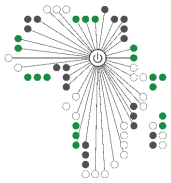

Universidade Virtual Africana

INFORMÁTICA APLICADA: CSI 4305

ENGENHARIA DE SOFTWARE

Martina Jennifer Zucule

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

Martina Jennifer Zucule de Barros

Par revisor(a)

Paulo Valdinacio

UVA - Coordenação Académica

Dr. Marilena Cabral

Coordenador Geral Programa de Informática Aplicada

Prof Tim Mwololo Waema

Coordenador do módulo

Robert Oboko

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.

Índice

Descrição Geral do Curso

Bem-vindo a Engenharia de Software

A Engenharia de Software é uma área que debruça-se sobre todos os aspectos relacionados com a produção de software, desde a especificação do sistema como um todo até a fase de manutenção deste depois de utilizado (Laurie Williams, 2004). Na perspectiva da engenharia, a Engenharia de Sistemas adopta uma abordagem sistemática e organizada de desenvolver sistemas.

O curso vai-lhe permitir fazer uso das várias práticas específicas ou técnicas para desenvolvimento de softwares. É importante que você se familiarize com as ferramentas de modelos Industriais com a finalidade de expor os mesmos ao estudo da arte de praticas com enfâse no desenvolvimento de softwares. Com a realização destes métodos, você terá uma melhor compreensão das complexidades bem como as subtilezas das várias actividades de desenvolver um produto final e com qualidade que inclui um trabalho em equipe ou grupo.

A engenharia de software é uma área necessário para desenvolver todas as espécies de projectos de software, inclusive projectos de softwares complexos. O desenvolvimento de um produto sempre passa pela fase de análise de requisitos, isto é o que é necessário para desenvolver, de seguida tem se outras fases como a modelagem do sistema até a finalização do mesmo.

Pré-requisitos

- Princípios de Sistemas de Base de dados
- Pricínpios de Programação
- Introdução a Programação Estruturada
- Introdução à Programação Orientada-a-Objectos

Materiais

Os materiais necessários para completar este curso incluem:

- Scripts das Aulas
- Ferrametas de Software
- Vídeos On-line
- Livros
- Forúm de Discussão
- Páginas Wiki

Objetivos do Curso

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

- Utilizar os conceitos da Ciência de Computação e princípios de engenharia de desenvolvimento de projectos de software.
- Desenhar, desenvolver e verificar sistemas e aplicações de softwares em Industrias, negócios e aplicações pessoais.
- Aplicar métodos sustentáveis para desenvolver um software de qualidade dentro de um tempo e com orçamentos previstos.

Unidades

Unidade 0: Diagnóstico

Nesta unidade você irá aprender sobre o que é Engenharia de Software e como esta disciplina auxilia as diversas actividades relacionadas com o desenho e produção de softwares.

Unidade 1: Fundamentos da Engenharia de Software

Nesta unidade você vai aprender todos sobre os elementos essências da engenharia de software, desde a sua natureza até as suas especificidades. Serão abordados alguns conceitos da engenharia de software relacionados com a Informática.

Unidade 2: Planeamento de Projecto de Software e Análise de requisitos do software e especificações.

A unidade dois versa sobre os aspectos relacionados com o desenvolvimento do software. O desenvolvimento de um software envolve vários sujeitos até a sua fase de conclusão. Você vai aprender como é que estes sujeitos interagem na produção de um software.

Unidade 3: Desenho de Sistemas

Esta unidade, desenho de sistemas está mais focada nos aspectos e sujeitos envolvidos no desenho do sistema isto é o desenho do software, assim, você vai aprender que aspectos são importantes nesta fase.

Unidade 4: Implementação e Testes

Depois de alguns processos como a análise de requisitos, modelação do sistema, temos a implementação e testes. Você vai aprender a usar as diversas ferramentas para desenvolver um software e como testar o mesmo depois de pronto.

Unidade 5: Gestão do Projecto e Manutenção

Nesta unidade você vai aprender que não basta só desenvolver um software ou produto, é necessário testar o mesmo, de modo que este possa ser utilizado pelos utilizadores finais ou clientes que encomendaram o determinado produto. Esta unidade fala dos tipos de testes que devem ser feitos depois de produzido o nosso software e como fazer a manutenção dos mesmos.

Avaliação

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

No final de cada módulo são apresentados instrumentos de avaliação sumativa, tais como testes e trabalhos finais, que compreendem os conhecimentos e as competências estudadas no 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:

	Título da Avaliação	Nota	Comentários
1	Avaliação – Exame no final do módulo (avaliação sumativa)	40%	Nota sobre 20 ou 100: Duração de 3 Horas Esta avaliação pode ser preferencialmente um projecto para avaliar o desenvolvimento de competências adquiridas.
2	Avaliação da Unidade 1: Fundamentos de Software (avaliação sumativa)	15%	Podem ser projectos de acordo com a matéria em curso.
3	Avaliação da Unidade 2: Planeamento de Projecto de Software e Análise de Requisitos do Software e especificações (avaliação sumativa)	15%	Podem ser projectos de acordo com a matéria em curso
4	Avaliação da Unidade 3: Desenho de Sistemas (avaliação sumativa)	15%	Podem ser projectos de acordo com a matéria em curso

5	Avaliação da Unidade 4:Implementação e Testes (avaliação sumativa)	15%	Podem ser projectos de acordo com a matéria em curso
6	Avaliação da Unidade 5: Gestão do Projecto e Manutenção (avaliação sumativa)	15%	Podem ser projectos que exijam uma conta pesquisa para além dos dados adquiridos durante o decurso do módulo.

Calendarização

Unidade	Temas e Actividades	Estimativa do tempo
Unidade 0	Nesta secção esta em causa a avaliação da Unidade 0	5h00
	Leitura do conteúdo da Unidade 0 pelos alunos	
	Os alunos devem fazer atividades de aprendizagem	
Unidade 1: Fundamentos de Software	Nesta secção está em causa a avaliação da Unidade 1	20h00
	Actividade 1:	
	Leitura do conteúdo da Unidade 1 pelos alunos	
	Leitura dos recursos obrigatórios da Unidade 1 pelos alunos	
	Os alunos devem fazer as atividades de aprendizagem	
Unidade 2: Planeamento de Projecto de Software e Análise de requisitos de Software e especificações	Nesta secção esta em causa a avaliação da Unidade 2	20h00
	Actividade 1:	
	Leitura do conteúdo da Unidade 1 pelos alunos	
	Leitura dos recursos obrigatórios da Unidade 1 pelos aunos	
	Os alunos devem fazer as atividades de aprendizagem	

Unidade 3: Desenho de Sistemas	Nesta secção esta em causa a avaliação da Unidade 3	20h00
	Actividade 1: Titulo	
	Leitura do conteúdo da Unidade 1 pelos alunos	
	Leitura dos recursos obrigatórios da Unidade 1 pelos aunos	
	Os alunos devem fazer as atividades de aprendizagem	
Unidade 4: Implementação e Testes	Nesta secção esta em causa a avaliação da Unidade 4	25h00
	Actividade 1:	
	Leitura do conteúdo da Unidade 1 pelos alunos	
	Leitura dos recursos obrigatórios da Unidade 1 pelos aunos	
	Os alunos devem fazer as atividades de aprendizagem	
Unidade 5: Gestão de Projecto e Manutenção	Nesta secção esta em causa a avaliação da Unidade 5	30h00
	Actividade 1:	
	Leitura do conteúdo da Unidade 1 pelos alunos	
	Leitura dos recursos obrigatórios da Unidade 1 pelos aunos	
	Os alunos devem fazer as atividades de aprendizagem	
Neste quadro faltam a indicação dos temas		
Penso que e importante indicar		

Leituras e outros Recursos

As leituras e outros recursos deste curso são:

- Agawal B. B., Tayal S. P. and Gupta M., (2008), "Software Engineering and Testing" Jones and Bartlett Publisher, ISBN: 978-1-934015-55-1
- Sommerville Ian, (2000): Software Engineering (6th Edition). Addison-Wesley, Boston USA
- Pressman Roger S., (2001), "Software Engineering, A Practitioner' S Approach" Fifth Edition, McGraw-Hill Higher Education, ISBN 0073655783
- Roger S. Pressman, (2000), adapted by Darrel Ince Software Engineering A Practitioner's Approach European Adaptation (5th Edition), McGraw-Hill, ISBN 0073655783
- RICHARD FAIRLEY, Software Engineering Concepts, McGraw- Hill, New York, 1985.
- [IEEE, Guide to the Software Engineering Body of Knowledge (SWEBOK), 2004, editores: Alain Abran e James W. Moore
- David Gustafson, (2002), "theory and Problems of Software Engineering", Schaum's Outline series, McGRAW-HILL, 0-07-140620-4

Unidade 0

Leituras e outros recursos obrigatórios:

- Pressman Roger S., (2001), "Software Engineering, A Practitioner' S Approach" Fifth Edition, McGraw-Hill Higher Education, ISBN 0073655783

Leituras e outros recursos opcionais:

- The New Product Development Game, HIROT AKA T AKEUCHI, IKUJIRO NONAKA, Harvard Business Review, 1986]
- WILSON DE PÁDUA PAULA FILHO, Engenharia de Software: Fundamentos, Métodos e Padrões, Editora LTC, 2a Edição, 2003

Unidade 1

Leituras e outros recursos obrigatórios:

- Agawal B. B., Tayal S. P. and Gupta M., (2008), "Software Engineering and Testing" Jones and Bartlett Publisher, ISBN: 978-1-934015-55-1
- Sommerville Ian, (2000): Software Engineering (6th Edition). Addison-Wesley, Boston USA

Leituras e outros recursos opcionais:

- WILSON DE PÁDUA PAULA FILHO, Engenharia de Software: Fundamentos, Métodos e Padrões, Editora LTC, 2a Edição, 2003
- [IEEE, Guide to the Software Engineering Body of Knowledge (SWEBOK), 2004, editores: Alain Abran e James W. Moore

Unidade 2

Leituras e outros recursos obrigatórios:

- Agawal B. B., Tayal S. P. and Gupta M., (2008), "Software Engineering and Testing" Jones and Bartlett Publisher, ISBN: 978-1-934015-55-1
- Sommerville Ian, (2000): Software Engineering (6th Edition). Addison-Wesley, Boston USA

Leituras e outros recursos opcionais:

- WILSON DE PÁDUA PAULA FILHO, Engenharia de Software: Fundamentos, Métodos e Padrões, Editora LTC, 2a Edição, 2003
- [IEEE, Guide to the Software Engineering Body of Knowledge (SWEBOK), 2004, editores: Alain Abran e James W. Moore

Unidade 3

Leituras e outros recursos obrigatórios:

- Agawal B. B., Tayal S. P. and Gupta M., (2008), "Software Engineering and Testing" Jones and Bartlett Publisher, ISBN: 978-1-934015-55-1
- Sommerville Ian, (2000): Software Engineering (6th Edition). Addison-Wesley, Boston USA

Leituras e outros recursos opcionais:

- WILSON DE PÁDUA PAULA FILHO, Engenharia de Software: Fundamentos, Métodos e Padrões, Editora LTC, 2a Edição, 2003

Unidade 4

Leituras e outros recursos obrigatórios:

- Agawal B. B., Tayal S. P. and Gupta M., (2008), "Software Engineering and Testing" Jones and Bartlett Publisher, ISBN: 978-1-934015-55-1
- Sommerville Ian, (2000): Software Engineering (6th Edition). Addison-Wesley, Boston USA

Leituras e outros recursos opcionais:

- V. Subramanyam, Sambuddha Deb, Priya Krishnaswamy, Rituparna Ghosh. An Integrated Approach to Software Process Improvement at Wipro Technologies veloci-Q. March 2004
- Forrest Shull and Roseanne Tesoriero. Software Engineering: Theory and Practice. Link: http://codecourse.sourceforge.net/materials/Software-Engineering-Theory-and-Practice.pdf, Acessado em Abril, 2016.
- http://www.mheducation.com/highered/home-guest.html

Unidade 5

Leituras e outros recursos obrigatórios:

- Agawal B. B., Tayal S. P. and Gupta M., (2008), "Software Engineering and Testing" Jones and Bartlett Publisher, ISBN: 978-1-934015-55-1
- Sommerville Ian, (2000): Software Engineering (6th Edition). Addison-Wesley, Boston USA

Leituras e outros recursos opcionais:

- V. Subramanyam, Sambuddha Deb, Priya Krishnaswamy, Rituparna Ghosh. An Integrated Approach to Software Process Improvement at Wipro Technologies veloci-Q. March 2004
- Forrest Shull and Roseanne Tesoriero. Software Engineering: Theory and Practice. Link: http://codecourse.sourceforge.net/materials/Software-Engineering-Theory-and-Practice.pdf, Acessado em Abril, 2016.
- http://www.mheducation.com/highered/home-guest.html

Unidade 0. Diagnóstico

Introdução à Unidade

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

O Ano de 1961 foi um ano em que o Mundo presenciou o surgimento de novos computadores. Estes computadores diferenciavam-se dos actuais por serem mais modernos e com um poder de carga computacional elevado. Com este advento surgiu também a necessidade do surgimento de programas e a busca pela qualidade dos mesmos introduziu a engenharia de software. O software teve um impacto grande, mas por outro lado esse boom trouxe também muitos problemas relacionados com o novatos. Assim em meados de 1968 surgiu a primeira crise do software.

A Engenharia de software foi um conceito muito debatido em 1968, e foi primeiramente ouvido na primeira conferência da NATO. Foi desta forma que a engenharia de software passou a fazer parte das nossas vida.

Figura 1: Reunião sobre Engenharia de Software

Objetivos da Unidade

Após a conclusão desta unidade, você deverá ser capaz de:

- Identificar os marcos importantes na evolução da Engenharia de Software
- Identificar os nomes sonantes de todos que estiveram por detrás do proc
- Definir os passos na criação de software

Termos-chave

Software:Software consiste em: (1) instruções (programas de computador) que, quando executadas, fornecem características, funções e desempenho desejados; (2) estruturas de dados que possibilitam aos programas manipular informações adequadamente; e (3) informação descritiva, tanto na forma impressa como na virtual, descrevendo a operação e o uso dos programas. (R. Pressman.2011)

Sistema: Um sistema é uma colecção de componentes que em conjunto realizam algumas formas de objectos de um sistema.

SDLC: É um ciclo de vida de desenvolvimento de um sistema que significa a combinação de várias actividades. Em outras palavras pode-se dizer que várias actividades juntas referem-se a ciclo de vida de desenvolvimento de um sistema.

IPO: Que significa Input (Entrada), Process (Processo) e Output (Saida) é o desenho básico de um programa.

Actividade 1: Desenho e Análise de Sistemas

Sistema de Computador

Sistemas são criados para resolver problemas. Pode-se pensar que a abordagem de sistemas como uma forma organizada de resolver problemas. No mundo dinâmico, a disciplina de análise de sistema e desenho do ingles Systems Analysis and Design (SAD) meramente trata das actividades de desenvolvimento de software.

O qué é um Sistema

Um sistema é um conjunto de elementos interconectados, de modo a formar um todo organizado. (Wikipedia)

Segudo (Tran Thi Phien, 2006), um sistema geralmente pode ser definido como uma colecção de componentes que trabalham em conjunto para realizar alguns formas de objectivos do sistema. Um sistema pode incluir programa, mecânico, eléctrico e hardware electrónico que pode ser operado por pessoas. Basicamente existem três maiores componentes em cada sistema que são conectados com os outros e são independentes como por exemplo:

Corpo humano que representa um sistema natural completo

Nós também estamos atados por muitos sistemas nacionais como sistema politico, sistema económico, sistema educacional e mais.

O objectivo da demanda dos sistemas exigem que alguns resultados são produzidos como resultado de processameto de entradas convenientes.

Tipos de Sistemas

De acordo com (Tran Thi Phien 2006), existem muitos tipos de sistemas, em que todos os dias nós estamos em contacto. Aqueles que são interessados que são automáticos, sistemas de informação computadorizados. Sistemas automáticos são sistemas que interagem com ou são controlados por um ou mais computadores. Nós podemos distinguir muitos tipos diferentes de sistemas automáticos, mas todos eles tem cinco componentes básicos em comum.

Infrastrutura: O físico e os componentes do sistema de hardware, isto é: servidores, hardware de computador como: CPUs, discos, terminais, ect.

Programa de Computador: Os programas e os programas de sistema incluindo sistemas operativos, sistema de base de dados, utilitários e aplicações (sistema financeiro).

Pessoas: para operar o sistema, para prover as entradas e consumir as saídas, e prover o manual de processamento de actividades no sistema como programas, operadores, utilizadores do sistema e gestão.

Dados: a informação capturada, utilizada e suportada pelo sistema incluindo ficheiros e bases de dados. A informação que o sistema lembra durante um perido de tempo.

Procedimentos: O programa e o manual de utilizador, instruções e passos involvidos na operação do sistema, incluindo procedimentos de tecnologias de informação (IT) para backup e manutenção.

Sistemas de negócio usam esses tipos de componentes para transformar entrada de dados em saídas de informação.

Partcipantes do Desenvolvimento de Sistemas

Num processo típico de desenvolvimento existem quatro categorias importantes de jogadores (Tran Thi Phien 2006):

a) Utilizador

O jogador é a pessoa mais importante no sistema (ou grupo de utilizadores) para quem o sistema é construído. Ela ou ele é a pessoa que vai interagir várias vezes e de forma detalhada, aprender que novas funções o sistema deve apresentar. O utilizador muitas vezes é tido como o dono no sentido de que ele ou ela recebe ou herda o sistema quando é construído. O utilizador é também o cliente em dois aspectos importantes:

- Como outros profissionais o cliente esta sempre certo, em relação á satisfação, ou não satisfação que o mesmo possa ter em relação ao uso do produto.
- O cliente é em último lugar a pessoa que paga pelo sistema e normalmente tem o direito e habilidade de recusar a pagar se não estiver satisfeito com o produto que recebeu.

Gestor

Gestor tem um pouco menos de acções no desenvolvimento do sistema. Existem algumas actividades de gestores que são:

Gestor do Utilizador: faz a gestão de várias pessoas na área operacional onde o sistema será utilizado. Existem normalmente gestores intermediários que querem que o sistema produza uma variedade de relatórios internos e análises de curto prazo.

Gestor Executivo do Desenvolvimento do Projecto (GEDP): a pessoa responsável pelo desenvolvimento do projeto num todo, e gestores de alto nível que são responsáveis por toda a gestão e alocação de recursos para a equipe técnica na organização do desenvolvimento do sistema.

Gestor Geral: gestores de nível alto que não estão directamente envolvidos no processo de (GEDP) ou na organização dos utilizadores. Este nível inclui o presidente ou CEO da organização.

Analista de Sistema

O analista de sistema é a pessoa chave de qualquer projecto de desenvolvimento de sistema. Um analista de sistema desempenha diversas funções.

i.**Arqueólogo**: Como analista de sistema uma das principais tarefas é revelar detalhes e documentar políticas de negócio que possam existir apenas como tradição, transmitida de geração em geração para utilizadores.

ii.**Inovador**: o analista de sistemas deve separar os sintomas do problema do utilizador da sua real causa. Com o seu conhecimento em Informática o analista deve ajudar o utilizador a explorar os novos aplicativos do computador.

iii.**Mediador**: O analista de sistema que muitas das vezes se encontra no meio dos utilizadores, gestores, programadores, auditores e vários outros jogadores que a maior parte deles nunca concordam com as ideias uns dos outros.

iv.**Líder do projecto:** porque o analista de sistema geralmente é mais experiente que os programadores do projecto e uma vez que este é atribuído o projecto antes dos programadores começarem a trabalhar há uma tendência natural de atribuir responsabilidades de gestão do projecto ao analista.

Projectistas do Sistema

Os projectistas do sistema é a pessoa ou grupo de pessoas) que receberá a saída do trabalho

de análise de sistemas. Sua tarefa é a de transformar uma tecnologia sem declaração de requisitos do utilizador em um projeto de arquitetura de alto nível que irá fornecer o quadro dentro do qual o programador pode trabalhar. Em muitos casos, o analista de sistemas e o designer de sistemas são a mesma pessoa ou membro do mesmo grupo unificado de pessoas.

Programadores

Particularmente em grandes projectos de desenvolvimento de sistemas, os designers de sistemas são susceptíveis de serem uma "cobertura" entre os analistas de sistemas e programadores.

Os analistas de sistemas fornecem seus produtos para o sistema designers e estilistas do sistema oferecem os seus produtos para o programador.

Analista de sistemas e o programador podem ter pouco ou nenhum contacto uns com os outros porque o trabalho é frequentemente realizado em uma sequência estritamente serial em muitos projectos de desenvolvimento de sistemas.

Pessoal de operações:

O pessoal de operações são responsáveis para o centro de informática, telecomunicações, segurança de dados e hardware de computador, bem como ao funcionamento actual dos programas de computador, montagem de conjuntos de discos e manuseio da saída do computador impressoras. Isto acontece depois de um novo sistema não só tem sido analisado e projectado, mas também tem sido programados e testados.

Actividade Ciclo de Vida de Desenvolvimento de Sistema

Ciclo de vida de desenvolvimento de sistema é um processo organizacional de desenvolvimento e manutenção de um sistema. O ciclo de vida do sistema ajuda a manter o plano de projecto, porque dá uma visão geral de todos processos e sub-processos necessários para desenvolver um sistema.

Ciclo de vida de desenvolvimento de sistema (CVDS) significa combinação de diversas actividades. Em outras palavras podemos dizer que diversas actividades juntos são referidos como o ciclo de vida do desenvolvimento do sistema. Na análise do sistema e Design (Engenharia de Software) terminologia, o ciclo de vida de desenvolvimento de sistema se refere ao ciclo de vida de desenvolvimento de software (CVDS). As fases do ciclo de vida de desenvolvimento do sistema podem ser identificadas por nomes diferentes.

Avaliação

1. O que é um sistema? Dê algumass definições de sistema.
2. Como é possível distinguir sistemas naturais de sistemas artificiais.
3. Liste alguns sistemas atomatizados e a regras para construidos
4. Quem participa do processo de desenvolvimento de um sistema? Explique o papel de cada um?

Princípios de Programação

Princípios de Programação Estruturada

A Programação Estruturada pode ser entendida como uma forma de programar que visa facilitar a escrita, compreensão, validação e manutenção de programas. Assim, a arte de programar consiste na arte de organizar e dominar a complexidade. A programação estruturada procura reduzir o nível de complexidade através de níveis como:

a. desenvolvimento do programa em diferentes fases por refinamento sucessivo (desenvolvimento top-down);

b. decomposição do programa total em módulos funcionais, organizados de preferência num sistema hierárquico;

c. uso de um número limitado de estruturas básicas de fluxo de controle dentro de cada módulo (Elias Pereira)..

Desenvolvimento Top-Down

Na Programação Estruturada, ao desenvolvermos um algoritmo, temos como objecto um produto final – o programa. Todavia, para termos esta transição, passamos por várias fases, no sentido "cima para baixo", onde cada fase é documentada e principalmente obtida por "refinamento" da fase anterior, até chegarmos a um nível de detalhamento que permita implementar o algoritmo directamente na linguagem de programação (Elias Pereira)..

Modularização

A solução final de um problema é obtida através de soluções de subproblemas, o que permite dividir o programa em módulos com subfunções claramente delimitadas, que podem, inclusive, ser implementados separadamente, por diversos programadores de uma equipe.

Estruturas de Controle

São representadas pela seqüência simples, o comando condicional e o comando repetitivo, e fornecem ao programador um aumento da legibilidade e compreensão de cada módulo de programa. Assim, temos como uma das principais normas da Programação Estruturada : não usar comandos de desvio (GOTO).

Confiabilidade

Medimos a confiabilidade de um sistema através de sua resposta ao uso constante, no tocante a: não apresentar erros e, corresponder às especificações.

Manutenção

As revisões sofridas por um programa (releases) tanto para correção de erros quanto para mudanças de especificação, são consideradas como manutenção de software. Os programas devem passar por testes exaustivos de confiabilidade antes de serem colocados em produção. Falhas nesta fase levam a altos níveis de manutenção, que consequentemente, levam a altos custos (Elias Pereira).

Algoritmos

Programas são formulações concretas de algoritmos abstractos, baseados em representações e estruturas específicas de dados". De forma bem simples, um algoritmo pode ser definido como "um conjunto de passos lógicos, bem definidos, que descreve a solução de um problema". Ao pensarmos na solução de um problema, encontramos acções imperativas que são expressas por comandos. Os algoritmos não são aplicados apenas ao mundo da Informática; pelo contrário, usamos – até sem perceber – algoritmos em todos os momentos de nossa vida. Uma receita de cozinha é claramente um algoritmo (Elias Pereira).

Avaliação

1. De forma clara descreva o ciclo de vida de processamento de dados em programas de computador.
2. De forma breve descreva os passos necessários para criar um programa
3. Descreva a importância de algoritmos em programação
4. Descreva com detalhe dois princípios de programação estruturada.

Resumo da Unidade

Existem muitas metodologias aplicadas no processo de desenvolvimento de software, uma delas é a abordagem estrutural. Abordagem estrutural é a metodologia mais usada. Técnicas que são empregues incluem diagrama de entidade e relacionamento, diagrama de sequencia e dicionário de dados. Nesta unidade introduzimos o conceito de programação estruturada, bem como os agentes envolvidos no processo de produção de um software.

Avaliação da Unidade

1. Qual é o papel de um gestor no desenvolvimento de um sistema?
2. Que tipos de gestores podemos ter num projecto de desenvolvimento de software?
3. O que entende por programação estruturada?
4. Dê exemplo de um algoritmo, e explique em suas palavras como seria o processo.

Leituras e Outros Recursos

Livros Recomendados

- http://pt.wikipedia.org/wiki/Sistema
- Agawal B. B., Tayal S. P. and Gupta M., (2008), "Software Engineering and Testing" Jones and Bartlett Publisher, ISBN: 978-1-934015-55-1
- Roger S. Pressman, Engenharia de Software. Uma abordagem Profissional. 2011. ISBN 978-85-8055-044-3
- Tran Thi Phien, (2006), "System Analysis and Design", Institute of Information Technology

Livro de Referência

- David Gustafson, (2002), "theory and Problems of Software Engineering", Schaum's Outline series, McGRAW-HILL, 0-07-140620-4
- Elias Andrade Pereira. Apostila de Lógica de Programação. Link: https://jornalismodedados.files.wordpress.com/2013/02/curso-bc3a1sico-de-lc3b3gica-da-programac3a7c3a3o-elis-andrade-pereira.pdf, Acessado em Abril 2016

As leituras e outros recursos desta unidade encontram-se na lista de "Leituras e Outros Recursos do curso".

Unidade 1. Fundamentos da Engenharia de Software

Introdução à Unidade

Software é definido pelas suas capacidades", capacidades relacionadas com as funções que este executa, as características que este prove, e as facilidades que este oferece. Para além das funções e facilidades o desenvolvimento de um software requere que os mesmos sejam desenvolvidos através de plataformas, isto é hardware, sistemas operativos, etc. Nesta unidade serão abordadas questões relacionadas com a necessidade de engenharia de software e processo de desenvolvimento de software. Os processos genéricos como modelo em cascata, evolucionário serão abordados em detalhes.

Objectivos da Unidade

Após a conclusão desta unidade, você deverá ser capaz de:

- Definir o termo software
- Descrever o modelo do processo genérico de desenvolvimento de um software
- Ilustrar as fases envolvidas no processo de desenvolvimento de um software, isto é a vida de desenvolvimento de um software.

Termos-chave

Software: Software é um conjunto de instruções com toda a documentação associada e dados configurados usados para adquirir entradas e para manipula-las para produzir o resultado desejado em termos de funções e desempenho como determinado pelo utilizador do software.

Processo de Software: Segundo Pressman (2011) um processo de software é definido como uma metodologia para as mudanças, acções e tarefas necessárias para desenvolver um software de alta qualidade.

Produto de Software: Produto de software são sistemas de software entregues ao cliente com a devida mudanças que descreve como instalar e manusear o sistema.

Processo: Um processo define um framework como um conjunto de mudanças de processos chaves que devem estabelecer-se para entrega eficaz da tecnologia de engenharia de software.

Actividades de Aprendizagem

Actividade 1: Natureza do Software

Introdução

Nesta unidade, a natureza do software, definição do software, tipos, classes de software, crise do software e mitos serão apresentados. Actualmente o software desempenha dois papeis. Primeiro o software é um produto que resulta de um processo, por outro lado é um veículo que serve para distribuir um produto.

Segundo Pressman (2011), o software como produto, fornece o potencial computacional representado pelo hardware ou, de forma mais abrangente, por uma rede de computadores que podem ser acessados por hardware local. Independentemente de residir em um celular ou operar dentro de um mainframe, software é um transformador de informações — produzindo, gerindo adquirindo, modificando, exibindo ou transmitindo informações que podem ser tão simples quanto um único bit ou tão complexas quanto uma apresentação multimídia derivada de da- dos obtidos de dezenas de fontes independentes. Como veículo de distribuição do produto, o software actua como a base para o controle do computador (sistemas operacionais), a comunicação de informações (redes) e a criação e o controle de outros programas (ferramentas de software e ambientes).

O que é Software?

O software não é somente um programa, um software conta com a documentação e configuração dos dados necessários para a fazer com o programa opere de forma correcta. De acordo com Agarwal et al (2010) software é um conjunto de instruções utilizadas para adquirir entradas e manipulá-las para produzir o mudanças esperadas em termos de mudança e desempenho que são determinadas pelo utilizador do software. Por outro lado, inclui um conjunto de documentos como manual do programa, destinado a ajudar o utilizador a entender o sistema de software. Hoje o software é composto por desenho de documentos, mudanças do sistema e manuais de mudanças. Em outras palavras pode-se dizer que o sistema de software consiste em:

- Instrução (um número de programas de computador separados) que quando executados fornecem as funções e desempenho desejado.
- Ficheiros de configuração que são utilizados para configurar os programas e estruturas de dados que permitem aos programas manipular apropriadamente a informação.
- Documentação do sistema que descreve as mudanças do sistema e
- Documentação do utilizador que explica o uso do programa e do sistema.

Crise do Software

Porque os projectos de software não eram entregues os produtos prometidos de acordo com as mudanças, dentro do tempo, dentro e com a qualidade especificada?

O desenvolvimento do software esteve em crise (desastre) porque os métodos (se houve algum) utilizados não foram suficientemente bons.

- Técnicas aplicadas a pequenos sistemas não eram adequadas.
- Projectos importantes algumas vezes demoravam muitos anos, acabando por custar mais do que foi previsto.
- Desenvolvimento de mudanças foi inseguro, executados de forma que os mesmos e tornaram-se difíceis de manter.

Durante o desenvolvimento de software muitos problemas foram surgindo e o conjunto de problemas foi denominado de crise de software. Quando o software começava a ser desenvolvido foi-se encontrado problemas associados aos passos de desenvolvimento.

Problemas

Os problemas encontrados foram:

- As estimativas de preço e mudanças muitas vezes eram inconsistentes.
- A produtividade de pessoas de software não acompanhou o passo de mudanças.
- A qualidade do mudanças algumas vezes é menos do que o adequado
- Sem uma mudanças sólida de qualidade de produtividade, não se pode avaliar sa mudanças no desenvolvimento de novas ferramentas.
- Comunicação entre o desenvolvedor e o cliente era muitas vezes probe.
- Tarefas de mudanças do software consumia a maioria de todos os fundos de software.

Causas

As causas dos problemas incluem:

- A qualidade do software não é boa porque muitos desenvolvedores faziam mudanças dos dados para desenvolver software.
- Se existe um atraso ou qualquer processo ou estágio (isto é, analise, desenho e teste) mudam o planeamento não combina com o tempo real.
- Comunicação entre gestores e clientes, desenvolvedores de equipe de suporte, ect., pode quebrar-se por causa das mudanças especiais do software e problemas associados ao seu software eram mal compreendidos.

Crise do Software do ponto de vista do programador

Do ponto de vista de programadores os problemas incluem: problemas de compatibilidade, portabilidade, documentação, problemas relacionados com a pirataria de software, na coordenação de trabalho de diferentes pessoas e problemas de devida manutenção.

Mitos Relativos ao Software

O processo de desenvolvimento do software também esteve rodeado de alguns mitos. Estes mitos envolvendo crenças infundadas remontam desde o surgimento da computação.

Assim, os mitos possuem uma série de atributos que os tornam insidiosos. Por um lado parecem ser de facto afirmaçõs que parecem conter alguma verdade isto é, tem uma sensação intuitiva e frequentemente são promulgados por praticamente experientes „que entendem do risco„.

Actualmente, a maioria dos profissionais de engenharia de software reconhece os mitos por aquilo que eles representam, atitudes enganosas que provocam sérios problemas tanto para gerentes quanto para praticantes da área. Apesar disso, existem alguns hábitos e atitudes que são difíceis de ser modificados e alguns deles permanecem.

Os mitos do software podem ser definidos em três categorias: Mitos de gestão, mitos do cliente e mitos dos profissionais da área (desenvolvedores de software.)

Mitos de Gestão

Muitas vezes os gerentes com a responsabilidade sobre o software, assim como os gerentes de outras áreas, frequentemente estão sobre pressão para manter o oraçamento do projecto, isto é evitar que haja deslizes no cronograma das actividades e elevar a qualidade do produto.

Mito: Já temos um livro que esta cheio de padrões e procedimentos para desenvolver software. Ele não supre meu pessoal com tudo que eles precisam saber?

Realidade: O livro com padrões pode muito bem existir, mas ele é usado? Os praticantes da área estão cientes de que ele existe? Esse livro reflecte a prática moderna da engenharia de software?É completo? É adaptável? Está alinhado para melhorar o tempo de entrega, mantendo ainda o foco na qualidade? Em muitos casos, a resposta para todas essas perguntas é não?(Pressman, 2011)

Mitos dos Clientes

O cliente que solicita o software pode ser uma pessoa que não esteja ao nosso lado da mesa, um grupo de técnicos de um andar inferior que o nosso, de um departamento de vendas e marketing ou de uma outra empresa que encomendou o proejcto. Muitas das vezes os clientes acreditam nos mitos de software porque nem sempre os gestores e profissionais da área fazem esforços para corrigir as informações falsas. Do lado do cliente este fica com falsas expectativas e cria também uma insatisfação com os desenvolvedores.

Mito: Uma definição geral dos objectivos é suficiente para começar a escrever os programas, pode-se preencher detalhes a posterior.

Realidade: Embora nem sempre seja possível uma definição ampla e estável dos requisitos, uma definição de objectivos ambígua é receita para um desastre. Requi- sitos não ambíguos (normalmente derivados da interatividade) são obtidos somente pela comunicação contínua e eficaz entre cliente e desenvolvedor.(Pressman, 2011)

Mitos dos Profissionais da área

Mitos que ainda sobrevivem nos profissionais da área têm resistido por mais de 50 anos de cultura de programação. Durante seus primórdios, a programação era vista como uma forma de arte. Modos e atitudes antigas dificilmente morrem.

Mito: Uma vez feito um programa e o colocado em uso, nosso trabalho está ter- minado.

Realidade: Uma vez alguém já disse que "o quanto antes se começar a codificar, mais tempo levará para terminá-lo". Levantamentos indicam que entre 60 e 80% de todo o esforço será despendido após a entrega do software ao cliente pela primeira vez.(Pressman, 2011)

Avaliação

1. Verifique a sua compreensão!
2. O que entende por software
3. O que é engenharia de sosftware?
4. Quais são os mitos e diferentes realidades sobre software?
5. De uma forma detalhada explique o processo de engenharia de software.

Actividade 2: O que é Enhengaria de Software

Introdução

Esta sub-unidade debruça sobre o conceito de aplicar as normas de engenharia no desenvolvimento de software, por isso o nome de „Engenharia de Software„

Definição

A IEEE define Engenharia de Software [1] da seguinte forma: A aplicação de uma abordagem sistemática, disciplinada e quantificável para o desenvolvimento, operação e manutenção de software, isto é, a aplicação de engenharia em software.

O critério chave para Engenharia de Software são: metodologias bem definidas, marcos previsíveis, rastreabilidade entre os passos, documentação e controle da manutenção. A engenharia de software preocupa-se com teorias, métodos e ferramentas que são necessárias para desenvolver software. A engenharia de software não é para produzir não se preocupa somente em produzir um sistema funcional, mas também em documentar o desenho do sistema, manual do utilizador, etc.

Princípios da Engenharia de Sosftware

Os principios da engenharia de software lidam com ambos processos de engenharia de software e o produto final. O processo certo vai ajudar a produzir o produto certo, mas o produto desejado vai também afectar a escolha sobre que processo usar. O problema tradicional na enganharia de software teve uma grande enfâse no processo ou produto ou na exclusão de um outro. Ambos são importantes. Para aplicar os princípios um engenheiro de software deve estar equipado com os métodos apropriados (guias gerais que guiam a execuação de algumas actividades, que são sistemáticas, e abordagens disciplinadas) e técnicas especificas que ajudem a incorporar as propriedades desejadas nos processos e produtos.

Produto de Software

Produto de software são sistemas de software entregues ao cliente com a documentação que descreve como instalar e usar o sistema. As características críticas do produto de software incluem:

- Utilidade: deve ser útil e melhorar a vida das pessoas
- Software é flexível. O programa pode ser desenvolvido para fazer tudo. A característica pode ajudar a acomodar qualquer tipo de mudança.

Avaliação

1. O que é Engenharia de Software?
2. Explique as características que um produto de software deve ter.
3. Como o software tem-se tornado mais pervasivo, risco ao público (devido a falta de programas) tornou-se uma enorme preocupação. Desenvolva um cenário realístico onde a falha de um programa de computador pode criar um enorme prejuízo (tanto económico ou humano).

Actividade 3: Áreas de Aplicações de Software

O aumento da complexidade do software faz com que a classificação do mesmo seja difícil. Assim sendo, iremos enumerar algumas das áreas de aplicação de software: (3)

Software Básico

Software de Tempo real: Estes softwares lidam com mudança de ambiente. Primeiro, colectam a entrada e convertem do analógico para o digital, controle de componente que respondem ao ambiente externo e realizam a acção. O software é utilizado para monitorar, controlar e analisar eventos do mundo real da forma como estes ocorrem.

Software comercial: Softwares comerciais são softwares desenhados para processar aplicações comerciais. Softwares comerciais podem ser processamento de aplicações dados e informações. Estes podem direccionar o processo de negócio através de transações processadas on-line ou no modo tempo real. Este tipo de softwares são utilizados para operações específicas como contabilização de pacotes, gestão de sistemas de informação e gestão de inventários.

Software de computador pessoal: Processamento de folhas de cálculos, computação gráfica, multimídia, entretenimento, gestão de base de dados, aplicações de negócios e pessoais, redes de computadores externas ou acesso a base de dados.

Software de inteligencia artificial: Software de inteligência artificial usam algoritmos não numéricos, que usam dados e informações geradas no sistema para resolver problemas complexos.

Avaliação

1. Enumera as várias áreas de aplicações de software
2. Explique o que é software embutido como uma das aplicações de software.

Actividade 4: Processo de Desenvolvimento de Software

Camadas da Engenharia de Software

Engenharia de Software é uma tecnologia de camadas. A figura 1.1 ilustra as camadas da Engenharia de Software.

Figura 1.1: Camadas de Engenharia de Software (3)

A base para a engenharia de software é a camada de processos. O processo de engenharia de software é a liga que mantém as camadas de tecnologia coesa e possibilita o desenvolvimento de forma racional e dentro do prazo. O processo define uma metodologia que deve ser estabelecida para a entrega efectiva de tecnologia de engenharia de software. O processo de software constitui a base para o controle da gestão de projectos de software e estabelece o contexto no qual são aplicados métodos técnicos, são produzidos derivados (modelos, documentos, dados, relatórios, formulários etc.) são estabelecidos marcos, a qualidade é garantida e mudanças são geridas de forma apropriada.

Os métodos da engenharia de software fornecem as informações técnicas para desenvolver software. Os métodos envolvem uma ampla gama de tarefas que incluem: comunicação, análise de requisitos, modelagem de projecto, construção de programas, testes e suporte.

Os métodos da engenharia de software baseiam-se em um conjunto de princípios básicos que governam cada área da tecnologia e inclui actividades de modelagem e outras técnicas descritivas.

As ferramentas de engenharia de software fornecem suporte automatizado ou semiautomatizado para o processo e para métodos. Quando as ferramentas são intergradas de modo que as informações criadas por uma ferramenta possam ser utilizadas por outra, é estabelecido um sistema para o suporte ao desenvolvimento de software, denomindado engenharia de software com o auxílo do computador (3).

Actividade 5: Ciclo de Desenvolvimento de Software

O ciclo de vida de desenvolvimento de software (CVDS) é inglês Software development life-cycle (SDLC) é utilizado para facilitar o desenvolvimento de uma vasta gama de produtos de forma sistemática, de forma bem definida e forma económica.

Um sistema de informação passa por uma série de fases desde a concepção até a execução. Este processo é chamado de ciclo de desenvolvimento de software. Várias razões para o uso de um modelo de ciclo de vida incluem:

- Ajuda a entender todo o processo
- Aplica o uso de uma abordagem estruturada para o desenvolvimento
- Permite o planeamento de recursos com antecedência
- Permite o controle destes processos a posterior
- Envolve a gestão para monitorar o progresso do sistema

O CVDS é composto por várias fases e estas fases devem ser identificadas em conjunto com a definição de critérios de entrada e saída para cada fase. Uma fase pode começar apenas quando forem satisfeitos os critérios de entrada da fase correspondente. Se não houver indicação clara de entrada e de saída de cada fase, torna-se muito difícil acompanhar o progresso do projecto (5).

O CVDS pode ser dividido em 5 a 9 fases, isto é, deve ter mo mínimo 5 fases e máximo nove fases. Em média tem 7 ou 8 fases que são:

- Início do Projecto e reconhecimento-planificação dos preliminares necessários da investigação.
- Identificação do projecto e seleção-estudo de viabilidade
- Análise do Projecto
- Desenho do Sistema
- Codificação
- Testes
- Implementação
- Manutenção

Actividade 6: Abordagens de Desenvolvimento de Software

Abordagens de desenvolvimento de sistemas em organizações profissionais normalmente baseiam-se em dois modelos básicos: o modelo em cascata e o modelo espiral.

Modelo em Cascata

No modelo em cascata os principais subprocessos são executados de acordo com uma sequencia, que permite demarcar pontos de controle bem definidos. Estes pontos de controle facilitam a muito a gestão de projectos, o que faz com que este processo seja em princípio confiável e utilizável em projectos de qualquer escala. Por outro lado, é tido como um processo rígido e burocrático onde as actividades de requisitos, análise e desenho como ilustra a figura 1.6.1 tem de ser muito bem dominadas pois não são permitidos erros. O modelo de cascata puro é de baixa visibilidade para o cliente, que só recebe o resultado final do projecto (5).

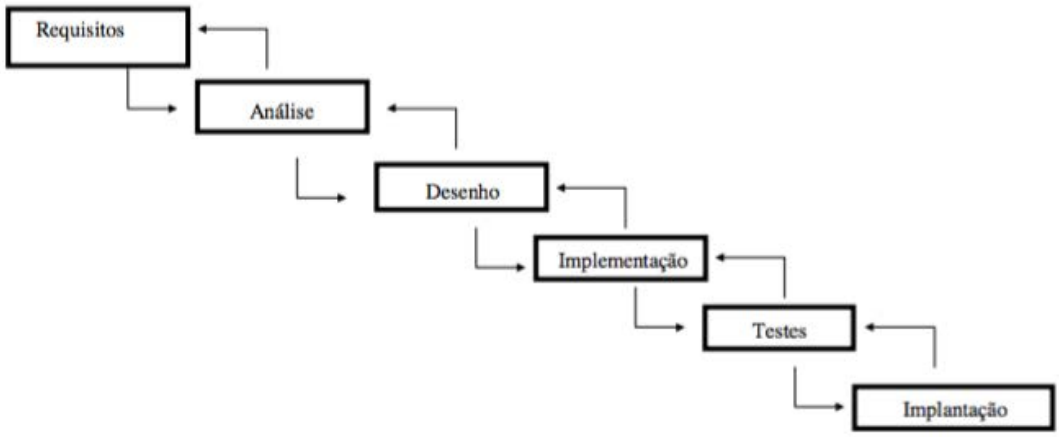

Figura 1.6.1: Modelo de ciclo de vida em Cascata (5)

Neste modelo, é necessário permitir que, em fases posteriores seja feita uma revisão e alteração de resultados das fases anteriores. Por exemplo, os modelos e documentos de especificação e desenho podem ser alterados durante a implementação, na medida em que problemas vão sendo descobertos. (5)

Modelo de ciclo de vida Espiral

O modelo em espiral é totalmente diferente do modelo em cascata, neste modelo o produto é desenvolvido em uma série de iterações. Cada nova iteração corresponde a uma volta na espiral como ilustra a figura 1.6.2. Desta forma é possível construir produtos em prazos curtos, com novas características e recursos que são agregados na medida em que a experiência descobre sua necessidade. As actividades de manutençãoo são utilizadas para identificar problemas e os registos fornecem dados para definir os requisitos das próximas libertações. O principal problema do ciclo de vida em espiral é que requer gestão muito sofisticada para ser previsível e confiável.(5)

Figura 1.6.2. Modelo de ciclo de vida em Espiral (5)

A variante do modelo espiral é o modelo de prototipagem evolutiva como ilustra a figura 1.6.3. Neste modelo a espiral é utilizada não para desenvolver o produto, mas para construir uma série de versões provisórias que são chamadas de protótipos. Os protótipos cobrem cada vez mais requisitos, até que se atinja o produto desejado. A prototipagem evolutiva permite que os requisitos sejam definidos progressivamente, a apresenta alta flexibilidade e visibilidade para os clientes. Entretanto, também requer gestão sofisticada e o desenho deve ser de excelente qualidade, para que a estrutura do produto não se degenere ao longo dos protótipos.

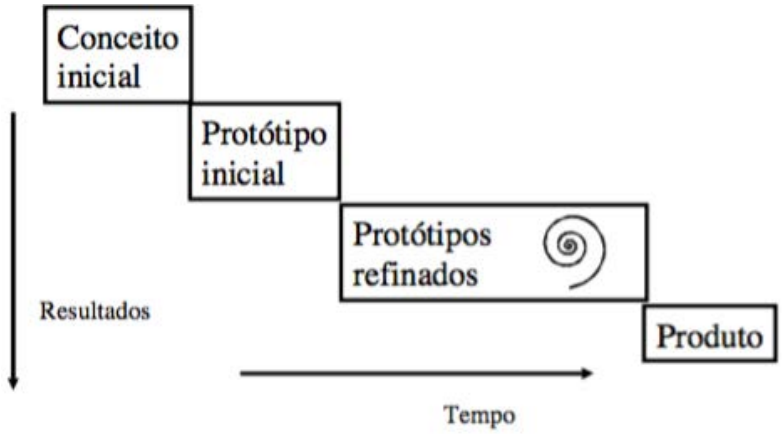

Figura 1.6.3: Modelo de ciclo de vida de prototipagem evolutiva (5)

Avaliação

1. Discuta o CVDS de forma muito resumida
2. Liste as fases básicas no ciclo de desenvolvimento de software
3. Quais são as diferentes fases no ciclo de vida de desenvolvimento de software. Quais são os produtos finais de cada etapa?
4. Explique as diferentes categorias de manutenção no processo de desenvolvimento do ciclo de vida.

Resumo da Unidade

O software tornou-se um elemento chave na evolução de produtos e sistemas baseados em computador. Os problemas encontrados em termos de crises de software e mitos fizeram com que os desenvolvedores de software aplicassem técnicas de engenharia no desenvolvimento de software, daí a disciplina de engenharia de software. Software é composto por programas, dados e documentos. Cada um desses itens, engloba a configuração que é criada como parte do processo de engenharia de software. A intenção da engenharia de software é prover um framework para construir software com qualidade.

Esta unidade apresenta a introdução de engenharia de software. Explica como a disciplina de engenharia foi empregue no desenvolvimento de software depois de analizados os problemas encontrados no desenvolvimento de software. A unidade apresentou com mais detalhes o processo de desenvolvimento de software apresentados sob a forma de ciclo de vida de desenvolvimento de software. Processos genéricos como modelo em cascata, evolucionário foram também abordados.

Avaliação da Unidade

Verifique a sua compreensão

1. O que é que percebe por software
2. 2O que é engenharia de software
3. Quais são os mitos e realidade sobre o software
4. De que forma detalhada explique o processo de engenharia de software.

Leituras e outros Recursos

- "IEEE Standard Glossary of Software Engineering Terminology", IEEE Std 610 12-1990, December 1990, p. 67
- Roger S. Pressman, Engenharia de Software: Uma Abordagem Profissional, Sétima Edicao. 2011
- Givanaldo Rocha de Sousa, Engenharia de Software. Introducao a Engenharia de Software.
- Agarwal B. B., Tayal S. P. and Gupta M., (2010), "Software Engineering & Testing, an Introduction", Jones and Bartlett Publishers, ISBN: 978-1-934015-55-1
- Wilson de Pádua Paula Filho. Engenharia de Software: fundamentos, métodos e padrões. Março, 2000

Unidade 2: Planeamento de Projecto de Software

e Análise de requisitos do software e especificações

Introdução da Unidade

Esta unidade introduz as bases para obter as especificações de requisitos de software empregando os requisitos de engenharia exigidos. Explica como entender o que os clientes querem, analisando e validando os seus requisitos e finalmente produzir as especificações de requisitos. A unidade também elabora um conjunto de actividades relacionadas com o planeamento de projecto de software. Planeamento do projecto de software foca estimativas de custo do projecto e planeamento do projecto.

Objectivos da Unidade

Após a conclusão desta unidade, deverá ser capaz de:

- Definir os seguintes termos: Engenharia de requisitos, planeamento de software.
- Descrever o processo adoptado na engenharia de requisitos.
- Demosntrar o processo de determinar os requisitos.
- Explicar o conceito de gestão de projecto e os quatro principais componentes envolvidos (pessoas, produto, processo e projecto).
- Descrever o conceito de gestão de projecto baseado em achar a estimativa de custos do projecto.
- Usar o guia que ilustra o processo de estimativa de projecto de software para obter o custo do projecto.

Termos-chave

Requisito: O requisito é uma característica do sistema ou uma descrição de alguma coisa que o sistema seja capaz de fazer no sentido de satisfazer o propósito do sistema.

Engenharia de requisitos: Engenharia de requisitos é o uso sistemático de princípios e técnicas aprovadas, e ferramentas de linguagens para analise do custo efectivo, documentação, e contínua avaliação das necessidades do utilizador e especificações de comportamento externo do sistema para satisfazer as necessidades desses utilizadores.

Estimativa do Projecto de Software: A estimativa do projecto de software é o processo de estimar vários recursos necessários para a realização de um projecto.

Actividades de Aprendizagem

Actividade 1: Engenharia de Requisitos

Introdução

Nesta actividade, serão apresentadas as definições de requisitos e engenharia de requisitos. Aborda também o processo adoptado na elaboração de engenharia de requisitos que levará a obtenção da especificação do requisito do sistema de software.

Requisito

Requisitos descrevem o quê do sistema e não o como. Engenharia de requisitos produz um vasto documento, escrito numa linguagem natural e contem a descrição do que o sistema irá fazer sem descrever como..

A engenharia de requisitos é o uso sistemático de princípios reconhecidos, técnicas e ferramentas de linguagens para a analise de custo efectivo, documentação e avaliação contínuia das necessidades do utilizador e as especificações do comportamento externo do sistema para satisfazer as exigências do utilizador. Pode ser definido como uma disciplina, que lida com requisitos de engenharia de objectos do inicio de um processo de desenvolvimento do sistema.

Tipos de Requisitos

Existem várias categorias de requisitos. De acordo com a sua prioridade, os requisitos são classificados de acordo com os três tipos seguintes:

- Aqueles que devem ser finalizados absolutamente
- Aqueles que são muito desejados, mas não necessários
- Aqueles que são possíveis mas podem ser eliminados

Na base de suas funcionalidades, os requisitos são classificados nos seguintes tipos:

Requisitos funcionais: Estes definem factores como formatos de entrada/saída, estructura de armazenamento, capacidades funcionais, regulação de tempo e sincronização.

Requisitos não funcionais: estes definem as propriedades ou qualidades do produto incluindo utilidade, eficiência, desempenho, espaço, confiança e portabilidade.

Os requisitos não funcionais têm origem nas necessidades dos utilizadores, em restrições de orçamento, em políticas organizacionais, em necessidades de interoperabilidade com outros sistemas de software ou hardware ou em fatores externos como regulamentos e legislações (SOMMERVILLE, 2007). Assim, os requisitos não funcionais podem ser classificados quanto à sua origem. Existem diversas classificações de requisitos não funcionais. Sommerville (2007), por exemplo, classifica-os em:

Requisitos de produto: especificam o comportamento do produto (sistema). Referem-se a atributos de qualidade que o sistema deve apresentar, tais como confiabilidade, usabilidade, eficiência, portabilidade, manutenibilidade e segurança.

Requisitos organizacionais: são derivados de metas, políticas e procedimentos das organizações do cliente e do desenvolvedor. Incluem requisitos de processo (padrões de processo e modelos de documentos que devem ser usados), requisitos de implementação (tal como a linguagem de programação a ser adotada), restrições de entrega (tempo para chegar ao mercado - time to market, restrições de cronograma etc.), restrições orçamentárias (custo, custo-benefício) etc.

Requisitos externos: referem-se a todos os requisitos derivados de fatores externos ao sistema e seu processo de desenvolvimento. Podem incluir requisitos de interoperabilidade com sistemas de outras organizações, requisitos legais (tais como requisitos de privacidade) e requisitos éticos.

No que se refere aos RNFs de produto, eles podem estar relacionados a propriedades emergentes do sistema como um todo, ou seja, propriedades que não podem ser atribuídas a uma parte específica do sistema, mas que, ao contrário, só aparecem após a integração de seus componentes, tal como confiabilidade (SOMMERVILLE, 2007). Contudo, algumas vezes, essas características podem estar associadas a uma função específica ou a um conjunto de funções. Por exemplo, uma certa função pode ter restrições severas de desempenho, enquanto outras funções do mesmo sistema não apresentam tal restrição.

Ainda que a classificação em requisitos funcionais e não funcionais seja a mais amplamente aceita, há outras classificações de requisitos. Por exemplo, Sommerville (2007) considera, além de requisitos funcionais e não funcionais, requisitos de domínio. Segundo esse autor, requisitos de domínio são provenientes do domínio de aplicação do sistema e refletem características e restrições desse domínio. Eles são derivados do domínio de aplicação e podem restringir requisitos funcionais existentes ou estabelecer como cálculos específicos devem ser realizados, refletindo fundamentos do domínio de aplicação.

Requisitos de domínio na concepção de Sommerville são o que outros autores, tal como Wiegers (2003), chamam de regras de negócio. Por exemplo, em um sistema de matrícula de uma universidade, uma importante regra de negócio diz que um aluno só pode se matricular em uma turma de uma disciplina se ele tiver cumprido seus pré-requisitos.

Essas regras de negócio geralmente incluem terminologia específica do domínio e fazem referência a conceitos do domínio (SOMMERVILLE, 2007). Assim, são mais facilmente capturadas na fase de modelagem conceitual.

Processo de Engenharia de Requisitos

A engenharia de requisitos de software é o ramo da Engenharia de software que engloba as actividades relacionadas coma definição dos requsitos de software de um sistema, desenvolvidas ao longo do ciclo de vida de software (KOTONYA; SOMMERVILLE, 1998).

O processo de engenharia de requisitos envolve criatividade, interacção de diferentes pessoas, conhecimento e experiência para transformar informações diversas (sobre a organização, sobre leis, sobre o sistema a ser construído etc.) em documentos e modelos que direcionem o desenvolvimento de software (KOTONYA; SOMMERVILLE, 1998).

A Engenharia de Requisitos pode ser descrita como um processo, ou seja, um conjunto organizado de actividades que deve ser seguido para derivar, avaliar e manter os requisitos e artefactos relacionados. Uma descrição de um processo, de forma geral, deve incluir, além das actividades a serem seguidas, a estrutura ou sequência dessas actividades, quem é responsável por cada actividade, suas entradas e saídas, as ferramentas usadas para apoiar as actividades e os métodos, técnicas e diretrizes a serem seguidos na sua realização.

Processos de engenharia de requisitos podem variar muito de uma organização para outra, ou até mesmo dentro de uma organização específica, em função de características dos projetos. A definição de um processo apropriado à organização traz muitos benefícios, pois uma boa descrição do mesmo fornecerá orientações e reduzirá a probabilidade de esquecimento ou de uma execução superficial. No entanto, não faz sentido falar em processo ideal ou definir algum e impô-lo a uma organização. Ao invés disto, as organizações devem iniciar com um processo genérico e adaptá-lo para um processo mais detalhado, que seja apropriado às suas reais necessidades (SOMMERVILLE; SAWYER, 1997).

Ainda que diferentes projectos requeiram processos com características específicas para contemplar suas peculiaridades, é possível estabelecer um conjunto de atividades básicas que deve ser considerado na definição de um processo de engenharia de requisitos. Tomando por base o processo proposto por Kotonya e Sommerville (1998), neste texto considera-se que um processo de engenharia de requisitos deve contemplar, tipicamente, as actividades mostradas na Figura 2.1: levantamento de requisitos, análise de requisitos, documentação de requisitos, verificação e validação de requisitos e gerência de requisitos.

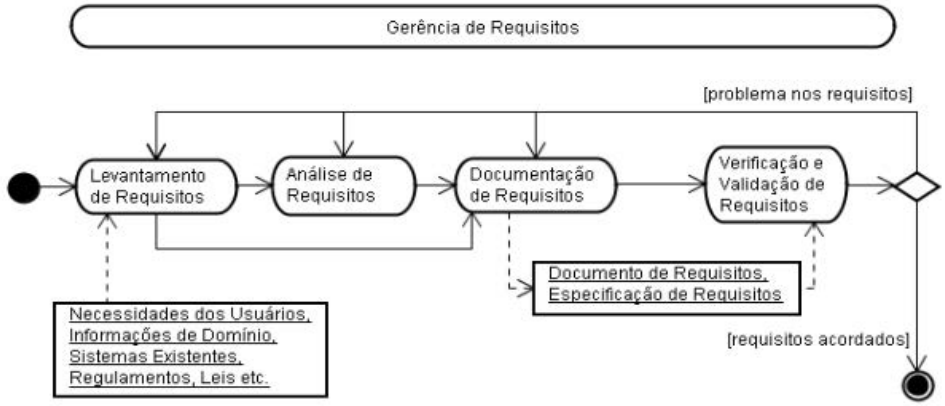

Figura 2.1: Processo de Engenharia de Requisitos Fonte: (Pressman, 2011)

O processo começa pelo levantamento de requisitos, que deve levar em conta necessidades dos utilizadores e clientes, informações de domínio, sistemas existentes, regulamentos, leis etc. Uma vez identificados requisitos, é possível iniciar a actividade de análise, quando os requisitos levantados são usados como base para a modelagem do sistema. Tanto no levantamento quanto na análise de requisitos, é importante documentar requisitos e modelos. Conforme discutido anteriormente, para documentar requisitos, dois documentos são normalmente utilizados: o Documento de Requisitos, contendo uma lista dos requisitos de usuário identificados, e o Documento de Especificação de Requisitos, que regista os requisitos de sistema e os vários diagramas resultantes do trabalho de análise. Os documentos produzidos são, então, verificados e validados. Adicionalmente, um esforço de garantia da qualidade deve ser realizado, visando garantir conformidade em relação a padrões e ao processo estabelecidos pela organização. Caso clientes, utilizadores e desenvolvedores estejam de acordo com os requisitos, o processo de desenvolvimento pode avançar; caso contrário, deve-se retornar à actividade correspondente para resolver os problemas identificados. Em paralelo a todas as actividades anteriormente mencionadas, há a gerência de requisitos, que se ocupa em gerenciar mudanças nos requisitos.

Documentação de requistsos (Especificação)

Documentação de requisitos é escrito depois de elucidação de requisitos e analise. Esta é a forma de representar num formato consistente. A documentação de requsitos é chamada de especificação de requisitos de software do inglês Software Requeirements Specification (SRS). O SRS é a especificação para um produto de software particular, ou um conjunto de programas que realizam certas funções em um ambiente especifico.

Revisão de requisistos (Validação)

Este é um processo manual que envolve múltiplos leitores de ambos os lados clientes e pessoal contratante analisando os documentos de requisitos anomalias e omissões validação de requisitos examina a especifição para garantir que todo os requisitos do sistema tenham sido iniciados inequivocamante; que inconsistência, omissões e erros tenham sido detectados e corrigidos; e que o produto de trabalho conforme os padores estabelecidos para o processo, o projecto e produto.

Gestão de requisitos

Requisitos define a capacidade que a solução do sistema de software deve entregar e os resultados desejados que devem resultar na sua aplicação de problemas de negocio. No sentido de gerar estes requisitos, uma aborgadem sistemática énecessaria através de uma gestão forma de processos chamada de gestão de requisitos.

Conclusão

A parte importante e difícil na construção de um sistema de software e em decidir precisamente o que construir e estabelecer os requisitos dos detalhes técnicos. Requisitos são identificados por obtebcoa de informacao do cliente. Requisitps sao analizados para avaliar a sua clareza, perfeicao e consistencia. Finalmente os requisitos do sistema são geridos para garatir que mudanças são propriamente controladas.

Avaliação

Verifique a sua compreensão

1. O que significa o termo requisitos?
2. Explica o processo de determinação de requisitos para sistemas baseados em software.
3. Discuta o significado de uso de engenharia de requisitos. Que são os problemas
4. Qual é o processo crucial na fase de engenharia de requisitos? Explique com ajuda de um diagrama.
5. Explique a importância de requisitos. Quantos tipos de requsitos são possíveis e porque?
6. O que é a elucidação de requsitos?

Actividade 2: Planificação de Projecto de Software

Objectivos de Planeamento de Software

O objectivo de planeamento de projecto de software é prover um framework que possibilita ao gestor efectuar estimativas de custos de recursos, custo e horário. O objectivo de planeamento e atingir através de processo de descoberta de informação que lida com estimativas razoaveis. Actividades associadas com o projecto de planeamento de software incluem:

Determinar o escopo do software

Funções e desempenho alocados ao software durante a engenharia de requsitos devem ser acessados para estabelecer o escopo do projecto que e não ambíguo e entendido no nível técnico e de gestão. Escopo de software descreve os dados e o control para ser processado, funções, desempenho, constrangimentos, interfaces e confiança.

Estimativa de recursos

A fugura 2.3 ilustra o desenvolvimento de recursos como uma pirâmide. O ambiente de desenvolvimento de, ferramentas de hardware e software localizam-se na fundação da pirâmide e fornece a infrastructura para suporte do esforço do desenvolvimento.

No nivel alto, encontra-se componentes de software reusáveis, blocos de construção de software que podem drasticamente reduzir o custo de desenvolvimento e acelerar a entrega. No topo da priamide esta o primeito recurso-pessoa

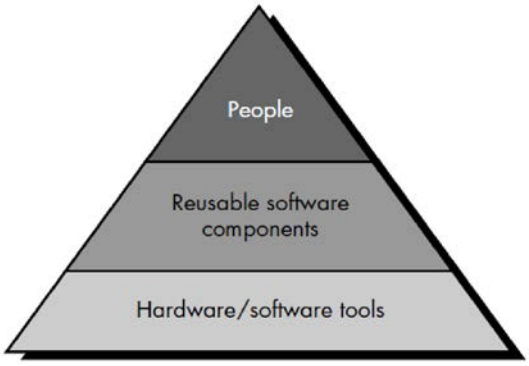

Figura 2.2: Projecto de Recurso Sorce: Pressman, 2011

Estimativa do projecto de software.

Estimativa efectiva do projecto de softwate é um dos desafios mais importantes e importante actividade no desenvolvimento de software. Planear um projecto adequado e controle não é possivel sem uma estimativa confiável. As seguintes sub unidades descrevem com mais detalhadamento sobre estimativa de projecto de software.

Razões para estimativas pobres e imprecisas

As razões a seguir são algumas das razões para estimativas pobres e imprecisas:

- Requisitos imprecisos. Requisitos mudam frequentemente
- projecto é novo e diferente do passado projecto elaborado.
- Não há disponibilidade suficiente de informação sobre projectos passados.
- Estimativas são forcadas a ser baseadas em recursos disponíveis
- Cost and time tradeoffs.
- Custo e tempo

Probelmas associados com estimativas:

Estimar o tamanho muitas vezes é posto de lado e o tempo estimado, que é mais relevante para gestão.

Estimar o tamanho é talvez o passo mais difícil, que tem uma influência em todas outras estimativas

Boas estimativas são somente projecções e sujitos de varios riscos

Muitas vezes organizações dão pouco importância para coleta e analise do histórico hostorio de dados de projectos desenvolvidos no passado. Histórico de dados e a melhor entrada para estimar um novo proejcto.

Gestores de projecto muitas vezes não estimam o tempo porque gestão e clientes as vezes hesitam em aceitar o tempo de execução realístico.

Linhas de estimativas de projecto

Algumas linhas para guias as estimativas de processo são:

Preservar e documentar dados relativos a projectos anteriores

Permitir tempo suficiente para estimação do projecto especialmente para grandes projectos.

Preparar estimativas de desenvolvimento básico realísticas. Associar as pessoas que vão trabalhar no projecto para alcançar uma estimativa mais realística e precisa.

Usar ferramentas de estimativa de software

Re-estimate the project during the life-cycle of the development process.

Analizar erros passados para estimativas do projecto

Conclusão

Um dos problemas no desenvolvimento de software é a estimativa de recursos necessarios para o proejcto. Desenvolver um plano de projecto prático é essencial para entender os recursos necessários, e como estes devem ser aplicados.

Avaliação da Unidade

Verifique a sua compreensão

1. Porque a estimativa de custo tem um papel importante no processo de desenvolvimento do software.
2. Claramente descreve o processo para estimativa de projecto.
3. Explica o termo esforço de estimativa na vista de custo de estimativa de software,
4. Quais são as várias razões para um poor/inaccurate estimation
5. Explique a importância de requisitos. Como muitos tipos de requisitos são possíveis e porque?

Resumo da Unidade

O plano de software de projecto deve estimar três coisas antes de se enviar o projecto: quanto tempo vai levar, quanto esforço será necessário e quantas pessoas estarão envolvidas. A unidade apresentou requisitos de engenharia e o processo para obter as especificações de requisitos. Tambem abordou sobre planeamento de projecto de software e especificação do custo de estimativa de projecto.

Leituras e Outros Recursos

Livros Recomendados

- Agawal B. B., Tayal S. P. and Gupta M., (2008), "Software Engineering and Testing" Jones and Bartlett Publisher, ISBN: 978-1-934015-55-1
- Sommerville Ian, (2000): Software Engineering (6th Edition). Addison-Wesley, Boston USA
- Pressman Roger S., (2001), "Software Engineering, A Practitioner' S Approach" Fifth Edition, McGraw-Hill Higher Education, ISBN 0073655783
- David Gustafson, (2002), "theory and Problems of Software Engineering", Schaum's Outline series, McGRAW-HILL, 0-07-140620-4

Unidade 3: Desenho de Sistemas

Introdução a Unidade

Desenho de sistema abrange um conjunto de principios, conceitos e praticas que conduzem ao desenvolvimento de sistemas ou produtos com uma elevada qualidade. Principios de desenho estabelecem uma filosofia primordial que guia o utilizador no trabalho de desenho que lele deve fazer. Os conceitos de desenho devem ser entendidos antes de toda a mecanica pratica do desenho ser aplicada. Assim sendo, esta unidade vai abordar todos os aspectos referentes ao desenho de um software.

Objectivos da Unidade

Após a conclusao desta unidade voce deverá ser capaz de:

- Indetificar todos os passos envolvidos no desenho de sistemas
- Descrever quais sao os pontos principais no desnho de sistema
- Descrver o que entende por desenho de sistemas

Termos-chave

Requisito: O Desenho de Software: É a pratica de pegar especificacoes de compotrtamento externos observaveis e adcionar detalhes necessarios para a implementacao de um sistema de computador, incluindo interacao humana, gestao de tareas e gestao detalhada de dados.

Princípios de Desenho: Desenho de software corresponde a processo e modelo. O processo de desenho é uma sequencia de passos que permitem o projectistas descrever todos os aspectos para construir o software.

Actividades de Aprendizagem

Actividade 1: Desenho no Contexto da Engenharia de Software

Desenho de sistema assenta no kernel tecnico da engenharia de software e é aplicado apesar de o modelo de processo de software que é usado. Iniciando depois dos requisitos de software serem analizados e modelados, desenho de sistema e a ultima accao da engenharia de software com o modelo de actividade e assenta nos estagios de construcao (geração de código e testes) como ilustra a figura 3.1.

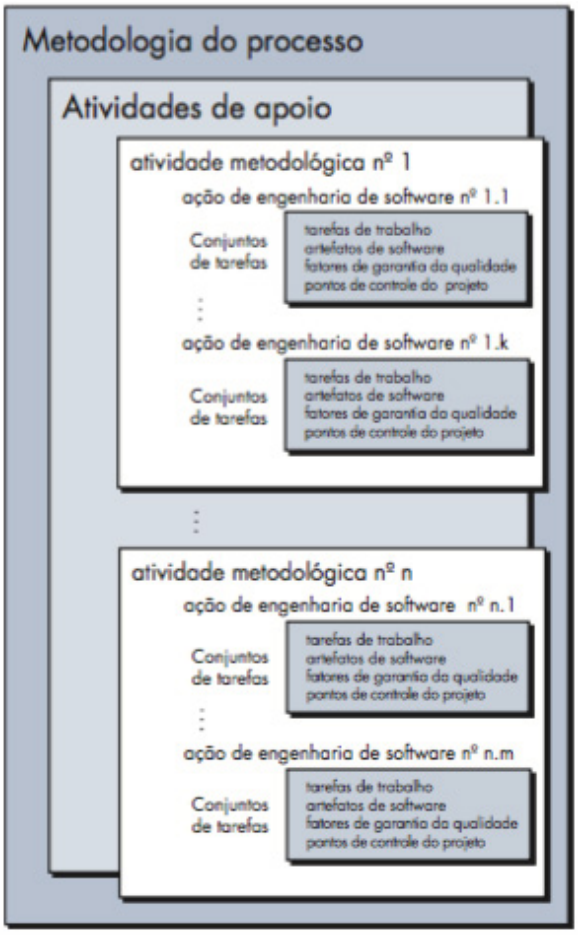

Figura 3.1: Modelo de processo de software (Roger Pressman.2011)

Actividade 2: O Processo de Desenho

Desenho de software e um processo interactivo sobre que requisitos sao traduzidos num blueprint para construcao de software.

Inicalmente, o blueprient representa uma vista holistica de software. Isto é o deseho é representado no nivel alto de abstracao, um nivel que pode ser direcatmanete esbocado para especifcar o objectivo do sistema e dados mais detalhaddos, funcionais, e requsitos comportamentais.

Qualidade de Sosftware, Guiões e atributos

Atravez do processo de desenho, a qualidade da envolvimento de desenho é feito com uma serie de tecnicas revistas. Existem tres caracteristicas sugeridas pelo McGlaughlin que servem de guia para avaliar um bom desenho:

Desenho deve implementar todas os requitiso explicitps contidos nos requsitos do modelo, e devemacomodar todas os requisitos implicitps desejaddo pelos stakeholders.

O desenho dese ser de facil leitura, um guai comprendido para aqueles que geram os codigos e para aqueles que testam e posteriomente trabalham como suporte do software.

O desenho deve prover uma completa figura do software, localizando os dados, funcional, e dominios comportados sob pontos de vista de uma perspectiva implementacao.

Guia de Linha de Qualidade

No sentido de avaliar a qualidade de representacao do desenho, voce e outros membros da euqipe de software devem estabeecerr criterios tecnicos para um bom desenho. Assim, sendo iremos analisar alguns crietriso consederando os seguintes:

1. O desnho deve exibir uma arquitectua (1) foi criado atraves de estilos de arquitecturas reconhecidas ou padroes, (2) é composto por componetes que mostram boas carateiriticas de desenho e (3) podem ser implementados num forma evolucionaria.
2. O desnho deve ser modular, isto é, o software deve ser particonado logicamente em elementos ou subsistemas.
3. O desenho deve conter representacaoes distintas de dados, arquitectura, interfaces e componentes.
4. O desenho deve lidar com estrutras de dados que sao apropriados para classes poderems ser implementadas e sao desenhadas de padores de dados reconhecidos.
5. O desenho deve lidar com componentes que mostrem caracteristicas fubcionais independentes.
6. O desnho deve lidar com interfaces que reduzem a complexidade entre os componestes e o ambiente externo.
7. O desenho deve ser derivado de metodos repetidos que é guiado por informacoes obtidas durante a análise de requisitos de software.
8. O desnho deve ser representado usando a notacao que efectivamente comunica o seu significado.

Estes guias de desenho nao sao alcandos por chances. Eles sao alcancados atraves de aplicacao de principios fundamentais de desenho, metodoliga sistematica e atraves de revisao.

Qualidade de Atributos

Hewlett-Packad desenvolveu um conjnto de atributos de qualidade de software que foi dado o acronimo de Funcionalidade, Usabilidade, Confianca, Desempenho e Suportabilidade (FURPS). Os atributos de qualidade FURPS representam uma mets para todos os desenhos de sistema.

Funcionalidade é avaliado pela avaliacao de um conjunto de carateiristicas e capacidades de um programa, a generalidade de funcoes que sao entregues e a seguranaca de todo o sistema.

Usabilidade é avaliada considerando factores humanos, todoa estetica, consistencia e documentacao.

Confinaca é avalidado medindo a frequebcia e a gravidade da falas, o exatidao dos resultados de saida, o Mean-Time-to-Failure (MTTF), a capacidade de recuperar-se de uma falha e a predicao do programa.

Desempenho é medido considerando o processo de velocidade, tempo de resposta, comsumo de recuros, eficiencai e throughput.

Portabilidade combina a abilidade de extender o programa (extensabilidade), adaptabilidade, utilidade, estes atributos representam o termo commum, manutencao e em adicao testavel, compativel, configuravel (a habilidade de reconhecer e controlar elementos de software de configuracao).

Nao é todo o atributo de qualidade de software é pesado igualmente como o desenho de software desenvolvido. Um a aplicacao pode baralhar a funcionalidade com um enfase especial na seguranca. Outra demanda de desempnho com particular enfase para a veclocodade de processamento é possivel. A terceira aposta pode focar na confinaca.

Avaliação

Verifique a sua compreensão

1. Defina o que é desenho arquitectural
2. Quais são os objetivos do desenho arquitectural?
3. Explique as várias técnicas de desenho que vem debaixo da categoria de desenhos de sistemas de baixo nível.

Actividade 3: Outros aspectos de Desenho de Software

Abordagem Funcional versus abordagem orientada a objetos

A tabela 3.4.1 explica as diferenças entre modelo funcional versus modelo orientado a objetos na abordagem de desenho de software.

S-N	Abordagem Funcional	Abordagem Orientada a Objectos
1	Abstrações básicas que são dadas aos utilizadores são funções do mundo real como, classificar, mesclar, visualizar, etc..	A abstração básica não são funções do mundo real, mas são abstrações de dados que são representadas por entidades do mundo real como imagens, máquinas, sistemas de radares, clientes, estudantes ,trabalhadores, etc..
2	Funções estão agrupadas em conjunto por funções de nível alto que são obtidas. Um exemplo desta técnica é a análise de Software-Design (SA-SD.)	As funções são agrupadas com base nos dados em que operam, tais como na classe pessoa.
3	O estado da informação é muitas vezes representado em memórias partilhadas centralizadas	O estado da informaçãoo não é representado em memórias partilhadas centralizadas, mas é implementado-distribuído entre os objetos do sistema.

Tabela 3.4.1: Diferença entre abordagem funcional e orientada a objectos

Especificções de Desenho

Especifições de desenho lidam com diferentes aspectos de modelo de desenho e são completas na medida em que os projectitas redefinem as suas representações de software.

Primeiro, o alcance global do esforço do projecto é descrito, que deriva da especificação do sistema e o modelo de análise (especificação de requisitos de software)

Depois, o desenho de dados é especificado, que inclui estruturas de dados, qualquer arquivo externo, as estruturas de dados internas e uma referencia cruzada que conecta objetos de dados em arquivos específicos

Depois, o desenho arquitectónico indica como a arquitectura do programa foi derivada do modelo de análise.

Especificação de desenho contém uma referencia cruzada de requisitos. A finalidade da referencia cruzada é:

Fazer com que todos os requisitos sejam satisfeitos pelo design de software

Indicar quais componentes são críticos para a implementaçãoo das exigências específicas.

A secção final da especificação de projecto contém dados complementares, tais como descrições de algoritmo, procedimentos alternativos e os dados tabulares.

Verificação do Desenho

Como em outras fases no processo de desenvolvimento, a sáida da fase de projecto do sistema deve ser verificada antes de prosseguir com as atividades da fase seguinte.

Se o desenho é expresso em algumas notações formais no qual ferramentas de análise estão disponíveis, então através da ferramentas pode ser verificado por inconsistência interna (por exemplo, os módulos usados por outro são definidos, a interface de um módulo é consistente com a forma como outros usam, o uso de dados é consistente com a declaração.etc.)

Se o desenho não é especificado em uma linguagem executável formal, este não pode ser processado através de ferramentas e outros meios de controle tem de ser utilizados.

Avaliação

Verifique a sua compreensão

1. Voce desenha software quando escreve um programa? O que torna o desenho de software diferente do código?
2. Defina o que é: Probelma de partição, Abstração e Abordagem Top-doen e bottom-up.

Resumo da Unidade

Desenho de software comeca com uma primeira interacao de engenharia de requisitos vêm a uma conclusao. A intencao de desnho de software é aplicar um conjunto de principios, conceitps e praticas que lidam com o desenvolviemnto de sistemas ou produtos de alta qualidade.

O objectivo de desnho é criar um modelo de software que vai implementar todos os requistiso do cliente correctamente e trazer o prazer a aqules que o usam. Desenhadores de software devem examinar minunciosamnete muitas alternativas de desenho e abrangir uma solucao que se enquadra nas necesidades de projecto dos stakeholders.

O processo de desenho move-se de uma vista grnade fugura de software para uma vista narrativa que define os detalhes necessaros para implementar o sistema. O processo imicia com a atenacao para a arquitectura. Subsistemas sao definidos; mecanismos de comunicacao sobre subsistemas sao estabelicidos; componetes sao identificados e descricoes detalhadas de cada componete e desenvolvida. Além disso, externa, interna e utilizadores de interface sao desenhados.

Avaliação da Unidade

Verifique a sua compreensão

1. Mencione duas importantes diferenças entre modelo funcional e abordagem orientada a objetos.
2. Discura as vantagens da abordagem orientada a objetos em ralação a obordagem funcional.
3. Explique o termo especificação de desenho
4. Discuta o termo verificaçãoo em referencia ao desenho de sistemas
5. O que é abstração? Quais são as métricas de verificaçãoo para desenho de sistemas?

Leituras e Outros Recursos

- 1. Roger S. Pressman, Software Engineering, A Practitioner's approach, 2010 ISBN 978–0–07–337597–7 MHID 0–07–337597–7
- 2. Agawal B. B., Tayal S. P. and Gupta M., (2008), "Software Engineering and Testing" Jones and Bartlett Publisher, ISBN: 978-1-934015-55-1
- 3. Acessado em Abril de 2016. http://www2.unemat.br/rhycardo/download/ciclo_de_vida.pdf

Unidade 4 : Implementação e Testes

Introdução a Unidade

Testar o software é forma de verificar se o mesmo responde as especificações definidas desde a fase de concepção até a implementação do mesmo.

Esta unidade vai abordar de uma forma geral todos os aspectos relacionados com a implementaçao e teste do software. Vai-se abordar técnicas usadas para implementar e os mecanismos usados para realizar teste do software desenvolvido, mas também os apresentar que técnicas exploradas são exploradas neste processo todo.

Objectivos da Unidade

Após a conculsão desta unidade deverá ser capaz de:

- Descrever as diferentes técnicas usadas para realizar testes
- Identificar as fases de realização de teste
- Definir as diferentes técnicas de testes e distinguir as suas diferenças
- Descrever o propósito de escrever planos de Teste

Termos-chave

Teste:É o processo pelo qual executa-se um programa com o objectivo de encontrar erros.

Teste de Oráculos:De acordo com Agarwal et all, teste de oráculo é um mecanismo diferente do programa em si que, pode ser utilizado para coraccao das saídas do programa para os casos de teste.

Teste de Software: Teste é um conjunto de atividades utilizadas para testar o código fonte no sentido de encontrar (e corrigir) erros antes que o software seja entregue ao cliente.

implementação.

Caso de Teste: Um caso de teste é um conjunto de instruções destinadas a descobrir um determinado tipo de erro ou defeito no sistema de software através de injecção de falhas.

Teste Estructural:Teste estructural é uma abordagem de testes quando os testes são derivados a partir do conhecimento da estrutura do software e

Testes Funcionais: Os testes funcionais referem-se a testes que envolvem apenas a observaçãoo da saída para determinados valores de entrada e não há qualquer tentativa para analisar o código que produz a saída.

Actividades de Aprendizagem

Actividade 1. Codificação do Software

Introdução

Durante a fase de implementação, cada componente do desenho é percebido como uma unidade do programa. Cada unidade deve ser verificada ou testada contra o sua especificação obtida na fase de desenho. Este processo é explicado na figura 4.1. Depois as unidades individuais do programa representando componentes do sistema são combinados e testados como um todo para garantir que os requisitos de software são atingidos. Quando os desenvolvedores estão satisfeitos com o produto, estes são testados depois pelo cliente. Esta fase termina quando o produto é aceite pelo cliente.

Codificação

Durante a fase de codificação o focu esta no desenvolvimento de progremas que são fáceis de ler e perceber e não é simples no desenvolvimento de programas que são simples de escrever.

Técnica Estruturada

Existem algumas técnicas utilizadas no desenvolvimento de softwares. A técnica estruturada engloba outras técnicas de produção de sistemas, como: análise estruturada, projecto estruturado, programação estruturada, desenvolvimento top-down, equipes de programação e revisões estruturadas.

Análise estruturada: a análise estruturada refere-se ao extremo inicial de um projecto de desenvolvimento de sistemas, durante o tempo em que os requisitos do utilizador são defenidos e documentados. Por que análise estruturada e tão importante? Simplesmente porque a análise clássica não respondeu as necessidades. O problema é muito simples de ser expressado: em qualquer coisa diferente de um projecto de desenvolvimento de sistemas trivial, o utilizador não tem uma boa compreensão do que esta ser feito para ele. Assim, os problemas da análise clássica podem se resumidos como:

- As especificações funcionais clássicas são monolíticas
- As especificações funcionais clássicas são redundantes
- As especificações funcionais clássicas são difíceis de se modificar ou manter
- As especificações funcionais clássicas fazem muitas suposições sobre os detalhes de implementação. ()

Projecto Estruturado

Programação Estruturada

De acordo com () a programação estruturada foi a primeira uma das primeiras técnicas a ser discutida e praticada amplamente. Para muitas pessoas, o termo programalçao estruturada é genérico e inclui filosofias de projecto, estrátegias de implementaçãoo, conceitos de organização de programas, e as virtudes básicas de prosperidade, lealdade e bravura.

A teoria por trás da programação estruturada é relativamente simples: qualquer lógica de programa pode ser construída pelas combinações das três estruturas como sequencia, decisão e repetição. Na maioria das línguas de programação de alto nível isto significa que os programas de computador podem ser construídas com as seguintes combinações:

- Comandos imperativos simples, como: leia, calcule, escreva e comandos lógicos;
- Comandos para a tomada de decisão como: se e caso;
- Comandos de repetição ou iteração como: repita até que ou faça enquanto;

O ponto principal é que este conjunto de constrições é suficiente para formar todo e qualquer tipo de lógica de programa em particular não é necessºario usar os camandos de desvio em programação.

Desenvolvimento Top-Down

O desenvolvimento top-down muitas vezes e confundido com o projecto top-down. Assim existem três aspectos relacionados, mas distintos da seguinte forma:

Projecto top-down: é uma estratégia de projecto que divide problemas grandes e complexos em problemas menores e mais simples, facilmente solucionáveis;

Codificação top-down: é uma estratégia para se codificar módulos de alto nível, antes que os módulos detalhados de nível mais baixo;

Implementação top-down: é uma estratégia para testar os módulos de alto nível de um sistema antes que os módulos de baixo níveç tenhma sido codificados e possivelmente, até antes que tenham sido projectados.

O desenvolvimento top-dow consiste em desenvolver inicialmente uma versão inicial com todos os módulos de alto nível do sistema. As versões subsequentes simplesmente envolvem o acréscimp de módulos de nível mais baixo ao esqueleto já existente dos módulos de alto-nível, até que uma versão final produz a saída satisfatória ao utilizador.

Ferramentas de Análise Estruturada

O Diagrama de Fuxo de Dados (DFD) é uma representação dos processos do sistemas e dos dados que ligam esses processos. Ele mostra o que o sistema faz e não como é feito. É a ferramenta de demonstraçãoo central da análise estruturada.

Um DFD apresenta as partes componentes de um sistema e as interfaces entre elas. É um conjunto integrado de procedimentos, sendo que as partes dos computados poderão estar inseridos ou não.

Na elaboração de um DFD, são utilizados quatro símbolos que permitem debater e apresentar ao utilizador todo o processo, sem assimir nenhum compromisso com implementações e demonstrar a sua fluência . Assim, os seguintes símbolos são utilizados na elaboraçãoo de um DFD como ilustra a Figura 4.2.

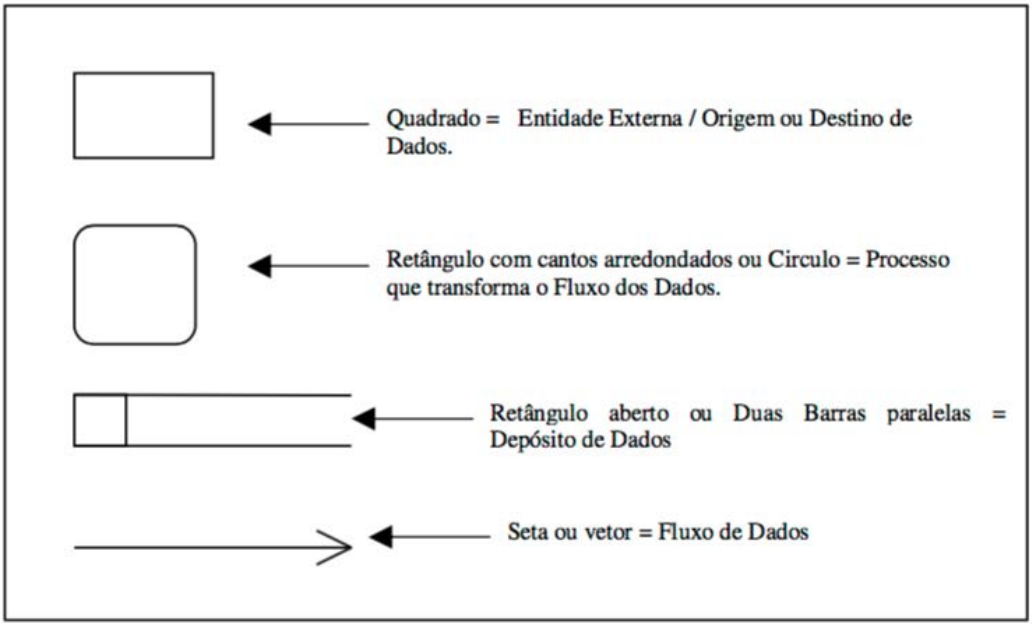

Figura 4.2: Simbolos de um DFD ()

Avaliação

Verifique a sua compreensão

1. O que é programação estructurada?
2. Como é que as linguagues de programação modernas, facilitam a escrita de programas estruturados? (Use qualquer linguagem de programação que você esta familiarizado com a mesma)
3. Qual é a vantagem de escrever programas estruturados versus programas não estructurados?

Actividade 2 Fundamentos de Teste de Software

Uma estratégia de teste de software deve acomodar testes de baixo nível, necessários para verificar se um pequeno segmento de código fonte foi implementado corretamente, bem como teste de alto nível que validam as funções principais do sistema de acordo com os requisitos do cliente. Uma estratégia deverá fornecer directtrizes para o profissional e uma séria de mestas para o agente.

Princípios de Teste

Introdução

Existem varias formas de testar um software que são aplicadas nos dias de hoje. Antes de um engenheiro de software fazer uso das técnicas de teste, este deve conhecer os princípios básicos que guião o processo de testes. Assim sendo, de acordo [Nome do Autor] com define que os seguintes princípios são os básicos:

a) Todos testes devem ser determinados de acordo com os requisitos do cliente

O propósito é descobrir possíveis defeitos ou falhas que façam com que o sistema a nao funcione de acordo com os requisitos do cliente.

b) Os testes devem ser planificados antes mesmo de iniciarem:

Depois de se terminar o processo de análise de requisitos o planeamento de teste pode iniciar. Casos de teste detalhados podem iniciar logo que o modelo de desenho termina.

c) O principo de Pareto é aplicado a teste de sosftware

Simples, o principio de Pareto defende que 80 por cento de todas as falhas descobertas durante a fase de testes pode afectar 20 por cento de todo os componentes do programa. O problema nesta fase é isolar esses componentes suspeitos testando os mesmos.

d) Teste devem começar de pequeno para um todo

Normalmente os primeiros testes planificados e realizados geralmente concentram-se em componentes individuais.

e) Não é possível realizar teste minuncioso

f) Para ser mais eficiente teste devem ser conduzidos por uma terceira parte independente.

O engenheiro de sosftware que cria um sistema nao é a melhor pessoa para realizar todos os teste do programa.

Objectivos de Teste

A fase de implementaçao e testes é composta por várias subfases e cada uma delas aborda com detalhes todos os aspectos necessários para garantir um processo eficaz. Por outro lado esta fase tambem apresenta alguns objectivos. Os objectivos asseguram que a partir do momento em que se realizam os testes no código do programa há maior probabilidade de descobrir erros. Isto vem sustentar que as funcionalidades de um software funcionam de acordo com as especificacoes relacionadas com a funcao, facildades, features e desempenho.

Outro aspecto que é abordado nos objectivos é que os teste de principio detectam eros no código do programa, mas estes nao vao mostar qualquer erro se o codigo nao corresponder as especificacoes estipulados na especificacao de requisisots de software (software requiremetns specification) SRS.

Os seguites objectivos abaixo mencionados sao considerados os objectivos de teste:

Teste é o processo de executar um programa com a finalidade de encontar erros.

Um bom caso de teste de é um teste que apresenta uma maior probabilidade de encontar qualquer erros que nao foi descoberto.

Um teste bem sucedido é o descobre qualquer erro que nao que nao tenha sido descoberto.

Teste de Oráculos

Teste de oraculos são considerados como humanos, eles podem realizar testes quando existe uma discrepancia entre os oraculos e os resultados do programa. Primeiro é necessário verificar os resultados produzidos pelos oraculos antes de declarar que existe uma falha no programa. Esta é uma da razoes de teste ser muito caro. A Figura 4.3 ilustra o principio de teste de oráculos.

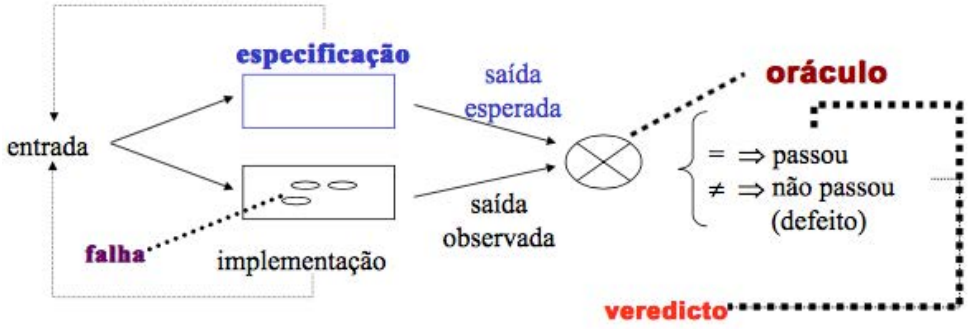

Figura 4.3 : Teste de Oráculos

Na imagem acima ilustrada tem-se o seguinte: o caso de teste é dados de acordo com o teste de oraculo e o programa sobre que vai ser testado. O resultado de cada um dos elementos no final é comparado para determinar se o progrma comportou-se correctamente de acordo com o caso de teste.

Os oraculos humanos geralmente usam as especificacoes do programa para decidir o qual deve ser o comportamento correcto do programa. Para ajudar nesse processo normalmente é essencial que o comportamento do sistema deve ser nao ambiguo (unambiguosly) especifico e a especifacao deve ser livre de falhas.

Por outro lado, existem nives de teste, isto é existem tres modulos individuais em todo os sistema de software que sao:

Teste de Unidade

No teste de unidade componentes individuais sao testados para garantir que funcionem correctamente. Concentram-se no esforco de verificacao. Nas pequenas unidades do desenho do software, cada componente é testado de forma independente sem os outros componentes de sistema. A figura 4.2 ilustra como é realizado o teste de unidades.

Neste teste focaliz-ase o esforço de verificação da menor unidade de projecto de software, o componente ou módulo de software. Utilizando como guai a descrição de projecto do nível de componente, caminhos de controle importantes são testados para descobir erros dento dos limites do módulo.

A interface do módulo é testada para ssegurar que as informações fluam corectamente para dentro e para fora da unidade de programa que esta sendo testada. A estrutura de dado local e examinada para garantir que os dados armazenados temporariamente mantenham sua integridade durante todos os passos na execuçãoo de um algoritmo. Todos os caminhos independentes da estrutura de controle são usados para assegurar que todas as instruções em um módulo tenham sido executadas pelo menos uma vez. As condicções limite são testadas para garantir que o módulo opere adequadamente nas fronteiras estabelecidas para limitar ou restringir o processamento. Finalmente, são testados todos os caminhos de manipulação de erro. (Pressman, 2011)

Fugura 4.4: Teste de Unidades (Pressman, 2011)

Teste de Integracao

Outro nivel de teste é o teste de integracao. Teste de intergacao é uma tecnica sistematica para contruir a estrutura do programa enquanto que ao mesmo tempo sao realizados testes de uncover erros associados com a interface.

Neste tipo de teste muitos modulos de unidades de teste sao combinados em subsistemas que sao testados. O objectivo é ver se os modulos podem ser intergardos de forma apropriada.

Abordagens para Teste de Integracao

Algumas das várias abordagens usadas para realizar teste de integracao sao:

- Abordagem Incremental
- Intergracao Top-dow
- Integracao Bottom-up
- Teste Regressivo
- Teste de Smoke
- Integracao Sandwich

Teste de Sistema

No teste de sistema subsistemas sao integrados para fazer o sistema todo. O processo de teste is conserned com encontrar erros que resultam de nao antecipadas interacaoes entre os subsistemas e os componentes do sistema. É tambem concerned com avalidacao que o sistema deve encontrar os requisistos funcionais e nao funcionais. Assim sendo, as tres essensias formas de testar o sistema sao:

- Tesde de Alpha
- Teste de Besta
- Teste de Aceitacao

A Figura 4.5 ilustra que a estratégia de teste de software pode ser visto também no conceito espriral.

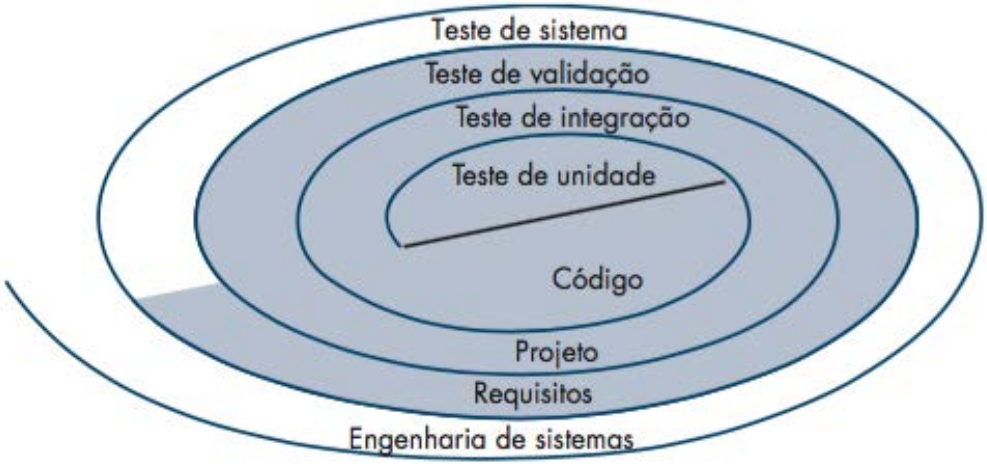

Figura 4.5: Teste de Software (Pressman, 2011)

Planeamento de Teste

O planeamento de teste é um documento que contem diferentes casos de teste desenhados para diferentes objectso de teste e diferentes teste de atributos. O plano poe o teste de uma forma logica e ordem sequencial de acordo com a estrategia escolhida, top-dow ou bottom-up. Normalmente o plano de teste é uma matriz de teste e casos de teste listadaos de acordo om a ordem de execuacao de cada tarefa. A tabela abaixo ilustra um exemplode um plano de execuacao.

Teste									
Id do Teste		Teste	1	2	3	....	N	Dias Planeados	
ID	Tester	Nome						Completos	Succsesseful

Avaliação

Verifique a sua compreensão

1. O que é teste?
2. Explique os diferentes tipos de testes realizados durante o desenvolvimento de software.
3. Defina os vários princípios de teste.
4. O que é teste de oráculos?

Actividade 3: Visões Interna e Extena do Teste

Segundo (Pressman, 2011) qualquer produto de engenharia pode ser testado por uma ou duas formas:

i, conhecendo a funlçao especifica para o qual o produto foi projectado para realizar , podem ser feitos teste que demostrem que cada uma das funções é totalmente operacional, embora os mesmo tempo procurem erros em cada função,

ii) conhecendo o funcionamento interno de um produto podem ser realizados testes para garantir que tudo se encaixa, isto é que as operações internas forma realizadas de acordo com as especificações e que todos componentes internos adequadamente exercitados. A primeira abordagem de teste usa uma visão externa e é chamada de teste caixa-preta. A segunda requer uma visão interna e é chamada de teste de caixa-branca.

O teste de caixa-preta faz referencia a teste realizados na interface do doftware. Um teste de caixa-preta examina alguns aspectos fundamentais de um sistema, com pouxa preocupação em relação a estrutura lógica interna do software.

O teste de caixa-branca fundamenta-se em um exame rigoroso do detalhe procedimental. Os caminhos lógicos do software e as colaborações entre componentes são testados exercitando conjuntos específicos de condições ou ciclos. (Pressman, 2011)

Teste de caixa-Branca

O teste de caixa-branca, é uma filosofia de projecto de casos de teste que usa a estrutura de controle descrita como parte do projecto no nível de componentes para derivar casos de teste. Usando métodos de teste de caixa-branca o engenheiro pode criar casos de teste que (1) garantam que todos os caminhos independentes de um módulo sejam exercitados pelo menos uma vez, (2) exercitam todas as decisões lógicas nos seus estados verdadeito e falso (3) executam todos os ciclos em seus limites e dento de suas fronteiras operacionais, e (4) exercitam estruturas de dados internas para assegurar a sua validade. (Pressman, 2011)

Teste de Caixa-Preta

Teste caixa-preta, também chamado de teste comportamental, focaliza os requisitos funcionais do software. As técnicas de teste caixa-preta permitem derivar séries de condições de entrada que utilizarão completamente todos os requisitos funcionais para um programa. O teste caixa-preta não é uma alternativa às técnicas caixa-branca. Em vez disso, é uma abordagem complementar, com possibilidade de descobrir uma classe de erros diferente daquela obtida com métodos caixa-branca.

O teste caixa-preta tenta encontrar erros nas seguintes categorias: (1) funções incorretas ou faltando, (2) erros de interface, (3) erros em estruturas de dados ou acesso a bases de dados externas, (4) erros de comportamento ou de desempenho, e (5) erros de inicialização e término.

Diferentemente do teste caixa-branca, que é executado antecipadamente no processo de teste, o teste caixa-preta tende a ser aplicado durante estágios posteriores do teste Devido ao teste caixa-preta propositadamente desconsiderar a estrutura de controle, a atenção é focalizada no domínio das informações. Os testes são feitos para responder às seguintes questões:

- Como a validade funcional é testada?
- Como o comportamento e o desempenho do sistema é testado?
- Que classes de entrada farão bons casos de teste?
- O sistema é particularmente sensível a certos valores de entrada?
- Como as fronteiras de uma classe de dados é isolada?
- Que taxas e volumes de dados o sistema pode tolerar?
- Que efeito combinações especí cas de dados terão sobre a operação do sistema?

Aplicando técnicas caixa-preta, você cria uma série de casos de teste que satisfaz aos seguintes critérios: (1) casos de teste que reduzem, de um valor maior do que um, o número de casos de teste adicionais que devem ser projetados para atingir um teste razoável; e (2) casos de teste que lhe dizem alguma coisa sobre a presença ou ausência de classes de erros, em vez de um erro associado somente com o teste especí co que se está fazendo. (Pressman, 2011)

Resumo da Unidade

Nesta unidade todos os apectos relacionados com a fase de implementacao e testes foram abordados de forma detalhada. Portanto, é tambem explicado algumas as prativas de engenharia de software implementadas durante a fese de programacao.

Ainda neste unidade forma discutidas a necessidade de documentaçao dos softwares produzidos e estrategias de solucionar falhas encontradas no decorrer do desenvolvimento do software. Por outro lado é aborda-se todos os aspectos relacinados com teste de programas. A diferença entre as várias técnicas de teste como teste de unidade, integraçao e sistema foram explicadas em detalhes. A necessidade de implementar um plano de teste, ou um ciclo de vida de testes de software e como as técnicas e ferramentas sao integradas neste foi discutido neste capítulo.

Avaliação da Unidade

Verifique a sua compreensão

1. O que é teste?
2. Defina os princípios de teste.
3. O que é teste de oraculos?
4. O que e teste de unidade?
5. O que e teste de sistema?
6. O que e teste de integraçao?
7. Suponha que um desenvolvedor de software passou com sucesso todas as fases de teste, desde da unidade até a integraçao. Podemos considerar que o software esta sem ameaças. Justique a sua reposta
8. Explique os diferentes tipos de teste realizados durante o processo de desenvolvimento de software?
9. Qual e a diferencça entre:

    - Teste de caixa branca e caixa preta
    - Explica as categorias de teste de caixa preta?

Leituras e Outros Recursos

- Agawal B. B., Tayal S. P. and Gupta M., (2008), "Software Engineering and Testing" Jones and Bartlett Publisher, ISBN: 978-1-934015-55-1
- Forrest Shull, Roseann Tesoriero, A study guide to acompany Shari Lawrence PfleegerSoftware Engineering: Theory and Practice
- Prof. Walteno Martins Parreira Júnior. Apostila de Engenharia de Software. Universidade do Estado de Minas Gerais. Disponivel em: http://www.dai.ifma.edu.br/~mlcsilva/aulas_modelagem/recursos/apostila_EngSoftware.pdf
- Roger S. Pressman. Engenharia de Software, Uma Abordagem Profissional, Sétima Edição. 2011

Unidade 5: Gestão do Projecto e Manutenção

Introducao a Unidade

Nesta unidade vai-se focar todos os apectos relacionados com a gestao e manutencao do software.De acordo com Agarwal et all gestao do projecto é uma actividade de tentar garantir que o processo de desenvolvimento de software seja bem sucedido. Neste aspecto o significado de sucesso difere de um para outros projectos, mas normalmente iclui atingir o deadline, implementado as carateristicas necessárias e meeting the budget.

Objectivos da Unidade

Após a conclusao deverá ser capaz de:

- Indentificar os requisitos essencias para gestao e mantencao do software desenvolvido
- Analisar o processo de gestao e manutecao
- Distiguir as fases de gestao e manutencao

Termos-chave

Gestão de configuração: É a arte de identificar, organizar e controlar modificacoes feitas no software por uma equipe de programacao. O objectivo é maximizar a produtividade minimizando os erros.

Gestão de configuração de software: É um conjunto de actividades desenhadas para controlar as mudancas identificando o produto de trabalho que sao frquentes de mudar, estabelecendo relacoes entre eles, definindo mecanismos diferentes versoes dos mesmos produtos de trabalho, controlando as mudancas impostas e adicionadas e reportadas na mudancas feitas

Gestão de Risco: Gestão de risco é area que tenta garantir que o impacto do custo dos riscos, qualidade e schedule é minimo.

Actividades de Aprendizagem

Actividade 1 Gestão do Projecto de Software

Gestao de projecto é a actividade de tentar garantir que o desenvolvimento do sosftware seja bem sucedido. O significado de sucesso difere de um projecto a outro, mas normalmente inclui terminar o mesmo dentro do limite traçados, implementado as caracteristicas requisitadas e o dinheiro alocado. [2)

A gestão da configuracao de um software é feita tendo em conta três fases essências.

1. Indentificacao da configuracao
2. Controlo da configuracao
3. Contabilidade da configuracao

De acordo com os autores Agarwal et all, as três actividades sao resumidas em:

Indentificacao da configuracao: consiste em identificar qual é a parte do sistema que deve ser guardada.

Controlo de configuracao: Assegura que as modificacoes de um componente acontenssam lisamente.

Contabildade de configuracao: guarda todas as fases do que foi modificado, quando e porque.

Necessidade de Manutenção do software

A gestão de software é uma actividade que esta associada com o processo de mater sempre continuamente operacional o sistema de computaçao em conjunto com os requisistos dos utilizadores e operacaoes de processamento de dados. O processo de gestao de software normalmente é caro e os riscos sao um desafio bastante grande. De acordo com os autores Agarwal et all, a necessidade de manutencao do software é tida pelas seguintes razoes:

- Mudanças dos requisistos do utilizador com o passar do tempo
- Problemas do programa e sistema
- Tornar o código facil de entender e trabalhar
- Eliminar qualquer desvio da sua especificaçao
- Rever os standards e eficiência
- Modificar os componenstes

Categorias de Gestão

A gestao do software é tambem calssificada nas seguintes categorias:

- Correctiva: alterações reactivas para corrigir problemas
- **Adaptativa**: modificações para manter o produtio em uso em ambientes de constante mudanças.
- **Perfectiva**: melhorar o desempenho ou a manutenção
- **Preventiva**: modificações para detectar e corrigir falhas latentes.

Manutenção correctiva: significa reparação de falhas em processamento e desempenho ou fazer mudanças devido a falhas prévias não corrigidas. Envolve a correçãoo de erros que não foram descobertos em fases anteriores do processo de desenvolvimento deixando a especificaçãoo inalterada.

Manutenção adaptativa: significa mudança de funções do programa. Isso é feito para adaptar as alterações no ambiente externo no qual o produto opera como novos regulamentos governamentais. Este tipo é conhecido como reforço de manutenção.

Manutenção Perfectiva: significa melhorar o desempenho ou modificar os programas para responder ao utilizador adicional ou alteração das necessidades. Envolve mudanças que o cliente pensa que irá melhorar a eficácia do produto, tais como uma funcionalidade adicional ou diminuiçãoo do tempo de resposta. Este tipo é também uma espécie de melhoria da manutenção.

Manutenção Preventiva: é o processo de evitar os sistemas obsoletos. A manutençãoo preventiva envolve o conceito de reengenharia e engenharia reversa no qual um sistema antigo com uma tecnologia antiga e reproduzido através do uso de novas tecnologias. Este manutenção impede a extinção do sistema.

Custos de Manutencao

Em 1970, a maior parte do dinheiro para os sistemas de sosftware era gasto no desenvolvimento do mesmo. A taxa do dinheiro de desenvolvmento para o dinheiro de gestao foi reservada em 1980 e vários estimativas de lugares de desenvolvimento 40% a 60 % do todo o custo do ciclo de vida do sistema (isto é, a partir do desenvolvimento até a manutençao para a evetual reforma ou substituicao). No entanto, este custo pode variar widely de um domínio de aplicaçao ao outro.

Portanto, aconselha-se a investir mais esforços na primeiras fases do ciclo de vida do software para reduzir os custos de manutençao. A tabela abaixo mostra como o defect ratio increases heavily from the analysis phase the implementation phase.

Fase	Taxa
Analise	1
Desenho	10
Implementacao	100

Tabel: [Agarwal et all]

Portanto, um esforço extra na fase de desnvolevimento pode certamente reduzir os custos de manutençao do software. Por outro lado, existem alguns factores que contribuem para esforço necessário para manter o sistema que sao:

- Tipo de aplicacao
- Sistema novetly
- Dependência da mudança do ambiente
- Características do Hardware
- Qualidade do desenho
- Qualidade do código
- Qualidade do documento
- Qualidade do teste

Actividade 2: Estimativa do Projecto de software

Estimativa do processo de software é o processo de estimar varios recursos necessarios para completar o projecto. No desenvolvimento do projecto de um sosftware fazer uma estimativa efectiva do projecto de sosftware é uma actividade muito importante.[1]
Assim sendo, estimativas de custo de software mailny encompasses the following steps:

Estimar o tamanho do Projecto: existem muitos procedimentos disponiveis para estimar o tamanho de um projecto, que sao baseados emabordagens quantitativas, como estimar a linha de código ou estimar a funcionalidade dos requisitos do projecto chamadas de funcoes de pontos.

Estimar o esforço:Pessoa por mês é base para estimar os recursos necessários por pessoa para o projecto.

Factores que afectam a Manutenção

Existema alguns factores que contribuem para o esforço necessário para manter um sistema. Esses factores incluem o seguinte:

Nova Aplicação: Como os utilizadores ganham experiência de uma nova aplicaçãoo, eles conseguem ver as possíveis melhorias e novas funcionalidades

Mobilidade do Pessoal: é sempre mais fácil para os programadores originais actualizar o código de algume. Quando a equipe se muda, torna-se difícil manter o código a menos que seja muito bem documentado.

Demasiadas Versões: pode ser difícil detectar alterações no código caso tenha havido um número de liberações

Documentação Insuficiente: se a documentação do desenho ou a documentaçãoo relativa a concepção do produto não for suficiente, então a manutenção será afectada. Uma boa prática é fazer comentários internos das variáveis descritivas.

Hardware Externo: Mudanças na plataforma de hardware, ou actualizações dos sistemas operacionais podem afectar os requisitos de manutenção.

Avaliação

Verifique a sua compreensão

1. O que é manutençãoo de software?
2. Descreva as várias categorias de manutenção que conhece
3. Explica os factores que influenciam o custo da manutenção.
4. Explica os vários tipos de manutençãoo
5. Porque que é que a munutenção é necessária?

Espector de Gestão

A gestão de desenvolvimento de software envolve 4 processos: Pessoas, produto, processo e projecto.

Pessoal

Os recursos humanso desempenham um papel importante no desenvolvimento de um produto de software. O Software Engineering Institute (SEI)

Desenvolveu um modelo para a maturidade de dos recursos humanos que chamou de People-CMM (People Capability and Maturity Model)- Modelo de capacidade e Maturidade para Recursos Humanos) em reconhecimento de que: `Toda a organizaçãoo precisa aprimorar continuamente a sua habilidade para atrair, desenvolver, motivar, organizar e reter a fora de trabalho necessária para atingr os objetivos estratégicos de seus negócios.

O People-CMM de ne as seguintes práticas-chave para o pessoal de software: formação de equipe, comunicação, ambiente de trabalho, gestão do desempenho, treinamento, compensação, análise de competência e de desenvolvimento, desenvolvimento de carreira, do grupo de trabalho, da cultura de equipe e de outros mais (3).

Produto

Antes de traçarmos um plano de projeto, devemos estabelecer os objetivos do produto e seu escopo, considerar as soluções alternativas e identi car as restrições técnicas e de gerenciamento. Sem tais informações, é impossível de nir de forma razoável (e precisa) a estimativa de custo, a avaliação efetiva dos riscos, a análise realística das tarefas do projeto ou um cronograma gerenciável do projeto que forneça a indicação signi cativa de progresso das atividades.

Como desenvolvedores, devemos nos reunir com os interessados no software para de nir os objetivos e o escopo do produto. Em muitos casos, tal atividade se inicia como parte da engenharia do sistema ou de engenharia do processo de negócio e continua como a 1a etapa da Engenharia de Requisitos do software.

Os objetivos identi cam as metas gerais do produto (do ponto de vista dos interessados) sem considerar como tais metas serão alcançadas. O escopo identi ca os principais dados, funções e comportamentos que caracterizam o produto e, mais importante, tenta mostrar as fronteiras e limitações dessas características de maneira quantitativa (3).

Processo

Um processo de software fornece a metodologia por meio da qual um plano de projeto abrangente para o desenvolvimento de software pode ser estabelecido. Poucas ativi- dades metodológicas são aplicáveis a todos os projetos de software, independentemente do seu tamanho e complexidade.

Uma quantidade de diferentes conjuntos de atividades-tarefas, pontos de controle, artefatos de software e pontos de garantia de qualidade possibilitam que as atividades metodológicas se- jam adaptadas às características do projeto de software e aos requisitos de equipe. Por m, as atividades de apoio, como Garantia de Qualidade de Software, Gerenciamento de Con guração e de Medições, sobrepõem-se ao modelo do processo. Estas são independentes de quaisquer das atividades metodológicas e ocorrem ao longo do processo. (3)

Projecto

Empregam-se projetos com planejamento e com controle por uma única e principal razão: é a única maneira de administrar a complexidade. E, mesmo assim, as equipes de software têm de se esforçar. Em um estudo de 250 grandes projetos de software entre 1998 e 2004, Caper Jones [Jon 04] constatou que "cerca de 25 obtiveram sucesso em cumprir cronograma, custos e objetivos quanto à qualidade. Em torno de 50 apresentaram atrasos em, no mínimo, 35% re- tardamentos sérios, enquanto 175 projetos tiveram uma experiência de atrasos e retardamentos sérios, ou ainda não conseguiram terminá-lo". Apesar de, atualmente, a taxa de sucesso nos projetos de software possa ter melhorado de algum modo, a taxa de falhas em projeto perma- nece mais alta do que deveria.(3)

Resumo da Unidade

Nesta unidade foram abordadas todas as questoes relacionadas com a gestao e manutencao do software. Fazer a gestao do projecto de sosftware nao é uma tarefa fácil, este processo involve o a selecao do modelo de processo, a organizacao de uma equipe de trabalho os métodos e ferramentas.

Uma das abordagens que foi detelhada foi a contagem de pontos de funcoes. Por outro lado têm-se o planeamento. Este processo envolve processo de tomada de decisoes e divisao das tarefas entres as pesoas que fazem parte da equipe de trabalho.

Um outro aspecto que também nao deve ser esquecido durante este processo todo sao os aspectos técnicos.

Avaliação da Unidade

Verifique a sua compreensão

1. O que entende por Gestao de configuracao de sosftware?
2. O que entende por gestao de risco?
3. Qual é a necessidade de manutencao de um software?
4. Enumere as categorias da gestao de software?
5. Suponha que você seja o líder de um projecto de desenvolvimento de software bastnate complexo e extenso. Faltando três meses para entrega do software, o cliente requisita mudanças que vao suscitar um esforço extra. Como você reagia?
6. Suponha que você seja o líder de um projecto de desenvolvimento de software. Um dos membros da equipe falhou a data limite para testar o código de um componente. Como você reagia?

Leituras e Outros Recursos

- Douglas Bell, Software Engineering for stiudents A Programing approach, fourth edition, 2005
- Software Engineering for students a programing approach
- Roger S. Pressman. Engenharia de Software, Uma Abordagem Profissional, Sétima Edição. 2011

Avaliação da Unidade

Verifique a sua compreensão

1. Defina Software
2. Quais são os diferentes componentes de software?
3. Quais são mitos de software
4. O que e engenharia de software
5. Por que é que precisamos de processos de desenvolvimento de software?
6. Descreva alguns processos fundamentais de atividades utilizadas no processo de software.
7. Fale um pouco sobre o ciclo de vida de desenvolvimento de um software.
8. Enumere as fases básicas no processo de desenvolvimento de software.
9. Esboçe a semântica de um modelo de desenvolvimento de software.
10. Como funciona o modelo em cascara? Explique o ser funcionamento com base no esquema de funcionamento do mesmo.
11. Quais são os diferentes mitos e verdades sobre software?
12. O que são requisitos?
13. Explica o processo para determinar os requisitos para desenvolvimento de sistemas
14. De forma clara explique o processo de estimativa de projecto.
15. O que é desemho de sistema
16. Quem é o utilizador?
17. Defina arquitectura de desenho
18. Qual é o objetivo do desenho da arquitectura
19. O que é programação estructurada?
20. O que é manutenção do software?

Referências do Curso

- Agarwal B. B., Tayal S. P. and Gupta M., (2010), "Software Engineering & Testing, an Introduction", Jones and Bartlett Publishers, ISBN: 978-1-934015-55-1
- Pressman Roger S., (2001), "Software Engineering, A Practitioner' S Approach" Fifth Edition, McGraw-Hill Higher Education, ISBN 0073655783
- Forrest Shull, Roseann Tesoriero, A study guide to acompany Shari Lawrence PfleegerSoftware Engineering: Theory and Practice
- Zhiming Liu, (2001), "Object-Otiented Software Development Using UML", The United Nations University, UNU-IIST International Institute for Software Technology, Tech Report 229.
- Sommerville Ian (2000), "Software Engineering (6th Edition)". Addison-Wesley, Boston USA
- David Gustafson (2002), "Theory and Problems of Software Engineering", Schaum's Outline Series, McGraw-Hill, 0-07-140620-4

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