Engenharia de Software (ES)

Software

- Produto projetado e construído por engenheiros de software
- Inclui
  - Programas de computador
  - Documentos físicos e virtuais
  - Dados de naturezas variadas

Criação e Utilização

- Desenvolvido por engenheiros de software
- Usado por todos no mundo industrializado

Importância

- Afeta inúmeros aspectos da vida
- Pervasivo no comércio, entretenimento, atividades diárias

Desenvolvimento

- Aplicar abordagem sistemática
- Obter resultado de alta qualidade
- Satisfazer demandas dos usuários

Produto

- Para engenheiros de software
  - Software desenvolvido
- Para usuários
  - Informação gerada pelo software

Complexidade

- Como garantir correção do software?

ES

- Framework que inclui
  - Processo
  - Métodos
  - Ferramentas

Processo de Software

Processo

- Para construir produto ou sistema
- Seguir série de passos
- Obter resultado, com qualidade requerida, conforme cronograma

Responsáveis

- Engenheiros de software e gerentes
- Adaptar processo às necessidades e segui-lo
- Clientes possuem papel no processo

Importância

- Provê
  - Estabilidade
  - Controle
  - Organização
- Atividade pode tornar-se caótica

Importante

- Processo depende do software
- Um processo apropriado para controle
- Outro completamente diferente para Web site

Produto

- Programas, documentos e dados
- Produzidos por atividade de engenharia
- Definidas pelo processo

Complexidade

- Mecanismos de avaliação de processos
- Determinar maturidade de processo
- Qualidade, conformidade com cronograma são melhores indicadores da eficácia

Processo

- Atividades aplicáveis a qualquer projeto de software
- Tarefas que possibilitam adaptá-las às peculiaridades do projeto
- Atividades guarda-chuva, garantia de qualidade, gerência de configuração

Modelo de Processo de Software

Modelo

- Engloba
  - Processo
  - Métodos
  - Ferramentas
- Diferentes modelos
  - Seqüencial Linear
  - Prototipagem

Seqüencial Linear

- Clássico, cascata
- Abordagem seqüencial, sistemática
- Começa no nível de sistema
- Progride através da análise, projeto, implementação, teste e apoio

Seqüencial Linear

- Engenharia de sistema
  - Software é parte de um sistema maior
  - Estabelecer requisitos para todos os elementos do sistema
  - Alocar subconjunto ao software

Seqüencial Linear

- Análise
  - Especificação de requisitos é intensificada
  - Entender
    - Domínio de informação para o software
    - Função
    - Desempenho
    - Interface

Seqüencial Linear

- Projeto
  - Foca quatro atributos de um programa
    - Estruturas de dados
    - Arquitetura de software
    - Representações de interface
    - Detalhes procedimentais
  - Traduz requisitos em uma representação do software

Seqüencial Linear

- Implementação
  - Projeto traduzido em uma forma que pode ser entendida pelo computador
  - Mecanicamente executada se projeto suficientemente detalhado

Seqüencial Linear

- Teste
  - Foca lógica interna do software
    - Garantir que todos comandos são testados
  - E funções externas
    - Conduzir testes para descobrir erros
    - Garantir que entradas produzem resultados conformes

Seqüencial Linear

- Apoio
  - Software sofrerá mudanças após entrega
  - Causas incluem
    - Erros descobertos
    - Acomodar mudanças no ambiente externo
    - Cliente requer melhorias
  - Aplica as fases precedentes a um software existente

Seqüencial Linear

- Problemas
  - Projetos raramente seguem fluxo seqüencial
  - Cliente tem dificuldade para explicitar requisitos
  - Cliente tem que ser paciente, programa disponível no final

Prototipagem

- Freqüentemente cliente define objetivos gerais para o software, não identifica requisitos detalhados
- Em outros casos, desenvolvedor incerto se algoritmo é eficiente, sistema apropriado, interação adequada
- Nesses casos, prototipagem é boa abordagem

Prototipagem

- Começa com obtenção de requisitos
- Cliente e desenvolvedor se encontram e definem objetivos gerais para o software
- Identificam requisitos conhecidos e destacam áreas que precisam de definições adicionais

Prototipagem

- Projeto rápido acontece
  - Foca representação de aspectos visíveis do software
- E leva à construção de um protótipo

Prototipagem

- Protótipo é avaliado pelo usuário e usado para refinar requisitos
- Iteração acontece conforme protótipo é aprimorado para satisfazer necessidades do usuário
- Ao mesmo tempo, permite ao desenvolvedor entender melhor o que precisa ser feito

Prototipagem

- Problemas
  - Cliente percebe “versão funcional” do software e exige poucos reparos para tornar o protótipo o produto final
  - Desenvolvedor compromete implementação para obter protótipo, com o tempo ele se acostuma com escolhas e esquece razões pelas quais elas são inapropriadas

RAD

RAD

- Rapid Application Development
- Modelo incremental
- Ciclo de desenvolvimento curto

RAD

- Adaptação de alta velocidade do seqüencial linear
- Desenvolvimento baseado em componente para desenvolvimento rápido

RAD

- Se requisitos bem compreendidos e escopo restrito
- Permite criar sistema funcional em período curto
- Principalmente aplicação de sistema de informação

RAD

- Restrições de tempo impostas a projeto demandam escopo escalável
- Aplicação modularizável com cada função completa em poucos meses
- Cada função tratada por time separado e então integradas

RAD

- Fases
  - Modelagem de negócio
  - Modelagem de dados
  - Modelagem de processo
  - Geração de aplicação
  - Teste

RAD

- Modelagem de negócio
  - Fluxo de informação entre funções de negócio
  - Que informação conduz processo de negócio
  - Que informação é gerada e por quem
  - Onde informação vai e quem a processa

RAD

- Modelagem de dados
  - Fluxo de informação refinado em conjunto de objetos de dados necessários para apoiar o negócio
  - Características de cada objeto identificadas
  - Relacionamentos entre objetos definidos

RAD

- Modelagem de processo
  - Objetos de dados transformados para obter fluxo de informação necessário para implementar função de negócio
  - Descrições de processamento criadas para adicionar, modificar, deletar ou recuperar objeto de dado

RAD

- Geração de aplicação
  - Reusar componentes de programas existentes
  - Criar componentes reusáveis
  - Ferramentas automatizadas usadas para facilitar construção de software

RAD

- Teste
  - Componentes já testados
  - Reduz tempo total de teste
  - Novos componentes testados
  - Interfaces completamente exercitadas

Incremental

Incremental

- Combina elementos do seqüencial linear aplicados repetidamente com iteratividade da prototipagem
- Aplica seqüências lineares periodicamente
- Cada seqüência produz incremento do software

Incremental

- Primeiro incremento é freqüentemente núcleo do produto
- Requisitos básicos são atendidos mas sem características suplementares

Incremental

- Núcleo do produto é usado pelo cliente ou passa por revisão detalhada
- Plano desenvolvido para próximo incremento

Incremental

- Plano define modificação do núcleo do produto para atender melhor necessidades do cliente
- Além da entrega de funções adicionais

Incremental

- Processo repetido seguindo liberação de cada incremento até produto completo ser produzido
- Enfatiza entrega de produto operacional com cada incremento

Incremental

- Útil se equipe indisponível para implementação completa no prazo estabelecido
- Incrementos podem ser planejados para gerir riscos técnicos

Espiral

- Combina natureza iterativa da prototipagem com aspecto controlado do seqüencial linear
- Provê potencial para desenvolvimento rápido de versões incrementais do software

Espiral

- Software desenvolvido em série de versões incrementais
- Nas primeiras iterações versão incremental pode ser modelo em papel ou protótipo
- Nas iterações posteriores versões incrementalmente mais completas são produzidas

Espiral

- Dividido em atividades de framework chamadas regiões de tarefas
- Existem de três a seis regiões

Espiral

- Regiões
  - Comunicação com cliente
  - Planejamento
  - Análise de risco
  - Engenharia
  - Construção e entrega
  - Avaliação de cliente

Espiral

- Comunicação com cliente
  - Tarefas necessárias para estabelecer comunicação efetiva entre desenvolvedor e cliente

Espiral

- Planejamento
  - Tarefas necessárias para definir recursos, linhas de tempo e outras informações relacionadas com projeto

Espiral

- Análise de risco
  - Tarefas necessárias para avaliar riscos técnicos e gerenciais

Espiral

- Engenharia
  - Tarefas necessárias para construir representações da aplicação

Espiral

- Construção e entrega
  - Tarefas necessárias para construir, testar, instalar e prover apoio ao usuário

Espiral

- Avaliação de cliente
  - Tarefas necessárias para obter retorno do cliente baseado na avaliação do software

Espiral

- Move ao redor de espiral
- Primeira volta pode resultar no desenvolvimento de especificação
- Voltas seguintes podem ser usadas para desenvolver protótipo e versões do software progressivamente mais sofisticadas

Espiral

- Cada volta através da região de planejamento resulta em ajustes no plano do projeto
- Custo e cronograma são ajustados baseado no retorno da avaliação do cliente
- Gerente do projeto ajusta número planejado de voltas na espiral

Desenvolvimento Baseado em Componentes

DBC

- Tecnologias OO provêem a infra-estrutura técnica para modelo baseado em componentes
- Paradigma OO enfatiza criação de classes que encapsulam dados e algoritmos
- Classes podem ser reusadas através de diferentes aplicações

DBC

- Modelo DBC incorpora características do espiral
- É evolutivo por natureza, demandando abordagem iterativa
- Entretanto compõe aplicações a partir de componentes de software existentes

DBC

- A atividade de engenharia começa com a identificação de classes candidatas
- Dados a serem manipulados pela aplicação e algoritmos aplicados para isso são examinados
- Dados e algoritmos correspondentes são encapsulados em uma classe

DBC

- Classes criadas em projetos passados são armazenadas em biblioteca de classes
- Biblioteca é examinada para determinar se classes candidatas identificadas já existem
- Se existe, é extraída da biblioteca e reusada; caso contrário, é criada

DBC

- Primeira iteração aplicação é composta usando classes novas e extraídas da biblioteca
- Fluxo de processo retorna ao espiral
- Entrará novamente na iteração de composição de componentes nas passagens pela atividade de engenharia

DBC

- Leva a reuso de software
- Reusabilidade leva a grande redução no tempo de ciclo de desenvolvimento e custo de projeto, além de aumento de produtividade

DBC

- Resultados são função da robustez da biblioteca de componentes
- Entretanto existe pouco questionamento quanto às vantagens significativas providas aos engenheiros de software