PROCESSOS DE SOFTWARE

Aula 04 – Introdução aos modelos de desenvolvimento de software (parte II)

Prof. Fabricio Freire

METAS DE APRENDIZAGEM

- Compreender os princípios do DSDM
- Compreender o Iconix Process
- Estudo de caso

DSDM - DYNAMIC SYSTEMS DEVELOPMENT METHODOLOGY

Definição

DSDM é uma metodologia ágil centrada em estabelecer recursos e tempo fixo para o desenvolvimento de projetos, ajustando funcionalidades para atender prazos estipulados.

Inversão Paradigmática

- Tempo e recursos: fixos e não-negociáveis
- Escopo: variável e ajustável através de priorização
- Funcionalidades: entregues por ordem de valor de negócio
- Foco: entregar valor rapidamente ao invés de completude tardia

Aplicabilidade

A metodologia não é adequada para sistemas críticos de segurança que não podem falhar, nem para contextos onde participação efetiva de stakeholders não está disponível.

DSDM - NOVE PRINCÍPIOS

Princípios que Caracterizam a Metodologia

- Participação ativa dos usuários e stakeholders
- Abordagem cooperativa e compartilhada entre equipe e negócio
- Equipes com poder de decisão e autonomia
- Entregas contínuas que fazem a diferença
- Desenvolvimento iterativo e incremental
- Feedback constante e incorporação de aprendizagem
- Todas as alterações durante desenvolvimento devem ser reversíveis
- Fixar apenas requisitos essenciais (Must Have)
- Teste em todo ciclo de vida, não apenas fase final

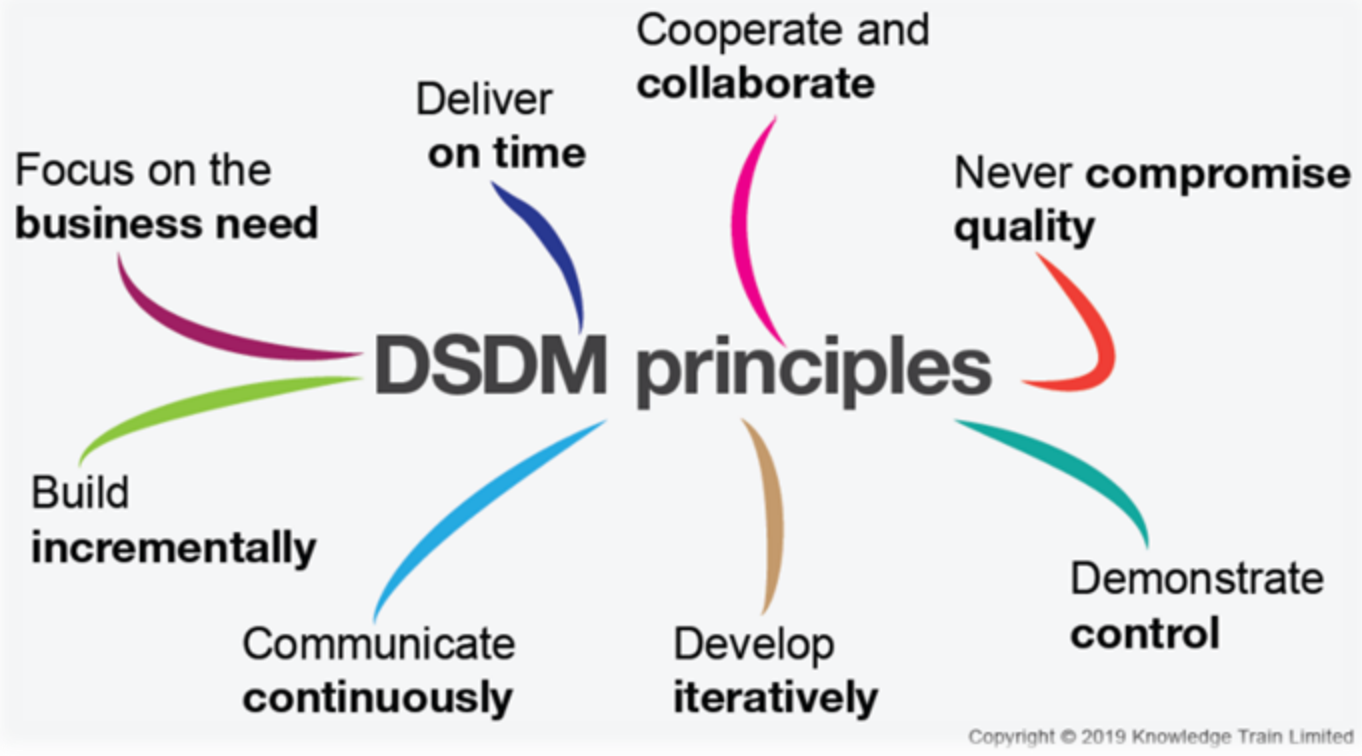
DSDM - NOVE PRINCÍPIOS

Conceito Timeboxing

- Encapsulamento do tempo reservado para desenvolvimento através da divisão do projeto em porções com orçamento próprio e data de entrega fixa.
- Funcionalidades são ajustadas para caber na timebox, não o contrário.

Conceito MoSCoW

- Técnica de priorização de requisitos durante desenvolvimento: Must have (crítico), Should have (importante), Could have (desejável), Won't have this time (excluído desta release).
- Permite negociação informada sobre escopo dentro de restrições temporais.


Cooperate and
collaborate
Deliver
on time
Focus on the
business need
Never compromise
quality
DSDM principles
Build
incrementally
Demonstrate
control
Communicate
continuously
Develop
iteratively
Copyright © 2019 Knowledge Train Limited

DSDM - FASES DO CICLO DE VIDA

Estudo da Viabilidade: Verifica se DSDM é solução adequada observando empresa como um todo, listando envolvidos, mapeando riscos, analisando técnicas e ferramentas de trabalho.

Estudo de Negócio: Observa regras de negócio, processos envolvidos, aspectos relevantes do sistema e necessidades de informação, definindo escopo do projeto.

Modelo de Iteração Funcional: Requisitos funcionais e não-funcionais são obtidos, lista de prioridades é montada, protótipo é desenvolvido e documentado.

Projeto e Construção de Iteração: Envolvidos revisam sistema e comentam resultados, dando feedback aos desenvolvedores. Sistema é efetivamente construído nesta fase.

Implementação: Sistema é colocado em funcionamento, transição do ambiente de desenvolvimento para operacional, treinamento e tarefas necessárias são executadas.

DSDM - PROTOTIPAGEM E EQUIPE

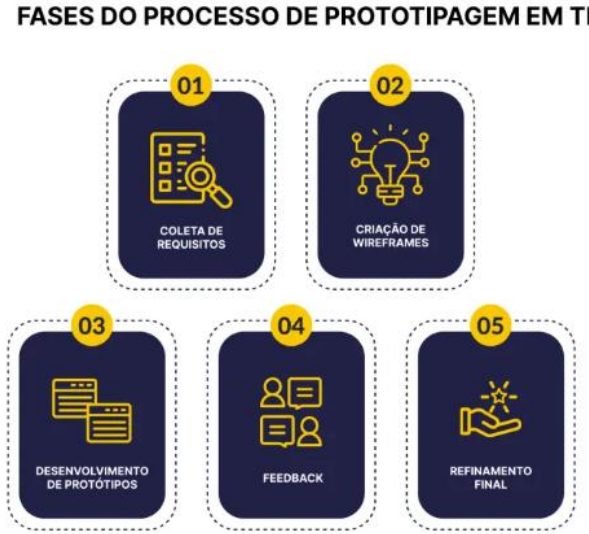
Prototipagem como Fator de Sucesso

- Permite verificação e descoberta de problemas e limitações do projeto antecipadamente.
- Considerado um dos principais fatores de sucesso da DSDM por possibilitar validação precoce de conceitos e funcionalidades com usuários reais antes de investimento substancial em implementação.

DSDM - PROTOTIPAGEM E EQUIPE

Papéis na Equipe de Projeto

- Coordenador técnico/arquiteto
- Desenvolvedor sênior
- Analista de sistemas
- Programador
- Designer de interface
- Testador
- Usuário (stakeholder ativo)
- Administrador de banco de dados
- Implantador
- Suporte técnico



Flexibilidade de Equipe

- Papéis podem ser acumulados entre envolvidos dependendo do tamanho e amplitude do projeto.
- Equipe pode variar entre duas a seis pessoas, podendo existir várias equipes pequenas em projeto maior.
- Em equipe de duas pessoas deve existir pelo menos um usuário e um desenvolvedor.

Imagem Fonte: https://www.levty.com/br/blog/voce-conhece-as-fases-do-processo-de-prototipagem-em-ti-66956cd5bdcfa1100e03fd2a

ICONIX PROCESS

Iconix Process é metodologia ágil baseada em modelagem dirigida por casos de uso que tem como objetivo estudar e comunicar comportamento do sistema sob ponto de vista do usuário final. Considerado método ágil não tão burocrático quanto RUP nem tão radical quanto XP, ocupando posição intermediária no espectro metodológico.

Características

- Utiliza subconjunto da UML: apenas três diagramas principais (Casos de Uso, Classes e Sequência) mais Robustez, ao invés dos catorze diagramas UML completos
- Minimiza paralisia da análise: ignora semântica de estereótipos UML como extend e include que retardam passagem da análise para projeto
- Possui rastreamento da análise à implementação: todos requisitos são associados a casos de uso e classes que formam eixo de sustentação do processo
- Iterativo e incremental: várias iterações ocorrem entre desenvolvimento do Modelo de Domínio e modelagem dos casos de uso
- Modelo estático é incrementalmente refinado pelo modelo dinâmico

DUAS VISÕES COMPLEMENTARES

Visão Dinâmica

- Apresenta aspectos comportamentais do software e interação do usuário com sistema. Artefatos utilizados:
- Diagrama de Caso de Uso: captura requisitos funcionais
- Diagrama de Robustez: ponte entre análise e design
- Diagrama de Sequência: especifica colaborações temporais entre objetos

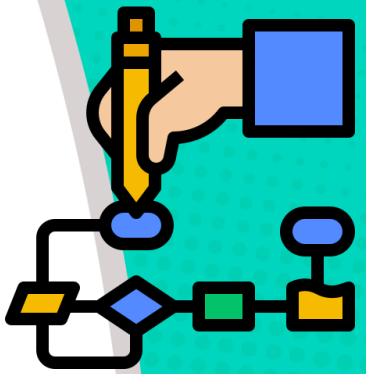
Visão Estática

- Apresenta aspectos estruturais do software, mostrando funcionamento do sistema sem dinamismo e interação do usuário. Artefatos utilizados:
- Modelo de Domínio: diagrama de classes de alto nível representando entidades conceituais
- Diagrama de Classes: design detalhado com atributos, métodos e relacionamentos de implementação

ICONIX PROCESS - DIAGRAMA DE ROBUSTEZ

Natureza Híbrida

- Diagrama de Robustez é gráfico híbrido entre Diagrama de Classes e Diagrama de Atividades da UML.
- Serve como ponte conceitual entre análise e design, ajudando a descobrir objetos necessários para realizar casos de uso.



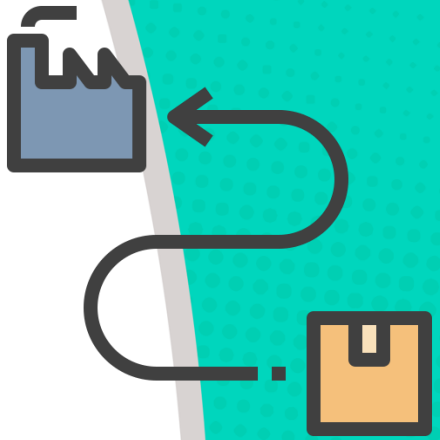
Três Estereótipos Básicos

- **Boundary (Fronteira/Interface):** objetos que fazem interface com atores externos ao sistema, representam telas, formulários, APIs
- **Entity (Entidade):** objetos que representam informação e conhecimento do domínio, tipicamente persistentes
- **Control (Controle):** objetos que coordenam e orquestram colaborações entre boundaries e entities, implementam lógica de negócio

RASTREABILIDADE DE REQUISITOS

Cadeia de Rastreabilidade

- Modelo de Casos de Uso: requisitos funcionais expressos como interações com usuários
- Descrição dos Casos de Uso: detalhamento textual de cenários e fluxos
- Diagrama de Robustez: objetos boundary, entity e control necessários para realizar caso de uso
- Diagrama de Sequência: colaborações temporalmente ordenadas entre objetos
- Diagrama de Classes: estrutura de implementação com métodos alocados
- Código: implementação efetiva rastreável até requisito original



ESTRUTURA DO PROCESSO

Análise de Requisitos

- Identificação de requisitos, casos de uso, objetos de domínio e desenvolvimento de protótipos de interface.
- Utiliza levantamento de documentos de regras de negócio, entrevistas e questionários aos usuários.

Análise e Desenho Preliminar

Descrição detalhada dos casos de uso usando cenários e desenvolvimento do Diagrama de Classes em nível conceitual representando modelo de domínio.

Desenho Detalhado

Especificação do comportamento do Diagrama de Classes o mais detalhado possível, incluindo assinaturas de métodos, tipos de atributos e diagramas de sequência mostrando colaborações.

Implementação

- Especificação de Diagrama de Componentes e instalação, escrita do código, realização de testes unitários, testes de integração e testes de aceitação.
- Cada fase possui revisões formais garantindo qualidade.

QUESTÕES FUNDAMENTAIS

Base Epistemológica do Processo

Iconix tem como base responder questões fundamentais sobre software através de técnicas específicas:

Quem são os usuários do sistema e o que estão tentando fazer? Utilizar casos de uso para capturar interações significativas.

O que são, no mundo real (domínio de problema), os objetos e associações entre eles? Utilizar diagrama de classe de alto nível (modelo de domínio).

QUESTÕES FUNDAMENTAIS

Que objetos são necessários para cada caso de uso? Utilizar análise de robustez com estereótipos boundary, entity e control.

Como objetos estão colaborando e interagindo dentro de cada caso de uso? Utilizar diagrama de sequência e de colaboração.

Como será manipulado em tempo real aspectos de controle? Utilizar diagrama de estado quando objetos têm comportamento dependente de histórico.

Como realmente será construído o sistema em nível prático? Utilizar diagrama de classe de baixo nível com detalhes de implementação.

ESTUDO DE CASO

Contexto: Automação de Processos com IA

Uma empresa de médio porte precisa desenvolver sistema de automação de processos utilizando técnicas de Inteligência Artificial para classificação automática de documentos, extração de informações e roteamento inteligente.

OSistema deve processar milhares de documentos diários em múltiplos formatos, aprender continuamente com feedback de usuários e integrar-se com sistemas legados existentes.

ESTUDO DE CASO

Requisitos Principais

- Upload e processamento de documentos em PDF, imagens e texto
- Classificação automática por tipo de documento usando machine learning
- Extração de entidades nomeadas e informações estruturadas
- Roteamento automático para departamentos apropriados
- Interface para validação e correção de classificações
- Retreinamento contínuo do modelo com feedback
- Integração via API com ERP e CRM existentes
- Dashboard de métricas e acurácia do sistema

Restrições e Desafios

- Prazo de seis meses para primeira versão operacional
- Equipe de cinco desenvolvedores com conhecimento variado em IA
- Requisitos de acurácia evoluirão conforme uso real
- Necessidade de demonstrações frequentes para stakeholders

ESTUDO DE CASO - PROCESSO UNIFICADO

Fase de Concepção

UP iniciaria com análise de viabilidade técnica dos componentes de IA, definição de casos de uso arquiteturalmente significativos como “Classificar Documento” e “Treinar Modelo”, e identificação de riscos técnicos principais relacionados a acurácia de modelos e integração com sistemas legados. Documento de visão estabeleceria objetivos de negócio e arquitetura conceitual do sistema.

Fase de Elaboração

Desenvolvimento de protótipo arquitetural validando viabilidade de frameworks de machine learning selecionados, definição de arquitetura em camadas separando lógica de IA de interface e integração, especificação detalhada de casos de uso prioritários, e criação de modelo de classes representando pipeline de processamento de documentos. Riscos de acurácia e performance seriam confrontados através de provas de conceito.

ESTUDO DE CASO - UP

Fase de Construção

- Implementação iterativa de funcionalidades através de múltiplas iterações de duas a quatro semanas, cada uma produzindo incremento executável.
- Primeiras iterações focariam em pipeline básico de classificação, iterações médias adicionariam extração de informações, iterações finais implementariam retreinamento e integrações.

Fase de Transição

Testes beta com usuários reais, ajustes baseados em feedback, migração de dados históricos para treinamento inicial do modelo, treinamento de usuários finais e implantação gradual por departamentos.

ESTUDO DE CASO - XP

Planejamento com Stories

Cliente escreveria user stories em cartões: “Como analista, quero fazer upload de PDF para que seja automaticamente classificado”, “Como supervisor, quero corrigir classificações incorretas para melhorar modelo”. Equipe estimaria esforço de cada story, cliente priorizaria por valor de negócio, e planning game determinaria escopo de iterações de uma a duas semanas.

Práticas Técnicas Aplicadas

Programação em pares seria crítica dado conhecimento variado em IA da equipe, permitindo transferência de expertise. TDD seria aplicado para lógica de negócio e pipeline de processamento, embora modelos de machine learning requeiram abordagem de testes adaptada. Integração contínua executaria testes automatizados e retreinaria modelos com dados de teste após cada commit. Design simples evitaria complexidade especulativa em arquitetura de IA.

Iterações Curtas e Feedback

Releases semanais demonstrariam progresso incremental para stakeholders. Cliente on-site seria essencial para definir critérios de acurácia aceitável e validar classificações. Refatoração contínua manteria código de pipeline limpo à medida que arquitetura de modelos evolui baseada em experimentação.

Desafios Específicos

Natureza experimental de IA cria tensão com design simples e YAGNI. Equipe precisaria balancear simplicidade com experimentação de diferentes arquiteturas de modelo até encontrar abordagem efetiva.

ESTUDO DE CASO - XP

Planejamento com Stories

- Cliente escreveria user stories em cartões: “Como analista, quero fazer upload de PDF para que seja automaticamente classificado”, “Como supervisor, quero corrigir classificações incorretas para melhorar modelo”.
- Equipe estimaria esforço de cada story, cliente priorizaria por valor de negócio, e planning game determinaria escopo de iterações de uma a duas semanas.

Práticas Técnicas Aplicadas

- Programação em pares seria crítica dado conhecimento variado em IA da equipe, permitindo transferência de expertise.
- TDD seria aplicado para lógica de negócio e pipeline de processamento, embora modelos de machine learning requeiram abordagem de testes adaptada.
- Integração contínua executaria testes automatizados e retreinaria modelos com dados de teste após cada commit.
- Design simples evitaria complexidade especulativa em arquitetura de IA.

ESTUDO DE CASO - FDD

Fase de Concepção e Planejamento

- Desenvolvimento de modelo abrangente do domínio de processamento de documentos com especialistas de negócio, identificando entidades como Documento, Classificação, Modelo, ExtractedEntity.
- Construção de lista de features usando template: “Classificar tipo de documento fiscal”, “Extrair número de nota fiscal”, “Rotear documento para departamento contábil”.
- Features são decompostas por áreas de negócio e atividades.

Planejamento por Features

- Features priorizadas por valor de negócio e dependências técnicas.
- Chief Programmer recebe conjuntos de features e aloca classes aos desenvolvedores.
- Features relacionadas a IA teriam estimativas maiores devido a incerteza e experimentação necessária.
- Planejamento consideraria features de infraestrutura de IA como dependências de features de negócio.

ESTUDO DE CASO - FDD

Detalhar e Construir por Features

- Para cada feature, equipe desenvolve diagramas de sequência mostrando colaborações, refina modelo de objetos adicionando métodos necessários, e implementa seguindo projeto técnico.
- Feature “Classificar documento” seria detalhada mostrando fluxo desde upload até invocação de modelo de IA e persistência de resultado.
- Após implementação, feature passa por testes de unidade, inspeção de código e integração ao build.

Vantagens para Projeto de IA

- Abordagem feature-driven permite rastreamento granular de progresso mesmo com incerteza de IA.
- Features podem ser demonstradas isoladamente, facilitando validação de acurácia incrementalmente.

ESTUDO DE CASO - DSDM

Timeboxing Rígido

- DSDM fixaria prazo de seis meses como não-negociável, dividindo em timeboxes de duas a quatro semanas.
- Funcionalidades seriam ajustadas para caber em timeboxes ao invés de estender prazos quando complexidade de IA se mostra maior que antecipado.

Priorização MoSCoW

- **Must Have**: classificação básica de três tipos principais de documentos, interface de upload, integração com um sistema legado crítico.
- **Should Have**: extração de informações estruturadas, dashboard de métricas, classificação de tipos adicionais.
- **Could Have**: retreinamento automático avançado, suporte a formatos exóticos.
- **Won't Have**: análise de sentimento, OCR para documentos manuscritos.

ESTUDO DE CASO - DSDM

Iteração Funcional com Prototipagem

- Protótipos de interface e classificação seriam desenvolvidos rapidamente para validação com usuários.
- Feedback sobre acurácia aceitável informaria ajustes no escopo.
- Se modelo de IA não atingir acurácia Must Have em timebox, funcionalidade seria simplificada ou convertida em Should Have.

Vantagens e Desafios

- Abordagem pragmática de DSDM se adequa bem a restrições de prazo empresariais.
- Contudo, natureza experimental de IA cria risco de subestimar complexidade em timeboxes, requerendo repriorização frequente.

ESTUDO DE CASO - ICONIX PROCESS

Análise de Requisitos com Casos de Uso

- Casos de uso capturariam interações principais: “Processar Documento”, “Validar Classificação”, “Consultar Histórico”.
- Protótipos de interface seriam desenvolvidos mostrando fluxo de upload e validação.
- Modelo de domínio identificaria classes conceituais: Documento, Classificador, ModeloIA, Resultado, Validação.

Análise de Robustez

- Para caso de uso “Processar Documento”, análise de robustez identificaria: Boundaries (TelaUpload, APIIntegração), Entities (Documento, Classificação, Modelo), Controls (ProcessadorDocumento, ClassificadorIA, GerenciadorTreinamento).
- Diagrama revelaria objetos necessários e refinaria especificação do caso de uso.

ESTUDO DE CASO - ICONIX PROCESS

Diagramas de Sequência

- Sequência para “Processar Documento” mostraria: TelaUpload recebe arquivo, ProcessadorDocumento extrai features, ClassificadorIA invoca modelo pré-treinado, Classificação persiste resultado, sistema notifica APIIntegração para roteamento.
- Métodos seriam alocados a classes durante elaboração da sequência.

Rastreabilidade de IA

- Rastreabilidade de Iconix permitiria verificar se cada requisito de acurácia tem implementação correspondente no pipeline de classificação e se cada classe do modelo de IA tem justificativa em requisitos funcionais.
- Crítico para validação sistemática de sistema de IA.

DOCUMENTAÇÃO E FORMALISMO

Processo Unificado

- Documentação mais completa e formal: documento de visão, especificação de requisitos, documento de arquitetura, modelos UML extensivos em múltiplas perspectivas.
- Apropriado quando rastreabilidade rigorosa e documentação detalhada são necessárias para conformidade ou manutenção de longo prazo.

Extreme Programming

- Documentação mínima e viva: user stories em cartões, testes automatizados como especificação executável, código limpo e auto-explicativo como documentação primária.
- Apropriado quando equipe é experiente, cliente está disponível e mudanças frequentes tornam documentação tradicional rapidamente obsoleta.

DOCUMENTAÇÃO E FORMALISMO

FDD

Documentação moderada e focada: lista de features, diagramas de sequência por feature, modelo de objetos evolutivo. Apropriado quando rastreabilidade granular por funcionalidade é desejada sem overhead de documentação completa do UP.

DSDM

Documentação pragmática e suficiente: apenas documentação que agrega valor e pode ser produzida dentro de timeboxes. Apropriado quando pressões de prazo requerem foco em entregáveis executáveis ao invés de documentação formal.

Iconix

Documentação intermediária e visual: casos de uso, diagramas de robustez, sequência e classes. Apropriado quando comunicação visual através de modelos é mais efetiva que especificações textuais extensivas.

COMPARAÇÃO - GESTÃO DE MUDANÇAS

Processo Unificado

- Mudanças são gerenciadas formalmente através de controle de mudanças com avaliação de impacto arquitetural.
- Baselines estabelecidos em milestones de fase fornecem pontos de estabilidade.
- Arquitetura robusta estabelecida em Elaboração facilita acomodação de mudanças sem reestruturação fundamental.

Extreme Programming

- Mudanças são abraçadas como característica inevitável. Design simples e refatoração contínua mantêm código maleável.
- Testes abrangentes fornecem confiança para fazer mudanças ousadas. Cliente reprioriza stories a cada iteração baseado em valor emergente.

COMPARAÇÃO - GESTÃO DE MUDANÇAS

FDD

Mudanças são absorvidas através de adição, modificação ou repriorização de features na lista. Modelo de objetos é refinado iterativamente acomodando novos requisitos. Features concluídas fornecem baseline incremental de funcionalidade validada.

DSDM

Mudanças são negociadas dentro de timeboxes fixas através de repriorização MoSCoW. Requisitos Must Have permanecem fixos, mas Should Have e Could Have podem ser ajustados. Prototipagem iterativa permite validação e refinamento antes de implementação final.

Iconix

Mudanças são rastreadas através de impacto em casos de uso, objetos de robustez, sequências e classes. Rastreabilidade bidirecional facilita análise de impacto e garante que mudanças são propagadas consistentemente através de artefatos.

LIÇÕES DO ESTUDO DE CASO

Não Existe Metodologia Universal

- Cada metodologia oferece trade-offs específicos entre previsibilidade e adaptabilidade, documentação e agilidade, disciplina e flexibilidade.
- Escolha depende de contexto organizacional, características da equipe, natureza do domínio de problema e restrições de projeto.

Metodologias Podem Ser Combinadas

- UP pode incorporar práticas de XP como programação em pares e TDD.
- FDD pode usar timeboxing de DSDM.
- Iconix pode ser processo dentro de fase de UP.
- Pragmatismo supera purismo metodológico.

LIÇÕES DO ESTUDO DE CASO

IA Adiciona Complexidade Única

- Desenvolvimento de sistemas com IA requer balanço entre experimentação necessária para encontrar arquitetura de modelo efetiva e disciplina de engenharia para produzir sistema robusto.
- Metodologias tradicionais devem ser adaptadas para acomodar natureza experimental e não-determinística de machine learning.

Sucesso Depende de Execução

- Metodologia perfeita executada mal falha.
- Metodologia adequada executada com disciplina, comprometimento e adaptação contínua sucede.
- Retrospectivas regulares e melhoria de processo são mais importantes que escolha inicial de metodologia.

BONS ESTUDOS