Manual de Boas Práticas para Desenvolvimento de Software

Table of contents

Apresentação

Olá, tudo bem?

Seja bem vindo ao **Manual de Boas Práticas para Desenvolvimento de Software em Institutos Federais de Educação**.

Utilize o menu ao lado para navegar entre as seções. Caso tenha dúvidas entre em contato comigo por aqui.

Boa leitura!

Relação de Boas Práticas apresentadas

Outras informações

Sobre o Manual

Este manual foi elaborado como produto de uma pesquisa de mestrado profissional e tem como objetivo auxiliar as equipes de tecnologia da informação e comunicação dos Institutos Federais de Educação a praticarem boas práticas de desenvolvimento de projetos de software na expectativa de entregá-los no prazo previsto e aumentar a satisfação dos clientes, com base em revisões de literatura e pesquisa de campo realizadas.

Nos capítulos a seguir será apresentado ao leitor o contexto da pesquisa (os IFs), as principais dificuldades encontradas e as principais técnicas propostas pelos mais renomados autores, a fim de combater os fatores críticos, entregar os produtos no prazo, melhorar a qualidade e cumprir objetivos institucionais.

O manual foi escrito em linguagem clara para que todas as pessoas possam utilizá-lo e sua estrutura foi embasada em livros sobre Padrões de Projetos, para facilitar a leitura e o

entendimento.

O objetivo é servir de pontapé inicial para que as equipes de TIC debatam e busquem cada vez mais a melhoria contínua do produto de software no Brasil.

Sobre o autor

Me chamo Lucas Bacciotti Moreira, sou formado em Sistemas de Informação pela UEMG, possuo especialização em Engenharia de Sistemas (ESAB) e estou concluindo meu Mestrado em Ciências da Computação no CIn - Centro de Informática da UFPE Universidade Federal de Pernambuco.

Há mais de dez anos venho trabalhando e estudando na área de Informática e Tecnologia. Tenho experiência em linguagens para web, banco de dados, infraestruturas de servidores web, entre outros.

Fiz este manual para ajudar as equipes de software de instituições de ensino a melhorarem a qualidade dos seus produtos, entregando-os no prazo estipulado e com os objetivos cumpridos.

Abaixo deixo meus contatos:
Email: lbm5@cin.ufpe.br
Twitter: @baciotti
GitHub: http://github.com/bacciotti
Cursos: Baciotti Cursos
Podcast: BaciottiCast

Como utilizar este Manual

Para utilizar este documento utilize o menu ao lado para navegar entre as seções (Fig. 1).

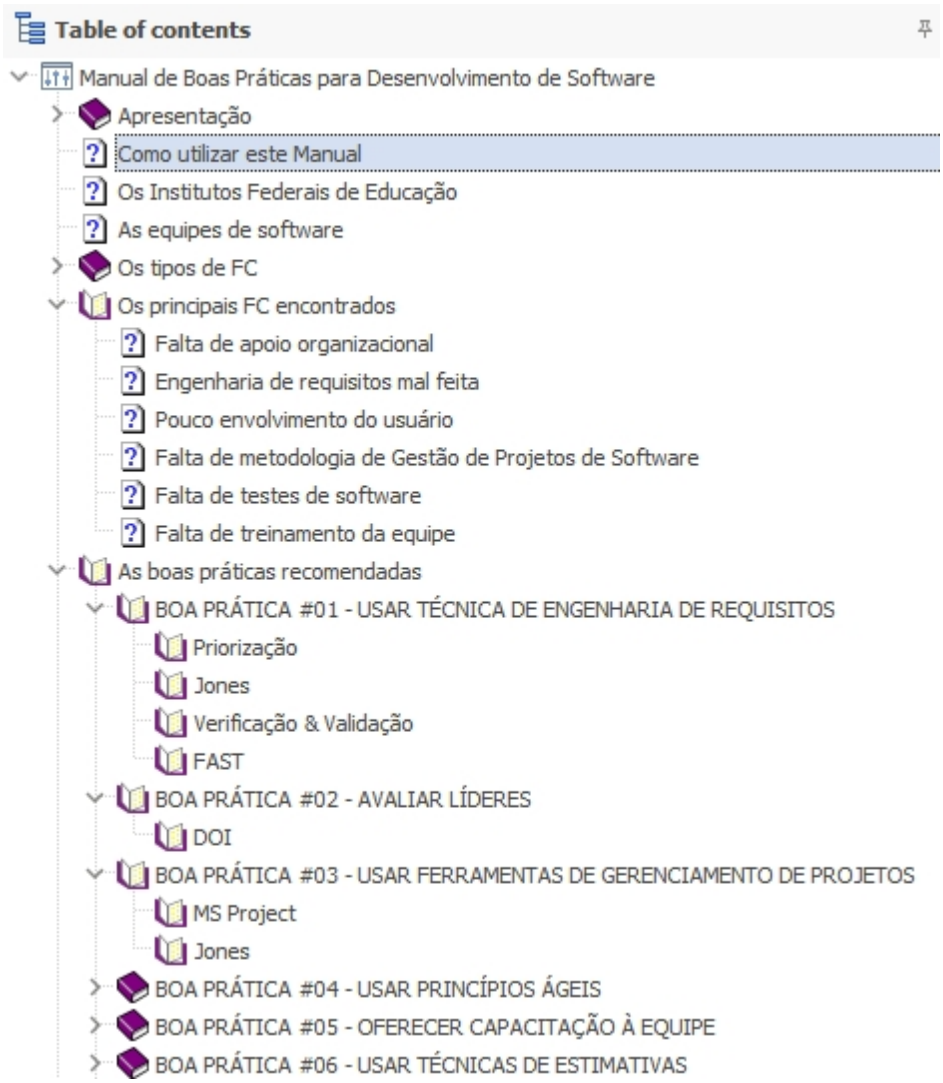


Fig. 1 - Menu de navegação lateral

Além disso, há duas setas no canto superior direito para navegar entre a seção anterior ou posterior (Fig. 2).

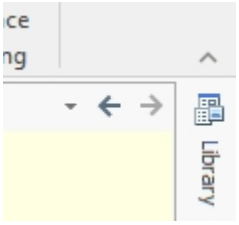


Fig. 2 - Setas de navegação

Você pode fazer o download deste conteúdo na seção Download. Fique à vontade para espalhar para seus colegas!

Este Manual utiliza um formato bastante utilizado em obras que tratam de **Padrões de Projetos** a fim de dividir as idéias e facilitar o entendimento do leitor. Desta forma, cada FC - Fator Crítico aqui mencionado possui:

- **Nome do FC:** como é popularmente conhecido.
- **Descrição:** resumo breve sobre o FC.
- **Também conhecido como**: outros nomes do FC.
- **Consequências:** desvantagens trazidas pelo FC.

E cada Boa Prática possui:

- **Nome da boa prática:** o nome da boa prática detalhada na seção.
- **Objetivo:** detalhes sobre a Boa Prática, conceituação e possíveis aplicações.
- **Também conhecida como:** outras denominações as quais a Boa Prática é referenciada.
- **FC relacionado(s):** os FC relacionados à Boa Prática.
- **Consequências:** vantagens e desvantagens do uso e nível de dificuldade.
- **Boas Práticas relacionadas:** informa quais outras boas práticas se relacionam com esta.
- **Dica:** alguma informação que pode auxiliar o leitor a pesquisar mais sobre o assunto tratado na seção.

Além das informações acima, cada Boa Prática possui algumas subseções com um maior detalhamento de algumas técnicas propostas, baseadas em bibliografia consultada (Fig. 3).

Fig. 3 - Boas Práticas e suas subseções

Os Institutos Federais de Educação

Os Institutos Federais (IFs) surgiram em 2008 por intermédio de uma lei federal, com o objetivo

de ofertar educação profissional, tecnológica, pública, gratuita e de qualidade nas diferentes áreas de ensino com base na conjugação de conhecimentos técnicos e tecnológicos com as suas práticas pedagógicas.

Por possuírem autonomia administrativa e financeira, ou seja, serem responsáveis pela gestão de seu pessoal, alocação de recursos, estabelecimento e cumprimento de metas, entre outras atividades correlatas, cada IF opta por gerenciar à sua maneira seus recursos de Tecnologia da Informação e Comunicação (TIC), muitos dos casos variando entre Diretorias de TIC ou Coordenações de TIC.

Em alguns casos, embora o IF possua um processo definido de desenvolvimento de software (52% dos casos), alguns produtos são entregues após o prazo estipulado (76,9%), o que pode ocasionar baixa satisfação do cliente, perda de qualidade de software e prejuízo ao cumprimento dos objetivos institucionais.

Após pesquisa realizada, levantou-se uma série de fatores que podem influenciar (positiva ou negativamente) a entrega dos projetos de software. Estes fatores serão descritos nos próximos capítulos. Além disso, uma série de técnicas e propostas, chamadas aqui de Boas Práticas, são informadas ao leitor.

A Fig. 1 a seguir exibe a Rede Federal de Ensino Técnico e Tecnológico (MEC).

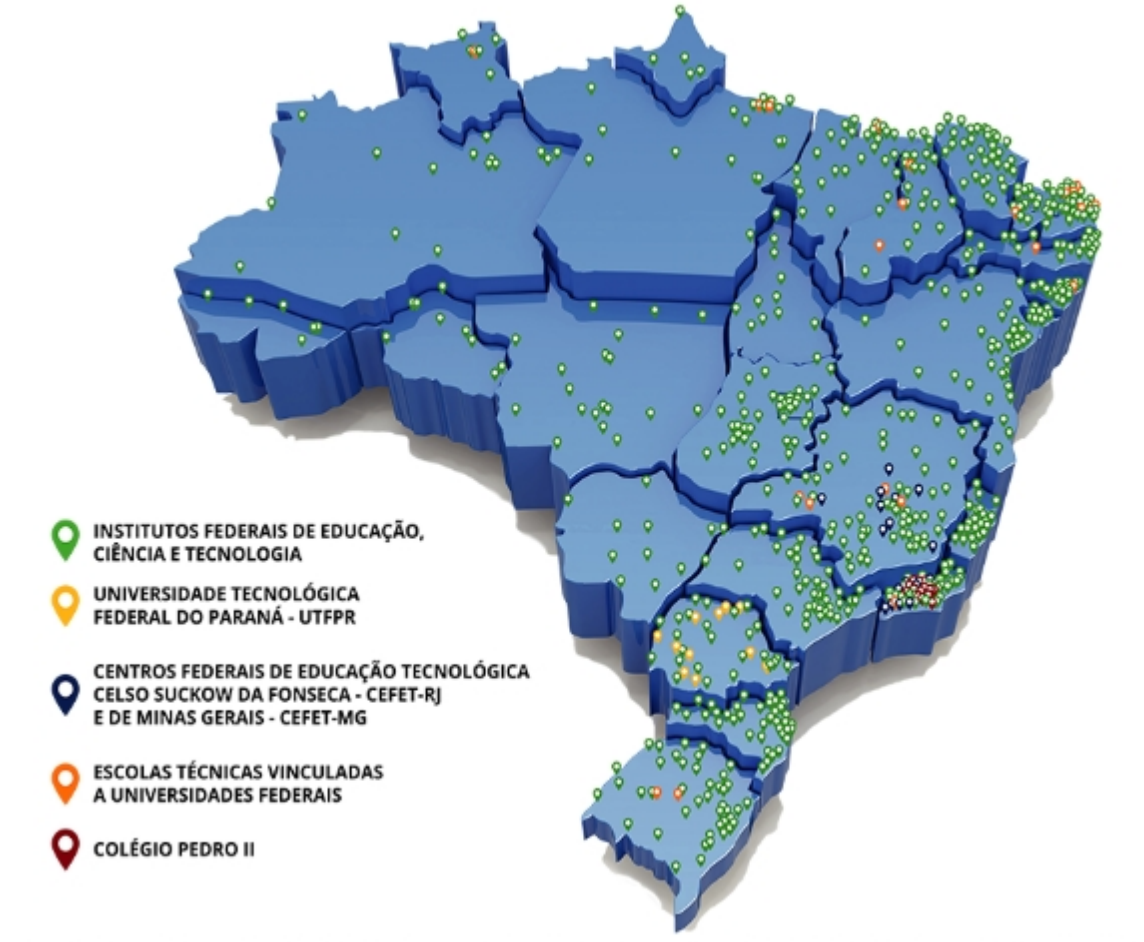


Fig. 1 - Lista de Campi da Rede Federal

As equipes de software

As equipes de Tecnologia da Informação e Comunicação (TIC) dos IFs são geralmente compostas por analistas de TI, técnicos de TI e estagiários, sendo gerenciadas, na maioria dos casos, pelo Diretor de TI, o qual possui o cargo de Professor.

Segundo dados coletados, apenas 52% das coordenações de sistemas dos IFs possuem um processo definido de desenvolvimento de software. Destes, 76,9% dos respondentes acreditam que os projetos de software de sua instituição não são entregues no prazo previsto e 88,5% dos respondentes acreditam que a não entrega no prazo prejudica a satisfação do cliente.

Desta forma, entende-se que os próximos capítulos podem auxiliar estas equipes a:

- Detectarem a ocorrência de certos FC - Fatores Críticos ao desenvolvimento;
- Buscarem padronização de processos;
- Começarem a propor meios para coletar de forma mais precisa os requisitos;
- Estimarem mais precisamente o tempo e custo dos projetos;
- Entregarem os produtos de software nos prazos previstos;
- Aumentarem a satisfação do cliente;
- Cumprirem objetivos institucionais.

Os tipos de FC

Existem diversos tipos de Fatores Críticos encontrados em Fábricas de Software.

Neste manual dividiremos nos seguintes tipos para facilitar o entendimento e aplicação.

TIPO P - FC RELACIONADO ÀS PESSOAS

Os FC Tipo **P** são aqueles relacionados às interações, relações e emoções humanas.

Aspectos que envolvem comunicação, resolução de conflitos, entendimento de necessidades por ambas as partes, motivação e treinamento, por exemplo, são pontos importantes dos FC Tipo P.

Para citar exemplos concretos, temos:

- Comunicação interna efetiva

- Comunicação externa efetiva
- Pouca atenção aos riscos
- Pouca atenção da alta gestão aos problemas do setor de TIC
- Treinamento
- Envolvimento do usuário/cliente

TIPO M - FC RELACIONADO AOS MÉTODOS

Este tipo envolve todos os FC relacionados à forma, ao passo a passo, ao protocolo de como é feita determinada tarefa. “Quais passos devem ser seguidos para alcançar este objetivo?” e “Qual a melhor maneira de realizar esta ação?” são perguntas que norteiam os principais FC do Tipo **M**.

Alguns exemplos:

- Metodologia para estimar e medir os passos do projeto
- Metodologia para coletar requisitos dos clientes
- Metodologia usada para gestão dos projetos

Os principais FC encontrados

Após elencamos os tipos de FC ao seu projeto de software, vamos agora conhecer alguns dos principais mais a fundo.

Falta de apoio organizacional

Nome do FC

Falta de apoio organizacional.

Tipo

P

Descrição

É um fator relacionado às pessoas (NASIR; SAHIBUDDIN, 2011) que refere-se à aprovação e

autorização de projetos por parte da alta gestão da organização, bem como a sua ciência de que o projeto possui alta prioridade, além da indicação de algum stakeholder como parte interessada e ativa nos projetos (AHIMBISIBWE; CAVANA, 2016).

A interferência deste FC num projeto tem relação ao quanto a alta gestão da organização vai apoiar a execução do projeto, seja fornecendo recursos ou autorizando atividades essenciais, por exemplo.

Algumas evidências deste problema incluem:

- Não disponibilização de recursos financeiros para realizar determinadas etapas do projeto;
- Não contratação de pessoal qualificado;
- Não disponibilização de recursos de infra-estrutura para a equipe: salas de reuniões, cabeamento de link para internet entre outros.

Ponto importante também é a Boa interação entre os membros da equipe, que pode trazer benefícios ao desenvolvimento do projeto pois facilita a comunicação e resolução de problemas encontrados

Também conhecido como

Falta de apoio da alta gestão.

Consequências

- Falta de pessoal qualificado para realização de determinadas tarefas;
- Atraso na entrega de incrementos de software;
- Mal uso de recursos disponíveis, gerando desperdício.

Engenharia de requisitos mal feita

Nome do FC

Engenharia de requisitos mal feita.

Tipo

M e P

Descrição

Uma efetiva análise de requisitos depende da qualidade dos *stakeholders* envolvidos no projeto. Jones (2004) destaca que a Engenharia de requisitos é um FC pois possui atividades-chave, como “lidar efetivamente com mudança de requisitos” e “alocar tempo suficiente para análise de requisitos”.

Hussain, Mkpojiogu e Kamal (2016) afirmam que a engenharia de requisitos é central para todo o projeto de desenvolvimento de software.

Cerpa e Verner (2009) destacam que a maioria dos projetos falham devido a requisitos inadequados. Já Hashim, Abbas e Hashim (2013) descrevem que requisitos irreais têm grande impacto em projetos de software.

Sem um bom levantamento de requisitos junto ao cliente, todo o esforço da equipe pode ser desconsiderado caso o cliente não fique satisfeito com um módulo ou funcionalidade solicitada, por exemplo, levando ao retrabalho.

Para se alcançar os objetivos do projeto é necessário entender muito bem o que o cliente necessita. Para tal, faz-se necessária uma coleta de requisitos detalhada, com uma análise posterior rigorosa. Uma coleta de requisitos mal feita pode levar a atrasos, desperdício de recursos e insatisfação do cliente.

Também conhecido como

Falta de análise requisitos, falta de coleta de requisitos.

Consequências

- O analista não entende o que o cliente quis dizer em relação à uma funcionalidade;
- O cliente não soube expressar bem o que deseja;
- O cliente obteve uma funcionalidade que não pediu ou que não esperava;
- Recursos foram desperdiçados em uma funcionalidade que não agregou valor ao cliente.

Pouco envolvimento do usuário

Nome do FC

Pouco envolvimento do usuário.

Tipo

M e P

Descrição

De acordo com o Manifesto Ágil, o cliente deve estar presente em todo o ciclo de vida do projeto para ajudar na condução dos requisitos e diminuir o impacto de mudanças de direcionamento.

Também conhecido como

Usuário distante do projeto, falta de comunicação intensa com o cliente.

Consequências

- Falta de feedback dos clientes em relação às funcionalidades entregues;
- Desinteresse do cliente;
- Clientes ausentes nas reuniões principais do projeto;
- Pouca comunicação entre equipe e clientes.

Falta de metodologia de Gestão de Projetos de Software

Nome do FC

Falta de metodologia de Gestão de Projetos de Software

Tipo

M

Descrição

Uma Gestão de Projetos de Software de alta qualidade conduz a um bom produto final, enquanto uma Gestão mal conduzida pode levar à falha do projeto (BILGAIYAN; MISHRA; DAS, 2016).

De acordo com Suliman e Kadoda (2017), a indústria de software contribui para as falhas nos projetos usando abordagens de gestão desestruturadas e pouco documentadas. Govindarajan (2015) reitera que a GPS é um dos fatores mais significativos a serem considerados para se obter os melhores resultados em um projeto. O planejamento (uma das fases do ciclo de vida do projeto de software e parte da Gestão de Projetos de Software) quando mal executado pode acarretar ineficiências no projeto e, então, conduzir a alterações na previsão de entrega, além de um orçamento inflacionado (AHIMBISIBWE; CAVANA, 2016).

Ayyubi, Ahmad e Faiz (2007) concordam que um mal planejamento, além de outros fatores, é a causa da perda de controle dos projetos de software: “Planejamento é a área chave a qual determina o sucesso ou o fracasso. Geralmente não se dá muita atenção à fase de planejamento, embora seja uma fase essencial.” (AYYUBI; AHMAD; FAIZ, 2007).

Logo, a não utilização de uma metodologia formal para gerir os produtos de software é um dos principais problemas que influenciam na entrega no prazo previsto. A falta de uma metodologia é aparente quando:

- Não se possui um manual, um protocolo ou um passo-a-passo a ser seguido na organização para se desenvolver e se entregar produtos de software;
- A equipe não possui conhecimento sobre métodos de desenvolvimento, como Método Ágil ou Cascata;
- Há dificuldade na implantação de alguma metodologia, seja por causa da cultura

organizacional ou seja por fatores externos à equipe.

Também conhecido como

Ausência de gerenciamento de projetos de software.

Consequências

- Projetos de software desenvolvidos apenas para resolver questões pontuais (“apagar incêndios”);
- Necessidade de refazer o mesmo projeto várias vezes;
- Desperdício de recursos.

Falta de testes de software

Nome do FC

Falta de testes de software.

Tipo

M

Descrição

Os testes de software são o meio de assegurar que o que foi desenvolvido funciona de acordo com o solicitado. Os testes de regressão, por exemplo, são testes que auxiliam na análise dos impactos que novos trechos de código adicionados ao software podem causar no funcionamento do sistema como um todo (GUPTA; CHAUHAN; DUTTA, 2013).0

Jones (2004) sugere planos de testes formais e a construção de casos de testes para prevenir defeitos no projeto. Segundo o autor, a alocação de tempo suficiente para os testes e reparos de defeitos é um dos principais FC. Como exemplo, temos os testes unitários, que são realizados logo no início do desenvolvimento. Sem eles, a chance de haver um erro de código em fases posteriores é muito grande, o que pode atrasar as atividades.

Também conhecido como

Testes de software inadequados.

Consequências

- Erros de código em fase de produção (quando o software já está sendo usado pelos clientes finais);
- Refatoramento de partes do software devido a *bugs*;
- Perda de credibilidade da equipe perante o cliente.

Falta de treinamento da equipe

Nome do FC

Falta de treinamento da equipe

Tipo

P

Descrição

Os treinamentos da equipe de desenvolvimento são úteis quando moldados aos requisitos de cada grupo de usuários. Além disso, os treinamentos não devem ocorrer em um evento único no tempo, devem ocorrer de forma regular a fim de garantir que os colaboradores se mantenham atualizados sobre as mudanças no processo (JING; QIU, 2007).

Um exemplo importante é o treinamento da equipe para a utilização de novas tecnologias de programação, o que pode dar velocidade à codificação evitando o atraso do projeto.

Também conhecido como

Falta de capacitação à equipe.

Consequências

- Dificuldade da equipe em conhecer novas tecnologias;
- Demora na execução de tarefas específicas que carecem novos conhecimentos.

As boas práticas recomendadas

Em consonância com o que foi descrito até então, tem-se as seguintes boas práticas como sugestão.

Quanto maior o número de boas práticas sendo seguidas, melhor será o resultado final.

BOA PRÁTICA #01 - USAR TÉCNICA DE ENGENHARIA DE REQUISITOS

Nome da boa prática

Usar técnica de engenharia de requisitos

Objetivo

Existem técnicas que facilitam a coleta de requisitos e que podem auxiliar neste processo, por exemplo a FAST - *Facilitated Application Specification Technique* (Técnica de Especificação de Aplicação Facilitada).

A utilização de técnicas já existentes não é obrigatória, mas é um bom começo para as equipes que buscam uma forma de padronizar a engenharia de requisitos da organização.

É necessário buscar uma forma de padronizar a coleta e análise dos requisitos do projeto, seja por meio de uma técnica já existente ou por um protocolo próprio criado pela equipe.

Um dos principais motivos para atraso e cancelamento de projetos de software são os requisitos mal levantados. Portanto, uma boa engenharia de requisitos (que consiste em todo o processo de coleta, análise, alteração e implantação de requisitos junto ao cliente) bem feita tende a diminuir os possíveis fracassos de um projeto.

Autores destacam que o uso de protótipos de software têm impacto positivo no gerenciamento das mudanças constantes dos requisitos.

Sugere-se também que se faça avaliações periódicas e constantes quanto à forma que os requisitos estão sendo coletados, com o objetivo de melhorar a Engenharia de Requisitos continuamente.

Além disso, sugere-se também a elaboração de formas de evitar ou minimizar mudanças drásticas de requisitos quando da mudança de pessoas envolvidas ao projeto (chefias, responsáveis etc.).

Também conhecida como

Usar método de coletar requisitos.

FC relacionado(s)

Pouco envolvimento do usuário e falta de apoio organizacional.

Consequências

Ao utilizar uma técnica para engenharia de requisitos é possível abstrair a maioria dos detalhes envolvidos e focar essencialmente no passo a passo que deve ser feito para coletar e analisar requisitos apropriadamente com um protocolo já pronto.

Mais uma vez, um ponto negativo pode ser a curva de aprendizado da equipe, que pode ser longa.

Há autores que propõem uma forma de priorização dos requisitos do cliente chamada CCR, que traduzido para o português significa Satisfação do Cliente, Custo e Contagem de Regressão. A técnica consiste em selecionar para implementação os requisitos essenciais ao usuário e que têm menor contagem de regressão (uma medida que define a quantidade de testes necessários). Desta forma, os testes de regressão gastam menor tempo durante o projeto.

Outros autores sugerem a utilização da técnica FAST (Técnica de Especificação de Aplicação Facilitada, em português) para que se obtenha uma melhor comunicação do cliente e consiga levantar os requisitos mais apropriadamente. A FAST consiste em uma roda de diálogo entre equipe de desenvolvimento, cliente e um facilitador, pessoa que conduz a sessão e é o

personagem-chave da técnica.

Por fim, também sugere-se uma forma de verificar e validar os requisitos através da ontologia, ramo da ciência da computação que estuda o relacionamento entre conceitos sobre dados coletados.

Boa(s) Prática(s) relacionada(s)

BOA PRÁTICA #04 - USAR PRINCÍPIOS ÁGEIS

Dica: pesquise sobre **Estórias de Usuários** (*User stories*) e utilize-as no processo de coleta e priorização de requisitos.

Priorização

Autor(es)
Gupta, Chauhan e Dutta (2013)

Proposta
Os autores propõem uma forma de priorização de requisitos (chamada CCR) para poder focar nos mais essenciais ao usuário e nos que tem menor carga aos testes de regressão (os testes de regressão são feitos com versões mais recentes do software para identificar novos erros). A adoção de técnicas de engenharia de requisitos pode aumentar a qualidade do software, pois requisitos mal interpretados, faltantes ou incompletos podem causar incorreções ao produto de software.

A prioriade do requisito é determinada pela Satisfação do usuário (*Customer satisfatcion*), Custo (*Cost*) e o Contagem da [do teste de] regressão (*Regression Count*). O objetivo é não somente focar na satisfação do usuário e no custo, mas também no custo computacional envolvido nos testes.

Um algoritmo é usado para definir os requisitos a serem implementados que tenham causem máximo de satisfação do usuário, tenham o mínimo de custo e necessitem o mínimo de testes de regressão. A Fig. 1 resume o passo a passo do método proposto.

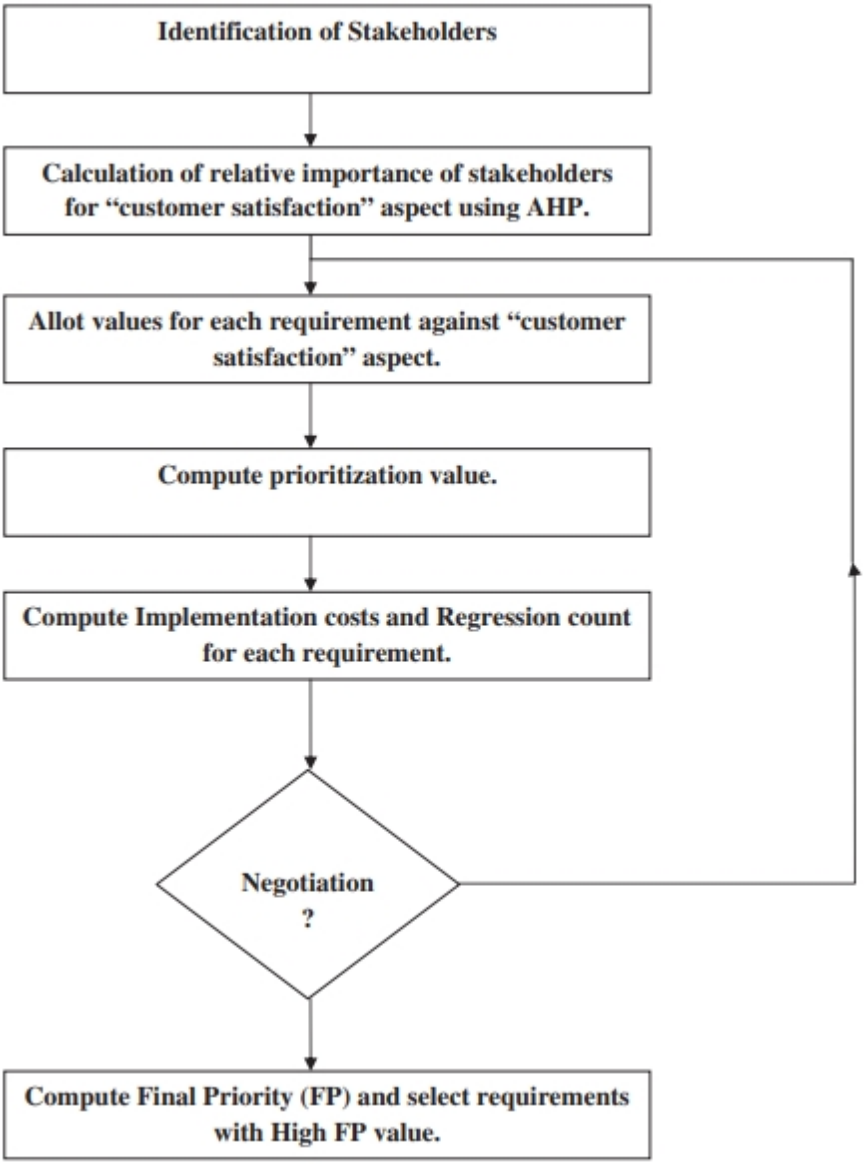


Fig. 1 - Modelo de priorização de requisitos proposto (Gupta, Chauhan e Dutta, 2013)

Exemplo prático dos autores

Por entenderem que institutos de educação demandam sistemas complexos de *E-learning*, os autores desenvolveram um projeto de software de ensino (*E-learning*) utilizando o algoritmo proposto (CCR) e sugeriram como trabalho futuro a aplicação deste software em outros contextos para coleta de impressões dos alunos.

Possível aplicação nos IFs

A aplicação de uma forma de priorizar os requisitos é essencial para a saúde do projeto de software. Desta forma, embora tenham técnicas complexas para priorização, é possível fazê-la de forma mais simples e independente de um algoritmo específico. Por exemplo, pode-se tentar entrar em consenso com os clientes para que as primeiras funcionalidades a serem entregues sejam as que menos carecem de processamento computacional ou utilização do banco de dados, desta forma o custo será mais baixo na realização dos testes.

JAD, priorização e outros

Autor(es)
Jones (2004)

Proposta
O autor elenca várias atitudes que podem ser tomadas pela equipe para que se possa ter uma engenharia de requisitos efetiva:

1. Criar um quadro onde os clientes e aquipe possam fazer o controle das mudanças dos requisitos;
2. Usar *JAD* (*Joint Application Design*: metodologia que acelera o projeto de sistemas criada pela IBM em 1977) para minimizar as alterações de requisitos;
3. Uso de protótipos (**BOA PRÁTICA #07 - USAR PROTÓTIPOS**);
4. Uso planejado de desenvolvimento iterativo (**BOA PRÁTICA #04 - USAR PRINCÍPIOS ÁGEIS**);
5. Revisões formais de todos os requisitos;
6. Priorização de requisitos (**Priorização**).

Exemplo prático do autor
O autor analisou 250 projetos de software entre 1995 e 2004 e constatou um padrão entre eles: em comparação entre os projetos que foram concluídos com sucesso, dentro das estimativas de custo e tempo, e os projetos mal sucedidos (atrasados, cancelados), seis problemas comuns são observados: mal planejamento, má estimativa de custo, métricas mal realizadas, definição de *milestone* mal elaborada, fraco controle de mudanças (aqui se incluem as mudanças de requisitos) e controle de qualidade pobre (Fig. 1). Em contraste, os projetos de sucesso tendem a ter bons resultados nesses seis aspectos. O interessante é que todos podem ser mais relacionados ao **planejamento do projeto** do que com fatores técnicos ou pessoais.

Successful Projects	Failing Projects
Effective project planning	Inadequate project planning
Effective project cost estimating	Inadequate cost estimating
Effective project measurements	Inadequate measurements
Effective project milestone tracking	Inadequate milestone tracking
Effective project change management	Ineffective change control
Effective project quality control	Inadequate quality control

Fig. 1 - Principais fatores que influenciam o sucesso dos projetos segundo Jones (2004)

Possível aplicação nos IFs
A criação de algum quadro (online ou offline) para que os clientes e a equipe consigam acompanhar (e discutir) o andamento dos requisitos pode ser aplicado nos IFs e auxiliar na comunicação entre as partes, além de agilizar as ações da equipe.

O uso de protótipos, como já indicado na **BOA PRÁTICA #07 - USAR PROTÓTIPOS** pode trazer um encurtamento de caminho entre os clientes e a equipe, uma vez que é uma técnica fácil de ser aplicada dado seu baixo custo.

Verificação & Validação

Autor(es)
Nazir et al. (2014)

Proposta
Por entenderem que é necessário resolver conflitos, reconhecer falhas e elaborar um documento de requisitos completamente útil, os autores propõem uma forma de verificar e validar tais requisitos através da ontologia (um modelo de dados que representa conceitos e o relacionamento entre eles). Tais atividades são essenciais ao sucesso da engenharia de requisitos e da qualidade do software (garantem que os requisitos não sejam desnecessários e sejam confiáveis).

A Fig. 1 resume o complexo modelo proposto.

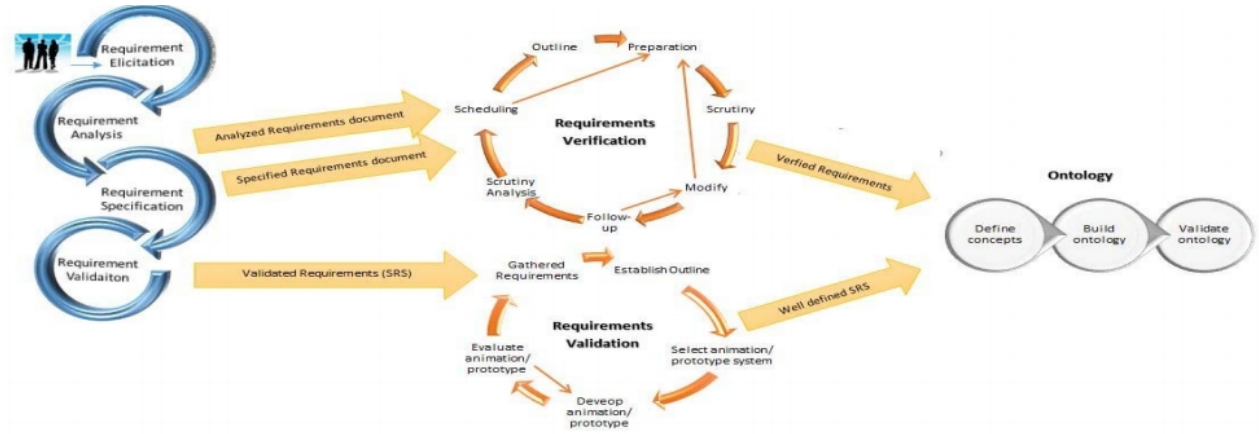


Fig. 1 - Modelo de verificação e validação de requisitos através da ontologia (Nazir et al.,2014)

O modelo proposto tenta resolver prolemas como ambiguidade, incompletude e inconsistências dos requisitos, a fim de minimizar o tempo e o custo dos projetos.

Possível aplicação nos IFs
Embora bastante técnico, a obra de Nazir et al. (2014) demonstra a importância de uma verificação e validação de requisitos bem feita, com o objetivo de entregar um software de qualidade, no prazo previsto e com o custo planejado.

Para aplicação nos IFs sugere-se, portanto, uma atenção maior nas fases de **verificação e validação** de requisitos.

FAST

Autor(es)
Shahzad e Mathkour (2009)

Proposta
A FAST (*Facilitate Application Specification Technique*) surge como uma técnica de encurtar o caminho entre o cliente e o desenvolvedor.

A técnica consite na realização de uma reunião entre a equipe de desenvolvimento e o cliente onde eles sentam-se em uma mesa redonda e são conduzidos por um **facilitador**.

O **facilitador** é a pessoa responsável por conduzir a reunião e é o papel mais importante na sessão FAST (o sucesso da aplicação da técnica dependede quão bem o facilitador irá conduzí-la). O papel do facilitador é conduzir a sessão FAST de modo que o máximo de requisitos sejam entendidos e as amibiguidades sejam eliminadas.

A eficácia do facilitador em uma sessão FAST depende de:
1. **Sucesso**: o facilitador deve saber conduzir apropriadamente toda a reunião;
2. **Autoridade**: o facilitador pode tomar decisões e resolver mal entendidos;
3. **Resolução de conflitos**: o facilitador deve identificar e resolver os requisitos conflitantes entre os dois grupos (equipe e clientes);
4. **Consolidação dos requisitos**: o projeto não pode começar enquanto os requisitos não estiverem em comum acordo entre os dois grupos. O facilitador deve assegurar que isso ocorra;
5. **Respeito**: a pessoa escolhida para essa função deve ser respeitada por ambos os grupos;
6. **Melhoria de relações**: o facilitador deve encontrar um lugar neutro para a realização da reunião;
7. **Definição de requisitos**: o faciltador deve incentivar o cliente a definir o máximo de requisitos possível e, após isso, executar a verificação e validação desses requisitos;
8. **Economia de tempo**: com a sessão FAST, elimina-se várias entrevistas com ambos os grupos, economizando o tempo dos profissionais envolvidos.

Exemplo prático dos autores
Um questionário de vinte e cinco questões foi foi feito pelos autores para que seja aplicado aos mais diversos tipos de pessoas da área de TIC com o objetivo de identificar as cinco principais caracterísitcas essenciais de um facilitador.

Como conclusão tem-se comprometido, honesto e confiável, resolvedor de conflitos, comunicativo e experiente (Fig. 1).

Q. No.	Question	Responses
10	Experience	35
19	Communication Skills	37
21	Conflict Resolution Approach	38
17	Honest and Reliable	40
9	Commitment	40

Fig. 1 - Resultado da pesquisa sobre as principais características que um facilitador deve possuir (Shahzad e Mathkour, 2009)

Os resultados mostram que os indivíduos preferem uma pessoa comprometida, honesta e confiável, além de poder resolver conflitos, o que são características chave para qualquer facilitador.

Possível aplicação nos IFs
A sugestão dos autores podem ser aplicadas na fase de levantamento de requisitos dos IFs desde que as instituições encontrem uma boa pessoa para fazer o papel do **facilitador**. Desta forma, o número de entrevistas tende a diminuir e os requisitos podem ser levantados de forma mais clara, objetiva e com o mínimo de ambiguidades possível.

BOA PRÁTICA #02 - AVALIAR LÍDERES

Nome da boa prática

Avaliar líderes

Objetivo

Escolher e avaliar os líderes de equipes e gerentes de projetos pode ser uma tarefa dispendiosa caso a organização não possua uma forma organizada para isso. Estas funções servem tanto para direcionar as equipes quanto para motivá-las, portanto há a necessidade de se fazer boas escolhas das lideranças do grupo.

Adicionalmente, nada impede de se fazer um rodízio de lideranças da equipe para aumentar a motivação e coesão de todos.

Para aplicação, sugere-se analisar a sinergia e a evolução da equipe no decorrer de certo período de tempo e verificar se há muitos conflitos entre os membros é um passo que pode ser dado no início. Em seguida, sugere-se aplicação de metodologias, como a Teoria da Difusão da Inovação para se avaliar as lideranças dos grupos.

Sugere-se também adicionar as avaliações dos líderes à própria metodologia de desenvolvimento utilizada.

Também conhecida como

Avaliação dos *CEO's*.

FC relacionado(s)

Falta de apoio organizacional.

Consequências

Fazer avaliação periódica dos líderes pode dar noção ao gestor do desempenho dos profissionais envolvidos, ajudando na tomada de decisão (troca de comando, por exemplo).

Boa(s) Prática(s) relacionada(s)

BOA PRÁTICA #05 - OFERECER CAPACITAÇÃO À EQUIPE

Dica: pesquise mais a fundo sobre a Teoria da Difusão da Inovação, de Everett Rogers, na

internet, e leia a subseção correspondente.

DOI

Autor(es)
Govindaraju, Hariadi e Sidiq (2015).

Proposta
Os autores propõem avaliar as caracterísiticas dos líderes (alta gestão da organização) através da *DOI - Diffusion Of Innovation* (Difusão da Inovação), com o objetivo de examinar suas influências em sistemas *ERP - Enterprise Resource Planning* (Sistema Integrado de Gestão). A Fig. 1 ilustra o ciclo de vida dos sistemas ERP.

O foco da utiliação da DOI é na fases pós-implementação (*Implementation*).

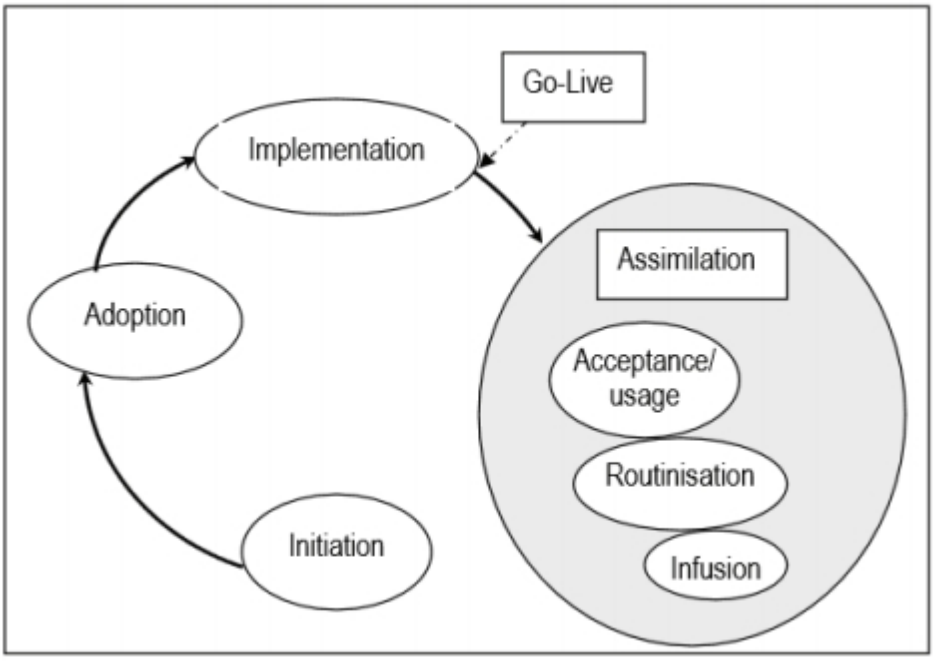


Fig. 1 - O ciclo de vida de sistemas ERP (Govindaraju, Hariadi e Sidiq, 2015)

O trabalho busca entender se a falta de envolvimento e falta de capacidade de inovar dos líderes causam falhas nos projetos (principalmente em sistemas ERP) e, com o uso da DOI, foi possível fazer essa relação, pois esta técnica ajudou a entender o nível de aceitação dos líderes para se utilizar novas tecnologias (**inovação**).

Os autores exploram três principais caracterísiticas dos líderes para testarem suas hipóteses: **Apoio**, **Capacidade de Inovar** e **Conhecimento**. São três pilares para se conduzir a uma pós-implementação com menos falhas.

Exemplo prático dos autores
Os autores testaram o modelo proposto usando o método SEM (*structural equation modeling*) em comparação com dados coletados através de questionários aplicados a usuários de sistemas ERP de 73 empresas. Os resultados mostram que apoio da alta gestão, capacidade de

inovar do CEO e conhecimento do CEO influenciam significativamente os sucesso dos projetos ERP.

Possível aplicação nos IFs
O uso da DOI permitiu concluir que as características dos líderes influenciam de forma direta no sucesso ou no fracasso de projetos de software, principalmente em sistemas ERP.

Desta forma, é importante para os projetos de software dos IFs que os **líderes motivem as equipes a continuarem usando, mantendo e sugerindo melhorias nos sistemas ERP**, provendo suporte, recursos e conhecimento.

Além disso, **a capacidade e a vontade de inovar do líder** motiva a equipe a buscar novas soluções frequentemente sem receio de discordância por parte da chefia.

Portanto, pode-se salienter que o estudo mostrou que é importante ao ERP a **busca por inovação** tanto dos **líderes** (sejam eles diretores, coordenadores, chefias, gerentes de projetos, entre outros) quanto das **equipes** de desenvolvimento (programadores, analistas, técnicos em TIC).

Por fim, o estudo sugere que é necessário que as **organizações ofereçam capacitação** (**BOA PRÁTICA #06 - OFERECER CAPACITAÇÃO À EQUIPE**) aos líderes a fim de eles sejam mais suscetíveis a aceitar propostas de inovação, uma vez que têm conhecimento sobre os benefícios da mesma, além de poder distribuir seu conhecimento às equipes posteriormente.

BOA PRÁTICA #03 - USAR FERRAMENTAS DE GERENCIAMENTO DE PROJETOS

Nome da boa prática

Usar ferramentas de gerenciamento de projetos

Objetivo

Consiste em utilizar softwares e técnicas para gerenciamento de projetos de software. Estas ferramentas auxiliam o gerente de projetos e a equipe a dividir tarefas, melhorar as estimativas de tempo e custo das atividades, acompanhar todas as fases do projeto, gerar relatórios entre muitas outras facilidades.

Alguns exemplos incluem a utilização de um quadro Kanban, listas de tarefas (*todo lists*) ou softwares como *MS Project*, *Tuleap*, *Redmine* e *Trello*.

Para aplicação, pode-se entrar em acordo com a equipe para seleção das melhores ferramentas disponíveis no mercado que se adequem às necessidades atuais. Posteriormente é necessário um treinamento para que todos saibam usar as ferramentas apropriadamente.

Além disso, há autores que sugerem algumas ferramentas para melhores estimativas para projetos: Before You Leap, Checkpoint, Software Life Cycle Management, SEER-SEM, entre

outras. Para a fase de planejamento, sugerem também as técnicas de Gráficos Gantt, gráficos PERT ou similares.

Outra sugestão da literatura consultada é a ferramenta MS-Project. Os autores destacam que atualmente uma ferramenta como esta é essencial na indústria de software. Planejar o projeto no MS-Project, por exemplo, pode reduzir o custo do atraso de todas as operações.

Sugere-se ainda:

- Utilização de um quadro Kanban físico fixado na parede das salas das equipes;
- Adaptar os projetos de software aos processos da instituição;
- Utilização de alguma ferramenta/pessoa para organizar os documentos e as agendas de reuniões de *sprint.*

Também conhecida como

Utilizar algum software de gestão de projetos.

FC relacionado(s)

Falta de metodologia de Gestão de Projetos de Software.

Consequências

O uso de alguma ferramenta auxilia a equipe no controle e divisão de tarefas, além de facilitar a previsão dos prazos. Isso pode ajudar a equipe a ter um melhor controle das atividades, evitando desperdício de tempo.

Boa(s) Prática(s) relacionada(s)

BOA PRÁTICA #06 - USAR TÉCNICAS DE ESTIMATIVAS

Dica: acesse **tuleap.org** para conhecer a ferramenta.

MS Project

Autor(es)
Pradhan, Rishiwal e Agarwal (2019)

Proposta
Os autores sugerem a utilização de uma ferramenta de gestão de projetos de software específica, chamada *Microsoft Project*, pois assim é possível melhorar todo o ambiente de desenvolvimento e reduzir o atraso do projeto.

Em geral, a falta de planejamento ou um planejamento mal feito causa atraso nas entregas dos projetos de software (a Fig. 1 exibe um gráfico da pesquisa realizada pelos autores demonstrando a importância de se planejar). A ferramenta *MS Project* é útil pois pode auxiliar na divisão de tarefas entre os membros das equipes e estabelecimento de prazos e *milestones* (um marco ou ponto significativo no projeto).

O planejamento de projetos de software usando ferramentas como o *MS Project* é a ação que todo gerente de projeto de software deve tomar para gerenciar efetivamente o software e as atividades de desenvolvimento durante todas as fases.
- Pradhan, Rishiwal e Agarwal (2019)

Outro recurso interessante é a criação de relatórios que auxiliam os gerentes na tomada de decisão.

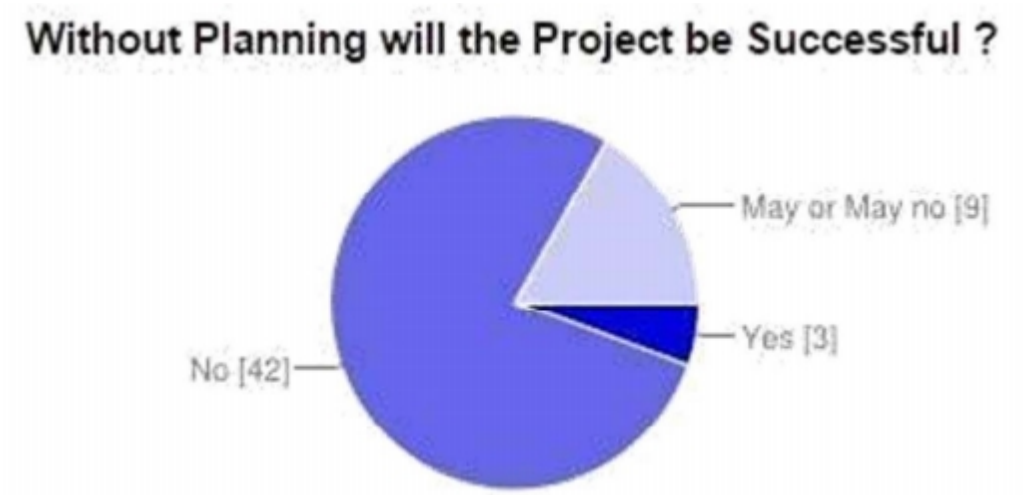


Fig. 1 - Gráfico de respostas para a pergunta "Sem planejamento o projeto terá sucesso?" (Pradhan, Rishiwal e Agarwal, 2019)

Para adquirir a ferramenta, acesse o site do desenvolvedor.

Exemplo prático dos autores
Os autores realizaram uma pesquisa de campo para compreender a importância do uso de ferramentas de gestão de projetos e constataram que seu uso adequado interfere positivamente no ambiente de desenvolvimento e pode reduzir o custo e o prazo do projeto.

Possível aplicação nos IFs
Embora os autores tenham sugerido o *MS Project*, **existem outras ferramentas**, tanto gratuitas quanto pagas, que podem auxiliar as equipes de TIC dos IFs a planejarem seus projetos de software.

O importante é utilizar alguma ferramenta para o gerenciamento, pois dessa forma é possível dividir as tarefas entre os membros da equipe, planejar entregas , estimar e medir os prazos de cada projeto, pois "se os times de desenvolvimento planejam com ajuda de uma ferramenta fica mais fácil para todos trabalharem e darem o seu melhor **sem atraso**, e isso conduz a uma melhor utilização de recursos de tempo, orçamento e custos".

Lista de ferramentas

Autor(es)
Jones (2004)

Proposta
Ao invés de focar em uma ferramenta somente, Jones cita sucintamente várias delas, a saber:

1. **Ferramentas de gerenciamento de projetos**

Termo aplicado para uma ampla família de ferramentas onde o objetivo é organizar e planejar as inúmeras tarefas dos projetos. Tais ferramentas são capazes de se expandir em inúmeras tarefas menores com alto nível de detalhe, por exemplo. O autor frisa que as ferramentas são importantes, mas não essenciais, pois desenvolver software requer outras capacidades.

Exemplos: Artemos Views, Microsoft Project, Primavera, Project Manager's Workbench.

2. **Ferramentas de métricas**

Algumas tarefas não são especialidades das ferramentas de gerenciamento de projetos. É o caso de ferramentas de métricas e estimativas (esse assunto também pode ser visto na **BOA PRÁTICA #06 - USAR TÉCNICAS DE ESTIMATIVAS**).

Exemplos: Before you Leap, CHECKPOINT, Constructive Cost Model (COCOMO) II, CostXpert, KnowledgePlan, PRICE-S, SEER-SEM e SLIM.

O autor destaca que **projetos de sucesso usam ao menos uma ferramenta, enquanto o uso de duas ou mais não é incomum**. Em contraste, a maioria dos projetos que falham não usam nenhuma ferramenta.

Exemplo prático do autor
Pode ser acessado aqui.

Possível aplicação nos IFs
Fica claro que existem diversas ferramentas para as mais diversas funcionalidades. É possível, inclusive, encontrar ferramentas gratuitas. Desta forma sua utilização e aplicação é facilitada, visto que se esquiva de passar pela burocracia da máquina pública (licitação, processo de liberação de pagamento, entre outras).

Portanto, frisa-se novamente a importância do uso de ferramentas especializadas em Gestão de Software, a fim de otimizar o planejamento, economizar recursos, entregar software no prazo e cumprir os objetivos organizacionais.

BOA PRÁTICA #04 - USAR PRINCÍPIOS ÁGEIS

Nome da boa prática

Usar princípios ágeis

Objetivo

O uso de uma metodologia ágil de desenvolvimento, com base nos princípios do "Agile Manifesto" tende a trazer melhoras em vários aspectos dos projetos, pois torna a equipe mais tolerante à mudanças, propõe maior interação com o cliente e evita um grande volume de documentação (o que às vezes é inútil), porém é importante ressaltar que é necessário seguir o protocolo ágil à risca para que os efeitos possam ser significativos.

Além disso, um dos princípios ágeis é o desenvolvimento iterativo, que percorre todas as fases do ciclo de vida de um projeto várias vezes, buscando aperfeiçoamento, correção de erros e otimização de funcionalidades. Usando este tipo de técnica, os produtos de software tendem a ser de maior qualidade e mais adaptáveis à mudanças de requisitos.

Para se adequar melhor à realidade da organização, pode-se ainda customizar a utilização do Scrum, que é um dos tipos de métodos ágeis. Pode-se substituir ou remover artefatos, alterar a duração e periodicidade de reuniões e reorganizar alguns processos. A equipe e a rotina de trabalho de TIC da organização são mais importantes que o método em si, portanto é esse que deve se adequar àquelas.

Para fazer uso desta Boa Prática, deve-se entender e aplicar aos poucos os princípios sugeridos pelo Manifesto Ágil. Posteriormente é necessário escolher uma metodologia que mais se adequa à realidade da organização. Exemplos como Scrum e XP são válidos.

É necessário também iniciar um processo de mudança de cultura da equipe, principalmente se esta faz uso de um modelo Cascata. É sugerido aplicar aos poucos o desenvolvimento iterativo, o qual percorre todas as fases do ciclo de vida de um projeto (concepção, elaboração, construção, transição...) diversas vezes antes de uma entrega. Para isso é necessário revisar e alterar o processo de desenvolvimento, caso exista algum.

Pode-se também mapear o processo da organização e compreender a sua cultura e rotina. Não há impedimento de se mesclar artefatos e técnicas de outras metodologias ágeis caso haja necessidade.

Alguns autores sugerem um framework para avaliar a eficácia da gestão dos projetos de software baseado em *soft computing*, ramo da informática que produz cálculos para auxiliar na resolução de problemas computacionais complexos.

O uso do Scrum personalizado também é defendido pela literatura. Trata-se de uma forma adaptada de usar o Scrum, porém sem seus maiores pontos fracos: os papéis e as responsabilidades de cada membro da equipe devem ser claramente apresentados a todos, principalmente àqueles que experimentam Scrum pela primeira vez, através de um passo-a-passo. Em segundo lugar, uma nova fase foi incorporada ao processo com o objetivo de planejar e esboçar o projeto como um todo. Já a dificuldade de planejar e coordenar sprints, devido às operações múltiplas e interdependentes, pode ser resolvida através do compartilhamento dos status das atividades.

Uma sugestão é a aplicação de avaliações periódicas aos envolvidos com o objetivo de captar os pontos fortes e pontos fracos da metodologia utilizada, a fim de melhora-la continuamente.

Também conhecida como

Utilizar Scrum, XP, etc. Fazer uso do *Agile Manifesto*.

FC relacionado(s)

Pouco envolvimento do usuário, falta de apoio organizacional e falta de metodologia de Gestão de Projetos de Software.

Consequências

A utilização de um Método Ágil para gerir e desenvolver software tem como vantagens a rápida adaptação a mudanças repentinas de requisitos, a entrega incremental ao cliente (evitando deixá-lo esperar muito tempo pelo produto) e um controle maior sobre as tarefas a serem realizadas. Desta forma, tem-se uma noção melhor do estado atual do projeto: o que já foi feito, o que falta fazer, quais os melhores caminhos percorrer e, ainda, uma estimativa mais exata de quanto tempo ainda pode demorar para o produto ser entregue.

Outra vantagem é a divisão do tempo em *sprints*, que são fatias temporais onde são entregues pedaços do produto de software ao cliente final.

Como desvantagens, tem-se a demora na adaptação da equipe, principalmente se esta estiver acostumada com modelos tradicionais de gestão de software (Modelo Cascata, por exemplo).

Boa(s) Prática(s) relacionada(s)

BOA PRÁTICA #01 - USAR TÉCNICA DE ENGENHARIA DE REQUISITOS

Dica: acesse os sites **www.agilemanifesto.org** e **www.desenvolvimentoagil.com.br/scrum**

Desenvolvimento iterativo

Autor(es)
Jones (2004)

Proposta
O autor propõe o uso planejado do desenvolvimento iterativo com o objetivo principal de se adequar rapidamente às possíveis mudanças de requisitos do software.

Desenvolvimento iterativo trata-se de um método em que você desenvolve o sistema através de ciclos repetitivos (Fig. 1). No ciclo tradicional (*Waterfall* - Cascata) você só constata que ficou livre de algum risco ao final do ciclo. No ciclo iterativo, em cada fase se produz um executável testável, possibilitando ver se os riscos diminuíram ou aumentaram.

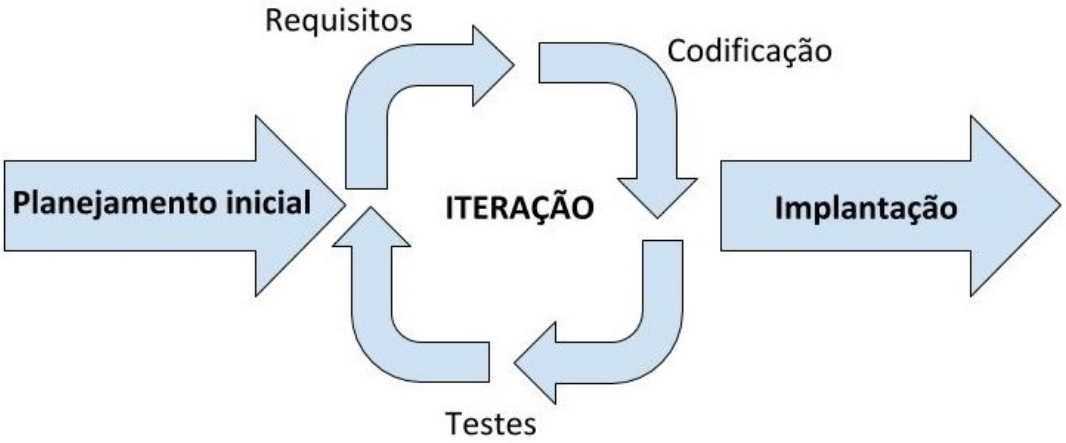


Fig. 1 - Ilustração sobre ciclo iterativo (do autor)

Principais recomendações
Uso do desenvolvimento iterativo de software.

Exemplo prático do autor
Pode ser acessado aqui.

Possível aplicação nos IFs
A aplicação do método iterativo de desenvolvimento é uma das propostas do desenvolvimento ágil de software (**BOA PRÁTICA #4 - USAR PRINCÍPIOS ÁGEIS**).

Sua aplicação pode ser feita utilizando-se também outros princípios do Manifesto Agil e sugere-se que treinamentos ou workshops sejam aplicados às equipes para que estas entendam os principais conceitos envolvidos (**BOA PRÁTICA #05 - OFERECER CAPACITAÇÃO À EQUIPE**).

A aplicação de um desenvolvimento iterativo não tem custo financeiro à organização, visto que é apenas um modelo conceitual de trabalho. Desta forma se esquiva de burocracias inerentes ao setor público, facilitando sua aplicação.

Dica: iterativo é diferente de **interativo**!

Scrum personalizado

Autor(es)
Hong, Yoo e Cha (2010)

Proposta
Segundo o estudo, a aplicação do Scrum customizado aumentou a qualidade do produto e reduziu o tempo necessário para completar o projeto.

O Scrum é popular pois ele pode ser aplicado rápida e facilmente, porém apresenta alguns problemas: os papéis dos membros das equipes são pouco claros no início; na fase de planejamento foca-se muito nos incrementos, deixando de lado o projeto como um todo, não permitindo ter-se uma visão macro de todo o projeto e os relatórios de progresso utilizando

gráficos *burn-down* são poucos compreensíveis.

Assim sendo, o autor propõe algumas modificações ao Scrum (detalhes abaixo).

Principais recomendações

1. Papéis e respnsabilidades devem ser apresentadas claramente aos membros da equipe que estão trabalhando com Scrum pela primeira vez (Fig. 1);
2. Uma fase adicional de planejamento foi inserida a fim de planejar o projeto como um todo;
3. Para a fase de *review*, o relatório deve ser baseado no número completo de páginas web, e não em pontos de esforço.

	Product backlog Review	Master Sprint Planning	Individual Sprint Planning	Daily meeting	Sprint review
Goal	Review and share product backlog	-Measure estimated scoring for product backlog -Determine the number of sprint -Set the milestone	-Set the goal for individual sprint planning -Produce Work progress table	-Share daily progress and updates -Check & resolve issues	-Demonstrate Sprint deliverables -Update the progress
Scrum Master	Review product backlog	-Hold meetings -Review entire product backlog -Determine the total number of Sprint -Review the milestone	-Hold meeting -Review work progress	-Manager progress -Manage issues -Manage schedules	-Update progress -Maintain issues -Maintain schedule
Product Owner	Delivery product backlog	-Measure estimated scoring for entire product backlog -Set goals for the entire Sprint	-Select backlog to execute during individual Sprint -Review work progress chart	Support issue resolution	-Check product -Check requirements that are missing in implementation
Scrum Team (Designer/Coder/ Developer/QA)	Review product backlog	-Measure estimated scoring for entire product backlog -Set the goals for entire Sprint	-Set the goals for individual Sprint by each operation process -Produce work progress chart	-Completed tasks -Planning tasks -Issues	-Demonstrate deliverables

Fig. 1 - Tabela de papéis e responsabilidades proposta pelo autor (Hong, Yoo e Cha, 2010)

Exemplo prático dos autores

Os autores aplicaram o Scrum cutstomizado em dois projetos e os resultados foram a melhoria da qualidade do produto e a redução no tempo necessário para conclusão. Além disso, mais de 80% das pessoas envolvidas afirmaram satisfação na proposta.

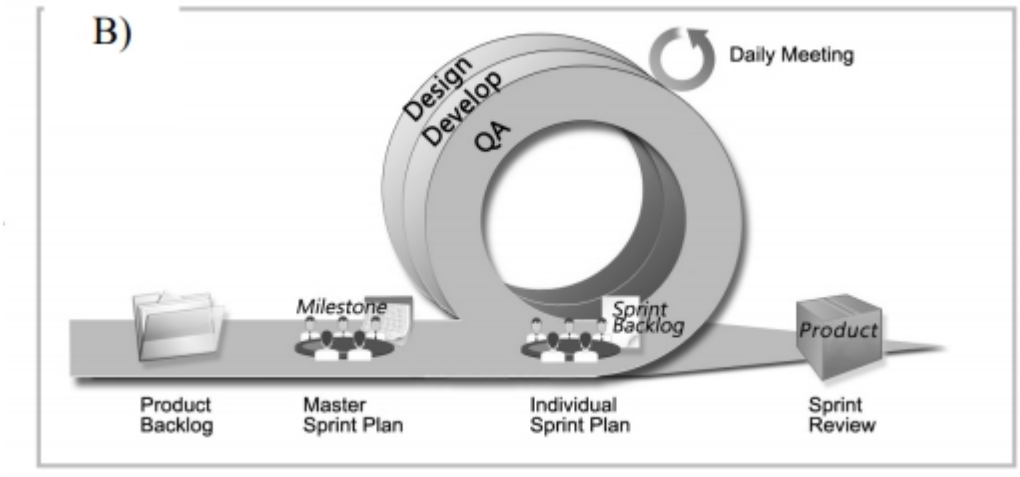


Fig. 2 - Ilustração do Scrum customizado (Hong, Yoo e Cha, 2010)

Possível aplicação nos IFs

O estudo ajuda a compreender que embora a aplicação de uma metodologia ágil de desenvolvimento possa ser bastante útil aos projetos, também **é importante customizar e adaptar ao contexto**.

Portanto, ao aplicar conceitos ágeis deve-se levar em consideração a cultura organizacional, o

tamanho da equipe, o número de projetos, entre outras variáveis. A sugestão de Hong, Yoo e Cha (2010) pode ser um pontapé inicial para experimentações (por exemplo, pode-se aplicar uma tabela de divisão de papéis e responsabilidades no início de cada projeto, ou no início de cada *sprint* do projeto).

BOA PRÁTICA #05 - OFERECER CAPACITAÇÃO À EQUIPE

Nome da boa prática

Oferecer capacitação à equipe

Objetivo

Aprimorar conhecimento e reciclar técnicas conhecidas são de vital importância, principalmente em equipes de desenvolvimento de software, dado o contínuo avanço das tecnologias. Assim sendo, promover treinamentos à equipe é uma boa prática para se obter novos conhecimentos e otimizar a execução das tarefas.

Atualmente o acesso à informação é facilitado, logo torna-se viável a aplicação de cursos, treinamentos e certificações online.

Também é possível selecionar algum membro experiente e que possua vasto conhecimento em determinado assunto para que se aplique um workshop ou minicurso *in loco* para toda a equipe.

É indicado entender quais tecnologias, métodos e ferramentas a equipe irá utilizar e, em seguida, fazer um levantamento das necessidades de cada membro. Em geral, há deficiências em tecnologias de desenvolvimento (linguagens de programação), tecnologias de versionamento (como *Git*), coleta de requisitos (**BOA PRÁTICA #01 - USAR TÉCNICA DE ENGENHARIA DE REQUISITOS**) ou gestão de projetos. Em seguida, sugere-se definir um período do dia para dedicação de treinamentos, sejam eles online ou presenciais.

Alguns autores sugerem uma cooperação intensa entre universidades e indústria de software, a fim de melhorar as capacidades dos estudantes.

Sugere-se também a auto-capacitação, que se dá quando o próprio funcionário reconhece suas deficiências e busca treinamentos por conta própria, com o objetivo de aprimoramento profissional.

Além disso, uma forma de equilibrar o conhecimento das equipes se dá por meio do Nivelamento. Para tal, pode-se incentivar a troca de informações e experiências entre os membros das equipes, com workshops ou seminários, para maior difusão do conhecimento.

Também conhecida como

Fornecer treinamento à equipe.

FC relacionado(s)

Pouco envolvimento do usuário e Falta de Apoio organizacional.

Consequências

Capacitar a equipe oferece novos meios de tratar as tecnologias existentes, podendo trazer celeridade às atividades dos projetos e novas maneiras de resolver problemas. Em contraponto, um profissional muito bem qualificado pode deixar a equipe devido propostas melhores da indústria.

Boa(s) Prática(s) relacionada(s)

BOA PRÁTICA #02 - AVALIAR LÍDERES

Dica: há vários cursos bons e gratuitos em **Udemy.com**

Treinamentos especializados

Autor(es)
Suliman e Kadoda (2017)

Proposta
Os autores destacam a importância das estimativas para a condução do projeto de software (**BOA PRÁTICA #06 - USAR TÉCNICAS DE ESTIMATIVAS**) e salientam que que a falta de um treinamento apropriado na área de gerenciamento de riscos é um fator de fracasso para os projetos de software.

Em pesquisa realizada, concluíram que o **treinamento** e a **experiência** de gerentes de projetos e desenvolvedores são um dos diversos fatores que influenciam o processo de estimativa.

Principais recomendações

1. Aplicar um sistema de feedback que conduz à uma análise de custo e de orçamento e determina as áreas que existem riscos de falhas. Uma vez que estas áreas sejam identificadas, pode-se aplicar treinamentos aos desenvolvedores e gerentes de projetos iniciantes, com o objetivo de explorar erros passados, discrepâncias, imprecisões, fatores que auxiliam as estimativas e métodos de estimativas.
2. A participação entre a indústria e as universidades podem aumentar as habilidades dos estudantes e aumentar suas experiências. A indústria pode auxiliar o sistema educacional provendo projetos reais onde os estudantes podem praticar antes de encararem problema semelhante em sua carreira.

Possível aplicação nos IFs
Um dos principais pontos a se destacar na obra de Suliman e Kadoda (2017) é o **item 2** acima: a constante cooperação entre indústria e universidade.

Além de fornecer a possibilidade de aprendizado e experiência aos alunos, pode-se fazer algo semelhante com a equipe de desenvolvimento: em uma provável cooperação entre os dois setores, novos conhecimentos e novas soluções podem ser descobertas.

Um exemplo concreto é a criação de **Empresas Júnior**, onde os estudantes, a equipe de TI e a

indústria podem trabalhar em conjunto.

Além disso, fica claro a importância de **treinamentos** à equipe, principalmente no tocante à novas tecnologias e especificaidades de ferramentas e métodos.

BOA PRÁTICA #06 - USAR TÉCNICAS DE ESTIMATIVAS

Nome da boa prática

Usar estimativas e métricas

Objetivo

Só é possível obter melhoria contínua de um projeto se for possível medí-lo. O uso de métricas de software (por exemplo, *LOC*, *kLOC*, APF etc.) combinado com o uso de ferramentas de estimativas auxiliam a equipe a obter prazos mais realísticos para organizarem seu cronograma e transmitirem maior transparência aos clientes.

Para aplicação, a princípio é necessário entender o ritmo da equipe, ou seja, quantos incrementos é possível realizar em determinado tempo. Isso pode ser medido pela própria equipe em conjunto ao gerente de projetos ou gestor da organização. Em seguida sugere-se o uso de planilhas eletrônicas ou algum software de gerenciamento (BOA PRÁTICA #03) para que se possa documentar, organizar e planejar a execução das tarefas. Técnicas como *LOC* e *kLOC*, embora antigas, ainda podem auxiliar no processo. Existe também a técnica APF, mais condizente com as tecnologias de desenvolvimento atuais. O importante é encontrar uma forma de medir e estimar o progresso e as entregas de produtos.

Entre as sugestões de maior relevância encontradas na literatura, destaca-se:

- Uma abordagem que combina técnicas de estatísticas e aprendizagem de máquina para reestimar as fases subsequentes do projeto através de um pré-processamento do esforço necessário para as fases anteriores.
- Algoritmos como COCOMO II, SLIM, CostXpert etc.
- Técnicas como Redes Bayesianas e Planning Poker como exemplos de sugestões que podem ser utilizadas para melhores estimativas de tempo do projeto.
- A união de técnicas como PSO (otimização por enxame de partículas), COCOMO (Constructive Cost Model) e LOC (Lines Of Code) para determinar o esforço e estimar o cronograma de um projeto de software.
- Uso de Análise por Pontos de Função, com o objetivo de medir a funcionalidade que o usuário solicita e recebe, e medir o desenvolvimento e manutenção do software independentemente da tecnologia utilizada para implementação.
- Técnicas de PSO (otimização por enxame de partículas) e COCOMO (Constructive Cost Model), as quais conseguiram prover boa capacidade de estimativa comparadas com modelos tradicionais.
- Framework genérico, chamado GQM (meta, questão e métrica, português), para ajudar na validação de técnicas de estimativas e métricas.

Além dos citados acima, sugere-se também a utilização de Métricas Individuais, por exemplo: número de erros encontrados no código, tempo de codificação de determinada funcionalidade ou número de linhas codificadas.

Também conhecida como

Estimar e medir as etapas do projeto de software.

FC relacionado(s)

Pouco envolvimento do usuário e engenharia de requisitos mal feita.

Consequências

Melhor previsão de prazos de entrega e melhor controle dos passos de cada tarefa. Previsões e estimativas mais realistas, tanto pra equipe quanto para o cliente.

Boa(s) Prática(s) relacionada(s)

BOA PRÁTICA #03 - USAR FERRAMENTAS DE GERENCIAMENTO DE PROJETOS

Dica: embora tenham soluções mais complexas, entender e utilizar o **Planning Poker** é um bom começo.

Fuzzy Logic

Autor(es)
Govindarajan (2015)

Proposta
Na área de estimativas e métricas, alguns autores propõem utilização de técnicas para diminuição de incertezas relacionadas ao projeto.

Neste caso, o autor apresenta um framework que auxilia os gerentes de projetos no processo de tomada de decisão com técnicas de inteligência artificial (*Fuzzy Logic*, ou "Lógica Difusa", é um conceito computacional que se baseia em como os seres humanos pensam, que combina a manipulação de valores, incertezas, julgamentos e ideias vagas).

A Lógica Difusa é uma forma de predizer valores e é mais efetiva e melhor aplicada em sistemas não-lineares, ou seja, que não seguem um passo a passo exato a cada execução. A Fig. 1 ilustra o passo a passo da Lógica Difusa.

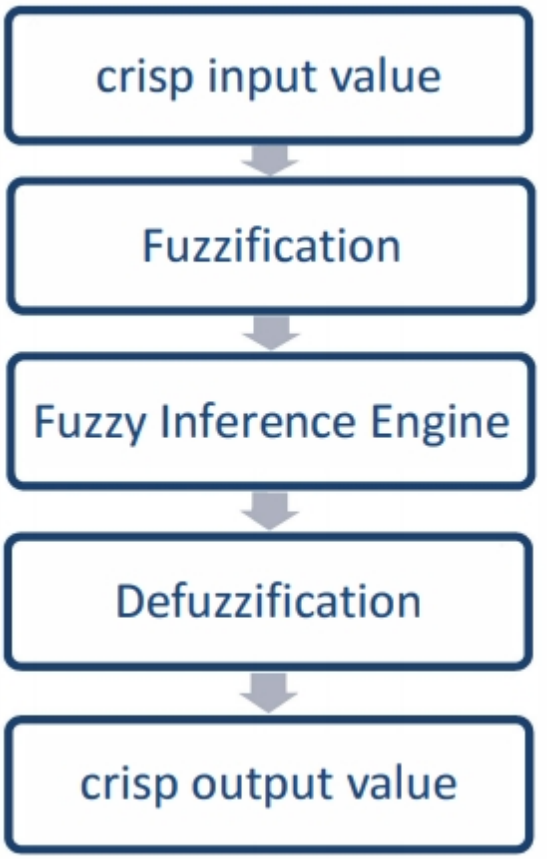


Fig.1 - Passo a passo básico da Lógica Difusa (Govindarajan, 2015)

Em seu trabalho, o autor concluiu que na indústria, na maioria das vezes, os gerentes de projeto não tem uma visão clara sobre os requisitos e isso resulta em perda de tempo, alto custo e falha de projeto. Esta proposta de utilização da Lógica Difusa usda para avaliar a eficácia do Gerenciamento dos Projetos de Software permite que os gerentes de projeto aliviem as deficiências que chegam no projeto sob circunstâncias vagas.

Principais recomendações
A utilização de Inteligência Artificial para auxiliar na tomada de decisão.

Exemplo prático do autor
O autor utiliza o *Sugeno Fuzzy Inference System* para diminuir as deficiências surgidas na gestão dos projetos de software devido à informações imprecisas ou incorretas. A proposta se mostrou eficaz, pois gerou informações que podem ajudar as equipes a tomarem decisões em situações imprecisas.

Possível aplicação nos IFs
Na indústria os gerentes de projetos tem, na maior parte do tempo, visões pouco claras de requisitos incertos, o que resulta em perda de tempo, atraso, e falha no projeto.

No estudo de Govindarajan (2015) a Lógica Difusa é utilizada para auxilia os gerentes de projetos a tomar decisões quando em situações incertas.

No âmbito dos IFs, é possível fazer uso de técnicas de Inteligência Artificial para o auxílio em todas as fases dos projetos, principalmente no tocante às **estimativas** e às **métricas**, técnicas que ajudam a ter uma melhor noção das datas de entrega e dos recursos necessários dos projetos.

BOA PRÁTICA #07 - USAR PROTÓTIPOS

Nome da boa prática

Usar protótipos

Objetivo

Usa-se os protótipos para encurtar a distância entre o que o cliente deseja e o que a equipe de desenvolvimento entendeu que o cliente deseja. Os protótipos podem ser fotografias de telas do sistema a ser desenvolvido ou até mesmo arquivos executáveis apenas com a interface pronta sem processamento.

Sugere-se ainda o uso de Mockups, que são desenhos da tela do sistema para discussão entre os interessados.

Também conhecida como

Usar artefatos de software.

FC relacionado(s)

Engenharia de requisitos mal feita

Consequências

Expectativas mais realistas do cliente e da equipe.

Boa(s) Prática(s) relacionada(s)

BOA PRÁTICA #01 - USAR TÉCNICA DE ENGENHARIA DE REQUISITOS

Dica: Existem váriоas ferramentas online e gratuitas para elaboração de mockups (nome dado a protótipos na área de design).

Principais telas do sistema

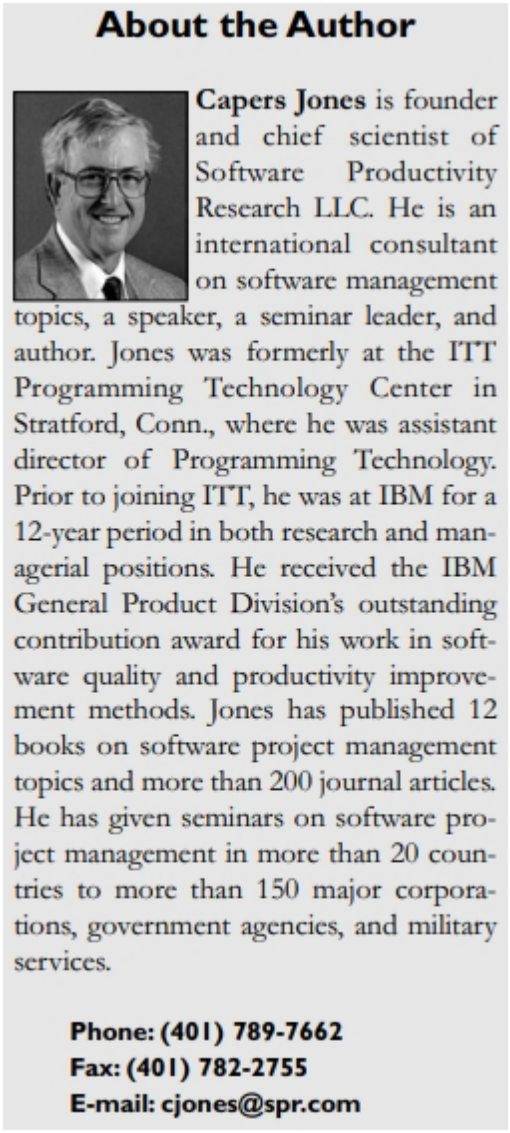
Autor(es)
Jones (2004)

Proposta
Assim como em **BOA PRÁTICA #03 - USAR FERRAMENTAS DE GERENCIAMENTO DE**

PROJETOS, Jones (2004, Fig. 1) apresenta sucintamente uma proposta em seu estudo.

Segundo o autor, usar protótipos de software é útil para minimizar os impactos de possíveis mudanças de requisitos.

Sugere-se a apresentação aos stakeholders das principais telas do sistema, e quais são as entradas possíveis e as saídas esperadas do fluxo do software. Assim, os usuários poderão ter uma experiência mais próxima à real, semelhantes ao sistema finalizado.



About the Author

Capers Jones is founder and chief scientist of Software Productivity Research LLC. He is an international consultant on software management topics, a speaker, a seminar leader, and author. Jones was formerly at the ITT Programming Technology Center in Stratford, Conn., where he was assistant director of Programming Technology. Prior to joining ITT, he was at IBM for a 12-year period in both research and managerial positions. He received the IBM General Product Division's outstanding contribution award for his work in software quality and productivity improvement methods. Jones has published 12 books on software project management topics and more than 200 journal articles. He has given seminars on software project management in more than 20 countries to more than 150 major corporations, government agencies, and military services.

Phone: (401) 789-7662
Fax: (401) 782-2755
E-mail: cjones@spr.com

Fig. 1 - Sobre Jones (Jones, 2004)

Principais recomendações
Utilização frequente de protótipos de software.

Exemplo prático do autor
Pode ser acessado aqui.

Possível aplicação nos IFs
Pode-se seguir à risca o proposto pelo autor para se aplicar os **protótipos** de software no

âmbito dos IFs.

Fotografias de tela dão uma ideia inicial aos usuários como o cliente irá se portar.

Também é interessante a **construção de uma interface** em linguagens de *frontend* (ou seja, construir apenas a "aparência" com HTML e CSS) para se poder discutir o layout com os interessados.

Definições

Brainstorming: técnica de dinâmica de grupo que tem como finalidade a descoberta de novas ideias explorando a criatividade da equipe.

FC: Fator Crítico - fator que interfere de alguma forma no projeto.

Framework: conjunto de práticas ou técnicas elaboradas já testadas em outros contextos.

Git: sistema de controle de versões de códigos-fonte.

IDE: Ambiente Integrado de Desenvolvimento, é a ferramenta utilizada por programadores para codificação.

IFs: Institutos Federais.

Loc, kLoc e APF: medidas para tamanho de códigos-fonte. *Loc* é *Lines Of Code* (linhas de código), *kLoc* é *Kilo Lines Of Code* (mil linhas de código) e APF significa Análise de Pontos por Função.

Manifesto Ágil: conjunto de princípios que norteiam o desenvolvimento ágil de software.

Método Ágil: disciplina baseada no Manifesto Ágil que estuda a entrega de incrementos de software ao cliente final de forma sistemática e periódica.

Método Cascata: método de desenvolvimento de software prega que uma etapa só pode começar a ser desenvolvida quando a anterior estiver finalizada. Atualmente está em desuso.

Quadro Kanban: artefato visual que controla os status das tarefas que estão sendo executadas, geralmente divididas em “A fazer”, “Sendo feito” e “Finalizado”.

Scrum: método baseado no Manifesto Ágil para gestão, planejamento e organização de projetos de desenvolvimento de software.

Stakeholder: qualquer pessoa envolvida de alguma forma em um projeto de software.

XP: outro exemplo de método baseado no Manifesto Ágil para gestão de projetos de desenvolvimento de software.

Download

Você pode fazer o download do arquivo PDF deste Manual **clicando aqui**.

Contato

Entre em contato comigo através dos dados abaixo

Email: lbm5@cin.ufpe.br

Twitter: @baciotti

GitHub: http://github.com/bacciotti

Cursos: Baciotti Cursos

Podcast: BaciottiCast

Referências

CANCIAN, M. H. **Uma proposta de guia de referência para provedores de software como um serviço.** 2009. Dissertação (Mestrado em Engenharia de Automação e Sistemas) – Universidade Federal de Santa Catarina, 2009.

GOVINDARAJAN, A., A soft computing framework to evaluate the efficacy of software project management. *In*: **2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO)**, 2015, pp. 1-6.

GOVINDARAJU, R.; HARIADI, R. A. R.; SIDIQ, A. Z. ERP assimilation and benefit realization: Analyzing the influence of leader characteristics. *In:* **2015 International Conference on Information Technology Systems and Innovation (ICITSI)**, 2015, pp. 1-6.

GUPTA, V.; CHAUHAN, D. S.; DUTTA, K. **Incremental development & revolutions of e-learning software systems in education sectors:** A case study approach Human-centric Computing and Information Science, 2013.

HONG N.; YOO, J.; CHA, S. Customization of Scrum Methodology for Outsourced E-Commerce Projects. *In:* **2010 Asia Pacific Software Engineering Conference**, 2010, pp. 310-315.

JONES, C. **Software Project Management Practices: Failure Versus Success**. 2004.

NAZIR S. et al., "A process improvement in requirement verification and validation using ontology". *In:* **Asia-Pacific World Congress on Computer Science and Engineering**, 2014, pp. 1-8.

PRADHAN, P.; RISHIWAL, V.; AGARWAL, A. A survey on effectiveness of tool based software project planning. *In:* **2016 2nd International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Fall)**, 2016, pp. 1-8.

SHAHZAD, B.; MATHKOUR, H. I. Role of Effective Facilitator: FAST Made Effective. *In:* **2009 International Conference on Management and Service Science**, 2009, pp. 1-4.

SULIMAN, S. M. A.; KADODA, G. Factors that influence software project cost and schedule estimation. *In:* **2017 Sudan Conference on Computer Science and Information Technology (SCCSIT)**, 2017, pp. 1-9.