OPEN UP - PROCESSO UNIFICADO ABERTO

ESTE BLOG É DEDICADO A APRESENTAR OS PRINCIPAIS CONCEITOS DO PROCESSO UNIFICADO ABERTO - OPEN UP, UMA DAS METODOLOGIAS ÁGEIS QUE FAZEM PARTE DA ECLIPSE PROCESS FRAMEWORK, DA COMUNIDADE ECLIPSE. ESPERO QUE POSSA TE AJUDAR.

3. Princípios do Open UP

O Open UP possui quatro princípios fundamentais que compõem os pilares de seu processo, o qual define as intenções por trás da metodologia. Este tratado gera uma pedra fundamental para a interpretação dos conceitos de Papéis, Produtos de Trabalho e as Tarefas que os geram.

- **Balancear as prioridades concorrentes para maximizar os valores dos Stakeholders**
  Promover práticas que permitam à equipe de desenvolvimento e aos Stakeholders desenvolver uma solução que contrabalancei-e todas as necessidades dos Stakeholders e que esteja de acordo com as restrições propostas no projeto;

- **Colaborar para alinhar os interesses e compartilhar os conhecimentos**
  Promover as práticas que promovam um ambiente saudável de desenvolvimento em equipe, possibilitando a colaboração e possibilitando a compreensão e concordância sobre os principais requisitos que definem o sistema.

- **Focar inicialmente na arquitetura para minimizar**

riscos e organizar o desenvolvimento
Promover as práticas que permitam à equipe de desenvolvimento focar suas ações na arquitetura, buscando minimizar os riscos e organizar o processo de desenvolvimento da solução proposta.

- **Envolver os *Stakeholders* para obter contínuo *feedback* do desenvolvimento**
  Promover práticas que permitam à equipe de desenvolvimento obter feedback contínuo dos stakeholders sobre a solução proposta e demonstrar o incremento de seu valor.

1º Princípio Básico
Balancear as prioridades Concorrentes para Maximizar os Valores dos *Stakeholders*

Raramente, um sistema (ou uma solução qualquer) é vista como um todo por todos os Stakeholders, ou seja, cada Stakeholder estará basicamente com vistas sobre a porção da solução que atende sua demanda. É de extrema importância que todos os Stakeholders tenham consciência de e concordância sobre o problema e a solução proposta como um todo. Para tanto, três fatores devem ser considerados:

- Problema a ser resolvido;
- As restrições inerentes à equipe de desenvolvimento (custo, recursos, habilidades, etc.);
- As restriçoes inerentes à solução proposta.

O grande desafio para as equipes de desenvolvimento é criar uma solução que maximize todos os valores (necessidades) propostos pelos Stakeholders. Isso envolve o balanceamento do custo benefício entre as características desejadas para a solução proposta e as subsequentes decisões referentes à arquitetura do sistema.

A definição do ponto de equilíbrio é um desafio constante,

considerando que o mesmo é dinâmico. Esse dinamismo varia em conjunto com as características e os requisitos da solução proposta. É importante lembrar do conceito fundamental que prega que os Requisitos são dinâmicos e podem variar de acordo com o decorrer do processo de desenvolvimento. A maior parte da variação de um requisito é resultado do refinamento do requisito, inerente a qualquer processo de desenvolvimento iterativo incremental. [RUP2007]

As mudanças ocorridas nas características e nos requisitos implicam também em mudanças nos riscos que implicam sobre o processo de desenvolvimento. Os membros da equipe de desenvolvimento e os Stakeholders devem estar preparados para reavaliar seus compromissos, reavaliar suas expectativas e ajustar seus planejamentos de acordo com o andamento do projeto.

Boas Práticas

Algumas práticas podem ser utilizadas para facilitar as ações acima descritas.

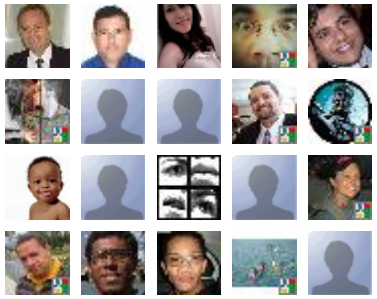
Conheça sua Audiência

Mantenha-se próximo aos seus Stakeholders. Você não poderá negociar adequadamente com eles se você não os conhece e, principalmente, se você não sabe exatamente do que eles precisam. É necessário destacar que um bom analista de negócios e requisitos, deve, além de tudo, detectar não somente o que os Stakeholders "querem", mas, principalmente, o que eles "precisam".

Sempre que possível, mantenha aberto um canal de comunicação entre sua equipe de desenvolvimento e os seus Stakeholders.

Separe o Problema da Solução

Frequentemente, é comum a equipe de desenvolvimento mergulhar profundamente no desenvolvimento de uma



solução sem conhecer com detalhes os problemas analisados. Em geral, aprendemos na escola apenas como resolver "problemas" sendo que o ideal é aprendermos primeiro a como "identificar" problemas. O foco na resolução dos problemas pode limitar a visão e compreensão dos mesmos pela equipe de analistas.

Garantir a real compreensão dos problemas facilita a busca por soluções alternativas para os mesmos.

Documente e Compartilhe a Compreensão do Domínio da Solução

Embora na Open UP aconselha-se o uso responsável de artefatos, é importante que o conjunto de artefatos que permitem a compreensão e a concordância sobre o domínio (escopo) da solução proposta sejam devidamente compartilhados entre todos os Stakeholders. Embora nem sempre seja possível, é importante que a equipe de desenvolvimento mantenha contato direto e initerrupto com os Stakeholders, gerando uma maior aproximação e maior confiança entre as partes.

Porém, é preciso percepção por parte dos analistas e desenvolvedores para que os mesmos não se tornem incômodos para os Stakeholders.

Os artefatos chaves para essa proposta são: Documento de Visão, Modelo de Caso de Uso, Especificação Suplementar e Glossário.

Utilize Modelo de Casos de Uso Para Especificar os Requisitos Funcionais

Muitas equipes de desenvolvimento ainda especificam seus requisitos funcionais através de listas de declarações. Embora fáceis de construir, essas listas podem dificultar o processo de compreensão dos requisitos pelos Stakeholders.

A aplicação de Modelos de Caso de Uso busca capturar os requisitos funcionais em um documento claro, escrito em uma linguagem própria do ambiente analisado. Por evitar tecnicismos e por falar a linguagem dos Stakeholders em geral, o Modelo de Caso de Uso torna-se uma importante referência para uma concordância de todos os envolvidos sobre os requisitos do sistema. [LARMAN2007]

Os requisitos não funcionais (Usabilidade, Confiabilidade, Performance, Suportabilidade) podem ser documentados da forma tradicional, utilizando como base o documento de Especificação Suplementar de Requisitos.

Estabelecer e Manter a Concordância em Relação às Prioridades

A decisão incorreta sobre o que desenvolver nos próximos passos pode resultar em um esforço indevido por parte da equipe de desenvolvimento, na entrega de funcionalidades inadequadas e principalmente, na identificação tardia de problemas, resultando no atraso e possível cancelamento do projeto).

Priorizar requisitos para implementação atráves de atividades desenvolvidas em parceria com os Stakeholders pode garantir o sucesso do projeto. Tomar decisões que levem à entrega de produtos que incrementem o valor da solução e que diminua o risco do projeto não é tão simples, mas é uma tarefa que principalmente, o Gerente de Projetos deve executar com maestria.

Gerenciar o Escopo do Sistema

Mudanças são inevitáveis. Como já afirmado, requisitos são dinâmicos, e, por consequência, o escopo do sistema também é dinâmico. Porém, isso não implica que mudanças nos requisitos e no escopo possam ser geradas a revelia. O excesso de mudanças pode resultar em um sistema inchado e deficiente, dificultando o processo de concordância dos Stakeholders

sobre os requisitos definidos.

O gerenciamento das mudanças solicitadas pode facilitar o processo de compreensão e concordância dos Stakeholders sobre os requisitos especificados.

Saiba Quando Parar

Extrapolar os limites do escopo definido para a solução proposta pode levar não só a perda de recursos por parte da equipe de desenvolvimento, mas também, à uma sobrecarga na complexidade da ferramenta entregue.

É importante parar o processo de desenvolvimento quando todos os requisitos foram atingidos, bem como, a qualidade proposta para o sistema. Segundo Philip Crosby, em seu livro Quality is Free: The Art of Making Quality Certain, "A qualidade é proporcional à consonância dos requisitos com o produto entregue."

2ª Princípio Básico

Colaborar para alinhar os interesses e compartilhar os conhecimentos

A prática de trabalhar colaborativamente tende a promover um ambiente saudável para o processo de desenvolvimento de software. Boas práticas colaborativas alinham os interesses dos participantes do projeto e os auxiliam a compartilhar os conhecimentos adquiridos.

O desenvolvimento de software, como qualquer outra atividade, é desempenhado (ou ao menos deve ser) por pessoas com interesses e habilidades diferentes, que devem trabalhar em sinergia para criar software de forma efetiva.

Um ambiente de trabalho saudável possibilita a colaboração efetiva entre os membros da equipe de desenvolvimento, alinhando os

interesses dos participantes do projeto (arquitetos, desenvolvedores, testadores e os demais Stakeholders) e permitem aos participantes do projeto a criar uma visão compartilhada dos conhecimentos adquiridos durante o desenvolvimento do projeto.

Boas Práticas

Algumas práticas podem ser utilizadas para facilitar as ações acima descritas.

Manter uma Visão Compartilhada dos Conhecimentos

O processo de desenvolvimento de software, por mais "enxuto" que seja, irá gerar um conjunto de documentos que formam um banco de conhecimento sobre o processo de desenvolvimento da solução em questão. Esse banco de conhecimento deve ser compartilhado entre todos os membros da equipe de desenvolvimento e entre os demais Stakeholders do projeto.

O uso da documentação permite o o alinhamento da compreensão do projeto e dos interesses de todos os Stakeholders do projeto. É importante que todos os participantes do processo de desenvolvimento sejam proativos e compartilhem com todos cada novo documento que compõe o banco de conhecimentos.

Para manter os alinhamentos devidos entre os membros da equipe de desenvolvimento é importante utilizar:

- Produtos de Trabalho da disciplina de Requisitos para manter o alinhamento entre os Stakeholders e a equipe de desenvolvimento;
- Produtos de Trabalho da disciplina de Arquitetura para manter o alinhamento entre os desenvolvedores.

Ao final de cada iteração é importante discutir quais metas da iteração

foram atingidas e, caso alguma meta não tenha sido atingida, quais as ações que serão tomadas para resolver as pendências.

Promover um Ambiente de Autoconfiança

As pessoas que não se sentem seguras no desempenho de suas atividades terão dificuldades para expor suas idéias, tomar iniciativas ou admitir possíveis falhas ou desconhecimento de técnicas. Em ambientes onde os membros da equipe sentem-se desvalorizados, o Gerente de Projetos tem uma responsabilidade maior que é acompanhar todas as atividades de forma minuciosa. É importante também que o Gerente de Projetos tome algumas atitudes que encoragem e incentivem os funcionários, como por exemplo:

- Gerenciar por objetivos: Criar um ambiente aonde a equipe de desenvolvimento se autogerencie e o Gerente de Projetos atue como um mentor, auxiliando à equipe a atingir seus objetivos;
- Derrubar as barreiras: Trabalhar visando a quebra das barreiras culturais e intelectuais da equipe pois as mesmas podem inibir o desenvolvimento da concordância dos assuntos entre os membros participantes do projeto;
- Caminhe um quilômetro com os sapatos de alguém: Respeitar e buscar compreender as perspectivas dos diversos membros da equipe antes de criticar suas idéias
- Manter diálogos pertinentes: as pessoas tecnicistas, em geral, quando questionados tendem a responder com argumentos ou pontos de vistas que levam a atritos e estabelecem um ambiente de trabalho inóspito, onde poucas pessoas conseguem contribuir realmente em um diálogo. Encorajar um comportamento que valorize a curiosidade e a busca por soluções criativas é fundamental para o bom andamento de um projeto.

Compartilhe Responsabilidades

Existem muitas desvantagens para um membro da equipe que se isola

dos demais membros. Nesses casos, a comunicação com a equipe torna-se esporádica ou até mesmo se encerra. Em geral, as pessoas que se isolam não conseguem se alinhar com os demais membros da equipe. Nas piores situações, a confiança entre os membros é quebrada e a sinergia acaba por completo.

Enquanto um membro da equipe tem responsabilidade primária sobre um Produto de Trabalho, a responsabilidade pelo produto de trabalho deve ser compartilhada. Nada é responsabilidade exclusiva de um único membro da equipe. Os membros mais experientes da equipe devem trabalhar em conjunto com os membros menos experientes, compartilhando seus conhecimentos e responsabilidades, incrementando assim a curva de aprendizagem dessas pessoas.

Aprenda Continuadamente

Desenvolver continuadamente tanto as habilidades técnicas quanto interpessoais é um fundamento importante para todos os membros da equipe de desenvolvimento. Busque sempre aprender com os exemplos de seus colegas. Aproveite a oportunidade de ser tanto tutor quanto pupilo de seus colegas de trabalho.

Gerenciar em Torno da Arquitetura

Conforme os projetos crescem em tamanho, aumenta também a comunicação entre os membros da equipe de desenvolvimento. Enquanto todos os membros compreendem o sistema como um todo, podem focar suas ações em um ou mais subsistemas pelos quais são responsáveis. A organização da equipe em torno da arquitetura também auxilia o processo de comunicação entre os membros da equipe, provendo-a com um vocabulário comum e compartilhando modelos mentais do sistema.

Porém, é necessário tomar cuidado para que não sejam criados silos

isolados de modelos mentais, ou seja, que alguns membros passem a visualizar e compreender apenas as funcionalidades nas quais estão focados.

3º Princípio Básico
Focar primariamente na arquitetura visando minimizar os riscos e planejar o processo

Uma arquitetura evoluída auxilia a equipe de desenvolvimento a identificar o nível de complexidade do sistema, auxilia nas ações para reduzir os riscos e auxilia no processo de organização e planejamento do processo de desenvolvimento do sistema proposto.

A arquitetura de um software pode ser entendida como a representação da organização ou estrutura dos componentes significantes do sistema, bem como, a representação dos comportamentos desses componentes. A estrutura dos componentes tem como função principal apresentar a interação entre os diversos componentes, principalmente no desenvolvimento em camadas, utilizando o padrão MVC (Modelo, Visão e Controle). Os componentes são representados em granularidades distintas (pacotes, componentes e classes).

Sem uma arquitetura bem especificada, o desenvolvimento de um sistema se tornará ineficiente e com uma tendência a se tornar mal fadado. Em geral, sempre haverá dificuldades no processo de manutenção, reuso, integração em um retrabalho substancial. Uma arquitetura mal estrutura também dificulta a organização e coordenação da equipe de desenvolvimento, bem como, no processo de comunicação inerente à arquitetura.

Boas Práticas

Algumas práticas podem ser utilizadas para facilitar as ações acima

descritas.

Elabore uma Arquitetura Considerando seus Conhecimentos

Como todo projeto, o desenvolvimento de software é limitado pelos recursos disponíveis. Em geral, analistas e desenvolvedores possuem uma tendência a extrapolar o escopo do sistema proposto ou supervalorizar determinadas funcionalidades de forma desnecessária.

A criação de uma arquitetura que atenda somente e tão somente aos requisitos definidos pelos Stakeholders provêm flexibilidade e velocidade no processo de desenvolvimento. Por mais bem intencionados que estejam analistas e desenvolvedores, é importante que os mesmos não tentem especular quais serão os futuros requisitos do sistema. Existe uma diferença em uma arquitetura superdimensionada e uma arquitetura flexível e expansível.

Influência da Arquitetura Como uma Ferramenta Colaborativa

A falta de uma compreensão comum sobre as características da ferramenta proposta podem levar a indecisões e opiniões contrárias entre os desenvolvedores, o que pode levar à paralização ou encerramento do projeto.
Os desenvolvedores, nesse caso, podem ter diferentes modelos mentais sobre o sistema e possuírem metas divergentes no processo.

Criar e desenvolver a arquitetura do sistema com a intenção de alinhar os objetivos dos desenvolvedores é uma prática que deve ser adotada por todas as equipes de desenvolvimento. Uma boa arquitetura facilita a colaboração e a sinergia entre os membros da equipe, criando um vocabulário comum e beneficiando a comunicação.

Tratar a Complexidade do Sistema Incrementando o Nível de

Abstração do Projeto

Software é complexo e as pessoas costumam sentir dificuldades ao lidar com altos níveis de complexidade. Conforme um sistema evolui, mais difícil se torna para a equipe de desenvolvimento para manter uma compreensão comum de sua estrutura, considerando que é mais difícil visualizar um problema como um todo do que suas partes.

A elaboração da arquitetura prega o uso de modelos (Diagramas de Classe, Padrão MVC, Design Patterns, etc.) para incrementar o nível de abstração do projeto, focando nos elementos mais importantes como relacionamentos e padrões, evitando focar em detalhes de baixo nível do projeto.

Organize a Arquitetura em Componentes Coesos e Fracamente Acoplados

Componentes fortemente acoplados tornam o sistema complexo e frágil. Em geral, softwares são caros para serem produzidos e, quanto maior a capacidade de reuso de seus componentes, menor o esforço necessário para criar novos sistemas.

É importante organizar a arquitetura de forma a minimizar a o acoplamento e maximizar a coesão dos componentes. Essa ação permite aumentar a compreensão e incrementar a flexibilidade e o reuso.

Reutilize Componentes Existentes

Aqui podemos utilizar a velha máxima: "Não queira reinventar a roda". Se você tem um componente estável, para que reescrevê-lo? É comum os desenvolvedores relutarem em reutilizar componentes, por mais estáveis que os mesmos possam estar, principalmente se os mesmos não atendem em 100% suas necessidades.

Já é mais do que posto e sabido que o reuso de componentes reduz o custo e o cronograma do processo de desenvolvimento de software. Algumas empresas com um processo maduro conseguem reaproveitar uma taxa muito alta de seus componentes, alterando somente quando necessário, questões de regras de negócios.

4º Princípio Básico
Envolver os *Stakeholders* para obter contínuo feedback do desenvolvimento

Muitas vezes, é muito difícil, quiçá impossível, conhecer todas as reais necessidades dos Stakeholders de um projeto nos primeiros momentos de desenvolvimento do projeto. Por consequência, muitos riscos inerentes à essas necessidades são ignorados durante o processo de desenvolvimento. É importante que a equipe de desenvolvimento promova práticas que permitam à equipe demonstrar o a agregação incremental de valor ao produto que está sendo gerado e obter feedback contínuo dos *Stakeholders*, evitando ao máximo que necessidades deixem de ser atendidas.

O principal intento desse princípio é obter feedback contínuo dos *Stakeholders* e assim, incrementar o valor agregado do produto e incrementar o próprio processo de desenvolvimento de software da equipe. Quando se oferece uma estrutura estável e confiável de desenvolvimento que possibilite o feedback contínuo, as mudanças ocorrem de forma menos traumática e se acomodam mais facilmente.

O ataque prematuro aos riscos do projeto e o constante trabalho em cima do feedback dos Stakeholders tendem a gerar um resultado final com maior valor agregado. Porém, é importante destacar que apenas os feedbacks não são suficientes para um resultado positivo. Se as ações corretas não forem tomadas em relação aos problemas e riscos que surgem no decorrer do desenvolvimento, então o resultado final não

apresentará a qualidade necessária.

Boas Práticas

Algumas práticas podem ser utilizadas para facilitar as ações acima descritas.

Desenvolva seus Projetos de Forma Iterativa

Desenvolver sistemas de forma linear, passo a passo é difícil e torna o processo mais complexo para incorporar mudanças e acrescentar novos valores. Além disso, o desenvolvimento lienar pode dificultar a descoberta de riscos, considerando que o processo de integração e o lançamento do produto é feito tardiamente no ciclo de vida do processo.

Dividir seu processo de desenvolvimento em pequenas etapas cíclicas (iterações) habilita sua equipe a incrementar a capacidade de entrega de produtos de trabalho continuadamente, como executáveis (builds) que vão sendo integrados e testados. Dessa forma, o custo e a quantidade de retrabalho exigidos por conta de solicitações de mudanças caem drasticamente, bem como, cresce a confiança e estabilidade do produto a ser entregue.

O desenvolvimento iterativo permite melhorias contínuas no processo de desenvolvimento de software durante todo o ciclo de vida de projeto.

Foque as Iterações de Forma a Atingir os Próximos Marcos

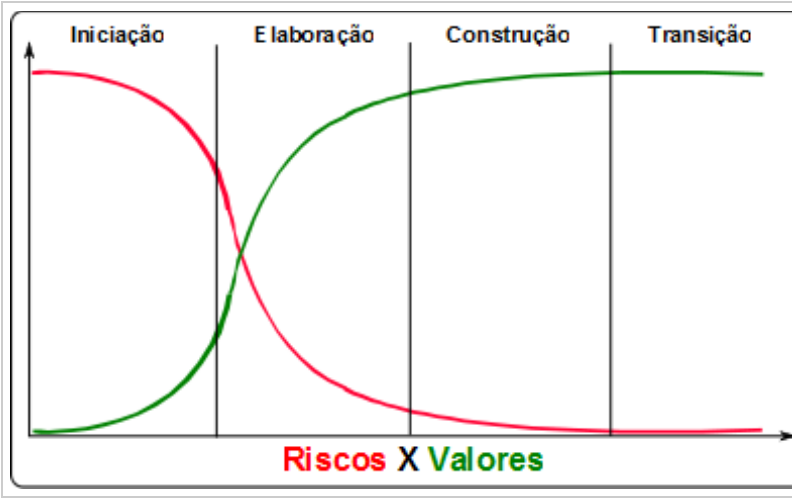
Uma equipe pode facilmente achar que está realizando progressos durante o desenvolvimento de um projeto, quando na verdade riscos e questões não resolvidas vão se

acumulando. Foque suas ações de forma a tornar visível todos os pontos fundamentais do projeto em desenvolvimento.
Use e abuse do conceito das fases do Processo Unificado e definindo claramente os marcos de cada fase. O foco de cada iteração de uma fase deve ser direcionado para atingir o marco de sua fase.

Gerencie Riscos
Adiar as decisões complexas e arriscadas para o final de um projeto aumentam significativamente os riscos de falha do mesmo. Esse adiamento pode levar ao investimento em tecnologias incorretas, em uma arquitetura instável ou em um conjunto de requisitos que não atendam às necessidades dos Stakeholders.

Ataque os riscos logo no início do processo de desenvolvimento para que os riscos não te ataquem. Priorize o ataque às funcionalidades que oferecem os maiores riscos e os refine constantemente. Determinar o foco das iterações baseado nos riscos.

O ataque prematuro aos riscos agrega valor e estabilidade ao projeto em desenvolvimento.



Gerencie Mudanças

Mudanças são inevitáveis e sempre devem ir ao encontro das necessidades dos Stakeholders. Mudanças podem ocorrer por necessidades de correções inerentes ao processo de desenvolvimento ou pela característica dinâmica dos requisitos.

É importante destacar que, quanto mais cedo em relação ao Ciclo de Vida de Projeto a necessidade de mudança é detectada, menos onerosa será sua manutenção.

Os Stakeholders também devem ter plena consciência de que mudanças geram custos. Mais uma vez, destaca-se a importância do processo de análise de requisitos. Sempre que necessário, é importante documentar as mudanças ocorridas. Apesar de tudo, o Open UP não indica como disciplina a Gerência e Configuração de Mudanças. Em projetos menos formais, discussões para obter concordância com os Stakeholders devem ser suficientes.

Mensure o Progresso do Projeto de Forma Objetiva

É responsabilidade do Gerente de Projetos saber avaliar o progresso do desenvolvimento do projeto em andamento. Uma má avaliação não conseguirá indicar se o projeto obterá sucesso ou falhará. Mudanças e incertezas tornam o processo de avaliação do desenvolvimento subjetivo. Uma forma objetiva de se avaliar o progresso de um projeto é avaliar a quantidade, qualidade e estabilidade dos Produtos de Trabalho que são entregues. Outras métricas mais simples podem ser adotadas para avaliar o progresso do desenvolvimento de um projeto como por exemplo: requisitos que foram implementados e validados; quantificação dos defeitos corrigidos; quantificação dos defeitos não corrigidos.

Avaliações mais objetivas podem ser realizados no Micro Incremento de Revisão de Iteração.

Referências

[RUP2007] Rational Unified Process; Direitos Autorais (C) IBM Corporation 2000, 2007.

[LARMAN2007] Larman, Craig; **Utilizando UML e Padrões**; Editora Bookman; 3ª Edição; São Paulo; 2007.

Página inicial

Assinar: Postagens (Atom)