Universidade Federal do Ceará
Campus de Quixadá

Disciplina	Gerência de Projeto de Software	**Semestre**	
Professor a	Diana Braga		
Aluno	Guilherme Moretti e Vitória Ashiley	**Matrícula**	

Questão 1 – Julgue os itens a seguir, **justificando os itens falsos**, se houver.

a) (F) Para que o gerenciamento de um portfólio de projetos seja efetivo, é necessário que os projetos ou programas que o compõem sejam inter-relacionados.
JUSTIFICATIVA: Um portfólio de projetos pode conter atividades que não sejam necessariamente inter-relacionadas, mas que são gerenciadas com uma mesma estratégia ou objetivo organizacional.

b) (V) Embora o gerente de um projeto seja responsável pelo seu orçamento e pelo cumprimento dos prazos estabelecidos, a elaboração dos demonstrativos financeiros e a garantia da qualidade do produto final são de responsabilidade de áreas específicas da organização.

c) (F) O planejamento do escopo é a etapa de levantamento e análise dos riscos inerentes ao projeto.
JUSTIFICATIVA: O planejamento do escopo refere-se à definição e documentação dos objetivos, entregas e limites do projeto. A análise de riscos é uma atividade separada que geralmente ocorre após o planejamento do escopo.

d) (V) O controle do projeto deve ocorrer também durante a sua execução, de forma que o gerente possa administrar as variações em relação aos planos e garantir a realização dos objetivos

e) (F) Como os projetos têm prazo definido, o gerente deve estabelecer para essas atividades um cronograma rígido, sem margem para negociação dos prazos de conclusão das etapas intermediárias para não ocorrer atrasos.
JUSTIFICATIVA: Mesmo com o projeto tendo um prazo definido, o gerente deve ter uma flexibilização para atividades do cronograma, permitindo adaptação em decorrência de problemas que possam surgir e afins.

Questão 2 – Reescreva os objetivos do projeto da esquerda na coluna à direita, tornando-os objetivos SMART.
S: specific (específico)
M: mensurável
A: atingível
R: relevante
T: temporal

Exemplo de projeto	Objetivo SMART
Quero emagrecer	Perder um total de 0.5kg durante 1 mês, seguindo a dieta
Aumentar o lucro da minha empresa	No periodo de 5 anos, aumentar o LL em 50% melhorando o CAGR

<algum projeto pessoal seu>	Quero terminar de ler o Livro "Criação de conhecimento na empresa" até o final do ano. (Gui) Quero conseguir fazer todos os cursos da cadeira de persistência e ficar com notas acima da média em todas as primeiras provas das cadeiras desse semestre. (Ash)

Questão 3 – Prática 1

Prática 1 – Análise de Relato
Gerenciamento de Projetos no Desenvolvimento de Software

Objetivo

Compreender quais fatores de fracasso e sucesso de projetos estão relacionados com uma abordagem correta de Gerenciamento de Projetos

Instruções: Ler material indicado e responder as questões.

Introdução

O relato a ser analisado são cenários elaborado por um profissional de larga experiência na área. O relato é uma anedota e exagera o cenário de fracasso de projetos comumente encontrado no mercado. Captar os fatos e eventos do cotidiano e relacionar a boas práticas e metodologias conhecidas permite ao profissional identificar soluções adequadas.

Leia o relato

No artigo "O Ciclo Vicioso do Software Retranqueiro", Klaus Wuestfeld apresenta um cenário caricato (porém real) de como organizações e profissionais podem entrar em um ciclo improdutivo e dispendioso de desenvolvimento de software, que se torna cada vez mais burocrático pelo próprio insucesso de ações anteriores.

Responda as questões

1. O que caracteriza o "software retranqueiro" mencionado no artigo.

   R: Projeto burocrático, alta complexidade, baixa interação com o cliente e qualidade comprometida devido a necessidade de rápida entrega.

2. Quais são os efeitos de um ciclo vicioso de desenvolvimento sobre a motivação e o desempenho de um time?

   R: Motivação prejudicada, desempenho comprometido e estresse.

3. Quais sinais podem indicar que um projeto de software está caindo no ciclo vicioso descrito por Wuestfeld?

   R: Atrasos recorrentes, cronogramas inflados, resistência a mudanças, dificuldade em estipular previsões de entregas, diminuição da qualidade e complexidade.

4. “Mas será que esse círculo vicioso é realmente inevitável? Será ele um fenômeno da natureza? Ou será que podemos acabar com isso?”

   R: O círculo vicioso do software retranqueiro não é inevitável. Ele pode ser interrompido com práticas ágeis, comunicação contínua com o cliente, adaptação constante e foco na qualidade. Isso exige vontade de abandonar processos ultrapassados e abraçar a flexibilidade e inovação

   De que forma um gerenciamento de projetos eficaz pode ajudar a evitar o ciclo vicioso do software retranqueiro?

   R: Encorajar a comunicação contínua e colaborativa, adotar metodologias ágeis como XP, Scrum, foco na qualidade, entregas menores e incrementais, redução de burocracia;

ANEXO

O Círculo Vicioso do Software Retranqueiro
por Klaus Wuestefeld

Projetos de desenvolvimento de software fracassam. Fracassam notoriamente e fracassam feio. De acordo com o Gartner Group, 80% a 90% dos projetos de TI fracassam, sendo cancelados ou entregues com prazo ou com o orçamento estourados.

Mas qual é a importância disso, afinal? Por que as organizações dependem tanto do desenvolvimento de software próprio? Por que não podem simplesmente comprar todo seu software pronto?

É claro que não estamos falando de commodities como sistemas operacionais ou software para automação de escritório. Assim como a escolha da marca de clips, cada vez menos importa qual editor de textos ou qual ambiente gráfico vai-se usar.

A maioria das empresas depende também dos famosos "pacotões": folha de pagamento, controle de estoque, contas a pagar, etc. Embora eles ainda tragam muitos problemas de implantação e integração, principalmente em grandes organizações, também não é a escolha do "pacotão" que vai GARANTIR vantagem competitiva a essa ou aquela empresa.

Onde é que a coisa pega, então? Qual é o tipo de software que realmente faz a diferença?

Organizações como restaurantes, grupos de teatro, clínicas médicas e bandas musicais não ligam muito para software. No caso delas, o que importa mesmo é a experiência, o conhecimento tácito de seus integrantes.

Para grande maioria das empresas, porém, informação e conhecimento explícito estão, cada vez mais, definindo o resultado do jogo. É justamente nos sistemas centrais dessas organizações, portanto, que agilidade e qualidade no desenvolvimento de software fazem a diferença entre o sucesso e a morte.

Software é a única coisa que consegue representar conhecimento explícito e, ao mesmo tempo, botá-lo para trabalhar em favor da empresa, em vez de ficar fazendo volume numa prateleira ou num lindo repositório de documentos.

Software é conhecimento explícito por excelência.

Jamais conseguirá uma empresa superar a outra, se comprar exatamente o mesmo software de prateleira para seu sistema central. O máximo que conseguirá atingir com isso é a mediocridade e, não sendo capaz de alterar esse mesmo software para capturar seus conhecimentos adquiridos frente ao dinamismo do mercado, ficar estagnado é o que lhe resta.

Mas por que isso acontece? Por que, mesmo sendo cada vez mais importantes para as organizações, os projetos de software continuam fracassando? Quais são as causas dessa situação vergonhosa para a minha profissão, como desenvolvedor de software?

Não existe causa única. Existe um círculo vicioso de causas que se exacerbam mutuamente.

1) O fracasso da grande maioria dos projetos de desenvolvimento de software causa grandes prejuízos aos clientes.

2) Os repetidos prejuízos alimentam a desconfiança dos clientes em relação ao desenvolvimento e aos desenvolvedores de software.

3) A desconfiança faz com que os clientes peçam orçamentos de preço fechado e, em muitos casos, com pesadas multas.

4) Para fazer orçamentos de preço fechado, os desenvolvedores têm que fazer estimativas precisas.

5) Para fazer estimativas precisas, os desenvolvedores necessitam de requisitos muito bem definidos.

6) Para obter requisitos muito bem definidos, cliente e desenvolvedores passam boa parte do tempo de um projeto, mesmo antes de seu início, levantando e detalhando requisitos. Trata-se de um investimento pesado para um projeto que nem se sabe ao certo se vai ser implementado.

7) Para minimizar esse investimento, que muitas vezes não é pago pelo cliente, os desenvolvedores comprometem a qualidade e abrangência do levantamento de requisitos (ver 10).

8) Depois de trabalhar tanto na produção de um lindo documento de requisitos, o cliente não quer mais saber de falar com o desenvolvedor até a entrega do sistema final.

9) O desenvolvedor, por sua vez, dá graças por não ter que aturar o cliente que só serve para atrapalhar, mudar de idéia e perguntar como anda o projeto.

10) Evitando o contato com o cliente e com pressa, o desenvolvedor passa a supor uma série de coisas a respeito de como o sistema deveria funcionar. Por mais pesado que tenha sido o investimento na fase de requisitos, eles sempre apresentam inconsistências, ambiguidades e omissões. Se não fosse assim, bastaria compilar o documento de requisitos e o sistema estaria pronto.

11) A maioria das suposições dos desenvolvedores está correta. A minoria, porém, está errada e causa surpresas desagradáveis para o cliente (ver 25).

12) As surpresas desagradáveis causam retrabalho (ver 24) para o desenvolvedor quando da entrega do produto.

13) Com pressa, a cada funcionalidade implementada, o desenvolvedor faz apenas alguns testezinhos manuais básicos. Ele não tem tempo para escrever código de teste automatizado para testar seu sistema. A qualidade do software fica prejudicada (ver 23).

14) O desenvolvedor é proibido de melhorar código que já está funcionando para eliminar as duplicações com novas funcionalidades. "Se tá funcionando, então não mexe!". A complexidade do sistema passa a crescer como uma colcha de retalhos com duplicações, dependências a torto e direito e acoplamento forte entre os módulos do sistema.

15) A complexidade faz as modificações no sistema serem cada vez mais demoradas e propensas a erros (ver 23).

16) A demora e o alto custo de modificações atrasam o projeto (ver 10, 13, 21, 25).

17) Os atrasos causam os ciclos de desenvolvimento a serem cada vez maiores.

18) Quanto maior o tempo de cada ciclo de desenvolvimento, piores as surpresas, menos freqüente a chance do cliente requisitar novas funcionalidades e maior o tempo de espera.

19) Quanto maior o tempo de espera, mais funcionalidades o cliente tem que requisitar nas raras chances que consegue: "Quero tudo, o mais configurável, flexível e genérico possível."

20) Exagero de requisitos torna o software mais complexo (ver 15) e aumenta o tempo de desenvolvimento (ver 16).

21) Projetos atrasados aceleram ou simplesmente ignoram a fase de testes.

22) Sem a fase de testes a qualidade que já estava ruim vai pro buraco.

23) Os defeitos causam retrabalho.

24) O retrabalho causa atraso (ver 10, 13, 21, 17) e ainda mais defeitos (ver 23).

25) As surpresas desagradáveis, os defeitos e os atrasos determinam o fracasso do projeto (ver 1).

Desenvolvedores experientes podem facilmente duplicar o tamanho dessa lista de sintomas do círculo vicioso. Todas essas coisas são tão corriqueiras na vida dos desenvolvedores que a grande maioria encara essa situação lastimável como sendo normal, inevitável.

A única forma de um projeto complexo nessas condições ser entregue dentro do prazo e dentro do orçamento é com pessoas excepcionalmente talentosas, experientes e motivadas, ou simplesmente com um prazo e com orçamento absurdamente altos.

Quando não estão preocupados com diagramas mais imponentes, cronogramas mais vistosos e coisas do gênero, os processos de desenvolvimento tradicionais trabalham alguns dos sintomas do círculo vicioso, buscando técnicas melhores para conseguir: levantamento de requisitos mais completos, estimativas mais precisas, modelos mais rastreáveis, etc.

Infelizmente, como em qualquer círculo vicioso, atacar os sintomas é perda de tempo, paliativo no melhor caso. Ainda por cima, para a maioria dos projetos doentes, o mero peso burocrático dos processos tradicionais, por si só, já representa uma quimioterapia por demais brutal. Não é à toa, portanto, que cansamos de ver até desenvolvedores sérios e responsáveis largando o processo, a quimioterapia, e preferindo encarar na raça a doença.

Esses desenvolvedores, cujo talento, como já vimos, é a única chance de sucesso desses projetos, ainda são condenados pelos feitores nas "fábricas de software" (triste metáfora), por não estarem atuando como deveriam: como "máquinas" burras seguindo o processo pseudo-definido.

A influência mais perniciosa nesse contexto todo, porém, ainda são as empresas fornecedoras de ferramentas para desenvolvimento de software e organizações certificadoras de processos. Para vender seu peixe, elas têm interesse em que o processo de desenvolvimento adotado pelo mercado seja o mais complexo e burocrático possível.

Mas será que esse círculo vicioso é realmente inevitável? Será ele um fenômeno da natureza? Ou será que podemos acabar com isso?

Os sucessos alcançados por equipes ágeis vem causando comoção justamente neste mercado cansado das metodologias retranqueiras tradicionais e do notório fracasso de seus projetos. O maior evento nacional de engenharia de software, até 2010, foi o Agile Brasil, onde tive a oportunidade de fazer a palestra de encerramento.

XP e as demais metodologias ágeis simplesmente elevam a um novo patamar o significado de "qualidade" e "agilidade" em desenvolvimento de software.

Sobre o autor:

Pioneiro de Extreme Programming (XP) no Brasil, Klaus Wuestefeld é autor do manifesto da computação soberana e do Prevayler, a camada de prevalência de objetos original. Klaus é sócio-fundador da Objective Solutions e já atuou como desenvolvedor, líder de projeto, gerente e até como cliente em projetos usando XP.

Twitter: @KlausWuestefeld