MARCOS RAMOS DA GRAÇA

TEMA:

ANÁLISE DE RISCOS E DIFICULDADES NO DESENVOLVIMENTO DE SOFTWARE

LICENCIATURA EM ENSINO DE INFORMÁTICA

ISE – 2006

MARCOS RAMOS DA GRAÇA

TEMA:

ANÁLISE DE RISCOS E DIFICULDADES NO DESENVOLVIMENTO DE SOFTWARE

TRABALHO CIENTÍFICO APRESENTADO NO ISE PARA OBTENÇÃO DO GRAU DE LICENCIATURA EM ENSINO DA INFORMÁTICA, SOB A ORIENTAÇÃO DO

ENG. LIONEL LONA SAMBÉ – MESTRE EM INFORMÁTICA

MARCOS RAMOS DA GRAÇA

TEMA.

ANÁLISE DE RISCOS E DIFICULDADES NO DESENVOLVIMENTO DE SOFTWARE

Trabalho científico apresentado ao Instituto Superior de Educação, aprovado pelos membros do júri e homologado pelo Concelho Científico, como requisito parcial à obtenção do grau de Licenciatura em ensino de Informática sob a orientaçao do eng Lionel Lona Sambe.

O Júri

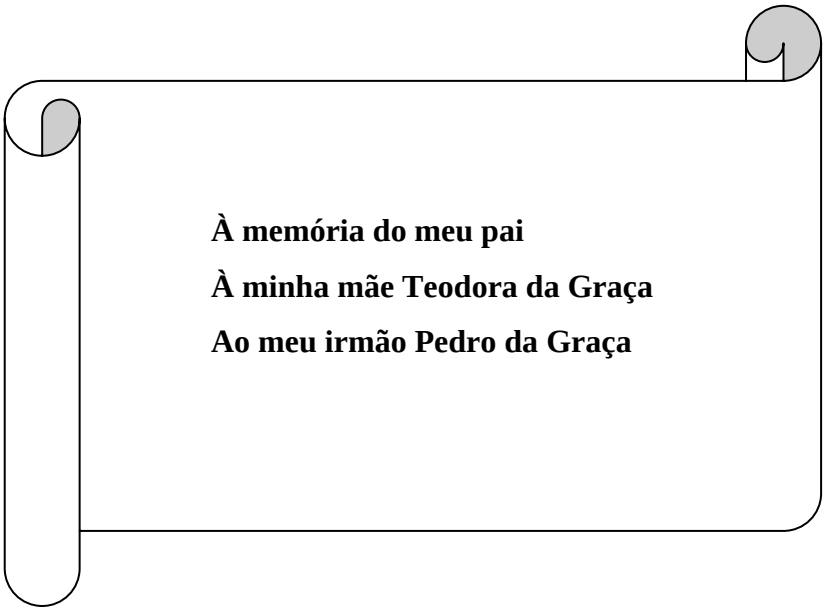
_____

_____

_____

Praia aos ____ de _____ de 2006

DEDICATÓRIA



À memória do meu pai

À minha mãe Teodora da Graça

Ao meu irmão Pedro da Graça

AGRADECIMENTOS

Um trabalho desta natureza, acarreta sempre consigo um conjunto de dívidas de gratidão. E assim sendo, são muitas as individualidades e instituições que merecem os nossos agradecimentos, por responderem calorosamente às nossas solicitações.

Um trabalho desta natureza, acarreta sempre consigo um conjunto de dívidas de gratidão. E assim sendo, são muitas as individualidades e instituições que merecem os nossos agradecimentos, por responderem calorosamente às nossas solicitações.

Em primeiro lugar queremos agradecer a **Deus Nosso Senhor e Maria** que nos proporcionou saúde e luz para a conclusão deste trabalho.

Um agradecimento muito especial vai ao **Eng. Lionel Sambé**, que desde muito cedo tomou a orientação do trabalho e dedicou-se muito para o seu sucesso.

Um obrigado à minha querida **mãe Teodora Da Graça** pelo seu empenho na minha educação.

Um obrigado aos irmãos **Pedro, António, Hirondina, Arlinda, Zuzu, Maria e Francisca** pelo apoio prestado nos momentos bons e menos bons ao longo do curso.

Um agradecimento extensivo à minha namorada, **Madalena Sousa** pela força, espírito de sacrifício e de compreensão que teve durante este ano, cuidando sozinha do nosso filho.

Um especial obrigado aos nossos amigos **Teixeira e Luis** pela correcção ortográfica deste trabalho para que pudesse ser apresentado o mais correcto possível.

Aos nossos colegas da mesma casa, **Miguel, Otelindo e Luís**, pela bonita caminhada durante todos estes anos e pelos apoios prestados.

Os nossos agradecimentos são também extensivos a **todos os colegas do curso de informática**, pelo apoio prestado e por tudo o que aprendemos juntos ao longo do curso.

A todos os verdadeiros **amigos** que estiveram sempre dispostos a nos ajudar, tanto material como moralmente. A esses, um muitíssimo obrigado!

Desenvolver software com o que de melhor existe tanto em termos de Gerenciamento quanto em termos de Engenharia é, portanto, o grande desafio das organizações de software hoje, se quiserem ser capazes de atender às exigências cada vez maiores de clientes e usuários, em termos de prazos, custos e qualidade do produto final.

Átila Belloquim

ÍNDICE

INTRODUÇÃO……..……...…1

CAPITULO I - FUNDAMENTAÇÃO TEÓRICA.....3

1.1 - Planeamento de projectos.....3

1.2 **–** Planeamento dos riscos no desenvolvimento de software.....4

1.4 - Análise e gestão de riscos.....5

CAPITULO II – MODELOS DE DESENVOLVIMENTO DE SOFTWARE.....6

2.1 – Introdução.....6

2.2 - Objectivos.....7

2.3 - Análise dos riscos associados ao modelo cascata.....8

2.4 - Análise de riscos associados ao modelo incremental.....10

2.5 - Análise de riscos associados ao Modelo de Prototipação.....12

2.6 – Análise de riscos associados ao modelo em espiral de Boehm.....14

CAPITULO III - METODOLOGIAS E GESTÃO DO RISCO NO DESENVOLVIMENTO DE SOFTWARE.....17

3.1 – Introdução.....17

3.2 - Metodologias ágeis e Metodologias tradicionais.....18

3.3 - Extreme Programming (XP)……..……19

3.4 - O Rational Unified Process……...…..23

CAPITULO IV – GESTÃO DO RISCO.....26

4.1 – Introdução…….…….26

4.2 - Objectivos da gestão de riscos.....27

4.3 – Categoria de riscos.....28

4.4 - Processo de gestão de riscos.....29

4.4.1 – Identificação de riscos.....29

4.4.2 – Análise de riscos.....32

4.4.3 - Planeamento de riscos……....37

4.4.4 - Monitoração de riscos……..……..40

CAPITULO V - CARACTERÍSTICAS DAS DIFICULDADES QUE SURGEM NO DESENVOLVIMENTO DE SOFTWARE.....43

5.1 – Introdução……..43

5.2 - Métodos de gestão…….……44

5.3 - Definição do produto a desenvolver……...……47

5.4 - Processo de desenvolvimento…….……47

5.5 – Recursos……...…..48

CAPITULO VI - CAUSAS E CONSEQUENCIAS DE RISCO E INCERTEZA.....50

6.1 - Alteração de requisitos……...……50

6.2 - Pressão excessiva sobre os prazos……..……51

6.3 - Metas não cumpridas……..……51

6.4- Derrapagens orçamentais…….……51

6.5- Baixa qualidade……..…….52

CAPITULO VII - OS RISCOS MAIS COMUNS E RISCOS MAIS PERIGOSOS NO DESENVOLVIMENTO DE SOFTWARE EM CABO VERDE…….....…….....53

CONCLUSÃO…… 56

RECOMENDAÇÕES…….59

TRABALHOS FUTUROS.....61

GLOSSÁRIO.....62

BIBLIOGRAFIA……...……..64

ANEXOS

FIGURAS

Figura 1 - Etapas do modelo cascata.....8

Figura 2 - Modelo Incremental 10

Figura 3 - Modelo de prototipação.....12

Figura 4 - Modelo espiral.....15

Figura 5 - Processo de gestão de riscos.....29

Figura 6 - Exemplo de Matriz de Exposição ao Risco.....35

Figura 7 - Indicadores das equipas de desenvolvimento de software.....55

INTRODUÇÃO

A análise de riscos constitui factor preponderante no desenvolvimento de qualquer software. Daí a importância do tema em estudo *"Análise de riscos e dificuldades no desenvolvimento de software"*, como velha ambição de contribuir nesta área que ainda não conseguiu conquistar o seu lugar no mercado Cabo-verdiano.

O objectivo fundamental deste trabalho é levar a cabo uma investigação que evidencie os efeitos negativos da má gestão dos riscos e apresentar alternativas quanto à identificação e o que fazer para evitar ou lidar com os riscos.

Sendo a prática o critério da verdade, nada melhor que aprender com os erros dos outros e fazer com que os outros aprendam com o trabalho que ora se realize, procurando sempre transmitir conhecimentos que permitam tratar os problemas dos riscos inerentes às actividades da gestão de projectos, assim como reduzir as incertezas associadas às macro-variáveis dum projecto e o impacto do sucesso ou insucesso do mesmo nos resultados da organização.

Dada a inexistência de um estudo do caso em Cabo Verde, fomos obrigados a fazer um levantamento da situação real quanto aos riscos e dificuldades que os profissionais da engenharia de software enfrentam no nosso país, utilizando diversos métodos científicos nomeadamente entrevistas e inquéritos.

Assim sendo, a primeira questão a surgir foi a seguinte: *"Sendo o sucesso do desenvolvimento de um software determinado pela análise e gestão de riscos, quais são as características das dificuldades observadas no desenvolvimento de software e quais os riscos mais comuns e perigosos de projectos de software desenvolvidos em cabo verde?"*

Para alcançar os objectivos preconizados utilizamos uma metodologia teórica-empirico que foi desenvolvida em três fases:

I. Fez-se a análise documental e entrevistas aos chefes/gestores de projectos de desenvolvimento de software;

II. Aplicação do inquérito aos chefes /gestores/ analistas de projectos de desenvolvimento de software;

III. Análise dos riscos mais comuns e mais perigosos com o objectivo de descobrir as eventuais causas e as consequências que estes podem trazer para os desenvolvedores e para o cliente.

O estudo realizado mostra que poucos dos que desenvolvem software em Cabo Verde fazem planeamento adequado do projecto e raros os que utilizam alguma metodologia de desenvolvimento de software. Hoje em dia com a popularidade da Internet, existem vários técnicos que acham que programam quando na verdade o que fazem são sistemas sem metodologia, documentação e, o pior, sem rumo.

Se programar fosse sentar no computador e começar a criar linhas de códigos não haveria necessidade das empresas de desenvolvimento de software contratarem analistas, mas sem este especialista responsável pelos estudos preliminares antes de começar a programar, de facto seria como um cego a andar num beco escuro.

Não avaliar os riscos num projecto de software ou ignora-los, significa, para muitos adiar situações que provavelmente deixarão de ser riscos e passarão a ser graves e complicados problemas.

Ao concluir este trabalho espera-se que os desenvolvedores dos projectos de softwares tirem proveito do mesmo e que sirva como valioso instrumento de orientação para gestão dos riscos de forma que num futuro próximo os projectos desenvolvidos a nível nacional possam concorrer na igualdade de oportunidade com os importados.

CAPÍTULO I-FUNDAMENTAÇÃO TEÓRICA

1.2 - PLANEAMENTO DE PROJECTOS

Muitas bibliografias mostram que a maioria dos projectos de desenvolvimento de sistemas fracassaram devido à falta de um planeamento adequado. Planeamento adequado significa ter a visibilidade real das complexidades existentes no projecto a fim de mensurá-las e geri-las, compreendendo as necessidades que envolvem a criação do produto de software e as demais necessidades do projecto.

Planeamento de projectos é a actividade de gestão que mais tempo consome. É uma actividade contínua e inclui: organização do projecto; estruturação das tarefas; cronograma do projecto e análise de risco.

Segundo (Pressman, 2002) o objectivo do planeamento de projecto, é fornecer um arcabouço que permita ao gerente fazer estimativas razoáveis de recursos, custo e cronograma. Essas estimativas são feitas dentro de um quadro de tempo limitado no início de um projecto de software e devem ser actualizadas regularmente à medida que o projecto avança. Além disso, as estimativas devem tentar definir cenários correspondentes ao melhor e aos piores casos de modo que o comportamento do projecto possa ser delimitado. O objectivo do planeamento é alcançado através de um processo de descoberta da informação que leva a estimativas razoáveis.

Reduzida à sua forma mais simples, a gerência de projectos é a disciplina de manter os riscos de fracasso em um nível tão baixo quanto necessário durante o ciclo de vida do projecto. O risco de fracasso aumenta de acordo com a presença de incerteza durante todos os estágios do projecto. Gestão de projectos é a disciplina de definir e alcançar objectivos ao mesmo tempo em que se optimiza o uso de recursos (tempo, dinheiro, pessoas, espaço, etc.).

Segundo (Sommerville, 2003), uma gestão eficaz de um projecto de software depende de um planeamento acurado do andamento do projecto. O gerente de projecto deve prever os problemas que possam surgir e preparar soluções experimentais para esses problemas. Um plano traçado no início do projecto deve ser utilizado como guia para esse projecto.

1.3 – PLANEAMENTO DOS RISCOS NO DESENVOLVIMENTO DE SOFTWARE

Inspirados nos estudos de (Sommerville, 2003), (Miguel, 2002) e (Pressman, 2002), tentamos fazer um apanhado das principais ideias e teorias defendidas por esses autores de forma a melhor compreender o tema.

Assim, segundo (Sommerville, 2003), o planeamento de riscos é um processo iterativo que só termina quando o próprio projecto for concluído. À medida que as informações sobre os riscos se tornam disponíveis durante o desenvolvimento do projecto, o plano deve ser regularmente revisado.

Ainda, segundo o mesmo autor, gerentes de projecto experientes não supõem que tudo correrá bem, pois sempre surgirão problemas de algum tipo durante um projecto. As hipóteses e as programações assumidas inicialmente devem ser pessimistas, em vez de optimistas. È preciso tentar prever todas as possíveis eventualidades no plano de projecto, para que as restrições e os marcos do projecto não tenham de ser renegociados a todo o tempo no loop do planeamento.

É neste sentido que, segundo (Miguel, 2002), o planeamento dos riscos destina-se a permitir ao chefe de projecto descobrir aquilo que ele desconhece e a planear o modo de o tratar no futuro. Desta forma, o plano do projecto é antecipadamente elaborado e as coisas podem ser controladas de modo mais eficaz. Desta forma quando chegar o momento de dominar esses riscos, o plano conterá as actividades necessárias para isso. E quando assim é, o chefe de projecto tem assim espaço de manobra. Ainda, segundo o mesmo autor, o planeamento dos riscos visa fornecer a informação necessária para a escolha de uma estratégia de desenvolvimento que tem como objectivo reduzir o risco técnico do projecto e o risco empresarial da organização.

O sucesso de um projecto de desenvolvimento de software é determinado pela sua grande parte por uma boa gestão dos riscos que eventualmente poderão pôr o projecto em causa.

Para resumir toda essa história sobre gestão de riscos, é oportuno lembrar uma frase do consultor Tom Gilb "Se você não atacar os riscos do projecto activamente, então estes irão activamente atacar você."

1.4 - ANÁLISE E GESTÃO DE RISCOS

Segundo (Pressman, 2002), a análise e gestão de riscos é uma série de passos que ajudam uma equipe de software a entender e administrar a incerteza. Ainda, segundo o mesmo autor, um risco é um problema potencial – pode ou não acontecer. Mas, independentemente do resultado, é realmente importante identifica-lo, avaliar a probabilidade de ocorrência, estimar seu impacto e estabelecer um plano de contingência para o caso de efectivamente ocorrer. Ele acha que software é um empreendimento difícil, e que muitas coisas podem dar errado e, francamente, com frequência dão. É por essa razão que os engenheiros de software têm que estar alertas de forma a entender os riscos e tomar medidas preventivas para evita-los ou administra-los.

Para isso há que ter uma visão do projecto global, de forma a poder reconhecer aquilo que pode dar errado, o que podemos chamar de identificação de risco. Após isso, cada risco é analisado para determinar a possibilidade de que venha ocorrer e o dano que vai causar, se efectivamente ocorrer. Após estar na posse dessas informações é vez de se estabelecer a probabilidade e o impacto de cada risco. E por fim, é desenvolvido um plano de atenuação e gestão dos riscos com alta probabilidade e alto impacto. Esse plano deve ser revisto à medida que o projecto evolui, para garantir que os riscos sejam actualizados.

A identificação, previsão, avaliação, administração e monitoração de riscos é um processo que consome tempo mas tem retorno de vários modos, nomeadamente: menos transtornos durante o projecto, maior capacidade de acompanhar e controlar o projecto, e a confiança de que advém do planeamento da resolução de problemas antes que eles ocorram.

CAPÍTULO II - MODELOS DE DESENVOLVIMENTO DE SOFTWARE

2.1 – INTRODUÇÃO

O desenvolvimento de um sistema de software envolve diversas fases, e em cada uma delas, desenvolvem-se diversas actividades. O encadeamento específico dessas fases para construção do sistema dá-se o nome de "modelo de ciclo de vida"[1], ou simplesmente modelo de desenvolvimento. Há diversos Modelos de Ciclo de Vida. A diferença entre um e outro está na maneira como as diversas fases são encadeadas, para transformar os requisitos do cliente em um sistema para o cliente.

O processo de desenvolvimento de software deve ser bem definido, eficiente, controlado, medido e gerido. Para alcançar tudo isso, é necessário utilizar algum modelo de processo de software de entre os vários modelos de processo de software (ou paradigmas de engenharia de software). Cada um representa uma tentativa de colocar ordem numa actividade inerentemente confusa.

Não podemos descrever todos os modelos de processo existentes e ilustrar todos os pontos fracos e fortes deles neste estudo, por eles serem tantos e variados o que impossibilita integrá-los todos neste trabalho. Mas pelo menos faremos uma análise dos riscos associados a cada um dos modelos que vão ser estudados que no nosso entender melhor adequa com os nossos objectivos para com o trabalho no que concerne à gestão de riscos no desenvolvimento de software. Tentaremos demonstrar os pontos fortes e fracos desses modelos escolhidos, bem como o impacto deles no processo de gestão dos riscos. Também não é nossa pretensão descrever o funcionamento de cada modelo de processo de software. Essas informações acerca da descrição de cada modelo são os pré-requisitos para entender aquilo que vamos tratar neste capítulo. Essas informações poderão ser encontradas em (Miguel, 2002); (Pressman, 2002) e também existe uma grande quantidade de informações sobre esses modelos disponível na word wide web.

[1] O termo "modelo de ciclo de vida" é utilizado para descrever um modelo que visa descrever um grupo de actividades e a forma como elas se relacionam

2.2 - OBJECTIVOS

Muitas organizações desenvolvem software sem usar nenhum processo. Isto ocorre porque os processos tradicionais geralmente não são adequados às realidades das organizações. Em particular, as organizações pequenas e médias não possuem recursos suficientes para adoptar o uso de processos pesados. A falta de sistematização na produção de software terá como resultado a baixa qualidade do produto final, além de dificultar a entrega do software nos prazos e custos predefinidos e inviabilizar a futura evolução do software.

“O objectivo de um modelo de desenvolvimento é proporcionar ao projecto uma estrutura que reduza os riscos” (Miguel, 2002, P.148). Que garanta que a exposição aos vários tipos de risco como produzir o sistema errado, ultrapassar o orçamento, produzir o sistema que nunca funcione, etc. se mantenha a um nível considerado aceitável.

Para cada desenvolvimento de software antes de escolher um modelo de processo, deve-se avaliar o nível de complexidade, confiabilidade e tamanho do sistema de forma a escolher a melhor que se adequa às suas características. Entre as quais destacam-se as características de incerteza no início ou da falta de conhecimento a cerca do sistema a ser desenvolvido, de modo a dar ao projecto uma estrutura que reduza os riscos que põem em causa o sucesso do projecto.

A estruturação e sistematização de um projecto de desenvolvimento de software são indispensáveis para a sua gestão. De um projecto sem estrutura não se pode fazer estimativas sobre o seu custo ou a sua qualidade, não se pode ter pontos críticos definidos e o seu progresso não pode ser monitorizado. Para evitar o fracasso de um projecto, a estruturação deve ser um dos pontos a ter em conta, o que se consegue com a utilização de um modelo de processo adequado ao projecto em si com o objectivo de reduzir o risco e a incerteza e aumentar a eficácia das decisões de gestão para com o projecto.

“O modelo do processo usado por um projecto é determinado por aquilo que se sabe e por aquilo que se desconhece sobre o sistema requerido e sobre o seu futuro provável e é desenhado para reduzir a exposição do projecto aos riscos” (Miguel, 2002, p.148).

2.3 - ANÁLISE DOS RISCOS ASSOCIADOS AO MODELO CASCATA

O modelo cascata também chamado de clássico ou linear, é modelo que requer uma abordagem sistemática, sequencial do desenvolvimento (para a análise começar, o levantamento de requisitos deverá estar terminado; para a fase de projecto começar, a fase de análise deverá estar terminada, etc.).

Neste modelo, o projecto vai progredindo com o passar do tempo, passando pelas fases que estão ilustradas na fig.1.

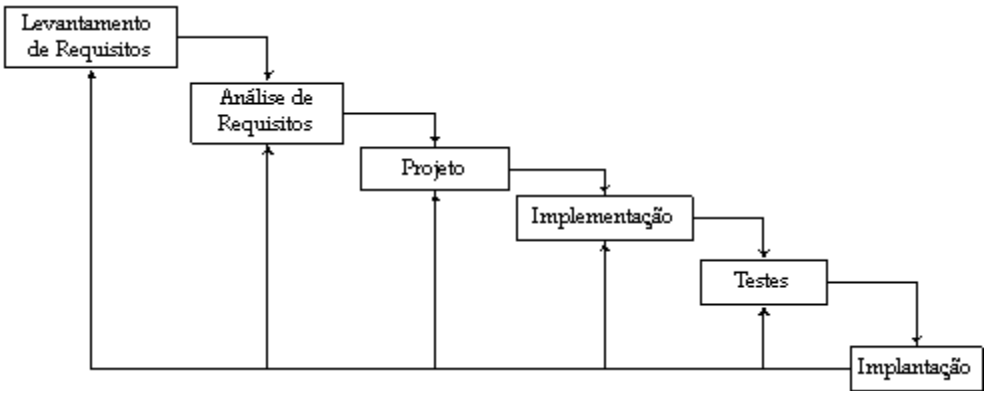


Fig 1 – Etapas do modelo cascata (Bezerra, 2003)

No modelo em cascata uma versão de produção do sistema de software não fica pronta até que o desenvolvimento chegue ao final. Em virtude das fases serem realizadas sequencialmente e a implantação do sistema pode ficar muito distanciada no tempo da fase inicial. Caso aconteça algum erro em alguma fase do desenvolvimento, por exemplo na análise, e vier a ser reconhecido só na fase final, o que é possível acontecer, visto que seres humanos podem cometer erros. Nesta situação tempo, dinheiro, mão-de-obra, etc. foram desperdiçados.

O problema do modelo em Cascata é sua inflexível divisão do projecto em fases distintas, o que dificulta possíveis alterações que são comuns no desenvolvimento de um projecto. E por isso esse modelo só deve ser utilizado somente quando os requisitos forem bem compreendidos. O que quer dizer que o software tem de ser totalmente especificado antes do seu início, visto que a alteração dos requisitos pode constituir um problema enorme devido a sequencialidade de tarefas.

Um outro ponto importante a ter em conta são os requisitos e a sua característica de volatilidade[2]. “No modelo em cascata, é comum a prática de ‘congelar’ os requisitos levantados antes de se iniciar a realização das demais fases do desenvolvimento. Isto é, os requisitos considerados eram os mesmos do início ao fim do projecto de desenvolvimento” (Bezerra, 2003).

De facto, a volatilidade dos requisitos é um facto com o qual a equipe de desenvolvimento tem de conviver.

Sistemas grandes em complexidade podem levar meses ou até anos para serem desenvolvidos. Na actualidade, devido a grande concorrência entre as empresas, é difícil pensar que o usuário espere pacientemente até que o sistema todo esteja pronto para ser utilizado. Mesmo que isto aconteça, pode haver o risco de que o sistema já não corresponda mais às reais necessidades do cliente, em virtude dos requisitos terem mudado durante o tempo de desenvolvimento.

Entretanto, é possível declarar detalhadamente todos os requisitos antes do início das demais fases do desenvolvimento de acordo com o modelo em cascata. Se esta hipótese for assumida, recursos poderão ser desperdiçados na construção de um requisito incorrecto. Esta falha pode propagar por todas as fases do processo, só sendo detectada após o cliente ter começado a utilizar o sistema.

Um outro problema do modelo em cascata é o facto da sua aplicação no desenvolvimento de um software não produzir informações suficientes para que o gerente do projecto possa medir o andamento do mesmo. Assim sendo, não é possível ao gerente realizar uma boa estimativa para o tempo de implementação com base no tempo que se levou para preparar os documentos de análise e projecto.

Uma contribuição importante do Modelo Cascata para o processo de desenvolvimento de software é que o processo de desenvolvimento de software deve ser sujeito à disciplina, planeamento e gestão. Por outro lado, este modelo é bem documentado e favorece uma abordagem top-down.

Sintetizando, podemos dizer que o modelo Cascata tem muitas fragilidades, mas ele é significativamente melhor do que uma abordagem casual ao desenvolvimento de software. Pode ser utilizado em pequenos projectos por ser mais fácil de atingir o objectivo que o modelo propõe que é a especificação de todos os requisitos no início.

[2] Um requisito volátil é aquele que pode sofrer modificações durante o desenvolvimento do sistema

2.4 - ANÁLISE DE RISCOS ASSOCIADOS AO MODELO INCREMENTAL

O modelo incremental foi proposto como uma resposta aos problemas encontrados no modelo cascata. Este modelo (fig.2) divide o desenvolvimento de software em iterações. Em cada iteração, são realizadas as actividades de levantamento de requisitos, análise de requisitos, projecto, implementação, testes e implantação para uma parte do sistema. Esta característica difere do modelo em cascata, na qual essas são realizadas uma única vez para o sistema como um todo.

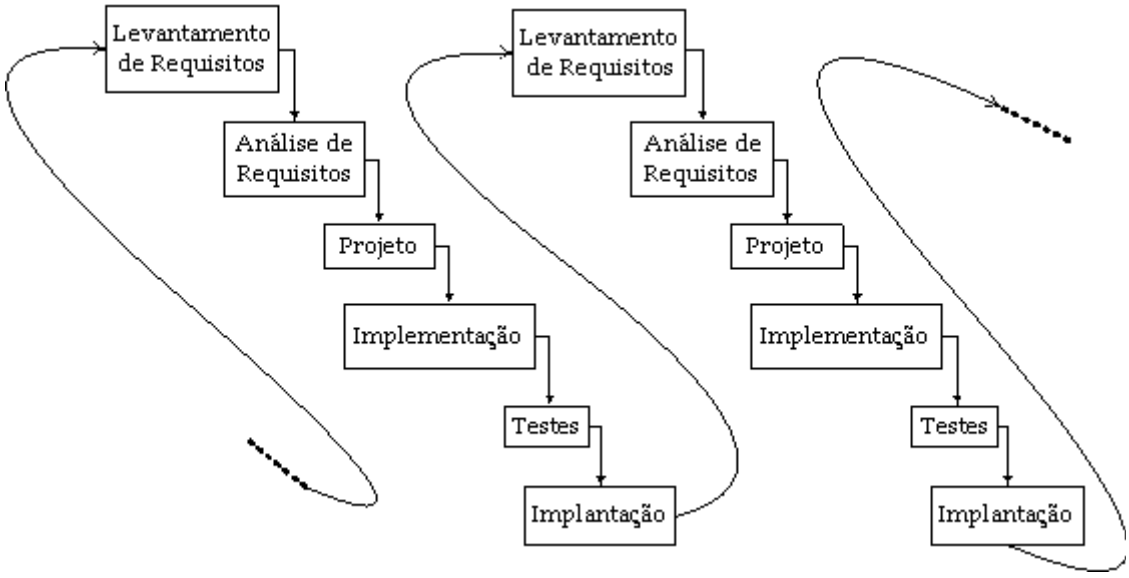


Fig. 2– Modelo Incremental (Bezerra, 2003)

Após a definição dos requisitos para uma iteração de desenvolvimento, estes são analisados, projectados, implementados, testados e implantados. Na iteração seguinte, um outro subconjunto de requisitos são considerados para serem desenvolvidos, o que produz uma nova versão ou incremento do sistema que contém extensões e refinamentos sobre a versão anterior. Desta forma, o desenvolvimento evolui em versões, através da construção incremental e iterativa de novas funcionalidades até que o sistema completo esteja construído.

Nas primeiras iterações, uma versão mínima do sistema é construída. Normalmente essas primeiras iterações ocorrem de uma forma sequencial, pois elas estabelecem a arquitectura geral do software. Podem fazer maior uso de técnicas de prototipagem. Após ter estabelecido essa arquitectura, algumas iterações seguintes podem ser desenvolvidas em paralelo.

Para ser utilizado, o modelo incremental requer um mecanismo para dividir o conjunto de requisitos do sistema e alocá-los a uma iteração. Os factores que devem ser considerados na

divisão dos requisitos são a prioridade (importância do requisito para o cliente) e o risco de cada requisito.

O modelo incremental defende a participação do usuário no processo de desenvolvimento através do uso de versões desde o início do desenvolvimento. O que compensa qualquer falsa expectativa que um usuário possa ter sobre a versão inicial do sistema.

De facto, as versões iniciais do sistema não contêm a implementação de todas as funcionalidades necessárias. Mas essa situação é muito melhor em comparação com o modelo cascata em que o usuário recebe o sistema somente no final do projecto.

Qualquer projecto de desenvolvimento de software está sujeito a riscos e estes não podem ser eliminados por completo. Portanto, todo o processo de desenvolvimento deve ter em conta esses riscos. E, no que toca a esse aspecto, o modelo incremental tem vantagem uma vez que os riscos do projecto podem ser geridos melhor. E também é menos custoso corrigir os erros cometidos em uma iteração se detectados quando a versão correspondente do software começar a ser utilizados. Isso evita a propagação de erros.

Se as inconsistências detectadas em uma versão recém liberada não são tão graves, elas são rapidamente removidas e uma nova versão do sistema é entregue ao usuário. Por outro lado, se as inconsistências descobertas forem graves e tiverem um risco grande no desenvolvimento do sistema, pelo menos a sua identificação torna possível reagir a elas mais cedo sem tantas consequências graves para o projecto. Ou seja, quanto mais cedo a equipa considerar os requisitos mais arriscados, menor é a probabilidade de ocorrer prejuízos.

Os requisitos a serem considerados primeiramente devem ser seleccionados com base nos riscos que eles fornecem. E os mais arriscados devem ser considerados o mais cedo possível.

Apesar do modelo incremental possuir todas as vantagens atrás referidas, esse modelo não resolve todos os problemas do desenvolvimento de software pelas seguintes razões:

Primeiramente, o envolvimento mais estreito do usuário no processo de desenvolvimento e o seu constante feedback em relação as versões que são liberadas resultam em uma quantidade de trabalho significativa para fazer as alterações e correcções necessárias. Assim, se o trabalho resultante da integração dos requisitos em evolução não for bem gerido, ele pode atrasar o ritmo do projecto.

Em segundo lugar, a utilização do modelo incremental torna mais complicada a tarefa de gestão das actividades do projecto. Gerir um projecto onde análise, projecto, implementação,

testes e implantação são feitas em paralelo, caso haja duas ou mais iterações ocorrendo ao mesmo tempo torna-se mais complicado do que gerir o desenvolvimento de um sistema que utilize o modelo em cascata, onde as fases ocorrem de forma sequencial.

Em contrapartida, o modelo incremental permite a produção de informações consistente que ajuda o gerente do projecto avaliar a continuação do desenvolvimento. A liberalização de uma versão de um incremento ao usuário, para fazer uso dela é no sentido descobrir os erros. Logo se uma versão produzida em um incremento tiver erros, a equipe de desenvolvimento fica sabendo que a versão tem erros tomando as devidas precauções antes de avançar com o desenvolvimento.

Ao desenvolvermos software complexo e de requisitos voláteis, estamos sujeitos a riscos. E ao utilizarmos o modelo incremental, reconhecemos os riscos como reais e tentamos atacá-los assim que apareçam para que não se compliquem nas fases subsequentes.

2.5 - ANÁLISE DE RISCOS ASSOCIADOS AO MODELO DE PROTOTIPAÇÃO

O modelo de prototipação (fig.3) pretende ser usado, principalmente para animar e demonstrar os requisitos de um sistema e ajudar os clientes e os responsáveis pelo desenvolvimento do projecto a testarem e a melhorarem o sistema antes deste estar finalizado.

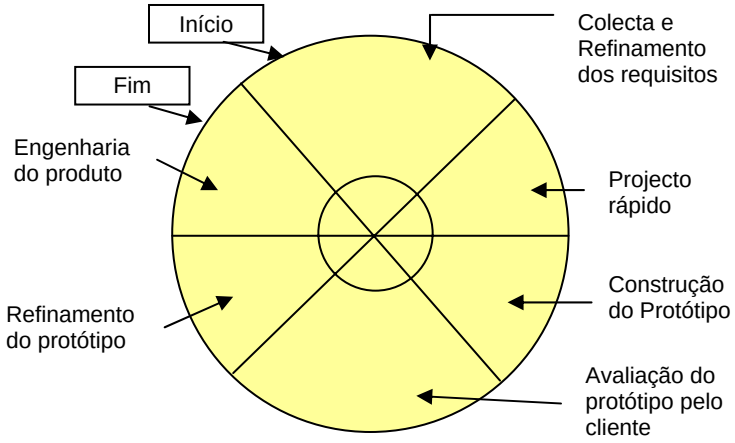


Fig. 3 – Modelo de prototipação

No modelo de prototipação não se produz código efectivo no início, ou seja, começar a desenvolver módulos do programa, uma vez que um protótipo pode ser usado para uma continuação do progresso do sistema ou pode ser completamente descartado. O início do desenvolvimento formal do processo acontece só depois de algumas especificações resultantes de experiências realizadas com o protótipo forem examinadas.

O importante para este modelo é definir as regras do jogo no início, isto é, o cliente e o desenvolvedor têm de estar deacordo que o protótipo seja construído para servir apenas para definição dos requisitos e depois é descartado. E o software real é submetido à engenharia com o objectivo de buscar qualidade e manutenibilidade.

O uso de protótipos nos ajudam quando temos um grande software para desenvolver e queremos envolver usuário na análise do projecto. Isso dá a garantia de responder cabalmente as necessidades do usuário para com o sistema, reduzindo desta forma os riscos dos requisitos.

O protótipo dá ao cliente uma noção das funções do sistema. A utilização de desenhos de tela para demonstrar ao cliente como os campos vão ficar e qual será o fluxo de manuseio, faz rentabilizar o tempo e qualidade na elaboração do protótipo, porque aquilo que se vê é o que se alcança.

Uma das desvantagens do modelo é que o cliente vê o que parece ser uma versão executável do software e desconhece que o protótipo apenas consegue funcionar precariamente. Isto porque na pressa de fazê-lo rodar, ninguém considera a sua qualidade global ou a manutenibilidade a longo prazo.

Após o cliente saber que o produto deve ser refeito de modo que altos níveis de eficácia possam ser atingidos, o cliente reclama e exige alguns reparos, para transformar o protótipo num produto executável. E os desenvolvedores com o desejo de terminar o trabalho o mais rápido possível, em geral concordam.

O desenvolvedor frequentemente faz concessões na implementação a fim de conseguir rapidamente um protótipo executável, poderá usar sistema operacional ou uma linguagem de programação inapropriada simplesmente por estar disponível e serem conhecidos. Poderá também implementar um algoritmo ineficiente simplesmente para indicar uma possibilidade. Com o tempo, o desenvolvedor pode ficar acostumado com essas escolhas e esquecer todas as razões por que elas eram inadequadas, e há que tomar muito cuidado com isso, dado que uma escolha muito abaixo da ideal se tornou parte integral do sistema.

A técnica de prototipagem na identificação de requisitos nem sempre é benéfica, isto porque algumas empresas podem encarar os protótipos como sendo os seus inimigos. Á adaptação ao protótipo, a eficiência de utilização, a aplicabilidade e o comportamento dos clientes que estão a comprar um software podem ter um impacto negativo.

Se a maqueta do sistema for construída sem cuidado especial, pode ser que esta resolva teoricamente o problema errado, ou seja, aparentemente o protótipo pode parecer muito bom e estar muito bem feito, mas na realidade não ir de encontro às necessidades dos potenciais utilizadores. Para além disso, o facto de um protótipo dar a conhecer apenas algumas partes, pode levar que se menosprezem partes fundamentais do sistema, tornando-se incompleto. Saltar passos fundamentais no desenho de um software pode levar-nos à solução mais fácil e simples em vez da melhor solução.

Um outro problema da prototipagem é que se o protótipo for demasiado perfeito e permitir que o utilizador navegue pelo sistema já com um grau de profundidade elevado, pode fazer com que o cliente pense que o projecto já está praticamente pronto, desvalorizando assim a quantidade de trabalho ainda por realizar.

O processo de prototipagem tem custos de produção elevados e ocupa uma quantidade de tempo exagerada. Todos estes pontos devem ser bem medidos tanto pelos engenheiros de software que estão à frente do projecto como também por parte dos clientes.

2.6 - ANÁLISE DE RISCOS ASSOCIADOS AO MODELO EM ESPIRAL DE BOEHM

“O modelo espiral, originalmente proposto por Boehm [Boe88], é um modelo de processo de software evolucionário que combina a natureza interactiva da prototipagem com os aspectos controlados e sistemáticos do modelo sequencial linear”(Pressman, 2002, p.32).

O Modelo em espiral é uma contribuição significativa para a compreensão do risco no planeamento do processo de desenvolvimento de software. Neste modelo o processo é orientado para os riscos, em vez de ser orientado para o produto. Boehm afirma que o modelo em espiral integra os outros modelos, como casos especiais e, dependendo dos riscos presentes no projecto, torna-se equivalente a eles, em algumas situações.

Este modelo, segundo (Miguel, 2002, p.199), procede através da repetição de um processo básico de cinco fases:

1) Determinar os objectivos, alternativos e restrições;
2) Avaliar as alternativas; identificar e resolver riscos;
3) Desenvolver e verificar o produto do nível seguinte;

4) Planear o próximo ciclo;

5) Rever o resultado do ciclo.

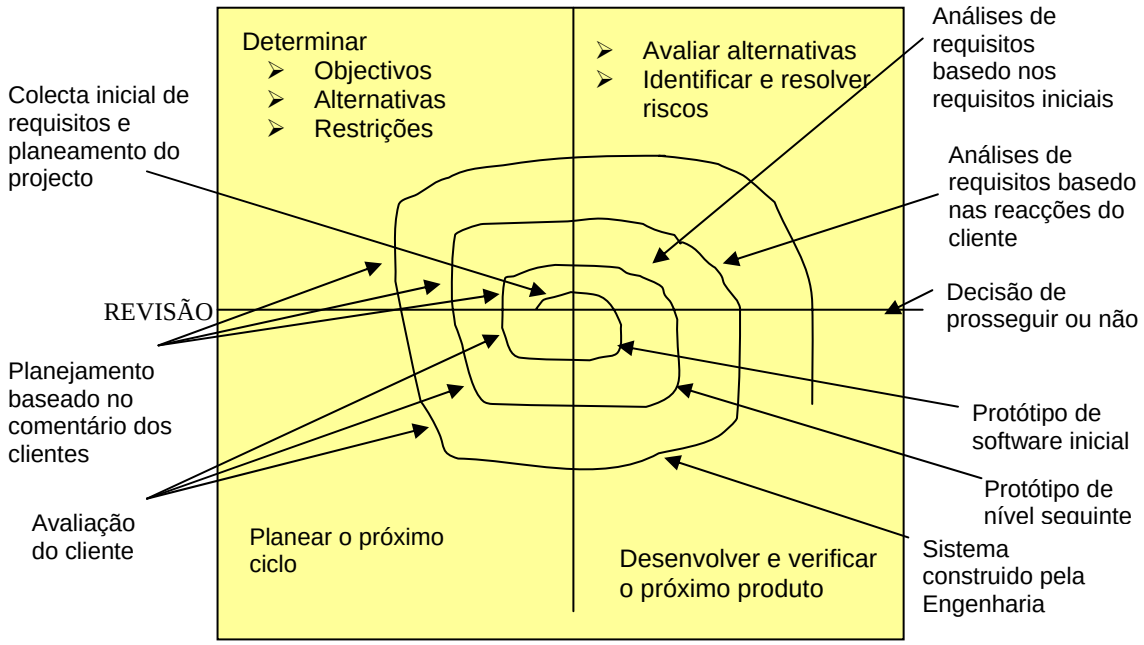


Fig. 4 – Modelo espiral

A fig. 4 indica o modo de manobra do modelo. O desenvolvimento processa-se em torno da origem, no sentido dos ponteiros do relógio, passando pelos quatro quadrantes que representam as fases 1 a 4 atrás mencionadas.

O modelo em espiral é uma abordagem para o desenvolvimento de software de grande porte. Em cada nível evolucionário o desenvolvedor e o cliente entendem melhor e reagem aos riscos à medida que o processo avança. “Ele mantém a abordagem sistemática passo a passo, sugerida pelo ciclo de vida clássico, mas o incorpora numa estrutura iterativa, que reflecte mais realisticamente o mundo real”(Pressman, 2002, p.141). Também ele usa a prototipagem como mecanismo de redução de risco, e o mais importante, é que permite ao desenvolvedor aplicá-la em qualquer estágio da evolução do produto. Exige a consideração directa dos riscos técnicos em todos os estágios do projecto e, se for aplicado adequadamente, reduz os riscos antes que eles fiquem problemáticos.

O modelo em espiral apresenta as seguintes vantagens:

- Flexibilidade do processo, mais adaptável a alterações no desenho e nos requisitos;
- Também pode resultar numa mais rápida disponibilização para o mercado.

- O produto a ser entregue poderá ter uma boa qualidade.
- O resultado final poderá ajustar-se melhor aos requisitos do utilizador, satisfazendo assim as necessidades dos clientes.
- A manutenção dos projectos é feita com uma dimensão reduzida, por exemplo, 2 a 3 analistas/programadores em dois a três meses de duração.

O modelo em espiral pode desenrolar de muitas formas distintas, sendo cada uma dessas formas um modelo de processo em si, dependendo das circunstancias do projecto, “Embora seja possível usar o Modelo em espiral directamente num projecto, é mais útil olhar para um certo número de possíveis processos da espiral que ocorrem frequentemente no desenvolvimento de software” (Miguel, 2002, P.209).

No que respeita aos pontos fracos deste modelo podemos dizer que ele é menos determinista e de gestão mais difícil, sendo difícil à gestão determinar a situação de um projecto em qualquer momento, assim como estimar quantas espirais são necessárias para um projecto.

O modelo em espiral tem um risco de processo maior do que o modelo cascata por ser mais difícil prever o nível de qualidade que o produto virá a ter num determinado momento, ficando cada vez mais difícil efectuar compromissos com os clientes.

“Torna-se igualmente difícil determinar o estado do produto, em termos da sua disponibilidade para sua comercialização em qualquer momento. Os processos tradicionais de medida e as métricas por eles utilizadas – por exemplo, densidade de defeitos ou perfil cumulativo de falhas – não se revelam muito úteis para a gestão deste tipo de projectos” (Miguel, 2002, p.203). O modelo espiral exige competência considerável na avaliação dos riscos e depende dessa competência para ter sucesso.

CAPÍTULO III METODOLOGIAS DE DESENVOLVIMENTO DE SOFTWARE

3.1 - INTRODUÇÃO

À semelhança dos modelos de desenvolvimento de software, não podemos falar de todas as metodologias existentes, por elas serem inúmeras e variadas, nesta ordem de ideia pretendemos deixar aqui uma reflexão sobre as metodologias tradicionais e metodologias ágeis e particularmente a metodologia Rational Unified Process (RUP) e Extreme Programming (XP).

“A metodologia de projecto é um plano para administrar um projecto qualquer e está normalmente dividida em três partes distintas: a especificação do escopo (objectivos do projecto), a especificação das actividades executadas (aplicações dos usuários) e as especificações de como as diversas partes constituintes deverão interagir entre si (Pinheiro, 2004).”

Qualquer metodologia deve ser estruturada de modo a incluir um projecto lógico antes de constituir um projecto físico e abordar os requisitos dos usuários do sistema antes de considerar outras variáveis. Ela também deve ser interactiva, ou seja, novas informações devem entrar progressivamente no projecto, à medida que se compreende melhor os requisitos definidos pelos usuários, a fim de corrigir desvios e eventuais falhas.

Não existe e nunca existirá uma metodologia boa para ser a única usada e nenhum software é igual ao outro. Cada software, equipe, processo, trabalho, vai ter uma metodologia diferente, nunca igual. E sempre se acrescentam alguns passos, pois ter regras ajuda, mas na hora certa temos que improvisar muita coisa e cada projecto acaba bem particular e peculiar. Portanto, regras demais é ruim, temos que seguí-las e sabermos virar sozinhos colocando a nossa experiência em prática.

Num projecto quase sempre se descobre novos bugs[3] que nos podem levar de novo ao início. Quando assim é, teremos que repetir todas as etapas e isso não tem fim, pois não existe software perfeito ou livre de bugs, por isso, temos que encarar os projectos como variáveis, dado que eles mudam frequentemente e quase sempre acabamos nos esbarrando com o começo.

3.2 - METODOLOGIAS ÁGEIS E METODOLOGIAS TRADICIONAIS

Existem vários processos de software definidos na literatura da Engenharia de Software. Algumas organizações por vezes criam o seu próprio processo ou adapta algum processo à sua realidade. Entre os vários processos existentes, se destaca as metodologias (abordagens) tradicionais, que são orientadas a documentação, e as metodologias ágeis, que procuram desenvolver software com o mínimo de documentação.

Entre todas as metodologias ágeis existentes, a Extreme Programming (XP) vem-se destacando em número de adeptos e projectos. Segundo (Soares, 2001) as metodologias ágeis surgiram com a proposta de aumentar o enfoque nas pessoas e não nos processos de desenvolvimento. Além disso, existe a preocupação de gastar menos tempo com documentação e mais com resolução de problemas de forma iterativa.

As abordagens tradicionais são também chamadas de pesadas ou orientadas a documentação, elas surgiram em um contexto de desenvolvimento de software muito diferente do actual. Na época, o custo de fazer alterações e correcções era muito alto, uma vez que o acesso aos computadores eram muito limitado e não existiam modernas ferramentas de apoio ao desenvolvimento do software, como depuradores e analisadores de código. Por isso o software era todo planeado e documentado antes de ser implementado.

As abordagens do tipo tradicional identificam uma sequência de passos a serem completados e bastante controversa, especialmente em projectos muito complexos, mesmo assim, tem conquistado adeptos em números crescentes.

Na abordagem tradicional, distinguimos cinco estágios no desenvolvimento de um projecto: iniciação; planeamento; produção; monitoração e fechamento de projecto.

[3] Bugs – é um erro involuntário de programação que pode não ser responsável pelo bloqueio de uma aplicação, programa ou sistema operativo mas que causa alguns problemas na operação ou utilização. [Matos 2004; pp 56]

Nem todos os projectos vão seguir todos estes estágios, já que projectos podem ser encerrados antes do seu término, pois podem existir projectos sem planeamento ou monitoração e muitas vezes passam pelos estágios 2, 3 e 4 múltiplas vezes.

No que toca as metodologias ágeis podemos apontar os seguintes conceitos chave:

- Indivíduos e interacções ao invés de processos e ferramentas;
- Software executável ao invés de documentação;
- Colaboração do cliente ao invés de negociação de contratos;
- Respostas rápidas a mudanças ao invés de seguir planos.

Isto não quer dizer que as metodologias ágeis rejeitam os processos e ferramentas, a documentação, a negociação de contratos ou o planeamento, mas simplesmente eles têm importância secundária quando comparado com os indivíduos e interacções, com o software executável, com a colaboração do cliente e as respostas rápidas às mudanças e alterações.

Projecto de software é algo difícil de se controlar. Por isso é necessário utilizar metodologias que minimizam os riscos garantindo que os engenheiros de software foquem em unidades menores de trabalho.

3.2 - EXTREME PROGRAMMING (XP)

XP é uma evolução do ciclo de vida baseada em prototipação, diferenciando-se desta na geração de documentação e de um produto no fim de cada ciclo do processo, mas mantendo as características que fazem o modelo de prototipação ser tão bem visto no mundo da engenharia de software.

Dentro do modelo de prototipação temos a geração de documentação específica, os relatórios e a avaliação dos riscos no fim de cada ciclo. Na XP o que é gerado como documentação é o código propriamente dito, evitando assim a geração de uma grande quantidade de documentação que deve ser actualizada exigindo muito trabalho para tal. Pois o desenvolvedor deve produzir linhas de código e bancos de dados, análise, projecto e depois actualizar os modelos, gráficos e relatórios gerenciais. Essas documentações formais existem na XP, porém se recomenda que elas sejam descartáveis e que a acumulação de papéis seja evitada.

Extreme Programming (XP) vem fazendo sucesso, por ajudar a criar sistemas de melhor qualidade, que são produzidas em menos tempo e de forma mais económica que o habitual. Tais objectivos são alcançados através de um pequeno conjunto de valores e práticas, que diferem substancialmente da forma como se desenvolve software na grande maioria dos projectos.

É uma forma mais humana, onde todos clientes, desenvolvedores e demais interessados no projecto são identificados como pessoas que falham e que acertam. A estrutura de desenvolvimento criada pelo XP procura ajudar o projecto a aproveitar o que as pessoas têm de melhor e solucionar suas falhas com rapidez e segurança.

XP prioriza as actividades da equipe permanentemente para evitar que trabalhos desnecessários sejam executados. Isto para poupar tempo e recursos. O mais importante em XP não é trabalhar muito e produzir muito, mas sim produzir a coisa certa, aquilo que o cliente realmente identifica como sendo valioso para resolver seus problemas. E isto deverá ser feito de forma consistente, segura e rápida ao longo de todo o andamento do projecto.

XP é uma metodologia ágil para equipas pequenas e médias que desenvolvem software baseado em requisitos vagos e que se modificam rapidamente. Este tipo de abordagem é muito interessante para projectos de pequeno porte e que não abarquem rotinas muito complexas onde a regra de negócio não é facilmente compreendida, dado que o XP prega que deve ser usado o mínimo, ou se possível, nada de documentos a respeito do software.

Se por um lado isso é interessante, já que o projecto parte muito rapidamente para o desenvolvimento, por outro lado isso é muito ruim em relação à questão da manutenção do sistema.

Características do XP:

- Pequenos ciclos com concreto e contínuo feedback;
- Abordagem incremental, que surge como um plano abrangente que se desenvolve durante toda a vida do projecto;
- Habilidade de agenda flexível da implementação de funcionalidades, respondendo a mudanças das necessidades do negócio;
- Confiança em testes automáticos escritos por programadores e clientes para monitorar o progresso do software;

- Confiança na comunicação oral, testes e códigos-fonte para comunicar a estrutura e objectivo do sistema;
- Confiança no processo de desenho evolucionário.

As principais diferenças da XP em relação às outras metodologias são: o feedback constante; abordagem incremental; e o encorajamento da comunicação entre as pessoas.

“A maioria das regras da XP causa polémica à primeira vista e muitos não fazem sentido se aplicadas isoladamente. É a sinergia de seu conjunto que sustenta o sucesso de XP, encabeçando uma verdadeira revolução no desenvolvimento de software.” (Soares, 2001).

XP enfatiza o desenvolvimento rápido do projecto e visa garantir a satisfação do cliente, além de favorecer o cumprimento das estimativas. As regras, práticas e valores da XP proporcionam um agradável ambiente de desenvolvimento de software para os seus seguidores, que são conduzidos por quatro valores: comunicação, simplicidade, feedback e coragem. A finalidade do princípio de comunicação é manter o melhor relacionamento possível entre clientes e desenvolvedores, preferindo conversas pessoais a outros meios de comunicação. A comunicação entre os desenvolvedores e o gerente do projecto também é encorajada. Procura-se o máximo possível comunicar-se pessoalmente, evitando-se o uso de telefone e o envio de mensagens por correio electrónico. A simplicidade visa permitir a criação de código simples (com o menor número possível de classes e métodos) e que não deve possuir funções desnecessárias.

Outra ideia importante da simplicidade é procurar implementar apenas requisitos actuais, evitando-se adicionar funcionalidades que podem ser importantes no futuro. A aposta da XP é que é melhor fazer algo simples hoje e pagar um pouco mais amanhã para fazer modificações necessárias, do que desenvolver algo complicado hoje que talvez não venha a ser usado, devido a mudança dos requisitos.

A prática do feedback constante significa que o programador terá informações constantes do código e do cliente. A informação do código é dada pelos testes constantes, que indicam os erros tanto individuais quanto do software integrado. Em relação ao cliente, o feedback constante significa que ele terá frequentemente uma parte do software totalmente funcional para avaliar e sugerir novas características e informações aos desenvolvedores.

Eventuais erros e não conformidades são rapidamente identificados e corrigidos nas próximas versões. Desta forma, a tendência é que o produto final esteja de acordo com as expectativas reais do cliente.

É necessário coragem para implantar os três valores anteriores. Por exemplo, não são todas as pessoas que possuem facilidade de comunicação e têm bom relacionamento. A coragem também dá suporte à simplicidade, pois assim que a oportunidade de simplificar o software é percebida, a equipe pode experimentar. Além disso, é preciso coragem para obter feedback constante do cliente.

XP começa com quatro valores: comunicação, feedback, simplicidade e coragem. Depois disso, segundo (Beck, 1999) são construídas 12 práticas[4] que os projectos XP devem seguir. Muitas dessas práticas são técnicas antigas e testadas. Além de ressuscitar essas técnicas, XP as tem como um todo sinérgico, onde cada técnica reforça as outras. Há uma confiança muito grande na sinergia entre elas, os pontos fracos de cada uma são superados pelos pontos fortes de outras

Uma das técnicas mais notáveis é a forte ênfase nos testes. Enquanto todos os processos mencionam a verificação através de testes, a maioria a faz de forma pouco enfática. Entretanto, XP coloca-a na base do desenvolvimento, com cada programador escrevendo testes à medida que escrevem código de produção. Os testes são integrados em um processo de integração contínua, o que leva a uma plataforma altamente estável para futuros desenvolvimento.

Dentro dos diversos pontos que a XP traz para a melhoria do desenvolvimento de software e sistema de informação aparece um que tem acompanhado as pequenas e médias empresas que é o envolvimento pessoal com os colegas de equipa. Estes laços, que são fundidos com companheirismo e amizade, podem trazer problemas que envolvem o lado pessoal, criando-se barreiras que muitas vezes podem comprometer o ambiente de trabalho e a produtividade. Este facto deve ser levado em conta na contratação de recursos humanos que além de talento, técnica e comprometimento com o trabalho o profissional, deve ser compatível com os restantes membros da equipa. Neste sentido torna-se fundamental, em qualquer empresa, uma administração voltada para a gestão de recursos humanos, visto que a continuidade da sua existência será determinada pela qualidade aliada aos seus produtos ou serviços, tendo como base pessoas motivadas e com alto nível de qualidade pessoal e profissional.

[4] Essas práticas poderão ser consultados em (Beck, 1999).

3.3 - O RATIONAL UNIFIED PROCESS (RUP)

Rational Unified Process, é uma metodologia de projecto que descreve como desenvolver software usando técnicas testadas e aprovadas comercialmente, aplicável a equipas de desenvolvimento de software.

O RUP é um processo de engenharia de software bem definido e bem estruturado. O RUP define claramente quem é responsável por o quê, como as coisas devem ser feitas e quando fazê-las. O RUP também provê uma estrutura bem definida para o ciclo de vida de um projecto RUP, articulando claramente os marcos essenciais e pontos de decisão.

As configurações do RUP podem ser criadas para suportar equipas grandes e pequenas, e técnicas de desenvolvimento disciplinadas ou menos formais. O produto IBM RUP contém várias configurações e visões de processos prontas que guiam analistas, desenvolvedores e gerentes de projecto, gerentes de configuração, analistas de dados, e outros membros da equipa de desenvolvimento em como desenvolver o software. Esse produto tem sido utilizado por muitas companhias em diferentes sectores da indústria.

O RUP utiliza a Linguagem Unificada de Modelagem (UML), e, por ser flexível e configurável, pode ser utilizado em projectos de pequeno, médio e grande porte.

3.3.1 - Os Princípios do RUP

Não há uma maneira exacta de aplicar o RUP, uma vez que ele pode ser aplicado de várias formas. Logo será diferente em cada projecto e organização. Porém, segundo (Luiz, 2004) existem alguns princípios que podem caracterizar e diferenciar o RUP de outros métodos iterativos entre eles:

- Atacar os riscos cedo e continuamente;
- Certificar-se de entregar algo de valor ao cliente;
- Focar no software executável;
- Acomodar mudanças cedo;
- Liberar um executável da arquitectura cedo;
- Construir o sistema com componentes;
- Trabalhar junto como um time;

- Fazer da qualidade um estilo de vida, não algo para depois.

3.3.2 - Importância da gestão de riscos com o RUP

A gestão de riscos é essencial em todo o projecto e, se não for iniciada com o início da engenharia de requisitos, pode criar grandes dissabores tanto às expectativas dos utilizadores e gestores, quanto às funcionalidades do sistema e à qualidade do projecto. O RUP tem uma estratégia interessante de tentativa de minimização de riscos baseada nas melhores práticas de desenvolvimento iterativo, com cada iteração a ser baseada nos requisitos e seus riscos associados. Desta maneira em vez de fechar os olhos aos riscos, toma-se decisões conscientes e medidas com base nas incertezas analisadas.

O gestor de projecto segue todo o processo da gestão de projecto. O processo começa com a identificação dos riscos, seguido de um plano de projecto (que inclui os requisitos a desenvolver), e seguidamente se define os passos das iterações:

1. Definir o plano das iterações;
2. Executar a iteração;
3. Avaliar a execução;
4. Revisitar a lista de riscos.

Estes quatro passos são repetidos para cada iteração, mas no final de cada um temos uma lista revisitada de riscos.

É desenvolvida uma lista inicial de requisitos que são priorizados e implementados em várias iterações consoante a prioridade. Esta priorização deve ser baseada nos riscos de implementação dos diferentes requisitos, tecnologias, etc. As primeiras iterações podem, por exemplo, servir como experimentação ou definição de um protótipo com os requisitos cujos riscos tecnológicos são mais elevados. Após as primeiras iterações pode-se reavaliar os riscos do projecto e, se necessário, alterar prioridades relativamente à implementação. Desta maneira o RUP tenta minimizar e identificar riscos que na aproximação tradicional só seriam detectados tardiamente, verificando a qualidade e controlando-os.

A ideia chave da gestão de risco é não esperar passivamente até o risco se materializar e tornar-se um problema ou mesmo acabar com o projecto para decidir o que fazer com o risco.

Para cada risco identificado é necessário definir o que se vai fazer com ele. O RUP sugere três possibilidades:

- Evitar o risco - implica reorganizar o projecto;
- Transferir o risco - reorganizar o projecto para outra pessoa ficar com o risco;
- Aceitar o risco - viver com o risco.

O RUP tem uma estratégia interessante de tentativa de minimização de riscos baseada nas melhores práticas de desenvolvimento iterativo, com cada iteração a ser baseada nos requisitos e seus riscos associados. Desta maneira, em vez de fechar os olhos aos riscos, toma-se decisões conscientes e medidas com base nas incertezas analisadas.

Com a utilização de uma metodologia de desenvolvimento de software como o RUP, é possível obter:

- Qualidade de software;
- Produtividade no desenvolvimento, operação e manutenção de software;
- Controle sobre desenvolvimento dentro de custos, prazos e níveis de qualidade desejados;
- Estimativa de prazos e custos com maior precisão.

Apesar dos benefícios, deve-se ter a consciência de que os benefícios não virão de maneira imediata. É necessário adquirir treinamento adequado, adaptação da metodologia no contexto no qual ela será utilizada, apoio especializado para as equipas de desenvolvimento e tempo para a absorção da metodologia.

CAPÍTULO IV - GESTÃO DO RISCO

4.1 - INTRODUÇÃO

Qualquer projecto implica risco. O objectivo de um projecto é estabelecer ou obter algo novo, apostar e arriscar. O risco é inerente às actividades. O que queremos saber é o que podemos fazer para evitar as causas e as consequências.

O desenvolvimento de software é por si só arriscado, mas não problemático desde que se tenha feito uma boa análise de risco. E para fazer uma boa análise de risco há que responder estas questões que são relevantes no nosso entender para poder tomar uma posição quanto a viabilidade do projecto de software. Que riscos corremos? Porquê? Quais as consequências? Qual a sua gravidade? O que podemos fazer para minimizar a exposição ao risco? Temos formas de controlo do risco? Como?

Apesar do risco ser algo inerente aos projectos de desenvolvimento de software, uma atitude comum face ao risco é ignorar e esperar que nada de grave aconteça. Contudo existem muitos projectos de software desastrosos que poderiam ter sido evitados caso tivesse sido realizada uma gestão de riscos desde a fase inicial do projecto, nomeadamente na Gestão de Requisitos.

No entanto, o desenvolvimento de software não constitui o único empreendimento humano arriscado. Novos procedimentos médicos, novos edifícios, novas medidas financeiras também possuem riscos ou seja todos podem falhar.

“O risco faz parte de qualquer actividade humana, e não pode ser nunca eliminado. O risco, em si, não é mau; o risco é essencial ao progresso e o insucesso constitui, muitas vezes, uma componente fundamental da aprendizagem” (Miguel, 2002, p.50). Para tal devemos aprender a equilibrar as possíveis consequências negativas do risco com os benefícios da respectiva oportunidade associada.

O processo de gestão de riscos, como todos os outros planeamentos de projecto, é um processo interactivo que continua ao longo do projecto. Uma vez traçado um conjunto inicial de planos, a situação é monitorada. A medida que as informações sobre os riscos se tornam

disponíveis, elas têm que ser reavaliadas e novas prioridades devem ser estabelecidas. Os planos para evitar riscos e os planos de contingência podem ser modificados com o surgimento de novas informações sobre os riscos. Os resultados do processo da gestão de riscos devem ser documentados em um plano. Esta fase deve incluir uma discussão sobre os riscos apresentados pelo projecto, uma análise desses riscos e os planos que são necessários para geri-los. Quando for apropriado, pode também incluir resultados de gestão dos riscos, isto é, planos específicos de contingência a serem activados caso o risco venha a ocorrer.

“Os tipos de risco que podem afectar um projecto dependem do projecto e do ambiente organizacional em que o software está sendo desenvolvido. Contudo, muitos riscos são considerados universais (Sommerville, 2003, p.71).” (ver quadro 1 em anexo)

4.2 - OBJECTIVOS DA GESTÃO DE RISCOS

O objectivo principal da informação de risco é providenciar o enquadramento para desenvolver a informação de risco necessária para ajudar no processo de decisão.

Durante o planeamento do futuro da empresa, a administração deve garantir que todos os cuidados foram tomados para que seus planos se concretizem. A formalização de uma análise de risco provê um documento que indica se este cuidado foi observado. O resultado da análise de risco dá à organização o controle sobre seu próprio destino. Através do relatório final, pode-se identificar quais controles devem ser implementados em curto, médio e longo prazo.

“Como analisar riscos sem estudar minuciosamente os processos de negócios que sustentam sua organização? O que quero proteger? Como classificar o risco destes processos sem antes avaliar as vulnerabilidades dos componentes de tecnologia relacionados a cada processo? Quais são os seus processos críticos? Aqueles que sustentam a área comercial, a área financeira ou a produção? Para cada pergunta, uma mesma resposta: conhecer para proteger” (Azevedo, 2002).

A transcrição acima demonstra claramente o objectivo da análise de risco e reflecte a preocupação que as empresas de desenvolvimento de software têm em saber qual o grau de exposição frente às ameaças capazes de comprometer os objectivos da sua operação. Isto torna a análise de risco uma actividade imprescindível.

4.3 – CATEGORIA DE RISCOS

O risco envolve duas características: Incerteza e perda. Incerteza porque o risco pode ou não acontecer, isto é, não há riscos 100% prováveis. Perda porque se o risco tornar real, consequências indesejadas ou perdas ocorrerão.

Na análise de riscos é muito importante quantificar o nível de incerteza e o grau de perda, associados com cada risco. Para conseguir isso, é necessário considerar as seguintes categorias:

Riscos de projecto - são ameaças ao plano do projecto. Isto é, se os riscos de projecto vieram a acontecer na realidade poderão atrasar o cronograma do projecto e aumentar os custos. Estes riscos identificam problemas relacionados com o orçamento, calendarização, quantidade e organização do pessoal, recursos, cliente, e de requisitos e seu impacto num projecto de software. A complexidade, o tamanho e o grau de incerteza estrutural de um projecto são também factores de risco de projecto.

Riscos Técnicos - são problemas que possam surgir e que põem em causa a qualidade do software a desenvolver bem como o prazo de entrega do mesmo. A concretização de um risco técnico na realidade poderá tornar a implementação difícil ou impossível. Os riscos técnicos identificam problemas potenciais de projecto, de implementação, de interface, de verificação e de manutenção.

Riscos do negócio – são problemas que põem em causa a viabilidade do software a ser construído e que poderão comprometer o projecto ou o produto. “Os prováveis 5 riscos principais de negócio são: (1) Construir um produto ou um sistema excelente que ninguém realmente quer (risco de mercado); (2) Construir um produto que não se encaixa mais na estratégia geral de negócios da empresa (risco estratégico); (3) Construir um produto que a equipe de vendas não sabe como vender; (4) Perda de apoio da gerência superior devido a mudança de enfoque ou de pessoal (risco gerencial); (5) Perda de comprometimento orçamentário ou de pessoal (risco orçamentário)” (Pressman, 2002, p. 141).

“**Riscos conhecidos** - são aqueles que podem ser descobertos após a avaliação cuidadosa do plano de projecto, do ambiente técnico e comercial, no qual o projecto está sendo desenvolvido, e de outras fontes de informação confiáveis (p. ex., prazo de entrega irreal, falta de documentação de requisitos ou do escopo do software e mau ambiente de desenvolvimento)” (Pressman, 2002, p. 141).

4.4 - PROCESSO DE GESTÃO DE RISCOS

O processo de gestão de riscos está ilustrado na fig. 5 e envolve vários estágios: identificação de riscos, análise de riscos, planeamento de riscos e monitoração de riscos. Estes estágios serão desenvolvidos neste capítulo.

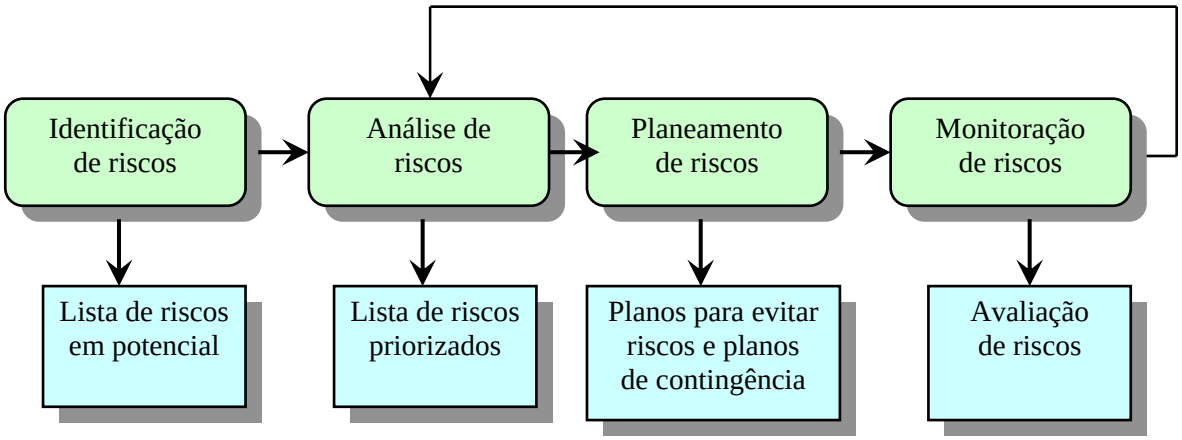


Fig 5: Processo de gestão de riscos[5]

4.4.1 – Identificação de riscos

A identificação de riscos é o primeiro estágio da gestão de riscos. Ela é uma tentativa sistemática de especificar ameaças ao plano do projecto no que toca a estimativas, cronograma, desempenho, recursos etc. Neste estágio em princípio os riscos não devem ser avaliados ou priorizados, embora na prática, segundo Ian Sommerville, os riscos com consequências muito pequenas ou com muita pouca probabilidade não sejam normalmente levados em consideração.

O estágio da identificação de riscos visa transformar incertezas acerca do projecto em riscos bem definidos, que podem ser descritos e medidos. Neste estágio deve-se descrever o risco e definir o contexto do risco.

O objectivo da descrição do risco é chegar a uma descrição concisa do risco, que possa ser facilmente compreendida e que permita a tomada de acções concretas e eficazes.

A actividade de descrição do contexto do risco evolve o registo de toda a informação adicional respeitante às circunstâncias, eventos e inter-relações dentro do projecto, que possam afectar o risco. Isto é feito com o intuito de providenciar os projectistas a informação

[5] Adaptado de (Sommerville, 2003, p. 72)

adicional do risco suficiente de modo a segurar a compreensão do significado original do risco por outras pessoas a medida que o tempo vai passando.

A descrição do contexto de um risco descreve “o que”, “quando”, “onde”, “como” e “porque” do risco, através das circunstancias que rodeiam o risco e possibilita a sua compreensão por outras pessoas.

Para realizar as tarefas desta fase, pode-se utilizar várias metodologias. O quadro 2, em anexo, mostra um resumo dessas metodologias.

Para cada uma das categorias apresentadas na secção anterior há dois tipos distintos de risco: riscos genéricos e riscos específicos do produto. “*Riscos genéricos* são uma ameaça potencial a todo projecto de software. *Riscos específicos do produto* podem ser identificados somente por aqueles que tem um claro entendimento da tecnologia, do pessoal e do ambiente que são específicos do projecto em mãos” (Pressman, 2002, p.141).

Um método para a identificação de riscos é a criação de uma lista de risco. Esta lista é constituída por um conjunto de riscos conhecidos e previsíveis das seguintes subcategorias genéricas:

- *Tamanho do produto* – riscos associados à dimensão geral do software a ser construído ou modificado.
- *Impacto no negócio* – riscos associados às restrições impostas pela gerência ou pelo mercado.
- *Características do cliente* – riscos associados à sofisticação do cliente e com a capacidade do desenvolvedor de se comunicar com o cliente a tempo.
- *Definição do processo* – riscos associados ao grau em que o processo de software foi definido e é seguido pela organização de desenvolvimento.
- *Ambiente de desenvolvimento* – riscos associados à disponibilidade e a qualidade das ferramentas a ser usadas para construir o produto.
- *Tecnologia para a construção* – riscos associados à complexidade do sistema a ser construído e a “novidade” da tecnologia que é incorporada ao sistema.
- *Tamanho e experiência da equipe* – riscos associados à técnica em geral e à experiência no projecto, dos engenheiros de software que vão fazer o trabalho.

Após a elaboração da lista de riscos, um conjunto de componentes e factores de risco é listado juntamente com a sua probabilidade de ocorrência. E por fim factores de desempenho, de custo e de cronograma são discutidos no estágio da análise de riscos.

Elaboração da lista de riscos

A lista de riscos de um projecto pode ser compilada em reuniões de equipa, a partir de experiência adquiridos em projectos anteriores e das listas dos riscos. O chefe da equipa e a sua equipa devem proteger o projecto actual dos riscos que fustigaram o projecto anterior. E se todos os membros da equipa se lembrarem dos enganos, dificuldades e dos riscos não geridos no projecto anterior, rapidamente chegarão a uma lista para o novo projecto.

A lista de riscos de projectos anteriores constitui o ponto de partida para ajudar a identificação de riscos do novo projecto. Cada risco deve ser confrontado com a sua relevância actual. Para isso há que interrogar: “isto foi um risco do projecto anterior, será que pode vir a ser um risco deste projecto?”

É necessário desenvolver uma lista de riscos completa, isto é, uma que cubra não apenas os riscos técnicos, como também os riscos políticos que podem diminuir o moral da equipa, ou os riscos organizacionais que podem afectar o desejo das pessoas em trabalhar na equipa ou um conjunto de outros riscos.

A análise de causa-efeito é muito importante para se tornar a identificação dos riscos ainda mais relevante. Começa-se com o efeito “o projecto falhou” e recua-se nas causas, a partir dai. O que é que poderia ter provocado a falha? De que modo falharam os objectivos? E o que poderia ser feito para que estas coisas não acontecem neste projecto? E assim se vai recuando, até se atingir um risco que se possa começar a tratar.

Com este tipo de análise é possível chegar às causas raiz de uma forma rápida no qual poderemos ser capazes de as atacar. À medida que recuamos na pesquisa das causas de um risco, começamos a descobrir formas de reduzir esse risco. Nesta fase de identificação dos riscos é comum pessoas começarem a fazer aquilo que poderemos considerar de análise de riscos. Mas isso não tem problema desde que se termine o assunto principal que é a elaboração de uma lista de todos os riscos potenciais.

Se tivermos um projecto muito grande poderemos ter também uma lista de riscos muito grande. Isto faz com que nós sentimos a necessidade de termos uma ferramenta de suporte para geri-los. Para tal podemos utilizar uma base de dados para manter o registo dos riscos ou

mesmo simplesmente utilizar uma folha de cálculo. Ter o registo de riscos em formato electrónica é muito importante porque encoraja as actualizações frequentes ou, pelo menos, simplifica o processo de actualização.

4.4.2 – Análise de riscos

Uma análise de riscos deve ser realizada sempre antecedendo um investimento. Antes da organização iniciar um projecto, um novo processo de negócio, desenvolvimento de uma ferramenta ou até mesmo uma relação de parceria, deve-se identificar e assegurar os requisitos do negócio.

A análise dos riscos visa identificar os aspectos do projecto que põem em causa o seu sucesso. Ao fazer-se a análise dos riscos, deve-se identificar: a natureza de cada risco, a causa de cada risco, o impacto de cada risco e a possibilidade de problemas acidentais.

A análise de riscos comprende três actividades básicas:

1. Avaliação da probabilidade e impacto do risco;
2. Classificação dos riscos;
3. Ordenação dos riscos em função da probabilidade e impacto.

“Cada risco pode ser decomposto numa **causa** e num **efeito**.

A causa tem uma **probabilidade** e o efeito tem uma dimensão ou **impacto”** (Miguel, 2002 p.77).

Durante o processo de análise de riscos, cada risco identificado é considerado individualmente e é feito um julgamento sobre a probabilidade e a seriedade desse risco. O resultado desse julgamento depende da experiência do gerente de projecto.

Em geral, não é preciso uma avaliação numérica exacta, mas essa análise deve ter como base uma análise com utilização de intervalos. O quadro 3, em anexo, mostra três níveis de detalhe possível para as escalas dos valores dos atributos.

Os resultados do processo de análise devem, então, ser apresentados numa tabela ordenada de acordo com a seriedade do risco. Obviamente que neste caso, a avaliação de probabilidade e seriedade é arbitrária. Para fazer essa avaliação é preciso ter informações detalhada sobre projecto, processo, equipe de desenvolvimento e da organização.

A probabilidade bem como a avaliação dos efeitos de um risco podem se modificar, à medida que mais informações sobre o risco se tornam disponíveis. Por isso, a cada iteração a tabela de risco deve ser actualizada.

Depois da identificação dos riscos, cada risco é analisado para determinar a possibilidade de que venha ocorrer e o dano que vai causar, se efectivamente ocorrer. Depois de ter estabelecida essa informação, os riscos são classificados por probabilidade e impacto. E por fim, é desenvolvido um plano para administrar aqueles riscos com alta probabilidade e alto impacto.

O processo de análise de riscos deve envolver especialistas em análise de riscos e especialistas no negócio da empresa. Esta sinergia possibilita o foco e a qualidade do projecto.

Uma Análise de Risco bem realizada dará informações à empresa para garantir a confidencialidade, disponibilidade e Integridade das suas informações.

a) Previsão de risco

A previsão de risco, ou estimativa de risco tenta avaliar cada risco de dois modos quanto a probabilidade do risco seja real e as consequências dos problemas associados ao risco, se ele ocorrer. Para tal é necessário desenvolver quatro actividades de previsão de risco: (1) estabelecer uma escala que reflecte a probabilidade do risco; (2) delinear as consequências do risco; (3) estimar o impacto do risco no projecto e no produto e (4) anotar a precisão da previsão de risco, de modo que não haja mal-entendidos.

b) Tabela de risco

A tabela de risco (ver quadro 4 no anexo) fornece uma técnica simples para a previsão de riscos.

Para construir uma tabela de risco a equipa de software começa por listar todos os riscos na primeira coluna da tabela. Depois, cada risco é classificado na segunda coluna, como por exemplo risco de tamanho do projecto, risco de negócio etc. A probabilidade de ocorrência de cada risco é colocada na próxima coluna. O valor da probabilidade de cada risco pode ser estimado individualmente pelos membros da equipa. Cada membro da equipa opina sequencialmente até que sua avaliação da probabilidade de risco comece a convergir.

De seguida, passa-se para o impacto de cada risco. Após ter calculado o impacto de cada risco, determina-se o valor global do impacto. O impacto global é calculado a partir da média dos quatro componentes de risco: desempenho, apoio, custo e cronograma.

Após ter completado as quatro primeiras colunas a tabela é ordenada por probabilidade e por impacto. Os riscos com alta probabilidade e alto impacto vão para o alto da tabela e aqueles com baixa probabilidade e impacto ficam na parte de baixo e assim se consegue a priorização de riscos de primeira ordem.

De seguida é definida uma linha de corte na tabela ordenada. A linha de corte traçada horizontalmente em algum ponto da tabela sugere que apenas riscos situados acima da linha receberão atenção especial. Riscos que ficam abaixo da linha são reavaliados para se chegar a priorização de segunda ordem.

Por último a última coluna contem ponteiro para um plano de atenuação, monitoração e administração de risco ou seja para uma colecção de folhas de informação de risco desenvolvida para todos os riscos que ficam acima da linha de corte.

c) Avaliação do risco

A natureza do risco, seu escopo e sua época são três factores que afectam as consequências em que o risco pode ocorrer. A natureza do risco indica os problemas que podem surgir se ele ocorrer. Por exemplo, se uma interface externa for mal definida prejudicará o início do projecto e dos testes que vao provavelmente trazer problemas de integração do sistema no fim do projecto. A época de um risco considera quando e quanto tempo o impacto será sentido. E, por fim, o escopo de um risco mostra a severidade com sua distribuição geral (quanto do projecto será afectado ou quantos clientes serão prejudicados).

Para determinar as consequências gerais de um risco é conveniente seguir os seguintes passos:

1. Determinar o valor médio da probabilidade de ocorrência de cada componente de risco;
2. Determinar o impacto de cada componente;
3. Completar a tabela de risco e analisar os resultados, como descrito na secção anterior.

d) Exposição ao risco

Para se avaliar um determinado risco há que estabelecer valores reais para:

- O impacto: o prejuízo, ou efeito, sobre o projecto caso o risco ocorra;
- A probabilidade: probabilidade de ocorrência do risco;
- A data de ocorrência: período em que será necessária uma acção, a fim de atenuar o risco.

A exposição ao risco (risk exposure, RE) é um atributo do risco, derivado de dois outros: o impacto (perda) e a probabilidade. A exposição ao risco é definida pela seguinte relação (Boehm 1989 p. 6):

ER = Prob(Ri) × Perda(Ri)

Em que Prob(Ri) é a probabilidade de ocorrência de um risco (Ri) e perda(Ri) é a perda, para as partes afectadas, caso esse risco se materialize.

A exposição ao risco pode ser calculada para cada risco da tabela de risco, assim que é feita a estimativa de custo do risco. A exposição total do risco, para todos os riscos acima da linha de corte da tabela de risco pode fornecer um modo de ajustar a estimativa final quanto ao custo do projecto. Também pode ser usada para fazer uma previsão sobre o aumento provável do pessoal, necessário em vários pontos do cronograma do projecto.

	Probabilidade		
Impacto	Elevada	Média	Baixa
Catastrófico	Elevada	Elevada	Moderada
Crítico	Elevada	Moderada	Moderada
Marginal	Moderada	Moderada	Baixa
Negligenciável	Moderada	Baixa	Baixa

Figura 6 – Exemplo de Matriz de Exposição ao Risco

Para ilustrar esse processo de avaliação de risco vamos a um exemplo prático:

Na análise de risco relacionada com a reutilização de componentes no desenvolvimento de um software a equipa identificou que apenas 60% dos componentes programados para serem reutilizados é que serão integrados na aplicação. Os outros terão que ser desenvolvidos pela equipa. A probabilidade do risco ocorrer é de cerca de 70% .

Partindo do princípio que o número de componentes que deveria ser reutilizados no sistema é de 50 componentes, o que quer dizer que a equipa terá que desenvolver cerca de 20

componentes a partir do zero. Como o tamanho de um componente é de 80 loc e o custo para cada loc é de 500$00, o custo geral (impacto) para desenvolver os componentes seria 20 × 80 × 500 = 800.000$00. e a exposição ao risco (RE) =0.70 × 800.000 =560000.

Durante a avaliação de risco há que examinar melhor as estimativas que foram feitas na previsão de risco, ordenar os riscos que foram descobertos e começar a pensar sobre modos de controlar e até mesmo evitar riscos que são prováveis de ocorrer.

Na avaliação de risco há que realizar os seguintes passos:

1. Definir os níveis de referência de risco para o projecto;
2. Tentar desenvolver uma relação entre cada tripla (risco, probabilidade do risco, impacto do risco) e cada um dos níveis de referência;
3. Prever o conjunto de pontos de referência que definem uma região de encerramento limitada por uma curva ou áreas de incerteza;
4. Tentar prever como combinações compostas de riscos afectarão um nível de referência.

e) Refinamento de risco

Após a identificação de um risco este pode ser identificado de uma forma generalizadA. Mas com o andar do tempo, prosseguindo nos estágios de planeamento de risco, o gerente vai obtendo informações do projecto e do risco que lhe possa refinar o risco num conjunto de riscos mais detalhados. O objectivo desse refinamento é conseguir uma forma mais fácil de atenuar, monitorar e administrar os riscos.

Retomando o exemplo da secção 4.4.2 alinea d) podemos escrever:

Considerando que os componentes de software reutilizáveis devem satisfazer padrões de projectos específicos, e que alguns não satisfazem, então a preocupação de que possivelmente apenas 60% dos componentes planeados possam ser integrados no software em construção, o que quer dizer que há necessidade de fazer cerca de 40% dos restantes componentes sob medida.

Esta condição poderá ser refinada do seguinte modo:

Subcondição 1. Certos componentes reutilizáveis foram desenvolvidos por terceiros sem conhecimento dos padrões internos do projecto.

Subcondição 2. O padrão de projecto para as interfaces de componentes não foi consolidado e pode não estar de acordo com alguns componentes reutilizáveis existentes.

Subcondição 3. Certos componentes reutilizáveis foram implementados em uma linguagem que não está disponível no ambiente alvo.

Após o refinamento do risco, as consequências do risco continuam na mesma isto é, 40% dos componentes terão que ser construídoS sob medida, mas o refinamento ajuda a isolar os riscos subjacentes e poderá levar a uma análise e uma resposta mais fácil.

4.4.3 - Planeamento de riscos

Planeamento de riscos é o estágio de gestão de risco onde são traçados planos para enfrentar os riscos, seja evitando-os, seja minimizando seus efeitos sobre o projecto.

Os objectivos da fase de planeamento são (Miguel, 2002, p.91):

1. Assegurar que se conhecem as origens e as consequências dos riscos;
2. Desenvolver planos eficazes (as medidas certas para os riscos certos);
3. Desenvolver planos eficientes (somente as medidas necessárias ou que beneficiarão o projecto e/ou a organização);
4. Produzir, ao longo do tempo, o conjunto correcto de acções que minimizam os riscos e os respectivos impactos (custos e prazos), ao mesmo tempo que maximizam as oportunidades;
5. Dar prioridade aos riscos considerados mais importantes.

A maioria das equipes de software ainda conta apenas com estratégias reactivas a risco. Quando uma estratégia reactiva monitora o projecto para riscos prováveis, são reservados recursos para lidar com eles quando se tornarem problemas reais. “Em geral a equipe de software não faz nada a respeito de risco até que algo se dê errado. Então, a equipe corre para a acção, numa tentativa de corrigir o problema rapidamente. Isso é frequentemente chamado *modo de combate ao fogo*.” (Pressman, 2002, p.140). E, quando assim é, o projecto estará em perigo eminente.

Os riscos devem ser planeados por pessoas que possuem conhecimento, capacidade e recursos necessários para lidar com eles de forma eficaz. Na fase de planeamento de riscos são respondidas as seguintes questões: (1) Quem é responsável por um risco? (2) O que se deve fazer e em que medida?

As informações sobre o risco colhidaS na fase de identificação e análise de risco são transformadAs nesta fase em acções e decisões. Nesta fase são desenvolvidas acções para enfrentar os riscos individuais, ou conjuntos de riscos relacionados por forma a estabelecer prioridades para as acções e criar um plano integrado de gestão de riscos.

A estratégia mais inteligente para a gestão de risco é ser preventivo. Uma estratégia preventiva começa muito antes do trabalho técnico ser iniciado. Riscos potenciais são identificados, avaliadas as suas probabilidades, impacto e também são classificados por importância. De seguida, a equipe estabelece um plano para administra-los. O objectivo principal é evitar riscos, mas, como é impossível evitar todos, a equipa trabalha para desenvolver um plano de contingência que vai permitir responder de modo controlado e efectivo.

Após ter-se analisado e priorizado os riscos, deve ser feito um julgamento sobre quais são os riscos mais importantes a serem considerados durante o projecto.

Esse julgamento é o resultado de uma combinação da probabilidade do risco surgir e do impacto daquele risco. De um modo geral todos os riscos catastróficos devem sempre ser considerados como também todos os riscos sérios que tenham mais do que uma probabilidade moderada de vir a ocorrer.

O número certo de riscos a ser monitorado depende do projecto e pode ser no total de cinco ou quinze. Embora haja quem recomenda identificar e monitorar os dez maiores riscos. Mas o mais importante é que o número de riscos escolhido deve ser gerenciável. Um número muito grande de riscos simplesmente poder-se-ia exigir que muitas informações fossem colectadas.

A partir do quadro 5, em anexo, poder-se-ia considerar os cinco riscos que podem ter consequências catastróficas ou sérios.

a) Estratégias de mitigação do risco

A pessoa que assumiu a responsabilidade do risco, é que deve decidir qual a estratégia a seguir para a sua mitigação. A determinação de uma estratégia de mitigação deve prosseguir os seguintes objectivos:

- Assegurar que se sabe o suficiente para se tomar uma decisão informada.
- Escolher a abordagem adequada para uma gestão eficaz dos riscos.
- Estabelecer objectivos de mitigação mensuráveis, de modo a providenciar uma meta que possibilite a avaliação do sucesso e uma orientação para o desenvolvimento dos planos de acção.

O quadro 6, em anexo, mostra possíveis estratégias que foram identificadas para os principais riscos, mostrados no quadro 5, em anexo.

Todas as actividades de análise de risco têm um único objectivo a atingir que é ajudar a equipe de projecto a desenvolver uma estratégia para lidar com o risco. É nesta óptica que após ter analisado os riscos mais importantes a equipa deve definir estratégias para geri-los. Estas estratégias podem ser dividdam em três pontos:

- *Estratégias preventivas* – são estratégias que reduzem a probabilidade de o risco surgir. Seguir estas estratégias significa que a probabilidade do risco surgir será reduzida. Um exemplo é a utilizada para lidar com componentes defeituosos, mostrado no quadro 6.
- *Estratégias de minimização* – são estratégias que diminuem o impacto do risco. Seguir estas estratégias significa que o impacto do risco será reduzido. Um exemplo é a utilizada quando alguém está doente, mostrado no quadro 6.
- *Planos de contingência* - são estratégias prontas para lidarem com o risco, caso acontecer. Seguir essas estratégias significa que, se o pior acontecer, a equipa estará preparada e tem uma estratégia pronta par lidar com o caso. Um exemplo de planos de contingência é a estratégia aplicada a problemas financeiras organizacionais, mostrado no quadro 6.

b) Mitigação de grupos de riscos relacionados

Ao analisar um conjunto de riscos deve-se procurar um conjunto de coisas fundamentais:

- Causas e efeitos, para identificar causas raiz ou efeitos comuns que necessitam ser evitados;
- Inter-relações entre riscos e causas, isto é, ciclos de relações. Como por exemplo: **A** causa **B** causa **C** causa **D** causa **A** – que podem ser quebrados ou redefinidos.

A mitigação para um conjunto de riscos é consideravelmente mais complexo que para um único risco. Mas a mitigação de um conjunto de riscos inter-relacionados traz muitas vantagens principalmente no que toca a rentabilidade dos planos de mitigação, através de eliminação de esforços duplicados. Também evita objectivos e acções de mitigação conflituantes. Por outro lado integra os esforços do planeamento e evita tempo desnecessário no desenvolvimento de planos.

Por estas razões é muito conveniente fazer agrupamento de riscos para poder fazer a mitigação. Isto principalmente quando se tem uma grande quantidade de riscos, visto que são mais facilmente mensuráveis através da sua classificação em grupos relacionados.

Isto tudo faz com que os responsáveis pela mitigação desses grupos de riscos tenham conhecimento o suficiente sobre esses riscos por forma a estabelecer as suas relações, causas, consequências e os objectivos de mitigar o conjunto de riscos.

4.4.4 - Monitoração de riscos

A monitoração de riscos consiste em avaliar constantemente os riscos e revisar os planos para a diminuição de riscos, a medida que mais informações sobre eles se tornam disponíveis. São avaliados regularmente cada um dos riscos identificados, a fim de decidir se esse risco está se tornando mais ou menos provável bem como se os efeitos desses riscos se modificaram.

É claro que isso não pode ser observado de modo directo. Portanto, devem ser examinados outros factores, para serem levantados indícios sobre a probabilidade e seus efeitos. Esses factores, como é obvio, são dependentes do tipo de risco.

A monitoração de riscos deve ser um processo contínuo durante todo o processo de desenvolvimento de software onde cada um dos riscos principais deve ser considerado separadamente e discutido numa reunião.

A medida que o projecto avança, as actividades de monitoração de riscos iniciam. O gerente do projecto monitora factores que podem indicar se o risco é mais ou menos provável. No caso de alta rotatividade de pessoal os seguintes factores podem ser considerados:

- Atitude geral dos membros da equipa com base nas pressões do projecto;
- Grau com que a equipa se aglutinou;
- Relacionamento interpessoal entre os membros da equipa;
- Problemas em potencial com remuneração e benefícios;
- Disponibilidade de emprego dentro da empresa e fora dela.

A monitorização tem como objectivo controlar a evolução dos riscos e das acções tomadas para os mitigar ou anular. Para tal há que recolher informação precisa, relevante e oportuna sobre os riscos, apresenta-la de um modo claro, compreensível e adequado à pessoa a quem é destinado o relatório da situação (ver quadro 7, em anexo).

a) Metodologias de monitorização dos riscos

Na literatura da gestão do risco podemos encontrar muitas metodologias e ferramentas desenhadas para monitorizar riscos, que utilizem as abordagens gerais destinadas à gestão de projectos. O quadro 8, em anexo, resume esses métodos e ferramentas que são utilizados para suportar cada uma das actividades da fase de monitorização.

b) O encerramento de riscos

O processo de gestão de riscos é um processo interativo que continua ao longo do projecto. A cada interação as informações dos riscos aumenta-se, possibilitando a tomada de decisão acerca de um determinado risco ou seja alguns riscos podem ser encerrados ou podem ser adoptadas estratégias para as mitigar.

A tomada de decisão do enceramento de riscos deve ser do responsável pelo risco. Este deverá notificar o pessoal que originou o risco sobre a tomada de decisão de encerramento. Também o encerramento do risco deve ser assinado pelo chefe do projecto ou de um elemento responsável dentro da equipa.

Um outro aspecto a ter enconta no processo de encerramento do risco é se ele faz parte de um grupo de risco, para poder tomar a decisão se deve encerrar o grupo completo ou apenas alguns riscos seleccionados dentro do conjunto.

Após o encerramento, as lições aprendidas devem ser discutidas e documentadas com o processo de observação ou mitigação. Esta informação poderá ser muito importante para o actual projecto bem como para futuros projectos da empresa. Dentro dessas informações que

devem ser retidas destacaremos: os planos de mitigação falhados e razões para a sua falha; relações e dependências de riscos que não eram óbvias; planos de mitigação bem sucedidos acompanhados das razões para o sucesso e dados relevantes da análise, especialmente os custos e benefícios do plano de mitigação.

CAPÍTULO V - CARACTRIZAÇÃO DAS DIFICULDADES NO DESENVOLVIMENTO DE SOFTWARE

5.1 - INTRODUÇÃO

Fazer software não é tarefa fácil. Fazer software de qualidade e entrega-lo dentro dos prazos previstos é ainda mais difícil.

Muitas das causas das aflições que surgem no desenvolvimento de software têm origem numa série de mitos que propagam confusões e mal-entendidos.

Actualmente, devido à existência de projectos cada vez mais complexos e com mais pessoas envolvidas, a gerência tem uma participação fundamental para o sucesso do projecto. Mas apenas as habilidades pessoais da gerência não são suficientes para o êxito do projecto. É necessária a utilização de ferramentas que automatizem determinadas actividades e facilitem a comunicação entre os membros da equipa.

O desenvolvimento de software, ao longo do tempo, tem apresentado padrões de baixa qualidade, de custos e prazos completamente ultrapassados. Mas não obstante, os obstáculos inerentes ao desenvolvimento de software tornam extremamente relevantes as várias iniciativas que possam ser desenvolvidas com o objectivo de ultrapassar estas dificuldades. Por isso vale a pena fazer uma análise dos diversos esforços que foram efectuados ao longo do tempo, e perceber porque alguns não foram totalmente efectivos na resolução dos problemas, enquanto outros, bem sucedidos, são apontados como melhores práticas a aplicar sistematicamente.

Existem diversas abordagens de desenvolvimento de sistemas. Diferentes tipos de problemas e desafios possuem características diferenciadas que requerem diferentes tipos de abordagens. O maior desafio está em escolher, adaptar e integrar esta abordagem, com as características

presentes num determinado ambiente. O desenvolvimento de software requer o uso de uma abordagem mais moderna associada à gerência de projectos, que responda a demanda de um ambiente distribuído. A globalização da economia tem levado muitas organizações a distribuírem geograficamente seus recursos e investimentos visando obter melhores resultados como maior produtividade, redução de custos, minimização dos riscos e melhoria na qualidade. Aliada a essas vantagens surge um novo problema no desenvolvimento de software, que envolve principalmente a distância física entre os participantes do processo. Desta forma, os já tradicionais problemas inerentes ao processo de desenvolvimento, fortemente centrado nas fases de especificação de requisitos e análise de sistemas, ganham contornos mais críticos. Para isso é necessário adoptar linguagens de especificação e processos de desenvolvimento mais formais e definidos. Caso contrário essa equipa distribuída pode fracassar.

Esse fracasso pode ser provocado pelos seguintes factores: falta de coordenação, dispersão geográfica, perda do espírito de equipa e diferenças culturais. Mas também existem alguns factores que podem levar uma equipa distribuída ao sucesso, tais como: Infra-estrutura de comunicação, arquitectura do produto, construção de uma equipa, metodologia de desenvolvimento, tecnologia de colaboração e técnicas de gerência.

5.2 - MÉTODOS DE GESTÃO

5.2.1 - Experiência do Chefe de Projecto

Segundo (Miguel 2002, p.13) o chefe de projecto deve orientar-se para o cliente e deve possuir a necessária autoridade, para poder operar eficazmente. É nesta óptica que o chefe de projecto deve ter as seguintes características para atacar com profissionalismo as diversas dificuldades inerentes a um projecto de software:

- Possuir uma profunda compreensão dos objectivos do projecto;
- Ser capaz de compreender as necessidades do seu pessoal;
- Ter uma boa reflexão sobre os detalhes;
- Estar fortemente comprometido com o projecto;
- Ser capaz de lidar com desaires e desilusões;

- Possuir boas aptidões de negociação, na medida em que uma grande parte da vida do projecto é gasta a tentar adquirir recursos;
- Ser prático e orientado para os resultados;
- Estar consciente dos custos e possuir aptidões de gestão básicas;
- Ser politicamente sensato – isto é, estar consciente daquilo que se deve ou não fazer;
- Possuir uma elevada tolerância à ambiguidade.

Os projectos funcionam com base nas datas e orçamentos fixos assim como especificações cuidadosamente concebidas como parte dos objectivos fundamentais. Independentemente das incertezas e dificuldades no desenvolvimento de software, o chefe de projecto deve ser capaz de traçar estratégias para contrariar as forças que visem impedir que os objectivos sejam alcançados.

Diversos problemas do desenvolvimento de software, são resultantes da omissão ou do mau uso de metodologias e técnicas adequadas. Conforme (vitiello 2001), os consumidores estão demandando rapidez no mercado porque o tempo de um projecto afecta as operações nos negócios. Eles exigem execução sem falhas para concretizar oportunidades de negócios. Ainda o mesmo autor destaca que o gerente de projectos tem que ter algumas habilidades para obter sucesso em suas actividades, entre elas destaca: a liderança, a comunicação, a resolução de conflitos, a negociação, a habilidade de escutar e uma boa gestão de relacionamento.

5.2.2 - Ambiente da Equipa

Hoje vivemos num mercado de concorrência recheado de ofertas dando vantagens à liberdade de escolha aos consumidores de produtos com preços cada vez mais baixo, acarretando dificuldades aos produtores que labutam para superar a concorrência e tentar atingir os seus objectivos, oferecendo produtos de maior qualidade. Contudo, o mercado exige que as empresas de desenvolvimento de software tenha um espírito inovador, aliada a capacidade de traçar estratégias que lhes permite fazer face às dificuldades inerentes ao desenvolvimento de software. É neste sentido que as empresas têm vindo a instituir-se transformações radicais nos seus modos de operação de forma a ser mais competitivas. Transformações essas como a diminuição dos níveis de gestão intermédia na medida em que foi percebido que a gestão intermédia acrescenta pouco valor às operações e contribui apenas para a manutenção da

burocracia. É neste sentido que muitas empresas estão a reestruturar as suas forças de trabalho, com o objectivo de eliminar os múltiplos níveis de burocracia que separam o(a) director(a) Geral do pessoal de limpeza. Fazendo com que os empregados lidam cada vez menos com chefes e subordinados claramente definidos, mas sim, com colegas sobre quem não tem controlo directo. Permitindo desta forma um ambiente onde as decisões tenham que ser tomadas geralmente através de consenso.

Esta eliminação de cadeiras de gestão intermédias traz uma redução da dimensão da empresa. Com isso as empresas ganham vantagens na luta contra a burocracia, redução de custos, aumento de produtividade e criação de um clima estável e afectivo entre o pessoal.

Uma outra transformação no seio das empresas tem a ver com a atribuição de poder aos empregados nos seus negócios com clientes. Como por exemplo, se um cliente quiser alterar a configuração de um dado equipamento, o empregado tem autoridade para garantir a alteração, se tal justifica. Isto é feito com o objectivo de acelerar o processo de decisão e, em simultâneo, satisfazer o cliente. Esta forma de exercício do poder altera o papel de gestor que passa da função de director de actividades para a função de suporte das actividades, isto é, o papel dos gestores é fazer o que for necessário para permitir que os seus empregados operem o mais eficaz possível.

E por último e não menos importante temos a subcontratação de serviços de produção.

Actualmente muitas empresas de desenvolvimento de software têm dependências cada vez mais de outras empresas para a consecução das suas operações. Isto é feito com o objectivo de reduzir os custos de investimentos em novos equipamentos e instalações, menos encargos com pensões, seguro de vida, saúde e o decréscimo da necessidade de recrutar e despedir pessoas para responder aos ciclos do mercado. Para além disso permite uma maximização de tempo de desenvolvimento, de forma que a conclusão de um produto não coincida com a sua desactualização provocada pela concorrência e alterações dos requisitos.

Um outro aspecto que merece destaque no domínio de desenvolvimento de software pelas implicações que tem na gestão de projectos é a globalização do desenvolvimento. Fazendo com que os gestores de desenvolvimento de projectos de software estejam sujeitos a ambiente mais complexos. Daí há que estabelecer alianças estratégicas e desenvolvimento em conjunto com outras divisões da mesma companhia. Cada um dos quais com os seus conjuntos de necessidades que exigem métodos especiais de organização e controlo. Esta situação torna mais complexa a gestão do risco desses projectos de desenvolvimento.

5.3 - DEFINIÇÃO DO PRODUTO A DESENVOLVER

5.3.1 - Especificações

A característica mais importante do planeamento é precisamente a listagem daquilo que não sabemos, porque o que precisamos descobrir é precisamente o que não sabemos. Por isso há que ter capacidade para reconhecer riscos e de conseguir traçar estratégias para os minimizar ou mesmo combate-los. Nesta fase não temos que ter respostas para todos os problemas mas sim saber que eles existem e planear a forma de os atacar mais tarde.

5.3.2 - A dificuldade de medição

As pessoas que dizem como um trabalho deve ser feito não são as mesmas que executam o trabalho. De facto, os líderes precisam de alguma forma de medir o quão efectivo são os trabalhadores.

Isto é particularmente relevante para software, devido à dificuldade de aplicar medições. Mesmo com nossos melhores esforços, somos incapazes de medir até mesmo as coisas mais simples a respeito de software - como por exemplo, produtividade.

Introduzir gestão por medidas, sem boas medidas, leva aos seus próprios problemas. Problemas esses que impedem a equipa de conseguir alcançar os seus objectivos quanto à qualidade do software em construção.

5.4 - PROCESSO DE DESENVOLVIMENTO

5.4.1 - Especificações de requisitos

As especificações de requisitos são um trabalho imprevisível ou seja estão sempre a mudar. Esta é uma realidade que não pode surpreender a ninguém. Isto porque podemos considerar que mudanças de requisitos de negócio ao construir software constituem uma regra. Agora o problema é o que devemos fazer a esse respeito.

Uma proposta a seguir na engenharia de requisitos é a obtenção de uma visão completamente clara dos requisitos antes de começar a construir o software, obter uma aprovação do cliente, e depois implantar procedimentos que limitam mudanças nestes requisitos.

Um problema com isto é que apenas entender as opções para os requisitos já é difícil. E mais difícil ainda é estabelecer as estimativas por vários motivos: 1º desenvolvimento de software é

uma actividade de design, portanto difícil de planear ou custear. 2º Os materiais básicos mudam rapidamente. 3º Dificuldade de previsão e quantificação do pessoal envolvente.

A natureza intangível do software também contribui. É muito difícil ver o valor de uma funcionalidade do software até que ela seja materializada. Apenas quando se vê uma versão preliminar do software é que realmente se começa a entender as funcionalidades valiosas e as que não são.

Mesmo que se concorda e realmente se consegue um conjunto preciso e estável de requisitos, ainda assim isso provavelmente não seria suficiente. Na economia de hoje, as forças de negócio estão mudando o valor dos requisitos de software muito rapidamente. O que pode ser um bom conjunto de requisitos agora, não será um bom conjunto de requisitos em 1 ano. Mesmo que os clientes possam fixar seus requisitos, o mundo dos negócios não vai parar para eles. E as mudanças no mundo dos negócios são completamente imprevisíveis.

Tudo mais no desenvolvimento de software depende dos requisitos. Se não for possível obter requisitos estáveis, então não será possível ter um plano de projecto estável.

5.5 - RECURSOS

Nos últimos anos, as organizações de desenvolvimento de software têm aumentado sua percepção em relação aos problemas que tipicamente as tocam. Software com bugs, prazos e orçamentos não cumpridos assim como insatisfação de clientes e usuários são eventos muito mais frequentes do que se desejaria. Todos estes problemas muitas vezes estão em grande parte relacionados ao facto de que no desenvolvimento de software se utilize métodos improvisados pelos desenvolvedores, os quais, por sua vez dependem muito mais do seu talento individual do que de uma sólida formação acompanhada de métodos formais que dirijam suas actividades. E se escolher alguma metodologia poderá ter a dificuldade de saber escolher a melhor para o projecto em si. Pois muitas vezes estas escolhas são feitas não em função daquele que melhor se adequa ao projecto mas sim daquele que a equipa é capaz de saber usar com menos dificuldade se é que se sabe usar alguma. Estas conclusões são reflexos do estudo efectuado sobre a realidade Cabo-verdiana que revela dificuldades de alguns chefes de equipa de desenvolvimento de software em responder quais são as metodologias utilizadas no desenvolvimento de software.

A ausência duma metodologia adequada no processo de desenvolvimento de software arrasta consigo sérios riscos que reflectem directamente na baixa qualidade do produto fazendo com que se torna difícil a sua manutenção às novas alterações dos requisitos face a dinâmica do desenvolvimento crescente do mercado.

CAPÍTULO VI CAUSAS E CONSEQUENCIAS DE RISCO E INCERTEZA

Este capítulo, apresenta o resultado de uma análise dos riscos que teve por objectivo tentar equacionar um conjunto de causas que provavelmente poderá causar algum risco e o seu respectivo impacto no desenvolvimento de projecto de software. Conforme vimos no capitulo IV, o risco no inicio é declarado de uma forma geral mas com o tempo vem aparecendo novas informações sobre o risco e poderá ser dividido em vários riscos diferentes com o objectivo de encontrar as melhores estratégias para lidar com eles. O conjunto de riscos resultantes dessa divisão do risco, em certos casos poderão ser considerados as causas que tenham originado o seu aparecimento.

Como é obvio, um risco poderá dar origem a um ou vários riscos. Dizer isso é dizer que o risco pode ter como consequência um conjunto de riscos e vice-versa. Se assim é, é natural dizer que existe alguma relação entre vários riscos dum projecto de software.

A causa de um risco pode ser devido a incerteza de um evento[6] ou à incerteza de uma estimativa. Devemos encarar ambas as causas de modo diferente, pois as repostas a elas devem ser igualmente diferentes.

Para analisar as causas e consequências de risco e incerteza escolhemos cinco riscos que melhor se adequam para este estudo quanto as suas frequências e perigosidades para o projecto de desenvolvimento de software.

6.1 - Alteração de requisitos

As possíveis causas que pode levar a alteração de requisitos no processo de desenvolvimento de software são de natureza diversa entre as quais podemos destacar: inexperiência de utilizadores, inexperiência de gestores, metodologias e estimativas de custos inadequadas.

[6] Os eventos que estamos a falar são eventos que não desejamos que aconteçam (riscos)

Para além dessas causas temos mais duas causas (fricção com utilizadores e fricção com gestores) que poderão alterar os requisitos provocando fricção com utilizadores e gestores.

A alteração de requisitos poderá trazer consequências para o projecto nomeadamente: prazos não cumpridos, derrapagens orçamentais, atrasos na comercialização, pressão excessiva sobre prazos e enfraquecimento da moral da equipa.

6.2 - Pressão excessiva sobre os prazos

Dentro das possíveis causas que pode levar a pressão excessiva sobre os prazos podemos apontar os seguintes: alteração de requisitos dos utilizadores, estimativas inadequadas, medições, planeamento e práticas de gestão incorrectas.

A pressão excessiva sobre os prazos podem ter várias consequências de vária ordem entre os quais destacamos os seguintes: fraca moral da equipa, elevada rotatividade de pessoal, projectos cancelados, derrapagens orçamentais e baixa qualidade do produto final.

6.3 - Metas não cumpridas

Dentro das possíveis causas que pode provocar o risco do não cumprimento das metas podemos apontar: inexperiência dos gestores, inexperiência do pessoal, alteração de requisitos dos utilizadores, estimativas inadequadas, medições e planeamento incorrectos.

Por seu lado, se as metas não forem cumpridas, isto podem causar consequências drásticas para o projecto e para a equipa de desenvolvimento, nomeadamente: fricção com utilizadores, fricção com gestão sénior, fraca moral da equipa, projectos cancelados, derrapagens orçamentais e baixa qualidade do produto final.

6.4 - Derrapagens orçamentais

Das possíveis causas que podem provocar derrapagens orçamentais, podemos apontar as seguintes: práticas incorrectas de gestão; pessoal inexperiente; alteração de requisitos do utilizador; estimativas incorrectas; planeamento inadequado e pressão excessiva sobre os prazos.

Para além dessas causas destacaria mais três que podem levar a derrapagens orçamentais e vice-versa como: projectos cancelados, prazos demasiado longos e metas não cumpridas.

As derrapagens orçamentais poderão originar consequências muito sérias nomeadamente: fricção com utilizadores; fricção com gestão sénior; atritos entre pessoal e fraca moral da equipa.

6.5 - Baixa qualidade

Das possíveis causas que podem fazer com que o software tenha baixa qualidade, podemos destacar as seguintes: inexperiência dos gestores e do pessoal; alteração de requisitos do utilizador; estimativas e planeamento incorrecto; correcção deficiente de erros e pressão excessiva sobre os prazos.

Neste ponto destacaremos mais duas causas que podem levar o software a ter baixa qualidade e vice-versa, como a baixa produtividade e prazos demasiados longos.

A baixa qualidade poderá provocar: uma insatisfação enorme no seio da equipa por não ter conseguido alcançar os objectivos no que toca a qualidade do produto; fricção com utilizadores; baixa satisfação dos utilizadores; fricção com gestão sénior; elevados custos de manutenção e fraca moral da equipa.

CAPÍTULO VII OS RISCOS MAIS COMUNS E MAISPERIGOSOS NO DESENVOLVIMENTO DE SOFTWARE EM CABO VERDE

Este capítulo apresenta o resultado de um levantamento que fizemos sobre os riscos e as dificuldades que as empresas de desenvolvimento de software enfrentam em Cabo Verde.

Para a obtenção dos dados foi aplicado um inquérito as empresas e depois fez-se os cálculos com base na média aritmética.

O estudo do caso que fizemos acerca dos riscos e dificuldades no desenvolvimento de software em Cabo Verde revelou-nos que os riscos mais frequentes no desenvolvimento de software são:

1. Pressão excessiva sobre os prazos;
2. Inexperiência de utilizadores;
3. Baixa produtividade;
4. Inexperiência do pessoal;
5. Má estrutura organizacional;
6. Práticas incorrectas de gestão;
7. Inexperiência de gestores.

No que toca aos riscos mais perigosos no desenvolvimento de software em Cabo Verde, destacam-se:

1. Estimativas inadequadas de custos;
2. Derrapagens orçamentais;
3. Fricção entre pessoal-empresa, cliente-fornecedor;
4. Baixa qualidade;

5. Práticas incorrectas de gestão;
6. Correcção deficiente de erros;
7. Inexperiência de gestores;
8. Estimativas inadequadas.

Em Cabo Verde as equipas de desenvolvimento de software têm uma taxa de insucesso de 19.3%. O orçamento é ultrapassado em cerca de 42%, sendo 22% do orçamento gastos em alterações e correcções. No que concerne aos projectos que incluíam algo que as organizações não tinham experiência a percentagem é de 18.8%. Também cerca de 11.1% dos projectos não tinham planos formais para testes e instalação do software. Dos projectos de software desenvolvidos cerca de 28.6% não chegaram ao seu término.

Nos projectos não contratados já desenvolvidos os utilizadores têm alterado os requisitos em cerca de 25.7%. Quanto ao cumprimento dos prazos, 22.8% dos projectos não foram cumpridos e o custo é ultrapassado em cerca de 22%. Cerca de 12% dos softwares desenvolvidos têm uma baixa qualidade e 10.3% têm um controlo de configuração inadequado.

Quanto aos projectos contratados, os custos de manutenção são cerca de 15%. E cerca de 2.7% dos projectos têm verificado atritos entre o cliente e a empresa contratada. Os utilizadores têm alterado os requisitos em cerca de 20%.

Em Cabo Verde cerca de 83% das organizações que desenvolvem software já entregaram pelo menos um software fora do prazo estabelecido.

Os estudos mostram que no nosso país sempre tem havido reconhecimento do trabalho realizado pelos membros das equipas e há um bom espírito de equipa no seio das organizações. Os gestores / chefes de equipa sempre envolvem os membros das equipas nas tomadas de decisões e também sempre controlam as alterações das especificações.

A fig. 8 mostra os indicadores das equipas de desenvolvimento de software numa escala de 0 a 4 pontos.

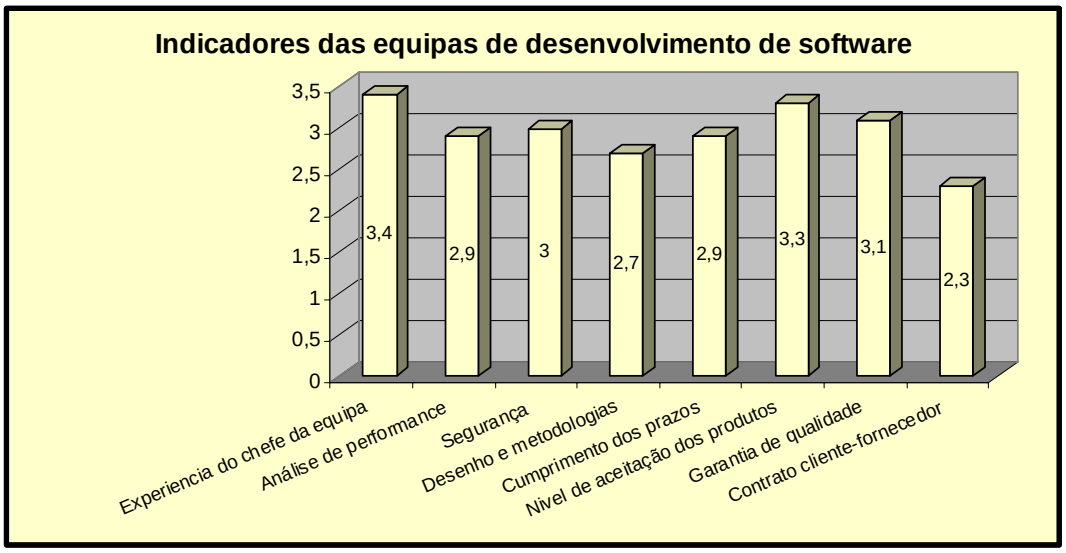


Fig. 7 – Indicadores das equipas de desenvolvimento de software

A fig. 7 mostra que em Cabo Verde, no que toca à experiência dos chefes de projecto, a maioria deles considera que tem uma boa experiência e uma boa análise de performance no desenvolvimento de software. Demonstra também que os produtos das organizações de desenvolvimento de software em Cabo Verde têm uma boa garantia de qualidade e um bom nível de aceitação no mercado. O mesmo não acontece com o estabelecimento de contrato cliente-fornecedor, bem como com o desenho e as metodologias de desenvolvimento de software, em que as empresas têm algumas dificuldades.

A dificuldade em estabelecer o contrato cliente-fornecedor é provocada pela falta de credibilidade nas capacidades técnicas de gestão e planeamento, aliada à concorrência com outros softwares importados. Isto faz com que as propostas financeiras fiquem aquém dos custos reais de desenvolvimento. A dificuldade no desenho e metodologias justifica-se por um lado pela falta de recursos humanos e financeiros para poder suportar a implementação de determinados processos pesados. Por outro lado é que as metodologias geralmente não são adequadas às realidades das organizações cabo-verdianas.

CONCLUSÃO

O processo de desenvolvimento de software deve ser bem definido, eficiente, controlado, medido e gerido. Para alcançar tudo isso é necessário que se utilize algum modelo de processo de software. Cada modelo representa uma tentativa de colocar ordem numa actividade inerentemente caótica.

É importante salientar que as vantagens dos modelos descritos neste trabalho são reais, mas não constituem panaceia que pode resolver todos os problemas do desenvolvimento de software. Desenvolver software é uma tarefa complexa. Cada projecto é diferente, isto porque os requisitos, as tecnologias necessárias e as pessoas envolvidas são diferentes. O que quer dizer que os modelos e as metodologias a serem adoptados poderão também ser diferentes.

O objectivo de um modelo de processo de software é proporcionar ao projecto uma estrutura que reduza a exposição aos vários tipos de riscos como: ultrapassar o orçamento, produzir o sistema errado e que nunca funcione, etc. se mantenha a um nível considerado aceitável.

Após a análise do estudo do caso que fizemos sobre o desenvolvimento de software em Cabo Verde, constatamos que muitas das organizações que desenvolvem software não usam nenhum modelo formal. Isto ocorre porque estes geralmente não são adequados às realidades das organizações cabo-verdianas. As organizações pequenas e médias não possuem recursos suficientes para adoptar o uso de processos pesados. Esta falta de sistematização na produção de software terá como resultado a baixa qualidade do produto, além de dificultar a entrega do software nos prazos e custos predefinidos e inviabilizar a futura evolução do software.

A estruturação e sistematização de um projecto de desenvolvimento de software são indispensáveis para a sua gestão. De um projecto sem estrutura não se pode fazer estimativas sobre o seu custo ou a sua qualidade, não se pode definir pontos críticos e o seu progresso não pode ser monitorizado. E por isso, para evitar o fracasso de um projecto, a estruturação deve ser um dos pontos essenciais a ter em conta. Isto se consegue com a utilização de um modelo

de processo adequado ao projecto em si, com o objectivo de reduzir o risco e a incerteza e aumentar a eficácia das decisões de gestão para com o projecto.

Nos últimos anos, as organizações de desenvolvimento de software têm aumentado sua percepção em relação aos problemas que tipicamente as tocam: Software com bugs, prazos e orçamentos não cumpridos, insatisfação de clientes e usuários. Estes problemas estão, em grande parte, relacionados com o desenvolvimento de software que, muitas vezes, é realizado através de métodos improvisados pelos desenvolvedores, os quais, por sua vez, dependem muito mais de seu talento do que de uma sólida formação acompanhada de métodos formais para dirigir as suas actividades.

A implantação das metodologias em algumas organizações tem tido resultados significativos, mas em outros nem por isso. Entre os motivos que levaram ao relativo insucesso das metodologias de desenvolvimento de sistemas, é o facto de estes darem mais relevância às actividades de engenharia, em detrimento das actividades de gestão. Isto porque as boas técnicas de desenvolvimento não adiantam se o projecto como um todo é mal conduzido.

Na realidade cabo-verdiana, as metodologias ainda não conquistaram o seu lugar. Cerca de 50% dos desenvolvedores ainda estão a construir software usando o método de codificar e consertar. Os métodos formais de desenvolvimento não são muito utilizados.

O processo de gestão de riscos, como todos os outros planeamentos de projecto, devem funcionar de forma interactiva ao longo de todo o projecto.

O risco faz parte de qualquer actividade humana, não podendo ser eliminado na plenitude. O risco, em si, não é mau, é essencial ao progresso e o insucesso constitui muitas vezes, uma componente fundamental da aprendizagem. Para tal, devemos aprender a equilibrar as possíveis consequências negativas do risco, com os benefícios da respectiva oportunidade associada.

Após a realização do estudo do caso em Cabo Verde, podemos concluir que os riscos mais frequentes no desenvolvimento de software em Cabo verde são: pressão excessiva sobre os prazos; inexperiência de utilizadores; baixa produtividade e inexperiência do pessoal. Quanto aos riscos mais perigosos temos: estimativas inadequado de custos; derrapagens orçamentais; baixa qualidade e fricção entre pessoal empresa cliente, fornecedor.

Para além desses riscos mais comuns e mais perigosos que as empresas de desenvolvimento de software estão sujeitas no nosso país, ainda deparam com outros problemas nomeadamente

a falta de credibilidade nas capacidades técnicas de gestão e planeamento. Um outro problema prende-se com a falta de recursos para adquirir as ferramentas actualizados fora do país para lhes ajudar no desenvolvimento de um produto de qualidade capaz de fazer face a concorrência dos softwares importados. Isto faz com que as propostas financeiras fiquem aquém dos custos reais de desenvolvimento. Um outro problema que as empresas nacionais enfrentam é a inexistência de uma cultura de assistência técnica pós venda por contrato.

RECOMENDAÇÕES

Depois de ter feito um análise profundo sobre o resultado do estudo realizado sobre os riscos de desenvolvimento de software no nosso país e o seu impacto sobre o mercado, estamos em condições de fazer algumas recomendações à aqueles que desenvolvem ou que pretendem desenvolver software de forma a alerta-los a tomar consciência das reais dificuldades e o risco que poderão vir a enfrentar no desenvolvimento de software no nosso país.

1º - Hoje em dia, a desenfreada penetração de softwares de gestão importados tem influído negativamente no desenvolvimento de softwares nacionais por se tratar de um mercado pequeno desprovido da cultura tecnológica. Face a esta avalancha de importação alimentada pela falta de credibilidade nas capacidades técnicas dos desenvolvedores nacionais recomenda-se desenvolver software para satisfazer situações pontuais e sempre por encomendas dos clientes.

2º - É urgente que as empresas nacionais de desenvolvimento de software façam um planeamento adequado antes de começar a desenvolver software e que utilizem métodos formais de desenvolvimento. Só assim poderão vencer a concorrência, evitar o desperdício de esforços em projectos de software que poderão ficar desactualizados antes da sua conclusão, desenvolver produtos que o cliente não quer e com o risco de ficar pelo caminho motivado por uma análise de risco inadequado. Tudo isto terá reflexos positivos na conquista da credibilidade do mercado, fácil manutenção e evolução do software.

2º - Recomenda-se aos desenvolvedores de software, que para cada projecto, antes de escolher um modelo de processo, deve-se avaliar o nível de complexidade, confiabilidade e tamanho do sistema, de forma a escolher o melhor que adequa as suas características. Entre as quais destacaremos as características de incerteza no início ou da falta de conhecimento acerca do sistema a ser desenvolvido, de modo a dar ao projecto uma estrutura que reduza os riscos que põem em causa o seu sucesso. É neste âmbito que não se pode ignorar o risco e esperar que nada de grave aconteça.

3º - Durante o planeamento do futuro da empresa, é necessário que a administração garanta que todos os cuidados foram tomados para que os planos se concretizem. Para tal há que fazer uma análise de risco para poder saber quais os controlos que devem ser implementados em curto, médio e longo prazo.

4º - Para analisar riscos há que estudar minuciosamente os processos de negócios que sustentam a organização de forma a conhecer aquilo que se quer proteger e avaliar as vulnerabilidades dos componentes de tecnologia relacionado a cada processo que possa comprometer os objectivos da operação.

5º - Recomenda-se as instituições de ensino superior a inclusão das metodologias ágeis nos currículos dos cursos de informática. Isto seria benéfica tanto para o meio académico que iria mostrar aos alunos novas práticas de desenvolvimento de software e prepará-los para um mercado em expansão, quanto para as comunidades de programação e a indústria que carecem de profissionais capacitados na aplicação de metodologias ágeis para acompanhar o dinamismo exigido pela economia actual.

TRABALHOS FUTUROS

Alargar o estudo a todas as empresas de desenvolvimento de software em Cabo verde de forma a recolher dados que permitam dizer em que nível de maturidade de desenvolvimento de software as empresas se encontram. Trabalho este que não é fácil devido à burocracia que ainda hoje, infelizmente persiste no nosso país. Por outro lado, as empresas, com o objectivo de deixar transparecer uma boa imagem, não divulgam os seus pontos menos fortes, limitando simplesmente aos seus pontos fortes por vezes um pouco desajustos à sua própria realidade.

GLOSSÁRIO

Software - são programas de computador e toda a documentação associada e os dados de configuração necessários para fazer com que esses programas operem correctamente.

Processo de software – é aplicação de uma abordagem sistemática, disciplinada e possível de ser medida para o desenvolvimento e manutenção do software

Modelo de processo de software - é uma descrição simplificada de um processo de software, que é apresentada a partir de uma perspectiva especifíca. Os modelos, pela sua natureza, são simplificações; e, assim, um modelo de processo de software é uma abstracção do processo real que está sendo descrito.

Engenharia de software - é uma disciplina da engenharia que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema, após ter entrado em operação.

Métodos - são meios organizados de produzir software. Eles incluem sugestões sobre o processo a ser seguido, as regras que regem as descrições de sistema produzidas e as directrizes do projecto.

Risco - é um contexto que inclui as ameaças, vulnerabilidade e o valor a proteger.

Gestão de risco – é um processo de tomada de decisões que implica a consideração de factores políticos, sociais, económicos e de engenharia, com a avaliação de risco relacionada a um perigo potencial a fim de desenvolver, analisar e comparar opções de controle, e escolher a melhor resposta para a segurança ante esse perigo.

Bugs – é um erro involuntário de programação que pode não ser responsável pelo bloqueio de uma aplicação, programa ou sistema operativo mas que causa alguns problemas na operação ou utilização.

Análise de Risco - é uma actividade que reflecte a preocupação que as empresas de desenvolvimento de software tem em saber qual o grau de exposição frente às ameaças capazes de comprometer os objectivos da sua operação.

Metodologia – é a descrição de uma sequência de actividades práticas a ser executadas durante o desenvolvimento.

BIBLIOGRAFIA

AZEVEDO, Felipe Vieiralves. **Análise de risco**. [online]. [7 Maio de 2002]. [citado em 15 de Abril de 2006; 17:00]. Disponível em:
http://www.cafesoftware.com.br/imprensa_artigo1.htm

BECK, K. **Programação Extrema Explicada**. Bookman, 1999.

BELLOQUIM, Átila. **SEI/CMM**. [online]. [24 Setembro de 1998]. [citado em 12 de Fevereiro de 2006; 17:00]. Disponível em:
http://www.dca.fee.unicamp.br/~eleri/ea976/01/SEI-MM.htm

BELLOQUIM, Átilia. **CMM (Capability Matury Model) com Metodologia [online**]. [12 de Agosto de 2002]. [citado em 10 de Março de 2006; 10:00]. Disponível em: http://www.sucesues.org.br/documentos/index.asp?cod_noticia=53

BEZERRA, Eduardo **. Desenvolvimento incremental e iterativo.** [online]. [Dezembro de 2003]. [citado em 14 de Fevereiro de 2006; 17:30]. Disponível em:
http://www.mundooo.com.br/php/modules.php?name=MOOArtigos&pa=showpage&pid=20

BOEHM, B. **Tutorial on software Risk Management**, IEEE Computer Society Press, New York, NY, 1989.

CARMEL, E. **Global Software Teams: Collaboration Across Borders and Time Zones**. Pretince-Hall, EUA, 1999.

COEN, Luciana**. Metodologias trazem maturidade à área de TI.** [online]. [18 de Novembro de 2003]. [citado em 5 de Março de 2006; 15:00]. Disponível em: http://www.sucesues.org.br/documentos/index.asp?cod_noticia=349

DURSCKI, Roberto; SPINOLA, Mauro; BURNET, Robert; REINEHR, Sheila. **Linhas de Produto de Software: riscos e vantagens de sua Implantação** [online]. [26 de Novembro

de 2004]. [citado em 11 de Janeiro de 2006; 15:30]. Disponível em: http://www.dimap.ufrn.br/~andre/gti-16/linha_de_producao.pdf

Júnior, José W. da Silva. **Uma disciplina para a engenharia de software: estudo do personal software process (PSP), proposto por Watts Humphey (a proficionalização do desenvolvedor de software).** [online]. [Dezembro de 2004]. [citado em 5 de Março de 2006; 15:00]. Disponível em:
http://www.ufpel.tche.br/prg/sisbi/bibct/acervo/info/2000/Mono-JoseWilson.pdf

LUIZ, Ronaldo Rezende Vilela. **Obtendo qualidade de software com RUP** [online]. [6 de Dezembro de 2004]. [citado em 7 de Janeiro de 2006; 11:00]. Disponível em: http://www.javafree.org/content/view.jf?idContent=7

MATOS A. José. **Dicionário de Informática e Novas Tecnologias** 2ª edição aumentada. Lisboa: FCA – Editora de Informática; 2004.

MIGUEL, António. **Gestão de Projectos de Software**. Lisboa: FCA – Editora de informática; 2003.

MIGUEL, António. **Gestão do risco e da qualidade no desenvolvimento de software**. S. ed. Lisboa: FCA – Editora de informática; 2002

O´BRIEN, james **A. Introducion to information systems; Essentials for the internetworked enterprise.** 9ª ed. United states of America: McGraw-Hill; 2000.

PINHEIRO, José Maurício Santos. **Projectos de redes.** [online]. [14 de Dezembro de 2004]. [citado em 14 de Janeiro de 2006; 10:00]. Disponível em: http://www.projectoderedes.com.br/artigos/artigo_entendendo_as_metodologias_e_padroes_para_projectos_php

PRESSMAN, Roger. **Engenharia de software**. 5ª ed. United states of America: McGraw Hill; 2002.

SOARES, M. Santos. **Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software**. [online]. [Dezembro de 2001]. [citado em 12 de Março de 2006; 14:00]. Disponível em: http://www.dcc.ufla.br/infocomp/artigos/v3.2/art02.pdf

SOMMERVILLE, Ian. **Engenharia de software**. 6ª ed. S. Paulo. Addison Wesley. 2003.

SOUZA, P. R. Rodrigues. **Como investir em tecnologia com segurança: critérios importantes para se adquirir e desenvolver software.** [online]. [Dezembro de 2000]. [citado em 5 de Março de 2006; 15:00]. Disponível em:

http://teses.eps.ufsc.br/defesa/pdf/5852.pdf

VARAJÃO, João Eduardo Quintela. **Arquitectura da gestão de sistemas de informação**. 2ª ed. Lisboa: FCA – Editora de Informática; 1998.

VITIELLO, Jill. **Fast Track into Management**. [online]. [16 Julho de 2001]. [citado em 12 de Fevereiro de 2006; 17:00]. Disponível em: http://www.pmi.org/articles/

ANEXOS

Quadro 1 – Tipos de riscos

Tipos de risco	Riscos possíveis
Ferramentas	O código gerado pelas ferramentas Case é ineficiente. As ferramentas CASE não podem ser integradas
Requisitos	São propostas mudanças nos requisitos. Os clientes não compreendem o impacto das mudanças nos requisitos
Tecnologia	O banco de dados utilizado no sistema não pode processar tantas transacções por segundo, como esperado Componentes do software que deviam ser integrados contem defeitos que limita a sua funcionalidade
Estimativa	O tempo requerido para desenvolver o software é subestimado. O tamanho do software é subestimado.
Organizacional	A organização está estruturada de maneira que diferentes gerências são responsáveis pelo projecto
Pessoal	É impossível treinar pessoal com a habilidade requerida. Pessoas importantes estão doentes em período cruciais. O treinamento necessário para o pessoal não está disponível

Quadro 2 – Metodologias e ferramentas utilizáveis na fase de identificação dos riscos (Miguel 2002, p. 67)

Actividade	*Ferramenta ou metodologia*	*Descrição*
Descrição dos riscos	Brainstorming	O pessoal do projecto identifica verbalmente os riscos, à medida que pensa neles, proporcionando, em seguida, aos participantes a oportunidade de construção sobre as ideias dos outros.
	Relato periódico de riscos	Relato periódico (obrigatório e agendado) dos riscos pelo pessoal do projecto.
	Questionário Taxionómico	Lista de questões, organizadas de acordo com uma Taxinomia de Riscos do Desenvolvimento de Software
	Checklists de riscos	Listas de riscos considerados comuns em projectos de desenvolvimento
Definição do contexto dos riscos	Todas as metodologias acima mencionadas são aplicáveis, desde que o contexto seja definido sempre que um risco é identificado	

Quadro 3 – Níveis de análise possíveis e respectivos atributos de avaliação dos riscos

Nível	**Impacto**	**Probabilidade**	**Exposição ao Risco**
Binário	➢ Sim ➢ Não	➢ Sim ➢ Não	4 valores possíveis para exposição ao risco. (Impacto * probabilidade): - Sim * Sim – **Elevada** (E) - Sim * Não – **Moderada** (M) - Não * Sim – **Moderada** (M) - Não * Não – **Baixa** (B)
3 Níveis	➢ Elevada (E) ➢ Moderada (M) ➢ Baixa (B)	➢ Elevada (E) ➢ Moderada (M) ➢ Baixa (B)	9 valores possíveis de exposição ao risco. (Impacto * Probabilidade): - E*E, E*M, M*E – **Elevada** (E) - E*B, B*E, M*M – **Moderada** (M) - M*B, B*M, B*B – **Baixa** (B)
5 Níveis	➢ Muito elevada ➢ Elevado (E) ➢ Moderada (M) ➢ Baixa (B) ➢ Muito Baixa (MB)	➢ Muito elevada ➢ Elevado (E) ➢ Moderada (M) ➢ Baixa (B) ➢ Muito Baixa (MB)	25 valores possíveis para a exposição ao risco. (impacto * probabilidade): ME*ME, ME*E, E*ME – **Muito elevada** (ME) ME*M, E*E, E*M, M*ME, M*E – **Elevada** (E) ME*B, ME*MB, E*B, E*MB, M*M, B*ME, B*E, MB*ME, MB*E – **Moderada** (M) M*B, M*MB, B*M, B*B, ME*M, - Baixa (B) B*MB, MB*B, MB*MB, **Muito Baixa** (MB)

Quadro 4 - Exemplo de tabela de risco antes da ordenação

Riscos	Categoria	Probabilidade	Impacto	RMMM[7]
Pessoal inesperiente	Tamanho e experiência da equipa	35%	Crítico	
Cliente modificará os requisitos	Tamanho do produto	90%	Crítico	
Prazo de entrega será apertado	Impacto do negócio	60%	Marginal	
Tecnologia não satisfará as expectativas	Tecnologia a ser usada	20%	Catastrófica	

Quadro 5 – Quadro de riscos ordenados

Risco	Probabilidade	Impacto
Problemas financeiros organizacionais forçam reduções no orçamento do projecto	Baixa	Catastróficas
É impossível recrutar pessoal com as habilidades requeridos para o projecto	Alta	Catastróficas
Pessoas-chave estão doentes em períodos cruciais do projecto	moderada	Sérios
Componentes de software que deviam ser reutilizadas contêm defeitos que limitam sua funcionalidade	Moderada	Sérios
O banco de dados utilizado no sistema não pode processar tantas transacções por segundo, como esperado	moderada	Sérios
As ferramentas CASE não podem ser integradas	Alta	Toleráveis
Tamanho do software é subestimado	Alta	Toleráveis
O código gerado pelas ferramentas CASE é ineficiente	Moderada	Insignificantes

[7] RMMM - contem ponteiro para um plano de atenuação, monitoração e administração de risco ou seja para uma colecção de folhas de informação de risco desenvolvida para todos os riscos que fica acima da linha de corte.

Quadro 6 – Estratégias de gestão de riscos

Risco	Estratégia
Problemas financeiras organizacionais	Prepare um documento informativo para a alta gerência, mostrando como o projecto presta uma contribuição muito importante para os objectivos da empresa.
Problemas de recrutamento	Alerte o cliente sobre as dificuldades em potencial e a possibilidade de atrasos; investigue a compra de componentes
Doenças de pessoas da equipa	Reorganiza a equipa de maneira que haja mais sobreposição de trabalho, portanto, as pessoas compreendam as tarefas umas das outras
Componentes defeituosos	Substitua componentes potencialmente defeituosos por componentes comprados e que tenha confiabilidade reconhecida
Desempenho do banco de dados	Investigue a possibilidade de comprar um banco de dados com maior desempenho

Quadro 7 – Formulário de informação de risco

Formulário de informação de risco			
Identificação do risco: P02-4-32	Data: 9/07/05	Probabilidade: 70%	Impacto: alto
Descrição: Alteração dos requisitos			
Refinamento / Contexto Subcondição 1: Tempo muito limitada dedicado para levantamento de requisitos Subcondição 2: Pouco envolvimento dos usuários e cliente no processo de levantamento de requisitos Subcondição 3: Envolvimento de pessoas erradas no processo de levantamento de requisitos Subcondição 4: Má interpretação dos requisitos do cliente			
Atenuação / monitoração 1. Planeia reuniões com o cliente e usuários de modo a surgirem o mínimo de dúvidas possível. 2. No caso da subcondição 1 negoceia com o cliente a possibilidade de aumentar o prazo e aumenta o tempo de levantamento de requisitos 3. Faz entrevistas as pessoas certas para poder obter os requisitos o mais cedo possível.			
Administração / Plano de contingência / disparo: Exposição ao risco calculado como sendo de 230.000$00. Reserva essa quantia no custo de contingência do projecto. Desenvolva cronograma revisado considerando que os requisitos terão de ser alterados			
Disparo: passos para atenuação improdutivos em 1 /09/05 Estado actual: 12/07/06 Passos de atenuação iniciada.			
Emissor: R. Manuel		Assinado: M. Lopes	

Quadro 8 – Metodologias de monitorização e respectivas ferramentas (Miguel 2002, p. 110)

Actividades	Metodologia	Ferramenta
Recolha da informação	• Reavaliar os atributos dos riscos (por ex., Atributos binérios ou três Níveis). • Entrevistar o pessoal do projecto. • Rever documentação técnica e relatórios (por ex., gráficos PERT, planos, orçamentos, registos, etc.). • Rever relatórios de situação. • Recolher dados de produtos do projecto, usando protótipos.	• Avaliação binária de atributos • Avaliação de atributos com três níveis
Compilação da informação	Os dados são analisados e compilados em relatórios de situação, de acordo com as normas do projecto. Esta é actividade em que são examinadas as tendências. Os relatórios podem incluir: ▪ Resumos da situação do plano de mitigação ▪ Resumo da situação dos riscos ▪ Resumos das tendências verificadas	• Gráficos de barras • Relatório de situação da mitigação • Folha da informação dos riscos • Matriz de riscos • Gráficos de correlação temporal • Gráficos temporais
Relato dos resultados	• Apresentar relatórios escritos e verbais • Apresentar formais ***NOTA***: Qualquer destes relatórios pode mostrar a situação de riscos individuais, de áreas agregadas de riscos, tendências ou um misto.	

INQUÉRITO A ANALISTAS / GESTORES / CHEFES DE EQUIPA

DE DESENVOLVIMENTO DE SOFTWARE

Este inquérito foi elaborado por mim, Marcos Ramos da Graça, finalista do curso de Licenciatura em ensino de Informática pelo Instituto Superior de Educação (ISE) e tem como objectivo recolher informações necessárias para fazer uma análise de riscos e dificuldades de desenvolvimento de software em Cabo Verde.

Os dados são confidenciais e anónimos e serão utilizados única e exclusivamente para a elaboração do meu trabalho de fim do curso para obter o grau de Licenciatura em ensino de Informática.

1 - Com base na sua experiência profissional, classifique os riscos mais frequentes e os riscos mais perigosos dos projectos de Software. Utilize uma escala de 1 a 5, sendo 1 os menos frequentes e os menos perigosos, e 5 os mais frequentes e os mais perigosos.

RISCO	FREQUÊNCIA	PERIGOSO
Alteração de Requisitos do Utilizador		
Práticas Incorrectas de Gestão		
Má estrutura organizacional		
Pressão excessiva sobre os prazos		
Baixa qualidade do software		
Estimativas inadequadas de custos		
Correcção deficiente de erros		
Metas não cumpridas		
Derrapagens orçamentais		
Estimativas inadequadas		
Ferramentas e metodologias inadequadas		
Métricas incorrectas		
Inexperiência de utilizadores		
Inexperiência de gestores		
Inexperiência do pessoal		
Fricção entre pessoal empresa, cliente fornecedor		
Elevados custos de manutenção		
Baixa produtividade		
Projectos cancelados		

2 - Com base na sua experiência de analista/gestor/chefe de equipa de desenvolvimento de software estimaria que:

a. A taxa de insucesso de software na sua equipa é de: _____%
b. O orçamento dos projectos é ultrapassado em cerca de: _____%
c. A percentagem do dinheiro do orçamento gasto em alterações e correcções é de: _____%
d. Qual a percentagem dos projectos que a sua equipa já desenvolveu e que incluíam alguma coisa que a empresa não tinha experiência? _____%
e. Qual é a percentagem dos projectos que não tinham planos formais para Testes e Instalação do Software? _____%
f. Quantos projectos desastrosos a sua equipa já teve? _____

3 - Nos projectos não contratados qual é a percentagem:

a) Em que os utilizadores têm alterado os requisitos? _____%
b) De excesso de pressão sobre os prazos? _____%
c) De projectos de baixa qualidade? _____%
d) De derrapagem de Custos? _____%
e) De controlo de configuração inadequado? _____%

4. - Nos Projectos Contratados qual é a percentagem de:

a) Custos de Manutenção? _____%
b) Atritos entre empregados do cliente e a vossa equipa? _____%
c) Alteração dos Requisitos do Utilizador? _____%
d) Critérios de Aceitação não previstos? _____%

5 - Responda às questões abaixo com <u>sim</u> ou <u>não</u> ou assinale a opção "<u>NR</u>" se não quiser responder.

Questões	Sim	Não	NR
A sua empresa já entregou alguma vez um software ao cliente fora do prazo estabelecido?			
Tem havido reconhecimento pelo trabalho realizado pelos membros da equipa?			
Considera que a sua equipa tem um bom espírito de equipa?			
Envolve os membros da equipa nas tomadas de decisões?			
Controlam as alterações das especificações?			
Existem planos formais para todas as actividades do projecto?			
O processo de desenvolvimento é bem compreendido pelos membros da equipa**?**			

Numa escala de 1 a 4 classifique a sua equipe em termos de:

Indicadores	Pontos
Experiência do chefe da equipa	
Análise de *Performance*	
Segurança	
Desenho e Metodologias	
Cumprimento do prazo para a conclusão dos trabalhos.	
Nível de aceitação dos vossos Produtos no mercado	
Garantia de Qualidade	
Contrato cliente-fornecedor	

1 – Insuficiente

2 – Suficiente

3 – Bom

4 – Muito bom

Costuma utilizar algum(s) modelo(s) formal(s) de desenvolvimento de software (orientado a objectos, análise estruturada, etc.)? Sim ☐ Não ☐

Qual(quais)?_____

_____

Que comentário tem a acrescentar sobre riscos e dificuldades no desenvolvimento de **projectos de software** em Cabo Verde?

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Obrigado pela sua colaboração!