ATIVIDADE 4

1 – Situação Geral

O artigo "A evolução do sistema de desenvolvimento de softwares da Microsoft" apresenta uma visão sobre o desenvolvimento de softwares dentro da Microsoft.

O grupo deverá fazer uma análise crítica entre os procedimentos de projeto, implementação e análise de qualidade adotados pela Microsoft e os procedimentos expostos na Unidade 4. O texto deverá comparar os procedimentos, as ferramentas, apresentando suas vantagens, desvantagens e implicações no desenvolvimento do software e na qualidade do mesmo.

3 – OBSERVAÇÕES

a) **A atividade é em grupo**. Os integrantes devem discutir sobre o assunto e elaborar um texto com no mínimo 15 linhas e no máximo 30.

b) O líder do grupo deverá estimular a participação dos demais integrantes. O líder deve postar uma mensagem de abertura da atividade no fórum no tema **Atividade 4** e os demais membros incluir as suas contribuições como respostas a essa mensagem.

C) O líder deverá postar a resposta consolidada do grupo via e-mail para o professor. O texto deve ser "colado" na mensagem de e-mail, sem nenhum arquivo em anexo. O e-mail deve conter logo no início:

Nome do centro:

Nome do grupo:

Membros que não participaram da atividade:

Atenção: não serão considerados os trabalhos postados no fórum.

c) A data de entrega do trabalho é até meia-noite do dia 25/06. Os trabalhos enviados após essa data serão analisados, mas poderão não ser considerados para efeito de obtenção da nota da atividade.

Participação dos integrantes:

A realização desta atividade é imprescindível para o entendimento do conteúdo ministrado, assim como base para a compreensão do conteúdo e

atividade seguinte. Além disso, **será cobrado em prova** o conhecimento adquirido, uma vez que a atividade é parte integrante da nossa disciplina.

A EVOLUÇÃO DO SISTEMA DE DESENVOLVIMENTO DE SOFTWARES DA MICROSOFT CORPORATION

O Antigo Estilo Administrativo

Em meados dos anos 80, o ambiente de trabalho dos programadores na Microsoft era "deliberadamente caótico". Quase não existia hierarquia. As equipes individuais de desenvolvimento de produto eram pequenas. Geralmente não mais que três pessoas. A criação de software tinha o controle diário direto do presidente fundador Bill Gates.

A filosofia de Gates rezava que, com menos estrutura, as pessoas poderiam ser mais inovadoras. Ferramentas do software que depois deveriam funcionar juntas eram construídas em unidades totalmente independentes, com um mínimo de troca de informações. Dentro da Microsoft, as promoções aconteciam porque as pessoas conseguiam descobertas técnicas e não por terem capacidade administrativa. E, para completar as coisas, o estilo mão-na-massa de Gates muitas vezes significava ele passar por cima da hierarquia e realizar mudanças importantes nos objetivos e mesmo em programas, sem lembrar-se de avisar os envolvidos. Como seria de se esperar, existia conflito perene entre o desejo de cada grupo de melhorar o produto cada vez mais e a necessidade de pôr esse mesmo produto na rua.

Dentro da organização adotada e desprovida de estrutura da Microsoft, as crises de desenvolvimento de produto, constantes, eram resolvidas em confrontações agressivas e com muita gritaria de parte a parte. Gates, sempre a par de todo o processo e claramente no controle da empresa, não dava a maior importância a esses encontros que classificava como "comunicações em banda larga superior". Seu evidente e claro objetivo de "ser o produtor líder de softwares para computadores pessoais" suplantava tudo o mais. Era voz corrente que ele demonstrava "paranóia competitiva", temendo que pudessem tomar a Microsoft de assalto, assenhorear-se dela ou mesmo a destruí-la. Gates jamais cessava de querer "vencer" e mesmo destruir tais ameaças.

O Mundo dos PCs Muda

Este estilo funcionou bem para os softwares aplicativos e para os primeiros computadores de mesa, mas perdeu eficiência quando a Microsoft começou a desenvolver sistemas maiores, como o Windows. Mesmo então, Gates se mostrou notavelmente flexível em matéria de organização. Entre 1983 e 1994, reconhecendo as próprias limitações, Gates passou por três diretores (responsável por setor) – James Towne, Jon Shirley e Michael Hallmann, cada um deles contribuindo com uma nova disciplina para a empresa – enquanto tratava de reorganizar seu próprio trabalho.

Numa jogada, mais ambiciosa, enquanto o projeto do Windows, de US$ 100 milhões, estava em andamento, Gates reorganizou a Microsoft em duas divisões independentes: a System Software e a Business Applications. Cada uma delas era dirigida por um vice-presidente. A idéia visada era a de que – com o MS-DOS assim dividido internamente – os grupos externos de programadores de aplicativos, cujos projetos giravam em torno do MS-DOS, poderiam ter comunicação direta com o grupo da System Software sem revelar informações potencialmente valiosas para a Microsoft Applications.

Mas, ironicamente, o resultado foi que a Microsoft teve muito mais conhecimento sobre todas as atividades em aplicativos de seus concorrentes. Gates procurou aplacar as reclamações do pessoal de fora da empresa, insistindo em criar uma "Muralha da China" entre as duas operações. Os cépticos comentaram que essa medida estava mais para peneira do que para muralha.

Os Melhores e Mais Trabalhadores

O estilo da Microsoft sempre fora o de contratar programadores melhores e de maior capacidade de trabalho, onde quer que estivessem, e colocá-los num ambiente com grande liberdade de ação. Para uma colocação, às vezes entrevistavam-se centenas de pessoas. Não existia orçamento para contratações. O pessoal de RH tinha toda a liberdade para contratar aquele talento especial, na hora em que fosse encontrado.

Ao contratar alguém, a Microsoft não dava atenção à educação formal ou experiência do candidato. Afinal de contas, nenhum de seus fundadores, Bill Gates ou Paul Allen, havia terminado a faculdade. Não importava quantos títulos aparecessem nas credenciais de um candidato. Ele só seria contratado após passar pelo crivo da empresa, respondendo sobre seus conhecimentos em programação e outras qualidades e capacidades.

O processo da entrevista era feroz: de quatro a seis programadores e diretores passavam de uma a duas horas examinando o candidato. Os entrevistadores faziam picadinho do novato; propunham-lhe perguntas técnicas muito difíceis e inesperadamente lhe estendiam lápis e papel dizendo: "resolva este problema". A ênfase era em como pensa o candidato e como ele trabalha, quando sob pressão. Durante muito tempo Gates insistiu em entrevistar pessoalmente todo programador que se candidatasse. E até 1990, Gates era capaz de viajar não importa para onde para descobrir um talento especial.

Dentre as características de organização mais notáveis da Microsoft estavam os "arquitetos", os sete samurais do software que tinham sido conselheiros de Gates, explorando novas tecnologias, e eram autores da estrutura de sistema mais importante. Abaixo deles, cada programador era classificado em 1 de 6 níveis, indo de 10 a 15. se algum programador conseguisse 15 de classificação e se tornasse um arquiteto, era provável que um escritório de advocacia o tornasse um sócio sênior.

Os pedidos empilhavam-se. As equipes de desenvolvimento eram, deliberadamente, conservadas com poucas pessoas. Mesmo com os projetos tornando-se cada vez mais complexos. Por exemplo, a Microsoft tinha apenas 18 desenvolvedores a cargo de todo o projeto de planilhas eletrônicas, enquanto a Lótus, no início dos anos 90, tinha cerca de 120. Conforme Gates: "*Só as equipes reduzidas trabalham direito. Quando começamos o Excel, tínhamos apenas dez pessoas trabalhando nele, eu inclusive. Hoje, sete pessoas trabalham nele.*" Cada grupo ocupado com um novo código trabalhava num regime darwiniano, isto é, a cada seis meses os desenvolvedores eram reexaminados e os 5% inferiores, eliminados.

O Processo de Desenvolvimento do Software

Já em 1994, o formato do desenvolvimento de um software na Microsoft havia evoluído para um sistema bem menos caótico. Segundo o Sr. Robert Muglia, diretor do Windows NT, "*uma das coisas que a Microsoft aprendeu muito bem foi criar produtos que satisfaçam a necessidade dos clientes, concentram-se naquilo que o cliente quer e, ao mesmo tempo, realizam tudo isso com a bossa de quem entende de negócios.*" A Microsoft criou duas funções básicas: gerentes de produtos e gerentes de programas.

Os *Gerentes de Produtos* controlavam o relacionamento com segmentos específicos de clientes. Eram responsáveis pela compreensão continuada das necessidades do cliente do ponto de vista descritivo e pela maioria das apresentações ao cliente, assuntos de venda, propaganda, composição do preço, formação da força de vendas, assuntos relativos ao gerenciamento dos canais, etc.

Os *Gerentes de Programação* trabalhavam com o pessoal do produto, para compreender a fundo as necessidades do cliente em nível técnico, e então detalhar essa compreensão em especificações para o design. Durante todo o processo, os gerentes de programa trabalhavam com grupos de desenvolvimento, grupos de teste e grupos de educação do usuário para garantir que o produto satisfizesse as necessidades previamente definidas.

O Sr. Peter Neupert, diretor Sênior, do Desenvolvimento de Produto Internacional destacou:

> *As especificações originais para a funcionalidade do programa - no que diz respeito ao "timing" e aos tipos de desempenho que são críticos - são estabelecidas pela alta direção da Microsoft. O segundo nível de especificações são as da interface programática, que tornam os sistemas operacionais compatíveis. O nível seguinte são as interfaces dos aplicativos ou APIs. Nesta altura já temos um maço de documentos internos que descrevem a interação entre alguns dos componentes, os sistemas de arquivo de administração de memória e coisas do gênero. Outra interface crítica é a interface programática que chega às mãos do usuário final. À medida que desenvolvemos o produto, precisamos nos certificar de que essa interface está presente; de outra maneira o produto falhará comercialmente. O grande desafio é manter cada pessoa da equipe de desenvolvimento ligada a esse fato, pela duração de todo o processo de programação. Essas interfaces são tão importantes que as pequenas equipes que fazem o programa geralmente são organizadas em função delas e das ferramentas específicas, necessárias para desenvolver o subsistema, exatamente como um sistema de arquivo.*

E o Sr. Muglia continua:

> *Os Gerentes de Programação têm as especificações e são os responsáveis da garantia que o produto dá ao consumidor de cumprir o que ele, consumidor, deseja, em todos os detalhes. Descem ao nível de dizer com precisão como será a aparência do produto para o cliente externo. Não descem até o nível das estruturas de dados que os implementam. Porém, as equipes de desenvolvedores são as reais donas dos códigos. Isto estabelece um bom relacionamento porque a maioria dos desenvolvedores não quer ser quem toma as decisões acerca do que fazer, mas deseja, isso sim, ficar com o controle absoluto do código e dos algoritmos.*

O Processo das Especificações

> *No nível da equipe de projeto, delineamos os assuntos e, para elementos mais importantes, usamos especificações detalhadas quanto ao que os produtos devem fazer - mas tudo isso tende a ser documentação verbal. Usamos quadros brancos para discutir as implementações tecnológicas (alvos) as quais, às vezes, acabam sendo escritas como especificações. É provável que nove entre dez vezes isso não aconteça. Não somos bons nessa história de manter as especificações de nossos produtos. Dentro de um sistema com 8 milhões de linhas como o NT há tanta coisa que, tentar documentá-las todas provavelmente dobraria o tamanho de nossa equipe e inibiria nossa capacidade de realizações no futuro. De modo geral, nosso código acaba sendo a especificação. Porém, as pessoas que o escreveram entendem os objetivos detalhados e as alternâncias que foram feitas. No caso de perdermos toda uma divisão, teríamos uma tarefa imensa para recriar o código e seu correspondente raciocínio. Temos, porém, pessoas que já trabalharam em "processamento de palavras" por, digamos, oito anos, e somos capazes de manter a continuidade nesse setor de modo que um núcleo de pessoas consegue manter os níveis de conhecimento necessários. (Operamos desse modo porque) nossa tecnologia muda com muita rapidez.*

Ainda o Sr. Muglia:

> *Olhamos as especificações como um ponto de partida muito importante para documentar quaisquer acordos que venham a existir. Geralmente o desenvolvimento de uma especificação nada mais é que o processo de uma equipe. Apenas O a 20% do tempo é gasto com a escrita das especificações. Porém, cada pessoa-chave interage com outras cinco ou seis, para chegarmos ao acordo de que isto é o que devemos fazer. Mas, quando as especificações já estão escritas, elas já (é comum) se transformaram em lixo porque nessa altura já aprendemos cinqüenta fatos ou mais, desde quando estabelecemos o primeiro acordo sobre as especificações. Mas nós não partimos para atualizar nossas especificações. Em vez disso, a cada quatro meses, juntamos todos os "consertos" que os clientes nos pediram para fazer e os consolidamos em pacotes de manutenção (que se transformam na documentação atualizada do programa). Os mesmos desenvolvedores acertam o código que, originalmente, serviu para escrever o programa.*

Alvos da Competição

Eis o que o Sr. Neupert comentou sobre este processo:

> *Nossa concentração na concorrência é incrível. Se é que existe um ponto mais fixado num programa, esse é vencer a concorrência. No negócio das redes, esse ponto é ser mais rápido que a Novell; no negócio de planilhas eletrônicas, é ser melhor que a Lotus. Desse ponto de vista, você pode começar a ver o que é necessário para alcançar o objetivo. Claro que esses não são os únicos alvos, e geralmente eles nem são quantificáveis. É comum precisarmos tomar decisões que implicam apenas em julgamentos. Por exemplo, um dos assuntos mais difíceis quando se lida com Sistemas Operacionais é decidir que tamanho de memória você quer instalar e quais são as compatibilidades que você precisa. Tais decisões são primeiramente abordadas em nível bem geral por Bill (Gates) e Paul (Maritz). Mas as especificações mudam constantemente à medida que descobrimos o que podemos e o que não podemos conseguir dentro de um objetivo e como isso*

afeta os outros. Por exemplo, na primeira versão do NT queríamos que o sistema trabalhasse em oito megabytes. Quando a coisa não funcionou, doze passou a ser nosso objetivo e, por fim, visamos um sistema de dezesseis megabytes. Essas decisões mais abrangentes afetam seriamente nosso posicionamento no mercado. Outro exemplo pode ser visto em nossa próxima versão do Windows onde foi preciso atualizar grande parte da base de máquinas já instaladas de modo a transformar o programa num fenômeno de massa. Isso significa estabelecermos critérios de programas tendo como base a composição do hardware que, acreditamos, será importante no futuro. A questão é que a definição de especificações é um processo dinâmico. Os pontos básicos importantes estão onde pensamos que a concorrência estará e no que achamos que o mercado espera.

Administração do Projeto

O Sr. Muglia continua:

Iniciado o projeto, passamos a projetar o cronograma. Uma de nossas ferramentas essenciais é o Microsoft Project. Os próprios desenvolvedores estabelecem o cronograma e os planos de integração. São eles que sabem como todas as peças se juntam no nível de maior detalhe; sabem qual é a ordem das dependências e projetam cronogramas para disciplinar as seqüências e as dependências. Marcam revisões do projeto mensais, no mínimo, para tudo isso. As equipes de desenvolvimento dão a partida e apresentam qual a exata situação de cada componente do grupo e em que função está trabalhando. Todos procuram antever com antecedência de um mês os problemas em potencial que podem surgir nas seqüências e nas funções. À medida que os projetos entram nos estágios mais abrangentes, aumentamos a freqüência das reuniões do núcleo das equipes de modo a descobrir e resolver os problemas. Todos compreendem exatamente o que significa contar problemas, a necessidades de consertá-los no futuro e quais são as prioridades.

Paul Maritz, Vice-Presidente para Sistemas Operacionais, prossegue com o relato:

Algumas empresas utilizam um design e realizam um ciclo de implementação. Nós não. A menos que você pense no modo como vai implementar o código na hora em que está fazendo o design dele, você não conseguirá fazer a implementação. Você deve pensar em termos de custos à medida que executa o design. Os níveis de abstração estão muito altos? E por aí afora. 0 primeiro passo é fixar a taxionomia. A seguir, dividimos o processo em áreas e funções diferentes. Então definimos as interfaces entre elas. Escrevemos as interações dessas interfaces. Depois tornamos a dividir o processo em funções ainda menores. Os usuários destas últimas podem criticá-las segundo os termos de seus objetivos e interfaces. Grande parte deste trabalho é feito de modo informal entre as equipes que interagem extensivamente. Procuramos cuidadosamente estabelecer, em primeiro lugar, os critérios para cada função e subfunção. Todos lêem os códigos uns dos outros de modo a terem um vocabulário em comum. Todo esse processo é inerentemente cheio de alternâncias. Cada grupo tem uma grande preocupação com sua própria funcionalidade, com seus propósitos e com a seqüência do tempo.

Neste processo, nota o Sr, Neupert,

> *Nós jamais colocamos as equipes em competição umas com as outras dentro da empresa. Claro que pode haver opiniões frontalmente divergentes dentro da equipe quanto ao que seja a abordagem correta ou a arquitetura acertada. Incentivamos essa atitude o tempo todo. O desafio é ter o assunto resolvido de um modo ou de outro. Mesmo após termos uma decisão inicial, pode acontecer de que, quem perdeu ir atrás da própria opinião só para provar que estava certo. Por exemplo, aconteceu um conflito entre o Windows e o OS/2. Internamente a maioria acreditava que o Windows não conseguiria ser melhor do que a nova arquitetura da Intel para explorar memória em modo protegido e coisas do gênero. Entretanto, um dos elementos do grupo continuou pesquisando o assunto apesar do fato de que noventa por cento de nossos recursos estarem, àquela altura, voltados para o OS/2. Ninguém imaginava que ele conseguiria: mas quando, finalmente, ele descobriu um modo de fazer o Windows funcionar, a solução encontrada tinha muitas características melhores e parecia servir ao mercado melhor do que o OS/2. Foi assim que mudamos nossa aposta no futuro do OS/2 para o apoio ao Windows.*

Nada de Fórmulas Mágicas

Paul Maritz continua:

> *Com 1500 pessoas trabalhando no Sistema Operacional e nos programas que costumam implicar em 2 milhões de linhas de código, o processo apresenta enorme complexidade. Não temos nenhuma mágica para escrevermos nosso código. Nossos programas devem rodar em múltiplas plataformas de hardware e devem se adaptar à medida que essas plataformas são modificadas. Não existe um corpo de procedimentos que possamos seguir. Pessoas inteligentes são a base de tudo, mas as equipes podem suplantar a experiência e capacidade desses indivíduos. O maior problema é lidar com a interação espacial entre os subsistemas. Centenas de coisas estão acontecendo em paralelo. O que excede em muito a complexidade de qualquer ferramenta CASE; por isso não usamos essa ferramenta. Também não usamos programas de "macrocontrole" como o Método 1 ou o Método 2 de Andersen. Preferimos, em vez disso, que os elementos do grupo documentem o programa à medida que avançam com o trabalho.*
>
> *Desenvolvedores veteranos ficam responsáveis pelo design geral. Entretanto, a maioria dos problemas acontece quando da implementação. Não se pode tolerar a perda do controle sobre o código de base. É preciso feedback constante de todos os elementos. Nos primeiros estágios, o costume é "deixar os portões relativamente abertos" de modo que todos possam ver tudo o que acontece. Entretanto, mais tarde, com mais de 200 desenvolvedores trabalhando num programa como o NT, se você permitir que os desenvolvedores verifiquem os detalhes dentro do sistema, você não conseguirá nem dar a partida. Tudo falharia.*

Para manter a coordenação, o grupo dos Sistemas Operacionais Microsoft usou uma técnica para levar o programa a um "teste de construção". Pelo menos uma vez por semana, no mais das vezes, duas a três vezes por semana, cada grupo recompila seu software de modo que toda a equipe possa construir um programa consistente e coerente, com todas as novas funções e características em seus devidos lugares.

É freqüente que a tentativa de unificar todo o programa ou subsistema falhe. Entretanto, essas "construções" forçam as pessoas a se esforçarem até encontrar e

consertar o que houve de errado desde a última “construção”. Se os erros não forem consertados neste estágio, as interações tornam-se rapidamente tão complexas que seria impossível fazê-las coerentes, mesmo que cada subsistema possa operar efetivamente cada um por sua conta.

Nas palavras de Paul Maritz:

Mais tarde fica difícil melhorar o desempenho de qualquer programa. O que significa que, além das “construções”, dividimos o processo em marcos, cada um dos quais vale por uma base de um dado cliente. Embora nossos “designers de teste” executem seqüências formais de testes para disciplinar cada etapa deste processo, não podemos adivinhar o que cada pessoa faz ou fará com o programa. Conseqüentemente, a cada marco fazemos os arranjos para que o programa possa ser usado para um dado propósito no mundo real. É assim que conseguimos feedback dos usuários do programa e trabalhamos a partir dessa base. A cada etapa sucessiva aumentamos a abrangência do programa. A cada marco zeramos a lista dos defeitos para essa clientela e para essa série de testes.

Para cada subsistema existe uma equipe composta de uma a dez pessoas. Impossível ter equipes maiores do que isto. As pessoas precisam ter a capacidade de entender toda a complexidade do subsistema e suas interfaces. Devem conhecer-se muito bem mutuamente e serem capazes de confiar e de julgar as mútuas necessidades e soluções. Conseqüentemente desenvolvemos uma série de rituais que forçam a todos se reunirem em torno do que estão construindo.

O Sr. Peter Neupert ressalta *“Cada um de nossos grupos de aplicativos tinha, geralmente, suas próprias ferramentas. Agora passamos a ter convenções e estamos fabricando algumas ferramentas; estas, porém, não são usadas com consistência pelos vários grupos. Em lugar de forçar a utilizar determinadas ferramentas ou impor interações, nossa atitude predominante é:” Quando as pessoas precisam saber de alguma coisa, devem ir e procurar as informações de que precisam. ”Geralmente isso acontece com o indivíduo. A permanência da pessoa dentro da empresa realmente é produtiva em termos do conhecimento de como se deslocar dentro do sistema, a quem fazer as perguntas e com quem falar.”*