Revista

INICIA

Revista de Iniciação Científica da FAI – Centro de Ensino Superior em Gestão, Tecnologia e Educação.

Publicação anual
Número 1, Ano 2001
FAI – Centro de Superior em Gestão, Tecnologia e Educação

Santa Rita do Sapucaí – MG – Brasil
Outubro de 2001

Revista Inicia, n. 1
Santa Rita do Sapucaí: FAI – Centro de Ensino Superior em Gestão, Tecnologia e Educação, 2001
Anual
ISSN 1806-8278
1. Administração. 2. Informática. 3. Educação
FAI – Centro de Ensino Superior em Gestão, Tecnologia e Educação.

EXPEDIENTE

Revista INICIA
Revista de Iniciação Científica de Informática e Administração da FAI
ISSN 1806-8278
Ano 01-Número 01

Tiragem
500 exemplares

Endereço para correspondência
Faculdade de Administração e Informática
Av. Antônio de Cássia, 472
Jardim Santo Antônio
CEP: 37540-000
Santa Rita do Sapucaí-MG
Brasil

Editorial

A Faculdade de Administração e Informática de Santa Rita do Sapucaí – MG, dando continuidade ao processo de engrandecimento que vem experimentando em todos os aspectos da Instituição, lança o primeiro número da revista INICIA – Revista de Iniciação Científica de Informática e Administração da FAI.

Dentro os objetivos estabelecidos para justificar a criação de mais um veículo de comunicação na FAI destacam-se: incentivar a pesquisa científica e a prática da educação continuada, levar à comunidade as pesquisas desenvolvidas na Faculdade, contribuir com a sociedade divulgando novas técnicas e métodos, favorecer a formação um profissional mais crítico, consciente e participativo, aproximar os acadêmicos da comunidade e divulgar a opinião dos alunos e professores, permitindo o debate.
A revista publicará artigos, resenhas e informações sobre pesquisas de modo geral. Para isto, conta com o empenho de toda comunidade acadêmica da FAI no sentido de gerar material publicável de alto nível.

Neste primeiro número serão publicados sete artigos, todos elaborados pelos alunos do 4º ano do curso de Ciência da Computação, dentro da Disciplina Engenharia de Software II, ministrada pela Profª. Adicinéia Aparecida Oliveira, que teve a iniciativa de solicitar a criação deste veículo de comunicação de idéias e de resultados de pesquisas.

Estamos certos de que, além de servir como um novo elo de ligação entre a Faculdade e a comunidade, a revista INICIA poderá contribuir de modo significativo para aumentar a sinergia no relacionamento professor-aluno dentro da FAI. Esta interação positiva é fundamental para que se possa comunicar, de modo natural, a unidade de propósitos necessária entre os segmentos de uma instituição que busca a excelência no ensino de gestão e sistemas de informação.

Prof. José Cláudio Pereira

Diretor

SUMÁRIO

PEOPLEWARE: ENTENDER É SINÔNIMO DE PRODUZIR

Conrado Chiaradia Navarro
Faculdade de Administração e Informática
conrado.navarro@bol.com.br

Valquíria de Cássia Pinto
Faculdade de Administração e Informática
val_cp@bol.com.br

Resumo *– A competitividade nos dias de hoje não depende mais de grandes investimentos em equipamentos ou infra-estrutura. O maior bem que a empresa possui são as pessoas que fazem parte de seu quadro de recursos humanos. O trabalho em equipe é capaz de transformar pequenos recursos em grandes resultados, transformando a imagem da empresa. A motivação dessas pessoas é que gera, ou não, maior competência no mercado e posteriormente maior participação nos resultados práticos da globalização. A maneira que essas pessoas trabalham é que muda todo cenário. O desafio é aplicar este raciocínio ao desenvolvimento de software.*

Abstract *– The new market race is no longer dependent on money or equipments. Not at all. The most important thing now is the team working on the companies. The teamwork makes possible the creation of huge projects with minimum resources, transforming the company's image. The motivation of the team is the difference between the pro's and the amateurs on this new global era. The way they work changes everything. The challenge today is to fit and apply all this new way of thinking on the software development process.*

Palavras-chave *– Peopleware, motivação, equipe, software, liderança, Microsiga, relacionamento, sucesso, projeto, pessoas, carreira, desenvolvimento de software, mercado de trabalho, organização, informática, cooperação.*

1. INTRODUÇÃO

O conhecimento técnico, puramente técnico não garante mais um lugar ao sol nos dias de hoje no mundo de tecnologia. As habilidades técnicas são mais que obrigatórias e por isso não são mais consideradas como fator decisivo. A nova visão de mercado, globalizada, requer profissionais que sejam dinâmicos, pró-ativos e principalmente que tenham "sentido" de cooperação suficiente para fazer parte de um time vencedor. Este time não é o seu departamento ou sua sala de trabalho, mas sim a corporação, a empresa, o mundo que está ao seu redor. Este é o novo cenário gerado pela globalização.

As empresas ao redor do planeta precisam de profissionais capazes de agregar valor aos seus negócios e conseqüentemente fazer parte dos seus planos futuros. Os conhecimentos técnicos tendem a ser cada vez mais obrigatórios porém menos decisivos. Precisamos trabalhar nosso lado humano.

2. QUAL O CAMINHO ENTÃO?

A tendência das organizações é cada vez mais aproximar e compartilhar seu sucesso com seus funcionários, sejam eles "soldados rasos" ou "coronéis". Mas para que isto aconteça é preciso que estes "oficiais" realmente vistam a camisa da empresa, ou melhor, a farda e lutem como verdadeiros guerreiros no campo de batalha que os cerca, quero dizer, no mercado de trabalho, demonstrando suas capacidades e valores através de muita perseverança.

Essa prática ofensiva e de cooperação vem dando muito certo em empresas com baixa rotatividade e pouca hierarquia aparente, mas dizer que esses são os motivos principais do sucesso organizacional é esconder o que de melhor fazem os verdadeiros líderes e gestores de pessoas: motivar e motivar. Estes líderes são responsáveis pela nova revolução silenciosa dentro das organizações responsável por tratar as pessoas como pessoas.

A grande diferença está em sentir que somos parte de um todo e que nosso trabalho colabora, e muito, para o sucesso e crescimento de todos dentro da organização. É ai que entram em cena estes especialistas em gerir conflitos, criar times vencedores e principalmente em fazer dos poucos recursos que têm sua grande fonte de inspiração e trabalho. O segredo de um time campeão fica claro quando percebemos que eles trabalham pelos resultados, e que, para alcançá-los são capazes de tudo, mas permanecendo sempre unidos, como começaram.

A tarefa destes líderes não é fácil. Criar esperança onde ela pode não existir mais, unir o que já era parte do vácuo empresarial e transformar um time normal em um time competitivo são algumas das tarefas que estas pessoas tem que enfrentar no seu dia-a-dia. Com o desenvolvimento de software, o cenário fica ainda pior. A pressão do cotidiano e dos clientes fica cada vez mais forte, uma vez que as regras de negócio podem mudar a qualquer instante e a perda de dinheiro é uma ameaça poderosa. Mas e ai? Será que existem equipes ou empresas de software que sejam desafiadoras, motivadoras e inovadoras? Claro que sim. E não são poucas. Um exemplo claro é a *Microsiga* que é a maior empresa de software empresarial do Brasil.

A *Microsiga* merece alguns pontos de destaque e observação que servem como parâmetros para um melhor desempenho das equipes de desenvolvimento. Vamos analisá-los: o primeiro é a participação dos funcionários em seus processos de decisão e ações, que mesmo de forma indireta em alguns casos, cria uma cultura única dentro da empresa capaz de formar uma identidade forte do todo perante os outros. O segundo aspecto positivo é o forte clima de amizade e descontração existente entre todos. Não é bem uma festa de formatura, mas vejam que interessante: a *Microsiga* cria anualmente reuniões para apresentar resultados e novas estratégias e faz questão que todos dêem a sua opinião e ofereçam novas alternativas. Esta prática aproxima as pessoas e faz com que os pensamentos de todos fiquem claros perante os diretores, gerentes e presidentes. Não há jogo de interesses aparente e assim, o processo de gerenciamento das pessoas fica mais fácil e menos burocrático. O terceiro e último ponto importante é o grande incentivo à reciclagem intelectual que existe entre os gestores dos departamentos. Nesta nova era, as pessoas valem pelo conhecimento e é apostando nesta premissa que a *Microsiga* investe em cursos e mais cursos para todo o seu time. Este comprometimento demonstra a preocupação da empresa com o bem estar, a produção intrínseca de seus funcionários, transformando o vínculo empregatício em uma relação de troca de experiências e conforto que acaba se revertendo em altíssima produtividade e satisfação por ambos os lados. E de quebra deixa a equipe realmente motivada e preparada para enfrentar qualquer desafio.

3. NÓS SOMOS O PROBLEMA

O ponto fraco da administração científica moderna está em sua dificuldade de reconhecer no ambiente os componentes não-racionais e lidar com eles, com tudo aquilo que não é resolvido pela tecnologia, pelas máquinas. O resultado é a falta de habilidade para resolver os conflitos do dia-a-dia e que são freqüentemente os maiores responsáveis por impasses ou discórdias entre membros de uma equipe.

O ser humano tem uma natureza complexa e freqüentemente contraditória, e os paradoxos do seu comportamento não devem surpreender. Onde há amor, há ódio também, cooperação e competição podem alternar-se, assim como ternura e hostilidade, alegria e tristeza, carinho e agressão, franqueza e dissimulação, lealdade e traição, razão e tolice.

As relações entre as pessoas percorrem essas dimensões alternativas, sem, entretanto, lograr uma posição fixa e definitiva em termos de normalidade. O conflito é apenas uma divergência, que pode ser sadia ou doentia, dependendo da maneira como é resolvido dentro das situações críticas do dia-a-dia. Por exemplo, você já deve ter passado por uma ou por várias das situações abaixo.

- Diferenças nos objetivos individuais: eu quero uma coisa e ele quer outra.
- Mal entendidos: é o que mais acontece... Ele quis dizer uma coisa e eu entendi outra.
- Pontos de vista diferentes xiii, isso então...
- Choques entre personalidades diferentes: imagine um gerente detalhista criticando o relatório todo resumido de um generalista?
- Competitividade e falta de cooperação: real ou imaginária.
- Problemas com autoridade e poder: será que existe alguém que não tenha uma história como essa pra contar?
- Divergências de métodos e técnicas: Concordo com os objetivos, mas discordo quanto aos meios para atingi-los.

Como você agiu quando passou por uma dessas situações? Seja qual for a nossa posição hierárquica em uma empresa, as habilidades psicossociais, aquelas que nos ajudam a lidar com as emoções e com os outros, integrando o individual e o coletivo, facilitam - e muito - nosso progresso profissional. Elas são fundamentais para auxiliar você a lidar com situações de conflito e é dessas tais habilidades que as empresas precisam.

A motivação de uma equipe está intimamente ligada com a cumplicidade e transparência de seus integrantes. Mas para que eles cheguem a este "grau de consciência" é preciso que os líderes e as empresas esteja atentos para algumas necessidades básicas do ser humano interessado no sucesso, que são:

1. Necessidade de reconhecimento: Desde pequenos, somos criados e tomamos atitudes em prol do

reconhecimento dos "maiores". Basta se lembrar do quanto fez birra por aquele novo carrinho ou aquela nova boneca. Precisamos chamar a atenção para que possamos ir para o próximo degrau da vida. Da mesma maneira, temos que perceber que nosso trabalho foi admirado e devidamente interpretado, e só assim podemos repetir nossa atuação criando mais resultados positivos.

2. Necessidade de poder falar: Trabalhar sobre pressão e ainda não poder expor seu ponto de vista é andar na contra-mão do sucesso. Alguns têm menos capacidade de expressão, mas mostrar o que pensa é essencial até porque sem isso, não é possível avaliar as expectativas dos integrantes da equipe. Afinal de contas, quem comanda é o piloto, mas além dele, quem precisa estar seguro no fim da viagem são os passageiros.

3. Necessidade de errar: Outra característica básica do ser humano é errar. Todo mundo erra, mas nem todos conseguem pedir ajuda antes de errar. É preciso poder pedir socorro nas horas certas e admitir que o erro existe e encará-lo como mais um fator de sucesso. A maioria dos chefes pensa que se você erra é porque não é capaz, mas a verdade está mais para a falta de capacidade dele de ter ensinado corretamente.

4. Necessidade de compartilhar: A gente precisa cooperar. Mas as vezes não temos espaço. Precisamos fazer parte de um time onde um zagueiro também marque gols e onde um atacante também consiga ajudar a defesa. Não entendam como o "faz tudo", mas como aquela pessoa capaz de orientar, ajudar e principalmente compartilhar seu sucesso e seus fracassos quando necessário.

5. Necessidade de ousar, desafiar: Quando assumimos um compromisso, sempre queremos de certa maneira surpreender as expectativas existentes ao nosso redor. Aliás, essa é a forma mais lucrativa de marketing existente até hoje. O ambiente onde trabalhamos tem que ser dinâmico e capaz de fornecer características particulares para o desenvolvimento individual com foco no objetivo global. Temos que crescer por méritos e desafios, não por imposições ou afeição. A sorte é bem vinda também, mas só ela não resolve.

6. Necessidade de vencer:
Essa é mais viva dentro de nós e a que mais é reprimida pelos gestores ainda reclusos. As empresas precisam de vencedores (isso todos admitem), mas quando elas se destacam demais (vencem) ai a repressão começa. Precisamos de espaço para vencer e de um belo "parabéns". Vencer é desfrutar de uma atitude de coragem e ousadia, por isso ouse.

Depois de conhecer muito bem seus colegas, o líder ainda precisa saber de alguns detalhes como:

- Conhecer e reconhecer sua equipe

Um líder não deve ser avaliado pelo que é, mas sim pela equipe que tem sob sua responsabilidade. Ao aceitar um desafio profissional, aceite-o na razão direta da capacidade contingencial da equipe e da infra-estrutura colocada à disposição da mesma.

- Provocar o Envolvimento

Demonstre interesse pela idéias e percepções dos membros da equipe. Busque feed-backs e informe às pessoas o que você, como líder, fará com as idéias delas. Isso as incentivará a dar novas idéias.

- Dar e solicitar feed-backs específicos

Em vez de citar pessoas, refira-se aos resultados ou aos comportamentos. O fato de um determinado resultado não ter sido alcançado, não necessariamente implicará incompetência de um profissional específico. Se for o caso, assuma que a incompetência permeou toda a cadeia: comando-ação-resultado. Comente o resultado e pergunte o que pode ser feito para melhorar. Feed-backs avaliativos, do tipo quem foi "o incompetente", podem dar margem a ofensas desnecessárias e que, certamente, não modificarão o resultado.

- Estar atento, aprender a ouvir tudo, verbal e não-verbal

Segundo os especialistas da área, 65% da comunicação é não-verbal. Esteja atento para o SIM com sabor de NÃO, e para o NÃO com sabor de SIM. Alguém pode estar apresentando um comportamento favorável a sua proposição e, no plano não-verbal, manter uma atitude (crença) desfavorável.

- Apurar o seu Sensor

Há um dispositivo pequeno colocado nos compartimentos de refrigeração de geladeiras, chamado sensor. Tem a função de detectar em tempo hábil as variações de temperatura (clima) no ambiente. E você, tem feito a manutenção preventiva do seu sensor de líder para sensibilizar-se e proagir em tempo hábil em relação às variações do clima na equipe pela qual responde? As pessoas têm problemas pessoais e necessitam ser ouvidas, o que não significa deixar de fazer o que precisa ser feito.

- Abaixar a Bola

É dia de chuva, muita chuva! Os sistemas de drenagem do campo podem não estar funcionando muito bem. Pode ser que saiam gols, mas também poderão ocorrer contusões e fraturas irreversíveis. Abaixe a bola, proponha passes curtos e, acima de

tudo, mantenha todos em campo, em forma, dispostos e motivados para o próximo jogo.

4. PEOPLEWARE X SOFTWARE

A engenharia de software busca, através de uma tecnologia intitulada Gestão do Conhecimento, estabelecer uma ligação entre os integrantes de uma equipe de desenvolvimento e seu objetivo maior de maneira que com o desenrolar dos acontecimentos todos possam se sentir motivados e sempre dispostos a empreender ainda mais em suas atividades diárias. Essa gestão é responsável por agregar valor à carreira do profissional de maneira que ele possa usar este conhecimento para fazer com que a empresa cresça. Aliás, este é o principal objetivo da gestão dos recursos humanos: maximizar os horizontes dos membros da equipe de forma que os objetivos sejam sempre comuns e que a produtividade seja proporcional ao crescimento profissional de cada um.

Ao idealizar um sistema de software, temos que pensar também na equipe que irá desenvolvê-lo. É muito comum ver idéias geniais serem mal interpretadas e executadas por pura falta de consideração e comprometimento. A solução para isso é tentar conscientizar o time de que aquela idéia trará benefícios para a carreira e a vida de cada componente. Não é fácil, mas é preciso. Tratar pessoas como commodities é o primeiro e mais cruel erro. Nós somos parte do processo sim, mas não somos o meio de produção. Somos mais que isso, somos o projeto, seu sucesso e sua repercussão. É isso mesmo. Afinal de contas, um sistema de software é raciocínio puro. A motivação por parte da equipe tem que existir para que as mentes estejam tranqüilas e eficientes para produzir, e com essa tarefa surgem os líderes que têm papel de integrar empresa e pessoas.

Este relacionamento empresa-equipe pode e deve se estender ainda mais. O que o funcionário sente e vive fora da empresa pode afetar, e muito sua produtividade na empresa. A empresa não tem que monitorar seu dia-a-dia, nem muito menos ficar mandando cartões e flores, mas deve fornecer uma maneira interessante e descontraída de poder ouvir seus funcionários, suas idéias, pensamentos e interesses. Quando ele perceber que compartilha de uma mesma identidade profissional, ele pode enfim, caracterizar-se como um funcionário exemplar e de elite. Sabe seus limites mas conhece bem onde está pisando. Isto é essencial. Se transportamos este aspecto para o desenvolvimento de software veremos que hoje existe uma falta de profissionais que trabalhe por crescimento profissional. A informática está muito caracterizada pelo trabalho mercenário, uma vez que é uma das profissões tem uma relação retorno/tempo muito curta. Estas atitudes são fatais e um profissional destes pode colocar em ruínas o sucesso de seus colegas e conseqüentemente do projeto. Daí vemos mais uma vez a necessidade de existir uma equipe forte e bem integrada.

Os fatores de sucesso de uma equipe de alto desempenho são tão nítidos que às vezes não acreditamos que só isso basta para se criar um time campeão. O principal aspecto que deve ser levado em consideração é a grande capacidade de gerir pessoas e de entendê-las, que o líder deve possuir. Esse é o passo principal. Depois, é preciso criar entre todos os membros uma cultura que mostre que o sucesso do projeto depende de cada atuação individual só que em conjunto. Confuso? É simples: cada um faz o seu trabalho, mas podendo se expressar, conversar, expor suas idéias, interesses e principalmente podendo se abrir com seus colegas, para que seus problemas não afetem diretamente a conclusão do projeto. Vale lembrar que o fracasso de um sistema de software é o fracasso da equipe e que seu sucesso é o sucesso de todos.

Vale lembrar que à medida que as equipes de desenvolvimento ou engenharia de software crescem, a comunicação entre elas torna-se tão difícil e consumidora de tempo como a comunicação entre as pessoas. E o que é pior, a comunicação (entre indivíduos ou equipes) tende a ser ineficiente – ou seja, um tempo demasiadamente grande é gasto transferindo-se muito pouco conteúdo de informação e, freqüentemente, informações importantes "caem por terra". Essa nova barreira da comunicação está sendo ultrapassada através de ajuda da própria tecnologia. Em muitas empresas, a correspondência eletrônica (e-mail) e os *bulletin boards* demonstraram ser meios muito eficientes de interligar um número grande de pessoas a uma rede de informações.

E, se a história passada servir de base, é justo dizer que as pessoas em si não mudarão. Porém, a maneira segundo a qual elas se comunicam, o ambiente em que trabalham, os métodos que usam, a disciplina que aplicam e, por conseguinte, a cultura global para o desenvolvimento de software já mudaram de forma significativa e muito profunda. A ênfase é utilizar as pessoas em seu potencial máximo e recompensá-las de maneira adequada visando sua satisfação como profissional e principalmente, assegurando seu lugar entre os mais produtivos e eficientes desenvolvedores da organização.

Até hoje, a imensa maioria de todo o software tem sido construída para processar dados ou informações. O novo paradigma, que já vem sendo explorado, está preocupado com sistemas que processam conhecimento. Alguns exemplos práticos e de sucesso são os sistemas de ERP (Enterprise Resource Planning) ou CRM (Customer Relationship Management) que criam bases de conhecimento sólidas com o intuito de facilitar e orientar o processo de tomada de decisões estratégicas nas empresas. Este

pensamento voltado para o saber vem das pessoas que através de dados e experiências adquirem conhecimento. E com pessoas foi possível transformar ideologia dos filmes em realidade. As pessoas formam a base da tecnologia a partir do momento em que utilizam seus conhecimentos.

Partindo dessa visão, acabamos de encontrar o Peopleware se defrontando com o Software, ou melhor, o PW interagindo com o SW. Essa interação tem como principal objetivo trazer benefícios e técnicas mais eficientes através do computador porém associadas ao conhecimento e experiência humana. Sem que estejamos dispostos a colaborar efetivamente para o progresso da tecnologia, não é possível transformar cultura em computação.

5. E O FUTURO?

Pensar em Internet ou em Tecnologia da Informação (TI) no futuro, daqui a 20, 30, 50 anos, é uma coisa bem complicada, mas indispensável para o profissional de TI. As previsões certamente irão todas falhar, aparecerão novos fenômenos inimagináveis; e o pior é que isso já estará acontecendo em apenas cinco, dez anos. Porém, o profissional que deixar de acompanhar as novas tendências, que parar de se preocupar com o futuro, corre o risco de ver tudo perdido. O que fazer então?

O primeiro passo é se preocupar com o assunto. O bom profissional de TI tem que estar constantemente preocupado com o mercado, com a sua carreira, com a evolução da tecnologia. Mais do que muitas outras, a área de TI se renova rapidamente e o profissional tem que saber se reinventar completamente, como teve ao sair do *mainframe* para o micro, ou quando aprendeu a desenvolver aplicativos na arquitetura Web. Logicamente, as formas de se preparar para o futuro são várias, tais como ler revistas e livros, fazer cursos, acompanhar sites na Internet, entre outras. Porém, há uma ferramenta um pouco diferente, mas que é interessante: fazer seu *marketing* pensando no futuro.

Sente com calma, olhe tudo o que está acontecendo no mercado e tente fazer o currículo que você gostaria de ter daqui a um, três, cinco anos. Pense em que posição você quer chegar, nas metas que você quer atingir, no que você deseja estar trabalhando, o que é necessário aprender para chegar lá e também como você terá aprendido! Pense seriamente nas experiências de trabalho que teriam sido necessárias, nos cursos, nos projetos que seriam importantes.

Este é um ótimo exercício de gerenciamento da sua carreira, coisa que definitivamente você deve fazer na área de TI. Logicamente, depois desse exercício (que é constante, você deve refazê-lo freqüentemente) você deverá adaptar sua carreira no presente para atingir suas metas daqui a um, três ou cinco anos.

Um segundo passo importante para você se inscrever no futuro, seja ele qual for, é buscar uma boa formação básica. Isso pode ser conseguido principalmente através de uma boa faculdade, mas também com alguns cursos técnicos ou mesmo aprendendo sozinho, lendo os livros certos. O profissional que entender o modo de pensar de TI, que souber aprender sozinho, que entender realmente de tecnologia, está fortemente habilitado a participar do futuro, pois vai poder sempre aprender melhor e mais rápido.

Um último ponto importante: você deve se preocupar com o futuro, mas de maneira nenhuma a quantidade de mudanças na área de TI é uma coisa ruim. Na verdade, é um dos maiores atrativos da nossa área, visto que quanto mais áreas aparecem mais chances de colocação profissional surgem automaticamente.

6. CONCLUSÃO

Trabalhar não é simplesmente produzir e produzir sem saber porquê ou pra que. Nossa necessidade vai além do trabalho escravo, é claro. Precisamos fazer parte de uma organização honesta e muito bem definida, para que possamos então trabalhar com realização e dedicação. Criar sistemas de software é ainda uma tarefa muito mais complexa. Depende de muita atenção e cooperação entre seus desenvolvedores e clientes. Por isso, essa equipe deve estar em sintonia com os interesses da organização para que todo o desenvolvimento seja proveitoso e obtenha sucesso ao ser concluído.

O trabalho em equipe voltado para o desenvolvimento de software tem como principal objetivo agregar conhecimento aos integrantes da equipe na mesma proporção em que seu andamento cresce e se encaminha para um desfecho de sucesso junto ao cliente.

Quando conseguimos criar valores pessoais e motivos especiais para trabalhar conseguimos formar uma família com nossos companheiros e isso faz com que o trabalho se transforme em lazer. Somos especiais e merecemos atenção e condições de trabalho especiais. Por isso que surge um ditado interessante nesta nova era do conhecimento: “Ser gente tem garantido melhores direitos do que ser máquina”. Alguém duvida?

AGRADECIMENTOS

Gostaríamos de agradecer nossos colegas que durante estes anos formaram nossa família temporária e que serão nossas eternas lembranças. Agradecemos ainda nossos professores que têm estado sempre alertas para os interesses e acontecimentos desta época universitária que é única e inesquecível. Obrigado(a) mesmo.

REFERÊNCIAS

JAMIL, George Leal. “Repensando a TI na empresa moderna”. Axcel Books, 2000. São Paulo.

PRESSMAN, Roger S. “Engenharia de Software”. Makron Books, 1995. São Paulo.

Revista EXAME de agosto de 2000.

Revista EXAME de julho de 2001.

Revista VOCE S/A de agosto de 2001.

DESVENDANDO OS MISTÉRIOS DO MUNDO DO SOFTWARE LIVRE

Andrei Simonini Souza
Faculdade de Administração e Informática
simonini@liveware.com.br

Luiz Gustavo Cunha Barbato
Faculdade de Administração e Informática
gustavo@liveware.com.br

Resumo *-- O desenvolvimento e a obtenção de softwares proprietários têm custos altos, com isso o desenvolvimento de softwares livres tem crescido muito nos últimos anos. Com isso, foi criada a General Public License, que tem como objetivo garantir a liberdade de alterar, compartilhar e distribuir softwares em todo o mundo. Com a experiência de uma comunidade que compartilhava programas de computador, foi fundada a Free Software Foundation, que tem como objetivo eliminar a restrição de cópias, redistribuição e modificação de programas de computadores. A partir dos conceitos de liberdade da Free Software Foundation foi lançado o manifesto e o projeto GNU, que tem por objetivo o desenvolvimento de softwares livres através de uma comunidade de colaboradores. Software livre se refere à liberdade dos usuários executarem, copiarem, distribuírem, estudarem, modificarem e aperfeiçoarem o software. O software livre está cada vez mais tomando o espaço do software comercial no mercado. Isto acontece por que os usuários querem softwares baratos, que sejam livres e que possuam o código fonte. Muitas organizações estão aderindo ao movimento, inclusive as do Brasil.*

Abstract *--- The development and the softwares proprietors obtaining have high costs, with that the development of free softwares has been growing a lot in the last years. With that, it was created General Public License, that has as objective to guarantee the freedom of altering, to share and to distribute softwares all over the world. With the community's experience that shared computer programs, Free Software Foundation was founded, that has as objective to eliminate the restriction of copies, distribution and modification of programs of computers. Starting from the concepts of freedom of Free Software Foundation it was thrown the manifesto and the project GNU, that has for objective the development of free softwares through a community of collaborators. Software gets rid he refers to the users' freedom to execute, they copy, they distribute, they study, they modify and they improve the software. The free software is more and more taking the space of the commercial software in the market. Why this happens the users they want cheap softwares, that are free and that possess the code source. A lot of organizations are sticking to the movement, besides the one of Brazil.*

Palavras-chave *--- General Public License, GNU, Free Software Foundation, Stallman, Software Livre, Open Source, código aberto, copyright, copyleft, free-software, software comercial, for free, freeware, software proprietário.*

1. INTRODUÇÃO

O desenvolvimento de software alavanca a tecnologia mundial, os softwares são criados com o intuito de facilitar a nossa vida. As licenças de softwares foram desenvolvidas com o intuito de restringir a liberdade de uso, compartilhamento e alterações. A GNU GPL, a General Public License, ao contrário, pretende garantir a liberdade de compartilhar e alterar softwares de livre distribuição - tornando-os de livre distribuição também para quaisquer usuários.

Quando nos referimos a softwares de livre distribuição, referimos-nos à liberdade. A GPL foi criada para garantir a liberdade de utilização, modificação e distribuição de cópias de softwares, o qual recebeu os códigos fontes, que podem ser alterados ou utilizados em parte de novos programas.

Para assegurar os direitos dos desenvolvedores, algumas restrições são feitas, proibindo a todas as pessoas a negação desses direitos, pois os outros querem o mesmo de nós. Essas restrições aplicam-se ainda a certas responsabilidades sobre a distribuição ou modificação do software.

Para a proteção da Free Software Foundation e do autor, é importante que todos entendam que não há garantias para softwares de livre distribuição, pois o software pode ser modificado e passado adiante.

Qualquer programa de livre distribuição é constantemente ameaçado pelas patentes de softwares. A Free Software Foundation busca evitar o perigo de que esses programas obtenham patentes individuais. Para evitar isso foram feitas declarações expressas de que qualquer solicitação de patente deve ser feita permitindo o uso por qualquer indivíduo, sem a necessidade de licença de uso.

2. QUANDO TUDO COMEÇOU

A idéia de software livre começou quando Richard M. Stallman pediu para um fabricante o driver aberto e as especificações do hardware de uma impressora e recebeu um solene não. De tão indignado com a negativa ele resolveu empreender o que seria o início do movimento Open Source. Uma espécie de John Lennon cibernético, com um longo currículo de batalhas, ele é o fundador da Free Software Foundation, a instituição que congrega o espírito libertário das plataformas de livre distribuição.

Figura 1 - Richard M. Stallman

3. FREE SOFTWARE FOUNDATION

A Free Software Foundation (FSF) foi criada em 1981, a partir da experiência concreta de uma comunidade que compartilhava programas de computadores no laboratório do Massachusets Institute of Technology. Indignados por não terem conseguido o código de programação de uma impressora Xerox, que não funcionava bem, descobriram que os programas, até então compartilhados por programadores e instituições universitárias e públicas, passaram a ser um produto de "mercado" e que os códigos de programação, agora secretos, tinham sido apropriados por grandes multinacionais. A única saída seria construir programas alternativos, totalmente livres. Liderados por Richard Stallman, criaram os conceitos do movimento, as licenças públicas (GPL), o copyleft (esquerda autoral) e o projeto GNU em 1984. O objetivo da FSF é de eliminar a restrição de cópias, redistribuição e modificação de programas de computadores. Site: http://www.fsf.org.

4. PROJETO GNU

A partir dos conceitos de liberdade da Free Software Foundation foi lançado, em 1984, o manifesto e o projeto GNU. O projeto tem por objetivo o desenvolvimento de programas livres através de uma comunidade de colaboradores. Os códigos de programação, as dificuldades, as documentações e o conteúdo dos programas são disponibilizados em sites na Internet e através de listas de discussões específicas que garantem o seu desenvolvimento colaborativo e o aperfeiçoamento permanente. De lá para cá, já foram desenvolvidos milhares de programas totalmente livres. O mais conhecido é o sistema operacional GNU/LINUX [Site: http://www.gnu.org.].

Figura 2 – Símbolo do Projeto GNU

5. LICENÇA GPL

É a General Public License, uma licença que protege o direito de liberdade do software livre. Esta proteção dos direitos se dá através de dois passos; o copyright do software; e a licença para copiar, distribuir e melhorar. O sistema operacional GNU-Linux é GLP [Site: http://www.gnu.org].

6. O QUE É SOFTWARE LIVRE?

Segundo o site oficial do GNU, software livre se refere à liberdade dos usuários executarem, copiarem, distribuírem, estudarem, modificarem e aperfeiçoarem o software. Mais precisamente, ele se refere a quatro tipos de liberdade, para os usuários do software:

- A liberdade de executar o programa, para qualquer propósito.
- A liberdade de estudar como o programa funciona, e adaptá-lo para as suas necessidades. Acesso ao código-fonte é um pré-requisito para esta liberdade.
- A liberdade de redistribuir cópias de modo que você possa ajudar ao seu próximo.
- A liberdade de aperfeiçoar o programa, e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie. Acesso ao código-fonte é um pré-requisito para esta liberdade.

É citado no site que um programa é software livre se os usuários tem todas essas liberdades. Portanto, você deve ser livre para redistribuir cópias, seja com ou sem modificações, seja de graça ou cobrando uma taxa pela distribuição, para qualquer um em qualquer lugar. Ser livre para fazer essas coisas significa que você não tem que pedir ou pagar pela permissão.

Você deve também ter a liberdade de fazer modificações e usá-las privativamente no seu trabalho ou lazer, sem nem mesmo mencionar que elas existem. Se você publica as modificações, você não deve ser obrigado a avisar a ninguém em particular, ou de nenhum modo em especial.

De modo que a liberdade de fazer modificações, e de publicar versões aperfeiçoadas, seja significativa, você deve ter acesso ao código-fonte do programa. Portanto, acesso ao código-fonte é uma condição necessária ao software livre.

Você pode ter pagado em dinheiro para obter cópias do software GNU, ou você pode ter obtido cópias sem custo nenhum. Mas independente de como você obteve a sua cópia, você sempre tem a liberdade de copiar e modificar o software.

Para que essas liberdades sejam reais, elas têm que ser irrevogáveis desde que você não faça nada errado; caso o desenvolvedor do software tenha o poder de revogar a licença, mesmo que você não tenha dado motivo, o software não é livre.

Entretanto, certos tipos de regras sobre a maneira de distribuir software livre são aceitáveis, quando elas não entram em conflito com as liberdades principais. Por exemplo, copyleft é a regra de que, quando redistribuindo um programa, você não pode adicionar restrições para negar para outras pessoas as liberdades principais. Esta regra não entra em conflito com as liberdades; na verdade, ela as protege.

Regras sobre como empacotar uma versão modificada são aceitáveis, se elas não bloqueiam a sua liberdade de liberar versões modificadas. Regras como "se você tornou o programa disponível deste modo, você também tem que torná-lo disponível deste outro modo" também podem ser aceitas, da mesma forma.

No projeto GNU, nós usamos "copyleft" para proteger estas liberdades legalmente para todos. Mas também existe software livre que não é copyleft. Nós acreditamos que existam razões importantes pelas quais é melhor usar o copyleft, mas se o seu programa é free-software mas não é copyleft, nós ainda podemos utilizá-lo.

Às vezes regras de controle de exportação e sansões de comércio podem limitar a sua liberdade de distribuir cópias de programas internacionalmente. Desenvolvedores de software não tem o poder para eliminar ou sobrepor estas restrições, mas o que eles podem e devem fazer é se recusar a impô-las como condições para o uso dos seus programas. Deste modo, as restrições não afetam as atividades e as pessoas fora da jurisdição destes governos.

Quando falamos sobre o software livre, é melhor evitar o uso de termos como "dado" ou "de graça", porque estes termos implicam que a questão é de preço, não de liberdade.

7. COPYLEFT

Obedece a quatro princípios: a permissão de liberdade de cópia; a não contraposição ao copyright; o direito autoral é preservado; e é regulado conforme a licença. Não há legislação brasileira que possa enquadrar como pirataria o uso de software com esta licença.

O modo mais simples de tornar um programa livre é colocá-lo em domínio público, ou seja, não possui copyright. Isto permite que as pessoas compartilhem o programa e suas melhorias, se elas estiverem dispostas a tal. Mas isto também permite que pessoas não-cooperativas transformem o programa em software proprietário. Eles podem fazer modificações, poucas ou muitas, e distribuir o resultado como um produto proprietário. As pessoas que receberem esta forma modificada do programa não têm a liberdade que o autor original havia lhes dado; o intermediário eliminou estas liberdades.

No Projeto GNU, o objetivo é dar a todos os usuários a liberdade de redistribuir e modificar o software GNU. Se algum intermediário fosse capaz de retirar a liberdade, teríamos muitos usuários, mas esses usuários não teriam liberdade. Então, em vez de colocar o software GNU em domínio público, o tornamos "copyleft". ("copyleft" é um trocadilho com o termo "copyright". Traduzindo literalmente, "copyleft" significa "deixar copiar".) O copyleft diz que qualquer um que distribui o software, com ou sem modificações, tem que passar adiante a liberdade de copiar e modificar novamente o programa. O copyleft garante que todos os usuários tenham liberdade.

O copyleft também fornece um incentivo para que outros programadores contribuam com o software livre.

Programas livres importantes como o Compilador GNU C++ existem por causa disso.

O copyleft também ajuda os analistas e programadores que desejam contribuir com melhorias para o software livre a obterem permissão de fazer isto. Eles freqüentemente trabalham para empresas ou universidades que fariam qualquer coisa para ganhar mais dinheiro. Um programador pode desejar contribuir suas modificações para a comunidade, mas seu empregador pode desejar transformar as mudanças em um produto de software proprietário.

Quando alguém explica ao empregador que é ilegal distribuir a versão melhorada exceto como software livre, o empregador geralmente decide liberá-lo como software livre em vez de jogá-lo fora.

Para tornar um programa copyleft, primeiro a Free Software Foundation registra o copyright; então ela adiciona termos de distribuição, que são instrumentos legais que garantem a qualquer pessoa os direitos de usar, modificar, e redistribuir o programa ou qualquer programa derivado dele se e somente se os termos de distribuição não forem modificados. Desta forma, o programa e as liberdades se tornam legalmente inseparáveis.

Utilizar os mesmos termos de distribuição para vários programas diferentes torna fácil copiar o código entre vários programas diferentes. Desde que eles todos tenham os mesmos termos de distribuição, não há necessidade de verificar se os termos são compatíveis.

Desenvolvedores de software proprietário usam o copyright para retirar a liberdade dos usuários; a Free Software Foundation utiliza o copyright para garantir a liberdade deles. É por isso que ela reverte o nome, mudando de "copyright" para "copyleft".

A Licença Pública Genérica do GNU é freqüentemente chamada de GPL. A GNU GPL foi criada de modo que você possa facilmente aplicá-la ao seu próprio programa,

se você é o detentor do copyright. Você não tem que modificar a GNU GPL para fazer isso, simplesmente adicione notas ao seu programa que façam referências adequadas a GNU GPL.

8. Software Comercial

Software comercial significa que o software não é livre. Um programa é comercial se ele foi desenvolvido para negócios. Um programa comercial pode ser livre ou não dependendo da sua licença.

No caso de softwares comerciais e proprietários você não compra um software, mas sim sua licença de uso do programa. Percebemos então que o software não pertence propriamente a você, pois ele permanece propriedade do editor (empresa, softhouse, etc.) ou propriedade do autor. As licenças dos sistemas operacionais e softwares proprietários visam apenas a limitação da liberdade. A limitação da liberdade de conhecer o código-fonte, de copiá-lo, redistribuí-lo, alterá-lo conforme às necessidades da sociedade.

9. Software For Free

Se você quer dizer que o seu programa é um software livre, nunca diga que ele é "for free." Esse termo específico significa que ninguém precisa pagar nada por ele, alguém pode modificá-lo e vendê-lo. Para não ter confusão você diz que seu programa é "as free software."

10. Software Freeware

Não use o termo "freeware" isto significa "free software." O termo "freeware" é usado desde os anos 80 para programas executáveis em teste, com código fonte não disponível.

11. Sell Software

Este termo "sell software" significa distribuir cópias de um programa por algum valor, ou seja, pagar para ter o direito de usá-lo, você desenvolve um programa, e quem quiser usá-lo ou obtê-lo paga por isso.

12. Theft

É infringir as leis de copyright. Nos Estados Unidos usá-se o termo "theft" para dizer que as leis foram violadas ou quando um software foi copiado sem autorização ou roubado.

13. Respostas de Stallman sobre o software livre

Essas perguntas foram feitas pela Revista do Linux a Richard Stallman.

RL - Muitas pessoas pensam que seus pontos de vista são muito radicais.

Stallman - Minha posição é firme, consistente e idealista, porém não radical. Está solidamente inclusa em uma tradição de pensamento nos Estados Unidos: de que as pessoas devem ter liberdade, e é bom cooperar com as outras pessoas. Eu aplico essa tradição em uma área e, que não é normalmente aplicada.

RL - Quando você escreveu a GPL, você imaginou que seria tão respeitável e importante nos dias de hoje?

Stallman - Eu não sabia que o software livre teria tal sucesso, fiz apenas o melhor que podia. Geralmente, eu não prevejo o futuro, porque sei que não posso fazê-lo. Especialmente quanto estou lutando por uma boa causa, não tento prever quem vai ganhar. Eu luto da melhor maneira que posso.

RL - E sobre a batalha Open Source X Free Software? O que você pensa sobre o movimento Open Source?

Stallman - O movimento Open Source tenta ganhar o suporte empresarial, apenas falando sobre os benefícios práticos em permitir que os usuários copiem e alterem o software. Eles nunca falam sobre a liberdade como beneficio em si. Eu concordo com que eles dizem, quanto a isso, porém eles não dizem o suficiente. A obtenção do suporte empresarial é útil, mas ainda mais importante é a difusão entre os usuários da idéia de que a liberdade por si só tem seu valor. Para manter nossa liberdade, freqüentemente precisamos nos esforçar para mantê-la e resistir à tentação de desistí-la. As pessoas não farão isso por meros benefícios práticos. Mas farão, se acharem que algo mais importante está em jogo. Portanto, precisamos falar sobre liberdade. O movimento Open Source não faz isso. O movimento Free Software faz. Não acho que o movimento Open Source seja ruim ou que deve desaparecer. Mas o movimento Free Software também é necessário.

14. Documentação do Software Livre

A Free Software Foundation possui dois tipos de licenciamento: um para software e outro para documentação. Assim ao lado da conhecida GPL existe uma outra modalidade de licença, a GNU Free Documentation License (GFDL ou FDL), que, por sua vez, complementa a GNU GPL. Ela não tem ainda o destaque devido, e muitos colaboradores publicam documentos e tutoriais usando uma referência a GPL. Seria importante que a comunidade do software livre no Brasil desse impulso a essa modalidade de licença, já que ao longo de 2000 houve um crescimento

significativo de documentação livre sobre Linux disponível na Web. Os usuários e profissionais querem e precisam aprender; o software livre precisa de documentação livre.

O propósito desta licença é permitir que um manual, livro texto, ou outro documento escrito seja "livre" no sentido de liberdade: assegurar a todo mundo a liberdade efetiva de copiá-lo e redistribuí-lo, com ou sem modificações, de maneira comercial ou não.

Num segundo termo, esta licença preserva para o autor ou para quem publicou uma maneira de obter reconhecimento por seu trabalho, ao mesmo tempo em que não se considera responsável pelas modificações realizadas por terceiros.

15. OPEN SOURCE NOS NEGÓCIOS

O modelo open-source tem muitas ofertas no mundo dos negócios. Muitas companhias estão colaborando para o código aberto.

No futuro, o código aberto pode causar a falência e o fechamento de muitas empresas que sobrevivem do software proprietário.

As primeiras empresas que forem colaboradoras comerciais em um nicho de mercado podem ganhar vantagens substanciais sobre as outras mais tarde. Por que? Porque elas recrutarão os melhores desenvolvedores disponíveis para o recrutamento, o que é limitado. E o provável é atrair os mais melhores colaboradores para investimentos.

O código aberto deve também ser bom para uma redução significativa em custos de produção do software por projeto, pois os códigos podem ser reaproveitados.

No desenvolvimento de software livre, o valor está realmente no serviço e na integração. O valor a ser cobrado é pelo serviço e não pelo software.

16. OPEN SOURCE PARA CONSUMIDORES

Além de todos os ganhos da confiabilidade e da qualidade, o modelo de código aberto tem uma vantagem para o cliente: você não é um prisioneiro. Porque você pode ter acesso ao código fonte, você pode sobreviver sem seu vendedor. E se as taxas de suporte do seu vendedor se tornarem exorbitantes, você pode procurar outro suporte.

Se o seu software livre foi desenvolvido internamente e nunca para a venda no mercado aberto, você deve possuir o código fonte, pois o que acontecerá quando os empregados saírem? Sua companhia estaria em uma furada... com seu software deteriorando. Mas se você possuir o código fonte, alguma outra pessoa pode dar manutenção e modificá-lo.

17. BOM E BARATO – OU MESMO DE GRAÇA. ENTÃO, POR QUE NÃO EXPERIMENTAR?

Esse raciocínio tem levado milhões de usuários, em todo mundo, a aderir ao conceito do free software – que envolve a drástica redução do custo do programa e a liberação de sua cópia, para instalação em todos os computadores de uma rede.

"O software mais usado é o que não é pago. Quando deixa de ser barato, ou de graça, o usuário começa a procurar outras opções", afirma Sandro Nunes Henrique, diretor da Conectiva Informática.

Segundo ele, o preço ainda é um fator fundamental para o usuário brasileiro – e de toda a América Latina – na hora da escolha do programa, ele conta o caso de uma escola de informática, que estava preste a fechar porque não tinha recursos para legalizar as cópias piratas de Windows e MS-Office, instaladas em seus micros. O dono chegou a ligar para a Conectiva, agradecido, depois de ter encontrado a alternativa Linux. "Como ele, muita gente está procurando opções que caibam no próprio bolso", afirma.

Estima-se que 61% dos programas de computador em operação no País são piratas, a prática de se colocar software free no mercado é natural e legal.

18. BUSH PODE APOIAR SOFTWARE LIVRE COM GASTO PÚBLICO

O novo presidente dos EUA, George W. Bush, parece não ter pressa quando o assunto é tecnologia da informação. Ao menos por enquanto a ordem é prolongar a existência do Pitac (Comitê Assessor da Presidência para Tecnologia da Informação).

Esse comitê de alto nível, criado pelo Congresso norte-americano, foi implementado por Clinton em fevereiro de 1997.

Bush prorrogou o mandato do Pitac. Entre os membros do Comitê estão desde Vinton Cerf (um dos pais da Internet) até representantes da Microsoft, IBM, Hughes e Intel, entre próceres da academia, ligados ao desenvolvimento da tecnologia da informação e representantes de entidades sem fins lucrativos.

Pouco antes do édito prolongando o mandato do Pitac, o Comitê enviara a Bush uma carta que resume o estado das artes e faz importantes sugestões.

Destaca-se a proposta de usar o poder de compra do governo para estimular o desenvolvimento de software de código aberto (software livre).

No entanto, o confronto entre software livre e códigos proprietários (cujo desenvolvimento e propriedade intelectual ficam sob o domínio de uma empresa) tem impactos econômicos e sociais intensos.

No relatório finalizado em setembro sobre software livre, citado na carta a George W. Bush, o Pitac insiste

num alerta que vem fazendo desde 1999: o desenvolvimento de software de código aberto deveria tornar-se uma "prioridade absoluta" entre os investimentos do governo federal.

Para países em desenvolvimento, como o Brasil, seria necessário encontrar uma expressão ainda mais forte que "prioridade absoluta".

Afinal, com o desenvolvimento da sociedade da informação, o crescimento no uso de software tende a ser exponencial.

Haverá dólares para pagar todos os royalties? Há competência local para desenvolver software? Há redes comunitárias para o desenvolvimento de software livre, a exemplo da bem-sucedida expansão mundial do Linux?

Por enquanto praticamente inexistem determinações nesse sentido. O debate sobre a sociedade da informação está apenas começando e, nos últimos anos, o Brasil foi palco de uma mini-bolha de Internet comercial.

Figura 3 – Incentivo do GNU/LINUX ao código aberto

19. CONCLUSÃO

Concluímos que o software livre está cada vez mais tomando o espaço do software comercial no mercado. Isto acontece por que os usuários querem softwares baratos, que sejam livres, para que não correrem o risco de serem multados pelo uso indevido de um software proprietário, pois, a lei do software prevê multa de até 3.1 vezes o valor da cópia encontrada e reclusão de um a quatro anos.

Somos apoiadores, do software gratuito. Mas, agora compreendemos que isso não basta. É preciso liberdade de verdade. Precisamos de acesso aos códigos fonte dos programas, para que possamos alterá-los e torná-los mais apropriados para o uso que faremos deles. Isso sim é liberdade. Precisamos ter consciência, também, de que outros querem o mesmo que nós. Por isso, devemos distribuir os softwares que contenham as nossas modificações cobrando por isso ou não, incluindo sempre o código fonte.

AGRADECIMENTOS

Agradecemos a todos os colaboradores do software livre, que nos disponibilizaram informações sobre o assunto e que trabalham diariamente na conquista de um mercado de software que seja justo.

REFERÊNCIAS

www.gnu.org

www.olinux.com.br

www.conectiva.com.br

www.rehat.com

Revista do Linux

Folha de S.Paulo

PARADIGMA DE DESENVOLVIMENTO DE SOFTWARE: "O SONHO DA PERFEIÇÃO"

Lílian de Azevedo
Faculdade de Administração e Informática
lilica_az@bol.com.br

Luciana Campos Bezerra Pereira
Faculdade de Administração e Informática
luciana@embassy-systems.com.br

Resumo — *O presente artigo tem como objetivo questionar o porque das dificuldades do desenvolvimento de software, porque as equipes de desenvolvimento muitas vezes não seguem metodologias de desenvolvimento ou ainda porque elas não conseguem seguir estas metodologias. Fabricar um bom programa de computador é, sem dúvida, um dos maiores desafios da indústria da informática. Satisfazer aos desejos dos usuários, atender a prazos e orçamentos limitados, escolher ferramentas de desenvolvimento que se mantenham atuais ao menos até o fim do projeto e selecionar pessoal qualificado não são tarefas simples.*
Apesar das dificuldades, existe uma grande demanda por software no mercado. Motivadas pelo aumento da competitividade, as empresas buscam no software formas de aumentar a sua produtividade e utilizar mais eficientemente a informação que dispõem, na busca por maiores lucros. Equipes buscam a perfeição no desenvolvimento de sistemas, ou ainda pelo menos, metodologias que satisfaçam suas necessidades. Todavia, uma pergunta ainda persiste na cabeça de membros de equipes de desenvolvedores: porque a perfeição de desenvolvimento de software ainda é um sonho?

Abstract — *This article tries to question why the software development process causes so many doubts and harm. Why, many times, the developers do not follow the patterns and methods created. Create a really good software is the biggest challenge today for the software houses. To satisfy all the users desires, build everything on schedule, right costs, choose the right tools and software, create high performance teams it's a very hard task today. Despite the difficulties, today we have a very large market asking and wanting high quality business software. Motivated by the increase of the competitiveness, the companies take software as a way of increasing productivity and use more efficiently all the information they collect. The teams are looking for perfection on the development process, but all they can find is a method of doing things. Though, a question is still alive in all developer's minds: why this development perfection is still a dream? Will us, sometime, reach this level? I guess no.*

Palavras chave — *Metodologias, qualidade, gerência de projetos, testes, complexidade, manutenção, orientação a objetos, UML, tecnologias, ferramentas CASE e RAD.*

1. INTRODUÇÃO

O sucesso de um software é reflexo imediato da qualidade do seu processo de desenvolvimento, que pode ser uma tarefa construtiva e organizada, bem como pode ser um caos de papéis e pessoas, resultando em inúmeros fatos e problemas no produto final.

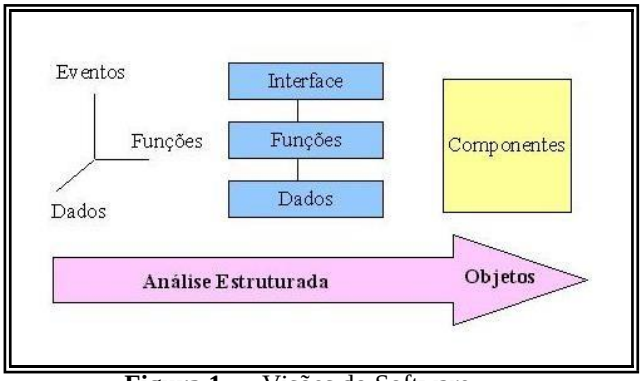

Figura 1 — Visões do Software.

Ser profissional hoje em dia é utilizar metodologias no desenvolvimento de sistemas, fazer valer as vantagens e facilidades que nos propõem e que muitas vezes são deixadas de lado. O que acontece é que nós, profissionais da área, não utilizamos as metodologias que auxiliam no desenvolvimento de software como profissionais de outras áreas fazem, desvalorizando assim nosso trabalho. Devemos valorizar nosso trabalho e a nossa profissão.

Muitas vezes fazemos perguntas como: por que os programas levam tanto tempo para serem concluídos? Por que os custos são tão elevados? Por que não se consegue encontrar todos os erros antes de entregar o aplicativo para o usuário? Por que se tem dificuldade em se medir o progresso enquanto o aplicativo é desenvolvido? As respostas a estas questões podem

ser várias, e certamente algumas não são óbvias, porém certamente uma boa parte delas sentem a necessidade de se adotar práticas formais de trabalho, como a utilização de uma metodologia de desenvolvimento, associada a uma metodologia de gerenciamento de projeto e um programa de qualidade/métricas.

2. METODOLOGIAS

Metodologia de Desenvolvimento de software é uma maneira de se utilizar um conjunto de métodos para atingir um objetivo. Em outras palavras, a metodologia deve definir quais as fases de trabalho previstas no desenvolvimento de sistemas.

Instalar ou implantar uma metodologia em uma organização para desenvolvimento de software é perigoso, por que pode ser que ela não seja apropriada. Uma boa metodologia mostra bons resultados se a introdução na organização for feito com sucesso e aceita por todos os envolvidos.

A adoção da notação UML pelas empresas tem ajudado no controle da documentação dos projetos, fixação de idéias e conceitos sobre o sistema a ser desenvolvido e padronização de desenvolvimento.

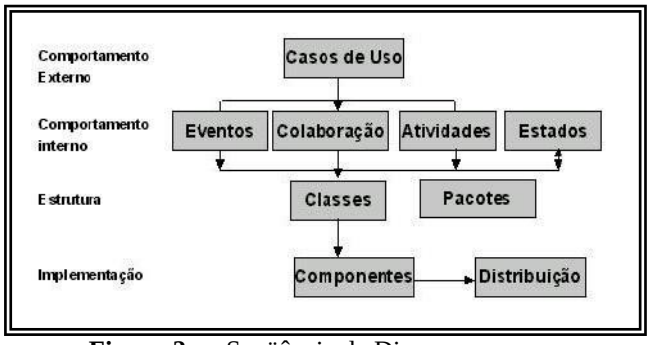

Figura 2 — Seqüência de Diagramas.

Não existem hoje em dia**,** metodologias que satisfaçam adequadamente a todos os tipos de projetos . No entanto, não há uma receita para adaptar, com sucesso, uma metodologia às necessidades de um projeto específico .

Os membros de uma equipe (pessoas envolvidas) de desenvolvimento e (ou) software sempre devem usar suas próprias experiências (vivências) e conhecimentos durante a adaptação de uma metodologia. Pois bem, a maioria dos profissionais de informática não utilizam as técnicas adquiridas durante seu período de formação na faculdade, causando insucessos na aplicação de metodologias impostas por eles mesmos.

Sendo assim, várias empresas procuram adotar metodologias em seus projetos para que assim possam garantir que eles tenham a chance de atingir seus objetivos e suas metas. Portanto, é necessário deixar para trás a idéia de que o desenvolvimento de software é um bicho de sete cabeças. Mas é claro que a aflição que contaminou o desenvolvimento do software não desaparecerá da noite para o dia. Reconhecer os problemas, as causas e desmascarar os mitos do software são os primeiros passos em direção às soluções. As próprias soluções devem oferecer prática ao desenvolvedor de software, e melhorar a qualidade. É mister desvendar os paradigmas de desenvolvimento de software.

Sendo assim, para um pequeno projeto as fases de desenvolvimento podem obter uma combinação para atender às necessidades do projeto e eliminar desempenhos desnecessários. Algumas tarefas específicas de desenvolvimento podem ser omitidas ou reorganizadas. No entanto, a adaptação de uma metodologia pode não somente implicar a mudança no desenvolvimento como pode significar que novas tarefas serão necessárias para satisfazer o projeto como ter um gerenciamento eficaz, organizar o trabalho, e que metodologia será adquirida com responsabilidades, objetivos e mecanismos de controle. Já para um grande e complexo projeto, talvez seja necessário passar por todas as fases de desenvolvimento, conduzindo as tarefas técnicas, uma de cada vez, em uma seqüência apropriada e com todo o rigor que projetos grandes exigem. Por fim, a equipe de desenvolvimento pode enfrentar diversas restrições, como tempo e recursos para desenvolvimento.

O método tradicional de desenvolvimento de software, no qual os programadores sozinhos completavam o projeto já foi ultrapassado. Hoje, temos que lidar com métodos, procedimentos e ferramentas para aumentar a produtividade e qualidade dos produtos. Faz-se necessária a adoção de metodologias para que os requisitos do software sejam atingidos que juntos cooperam para que o desenvolvimento atinja os requisitos pré-estabelecidos.

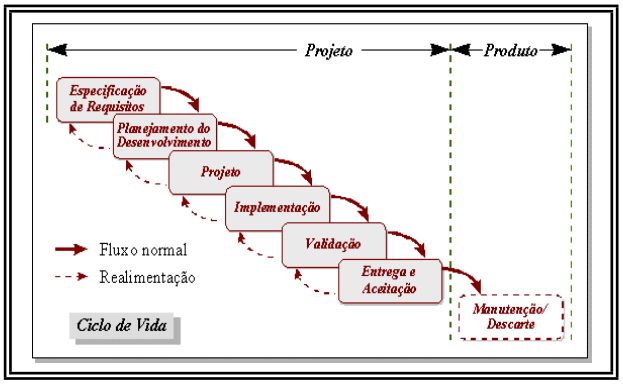

Figura 3 — Ciclo de vida clássico.

Existem vários métodos para o desenvolvimento de software, tais como prototipação, modelo espiral, processo evolutivo (proposto por Meyer), modelo incremental e outras técnicas de quarta geração como a RAD. O ciclo de vida clássico da engenharia de software, também chamado de modelo cascata, ou ainda modelo de processo linear, utiliza uma abordagem sistemática, seqüencial ao desenvolvimento, iniciando no nível do sistema e segue na seqüência de análise, projeto, codificação, teste e manutenção.

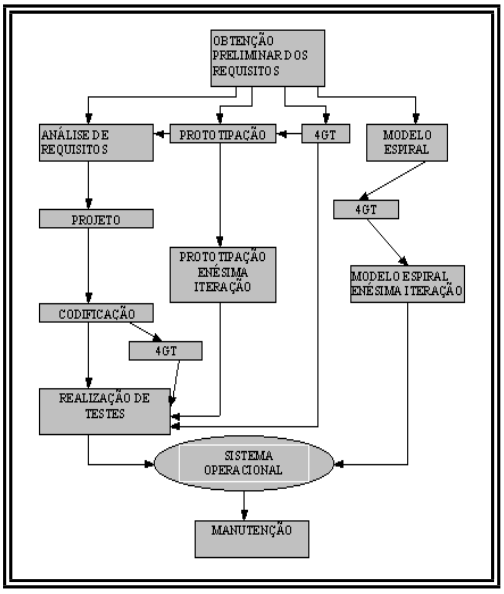

Figura 4 — Combinando paradigmas.

2.1 Que ferramenta utilizar?

Equipes prestes a iniciar um projeto de software se defrontam com um velho dilema. Que ferramenta utilizar? Uma ferramenta que adiante o projeto, devido ao curto cronograma? Ou uma ferramenta que padronize a documentação? Duas frentes disputam a atenção dos projetistas de software. A primeira defende o uso de ferramentas RAD (Rapid Application Development), para abreviar o desenvolvimento de software; enquanto a outra, propõe uma abordagem metodológica para a criação de um novo programa de computador, fazendo-se uso de ferramentas CASE (Computer-Aided Software Engineering).

Todas as modernas ferramentas de programação visual oferecem um variado conjunto de recursos que permitem construir rapidamente um aplicativo. Com o auxílio de agentes e bibliotecas com centenas de milhares de linhas de código, o desenvolvimento de software torna-se uma tarefa muito mais rápida. Em apenas alguns minutos cria-se uma interface de acesso a um banco de dados com todas as facilidades de uso do mundo Windows, algo que seria impensável a alguns anos atrás.

Entusiasmados pela possibilidade de construir rapidamente um programa, os analistas partem de imediato para a programação do aplicativo, antes mesmo de que o problema esteja entendido. A aceitação de um método para conduzir o desenvolvimento, faz com que o analista seja estimulado a entender o problema e a criar documentos de projeto antes de iniciar a programação. Como os sistemas atuais são bastante complexos, há necessidade de se apoiar o método de desenvolvimento em uma ferramenta CASE. Esta , aparentemente distancia o programa executável das fases iniciais do projeto, e é aí que o RAD cumpre com seu papel.

Muitos consideram que os projetos de software sofrem os problemas de uma tecnologia imatura. Não se começa a construir uma casa sem uma planta, sem um projeto aprovado pelo cliente, ainda que ela seja pré-fabricada. A engenharia de software não é diferente. Antes de se iniciar a construção de um programa deve-se projetá-lo, analisando os requisitos dos clientes, a complexidade , e avaliando as possíveis soluções, que incluem o uso de ferramentas RAD.

Conclui-se que ao iniciar um novo projeto de software não se pode dispensar uma ferramenta CASE para a analisar e gerenciar a complexidade do problema, assim como não se pode deixar de usar ferramentas de programação RAD e suas incontestáveis facilidades para construção de software.

A dificuldade encontrada pelo analista passa a ser agora a de dominar estes novos recursos. Deve-se estar em permanente reciclagem profissional, através de seminários e congressos, complementando o treinamento recebido em cursos formais. Também para manter-se atualizado.

3. COMPLEXIDADE DE SOFTWARE

Por volta dos anos 70, teve início pesquisas sobre a complexidade de desenvolvimento de software e ao mesmo tempo começava a utilização da programação estruturada. Ambas eram resultado de experiências pobres com grandes sistemas, surgindo assim a necessidade de um controle de qualidade de software culminando em duas formas. A primeira envolvendo o desenvolvimento de novos padrões para a programação e a segunda requisitando o desenvolvimento de medidas para monitorar a "complexidade" do código produzido, a fim de obter maior qualidade nos sistemas e facilitar o desenvolvimento de sistemas.

4. O PAPEL DA ORIENTAÇÃO A OBJETOS

Não podemos deixar de comentar um pouco sobre a Orientação a Objetos. A OO é considerada atualmente

como uma tecnologia fundamental no desenvolvimento de softwares. As tendências da computação favorecem a utilização da mesma. Existe uma grande quantidade atualmente de sistemas distribuídos, com interface gráfica para os usuários, arquitetura cliente – servidor, com uma escala elevada da utilização da OO.

A orientação a objetos promete ganhos de produtividade no desenvolvimento decorrentes da construção de sistemas a partir de componentes já existentes (reutilização). Sua tecnologia possibilita também maior modularidade, maior abstração, maior reutilização de códigos, e uma seqüência de vantagens. Ou seja, seus benefícios são: desenvolvimento mais rápido, melhor qualidade, fácil manutenção, redução de custos (devido a reutilização de códigos), maior facilidade de adaptação. No entanto, existem algumas dificuldades ao utilizar a OO, tais como: falta de ferramentas especializadas, disponibilidade de pessoal qualificado, custo, e algumas outras dificuldades que acabam sendo desprezadas graças aos seus benefícios trazidos ao desenvolvimento de software.

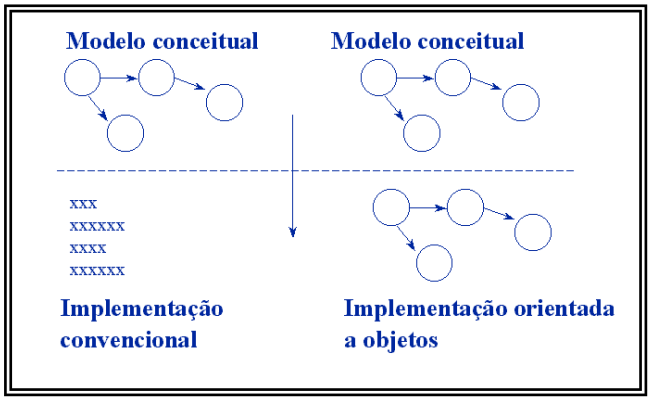

Figura 5 — Modelo conceitual OO

5. TESTES

Com o passar dos tempos, as empresas e as próprias pessoas passaram a ter visões diferentes dos sistemas que encomendavam, foram ficando exigentes e passaram a prestar atenção em detalhes do sistema que somente as pessoas que participaram da equipe de desenvolvimento haviam percebido e muitas vezes nem perceberam. O que acontece é que as pessoas que desenvolvem sistemas não percebem os erros que cometeram ao desenvolvê-lo. Afinal foram elas mesmas que desenvolveram e estes erros acabaram encapsulados em suas visões e idéias e acabaram deixando passar estas falhas.

Porém, são utilizadas duas técnicas de teste que é a caixa preta e caixa branca. O teste de caixa preta é um teste funcional, enfoca o ponto de vista do usuário, desconsiderando a estrutura interna e forma de implementação do sistema. Já o de caixa branca visa testes de código, necessitando do código fonte de todo o sistema. Sendo assim, o teste mais aplicado é o da caixa preta.

5.1 Técnicas de Teste de Software

A atividade de teste é o processo de executar um programa com a intenção de "Descobrir um erro". Um bom caso de teste é aquele que tem uma elevada probabilidade e chance de revelar um erro ainda não descoberto. Um teste bem sucedido é aquele que revela um erro ainda não descoberto. Tendo a aplicação de uma boa metodologia, diminui o tempo, já que suas técnicas filtram esses erros já no desenvolvimento, não chegando a esta fase.

5.2 Estratégias de Teste de Software

A equipe de desenvolvimento do software é sempre responsável por testar as unidades individuais (módulos) do programa, para garantir que cada uma execute a função para a qual foi projetada. Em muitos casos, também realiza testes de integração - a etapa de teste que leva à construção (e teste) do de programa completo. Só depois que a arquitetura de software está completa é que um grupo de teste independente envolve-se.

Em suma, o teste de software é responsável pela maior percentagem de esforço técnico no processo de desenvolvimento do software, resultando em aumento de gastos, demora na entrega do software ao cliente e desgaste da equipe. Mas afinal alguém já relatou algum desenvolvimento onde o que foi previsto aconteceu? Então, se não saiu o que era para sair, é para isto que existe a manutenção de software.

6. MANUTENÇÃO DE SOFTWARE

A manutenção de software foi caracterizada [CAN72] como o "iceberg". Sua presença pode ser responsável por mais de 70% de todo o esforço gasto por uma organização de software. A manutebilidade pode ser definida como a facilidade com que um software pode ser entendido, corrigido, adaptado e/ou aumentado.

As tarefas associadas à manutenção de um software iniciam-se muito tempo antes que um pedido de manutenção seja feito pelo cliente. Inicialmente, uma organização de manutenção deve ser organizada, procedimentos de avaliação e relatórios devem ser descritos e uma seqüência de eventos padronizada deve ser definida para cada pedido de manutenção. Além disso, um sistema de registro de manutenção deve ser estabelecido e critérios de avaliação e revisão devem ser definidos. É importante, manter a documentação do software sempre em dia, de acordo com a última atualização, pois isso afetará a manutebilidade do software.

Modificar software é uma tarefa arriscada. Infelizmente, toda vez que uma mudança é introduzida num procedimento complexo, o potencial de erros cresce. Na correção de um erro, muitas vezes o desenvolvedor não prevê as implicações dessa correção em outros módulos do software, provocando assim efeitos colaterais no software.

Manutenção em códigos desconhecidos é um problema que para as organizações de software. Tais programas são às vezes denominado "código alienígena", porque nenhum membro atual do pessoal técnico trabalhou no desenvolvimento do programa; nenhuma metodologia de projeto foi aplicada e, por conseguinte, o resultado é um projeto arquitetural e de dados ruim; a documentação é incompleta e o registro das mudanças passadas é superficial.

A engenharia reversa é um recurso importante para se recuperar as informações de projeto de um software existente. Significa analisar um programa, num esforço para criar uma representação do programa em um nível de abstração maior do que o código fonte.
A manutenção, a última fase do processo de engenharia de software, é responsável pela maior parte de todos os dólares gastos em software de computador.

7. QUALIDADE

A qualidade de software é uma combinação complexa de fatores que variarão de acordo com as diferentes aplicações e clientes que as solicitam. Para que a qualidade de software seja garantida ela deve ser visada em todo o processo de desenvolvimento, todas as etapas do processo devem ser planejadas, gerenciadas e aplicadas de forma a produzir software com qualidade. O problema é que a prática de utilização de metodologias não manteve passo com a crescente demanda por qualidade, complexidade de funções executadas e avanços tecnológicos em áreas que necessitam de software.

A grande preocupação com a qualidade, tanto do produto quanto do processo que o produz, deve ter como meta à criação de mecanismos que possibilitem que este aspecto se realize a contento de todos os envolvidos. O estabelecimento de um processo de desenvolvimento com atividades e etapas definidas, cada uma com suas entradas e produtos especificados e responsabilidades atribuídas, juntamente com marcos para controle do projeto e manutenção da documentação associada, constitui uma metodologia de desenvolvimento.

7.1Garantias de Qualidade de Software

A garantia de qualidade é uma atividade fundamental para qualquer negócio que gere produtos que são usados e percebidos por outros(cliente). Para se ter qualidade no software, é preciso aplicar qualidade durante todo o trabalho antes do uso do software. Os requisitos de software são a base a partir da qual a qualidade é medida. A falta de conformidade aos requisitos significa falta de qualidade.

Há um conjunto de requisitos implícitos que freqüentemente não são mencionados. Se o software se adequar aos seus requisitos explícitos (solicitado pelo cliente), mas deixar de cumprir seus requisitos implícitos, a qualidade de software será suspeita.

Existem padrões definem um conjunto de critérios de desenvolvimento que orientam a maneira segundo a qual o software. Se os critérios não forem seguidos, o resultado quase que seguramente será a falta de qualidade.

As revisões de software são um "filtro" para o processo de software. Ou seja, as revisões são aplicadas em vários pontos durante o desenvolvimento de software e servem para descobrir defeitos que possam ser eliminados.

Uma revisão técnica é uma atividade de garantia de qualidade de software executada por profissionais da engenharia de software. A subjetividade e a especialização também se aplicam na determinação da qualidade do software.

Para ajudar a resolver esse problema, uma definição mais precisa da qualidade de software é necessária, bem como uma forma de derivar medições quantitativas da qualidade de software para análise objetiva. Uma vez que não existe essa coisa de conhecimento absoluto, ninguém deve esperar medir a qualidade de software exatamente, porque cada medição é parcialmente imperfeita.

Resumindo "A garantia de qualidade de software é o mapeamento dos preceitos administrativos e disciplinas de projeto de garantia de qualidade até o espaço administrativo tecnológico aplicável da engenharia de software".

8. PESQUISA

Segundo Stamdish Group [DEVELOPERS], 31% dos projetos de tecnologia de informação são cancelados antes de serem completados, estimando-se um prejuízo de 81 bilhões. Outros 53% destes projetos ultrapassam, em quase 90% as estimativas iniciais de custo e prazo, representando um prejuízo de 59 bilhões. Somente 16% destes projetos foram concluídos dentro do prazo e do orçamento previsto. Pela visão do gerente de desenvolvimento, os projetos que compõem os 16% dos projetos de sucesso, foram bem sucedidos. A mesma pesquisa ainda informa que para os projetos de sucesso, somente 42% das funcionalidades previstas foram implementadas.

Portanto, segundo a visão de qualidade do nosso usuário final, 100% dos projetos fracassaram gerando usuários insatisfeitos.

9. CONCLUSÃO

Não há métodos técnicos ou gerenciais capazes de acelerar o desenvolvimento de software, a solução está na disciplina, no esforço, na persistência e no cuidado com o desenvolvimento, o problema não é a lentidão no desenvolvimento do software, mas a rapidez no desenvolvimento do hardware.

No entanto, o sonho da perfeição fica preso à uma cultura organizacional bem definida, bom entendimento, contextualização adequada do problema, objetivos bem definidos, apoio dos gerentes de alto nível, comprometimento dos usuários em todos os estágios, utilização de técnica adequada de desenvolvimento, gerenciamento de mudanças, bons programas de treinamento, dentre outros como saber escolher a metodologia a ser adquirida.

AGRADECIMENTOS

Primeiramente agradecemos a Deus que nos da força para vencermos os obstáculos da vida. Agradecemos a professora Adicinéia por nos proporcionar um alto nível de conhecimento. Agradecemos aos nossos pais por nos possibilitar o estudo, amor, carinho e acreditar em nossos propósitos. Aos amigos por estar sempre conosco, nas horas difíceis, nos consolando e auxiliando.

REFERÊNCIAS

PRESSMAN, R. S.. **Engenharia de Software**. São Paulo,
Makron Books,1995.

ANDRADE, R. **Métricas para Desenvolvimento de Software Orientado a Objetos**. Relatório Técnico, COPPE/UFRJ, 1997.

www.google.com.br
www.developers.com.br
www.altavista.com.br
www.miner.com.br

QUALIDADE NO DESENVOLVIMENTO DE SOFTWARE

Alfredo Tavares da Silva
Embassy Systems Ltda
alfredo@embassy-systems.com.br

Esley Bonomo
Embassy Systems Ltda
esley@embassy-systems.com.br

Leandro Fernandes de Paiva
Embassy Systems Ltda
leandro@embassy-systems.com.br

Resumo – *Atualmente todas as organizações do mundo de uma forma ou de outra estão envolvidas com o desenvolvimento ou uso de software. Então podemos afirmar que não apenas o custo de um projeto é que faz o diferencial mas também a qualidade de um software poderá garantir a permanência ou não do produto no mercado. Mas para que alcancemos a qualidade é preciso mudar o processo de desenvolvimento de software. A implantação de um processo de medição do desenvolvimento é o primeiro passo para que a organização possa atingir um alto índice de desenvolvimento, além de definir metas de melhoria , assim fornecendo um conjunto de informações, permitindo que as empresas possam adotar padrões e melhorias para o nivelamento do desenvolvimento de software.*

Abstract – *All organizations around the world are, one way or another, envolved with software development and it's use for business. So, we can say that the cost of a software project, in this new competitive market and era, is not the only competitive branch today. What makes a software survive is it's quality. But to achieve this software quality we need to change all the development process. The first step is to create a measure process that could let us reach a high satisfaction on the developing phase. The next step is the creation of goals, which can guide the companies to the so said "development patterns" and it's good feedbacks. This way, they can build a solid development process and try to create a baseline for the next projects and softwares.***Palavras-chave –** Qualidade de Software, Certificados, Métricas, CMM – Modelo de Maturidade e Capacidade, Garantia de Qualidade de Software.

1. INTRODUÇÃO

Devido a um mau planejamento do processo de desenvolvimento, mau gerenciamento de projeto, má elaboração de prazos, e que muitas empresas acabam perdendo parte do mercado. Esses fatores e que comprometem a qualidade do produto durante o tempo de desenvolvimento, mas ao final é possível diminuir ao máximo os defeitos encontrados no sistema para se obter uma boa qualidade, vide figura 1.

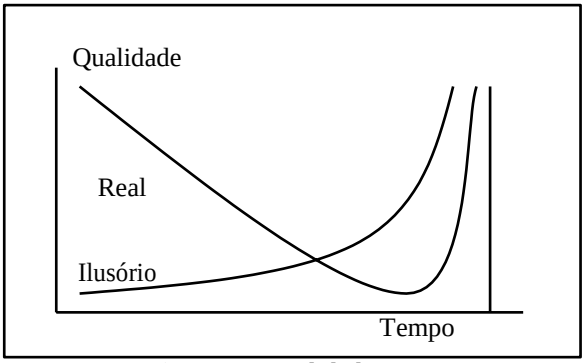

Figura 1 - Qualidade no processo.

Podemos identificar que mesmo com todas as ferramentas para o auxilio ao desenvolvimento, a maioria das organizações não adotam um processo de desenvolvimento. Para mudarmos e solucionar o problema de falta de qualidade é preciso:

- Melhorar o processo de produção de software;
- Melhorar a gerência de projetos de software;
- Um modelo mais amplo do que uma simples metodologia;
- Algo maior do que técnicas, ferramentas e métodos isolados.

De modo geral podemos definir software de qualidade aquele que produz resultados úteis e confiáveis, e fácil de ser modificado e evoluído, mas devemos lembrar que software de qualidade não deve apenas ser correto e ter uma boa documentação, mas além disso, técnicas e métodos bem definidos e especificados, garantindo assim as necessidades dos usuários.

2. DEFINIÇÃO DE QUALIDADE DE SOFTWARE

Encontramos diversas definições para a qualidade de software. Por exemplo:

- "Qualidade é estar em conformidade com os requisitos dos clientes".
- "Qualidade é antecipar e satisfazer os desejos dos clientes atendendo assim os requisitos pedidos".
- "Qualidade é escrever tudo o que se deve fazer e fazer tudo o que foi escrito, de uma maneira correta eficiente e eficaz".
- "Qualidade é um conjunto de propriedades de software a serem satisfeitas em determinado grau, de modo a satisfazer as necessidades de seus usuários".
- "Qualidade de software é conformidade a requisitos funcionais e de desempenho explicitamente declarados, a padrões de desenvolvimentos claramente documentados e a características implícitas que são esperadas de todo software profissionalmente desenvolvido" [PRESSMAN, 1995].
- "Qualidade de software é a totalidade das características de uma entidade que lhe confere a capacidade de satisfazer às necessidades explícitas e implícitas" [NBR ISO 8402].
- "A garantia da qualidade de software é um padrão sistemático e planejado de ações" [SCHULMEYER, 1987].
- "Produto ou serviço de qualidade é aquele que atende perfeitamente, de forma confiável, de forma acessível, de forma segura e no tempo certo as necessidades dos clientes"[FALCONI].

Devemos lembrar que :

- a qualidade depende de um conjunto de propriedades;
- as propriedades são possíveis de serem avaliadas;
- a qualidade depende da satisfação dos diversos tipos de usuários (direto, operador, outro sistema, etc.);
- as propriedades podem ser conflitantes entre si.

3. FATORES DE QUALIDADE

Os fatores que afetam o desenvolvimento de um software de qualidade, podem ser divididos em duas partes:

- Fatores medidos diretamente no desenvolvimento de software.
- Fatores medidos indiretamente no desenvolvimento de software.

Para que possamos medir esses fatores é necessário comparar o software como um todo com uma medida de indicação de qualidade para chegarmos mais perto da perfeição.

Segundo o modelo proposto por McCall e seus colegas, mostrado na Figura 2, existem três aspectos importantes no desenvolvimento de um software de qualidade, que são:

- Características operacionais;
- Facilidade de revisão;
- Facilidade de adaptação a novos ambientes.

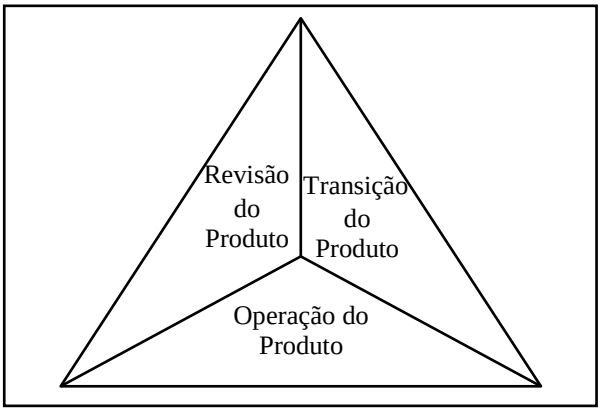

Figura 2 - Fatores de Qualidade.

Com relação aos fatores de qualidade podemos afirmar:

- Revisão do Produto:
  - Facilidade de manutenção;
  - Flexibilidade do sistema;
  - Capacidade de ser testado.
- Transição do Produto:
  - Portabilidade;
  - Reusabilidade;
  - Interface com outros sistemas.
- Operação do Produto:
  - Facilidade de correção;
  - Confiabilidade;
  - Eficiência;
  - Integridade;
  - Usabilidade.

4. BAIXA QUALIDADE DE SOFTWARE

Os problemas relativos à baixa qualidade de software ocorrem devido as seguintes falhas:

- Falha na qualidade de conformidade – diz respeito se o software atende as necessidades do cliente;
- Falha na qualidade de desempenho – quer dizer boa performance, ausência de erros, tolerância a falhas de infra-estrutura e tolerância a erros de usuários.

As principais motivações para atribuir qualidade ao software são:

- Redução dos atrasos do projeto: Imagine o quanto de horas seriam gastas em um projeto com as correções;
- Evitar cancelamento de projeto: A maioria dos projetos é cancelada devido à baixa qualidade;
- Redução dos custos de manutenção: É o que foi dito na redução de atraso dos projetos, quanto menos erro, menos dinheiro seria gasto com manutenção;
- Evitar desgastes com os clientes: Desgaste da imagem da empresa com problemas no software.

Ou seja, todas estas motivações servem para evitar perda de dinheiro.

5. GARANTIA DE QUALIDADE

Podemos dizer que a garantia da qualidade depende diretamente da melhoria do processo de desenvolvimento de software. Para que possamos fazer a mudança no processo devemos antes adotar uma metodologia que siga praticas padronizadas e documentação para avaliarmos cada fase de desenvolvimento, ou seja, obrigatoriamente a melhoria requer mudanças, às vezes mudanças bruscas em relação ao que se tem. Mas sabemos que as mudanças na cultura da empresa, levam um tempo maior para que sejam aceitas e implantadas, que é sem dúvida a pior de todas as mudanças, pois acontece em pessoas, e mudar hábitos é muito complicado.

Para que se tenha um total controle sobre as mudanças no processo de desenvolvimento de software, deve-se levar em consideração a consistência, gerência do projeto e valor do sistema de software.

- Consistência: As mudanças ocorridas durante o processo de desenvolvimento devem ser documentadas para que a equipe fique ciente da responsabilidade de cada um e das fases que se encontra o projeto.
- Gerência de Projeto: Garantindo um bom gerenciamento das mudanças, podemos diminuir a quantidade de surpresas durante as fases do projeto.
- Valor do Sistema de Software: O valor do sistema está ligado diretamente ao gerenciamento do projeto, pois garante economia de tempo e dinheiro, reduz esforços futuros e custos de manutenção.

Podemos dizer que a melhoria do processo pode ser descrita como um paradigma que a empresa deverá adotar para alcançar a qualidade de software.

6. CERTIFICAÇÃO DE QUALIDADE

Certificação é uma norma ou padrão, é a emissão de um documento oficial indicando a conformidade da norma ou padrão [BARRETO, 1997].

Hoje existem várias maneiras para se certificar um produto no mercado, dentre delas podemos citar os certificados de qualidades da série ISO- 9000, e muitos outros.

Antes de se ter uma emissão do certificado, é necessária a realização de um processo de avaliação e julgamento de acordo com uma determinada norma. Existem várias empresas que desenvolvem esse tipo de trabalho, prestando serviço de consultoria auxiliando as empresas a poderem estar aptas para tais certificados.

Os órgãos mais conhecidos de certificação de qualidade de software são os seguintes:

ISO - International Organization for Standardization;

IEEE - Instituto de Engenharia e Eletrônica;

ABNT – Associação Brasileira de Normas Técnicas.

No Brasil, o INMETRO é o órgão do governo responsável pelo credenciamento destas instituições que realizam a certificação de sistemas de qualidade.

7. MODELOS DE QUALIDADE

Quando falamos de qualidade de processo estamos na verdade analisando como o produto foi desenvolvido, acredita-se que se o processo de desenvolvimento for realizado com qualidade resultará em um ótimo produto. Para isto existem diversos modelos que garantem a qualidade no desenvolvimento de software dos quais podemos destacar:

- SMM – Software Management Model;
- CMM – Capability Maturity Model;
- PSP – Personal Software Process;
- SPICE – Software Process Improvement and Capability determination;
- IDEAL – Initiation, Diagnosing, Establishing, Acting e Leveraging;

- ISO 9000-3;
- ISO 12207 – Processo de Ciclo de Vida;
- ISO 9126 – Qualidade de Produtos de Software;
- SPIG – Software Process Improvement Guidebook (NASA).

De todos os modelos apresentados acima, estaremos enfocando especificamente o CMM.

7.1. Modelo CMM – Capability Maturity Model

Este modelo foi inicialmente desenvolvido pelo SEI (Software Engineering Institute) com o objetivo de descrever como um processo de software imaturo pode-se tornar um processo maduro.

O modelo permite direcionar a organização a ter um melhor controle do processo de desenvolvimento de software, e melhorar a cultura de excelência na gestão de Software. Este modelo identifica pontos chaves para alcançar um nível de maturidade, organizada em 5 níveis, que são:

- Inicial;
- Repetitivo;
- Definido;
- Gerenciado;
- Otimizado.

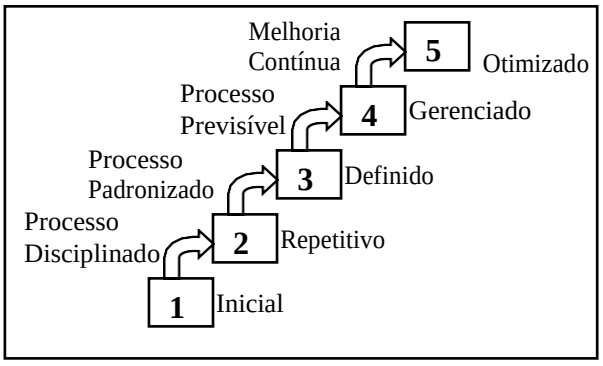

Figura 3 - Níveis de Maturidade

Na figura 3, podemos verificar de forma resumida os cinco níveis do CMM, suas características e ações necessárias para mudança de nível.

Em cada nível do CMM é definido um conjunto de objetivos para se atingir o nível seguinte, onde cada nível de maturidade representa as fases de um processo de melhoria.

A maturidade do processo de desenvolvimento deve ser utilizada com uma metodologia de avaliação e gerência, onde a avaliação tem como objetivo identificar seu estado de maturidade específico e a gerência tem como objetivo estabelecer uma estrutura para implementação de melhorias prioritárias.

7.1.1. Características do modelo CMM

Para atingir um nível especifico de maturidade a organização deve realizar um conjunto de práticas com objetivo de alcançar um novo nível.

Nível 1: Inicial

Tem como principal característica ser considerado um nível caótico, com custo, cronograma e desempenho de qualidades imprevisíveis. Para amenizar este problema deve-se planejar, ou seja, estimar tamanho, custo e cronogramas, controlar mudanças para garantir a qualidade do software.

Nível 2: Repetitivo

Este nível é caracterizado por ser intuitivo, ou seja, com custo e qualidade altamente variáveis, controle de cronograma, métodos do processo e procedimentos informais e genéricos. Por ser um nível intuitivo permite desenvolver definições e padrões de processo, designar recursos de processos e estabelecer métodos.

Nível 3: Definido

Este nível é caracterizado por ser qualitativo, ou seja, possui custos e cronogramas confiáveis, melhoramento imprevisível do desempenho da qualidade. Este nível permite estabelecer medições de processos e objetivos da qualidade quantitativos.

Nível 4: Gerenciado

Este nível é caracterizado por ser quantitativo, ou seja, tem um controle estatístico racional além da qualidade do produto. Este nível permite a definição de: plano e controle quantitativos, ambiente e processo documentados e investimentos tecnológicos justificados economicamente.

Nível 5: Otimizado

Este nível é caracterizado por ter uma quantitativa para investimentos de capital continuado na automação e melhoria do processo. Este nível tem uma ênfase continua na medição do processo e métodos de processos para prevenção de erros.

De todos os modelos existentes hoje em dia o CMM é o mais aceito pelas organizações para o processo de desenvolvimento, sendo utilizado com sucesso por diversas organizações produtoras de software.

O CMM é um modelo proposto como uma poderosa ferramenta que tem como objetivo ajudar as organizações a melhorar seus processos de desenvolvimento de sistemas, que deve ser utilizada para encontrar as fraquezas das organizações, propondo instrumentos concretos para facilitar a implementação prática e transformar boas intenções em planos de ações.

8. IMPLANTAÇÃO DE UM PROCESSO DE MEDIÇÃO DE SOFTWARE

Para gerenciar, controlar e melhorar o processo de desenvolvimento de software é importante a medição. Nesse processo é fundamental a implantação de um processo de medição na organização. Se este processo for bem implantado possibilita a organização identificar, melhorar e padronizar um modelo de desenvolvimento de sistemas de software.

Para a medição de software é indispensável ter estimativas, melhorias continuas no processo e gerenciamento de riscos. Mas não devem ser apenas medidos os atributos ou contados os pontos de função e sim conseguir abstrair informações das medições.

8.1. Razões para medição de software

Existem várias razões para aplicar um processo de medição de software. A frase "você não pode gerenciar o que você não pode medir" tem um significado muito importante para muitos programas de medição.

Hoje as organizações se preocupam muito em melhorar a gestão e o controle de seus processos através da medição de software.

Segundo Tom DeMarco, há pelo menos três razões para se coletar métricas:

- Para descobrir fatos a respeito do mundo;
- Para direcionar nossas ações;
- Para modificar o comportamento humano.

DeMarco, diz ainda, que o objetivo de um processo de medições é prover métricas de descoberta para o melhoramento continuo e medição do gerenciamento de risco. Para isso ele sugere um espectro de métricas que diz que quanto mais à direita você estiver no espectro maior será o sucesso, veja a figura 4.

Modificação de _ Direcionamento _ Descoberta Comportamento

Figura 4 – Espectro de Métricas de DeMarco.

As principais razões para medir software são:

- Formar uma base para estimativas;
- Determinar se as metas de produtividade e qualidade do processo estão sendo atingidas;
- Avaliar os benefícios de novos métodos e ferramentas de software;
- Melhorar o relacionamento com os clientes;
- Melhorar a gerência de contratos de software e relacionamentos com terceiros;
- Reduzir o risco de pressão excessiva do cronograma;
- Melhorar a gerência de projetos de desenvolvimento de software.

Os objetivos de se fazer medições de software são para entender, prover e controlar o processo de desenvolvimento de sistemas de software.

8.2. Classificação das métricas

Devem ser utilizadas no processo de medição, definições baseadas nas necessidades de informação de cada nível organizacional. Isso é levantado a partir de informações das áreas interessadas. Para isso devemos levar em consideração os seguintes tópicos:

- Fornecer resultados consistentes;
- Ser de fácil aprendizado e compreensível pelas partes interessadas no negócio;
- Servir para estimativas;
- Permitir automatização;
- Possibilitar obtenção de séries históricas.

Segundo Jones, existem três tipos de informações presentes em processo de medição de software:

- Dados HARD – refere-se à alta precisão dos dados;
- Dados SOFT – refere-se a opiniões humanas;
- Dados de Padronização – determina se o projeto esta de acordo com a meta.

8.3 – Problemas encontrados

Apesar dos benefícios de um processo de medição de software, existem muitas dificuldades e problemas encontrados:

- É muito comum nas organizações que a métrica se transforme em um objetivo, ao invés de uma informação útil;
- Pouca noção de como analisar e interpretar os dados coletados, deixando de utilizar suas medições;
- Dados coletados são inválidos, e assim permitindo uma má interpretação;
- Pessoas não são motivadas para coletar dados.

9. CONCLUSÃO

Devido à diversidade e complexidade dos modelos de qualidade de software encontrados hoje nas organizações, existe uma grande dificuldade na escolha de qual método será o mais adequado para cada sistema de software a ser desenvolvido, conseguindo assim atender as necessidades do cliente. Entre elas encontramos o CMM, que permite ter um controle do processo de desenvolvimento e da gestão da empresa. Com isso é possível atribuir a organização níveis de

maturidade indicando que ela possui um conjunto de práticas que garantem a qualidade do produto.

O desenvolvimento de software ainda não é uma tarefa fácil. Porém, pesquisadores preocupados com este problema vem desenvolvendo várias ferramentas, metodologias, métricas e muitas outras técnicas, com o único objetivo de desenvolver um software com qualidade, que atenda as necessidades de uma forma ampla e correta. Com certeza teremos que trabalhar muito ainda até chegarmos a um ponto que nos leve bem próximo da perfeição no processo de desenvolvimento de software, caso isso seja possível. Contudo, pela própria característica da evolução da tecnologia, muita melhoria já está por vir para auxiliar as organizações no desenvolvimento de software com qualidade.

AGRADECIMENTOS

Agradecemos primeiramente a Deus e aos nossos pais que acima de tudo acreditaram em nós, não desistiram nem mesmo na hora em que pensávamos em desistir de um sonho que poucas pessoas no Brasil podem desfrutar, ter um diploma de nível superior, mesmo com todas as dificuldades sofridas durante esses 4 anos de luta.

Agradecemos a Prof. Adicinéia Aparecida Oliveira por nos dar a oportunidade de elaborar este artigo, não só para obtenção da nota de sua matéria, mas também a possibilidade de publicá-lo na 1ª. Revista Científica da FAI – INICIA e por incentivar o nosso ingresso no mestrado.

Agradecemos a empresa Embassy Systems pelo apoio ao nosso desenvolvimento e formação acadêmica.

REFERÊNCIAS

[PRESSMAN, 1995] Roger S. Pressman, Engenharia de Software, 3ª ed., Editora MAKRON Books.

[BASILI] Victor R. Basili, Gianligi Caldeira, H. Dieter Rombach – The Experience Factory, Institute for advanced Computer Studies.

[BEAFOUND, 1997] Clifton Eduardo Clunie Beaufond, Claúdia Maria L. Werner e Ana Regina C. da Rocha – Manual para controle da Qualidade de Especificações Orientada a Objeto. Programa de Engenharia de Sistemas e Computação da Universidade Federal do Rio de Janeiro.

Documentação oficial da UML. Disponível na internet no endereço: www.rational.com/uml.

[WANGENHEIM, 2000] – Christiane Gresse Von Wangenheim, Melhoramento de Qualidade e Produtividade de Software, 2000GeNESS.

CMM – MODELO DE CAPACITAÇÃO DE MATURIDADE

Fabrício Bruno
Faculdade de Administração e Informática
fabrício_bruno@bol.com.br

Fernanda Barros
Faculdade de Administração e Informática
fe_barros@yahoo.com.br

Resumo — *CMM ou modelo de capacitação de maturidade de uma organização para a produção de software é um conjunto de conceitos e práticas que visam aperfeiçoar o processo de software como um todo.O CMM procura orientar a organização no sentido de implementar a melhoria contínua do processo de software, e o faz através de um modelo de cinco níveis, priorizado de forma lógica as ações a serem realizadas. Quanto maior o nível, maior a maturidade da organização, o que se traduz em maior qualidade dos produtos final, prazos e custos mais baixos e maior previsibilidade em cronogramas e orçamentos.Os cinco níveis são : Iniciar, Diagnosticar, Estabelecer, Agir, Implantar. Sendo seu objetivo desenvolver software de qualidade, com produtividade, dentro dos prazos estabelecidos.Com o CMM, você consegue identificar e definir o status atual do desenvolvimento, ou seja, o nível da maturidade que deseja alcançar, estabelecendo-o como meta e tomar ações para reduzir a distância entre o status atual e o almejado.*

Abstract —*CMM or model of qualification of maturity of an organization for the software production is a set of practical concepts and that they aim at to perfect the software process as a whole. The CMM look for to guide the organization in the direction to implement the continuous improvement of the software process, and it makes it through a model of 5 levels, prioritized of form logical the actions to be carried through. How much bigger the level, greater the maturity of the organization, what it is expressed bigger lower stated period, product quality final and cost and bigger previsibility in chronograms and budget. The five levels are: To initiate, To diagnosis, To establish, To act, To implant. Being its objective to develop quality software, with productivity, inside of the established stated periods. With the CMM, you obtain to identify and to define the current status of the development, or either, the level of the maturity that she desires to reach, establishing it as goal and to take actions in the distance to reduce between the current status and the longed for one.*

Palavras Chaves — CMM, ACP's, SEPG,SEI, SQA, KPA.

1. INTRODUÇÃO

CMM (Capability Maturity Model), ou modelo de capacitação de maturidade de uma organização para a produção de software é um conjunto de conceitos e práticas que visam aperfeiçoar o processo de software como um todo, seja do ponto de vista gerencial ou de engenharia de software, aumentando a confiabilidade dos resultados, a pontualidade dos compromissos e a precisão dos orçamentos, principalmente.O CMM, que difere de outros modelos pela sua ênfase na melhoria contínua de processos, possui cinco fases, atividades e recursos necessários para tomar iniciativas de melhoria de processo com sucesso:

Iniciar - através de estímulos visando a melhoria é estabelecida uma infra-estrutura para a melhoria, com o patrocínio da gerência sênior;

Diagnosticar - é produzido um diagnóstico documentado das práticas correntes e são geradas recomendações para melhorá-las;

Estabelecer - são estabelecidas e documentadas as estratégias e prioridades, são identificados os processos e recursos necessários. É elaborado um plano de ação;

Agir - é realizada a melhoria de acordo com o planejado. São definidos processos e medições, são planejados e executados projetos piloto e a instalação é planejada, executada e acompanhada;

Implantar - são documentadas e analisadas as lições aprendidas e é revista a proposta organizacional.

O objetivo do CMM é desenvolver software de qualidade, com produtividade, dentro dos prazos estabelecidos e sem necessitar de mais recursos do que os alocados tem sido o grande desafio da Engenharia de Software. A principal causa dos problemas, apontada pelos especialistas é a falta de um processo de desenvolvimento claramente definido e efetivo. Conhecer processos significa conhecer como os produtos e serviços são planejados produzidos e entregues.

O CMM enfatiza a documentação dos processos, seguindo a premissa de que para construir ou realizar alguma melhoria no processo é preciso conhecê-lo e entendê-lo e que a qualidade de um produto é reflexo da qualidade e gerenciamento do processo utilizado em seu desenvolvimento.

2. NÍVEIS DE MATURIDADE

O CMM é composto de cinco níveis de maturidade que determinam qual é a capacitação do processo. Com exceção do nível um cada nível é composto por várias Áreas-Chave de Processo. Estas Áreas-Chave conduzem ao alcance de metas de melhoria do processo para um determinado nível.

Um nível de maturidade é um patamar evolutivo em definido visando alcançar um processo de software maduro. Os níveis são uma forma de priorizar as ações de melhoria de tal forma que se aumente à maturidade do processo de software. Cada nível de maturidade compreende um conjunto de metas de processo que, quando satisfeitas, estabilizam um importante componente do processo de software. Cada nível da estrutura de maturidade estabelece um componente diferente no processo de software, resultando em um aumento na capacitação do processo da organização. Existem cinco níveis de maturidade que caracterizam o nível de capacitação do processo da organização Os cincos níveis são:

Inicial - a capacitação do processo de software de ma organização neste nível é imprevisível, pois o processo ou não está explicitamente definido ou é constantemente alterado com o transcorrer do trabalho.Os cronogramas, orçamentos, funcionalidades e qualidade de produtos são geralmente imprevisíveis.

Repetitivo - o processo neste nível é caracterizado como disciplinado porque o planejamento e o acompanhamento do projeto de software são estáveis, tornando possível repetir desenvolvimentos bem sucedidos. O processo do projeto está sob controle efetivo de um sistema de gerenciamento de projetos, seguindo planos realistas, baseados em desempenhos de projetos anteriores.

Definido - a capacitação do processo de software de organizações neste nível pode ser resumida como padronizada e consistente porque ambas as atividades de gerência e engenharia de software são estáveis e repetíveis. Com relação às linhas de produtos estabelecidas, custos, cronograma e funcionalidades estes estão sob controle e a qualidade de software é acompanhada. Esta capacitação de processo é baseada num entendimento comum das atividades, papéis e responsabilidades num processo de software definido ao longo de toda a organização. A capacitação do processo deixa de ser uma habilidade das pessoas passando a ser uma habilidade da organização.

Gerenciado - a capacitação do processo de software para organizações neste nível é quantificável e previsível uma vez que o processo é medido e opera dentro de limites aceitáveis. O nível da capacitação do processo, neste nível, permite a uma organização prever tendências em processos e em qualidade de produtos dentro de limites quantitativos. Visto que o processo é estável e medido, quando ocorrer algum evento inesperado, as conseqüências poderão ser identificadas e abordadas minimizando o seu impacto.

Otimizado - a capacitação do processo de software neste nível pode ser caracterizada como sendo de melhoria contínua, pois organizações neste patamar estão continuamente se empenhando para melhorar a capacitação do processo. As melhorias ocorrem tanto por avanços incrementais no processo existente quanto por inovações fazendo uso de novas tecnologias e métodos. Melhorias de tecnologias e processos são planejadas e gerenciadas como atividades de negócios rotineiras.

Organizações Maduras	**Organizações Imaturas**
Papéis e responsabilidades bem definidos	Processo improvisado.
Existe base histórica.	Não existe base histórica.
É possível julgar a qualidade do produto	Não há maneira objetiva de julgar a qualidade do produto
A qualidade dos produtos e processos é monitorada	Qualidade e funcionalidade do produto sacrificado
O processo pode ser atualizado	Não há rigor no processo a ser seguido
Comunicação entre o gerente e seu grupo	Resolução de crises imediatas

Tabela 1 – Comparação entre as organizações.

3. CERTIFICAÇÃO CMM

O CMM pode ser utilizado de diversas formas. A mais comum é usá-lo como uma diretriz para implementar boas práticas de engenharia de software. Neste caso selecionam-se ACP's que trarão o maior impacto positivo para a organização e implementam-se as correspondentes ACP's. A vantagem disto é estar construindo um ambiente coerente e homogêneo, ao qual se poderá adicionar mais ACP's à medida que vai se galgando níveis de sofisticação ou de qualidade exigida.

Uma outra forma de usá-lo é na busca de um certificado de conformidade, atestando que se está em determinado e desejável nível de maturidade. Neste caso realiza-se uma avaliação de processo de software. Para tal é necessário que se implemente todas as ACP's do nível e interagindo com uma agência certificadora, se obtenha o certificado desejado. A vantagem disto é poder utilizar este certificado como um indicador idôneo da capacitação de desenvolver software de qualidade assegurada fornecido por uma agência certificadora independente.

Uma terceira forma é examinar um fornecedor de software quanto à sua capacitação em produzir o software desejado no nível de qualidade requerido. Neste caso realiza-se uma avaliação da capacitação do software, sendo que o laudo se torna parte da análise de riscos realizada pela organização contratante. A vantagem disto é dispor de fornecedores de software melhor qualificados.

4. PREPARAÇÃO

Nessa fase é escolhido o grupo de trabalho CMM. Na literatura, ele é conhecido como SEPG - Software Engineering Processo Group. Dependendo do tamanho e complexidade da empresa e da urgência de resultados, o SEPG pode ser representado por um único especialista, uma equipe de três pessoas ou um grupo maior de até dez pessoas. Pode ser composto de consultores especialista, uma equipe de três pessoas ou um grupo maior de até dez pessoas. Pode ser composto de especialistas em CMM contratados, profissionais da própria empresa ou uma equipe mista dos dois. O importante é que eles estejam todos profundamente familiarizados com a filosofia e os termos utilizados nos textos CMM. Também é fundamental que pelo menos parte da equipe tenha dedicação integral ao projeto. Caso sejam utilizados profissionais da casa nesse grupo, eles deverão não só receber bom treinamento, mas também verba para compra de livros, ferramentas de apoio e tempo para estudo. Desnecessário dizer que os consultores devem demonstrar sua experiência e capacidade, mas não seja muito exigente quanto a certificações formais pelo SEI, porque já que elas hoje só podem ser obtidas no exterior, são extremamente raras. O SEPG deve planejar e acompanhar todas as tarefas do projeto de melhoria de capacidade. Eles são o coração do projeto, aquilo que vai garantir que todo o processo de implantação do modelo flua de maneira adequada.

Também nessa fase deve-se fazer circular pela empresa informações a respeito do que é o CMM. É importante divulgar que não se trata de uma reengenharia e que não há previsão de demissões; isso pode evitar resistências. Use formas de divulgação como jornais internos, circulares, e-mail com resumos ou circulação de revistas e livros que tratem sobre o tema.

Durante a preparação, devem ser definidos dois outros grupos: o Comitê Estratégico é uma extensão do SEPG que inclui também a alta gerencia patrocinadora do projeto - quem paga a conta. As pessoas-chave são as que detêm o conhecimento de como é o desenvolvimento e manutenção de software. Essas pessoas são as que deverão ser treinadas e entrevistadas primeiras. Esse grupo tende a mudar, à medida que temos melhor compreensão dos processos de software.

O Comitê Estratégico deve aprovar um cronograma de trabalho que será de responsabilidade do SEPG acompanhar e atualizar esse cronograma. Ë importante que esse documento exista e que seja levado a sério, mas não deve se esquecer que ele é um instrumento, e não o objetivo final do trabalho. O Comitê Estratégico terá a responsabilidade de modificá-lo sempre que isso se fizer necessário ao processo de avaliação. Um cronograma com as macroatividades funciona melhor do que um excessivamente detalhado. As entrevistas com as pessoas-chave devem ser marcadas de acordo com a necessidade e disponibilidade, mas as reuniões do Comitê Estratégico devem acontecer de acordo com o previsto no cronograma.

5. TREINAMENTO

Deve ser preparado e, preferencialmente aplicado por integrantes do SEPG. Atenção especial deve ser dada à adaptação do CMM à cultura da empresa. Todos os termos (KPs, SQA, metas, habilidade, ciclos de venda, etc)devem ser definidos e explicados, bem como a filosofia geral do CMM, as próximas etapas do projeto e as responsabilidades de cada um. As pessoas-chave devem ser treinadas primeiro, mas não restrinja o número de pessoas envolvidas; treine quantas puder. Mesmo se você não tiver certeza de que usará uma determinada pessoa no projeto, treine-a. Isso pode até evitar problemas.

Uma estratégia do treinamento é a tradução do vocabulário CMM para a cultura da empresa - ou seja, procura-se identificar elementos fundamentais de cada KPA (manuais, políticas, prática, etc) que já são utilizados e difundidos na organização, apesar de os técnicos até então acharem que SQA só serve para engrossar o caldo da sopa de letrinhas! Porém, essa tática só será boa se o SEPG conseguir manter o foco no verdadeiro espírito de cada termo usado pelo SEI, mesmo que isso dificulte um pouco o uso do material gerado na avaliação e publicado na empresa.

Durante os treinamentos, o SEPG deve estimular a participação das pessoas chaves. Deixe que façam perguntas e contem casos de sua própria experiência. Podem surgir fatos relevantes, e a troca de experiência entre a equipe não deve ser desperdiçada, especialmente em empresas que tem diferentes site de desenvolvimento, cada qual trabalhando com um ambiente e características próprias.

É interessante que um componente do SEPG anote informações surgidas nos treinamentos para uso futuro. E, se existir mais de uma turma, use os feedbacks das pessoas chaves para melhor treinamento dos próximos.

6. LEVANTAMENTO DE PROCESSOS

Antes de melhorar alguma coisa, devemos primeiro conhece-la. Uma empresa pode ter várias maneiras de executar uma mesma tarefa, como desenvolver e manter software. Cada uma delas representa um processo. Pode existir um processo para os sistemas corporativos, outro

para os departamentais e um terceiro para desenvolver sistemas de pequeno porte. Mais de uma metodologia pode estar sendo empregada ou nenhuma! Pode haver controle total das bibliotecas de programas no mainframe, nas aplicações cliente/servidor, tudo feito de maneira totalmente artesanal. Ë importante que o SEPG tome conhecimento de todos esses processos.

Esse conhecimento deverá ser provido por reuniões, entrevistas e coletas de documentos. Sempre que um documento for descoberto, ele deve receber um código. O SEPG deve armazenar um modelo ou um exemplar de cada documento. Isso deverá se tornar permanente na empresa.

Quando o SEPG já possuir informações suficientes, deve formalizar esse conhecimento utilizando uma ferramenta descritora de processos. A maneira clássica é os diagrama IDEF0, mas há outras opções. Os diagramas devem ser representados e validados tanto pelas pessoas-chave quanto pelo Comitê Estratégico.

7. APLICAÇÃO DE QUESTIONÁRIOS - SEI/CMM

Faça reuniões com as pessoas-chave utilizando os questionários da SEI. Selecione as pessoas que podem lhe dar mais informação sobre cada Key Process Area. Não é necessário que se use uma reunião por KPA, mas não é produtivo utilizar muitos questionários em uma mesma reunião.

A dinâmica de cada reunião pode ser adaptada. Bons resultados são obtidos com a leitura e explicação de cada questão. Devem ser gastos alguns minutos para que cada participante possa comentar a questão e fazer anotações. O SEPG deve fazer suas próprias anotações.

Após cada reunião, os participantes devem ter alguns dias para entregar suas respostas por escrito. O SEPG deverá comparar essas respostas por escrito. O SEPG deverá comparar essas respostas com suas próprias anotações e preparar as respostas consolidadas. Cada participante deve receber uma cópia dessas respostas, e uma reunião de revisão de métodos deve ser marcada.

8. MAPEAMENTO E MÉTRICAS

Com todas as informações coletadas, o SEPG deve organizá-las de acordo com o modelo CMM. A maturidade de cada KPA deve ser estimada e apresentada de forma gráfica. Nenhuma empresa deve esperar atingir altos índices nessa primeira avaliação, mas sim um resultado médio de 15% para as KPAs de nível 2. Também não é comum encontrar resultados homogêneos, por exemplo , uma empresa pode atingir 40% na KPA de Gerência de Requerimentos, enquanto fica com 1% na de Software Quality Assurance (SQA).

9. CONCLUSÃO

Desenvolver software com uma melhor qualidade 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.

Segundo o CMM, a organização de software não deve iniciar o desenvolvimento a partir de requisitos "informais", isto é, orais e não documentados. Além disso, qualquer modificação deve passar por um processo de revisão definido e documentado. Todo requisito deve estar documentado, e deve ser possível a identificação das características do produto final em relação aos requisitos documentados.

Assim, dada uma característica do software, deve ser possível encontrar na documentação do software o requisito que gerou esta característica.

Cada vez mais encontramos profissionais da área, em todos os níveis, que começam a preocupar-se com esta questão tão complexa e delicada.

Após anos de promessas não cumpridas de solucionar-se os problemas de tecnologia através de mais tecnologia, parece que começa a haver um movimento de compreensão de que os problemas de desenvolvimento de software não são tecnológicos, mas gerenciais e organizacionais.

É importante citar por quê tão poucas organizações estão no nível 2. Qualquer um que tenha alguma experiência na área sabe que a maioria das atividades para o desenvolvimento de software são realizadas de forma aleatória, ou, na maioria das vezes, não são executadas de forma alguma.

No entanto, são todas atividades necessárias para que exista um mínimo de ordem no desenvolvimento de software, e será difícil defender a tese de que é possível desenvolver software com qualidade na ausência de qualquer uma das atividades descritas.

Fica claro, portanto, que, em vez de burocracia sem sentido, padrões e processos bem definidos e usados podem fazer muito pela qualidade dos processos de desenvolvimento de uma empresa, aumentando a qualidade dos produtos e reduzindo prazos e custos por conta de muito menos "retrabalho", que seria necessário como decorrência de um ambiente caótico.

AGRADECIMENTOS

Agradecemos em primeiro lugar a Deus, que sabe nos amar da maneira mais pura existente no mundo.

Agradecemos aos nossos pais, que com luta e dedicação nos ensinaram o dom de apreciarmos o fantástico mundo da educação

Em especial, agradecemos nossa professora Adicinéia.

REFERÊNCIAS

CHAVES, Kival Weber e CAVALCANTI, Ana Regina da Rocha. "Qualidade e Produtividade em Software". (3nded.) Makron Books.

http://www.altavista.com.br
http://www.cade.com.br
http://www.todobr.com.br

AGENTES INTELIGENTES MÓVEIS

André Vinícius Alvarenga Alves
Faculdade de Administração e Informática
deco_aa@yahoo.com

Josenete Flórido Andrade
Faculdade de Administração e Informática
josi_fandrade@yahoo.com

Resumo — *A utilização da tecnologia de agentes móveis inteligentes tem sido campo de extensas pesquisas por grandes empresas que se esforçam em disponibilizar aplicações práticas que facilitem a interação do usuário com sistemas informatizados. No modelo clássico de gerenciamento de redes, um agente é um módulo de software capaz de interagir com um dispositivo de rede e recolher informações importantes para o gerenciamento do mesmo. Um gerente pode então solicitar aos agentes tais informações de gerenciamento através de um protocolo de gerência (SNMP, CMIP). A tecnologia de Agentes Móveis tem sido reconhecida como uma técnica bastante eficaz no desenvolvimento e gerenciamento de sistemas Data Mining, principalmente quando se trata de extrair conhecimento em grandes bases de dados localizadas em nós diferentes da rede.*

Abstract — *The use of the technology of intelligent mobile agents has been a field of extensive research for great companies who if strengthen to arrange practical applications that make easy the interaction of the user with information systems. In the classic model of management of networks, an agent is a module of software capable to interact with a network device and to collect important information for the management of exactly. A manager can then request to the agents such information of management through a management protocol (SNMP, CMIP). The technology of Mobile Agents has been recognized as one sufficiently efficient technique in the development and management of systems Data Mining, mainly when it is treated to extract knowledge in great databases located in terminals different of the network.*

Palavras-chave — *Agentes móveis inteligentes; Gerenciamento de Redes, Data Mining.*

1. INTRODUÇÃO

Agentes móveis é uma emergente tecnologia que promete tornar sistemas distribuídos mais fáceis de projetar, implementar e manter. Agentes móveis não estão restritos aos sistemas em que iniciaram sua execução, pois eles têm a habilidade de se transportar de um sistema para outro através de uma rede. Este recurso permite um agente móvel mover-se para o sistema que possui um objeto com o qual o agente deseja interagir, obtendo a vantagem de residir na mesma máquina ou rede do objeto [ARI 98].

Essa tecnologia é, atualmente, uma das áreas de pesquisas que representa um grande interesse em desenvolvimento de novas aplicações. Ela expõe ao usuário facilidades que são baseadas em conceitos da inteligência artificial distribuída.

No dicionário "The Random House", Marvin Minsky define agente como: 1. uma pessoa ou negócio autorizado a agir em nome de outra. 2. uma pessoa que gerencia ou trabalha em uma agência. 3. um representante de firma, particularmente em se tratando de uma agência de viagens. 4. uma substância que causa reação química.

Um agente pode ser definido como algo ou alguém que age representando outra parte, com propósito de desempenhar funções específicas em benefício da parte representada. Um agente de software é um programa computacional que realiza tarefas para seu usuário.

A definição mais geral sobre agentes refere-se a agentes como: uma entidade real ou virtual que emerge num ambiente onde pode tomar algumas ações, que é capaz de perceber e representar parcialmente esse ambiente, que é capaz de comunicar-se com outros agentes e que possui um comportamento autônomo que é uma conseqüência de sua observação, seu conhecimento e de suas interações com outros agentes.

Agente é uma entidade cognitiva, ativa e autônoma, ou seja, que possui um sistema interno de tomada de decisões, que age sobre o mundo e sobre os outros agentes que o rodeiam e, por fim, que é capaz de funcionar sem necessitar de algo ou de alguém para o guiar (tem mecanismos próprios de percepção do exterior).

Embora não haja ainda um consenso sobre uma definição formal do que seja o agente tal que englobe todo o espectro possível, algumas características esperadas foram estabelecidas. A analogia feita com agentes no mundo real nos leva a conceituar um agente como uma entidade ativa, sempre ao lado do usuário e que possui conhecimento especifico sobre um determinado domínio.

De posse de bases de conhecimento e de mecanismos de raciocínio os agentes devem ser capazes de reconhecer situação em que devam se ativar, sem que o usuário perceba, ou seja, de forma transparente ao usuário.

2. AGENTES MÓVEIS

Agentes móveis é uma classe especial de agentes que possui a capacidade e a autonomia de migração, podendo se deslocar na rede e migrando de um nó para outro a fim de executar sua tarefa.

A tecnologia de agentes vem mudar radicalmente o modo como o usuário utiliza seu computador, permitindo que o software seja um assistente ao usuário. Esta tecnologia deverá aproximar ainda mais o usuário ao seu computador.

Como todo agente inteligente, um agente móvel precisa ter as seguintes características: autonomia, habilidade de comunicação, capacidade de cooperação, capacidade de raciocínio, comportamento adaptativo e confiabilidade. Além destas características, um agente móvel deve ter, também, mobilidade ou poder de migração.

Uma das propriedades mais atrativas de um agente é a sua capacidade de se mover de um lugar para outro autonomamente. Diferentes tipos ou graus de mobilidade podem existir (Falsarella et al, 1996):

Estático: agente que só será executado na própria plataforma em que foi originalmente criado;

Execução Remota: agente que é enviado para um nó remoto para executar suas tarefas. Este agente inicia sua execução no local remoto sempre a partir do seu ponto inicial;

Migração (com estado): agente que pode parar sua execução em um nó, migrar para qualquer outro nó e continuar sua execução do ponto onde parou antes da migração. Para isso ele deve levar consigo seu estado. Entende-se por estado do agente todas as informações necessárias (valores de variáveis, próxima instrução a ser executada, valores de parâmetros, etc.) para que o mesmo possa dar continuidade a sua tarefa no novo nó.

Segundo LANGE e OSHIMA em [OSH 98], agentes móveis apresentam uma série de vantagens como:

Redução do tráfego da rede - Sistemas distribuídos demandam um grande volume de comunicação (interação) para realizar uma determinada tarefa, principalmente quando há restrições de segurança envolvidas. Agentes móveis permitem reduzir o tráfego da rede, pois permitem despachar tarefas que podem executar suas interações localmente. Agentes móveis podem ainda reduzir o tráfego de dados da rede, pois permitem mover o processamento para o local onde os dados estão armazenados ao invés de transferir os dados para depois processá-los. O princípio é simples: "Mover o processamento para os dados ao invés de mover os dados para o local de processamento".

Oculta a latência da rede - Sistemas críticos necessitam de respostas em tempo real para mudanças no ambiente. O controle desses sistemas através de uma rede substancialmente grande ocasiona uma latência inaceitável. Agentes móveis oferece uma solução, pois podem ser despachados pelo controlador central para realizarem suas tarefas localmente.

Encapsulamento de protocolo - Cada máquina em um sistema distribuído possui seu próprio código necessário para implementar a transferência de dados. Porém, novos requisitos de segurança e eficiência demandam mudanças no protocolo que podem ocasionar problemas na manutenção do código existente. Agentes móveis, por outro lado, podem mover-se para máquinas remotas a fim de estabelecer canais de comunicação baseados em protocolos proprietários.

Execução assíncrona e autônoma - Tarefas podem ser embutidas em agentes móveis que podem ser despachados pela rede. Após serem despachados, os agentes são autônomos e independentes da criação de processo, podendo executar assincronamente. Este recurso é útil principalmente porque um dispositivo móvel (ex. laptops) pode se reconectar na rede para coletar o agente mais tarde.

Adaptação dinâmica - Agentes móveis possuem a habilidade de perceber mudanças no ambiente de execução e reagir autonomamente. Múltiplos agentes podem interagir entre si e se distribuir pela rede, de modo a manter uma configuração ótima para resolver um problema em particular.

Independência de plataforma - Redes de computadores, geralmente são heterogêneas, tanto na perspectiva de hardware como a de software. Agentes móveis são independentes da máquina e também da rede, sendo dependentes somente do seu ambiente de execução, não dificultando a integração de sistemas.

Robustez e tolerância à falhas - A habilidade dos agentes móveis de reagirem dinamicamente a situações e eventos desfavoráveis torna fácil à

construção de sistemas distribuídos robustos e tolerantes a falhas. Se uma máquina está para ser desligada, todos os agentes em execução na máquina podem ser advertidos para que possam ser despachados e continuar suas tarefas em outra máquina da rede.

Agentes móveis são uma atraente alternativa para construções de ambientes distribuídos, principalmente em aplicações como comércio eletrônico, sendo que suas "habilidades" podem ser utilizadas como ilustrado abaixo:

Observação: em um ambiente de investimento um cliente pode despachar um agente móvel para monitorar um mercado eletrônico até que determinado produto atinja um determinado preço, assim retornando e notificando o usuário.

Busca: o usuário pode delegar a um agente para procurar um produto a um determinado preço, assim o agente cliente movimenta-se para um servidor para interagir com outros agentes(vendedores) para a busca do melhor preço.

Organização: em uma aplicação voltada para o entretenimento, um cliente pode delegar a um agente a tarefa de organizar uma agenda. Com isso o agente deve interagir e buscar a melhor solução para cobrir todos os requisitos determinados pelo cliente.

O principal paradigma utilizado atualmente em sistemas de objetos distribuídos é baseado na passagem síncrona de mensagens entre objetos estacionários que interagem utilizando este mecanismo. Entretanto, esse paradigma é incompleto, e necessita ser melhorado adicionando alguns recursos como passagem de mensagem assíncrona, mobilidade de objetos e objetos ativos.

Agentes móveis podem oferecer um paradigma uniforme para objetos distribuídos, englobando passagem de mensagens síncronas e assíncronas, passagem de objetos e objetos móveis e estacionários. Além de suportar os serviços existentes em uma rede, agentes móveis também tornam possíveis novos serviços e novas oportunidades de negócios.

A tecnologia de agentes móveis é promissora e relativamente nova, e muitas pesquisas e esforços estão sendo realizados. Espera-se que diversas aplicações sejam implementadas utilizando-se esta tecnologia.

3. CARACTERÍSTICAS DOS AGENTES MÓVEIS

As características importantes relacionadas à mobilidade de agentes são (Lange, 1996):

Passagem de objeto: quando um objeto se move, o objeto como um todo é passado; isto é, seu código, dados, estado de execução e itinerário de viagem, são passados junto.

Autonomia: o agente móvel contém informação suficiente para decidir o que fazer, aonde ir e quando ir.

Assincronismo: o agente móvel tem seu próprio thread de execução.

Interação local: o agente móvel interage com outros agentes móveis ou objetos estacionários localmente. Se necessário, ele pode enviar agentes mensageiros ou agentes substitutos, que são todos agentes móveis, para facilitar interação remota.

Operação sem conexão: o agente móvel pode desempenhar suas tarefas quando a conexão da rede está aberta ou fechada. Se a conexão de rede está fechada e ele necessita mover-se, ele pode esperar até que a conexão seja reaberta.

Execução paralela: mais de um agente móvel pode ser enviado para diferentes sítios para executar tarefas em paralelo.

4. VANTAGENS DOS AGENTES MÓVEIS

De acordo com (Falsarella et al, 1996), o uso de agentes móveis em certas aplicações pode ser justificado por benefícios e vantagens que eles podem trazer, tais como:

Redução do custo de comunicação: dependendo da quantidade de informações a serem transmitidas pelo agente entre dois nós da rede, pode ser mais eficiente em termos de custo de comunicação enviar um agente para onde estas estão localizadas. Desta forma, o agente pode filtrar e selecionar somente as informações relevantes e transferi-las para o seu nó de origem.

Processamento assíncrono: um agente pode ser enviado através da rede, continuando suas tarefas em outros nós. Enquanto o agente se encontra fora de seu nó de origem não é necessário que este nó permaneça em operação.

Divisão de carga de processamento: computadores com baixa capacidade de processamento podem ser usados eficientemente para armazenar e processar dados que são filtrados e enviados por agentes móveis a partir de grandes servidores de informações.

Flexibilidade: agentes móveis fornecem uma única arquitetura de processamento distribuído que funciona de forma diferente de set-ups estáticos. São uma alternativa para o tradicional modelo cliente-servidor.

5. AGENTES MÓVEIS NO GERENCIAMENTO DE REDES

O gerenciamento de redes se baseia de um do geral em um paradigma centralizado, podendo ser realizado com maior eficiência quando os agentes móveis são utilizados. Os protocolos de gerenciamento SNMP (Simple Network Management Protocol) e CMIP (Common Management Information Protocol) baseiam-se no modelo cliente-servidor, no qual o gerente centraliza as informações e fornece a ordem para executar as ações corretivas, e o agente coleta as informações, armazena-as na MIB (Management Information Base) e executa as ordens do gerente. Devido à centralização, o gerenciamento não é escalável quando o tamanho ou a complexidade da rede cresce e, neste modelo, uma falha em um ponto da rede ou no gerente pode deixá-la sem gerenciamento. Os agentes móveis descentralizam o processamento e o controle, reduzindo o tráfego da estação de gerenciamento, tornando assíncrona a comunicação com a estação de gerenciamento (bom quando há enlaces não-confiáveis ou com grandes perdas), permitindo um balanceamento de carga de processamento e aumentando a flexibilidade do comportamento dos agentes de gerenciamento.

Para cada dispositivo gerenciado deve obrigatoriamente existir um agente que reporte um conjunto de informações do dispositivo. Este agente está então associado ao dispositivo de que captura informações. Não existe a possibilidade do agente interagir com outros dispositivos recolhendo informações dos mesmos. Neste sentido, um agente é estático e comunica-se apenas com o gerente e com o dispositivo associado, vide figura 1.

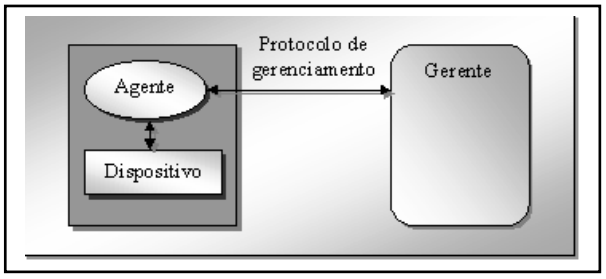

Figura 1 - Modelo gerente/agente de gerenciamento.

Um agente móvel difere muito de um agente padrão. Primeiramente um agente móvel é definido como sendo um *agente de software*. Um agente de software é uma entidade autônoma que possui a capacidade de *aprender* e *cooperar* com outros agentes de software. Além disso, um agente móvel é um *agente de software móvel*, capaz de mover-se baseado em algum padrão ou regra. Esta mobilidade é particularmente interessante porque permite que um agente possa atuar em diferentes localizações de uma rede.

Cada agente móvel possui um conjunto de modelos que definem seu comportamento. Existe um modelo para o *ciclo de vida* do agente, um modelo *computacional*, um modelo de *segurança* e um modelo de *comunicação* [BIE98]. O ciclo de vida de um agente define como o mesmo se comportará durante sua existência, em que circunstâncias haverá clonagem do agente, quando uma instância deve ser excluída, etc. O modelo de comunicação não define um protocolo, como o SNMP, por exemplo, mas sim a forma como os agentes se comunicam entre si e com o ambiente. Além dos modelos citados, o modelo mais importante de um agente móvel é o modelo de *locomoção* que define como um agente move-se na rede. De acordo com este modelo, um agente é capaz de decidir, por exemplo, qual o próximo nó de uma rede a ser visitado. Se isso já ocorreu que outra ação de locomoção deve ser tomada.

As capacidades de aprender e cooperar dos agentes móveis tem impacto direto no comportamento dos mesmos. Isso significa que um agente é capaz de analisar os dados coletados e com isso tomar alguma decisão, mas principalmente registrar essa ação internamente e poder assim agilizar futuras operação. Isto é, aprender. Além da interação com os dispositivos de rede um agente deve interagir com outros agentes, para que as tarefas possam ser executadas cooperativamente. Esta interação permite ainda que um agente possa aprender não apenas com as suas ações, mas também através do conhecimento das ações de outros agentes. Isto é: colaboração.

Como exemplo, podemos citar um sistema de gerenciamento onde um conjunto de agentes móveis é responsável pela detecção de falhas de performance em segmentos de uma rede. Um segundo conjunto de agentes é responsável por alterar parâmetros de configuração do conjunto de dispositivos que podem estar gerando as falhas de performance.

Os agentes cooperam entre si no sentido de verificar segmentos distintos de uma mesma rede. Quanto mais agentes forem usados, mais fácil será a detecção dos problemas, pois haverá mais áreas de uma mesma rede sendo analisadas ao mesmo tempo (fig 2).

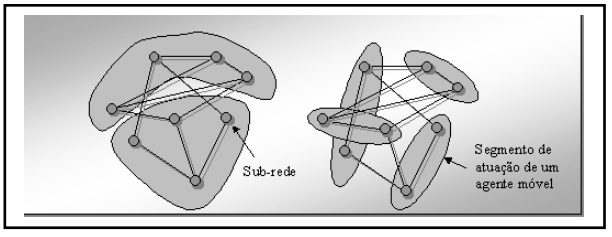

Figura 2 - Exemplo de agentes móveis utilizados na detecção de problemas de performance em uma rede.

O primeiro grupo de agente é então "lançado" em uma rede. De acordo com seu modelo de locomoção, os agentes vão recolhendo dados de performance a

analisando os mesmos para a detecção de problemas. Sempre que um problema de performance for detectado, o agente do primeiro grupo, que detectou o problema poderá informar esta situação a um agente do segundo grupo. O segundo grupo de agentes é então utilizado sempre que um problema for detectado.
Os agentes do primeiro grupo informam o problema existente. Ações de reparo são tomadas de acordo com a análise dos dados de performance. Se necessário agentes especializados podem locomover-se até um dispositivo problemático e alterar os parâmetros de funcionamento necessário. A falha pode ocorrer, entretanto por uma determinada combinação de parâmetros de dispositivos diferentes. Os agentes responsáveis pelo reparo devem reconhecer esta situação e proceder com as alterações necessárias para a solução do problema.

Aparentemente os agentes móveis possuem uma complexidade grande, por realizarem tarefas complexas como as citadas anteriormente. Entretanto, isso não necessariamente é verdade. Cada agente de um sistema pode ser muito simples, executando um conjunto de tarefas bem limitado. Por outro lado, o comportamento resultante da interação entre estes agentes simples é capaz de solucionar problemas complexos de uma rede. Isto é, um agente móvel sozinho é muito simples, mas o comportamento resultante de um grupo destes agentes é complexo. Isso facilita muito a tarefa de criação e manutenção dos agentes por parte do usuário, sem que se perca com isso capacidade de resolução dos problemas.

Por fim uma capacidade importante dos agentes móveis é sua comunicação com uma base de gerenciamento. A base de gerenciamento é um computador principal que substitui agora a estação de gerenciamento. Esta base é ponto inicial do gerenciamento e é ela a responsável pela colocação inicial de agentes móveis em uma rede. Como os agentes são autônomos, depois de criados este tem vida própria e são capazes de executar tarefas apenas com seu próprio ferramental. Tarefas mais simples sequer precisam ser reportadas a base de gerenciamento. Por exemplo, quando um agente móvel detecta uma interface que se encontra desativada, este pode ativar a interface problemática sem avisar a base. Porém, existem outras tarefas que precisam ser reportadas ao gerente da rede, que por sua vez interage através da base de gerenciamento. Para estas tarefas, os agentes móveis devem ser capazes de se comunicar com a base, enviando resultados de suas tarefas, relatórios de atividades e logs de execuções. Com estas informações, a base de gerenciamento pode informar ao usuário o estado atual da rede gerenciada.

Resumindo, um agente móvel deve possuir as seguintes características para executar tarefas de gerenciamento de uma rede:

- Devem ser autônomos;
- Possuir um tempo de vida determinado;
- Executar tarefas simples;
- Cooperar entre si;
- Possuir mobilidade;
- Possuir uma inteligência coletiva;
- Comunicar-se com a base de gerenciamento.

6. Aplicação da Tecnologia de Agentes Móveis no Desenvolvimento e Gerenciamento de Sistemas Data Mining

Data Mining surgiu com o intuito de revelar as informações estratégicas escondidas em grandes bases de dados, através da pesquisa dessas informações e da determinação de padrões, classificações e associações entre elas. Essas informações valiosas podem ser utilizadas em muitas áreas, tais como: análise de risco, tomada de decisão, marketing direto, controle de qualidade de produção industrial, etc.

Data Mining significa um processo não trivial de extração de informações de dados previamente desconhecidas e potencialmente usuais em grandes bases de dados. Com Data Mining, o sistema pesquisa os dados e determina padrões, classificações e associações, enquanto o analista determina o que fazer com os resultados (IBM, 1998).

Os resultados podem ser aplicados a gerenciamento de informação, processamento de pedidos de informação, tomada de decisão, controle de processo e muitas outras aplicações.

Segundo Simoudis, em (Simoudis, 1996), Data Mining tem duas formas. Data mining dirigida à verificação, que extrai informação no processo de uma hipótese sugerida por um usuário. Ela envolve técnicas tal como análises estatística e multidimensional. Data Mining dirigida à descoberta, que usa ferramentas, tais como clustering simbólico e neural, descoberta de associação, indução supervisionada para extrair informação automaticamente.

7. Considerações sobre o uso da Tecnologia de Agentes Móveis em Data mining

A aplicação de agentes móveis na busca em bases de dados corporativas requer, em geral, a solução de algumas questões (Mitsubishi, 1998).

A primeira é o desenvolvimento de um servidor de agentes. Esta é uma operação direta que requer a seleção de uma plataforma dentro da rede corporativa, talvez ela já exista ou então o sistema pode fornecê-la.

A segunda questão é a necessidade de plataformas de computação portáveis, adequadas para agentes. Uma plataforma de agentes é uma infraestrutura de serviços

instalados em um nó de rede que suporta todo ciclo de vida de um agente (Falsarella et al, 1996).
O terceiro passo é definir credenciais de segurança e permissões que possam ser usadas para identificar agentes e seus usuários para o serviço de base de dados existente, ou quaisquer serviços escolhidos para serem exportados pela corporação. Um processo administrativo é então encarregado de gerenciar e mapear estas permissões e as credenciais que são assinaladas aos usuários.

Finalmente, é necessário criar os agentes que efetivamente executarão as requisições.

Uma estrutura ou plataforma de agentes móveis deve incluir as seguintes funções:

- Gerenciar o ciclo de vida de agentes (criação, destruição, cópia no nó destino do agente e seu contexto).
- Armazenar os dados, o estado de execução e o código do agente.
- Fornecer o controle necessário para garantir proteção e segurança de objetos móveis e de recursos computacionais.
- Possuir um esquema de nomeação único e global para agentes.
- Definir um itinerário de viagem para especificar os complexos padrões de viagem com múltiplos destinos e manipulação automática de falhas.
- Fornecer um mecanismo que permita colaborar e compartilhar informações entre múltiplos agentes, de forma assíncrona.
- Possuir um esquema de passagem de mensagem que suporte livremente assincronismo, bem como comunicação síncrona entre agentes.
- Fornecer um contexto de execução que provê agentes com um ambiente uniforme, independente do sistema de computação atual sobre o qual eles estão executando.
- Garantir a transmissão segura e eficiente de agentes e seus estados de um nó a outro da rede.
- Reduzir overhead de migração, protegendo uma máquina de agentes maliciosos (e um agente de máquinas maliciosas), e isolando o agente de falhas de rede e de máquina.
- Fornecer um mecanismo de controle de ordem de execução dos serviços de agentes nos nós da rede.

A construção de uma plataforma para dar suporte à mobilidade de agentes pode ser baseada nos serviços da OMG OMA (Object Management Group-Object Management Architecture), como ocorre nos projetos MAGNA - (Mobile AGent Architecture) e PAGE (Prototyping an AGent Environment), conforme apresentado em [Falsarella 96]. A arquitetura OMA já tem serviços especificados (como por ex: Life Cycle, Externalization, Naming, etc.), que podem ser utilizados. Esses serviços cobrem a maioria das funções relacionadas acima, sendo necessário apenas definir serviços relacionados às funções de migração dos agentes, tais como (Falsarella, 1996):

- Serviço de Persistência de Agente, para armazenar os dados e o estado de execução do agente;
- Serviço de Segurança de Agente, para realizar as ações necessárias de segurança antes, durante e após o transporte;
- Serviço de Repositório de Implementação de Agentes, para armazenar o código do agente;
- Serviço de Instanciação de Agentes, para instanciar o agente na agência destino e colocá-lo em execução.

8. PARADIGMA DOS AGENTES MÓVEIS

Uma característica chave do paradigma dos agentes móveis é a possibilidade de qualquer computador em uma rede possuir uma combinação de processamento, recursos e conhecimento. O conhecimento não está vinculado a um único computador, ao contrário está disponível para toda a rede.

Como exemplo podemos citar os Aglets, desenvolvidos pela IBM em Tóquio. Os Aglets são agentes móveis Java que suportam os conceitos de execução autônoma e roteamento dinâmico em seu itinerário. Outro exemplo é a linguagem Telescript da General Magic. A linguagem Telescript é uma linguagem orientada a objetos concebida para o desenvolvimento de aplicações distribuídas de grande porte.

Embora se tem sugerido a utilização de agentes móveis para vários tipos de aplicações, o escopo de aplicabilidade de agentes móveis será determinado dependendo-se do desenvolvimento de soluções apropriadas para um conjunto de questões. Parte destas questões está relacionada ao que se pode chamar de *capacidade de controle* de atividades baseadas em agentes móveis. Usuários ou organizações utilizando agentes móveis para executar aplicações em um ambiente distribuído devem ter controle sobre a sua execução. Dois dos aspectos relacionados à capacidade de controle sobre atividades baseadas em agentes móveis, que serão considerados no contexto deste trabalho, são: suporte à *confiabilidade* da execução de agentes móveis e suporte à geração de implementações de aplicações baseadas em agentes móveis *corretas*. Outros aspectos são segurança e contabilização de recursos (*accounting*).

9. CONCLUSÃO

A tecnologia de agentes móveis tem provado ser uma técnica bastante eficaz no desenvolvimento e gerenciamento de sistemas devido à capacidade de mobilidade que todo agente móvel deve ter, podendo navegar por redes de longa distância. Com o poder de migração um agente pode parar sua execução em um

nó, migrar para qualquer outro nó e continuar sua execução do ponto onde parou antes da migração. A capacidade de mobilidade também possibilita a execução remota, ou seja, um agente pode ser enviado para um nó remoto para executar suas tarefas.

Sendo um agente capaz de mover-se de um nó a outro da rede, pode-se atribuir funções de extração de dados a ele e fazer com que ele seja enviado ao(s) nó(s) onde esses dados estão localizados para que ele faça uma seleção dos mesmos, havendo assim uma melhor divisão de trabalho entre os nós da rede e uma redução do custo de comunicação.

O uso da tecnologia de agentes móveis em sistemas exige que algumas questões importantes sejam solucionadas e que seja definida uma estrutura ou plataforma de agentes para o desempenho das funções necessárias.

Este artigo teve como objetivo verificar o quanto à tecnologia de agentes móveis pode ajudar no desenvolvimento e gerenciamento de sistemas. Para isto, foram apresentados conceitos, vantagens, técnicas, etc. de agentes móveis.

REFERÊNCIAS

IBM. Data warehousing concepts.
URL: http://iws.as400.ibm.com/db2/dataware.html, 1998.

GENERAL MAGIC. Tabriz. URL: http://www.genmagic.com/tabriz, 1996.

LANGE, D.B. & CHANG, D.T. IBM aglets workbench: programming mobile agents in java. URL: http://aglets.trl.ibm.co.jp/whitepaper.html, 1996.

LANGE, D.B. & CHANG, D.T. IBM Aglets Workbench.
URL: http://aglets.trl.ibm.co.jp/whitepaper.html, 1998

LANGE, D.B. Java aglet application programming interface.
URL: http://aglets.trl.ibm.co.jp/JAAPI-whitepaper.html, 1998.

SISTEMAS DE PESQUISAS INTELIGENTES NA INTERNET

Tânia Leite de Vitor
Faculdade de Administração e Informática
tania_vitor@hotmail.com

Resumo *– O objetivo desse artigo é mostrar que a internet deve ser vista como um espaço heterogêneo e dinâmico, com um continuo crescimento de informação e de usuários. Como a internet cresce em grandes proporções é necessário que novas tecnologias devam ser adotadas para minimizar o trabalho e poupar tempo dos usuários. Neste artigo, faremos uma introdução sobre a comunicação entre agentes, as linguagens KQML e o Telescript e os diferentes tipos de sistemas de busca que utilizam agentes inteligentes, incluindo sistemas especializados em assuntos específicos, seja no comércio eletrônico ou na área médica. A integração de tecnologias é a chave para alcançar e sustentar um novo nível de performance e uma posição mais competitiva num mercado com rápidas mudanças*

Abstract *– The objective of this article is to show that the Internet must be seen as a heterogeneous and dynamic space, with continuous growth of information and users. As the Internet grows in great ratios is necessary that new technologies must be adopted to minimize the work and to save time of the users. In this article some basic information on the communication between agents will be shown to you.Tthe language KQML and Telescript and the different types of fetching systems that use intelligent agents, including systems specialized in one determined subject, either in the electronic commerce or the medical area will also be shown. This integration, between the technologies offered to the users, is the key to reach and to support a new level of performance and a more competitive position in a market with fast changes*

Palavras-chave *– Agentes Inteligentes, KQML, Telescript, Amalthaea; BargainFinder;, Homer; Letizia, MedScopio .*

1. INTRODUÇÃO

O contínuo crescimento da rede mundial de computadores permitiu o desenvolvimento de mecanismos computacionais onde suas ações são realizadas a favor do usuário, otimizando e facilitando o trabalho em uma base de comunicação entre eles. Assim a Internet transformou a visão clássica de base de dados como uma coleção de dados centralizados e homogêneos, em coleções de dados distribuídos e heterogêneos.

O funcionamento dos agentes em computação distribuída, através da programação remota, requer que pelo menos dois computadores combinem as instruções e os tipos de dados permitidos, e um agente contendo a rotina e o seu estado atual é enviado ao servidor. O procedimento contido no agente faz os pedidos necessários ao servidor, baseado no seu estado (condição), tornando assim a troca de mensagens bem menor. Não importa quantos pedidos sejam necessários ao servidor, é necessária apenas a mensagem que transporta o agente entre os computadores. Com isso ganhamos em performance e personalização já que a concorrência e a distribuição dos processos em diferentes máquinas podem aumentar a velocidade de processamento, haja vista o paralelismo na execução das tarefas.

Os agentes estão espalhados pela rede para beneficiar os usuários, trazendo informações já filtradas de seus detalhes. Além de proporcionar vantagens como a diminuição do tráfego de informações na rede esse mecanismo inteligente contribui para otimizar o trabalho do internauta. Assim, ele evita que uma transmissão longínqua seja interrompida, visto que a taxa de transferência é imprescindível. Uma aplicação natural para os agentes permite a distribuição de tarefas de monitoramento do sistema e a agilização no processo de tomada de decisão no caso de ausência do usuário.

Uma outra área a ser beneficiada com a aplicação das tecnologias de agentes é o comércio eletrônico. Dentro de um futuro não muito distante, o internauta poderá, por exemplo, agendar a compra de um determinado livro, especificando apenas os dados que o identifiquem e a faixa de preço que está disposto a pagar. Assim, quando a data se aproximar o agente se responsabilizará por trazer ao usuário todas as opções encontradas dentro das limitações preestabelecidas, para a compra do livro. O agente, não o cliente, coordena todo o trabalho, interagindo com o servidor nele mesmo, sem precisar da rede. Dessa forma o internauta evitaria estar se conectando somente para a busca dos sites e ainda contribuiria para que não haja aumento no trafego da rede, além de poupar seu tempo, tudo isso sem exigir o menor esforço.

2. ARQUITETURA E CARACTERÍSTICA DE UM AGENTE

Segundo (Brenner et al, 1998), a arquitetura de um agente deve permitir a captação e interpretação das percepções as quais são processadas de maneira inteligente, e que originam uma tomada de decisão e consequentemente a uma ação.

O agente, através dos sensores percebe as alterações e recebe as informações ao ambiente ao qual está inserido. A tomada de decisões interpreta, analisando e identificando as informações percebidas pelos sensores. Nesse momento ocorre, no módulo de fusão sensorial, a fusão da informação recebida de diversas fontes por forma de retirar-lhe possíveis inconsistências e obter uma decisão coerente do ambiente. O objetivo do processamento é interpretar os dados e formular planos, para a tomada de decisão. Posteriormente estes planos são passados ao módulo de seleção e finalmente a execução da ação.

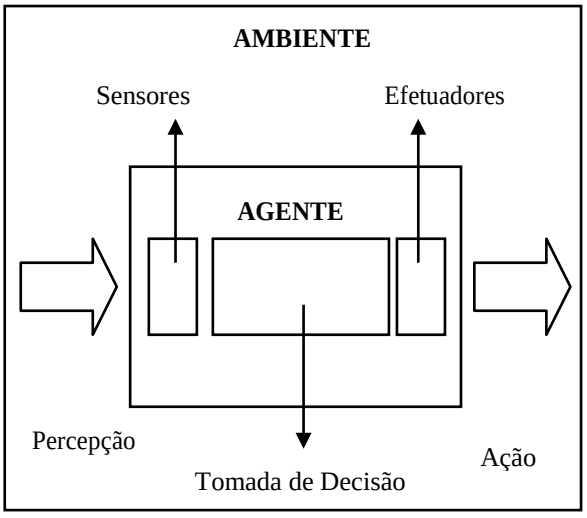

Figura 1 – Arquitetura básica de um agente.

As características como inteligência, mobilidade, autonomia, adaptabilidade e a capacidade de reação fazem com que os agentes se diferenciem de outros programas. De uma forma geral os agentes podem ser definidos como entidades que realizam um conjunto de operações em favor de um usuário, com certa autonomia, representando seus objetivos (inteligência). Possuem a tecnologia de transportar-se pela rede levando informações e solicitando outras (mobilidade). Capacidade de agir e determinar sobre o ambiente ao qual pertence (autonomia). Podem realizar tarefas específicas coordenadas, de forma a se completarem. Observa o comportamento do usuário, criando um conjunto de conceitos sobre os quais pode basear melhores suas decisões e otimizar seus serviços (adaptabilidade). Os agentes são independentes uns dos outros e podem em grupos reagirem a um conjunto de estímulos, comportando de maneira inteligente. Ainda podem ser adicionados ou removidos dentro de um contexto. E, finalmente podem raciocinar sobre ações já executadas, podendo planejar as suas ações futuras.

3. COMUNICAÇÃO MULTI -AGENTES

Para que ocorra uma comunicação entre agentes é necessário que esteja envolvido em uma infraestrutura capaz de proporcionar mecanismos de execução, comunicação, transporte e empacotamento.

A execução consiste no hardware e software necessários para executar programas agentes em tempo de execução, no ambiente.

O hardware pode ser qualquer tipo de computador, desde um mainframe até mesmo um computador pessoal, que nesse caso não é objeto de estudo.

Considerando a fundo a questão do software, muito se tem discutido sobre a aplicação de um agente em uma linguagem apropriada para desenvolvimento, porém, não se chegou a um consenso. Linguagens de código móvel como o Java e o C++ (orientadas a objeto) já podem ser encontradas aplicações de agentes inteligentes. Existe uma proposta para que a Telescript, da General Magic possa servir como padrão e enquanto isso a JKQML, da IBM e a KQML vem sendo usada com o mesmo propósito.

A comunicação entre agentes ocorre de duas maneiras: através de uma estrutura compartilhada chamada de quadro negro e através de troca de mensagens.

Na estrutura de quadro negro o agente pertence a um determinado nível e este interage com o nível superior, o quadro negro, que por sua vez além de fornecer uma estrutura de dado central, faz o escalonamento entre os agentes, participa do gerenciamento dos processos.. Dessa forma o agente pode ler ou escrever em um dos níveis.

O processo de troca de mensagens é o procedimento pelo qual um agente se comunica. As mensagens devem possuir um conteúdo, os dados, e o contexto que transforma esses dados em informações.

O movimento de um agente na rede caracteriza o transporte de um ambiente de execução para outro. A transmissão de dados suporta os padrões HTTP (HyperText Transfer Protocol ou protocolo de transferência de hipertexto), SMTP (Simple Message Transfer Protocol ou protocolo de transferência de mensagens simples), TCP/IP.

Atualmente, a maior parte dos ambientes que suportam agentes é resultado de pesquisas no gerenciamento de rede através de CORBA (Common Object Request Broker Architecture). Essa interação entre os

dispositivos é gerenciada através do protocolo SNMP (Simple Network Management Protocol ou protocolo de gerenciamento de redes). Existem momentos, entretanto onde o protocolo utilizado é proprietário do fabricante do dispositivo, nesse caso a comunicação é feita geralmente via porta serial.

Conceituamos o empacotamento como a capacidade que o agente possui de encapsular suas informações de estado, autenticação, objetivo, capacidade e plano de ação independente da sua estrutura interna. Ainda não segue a nenhum padrão.

4. A LINGUAGEM KQML

A KQML – Knowledge Query Manipulation Language Fazendo parte de um projeto de pesquisa chamado DARPA, desenvolvido pela Arpa Knowledge Sharing Effort, segundo (Labrou and Finin, 1997).

A KQML é uma linguagem e um protocolo para trocar informações e conhecimentos entre agentes. Caracteriza-se tanto pela formatação de mensagens como por um protocolo para manipular essas mensagens que suportam em tempo de execução, o compartilhamento de conhecimento e informação**.** Surgiu da necessidade de uma linguagem que seja aceita universalmente onde a comunicação não gere ambigüidade.

A KQML possui três camadas: comunicação, que trata o nível mais baixo como a indicação do receptor e emissor; mensagem, que especifica qual o protocolo deve ser usado e quais parâmetros são relativos ao conteúdo; conteúdo, as mensagens que o usuário vai transmitir.

Um modelo de uma expressão KQML representa além do protocolo pretendido, uma lista de atributos, como indicados a seguir:

:linguagem <indica qual linguagem está expresso o conteúdo da comunicação>
:conteúdo <conteúdo da mensagem>
:ontologia <indicam quais tipos de conceitos representados deverão ser usados para interpretar a mensagem>
:de <agente de origem>
:para <agente de destino>

Foi usada com sucesso para implementar uma variedade de sistemas de informações usando diferentes arquiteturas de software. A implementação da comunicação pode ser feita usando memória partilhada ou através da passagem de mensagem.

O uso da memória partilhada implica na existência de uma base de dados partilhados pelos agentes, dando suporte à comunicação, permitindo o envio de pedidos, respostas ou acessos a dados. A desvantagem é a possibilidade de atrasos no processo de resolução. Se ocorrer uma interrupção ou qualquer problema no suporte de comunicação o sistema pode falhar.
Na comunicação através de mensagens, os agentes se comunicam de modo transparente via sockets. Um socket está associado a uma porta, de modo que a camada TCP possa identificar a que aplicações se destinam os dados.

Comercialmente a KQML possui um ambiente que suporta a localização através de um software a um agente em um ambiente distribuído. A maioria dos ambientes adota o uso de uma ou mais estruturas como o CORBA ou o OLE2.

5. A LINGUAGEM TELESCRIPT

O Telescript é uma linguagem criada pela General Magic. Possui a facilidade de permitir aos desenvolvedores que construam aplicações comerciais seguras e fáceis, através da programação remota, onde o cliente e o servidor podem interagir sem o auxilio da rede.

A segurança das informações é garantida pela habilidade de controlar os acessos através de interações do processo, além de possuir chaves públicas e privadas para a autenticação. É assegurado que tanto os dispositivos físicos quanto à memória possam sequer ser acessados.

O agente implementado em Telescript realiza todas as transações solicitadas pelo cliente, podendo transportar por toda a rede, interagindo com o ambiente, através da troca de mensagens e voltar ao local de início, caracterizando a programação remota. Todas as ações executadas pelos agentes são limitadas.

6. O COMÉRCIO ELETRÔNICO

Que a Internet é a grande vedete das negociações não é novidade para ninguém. Uma área que está revolucionando os paradigmas das transações comerciais e da negociação é o comércio eletrônico, que vem provocando uma forte integração entre tecnologia a serviço do consumidor.

A grande oportunidade do mercado é a maneira como consumidores e fornecedores se encontram nesse mercado. Os clientes podem encontrar produtos que melhores se adeqüem ao seu perfil, através de uma busca pelos fornecedores em potenciais, via Internet. O agente se responsabiliza pela busca e o consumidor ficaria apenas aguardando a resposta.

O segredo para se alcançar sucesso é manter a agilidade das informações, integradas a redução de custo, economia de trabalho a asserção das tomadas de decisão. Com isso as empresas podem se posicionar frente a uma fatia significativa do mercado.

Existem estudos e pesquisas que abordam o uso de agentes no processo da negociação. Nesse caso um agente assume a posição do cliente e um outro a posição do fornecedor. O problema é como proceder já que nesse processo cada qual busca o melhor para o seu usuário e os conflitos de interesses sempre vão existir. O processo da negociação deve justamente encontrar uma solução que seja agradável para ambas às partes.

7. SISTEMAS DE BUSCA INTELIGENTES

Para a construção de agentes inteligentes é necessário que ele possua a habilidade de definir o conhecimento para que o agente possa decidir quando, com o que e como ajudar o usuário e ainda garantir que seja executada corretamente a tarefa a ele destinado.

Nos últimos anos foram desenvolvidos diversos sistemas que são baseados em agentes, que objetivam a busca de informação na Internet de interesse do usuário, nas mais diversas áreas. Eis alguns:

7.1 AMALTHAEA (Information Discovery and Filtering using a Multiagent Envolving Ecosystem)

O AMALTHAEA é um sistema cujo objetivo é ajudar o usuário a encontrar páginas que vão ao encontro de seu interesse.

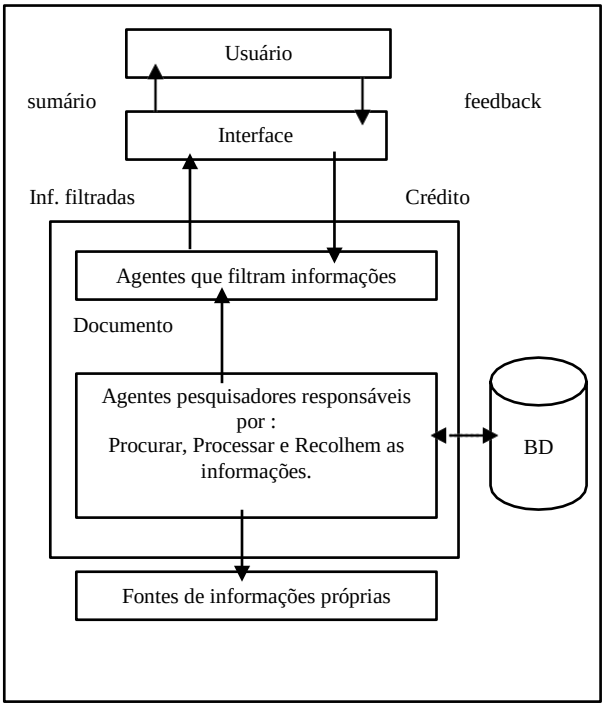

Figura 2 – Arquitetura do sistema Amalthaea.

O usuário através de um browser, faz uma solicitação de pesquisa usando uma interface que pode ser uma aplicação em Java, por exemplo. Esta interface estabelece uma comunicação entre aplicação e agente.

Existem dois agentes que trabalham juntos: os agentes de filtragem e os agentes pesquisadores.

Os agentes pesquisadores lançados pesquisam nas fontes de informações próprias e simultaneamente procuram encontrar na base de dados da Web os sites com informações relevantes, lançando diversos agentes na rede. As informações são comparadas e a medida em que houver uma modificação esta informação é recolhida e indexada à fonte própria. Os agentes pesquisadores retornam a pesquisa ao agente responsável por filtragem, que selecionará as páginas mais convenientes, removendo–lhes palavras irrelevantes, prefixos e sufixos. Esse procedimento de monitorização entre as URL's é feito em intervalo de tempos regulares.

Após este trabalho o agente de filtragem passa as informações filtradas à interface, que recebe os dados em formato hipertexto e converte em uma linguagem natural ao usuário. Nesse momento é exibida uma página contendo os vários índices das páginas encontradas.

Um mecanismo que verifica o sucesso da pesquisa é o feedback que o usuário retorna a interface, ou seja o usuário quando recebe os resultados efetua uma classificação da documentação apresentada, que serve de base de aprendizagem dos agentes. A interface por sua vez, dá um crédito ao agente. Assim podemos observar que os agentes cujo desempenho é considerado bom, se desenvolvem enquanto agentes que obtêm desempenho inferior são automaticamente eliminados. Ainda podemos verificar que os agentes rapidamente se adaptam aos interesses do usuário.

7.2 BargainFinder

O BargainFinder é um protótipo desenvolvido pela empresa Andersen Consulting que tem como principal objetivo demonstrar o uso de agentes inteligentes no comércio eletrônico.

Particularmente, o BargainFinder permite que os agentes atuem diretamente no processo de compra. Seu papel, além de fazer um levantamento de fornecedores distribuídos pela Internet, é fazer também desde a cotação dos preços até o retorno ao usuário, quando este autoriza compra do produto.

7.3 HOMER

O HOMER representa a simulação de um robô submarino que atua em um "mundo oceânico".

A arquitetura do agente o habilita a obter instruções do usuário utilizando um vocabulário restrito de 800 palavras, e estas instruções podem conter referências

temporais moderadamente sofisticadas, já que possui apenas um conhecimento parcial.

Outra característica do agente é a capacidade de elaborar seus planos, executá-los e modificá-los quando requerido durante a execução.

7.4 LETIZIA (An Agent That Assists Web Browsing)

Caracterizado por ser um agente que ajuda o usuário nas pesquisas via Web (Lieberman,1995). À medida que o usuário vai navegando em browsers, o LETIZIA vai tomando nota das páginas que este segue, por forma a poder antecipar os itens de interesse para o usuário. O agente segue uma estratégia automática de procura de páginas, segundo heurísticas denotadas pelo usuário. O comportamento do usuário que causa no agente a certeza de que uma página é realmente de seu interesse, é o de este adicionar às suas preferências no browser.

O modelo adotado pelo LETIZIA é uma combinação entre os passos dados pelo usuário e um agente de software inteligente. O LETIZIA e o usuário percorrem o mesmo espaço de procura, a Web, não havendo quaisquer objetivos pré-definidos. Existe uma confiança total do usuário relacionada ao agente que, por outro lado, pode explorar um conjunto de alternativas de modo mais rápido que o usuário. O papel do LETIZIA é apenas de observação e inferência dos comportamentos do usuário, para futuras pesquisas. Seu objetivo é recomendar dentro das ligações existentes nessa página, aquelas que serão de maior interesse do usuário.

A arquitetura do LETIZIA é orientada a objetos. Uma característica importante é que este dá uma explicação da razão porque escolheu determinada página: ou esta estava na preferência do usuário, ou tem uma palavra-chave de uma pesquisa efetuada antes.

O LETIZIA adota o método de procura em largura, pois o uso de procura em profundidade leva a que o agente se perca na Web, sem atingir seus objetivos. Além disso, as ligações adotadas pelo método em profundidade podem aumentar de forma exponencial. Para limitar este problema foi fixado o número de nós visitados por minuto.

7.5 MedScopio

O MedScopio é um sistema de busca na Internet especializado em pesquisas na área médica. Sua característica peculiar é de classificar todos os seus documentos de acordo com a Classificação Internacional de Doenças proposta pela Organização Mundial de Saúde.
O Medscopio permite com que as pesquisas apresentem precisão já que são utilizadas diversas fontes e bibliotecas médicas como base de dados. Ao contrário dos demais sistemas a busca é feita localmente devido a coleta contínua de informações anexas à própria base de dados, posteriormente em bases de dados pré-estabelecidas espalhadas pela rede. As respostas às solicitações são relacionadas ao usuário em inglês e português.

8. CONCLUSÃO

Neste artigo, procuramos ampliar o conhecimento em agentes inteligentes. Conceituamos as principais técnicas e sistemas desenvolvidos que utilizam esta tecnologia.

Foi observada também a inexistência de uma linguagem padrão. Nesse contexto denota-se que diversas linguagens como o Java e o C++ e mesmo os protocolos existentes hoje permitem facilmente a implementação da técnica de agentes.

Nesse momento é importante entender as características dos agentes inteligentes, já que muitos sistemas de pesquisa e alguns softwares se intitulam inteligentes quando na verdade não são.

Os sistemas apresentados que se utilizam agentes inteligentes destacam-se dos sistemas de busca como o Yahoo, Altavista e Miner, por:

- conseguir aprender com o usuário, através de feedback;
- aproximar da verdadeira necessidade do usuário;
- possuir, geralmente, uma base de dados própria;
- poderem ser acessados sempre que seja necessário;
- poderem usar quaisquer serviços disponíveis na WEB, como ftp ou telnet;
- estar em constante atualização.

O intuito da técnica de Agentes Inteligentes é facilitar e melhorar as pesquisas feitas pelo usuário, de uma maneira amigável e rápida, aproximando realmente as necessidades do mesmo. Assim sua aplicabilidade varia além da imaginação tanto de analistas quanto de desenvolvedores. Hoje, existem agentes controlando desde satélites e robôs até mesmo no ensino à distância e no comércio eletrônico.

AGRADECIMENTOS

Este trabalho recebeu uma especial atenção da professora Adicinéia que nos incentivou a desenvolver este artigo científico. A FAI, que nos proporcionou um excelente nível de aprendizado e que se mantém empenhada em despertar nossos talentos na comunidade acadêmica. Em especial aos nossos pais por acreditarem em nossos potenciais e que com tanto amor e carinho nos possibilitou o estudo. Aos nossos colegas que nos incentivam e nos auxiliam e a todos que acreditam no futuro da educação.

REFERÊNCIAS

BRENNER, W.e ZARNEKOW, R. e WITTING, H. "Intelligent Software Agents": Foundations and Applications Springer – Verlag, Berkin, Heidelberg, 1998.

LABROU, Y. e FININ, T. " A proposal for a new KQML Specification". Computer Science and Electrical Engineering Department , CSEE, 1997.

LIEBERMAN, H. "Letizia: An Agent that Assists Web Browsing" In Internacional Joint Conference of Artificial Intelligence, Montreal, 1995.

MADLENIC, D. Technical Report IJS-DP, 1996.

MOUKAS, A. "Amalthaea: Information Discovery and Filtering using a Multiagent Evolving Ecosystem". In Proceedings of the Conference on Pratical Application of Intelligence Agents and Multi-Agent Technology, London, 1996.

Introdução aos Agentes Inteligentes [disponível on-line: http://www.pcs.usp.br/~ito/ai01.htmal]

KQML [disponível on-line: http://www.inf.ufsc.br/iad/ users/f/fernando]

MedScopio [disponível on-line: http://www.medscopio.com.br]

Mini-Curso sobre Arquitetura de Agentes Inteligentes por Fabiana Maria Mendes C. Silva [disponível on-line: http://www.di.ufpe.br/~fmmcs/agentes/resumo. html]

Telescript [disponível on-line: http://www.genmagic. Com e http://www.inf.ufsc.br/iad/users/h/homero]

Uma estrutura de agentes para assessoria na Internet, Eliane Moreita Sá de Souza [disponível on-line: http:// Eps.ufsc.br/disserta97/souza]

Uma solução para a Transferência de dados via Internet [disponível on-line**:** http://www.inf.ufsc.br/marcel]