Desenvolvimento distribuído de software para sistemas pervasivos: um estudo de caso

Alexandre L'Erario[1,2], Tiago Faustino Prado[1], Mauro de Mesquita Spinola[1], Marcelo Schneck de Paula Pessôa[1], José Augusto Fabri[1,2]

[1]Engenharia de Produção - Escola Politécnica da Universidade de São Paulo
Av. Professor Almeida Prado, trav 2 n. 128 CEP 05508-900
Cidade Universitária São Paulo – Brasil Tel. 55 11 3091-5363

[2]Centro de Pesquisas em Informática da Fundação Educacional do Município de Assis.
Av. Getúlio Vargas, 1200 Vila Nova Santana
Assis – SP – Brasil CEP: 19807-634 - Fone/Fax: (18) 3302-1055

alerario@femanet.com.br, tiagopoli@uol.com.br, mauro.spinola@poli.usp.br, mpessoa@usp.br, fabri@femanet.com.br

Resumo. *Este artigo aborda o desenvolvimento distribuído de componentes para aplicações pervasivas. É ilustrado de que modo os componentes de software podem agregar funcionalidades a dispositivos móveis, tais como celulares, handhelds, e outros, além do impacto que estes tipos de agregações têm no processo de desenvolvimento distribuído. Este trabalho apresenta um estudo de caso onde um processo distribuído foi utilizado para construir uma aplicação.*

Palavras Chaves: *Componentes Pervasivos, Produção Distribuída, Processo Distribuído de Software.*

Abstract. *This article is related to distributed development of components for pervasive applications. It shows how the software components can add functionalities to mobile devices (such as cell phones, handhelds, and a few others), as well as the impact that these types of aggregations have on the distributed development process. This work presents a case study in which a distributed process was used to build up an application.*

Keywords: *Pervasive components, Distributed Production, Distributed Software Process.*

1. Introdução

A Tecnologia de Informação móvel é uma ferramenta capaz de permitir a distribuição ordenada da informação provendo auxílio a negócios, comodidade e agilidade a transações, que exigem versatilidade e velocidade. Atualmente, o termo "pervasivo" é empregado para abordar a façanha de distribuição, de mobilidade e de transparência da computação [Burkhardt 2002][Hansmann 2001]. Neste cenário, o usuário tem acesso à informação em qualquer dispositivo que não necessariamente seja um computador *desktop* convencional.

No entanto, o desenvolvimento de aplicações pervasivas é direcionado por um conjunto de requisitos não funcionais. Tais requisitos são resultantes das limitações e capacidades dos equipamentos que, ainda em sua maioria, apresentam características

heterogêneas. Agregar funcionalidades a estes dispositivos significa instalar componentes de software, que executam processamento local e/ou distribuído adequado a suas características físicas. Por esta razão, as informações que são acessadas em um computador convencional, em diversos casos, precisam ser adaptadas para os dispositivos portáteis. Este tratamento inclui, por exemplo, a redução de casas decimais e o ajuste da resolução de uma imagem entre outros.

O eLabSoft, um laboratório do Departamento de Engenharia de Produção da Escola Politécnica da USP, em parceria com a Fundação Educacional do Município de Assis, atualmente pesquisa o desenvolvimento e o processo de software. O projeto ManWApp, como um de seus projetos pioneiros, serviu de subsídio para aplicação de processos e tecnologias. Entre estes processos destaca-se o desenvolvimento distribuído e a implementação de um aplicativo capaz de conduzir a informação o mais próximo possível do usuário. O resultado foi o desenvolvimento de uma aplicação pervasiva chamada ManWApp.

Este artigo descreve um estudo de caso do desenvolvimento de um sistema pervasivo, elaborado de maneira distribuída, em que vários nós de rede (unidades de produção independente de software) agregaram funcionalidades ao sistema. Como limitação do projeto, o desenvolvimento foi voltado para *pda's* e celulares, desconsiderando aplicações desenvolvidas para eletrodomésticos e demais dispositivos que não se enquadram nestas categorias. É abordado o processo de produção distribuída e componentes de software para dispositivos móveis/portáteis.

1.1 Arquitetura do software utilizado no estudo de caso

A arquitetura multicamada foi adotada no projeto ManWApp com o objetivo de maximizar a capacidade de escalonamento, distribuição dos dados e serviços. Alguns requisitos, a princípio, foram concebidos somente para funcionarem em equipamentos *desktop* e, posteriormente, foram adaptados para dispositivos portáteis. Outros requisitos foram concebidos inicialmente para ambos os equipamentos. Alguns requisitos foram projetados apenas para dispositivos portáteis. Esta política serviu como subsídio para verificar os impactos no projeto e no processo, principalmente na dependência e delegação de tarefas (exemplo: para implementar seu subproduto um nó pode precisar de um subproduto de outro nó). A Figura 1 a seguir ilustra a arquitetura de camadas utilizada no ManWApp.

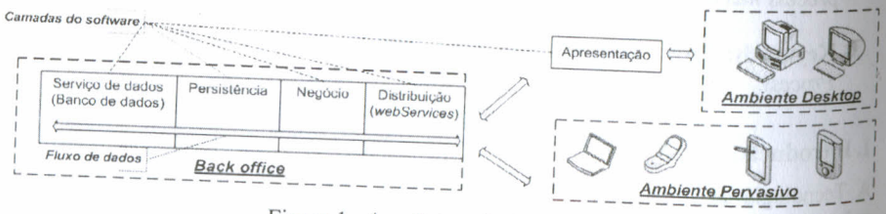

Figura 1 - Arquitetura do ManWApp

O *back office* é o núcleo do conjunto de funcionalidades (como regras de negócio) do ManWApp. O ambiente *desktop* envolve qualquer computador convencional, que tenha acesso à aplicação por meio de um navegador *web* e que, por isso, dispensa a instalação de módulos de software. O ambiente pervasivo é composto pelo conjunto de dispositivos móveis/portáteis que acessam a aplicação.

A camada de apresentação gera a interface visível do programa, que é utilizada pelo usuário para acessar a aplicação. As regras de negócio determinam de que maneira os dados serão utilizados. Na camada de persistência tem-se o acesso ao serviço de dados, no qual reside toda a informação necessária para o funcionamento da aplicação. A camada de distribuição (*web services*) provê a comunicação de dados e serviços para outras aplicações, através de protocolos e formato de dados em padrões abertos e independentes de plataforma de tecnologias. Com o uso de *web services,* é possível acessar dados em diversas aplicações, desenvolvidas com tecnologias diferentes e apresentar o resultado final, na forma de aplicação virtual, para um usuário específico [Newcomer 2002].

Nos dispositivos móveis/portáteis, foi necessário desenvolver e instalar um módulo de software. Este software contém componentes que prevêem as limitações do dispositivo tais como o tamanho da tela, a limitação de processamento, de memória, de comunicação, etc.

2. Sistemas pervasivos

A finalidade da computação pervasiva é aproximar o sistema do usuário de maneira que haja interações imediatas. Seu princípio básico de arquitetura funcional está alicerçado no mesmo princípio dos sistemas distribuídos. Este modelo computacional utiliza-se da descentralização para obter melhor performance e oferecer uma gama maior de ferramentas, que auxiliam as pessoas a resolverem problemas de maneira rápida e aparentemente simples, do ponto de vista do usuário [Hansmann 2001].

Hansmann [Hansmann 2001] e Burkhardt [Burkhardt 2002] definem que os sistemas pervasivos são constituídos da descentralização, diversificação, conectividade e simplicidade. A descentralização é uma característica na qual as informações e/ou o processamento podem ser distribuídos. Os usuários de um sistema pervasivo podem obter informações a partir de dispositivos diferentes, tais como *pagers*, telefone celular, *pda's* entre outros dispositivos. Todo dispositivo, por mais diferente que seja, precisa estar conectado logicamente, pois é necessário buscar informações remotas ou enviar dados armazenados. Além disso, os aplicativos precisam ser simples, pois o usuário precisa ter acesso fácil a informações. Estes itens são considerados não só no desenvolvimento de componentes para dispositivos móveis/portáteis, mas também no desenvolvimento do sistema *back office* pois, ambos trocam dados que são, de alguma maneira, disponibilizados para o usuário por meio do dispositivo portátil.

As limitações dos dispositivos são requisitos não funcionais, que precisam ser consideradas no projeto. Utilizando estes equipamentos, o usuário tem o sistema em qualquer lugar e hora dependendo é claro da disponibilidade de comunicação e do servidor. No entanto, quase que a totalidade dos dispositivos móveis (celulares e *pda's*) apresenta um visor reduzido, tem limitação com relação à entrada de dados (*script pen* ou pequenos botões) e não tem a mesma capacidade de processamento de um computador convencional (modelo *desktop*). Somente a parte realmente significativa do sistema de informação deve ser levada para o usuário por meio destes dispositivos.

2.1 Componentes dos sistemas pervasivos

Basicamente, há dois conceitos importantes para construção de aplicações pervasivas no contexto deste trabalho: componentes de software e sistemas distribuídos.

Um componente de software é um conjunto de código que encapsula uma ou mais funcionalidades. Segundo Yacoub [Yacoub 1995], os componentes têm três descrições básicas: informal, externa e interna. A descrição informal busca responder as questões relacionais entre o homem e o componente. A descrição externa define, formalmente, a comunicação entre o componente e a plataforma por este utilizado. Finalmente, as características internas refletem aspectos internos do componente. Já Brown [Brown 1999] além de reafirmar as características citadas acima assevera que para assegurar-se de que um sistema de software, baseado em componente possa funcionar correta e eficazmente, a descrição externa é o fator mais importante a ser observado, já que esta parte terá o papel de comunicar-se com a arquitetura do sistema.

Um software desenvolvido para um ambiente pervasivo pode ser construído a partir de pequenos componentes, chamados *facets* [Belamani 2003]. Várias adaptações precisam ser realizadas para que estes se comuniquem com uma aplicação *back office*. Tais adaptações incluem: adaptação de dados, adaptação do nível de rede, adaptação de energia (entende-se por energia o consumo da bateria de um dispositivo portátil/móvel), adaptação de migração e, finalmente, adaptação de funcionalidade [Belamani 2003].

Os sistemas pervasivos estão fortemente relacionados aos sistemas distribuídos, pois os equipamentos portáteis não disponibilizam um volume de memória e de processamento adequados, restando então fazer chamada remota, ou seja, invocar um procedimento que é executado em um computador e retornar somente os dados para o dispositivo móvel/portátil.

Existem diversas tecnologias para construção de aplicações distribuídas, tais como RMI [Grosso 2001], CORBA [Orfalli, 1998] entre outras.

3. Desenvolvimento distribuído de componentes pervasivos

3.1 Processo distribuído de software

O desenvolvimento distribuído ocorre quando vários nós concorrem para desenvolver um mesmo produto ou parte dele. Neste cenário, a complexidade do processo de desenvolvimento se amplia. A redução do tempo é a principal razão da divisão de tarefas na produção distribuída de software, porém o tempo de comunicação e de resposta entre os nós podem ser incomensuráveis. Há muitas variáveis neste cenário, tais como a cultura, a língua, a capacidade de cada nó entre outras. [Herbsleb 2003][Suzuki, 1999]. Além disso, desenvolver software em uma rede de produção requer gerenciamento mais eficaz de tarefas [Jäger 2001], ferramentas, capacidades e informação. [Martin 1996]

Embora a internet ofereça inúmeras vantagens em relação à comunicação, um projeto de software é muito complexo e grande, surgindo então uma necessidade de *framework* [Suzuki, 1999]. Este deve esclarecer como a informação é gerenciada na rede e como as tarefas são delegadas, por exemplo.

Para o desenvolvimento desta aplicação (ManWApp), foram escolhidas duas variáveis críticas de controle de processo que compõem um *framework*. Tais variáveis são: gestão de tarefas e gestão de repositório. Todas as variáveis críticas eram compartilhadas e todos os nós tinham a mesma hierarquia dentro da rede de produção.

Nesta rede a gestão de tarefas teve o objetivo de distribuir e rearranjar atividades de desenvolvimento por todos os nós. A delegação de tarefas foi baseada na capacidade de desenvolvimento individual e no tempo de entrega do subproduto de cada nó. Algumas dificuldades foram encontradas na dependência de atividades. Por exemplo, para testar uma funcionalidade de um dispositivo móvel era preciso que um módulo *web service* fosse desenvolvido na aplicação *back office* e outro módulo no dispositivo móvel.

Em muitos casos, a funcionalidade pervasiva era adicionada após a implementação do sistema no modelo clássico (*desktop*). Por isso, algumas funcionalidades foram adicionadas ao projeto *back office* para suportar comunicação remota e sincronização de dados. Em alguns casos foi necessário compatibilizar informações. Por exemplo, em alguns dispositivos móveis não foi possível tratar números com grande quantidade de casas decimais, enquanto que no *desktop* este tipo de tratamento é convencional.

A heterogeneidade dos nós da rede de produção pode causar impasses. O tempo para resolver os impasses tende a ser grande e atrasar demasiadamente a entrega do produto final.

A latência de resposta, ou seja, tempo médio que um nó leva para responder a um estímulo da rede é difícil de ser calculado. Uma solução encontrada foi a utilização de *checkpoints* : um nó gera um subproduto toda vez que armazena uma nova versão no repositório. A escolha de um repositório centralizado facilitou o gerenciamento de versões e evitou ambigüidades, provendo rastreabilidade do estado das tarefas de cada nó. O repositório gerenciou o acesso aos arquivos do projeto, identificando qual nó estava alterando qual arquivo.

Assim que um ciclo de atividades, de acordo com o projeto, fosse concluído, uma versão era entregue e disponibilizada para testes com usuários e demais membros do grupo. Após validação e fechamento de versão, uma versão do produto era atualizada em um servidor de aplicações. Este processo está ilustrado na Figura 2.

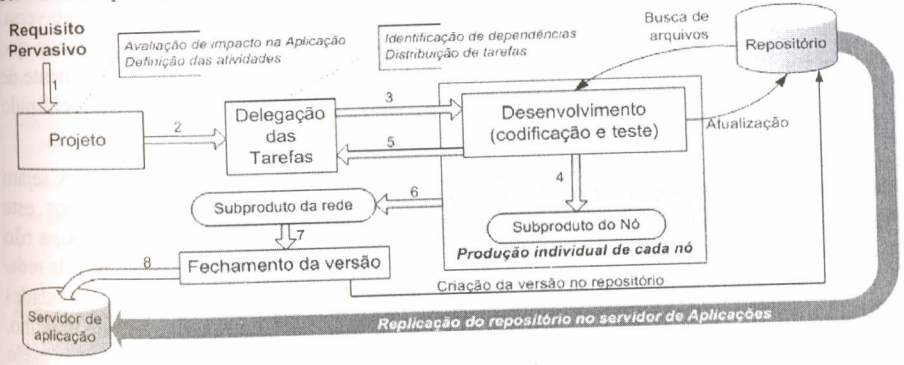

Figura 2 - Processo adotado para o estudo de caso

O fluxo 1 da Figura 2 indica a introdução de um requisito pervasivo, ou seja, a necessidade de aproximar a informação do usuário por meio de um dispositivo móvel/portátil no projeto. Após determinar as especificações pervasivas, fase de projeto ilustrada pela Figura 2, um conjunto de atividades, fluxo 2 da Figura 2, é então selecionado para distribuição.

A divisão de tarefas neste estudo de caso teve como norteador o diagrama de seqüência. Este pode representar a interação entre os módulos funcionais e camadas do sistema. Sendo assim, a delegação de tarefas foi condicionada a capacidade que cada nó tem em produzir determinada funcionalidade e o tempo estimado de entrega do subproduto do nó. Uma análise de impacto de acordo com a agregação da funcionalidade, discutido a seguir no item 3.2, era feita em um componente pervasivo. Cada nó, recebia como tarefa, a codificação de um objeto do diagrama de seqüência. Em alguns casos, o mesmo objeto era desenvolvido

por vários nós, no entanto, um grau mais refinado deste mesmo diagrama de seqüência era elaborado exclusivamente para tal objeto.

O fluxo 3 da Figura 2, identifica a tarefa sendo enviada para um nó. Esta pode ser cumprida ou não, dependendo da capacidade e do tempo de desenvolvimento. Caso seja desenvolvido, o nó gera um subproduto, fluxo 4 da Figura 2, e atualiza um repositório central de projeto. Caso o nó não cumpra o desenvolvimento, a tarefa é novamente delegada, fluxo 5 da Figura 2. Quando um nó termina por completo sua tarefa, conclui então um subproduto, fluxo 6 da Figura 2. Quando todos os nós concluírem suas tarefas, ocorre o fechamento da versão, fluxo 7 da Figura 2. O repositório é notificado para manter esta versão e a aplicação é então disponibilizada no servidor de aplicações para uso, fluxo 8 da Figura 2.

3.2 Agregação de funcionalidades a dispositivos móveis

Os componentes pervasivos desenvolvidos neste estudo de caso foram avaliados principalmente sob dois pontos de vista: interação com o equipamento e interação com o sistema *back office*. Além disso, foram identificadas quatro maneiras de como os componentes podem agregar funcionalidades. São elas: agregação por componente local, agregação por sincronização *on-line*, agregação por sincronização *off-line* e agregação por chamada remota.

Na agregação local o componente de software interage somente com o dispositivo móvel/portátil. Na agregação por sincronização as informações são trocadas entre aplicativo *back office* e aplicativo pervasivo, resultando em atualizações no banco de dados da aplicação *back office* e/ou da aplicação pervasiva. A sincronização *on-line* ocorre automaticamente, ou seja, os dados que estão no dispositivo são transferidos para outro sistema ou vice-versa, assim que uma comunicação estiver disponível. No modelo *off-line*, o usuário solicita que seja feita uma sincronização. A agregação por chamada remota implica que o componente de software instalado no dispositivo móvel invoque remotamente um procedimento localizado no *back office*.

O componente que utiliza agregação local deve prever que todos os dados sejam armazenados no dispositivo e todo o processamento seja efetuado também por este dispositivo. Neste caso, o componente não sofre interação com outro sistema externo, a não ser pelo usuário. Um mesmo componente pode ser desenvolvido por um ou mais nós da rede. Quando vários nós desenvolvem um mesmo componente, a dependência de atividades aumenta e consequentemente, o repositório é fundamental para evitar duplicações de trabalho. O projeto todo fica centrado no dispositivo móvel. Todos os requisitos não funcionais estão centrados nas limitações do equipamento.

Com exceção da agregação local, todos os demais tipos de agregação necessitam de que a aplicação *back office* e a aplicação instalada no dispositivo móvel/portátil estejam preparadas para troca de dados. O diferencial entre um tipo e outro de agregação é a disponibilidade do serviço de sincronização e a capacidade de processamento/armazenamento de informações do dispositivo móvel.

Na agregação por sincronização *off-line*, o dispositivo portátil deve comportar o processamento e não necessariamente deve armazenar todas as informações, mas somente as necessárias. Em muitos casos, as informações são resultados de consultas a várias tabelas, parte de uma tabela no banco de dados, e outras consultas. O componente interage indiretamente com outra aplicação. O serviço de sincronização não necessariamente precisa estar disponível o tempo todo. Os dados armazenados no dispositivo pervasivo devem ser suficientemente completos para o usuário trabalhar, porém precisam ser pequenos o bastante para que dispositivo móvel suporte-os.

Na sincronização *on-line*, o dispositivo deve comportar o processamento do componente e um volume de dados igual ao *off-line*, pois segue a mesma simetria do modelo. Porém, em alguns casos, o volume de dados que o dispositivo precisa armazenar pode ser menor, pois, toda vez que uma informação não estiver disponível no banco de dados do dispositivo móvel, uma conexão de rede é estabelecida e o banco de dados local é atualizado. Esta aplicação é decorrente da interação com o *back office*.

Na agregação por chamada remota, o *facet* invoca remotamente um procedimento. O dispositivo deve ter capacidade suficiente para processar o volume de dados gerados pelo procedimento remoto. Diretamente o componente interage com outra aplicação e, além disso, o sistema *back office* deve disponibilizar sempre o serviço de troca de dados.

A tabela 2 mostra os modelos de agregação, sua avaliação com relação à interação com o dispositivo e ao sistema *back office* e as tarefas a serem cumpridas. Um *facet* pode ser construído também utilizando mais de um modelo de agregação.

Tabela 1 - Modelos de agregação de funcionalidade em dispositivos móveis

Modelo de agregação	Interação com o dispositivo	Interação com o sistema *back office*	Tarefas
1- Local	Armazenamento de toda a informação Processamento de todas as funcionalidades	---	• Desenvolver um *facet*: acesso a recursos locais
2-Sincronização *off-line*	Armazenamento parcial de informações Processamento de funcionalidades locais	Disponibilidade esporádica de sincronização	• Desenvolver um *facet*: acesso ao banco de dados local • Desenvolver o componente de sincronização no sistema *back office*
3-Sincronização *on-line*	Armazenamento parcial de informações Processamento de funcionalidades locais	Média disponibilidade freqüente de sincronização	• Desenvolver um *facet*: acesso ao banco de dados local; solicitação automática de sincronização quando os dados estão desatualizados ou incompletos. • Desenvolver o componente de sincronização no sistema *back office*
4-Chamada remota	Nenhum armazenamento Processamento de informações resultantes da chamada remota	Alta disponibilidade freqüente de troca de dados	• Desenvolver um *facet*: capacidade de processar um volume razoável de informações • Desenvolver o componente capaz de disponibilizar um *web service*

4. Conclusões

Este artigo apresentou o estudo de caso do desenvolvimento distribuído de uma aplicação pervasiva chamada ManWApp. O desenvolvimento em rede de produção pode reduzir consideravelmente o tempo de entrega do produto final, desde que um *framework* seja detalhado antes do desenvolvimento de software. Neste estudo de caso os pesquisadores e desenvolvedores residem em locais distantes (acima de 400km), o que obrigou a adoção de procedimentos para desenvolver software em uma rede de produção. Um processo precisou ser desenvolvido (Figura 2) para solucionar o problema da descentralização das pessoas envolvidas.

Foram impostas algumas modificações no projeto, tais como novos requisitos pervasivos e novos requisitos funcionais. Neste caso, após reavaliação do projeto, foi percebido que a dependência de atividades era muito grande, causando desenvolvimento seqüencial em algumas etapas. Se a alteração de uma regra de negócio causasse algum impacto em um *facet*, o desenvolvimento seria praticamente seqüencial.

A grande vantagem neste processo está na sua simplicidade e agilidade. Em curtos espaços de tempo, os nós já tinham subprodutos que eram armazenados e disponibilizados em um repositório. No entanto, periodicamente era necessário uma reunião com os projetistas, pois o processo descrito neste trabalho aborda o desenvolvimento, não tratando as demais áreas do projeto.

Do ponto de vista do usuário final, a vantagem do desenvolvimento descentralizado é que o tempo de entrega de uma nova funcionalidade pode ser reduzido drasticamente. Uma rede de produção de software pode ser constituída de vários nós com vários especialistas diferentes. Além disso, a tarefa pode ser delegada de acordo com a capacidade, o tempo de entrega e o fuso horário onde se encontra o nó.

O processo apresentado neste artigo aborda somente o desenvolvimento distribuído. A próxima etapa é desenvolver um processo, compatível a este, porém capaz de atingir uma escala de análise e projeto.

Referencias

Belamani, et al. (2003) "Dynamic component composition for functionality adaptation in pervasive environments". IEEE Workshop of future trends of Distributed Computing Systems.

Brown, A. K. Wallnau. (1999) "The Current State of CBSE". IEEE Software, 15 (5): 37---46, Sep-Oct.

Burkhardt, Jochen , et al. (2002) "Pervasive computing – technology and architecture of mobile internet applications". Addison Wesley.

Grosso, Willian. (2001) "Java RMI. Oreilly & Assoc." 1ª edição.

Hansmann, Uwe et al. (2001) "Pervasive computing – the mobile world". Springer 2ª. Edição

Herbsleb James D. Morcrkus, Audris. (2003) "An empirical Study of Speed and Communication in Globally Distributed Software Development". IEEE transactions on Software Engineering, June n.6 vol29 pp. 481-494

Jäger, et al. (2001) "A delegation based model for distributed software process management." EWSPT Springer-Verlag Berlin.

Martin, Patric e Poweley, Wendy. (1996) "A management information repository form distribute application management". in ACM Conference of the Centre for Advanced Studies on Collaborative research

Newcomer, Eric. (2002) "Understanding Web Services," 1ª edição. David Chappel Series Editor.

Orfalli, Robert. (1998) "Client/Server Programming with Java and Corba". 2ª edição, Canadá.

Suzuki, Junichi e Uamamoto Yoshikazy. (1999) "Leveraging Distributed Software Development". IEEE computer, September. Vol32 pp. 59 - 65

Yacoub, Sherif. Ammar, Hany e Mili, Ali. (1995) "Characterizing a Software Component". http://www.sei.cmu.edu/cbs/icse99/papers/34/34.htm, June.