UMA PROPOSTA PARA ARQUITETURA DE DESENVOLVIMENTO DE SOFTWARE

Flávia Gonçalves Fernandes, Luciene Chagas de Oliveira
UNIUBE – Universidade de Uberaba
Uberlândia – MG, Brasil
flavia.fernandes92@gmail.com, luciene.oliveira@uniube.br

Resumo – No mundo globalizado e tecnológico em que se vive, é necessário buscar maneiras de melhorar o sistema informatizado digital. Os softwares, por exemplo, encontram-se em constante mudança, pois há a necessidade de corrigir erros existentes ou de adicionar novos recursos e funcionalidades. Essa necessidade evolutiva do sistema de software o torna "não confiável" e predisposto a defeitos, atraso na entrega e com custo acima do estimado. A complexidade dos sistemas de software exige que os profissionais da área raciocinem, projetem, codifiquem e se comuniquem por meio de componentes de software. A partir dessa necessidade, a engenharia de software mostrou-se mais atrativa e tornou-se uma área muito abordada recentemente. Também por meio dela, surgiu o ramo da arquitetura de software, que trabalha com o uso de camadas, objetivando facilitar a alocação da funcionalidade dos requisitos e oferecer suporte à flexibilidade e portabilidade, o que resulta em facilidade de reuso e manutenção. Isso ocorre devido às interfaces padrões bem definidas que encapsulam componentes. Além das modularizações, a arquitetura oferece suporte a um conjunto de atributos de qualidade, como desempenho, portabilidade e confiabilidade. Este trabalho tem como objetivo apresentar uma abordagem das arquiteturas de software existentes e propor uma nova arquitetura de software.

***Palavras-Chave* – arquitetura de software, desenvolvimento de software, engenharia de software.**

A PROPOSED ARCHITECTURE FOR SOFTWARE DEVELOPMENT

***Abstract* - In the globalized and technological world in which we live, it is necessary to seek ways to improve digital computerized system. The software, for example, are constantly changing, because there is a need to fix existing bugs or add new fetures and functionality. This evolutionary necessity of the software system makes it "unreliable" and prone to defects, delivery delays and cost over the estimate. The complexity of software systems requires professionals to reason, design, coding, and communicate by means of software components. From this need, software engineering was more attractive and has become an area much discussed recently. Also through her came the business of software architecture that works with the use of layers, aiming to facilitate the allocation of the functionality requirements and support the flexibility and portability, which results in ease of reuse and maintenance. This is due to well-defined standard interfaces that encapsulate components. Besides modularizações, the architecture supports a set of quality attibutes such as performance, portability and reliability. This paper aims to present an approach of existing software architectures and propose a new software architecture.**

***Keywords* - software architecture, software development, software engineering.**

I. INTRODUÇÃO

Este trabalho trata da arquitetura de software, uma disciplina em ascendência nas áreas acadêmicas e no mercado, na qual têm sido realizados diversos estudos com o intuito de melhorar a qualidade dos produtos que são desenvolvidos.

Segundo [7], a arquitetura de software alude a duas importantes características de um programa de computador: a estrutura hierárquica de componentes (módulos) procedimentais e a estrutura de dados. A arquitetura de software deriva de um processo de divisão em partições que relaciona elementos de uma solução de software a partes de um problema do mundo real implicitamente definidas durante a análise de requisitos. A evolução da estrutura de dados e do software inicia-se com uma definição do problema. A solução efetua-se quando cada parte do problema é resolvida por um ou mais elementos de software.

Além disso, a arquitetura de software possui grande importância atual na vida acadêmica e no mercado de trabalho, principalmente na área de tecnologia de informação, além de possibilitar maior desenvolvimento da economia do país.

O objetivo principal desse trabalho é propor uma nova técnica de arquitetura de software a partir das arquiteturas já existentes.

II. FUNDAMENTOS E ARQUITETURAS DE SOFTWARE

Neste trabalho, as metodologias de pesquisa empregadas foram a análise e a síntese, uma vez que apresenta um processo metódico de tratamento de um problema, que

implica a decomposição de um todo em suas partes, além de realizar a composição geral a partir das conclusões da análise, consistindo em reconstruir ou recompor os tópicos analisados numa sequência compacta e lógica.

Nessa perspectiva, os instrumentos e técnicas de coleta de dados foram as pesquisas feitas em livros, revistas e artigos na área de engenharia e arquitetura de software.

Dentre os conceitos básicos encontram-se: software, engenharia de software e arquitetura de software.

Software é o conjunto de instruções (programas de computador) que, quando executadas, produzem a função e o desempenho desejados.

Engenharia de software é o estabelecimento e uso de sólidos princípios de Engenharia para que se possa obter, economicamente, um software que seja confiável e que funcione eficientemente em máquinas reais.

Arquitetura de software é o estudo da organização global dos sistemas de software bem como do relacionamento entre seus subsistemas e componentes. Ela serve como uma estrutura que permite o entendimento de componentes de um sistema e seus inter-relacionamentos, especialmente daqueles atributos que são consistentes ao longo do tempo e de implementações, como visto em [7].

Arquitetura de software é uma subárea da disciplina de Engenharia de Software, cujo objetivo é estudar os componentes do sistema, suas propriedades externas, e seus relacionamentos com outros softwares.

Uma boa arquitetura pode possibilitar que um sistema satisfaça às exigências principais de um projeto, tais como: desempenho, confiabilidade, portabilidade, manutenibilidade, interoperabilidade .

Com o amadurecimento da engenharia, percebeu-se que uma boa arquitetura é um fator decisivo de sucesso para o projeto e o desenvolvimento do sistema, visto que ela desempenha o papel de uma ponte entre requisitos e código.

A arquitetura de software tem por objetivo fazer o melhor uso dos recursos técnicos de um projeto para garantir a maior eficiência possível aos objetivos do projeto, produto e mesmo à visão e missão de uma empresa.

Quando analisamos, é importante considerar não somente os requisitos funcionais do sistema, mas também atributos de qualidade, porque a arquitetura de software pode ser alterada se houver mudança dos requisitos não funcionais.

Conforme [3], questões arquiteturais englobam organização e estrutura geral de controle, protocolos de comunicação, sincronização, alocação de funcionalidades a componentes e seleção de alternativas de projeto. Trabalha com o uso de camadas, objetivando facilitar a alocação da funcionalidade dos componentes e oferecer suporte à flexibilidade e portabilidade, o que resulta em facilidade de reuso e manutenção. Isso ocorre devido às interfaces padrões bem definidas que encapsulam componentes.

Além das modularizações, a arquitetura pode servir de base à estimação de custos de gerência do projeto, e para análise da consistência e dependência, oferecendo suporte a um conjunto de atributos de qualidade (como desempenho, portabilidade e confiabilidade).

Apenas recentemente houve o foco na arquitetura de software e forte ênfase na disciplina de engenharia de software devido à economia e reuso.

Habilidades desejadas do Arquiteto de Software:

- Conhecimento do domínio e tecnologias relevantes;
- Conhecimento de questões técnicas para desenvolvimento de sistemas;
- Conhecimento de técnicas de levantamento de requisitos, e de métodos de modelagem e desenvolvimento de sistemas;
- Conhecimento das estratégias de negócios da empresa;
- Conhecimento de processos e produtos de empresas concorrentes.

Tarefas atribuídas: modelagem; análise de compromisso e viabilidade; prototipagem, simulação e experimentação; análise de tendências tecnológicas; “evangelizador” de novos arquitetos.

Logo, a arquitetura é o “universo” em que o ciclo de vida do software vive, ou seja, o código é o artefato primário; também há a divisão entre as camadas arquiteturais, a divisão entre os componentes de negócio, regras de integração entre as camadas, componentes e entre os sistemas (novos e legados).

Como visto em [6], podemos identificar, nesta visão, atividades tais como:

- garantir o alinhamento técnico do projeto às diretrizes e estratégias tecnológicas de uma área ou organização, bem como à sua cultura organizacional;
- atender as restrições gerenciais de um projeto tais como custo, prazo e competências técnicas da equipe;
- identificar e detalhar requisitos significativos para a arquitetura de software;
- antecipar e mitigar os riscos técnicos de um projeto;
- realizar a análise e desenho técnico da arquitetura através de técnicas de modelagem arquitetural;
- construir provas de conceito e gerar código executável para os principais pontos de risco do projeto;
- orientar e acompanhar o time técnico para garantir a aderência do código construído às diretrizes arquiteturais do projeto;
- validar a estabilidade e objetivos da arquitetura do produto;
- coletar lições aprendidas e promover um novo ciclo de melhorias.

O ciclo de vida do produto é mais abrangente que o do projeto. Um produto pode ser criado e evoluído por vários projetos.

As atividades da gestão da arquitetura de software são um subconjunto das atividades executadas em um projeto de software.

De acordo com [3], os grupos de atividades do ciclo de vida da gestão da arquitetura que evidenciamos como IDEA são: Iniciação-Definição-Edificação-Avaliação.

Iniciação: A formulação da estratégia arquitetural é a uma atividade do grupo de iniciação do IDEA. Em suma, o arquiteto deve realizar uma análise estratégica e desenvolver

a estratégia da arquitetura de software. O time de arquitetura também pode endereçar oportunidades para a empresa com o desenvolvimento de uma arquitetura de software robusta para o produto. A estratégia arquitetural também endereça aspectos como o estilo arquitetural e a metáfora sistêmica. O estilo arquitetural, como na arquitetura da construção civil, classifica um produto quanto ao seu estilo primário.

Definição: A definição é o grupo de atividades do IDEA que lida com análise e desenho arquitetural dos requisitos sob análise. Grande parte do trabalho que arquitetos usualmente praticam, principalmente a modelagem arquitetural, está situada neste grupo de atividades. No artigo corrente, abordaremos a definição arquitetural com um nível muito superficial de detalhes. A definição no IDEA, entretanto, tende a ser mais ampla e lida com:

- O planejamento das atividades de definição da arquitetura para a iteração ou fase atual do projeto;
- O desenvolvimento dos requisitos a partir das diretrizes arquiteturais;
- A modelagem arquitetural através do uso de múltiplas visualizações;
- O detalhamento das táticas arquiteturais;
- A exploração de alternativas de análise e desenho.

Edificação: A edificação realiza a arquitetura, isto é, fornece evidências concretas através de código executável que a arquitetura acomoda os requisitos do projeto. A edificação é realizada através do esforço conjunto do time de arquitetura do projeto (arquiteto e desenvolvedores especialistas). O resultado da edificação é uma arquitetural executável, que poderia ser gerada em pequenos incrementos (builds) nas fases iniciais de um projeto.

Avaliação: A avaliação deve ser ampla e envolver o time técnico, time gerencial, time de analistas e demais interessados no projeto. As ações de arquitetura devem ser avaliadas e os erros e lições aprendidas devem ser discutidas e comunicadas para toda a equipe.

Portanto, a disciplina de arquitetura de software tende a ser mais eficaz se aplicada em um contexto mais amplo e responsável – a gestão da arquitetura de software. IDEA, o ciclo de vida da gestão da arquitetura de software, encadeia as atividades de arquitetura de software de uma forma compreensível e organizada.

A. Arquitetura Orientada a Objetos

Objetos são entidades em um sistema de software que representam instâncias de entidades do mundo real e de algum sistema.

A arquitetura orientada a objetos combina dados com funções numa única entidade (objeto), facilitando a decomposição do problema, manutenção e reuso. É comum a sua utilização em sistemas de informação como de consultas e emprésti-mos online de bibliotecas de instituições de ensino que dispõem de componentes de cadastro de usuários e componentes de autenticação de usuários, de acordo com [3].

Essencialmente, a arquitetura orientada a objetos estabelece as regras, diretrizes e convenções definindo como as aplicações podem se comunicar e interoperar. Dessa forma, o foco da arquitetura não é em como a implementação é realizada, mas sim na infra-estrutura e na interface entre os componentes da arquitetura. É uma forma mais natural de se analisar o mundo. Ela nos permite construir sistemas melhores e, além disso, de maneira mais fácil.

O estilo arquitetural de objetos tem como base o uso de um tipo de dados abstrato – o objeto, o qual possui várias propriedade importantes, como a possibilidade de fazer o mapeamento entre as entidades reais e os objetos que atuam no controle de um sistema, resultando numa melhor compreensão dos sistemas.

Além disso, os objetos fazem uso de um mecanismo de troca de mensagens para se comunicar em vez de utilizar variáveis compartilhadas, além de poderem ser reutilizados devido a sua independência, estar distribuídos e executar seqüencialmente ou em paralelo, a depender das decisões tomadas no início do projeto, possuindo como base a ocultação da informação

As suas vantagens são: facilidade, reutilização, manutenção, modularidade, encapsulamento, agilidade, portabilidade, exatidão e escalabilidade, e as desvantagens são consumo de memória, apropriação, fragilidade e linearidade de desenvolvimento.

B. Arquitetura Orientada a Serviços

Serviço é uma função de um sistema computacional que é disponibilizado para outro sistema. Além disso, é independente e possui interface bem definida.

A arquitetura orientada a serviços tenta aproximar o entendimento entre as áreas de tecnologia de informação (TI) e de negócios a uma visão mais semelhante de como um sistema de informação deve ser abordado.

Além disso, ela propõe uma solução mais próxima dos processos de negócio, uma vez que se tem um fluxo de execução montado a partir de peças ou unidades bem definidas que executam determinada funcionalidade. Sendo assim, uma unidade demanda, como pré-requisito, informações na entrada e, necessariamente, criará algo modificado como resultado do sucesso de sua execução, conforme [4].

SOA propõe que se faça a modelagem de um processo, não mais como um monobloco, mas sim por um conjunto ordenado de unidades bem definidas, cada qual com sua funcionalidade convenientemente descrita, de modo que, ao final da execução de todas as unidades, teremos o resultado esperado. Esse conjunto ordenado de unidades recebe o nome de workflow (fluxo de trabalho) e cada unidade recebe o nome de serviço, segundo [8].

As suas vantagens são: reutilização de software, aumento de produtividade, maior agilidade, interoperabilidade e escalabilidade, e as desvantagens são: performance, segurança, robustez, disponibilidade, testabilidade e usabilidade.

C. Arquitetura Orientada a Modelos

Modelo é uma abstração de algo que existe no mundo real e descreve a representação de sistemas de software de um domínio.

Nessa arquitetura, os modelos são parte constituinte do software, assim como o código. A proposta é reduzir a distância semântica entre o domínio do problema e o domínio da implementação/solução, através de modelos de alto nível que protegem os desenvolvedores de software das complexidades da plataforma de implementação, de acordo com [9].

A MDA propõe separar a lógica de negócio, plataforma e tecnologia, de tal forma que as modificações na plataforma não afetem as aplicações existentes e a lógica de negócio evolua independente da tecnologia. Com isso, os benefícios são evidentes, tais como a redução de custo do ciclo de vida do projeto, a redução do tempo de desenvolvimento para novas aplicações, o aumento da qualidade da aplicação e o aumento do retorno no investimento em tecnologias, conforme [10].

Para obter mais benefícios, a MDA faz uso de modelos, os quais podem ser independentes ou específicos de uma plataforma. É responsável por diversos padrões da computação para especificação de sistemas e interoperabilidade. O seu ciclo de desenvolvimento não é muito diferente do tradicional.

Através da MDA, apesar de ser uma abordagem relativamente recente, é possível ter um aumento significativo na produtividade, garantir a documentação do sistema em vários níveis de abstração e a interoperabilidade entre os modelos, com a utilização de ferramentas apropriadas, as quais vêm surgindo com bastante força no mercado. Na MDA, os modelos são elementos ativos e participantes de todo ciclo de desenvolvimento e não apenas um meio de comunicação entre os participantes do projeto.
No processo de desenvolvimento de software, os modelos podem ser usados na geração de programas, scripts de banco, documentação de usuário, configurações e quaisquer outros elementos que fazem parte do processo de desenvolvimento.

As suas vantagens são: praticidade, exatidão, portabilidade, agilidade, intuitividade, documentação, engenharia reversa, reutilização, manutenção, modularidade e encapsulamento, e as desvantagens são: falta de qualidade em codificação, inconsistência entre a modelagem e o código-fonte final, consumo de memória, código-fonte automatizado nem sempre é o ideal e os testes são realizados diretamente no produto final.

D. Arquitetura Orientada a Aspectos

Aspectos são visões das funções ou dos atributos de qualidade. Tem o mesmo comportamento de uma classe e é encarregado de encapsular as funcionalidades transversais dispersas no sistema.

Conforme [10], o desenvolvimento de software orientado a aspectos é um novo paradigma com abstrações e mecanismos de composição que possibilita tratar os requisitos do software de cada aspecto separadamente, classificá-los por categorias e realizar uma análise geral.

Dessa maneira, essa arquitetura é caracterizada pelo não-entrelaçamento do código e divide o sistema em dois níveis: componentes (ou requisitos) funcionais e aspectos (requisitos não-funcionais ou entrelaçados)

O requisito não-funcional é ortogonal à decomposição funcional e é desmembrado em fatores.

A arquitetura orientada a aspectos mostra a estrutura estática do aspecto e especifica pontos de conexão bem definidos, que são a base da ligação entre componentes. Como essa abordagem utiliza o conceito de porta, a expansibilidade da conexão se torna possível graças a esses "pontos de extensão".

As suas vantagens são: reusabilidade, manutenibilidade, legibilidade, modularidade e flexibilidade, e as desvantagens são: falta de padronização em UML, inexistência de ferramentas específicas para suporte ao projeto arquitetural e falta de padronização da arquitetura.

E. Arquitetura Orientada a Componentes

Componente é um elemento independente, que possui interface bem definida, não compartilha estado e comunica-se por troca de mensagens contendo dados.

A arquitetura orientada por componentes busca reduzir a complexidade de desenvolvimento resultante da heterogeneidade dos códigos de acesso e da dificuldade em capturar e gerir informações que descrevem o contexto de utilização. A arquitetura fornece mecanismos de adaptação, tanto no momento da instalação quanto durante a execução, de funcionalidades, conteúdo e interfaces de sistemas Web de administração de ensino em função do contexto de utilização do sistema, como visto em [7].

Apesar da intensa pesquisa relacionada aos sistemas sensíveis ao contexto realizada nos últimos anos, o desenvolvimento e a implantação de tais sistemas ainda corresponde a uma operação complexa, que é resultante principalmente dos três pontos críticos a seguir:

I) a heterogeneidade dos DM de acesso;

II) a dificuldade em capturar/gerir as informações que descrevem o contexto de utilização;

III) a complexidade relacionada à implementação dos mecanismos de adaptação e de filtragem de conteúdo baseados no contexto.

As suas vantagens são: facilidade, reuso, agilidade, confiabilidade e qualidade, e as desvantagens são: falta de robustez e adaptação.

F. Arquitetura Orientada a Características

Características são entidades de primeira classe do projeto, que encapsulam representações de programas estruturadas hierarquicamente (como código, arquivos de configuração, etc.) e que descrevem em alto nível de abstração os requisitos de um sistema, ou seja, é um incremento na funcionalidade do programa.

A ideia central proposta pela arquitetura orientada a características é permitir a geração de aplicações através da composição de características. Segundo [5], ela propõe uma abordagem de refinamento incremental onde características representadas por unidades de implementação modular são gradativamente aplicadas a uma base inicial de código-fonte.

Dessa forma, como apresentado em [2], as características são criadas/adaptadas em função de um contexto considerada a presença de uma característica base (um ponto de partida para os produtos de uma linha) e seu comportamento e refinado por sua combinação com outras características. Esta

mudança de comportamento altera a semântica do produto final. Em essência, a arquitetura orientada a características defi-ne uma linha de produtos.

Na prática, muitas características são independentes de outras, e não modificam os seus estados quando agrupadas. Estas podem ser combinadas sem mudar as demais características.

Entretanto, segundo a abordagem de [1], uma característica pode refinar o comportamento de outras existentes ou necessitar a exclusão/substituição de uma outra por questões de incompatibilidade. O resultado de análises como esta e um conjunto de critérios (restrições especificas do domínio) para composição de características, que permite um raciocínio modular e o refinamento sobre as composições.

Logo, este paradigma é realizado para linhas de produtos de software onde programas são sintetizados pela composição de características. Uma linha de produtos de software define uma família de produtos similares onde as características correspondem as funcionalidades que os programas proveem.

As suas vantagens são: encapsulamento, gerência, facilidade de modificação, suporte e modularização, e as desvantagens são: falta de capacidade de endereçar o projeto e de implementar características transversais.

III. PROPOSTA DE ARQUITETURA DE SOFTWARE

A partir das arquiteturas apresentadas, propõe-se uma nova técnica de arquitetura de software que une os tipos já existentes e mencionados anteriormente.

A nova arquitetura de software propõe a partir dos modelos existentes, extraem-se os objetos, os aspectos e as características do software, e projetam-se os componentes e serviços do sistema, conforme ilustrado na Figura 1.

Dessa forma, sabe-se que os modelos possuem maior grau de abstração que os objetos, características e aspectos.

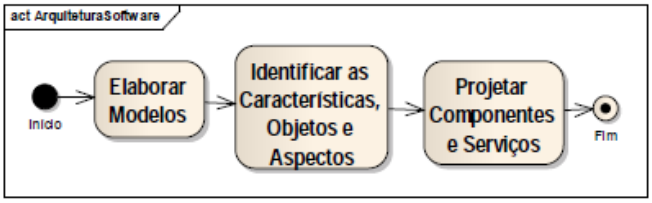

Figura 1. Atividades da Arquitetura de Software Proposta..

Os modelos são elaborados na fase de concepção do sistema. Os requisitos funcionais são utilizados para extração de objetos e os requisitos não funcionais são utilizados para extração de aspectos que compõem o sistema. As características serão utilizadas para compor programas de linhas de produtos de software. Assim, por meio dessa hierarquia, modelos, objetos, aspectos, características, serviços e componentes são responsáveis por determinada parte do programa. Os programas são construídos através da composição de características, objetos e aspectos, formando os componentes e serviços de software.

Com a enfatização destas arquiteturas em uma única arquitetura, vantagens como, por exemplo, reutilização, agilidade, facilidade de manutenção e engenharia reversa poderão ser alcançadas e as desvantagens serão amenizadas, já que os benefícios de uma modalidade poderão suprir os defeitos da outra.

IV. CONCLUSÕES

A construção de software através da combinação de várias arquiteturas possibilita a utilização delas conforme a necessidade, podendo-se dar maior enfoque a uma ou mais arquitetura, dependendo do sistema implantado.

Portanto, a arquitetura proposta poderá ser mais completa e eficiente do que cada uma isoladamente, o que ocasiona aumento dos lucros para as empresas, produtividade e satisfação dos clientes, visto que a construção de programas através da combinação de várias arquiteturas possibilita a utilização delas conforme a necessidade, podendo-se dar maior enfoque a uma(s) determinada(s) arquitetura(s), dependendo do sistema implantado.

Para a validação deste trabalho será realizado um estudo de caso envolvendo a implementação de um sistema usando a abordagem proposta.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] BATORY, D.: Feature-Oriented Pro-gramming and the AHEAD Tool Suite, 2004.

[2] DIJKSTRA, E.W. A Discipline of Programming. Prentice-Hall,1976.

[3] FILHO, Antônio Mendes da Silva. Ar-quitetura de software: desenvolvimento orientado para arquitetura. Engenharia de Software Magazine, Brasil, 1 ed., p. 38-45, 2007.

[4] JÚNIOR, Jorge Dias. Arquitetura ori-entada a serviços. Engenharia de Software Magazine, Brasil, 22 ed., p. 30-35, 2008.

[5] MEDEIROS, Ana Luisa et al. Requisi-tos e arquitetura de software orientada a aspectos e a características. Simpósio Brasileiro de Engenharia de Software, 21, 2007, São Paulo.

[6] MENDES, Marco Aurelio S.. VIGGI-ANO, Eros. Ciclo de vida da gestão em arquitetura de software. Engenharia de Software Magazine. Ano 2, 14ª edição, 2008, p. 40-48.

[7] PRESSMAN, R. S. Engenharia de Software. McGraw-Hill, São Paulo, 2006, 1027 p.

[8] RIBEIRO, Anderson Dias. GIGLIO, Giuliano Prado de Morais. Arquitetura orientada a serviços. Engenharia de Soft-ware Magazine. Ano 2, 19ª edição, 2008, p. 24-32.

[9] SEVERIANO, Rodrigo Henrique; BRAGA, Regina Maria Maciel; ARAÚJO, Marco Antônio Pereira. Desenvolvimento orientado a componentes. Engenharia de Software Magazine, Brasil, 13 ed., p. 42-48, 2008.

[10] SOUZA, Italo Fernando Comini; ARAÚJO, Marco Antônio Pereira. MDA – Arquitetura Orientada por Modelos. Engenharia de Software Magazine, Brasil, 9 ed., p. 28-34, 2007.