SEGURANÇA NA ARQUITETURA DE SISTEMAS INFORMATIZADOS

SECURITY IN COMPUTER SYSTEMS ARCHITECTURE

SEGURIDAD EN INFORMÁTICA DE SISTEMAS DE ARQUITECTURA

SAMÁRIS RAMIRO PEREIRA[1]

PAULO BANDIERA PAIVA[2]

LEANDRO ZEIDAN TOQUETTI[3]

DELCÍNIO RICCI[4]

SUELI APARECIDA LODDI[5]

Recebido em setembro de 2010. Aceito em novembro de 2010.

[1] Graduada em Matemática com ênfase em Processamento de Dados e em Tecnologia com ênfase em Técnicas Digitais. Mestre em Informática. Doutoranda em Informática na Saúde pela UNIFESP. Professora da Faculdade de Tecnologia de São Bernardo do Campo.

[2] Graduado em Tecnologia da Computação. Mestre em Ciências Biológicas. Doutor em Ciências Biológicas pela UNIFESP. Professor da UNIFESP.

[3] Graduado em Materiais, Processos e Componentes Eletrônicos. Mestre em Engenharia Elétrica. Doutor em Engenharia Elétrica pela USP. Professor e, atualmente, diretor da Faculdade de Tecnologia de São Bernardo do Campo.

[4] Graduado em Matemática e em Pedagogia. Mestre em Educação Matemática. Doutor em Educação pela PUC-SP. Professor e, atualmente, coordenador do curso de tecnologia em Informática para Gestão de Negócios da Faculdade de Tecnologia de São Bernardo do Campo.

[5] Graduada em Matemática com ênfase em Processamento de Dados. Mestre em Administração pela Universidade Municipal de São Caetano do Sul. Professora da Faculdade de Tecnologia de São Bernardo do Campo.

SEGURANÇA NA ARQUITETURA DE SISTEMAS INFORMATIZADOS

RESUMO

É um princípio básico da Segurança da Informação o fato de não existir sistema 100% seguro, devendo-se optar por um nível de segurança conforme a necessidade. Porém, é inviável definir um nível de segurança em um sistema informatizado que apresente falhas decorrentes de um processo de produção precário ou que não atenda aos requisitos do usuário. A qualidade no funcionamento de um sistema não é um objetivo da Segurança da Informação, é um pré-requisito. O objetivo deste artigo é ressaltar a importância para a Segurança da Informação, da qualidade no desenvolvimento de sistemas informatizados, meta a ser atendida sem detrimento do desenvolvimento ágil. Para que se possam obter informações seguras, que garantam autenticidade e confidencialidade assegurada por lei, há a necessidade de se aliar às tecnologias de Segurança da Informação como criptologia e certificação digital, o cumprimento das metas da Engenharia de Software referentes à qualidade e agilidade. Para tal, conta-se com várias tecnologias e ferramentas desenvolvidas e aprimoradas ao longo dos anos.

PALAVRAS-CHAVE: Arquitetura de sistemas em multicamadas. Engenharia de Software. Sistemas informatizados. Segurança da Informação. Transação em banco de dados.

SECURITY IN COMPUTER SYSTEMS ARCHITECTURE

ABSTRACT

It is a basic tenet of Information Security, the fact that there is no 100% secure system, one should choose a level of security as needed. However, it is impossible to define a level of security in a computerized system to produce failures due to a poor production process or not meeting user requirements. The quality in the functioning of a system is not an objective of information security is a prerequisite. The aim of this paper is to highlight the importance to information security, quality in the development of computerized systems, a goal to be met without the expense of agile development. In order to obtain reliable information, to ensure authenticity and confidentiality guaranteed by law, there is a need to combine the technologies of information security and cryptology and digital certification, the goals of Software Engineering concerning the quality and agility. To this end, it counts with several technologies and tools developed and refined over the years.

KEYSWORDS: Computerized Systems. Information Security. Multi-tier Architecture. Software Engineering. Transaction Database.

SEGURIDAD EN INFORMÁTICA DE SISTEMAS DE ARQUITECTURA

RESUMEN

Es un principio básico de la seguridad de la información, el hecho de que ningún sistema es 100% seguro. Uno debe elegir un nivel de seguridad según sea necesario. Sin embargo, es imposible definir un nivel de seguridad en un sistema computarizado para producir fallos debidos a un proceso de producción deficiente o que no cumplan los requisitos del usuario. La calidad en el funcionamiento de un sistema no es un objetivo de seguridad de la información, es un requisito previo. El objetivo de este artículo es poner de relieve la importancia de la seguridad de la información, calidad en el desarrollo de la informática, un objetivo que debe cumplir sin el gasto de desarrollo ágil. Con el fin de obtener información fiable, para garantizar la autenticidad y confidencialidad garantizada por la ley, existe la necesidad de combinar las tecnologías de seguridad de la información como la criptografía y certificación digital, a los objetivos de la Ingeniería de Software sobre la calidad y agilidad. Para ello, cuenta con varias tecnologías y herramientas desarrolladas y perfeccionadas a lo largo de los años.

PALABRAS-CLAVE: Arquitectura de Sistemas de Múltiples Capas. Ingeniería de Software. Los sistemas computarizados. Seguridad de la Información. Base de datos de transacciones.

1 INTRODUÇÃO

Com o constante avanço da TI (Tecnologia da Informação) e sua disponibilidade, as empresas passaram a depender cada vez mais da informação e desfrutar de sistemas computacionais que continuamente se sofisticam, para suportar suas atividades. Nos tempos atuais, mais que nunca, informação significa poder, e seu uso apropriado pode estabelecer o diferencial competitivo e a projeção de um cenário com vistas a um melhor atendimento a clientes, com a otimização de toda a cadeia de serviços, produtos e pesquisas.

Sistemas informatizados são essenciais, sendo ativos de valor. Porém, esses sistemas necessitam de métodos e técnicas para garantir a integridade das informações mantidas e fornecidas por eles, caso contrário, as consequências podem ser graves, como processos judiciais ou a indução ao erro médico.

Há uma forte necessidade na Engenharia de Software de um desenvolvimento ágil dos sistemas. As diferentes experiências vivenciadas pelos autores mostram que os usuários de sistemas informatizados consideram que um sistema funcionando valha mais do que sua documentação precisa e extensa. Da mesma forma, consideram que a colaboração do usuário valha mais do que renegociar o contrato, e que resposta rápida às mudanças do sistema valha mais que seguir rigorosamente o plano definido e documentado.

A agilidade no desenvolvimento de sistemas é obtida por meio do uso das tecnologias modernas e do comprometimento dos envolvidos que devem ser especializados, multifuncionais, estando aptos e dispostos a executar tarefas que, em metodologias tradicionais, são feitas por diferentes profissionais e sem integração (KOSCIANSKI e SOARES, 2007).

Não se pode confundir agilidade com falta de qualidade no funcionamento do sistema. Este é o grande desafio: obter qualidade com agilidade (KOSCIANSKI e SOARES, 2007).

Um princípio básico da Segurança da Informação é o fato de não existir sistema 100% seguro, devendo-se optar por um nível de segurança conforme a necessidade. Porém, é inviável se definir um nível de segurança em um sistema informatizado que não apresente funcionamento correto ou que não atenda às necessidades do usuário. A qualidade no funcionamento de um sistema não é um objetivo da Segurança da Informação, é um pré-requisito (RAWAT et al., 2010).

O tema software seguro tem recebido mais atenção dia a dia, impulsionado pela necessidade de segurança em todas as áreas da TI, pois os

incidentes de segurança apresentam um crescimento exponencial. Crescem os focos de análises para se estabelecer segurança, entre elas:

(1) nas redes, englobando estações de trabalho e servidores em geral;

(2) nas instalações físicas evitando, por exemplo, perda de disponibilidade por cabo desconectado;

(3) nos procedimentos do usuário, evitando que este contribua com ataques por Engenharia Social, seja como invasor ou como vítima;

(4) na Engenharia de Software, ramificação da Ciência da Computação que produz a camada de aplicação, evitando execução de processos de forma incorreta;

(5) na legislação vigente no local, evitando processos judiciais.

Em Segurança da Informação não existe análise de qual ação é mais importante. Qualquer vulnerabilidade não controlada possibilita uma ameaça à organização. Há a necessidade da elaboração de um Plano Diretor de Segurança, que preveja o levantamento dos ativos da TI da organização, e, para cada um deles, analisem-se as vulnerabilidades e suas respectivas ameaças.

Após o planejamento de controle de riscos de acordo com o nível de segurança estabelecido para a organização, é imprescindível o constante controle, manutenção e monitoramento do plano. Essa atividade contínua, muitas vezes acaba sendo esquecida ou colocada em segundo plano, atitude que leva a uma das maiores ameaças para o processo de segurança. Os mecanismos de segurança precisam ser eficientes sempre, pois o invasor, apenas com um ataque, pode prejudicar todo o plano segurança mantido ao longo do tempo.

Segurança da Informação é uma atividade que envolve todos os colaboradores da organização, incluindo os não usuários da TI, pois estes podem proporcionar ameaças de várias formas. Ela é sustentada por diferentes pilares envolvendo aspectos culturais, tecnológicos e legais. Com essa abrangência, Segurança da Informação se insere em duas áreas de Conhecimento: a Ciência da Informação e a Ciência da Computação. Pode-se observar a complexidade que envolve Segurança da Informação, tanto na quantidade de envolvidos (com diferentes especializações), como na diversidade de fatores e tecnologias a serem consideradas.

Dentro do grande universo monitorado pela Segurança da Informação, o foco deste artigo é a segurança em termos de aplicação, tendo como objetivo ressaltar a importância da qualidade no desenvolvimento de sistemas informatizados para Segurança da Informação, atendida sem detrimento da agilidade no desenvolvimento. Para tal, o

artigo apresenta tecnologias e ferramentas de auxílio.

Para escalar níveis de desenvolvimento de software seguro, é necessário digerir uma quantidade infinita de informações descentralizadas. Isso acontece pela dificuldade em se encontrar programas de treinamento que considerem esse tema e pela dinamicidade associada: a cada dia, novas vulnerabilidades e novas contramedidas (MERKOW e RAGHAVAN, 2010).

No desenvolvimento de um software, assim como na sua utilização, o fator humano é o elo mais fraco para a Segurança da Informação. Para o engenheiro de software, a prioridade é desenvolver, produzir. Sob pressão, caso não haja uma política de reconhecimento e valorização da qualidade e segurança no software, facilmente o engenheiro irá esquecê-la ou ignorá-la.

Infelizmente as prováveis consequências drásticas não serão identificadas de um primeiro momento. Torna-se de grande importância sensibilizar os envolvidos direta ou indiretamente com o desenvolvimento de sistemas informatizados para essa necessidade.

Este artigo discute três tópicos vitais para prover a Segurança da Informação durante o desenvolvimento de softwares: Bancos de Dados, Arquitetura em Multicamadas e Métodos Ágeis.

Foi utilizada metodologia (LAKATOS e MARCONI, 2005) de pesquisa constituída de livros, artigos e materiais disponibilizados na internet, em páginas criteriosamente selecionadas pelos autores quanto ao conteúdo e à autoria, aliada à experiência vivenciada por eles em mais de 20 anos de desenvolvimento de sistemas informatizados.

2 RESULTADOS E DISCUSSÃO

2.1 Banco de Dados Relacional Orientado a Objetos

Um banco de dados é um sistema de armazenamento de dados baseado em computador, cujo objetivo é registrar e manter informações consideradas significativas à organização. Sua correta utilização possibilita um ambiente eficiente e adequado de recuperação e armazenamento de grandes quantidades de informações.

O conceito de banco de dados originou-se na década de 1970, quando as organizações tinham um alto custo na contratação de pessoas para armazenar e organizar os dados de forma manual.

A automatização de rotinas com auxílio de banco de dados reduzia os custos e erros humanos e tem mantido essas vantagens até os dias atuais

(SILBERSCHATZ e SUDARSHAN, 2006). Ainda existe grande quantidade de sistemas não informatizados em diversas áreas, entre elas a Saúde.

Um banco de dados é criado e mantido por aplicações desenvolvidas para esta tarefa, chamados de Sistemas Gerenciadores de Banco de Dados (SGBD). Um SGBD deve isolar o usuário dos detalhes internos do banco de dados, promovendo a abstração de dados, a independência dos dados em relação às aplicações, a estratégia de acesso e a forma de armazenamento. A complexidade está abstraída em três níveis:

(1) Interno (Físico). Descreve a estrutura de armazenamento físico do banco de dados.

(2) Estrutural (Lógico). Descreve a estrutura de todo o banco de dados, ocultando detalhes das estruturas de armazenamento físico.

(3) Externo (Conceitual). Constitui-se da visão que o usuário tem do banco de dados (como os dados são visualizados por ele) (SILBERSCHATZ e SUDARSHAN, 2006).

Um recurso dos SGBDs relacionais o de Transação, uma unidade de execução de programa que acessa e, possivelmente, atualiza itens de dados. Uma transação é geralmente o resultado da execução do SGBD, ou de algum código em linguagem de programação. Um SGBD deve garantir a execução apropriada das transações com relação às falhas: ou a transação é executada (chamada então de *commited*) por completo ou nada é executado (processo é conhecido como *rolled back*). O controle através de transações é essencial para manter a integridade dos dados (ELMASRI e NAVATHEM**,** 2005).

Atualmente, predomina o modelo relacional de banco de dados nos sistemas já desenvolvidos. A tendência para novos desenvolvimentos é o modelo relacional orientado a objetos. O paradigma Orientação a Objetos favorece no desenvolvimento de software, visto que se propõe a resolver problemas básicos: diminuir o tempo de desenvolvimento, torná-lo confiável, com baixo custo e possibilidade de reaproveitamento de códigos (ELMASRI e NAVATHEM**,** 2005).

2.2 Arquitetura Multicamadas

Segundo Neward (2003), camadas são separações lógicas na estrutura do código fonte de um sistema, em que cada parte é independente o que torna mais fácil dividir as responsabilidades do sistema. A utilização de camadas ajuda a estruturar aplicativos, que podem ser decompostos em grupos de subtarefas, sendo que cada grupo representa um nível de abstração. Os primeiros sistemas produzidos possuíam uma camada, ou seja, feito em um único código que acessava os dados, tratava as

regras de negócios e fazia a interface com o usuário.

Esses sistemas eram armazenados em um único servidor. As manutenções eram complicadas e demoradas. Na década de 1990, os computadores pessoais passaram a contar com capacidade de processamento suficiente para "rodar" os programas, sem a necessidade de um servidor. Aliada a evolução das redes de computadores, os métodos de desenvolvimento de sistemas evoluíram para uma arquitetura descentralizada, em que os computadores pessoais passaram a ter um importante papel nos sistemas e aplicações, pois parte do processamento era feito na máquina do usuário, e um servidor era responsável por acessar o banco de dados. Baseado nesse modelo, surgiu a arquitetura em duas camadas, sendo que uma camada é armazenada na máquina do usuário e outra no servidor (O'DOCHERTY, 2005).

Com o advento da Internet, a arquitetura em duas camadas ficou inviável, pois a carga de complexos algoritmos de negócios na máquina cliente era muito demorada. Esse problema levou a criação de novas técnicas de desenvolvimento em multicamadas, visando melhorar o desempenho e/ou diminuir a necessidade de carregar todos esses algoritmos nas máquinas clientes.

No modelo três camadas (apresentação, negócios e dados), a camada de apresentação, localizada fisicamente na máquina cliente é responsável pela interface entre cliente e outras camadas, sendo leve, com código fonte pequeno, composto por telas do sistema e validações. A camada de negócios faz requerimentos e o tratamento no banco de dados, sendo armazenada em servidor dedicado com elevados recursos de hardware. Várias estações clientes podem se conectar a um mesmo servidor de camada de negócios.

No modelo em quatro camadas, foi retirada a camada de apresentação da máquina do cliente e colocada em um servidor. O usuário acessa o programa por meio de um Navegador de Internet e pormeio dele faz todas as operações necessárias (NEWARD, 2003).

Com a criação dos Web Services, essa tecnologia também evoluiu e os desenvolvimentos atuais utilizam a Arquitetura de Desenvolvimento de Sistemas em *n* Camadas**,** visando a uma definição de camadas mais refinadas (CANACHO, 2009).

Há a necessidade da utilização de uma linguagem de programação que suporte a arquitetura multicamadas. Um exemplo, entre as várias já disponíveis, é a linguagem C# que foi desenvolvida pela Microsoft, dentro do projeto da arquitetura Dot Net (.Net), especificamente para a plataforma multicamadas (CANACHO, 2009).

Outra tecnologia que auxilia no desenvolvimento de sistemas modernos é a .Net Framework[6], uma arquitetura de desenvolvimento criada pela Microsoft, com o objetivo de fornecer um ambiente de desenvolvimento de aplicativos distribuídos, escaláveis e poderosos.

Dentro da arquitetura.Net existe uma infraestrutura composta por um pacote com diversas tecnologias que, em conjunto, permite formar um ambiente de desenvolvimento (NAGEL, 2010). Este pacote, conhecido como .NET Framework SDK, é gratuito (MICROSOFT, 2010) e inclui compiladores de linguagens como C#, Visual Basic.NET, Managed C++ e JScript.NET, todos desenhados para criar aplicações .NET, ou seja, aplicações orientadas ao desenvolvimento de software em arquitetura de multicamadas na Internet (MICROSOFT, 2010).

Existem diversas opções de pacotes que visam à Segurança da Informação em arquitetura multicamadas. Não há uma análise que determine qual a melhor solução. Deve-se analisar a melhor opção para a implementação em questão.

A arquitetura em multicamadas trouxe uma importante contribuição para a Engenharia de Software, porém, suas peculiaridades no tocante à Segurança da Informação não podem ser ignoradas. Há por exemplo, a necessidade de autenticação por camadas, e esta apresenta suas respectivas vulnerabilidades e ameaças (PAUTOV, 2008).

A Microsoft possibilita um framework com solução para autenticação de fácil utilização para as aplicações, em que pode ser concedido acesso aos usuários de acordo com sua função dentro da organização ou da aplicação. Este framework é totalmente compatível com o sistema operacional Windows NT Server, com capacidade de segurança tanto do Active Directory como do NTFS, desde que o armazenamento de políticas de autorização se localize em um sistema confiável (MICROSOFT, 2010).

A Microsoft, reconhecendo a importância da Segurança da Informação na Engenharia de Software, desenvolveu, em 2005, um relatório de 51 páginas, o "Security Engineering Explained", considerando as preocupações elementares de seis dos seus especialistas no desenvolvimento de software seguro (MEIER, 2005).

2.3. Métodos Ágeis

Um método de desenvolvimento de sistemas é um conjunto de atividades que

[6] Em Engenharia de Software, framework é uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica (Microsoft, 2010).

auxiliam na produção do software. O resultado dessas atividades é um produto que reflete a forma como todo o processo foi conduzido. Os problemas com os projetos e a insatisfação com as abordagens pesadas levou os desenvolvedores de software na década de 1990 a proporem novos métodos. Desde então, surgem métodos, chamados de ágeis, que focam de forma diferenciada o desenvolvimento, entre eles: Scrum, eXtreme Programming (XP), Crystal, Adaptive Software Development, DSDM e Feature Driven Development (SOMMERVILLE, 2007).

A grande motivação para a agilidade e métodos iterativos está em fatos como:

(1) clientes ou usuários não têm certeza do que querem;

(2) eles têm dificuldade de dizer o que querem e o que sabem;

(3) detalhes do que eles precisam serão revelados durante o desenvolvimento;

(4) os detalhes são complexos para os usuários;

(5) como eles veem o desenvolvimento do produto, eles mudam seus pensamentos;

(6) forças externas (como um competidor, produto ou serviço) podem levar a modificações (LARMAN, 2003).

O método ágil XP é voltado ao gerenciamento do desenvolvimento do sistema, dando enfoque especial às importantes rotinas de testes. Tem foco a programação em par e a criação de soluções simples, com a liberação constante de versões funcionais do software. Sua base consiste de doze práticas, e uma delas é a produção orientada a caso-teste.

As falhas desta metodologia residem na falta de documentação e no uso constante da intuição para a resolução de problemas. Porém, mesmo não se adotando o XP como método de desenvolvimento, algumas de suas práticas podem proporcionar melhoria no processo de desenvolvimento principalmente no contexto de Segurança da Informação. Sua orientação a caso-teste atende às necessidades básicas da Segurança da Informação, como por exemplo, a criação dos testes antes do código que o mesmo irá testar e ter um teste para cada unidade desenvolvida. Esta prática pode ser utilizada em outros métodos. O XP tem sido utilizado emconjunto com o Scrum, de forma a se completarem (PEARMAN, 2006).

O Scrum é voltado ao gerenciamento do projeto e dá enfoque para o tratamento de mudanças frequentes de requisitos de software e outras situações, tais como: mudança de equipe, adaptações de cronogramas, trocas de ferramentas de desenvolvimento ou de linguagens de programação. O Scrum propõe uma forma

de trabalho flexível que se adapta a ambientes muito dinâmicos. Visa a tratar mudanças. A principal falha neste método e também sua principal qualidade: em cada projeto é necessário adaptar o processo à situação específica deste novo projeto (BHANDARKAR, 2010).

Uma falha dos métodos ágeis é a necessidade de um representante dos *stakeholders*[7] estar disponível quase o tempo todo. Isso se torna um problema principalmente se houver muitos *stakeholders* com prioridades distintas. Porém, cada *stakeholder* irá compensar o tempo gasto em curto prazo, com a eficiente e eficaz utilização do sistema em questão (SOMMERVILLE, 2007).

Como apoio, é utilizada a diagramação UML[8]. Diagramas UML podem ser empregados para visualizar, especificar, construir e documentar os artefatos de sistemas de software. Ela é o resultado da unificação da linguagem de modelagem de objetos entre os métodos líderes do mercado (O'DOCHERTY, 2005).

3 CONSIDERAÇÕES FINAIS

Independente do tipo de aplicação, há necessidade de garantir a integridade das informações que ela disponibiliza: corretas, atualizadas e em tempo hábil para a tomada de decisões.

Para tal, é preciso um controle corretamente elaborado das transações em banco de dados, assim como uma arquitetura em multicamadas para preservar a integridade dos programas na implantação e nas atualizações de versão, evitando manutenções diretamente nas máquinas clientes.

O uso crescente de sistemas informatizados em todas as funções de uma organização e sua constante evolução fizeram surgir novos métodos de desenvolvimento para que estes atendessem, em tempo hábil e de forma fácil, aos *stakeholders*. O paradigma da orientação a objetos aliado a métodos ágeise UML, agregou valor ao processo de desenvolvimento.

O emprego de novas tecnologias deve ser estudado em relação ao seu uso nas atividades realizadas pela organização e principalmente na dinamização destas tecnologias para obter a integridade de dados e o desenvolvimento e manutenção

---

[7] S*takeholder* é qualquer pessoa ou entidade que afeta ou é afetada pelas atividades de uma empresa (Sommerville, 2007).

[8] UML (*Unified Modeling Language*) é uma linguagem gráfica para a elaboração da estrutura de projetos de software, promovendo: (1) Disponibilizar mecanismos de especificação em níveis conceituais; (2) Tornar o processo de desenvolvimento independente da linguagem em níveis conceituais; independente da linguagem de programação; (3) Incentivar o uso do paradigma OO; e (4) suportar conceitos de programação de alto nível.

em tempo ágil dos programas de um sistema informatizado.

É fundamental ressaltar a importância para a Segurança da Informação, da qualidade no desenvolvimento ágil de sistemas informatizados, pois, caso não haja uma política de reconhecimento e valorização da qualidade e segurança no software (pelos desenvolvedores e *stakeholders*), facilmente o engenheiro de software irá esquecê-la ou ignorá-la. Infelizmente as drásticas consequências não serão identificadas de momento.

REFERÊNCIAS

BHANDARKAR, J. **Scrum software development.** CRC Press. 2010.

CANACHO JR, C. O. de A.

Desenvolvimento em camadas com C#.NET. VISUAL BOOKS, **2008.**

ELMASRI, Ramirez; NAVATHEM**,**

Shamkant B**. Sistema de banco de dados**. 4ª ed. São Paulo: Pearson Addison Wesley. 2005.

KOSCIANSKI, A.; SOARES, M. dos S.

Qualidade de software: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. 2ª ed. São Paulo: Novatec, 2007.

LAKATOS, E. M.; MARCONI, M. A.

Fundamentos de metodologia científica. 6ª ed. São Paulo: Atlas, 2005.

LARMAN, Craig. **Agile and iterative development: a manager's guide.** São Paulo: Addison-Wesley Professional, 2003.

MEIER, J.D et al. **Microsoft**: security engineering explained. Microsoft Corporation. 2005.

MERKOW, M. S.; RAGHAVAN L. **Secure and resilient software development**. **CRC** Press. 2010.

Microsoft. **.NET Framework SDK.** Disponível em: http://www.microsoft.com/ Acesso: 03 mar. 2010.

NAGEL, C. et al. **Professional C# 4 And .Net 4.** Wrox **Press**, Inc. 2010.

NEWARD, T. **Arquitetura pragmática.** 2003. Disponível em: http://msdn.microsoft.com/pt-br/library/aa905336.aspx. Acesso: 09 set. 2010.

O'DOCHERTY, Mike. Object-oriented analysis and design: understanding system development with UML 2.0. John Wiley & Sons, Inc. 2005.

PAUTOV, P. A. The problem of authentication in the multi-tier applications. Prikl. Diskr. Mat. no. 2, 87– 90. 2008.

PEARMAN, G. Pro .Net extreme programming. APRESS. 2006.

RAWAT, S.; SAXEBA, A.; GOPAL, P. KB H. Software vulnerabilities exposed. Series: Infosys Press. CRC Press. 2010.
SILBERSCHATZ, A; KORTH, H. F;

SUDARSHAN, S. Sistemas de banco de dados. São Paulo: Ed. Campus. 2006.

SOMMERVILLE, Ian. Engenharia de software. 8a. ed. São Paulo: Pearson Addison Wesley. 2007.