

# Da Estratégia à Solução: Uma abordagem leve e semi-prescritiva para o Ciclo de Vida de Desenvolvimento de Software com Suporte à Terceirização

Nélio Alves  
Instituto Federal do Triângulo  
Mineiro  
nelio@iftriangulo.edu.br

Sérgio Paim  
especificacoes.com  
sergio.paim@especificacoes.com

Alexandre Cardoso  
Universidade Federal de  
Uberlândia  
alexandre@ufu.br

Edgard Lamounier  
Universidade Federal de  
Uberlândia  
lamounier@ufu.br

*Resumo - Este artigo apresenta um trabalho de pesquisa, em andamento através de uma colaboração empresa-escola, de especificação e implementação de um método leve e semi-prescritivo para o ciclo de vida de desenvolvimento de software, combinando características ágeis e prescritivas. Este método propõe características e decisões de projeto que serão descritas e discutidas. Por exemplo, escolhemos gerenciamento formal de requisitos com a devida rastreabilidade de mudança, bem como uma arquitetura de software prescritiva a fim de proporcionar vantagens como escalabilidade de equipe, realizações confiáveis de requisitos e gestão do escopo. Introduzimos o conceito de cenário de implementação: um subconjunto específico de requisitos – incluindo não-funcionais – que melhoram consideravelmente o gerenciamento de risco e a eficiência geral na estabilização da arquitetura, e também é um dos aspectos chaves para a integração ágil-prescritivo. O método também propõe um suporte nativo para a terceirização de desenvolvimento de software, que consiste em uma interface entre cliente e fornecedor centrada em processos, baseada em atribuição de atividades, custódia de artefatos e a própria concepção do método. Até o momento, o método foi testado informalmente no desenvolvimento de projetos de pequeno e médio portes. Os testes mostraram resultados significativos que serão apresentados e discutidos.*

## I. INTRODUÇÃO E MOTIVAÇÃO

Uma recente pesquisa sobre metodologias de desenvolvimento de software orientadas a processos (1) analisou o estado da arte deste campo de pesquisa, examinando a essência dos métodos disciplinados e ágeis. Dentre as conclusões desta pesquisa, discutimos aquelas que servem como parte da motivação para nosso trabalho:

- **Necessidade de integração:** Segundo análises, métodos disciplinados e seus homólogos ágeis não têm outra escolha senão a convergência. Outra fonte de inspiração para esse artigo é o trabalho publicado em (2).
- **Negligência metodológico:** Métodos de desenvolvimento de software são geralmente desenvolvidos sem considerar uma base metodológica adequada ou uma abordagem sistêmica.
- **Problemas comuns de abordagens ágeis:** Foi concluído que, apesar de notáveis realizações, métodos ágeis não estão maduros o suficiente (2) (3) (4) (5) e os problemas mais comumente citados são a falta de escalabilidade, os pressupostos irrealistas e a falta de um processo específico e não-ambíguo.

Destacamos ainda algumas outras questões que motivam este trabalho:

- **Terceirização de desenvolvimento de software:** A pesquisa mencionada (1) sequer fala em terceirização. Podemos observar a falta de uma abordagem metodológica centrada em processos para suportar terceirização. Em outras palavras, os processos de

desenvolvimento de software geralmente não incluem apoio específico para a terceirização.

- **Requisitos:** Gerenciamento formal de requisitos é crucial para garantir uma adequada comunicação e concordância entre cliente e fornecedor. Em (1), Engenharia de Requisitos é considerado ponto fraco em muitos métodos. Também observamos isto em nossa experiência e foi decidido que uma abordagem formal com a adequada rastreabilidade de mudanças de requisitos é obrigatório para o propósito deste trabalho.

A fim de solucionar os problemas mencionados, este artigo apresenta um método para gerenciamento do ciclo de vida de aplicações e discute suas características gerais e específicas que tratam as questões citadas.

Uma experiente empresa de TI está desempenhando um papel fundamental nessa pesquisa com grandes contribuições na especificação e teste do processo. O método tem sido testado em projetos de pequeno e médio porte e mostraram resultados significativos, principalmente na melhoria da eficiência e no gerenciamento de riscos.

Este artigo é considerado uma proposta, pois ainda não foram realizadas validações quantitativas.

## II. TRABALHOS RELACIONADOS

Consideramos que os principais trabalhos relacionados a esta pesquisa estão classificados em vertentes: 1 - metodologia de desenvolvimento de software e 2 - terceirização de software.

Existem várias abordagens metodológicas ágeis e disciplinadas para o ciclo de vida de desenvolvimento de software. Esta breve pesquisa focou nas abordagens que influenciaram este trabalho. Mais informações sobre metodologias de softwares podem ser encontradas em (1).

Rational Unified Process (RUP) (6) (7) e seu correspondente não-proprietário USDP (8) foram os principais provedores da base teórica e prática para este trabalho. Além disso, é amplamente utilizado por mais de seis anos pela empresa onde este trabalho está sendo aplicado. O RUP é um método dirigido por casos de uso, centrado em arquitetura, iterativo e incremental e utiliza a Unified Modeling Language (UML) (9) como linguagem de modelagem padrão. O RUP é caracterizado por ser intenso em especificação e pode ser customizado para melhorar sua gerenciabilidade. Uma variação derivada do RUP chamada Enterprise Unified Process (10) foi proposta após sua criação.

Scrum (11) é um framework para desenvolvimento de software, primeiramente apresentado em (12), focado numa alta interação da equipe e na estruturação de iterações de desenvolvimento de 2 a 4 semanas chamadas de sprints. Este framework possui três fases cíclicas: pré-game (planejamento, projeto em alto nível, arquitetura), desenvolvimento (execução do sprint) e pós-game (integração e atividades de entrega). Scrum geralmente é utilizado combinado a outro método, normalmente ágil, e prescreve a adoção de algumas práticas como reuniões diárias, utilização de um quadro de tarefas visível contendo as ordens de trabalho e a divulgação

de um gráfico Burndown. Nossa abordagem incorporou a utilização do quadro de tarefas prescrito pelo Scrum, como parte de uma das atividades de gerenciamento, o que se mostrou bastante efetivo.

A maioria das pesquisas trata sobre a terceirização de sistemas de informação, mas poucas trataram, de forma específica, a terceirização do desenvolvimento de software (13). Encontramos muitas publicações tratando sobre o abrangente campo de terceirização de TI e mais ainda, tratando genericamente a dinâmica de terceirização entre empresas. Vários tópicos sobre terceirização são freqüentemente apresentados no formato de listas de fatores de sucesso (15), estudos de casos (16) e guidelines (17).

Não encontramos nenhuma abordagem metodológica para a terceirização do desenvolvimento de software similar a nossa proposta. A análise da literatura mostra que ainda há muito que fazer em se tratando de abordagens metodológicas centradas em processos.

### III. VISÃO GERAL DO MÉTODO

Nosso método de ciclo de vida de desenvolvimento de software foi definido em sete processos de TI, agrupados de acordo com as áreas funcionais Projeto e Transição de Serviço da estrutura de serviço do ITIL – uma biblioteca amplamente aceita para boas práticas no gerenciamento de serviços de TI (18). A Tabela I mostra estes sete processos agrupados.

TABELA I

Processos de TI para Projeto e Transição de Serviço

Projeto	Transição
<ul style="list-style-type: none"><li>Projeto de Serviço</li><li>Engenharia de Processos</li><li>Engenharia de Requisitos</li></ul>	<ul style="list-style-type: none"><li>Desenvolvimento de Mudança</li><li>Engenharia de Software</li><li>Validação de Mudança</li><li>Implantação de Mudança</li></ul>

Uma breve descrição da responsabilidade de cada processo de TI é apresentada abaixo. Na próxima seção, daremos maiores informações sobre o padrão de especificação do processo.

**Projeto de Serviço:** este processo primeiramente utiliza o direcionamento estratégico e revisa o portfólio de serviços através da identificação e descrição dos serviços de TI que suportam a demanda de automação em análise. A seguir, entra em ação o processo de Engenharia de Requisitos cujo propósito é levantar o escopo inicial da aplicação que suportará o serviço e realiza análises de viabilidade funcional e econômica, estruturando as iniciativas para o desenvolvimento da solução, caso a viabilidade seja confirmada. Se necessário, o processo de Engenharia de Processos é previamente realizado, gerando o modelo de negócio e as demandas de automação. O propósito do Projeto de Serviço é similar à fase de Concepção do RUP.

**Engenharia de Processos:** responsável pela modelagem de negócio com alinhamento estratégico e melhoria contínua. Mais detalhes serão discutidos na próxima seção.

**Engenharia de Requisitos:** responsável pela execução do ciclo de requisito. Este ciclo consiste na identificação e detalhamento dos requisitos de automação (sem violar a integridade da arquitetura), gerenciamento formal da rastreabilidade de mudanças, estimativa de escopo e validação formal entre cliente e fornecedor.

**Desenvolvimento de Mudança:** responsável pela execução das atividades de gerenciamento de projeto durante o estágio de Transição de Serviço.

**Engenharia de software:** responsável pela execução do ciclo de desenvolvimento, composto por atividades de refinamento da arquitetura, projeto, implementação, testes, integração e confecção de material de suporte. Este é um processo adaptado aos métodos ágeis. Mais detalhes são apresentados na Seção V.

**Validação de mudança:** responsável pela execução do ciclo de validação, o qual consiste em atividades relacionadas à implantação do ambiente de homologação, validação do software aplicativo e dos serviços de TI suportados.

**Implantação de mudança:** responsável pela execução de atividades necessárias à implantação da nova solução em produção (como serviço de TI), tais como definição dos recursos de infra-estrutura, execução do plano de migração, testes de estabilidade e treinamento de usuários.

### IV. METODOLOGIA UTILIZADA

Todos os sete processos de TI foram modelados de acordo com os preceitos de Engenharia de Processos. Na verdade, o método utilizado foi exatamente o processo “Engenharia de Processos” que nós criamos (ver Tabela I). Observe que este é um meta-processo, visto que é um processo de negócio que detalha como especificar os processos de negócio (onde os processos de TI são apenas uma especialização).

Nosso processo de “Engenharia de Processos” foi altamente inspirado pela disciplina Modelagem de Negócio do RUP, onde alguns aspectos foram simplificados e outros foram criados.

Definimos uma especificação textual complementada pela representação gráfica de cada processo de TI e suas atividades. Por restrições de espaço, apresentamos apenas os modelos comportamental e estrutural do processo de “Engenharia de Processos” na Figura 1 e 2. Observe que foram utilizados diagramas UML do tipo atividades e classes, com o uso apropriado de estereótipos.

### V. DECISÕES E CARACTERÍSTICAS DE PROJETO

Nesta seção, discutimos as principais decisões de projeto tomadas e apresentamos as razões que nos levaram a estas escolhas. Esta discussão é complementada na próxima seção, onde apresentamos as decisões relacionadas à terceirização.

**Gerenciamento formal de requisitos:** o processo de Engenharia de Requisitos foi definido para apoiar a logística de terceirização de requisitos. Em cada ciclo de requisitos (incluindo o primeiro), gera-se um documento de Alteração de Especificação e cria-se uma linha base com os requisitos relacionados a esta alteração. Após a identificação e detalhamento dos requisitos, feita em colaboração entre cliente e fornecedor, os requisitos são validados formalmente pelo cliente e então é criada uma linha base pelo Gerente de Projetos do cliente e comunicado o valor de contrato para ambas as partes. Esta estrutura metodológica garante o correto gerenciamento de requisitos e a rastreabilidade das mudanças.

**Arquitetura prescritiva:** um aspecto que priorizamos, especialmente se tratar-se de um ambiente com terceirização, é a arquitetura de solução feita de forma prescritiva. Uma boa abordagem arquitetural da solução garante a concordância sobre mitigação de riscos técnicos, decisões tecnológicas e realização de requisitos. Também deve prover escalabilidade da equipe e auxiliar na definição realista do escopo.

**Construção ágil de software:** nosso processo de Engenharia de Software é a parte ágil de nosso método. Foi projetado baseado em alguns princípios ágeis (19). Além disso, como mencionamos anteriormente, adotamos algumas práticas do Scrum como o quadro de tarefas, reuniões diárias e iterações curtas (como se fossem sprints). Entretanto surge uma importante questão: é conhecido que o refatoramento de arquitetura e o uso de requisitos exploratórios são os aspectos centrais da maioria dos métodos ágeis, mas utilizamos uma abordagem contrária para garantir as vantagens mencionadas anteriormente. Então, como é possível melhorar a dinâmica da fábrica de software com abordagens ágeis utilizando gerenciamento de requisitos e arquitetura prescritivos? A natureza exploratória e dinâmica do desenvolvimento ágil pode ser aplicada pela utilização de Cenários de Implementação apresentados abaixo.

**Cenário de Implementação:** introduzimos o conceito de cenário de implementação: um subconjunto do requisito – incluindo requisitos não-funcionais – baseado em risco, convenientemente escolhido pelo Arquiteto de Solução de acordo com o momento do ciclo de vida, com foco na implementação para priorização arquitetural. No RUP, o requisito é a unidade básica da priorização arquitetural, o que implica na perda de tempo e de capacidade de mitigação de riscos devido à implementação de porções não arquiteturais na fase de elaboração. A função principal do cenário de implementação é dividir o requisito e priorizar apenas as porções apropriadas para cada fase de desenvolvimento. Além disso, os cenários de implementação desempenham importante papel na integração ágil e prescritiva: a arquitetura é prescritiva, mas a estabilização arquitetural e a construção do software são feitas através da identificação e priorização exploratória dos cenários.

## VI. PROPOSTA DE TERCEIRIZAÇÃO

A contribuição que este trabalho realiza para a terceirização de desenvolvimento de software está relacionada com sua estrutura intrínseca, a qual consiste em uma interface cliente-fornecedor bem definida sob três pontos de vista: 1 - atribuição de atividades, 2 - custódia de artefatos e 3 - a própria concepção do método.

**Atribuição de atividades:** especifica se a atividade é realizada no cliente, no fornecedor ou em ambos. Representamos isto graficamente pela diferenciação de cores de cada atividade nos diagramas de atividade.

**Custódia de artefatos:** definimos um mapeamento complementar dos artefatos a serem desenvolvidos e entregues ao longo do ciclo de vida de desenvolvimento. Este mapeamento deve ser instanciado para cada projeto a ser executado. Os artefatos são classificados por 1 - fase (fase inicial onde é produzido), 2 - obrigatoriedade, 3 - intervenção (como o cliente trabalha com o artefato: revisão, acompanhamento, aprovação ou aceite) e 4 - formato/ferramentas (quais ferramentas ou formatos de arquivos são utilizados para a produção do artefato).

**Concepção do método:** embora o método defina requisitos formais e arquitetura prescritiva, a construção do software (dentro da fábrica de software) é baseado na implementação exploratória dos cenários de implementação, empregando o estilo ágil. O principal aspecto técnico da interface de terceirização é a arquitetura: inicialmente ela é projetada prescritivamente com participação mútua, mas o Arquiteto de Solução do fornecedor é livre para escolher os cenários de implementação exploratórios no processo de Engenharia de Software. Por outro lado, o principal aspecto de gerenciamento da interface de terceirização é a gestão formal dos requisitos, o qual fornece base para o escopo,

orçamento, planejamento, gerenciamento da qualidade e, além de tudo, o aceite do projeto.

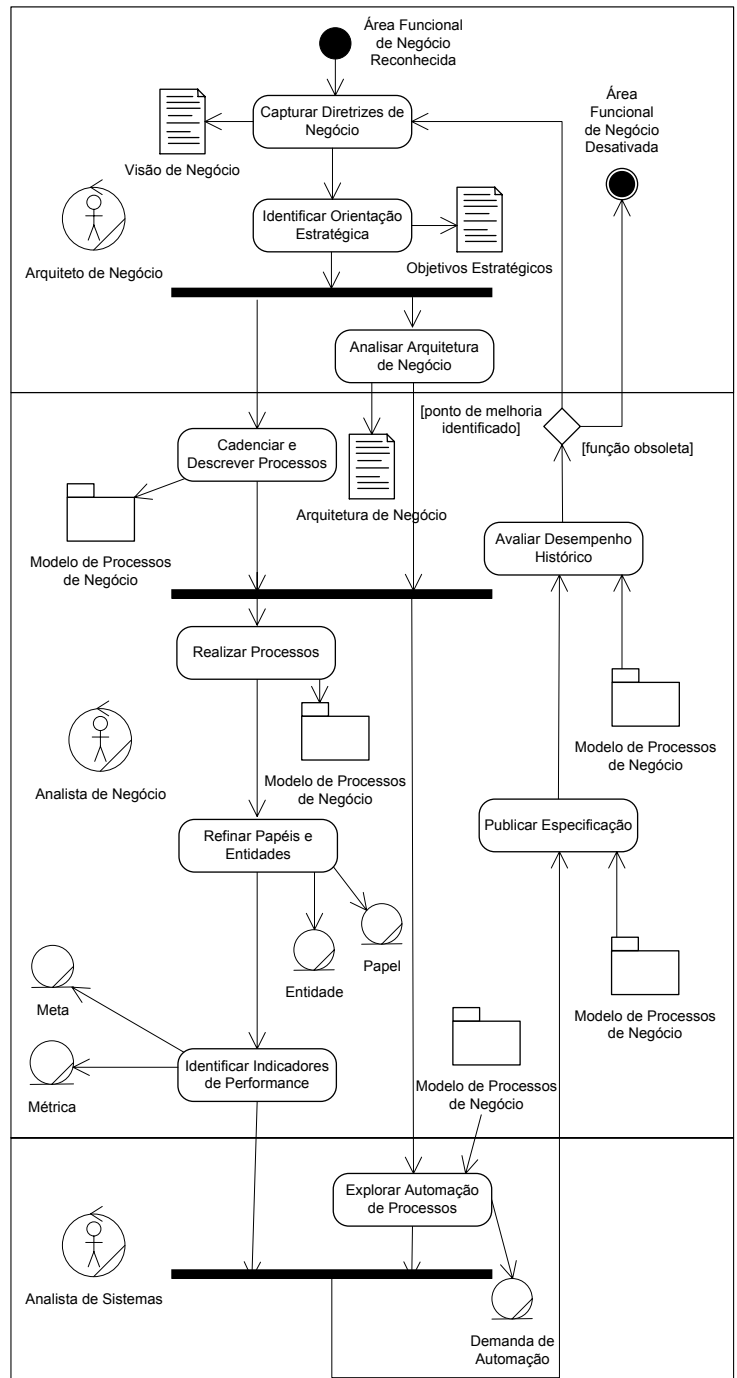


Fig. 1 - Processo de "Engenharia de Processos."

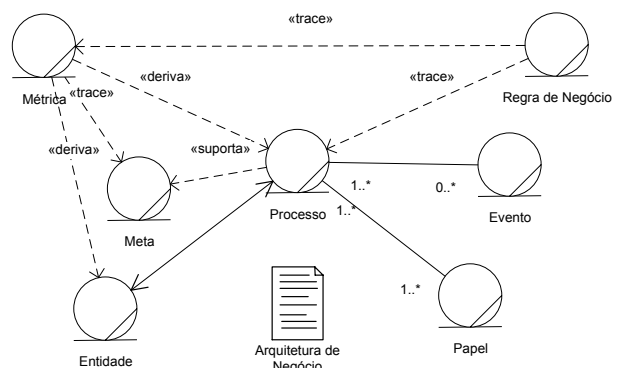


Fig. 2 - Entidades e relacionamentos referentes à engenharia de processos.

## VII. RESULTADOS E TESTES EMPÍRICOS

Como mencionado anteriormente, uma experiente empresa de TI tem testado este método por mais de seis meses

em quatro projetos de pequeno e médio porte (variando de 1.000 a 7.000 horas cada), fazendo o “papel de fornecedor”. O “papel de cliente” tem sido realizado por um de seus clientes (um grande grupo empresarial que possui um departamento de TI).

O Gerente de Projetos e o Diretor de Tecnologia foram entrevistados para relatar as melhorias percebidas com a utilização do novo método. As principais melhorias estão listadas abaixo:

**Gerenciamento de risco:** a utilização de cenários de implementação melhorou muito o gerenciamento de riscos técnicos e a estabilização da arquitetura. Através da orientação a riscos feita de forma exploratória, prioriza-se porções de requisitos com foco em aspectos realmente importantes para a estabilização da arquitetura, melhorando a efetividade e mitigando riscos de atraso nas fases posteriores.

**Produtividade:** a utilização de abordagens ágeis para a construção do software melhorou a dinamismo e flexibilidade da fábrica de software. O aumento da eficiência no desenvolvimento foi especialmente percebido pela utilização de iterações menores e do quadro de tarefas físico.

## VIII. CONCLUSÕES

Apresentamos um método do ciclo de vida de desenvolvimento de software semi-prescritivo definindo sete processos de TI. O método foi construído em uma base bem definida de Engenharia de Processos, como uma atitude positiva frente à negligência metodológica geralmente apresentada na concepção de métodos de software (1).

Um resumo das principais contribuições é apresentado abaixo:

- Foi proposta uma solução para a integração dos métodos ágeis e disciplinados. Essencialmente, o gerenciamento formal de requisitos e a arquitetura prescritiva foram inspirados no mundo disciplinado e a exploração dos cenários de implementação (orientados a risco) e as práticas ágeis para fábrica de software foram obtidos do mundo ágil. Em outras palavras, desta maneira podemos melhorar as dinâmicas de desenvolvimento sem prejudicar a definição do escopo e a escalabilidade da equipe.
- Propusemos um método com suporte nativo ao desenvolvimento de software terceirizado. Este suporte consiste de uma interface centrada no processo entre o cliente e o fornecedor baseada na atribuição de atividades, custódia de artefatos e a própria concepção do método.
- Testes informais foram executados e resultados significativos foram reportados, principalmente referentes ao gerenciamento de riscos e produtividade.

Pode-se verificar que todos os aspectos citados na Seção I foram desenvolvidos nesta abordagem.

## IX. TRABALHOS FUTUROS

Estão sendo executados ou planejados incrementos a este trabalho. Estamos definindo métricas de desempenho deste método a fim de validá-lo através de evidências quantitativas tangíveis. Outra direção futura é sobre a integração com outras áreas de TI (como consultoria ao negócio) e refinamento contínuo das proposições aqui colocadas.

Este método também tem o objetivo de ser aplicado combinado com a abordagem de Application Lifecycle Management (ALM), como parte do modelo geral de governança de TI para ambientes multi-sourcing.

## X. REFERÊNCIAS

1. Ramsin, R. e PAIGE, R. F. Process-Centered Review of Object Oriented Software Development Methodologies. ACM Computing Surveys. Fevereiro de 2008. Vols. 40, No. 1, Article 3
2. Boehm, B. e Turner, R. Balancing Agility and Discipline: A Guide for the Perplexed. Reading, MA : Addison Wesley, 2003.
3. Abrahamsson, P., et al. New directions on agile methods: A comparative analysis. Proceedings of the International Conference on Software Engineering, (ICSE). 2003.
4. Boehm, B. e Turner, R. Management challenges to implementing agile processes in traditional development organizations. IEEE Software, 22, 5. (Setembro/Octubre) de 2005. pp. 30-39.
5. Nerur, S., Mahapatra, R. e Mangalaraj, G. Challenges of migrating to agile methodologies. Communication of the ACM 48. Maio de 2005. pp. 73-78.
6. Kruchten, P. Rational Unified Process: An Introduction. Reading, MA : Addison-Wesley, 2003. 3rd ed.
7. Kroll, P. e Kruchten, P. The Rational Unified Process Made Easy: A Practitioner's Guide to Rational Unified Process. Reading, MA : Addison-Wesley, 2003.
8. Jacobson, I., Booch, G. e Rumbaugh. Unified Software Development Process. Reading, MA : Addison-Wesley, 1999.
9. Unified Modeling Language Specifications (v2.0). [Online] Object Management Group, 2004. <http://www.omg.org/technology/documents/formal/uml.htm>.
10. Ambler, S. W., Nalbone, J. e Vizdos, M. J. The Enterprise Unified Process: Extending the Rational Unified Process. Englewood Cliffs, NJ : Prentice-Hall, 2005.
11. Schwabe, K. e Beedle, M. Agile Software Development with Scrum. Englewood Cliffs, NJ : Prentice-Hall, 2001.
12. Schwaber, K. SCRUM development process. Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'95). 2005.
13. Gonzalez, R., Gascoa, J. e Llopis, J. Information systems outsourcing: A literature analysis. Vols. 43, Issue 7.
14. Lee, J., et al. IT outsourcing evolution---: past, present, and future. Communications of the ACM. Maio de 2003. Vols. 46, Issue 5.
15. Pei, Z., Zhen-xiang, Z. e Chun-ping, H. Study on Critical Success Factors for IT Outsourcing Lifecycle. Int Conf. on Wireless Communications, Networking and Mobile Computing. 2007.
16. Martin, A., Biddle, R. e Noble, J. When XP Met Outsourcing. Extreme Programming and Agile Processes in Software Engineering, Lecture Notes in Computer Science. 2004, Vol. 3092.
17. Yalaho, A. A Conceptual Model of ICT-Supported Unified Process of International Outsourcing of Software Production. IEEE International Enterprise Distributed Object Computing Conference Workshops, IEEE Computer Society. 2006.
18. ITIL - Information Technology Infrastructure Library. [Online] <http://www.itil-officialsite.com>.
19. Bech, K. Manifesto for agile software development. [Online] et al. 2001. <http://agilemanifesto.org>.

**Tradução do artigo:**

**Felipe Rezende e Thiego Bisco - especificacoes.com**