DOI: 10.5102/un.gti.v5i2.3413

Uma proposta de customização de metodologia para o desenvolvimento de sistemas de informação de nível acadêmico*

A proposal for customizing a system development methodology

José Gladistone da Rocha[1]
Carlo Kleber da Silva Rodrigues[2]

Resumo

O desenvolvimento de um sistema de informações consiste na construção de um software por meio de métodos e técnicas da Engenharia de Software. Existem várias metodologias de desenvolvimento, com abordagens peculiares para orientar, organizar e ordenar as etapas envolvidas nesse processo. Essas metodologias, entretanto, podem tornar-se complexas e extrapolar as necessidades reais para o desenvolvimento de sistemas mais simples, especialmente aqueles de nível acadêmico. Dentro deste contexto, este artigo tem o objetivo de realizar uma proposta de customização de metodologia de desenvolvimento de sistemas. A fundamentação desta proposta está na identificação das principais dificuldades de discentes de graduação ao realizarem seus trabalhos de conclusão de curso, bem como na análise de metodologias já consagradas da literatura.

Palavras-chave: Software. Metodologia. Engenharia. Acadêmico. Sistema.

Abstract

The development of an information system involves the construction of a software product by means of methods and techniques of the Software Engineering. There are several development methodologies with unique approaches to guide, organize and sort the tasks involved in this process. These methodologies may though become complex and extrapolate the actual needs for developing more simple systems, especially those at academic level. Within this context, this article aims at proposing a how-to solution to customize a system development methodology. The basis of this proposal lies on the identification of the main difficulties faced by graduate students when elaborating their final thesis as well as on the analysis of literature well-known development methodologies.

Keywords: Software. Methodology. Engineering. Academic. System.

* Recebido em: 08/06/2015.
Aprovado em: 17/09/2015.

[1] Graduado em Formação de Oficiais pela Academia Militar das Agulhas Negras (1985). Bacharel em Sistemas de Informação pelo Centro Universitário do Sul de Minas (2014). É Pós-Graduado em Análise de Sistemas pelo Centro de Estudos de Pessoal do Exército; Pós-Graduado em Criptografia e Segurança em Redes de Computadores pela Universidade Federal Fluminense; e Pós-Graduado em Docência do Ensino Superior pela Universidade Castelo Branco. Atua na área de Ciência da Computação e é professor em Instituições de Ensino Superior de Brasília, DF.

[2] Possui graduação em Engenharia Elétrica pela Universidade Federal da Paraíba (1993), mestrado em Sistemas e Computação pelo Instituto Militar de Engenharia (2000) e doutorado em Engenharia de Sistemas e Computação pela Universidade Federal do Rio de Janeiro (2006). Trabalha como engenheiro no Estado-Maior do Exército Brasileiro e é professor do Centro Universitário UniCEUB, em Brasília, DF. Suas principais linhas de pesquisa são: redes de computadores, modelagem e análise de sistemas computacionais, streaming e arquiteturas P2P.

1 Introdução

Para o desenvolvimento de um Sistema de Informações (SI), tornam-se essenciais o entendimento dos processos de negócio a serem automatizados, a identificação detalhada dos requisitos (PATIL; YOGI, 2011) que devem ser atendidos e, ainda, a análise criteriosa para a definição do Projeto de Desenvolvimento de Software (PDS) e a arquitetura a ser implementada (AL-HAGERY, 2012).

Em específico, o desenvolvimento de um SI consiste na construção de um produto que é fruto de um trabalho de Engenharia de Software (ES), com aplicação de métodos e técnicas para garantir resultados com qualidade, sem comprometer orçamento e prazo, bem como prover a satisfação final do cliente (AL-HAGERY, 2012; PAULA FILHO, 2006; SHARMA et al., 2012; SOMMERVILLE, 2011).

A ES vem evoluindo ao disponibilizar as chamadas Metodologias de Desenvolvimento de Sistemas (MDS), as quais podem ser prescritivas ou ágeis (PATHAK et al., 2012; AL-HAGERY, 2012; SHARMA et al., 2012; DYCK; MAJCHRZAK, 2012). Cada MDS possui uma abordagem peculiar que visa orientar, organizar e ordenar as etapas de construção de um software para possibilitar o gerenciamento e controle (ALFONSO; BOTÍA, 2005) na execução do PDS.

Essas metodologias, entretanto, podem tornar-se complexas e extrapolar as necessidades reais para o desenvolvimento de sistemas mais simples, especialmente aqueles desenvolvidos ainda em nível acadêmico.

Ante o exposto, este artigo tem o objetivo de realizar uma proposta de como customizar uma MDS para o desenvolvimento de um SI mais simples, com enfoque principal no meio acadêmico. Esta proposta tem sua fundamentação estabelecida a partir da identificação das principais dificuldades de discentes de graduação ao realizarem seus trabalhos de conclusão de curso (TCC), em Instituições de Ensino Superior (IES), bem como da análise de diferentes MDS já consagradas na literatura.

O restante deste texto é organizado como descrito a seguir. A seção 2 aborda os trabalhos relacionados. A seção 3 apresenta a caracterização do ambiente de pesquisa, as metodologias de coleta de dados e, por fim, a análise desses dados, com especial enfoque no contexto acadêmico. A seção 4 avalia as MDS mais consagradas da literatura. A Seção 5 traz a proposta de customização. Por fim, a seção 6 apresenta as conclusões finais e indicações de possíveis trabalhos futuros.

2 Trabalhos relacionados

O desenvolvimento de soluções em Tecnologia da Informação e Comunicações (TIC) é uma tarefa complexa (SHARMA et al., 2012) que exige de seus integrantes a capacidade de comunicação, a vocação para gerência de pessoas, e a detenção de conhecimentos técnicos. Isso para que se obtenha pleno sucesso no projeto a ser desenvolvido (AL-HAGERY, 2012; LANGLEY; RONEN, 2011).

Dentre as MDS que podem ser adotadas, as mais conhecidas são: Cascata, Espiral, Prototipação, Iterativo, Incremental, Processo Unificado e *Rapid Application Development* (RAD), como prescritivas; e Scrum, *Extreme Programming* (XP) e Desenvolvimento Voltado a Funcionalidades, como ágeis (AL-HAGERY, 2012; PATHAK et al., 2012; PATEL et al., 2004; SHARMA et al., 2012; SOMMERVILLE, 2011; DYCK; MAJCHRZAK, 2012).

MDS tradicionais e ágeis têm notadamente vantagens e desvantagens. Idealmente, é necessário encontrar uma solução que combine as vantagens de ambas, com a eliminação ou redução das desvantagens. Porém, equilibrar abordagens ágeis e tradicionais é uma tarefa bastante intrincada. Na prática, isso tem levado os profissionais de TI a preferirem aplicar um método ágil ou tradicional puro (SEYAM; GALAL-EDEEN, 2011; BOEHM; TURNER, 2005).

A escolha de qual MDS deve ser adotada depende do tipo de *software* a ser construído e da maturidade do grupo de desenvolvedores (SHARMA et al., 2012; DYCK; MAJCHRZAK, 2012). Essa é uma decisão gerencial importante para o sucesso de projetos (ALFONSO; BOTÍA, 2005). O desconhecimento dessa metodologia pela equipe desenvolvedora é fator de insucesso, particularmente quanto a prazos e custos envolvidos (RUP, 2007; AL-HAGERY, 2012).

Muito raramente se utiliza uma MDS de forma rígida. Na prática, as empresas utilizam método adaptados. Patel et al.(2004) apresentam quatro abordagens que podem ser adotadas para definir uma metodologia: criar a partir do zero; selecionar dentre as disponíveis no mercado ou literatura; misturar e combinar fragmentos de métodos de diferentes metodologias; e customizar baseando-se, em uma metodologia para projetos específicos. Acrescentam, ainda, que a metodologia customizada

representa uma solução equilibrada, pois mantém os benefícios da padronização, enquanto permite flexibilidade controlada e adaptação a contextos específicos.

3 Ambiente de pesquisa, coleta de dados e análise

3.1 Caracterização do ambiente

Nos cursos de Sistema de Informações das IES pesquisadas, o TCC é desenvolvido em grupos de dois a cinco alunos, sendo materializado em um PDS para atender a um problema identificado em uma empresa alvo.

O PDS supracitado possui as seguintes características: 1) escopo reduzido, envolvendo de oito a quinze Casos de Uso (UC); 2) não há mudanças significativas nos requisitos ao longo do desenvolvimento; 3) todos os UC são especificados; 4) no mínimo, são implementados três UC, sendo um do tipo CRUD (*Create*, *Retrieve*, *Update*, *Delete*), outro peculiar ao negócio, e o último referente a algum relatório gerencial; e 5) constam nos trabalhos escritos os artefatos de projeto de *software* e de gestão do projeto.

Nas IES pesquisadas, os conceitos e estudos de casos aplicados à ES são praticados pelos alunos em duas disciplinas de TCC, englobando dois semestres letivos. Nessas oportunidades são apresentadas MDS e ferramentas de modelagem que auxiliam as atividades práticas, além do estabelecimento de um plano de comunicação entre os integrantes dos grupos e seus orientadores.

Ainda, nas IES pesquisadas, a prática de orientação ocorre em dois eixos distintos. O primeiro presencial, para retiradas de dúvidas pontuais, e o segundo, mais dinâmico, por meio de troca de arquivos eletrônicos. Dessa forma, disponibiliza-se mais tempo aos alunos para trabalharem em seus grupos.

3.2 Coleta de dados

Os dados para embasar a justificativa da pesquisa deste trabalho foram coletados no período dos anos de 2009 a 2014, considerando-se a realização do TCC por 140 alunos do último semestre letivo do curso de bacharelado em Sistema de Informações, de duração de quatro anos e em duas distintas IES da cidade de Brasília, DF.

Em específico, a metodologia aplicada para a coleta dos dados da Tabela 1, que mostra a distribuição de alunos por anos e grupos então considerados nesta pesquisa, consistiu na compilação de informações provenientes de atas de bancas examinadoras de TCC, disponibilizadas pela coordenação de curso das duas IES pesquisadas.

Com relação aos dados da Figura 1, a metodologia de coleta consistiu na compilação de informações provenientes das trocas de mensagens eletrônicas entre os orientadores e seus respectivos grupos de orientandos, então utilizadas para o envio e o recebimento de artefatos dos projetos de TCC desenvolvidos no período considerado. Essa figura apresenta a evolução de remessa de arquivos aos orientadores, referentes aos artefatos de projeto construídos pelos grupos de alunos orientandos.

Tabela 1 – Distribuição de alunos por anos e grupos

Ano	Total de alunos	Total de grupos formados
2009	33	7
2010	26	7
2011	26	8
2012	16	5
2013	28	11
2014	11	4
	140	37

Fonte: do autor

Por fim, ainda como metodologia de coleta de dados, também foram realizadas entrevistas individuais com os orientadores dos TCCs das IES pesquisadas no período dos anos já assinalados, conforme Tabela 1. Essas entrevistas tiveram o objetivo de identificar os principais problemas enfrentados pelos grupos de alunos orientandos durante a elaboração dos TCCs. Esses dados são apresentados no Quadro 1 e estão agrupados em seis categorias, expondo inclusive suas prováveis causas e efeitos.

3.3 Análise de dados

A partir da Figura 1, se pode observar que, nos meses iniciais, pouco se constrói, deixando-se o esforço maior para o final do período. Também é possível perceber que muitos TCCs, dentro dos anos de execução, são desenvolvidos em ritmos diferentes pelos grupos de alunos. Isso é reflexo das dificuldades enfrentadas, seja pela carência de conhecimentos técnicos e/ou falta de gerenciamento de trabalhos em grupo (BARBARA et al., 2004), ou mesmo pelo baixo controle afetivo e emocional relacionados a seus integrantes, observado pelos orientadores de TCCs das IES pesquisadas, conforme indicado no Quadro 1.

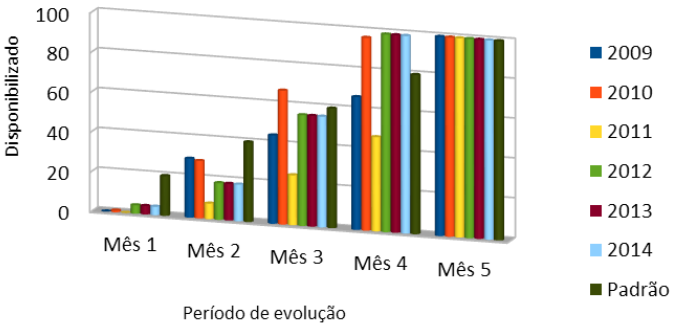
Evidências constatadas por Paula Filho (2006) e

também observadas pelos orientadores de TCCs das IES pesquisadas são que os alunos possuem pouca experiência e têm dificuldades em realizar uma coleta de dados eficiente para caracterização do problema e identificação de uma possível solução a ser proposta e executada, conforme indicado no Quadro 1. Muito tempo é gasto nessas atividades, podendo comprometer o cronograma de trabalho dos TCCs.

Os orientadores de TCCs das IES pesquisadas também percebem uma grande dificuldade dos alunos em definir uma estratégia eficiente, quanto ao uso do tempo e quanto à qualidade técnica e documental dos trabalhos, para implementar a solução então identificada, conforme indicado no Quadro 1.

Finalmente, em conclusão, é possível inferir-se que grande parte dos problemas está principalmente relacionada ao uso de MDS e à gestão de trabalho em grupo.

Figura 1 – Percentual do total de arquivos disponibilizados dos TCCs.



Fonte: do autor

Quadro 1 – Problemas identificados no desenvolvimento de TCC

Categorias	Causas	Efeitos
Artefatos	- A metodologia RUP, adotada nos projetos, é muito ampla e abrangente, apresentando uma série de artefatos para atenderem a projetos variados; - Os *templates* dos artefatos não são explicativos, por completo, cabendo aos alunos deduzirem alguns itens propostos.	- Dificuldade em elencar os artefatos que devem ser produzidos; - Falta de artefatos mínimos necessários para entendimento da solução desenvolvida; - Artefatos incompletos; - Diversidade de artefatos apresentados tornando os trabalhos muito heterogêneos entre si.
Processos	- Grande diversidade de processos em cada fase da metodologia RUP em virtude de sua grande abrangência, mesmo na versão *Small Projects*; - Existência de processos no RUP não adequados a um Projeto Final de Curso; - Planejamento não aderente ao executado pelo grupo.	- Dificuldade em elencar os principais processos que devem ser executados para atender a uma solução acadêmica; - Desalinhamento ao executar processos não adequados ao trabalho acadêmico; - Dificuldade em dividir tarefas (papeis) aos integrantes do grupo; - Alocação exagerada de tempo em atividade de programação e suprimindo destinado a outros artefatos importantes para construção da solução; - Deixar de abordar, nos trabalhos, aspectos técnicos relevantes, que abranjam as áreas de Engenharia de Software, Redes de Computadores, Banco de Dados, Programação e Gerência de Projetos.
Requisitos e Regras de Negócio	- Poucas informações coletadas; - Superficialidade na coleta de dados; - Falta de coleta de documentação de negócio para melhor entendimento do problema; - *Templates* muito extensos e/ou faltando incluir aspectos inerentes a um trabalho acadêmico	- Dificuldade na coleta de dados de forma eficiente para atender ao problema; - Falta de informações necessárias para mapear todo o negócio estudado; - Dificuldade no entendimento do negócio e identificação correta das necessidades a serem abrangidas na solução; - Especificação de UC de forma superficial, sem expressar a realidade do cliente; - Não identificação de regras de negócio, formais ou informais, praticadas pelo cliente.
Participação e Envolvimento	- Distribuição de papeis inadequados aos integrantes sem levar em consideração as competências individuais; - Pouca ou nenhuma reunião de coordenação dos trabalhos do grupo; - Falta de ajustes de papeis ao longo da execução do projeto; - Desmotivação do aluno por não atender a contento o esperado pelo grupo; - Pouca carga horária destinada, pelo aluno, ao desenvolvimento de suas tarefas do projeto; - Falta de tempo do aluno em virtude de seu trabalho diário.	- Falta de sinergia do grupo com esforços divergentes e não objetivos; - Não cumprimento do cronograma do projeto; - Não comprometimento com a qualidade do trabalho realizado.
Acompanhamento	- Não divulgação, por todos os integrantes, dos artefatos gerados para o projeto; - Falta de reuniões regulares para discussão sobre todos os artefatos gerados; - Falta de um ambiente para compartilhamento a todos os integrantes do grupo contendo todos os artefatos gerados.	- Integrantes dos grupos não conhecem todos os artefatos gerados no projeto; - Artefatos necessitando serem debatidos pelo grupo para seu refinamento e melhoria na qualidade.
Definição e distribuição de Papéis	- Grande diversidade de papeis da metodologia RUP que muitas vezes são dispensáveis para um trabalho acadêmico; - Não identificação de todas as competências individuais dos integrantes dos grupos.	- Dificuldade de identificação dos papeis necessários para a execução do projeto; - Má distribuição de papeis pelos integrantes dos grupos, não aproveitando as competências individuais.

Fonte: do autor

4 Metodologias de desenvolvimento de software

Nesta seção são analisadas três metodologias que constituem a base de conhecimento para se apontar uma possível customização: Processo Unificado da *Rational* (RUP), *Extreme Programming* (XP) e Scrum. A razão que leva à escolha dessas metodologias é o fato de serem as mais conhecidas e praticadas no mercado (SMITH, 2001; ALFONSO; BOTÍA, 2005; SHARMA et al., 2012; HANSSEN et al., 2012).

4.1 Processo Unificado da Rational

O RUP (RUP 2007) possui duas dimensões distintas: quatro fases, como aspectos dinâmicos, e nove disciplinas (SMITH, 2001), que representam os aspectos estáticos do processo. Suas características essenciais são: 1) direcionado a casos de uso – o elemento chave que orienta a construção do sistema e a unidade de gestão do projeto de *software*; 2) focado na arquitetura – preocupação, desde a fase inicial, em definir a arquitetura de *software* a ser adotada para minimizar riscos, seja de desempenho ou viabilidade de sua construção; 3) ser interativo e incremental – o processo é cíclico, isto é, adotadas várias iterações, onde a cada iteração novas funcionalidades são disponibilizadas ao cliente.

4.1.1 Pontos fortes

Na maioria dos projetos em RUP, a duração de uma iteração ocorre entre duas e seis semanas. O número de interações de um projeto está entre três e nove. Essa configuração padrão do RUP abrange projetos que podem durar de seis semanas a 54 meses (SMITH, 2001; RUP, 2007). Nesse aspecto, projetos acadêmicos de TCC têm duração de 16 a 20 semanas, portanto, compatíveis com essa configuração.

Na fase de elaboração, a arquitetura da solução deve estar estabilizada. Nesse aspecto, apesar de ser extremamente importante para o produto de *software*, não é o fator crucial para projetos acadêmicos uma vez que são de características mais simples, tanto em tamanho como em complexidade.

O RUP incentiva a adaptação das atividades e artefatos para atender às peculiaridades de projetos e equipes de desenvolvimento, mantendo-se as melhores práticas para desenvolvimento de *software*. Possui, portanto, customização nativa para atender a projetos pequenos (aqueles que envolvem equipes de três a dez pessoas e que tenham duração menor que um ano), como é o caso dos TCCs. Embora, com grande nível de detalhamento, apresenta orientações de como se adaptar o processo, inclusive alterando suas características com a inclusão de outras disciplinas e/ou fases, se for conveniente.

4.1.2 Pontos desfavoráveis

Por ser uma metodologia bastante abrangente e detalhada pode resultar em uma maior dificuldade ao se identificar os pontos-chave de interesse da metodologia a ser customizada, consumindo maior tempo que outras metodologias mais *enxutas*.

4.2 Extreme Programming

O *Extreme Programming* (XP) é uma metodologia ágil usada por pequenas e médias equipes que desenvolvem *software* com requisitos vagos e em constante mudança. Adota uma estratégia de acompanhamento contínuo e a realização de vários pequenos ajustes durante a execução do processo de desenvolvimento (SOARES, 2004; LACERDA et al., 2011; COSTA FILHO et al., 2005).

É composta de cinco valores fundamentais: comunicação, simplicidade, *feedback*, coragem e respeito. A partir desses valores, emprega como princípios básicos: realizar rápidos feedbacks, presumir simplicidade, realizar mudanças incrementais, abraçar mudanças, e desenvolver trabalho de qualidade (SOARES, 2004; LACERDA et al., 2011).

Para o XP, dentre as variáveis de controle em projetos (custo, tempo, qualidade e escopo), há um foco explícito em escopo. Para isso, recomenda-se a priorização de funcionalidades de maior valor ao negócio. Caso seja necessária a diminuição de escopo, as funcionalidades menos valiosas serão adiadas ou canceladas (SOARES, 2004; LACERDA et al., 2011; COSTA FILHO et al., 2005).

Além disso, adotam-se quatro atividades básicas que são executadas com uso de um conjunto de práticas que estão presentes nas disciplinas do RUP: codificação, teste, monitoramento e design. Por fim, as práticas utilizadas pelo XP (SOARES, 2004) são: 1) jogo de planejamento – desenvolvedores e cliente se reúnem para priorizarem as funcionalidades; 2) pequenos releases – liberação de pequenas versões funcionais que auxilia na aceitação pelo cliente; 3) metáfora – facilidade na comunicação com o cliente; 4) design simples – adoção de uma solução simples para cada parte do código; 5) reuniões em pé (*stand-up meetings*) – rápidas reuniões focadas nas tarefas realizadas e a serem realizadas; 6) teste – continuo

e sem prejuízo no prosseguimento do desenvolvimento; 7) refatoração – redução de erros e manutenção da compatibilidade do código já existente, evitando-se sua duplicação; 8) programação em pares – programa construído por duas pessoas, num único computador; 9) propriedade coletiva – código fonte não tem dono, podendo ser modificado livremente. Isto permite à equipe conhecer todo o sistema; 10) integração contínua – integra e cria o sistema muitas vezes sempre que uma tarefa for concluída; 11) Semana de 40 horas; 12) cliente no local de desenvolvimento; 13) padrões de codificação – código escrito de acordo com regras que enfatizem a comunicação.

4.2.1 Pontos fortes

O emprego de pequenos releases pode ser uma maneira mais simples e eficiente para os acadêmicos desenvolverem seus trabalhos, pois permite que eventuais falhas sejam detectadas mais precocemente.

O tempo de duração das iterações do XP, cerca de duas semanas, e as suas liberações, cerca de dois meses (SMITH, 2001), parece adequar-se perfeitamente a trabalhos acadêmicos (TCC).

O emprego do jogo de planejamento assegura que a equipe trabalhe no mais importante a cada momento do projeto, evitando esforços em funcionalidades secundárias (SOARES, 2004). Neste contexto, a peculiaridade acadêmica nos cenários estudados visa a uma implementação mínima de três casos de uso ou funcionalidades representativas do sistema a ser desenvolvido.

4.2.2 Pontos desfavoráveis

Processos ágeis parecem ser melhores aplicados quando existem equipes com integrantes que possuem níveis altos de habilidades (SOMMERVILLE, 2011), o que nem sempre ocorre com o público discente.

Como o escopo é reavaliado semanalmente, isso perece ser prejudicial em um contexto acadêmico, pois o tempo disponível, tanto para desenvolvimento como para planejamento com o cliente, é relativamente curto.

Em face da restrição de tempo para a realização do TCC, a variação constante do escopo pode trazer dificuldades para a documentação (SOARES, 2004), requisito importante para materializar trabalhos acadêmicos e possibilitar sua avaliação de forma objetiva.

Pela observação prática em realização de TCC, há dificuldade dos discentes em manterem o cliente sempre presente como sugere a metodologia. Apesar do efeito positivo dessa parceria, isso nem sempre é possível de se realizar e podem surgir situações que inviabilizem a introdução desta prática (BOEHM; TURNER, 2005).

Por fim, a prática de refatoração pode demandar um tempo valioso para a equipe, se o trabalho não for bem coordenado. Isso provavelmente demandará maior atenção do grupo de acadêmicos.

4.3 Scrum

O Scrum é um *framework* baseado em metodologia ágil que funciona de maneira iterativa e incremental para o gerenciamento de projetos (RESOURCES, 2012). Tem por objetivo entregar versões incrementais do produto potencialmente utilizáveis em um determinado espaço de tempo (CAJU et al., 2007), e encontrar uma forma de trabalho para a equipe produzir um *software* de forma flexível em um ambiente de constante mudança (SOARES, 2004).

Contém grupos de práticas e papéis pré-definidos, entre eles: 1) Scrum Master – responsável por manter os processos; 2) Product Owner – representa o *stakeholder* e o negócio; 3) Scrum Team – equipe multifuncional responsável pela análise, projeto, implementação, teste, dentre outras atividades.

No Scrum, a sprint é a unidade básica de desenvolvimento (CAJU et al., 2007). Sprints tendem a durar entre uma semana e um mês e são precedidas por reuniões de planejamento continuado, onde as tarefas são identificadas e um objetivo é definido. Após cada sprint, ocorre uma revisão ou retrospectiva, onde o progresso é revisado e lições, para as próximas sprints, são identificadas. A equipe deve produzir, ao seu final, um incremento de produto funcional devidamente testado.

O Scrum permite a criação de equipes auto-organizadas que encorajam em seus membros à comunicação verbal entre todos (CAJU et al., 2007). Os clientes são livres para mudarem os requisitos, o que tornam os desafios imprevisíveis e dificultam o planejamento tradicional. Foca sempre na maximização da habilidade da equipe em entregar rapidamente e responder às necessidades emergentes (CAJU et al., 2007; SOARES, 2004).

A seguir, listam-se alguns aspectos adotados em metodologia ágeis como o Scrum (PATHAK et al., 2012): 1) receptividade quanto às mudanças de requisitos; 2) satisfação dos clientes; 3) simplicidade; 4) a equipe deve trabalhar junta durante o projeto; 5) comunicação face à face; 6) motivação individual; 7) auto-organização das equipes; 8) o cliente se torna parte da equipe; 9) cada sprint entrega uma funcionalidade 100% desenvolvida;

10) problemas não são ignorados; 11) velocidade; 12) o backlog (lista de requisitos da sprint) torna o projeto mais claro para todos os membros da equipe; e 13) reduz os custos de manutenção, a longo prazo.

4.3.1 Pontos fortes

Apesar de o Scrum estar centrado na gerência de projetos, é importante considerá-lo neste estudo, pois alguns problemas relatados na Seção 3 retratam esta realidade.

O emprego do conceito de sprint é adequado à realidade acadêmica, uma vez que se trata de uma prática rápida e de resultados concretos. Além disso, a estrutura de equipe adotada (Scrum Team) de forma reduzida é aderente a trabalhos acadêmicos de TCC, que são realizados por um número pequeno de discentes, conforme já descrito anteriormente.

4.3.2 Pontos desfavoráveis

No Scrum, como os requisitos são passíveis de alterações constantes, acredita-se que tal abordagem pode não ser adequada a uma realidade acadêmica na elaboração de um TCC, em face das restrições de tempo já mencionadas. Não há uma delimitação de escopo bem definida, o que pode comprometer o entendimento dos discentes para efeito de planejamento das etapas a serem realizadas. Como prevê a existência de poucos papéis, subentende-se a exigência de uma maior multidisciplinaridade e experiência de seus integrantes, o que não retrata a realidade do público discente.

5 Proposta de customização

De forma geral, propõe-se o RUP como metodologia base para customização pretendida. A estrutura proposta visa a solucionar os problemas identificados na seção 3. A definição da documentação sugerida leva em consideração as características dos ambientes estudados.

Destaca-se que um mínimo de documentação é necessário ser produzido para o desenvolvimento de *software* (PATIL; YOGI, 2011), e é isto que se espera de um trabalho acadêmico de final de curso.

5.1 Solução por meio da metodologia

O Quadro 2 apresenta sumariamente os aspectos favoráveis que foram identificados dentre as metodologias estudadas e que são indícios para solução dos problemas apresentados no Quadro 1.

Quadro 2 – Possível solução, pela metodologia, aos problemas identificados

Problema	Metodologias envolvidas		
	RUP	XP	Scrum
Artefatos	Prevê muitos artefatos (selecionar os essenciais para um contexto acadêmico).	Prevê poucos artefatos.	Prevê poucos artefatos: - *Product Backlog*; - *Sprint backlog*; - Planejamento de *sprint*
Processos	Prevê variados processos nas disciplinas, distribuídas entre as fases e envolvendo vários papéis (selecionar os essenciais para um contexto acadêmico)	Emprego de small releases: execução de uma funcionalidade de cada vez previne falhas detectadas precocemente	Planejamento de Sprints: prática rápida e de resultados concretos.
Requisitos e Regras de Negócio	Fornece melhores práticas e dá ênfase ao fluxo de trabalho	Aplica conceitos como metáfora para facilitar a comunicação com o cliente	
Participação e Envolvimento		- Emprego do "jogo de planejamento" centra o foco no que é mais importante - Emprego de "Propriedade Coletiva" onde todos podem acessar e alterar códigos	Reuniões do Scrum motivam a participação dos envolvidos
Acompanhamento	Contempla vários processos para gerenciamento do projeto	- Reuniões em pé: com foco nas tarefas realizadas e a serem realizadas	Reuniões do Scrum podem: - favorecer o desempenho quanto à coordenação dos trabalhos; - proporcionar motivação ao grupo no acompanhamento dos resultados.
Definição e distribuição de Papeis	Preveem variados papéis. Devem-se selecionar os mais importantes.		- Adoção do Scrum Team como equipe reduzida. - Permite ajuste de papeis

Fonte: do autor

5.2 Estrutura proposta

Depois de considerados os aspectos vantajosos de cada metodologia, busca-se então definir uma estrutura base para customização. Assim, as subseções seguintes retratam quais elementos podem compor a metodologia customizada.

5.2.1 Fases e processos

Manter as quatro fases existentes no RUP, com uma iteração cada, por se tratar de projetos pequenos, e dispostos conforme apresentado na Tabela 2.

Tabela 2 – Estrutura de fases do método adaptado

Fases		Duração (em semanas)	
Nr	Descrição	TCC com 16 semanas	TCC com 20 semanas
1	Iniciação	4	5
2	Elaboração	7	9
3	Construção	3	4
4	Transição	2	2

Fonte: do autor

Em cada fase devem ser considerados, como meta mínima, os objetivos apresentados no Quadro 3 para adequar-se às características inerentes aos projetos acadêmicos de final de curso.

Quadro 3 – Objetivos das fases para a metodologia customizada

Fases	Objetivos a serem considerados para adaptação
Iniciação	a) Definir a equipe do projeto e artefatos iniciais de gestão de projeto; b) Conhecer os processos de negócio que envolvem o sistema; c) Levantar os requisitos iniciais do sistema; d) Estabelecer o escopo do software; e) Identificar e avaliar os riscos; e f) Preparar o ambiente de suporte para o projeto.
Elaboração	a) Estabelecer e detalhar a arquitetura do sistema; b) Especificar requisitos de software; c) Especificar os critérios de aceitação do produto; d) Detalhar os custos para o projeto; e) Definir e discriminar os casos de uso do sistema a serem implementados; f) Elaborar o projeto de software (Classes); g) Elaborar o projeto de banco de dados e de infraestrutura; e h) Definir interfaces com o usuário.
Construção	a) Desenvolver a codificação dos casos de uso definidos para implementação; b) Criar protótipo incluindo as interfaces de UC que não serão implementados; c) Descrever a implementação dos testes; d) Realizar os testes de todas as funcionalidades implementadas; e e) Descrever os demais casos de uso.
Transição	a) Desenvolver um Plano de Implantação; b) Desenvolver Material para suporte ao usuário (treinamento); e c) Encerrar o Projeto.

Fonte: do autor

Os processos devem estar inseridos nas fases e distribuídos nas disciplinas a serem consideradas, conforme apresentado no Quadro 4. O objetivo não é detalhar os processos em si, mas apontar as ações principais que conduzam a uma solução acadêmica para os TCCs.

Quadro 4 – Processos selecionados para a metodologia customizada

Disciplinas	Processos e suas Fases			
	Iniciação	Elaboração	Construção	Transição
Modelagem de Negócios	- Avaliar organização-alvo - Identificar processos de negócio	- Definir requisitos - Projetar processo de negócio		
Requisitos	- Desenvolver visão - Identificar necessidades - Formular escopo	- Especificar Requisitos - Detalhar casos de uso		
Análise e Design	- Definir funcionalidades	- Sintetizar arquitetura - Estruturar modelo de UC - Projetar design - Analisar casos de uso - Construir diagrama de classes - Projetar banco de dados		
Implementação e Teste		- Modelar interface - Criar modelo de implementação	- Implementar componentes - Estruturar testes - Revisar código - Realizar testes	- Desenvolver material de suporte - Criar plano de Implantação
Gerenciamento de Projeto	- Definir equipe de projeto - Definir plano de projeto; - Identificar e avaliar riscos - Preparar ambiente - Monitorar projeto - Definir tarefas	- Monitorar projeto - Desenvolver Plano de Aceitação do Produto - Definir tarefa	- Definir tarefas - Monitorar projeto	- Definir tarefas - Monitorar projeto

Fonte: do autor

5.2.2 Disciplinas e artefatos

Opta-se por realizar uma fusão de algumas disciplinas do RUP em virtude de apresentarem, para esta customização, poucos processos e assim tornar a metodologia mais simplificada e objetiva. Para ser aderente aos trabalhos de um TCC, verifica-se, pela sua importância, que a metodologia adaptada pode utilizar as disciplinas constantes no Quadro 5, que mostra os artefatos sugeridos para construção nas fases, e dentro de cada disciplina.

Quadro 5 – Fases, disciplinas e artefatos para a metodologia customizada

Disciplinas	Artefatos e suas Fases			
	Iniciação	Elaboração	Construção	Transição
Modelagem de Negócios	- Glossário - Processo de Negócio			
Requisitos	- Documento de visão	- Especificação Suplementar - Descrição de Casos de Uso		
Análise e Design	- Requisitos funcionais	- Modelos de Casos de Uso - Arquitetura de Software - Diagrama de Atividade - Diagrama de Sequência - Diagrama de Classe - Projeto de Banco de Dados		
Implementação e Teste		- Interfaces do sistema	- Codificação - Plano de Teste	- Manual Usuário - Plano Implantação
Gerenciamento de Projeto	- Papéis e Responsabilidades - Cronograma - Plano de Comunicação - Estrutura Analítica - Dicionário da EAP - Análise de Riscos - Custos e Aquisições	- Plano de Aceitação		

Fonte: do autor

5.2.3 Papéis

Sugere-se a adoção de quatro papéis (Quadro 6) a serem considerados para a metodologia customizada, de forma a atender ao número provável de integrantes de grupos de um TCC. São associados outros papéis que podem ser abrangidos e sua adoção depende de como é estruturado o trabalho pelos grupos.

Quadro 6 – Papéis a serem desempenhados

Papéis	Papéis associados
Analista	Analista de Sistemas
	Analista do Processos de Negócio
	Especificador de Requisitos
Projetista	Projetista de Banco de Dados
	Projetista de Infraestrutura
	Projetista de *Software*
Desenvolvedor	Implementador/Programador
	Testador
Gerente	Gerente de Projeto/*Scrum Master*

Fonte: do autor.

5.2.4 Considerações sobre a proposta

Como vantagens da sua aplicação, destaca-se que a mesma é bem simples, abrange as boas práticas de desenvolvimento e, assim, é bem viável para a execução de TCCs, que normalmente são de baixa complexidade e escopo reduzido.

É exequível para grupos de dois a cinco alunos integrantes dos projetos, sem prejuízo da qualidade dos produtos a serem apresentados, e é flexível o suficiente para possibilitar, aos alunos, executarem os principais artefatos de um PDS no tempo destinado.

Assim, tanto professores como alunos sabem o contexto a ser avaliado nos trabalhos e o que deve ser produzido em seus projetos. Este conhecimento pode ser potencializado se praticado nas disciplinas de ES do curso, antes mesmo de se chegar à etapa final da graduação.

Sua adoção proporciona uma padronização do curso de Sistema de Informações para a construção de TCCs e permite aos professores ter um parâmetro mais homogêneo para avaliação dos trabalhos.

Por fim, sua implementação deve contemplar as diversas fases, disciplinas, processos, papéis e artefatos, preferencialmente de forma visual, para que seja de fácil acesso em ambiente web, contendo templates dos artefatos componentes para fins de facilidade de utilização.

6 Conclusão e trabalhos futuros

Este artigo teve o objetivo de realizar uma proposta de customização de metodologia para o desenvolvimento de sistemas. A fundamentação desta proposta está na identificação das principais dificuldades de discentes de graduação ao realizarem seus trabalhos de conclusão de curso, em Instituições de Ensino Superior, bem como na análise de metodologias de desenvolvimento já consagradas da literatura. Como trabalho futuro, sugere-se a implementação e utilização da proposta aqui apresentada em ambientes acadêmicos para fins de validação prática final.

Referências

ALFONSO, M. I.; BOTÍA, A. An interative and agile process model for teaching software engineering. In: CONFERENCE ON SOFTWARE ENGINEERING EDUCATION e TRAINING, 18., 2005, Otawa. *Proceedings*… Washington: IEEE, 2005. p. 9-16.

AL-HAGERY, M. A. H. Problems discovery of final graduation projects during the software development processes. *International Journal of Computer Applications*, New York, v. 37, n. 5, p. 19-24, Jan. 2012.

BARBARA, O.; FELDER, R. M.; REBECCA, B.; IMAD, E. Turning student groups into effective teams. *Journal of Student Centered Learning*, New Orleans, v. 2, n. 1, p. 9-34, 2004.

BOEHM, B.; TURNER, R. Management Challenges to Implementing Agile Processes in Traditional Development Organizations. *Software*, Los Angeles, v. 22, n. 5, p. 30-39, Sept./Oct. 2005.

CAJU, P. P.; COELHO, P. T.; MARÇAL, A. S. *Entendendo Scrum para gerenciar projetos de forma ágil*. Disponível em: <http://www.siq.com.br/DOCS/EntendendoScrumparaGerenciarProjetosdeFormaAgil.pdf>. Acesso em: 06 jan. 2013.

COSTA FILHO, E. G. et al. Padrões e métodos ágeis: agilidade no processo de desenvolvimento de software. In: LATIN AMERICAN CONFERENCE ON PATTERN LANGUAGES OF PROGRAMS. 5., 2005. Campos do Jordão. *Proceedings…* São Paulo: USP, 2005. v. 1. p. 172-185.

DYCK, S.; MAJCHRZAK, T. A. Identifying common characteristics in fundamental, integrated, and agile software development methodologies. In: HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES. 45., 2012, Maui. *Proceedings*… Washington: IEEE, 2012. p. 5299-5308.

HANSSEN, G. K.; WESTERHEIM, H.; BJØRNSON, F. O. Tailoring RUP to a defined project type: a case study. In: INTERNATIONAL CONFERENCE PROFES, 6., 2005, Oulu. *Proceedings...* Heidelberg: Springer-Verlag Berlin Heidelberg, 2015. v. 3547. p. 314-327.

LACERDA, G. S.; BARBOSA, A. B.; RIBEIRO, V. G. Adoção do CMMI e das metodologias ágeis em empresas brasileiras. *Revista Avances en Sistemas e Informática*, Medellín, v. 8, n. 3, p. 33-42, dez. 2011.

LANGLEY, D; RONEN, M. Responding to the employability challenge: final projects for it-based organizational training. *Informing Science and Information Technology*, California, v. 8, p. 125-142, jan. 2011.

PATEL, C. et al. A framework for method tailoring: a case study. In: OOPSLA WORKSHOP ON METHOD ENGINEERING FOR OBJECT-ORIENTED AND COMPONENT-BASED DEVELOPMENT, 2., 2004, Vancouver. *Proceedings…*Sydney: COTAR, 2004. p. 23-37.

PATHAK, S.; PATERIYA, P.; PAL, P. A case study on software development projects in academic knowledge centers using SCRUM. *International Journal of Computer Applications*, New York, v. 43, n. 10, p. 20-24, Apr. 2012.

PATIL, M. V.; YOGI, A. M. N. Importance of data collection and validation for systematic software development process. *International Journal of Computer Science and Information Technology*, EUA, v. 3, n. 2, p. 260-278, Apr. 2011.

PAULA FILHO, W. P. A software process for time-constrained course projects. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 28., 2006, Shanghai. *Proceedings…* New York: ACM, 2006. p. 707-710.

RESOURCES to broaden your knowledge of Scrum e Agility. Available in: <http://www.scrum.org/Scrum-Guides>. Accessed: 13 Dec. 2012.

RUP 2007: Rational Unified Process. Available: <http://www.wthreex.com/rup/portugues/index.htm>. Accessed: 16 Set. 2012.

SEYAM, M. S.; GALAL-EDEEN, G. H. Traditional versus agile: the tragile framework for information systems development. *International Journal of Software Engineering*, Cairo, v. 4, n. 1, p. 63-93, Jan. 2011.

SHARMA, A. K.; SHARMA, S. A.; MEHTA, I. C. A comparative analysis of software process models. *International Journal of Computer Applications,* New York, v. 82, n. 18, p. 16-19, Nov. 2013.

SMITH, J. *A Comparison of RUP and XP*. California: Rational Software, 2001. Available in: <http://citeseerx.ist.psu.edu/viewdoc/download?-doi=10.1.1.136.4963&rep=rep1&type=pdf>. Accessed: 28 Oct. 2015.

SOARES, M. S. Metodologias ágeis Extreme Programming e Scrum para o desenvolvimento de software. *Revista Eletrônica de Sistema de Informação*, v. 3, n. 1, p. 1-8, 2004. Disponível em: <http://189.16.45.2/ojs/index.php/reinfo/article/view/146/38 >. Acesso em: 28 out. 2015.

SOMMERVILLE, I. *Software engineering*. 9. ed. São Paulo: Pearson, 2011.