UNIVERSIDADE FEDERAL DO PARÁ

INSTITUTO DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

RAFAEL OLIVEIRA CHAVES

UM JOGO SÉRIO PARA APOIO AO ENSINO DE MODELAGEM DE PROCESSO DE SOFTWARE BASEADO EM MAPAS CONCEITUAIS: UMA AVALIAÇÃO EXPERIMENTAL

TD 13/2015

BELÉM

2015

RAFAEL OLIVEIRA CHAVES

UM JOGO SÉRIO PARA APOIO AO ENSINO DE MODELAGEM DE PROCESSO DE SOFTWARE BASEADO EM MAPAS CONCEITUAIS: UMA AVALIAÇÃO EXPERIMENTAL

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, do Instituto de Tecnologia, da Universidade Federal do Pará, como requisito parcial à obtenção do grau de Doutor em Engenharia Elétrica.

Orientador: Prof. Dr. Eloi Luiz Favero

BELÉM
2015

Dados Internacionais de Catalogação-na-Publicação (CIP)
Sistema de Bibliotecas da UFPA

Chaves, Rafael Oliveira, 1980-

Um jogo sério para apoio ao ensino de modelagem de processo de software baseado em mapas conceituais: uma avaliação experimental / Rafael Oliveira Chaves. - 2015.

Orientador: Eloi Luiz Favero.

Tese (Doutorado) - Universidade Federal do Pará, Instituto de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Belém, 2015.

1. Engenharia de software auxiliado por computador - jogos. 2. Jogos educativos. 3. Inovações educacionais - projetos. I. Título.

CDD 22. ed. 005.1

RAFAEL OLIVEIRA CHAVES

UM JOGO SÉRIO PARA APOIO AO ENSINO DE MODELAGEM DE PROCESSO DE SOFTWARE BASEADO EM MAPAS CONCEITUAIS: UMA AVALIAÇÃO EXPERIMENTAL

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, do Instituto de Tecnologia, da Universidade Federal do Pará, como requisito parcial à obtenção do grau de Doutor em Engenharia Elétrica.

Data: ___/____/_____

Banca Examinadora

_____
Prof. Dr. Eloi Luiz Favero (Orientador)
Faculdade de Ciências da Computação
Universidade Federal do Pará

_____
Prof. Dr .Sandro Ronaldo Bezerra Oliveira
Faculdade de Ciências da Computação
Universidade Federal do Pará

_____
Prof. Dr. Eduardo Coelho Cerqueira
Faculdade de Engenharia da Computação e Telecomunicações
Universidade Federal do Pará

_____
Prof. Dr. Manoel Ribeiro Filho
Instituto de Geociências e Engenharia
Universidade Federal do Sul e Sudeste do Pará

_____
Profa. Dra. Andrea Silva Miranda
Instituto Ciberespacial
Universidade Federal Rural do Pará

_____
Profa. Dra Christiane Gresse von Wangenheim
Departamento de Informática e Estatística
Universidade Federal de Santa Catarina

RESUMO

A Modelagem de Processo de Software (MPS) é um importante tópico na engenharia de software, pois fornece subsídios para gerenciar, automatizar e apoiar a Melhoria de Processo de Software. Seu ensino é um desafio, principalmente em consequência da grande ênfase teórica e de poucos exercícios práticos. Além disso, ainda são poucas as abordagens de ensino que visam o MPS por meio da introdução de recursos inovadores, tais como jogos. O uso de jogos é principalmente focado em outras áreas da engenharia de software, por exemplo, gerência de projetos de software. Visando contemplar esta lacuna, este trabalho relata o desenvolvimento de um jogo (DesigMPS – que apoia o ensino da MPS) e um experimento formal para avaliar a sua eficácia no ensino de MPS, bem como compará-lo com o método instrucional de aprendizagem baseada em projetos (ABP) (*project-based learning*). Neste jogo, o aluno é desafiado a modelar um processo de software no contexto do Modelo de Melhoria de Processos do Software Brasileiro (MPS.BR). Também foram analisadas características como: adequação em relação ao conteúdo, suficiência, grau de dificuldade, sequência e método de ensino; atratividade; pontos fortes e fracos do jogo. Os resultados indicaram que o DesigMPS tem eficácia de aprendizagem positiva (diferença estatisticamente significativa) nos níveis de compreensão e aplicação na taxonomia revisada de objetivos educacionais de Bloom, mesmo havendo alguns pontos fracos como: a falta um de *help* sobre o uso de jogo e de um *feedback* instrucional mais claro com explicação das causas dos erros. O jogo também tem uma eficácia de aprendizagem positiva (diferença estatisticamente significativa) comparada ao ABP no nível de aplicação, mas igual no nível de compreensão. A maioria dos alunos considerou que o jogo é atrativo e relevante em respeito ao conteúdo da disciplina de Qualidade de Processos de Software, e assim pode ser considerado uma alternativa para ensinar MPS.

Palavras-chave: Aprendizagem Baseada em Projetos. Jogos Sérios. Educação em Engenharia de Software. Modelagem de Processo de Software.

ABSTRACT

Software process modeling (SPM) is an important area of software engineering because it provides a basis for managing, automating, and supporting software process improvement (SPI). Teaching SPM is a challenging task, mainly because it lays great emphasis on theory and offers few practical exercises. Furthermore, as yet few teaching approaches have aimed at teaching SPM by introducing innovative features, such as games. The use of games has mainly been focused on other areas of software engineering, for example software project management. In an attempt to fill this gap, this paper describes a formal experiment carried out to assess the learning effectiveness of a serious game (DesigMPS), designed to support the teaching of SPM, and to compare game-based learning with a project-based learning method. In the DesigMPS game, the student models a software process from an SPI perspective, based on the Brazilian SPI model (MPS.BR). Also features of the game were analyzed as: appropriateness in terms of content relevance, sufficiency, degree of difficulty, sequence, teaching method; engagement; strengths and weakness. The results indicate that playing the game can have a positive learning effectiveness (statistically significant difference) on the cognitive levels of understanding and applying, according to the revised version of Bloom's taxonomy of educational objectives, despite some weaknesses as: the lack of “help” on the use of the game and the lack of clearer instructional feedback with explanations regarding the causes of errors. Also, there is positive learning effectiveness (statistically significant difference) when compared to the PBL on the applying level, but no significant difference on the understanding level. Most students considered that the game is engaging and relevant concerning the content of Software Quality course. Based on the results the game could be an option to teach SPM.

Keywords: Games Project-Based Learning. Serious Game. Software Engineering Education. Software Process Modeling.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

ABP -	Aprendizagem Baseada em Projetos
CMMI -	Capability Maturity Model Integration
CMMI-DEV -	Capability Maturity Model Integration for Development
DF -	Fator de Confusão
ES -	Engenharia de *Software*
GQA -	Garantia da Qualidade
GQM -	Goal, Question, Metric
ITEC -	Instituto de Tecnologia
LMPS -	Linguagem de Modelagem de Processos de Software
MC -	Mapa Conceitual
MCMPS -	Modelo de Capacidade/Maturidade de Processos de Software
MPS -	Modelagem de Processos de Software
MPS.BR -	Modelo de Melhoria de Processos do Software Brasileiro
MR-MPS-SW -	Modelo de Referência MPS para Software
SEI -	*Software Engineering Institute*
SOFTEX -	Associação para Promoção da Excelência do Software Brasileiro
SPEM -	Software Process Engineering Meta-Model
SWEBOK -	Software Engineering Body of Knowledge
UFPA -	Universidade Federal do Pará
UML -	Unified Modeling Language
VAL -	Validação
VER -	Verificação

SUMÁRIO

1. INTRODUÇÃO

1.1 Contextualização e motivação

A modelagem de processos é uma atividade chave para a melhoria do processo de *software*, auxiliando na identificação de deficiências e estimativas de sua execução (BECKER-KORNSTAEDT, 2000). Entre os seus principais objetivos estão ajudar a conhecer, controlar e gerenciar as atividades de um processo de software (ACUÑA; JURISTO, 2004). Dentro dessa perspectiva, Rai e Al-Hindi (2000), Krishnan et al. (2000) e Armenise et al. (1993) sugerem ainda que existe uma relação positiva entre a Modelagem de Processos de Software (MPS) e a maior qualidade do produto e do processo de *software*. A partir disso, a MPS tem sido associada com melhores produtos de software e com a qualidade do processo de software, ambos estes que dependem consideravelmente da maturidade do processo de software (GARCIA-BORGOÑON et al. 2013).

Apesar da importância do conhecimento da MPS, foi encontrado que os profissionais têm aprendido mais sobre processos de *software* em suas atividades de trabalho do que nos seus cursos universitários/educação (LETHBRIDGE, 2000). Isso pode ser porque, no momento, a MPS geralmente representa apenas uma pequena parte do que é ensinado nos cursos de graduação em computação (ACM/AIS/IEEE-CS, 2005), ou porque a maioria dos currículos universitários considera processo de *software* como sendo um conhecimento marginal da Engenharia de *Software* (ES) (KUHRMANN et al. 2013). Também, é evidenciada a relevância do MPS no Curriculum IEEE-CS/ACM (IEEE-CS/ACM, 2004) quando recomenda que a área de conhecimento sobre processo de software deve fazer parte do ensino de graduações em ES, sendo que um dos objetivos de aprendizagem nesta área do conhecimento é a modelagem e especificação de processos de *software*.

As disciplinas que relataram o ensino de MPS no ambiente universitário (JACCHERI, 1997; GROTH; ROBERTSON, 2001; HAWKER, 2009; WANGENHEIM; HAUCK, 2010) normalmente usam a instrução direta, por meio de aulas expositivas, para a apresentação de conceitos iniciais sobre MPS. Essas disciplinas citadas têm como principal estratégia a instrução indireta, por meio de projetos acadêmicos. Outras estratégias instrucionais com menor uso nas disciplinas são: aprendizagem experiencial, por meio de simulação, mas sem apoio computadorizado; e instrução interativa por meio de aprendizagem em pares. Não se observou a utilização de jogos como recurso de uma estratégia instrucional para ensino de MPS.

Algumas disciplinas de ES dispõem de jogos para apoiar o seu ensino, a exemplo de gerência de engenharia de software, processos de engenharia de software, requisitos de software, projeto de software, construção de software (CAULFIELD; VEAL; MAJ, 2011). Todavia, inexistem jogos específicos com o objetivo de ensinar MPS, fato identificado em revisões sistemáticas da literatura sobre jogos para ensino de ES (WANGENHEIM; SHULL, 2009; CONNOLLY; STANFIELD; HAINEY, 2007; CAULFIELD, 2011).

Embora existam poucas evidências de que jogos sérios são eficazes no ensino de ES, quando comparado com outros métodos de ensino (WANGENHEIM; SHULL, 2009), existem algumas vantagens significativas para uso de jogos sérios:

- Podem tornar o aprendizado mais divertido (KAFAI, 2001), por permitir o "aprender fazendo" em situações realistas, produzindo resultados positivos em termos de aquisição de conhecimento e compreensão dos conteúdos (CONNOLLY et al. 2012).
- Possibilitam aos alunos trabalhar em seu próprio ritmo, requerendo pouca ou nenhuma presença de um instrutor, nem interação com seus pares (WANGENHEIM; THIRY; KOCHANSKI, 2009).
- Oferecem atividades que estão em conformidade com modernas teorias de aprendizagem, estas que sugerem que a aprendizagem é mais eficaz quando é empírica, baseada em problemas, e fornece um *feedback* imediato (BOYLE; CONNOLLY; HAINEY, 2011).

Em virtude da reconhecida importância da MPS para o processo de software, e de que o seu aprendizado está ocorrendo tardiamente no ambiente profissional, verificou-se a possibilidade de melhorar o seu ensino na graduação por meio de jogos sérios, pois estes possuem vantagens significativas quando comparados com outros métodos de ensino. Esse tipo de jogo já é aplicado em outras áreas de ES, mas não estão disponíveis para ensino de MPS. Assim, o problema dessa pesquisa é: jogos sérios podem melhorar o aprendizado sobre MPS nos cursos de graduação?

Uma vez estabelecido problema de pesquisa, um jogo chamado DesigMPS foi desenvolvido com o intuito de apoiar o ensino de MPS - em que o aluno é desafiado a modelar processo de software no contexto de modelos de capacidade/maturidade de processos de software (MCMPS). Este jogo se baseia em trabalhos de avaliação adaptativa usando Mapas Conceituais (MC).

1.2 Hipótese

Em virtude da pequena quantidade de abordagens de ensino que visam a MPS de forma inovadora, e também para tentar diminuir a lacuna existente entre a grande ênfase teórica e os poucos exercícios práticos, esta tese tem a seguinte hipótese: *Jogos sérios sobre modelagem de processos de software são eficazes para melhorar a aprendizagem dos alunos em relação a esta área de conhecimento*.

A partir desta hipótese podem-se levantar as seguintes questões:

Questão 1: Jogar DesigMPS aumenta a eficácia de aprendizagem de MPS no nível de aplicação?

Questão 2: Como é a eficácia de aprendizagem de MPS com o DesigMPS comparada com a do ABP nos níveis de conhecimento, compreensão e aplicação?

Questão 3: O jogo é considerado adequado em termos de relevância de conteúdo, suficiência, grau de dificuldade, sequência, e método de ensino para o qual ele foi desenvolvido?

Questão 4: O jogo é atrativo?

Questão 5: Quais são os pontos fortes e os pontos fracos do jogo?

1.3 Objetivo geral

Desenvolver um jogo de computador para apoio do ensino de MPS, em que o aluno é desafiado a modelar processos de software no contexto de modelos de capacidade/maturidade de processos de software.

1.4 Objetivos específicos

1) Identificar uma abordagem de ensino que já tenha eficácia reconhecida.
2) Desenvolver o jogo baseado nesta abordagem de ensino
3) Aplicar o jogo em uma unidade instrucional sobre MPS, em que o aluno deverá modelar processos pertencentes ao MR-MPS-SW do MPS.BR.
4) Avaliar eficácia de aprendizagem do jogo nos níveis cognitivos de conhecimento, compreensão e aplicação, por meio de um experimento formal.
5) Comparar a eficácia de aprendizagem do jogo com método instrucional de aprendizagem baseada em projetos (*project-based learning*).

6) Avaliar a percepção dos alunos em relação à adequação em termos de relevância de conteúdo, suficiência, sequência, grau de dificuldade e, método de ensino, atratividade, pontos fortes e pontos fracos do jogo.

1.5 Contribuição e relevância

Relacionam-se a seguir os pontos relevantes e novos desenvolvidos nesta tese para superar os problemas e as limitações citadas:

- Desenvolvimento de um jogo para ensino de MPS que flexibilize e amplie a aplicação de exercícios práticos sobre o assunto.
- O jogo pode ser usado para o ensino de modelagem de qualquer processo de um MCMPS, desde que o processo possa ser modelado via sintaxe e semântica do SPIDER_ML.
- Adaptação da abordagem de ensino por meio de MC para que possam ensinar especificamente MPS.
- Contribuição com o estudo e resultados da aplicação do jogo proposto. Assim estudos semelhantes podem usufruir das conclusões e resultados obtidos por essa pesquisa.

1.6 Metodologia

O processo de pesquisa adotado nesse trabalho pode ser dividido em cinco fases:

Fase 1: Revisão da literatura

Inicialmente foi realizada a fundamentação teórica sobre os temas: MPS, abordagens para ensino de MPS, jogos para ensino de ES, e desenhos de pesquisa para avaliação de jogos. Em seguida, por meio de 3 revisões sistemáticas da literatura sobre jogos para ensino de ES, foram levantadas as áreas de conhecimento da ES que fazem uso de jogos para apoiar seu ensino. Também, uma revisão complementar foi realizada para verificar se houve novos trabalhos sobre esse tema.

Fase 2. Desenvolvimento da abordagem de ensino por meio de jogo para ensino de MPS

A partir do estudo inicial foram definidos os objetivos e os requisitos de um jogo para apoiar o ensino de MPS, assim como seu método de ensino. Logo após se iniciou o seu desenvolvimento. Nesse caso o método de ensino deve ser adequado para o ensino de MPS e ter eficácia relatada por meio de estudos experimentais. Fundamentando-se nisto, foi decidido

o uso de MC. Também foram elaborados questionários para avaliar a percepção dos alunos em relação à adequação em termos de relevância de conteúdo, suficiência, sequência, grau de dificuldade e, método de ensino, atratividade, pontos fortes e pontos fracos.

Fase 3. Definição do processo de aplicação do jogo

Após o desenvolvimento do jogo foi definido um processo com os passos para executar a avaliação na prática. Esse processo foi baseado em (Cohen et al. 2000), que estabelecem métodos para pesquisa em educação.

Fase 4: Aplicação do jogo educacional

A estratégia de pesquisa empregada foi um experimento formal com pré/pós-testes e grupo controle. O jogo foi usado durante uma unidade instrucional sobre MPS que fazia parte de uma disciplina de Qualidade de Software. A eficácia educacional do jogo foi avaliada nos níveis cognitivos de conhecimento, compreensão e aplicação de acordo com taxonomia revisada de objetivos educacionais de Bloom (ANDERSON; KRATHWOHL, 2001), e comparada com a eficácia de um ABP; e também foi avaliada a percepção dos alunos em relação à adequação em termos de relevância de conteúdo, suficiência, sequência, grau de dificuldade e, método de ensino, atratividade, pontos fortes e pontos fracos.
A avaliação ocorreu na sequência: ensino da unidade instrucional de MPS, pré-testes, uso do jogo, pós-testes, aplicação de questionários e análise dos dados.

Fase 5: Avaliação e análise do jogo

A eficácia de aprendizado causada pelo jogo nos níveis cognitivos foi avaliada por meio de testes de hipóteses com base nas notas obtidas nos pré/pós-testes. As análises foram realizadas fundamentadas com as informações recolhidas nos questionários.

1.7 Organização do trabalho

Capítulo 2: oferece um background sobre os assuntos necessários para melhor compreensão da pesquisa. Estes assuntos estão relacionados à MPS, linguagem de modelagem de processos de software, melhoria de processos de software, modelos de qualidade de processos de software, o programa MPS.BR e os seus processos de Verificação (VER), Validação (VAL) e Garantia de Qualidade (GQA), assim como ensino de modelagem

de processos de software apoiado por jogos, e avaliação adaptativa por meio de mapas conceituais.

Capítulo 3: descreve o estado da arte dos jogos sobre processos de software encontrados em revisões sistemáticas da literatura.

Capítulo 4: descreve sobre o DesigMPS, seus objetivos educacionais, *game flow*, estágios, modelagem dos processos de referência, requisitos, modificações necessárias nos métodos de avaliação adaptativa usando MC para que proporcionar avaliação de modelos de processos de software.

Capítulo 5: trata sobre a avaliação do DesigMPS, estratégia de pesquisa, questões de pesquisa, hipóteses, formulações e variáveis, assim como a execução.

Capítulo 6: trata sobre a análise das questões de pesquisa e o teste das hipóteses.

Capítulo 7: é realizada a discussão dos resultados, assim como as considerações sobre as ameaças à validade do trabalho.

Capítulo 8: considerações finais, discorre-se sobre os resultados confrontados com a hipótese e questões levantadas, trabalhos futuros, e apresentam-se as publicações que foram geradas dessa pesquisa.

1. REFERENCIAL TEÓRICO

1.1. Modelagem de processos de software

Existem diversas definições para o processo de software. Segundo Pfleeger (2001), um processo de software são as atividades que precisam ser realizadas para a construção de um software, não se limitando a isso, pois devem ser levados em consideração as pessoas, as ferramentas, a definição das atividades, os recursos, os artefatos consumidos e gerados, os procedimentos adotados e o modelo de ciclo de vida utilizado.

Os processos de software podem ser considerados como um conjunto de atividades, métodos, práticas e transformações que são usados para desenvolver e manter um software e seus produtos associados (planos de projetos, modelos de software, código fonte, casos de teste e manuais de usuários) (PAULK et al., 1993). Para Pressman (2011), um processo é a definição de uma metodologia indicada para conduzir atividades, ações e tarefas, propiciando estabilidade, controle e organização, que são indispensáveis para que o software produzido seja de qualidade. Em suma, o processo de software possui um foco em melhorar as etapas de elaboração do produto, para que o resultado possua um grau de qualidade considerado satisfatório.

Um processo de software pode ser descrito por meio de modelos que representam a definição de um processo de software (ALDAZABAL et al., 2008). A modelagem de processos de software descreve a criação de modelos de processo de software. Um modelo de processo de software é uma representação abstrata da arquitetura, design ou definição de um processo de software (FEILER; HUMPHREY, 1993). Cada representação descreve, em diferentes níveis de detalhes, uma organização de elementos de um processo e provê a definição do processo a ser usado para avaliação e melhoria. Um modelo de processo pode ser analisado, validado e simulado, caso seja executável. Os modelos de processo são usados principalmente no controle dos processos de software de uma organização (ACUÑA et al., 2000).

Para se entender e implementar sistematicamente um processo de software pode-se fazer uso da modelagem de processos. Isso pode ser feito de forma descritiva, prescritiva ou híbrida. A modelagem descritiva, como o nome sugere, descreve o processo de software da forma que ele ocorre na empresa; o modelo prescritivo descreve o processo como ele deveria ser (BECKER-KORNSTAEDT, 2001); e a abordagem híbrida engloba ambos os aspectos descritivo e prescritivo.

Podemos citar a importância desses modelos para o processo de software no que concerne ao suporte de: gerência, execução automatizada e, melhoria de processos (CURTIS; KELLNER; OVER, 1992). Assim, modelos de processos de software são um fator importante para entrega de softwares com qualidade, porque estes modelos permitem a gestão de processos de software e, ao mesmo tempo, ajudam a transformar as necessidades do usuário em um produto de software (ACUÑA et al., 2000). A MPS normalmente faz parte de programas de melhoria de processos de software. Pois com esses modelos pode-se realizar uma análise verificando os pontos fracos e oportunidades de melhoria (BECKER-KORNSTAEDT, 2001). Também é importante ressaltar que a MPS se difere de outros tipos de modelagens do campo da ciência da computação, porque muitos fenômenos modelados são executados por pessoas ao invés de máquinas (CURTIS; KELLNER; OVER, 1992).

Diferentes elementos de um processo podem ser modelados, por exemplo: atividades (tarefas), artefatos (produtos), papéis e ferramentas (HUFF, 1996). Os elementos mais comumente modelados (BENALI; DERNIAME, 1992; FINKELSTEIN; KRAMER; NUSEIBEH, 1994; MCCHESNEY, 1995) são:

- Papel: descreve um conjunto de agentes ou conjunto de responsabilidades, atribuições e habilidades para executar uma atividade específica do processo de software.
- Atividade ou tarefas: é uma etapa de um processo que produz mudanças externamente visíveis do estado de um produto de software. Uma atividade deve ter *inputs* e *outputs*. Tarefas incluem e implementam procedimentos, regras, políticas e objetivos para gerar e modificar artefatos. As tarefas podem ser executadas por pessoas ou sistemas automatizados.
- Ferramenta: Instrumentos utilizados pelos papéis na execução de uma atividade, por exemplo: ferramentas *case* e linguagens de programação.
- Artefato ou produto: é o produto, subproduto (*output*) ou matéria-prima (*input*) de uma tarefa ou processo. Um artefato produzido por uma tarefa pode ser usado mais tarde como *input* da mesma ou outra tarefa. Os artefatos podem ser criados, acessados ou modificados. Produto é o conjunto de artefatos de software entregues a um usuário.

1.2. Linguagem de modelagem de processos de software

Com o objetivo de modelar processos de software utilizam-se: i) linguagens, ii) abstrações, e iii) formalismos que são adequados para esse propósito (FINKELSTEIN et al., 1994) (ACUÑA et al., 2000). Neste contexto, linguagens de modelagem de processos de software (LMPS) são aquelas usadas para descrever modelos de processos de software (ZAMLI; LEE, 2001).

Segundo García-Borgoñon et al. (2014), as LMPS podem ser classificadas em três grupos: 1) linguagens baseadas em gramática que incluem todas que focam em linguagens formais, matemáticas e de programação, por meio de regras ou restrições (e.g. linguagens de programação, teoria dos grafos, Redes de Petri); 2) as que são baseadas em UML – *Unified Modeling Language* (e.g.: UML 1.1, UML 1.4 e UML 2.0); e 3) as baseadas em meta-modelos (e.g.: SPEM 1.1, SPEM 2.0 e *powertypes*).

Para se conseguir um processo de software bem definido e executável, é necessário estabelecer uma LMPS que abranja os principais aspectos gerais de processos de software, e também um ambiente que seja flexível o suficiente para modelar diferentes categorias de projetos (GARCIA-BORGOÑON et al., 2013). Assim, a abstração da linguagem, deve permitir a expressão de diferentes pontos de vista e perspectivas de processo, atendendo o nível de detalhe no qual um processo de software é representado (ACUÑA et al., 2000). O formalismo assegura a maneira na qual a linguagem é representada (e.g.: baseadas em gramática, baseadas na UML, baseadas em modelos), com sintaxe precisa e semântica detalhada (ARMENISE et al., 1993). As LMPS representam graficamente ou textualmente, pelo menos, os seguintes elementos básicos que constituem os processos de software (FUGGETA, 2000): atividades, papéis, artefatos, ferramentas e relações de precedência entre as atividades (e.g., *start_to_start, end_to_start*) (BENDRAOU et al., 2010).

Neste contexto, o SPEM 2.0 (*Software Process Engineering Meta-Model*) é um meta-modelo que suporta MPS (OMG, 2008). Sua notação, sintaxe e semântica são usadas como base para a especificação de novas LMPS (GARCIA; VIZCAINO; EBERT, 2011). Um exemplo é o SPIDER_ML (SPIDER, 2009), que procura integrar e formalizar práticas de modelagem de processos adotados pela indústria do software.

1.3. A Linguagem de Modelagem de Processos de Software – SPIDER_ML e a Ferramenta de Modelagem – SPIDER_PM

Dentro do contexto da melhoria de processos de software, a SPIDER_ML foi criada como uma linguagem para modelagem de processos padrão e instanciados de uma organização. Para esse fim, a SPIDER_ML possui um conjunto de elementos que podem ser usados para documentação, análise, e comunicação de um processo de desenvolvimento de software. A SPIDER_ML tem o objetivo de ser de fácil utilização e adequada aos aspectos de fato encontrados nos processos de desenvolvimento de software utilizados pelas organizações (SPIDER_ML, 2009). A SPIDER_ML tem 17 componentes que são utilizados para modelar aspectos dos processos de desenvolvimento de software. Mais 3 diagramas, que possuem seus próprios relacionamentos e componentes.

Um subconjunto de SPIDER_ML foi adotado para a MPS nas atividades do DesigMPS, conforme apresentado na Tabela 1.

Tabela 1. Subconjunto de elementos e relacionamentos do SPIDER_ML usados pelo DesigMPS

Nome	Notação gráfica	Descrição
Tarefa instanciada		Representa uma tarefa elementar do processo, que não pode ser decomposta em tarefas menores.
Artefato instanciado		Representa um resultado de uma tarefa instanciada. Também pode representar um artefato consumido ou modificado por essa tarefa
Papel instanciado		Representa um conjunto de recursos que possui um conjunto de capacidades, habilidades e características que o permitem realizar ou contribuir para a execução de tarefa
Ferramenta instanciada		Recurso auxiliar utilizado por um papel para realizar uma determinada tarefa
Estado inicial		Representa o ponto de partida do fluxo do processo
Estado final		Representa um ponto para a finalização de um fluxo do processo
Barra de separação		Representa um ponto que inicia a execução em paralelo de dois ou mais fluxos
Barra de junção		Representa um ponto onde dois ou mais fluxos que ocorrem em paralelo são finalizados
Transição		É um relacionamento entre duas tarefas do processo. Podem ser do tipos *end_to_start, start_to_start, start_to_end*
Associação		É um relacionamento em que a extremidade que apresenta a ponta da seta indica o destino do relacionamento e a outra extremidade é a origem do relacionamento
Decisão		Indica um desvio condicional de um fluxo

Fonte: Elaborada pelo autor (2015)

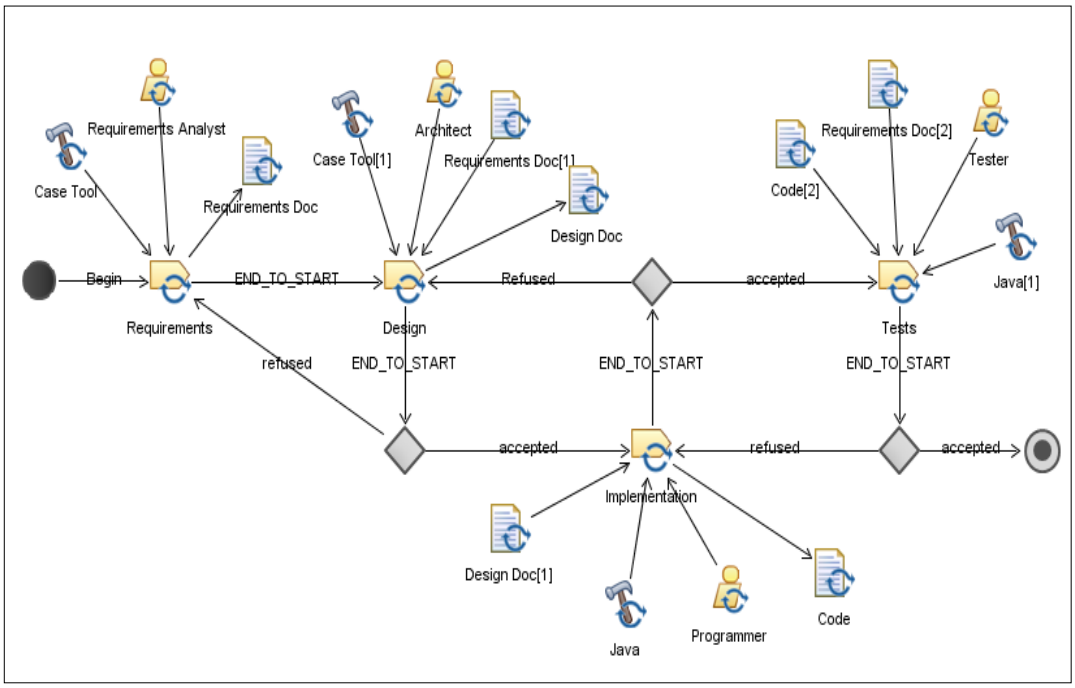
Para suporte computadorizado à modelagem de processo de software usando a SPIDER_ML se desenvolveu a ferramenta SPIDER_PM (SPIDER, 2010). Trata-se de um *software* livre que tem como objetivos: incorporar as práticas de modelagem adotadas pelas organizações; permitir a produção de modelos compatíveis com processos de software do mundo real; e facilitar e agilizar a construção desses modelos. Além disso, a Spider_PM também permite a modelagem de processos compatíveis com programas de melhoria como MPS.BR e CMMI. As principais características da SPIDER_PM são:

- Reutiliza as notações do SPEM 2.0 e da UML.

- Apresenta uma separação entre os elementos do processo que podem ser reutilizados e os que não podem.
- Faz persistência dos processos modelados em arquivos XML.
- Permite a extensão semântica dos componentes da modelagem.
- Valida os diagramas enquanto eles estão sendo definidos.

A Figura 1 apresenta como exemplo o clássico modelo de processo de software em cascata modelado em SPIDER_ML por meio do SPIDER_PM.

Figura 1. Exemplo de um modelo de processo de software na notação do SPIDER_ML



Fonte: Elaborada pelo autor (2015)

1.4. Melhoria de processos de software

Esta seção trata sobre a melhoria de processos, MCMPS, assim como o MPS.BR. Estes conhecimentos são relevantes para a pesquisa, pois o DesigMPS utiliza os processos dos modelos de melhoria de processos de software como contexto para ensinar MPS. Desta forma, foram usados no experimento (seção 5.1) os processos de software Verificação, Validação e Garantia da Qualidade pertencentes ao MR-MPS-SW do MPS.BR.

A melhoria de processo de software é uma área da ES que objetiva melhorar a qualidade do produto e a produtividade do processo de desenvolvimento (AAEN et al., 2001). Existem diversas normas e modelos de referência para melhorar o processo de software de

forma gradativa a partir do uso de boas práticas e recomendações (FUGGETTA, 2000), tais como os modelos CMMI-DEV – *Capability Maturity Model Integration for Development* (SEI, 2010); as normas ISO/IEC 12207 (ABNT, 2009) e ISO/IEC 15504 (ABNT, 2008); e o programa MPS.BR e seu modelo de referência MR-MPS-SW (SOFTEX, 2012b).

1.4.1. Modelos de qualidade de processo de software

A indústria de software e seu mercado estão em constante transformação, deste modo o acompanhamento destas mudanças pode ser facilitado quando existem padrões e modelos de referência de processos, direcionando as organizações a melhorarem seu desempenho e se fortalecerem (DE MELLO, 2001). Logo, o surgimento de diversas normas e modelos de referência para processo de software foi incentivado pelo crescente interesse nesta área nas últimas décadas (BIRK; PFAHL, 2002). Os modelos têm como principal objetivo atingir a organização como um todo, em detrimento das peculiaridades isoladas de cada projeto de software, cada processo de trabalho e cada equipe envolvida (SPINOLA; TONINI; CARVALHO, 2008).

Segundo o *Software Engineering Institute* (SEI, 2010), os primeiros modelos de qualidade de software basearam-se em estudos de melhoria de processos em outras áreas, que foram estendidos e aplicados a software nas pesquisas realizadas por Humphrey (1989). Estas pesquisas originaram princípios e conceitos básicos nos quais muitos dos modelos de maturidade capacidade estão baseados.

Vários modelos e normas de qualidade de processo de software estão disponíveis para orientação das organizações, e podem ser utilizados dependendo do mercado que se deseja atingir. Segundo Koscianski e Soares (2007), algumas das principais referências são: a norma ISO/IEC 12207 (ABNT, 2009), que estabelece terminologias e práticas relacionadas ao ciclo de vida do *software*; e os modelos CMMI (SEI, 2010) e MPS.BR (SOFTEX, 2012a).

O modelo CMMI exerce uma forte influência para que a comunidade de ES considere seriamente a melhoria de processo de software (PRESSMAN, 2011). Criado pelo SEI (*Software Engineering Institute*), originou-se a partir da integração de modelos existentes anteriormente denominados CMM (SEI, 2010).

O CMMI está dividido em três principais documentos: CMMI-DEV (*CMMI for Development*), voltado para o processo de desenvolvimento de software, CMMI-ACQ (*CMMI for Acquisition*), voltado para aquisição e terceirização e CMMI-SVC (*CMMI for Services*), voltado para empresas prestadoras de serviços.

O CMMI-DEV possui duas representações (por estágio e contínua), para orientar de forma distinta as organizações que desejem melhorar seu processo de *software*, ambas utilizam níveis cumulativos, ou seja, um nível de capacidade (ou maturidade) mais alto inclui os atributos dos níveis mais baixos. A representação por estágios distribui as diversas áreas de processos relacionadas ao desenvolvimento de software em níveis de maturidade, que vão em uma escala a partir do nível "1", no qual o processo da organização é caótico e indefinido, até o nível "5", no qual a organização deve buscar inovações no processo por meio da melhoria contínua.

A representação contínua orienta a evolução de cada área de processo individualmente. Cada uma destas áreas deve possuir um nível de capacidade que vai em uma escala de "0" no qual as definições de determinada área de processo encontram-se incompletas, até o nível "3", que indica que o processo está definido, gerenciado e realizado. Além disso, o CMMI-DEV está organizado em 22 distintas áreas de processo, no qual cada uma delas possui orientações à implementação, composta pelo propósito, notas introdutórias, outras áreas de processo relacionadas e objetivos específicos.

1.4.2. O programa MPS.BR

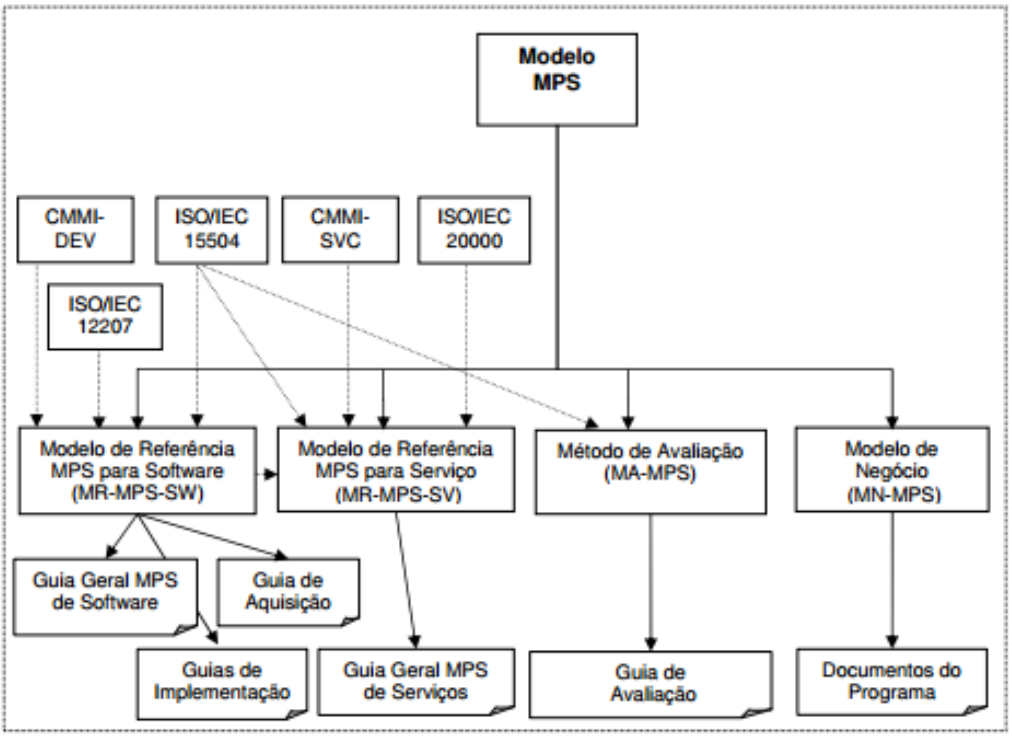
Apesar do modelo CMMI ser bastante consistente, a realidade do Brasil necessita abordagens diferentes, assim surgiu o programa MPS.BR. Ele defende a melhoria do processo de software de forma mais gradativa, permitindo a avaliação de múltiplas organizações por meio de um consórcio (SOFTEX, 2012a).

Através de parcerias coordenadas pela Associação para Promoção da Excelência do Software Brasileiro (SOFTEX), o programa foi criado em 2003 com o intuito de promover a melhoria de processo de software e serviços correlatos no país, por meio de duas principais metas: (i) meta técnica, visando a criação e o aprimoramento do modelo MPS e (ii) meta de negócio, pretendendo à disseminação e adoção do Modelo MPS em todas as regiões do país, em um intervalo de tempo justo, a um custo razoável, tanto em micro, pequenas e médias empresas, quanto em grandes organizações privadas e governamentais (SOFTEX, 2012a).

As diretrizes fornecidas pelo modelo MPS foram baseadas nas principais orientações fornecidas por padrões internacionais, como as normas ISO/IEC 12207, ISO/IEC 15504 (ABNT, 2008), ISO/IEC 20000 (ISO/IEC, 2011) e no CMMI, adaptadas à realidade das empresas brasileiras por meio de parcerias entre a SOFTEX, o Governo e as universidades.

O Modelo MPS atualmente é composto por quatro principais modelos: (i) o MR-MPS-SW (Modelo de Referência MPS para Software), que será abordado em detalhes na seção seguinte; (ii) o MR-MPS-SV (Modelo de Referência MPS para Serviço), que aborda diretrizes para empresas prestadoras de serviços; (iii) o MA-MPS (Método de Avaliação MPS), que aborda o método de avaliação das organizações que pretendem avaliar a melhoria do seu processo; (iv) e MN-MPS (Modelo de Negócio MPS), que descreve regras de negócio para implementação e avaliação de processos, certificação de consultores e programas anuais de treinamento. A Figura 2 apresenta graficamente a estrutura do modelo, bem como a relação com os documentos no qual foram baseados.

Figura 2. Componentes do Modelo MPS



Fonte: SOFTEX (2012a)

Existem três principais papéis que atuam com os modelos fornecidos: implementador, avaliador e consultor de aquisição. O implementador deve realizar e ser aprovado em uma prova, denominada P2, que tem como pré-requisito a realização do curso C1, de introdução ao MPS.BR. Entre o curso C1 e a prova P2 existem, na sequência, a prova P1 (relacionada ao conteúdo de C1) e o curso C2, exclusivo para implementadores. A prova P1 e o curso C2 são facultativos, sendo utilizados apenas para avaliação de conhecimento.

O avaliador também deve cumprir alguns requisitos: já ter sido implementador em algumas avaliações prévias e também realizar o curso C3 e a prova P3, exclusivos para avaliadores MPS.BR.

O consultor de aquisição também deve cumprir alguns requisitos: o curso C4 e a prova P4, referente ao conteúdo do guia de aquisição fornecido pelo programa; estes podem ser realizados independentemente dos cursos e provas citados previamente.

1.4.2.1. O modelo MR-MPS-SW

Este modelo de referência é voltado para maturidade e capacidade do processo de software, e contém os requisitos que os processos das unidades organizacionais devem atender para estar em conformidade com o MR-MPS-SW. Como mostra a Figura 2, é composto por três documentos:

- O guia geral: contém a descrição geral do modelo MPS e detalha o modelo de referência MR-MPS-SW, seus componentes, tais como áreas de processos e resultados esperados, e também descreve as definições comuns necessárias ao seu entendimento e aplicação.
- O guia de implementação: é, na verdade, um conjunto de documentos que fornecem orientações para implementar os níveis de maturidade nas organizações, fornecendo documentações a respeito de cada área de processo com um maior nível de detalhe.
- O guia de aquisição: é um documento que descreve um processo de aquisição de software e serviços correlatos, de forma que seja possível adquirir produtos de software apoiando-se no MR-MPS-SW.

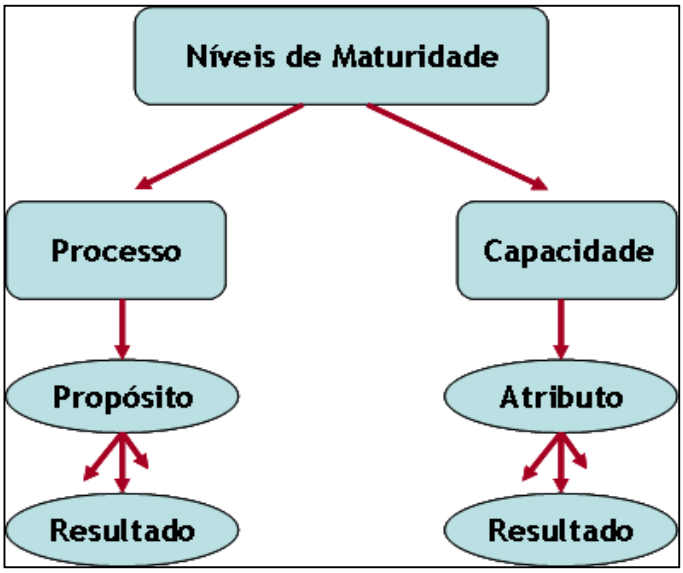
O MR-MPS-SW é composto por 23 áreas de processo distribuídas em sete níveis de maturidade denominados por letras do alfabeto, começando do nível de menor maturidade até o último nível: G (Parcialmente gerenciado), F (Gerenciado), E (Parcialmente definido), D (Largamente definido), C (Definido), B (Gerenciado quantitativamente) e A (Em otimização).

Cada uma das áreas de processos que compõem os níveis de maturidades possui a descrição de seu propósito e de seus resultados esperados, que são os objetivos menores, que devem ser alcançados como resultado da implementação do processo. Além disso, cada nível de maturidade também possui uma descrição da capacidade de processo que deve ser

alcançada, por meio dos resultados de atributos de processo, que devem ser aplicados a todas as áreas de processo que estão sendo implementadas.

A Figura 3 apresenta graficamente a estrutura dos níveis de maturidade, que é alcançada com todos os resultados esperados e resultados de atributos de processo são satisfeitos para as áreas de processos daquele nível.

Figura 3. Estrutura do MR-MPS



Fonte: SOFTEX (2012b)

Analisando a Figura 3, podemos identificar com detalhes que um nível de maturidade possui requisitos relacionados tanto à maturidade, quanto à capacidade do processo organizacional. A maturidade é composta por um conjunto de áreas de processos, sendo que cada uma possui um propósito e um conjunto de resultados esperados. Com relação à capacidade, são listados um conjunto de resultados de atributos de processos, que devem ser aplicados a cada uma das áreas de processo já implementadas na organização.

1.4.2.2. Verificação

Boehm e Basili (2001) afirmam que uma grande parte do esforço de um projeto de software - de 40% a 50% - é consumido em atividades relacionadas a retrabalho, que poderiam ser contidas por meio da aplicação de conceitos abordados na ES.

Com o esforço voltado para essas atividades, a tendência é o aumento do custo total, pois o custo para correção de defeitos cresce conforme o avanço às fases finais do projeto.

Segundo Andersson (2003), o custo para correção de defeitos na etapa de testes é de dez a cem vezes maior, se comparado às fases de codificação ou projeto. Logo, uma estratégia para a detecção de erros desde as fases iniciais do projeto é uma solução importante para que se obtenha um produto de qualidade e sem custos excessivos. Portanto, tornam-se útil as abordagens de Verificação em um processo de *software*.

Pressman (2011) define a verificação como um conjunto de tarefas que garantem que o software implementa corretamente uma função específica via avaliações. Elas devem ser realizadas desde a fase de requisitos. Para Shculmeyer e Mackenzie (1999), a verificação deve assegurar que produtos gerados em uma fase do processo estão alinhados aos requisitos das fases atual e anterior. A norma ISO/IEC 12207 (ABNT, 2009) delimita o objetivo da verificação, definindo que tem o intuito de determinar se os produtos de software atendem aos requisitos ou condições impostas nas atividades anteriores, por intermédio de exame e por fornecimento de evidência objetiva.

Todas essas definições corroboram o fato de que verificação trata da corretude dos produtos de trabalho de acordo com os requisitos de qualidade especificados previamente.

No modelo MR-MPS-SW a verificação situa-se no quarto nível de maturidade - nível D - no qual estão os processos relacionados à engenharia de sistemas. No total, o modelo determina seis resultados esperados para esta área de processo.

Os resultados esperados definem que os produtos de trabalho que serão verificados devem ser identificados. Antes da execução das tarefas de verificação, deve ser adotada uma estratégia para tal, assim como devem ser definidos critérios para a verificação destes produtos de trabalho.

Além disso, os demais resultados esperados especificam detalhes de que as atividades de verificação em si, devem ser realizadas, por exemplo, via testes e revisões por pares; cada defeito identificado deve ser registrado. Ainda, os resultados obtidos nas v erificações devem ser analisados e disponibilizados às partes interessadas.

1.4.2.3. Validação

A validação é comumente associada à verificação. Boehm (1981) afirma que comumente elas são executadas em conjunto, devido ao fato de ser árdua a identificação de ambas separadamente durante um projeto. Além disso, pode-se dizer que a verificação tem o objetivo de avaliar se o produto está sendo desenvolvido corretamente, enquanto que a Validação preocupa-se em avaliar se o produto está de acordo com as aspirações do cliente.

Rocha, Maldonado e Weber (2001) definem que o objetivo da validação é garantir que o software que está sendo desenvolvido está de acordo com os requisitos do usuário, e que sua execução consiste em verificar a presença de defeitos e aumentar a confiança de que o produto esteja correto.

As diferenças entre verificação e validação estão relacionadas à equipe, no qual a primeira está mais alinhada à equipe interna, e a segunda à equipe externa (cliente e usuários). Do mesmo modo, a verificação preocupa-se com o atendimento das especificações e a identificação de defeitos, enquanto que a validação empenha-se em atender aos requisitos do cliente e à identificação de problemas.

É importante ressaltar que a validação, diferentemente da verificação não trata somente de testes. Devem ser especificados requisitos e o uso pretendido para estes requisitos deve ser avaliado. Portanto, além dos testes, diversas outras técnicas podem ser utilizadas na validação, dentre elas: prototipação, simulações, *model checking*, utilização de cenários de caso de uso (PRESSMAN, 2011).

No MR-MPS-SW, o processo de validação também está localizado no nível D. Ele possui um total de sete resultados esperados. Pode ser feito um paralelo com os resultados esperados do processo de verificação, com exceção do último resultado. Assim como na verificação, os resultados esperados demandam a identificação dos produtos de trabalho, e que sejam especificados previamente a estratégia para validação e quais critérios serão adotados. Também, deve haver evidências da execução de atividades de validação durante todo o ciclo de vida do projeto; os problemas devem ser registrados e identificados; e os resultados dessas atividades devem ser analisados e disponibilizados às partes interessadas.

O resultado esperado que diferencia a validação da verificação, refere-se à necessidade de fornecer evidências de que os produtos de *software* desenvolvidos estão prontos para uso.

1.4.2.4. Garantia da qualidade

A definição do processo garantia da qualidade, segundo a ABNT (2009), é de garantir que tanto os produtos, quanto os processos de software estejam em conformidade com os padrões, procedimentos e descrições de processos definidos na organização para o projeto. Outra definição, fornecida por IEEE (1990) diz que a garantia da qualidade envolve um conjunto de atividades de avaliação do processo pelo qual os produtos são desenvolvidos, com o intuito de fornecer confiança necessária de que estejam em conformidade com os requisitos técnicos previamente especificados.

Por tratar de avaliações objetivas tanto de processos, quanto de produtos, é importante que os responsáveis pela garantia da qualidade tenham autoridade e autonomia organizacional, de forma independente às pessoas responsáveis pela execução do processo. Esta independência pode ser alcançada por meio de um grupo de garantia da qualidade ou da atuação de um profissional externo ao projeto que será avaliado (SOFTEX, 2012c).

No modelo de referência elaborado pelo programa MPS.BR, o processo de Garantia da Qualidade se encontra no nível F, segundo nível de maturidade a ser alcançado. Seu propósito é fornecer visibilidade do projeto para todos da organização por meio de uma visão independente em relação ao processo e ao produto. A garantia de qualidade é um apoio para o gerente e agrega valor à equipe de projeto, ajudando-a a preparar e rever procedimentos, planos e padrões, desde o início do projeto até o seu encerramento (SOFTEX, 2012c). O processo de garantia da qualidade possui quatro resultados esperados:

- O primeiro resultado esperado define que cada produto de trabalho a ser entregue ao cliente, deve antes ser avaliado por meio de critérios objetivos, que podem ser estabelecidos pela definição e utilização de *checklists*, questionários etc.
- O segundo resultado esperado também trata de avaliações objetivas que devem ser realizadas, porém desta vez o foco é no processo, ou seja, cada área de processo já implementada na organização deve ser avaliada.
- O terceiro resultado esperado sugere que os problemas e não conformidades identificados durante a avaliação devem ser registrados e comunicados aos interessados.
- Finalmente, o quarto resultado esperado trata das ações corretivas para as não conformidades identificadas. Elas devem ser registradas e acompanhadas até as efetivas conclusões, sendo que em casos que necessário, as ações devem ser escalonadas aos níveis superiores, de forma a garantir a solução.

A adoção da Garantia da Qualidade, assim como da Verificação e da Validação, podem trazer alguns possíveis benefícios como: (i) a maior satisfação do cliente, devido ao melhor atendimento de suas necessidades; (ii) a redução do retrabalho; (iii) a redução de custos, em consequência da redução do retrabalho e; (iv) a maior competitividade, originada a partir do aumento.

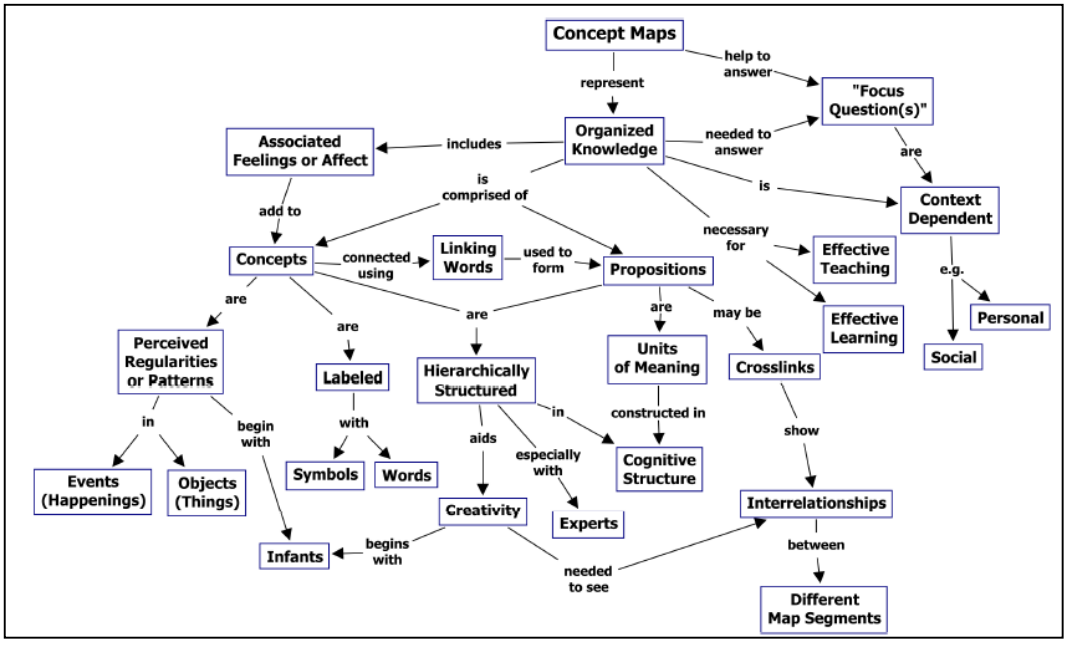
1.5. Mapas conceituais

A avaliação adaptativa por meio de mapas conceituais (MC) foi a abordagem de ensino que orientou o desenvolvimento do DesigMPS. Assim, faz-se necessário o embasamento teórico tanto sobre MC quanto de seus métodos de avaliação adaptativa.

Um MC é uma maneira gráfica de representação e organização de conhecimento (Figura 4). Ele é composto por nós e ligações (relacionamentos), dispostos em alguma ordem de modo a refletir o domínio das informações. Os nós simbolizam conceitos e as ligações representam relações entre os conceitos. Os nós e as ligações podem ser categorizados.

Mapeamento é o processo de construção de MC. Neste sentido mapeamento é definido como um método para representar graficamente o conhecimento e a informação (NOVAK, 1977). Um MC também pode ser usado "para exteriorizar e tornar explícito o conhecimento conceitual que o aluno realiza em um domínio do conhecimento" (CAÑAS, 2003). Em virtude disso, o MC tornou-se uma ferramenta útil de ensino e aprendizagem (BRUILLARD et al., 2000). Uma das razões do MC ser tão poderoso para a promoção aprendizagem é que ele serve como uma espécie de modelo ou andaime que ajuda a organizar e a estruturar o conhecimento (NOVAK; CAÑAS, 2008).

Figura 4. Representação do mapa conceitual através de um mapa conceitual



Fonte: Calvo et al. (2011)

Normalmente, os mapas conceituais são representados como uma hierarquia com a maioria dos conceitos gerais no topo do mapa e os conceitos mais específicos colocados nos níveis mais baixos (NOVAK; CAÑAS, 2006). É possível fornecer aos alunos tarefas com vários níveis de dificuldade. Assim, pode-se avaliar diferentes níveis de conhecimento, oferecendo aos alunos tarefas com diferentes graus de dificuldade (RUIZ-PRIMO, 2004).

O direcionamento (do inglês: *directeness*) está relacionado com as informações fornecidas aos alunos (RUIZ-PRIMO, 2004). As tarefas podem variar de um alto nível de direcionamento até baixos níveis de direcionamento. Tarefas com alto nível de direcionamento oferecem ao aluno os conceitos, as frases de ligação e uma estrutura de mapa. Em contraste, tarefas com baixo nível de direcionamento deixam o aluno livre para decidir quais os conceitos e quantos deles devem ser incluídos e como eles serão relacionados em seus mapas. Em outras palavras, as tarefas podem ser divididas em dois subconjuntos de tarefas:

- *Fill-in*: onde é fornecida ao aluno uma estrutura em branco de um mapa e listas de conceitos e frases de ligação para colocar nos lugares corretos na estrutura.
- *Construct a map*: os alunos são livres para fazer suas próprias escolhas ao construir um mapa.

Tarefas *fill-in* podem variar. A primeira variação é referente ao que é fornecido para o aluno (lista de conceitos e/ou lista de rótulos de relacionamentos), que ele pode usar. Em segundo lugar, elas variam de acordo como a estrutura do mapa pré-definido que é fornecido: a estrutura já contém alguns conceitos preenchidos e/ou relacionamentos rotulados, ou ele está vazio. Tarefas *construct a map*, assim como as tarefas *fill-in* podem oferecer variedade e restrições sobre conceitos e rótulos disponíveis necessários para usar no mapeamento. Nesse caso a estrutura em branco não é fornecida.

1.6. Avaliação adaptativa

A identificação do nível de conhecimento de um aluno constitui parte essencial para que as atividades a ele dirigidas sejam mais apropriadas e possibilitem que suas atuais necessidades de aprendizagem sejam melhor atendidas. A avaliação adaptativa baseada em computador é um tipo específico de avaliação (BRUSILOVSKY, 1996; PAPANASTASIOU, 2003), em que são disponibilizadas aos estudantes atividades (testes) com maior ou menor nível de dificuldade, de acordo com o desempenho nas tarefas precedentes. Quanto mais erros cometidos, maior será o nível de facilidade da nova atividade a ele atribuída, e vice-versa.

1.6.1. Pontos fortes da avaliação adaptativa

Entre as vantagens da avaliação adaptativa destacam-se (PAPANASTASIOU, 2003):

- Permite que o nível de conhecimento dos alunos seja avaliado com maior precisão, favorecendo a adequação do grau de desafio dos testes ao grau de desenvolvimento de cada aluno, evitando que as tarefas se tornem desestimulantes.

- Mantém um histórico dos testes realizados por aluno, que serve como base para a determinação dos próximos testes. Além de possibilitar a aplicação de testes menores de resolução mais rápida, para avaliar progressivamente, em lugar de um único teste longo e demorado aplicado para avaliar todo o conhecimento de uma única vez.

- Oferece um *feedback* imediatamente após a realização da tarefa, permitindo o redirecionamento do ensino por meio de atividades adequadas para superar as deficiências detectadas.

1.6.2. Avaliação adaptativa e mapas conceituais

O uso de MC como estratégia de avaliação tem sido usado como instrumento de avaliação desde o inicio da década de 1980 (NOVAK; GOWIN, 1984), mas não dentro de um contexto adaptativo, que se constitui uma área de pesquisa relativamente recente.

A avaliação adaptativa, por meio de MC, possibilita que a aprendizagem possa ser desenvolvida respeitando o ritmo próprio de cada aluno (ROCHA et al., 2008), porque assegura que a construção de um MC atribuído como tarefa terá sempre como base o real conhecimento do aprendiz. A avaliação adaptativa também garante que o grau de dificuldade aumentará proporcionalmente à evolução deste conhecimento.

A avaliação adaptativa automatizada possibilita a produção de indicadores de evolução do conhecimento do aluno, tornando mais fácil para o professor a elaboração de testes, de modo que o conteúdo não se distancie demais do nível atual de conhecimento do aprendiz, nem para menos nem para mais. Além disso, torna possível o gerenciamento da evolução de cada aluno pelo professor com menos esforço, o que seria inviável com o uso de um método de avaliação não automatizado.

1.6.3. Avaliação baseada na estrutura do mapa conceitual

Nesta forma de avaliação, um MC construído por um especialista (professor) é usado como base para avaliar os MC propostos pelos alunos (ANOHINA et al., 2007). A avaliação se funda na procura de padrões entre os conceitos dos MC dos alunos e os do professor, aos quais é atribuída uma pontuação que varia numa escala de pontos. Essa abordagem está mais concentrada na estrutura dos MC, na disposição dos conceitos e nas relações entre eles. Sempre que um novo conteúdo é ensinado, novos MC são elaborados pelos especialistas, sendo que cada nova versão é uma complementação do MC anterior. A nova versão representa conhecimentos mais completos sobre um determinado assunto.

O aluno deve completar os conceitos que estão faltando na estrutura do MC proposto pelo professor, num processo gradual, até que o MC contenha os conceitos pertinentes a determinado assunto na sua completude. Desta forma os conhecimentos são avaliados gradualmente na medida em que novos conceitos são apresentados ao aluno. Apesar da vantagem de não sobrecarregar o aluno com uma única avaliação envolvendo todo o conteúdo, essa abordagem, por desconsiderar a avaliação semântica, apresenta a desvantagem de limitar a expressão da solução do problema pelo aluno, exigindo que ele empregue os mesmos termos para rotular conceitos e frases de ligação usados pelo professor.

A adaptação das avaliações utiliza duas estratégias:

1- A primeira é o monitoramento do preenchimento dos conceitos na estrutura do MC em branco pelo aluno, para detectar se ele está tendo dificuldades.

   Os desafios que se apresentam ao monitoramento do preenchimento da estrutura do MC em branco pelo aluno envolvem: a decisão do parâmetro que o sistema deve considerar para identificar quando o aluno tem dificuldade, que pode ser o tempo que ele permanece sem realizar nenhuma tarefa no sistema antes de inserir o próximo conceito, ou a quantidade de vezes que ele modifica um mesmo conceito em branco com valores diferentes; o princípio que definirá que o sistema deve intervir colocando um conceito correto num lugar em branco; e o que o sistema deve fazer com os conceitos inseridos pelo aluno no lugar errado na estrutura no MC.

2- A segunda é o grau de direcionamento (tradução livre de *directedness*) (GRAUDINA; GRUNDSPENKIS, 2006) de uma avaliação que o sistema oferece ao aprendiz durante a construção de um MC. Este grau pode ser alto, quando são

fornecidas a estrutura do MC e a lista de conceitos para que ele preencha nos espaços em branco; ou baixo, quando o aluno precisa definir os conceitos e as ligações entre eles.

Segundo Anohina et al. (2007), embora o uso dos MC para avaliação seja viável, por se tratar de uma área recente, muita pesquisa ainda precisa ser feita sobre as estratégias de adaptação das avaliações.

1.7. Sistema de Avaliação Inteligente de Conhecimento - Intelligent Knowledge Assessment System

O *Intelligent Knowledge Assessment System* (*IKAS)* (ANOHINA; STRAUTMANE; GRUNDSPENKIS, 2010; ANOHINA; GRAUDINA; GRUNDSPENKIS, 2007) é um modelo computadorizado para avaliação de MC criados por alunos. Ele tem dois objetivos: a) promover a autoavaliação dos alunos; b) e apoiar o professor na melhoria dos cursos por meio dos resultados da avaliação sistemática do conhecimento dos alunos.

O DesigMPS se baseou no *IKAS* para estabelecer o mecanismo de avaliação dos modelos de processos de software, visto que esses modelos tem uma equivalência com os MC (ver Tabela 3). Os conceitos dos MC podem representar tarefas, artefatos, papéis e ferramentas, os *links* do MC podem representar as associações e a dependência dos modelos de processos de *software*.

O *IKAS* é usado na seguinte maneira: o professor define estágios do conhecimento a ser avaliado e cria MC para todos os estágios, especificando os conceitos e os relacionamentos, de forma que um estágio é nada mais do que a extensão do anterior. Os estágios correspondem a avaliações de conhecimento por meio de MC. Depois o estudante submete o MC que ele criou e o sistema compara com o MC do professor e gera o *feedback*. Duas formas são possíveis: a) um modo de autoavaliação do conhecimento, em que o propósito é permitir a um estudante avaliar seu nível de conhecimento e aprender mais sobre um tópico específico no caso de ter um nível de conhecimento considerado insuficiente, e b) um modo de controle de conhecimento objetivando a determinação do nível de conhecimento por um professor.

O *IKAS* oferece seis tarefas de diferentes graus de dificuldade: quatro delas são do tipo *fill-in-the-map* (1ª Inserção de alguns conceitos na estrutura de um MC já contendo *links* rotulados e alguns conceitos inseridos em suas posições corretas; 2ª Inserção de conceitos na estrutura de um MC que contém *links* rotulados; 3ª Inserção de conceitos na estrutura de um

MC sem *links* rotulados; 4ª Inserção de conceitos e rotulação de *links* na estrutura de um MC, essa estrutura está vazia), e duas tarefas do tipo *construct-the-map* (5ª Criação de um MC a partir de um conjunto de conceitos; 6ª Criação de um CM a partir de um conjunto de conceitos e rótulos de *links*).

O MC do professor serve como um modelo padrão com o qual o MC do aluno é comparado. Além disso, um algoritmo de comparação (ANOHINA; VILKELIS; LUKASENKO, 2009) tem sido desenvolvido e leva em consideração critérios como: existência de relacionamentos (*links*), posições de ambos os conceitos, tipo e direção do relacionamento, rótulo correto.

1.8. Ensino de Engenharia de Software apoiado por jogos

De acordo com Zyda (2005), jogo sério de computador é um desafio mental com regras especificas para capacitar e/ou educar de forma lúdica. Para Dempsey, Lucassen e Rasmussen (1996), jogos sérios são aqueles projetados para ensinar sobre determinados assuntos, expandir conceitos, ou ajudar a exercitar ou aprender uma habilidade ou mudar atitudes. E esse tipo de jogo vem sendo utilizado para o ensino de áreas de conhecimento da ES e geralmente estão focados nas seguintes áreas de conhecimento do SWEBOK (IEEE-CS, 2004): gerência de engenharia de software (13 jogos), processos de engenharia de software (7 jogos), requisitos de software (6 jogos), projeto de software (2 jogos), construção de software (1 jogo) (CAULFIELD et al., 2011). Esta evidência mostra que os jogos para o ensino de ES não lidam especificamente com o ensino de MPS.

A efetividade educacional desses jogos tem sido pouco avaliada, dado em partes pela complexidade de se realizar experimentos adequados que indiquem níveis de evidência de alta confiança (ALMSTRUM et al. 1996; NAVARRO; VAN DER HOEK, 2007). Na revisão sistemática da literatura de (WANGENHEIM; SHULL, 2009) apenas um jogo foi avaliado experimentalmente em 4 estudos (PFAHL;KOVAL; RUHE, 2001; PFAHL et al. 2003; PFAHL et al. 2004; RODRIGUEZ et al. 2006) e teve um impacto positivo na aprendizagem. Um estudo relatou sobre um jogo que não teve nenhum impacto no aprendizado (DRAPPA; LUDEWIG, 2000), e outro estudo relatou que um jogo teve impacto menor do que comparado com outros métodos de ensino (NAVARRO, 2006). Porém, com base nos *feedbacks* subjetivos, muitos estudos relataram que os alunos preferem atividades com jogos do que os métodos tradicionais de instrução.

2. ESTADO DA ARTE SOBRE JOGOS PARA ENSINO DE ENGENHARIA DE SOFTWARE

Para fundamentar uma análise sistemática do estado da arte referente a jogos sérios (educacionais) com o objetivo de ensinar MPS, os seguintes artigos de revisão sistemática da literatura sobre o tema foram usados: Wangenheim e Shull (2009), Connolly, Stansfield e Hainey (2007) e Caulfield (2011). Em nenhum deles foi relatado jogo específico para ensinar MPS.

Como forma de complementar e atualizar o conhecimento contido nesses artigos se realizou uma pesquisa no *Googlescholar*, em janeiro de 2013. Os termos de pesquisa utilizados foram: <"*software process modeling*" *game teaching education training*>. Na busca inicial foram obtidos 70 resultados, em que se realizou uma avaliação superficial pelos títulos e resumos, destes foram descartados 59 trabalhos que não tratavam sobre jogos.

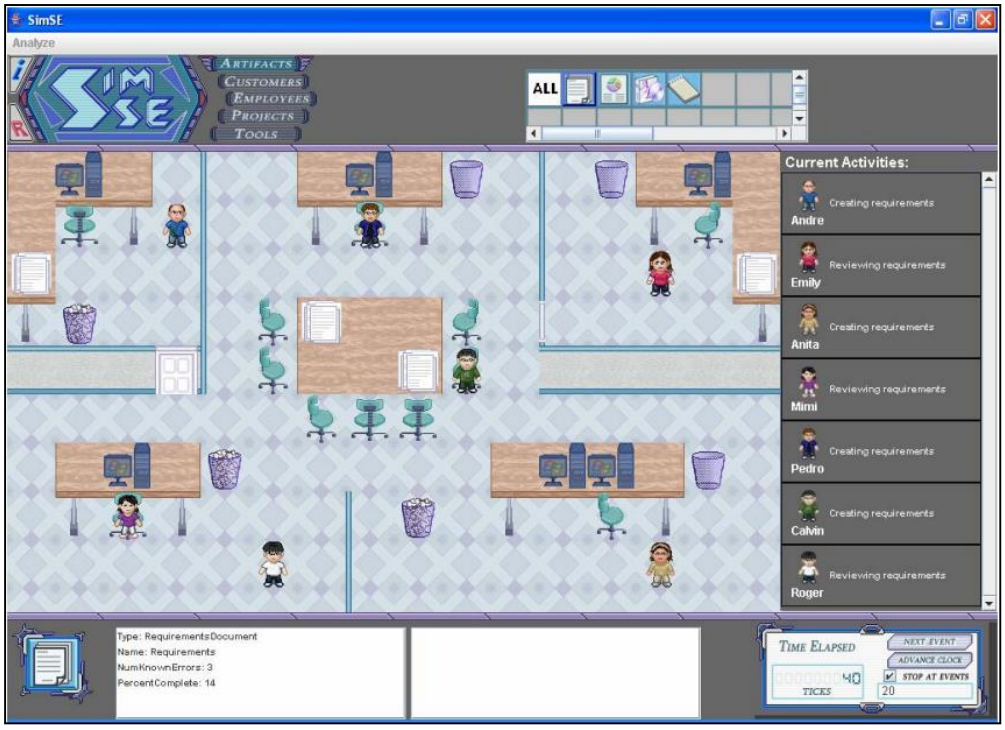
Em um segundo momento, avaliando-se integralmente os 11 trabalhos restantes, excluiram-se 10 jogos de simulação, porque não ensinam modelagem de processos em si, pois neles os alunos são treinados na execução de processos específicos. Apenas o jogo SIMSE (NAVARRO, 2006) foi considerado relevante e classificado como jogo que pode proporcionar o ensino da MPS dentro do foco da nossa pesquisa.

O SIMSE é um jogo de simulação de processos de *software*, que possibilita a MPS em um ambiente gráfico e interativo chamado de *ModelBuilder*. Apesar do principal propósito educacional do SIMSE não ser o ensino da modelagem, ele já foi usado em uma atividade de aprendizagem em que os alunos deveriam modelar seus próprios processos (BIRKHOELZER; NAVARRO; HOEK, 2005). Porém, encontraram-se as seguintes limitações do uso do *ModelBuilder*: modelar é uma tarefa muito complexa, pois o *ModelBuilder* não possui uma LMPS gráfica para este fim; a sua usabilidade não é centrada no aluno; e são necessárias muitas interações de testes de simulação para ajustar o modelo, isto acontece porque os modelos descritos pelo *ModelBuilder* necessitam de variáveis específicas para simulação. O resultado desta busca, conjuntamente com as revisões sistemáticas da literatura (WANGENHEIM; SHULL, 2009; CONNOLLY; STANSFIELD; HAINEY, 2007; CAULFIELD et al., 2011), evidenciam a falta de jogos específicos com o objetivo de ensinar MPS.

2.1. SIMSE (Software Engineering Simulation Environment)

Desenvolvido como tese de doutorado de Navarro (2006), este é um jogo de simulação de processo de software. O SIMSE permite que sejam criados modelos de processos de software mais complexos e completos do que os comumente usados em práticas de disciplinas de engenharia de software, tais como: grandes equipes, projetos de larga escala, decisões críticas, múltiplos *stakeholders*, orçamento, planejamento, e eventos inesperados e aleatórios. O SIMSE fornece ao aluno possibilidade de experimentar os diferentes aspectos do processo de software, por meio de modelos customizados que representem os fenômenos necessários ao aprendizado do aluno. A Figura 5 mostra uma tela do SIMSE.

Figura 5. Tela do jogo SIMSE



Fonte: Elaborada pelo autor (2015)

Os modelos do SIMSE são descritos com um conjunto de quatro elementos: *Object Type*, *template* para todos os elementos do processo de software (*artefacts*, *roles*, *costumers*, *tools*); *Start state*, instancia os elementos no começo da simulação com valores para os seus estados iniciais; *Actions*, atividades que os elementos participam ou se relacionam; *Rules*, definem o comportamento e os efeitos que a as ações tem durante a simulação. Esses construtores básicos são característicos a maioria dos processos de *software*, e permitem modelar uma grande variedade de situações peculiares para cada tipo de processo.

Para o desenvolvimento dos modelos, existe o módulo chamado *ModelBuilder*, que é um ambiente que auxilia de modo semiautomático os instrutores a construírem modelos de simulação customizados. Nesses modelos estão definidas as lições sobre processo de software que o aluno deve aprender.

O *ModelBuilder* fornece elementos gráficos que deixam transparente a linguagem de modelagem para o modelador. Essa característica proporciona mais simplicidade, facilidade e rapidez para construir modelos do que por meio de codificação direta em uma linguagem de modelagem.

Para o SIMSE foi desenvolvida uma LMPS, puramente textual, que atende os seguintes requisitos da simulação: caráter educacional, interatividade, *game-based*, e elementos gráficos de interação. Essa linguagem é baseada em XML, e não possui construtores comuns às linguagens de programação: *if-then*, *loops* e predicados. A falta destes compromete um pouco a expressividade e a flexibilidade da linguagem, porém torna maior a facilidade e a rapidez do desenvolvimento dos modelos. Na definição da linguagem alguns *tradeoffs* foram necessários, o mais importante a destacar é que foram preteridas a sofisticação e a qualidade dos gráficos dos jogos comerciais em função de se permitir modelos facilmente customizáveis. A linguagem não tem a separação entre o modelo de domínio e o modelo comportamental, ou seja, apenas um único modelo representa a estrutura e o comportamento. Assim, esse forte acoplamento entre eles dificulta o reuso dos modelos.

Uma contribuição importante desse trabalho é a compilação do que a autora chamou de “regras fundamentais da engenharia de software”, que são fenômenos que podem ocorrer durante o processo de software. Esses fenômenos serviram para que a autora entendesse melhor as características da simulação de processo de software e orientaram as funcionalidades necessárias para modelagem e simulação do SIMSE.

A interface com o aluno é totalmente gráfica, todas as possíveis interações são realizadas via *mouse*. Cada ícone representa um elemento do modelo de processo de software e suas respectivas propriedades. Atributos numéricos e *string* são apresentados apropriadamente para os alunos.

Além dos elementos do modelo de processo de software, existe um cenário gráfico que simula o espaço físico onde ocorre o processo de desenvolvimento. Nesse cenário são posicionados os *roles*, divisórias, cadeiras, mesas e computadores. Todos os elementos dos cenários são fixos, ocupando um mesmo lugar (uma coordenada X, Y) durante todo o andamento do jogo. Essa característica estática dos elementos que compõe o cenário é desfavorável à jogabilidade, tornando o SIMSE enfadonho com simulação de longa duração.

Muitos alunos demonstraram que esperavam gráficos sofisticados para que o jogo tornar-se mais interessante. Limitação dos recursos tecnológicos do jogo é justificada para não adicionar complexidade à linguagem de modelagem de processo.

O SIMSE, um dos jogos mais relevantes em educação de engenharia de software, é referência por outros trabalhos, e serviu de fundamento e inspiração para outros, a exemplo de Hsueh, et al. (2008) e MO-SEProcess (YE; LUI; POLACK-WAHL, 2007). Ele está disponível gratuitamente, e é aberto para melhorias e novas funcionalidades adicionadas pela comunidade interessada.

Apesar do SIMSE se mostrar uma ferramenta útil no ensino de ES, seu uso deve ser complementar a outras técnicas educacionais e contar com a orientação de preceptores.

Entre os problemas apresentados pelo SIMSE, destacam-se:

- Dificuldade de criar modelos de simulação: apesar do uso do *ModelBuilder* abstrair a linguagem textual, esta apresenta algumas deficiências que diminuem sua flexibilidade e expressividade. O desenvolvimento de modelos é ainda muito complexo e precisa passar por sucessivos refinamentos.
- Melhores recursos gráficos: gráficos sofisticados podem deixar os jogos mais atrativos e interessantes para os alunos. Adicionar semântica ao posicionamento dos roles entre si e aos outros elementos do cenário, por exemplo, um *role* próximo a algum *role* ao qual ele é “simpático” aumentará a produtividade de ambos; ou o incremento do número de páginas da documentação evidenciando a progressão do trabalho realizado pela equipe.
- Melhor *feedback* sobre o efeito das ações dos aprendizes: o *feedback* às ações dos alunos é apresentado somente ao final da simulação, impossibilitando que os alunos possam rever suas atitudes e mudar seu comportamento durante a simulação.

3. DESIGMPS

O DesigMPS (CHAVES et al., 2011) é um jogo de computador que visa apoiar o ensino da MPS, por meio do reforço de conceitos pertinentes e do exercício de MPS. Em termos de objetivos de aprendizagem, o jogo foca os níveis cognitivos de conhecimento, compreensão e aplicação de acordo com a taxonomia revisada de objetivos educacionais de Bloom (ANDERSON; KRATHWOHL, 2001).

Ao final da aplicação do jogo espera-se que os alunos tenham maior capacidade de: 1) nomear os elementos de processo de software (atividade, papel, artefato e ferramenta), lembrar seus respectivos usos, assim como lembrar as regras de modelagem – nível de conhecimento; 2) reconhecer e descrever os elementos do processo de software e seus relacionamentos de dependência (e.g. start_to_start, end_to_start), a partir de uma descrição textual do processo - nível de compreensão e; 3) modelar um processo de software usando a SPIDER_ML em um contexto de melhoria de processos do MPS.BR – nível de aplicação.

O jogo tem como público-alvo estudantes de cursos de graduação ou profissionais já atuantes no mercado. O jogo foi projetado para ser jogado individualmente durante aulas presenciais com duração máxima de 90 minutos.

Como tentativa do DesigMPS superar as limitações do *ModelBuilder*/SIMSE supracitadas, as seguintes estratégias foram usadas: 1) o uso de LMPS gráfica, SPIDER_ML, como ferramenta de apoio à modelagem; 2) a usabilidade é centrada no aluno, não no especialista em modelagem; e 3) não sobrecarrega a atividade de modelagem com variáveis de simulação.

3.1. Game flow

No jogo o aluno assume o papel de um engenheiro de processos que deve modelar um processo de software com base no modelo MR-MPS-SW, usando a SPIDER_ML. Para tal, o DesigMPS conduz o aluno em atividades de modelagem de processos, sendo estruturado em quatro estágios em ordem crescente de dificuldade: A, B, C e D (Figuras 9-12).

O DesigMPS é baseado nos trabalhos de Anohina, Graudina e Grundspenkis (2007); Anohina, Strautmane, Grundspenkis (2010) sobre avaliação adaptativa de conhecimento usando mapas conceituais (MC). Esta avaliação ocorre por meio de 6 tarefas em nível crescente de dificuldade. Os estágios A, B, C e D do DesigMPS são equivalentes às tarefas nº 1, nº 2, nº 4 e nº 6 propostas por Anohina, Strautmane e Grundspenkis (2010). As tarefas nº 3

e nº 5 não foram implementadas porque elas eram muito similares as tarefas nº 2 e nº 6, respectivamente.

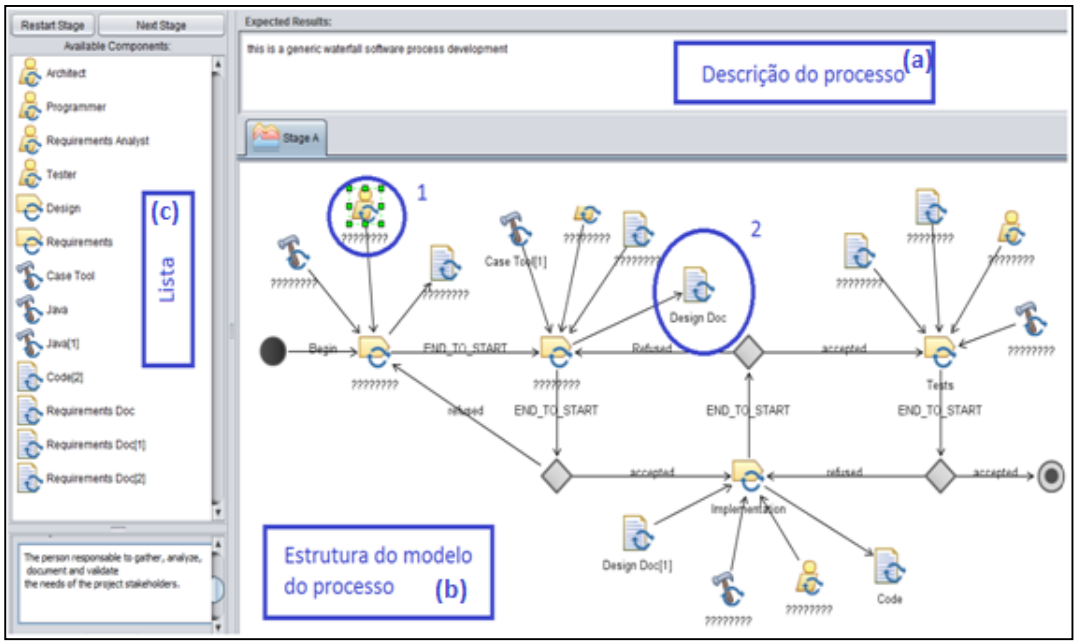
Para que o aluno possa completar os estágios, o DesigMPS fornece as seguintes informações apresentadas na Figura 9: a) uma descrição textual de um processo de software do MR-MPS-SW; b) a estrutura do modelo do processo, esta é a configuração de como os elementos estão relacionados, por meio de associações e transições; e c) uma lista com os elementos e os relacionamentos do processo.

O aluno deve modelar o processo colocando os elementos e relacionamentos da lista (Figura 9.c) em suas posições corretas na estrutura do modelo de processo (Figura 8.b). Os elementos e os relacionamentos da lista são extraídos de um modelo de gabarito (modelo de referência), criado por um especialista em MPS e baseado no MR-MPS-SW.

Nos estágios mais fáceis é fornecida ao aluno a estrutura do modelo de processo com uma maior quantidade de elementos preenchidos e relacionamentos rotulados nas suas posições corretas. Esta é a abordagem *fill-in-the-map* (estágios A, B e C). Na medida em que os estágios vão se tornando mais difíceis, é fornecida uma quantidade cada vez menor de elementos e relacionamentos preenchidos em suas posições na estrutura. No estágio mais difícil, os alunos precisam definir o modelo de processo por eles mesmos, do princípio, sem o auxílio da estrutura do modelo do processo. Esta é a abordagem *construct-the-map* (estágio D).

A Figura 8 mostra o mais fácil, o estágio A: é fornecida a estrutura do modelo do processo com todos os relacionamentos rotulados (*END_TO_START*, *accepted*, *refused*) e alguns elementos preenchidos nas suas posições corretas. Os elementos que devem ser preenchidos pelos alunos estão marcados com “???????”. Nesse exemplo, os elementos preenchidos são: Design Doc, Code e Tests.

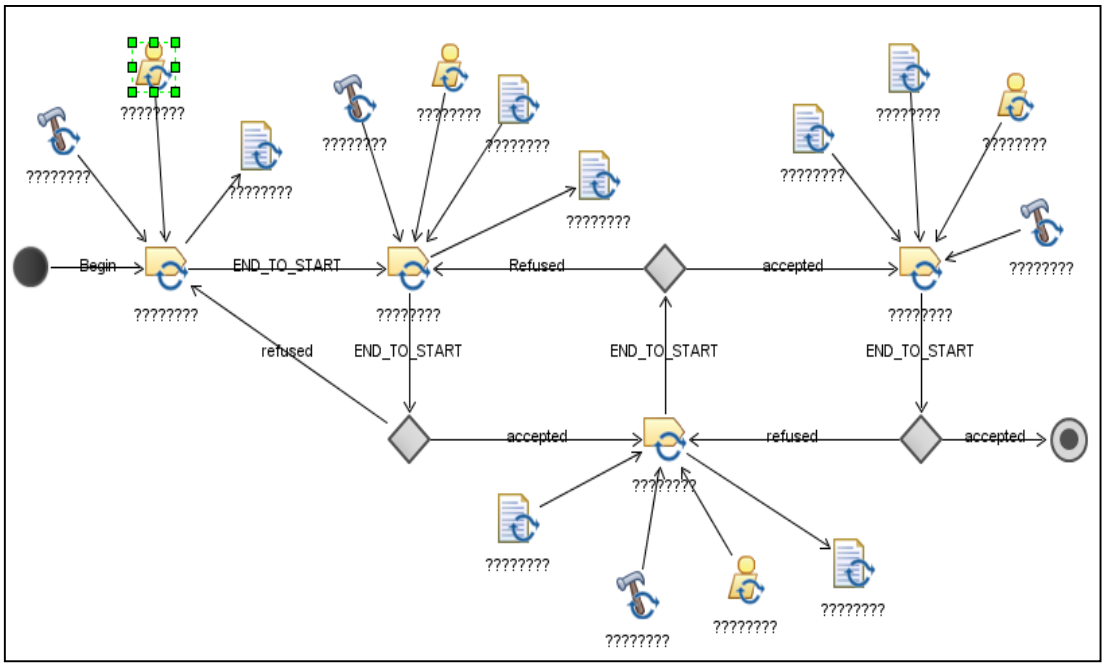
Figura 6. Estrutura do modelo do estágio A apresentando 4 elementos preenchidos e todos os relacionamentos rotulados



Fonte: Elaborada pelo autor (2015)

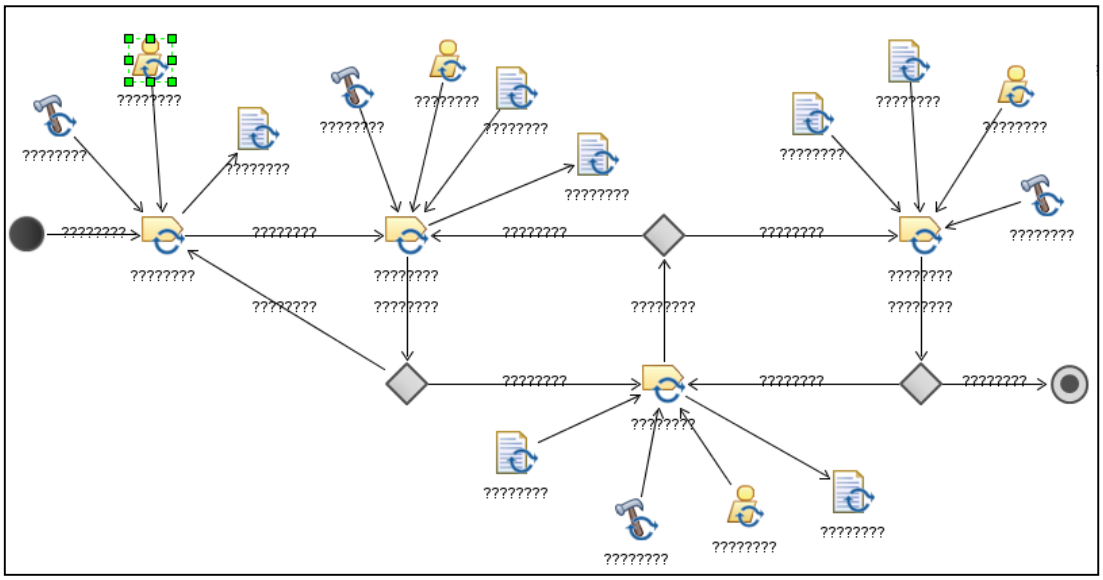
A Figura 10 mostra o estágio B: é fornecida a estrutura do modelo do processo com todos os relacionamentos rotulados corretamente, porém todos os elementos estão marcados com “???????”, requerendo o preenchimento por parte do aluno. A Figura 11 mostra o estágio C: em que é fornecida a estrutura do modelo do processo. No entanto, não é disponibilizada nenhuma informação acerca dos elementos e relacionamentos, estes devem ser preenchidos pelo aluno, tanto relacionamentos e elementos estão marcados com “???????”. Por fim, a Figura 12 mostra o estágio D, o mais difícil, pois nele a estrutura do processo não é fornecida, portanto, os alunos devem criar o modelo do princípio, sem nenhuma informação *a priori* sobre a estrutura do processo.

Figura 7. Estágio B: apresentando todos os elementos marcados como "??????" e os relacionamentos rotulados



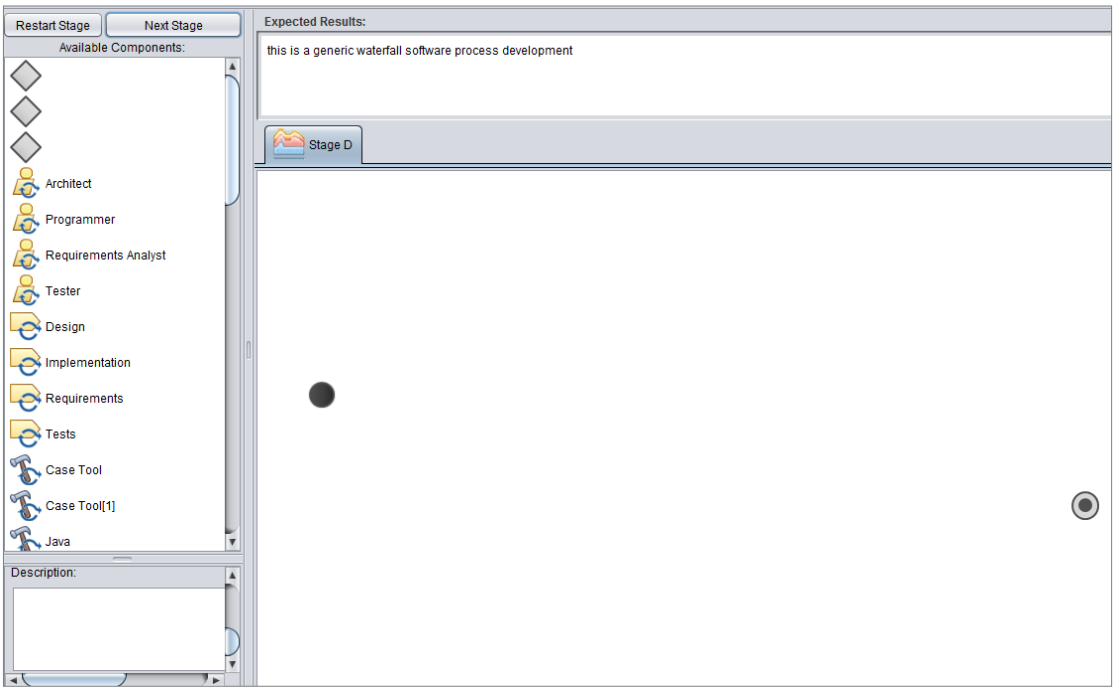
Fonte: Elaborada pelo autor (2015)

Figura 8. Estágio C: nenhum elemento preenchido ou relacionamento rotulado estão disponíveis na estrutura



Fonte: Elaborada pelo autor (2015)

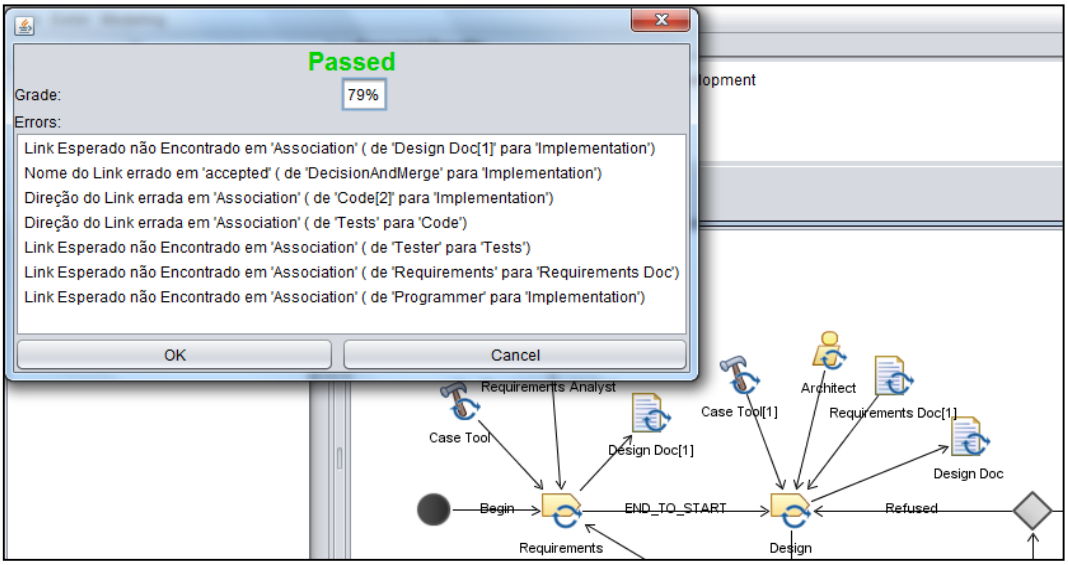
Figura 9. Estágio D: nenhuma parte da estrutura está disponível



Fonte: Elaborada pelo autor (2015)

A comparação entre o modelo do aluno e o modelo de gabarito (modelo de referência) é realizada ao final de cada estágio, de acordo com critérios de similaridades (ver Tabela 4), gerando uma pontuação de forma automática. A pontuação – diferenças nos critérios de similaridade entre os modelos do aluno e de referência -, juntamente com a indicação dos erros constituem o *feedback* aos alunos (ver Figura 13). Se a pontuação for maior do que um limiar mínimo, o aluno passa para o próximo estágio, caso contrário, permanece no mesmo.

Figura 10. Pontuação e a indicação dos erros como feedback aos alunos no final de cada estágio



Fonte: Elaborada pelo autor (2015)

A Tabela 2 sumariza os estágios e as respectivas informações disponíveis para os alunos.

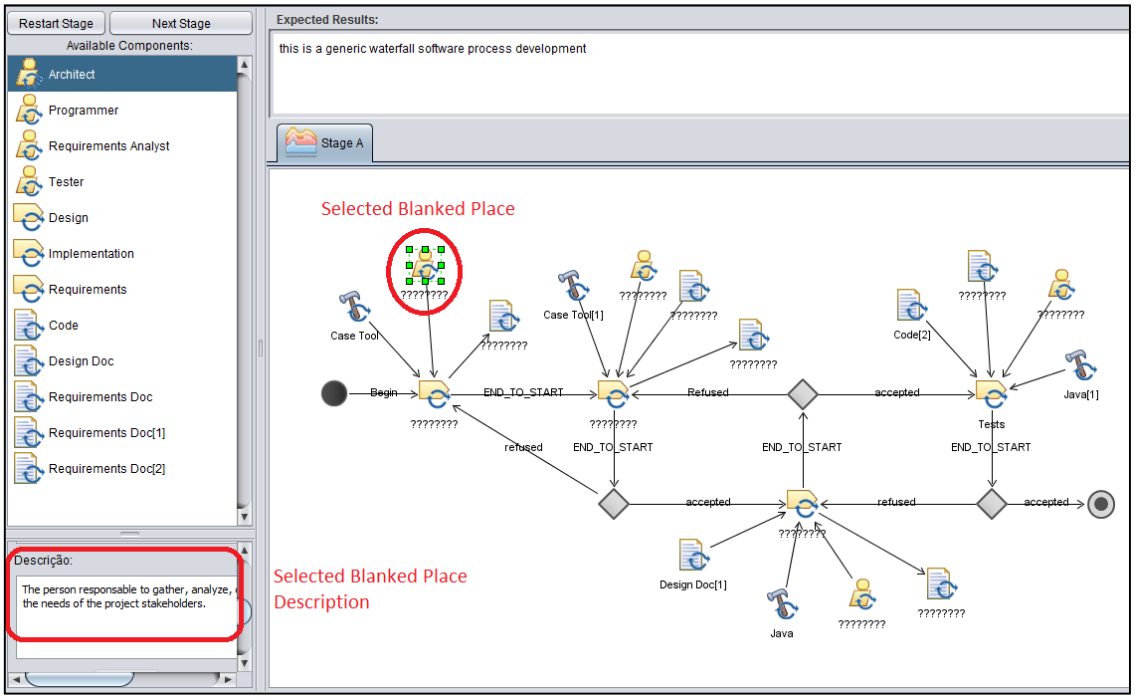
Tabela 2. Sumário dos componentes dos estágios

	Estrutura do modelo de processo		
Estágio	Vazio	Com alguns elementos preenchidos	Com todos os relacionamentos rotulados
A		X	X
B			X
C	X		
D			

Fonte: Elaborada pelo autor (2015)

Para os estágios A, B e C, o DesigMPS fornece uma ajuda opcional para os alunos: quando clicarem em alguma posição na estrutura do processo em que um elemento não está preenchido, ao mesmo tempo, o frame inferior esquerdo mostra a descrição do elemento. Esta descrição é uma breve declaração sobre a função de elemento no processo de software, como mostrado na Figura 14.

Figura 11. Seleção de uma posição não preenchida e sua descrição

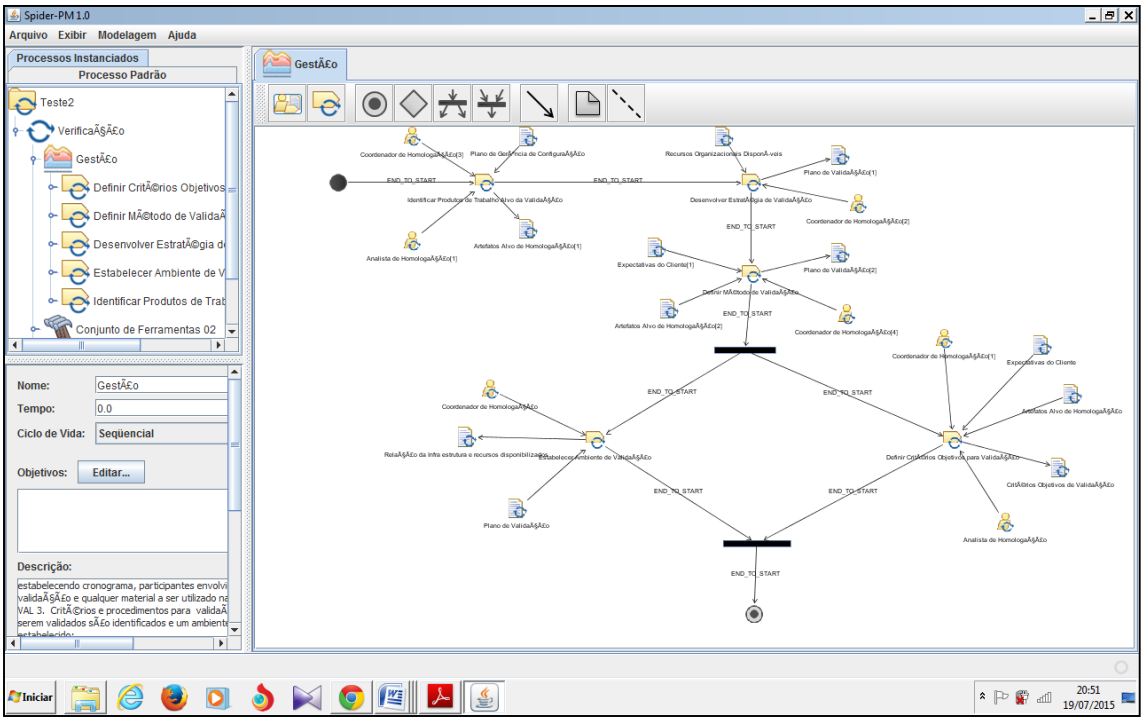


Fonte: Elaborada pelo autor (2015)

3.2. Criar modelos de processos de software de referência

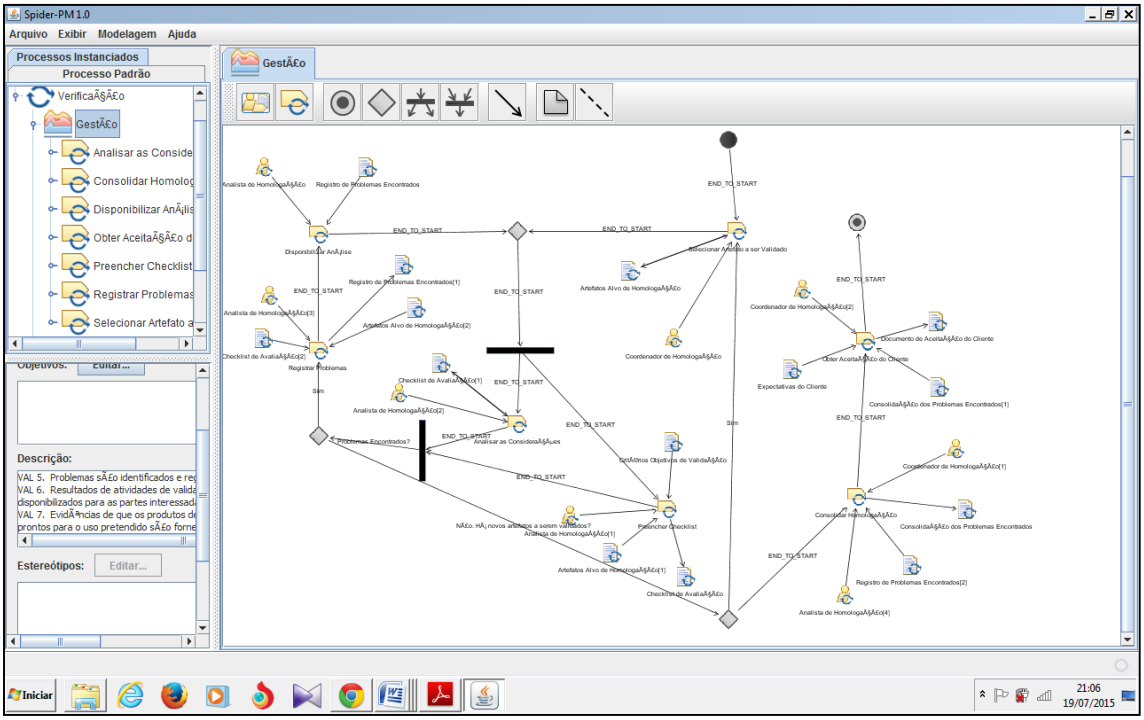
Existe uma versão modificada do SPIDER_PM para que o especialista modele os processos de software de referência. Essa versão permite estabelecer relacionamentos de transição entre as tarefas e gerar um arquivo XML compatível com o DesigMPS. A Figura 12 e a Figura 13 mostram os modelos de referência usados nas atividades de aprendizagem durante o experimento executado nesse trabalho. Estas atividades foram a intervenção do grupo experimental.

Figura 12. Primeiro modelo de referência usado na atividade de aprendizagem para o grupo experimental



Fonte: Elaborada pelo autor (2015)

Figura 13. Segundo modelo de referência usado na atividade de aprendizagem para o grupo experimental

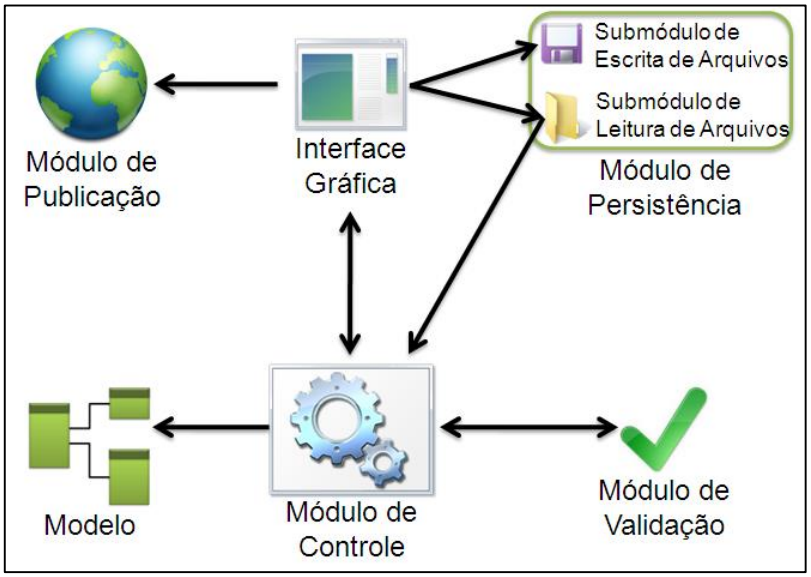


Fonte: Elaborada pelo autor (2015)

3.3. Desenvolvimento do DesigMPS

O DesigMPS tomou com base para o seu desenvolvimento o código e a arquitetura do SPIDER_PM. Desta forma, os Módulos de Persistência e o Modelo foram reutilizados integralmente, enquanto que houve adaptações dos Módulos de Controle, Interface Gráfica e Validação. A Figura 17 exibe a arquitetura do Spider_PM, que segue o modelo arquitetural *Model-View-Controller*.

Figura 14. Arquitetura da ferramenta SIDER_PM



Fonte: SPIDER (2010)

3.3.1. Requisitos

No Quadro 1 são apresentados os requisitos funcionais do jogo em alto nível de abstração.

Quadro 1. Descrição dos requisitos funcionais do DesigMPS em alto nível de abstração

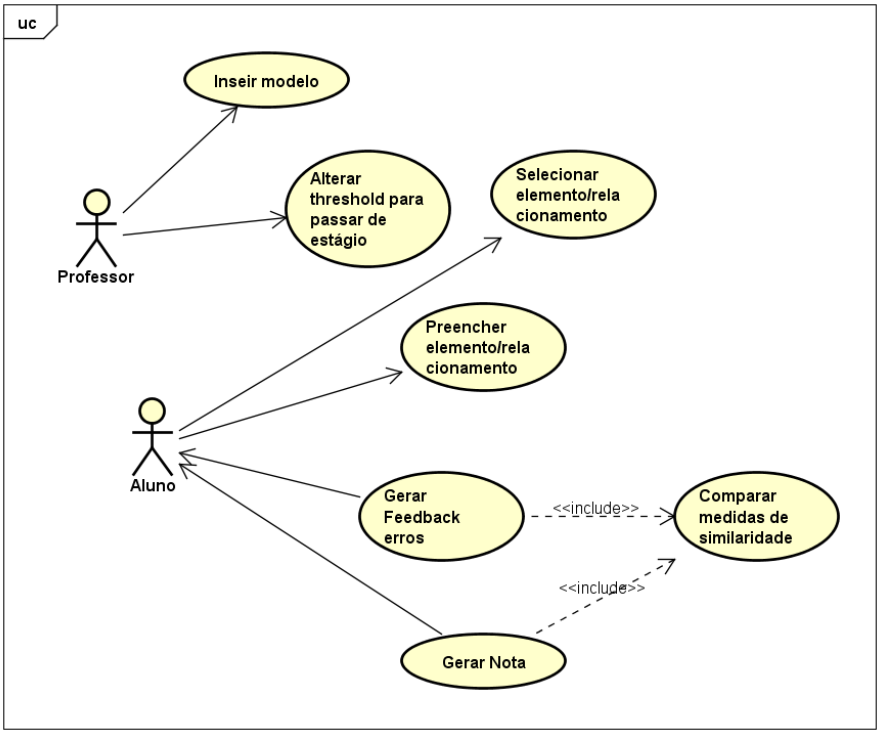
Nº	Requisito
1	A interface do jogo deve apresentar ao aluno uma descrição textual do processos de software, a lista de elementos e relacionamentos que compõem o processo de software, e também a estrutura do processo de software quando a atividade for do tipo *fill-in-the-map*.
2	O jogo deve carregar um arquivo criado no SPIDER_PM que contenha a descrição textual e o modelo de processo de software que será usado em uma partida
3	Avaliar, baseado em critérios de similaridade, quais os relacionamentos e

	posicionamentos dos elementos estão corretos, e retornar uma nota.
4	Retornar *feedback* indicando quais relacionamentos e elementos posicionados estão errados e quais seriam os esperados corretamente.
7	Possibilitar que a nota mínima para passar de estágios seja alterada
8	Adaptar os elementos e relacionamentos de MPS para os elementos e relacionamentos de MC.
9	Adaptar a avaliação de MC para o contexto de MPS.
10	O jogo deve ter 4 níveis de dificuldade (estágios) que são equivalentes as tarefas nº 1, nº 2, nº 4 e nº 6, propostas por Anohina, Strautmane e Grundspenkis (2010).
11	O estágio 1 do jogo apresenta a estrutura completa do processo, ocultando aleatoriamente o nome de quatro quintos dos elementos do processo e exibindo todos os relacionamentos rotulados .
12	O estágio 2 apresenta a estrutura completa do processo, sem fornecer o nome de todos os elementos do processo, mas exibindo todos os relacionamentos rotulados.
13	O estágio 3 apresenta a estrutura completa do processo, mas ocultando o nome de todos os elementos e relacionamentos sem rótulos.
14	O estágio 4 não oferece estrutura do processo. O aluno deve realizar a modelagem *from scratch*

Fonte: Elaborada pelo autor (2015)

A Figura 15 apresenta o modelo de caso de uso com as interações entre os atores e o jogo

Figura 15. Modelo de caso de uso do DesigMPS



Fonte: Elaborada pelo autor (2015)

O Quadro 2 descreve os atores, e o Quadro 3 descreve cada caso de uso de acordo com os seguintes campos: nome do caso de uso, sumário, ator primário, atores secundários, precondições, fluxo principal, fluxo alternativo, pós-condições.

Quadro 2. Descrição dos atores do modelo de caso do uso do DesigMPS

#Ator	Nome do ator	Descrição do ator
Ator-01	Professor	Responsável pela disciplina em que o jogo será aplicado
Ator-02	Aluno	Pessoas que estão cursando à disciplina e usarão o jogo

Fonte: Elaborada pelo autor (2015)

Quadro 3. Descrição dos casos de uso do DesigMPS

Nome do caso de uso	Inserir modelo
Sumário	O professor abre no jogo o arquivo do modelo que foi gerado no SPIDER_PM
Ator primário	Professor
Atores secundários	
Precondições	Modelo deve estar criado, O arquivo deve estar salvo
Fluxo principal	**1.** Selecionar o arquivo do modelo a partir de um gerenciador de arquivos

	2. Abrir o modelo no jogo
Fluxo alternativo	**1.** No passo 2 do fluxo principal, caso exista algum erro no arquivo: **a.** Deve retornar um aviso especificando o erro **b.** O arquivo não deve ser aberto
Pós-condições	As seguintes informações devem ser apresentadas na interface do jogo: descrição textual, lista de elementos e relacionamentos, e a estrutura do modelo de processo
Nome do caso de uso	Alterar *threshold* da nota
Sumário	O professor pode alterar o valor mínimo que o aluno deve obter para passar para o próximo estágio
Ator primário	Professor
Atores secundários	
Precondições	A partida não deve ter sido iniciada
Fluxo principal	**1.** O professor deve digitar o valor em porcentagem entre o intervalo fechado de 0% e 100%
Fluxo alternativo	**1** No passo 1 do fluxo principal, caso o valor inserido esteja fora do intervalo: **a.** Deve retornar um aviso requisitando para redigitar o valor no intervalo correto **b.** Permitir que o valor seja redigitado
Pós-condições	O *threshold* deve ter o valor digitado
Nome do caso de uso	Selecionar elemento/relacionamento
Sumário	O aluno deve selecionar a partir de uma lista os elementos/relacionamentos que compõem o processo
Ator primário	Aluno
Atores secundários	
Pré-condições	O arquivo do modelo deve estar aberto
Fluxo principal	**1** O aluno deve selecionar o elemento/relacionamento. **2** Deve arrastar e soltar o elemento até alguma posição na estrutura do modelo (para os estágios A, B, C) ou arrastar e soltar em algum lugar no frame inferior direito da interface do jogo (para o estágio D).
Fluxo alternativo	**1** No passo 2 do fluxo principal, caso o elemento/relacionamento seja arrastado e soltado para fora da estrutura do processo (válido somente para o estágios A, B, C): **a.** O elemento/relacionamento deve retornar para a lista, ficando disponível para ser selecionado
Pós-condições	Elemento/relacionamento selecionado
Nome do caso de uso	Preencher elemento/relacionamento
Sumário	O aluno deve preencher o elemento/relacionamento em alguma posição que seja do mesmo tipo (ex. um ator da lista só deve ser preenchido em uma posição equivalente de um ator).

Ator primário	Aluno
Atores secundários	
Precondições	O elemento/relacionamento deve ter sido selecionado, arrastado e solto em uma posição da estrutura do processo.
Fluxo principal	**1** O elemento/relacionamento é solto em alguma posição. **2** É avaliado se aquele elemento/relacionamento é do mesmo tipo da posição. **3** O elemento/relacionamento fica preenchido na posição. **4** O elemento/relacionamento é subtraído da lista.
Fluxo alternativo	**1** No passo 2 do fluxo principal, caso o elemento/relacionamento seja preenchido em uma posição de tipo diferente (válido somente para o estágios A, B, C): **a.** O elemento/relacionamento deve retornar para a lista, ficando disponível para ser selecionado
Pós-condições	O elemento/relacionamento fica preenchido em alguma posição do mesmo tipo na estrutura do processo
Nome do caso de uso	Comparar medidas de similaridade
Sumário	O modelo criado pelo aluno é comparado com o modelo do arquivo inserido pelo professor, por meio de medidas de similaridades.
Ator primário	Aluno
Atores secundários	
Pré-condições	Todos os elementos/relacionamentos devem estar preenchidos em alguma posição da estrutura do processo, a lista deve estar vazia
Fluxo principal	**1** executar MSL1 **2** executar MSL2 **3** executar MSL3 **4** executar MSL4 **5** executar MSL5 **6** executar MSL6 7 executar MSL7
Fluxo alternativo	
Pós-condições	Retorno das condições satisfeitas por MSL1 Retorno das condições satisfeitas por MSL2 Retorno das condições satisfeitas por MSL3 Retorno das condições satisfeitas por MSL4 Retorno das condições satisfeitas por MSL5 Retorno das condições satisfeitas por MSL6 Retorno das condições satisfeitas por MSL7
Nome do caso de uso	Gerar *feedback* erros

Sumário	São apresentados os erros cometidos pelo aluno.
Ator primário	Aluno
Atores secundários	
Pré-condições	Todos os elementos/relacionamentos devem estar preenchidos em alguma posição da estrutura do processo, a lista deve estar vazia; Retorno das condições satisfeitas por MSL1 Retorno das condições satisfeitas por MSL2 Retorno das condições satisfeitas por MSL3 Retorno das condições satisfeitas por MSL4 Retorno das condições satisfeitas por MSL5 Retorno das condições satisfeitas por MSL6 Retorno das condições satisfeitas por MSL7
Fluxo principal	**1** Apresentar quais condições não foram satisfeitas pelas medidas de similaridade **2** Dar opção de cancelar ou continuar
Fluxo alternativo	
Pós-condições	Todos os erros cometidos pelo aluno devem estar listados
Nome do caso de uso	Gerar nota
Sumário	A nota é calculada baseada no retorno das condições satisfeitas pelas medidas de similaridade
Ator primário	Aluno
Atores secundários	
Precondições	Todos os elementos/relacionamentos devem estar preenchidos em alguma posição da estrutura do processo, a lista deve estar vazia; Retorno das condições satisfeitas por MSL1 Retorno das condições satisfeitas por MSL2 Retorno das condições satisfeitas por MSL3 Retorno das condições satisfeitas por MSL4 Retorno das condições satisfeitas por MSL5 Retorno das condições satisfeitas por MSL6 Retorno das condições satisfeitas por MSL7
Fluxo principal	**1** Apresentar a nota obtida pelo aluno **2** Caso nota >= threshold **3** Passar para o próximo estágio
Fluxo alternativo	**1** No passo 2 do Fluxo principal, caso nota <= threshold a. Permanecer no mesmo estágio
Pós-condições	Nota disponível

Fonte: Elaborada pelo autor (2015)

3.3.2. Tecnologias

O JAVA foi a linguagem de programação escolhida, pois é a utilizada no SPIDER_PM. Também foram mantidas duas bibliotecas: a XStream (Disponível em: http://xstream.codehaus.org/), responsável pela persistência dos dados em formato XML por realizar a conversão de objetos para esse formato; e a JGraph (Disponível em: http://www.igraph.com), que permite a manipulação dos objetos gráficos que compõem os elementos do processo de software.

3.4. Adaptação do IKAS para o contexto de avaliação de modelos de processo de software

A adaptação IKAS para o contexto da avaliação de modelos de processo de software é uma das principais atividades para o desenvolvimento do DesigMPS, pois permitirá que o modelo do aluno seja avaliado usando regras estabelecidas comparando com o modelo de gabarito desenvolvido pelo especialista. Essa adaptação diz principalmente respeito aos requisitos 8 e 9.

Para realizar a adaptação das tarefas dos MC de Anohina, Graudina e Grundspenkis (2007) e Anohina, Strautmane e Grundspenkis (2010) para os estágios do DesigMPS, estabeleceram-se dois requisitos:

1. Estabelecer equivalência entre os elementos do MC com elementos do SPIDER_ML, e equivalência entre os relacionamentos do MC com os relacionamentos do SPIDER_ML. A Tabela 3 apresenta as equivalências (requisito 8).

2. Definir medidas de similaridades locais (MSL) e geral (SG) para comparação do modelo do aluno com o gabarito, resultando na pontuação. A tabela 4 discorre as medidas (requisito 9).

As MSL2, MSL3, MSL4, MSL5, MSL6, MSL7 derivam do padrão 2 de (ANOHINA; GRAUDINA; GRUNDSPENKIS, 2007), pois verificam, comparando com o modelo de referência, se: artefatos, papéis, ferramentas, artefatos, barras de separação/junção e decisão estão ligados pelas associações e transições incorretas, e independente da posição dos elementos (padrão 2). A avaliação DesigMPS não faz distinção de importância entre os seus elementos, por isso, nesse caso os outros padrões de Anohina, Graudina e Grundspenkis

(2007) não foram utilizados. O MSL 1 somente adota os padrões 1 e 2:) tarefas, barra de separação/junção e decisão estão ligados pelas transições corretas, e todos os elementos estão em seus lugares próprios (padrão 1); ou tarefas, barra de separação/junção e decisão estão ligados pelas transições incorretas (padrão 2). O MLS 1 serve para assegurar que a ordem de procedência entre as atividades seja avaliada, pois as posições das atividades devem estar corretas, assim como também a transição deve estar correta para que a ordem de precedência seja respeitada.

Tabela 3. Equivalências entre os elementos da SPIDER_ML dos elementos de mapas conceituais

Elementos de MC	Elementos da SPIDER_ML
Conceito	• Tarefa • Papel • Artefato • Ferramenta • Barra de junção • Barra de separação • Decisão
Relacionamento	• Associação • Transição

Fonte: Elaborada pelo autor (2015)

Tabela 4. Medidas de similaridades locais

Identificador	Descrição	Peso	Intervalo de valores	Cálculo
MSL1	Verificar se todas as transições entre as tarefas de origem e destino estão corretas, garantindo a ordem de precedência entre as tarefas.	P1	[0,1]	Número de transições entre tarefas corretas no modelo resposta dividido pelo número de transições entre tarefas do modelo gabarito.
MSL2	Verificar se as transições entre as tarefas são do tipo correto.	P2	[0,1]	Número de transições com o tipo correto no modelo resposta dividido pelo número de transições com o tipo correto do modelo gabarito.
MSL3	Verificar se as transições entre as tarefas e as barras de separação/junção estão corretas.	P3	[0,1]	Número de transições entre tarefas e as barras corretas no modelo resposta dividido pelo número de transições entre tarefas e as barras do modelo

				gabarito.
MSL4	Verificar se as transições entre as tarefas e as decisões estão corretas.	P4	[0,1]	Número de transições entre tarefas e as decisões corretas no modelo resposta dividido pelo número de transições entre tarefas e as decisões do modelo gabarito.
MSL5	Verificar se os papéis estão associados corretamente às tarefas deles.	P5	[0,1]	Número de papéis associados corretamente às tarefas do modelo resposta dividido pelo número de papéis associados do modelo gabarito.
MSL6	Verificar se as ferramentas estão associadas corretamente às tarefas delas.	P6	[0,1]	Número de ferramentas associadas corretamente as tarefas no modelo resposta dividido pelo número de ferramentas associadas do modelo gabarito.
MSL7	Verificar se os artefatos de entrada/saída estão associados corretamente às tarefas correspondentes.	P7	[0,1]	Número de ferramentas associadas corretamente as tarefas no modelo resposta/ número de ferramentas associadas modelo referência.

Fonte: Elaborada pelo autor (2015)

Depois de realizar o cálculo das similaridades locais, é calculada a similaridade global (SG), que define a nota obtida pelo aluno no presente estágio. O cálculo consiste na multiplicação dos pesos pelos valores obtidos por cada similaridade que, somados, são divididos pelo número total de similaridades existentes no modelo (NMSL), como demonstra a Equação (1). Os pesos podem ser ajustados de acordo com as necessidades educacionais deliberadas pelos professores.

$$SG = \sum_{i=1}^{NMSL} (MSLi \times PesoMSLi) / (NMSL) \qquad (1)$$

3.5. Comparação entre as funcionalidades do DesigMPS e Knowledge Assessment System - KAS

Existem sistemas que realizam avaliações adaptativas usando MC. Desses sistemas o que mais se aproxima ao proposto por Anohina, Graudina e Grundspenkis (2007) e Anohina, Strautmane e Grundspenkis (2010) é o protótipo da 4ª versão Knowledge Assessment System (KAS) (GRUNDSPENKIS; ANOHINA, 2009). A Tabela 5 apresenta a comparação entre o DesigMPS e a 4ª versão do protótipo do KAS.

Tabela 5. Comparação entre as funções do 4º protótipo do KAS e as do DesigMPS

	DesigMPS	**4º protótipo do KAS**
Criação de MC		
Relacionamentos com pesos diferentes	Não	Sim
Semântica dos relacionamentos	Sim	Sim
Relacionamentos direcionados	Sim	Sim
Definição de sinônimos dos rótulos dos relacionamentos	Não	Sim
Definição de sinônimos dos conceitos	Não	Sim
Uso de rótulos padrões para os relacionamentos	Não	Não
Feedback fornecido aos alunos		
Máxima pontuação da fase atual	Não	Sim
Pontuação atual do aluno da fase atual	Sim	Sim
Pontuação adicional em comparação com o estágio anterior	Não	Sim
MC dos alunos com os pontos para cada relacionamento	Não	Sim
MC do professor	Não	Sim
Checagem das proposições	Sim	Sim
Feedback fornecido ao professor		
MC dos alunos com erros em destaque	Não	Sim
Pontuação máxima para a fase corrente	Não	Sim
Pontuação dos alunos	Não	Sim
Informações estatísticas sobre as diferenças entre os MC dos professores e dos alunos	Não	Sim
Tracking dos erros cometidos pelos alunos	Sim	Não

Fonte: Elaborada pelo autor (2015)

Algumas diferenças encontradas na Tabela 5 são motivadas para atender as particularidades da modelagem e da avaliação de modelos de processo de software, o que não representa desvantagens em relação à versão 4 do protótipo do KAS. Por exemplo, a característica “relacionamentos com pesos diferentes” não é implementada no DesigMPS, pois se considera que todos os relacionamentos em um processo de software são equitativamente de mesma importância, não havendo pesos diferentes para eles.

4. AVALIAÇÃO

Após o desenvolvimento do jogo e a criação do modelo de processos de software sobre o processo de validação do MR-MPS-SW, um estudo empírico foi planejado e executado para identificar evidência da eficácia de aprendizado do jogo quando aplicado em um contexto de uma disciplina que tem uma unidade instrucional em que MPS é ensinada . O estudo também procurou capturar a percepção dos alunos em relação a sua adequação em termos de relevância de conteúdo, suficiência, grau de dificuldade, sequência e, método de ensino, atratividade, e para identificar os pontos fortes e pontos fracos; além de comparar a sua eficácia de aprendizagem com a atividade instrucional de uma Aprendizagem Baseada em Projetos (ABP).

O método instrucional ABP foi utilizado nesse experimento, pois o interesse da sua aplicação na educação em engenharia é crescente (HADIM; ESCHE, 2002). Além de ser uma das formas mais efetivas para que os alunos sejam participantes ativos no aprendizado de "design" (DYM et al., 2005). No ABP, os alunos devem realizar tarefas que levam a produção de um produto, que pode ser um projeto, um modelo, um dispositivo ou uma simulação (RÜÜTMANN; KIPPER, 2011). Normalmente ABP são combinados com aulas expositivas, e focam na aplicação e integração de conhecimentos adquiridos anteriormente (MILLS; TREAGUST, 2004).

A ABP para o experimento consistiu de uma questão solicitando que o aluno modelasse um processo usando o SPIDER_PM, a partir de uma descrição textual de um cenário no contexto de melhoria de processos de software do MR-MPS-SW.

Os objetivos do estudo experimental estão de acordo com as avaliações do jogo definidas nesse estudo, e descritos usando o GQM – *Goal, Question, Metric* (BASILI; CALDIERA; ROMBACH, 1994):

Objetivo do estudo experimental 1:

Analisar o DesigMPS para avaliar sua eficácia de aprendizagem sobre o conhecimento de MPS, do ponto de vista do aluno nos níveis cognitivos de conhecimento, compreensão e aplicação. E então comparar com a eficácia de aprendizagem de uma ABP aplicada em uma unidade instrucional de MPS que faz parte de uma disciplina de Qualidade de Processos de Software em cursos de graduação em Ciência da Computação e Sistemas de Informação.

Objetivo do estudo experimental 2:

Analisar o DesigMPS, com o propósito de avaliá-lo em relação à adequação em termos de relevância de conteúdo, suficiência, sequência, grau de dificuldade e, método de ensino; atratividade; pontos fortes e pontos fracos, do ponto de vista do aluno, no contexto de uma unidade instrucional sobre modelagem de processo de software em uma disciplina de Qualidade de Processos de Software, em cursos de graduação em Ciência da Computação e Sistemas de Informação.

4.1. Estratégia de pesquisa e avaliação

O desenho de um experimento formal foi aplicado (grupos controle e experimental randomizados com pré-teste/pós-teste) para comparar e avaliar a eficácia das atividades de aprendizagem (COHEN; MANION; MORRISON, 2000), permitindo uma comparação estatística da mudança observada no grupo experimental em relação com a observada no grupo controle (CAMPBELL; STANLEY, 1963). O desenho foi: (1) os alunos assistiram a aulas expositivas sobre (i) modelos de capacidade/maturidade de processos de software (MCMPS); (ii) os processos VER (Verificação), VAL (Validação) e GQA (Garantia da Qualidade) do MR-MPS-SW; e (iii) conceitos e prática de MPS, com treinamento no uso do SPIDER_PM e DesigMPS; (2) para reduzir o viés de seleção, a distribuição dos participantes nos Grupos A (experimental) e B (controle) foi randomizada por sorteio, formando grupos balanceados, como evidenciado pela análise de dados dos fatores de confusão (ver Apêndice III) – *background* pessoal (DF.1) e motivação (DF.2) – assim, para propósitos estatísticos, ambos os grupos são considerados equivalentes; (3) todos os alunos fizeram os mesmos pré-testes antes da aplicação das intervenções (atividades de aprendizagem); (4) as intervenções foram aplicadas. O grupo A jogou DesigMPS e o grupo B completou uma atividade de ABP; e (5) ao final, os grupos A e B fizeram os mesmos pós-testes.

O desenho geral do experimento é descrito na Tabela 6.

Tabela 6. Sumário do desenho do experimento

Grupos	Aulas		Pré-teste	Intervenções	Pós-teste
Grupo A Grupo B	Aulas expositivas e treinamento em SPIDER_PM e DesigMPS	Alocação aleatória dos grupos	Testes para os níveis de conhecimento, compreensão e aplicação	DesigMPS ABP-Modelagem com SPIDER_PM	Testes para os níveis de conhecimento, compreensão e aplicação

Fonte: Elaborada pelo autor (2015)

As notas do pré/pós-testes foram disponibilizadas para os alunos ao final do experimento. Os pré/pós-testes e as atividades de aprendizagem serão detalhadas na seção de Instrumentação e se encontram no Apêndice I.

4.2. Questões de pesquisa, hipóteses, formulações e variáveis

Os objetivos do estudo experimental, as respectivas questões de pesquisa, as variáveis, a formulação, as hipóteses nulas e os instrumentos são mostrados nas Tabelas 7 e 8. As variáveis Y.1, Y.2 e Y.3 foram adaptadas de (ARMITAGE; KELLNER, 1994), e correspondem a outros trabalhos sobre MPS em termos de elementos (papel, ferramenta, tarefa e artefato) e relacionamentos (associação e dependência) necessários para representar processos de software (ACUÑA et al., 2000), (FUGGETTA, 2000), (CURTIS; KELLNER; OVER, 1992). Esses elementos e relacionamentos também são parte do SPEM 2.0 (OMG, 2008) e do SPIDER_ML (SPIDER, 2009). Para esse estudo, os fatores de confusão usados em wohlin et al. (2012) foram adaptados: DF.1, *background* pessoal em termos de treinamento e experiência acadêmica/profissional ; DF.2, motivação em termos de importância do MPS e interesse em aprender mais sobre o assunto. As respostas de DF.2 têm um formato baseado em uma escala Likert de 5 pontos entre -2 e 2 (-2 discordo fortemente, -1 discordo, 0 neutro, 1 concordo, 2 concordo fortemente).

Tabela 7. Questões de pesquisa, variáveis, formulações e hipóteses do objetivo do experimento 1

Objetivo do estudo experimental 1	
Questão de pesquisa 1: Jogar DesigMPS aumenta a eficácia de aprendizagem de MPS no nível de aplicação?	
Hipótese 1 H01**:** Não existe diferença entre as pontuações médias do pré/pós-testes no Grupo A (experimental)	
Variáveis dependentes	
Y.1. Elementos do processo: elementos que compõem o processo (tarefas, papéis, artefatos e ferramentas) na descrição textual que devem estar presentes no modelo.	Pontuação parcial do pré e pós testes $\in [0,1]$
Y.2. Descrição dos elementos do processo: a descrição correta dos elementos do processo de Y.1	Pontuação parcial do pré e pós testes $\in [0,1]$
Y.3. Relacionamentos entre os elementos do processo: relacionamentos corretamente definidos entre os elementos de processo de Y.1.	Pontuação parcial do pré e pós testes $\in [0,1]$

Y.4. Uso correto do SPIDER_ML para representar Y.1 e Y.3	Pontuação parcial do pré e pós-testes ∈ [0,1]

Formulação : pos(Mpos; A) > pre(Mpre; A),
Onde:

1) A = Grupo A

2) As pontuações do pré e pós-testes para cada aluno são, respectivamente:

$LSpre_{\text{j}} = \frac{100(Y_1+Y_2+Y_3+Y_4)}{4}$ onde *j*= aluno do Grupo A;

$LSpos_{\text{j}} = \frac{100(Y_1+Y_2+Y_3+Y_4)}{4}$ onde *j*= aluno do Grupo A.

3) Média das pontuações dos estudantes nos pré e pós-testes, respectivamente:

$Mpre = \frac{\sum_{j=1}^{m} LSpre_j}{m}$ onde *m* = número de estudantes no Grupo A;

$Mpos = \frac{\sum_{j=1}^{m} LSpos_j}{m}$ onde *m* = número de estudantes no Grupo A.

Instrumentos: Pré e pós-testes

Questão de pesquisa 2: Como é a eficácia de aprendizagem de MPS com o DesigMPS comparada com a do ABP, nos níveis de conhecimento, compreensão e aplicação?

Hipótese 2a

H02a: Não existirá diferença nas pontuações do pré-teste entre os grupo A e grupo B (os dois grupos selecionados aleatoriamente irão ter as mesmas habilidades) no nível de aplicação.

Variáveis dependentes

Y.1, Y.2, Y.3 e Y.4 e suas respectivas pontuações parciais, como descrito para a Questão de Pesquisa 1

Formulação 2a: pre(Mpre; A) > pre(Mpre; B),

Onde:

1) A ou B = Grupo A ou B

2) A pontuação do pré-teste de cada aluno respectivamente:

$LSpre_{\text{j}} = \frac{100(Y_1+Y_2+Y_3+Y_4)}{4}$ onde *j*= aluno do Grupo A ou B.

3) Média do pré-teste dos alunos respectivamente:

$Mpre = \frac{\sum_{j=1}^{m} LSpre_j}{m}$ onde *m* = número de alunos do Grupo A ou B.

Hipótese 2b

H02b: Não existirá diferença entre as pontuações dos pós-testes entre o grupo A e o grupo B (os dois grupos ainda serão igualmente capazes após as atividades de aprendizagem) no nível de aplicação.

Variáveis dependentes

Y.1, Y.2, Y.3 e Y.4 e suas respectivas pontuações parciais, como descrito para a Questão de Pesquisa 1

Formulação b: pos(Mpos; A) > pos(Mpos; B),

Onde:

1) A ou B = Grupo A ou B

2) A pontuação do pós-testes de cada aluno respectivamente:

$LSpos_{\text{j}} = \frac{100(Y_1+Y_2+Y_3+Y_4)}{4}$ onde *j*= aluno do Grupo A ou B.

3) Média da pontuação do pós-teste dos alunos respectivamente:

$Mpos = \frac{\sum_{j=1}^{m} LSpos_j}{m}$ onde *m* = número de alunos no Grupo A ou B.

Instrumentação: Pré/pós-testes

Hipótese 2c

H02c: Não existirá diferença entre as pontuações dos pré-testes dos Grupo A e Grupo B (Os dois grupos selecionados aleatoriamente terão habilidades iguais) no nível de conhecimento

Variáveis dependentes

Y.5 média da pontuação de 10 questões ∈ [0,10]
Hipótese 2d H02d: Não existirá diferença entre as pontuações dos pós-testes entre o Grupo A e o Grupo B (os dois grupos ainda serão igualmente capazes após as atividades de aprendizagem) no nível de conhecimento.
Variáveis dependentes Y.5 média da pontuação de 10 questões ∈ [0,10]
Instrumentação: Pré e pós-testes
Hipótese 2e H02e: Não existirá diferença entre as pontuações do pré-testes dos Grupo A e Grupo B (Os dois grupos selecionados aleatoriamente terão habilidades iguais) no nível de compreensão.
Variáveis dependentes Y.6 média da pontuação de 5 questões ∈ [0,10]
Hipótese 2f H02f: Não existirá diferença entre as pontuações dos pós-testes entre o Grupo A e o Grupo B (os dois grupos ainda serão igualmente capazes após as atividades de aprendizagem) no nível de compreensão.
Variáveis Dependentes Y.6 média da pontuação de 5 questões ∈ [0,10]
Instrumentos: Pré e pós-testes

Fonte: Elaborada pelo autor (2015)

Tabela 8. Questões de pesquisa, variáveis, formulações e hipóteses do objetivo do experimento 2

Objetivo do estudo experimental 2			
Questão de Pesquisa 3	O jogo é considerado adequado em termos de relevância de conteúdo, suficiência, grau de dificuldade, sequência, e método de ensino para o qual ele foi desenvolvido?		
	Variáveis dependentes	Y.7 Adequação em termos de: Y.7.1 Relevância de conteúdo: o conteúdo de ensino do DesigMPS é relevante Y.7.2 Suficiência: jogar o DesigMPS é suficiente para apoiar o ensino da modelagem de processos de software Y.7.3 Grau de dificuldade: cada nível testou progressivamente o conhecimento sobre modelagem de processos de software Y.7.4 Sequencia: Houve aumento coerente do nível de dificuldade entre os estágios do DesigMPS Y.7.5 Método de Ensino: a abordagem para ensinar MPS pelo DesigMPS é adequado	{discordo fortemente -2, discordo -1, neutro 0, concordo 1 e concordo fortemente 2}
	Formulação	Distribuição de frequência (Y.7.1) Distribuição de frequência (Y.7.2) Distribuição de frequência (Y.7.3) Distribuição de frequência (Y.7.4)	
	Instrumento	Questionário sobre a adequação e atratibilidade do DesigMPS (ver a Tabela 9).	

Questão de pesquisa 4	O jogo é atrativo?		
	Variável Dependente	Y.8 atratividade	{discordo fortemente -2, discordo -1, neutro 0, concordo 1 e concordo fortemente 2}
	Formulação	Distribuição de frequência (Y.8)	
	Instrumento	Questionário sobre a adequação e atratividade do DesigMPS (veja a tabela 9).	
Questão de pesquisa 5	Quais são os pontos fortes e pontos fracos do jogo?		
	Variáveis Dependentes	Y.9 Pontos fortes Y.10 Pontos fracos	
	Instrumento	Duas questões abertas	

Fonte: Elaborada pelo autor (2015)

4.3. Instrumentação

Os pré/pós-testes anônimos foram usados para coletar dados relevantes para as questões de pesquisa 1 e 2. Esses testes estão disponíveis do Apêndice I. No nível de aplicação, os testes consistiram de uma única questão pedindo aos alunos modelarem processos de software baseados em MR-MPS-SW (pré-teste, Figura 18). Os testes foram similares em conteúdo e grau de dificuldade e foram corrigidos por dois especialistas e um verificador. A pontuação desse nível foi calculada por uma escada de Percentage-Mastery (ANGOFF, 1984). Os níveis de conhecimento e compreensão foram avaliados por pré/pós-testes com questões idênticas respectivamente para cada nível. As pontuações de conhecimento (Y.5) e compreensão (Y.6) foram a média aritmética das questões corretas. Nenhuma informação relacionada aos participantes estava contida nos testes. As pontuações dos pré/pós-testes foram disponibilizadas para os participantes somente ao fim do experimento.

Figura 16. Pré-teste usado no experimento

Modele um processo de software alinhado ao VER 1, VER 2 e VER 3 (Processo de Verificação do Nível D do MR-MPS-SW) usando a SPIDER_ML. Identifique os elementos de processo de software (papel, artefato, tarefa e ferramenta) e faça a respectiva descrição, ou seja, a sua característica e importância do processo. Defina os relacionamentos entre os elementos e a ordem de precedência (end_to_start, start_to_start, start_to_end) entre as tarefas.

Fonte: Elaborada pelo autor (2015)

O objetivo das questões de pesquisa 3, 4 e 5 é a obtenção de uma avaliação subjetiva da adequação do jogo em termos de relevância de conteúdo (Y.7.1), suficiência (Y.7.2), grau de dificuldade (Y.7.3), sequência (Y.7.4), método de ensino (Y.7.5), atratividade (Y.8), bem como os pontos fortes (Y.9) e os pontos fracos (Y.10) a partir do ponto de vista dos alunos. Essas variáveis foram selecionadas a fim de ajudar a identificar eventuais falhas no *design* do jogo, por exemplo, aspectos ausentes, não abordados que os alunos consideram importantes a serem incluídos. No entanto, os alunos que ainda estão aprendendo sobre MPS podem não ser capazes de fornecer um *feedback* valioso sobre essas questões e / ou ser tendenciosos. Devido a essas ameaças à validade, essas questões também foram analisadas por especialistas em ES.

Os dados das questões de pesquisa 3 e 4 foram coletados por meio do questionário sobre adequação em relação a relevância de conteúdo, suficiência, grau de dificuldade, sequência, método de ensino e atratividade. Este questionário é constituído de 6 itens, sendo que cada item era referente a uma variável das questões de pesquisa. As respostas desses itens têm um formato baseado em uma Escala Likert de 5 pontos, em que as alternativas variam entre discordo fortemente (-2) e concordo fortemente (2) (ver Apêndice IV).

Para obter *feedback* sobre os pontos fortes (Y.9) e os pontos fracos (Y.10) do jogo, foram aplicadas duas questões abertas aos participantes do grupo A. Em relação aos fatores de distúrbio, eles foram coletados por meio de um questionário de background pessoal (DF.1) com 13 itens e um questionário de motivação (DF.2.1 importância em aprender modelagem de processo de software; DF2.2 interesse em aprender mais sobre MPS), ver Apêndice II.

4.4. Intervenções (atividades de aprendizagem)

O Grupo A jogou DesigMPS e o Grupo B usou o SPIDER_PM para modelar a atividade do ABP. Ambas as intervenções (atividades de aprendizagem) foram realizadas simultaneamente em duas sessões de 90 minutos, ao longo de dois dias de aula. Nos primeiros dez minutos da primeira sessão o instrutor explicou as atividades de aprendizagem, e explicou que não havia competição entre os participantes. Ambos os grupos realizaram as atividades de aprendizagem a partir da descrição textual do processo de Validação (VAL) do MR-MPS-SW.

4.5. Execução

O experimento foi realizado em 2013 paralelamente em duas turmas da disciplina Qualidade de Processos de Software (EM-05137 Tópicos Especiais em Engenharia de Software) que faz parte do currículo dos cursos de bacharelado em Sistemas de Informação (24 alunos no experimento) e bacharelado em Ciência da Computação (17 alunos no experimento). Ambos os cursos pertencem à Faculdade de Computação da Universidade Federal do Pará. A disciplina tem exatamente o mesmo currículo, mesma carga horária, ensinada pelo mesmo professor e apoiada pelos mesmos mentores. O objetivo da disciplina é ensinar conceitos chave sobre qualidade de produto de software, e também, melhoria de processo de software por meio de MCMPS. A disciplina inclui uma unidade instrucional sobre MPS (ver Figura 19). Os participantes do experimento foram alunos voluntários.

A agenda do experimento foi: dia 1, aplicação do questionário de *background* pessoal e motivação (10 minutos), em seguida aulas expositivas sobre Modelos de Capacidade/Maturidade de Processos de Software - MCMPS (80 minutos); dia 2, aulas expositivas sobre os processos VER (Verificação), VAL (Validação) e GQA (Garantia da Qualidade) do MR-MPS-SW (90 minutos); dia 3, aulas sobre MPS e SPIDER_ML (90 minutos); dia 4, aulas de laboratório para treinamento sobre SPIDER_PM (45 minutos) e DesigMPS (45 minutos); dia 5, os pré-testes foram administrados; dia 6, a apresentação das intervenções (atividades de aprendizagem) (10 minutos) e os alunos realizaram as atividades de aprendizagem (DesigMPS para o grupo A e o ABP para o grupo B); dia 7, houve uma repetição da realização das intervenções (90 minutos); e finalmente, no dia 8, os pós-testes foram aplicados.

Figura 17. Esboço da unidade instrucional sobre modelagem de processos de software

Objetivo Instrucional: que os alunos consigam modelar processos de software no contexto do MR-MPS-SW, usando a LMPS SPIDER_ML.

Pré-requisitos: Os alunos devem conhecer conceitos básicos de engenharia de software, de processo de software e MR-MPS-SW. Eles tipicamente têm pouca ou nenhuma experiência com SPM.

Descrição da unidade instrucional: conceitos e terminologia de modelagem de processos de software; relevância da modelagem de processos de software no contexto dos MCMPS: CMMI-DEV e MR-MPS-SW; o SPIDER_ML e processo de MPS.

Objetivos de desempenho: Os alunos devem ser capazes de ter uma compreensão básica sobre SPM, e de modelar processos de software no contexto do MR-MPS-SW usando a SPIDER_ML. Como resultados dessa unidade instrucional, os alunos devem ser capazes de:

- Identificar e descrever os elementos de processo de software e seus relacionamentos a partir de uma descrição textual;
- Desenvolver um modelo que represente o processo analisado usando a SPIDER_ML.

<table>
<tr><td rowspan="3">Modelos de capacidade/maturidade de processos de software</td><td rowspan="3">Processos VER, VAL e GQA do MR-MPS-SW</td><td></td><td colspan="3">Modelagem de processos de software</td><td rowspan="3">Métodos de avaliação MA-MPS</td></tr>
<tr><td>Conteúdo</td><td>Conceitos, terminologias e LMPS; SPIDER_ML: sintaxe e semântica.</td><td>Aplicação dos processos de software</td><td>Unidade de avaliação</td></tr>
<tr><td>Estratégia Instrucional</td><td>Aulas expositivas e exemplos de modelagem de processos de software em aulas de laboratório</td><td>Aplicação do DesigMPS e ABP</td><td>Atividades de modelagem de processo de software</td></tr>
</table>

Fonte: Elaborada pelo autor (2015)

O experimento foi conduzido seguindo a agenda apresentada na Tabela 9, durante 3 semanas consecutivas.

Tabela 9. Agenda do experimento

Dia	Conteúdo	Instrumentos	Duração
1	Aulas sobre modelos de capacidade/maturidade de processos de software	Questionário sobre o background pessoal e motivação (DF.1 e DF.2)	10min 80min
2	Aulas sobre os processos: VER, VAL e GQA do MR-MPS-SW		90min
3	Aulas sobre MPS e SPIDER_ML		90min
4	Aulas de laboratório para treinamento sobre SPIDER_PM e DesigMPS	SPIDER_PM DesigMPS	45min (SPIDER_PM) 45min (DesigMPS)
5	Pré teste	Teste para os níveis de conhecimento, compreensão e aplicação	90min
6	Intervenção	DesigMPS (grupo A) e ABP (grupo B)	90min
7	Intervenção	DesigMPS(grupo A) e ABP (grupo B)	90min
8	Pós-teste	Teste para os níveis de conhecimento, compreensão e aplicação do questionário sobre adequação e atratividade (Y.7.1, Y.7.2, Y.7.3, Y.7.4, Y.7.5 e Y.8) (grupo A) perguntas sobre os pontos fortes (Y.9) e pontos fracos (Y.10) (grupo A)	90min 5min 10min

Fonte: Elaborada pelo autor (2015)

Ao final do pós-teste, os alunos que jogaram o DesigMPS (grupo A) preencheram o questionário sobre a adequação e atratividade do jogo, como mostrado na tabela 10. Eles responderam também duas questões abertas sobre os pontos fortes e os pontos fracos, e fizeram sugestão de melhorias.

Tabela 10. Estrutura do questionário sobre adequação atratividade e as respectivas variáveis e questões de pesquisa

Item	Variável	Questão de pesquisa
O conteúdo de ensino do DesigMPS é relevante	Y.7.1 Relevância de conteúdo	3
Jogar o DesigMPS é suficiente para apoiar o ensino da modelagem de processos de software	Y.7.2 Suficiência	3
Cada nível testou progressivamente o conhecimento sobre modelagem de processos de software	Y.7.3. Grau de dificuldade	3
Houve aumento coerente do nível de dificuldade entre os estágios do DesigMPS	Y.7.4. Sequencia	3
A abordagem para ensinar modelagem de processo de software pelo DesigMPS é adequado	Y.7.5 Método de ensino	3
O DesigMPS é atrativo	Y.8. Atratividade	4

Fonte: Elaborada pelo autor (2015)

O DesigMPS para esse experimento estava configurado com uma pontuação mínima de 50% de acertos de modelagem, para que o aluno pudesse prosseguir para o estágio seguinte. Tal valor foi estipulado em virtude da Universidade Federal do Pará ter em seu regimento que o aluno precisa ter no mínimo 50% de aproveitamento (equivalente ao conceito "Regular") para aprovação em uma disciplina.

5. ANÁLISE DOS DADOS

5.1. Procedimentos

Para a coleta de dados das variáveis de estudo descritas, utilizou-se o seguinte esquema:

- DF.1 e DF.2 foram extraídas a partir da aplicação de questionários de *background* pessoal e motivação no início do experimento;
- Y.1 a Y.6 foram coletados durante o pré e pós-testes;
- Y.7 a Y.10 foram coletadas ao final do pós-teste;

5.2. Análise das questões de pesquisa

O teste de normalidade *Shapiro-Wilk* (SHAPIRO; WILK, 1965) foi aplicado para verificar a distribuição das pontuações dos pré/pós-testes (pontuações dos testes no Apêndice V). Uma vez que os resultados deste teste para os níveis de compreensão e aplicação têm compatibilidade com uma distribuição de Gauss, o Teste t de *Student* foi aplicado. O resultado no nível de conhecimento não apresentou uma distribuição normal, de modo que o Teste U de *Mann–Whitney* foi aplicado. Ambos os testes de hipóteses tiveram nível de significância $\alpha = 0,05$.

5.2.1. Análise da questão de pesquisa 1

Para a questão de pesquisa 1: Jogar DesigMPS aumenta a eficácia de aprendizagem de MPS no nível de aplicação?, a Tabela 11 mostra a comparação entre o pré e o pós-teste usando o Teste t de *Student* bicaudal para amostras pareadas. O grupo A pontuou *57,99%±24.19* no pré-teste, com um ganho de *77,64%±10.26* no pós teste. A diferença entre as duas médias, o ganho médio *(Δ=19.65)*, indica um incremento no aprendizado, porque o p-valor*<0.0001** é estatisticamente significante. Então a hipótese nula (H01: Não existe diferença entre as pontuações médias dos pré/pós-testes no Grupo A) foi rejeitada para o grupo A.

Tabela 9. Eficácia de aprendizagem a partir da pontuação da modelagem de processo de software (%) – Nível de aplicação

Variáveis	Grupo A (Experimental)		Grupo B (Controle)	
	Pré	Pós	Pré	Pós
Tamanho da amostra	21	21	20	20
Mínimo	7.50	54.00	0.00	20.00
Máximo	95.00	100.00	97.50	86.50
Mediana	63.00	78.75	46.00	57.25
Primeiro quartile	46.25	72.50	37.81	47.13
Terceiro quartile	68.05	83.88	64.06	71.38
Média aritmética	57.99	77.64	48.89	58.25
Desvio padrão	24.19	10.26	22.71	18.00
Poder do teste	**0.9626**		**0.4202**	
p-value (pré × pós testes)	**<0.0001***		**0.0169***	

Fonte: Elaborada pelo autor (2015)

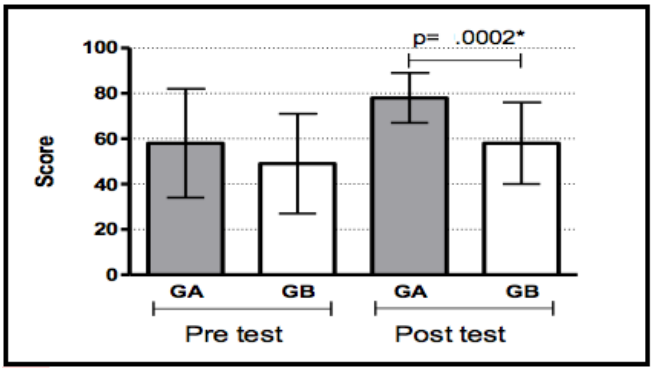
*Teste t de *Student* bicaudal para amostras pareadas para comparar as pontuações do pré/pós-teste para cada grupo.

5.2.2. Análise da questão de pesquisa 2

As hipóteses H02a e H02b foram combinadas para analisar a questão de pesquisa 2 (Como é a eficácia de aprendizagem de MPS com o DesigMPS comparada com a do ABP nos níveis de conhecimento, compreensão e aplicação?) para o nível de aplicação. No pré-teste (ver Tabela 11), a comparação entre o grupo A (GA) (57,99%±24.19) e o grupo B (GB) (48,89%±22.71) foi realizada pelo Teste de t *Student* bicaudal para amostras independentes. O resultado é um p-valor=0.2223 que não é estatisticamente significativo, não existindo desse modo, evidência suficiente para rejeitar a hipótese nula H02a: Não existirá diferença nas pontuações do pré-teste entre o grupo A e o grupo B (os dois grupos selecionados aleatoriamente irão ter as mesmas habilidades) no nível de aplicação, como descrito na Figura 20.

No pós-teste (ver Tabela 11), a comparação entre GA (77,64%±10.26) e GB (58,25%±18.00) o Teste de t *Student* bicaudal para amostras independentes resulta em um p-valor=0.0002* estatisticamente significativo, rejeitando-se a hipótese nula H02b: Não existirá diferença entre as pontuações dos pós-testes entre o grupo A e o grupo B (os dois grupos ainda serão igualmente capazes após as atividades de aprendizagem) no nível de aplicação. Assim, concluiu-se que existe uma diferença real entre os grupos, a pontuação de GA é maior do que a pontuação de GB por um Δ=19.39. Pode-se afirmar que GA foi melhor no pós-teste, como descrito na Figura 18.

Figura 18. Média e desvio padrão do GA (n=21) e GB (n=20) nos pré/pós-testes no nível de aplicação



Fonte: Elaborada pelo autor (2015)

As hipóteses H02c e H02d foram combinadas para analisar a questão de pesquisa 2 (Como é a eficácia de aprendizagem de MPS com o DesigMPS comparada com a do ABP nos níveis de conhecimento, compreensão e aplicação?) para o nível de conhecimento. No pré-teste, a comparação entre GA e GB foi realizada pelo Teste U de *Mann–Whitney* bicaudal para amostras independentes, que resultou em p-valor=0.2677 que não é estatisticamente significante, assim não existe evidência suficiente para rejeitar a hipótese nula H02c: Não existirá diferença entre as pontuações do pré-testes do grupo A e do grupo B (os dois grupos selecionados aleatoriamente terão habilidades iguais) no nível de conhecimento, como descrito na Tabela 12.

No pós-teste, na comparação entre GA e GB, o Teste U de Mann–Whitney bicaudal para amostras independentes resulta em um *p-value*=0.8043 que não é estatisticamente significativo, assim não existe evidência suficiente para rejeitar a hipótese nula H02d: Não existirá diferença entre as pontuações dos pós-testes entre o Grupo A e o Grupo B (os dois grupos ainda serão igualmente capazes após as atividades de aprendizagem) no nível de conhecimento, como descrito na Tabela 12.

Tabela 10. Comparação da eficácia de aprendizagem nas pontuações da modelagem de processo de software - nível de conhecimento

Pré-teste	**N**	**Mediana**	**Soma dos postos**
GA	21	10	483.5
GB	20	10	377.5
p-value = 0.2677	U=167.50		
Pós-teste	**N**	**Mediana**	**Soma dos postos**
GA	21	10	431.5
GB	20	10	429.5
p-value= 0.8043	U= 200.50		

Fonte: Elaborada pelo autor (2015)

As hipóteses H02e e H02f foram usadas para analisar a questão de pesquisa 2 (Como é a eficácia de aprendizagem de MPS com o DesigMPS comparada com a do ABP nos níveis de conhecimento, compreensão e aplicação?) no nível de compreensão. No pré-teste, a comparação entre o GA (6.97±1.83) e o GB (6.46±1.78) foi realizado pelo Teste de t *Student* bicaudal para amostras independentes, que resultou em um p-valor=0.3701 que não é estatisticamente significativo, assim não existe evidência suficiente para rejeitar a hipótese nula H02e: Não existirá diferença entre as pontuações do pré-teste dos grupos A e B (Os dois grupos selecionados aleatoriamente terão habilidades iguais) no nível de compreensão, como descrito na Tabela 13.

No pós-teste, a comparação entre GA (8.34±0.96) e GB (7.62±1.40) o Teste de t *Student* bicaudal para amostras independentes resulta em um p-valor=0.0578 que não é estatisticamente significativo, assim não existe evidência suficiente para rejeitar a hipótese nula H02f: Não existirá diferença entre as pontuações dos pós-testes entre o Grupo A e o Grupo B (os dois grupos ainda serão igualmente capazes após as atividades de aprendizagem) no nível de compreensão, como descrito na Tabela 13.

Tabela 11. Comparação da eficácia de aprendizagem nas pontuações da modelagem de processo de software - nível de compreensão

Variáveis	Grupo A Pré	Grupo B Pré	Grupo A Pós	Grupo B Pós
Tamanho da amostra	21	20	21	20
Média aritimética	6.97	6.46	8.34	7.62
Desvio padrão	1.83	1.78	0.96	1.40
Poder do teste	0.2300		0.6179	
p-value (pré × pós-teste)	0.3701*		0.0578*	

Fonte: Elaborada pelo autor (2015)

*Teste t de *Student* bicaudal para amostras independentes para comparar as pontuações dos pré/pós-testes entre os dois grupos

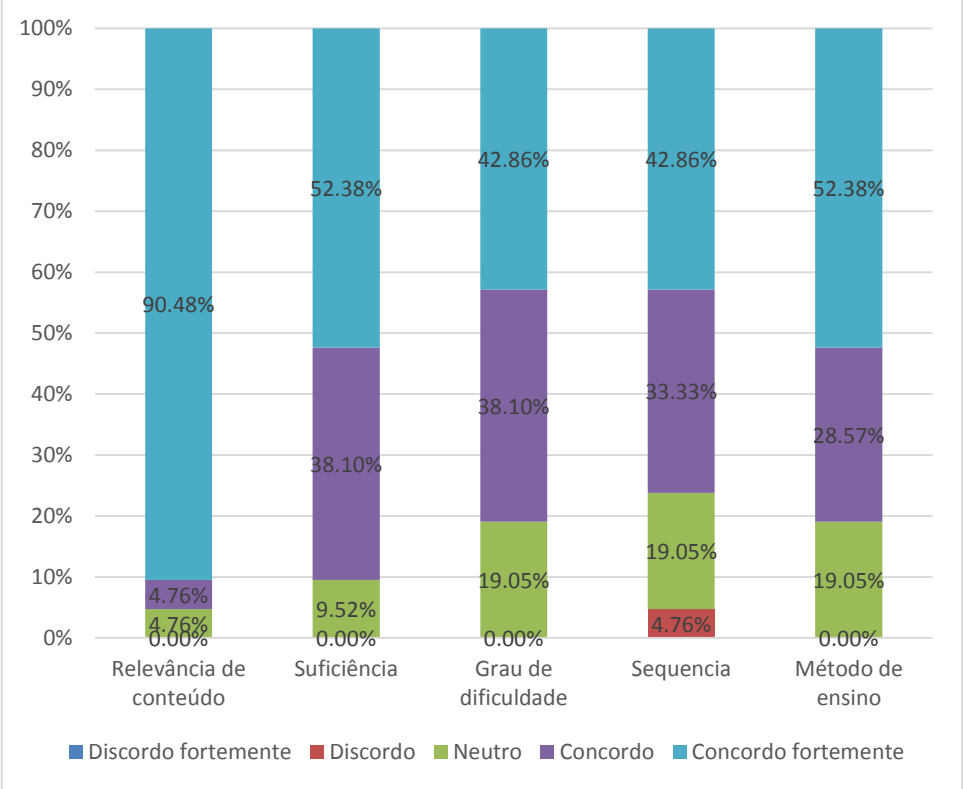
5.2.3. Análise de questão de pesquisa 3

Em consideração à questão de pesquisa 3: o DesigMPS é considerado apropriado em termos de relevância de conteúdo, suficiência, grau de dificuldade, sequência, e método de

ensino para o qual foi projetado? A avaliação obteve a resposta “concordo fortemente” como a mais frequente em todos os itens: Y.7.1 relevância de conteúdo (90.48%), Y.7.2 suficiência (52.38%), Y.7.3 grau de dificuldade (42.86%), Y.7.4 sequencia (42.86%) e Y.7.5 método de ensino (52.38%) (ver Figura 19). Para os itens da Y.7 o coeficiente de consistência interna Alfa de *Cronbach* (CRONBACH, 1951) foi considerado “bom” (Alfa de *Cronbach* = 0.817). O maior percentual de “concordo fortemente” para relevância de conteúdo (Y.7.1), seguido por suficiência (Y.7.2) e método de ensino (Y.7.5).

As dimensões com as menores porcentagens para “concordo fortemente” foram grau de dificuldade (Y.7.3) e sequencia (Y.7.4). Em que Y.7.4 apresentou a maior divergência de opiniões entre os participantes do GA.

Figura 19. A avaliação da adequação jogo em relação a relevância de conteúdo (Y.7.1), suficiência (Y.7.2), grau de dificuldade (Y.7.3), sequencia (Y.7.4), método de ensino (Y.7.5) no Grupo A (n=21) composto de alunos de Sistemas de Informação e Ciência da Computação. Belém-PA, Brasil, 2013.

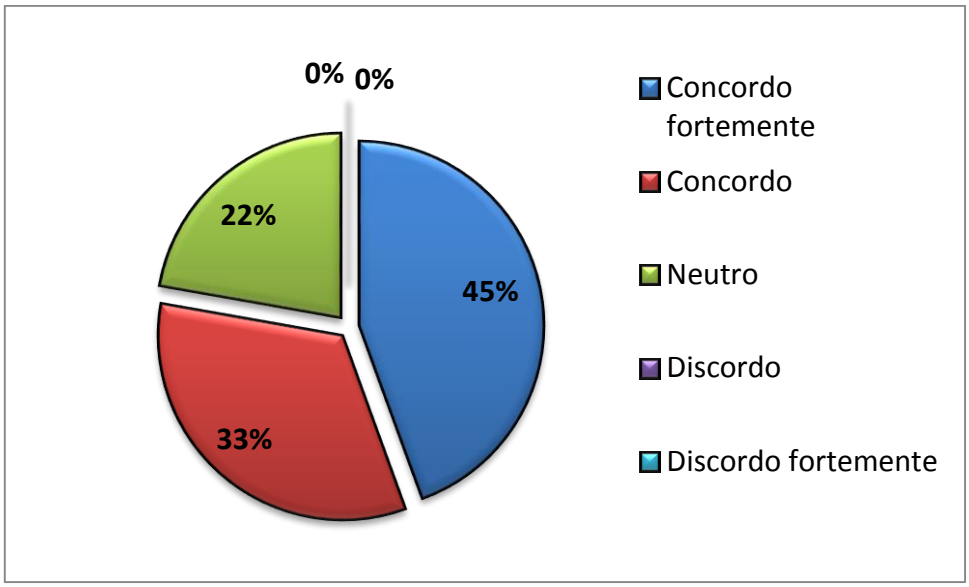


Fonte: Elaborada pelo autor (2015)

5.2.4. Análise da questão de pesquisa 4

Em relação à questão de pesquisa 4: O jogo é atrativo (Y.8)? A maioria dos alunos que jogou DesigMPS opinou concordo fortemente ou concordo (78%) (ver Figura 20), a distribuição mostra que a opinião mais frequente foi "concordo fortemente" (45,00%). Considerando a consistência interna dos itens Y.7 e Y.8, o Alfa de *Cronbach* =0,807 (Bom).

Figura 20. A avaliação da atratividade (Y.8) do jogo no Grupo A (n=21) composto por alunos de Sistemas de Informação e Ciência da Computação. Belém-PA, 2013



Fonte: Elaborada pelo autor (2015)

5.2.5. Análise da questão de pesquisa 5

Foi solicitada a opinião dos alunos do grupo A sobre (Y.9) pontos fortes e (Y.10) pontos fracos do jogo. A compilação dos pontos fracos (Y.10) está presente na Tabela 14, enquanto que os pontos fortes (Y.9) são mostrados na Tabela 15. Em ambas as tabelas estão descritos o motivo da opinião e o número de alunos que tiveram opiniões convergentes.

Tabela 12. Compilação dos pontos fracos (Y.10), motivos e número de estudantes

Pontos fracos	Motivos	Número de estudantes
Feedback dos erros	O *feedback* não é claro ou não explica o motivo do erro	5
Ausência de *help* e de material de consulta sobre	Necessita de *help* com instruções sobre o jogo e de material de consulta sobre MPS	6

MPS

Poucas fases	Jogos normalmente tem mais fases	3
Usabilidade	Necessidade de *scrolling* quando o modelo é grande	3
	Disposição inadequada dos rótulos dos relacionamentos	1
Desnível de dificuldade muito grande entre os estágios 3 e 4	O estágio 4 é muito mais difícil que o estágio 3	2
Efeito de decorar o modelo do processo	Nas últimas fases o aluno já pode ter decorado o modelo	1
Modelos grandes são cansativos e desmotivadores	Evitar modelos grandes, pois eles tornam o jogo cansativo e desmotivador	1
Não é relevante para quem trabalha com métodos ágeis	Crença de que SPM é desnecessária para quem trabalha com métodos ágeis	1
Restringir a criatividade	Completar modelos desenvolvidos por especialistas restringem a criatividade do aluno	1

Fonte: Elaborada pelo autor (2015)

Tabela 13. Compilação dos pontos fortes (Y.9), motivos, e números de estudantes

Pontos fortes	Motivo	Número de estudantes
Relevância da modelagem de processos de software	Ajuda a evidenciar a importância de se modelar processos de software	7
Facilidade de uso	A interface e interações são simples	6
Integração da teoria e prática	Ajuda a integrar a teoria estudada na disciplina com a prática	6
Modelo de referência ajuda a entender a modelagem	Analisar o modelo de referência dos estágios ajuda a entender o processo de modelagem	4
Aprendizado de forma divertida/desafiadora	Por ser um jogo, a aprendizagem foi divertida e desafiadora	3
Ter um modelo de referência ajuda os alunos a criarem seus próprios modelos	Analisar o modelo de referência dos estágios ajuda os alunos a criarem seus próprios modelos	2
Visualizar graficamente o processo é melhor do que apenas a descrição textual do mesmo	O mapeamento entre a descrição textual representação gráfica ajuda a compreender o modelo de processo de software	2
Algum auxílio no processo de modelagem é melhor do	As fases auxiliam que os alunos progressivamente completem a	2

que tentar modelar sozinho	modelagem.	
Aprendizado por repetição	O aprendizado pode ocorrer pela repetição da modelagem em níveis diferentes de dificuldade	1

Fonte: Elaborada pelo autor (2015)

A Tabela 16 apresenta quais são as propostas dos alunos quando foram requisitados a opinar sobre melhorias nas futuras versões do jogo.

Tabela 14. Melhorias citadas

Melhorias	Número de estudantes
O jogo deveria ser usado por mais tempo na disciplina	6
Mais modelos disponíveis para os alunos exercitarem	3
Permitir que o aluno criasse modelos que sejam avaliados independentemente de um modelo de referência	1
Feedback na momento em que o erro acontece e não ao final de cada fase	1
Permitir o que aluno possa consertar os erros antes de passar para a próxima fase	1

Fonte: Elaborada pelo autor (2015)

6. DISCUSSÃO

Os resultados do teste da hipótese H01 (Não existe diferença entre as pontuações médias do pré e do pós-teste no grupo A) dão um indicativo inicial de que o jogo tem uma eficácia de aprendizagem positiva no nível cognitivo de aplicação. Além disso, os resultados dos testes de H02a (Não existirá diferença nas pontuações do pré-teste entre os grupos A e B - os dois grupos selecionados aleatoriamente irão ter as mesmas habilidades - no nível de aplicação) e H02b (Não existirá diferença entre as pontuações dos pós-testes entre o grupo A e o grupo B - os dois grupos ainda serão igualmente capazes após as atividades de aprendizagem - no nível de aplicação) indicam que jogar DesigMPS tem uma eficácia de aprendizagem maior no nível de aplicação do que a eficácia causada pela atividade de ABP no nível de aplicação, uma vez que os ganhos médios dos pré-testes não foram estatisticamente significativos e os ganhos médios no pós-teste foram estatisticamente significativos. Os resultados H01 e H02b podem ser atribuídos em parte ao fato do DesigMPS ser baseado em MC, este que é reconhecido por produzir resultados positivos no ensino de tópicos relacionados a computação (CALVO et al., 2011; YU; KLEIN, 2008).

O efeito de aprendizagem do jogo também foi comparado com ABL nos níveis de conhecimento e compreensão. Os resultados dos testes de H02e (Não existirá diferença entre as pontuações dos pré-testes do grupo A e do grupo B - os dois grupos selecionados aleatoriamente terão habilidades iguais - no nível de compreensão) e H02f (Não existirá diferença entre as pontuações dos pós-testes entre o grupo A e o grupo B - os dois grupos ainda serão igualmente capazes após as atividades de aprendizagem - no nível de compreensão) mostram que não houve diferenças estatisticamente significativas entre os dois grupos nos níveis de compreensão. Por outro lado, houve uma diferença estatisticamente significativa entre os pré/pós-testes de GA, uma vez que foi observada uma diferença entre as duas médias: ganho médio ($\Delta = 1{,}37$) e de p-valor = 0,0001, por meio de Teste de t *Student* bicaudal para amostras pareadas. Houve também uma diferença estatisticamente significativa entre o pré/pós-teste de GB, uma vez que foi observada uma diferença estatisticamente significativa entre as duas médias, ganho médio ($\Delta = 1{,}16$) e p-valor = 0,0023, por meio de Teste de t *Student* bicaudal para amostras pareadas. Isto indica que ambos os métodos de ensino são igualmente capazes de melhorar o nível de compreensão.

Os resultados dos testes H02c (Não existirá diferença entre as pontuações dos pré-testes do grupo A e do grupo B - os dois grupos selecionados aleatoriamente terão habilidades iguais - no nível de conhecimento) e H02d (Não existirá diferença entre as pontuações dos pós

testes entre o grupo A e o grupo B - os dois grupos ainda serão igualmente capazes após as atividades de aprendizagem - no nível de conhecimento) não mostram diferenças estatisticamente significativas entre os dois grupos no nível de conhecimento. Isto pode ter ocorrido devido ao fato de que a média do pré-teste (Δ> 90%) já foi muito elevada em ambos os grupos, o que dá pouco espaço para pontuações melhores no pós-teste. Estas pontuações altas do pré-teste provavelmente podem ser atribuídas às aulas expositivas e aulas de laboratório usadas para ensinar conceitos MPS antes da aplicação das atividades de aprendizagem.

A pontuação mínima de acertos para passar de estágio pode ter uma grande influência no aprendizado, pois caso a pontuação seja demasiadamente baixa, o aluno pode não conseguir obter um grau de conhecimento real necessário para modelar processos de software. Porém, como pontuações muito altas podem ser difíceis de ser alcançadas em modelos de referência mais complexos, com isso, o aluno pode se desestimular com o jogo. Assim, outros resultados podem ser possíveis caso a pontuação mínima fosse diferente da estabelecida nesse experimento.

Concluindo, o DesigMPS é mais eficaz do que ABP ao nível de aplicação, e por si só conduz a uma melhoria no nível de compreensão.

Um resultado surpreendente foi que a maioria dos alunos consideraram que o DesigMPS é atrativo (Y.8) (45% concordam fortemente e 33% concordam), haja vista que o jogo não usa recursos sofisticados de gráficos ou sons, que normalmente estão presentes em jogos comerciais e que muitas vezes são considerados indispensáveis para que jogo seja atrativo. Acredita-se que esse resultado possa ter sido obtido, em parte, pela proposta e pelo objetivo educacional do jogo, visto que 7 alunos citaram como ponto forte do jogo ele apoiar o ensino de MPS. Mesmo com as limitações tecnológicas o jogo foi considerado uma forma divertida e desafiadora por alguns alunos.

Dois pontos fracos observados pelos alunos já eram esperados. A primeira limitação citada é que completar modelos desenvolvidos por especialistas podem restringir a criatividade; e segunda é que os alunos apenas “decoram” modelos prontos. Com o objetivo de diminuir a primeira limitação esperamos desenvolver um sistema de avaliação de modelos mais inteligente do que o atual, que envolva análise semântica. Em relação à segunda limitação, “decorar” realmente é um efeito colateral em virtude da repetição do jogo com os mesmos modelos de referência. Porém, isso pode ser útil quando se deseja educar alunos/profissionais no processo nos quais irão atuar.

Um ponto fraco atípico citado por um aluno foi que ele não considerou o jogo relevante para quem trabalha com métodos ágeis. Isto pode ser explicado pelo fato de que frequentemente existe a percepção de que as melhores práticas dos MCMPS e dos métodos ágeis são contrárias umas às outras (GLAZER, 2008).

O pouco número de estágios do DesigMPS e o grande desnível de dificuldade entre os estágios 3 e 4 foram considerados pontos fracos. O pequeno número de estágios foi motivado para que o tempo de uma partida fosse o suficiente para ser usado durante as aulas. Porém, alguns alunos conseguiram concluir os 4 estágios mais rapidamente, e sem mais estágios para competir, eles se desinteressaram pelo jogo. As futuras versões terão mais estágios entre os estágios 3 e 4 com o objetivo de diminuir o desnível entre elas, desta forma lidaremos com os dois pontos fracos de uma única vez.

6.1. Ameaças à validade

Existe cautela sobre a generalização dos resultados positivos obtidos no experimento. Avaliações de outros jogos para ensinar ES não conseguiram confirmar um efeito de aprendizagem significativo (DRAPPA; LUDEWIG, 2000). Duas críticas foram levantadas sobre os estudos que foram experimentalmente avaliados: a amostragem dos alunos não foi grande o suficiente (WANGENHEIM; THIRY; KOCHANSKI, 2009; NAVARRO; VAN DER HOEK, 2007; DRAPPA; LUDEWIG, 2000); e o jogo não foi utilizado como um recurso complementar para outros métodos de ensino (NAVARRO; VAN DER HOEK, 2007; DRAPPA; LUDEWIG, 2000). Neste trabalho foi tentado superar essas limitações: primeiro, pela formação de uma amostra que permitisse um resultado estatisticamente significativo e, portanto, a aplicação de teste de hipótese mais robusto (paramétrico); e, segundo, o DesigMPS foi usado como um recurso complementar para outros métodos educacionais (aulas expositivas e aulas de laboratório) em uma unidade instrucional sobre MPS.

Neste ponto, analisaremos quatro tipos de ameaças à validade de um experimento: interna, externa, construção e conclusão.

6.1.1. Validade interna

É o grau em que mudanças nas variáveis dependentes podem ser seguramente atribuídas às mudanças nas variáveis independentes. Algumas possíveis ameaças à validade interna do experimento são: os grupos podem se comportar de forma diferente como resultado de um fator de confusão; um efeito instrumentação ou; um efeito de maturação.

A divisão randomizada, por meio de sorteio, dos participantes no grupo experimental (A) e grupo de controle (B) gerou grupos estatisticamente equivalentes (ver Apêndice III), assim diminuindo a influência de fator de confusão.

Além disso, para reduzir o efeito de maturação, as pontuações individuais dos pré-testes foram publicadas somente no final do experimento. No entanto, o simples fato da realização do pré-teste pode ter tido um efeito de aprendizado positivo. Como todos os grupos realizaram testes similares, além das duas questões nas atividades de aprendizagem que foram semelhantes aos testes do nível de aplicação, e todo o experimento durando três semanas (um período relativamente longo), essa combinação de fatores pode ter causado um prejuízo no entusiasmo e motivação por parte de alguns participantes, e/ou um efeito de aprendizado positivo. A aprendizagem não planejada pode ter ocorrido simultaneamente a partir de atividades não relacionadas com o experimento. Para diminuir a influência desse fator, os participantes foram instruídos a não estudar ou trabalhar sobre questões relacionadas com MPS e / ou MCMPS durante o experimento. No entanto, na prática, não há nenhuma maneira de monitorar se os alunos seguiram essa instrução.

Com o objetivo de reduzir o viés de instrumentação, tentou-se diminuir a influência de fatores subjetivos na pontuação, garantindo que todos os testes fossem corrigidos por dois especialistas e um verificador. Além disso, nenhuma informação sobre os participantes foi fornecida nos testes. Apesar dos ganhos médios terem sido estatisticamente significativos em H01 e H02b, esses resultados não podem ser atribuídos ao tratamento por si só, mas certamente, em parte, envolve uma regressão à média (CAMPBELL; STANLEY, 1963).

A pontuação mínima de acertos para passar de estágio foi de 50%, assim os resultados alcançados estão diretamente relacionados e assegurados para essa pontuação de 50%. Outros valores para tal pontuação podem ter resultados diferentes para o mesmo tipo de experimento executado nesse trabalho.

Como os pré/pós-testes (nos níveis de conhecimento e compreensão) foram questões idênticas, existe um risco de que a aprendizagem possa ter sido conseguida em partes, por meio da realização do próprio teste.

6.1.2. Validade externa

É a capacidade de generalizar os resultados de um estudo para outras populações em outros contextos. As possíveis ameaças identificadas foram:

Como ameaças à validade externa, o experimento ocorreu em um ambiente acadêmico, e a maioria dos participantes teve apenas experiência acadêmica por meio de pesquisa de graduação e estágio supervisionado (ver a análise de dados do background pessoal dos participantes no Apêndice III). Assim, os resultados só podem ser generalizados a uma extensão limitada pelo contexto acadêmico. Além disso, o experimento foi realizado em uma amostra muito pequena de participantes e não foi replicado por outros grupos de investigação em outras universidades. Esses fatores tornam mais difícil a generalização dos resultados. Os alunos já tinham experiência em UML e modelagem Entidade-Relacionamento, o que pode ter favorecido a aprendizado da modelagem de processos de software. Os resultados podem não ser semelhantes em situações em que os alunos não têm um conhecimento prévio dessas ferramentas de modelagem. Como o uso de SPIDER_ML foi obrigatório, alguns alunos podem ter tido aptidões variáveis na utilização desta LMPS, e os resultados podem ser diferentes se forem usadas outras LMPS.

A atividade de MPS do DesigMPS é similar às atividades de modelagem típicas em ambientes acadêmicos. Nesse âmbito os alunos devem criar modelos a partir da descrição textual de um processo de software, os quais normalmente têm um contexto e objetivos muito bem definidos, o que facilita a modelagem. Tais atividades são bastante didáticas, representando simplificadamente cenários reais de MPS. Desta forma, não existem garantias de que, se o experimento for executado em um ambiente real de melhoria de processo de *software*, possam ser obtidos resultados similares.

6.1.3. Validade de construção

É o grau em que as variáveis independentes e dependentes medem acuradamente os conceitos que se propõem a medir. Os seguintes assuntos relacionados com a validade de construção foram identificados:

Os problemas a seguir são pertinentes à validade de construção. Os potenciais efeitos de aprendizado foram medidos pela diferença da pontuação entre os pré e os pós-testes. No entanto, isso pode não ser suficiente para medir o efeito de aprendizagem real em relação ao nível de aplicação, ou seja, se os participantes aprenderam a utilizar os seus novos conhecimentos em um contexto diferente do MCMPS usados no experimento. Um possível

risco seria se os pós-testes fossem mais fáceis do que os pré-testes, porque neste cenário, não haveria uma forte probabilidade das pontuações dos pós-testes serem maiores do que as dos pré-testes. Para tentar eliminar essa ameaça, os dois testes tiveram a mesma estrutura - uma questão para modelar um processo de software no contexto do MR-MPS-SW; mas cada teste deveria modelar processos diferentes (o pré-teste foi sobre o processo de Verificação(VER e o pós-teste sobre o processo de Garantia da Qualidade (GQA)). Os dois processos foram escolhidos por serem considerados de complexidade similar de acordo com a experiência de especialistas da área.

Pode haver ameaça em relação às medidas para avaliar adequação (Y.7) uma vez que trata de assuntos que são difíceis de medir e foram capturados por meio de medidas subjetivas. A avaliação subjetiva da variável suficiência (Y.7.2) do jogo pode ser ameaça em especial à validade. Isto ocorre em virtude de que os alunos ainda estão aprendendo sobre MPS e, talvez, não possam realizar sozinhos julgamentos mais adequados.

6.1.4. Validade de conclusão

Preocupa-se em assegurar que existe uma relação estatística entre o tratamento e os resultados.

O chamado *Simpson's paradox* é uma ameaça em potencial que pode ter ocorrido neste estudo. Este é um fenômeno que pode acontecer quando uma tendência que aparece em diferentes grupos de dados desaparece quando esses dados são reunidos, e uma tendência invertida surge no novo conjunto de dados agregados.

Mesmo reunindo todos os dados, a amostra ainda permanece muito pequena. Desta forma, usando essa amostra basicamente não se pode demonstrar qualquer relação estatística válida. Porém, ao invés de abandonarmos a pesquisa, nós adotamos um teste estatístico mais robusto aceitando um baixo poder estatístico. Esse tipo de decisão já foi tomada em estudos similares (PFAHL et al., 2003; WANGENHEIM et al., 2009).

Os testes no nível de aplicação foram compostos apenas por uma única questão (instrução). Como consequência as notas podem limitar a representação da aprendizagem – obviamente, quanto maior o número de questões, as notas tornam-se mais representativas em respeito à aprendizagem. Isso pode explicar o fato de que alguns alunos obtiveram uma nota pior no pós-teste do que no pré-teste, pois um erro ou equívoco cometido pelo aluno tem um grande impacto na nota dele.

Este trabalho é considerado um estudo explanatório para se obter os primeiros *insights* sobre a eficácia de aprendizagem do DesigMPS, então aceita-se, portanto, que a significância dos resultados são fracos em consequência das ameaças à validade encontradas nesse trabalho.

7. CONCLUSÃO

No âmbito desta tese, apresentou-se uma forma de melhorar enfoques práticos do ensino relacionados a MPS, permitindo a aplicação de conhecimentos teóricos em contextos mais próximos da prática, o que foi possível por intermédio de um jogo, que foi usado complementarmente a uma unidade instrucional sobre MPS, em cursos de graduação de Computação. Apesar da relevância da MPS, este ainda não contava com jogos como uma estratégia instrucional, opostamente a outras áreas da ES que já fazem uso de jogos, o que foi constatado em 3 revisões sistemáticas da literatura sobre jogos para ensino de ES.

O objetivo deste jogo é que o aluno melhore a sua capacidade de modelar processos de software. E para esse fim, o DesigMPS tem uma abordagem de ensino baseada em MC, em que esses são avaliados adaptativamente. Abordagens baseadas em MC são amplamente reconhecidas como eficazes no ensino de várias áreas do conhecimento, inclusive relacionadas à computação.

A fim de examinar os efeitos de aprendizagem do DesigMPS, realizou-se um experimento formal com pré/pós testes e grupo controle com alunos de graduação em uma unidade instrucional sobre MPS. Como resultado desse experimento se pode comprovar a hipótese de pesquisa - *jogos sérios sobre modelagem de processos de software são eficazes para melhorar a aprendizagem dos alunos em relação a esta área de conhecimento* - para os níveis de compreensão e aplicação em virtude, respectivamente, dos achados da Questão 1 e da diferença estatisticamente significativa ao se comprar os pré/pós-testes do grupo A no nível de compreensão. Porém, não se conseguiu provar uma diferença estatisticamente significativa no nível de conhecimento.

Ainda como resultados do experimento, as questões levantadas da Introdução foram respondidas:

Questão 1: Jogar DesigMPS aumenta a eficácia de aprendizagem de MPS no nível de aplicação?

A resposta para essa questão é que existe a indicação de que jogar o DesigMPS realmente melhora a eficácia no aprendizado no nível de aplicação, o que foi demonstrado pela rejeição da hipótese nula (H01: Não existe diferença entre as pontuações médias do pré e pós testes no Grupo A). Assim, demonstrou-se que os alunos conseguiram obter notas realmente maiores depois de usar o jogo.

Questão 2: Como é a eficácia de aprendizagem de MPS com o DesigMPS comparada com a do ABP nos níveis de conhecimento, compreensão e aplicação?

A resposta dessa questão se deu em partes para cada nível cognitivo. Os resultados dos testes das hipóteses nulas H02a e H02b demonstraram que o DesigMPS proporciona um melhor aprendizado de MPS no nível de aplicação do que o ABP; os resultados dos testes das hipóteses nulas H02c e H02d demonstraram que o DesigMPS proporciona um aprendizado em MPS no nível de conhecimento igualmente ao ABP; e os testes das hipóteses nulas H02e e H02f para o nível de compreensão demonstraram que o DesigMPS proporciona um aprendizado igual ao do ABP. Assim, pode-se evidenciar que o jogo proporciona um melhor aprendizado apenas no nível de aplicação quando comparado ao ABP

Questão 3: O jogo é considerado adequado em termos de relevância de conteúdo, suficiência, grau de dificuldade, sequência, e método de ensino para o qual ele foi desenvolvido?

Pode-se considerar que o jogo é adequado em todas essas dimensões, pois foram avaliadas positivamente por mais de 70% dos alunos: relevância de conteúdo (95,24%), suficiência (90,48%), grau de dificuldade (80,96%), sequência (76,19%), e método de ensino (80,95%).

Questão 4: O jogo é atrativo?

O jogo pode ser considerado atrativo, visto que 78% dos alunos avaliaram o jogo como atrativo.

Questão 5: Quais são os pontos fortes e pontos fracos do jogo?

Os alunos do grupo experimental apontaram os seguintes pontos fortes do jogo: relevância da modelagem de processos de software, facilidade de uso, integração entre teoria e prática, modelo de referência ajuda a entender a modelagem, aprendizado de forma divertida/desafiadora, ter um modelo de referência ajuda os alunos a criarem seus próprios modelos, visualizar graficamente o processo é melhor do que apenas a descrição textual do mesmo, algum auxílio no processo de modelagem é melhor do que tentar modelar sozinho, aprendizado por repetição.

Estes alunos também relataram os pontos fracos do jogo: *feedback* dos erros, ausência de *help* e de material de consulta sobre MPS, poucas fases, usabilidade, desnível de dificuldade muito grande entre os estágios 3 e 4, efeito de decorar o modelo do processo, modelos grandes são cansativos e desmotivadores, não é relevante para quem trabalha com métodos ágeis, restringe a criatividade

A principal contribuição desta tese é oferecer um jogo baseado em avaliações adaptativas de MC eficaz no ensino de MPS, principalmente, nos níveis cognitivos de

compreensão e aplicação, quando o jogo é empregado no contexto de uma disciplina que trate de MPS. Outras contribuições são:

- O jogo pode ser uma opção a mais para o ensino de MPS frente às abordagens já existentes.
- O jogo pode ensinar sobre qualquer processo de um MCMPS (ex. MPS.Br e CMMI), desde que possa ser modelado pelo SPIDER_ML. Isso permite que o jogo tenha vários escopos, seja adaptável às necessidades de ensino de novos processos, pois novos modelos de processo de software podem ser criados por um especialista em MPS.

Apesar dos resultados não serem suficientes para generalização, em virtude da pequena amostra e da falta de replicação do experimento, os resultados encorajam que mais pesquisas sejam feitas com objetivo de investigar a comparação de aprendizado com outros métodos de ensino. E também que a mesma abordagem de ensino usada no DesigMPS possa ser aplicada no ensino de outros tipos de modelagem pertinente à ES (ex. modelos da UML e Entidade-Relacionamento).

8.1 Trabalhos futuros

A complexidade de se avaliar experimentalmente os jogos educativos provavelmente resultou que poucos jogos para o ensino de ES tenham sido avaliados para medir sua eficácia de aprendizagem. Assim, outros modelos mais simples para a avaliação do jogo devem ser investigados, a exemplo de Savi et al. (2011), para obter evidência mais confiável do impacto dos jogos sérios em educação de ES nos níveis cognitivos de aplicação e superiores.

Os alunos contribuíram com opiniões em relação aos pontos fortes, pontos fracos e sugestão de melhorias. Os pontos fracos serão avaliados para decidir quais devem ser corrigidos, e as sugestões de melhoria serão avaliadas quanto à viabilidade de serem incorporadas às futuras versões do DesigMPS.

Baseados nos *feedbacks* dos alunos, o jogo está sendo aperfeiçoado a fim de melhorar o *feedback* sobre erros cometidos na modelagem, com a implementação de um tutor inteligente (BURNS; CAPPS, 1988) que indique o erro e explique a causa. Também está sendo aperfeiçoado o método de correção do modelo, para possibilitar que o aluno tenha mais liberdade criativa para modelar, visando eliminar o efeito de decorar por repetição de um mesmo modelo. Para este fim está sendo usado Wordnet (FELLBAUM, 1998) para análise semântica. Além disso, a pesquisa futura sobre de DesigMPS deve examinar recursos que

aumentam a motivação do jogador, estratégias de cooperação e/ou competição entre os alunos. Complementarmente, para uma melhor avaliação, o *log* das ações dos alunos deve ser refinado, objetivendo, por exemplo, registrar o tempo gasto em cada nível, bem como os números e tipos de erros.

Ainda é relevante o desenvolvimento de um material de apoio teórico detalhado sobre SPM, para disponibilizar juntamente com o jogo, com o objetivo de possibilitar um maior grau de autoestudo. Pensa-se em continuar a explorar e adaptar os MC como ferramentas educacionais para o ensino de SPM. Além disso, em virtude dos resultados alcançados, esperamos que outros tipos de modelagens, e.g., Orientação a Objetos e Entidade-Relacionamento, possam ter seu ensino melhorado com abordagem semelhante à deste trabalho.

8.2. Publicações

Eventos

CHAVES, R.; MIRANDA, T.; TAVARES, E.; OLIVEIRA, S.; FAVERO, E. Desigmps: um jogo de apoio ao ensino de modelos de qualidade de processos de software, baseado em mapas conceituais. In: CONGRESSO BRASILEIRO DE EDUCAÇÃO EM ENGENHARIA, 39. Blumenau-SC, 2011. **Anais...** 2011.

Periódicos (Qualis A1 em Engenharias IV)

CHAVES, R.; GRESSE VON WANGENHEIM, C.; FURTADO, J.; OLIVEIRA, S.; SANTOS, A.; FAVERO, E. Experimental evaluation of a serious game for teaching software process modeling. **IEEE Transactions on Education**, v. 58, n. 4, p. 289-296, nov. 2015.

REFERÊNCIAS

ABNT – Associação Brasileira de Normas e Técnicas. **NBR ISO/IEC 12207:2009** – Engenharia de Sistemas de Software – Processos de Ciclo de Vida de Software. Rio de Janeiro, 2009.

ABNT – Associação Brasileira de Normas e Técnicas. **NBR ISO/IEC 15504-1:2008** - Tecnologia da Informação - Avaliação de processo Parte 1: Conceitos e Vocabulário. Rio de Janeiro, 2008.

ACM/AIS/IEEE-CS. **The Joint Task Force for Computing Curricula**, Computing Curricula 2005 – The Overview Report.

ACUÑA, S. T.; ANTONIO, A.; FERRÉ, X.; LÓPEZ MARTA, Maté Luis. The Software Process: Modelling, Evaluation and Improvement. In: CHANG, S. K. (eds) **Handbook of Software Engineering and Knowledge Engineering**. Singapore: World Scientific Publishing Company, 2000. p. 193-237.

ACUÑA, S. T.; JURISTO, N. Assigning people to roles in software projects. **Software: Practice and Experience**, New York: John Wiley & Sons, Inc, v. 34, n. 7, p. 675-696, 2004.

AAEN, I.; ARENT, J.; MATHIASSEN, L.; NGWENYAMA, O. A conceptual MAP of software process improvement. **Scandinavian Journal of Information Systems**, Special Issue on Trends in the Research on Software Process Improvement in Scandinavia, v. 13, p. 123-146, 2001.

AHLBER, M. Varieties of Concept Mapping. In: FIRST INTERNACIONAL CONCEPT MAPPING, 1. **Proceedings…** 2004.

ALDAZABAL, A., BAILY, T.; NANCLARES, F.; SADOVYKH, A.; HEIN, C, ESSER M, RITTER T. Automated model driven development processes. In: ECMDA WORKSHOP ON MODEL DRIVEN TOOL AND PROCESS INTEGRATION, Fraunhofer IRB Verlag, Stuttgart, 2008. **Proceedings…** 2008. p. 361-375

ALMSTRUM, V. L.; DALE, N.; BERGLUND, A.; GRANGER, M.; CURRIE LITTLE, J.; MILLER, D. M. Evaluation: turning technology from toy to tool. In: CONFERENCE ON INTEGRATING TECHNOLOGY INTO COMPUTER SCIENCE EDUCATION (ITiCSE '96), 1. Barcelona, Spain, 1996. **Proceedings…**1996. p. 201-217

ANDERSSON, C. **Exploring the software verification and validation process with focus on efficient fault detection**. 2003. Licentiate Thesis, Lund Institute of Technology (LTH), Lund University, Sweden, 2003.

ANDERSON, L. W.; KRATHWOHL, D. R. (eds.) **A taxonomy for learning, teaching, and assessing**: a revision of bloom's taxonomy of educational objectives. New York: Longman, 2001.

ANGOFF, W. H. **Scale, norms, and equivalent scores**. Princeton: Educational Testing Service, 1984.

ANOHINA, A.; GRAUDINA, V.; GRUNDSPENKIS, J. Using concept maps in adaptive knowledge assessment. In: MAGYAR, G. et al. (ed.) **Advances in information systems development**, new methods and practice for the networked society. 2007. v. 1, Springer, p. 469-479.

ANOHINA, A.; VILKELIS, M.; LUKASENKO, R. Incremental improvement of the evaluation algorithm in the concept map based knowledge assessment system. **International Journal of Computers**, Communication and Control, v. 4, n. 1, p. 6-16, 2009.

ANOHINA, A.; STRAUTMANE, M.; GRUNDSPENKIS, J. Development of the scoring mechanism for the concept map based intelligent knowledge assessment system. In: INTERNATIONAL CONFERENCE ON COMPUTER SYSTEMS AND TECHNOLOGIES, 11./Workshop for PhD Students in Computing. Nova York, 2010. **Proceedings…** 2010.

ARMENISE, P.; BANDINELLI, S.; GHEZZI, C.; MORZENTI, A. A survey and assessment of software process representation formalism. **International Journal of Software Engineering and Knowledge Engineering**, v. 3, n. 3, p. 401-426, 1993.

ARMITAGE, J. W.; KELLNER, M. I. (1994) A Conceptual Schema for Process Definitions and Models. In: INTERNATIONAL CONFERENCE ON THE SOFTWARE PROCESS, 3. Virginia, USA, 1994. **Proceedings**… 1994. p. 153-165.

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. Goal/question/metric approach. In: MARCINIAK, J. (ed) **Encyclopedia of software engineering**. New York: John Wiley & Sons, 1994. v. 1, p. 528-532.

BECKER-KORNSTAEDT, U.; BELAU, W. Descriptive process modeling in an industrial environment: experience and guidelines. **Software Process Technology**, 2000. Disponível em: http://www.springerlink.com/index/t43q70180l337w16.pdf. Acesso em: jan. 2013

BECKER-KORNSTAEDT, U. Descriptive software process modeling - how to deal with sensitive process information. **Empirical Software Engineering**, n. 88, 2001. Disponível em: http://www.springerlink.com/index/X17X26738785538G.pdf. Acesso em: jan. 2013

BENALI, K.; DERNIAME, J. C. Software processes modeling: what, who, and when. In: EUROPEAN WORKSHOP ON SOFTWARE PROCESS TECHNOLOGY, 2. Trondheim, Norway, 2002. **Proceedings…** 1992. p. 21-25

BENDRAOU, R.; JÉZÉQUEL, J.M.; GERVAIS, M. P.; BLANC, X. A comparison of six UML-based languages for software process modeling, **IEEE Transactions on Software Engineering**, v. 36, n. 5, p. 662–675, 2010.

BIRK, A.; PFAHL, D. A. Systems perspective on software process improvement. In: INTERNATIONAL CONFERENCE ON PRODUCT FOCUSED SOFTWARE PROCESS IMPROVEMENT, 4. **Proceedings…**v. 2559, 2002. p. 4-18.

BIRKHOELZER, T.; NAVARRO, E. O.; HOEK, A. V. D. Teaching by Modeling instead of by Models. In: INTERNATIONAL WORKSHOP ON SOFTWARE PROCESS

SIMULATION AND MODELING, 6. St. Louis, MO, EUA, 2005. **Proceedings…** 2005. p. 1–4.

BOEHM, B. **Software engineering economics**. Prentice-Hall, 1981

BOEHM, B.; BASILI, V. Software defect reduction top 10 list. **IEEE Computer**, v. 34, n. 1, p. 135-137, 2001.

BOYLE EA, CONNOLLY TM, HAINEY T (2011) The role of psychology in understanding the impact of computer games. Entertainment Computing, 2, pp 69–74.

BRUILLARD, E.; BARON, G. L. Computer-based concept mapping: a review of a cognitive tool for students. In: ICEUT2000, 16th IFIP World Computer Congress (eds. Benzie, D. & Passey, D.), **Proceedings…** 2000, p. 331-338.

BRUSILOVSKY, P. Methods and Techniques of Adaptative Hypermedia. **User Modeling and User Adapted Interaction**, v. 6, n. 2-3, p. 87-129, 1996.

BURNS, H. L.; CAPPS, C. G. Foundations of intelligent tutoring systems: an introduction. In: POLSON, M. C., RICHARDSON, J. J. (eds) **Foundations of intelligent tutoring systems**. London: Lawrence Erlbaum, 1988. p. 1-19.

CALVO, I.; ARRUARTE, A.; ELORRIAGA, J. A.; LARRAÑAGA, M.; CONDE, A. The use of concept maps in computer engineering education to promote meaningful learning, creativity and collaboration. Proceedings of the 41st Frontiers in Education Conference (FIE'2011*).* South Dakota, USA. 2011. p. T4G-1 - T4G-6.

CAMPBELL, D. T.; STANLEY, J. C. **Experimental and quasi-experimental designs for research**. Houghton Mifflin Company, Boston, MA, 1963.

CAULFIELD, C.; XIA, J.; VEAL, D.; MAJ, S. P. A systematic survey of games used for software engineering education. **Modern Applied Science**, v. 5, n. 6, p. 28-43, 2011.

CAÑAS, A. J. **A summary of literature pertaining to the use of techniques and concept mapping technologies for education and performance support**. 2003. Disponível em: http://www.ihmc.us/users/acanas/Publications/ConceptMapLitReview/. Acesso em: jan. 2011.

CHAVES, R. O.; MIRANDA, T. C.; TAVARES, E. M. C.; OLIVEIRA, S. R. B.; FAVERO, E. L. Desigmps: um jogo de apoio ao ensino de modelos de qualidade de processos de software, baseado em mapas conceituais. In: CONGRESSO BRASILEIRO DE EDUCAÇÃO EM ENGENHARIA, 39. 2011. **Proceedings…**Blumenau-SC, 2011.

COHEN, L.; MANION, L.; MORRISON, K. **Research methods in education**. Routledge Falmer, 2000.

CONNOLLY, T. M.; STANSFIELD, M.; HAINEY, T. An application of games-based learning within software engineering. **British Journal of Educational Technology**, v. 38, n. 3, p. 416-428, 2007.

CONNOLLY, T. M.; BOYLE, E. A.; MACARTHUR, E.; HAINEY, T.; BOYLE, J. M. A systematic literature review of empirical evidence on computer games and serious games. **Computers & Education**, v. 59, n. 2, p. 661-686, 2012.

CRONBACH, L. J. Coefficient alpha and the internal structure of tests. **Psych ometrika**, n. 16, p. 297-334, 1951.

CURTIS, B.; KELLNER, M. I.; OVER, J. Process modeling. **Communications of the ACM**, New York, v. 35, n. 9, p. 75-90, 1992.

DANTAS, A.; BARROS, M.; WERNER, C. A. Simulation-based game for project management experiential learning SEKE. - 2004. p. 19-24.

DE MELLO, M. S. **Melhoria de processos de software multi-modelos baseada nos modelos MPS e CMMI-DEV**. 2001. 232f. Dissertação (Mestrado em Engenharia de Sistemas e Computação) - COPPE/UFRJ, Rio de Janeiro, 2001.

DEMPSEY, J. V.; LUCASSEN, B.; RASMUSSEN, K. **The instructional gaming literature**: implications and 99 sources, tech. report 96-1. College of Education, Univ. of South Alabama, 1996.

DRAPPA, A.; LUDEWIG, J. Simulation in software engineering training. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 22. Limerick, Ireland, **Proceedings…** 2000. p 199–208

DYM, C. L.; AGOGINO, A. M.; ERIS, O.; FREY, D. D.; LEIFER, L. J. Engineering design thinking, teaching, and learning. **Journal of Engineering Education**, v. 94, n. 1, p. 103-120, 2005.

FEILER, P. H.; HUMPHREY, W. S. Software process development and enactment: concepts and definitions. In: INTERNATIONAL CONFERENCE ON THE SOFTWARE PROCESS, 2. Berlin, Germany, 1993. **Proceedings…** 1993. p. 28-40.

FINKELSTEIN; KRAMER, J., NUSEIBEH, B. **Software process modelling and technology**. 1994. (Research Studies Press)

FUGGETA, A. Software process: a roadmap. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE'00), 22. Limerick, Ireland, 2000. **Proceedings…** p. 27-34.

GARCIA, F.; VIZCAINO, A.; EBERT, C. Process management tools. **Journal IEEE Software**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 28, n. 2, p. 15-18, 2011.

GARCÍA-BORGOÑON, L.; GARCIA-GARCIA, J. A.; ALBA, M.; ESCALONA, M. J. (2013) Software process management: a model-based approach. In: INTERNATIONAL CONFERENCE ON INFORMATION SYSTEMS DEVELOPMENT (ISD2012), 21. **Proceedings…** 2013.

GARCÍA-BORGOÑON, L.; BARCELONA, M. A.; GARCÍA-GARCÍA, J. A.; ALBA, M.; ESCALONA, M. J. Software process modeling languages: A systematic literature review. **Information & Software Technology**, v. 56, n. 2, p. 103-116, 2014.

GLAZER, H.; DALTON, J.; ANDERSON, D.; KONRAD, M. D.; SHRUM, S. **CMMI or Agile**: why not embrace both! Technical Note CMU/SEI-2008-TN-003. Software Engineering Institute/Carnegie Mellon University, Pittsburgh, Pennsylvania, 2008.

GOOLD, A.; HORAN, P. Foundation software engineering practices for capstone projects and beyond. In: CONFERENCE ON SOFTWARE ENGINEERING EDUCATION AND TRAINING, 15. IEEE: Covington, KY, USA. **Proceedings… 2002.** P. 140-146.

GRAUDINA, V.; GRUNDSPENKIS, J. Conceptual model for ontology-based adaptive assessment system. In: E-LEARNING CONFERENCE-COMPUTER SCIENCE EDUCATION, 3. Coimbra, Portugal, 2006. **Proceedings…** 2006.

GROTH, D. P.; ROBERTSON, E. L. It's All About Process: Project-Oriented Teaching of Software Engineering. In: CONFERENCE ON SOFTWARE ENGINEERING EDUCATION AND TRAINING, 14. Charlotte, USA, 2001. **Proceedings…** 2001. p. 7-17

GRUNDSPENKIS, J.; ANOHINA, A. **Evolution of the concept map based adaptive knowledge assessment system**: implementation and evaluation results. Disponível em: <https://ortus.rtu.lv/science/en/publications/6110/fulltext>. Acesso em: maio 2011

HADIM, H. A.; ESCHE, S. K. Enhancing the engineering curriculum through project-based learning. In: FRONTIERS IN EDUCATION CONFERENCE, 32. Boston, USA, Section F3F, **Proceedings…** 2002. p. 1-6.

HAWKER, J. S. A software process engineering course. In: 2009 AMERICAN SOCIETY FOR ENGINEERING EDUCATION ANNUAL CONFERENCE. Austin, TX, 2009. **Proceedings…** 2009.

HSUEH, Nien-Lin et al. Applying UML and software simulation for process definition, verification, and validation. **Information and Software Technology**, v. 50, n. 9-10, ago. 2008.

HUFF, K. Software process modeling. In: **Software process**. John Wiley & Sons, 1996.

HUMPHREY, W., S. **Managing the software process**. The SEI Series in Software Engineering. Addison-Wesley, 1989.

IEEE-CS/ACM. **The Joint Task Force on Computing Curricula**. Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, 2004.

IEEE-CS. **SWEBOK**: Guide to the Software Engineering Body of Knowledge, 2004.

IEEE - INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. **Std 610.12** - IEEE Standard Glossary of Software Engineering Terminology, 1990.

ISO/IEC INTERNATIONAL ORGANIZATION FOR STANDARDIZATION/ INTERNATIONAL ELECTROTECHNICAL COMISSION. **ISO/IEC 20000** Information Technology– Service Management, Geneve: ISO, 2011.

ITIN, C. M. Reasserting the philosophy of experiential education as a vehicle for change in the 21st century. **The Journal of Experiential Education**, n. 22, p. 91-98, 1999.

JACCHERI, M. L.; LAGO, P. Applying Software Process Modeling and Improvement in Academic Setting. In: CONFERENCE ON SOFTWARE ENGINEERING EDUCATION & TRAINING, 10. Virginia Beach, Virginia, 1997. **Proceedings…** IEEE Computer Society Press, 1997. p. 13-27.

KAFAI, Y. B. The educational potential of electronic games: from games-to-teach to games-to learn. In: CONFERENCE ON PLAYING BY THE RULES: the cultural policy challenges of video games. Chicago, Illinois, 2001. **Proceedings…** 2001.

KOSCIANSKI, A.; SOARES, M. S. **Qualidade de software**. 2. ed. São Paulo, Novatec, 2007.

KRISHNAN, M. S.; KRIEBEL, C. H.; KEKRE, S.; MUKHOPADHYAY, T. An Empirical Analysis of Productivity and Quality in Software Products. **Management Science**, v. 46, n. 6, p. 745-759, 2000.

KUHRMANN, M.; FERNÁNDEZ, D. M.; MÜNCH, J. Teaching Software Process Modeling. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE'2013), 35. San Francisco, United States, **Proceedings…** 2013. p. 18-26.

LETHBRIDGE, T. C. What knowledge is important to a software professional? **Journal Computer**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 33, n. 5, p. 44-50, 2000.

MCCHESNEY, R. Toward a classification scheme for software process modelling approaches. **Information and Software Technology**, v. 37, n. 7, p. 363-374, 1995.

MANDL-STRIEGNITZ, P. How to successfully use software project simulation for educating software project managers [Conferência] // FIE '01 Proceedings of the Frontiers in Education Conference, 2001. on 31st Annual - Volume 01. - Washington, DC, USA : IEEE Computer Society, 2001. - pp. 1-6.

MILLS, J. E.; TREAGUST, D. F. Engineering education: is problem-based or project-basedlearning the answer? **Australasian Journal of Engineering Education**, 2003–04, online publication. 2004. Disponível em: http://www.aaee.com.au/journal/2003/ mills_treagust03.pdf Acesso em: 06 ago. 2011.

NAVARRO, E. O. **SimSE**: A Software Engineering Simulation Environment for Software Process Education. 2006. Tese (Doutorado em Information and Computer Science) - University of California, EUA, 2006.

NAVARRO, E. O.; VAN DER HOEK, A. Comprehensive evaluation of an educational software engineering simulation environment. In: CONFERENCE ON SOFTWARE

ENGINEERING EDUCATION AND TRAINING, 20. Dublin, Ireland, 2007. **Proceedings...** 2007. p. 195-202

NOVAK, J. D. **A theory of education**. Ithaca, NY: Cornell University, 1977.

NOVAK, J. D.; GOWIN, D. B. **Learning how to learn**. Cambridge University Press, 1984.

NOVAK, J. D.; CAÑAS, A. J. **The theory underlying concept maps and how to construct them**. Technical Report IHCM CmapTools, 2006.

NOVAK, J. D.; CAÑAS, A. J. **The theory underlying concept maps and how to construct and use them** [on line]. Technical Report IHMC CmapTools 2006-01 Rev 01-2008, Florida, Institute for Human and Machine Cognition. Acesso em: 15 jun. 2010.

OLIVEIRA, S. R. B. **SPIDER** - Uma Proposta de Solução Sistêmica de um Suite de Ferramentas de Software Livre de apoio à implementação do modelo MPS.BR. Research Project. Instituto de Ciências Exatas e Naturais, Universidade Federal do Pará, Belém, 2009.

OMG - Object Management Group. **Software & Systems Process Engineering Metamodel Specification (SPEM)**. Version 2.0, Technical. Report, Object Management Group, 2008.

PAPANASTASIOU, E. C. Computer-adptive testing in science education. In: INTERNATIONAL CONFERENCE ON COMPUTER BASED LEARNING IN SCIENCE, 6. **Proceedings…**2003.

PAULK, M. C.; WEBER, C. V.; GARCIA, S. M.; CHRISSIS, M. B.; BUSH, M. **Key Practices of the Capability Maturity Model**, Version 1.1 Technical Report SEI-93-TR-25, Software Engineering Institute, Carnegie Mellon University, USA, 1993.

PFAHL, D.; KOVAL, N.; RUHE, G. An experiment for evaluating the effectiveness of using a system dynamics simulation model in software project management education. In: INTERNATIONAL SOFTWARE METRICS SYMPOSIUM, 7. London, 2001. **Proceedings…**United Kingdom, p. 97-109, 2001.

PFAHL, D.; LAITENBERGER, O.; DORSCH, J.; RUHE, G. An externally replicated experiment for evaluating the learning effectiveness of using simulations in software project management education. **Empirical Software Engineering**, Kluwer Academic, The Netherlands, v.8, p. 367-395, 2003.

PFAHL, D.; LAITENBERGER, O.; RUHE, G.; DORSCH, J.; KRIVOBOKOVA, T. Evaluating the learning effectiveness of using simulations in software project management education: results from a twice replicated experiment. **Information and Software Technology**, v. 46, n. 2, p. 127-147, 2004.

PFLEEGER, S. L. **Software Engineering: theory and practice, 2nd edition**. Prentice-Hall, Inc., ISBN 0-13-029049-1, 2001.

PRESSMAN, R. S. **Engenharia de software**: uma abordagem profissional. McGraw Hill, 2011.

RAI, A.; AL-HINDI, H. The effects of development process modeling and task uncertainty on development quality performance. **Information & Management**, v. 37, n. 6, p. 335-346, 2000.

ROCHA, A. R. C.; MALDONADO, J. C.; WEBER, K. C. **Qualidade de software** – Teoria e Prática. Prentice Hall, São Paulo, 2001.

ROCHA, F. E. L.; COSTA, J. V.; FAVERO, E. L. An Approach to Computer-Aided Learning Assessment. In: CONFERENCE ON CONCEPT MAPPING, 3. 2008, Tallinn, Estonia and Helsinki, Finland **Proceedings…** 2008.

RODRÍGUEZ, D.; SICILIA, M. A.; CUADRADO, J. J.; PFAHL, D. E-learning in project management using simulation models: a case study based on the replication of an experiment. **IEEE Transactions on Education**, v. 49, n. 4, p. 451-462, 2006.

RUIZ-PRIMO, M. A. Examining concept maps as an assessment tool. In: CONF. IN CONCEPT MAPPING, 1. **Proceedings…** 2004.

RÜÜTMANN, T.; KIPPER, H. Teaching strategies for direct and indirect instruction in teaching engineering. In: INTERNATIONAL CONFERENCE ON INTERACTIVE COLLABORATIVE LEARNING (ICL'2011), 14. Piestany, Slovakia, 2011. **Proceedings…** 2011. p. 107-114.

SASKATCHEWAN Education. **Instructional approaches**: a framework for professional practice, 1991.

SAVI, R.; VON WANGENHEIM, C. G.; BORGATTO, A. F. **Analysis of an Evaluation Model of Educational Games**. Technical Report INCOD – National Institute for Research and Technology on Digital Convergence (INCOD)/UFSC, Brazil, 2011.

SOFTEX. **Melhoria do Processo de Software Brasileiro (MPS.BR)** - Guia Geral 2012, Brasil. 2012a.

SOFTEX. **Melhoria do Processo de Software Brasileiro (MPS.BR)** - Guia de Implementação - Parte 11: Implementação e Avaliação do MR-MPS-SW em conjunto com o CMMI-DEV v1.3, Brasil, 2012b.

SOFTEX. **Melhoria do Processo de Software Brasileiro (MPS.BR)** - Guia de Implementação - Parte 2: Fundamentação para Implementação do Nível F do MR-MPS-SW:2012, Brasil, 2012c.

SCHULMEYER, G. G.; MACKENZIE, G. R. **Verification & validation of modern software-intensive systems**. New Jersey: Prentice-Hall Inc, 1999.

SEI – Software Engineering Institute (2010) CMMI Product Team. CMMI for Development (CMMI-DEV), Version 1.3, Technical Report, CMU/SEI-2010-TR-033.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). **Biometrika**, London, v. 52, p. 591-609, 1965.

SPIDER – Software Process Improvement: Development and Research, "SPIDER_ML - Technical Specification," Federal University of Pará, Brazil, 2009. Disponível em: http://www.spider.ufpa.br/projetos/spider_pm/SPIDER_ML%5B1.1%5D.pdf. Acesso em: 25 fev. 2015.

SPIDER – Software Process Improvement: Development and Research, SPIDER_PM Technical Specification, Federal University of Pará, Brazil, 2010. 35p. Disponível em: http://www.spider.ufpa.br/projetos/spider_pm/Spider-PM.pdf. Acesso em: 25 fev. 2015.

SPINOLA, M. M., TONINI, A. C.; CARVALHO, M. M. Contribuição dos modelos de qualidade e maturidade na melhoria dos processos de software. **Revista Produção**, São Paulo: USP, 2008.

WANGENHEIM, C. G. V.; SHULL, F. To Game or Not to Game? **IEEE Software**, IEEE Computer Society Press Los Alamitos, CA, USA, v. 26, n. 2, p. 92-94, 2009.

WANGENHEIM, C. G.; THIRY, M.; KOCHANSKI, D. Empirical evaluation of an educational game on software measurement. **Empirical Software Engineering**, Kluwer Academic Publishers Hingham, MA, USA, v. 14, n. 4, p. 418-452, 2009.

WANGENHEIM, C. G. V.; HAUCK, J. C. R. Teaching Software Process Improvement and Assessment. In: EUROPEAN SYSTEMS & SOFTWARE PROCESS (EuroSPI'2010), 17. **Proceedings…** Grenoble, France, 2010.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. **Experimentation in software engineering** - an introduction. Kluwer Academic, Norwell, 2012.

YE, E.; LIU, C.; POLACK-WAHL, J. A. Enhancing Software Engineering Education Using Teaching Aids in 3-D Online Virtual Worlds. In: ASEE/IEEE FRONTIERS IN EDUCATION CONFERENCE, 37. **Proceedings…** p. T1E-8 - T1E-13, 2007.

YU, X.; KLEIN, S. Enhancing student learning using concept mapping and learning by teaching environment. **Journal of Computer Science in Colleges**, Consortium for Computing Sciences in Colleges , USA, v. 23, n. 4, p. 271-278, 2008.

ZAMLI, K. Z.; LEE, P. A. Taxonomy of process modeling languages. In: ACS/IEEE INTERNATIONAL CONFERENCE ON COMPUTER SYSTEMS AND APPLICATIONS AICCSA 2001, Beirut, Lebanon. **Procedings…** IEEE Computer Science Press, 2001. p. 435-437

ZYDA, M. From visual simulation to virtual reality to games. **IEEE Computer**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 38, n. 9, p. 25-32, 2005.

APÊNDICES

APÊNDICE I – PRÉ/PÓS-TESTES

Questões do pré/pós-testes – Nível de conhecimento

A) 	B) 
c) 	D) 
E) 	F) 
G) 	H) 
I) 	J) 

Questões do pré/pós-testes – Nível de compreensão

1- Uma tarefa T2 recebe como input um artefato A1 produzido por uma tarefa T1. Caso T2 seja concluída com êxito, o artefato A1 é transformado no output A2 que é input de uma tarefa T3, caso contrário A1 deve retornar para T1.

2- Uma tarefa T3 recebe como input um artefato A1 produzido por uma tarefa T1 e um artefato A2 produzido por uma tarefa T2, sendo que T1 e T2 são executadas em paralelo. T3 gera um output A3 que é o input de uma tarefa T4 e de uma tarefa T5 que devem ser executadas em paralelo.

3- Uma tarefa T1 é executada por um ator AT1 e por duas ferramentas F1 e F2, gera como output um artefato A1. A1 é input de uma tarefa T2 e de outra tarefa T3, sendo que essas duas são executadas em paralelo. T2 recebe um artefato A4 e é executada por um ator AT2 e gera um artefato A2.

4- Uma tarefa T1 e uma tarefa T2 não são executadas em paralelo. Sendo que os artefatos A1 e A2 outputs gerados respectivamente por T1 e T2, são inputs de uma tarefa T3. T1 e T2 têm como atores AT1 e AT2, respectivamente. Depois que T3 for executada, o artefato T3 é gerado e o processo finalizado.

5- Uma tarefa T1 produz como outputs os artefatos A1 e A1.1. A1 é input de uma tarefa T2 e A1.1 de uma tarefa T3. T3 é executada em paralelo com T4. Caso T3 não seja concluída com êxito, o processo termina. Caso T3 e T4 sejam executadas com êxito, a tarefa T5 é iniciada e recebe como input os outputs artefato A3 e A4, produzidos respectivamente por T3 e T4.

Questão do pré–nível de aplicação

Modele um processo de software alinhado ao VER 1, VER 2 e VER 3 (Processo de Verificação do Nível D do MR-MPS-SW) usando a PML SPIDER_ML. Identifique os elementos de processo de *software* (papel, artefato, tarefa e ferramenta) e faça a respectiva descrição, ou seja, a sua característica e importância do processo. Defina os relacionamentos entre os elementos e a ordem de precedência (end_to_start, start_to_start, start_to_end) entre as tarefas.

APÊNDICE II - FATORES DE CONFUSÃO (DF.1, DF.2), SEUS ITENS DOS QUESTIONÁRIOS DE BACKGROUND E MOTIVAÇÃO, COM AS RESPECTIVAS PONTUAÇÕES

Variável	Item	
Background DF.1	Profissional	Valor
	1- Experiência (ao menos 6 meses) como desenvolvedor/analista de SW	Se o item for verdadeiro, adicione 1 à pontuação de background
	2- Experiência (ao menos 6 meses) como gerente de projetos de SW	Se o item for verdadeiro, adicione 1 à pontuação de background
	3- Experiência (ao menos 6 meses) responsável pelo controle da qualidade de SW	Se o item for verdadeiro, adicione 1 à pontuação de background
	4- Participação na implementação em programas de melhoria de processos	Se o item for verdadeiro, adicione 1 à pontuação de background
	Acadêmica:	
	5- Experiência (ao menos 6 meses) como desenvolvedor/analista de SW	Se o item for verdadeiro, adicione 1 à pontuação de background
	6- Experiência (ao menos 6 meses) como gerente de projetos de SW	Se o item for verdadeiro, adicione 1 à pontuação de background
	7- Experiência (ao menos 6 meses) responsável pelo controle da qualidade de SW	Se o item for verdadeiro, adicione 1 à pontuação de background
	8- Participação na implementação em programas de melhoria de processos	Se o item for verdadeiro, adicione 1 à pontuação de background
	Treinamento	
	9- Treinamento (ao menos 30h) sobre processo de software	Se o item for verdadeiro, adicione 1 à pontuação de background
	10- Treinamento (ao menos 30h) sobre qualidade de software	Se o item for verdadeiro, adicione 1 à pontuação de background
	11- Treinamento (ao menos 30h) sobre CMMI	Se o item for verdadeiro, adicione 1 à pontuação de background
	12- Treinamento (ao menos 30h) sobre MPS.BR	Se o item for verdadeiro, adicione 1 à pontuação de background
	13- Treinamento (ao menos 30h) sobre gerência de projetos	Se o item for verdadeiro, adicione 1 à pontuação de background
		$Pontuação\ de\ Background = \sum_{i=1}^{13} item_i$
Motivação		
DF.2.1	Importância da modelagem de processos de software	-2— discorda fortemente, -1— discorda, 0— neutro, 1— concorda, 2— concorda fortemente
DF.2.2	Interesse em aprender mais sobre modelagem de processos de software	-2— discorda fortemente, -1— discorda, 0— neutro, 1— concorda, 2— concorda fortemente

APÊNDICE III – ANÁLISE DE DF.1 E DF.2

DF.1 Análise dos dados sobre o background pessoal

Aluno	Experiência acadêmica	Experiência profissional	Treinamento	Total
Grupo A				
2	2	0	0	2
3	1	0	0	1
5	1	1	0	2
6	1	0	0	1
10	1	0	0	1
12	1	0	0	1
15	2	1	0	3
16	1	3	2	6
18	1	0	0	1
19	2	1	1	4
22	1	0	0	1
25	1	0	0	1
26	1	1	0	2
27	2	0	0	2
28	1	0	0	1
31	1	1	0	2
34	0	1	1	2
35	1	0	0	1
36	1	0	0	1
39	2	1	1	4
40	1	0	0	1
Mediana				1
Variação interquartil				1
Grupo B				
1	1	0	0	1
4	2	1	1	4
7	1	0	0	1
8	2	2	2	6
9	1	1	0	2
11	1	0	0	1
13	2	1	1	4
14	1	0	0	1
17	1	0	0	1
20	1	1	0	2
21	1	1	0	2
23	1	0	0	1
24	0	0	0	0
29	1	1	0	2
30	1	0	0	1

32	2	1	0	3
33	1	0	0	1
37	2	0	0	2
38	1	0	1	2
41	1	0	0	1
Mediana				1.5
Variação interquartil				1

DF.2 Análise dos dados sobre motivação sobre estudos de modelagem de processo de software

	Motivação	
Aluno	DF.2.1 Importância	DF.2.2 Interesse
Grupo A		
2	1	2
3	2	2
5	2	2
6	2	2
10	2	2
12	2	1
15	1	1
16	1	2
18	2	2
19	-2	0
22	2	2
25	2	1
26	2	2
27	2	2
28	2	1
31	2	2
34	1	1
35	1	1
36	2	2
39	2	2
40	1	2
Mediana	2	2
Variação interquartil	1	1
Grupo B		
1	2	1
4	2	2
7	2	2
8	2	2
9	2	2
11	1	1
13	2	1
14	2	2

17	2	2
20	1	2
21	1	1
23	2	2
24	2	2
29	2	2
30	1	1
32	2	2
33	1	1
37	1	1
38	1	2
41	2	2
Mediana	2	2
Variação interquartil	1	1

APÊNDICE IV - QUESTIONÁRIO APLICADO PARA CAPTURAR A PERCEPÇÃO DOS ALUNOS SOBRE A VARIÁVEL Y.7 – ADEQUAÇÃO

Por favor, para cada afirmação abaixo marque a sua opinião:

O conteúdo de ensino do DesigMPS é relevante.

( ) Discordo fortemente - ( ) Discordo - ( )Neutro - ( ) Concordo - ( ) Concordo fortemente

Jogar o DesigMPS é suficiente para apoiar o ensino da modelagem de processos de software.

( ) Discordo fortemente - ( ) Discordo - ( )Neutro - ( ) Concordo - ( ) Concordo fortemente

Cada nível do DesigMPS testou progressivamente o conhecimento sobre modelagem de processos de software.

( ) Discordo fortemente - ( ) Discordo - ( )Neutro - ( ) Concordo - ( ) Concordo fortemente

Houve aumento coerente do nível de dificuldade entre os níveis do DesgiMPS.

( ) Discordo fortemente - ( ) Discordo - ( )Neutro - ( ) Concordo - ( ) Concordo fortemente

A abordagem para ensinar modelagem de processo de software pelo DesigMPS é adequada.

( ) Discordo fortemente - ( ) Discordo - ( )Neutro - ( ) Concordo - ( ) Concordo fortemente

O DesigMPS é atrativo.

( ) Discordo fortemente - ( ) Discordo - ( )Neutro - ( ) Concordo - ( ) Concordo fortemente

APÊNDICE V – PONTUAÇÕES DOS PRÉ/ PÓS-TESTES

Grupo A (Experimental)	Pontuações do pré-teste (nível cognitivo aplicação)					
ID aluno	Y.1 Elementos do processo	Y.2 Descrição dos elementos do processo:	Y.3 Relacionamentos entre os elementos do processo	Y.4 Uso correto do SPIDER_ML para representar Y.1 e Y.3	Total	Percentage-Mastery scale
2	0,68	1	0,605	1	3,285	82,125
3	1	0,76	1	1	3,76	94
5	0,442	1	0,55	0,33	2,322	58,05
6	0,64	1	0,6	0,4	2,64	66
10	1	0,8	1	1	3,8	95
12	0,66	0,75	0,75	0,32	2,48	62
15	0,3	0	0	0	0,3	7,5
16	0,4	1	0,75	1	3,15	78,75
18	0,7	0,75	0,25	0,3	2	50
19	0,2	0,15	0	0	0,35	8,75
22	0,2	0,5	0	0,2	0,9	22,5
25	0,804	0,75	0,75	1	3,304	82,6
26	0,4	0,25	0,35	0,8	1,8	45
27	0,722	0,5	0,5	1	2,722	68,05
28	0,5	0,75	0,5	0,8	2,55	63,75
31	0,8	0,82	0,5	0,4	2,52	63
37	0,4	0,7	0,6	0,6	2,3	57,5
38	0,62	0,5	0,55	1	2,67	66,75
39	0,8	0,75	0,5	0,5	2,55	63,75
40	0,46	0,75	0,25	0	1,46	36,5
41	0,4	0,9	0,25	0,3	1,85	46,25

Grupo A (Experimental)	Pontuação do pós-teste (nível cognitivo aplicação)					
ID aluno	Y.1 Elementos do processo	Y.2 Descrição dos elementos do processo:	Y.3 Relacionamentos entre os elementos do processo	Y.4 Uso correto do SPIDER_ML para representar Y.1 e Y.3	Total	Percentage-Mastery scale
2	0,8	0,965	0,77	1	3,535	88,375
3	1	1	1	1	4	100
5	0,6	1	0,75	0,82	3,17	79,25
6	0,8	1	0,555	1	3,355	83,875
10	0,774	0,9	0,9	1	3,574	89,35
12	0,8	0,75	0,85	0,8	3,2	80
15	0,6	0,8	0,6	0,7	2,7	67,5
16	0,5	1	1	1	3,5	87,5
18	0,42	0,85	0,72	1	2,99	74,75
19	0,56	0,7	0,5	0,73	2,49	62,25
22	0,9	0,8	0,61	1	3,31	82,75
25	0,8	0,565	0,55	1	2,915	72,875
26	0,74	0,75	0,61	1	3,1	77,5
27	0,42	0,905	0,45	1	2,775	69,375
28	0,46	0,85	0,15	0,7	2,16	54
31	0,8	1	0,565	1	3,365	84,125
37	0,54	0,5	0,7	1	2,74	68,5
38	0,6	0,7	0,6	1	2,9	72,5
39	0,8	0,85	0,6	0,9	3,15	78,75
40	0,74	0,75	0,5	1	2,99	74,75
41	0,8	1	0,5	1	3,3	82,5

Grupo B (Controle)	Pontuação no pré-teste (nível cognitivo aplicação)					
ID Aluno	Y.1 Elementos do processo	Y.2 Descrição dos elementos do processo:	Y.3 Relacionamentos entre os elementos do processo	Y.4 Uso correto do SPIDER_ML para representar Y.1 e Y.3	Total	Percentage-Mastery scale
1	1	1	0,9	1	3,9	97,5
4	0,4	0,5	0,205	1	2,105	52,625
7	0,2	0	0	0,4	0,6	15
8	0,7	0,75	0,75	0,2	2,4	60
9	0,28	0,5	0,5	0,5	1,78	44,5
11	0	0	0	0	0	0
13	0,6	0,75	0,5	0,25	2,1	52,5
14	0,3	0,5	0,25	0,5	1,55	38,75
17	0,45	0,5	0,5	0,25	1,7	42,5
20	0,4	0,1	0,5	0,3	1,3	32,5
21	0,2	0,25	0,25	0,7	1,4	35
23	0,72	1	0,5	0,76	2,98	74,5
24	0,4	0,5	0,5	0,5	1,9	47,5
29	0,24	0,25	0,25	0,2	0,94	23,5
30	0,8	0,75	0,75	0,5	2,8	70
32	0,4	0,65	0,5	1	2,55	63,75
33	0,6	1	0,5	0,5	2,6	65
34	0,8	1	0,5	0,8	3,1	77,5
35	0,26	0,55	0,6	0,3	1,71	42,75
36	0,4	0,5	0,4	0,4	1,7	42,5

Grupo B (Controle)	Nota Pós-teste (nível cognitivo aplicação)					
ID aluno	Y.1 Elementos do processo	Y.2 Descrição dos elementos do processo:	Y.3 Relacionamentos entre os elementos do processo	Y.4 Uso correto do SPIDER_ML para representar Y.1 e Y.3	Total	Percentage-Mastery scale
1	0,8	0,8	0,86	1	3,46	86,5
4	0,834	0,7	0,7	1	3,234	80,85
7	0,2	0	0	0,6	0,8	20
8	0,7	1	0,75	0,34	2,79	69,75
9	0,4	0,5	0,5	0,36	1,76	44
11	0,64	0,6	0,43	0,2	1,87	46,75
13	0,6	0,75	0,5	0,5	2,35	58,75
14	0,26	0,5	0,5	0,2	1,46	36,5
17	0,2	0,5	0,5	0,2	1,4	35
20	0,8	0,75	0,5	0,5	2,55	63,75
21	0,68	0,5	0,5	0,49	2,17	54,25
23	0,42	0,75	0,75	0,39	2,31	57,75
24	0,6	0,5	0,645	1	2,745	68,625
29	0,46	0	0,58	1	2,04	51
30	0,84	1	0,56	1	3,4	85
32	0,62	0,55	0,5	0,6	2,27	56,75
33	0,4	1	0,85	0,8	3,05	76,25
34	0,9	0,75	0,5	1	3,15	78,75
35	0,3	0,75	0,15	0,7	1,9	47,5
36	0,54	0,15	0,8	0,4	1,89	47,25

Grupo A (experimental)	Pontuação (nível de conhecimento)	
ID aluno	Pré-teste	Pós-teste
2	10	10
3	10	10
5	10	10
6	9	10
10	10	10
12	10	9
15	10	10
16	9	10
18	10	10
19	10	10
22	9	10
25	10	10
26	10	10
27	10	10
28	9	10
31	10	9
34	9	10
35	10	10
36	10	10
39	8	10
40	10	10

Grupo B (Controle)	Pontuações (nível de conhecimento)	
ID aluno	Pré-teste	Pós-teste
1	9	10
4	9	10
7	9	10
8	8	10
9	10	10
11	9	10
13	9	9
14	10	10
17	7	10
20	10	10
21	10	10
23	10	10
24	8	10
29	8	10
30	10	10
32	10	10
33	10	10
37	10	10
38	10	10
41	10	10

Grupo A (experimental)	Pontuações (nível de compreensão)	
ID aluno	Pré-teste	Pós-teste
2	8.0	9.5
3	10.0	9.0
5	6.5	8.0
6	8.8	9.7
10	9.5	9.5
12	7.5	8.2
15	2.8	7.0
16	7.5	8.2
18	5.7	7.5
19	3.5	6.0
22	4.3	8.0
25	8.0	9.5
26	6.0	7.5
27	7.0	8.4
28	7.5	9.2
31	8.0	9.0
34	7.0	8.8
35	7.2	8.0
36	5.5	7.5
39	8.5	9.0
40	7.5	7.7

Grupo B (Controle)	Pontuações (nível de compreensão)	
	Pré-teste	Pós-teste
1	9	9.5
4	6	8.3
7	4	6.5
8	8.5	9
9	6.5	8.5
11	3.5	5.7
13	6.8	8
14	5.5	6.2
17	6	5.6
20	7	6.5
21	5.7	7.5
23	9.7	9
24	5.5	7
29	4.2	4.5
30	8.5	8.5
32	7	7.7
33	8.2	8
37	8	8.2
38	5	8.5
41	4.5	9.6