Extração de casos de teste a partir de modelos de processos de negócio

Henrique Prado Sousa[1], André Luiz de Castro Leal[1,2], Arndt von Staa[1], Julio Cesar Sampaio do Prado Leite[1]

[1] Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Rio de Janeiro, RJ, Brasil

[2] Departamento de Matemática
Universidade Federal Rural do Rio de Janeiro (UFRRJ)
BR-465, Km 7, Seropédica, Rio de Janeiro - CEP 23890-000
hps.infotec@gmail.com, andrecastr@gmail.com, arndt@inf.puc-rio.br, www.inf.puc-rio.br/~julio

Resumo. A modelagem de processos de negócio é um recurso que vem sendo empregado com diversas finalidades, entre as quais, planejamento e alinhamento estratégico. Na engenharia de requisitos, os modelos de processos de negócio também são utilizados como insumos na extração de requisitos de software, os quais deverão ser utilizados com base para testes posteriores. Neste sentido, propomos um método de extração de suítes de testes a partir de modelos de processo de negócio. A identificação dos casos de testes é baseada na junção de heurísticas de identificação de requisitos e novas heurísticas inspiradas em métodos de geração de casos de testes a partir de casos de uso e diagrama de estados. O produto é uma suíte de casos de teste definida ainda nas fases iniciais do desenvolvimento, com base em informações de contexto.

Keywords: casos de teste, suíte de teste, software, processos de negócio, BPM.

1 Introdução

Nos projetos de desenvolvimento de software, a correta definição dos requisitos ainda é um desafio para a Engenharia de Requisitos (ER). Como fator agravante, o custo da identificação de problemas inseridos nas fases iniciais da construção de software é crescente ao longo do ciclo de desenvolvimento, muitas vezes de forma exponencial [6] [20]. Mesmo quando bem definidos, os requisitos podem sofrer problemas de implementação, projeção de arquitetura e tecnologias, e acarretar prejuízos crescentes ao longo do tempo incorridos nas tentativas de reverter o problema.

O custo relativo para corrigir um defeito nas diferentes fases de construção de um software se eleva conforme o andamento do projeto. Este custo é menor durante as fases iniciais porque ainda existem poucos artefatos que possam vir a necessitar de correção [18]. Na existência de erros, o maior número de produtos desenvolvidos provavelmente demandará um efeito cascata para a correção dos erros encontrados.

Alguns projetos de desenvolvimento de software gastam de trinta a cinquenta por cento do orçamento na realização de testes, entretanto, a maioria dos usuários identifica que os softwares não são bem testados antes da entrega [14]. Uma possível explicação para a ineficiência dos testes é a sua realização apenas na última etapa do processo de desenvolvimento, momento em que orçamento e prazo já se encontram bastante limitados. Somado a isso, além de ser uma tarefa complexa, muitas vezes, os testes são realizados sem seguir métodos bem definidos.

Para auxiliar a tarefa de testes de software, o presente trabalho sugere a projeção de suítes de testes logo nas fases iniciais do desenvolvimento, com base em informações do contexto presentes em modelos de processos de negócio, as quais também poderão ser utilizadas na

elicitação de requisitos. Os casos de teste são definidos a partir da identificação de funcionalidades em um modelo de processo, bem como outras características pesquisadas nas maiorias dos métodos de geração de casos de teste a partir de artefatos de requisitos. Como os modelos de processos de negócio são elementos fundamentais nesta proposta, ela somente se aplica aos projetos os quais possuem este artefato disponível.

Com uma suíte de testes projetada praticamente em conjunto com os requisitos, espera-se que ela oriente a fase de construção do software ao fornecer informações prévias sobre o fluxo dos casos de testes que deverão ser realizados, produzindo um efeito colateral que pode ser benéfico à qualidade dos artefatos a serem produzidos. Isso pode ocorrer porque a formulação de casos de teste requer a identificação dos cenários, a valoração dos dados e a definição de oráculos ou resultados esperados, o que reforçará a demanda pelo o conhecimento necessário do domínio para a definição desses elementos. Os processos que geram esses artefatos são, portanto, benéficos, porque antecedem informações que podem auxiliar ao engenheiro de requisitos na identificação de possíveis erros, impedindo que sejam propagados para artefatos/produtos posteriores. Reduz-se assim o custo do projeto através da redução significativa do retrabalho.

Outro benefício é que o uso de modelos de processos de negócio como insumo em um processo de elicitação beneficia o alinhamento dos requisitos ao negócio, desta forma, os casos de testes extraídos do mesmo modelo, e incorporando o mesmo processo de identificação de requisitos, além de garantir uma boa cobertura, também incluirá nos testes cenários alinhados ao negócio.

Diante disso, o objetivo principal deste trabalho é apresentar como os modelos de processos de negócio podem contribuir para a definição de casos de testes de software. Entendemos que a modelagem de processos possui grande variação de elementos com semântica rica, e conseguem representar os requisitos de sistema em diferentes níveis de abstração. A partir disso, unimos esta capacidade a algumas heurísticas de um método de identificação de serviços, e novas heurísticas complementares baseadas em outros métodos de definição de casos de teste presentes na literatura. Para alcançar este objetivo, iniciamos com o estudo de métodos para geração de casos de teste a partir de artefatos de requisitos. Isso permitiu a identificação dos elementos necessários para a definição dos casos de teste e as técnicas de extração das informações para a composição dos casos de teste a partir de processos de negócio. Por fim, criamos um pequeno exemplo de aplicação destas heurísticas e analisamos os resultados baseado no nível de detalhe existente no modelo e nível de abstração.

O presente artigo está dividido nas seguintes seções: na seção 2, estão descritas algumas características de uso de requisitos na extração de testes; na seção 3 são analisadas as características presentes em um modelo de processo de negócio em relação aos elementos necessários para permitir a extração dos casos de teste, na seção 4, discorre-se a respeito do método de identificação de requisitos a partir de modelos de processos de negócio; na seção 5 é apresentado o método proposto; e por fim, na seção 6, estão as conclusões.

2 Uso de artefatos de requisitos na extração de testes

Nos processos de desenvolvimento de software tradicionais, as atividades de teste costumam ser uma das últimas a serem realizadas. Este tipo de abordagem possui falhas, uma vez que são comuns os erros na definição e/ou interpretação dos requisitos, o que resulta em projeto e implementação do software de forma equivocada. Os requisitos definem o que o software deve fazer e como se comportar, sendo obrigatório satisfazê-los integralmente para que o projeto de software alcance os seus objetivos. Por isso, é importante considerar os requisitos na elaboração de testes, a fim de verificar se o produto de software corresponde às necessidades esperadas.

Os requisitos são expressos de diversas formas, comumente utilizando documentos textuais e modelos. Devido ao maior grau de formalidade, os modelos são os artefatos mais

utilizados na elaboração de testes de software. Entre os modelos mais usados, são os que compõem a UML, como o OCL (*Object Constraint Language*), e diagramas de objetos e especificação do tipo Z [21]. Cada modelo oferece um conjunto de informações distintas que, baseado em suas características, auxiliam a geração de testes em diferentes níveis, por exemplo, testes de sistema, testes de integração, testes de unidade, testes de regressão, e testes de *stress* [21]. Entretanto, é importante destacar que a abordagem de testes é baseada em requisitos, portanto, demanda modelos corretamente descritos. Uma forma de verificar isso é através da aplicação das propostas de checagem de modelo [19], [12], [21], muitas delas desenvolvidas para se integrarem às abordagens de geração de testes a partir de requisitos.

Os modelos de requisitos utilizados como insumo devem oferecer um conjunto de informações mínimas que possam ser extraídas e disponibilizadas para a construção de casos de testes. Devem contemplar, se possível, cada uma das possibilidades de desvios na execução de procedimentos que visam à satisfação de um requisito. Independente dos artefatos utilizados na geração de casos de teste, os seguintes elementos são necessários: comportamento do sistema [21], responsável por definir como ele deve responder aos comandos do usuário e aos estados alcançados; informações que são tratadas e geradas no requisito, ou seja, os dados (em seu devido formato) necessários para oferecer o insumo correto na execução dos testes e a verificação do produto final; momento de inicio de execução do requisito, caracterizado pelo estado que dispara a execução do requisito; produtos gerados na execução do requisito (se houverem); momento de finalização do requisito, caracterizado pelo estado que deve ser alcançado após a execução do teste.

Não são todos os artefatos que possuem este conjunto de informações, por outro lado, os projetos de desenvolvimento de software não produzem todos os artefatos possíveis. Portanto, para cada projeto, deve-se escolher os artefatos de acordo com suas necessidades [13], uma vez que não há melhores práticas definidas no contexto de testes de software [15]. Estas informações sobre os requisitos constituem os elementos básicos necessários para definição de um caso de teste: Estado inicial - momento que caracteriza o início do caso de teste. Dados de entrada - informações necessárias para execução do teste. Comportamento - quais tarefas e como devem ser executadas. Dados de saída - produtos gerados na execução satisfatória do cenário. Estado final - momento que caracteriza o fim do caso de teste (estes elementos são mais detalhados posteriormente).

De forma geral, os métodos buscam identificar as possibilidades de execução dos requisitos no sistema, seus eventos de disparo e eventos finais. A partir deste conjunto de informações, são geradas listas de casos de testes, o que muitas vezes resulta em um número extenso. Alguns autores propõem métodos para reduzir o conjunto de casos de testes com critérios para identificação de casos de teste fora do escopo (ao reduzir o escopo) ou não possíveis de executar no mundo real. Alguns métodos utilizam diagramas individualmente ou combinados [27], de acordo com os tipos de testes desejados que possam ser mais ou menos abstratos e abrangentes.

Além da presença de elementos básicos nos modelos, ainda devem ser definidos os critérios de cobertura, que consistem em regras que visam alcançar de forma adequada os cenários possíveis baseados em suas características e recursos oferecidos no modelo. Por exemplo, [28] aplica um conjunto de critérios específicos como métrica para cobertura dos requisitos a partir de diagramas de estados. Alguns deles são: Critério de cobertura dos estados: Todos os estados de um diagrama de estados devem ser atingidos pelo menos uma vez. Critério de cobertura das transições: Todas as transições devem ser exercidas. Critério de cobertura dos caminhos: Todos os caminhos criados segundo critérios de seleção devem ser exercidos. Condição de cobertura: Uma condição é considerada coberta se o resultado de sua avaliação for verdadeiro ou falso em algum momento durante a execução do teste.

3 Características dos modelos de processos de negócio

Modelos de processos de negócio podem ser utilizados como material de apoio na gestão da organização, bem como em projetos de informatização dos processos. Para isso são projetados em momento anterior ao desenvolvimento de software, com informações que permitem indicar onde, quando e como as funcionalidades são executadas no processo. Um modelo de processo de negócio pode ser utilizado tanto para conhecimento do domínio como uma fonte de informação para a elicitação de requisitos. Em [4] é proposto um conjunto de heurísticas para extração de requisitos a partir de modelos de processo de negócio, de forma automática.

Os modelos de processos de negócio possuem a capacidade de expressar grande número de elementos de negócio em seus diagramas, como fluxo de atividades, eventos iniciais, intermediários e finais, atores, fluxo de informações, regras e requisitos de negócio, pontos de controle, artefatos, termos, telas, regras de negócios.

Outra característica importante é que um modelo que apresenta o nível de fluxo de atividades de um processo de negócio detalha os diversos caminhos que uma instancia de processo pode seguir baseado em seus possíveis estados intermediários. Estes estados intermediários podem ser comparados aos fluxos alternativos de um caso de uso, ou ainda a um diagrama de estados, portanto, cada possível caminho pode gerar um caso de teste. Os elementos representativos que compõem modelos de processo de negócio podem representar informações suficientes para a geração de casos de teste devido a sua riqueza de representação. Além disso, é possível explorar ainda outros elementos que detalham mais as informações e procedimentos envolvidos nas execuções das atividades, por exemplo, as regras de negócio e os requisitos de negócio.

A execução de uma atividade em um processo deve contribuir de alguma forma ao negócio, ou seja, deve agregar valor para algum elemento envolvido no contexto, uma vez que, se atividade não traz nenhum benefício ao processo, ela não deveria existir.

Portanto uma atividade, mesmo que com pequena contribuição, leva o processo de um estado para outro, ainda que esta mudança de estado não signifique um elemento "concreto" para a organização. Algo sempre é processado, mesmo que manualmente, resultando na transformação de um elemento alvo (ex. informação ou artefato), mesmo que seja, por exemplo, uma alteração do estado de uma informação, que sai da memória de um computador para um papel impresso.

As transformações realizadas nos elementos que trafegam pelos processos de negócio podem ser executadas integralmente por meio computacional, parcialmente com o auxílio de software operados por pessoas ou completamente manual. Algumas das atividades manuais podem posteriormente sofrer alterações em sua execução caso a organização implante um sistema para apoiar os processos, ou ainda, serem substituídas.

Quando tratamos processos que já possuem o mapeamento de sistemas, podemos considerar que na dada atividade onde o sistema aparece há um ou mais requisitos implementados. Em uma mesma atividade é possível que o papel executor faça a leitura de um cadastro (funcionalidade 1), altere as informações (funcionalidade 2), obedecendo a uma regra de negócio (funcionalidade 3), e ao final grave a informação (funcionalidade 4). Neste exemplo, identificam-se quatro funcionalidades de sistema que podem estar mapeadas em uma única atividade. Entretanto, essas funcionalidades representam requisitos que possuem características atômicas e podem compor requisitos mais abstratos, que abrangem maior número de funcionalidades. Partindo deste pressuposto, o processo também pode ser analisado considerando seus fluxos de atividades. Por exemplo, um ciclo presente em um processo poderia representar um requisito que é executado diversas vezes até alcançar um dado estado que permita prosseguir com o processo. O ciclo pode ser composto por um conjunto de atividades que possuem diversas funcionalidades e regras de negócio.

Entre as diversas notações que estão disponíveis para a modelagem de processos, a UCM (*Use Case Map*) [31], apresenta características especiais que demonstram a utilidade dos modelos de processos de negócio na extração de casos de teste. Ela foi inicialmente desenvolvida para

modelar cenários comportamentais durante o projeto de alto nível em sistemas reativos distribuídos. Posteriormente foi considerado um recurso adequado e útil para a modelagem e análise de processos de negócio, e que satisfaz os objetivos de uma linguagem BPM [10].

A reutilização de uma notação inicialmente projetada para a modelagem de cenários, na modelagem de processo, evidencia que se uma linguagem para modelagem de cenários pode ser abstraída para a modelagem de processos de negócios, então também poderíamos utilizar a notação de processos de negócio para gerar casos de teste, uma vez que é possível gerar testes a partir de cenários.

Entretanto, existem outras as notações com maior capacidade de representação de elementos do negócio [22], tais como a BPMN [8] e a EPC (*Event-driven Process Chain*) [29], que podem ser utilizadas em conjunto ao método proposto neste trabalho [7], [22], [30].

4 Identificação de requisitos a partir de processos de negócio

A modelagem de processos geralmente segue a representação dos processos organizacionais em diferentes níveis (três níveis) de abstração: o primeiro nível é formado pelos macroprocessos (nível mais abstrato, respectivo à cadeia de valor organizacional); o segundo nível detalha o fluxo de atividades do processo, composto por um ou mais papéis executores, atividades, fluxos, operadores lógicos e eventos; o terceiro nível detalha o nível operacional demonstrando informações sobre a execução de cada atividade e os elementos envolvidos [29]. A partir do segundo e terceiro nível é possível identificar requisitos de software, uma vez que estão presentes elementos operacionais.

O método de identificação de requisitos sugerido nesse trabalho é adaptado do método de identificação de serviços a partir de modelos de processos de negócio [2], [3], seguindo uma análise *top-down* dos modelos. Essa análise identifica os requisitos baseados nos objetos que compõem o modelo, seja por sua definição (por exemplo, regras de negócio, requisitos de negócio e *cluster*) ou por características padronizadas no fluxo de atividades do processo (por exemplo, ciclo de atividades e sequência de atividades) [22].

Os requisitos identificados através deste método são considerados "requisitos candidatos", uma vez que se torna necessária a verificação posterior de um engenheiro de requisitos para identificar entre os elementos extraídos, quais não são de interesse. Observa-se que o método é baseado em heurísticas, e suas identificações não necessariamente podem estar no escopo de um dado projeto. O "Requisito candidato" é uma abstração (não implementada) de um requisito, que pode vir a ser implementada como um serviço físico (por exemplo, como *Web Service*) ou como uma funcionalidade de uma aplicação.

O método de identificação de requisitos candidatos é subdividido em três etapas: A) Seleção de atividades: são selecionadas atividades automatizadas (executadas por sistema), apoiadas por sistema (executadas por pessoas, mas utilizando um sistema) e automatizáveis (atividades manuais que se pretende automatizar). Atividades manuais e que não estão sendo consideradas para automatização são descartadas, já que não há requisitos envolvidos. B) Identificação de requisitos candidatos: os modelos de processos de negócio são analisados de acordo com um conjunto de heurísticas definidas por [2], [3]. As heurísticas analisam a estrutura e semântica dos modelos de processos: - Para análise da estrutura baseiam-se nos padrões de *workflow* [5], [23] e [26] e cobrem todas as possibilidades de fluxos de atividades que podem ser representadas por um modelo de processo; Para análise semântica, são considerados os elementos dos modelos, tais como os requisitos de negócio, informações de entrada e saída e regras de negócio. A semântica destes elementos indica as funcionalidades que podem ser implementadas em serviços de apoio ao processo. O resultado desta etapa é a identificação de requisitos candidatos de dados (executam operações CRUD (*Create, Read, Update, Delete)* sobre os dados) e/ou serviços candidatos de lógica (executam regras de negócio e eventualmente podem incluir

operações CRUD). C) Consolidação de requisitos candidatos: na terceira etapa, os requisitos candidatos identificados na etapa anterior são analisados e são geradas informações referentes à granularidade, grau de reuso, dependência entre serviços, associações dos serviços com as atividades de onde foram identificados, papéis que os executam e sistemas que poderão invocá-los.

As informações geradas na etapa de consolidação são apresentadas em tabelas, gráficos e grafos de dependência, para tornar possível a manipulação dos resultados pela equipe de requisitos. Estas informações são utilizadas nas fases de análise e projeto de software, a fim de definir a melhor forma de implementação do conjunto de requisitos candidatos [22]. Assim uma representação da proposta seria a partir de:

⊢→ MODELO DE NEGÓCIO → HEURÍSTICAS → REQUISITOS CANDIDATOS

Considerando que apesar da UCM ser utilizada para a modelagem de processos de negócio e que alguns autores ainda a consideram inadequada [24], [17], [7] por oferecer poucos recursos, então podemos esperar que uma notação mais abrangente como a BPMN [8] possua maior capacidade de oferecer a geração de casos de teste devido a maior riqueza sintática e semântica de seus componentes, uma vez que a UCM, mesmo com suas limitações, é utilizada com este objetivo [1].

5 Método para geração de casos de teste a partir de processos de negócio

Os modelos de processos de negócio podem expressar grande número de elementos do negócio, podendo representar diversas camadas de abstração, como também diferentes níveis de detalhe. Portanto, existe um problema de abstração e detalhamento que afeta o resultado dos casos de teste. A geração de casos de testes estará limitada pela complexidade e detalhamento do conteúdo do modelo de processo. Por exemplo, um diagrama não precisa conter obrigatoriamente detalhes sobre regras de negócio relacionadas às atividades as quais se aplicam, ou uma atividade pode ser abstrata o suficiente de forma a "esconder" detalhes que definem outras atividades ou ficando muito densa, com diversos elementos de entrada e saída e muitas tarefas em sua descrição.

Independente desses fatores, o mais importante é que a modelagem esteja adequada (tanto no contexto da notação quanto na qualidade de corretude do mapeamento dos elementos do negócio para o modelo), já que o produto desta proposta será uma *suíte* de casos de teste que deverá ser criticada por um engenheiro de software. A necessidade de mais ou menos informação, inclusão de novos casos, exclusão de testes similares ou fora de escopo, é de sua responsabilidade.

Na identificação de requisitos a partir de processos de negócio, as heurísticas geram "requisitos candidatos". A palavra "candidato" foi incluída justamente porque, devido à aplicação das heurísticas e à natureza automatizada das identificações, não se pode afirmar que estes elementos são de fato requisitos de um projeto [2]. O mesmo pensamento é aplicado à lista de casos de teste resultante do método. Outro fator importante é o da cobertura da identificação de requisitos que, segundo [25], é um problema NP. Portanto, isso justifica o uso de heurísticas, já que definir um método preciso e plenamente correto ainda é considerado inviável.

Dividida em passos, o processo de geração de casos de teste a partir de processos de negócio pode ser resumidos cômoda seguinte forma: PRIMEIRO PASSO: consiste na modelagem dos processos de negócio. Recomendamos os cuidados na modelagem das regras de negócio e nas boas práticas de modelagem. Cada atividade deve agregar valor ao negócio, ou seja, ela deve alterar o estado de algum elemento do processo, auxiliando em algum nível o processo a alcançar seus objetivos. Assim, as atividades representarão tarefas e elementos que os sistemas de software muito possivelmente deverão abordar em suas funcionalidades. SEGUNDO PASSO: consiste na aplicação do método de identificação de requisitos, apresentado na sessão anterior. A aplicação deste método é importante para que os requisitos sejam identificados ao longo dos elementos de processo de negócio, já que poderemos agrupá-los posteriormente à lista de funcionalidades a serem testadas nos casos de teste. Um caso de teste não verificará uma, e somente uma

funcionalidade do sistema, uma vez que um requisito pode agregar muitas funcionalidades que devem ser testadas, ou seja, torna-se um requisito "denso", principalmente devido ao seu nível de abstração. TERCEIRO PASSO: consiste em delimitar os casos de teste. Neste passo, baseamo-nos nos métodos de geração de casos de testes aplicados em outros modelos, por exemplo, um diagrama de estados ou caso de uso. Um caso de uso pode possuir diversos "caminhos" a serem testados, já que define além do fluxo principal, os "n" possíveis fluxos alternativos. Isso se assemelha a um processo de negócio, que através dos operadores lógicos e eventos, criam diversos caminhos possíveis para a execução do processo. QUARTO PASSO: os engenheiros de software devem criticar a lista final de casos de teste de forma a identificar quais se aplicam às suas necessidades de teste, que podem, por exemplo, ser reduzida pelo escopo de atuação dos testes que desejam no dado momento.

No método proposto, dois elementos são variantes: os requisitos candidatos identificados através de elementos explícitos que afetam diretamente os requisitos do sistema; e os requisitos candidatos identificados através de sequências recorrentes nos processos (padrões de *workflow*) que podem representar funcionalidades compostas que possuem grande potencial de reuso.

No primeiro caso, o método de identificação de requisitos [2] busca por elementos específicos que expressam diretamente "requisitos básicos" em um sistema: regras de negócio, informações de entrada e saída que sofrem operações CRUD, e requisitos de sistemas explícitos[1]. Neste trabalho, utilizamos as seguintes heurísticas de requisitos básicos (HRB) (adaptado de [4], [22]):

HRB1 (regra de negócio): Um requisito candidato deve ser identificado a partir de uma regra de negócio.

HRB2 (informação de entrada e saída): Um requisito candidato deve ser identificado para cada informação de entrada de uma atividade (caracterizado como requisito de leitura), assim como um requisito candidato de dado deve ser identificado para cada informação de saída de uma atividade (caracterizado como requisito de escrita) desde que as informações estejam associados a portadores de informação (por exemplo, banco de dados, e-mail e documentos eletrônicos).

Adicionamos nova heurística para abordar as atividades automatizadas, uma vez que representam uma funcionalidade e devem ser listados como itens para possíveis testes isolados.

HRB3 (atividades automatizadas): Um requisito candidato deve ser identificado para cada atividade automatizada.

No segundo caso, o método de identificação de requisitos [2] busca pelos fluxos de processo que se enquadram nos padrões de *workflow*. Estes padrões identificam fluxos específicos que evidenciam funcionalidades dentro de um processo devido a sua característica de execução sequencial de atividades e repetição. Por exemplo, podemos citar um fluxo de ciclo que pode ser executado diversas vezes seguidas até um evento ser satisfeito. A repetição característica desse fluxo remete a uma funcionalidade bem definida que é executada repetidamente enquanto um estado for verdadeiro. Neste trabalho, utilizamos as seguintes heurísticas de padrão de *workflow* (HPW) (adaptado de [4], [22]):

HPW1 (atividades sequenciais): Um requisito candidato deve ser identificado a partir de um conjunto de atividades sequenciais automatizadas ou apoiadas por sistema (uma atividade sequencial é definida por uma sequência de no mínimo duas atividades executadas seguidamente, sem a interferência de operadores lógicos (OR, XOR ou AND)).

HPW2 (ciclo de atividades - *Loop*): Um requisito candidato deve ser identificado a partir de uma estrutura do workflow onde uma ou mais atividades podem ser executadas repetidamente.

Ainda existem as heurísticas de padrão de workflow OR, AND, XOR, Interface de processo e atividades de múltiplas instâncias que podem ser acessadas em [4] e [22].

---

[1] A maioria das notações não oferece o elemento "Requisito de sistema".

Os fluxos de processos também definem os diferentes cenários de um processo. Estes cenários são divisões que ocorrem a partir dos operadores lógicos, e os caminhos a seguir são definidos através dos diferentes eventos intermediários. A partir dos padrões de *workflow*, da representação de possíveis cenários nos processos de negócio e estudo dos critérios de cobertura presentes em [28], propomos duas novas "heurísticas de cobertura de cenários" (HCC):

HCC1 (eventos inicial e final): um fluxo de cenário é identificado quando houver uma sequência de atividades presentes entre um evento inicial e um evento final. Se houver mais de um fluxo a ser percorrido, todos deverão ser considerados. A partir dessas heurísticas, os fluxos dentro de um processo serão identificados e corresponderão a um caso de teste. Unindo o procedimento de identificação de requisitos ao procedimento de identificação de fluxos, as funcionalidades (em um baixo nível) que devem ser testadas nos casos de teste são definidas, podendo ser mais ou menos granulares, de acordo com o numero de funcionalidades que deverão ser abordadas em cada caso.

HCC2 (eventos intermediários): um fluxo de cenário é identificado quando houver um fluxo de atividades presentes entre dois eventos intermediários, evento inicial e intermediário ou evento intermediário e evento final. Esta heurística permite a identificação de fluxo e subfluxos que particionam o processo de negócio em seus diferentes ramos. Um fluxo pode possuir subfluxos quando uma partição maior possui em seu interior outras partições, entretanto, não necessariamente esses fluxos existirão, podendo apenas ocorrer o fluxo principal, delineado pelo evento inicial e final, os quais também são respectivamente eventos iniciais e finais do processo.

Baseado nas heurísticas apresentadas, definimos dois critérios de cobertura (CC):

CriCob1 (cobertura de requisitos): todos os requisitos identificados pelas heurísticas devem ser considerados nos casos de testes pelo menos uma vez, ou seja, os requisitos devem estar presentes em algum fluxo extraído do processo de negócio.

CriCob2 (cobertura de fluxos): todos os fluxos identificados pelas heurísticas devem ser considerados nos casos de testes pelo menos uma vez.

O produto final da aplicação das heurísticas é uma lista composta por casos de teste (suíte de testes). Se for desejado utilizar casos específicos, deverá ser feita a análise e seleção dos casos de testes convenientes. As informações consolidadas presentes na lista são compostas pelas informações oferecidas no modelo através de seus elementos. Por exemplo, um objeto representando um artefato pode disponibilizar informações sobre o conteúdo deste artefato. Se este conteúdo for relevante, deve estar presente na lista de casos de teste.

Um conteúdo importante para um artefato de informação que serve como insumo em uma atividade pode ser o "tipo de dado" o qual esse elemento representa. Desconsiderando a possibilidade de informações extras, definimos os elementos básicos que devem compor o caso de teste como: Caso de teste - representa o nome do caso de teste. Normalmente representado apenas pela enumeração da lista. Condição de cenário - representa a condição de disparo do caso de teste, ou seja, o evento inicial ou intermediário do fluxo que representa o cenário. Entradas - representa as informações necessárias para a realização do cenário, ou seja, as informações que servirão como insumo das atividades que compõem o cenário. Informações geradas dentro do cenário não são consideradas como entradas. Requisitos - representa os requisitos envolvidos no caso de teste, ou seja, uma lista de requisitos candidatos que estão relacionados às atividades que compõem o cenário de teste. Fluxo envolvido - representa o conjunto de atividades que estão envolvidas no cenário, incluindo seus eventos que são classificados como informações de "mensagens". Regras de negócio - representa o conjunto de regras de negócio associadas às atividades que estão envolvidas no cenário. Saídas - representam as informações (ou tipos/objetos) e artefatos gerados como produto da execução satisfatória do cenário. Elementos parciais gerados durante a execução do cenário não são consideradas como produtos de saída. É possível que em alguns casos o cenário não gere produtos, mas alcance a satisfação de um estado específico. Resultado esperado - representa a condição de finalização do caso de teste, ou seja, o evento final ou intermediário-final do fluxo que representa o cenário.

Em um exemplo de aplicação é demonstrado como o nível de abstração da modelagem pode afetar o resultado da geração dos casos de testes. O exemplo proposto trata o registro de um cliente, que pode ser composto por pequenas tarefas e regras de negócio. Porém, em um modelo de processo, dependendo da abstração utilizada estes detalhes podem ser reduzidos, transformando-se em informações textuais e não em elementos de modelagem (Fig. 3). Nesta forma a capacidade da aplicação de heurísticas é reduzida, porém não impede a identificação do requisito. No entanto, menos informações estarão disponíveis para extração e geração dos casos de testes, já que as heurísticas de aplicação em modelos não atuam na análise das descrições textuais dos elementos.

O registro de cliente é apenas um trecho de um processo de negócio, no entanto, demonstraremos que um pequeno trecho pode ser rico em informações. Apresentaremos este exemplo de forma visual, através do modelo, como também através de descrição textual. A Fig. 3 apresenta a atividade "Registrar cliente". A descrição da atividade é: a atividade recebe como entrada a senha e o nome do usuário e produz como saída o registro do cliente que é armazenado no banco de dados Clientes. A atividade é apoiada pelo Sistema X. Ao final da atividade, o cliente estará cadastrado na base de dados.

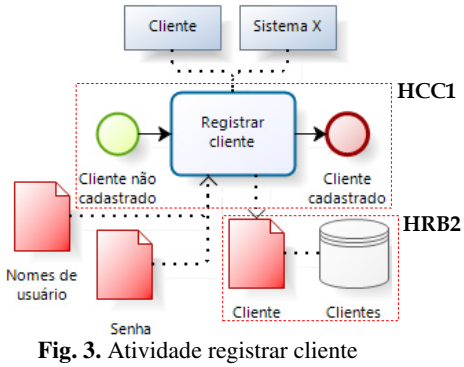

Fig. 3. Atividade registrar cliente

Ao aplicar a heurística de requisitos básicos HRB2 (informação de entrada e saída) (somente este heurística é aplicável), obtemos o requisito candidato descrito na Tabela 1.

Tabela 1. Lista preliminar de requisitos

Índice	Heurística	Requisitos candidatos	Sistemas	Papéis
1	HRB2	Gravar cliente	Sistema X	Sistema X; Cliente;

Posteriormente aplicamos a heurística de cobertura de cenários HCC1 (eventos inicial e final) que identifica o caso de teste descrito na Tabela 2 (a coluna "Heurística" não é acrescentada nas tabelas 2 e 4 devido ao espaço limitado). A Fig. 3 mostra onde as heurísticas são aplicadas.

Tabela 2. Exemplo de caso de teste

Caso de teste	Condição de cenário	Entradas	Requisitos *(Tabela 3)*	Fluxo envolvido	Regra de negócio	Saídas	Resultado esperado
1	Cliente não cadastrado	Nome, Senha	1	**Ação:** Registrar cliente	--x--	Cliente	Evento: Cliente cadastrado

Este caso de teste, ao ser executado, envolverá o conjunto de requisitos que estão presentes no fluxo de processo (neste caso, somente um). Cada requisito é um elemento distinto, porém a união destes elementos definirá o grau de granularidade do caso de teste. É possível identificar que este exemplo de cadastro de cliente é simples, principalmente por não considerar fluxos alternativos. No próximo exemplo será utilizado o mesmo processo, porém, sendo expandido para considerar os fluxos alternativos. Ao expandir este cenário, obtém-se uma visão mais detalhada, desta forma outros elementos surgem, como regras de negócio e novas atividades.

Registro de cliente – Modelo detalhado

A Fig. 4 apresenta o trecho de processo que tem como objetivo o cadastro do cliente. A descrição do processo é: o processo inicia quando o cliente não está cadastrado. O usuário informa nome de usuário e senha, o sistema verifica se o nome de usuário já se encontra registrado na base de dados. Caso o nome de usuário já esteja presente na base de dados, o usuário deverá informar novamente o nome de usuário e senha, porém alterando sua proposta de nome de usuário. Caso o nome do usuário não esteja presente na base de dados, o sistema registra o cliente. Ao final do processo, o cliente estará cadastrado.

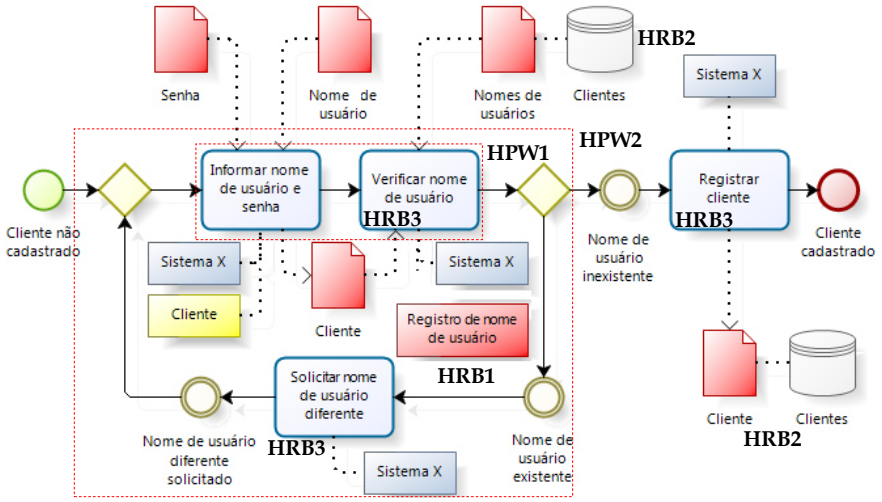

Fig. 4. Trecho de processo referente ao cadastro do cliente

Com este novo modelo, as heurísticas de identificação de requisitos aplicáveis ao modelo se expandem para as demais, apresentadas anteriormente (HRB1 (regra de negócio); HRB2 (informação de entrada e saída); HRB3 (atividades automatizadas); HPW1 (atividades sequenciais), HPW2 (ciclo de atividades - *Loop*)). A Fig. 4 demarca os padrões de *workflow* e registra as sigas das respectivas heurísticas próximas aos objetos nos quais são aplicadas. A Tabela 3 apresenta serviços candidatos a partir da aplicação das heurísticas.

Tabela 3. Requisitos extraídos do modelo expandido

Índice	Heurística	Serviço candidato	Sistemas	Papéis
1	HRB1	Registrar nome de usuário	Sistema X	Sistema X
2	HRB2	Consultar nomes de usuários	Sistema X	Sistema X
3	HRB2	Gravar Cliente	Sistema X	Sistema X
4	HRB3	Verificar nome de usuário	Sistema X	Sistema X
5	HRB3	Solicitar nome de usuário diferente	Sistema X	Sistema X
6	HRB3	Registrar cliente	Sistema X	Sistema X
7	HPW1	Informar nome de usuário e senha_Verificar nome de usuário	Sistema X	Sistema X; Cliente
8	HPW2	Informar nome de usuário e senha_Verificar nome de usuário_Solicitar nome de usuário diferente	Sistema X	Sistema X; Cliente

Posteriormente são aplicadas as heurísticas de cobertura de cenários HCC1 (eventos inicial e final) e HCC2 (eventos intermediários), delimitando os casos de testes. As informações são formatadas conforme a tabela proposta (alguns nomes foram abreviados por motivo de espaço):

Tabela 4. Casos de testes extraídos do modelo expandido

Caso teste	Condição de cenário	Entradas	Req. *(Tab 3)*	Fluxo envolvido	Reg. Neg.	Saídas	Resultado esperado
1	Cliente não cadastrado	Nome, Senha	2, 3, 4, 5, 6, 8	**Ação**: Cliente informa nome de usuário e senha; Sistema verifica nome de usuário; **Mensagem**: Nome de usuário existente; **Ação**: Sistema solicita nome e usuário diferente; **Mensagem**: Nome de usuário diferente solicitado; **Ação**: Cliente informa nome de usuário e senha (inexistentes); Sistema verifica nome de usuário; **Mensagem**: Nome de usuário inexistente; **Ação**: Sistema registra cliente;	1	Cliente	Evento: Cliente cadastrado
2	Cliente não cadastrado	Nome, Senha	2, 3, 4, 6, 7	**Ação**: Cliente informa nome de usuário e senha (inexistentes); Sistema verifica nome de usuário; **Mensagem**: Nome de usuário inexistente; **Ação**: Sistema registra cliente;	1	Cliente	Evento: Cliente cadastrado
3	Cliente não cadastrado	Nome, Senha	2, 4, 7	**Ação**: Cliente informa nome de usuário e senha (inexistentes); Sistema verifica nome de usuário;	1	--x--	Evento: Nome do usuário inexistente
4	Cliente não cadastrado	Nome, Senha	2, 4, 7	**Ação**: Cliente informa nome de usuário e senha (existentes); Sistema verifica nome de usuário;	1	--x--	Evento: Nome do usuário existente
5	Cliente não cadastrado	Nome, Senha	2, 4, 5, 7	**Ação**: Cliente informa nome de usuário e senha (existentes); Sistema verifica nome de usuário; **Mensagem**: Nome de usuário existente. **Ação**: Sistema solicita nome de usuário diferente;	1	--x--	Evento: Nome do usuário diferente solicitado
6	Nome de usuário inexistente	Nome, Senha	3, 6	**Ação**: Registrar cliente;	--x--	Cliente	Evento: Cliente cadastrado
7	Nome de usuário existente	--x--	2, 4, 5, 7	**Ação**: Sistema solicita nome de usuário diferente; **Mensagem**: Nome de usuário diferente solicitado. **Ação**: Cliente informa nome de usuário e senha (inexistentes); Sistema verifica nome de usuário;	1	--x--	Evento: Nome de usuário inexistente
8	Nome de usuário existente	--x--	5	**Ação**: Solicitar nome de usuário diferente;	--x--	--x--	Evento: Nome de usuário diferente solicitado
9	Nome de usuário existente	--x--	2, 3, 4, 5, 6, 7	**Ação**: Sistema solicita nome de usuário diferente; **Mensagem**: Nome de usuário diferente solicitado; **Ação**: Cliente informa nome de usuário e senha; Sistema verifica nome de usuário;	1	Cliente	Evento: Cliente cadastrado
10	Nome de usuário diferente solicitado	Nome, Senha	2, 4, 7	**Ação**: Cliente informa nome de usuário e senha (inexistentes); Sistema verifica nome de usuário;	1	--x--	Evento: Nome de usuário inexistente
11	Nome de usuário diferente solicitado	Nome, Senha	2, 3, 4, 6, 7	**Ação**: Cliente informa nome de usuário e senha (inexistentes); Sistema verifica nome de usuário; **Mensagem**: Nome de usuário inexistente; **Ação**: Sistema registra cliente;	1	Cliente	Evento: Cliente cadastrado
12	Nome de usuário diferente solicitado	Nome, Senha	2, 4, 7	**Ação**: Cliente informa nome de usuário e senha (existentes); Sistema verifica nome de usuário;	1	--x--	Evento: Nome de usuário existente

Esta tabela é construída baseada nos elementos delimitados ("trecho" do processo) pelas HCC1 e HCC2. Os itens básicos para a construção dos casos de testes são extraídos do trecho de processo, tais como, "**Condição do cenário**" (representada pelo evento que inicia o caso de teste), "**Entradas**" (representadas pelos insumos da primeira atividade), "**Saídas**" (representadas pelos produtos da última atividade), "**Fluxo envolvido**" (representado pelas atividades e eventos intermediários que expressam o comportamento de uma funcionalidade composta) e "**Resultado esperado**" (representado pelo evento que finaliza o caso de teste). Além disso, são listados os "**Requisitos**" identificados pelas heurísticas de identificação de requisitos (neste caso HRB1-3, HPW1-2) conforme número de identificação na respectiva tabela (Tabela 3). Da mesma forma são destacadas as "**Regras de negócio**" envolvidas no trecho do processo delimitado (identificadas pela HRB1). A nomenclatura usada nos requisitos candidatos (Tabela 3) é baseada em [2]. A descrição da coluna "Fluxo envolvido", na Tabela 4, foi organizada para descrever o fluxo representando as atividades como "Ações" e eventos como "Mensagens".

Em especial, a coluna "Requisitos" é importante porque descreve os requisitos envolvidos no caso de teste. Entretanto, pode haver redundância neste campo porque alguns requisitos podem ser identificados por diferentes heurísticas e apresentar níveis de abstração distintos. Por exemplo, no caso de teste 1 há um ciclo representado pelo índice 8, que é uma composição dos requisitos de índices 2, 4 e 5. O caso de teste deve ser projetado de acordo com a solução de implementação adotada e projeção de testes dos requisitos. Ou seja, o teste poderá verificar o ciclo como uma funcionalidade, ou considerar separadamente cada um dos requisitos "atômicos" identificados.

Em relação aos critérios de cobertura CriCob1 (cobertura de requisitos) e CriCob2 (cobertura de fluxos) propostos neste trabalho, verifica-se que o primeiro se relaciona com as heurísticas de identificação de requisitos, enquanto o segundo, com as heurísticas de cobertura de cenários. Desta forma, o cumprimento destes critérios depende da relação entre os requisitos presentes na Tabela 3 e os casos de testes da Tabela 4, que já expressam todos os fluxos identificados. Desta forma verifica-se que todos os requisitos (1 a 8) estão presentes em alguma das linhas que compõem a Tabela 4, satisfazendo ambas as heurísticas. No exemplo, os casos de testes 1 e 2 foram identificados pela HCC1 e os casos de teste 3, 4, 5, 6, 7, 8, 9, 10, 11 e 12, pela HCC2. A apresentação de todos os resultados destas heurísticas buscam satisfazer o CriCob2.

Entretanto, para satisfazer o critério de cobertura de fluxos (CriCob2) muitos subfluxos são repetidos em fluxos maiores na lista de casos de teste. Isso evidencia a necessidade da fase de consolidação das informações, que consiste em identificar tanto possíveis duplicidades como especificar os casos de testes mais adequados para as necessidades momentâneas do projeto. Por outro lado, uma lista abrangente permite maior possibilidade de seleção de trechos específicos do processo para teste das funcionalidades de software presentes. A escolha de realizar um caso de teste mais específico ou mais abrangente fica a cargo do engenheiro de software.

6 Conclusões

O presente trabalho apresentou uma proposta para extração de casos de testes a partir de modelos de processos de negócio. O método faz uso de heurísticas definidas em [2], o que inclui padrões de *workflow* presentes em [5], além de heurísticas complementares, proposta neste trabalho. Algumas heurísticas são influenciadas por propostas que utilizam outros artefatos de requisitos como insumo para extração de casos de teste, tais como [9], [11], [14], [16], [25], [28].

Observamos que os modelos de processos de negócio oferecem grande potencial como insumo para geração dos casos de testes, principalmente porque possuem maior capacidade em representar detalhes do negócio. Estes detalhes são traduzidos em requisitos de sistema e devem ser incluídos nos casos de teste. Identificamos que quanto maior a abstração do modelo de processo, maior é o impacto nos casos de teste. É importante que o processo seja modelado de forma detalhada, principalmente em relação aos fluxos que podem ser considerados como

"alternativos". Por exemplo, os fluxos caracterizados por desvios em regras de negócio ou situações que demandam decisões que levem a um fluxo executado por diferentes atividades.

Apesar destes cuidados beneficiarem o método de identificação de requisitos e de extração de casos de teste, os modelos de processos não abordam todos os requisitos de um sistema, mas expressam os de natureza operacional, assim como são os elementos que o compõem. Requisitos não funcionais são ainda mais difíceis de serem representados em modelos e identificados por heurísticas. Isso demonstra que a aplicação dessas propostas não deve ser considerada como único recurso para elicitação de requisitos e construção de casos de teste.

Entre os benefícios do método proposto, temos a aplicação direta nos modelos, sem a necessidade de transformações ou construção de modelos intermediários que auxiliem no processo de definição de casos de teste, como ocorrem em algumas propostas [9], [28], [32]. O método proposto é baseado em heurísticas que analisam diretamente os modelos e extraem as informações a partir dos objetos e fluxos presentes. A identificação dos requisitos [2] auxilia a construção de casos de testes, mas ainda demanda a delimitação dos cenários e tratamento das informações coletadas. Outro fator importante é a possibilidade de alinhar a estratégia de uso dos modelos de processos de negócio na engenharia de requisitos, com a geração de casos de teste, ampliando o potencial dos processos de negócio na construção de software. Além disso, o método de identificação de requisitos utilizado já se encontra automatizado para a ferramenta ARIS [4] e BPMN [30]. Desta forma, a automatização da geração de casos de teste facilitaria o acesso aos artefatos com significativa redução de custos. Em organizações que utilizam processos de negócio como auxílio na gestão, as alterações nos modelos poderiam ser replicadas nos artefatos de software de forma imediata, reduzindo o seu custo de manutenção.

Em trabalhos futuros, além da automação da proposta, devemos aplicar o método em um exemplo real para identificar com mais precisão o impacto do uso dos casos de testes candidatos. Também devemos investigar como classificar e consolidar a lista de casos de testes, gerando novas informações que possam auxiliar engenheiro de software durante a aplicação da suíte de testes. Outra questão é qual o impacto dos casos de teste no alinhamento do software com os objetivos organizacionais. Ou seja, deseja-se saber se os testes projetados a partir dos modelos de processo de negócio podem contribuir positivamente ao alinhamento entre a TI e o negócio. A relação com TMM (*Testing Maturity Model*) e TMMI (*Testing Maturity Model Integration*) também é desejada, com o objetivo de integrar as boas práticas registradas nestes modelos.

Referências bibliográficas

1. Amyot, D., Mussbacher, G., "*URN: Towards a New Standard for the Visual Description of Requirements*". In Proceedings of SAM'2002. pp.21-37, (2002)
2. Azevedo, L., Baião, F., Santoro, F., Souza, J., Revoredo, K., Pereira, V., Herlain, I., "Identificação de serviços a partir da modelagem de processos de negócio", Simpósio Brasileiro de Sistemas de Informação (SBSI'09), Brasília, (2009)
3. Azevedo, L., Santoro, F., Baião, F., Souza, J., Revoredo, K., Pereira, V., Herlain, I., "*A Method for Service Identification from Business Process Models in a SOA Approach*", 10th International Workshop on Business Process Modeling, Development, and Support (BPMDS), Enterprise, Business-Process, and Information Systems Modelling. V. 29, Amsterdam, (2009)
4. Azevedo, L., Sousa, H. P., Souza, J. F., Santoro, F. Baião, F., Identificação automática de serviços candidatos a partir de modelos de processos de negócio. Conferência IADIS Ibero Americana WWW/INTERNET 2009 (CIAWI'09), Outubro, 21-23, Madrid, Espanha, (2009)
5. van der Aalst, W., Ter Hofstede, A., Kiepuzewski, B., Barros, A., "*Workflow patterns*", In Distributed and Parallel Databases 14(1), p.5–51, (2003)
6. Westland, J. C., "*The cost of errors in software development: evidence from industry*", Journal of Systems and Software", Volume 62, Issue 1, Pages 1-9, ISSN 0164-1212, (2002)
7. Sousa, H. P., "Integrando modelagem intencional à modelagem de processos", dissertação de

mestrado, PUC-Rio, Rio de Janeiro, (2012)

8. OMG, "*Business Process Model and Notation*", http://www.bpmn.org
9. Swain, S.K., Mohapatra, D.P., Mall, R., "*Test Case Generation Based on Use case and Sequence Diagram*", International Journal of Software Engineering & Applications (IJSEA), Vol.3, (2010)
10. Weiss, M., Amyot, D., "*Business process Modeling with URN*", International Journal of E-Business Research, Vol. 1, No. 3, (2005)
11. Fraikin F., Leonhardt, T., "*SeDiTeC – Testing Based on Sequence Diagrams*", Automated Software Engineering, 2002. 17th IEEE International Conference on, ISSN: 1938-4300, 23-27, (2002)
12. Gargantini, A.; Heitmeyer, C., "*Using model checking to generate tests from requirements specifications*", In: ESEC/FSE- 7, ACM Press, (1999)
13. Hartman, A., Katara, M., Olvovsky, S., "*Choosing a test modeling language: a survey*". Proceedings of the 2nd international Haifa verification conference on Hardware and software, verification and testing, Haifa, Israel, (2006)
14. Heumann, J., "*Generating Test Cases From Use Cases*", The Rational Edge: e-zine for the rational community, (2001)
15. Kaner, C., Bach, J., Pettichord, B., "*Lessons Learned in Software Testing*", Wiley (2001).
16. Wang, Y., Zheng, M., "*Test Case Generation from UML Models*", Midwest Instruction and Computing Symposium 2012, (2012)
17. Sikandar-gani, S. B., "*User Requirement Notation (URN)*", Graduate Student, Department of Electrical and Computer Engineering, Mississippi State University, MS, USA, 2003.
18. Mogyorodi, G., "*Requirements-based testing: an overview*", Process engineering for systems, software and people, 2003.
19. Mingsong. C., Xiaokang. Q., Xuandong, L., "*Automatic test case generation for UML activity diagrams*", In: AST '06, ACM Press, 2006.
20. Myers, G. J.; Sandler, C.; Badgett, T., "*The Art of Software Testing*", 240 pages, Wiley; 3 edition, ISBN-10: 1118031962, ISBN-13: 978-1118031964, (2011)
21. Neto, A.C., Subramanyan, R. Vieira, M. Travassos, G.H., "*A Survey on Model-based Testing Approaches: A Systematic Review*", WEASELTech'07, Atlanta Georgia, USA, (2007)
22. Sousa, H. P., Azevedo, L.G., Santoro, F.M., "Identificação Automática de Serviços em Uma Abordagem SOA". Relatórios Técnicos do Dep. de Informática Aplicada da UNIRIO, n 0002, (2011)
23. van der Aalst, W., ter Hofstede, A, Kiepuzewski, B., Barros, A., "*Advanced workflow patterns*", 7th International Conference on Cooperative Information Systems, LNCS 1901, pp. 18–29, (2000)
24. Pourshahid, A., Amyot, D., Peyton, L., Ghanavati, S., Chen, P.; Weiss, M.; Forster, A. J., "*Business process management with the user requirements notation*", International MCETECH Conference on e-Technologies, pp.3-15, (2009)
25. Rosenberg, D., "*Use Case Thread Generation for Acceptance Testing From Theory to Practice*", ICONIX, http://iconixsw.com/
26. Russell, N., Ter Hofstede. A., Edmond, D., van der Aalst, W., "*Workflow Data Patterns*", QUT Technical report, FIT-TR-2004-01, Queensland University of Technology, Brisbane, (2004)
27. Srivastava, P.R., "*Combining UML Interaction Diagrams and State-Charts for Testing of Object Oriented Software Systems*", Journal of Information and Computing Science, Vol. 4, No. 1, pp. 056-064, ISSN 1746-7659, England, UK, (2009)
28. Swain, R.K., Behera, P.K., Mohapatra, D.P., "*Minimal Test Case Generation for Object-Oriented Software with State Charts*", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.4, (2012)
29. Scheer, A.G., "Methods ARIS 7.0", (2005)
30. Brandão, B. C. P.; J. C., Silva, "Identificação automática de serviços a partir de processos de negócio em BPMN", TCC, Departamento de Informática, UNIRIO, (2013)
31. R.J.A. Buhr, R.S. Casselman, "*Use Case Maps for Object-Oriented Systems*", Prentice Hall, (1996)
32. D. Buchs, L. Lucio, A. Chen, "*Model checking techniques for test generation from business process models*", Reliable Software Technologie-Ada-Europe, LNCS, Vol 5570, pp59-75, Springer, (2009)