PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 22/11

Contexto de Transparência Aplicada a Modelos de Negócio e a Sistema de *Software*

André Luiz de Castro Leal
Henrique Prado Sousa

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900
RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 22/11 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena Dezembro 2011

Contexto de Transparência Aplicada a Modelos de Negócio e a Sistema de Software

André Luiz de Castro Leal
aleal@inf.puc-rio.br

Henrique Prado Sousa
hsousa@inf.puc-rio.br

Abstract: The present work makes an analysis of transparency in the context of software and business processes, estabilishing a relationship among the transparency operations applied in the Lattesscholar software and make an evaluation of paradigm aspect in transparency of business process model.

Keywords: Transparency, software transparency, aspect paradigm, business process transparency.

Resumo: O presente trabalho faz uma analise de transparência no contexto de software e processos de negócio, estabelece uma relação de operações de transparência aplicada ao software Latesscholar e faz uma avaliação do paradigma de aspectos em transparência de modelos de processos de negócio.

Keywords: Transparência, transparência de software, paradigma de aspectos transparência de processos de negócio.

Responsável por publicações:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3114-1516 Fax: +55 21 3114-1530
E-mail: bib-di@inf.puc-rio.br

Sumário

1. INTRODUÇÃO 1

2. OPERACIONALIZAÇÃO DA TRANSPARÊNCIA 2

3. TRANSPARÊNCIA NOS PROCESSOS ORGANIZACIONAIS 4

3.1. TRANSPARÊNCIA APLICADA AOS PROCESSOS ORGANIZACIONAIS 4
3.2. MODELO DE PROCESSO DE NEGÓCIO SEM INSERÇÃO DE ELEMENTOS DE TRANSPARÊNCIA 6
3.3. MODELO DE PROCESSO DE NEGÓCIO COM INSERÇÃO DE ELEMENTOS DE TRANSPARÊNCIA 7
3.4. MODULARIZAÇÃO DA TRANSPARÊNCIA NO MODELO: IMPLEMENTAÇÃO SOB A ÓTICA DE ASPECTOS 9

4. CONTEXTO DE TRANSPARÊNCIA EM *SOFTWARES* 10

4.1. OPERACIONALIZAÇÃO DE *SOFTWARE* COM VIÉS DE CÓDIGO FONTE 11

5. CONSIDERAÇÕES FINAIS 15

REFERÊNCIAS 16

Lista de Figuras

Figura 1: Diagrama de relacionamento de transparência com outros requisitos não funcionais **4**

Figura 2: Modelo de processo de negócio de solicitação de financiamento **6**

Figura 3: Modelo de processo de negócio com inserção de elementos de transparência. **8**

Figura 4: Modelo de processo de negócio com transparência evidenciando as características transversais. **9**

Lista de Tabelas

Tabela 1: Detalhamento dos atributos de Graus de Transparência. **2**

Tabela 2: Detalhamento do atributo Validade. **14**

1. Introdução

O conceito de transparência perpassa por características de diafaneidade. Diáfano é aquilo que sendo compacto, dá passagem a luz, ou seja, é translúcido [Ferreira, 1986]. Essa característica é trazida para um contexto da informação, onde cidadãos críticos almejam por maiores esclarecimentos de determinados fatos ou processos. Segundo Fung [2007], a transparência poderá auxiliar decisões se prover às pessoas informações pertinentes, melhorando suas escolhas sem grandes custos.

Para [HOLZNER, 2006], transparência é a crescente demanda no contexto de mudanças globais devido a necessidade de se criar confiança através da vasta distancia cultural e geográfica. No contexto empresarial, as transações comerciais necessitam validar informação sobre mercados, seus riscos e oportunidades, neste caso conceituado por Fung [2007] como transparência dirigida. Na política há a necessidade de se validar informação sobre intenções e estratégias entre países, partidos e interesses da sociedade.

Atualmente é perceptível o aumento por ações de transparência através de políticas do governo, além disso, transparência vem sendo impulsionada pela transformação e poder dos computadores e internet, que pode ajudar a criar uma nova geração mais eficiente de políticas de transparência colaborativas [FUNG, 2007]. Nesse sentido, Fung [2007] cita que a eficácia da transparência vai depender em grande parte de dois fatores: 1. Políticas de transparência centradas no usuário, em suas necessidades e interesses. 2. Políticas eficazes de transparência que sejam sustentáveis para que sejam efetivas, ou seja, que ganhem em uso, precisão, escopo e perdurem no tempo.

Há de se verificar portanto, que a transparência passa a ter um papel importante no contexto da sociedade, tornando-se uma frente inexplorada para sua característização e aplicação. Essa frente pode ser direcionada aos processos organizacionais e a *softwares* utilizados de forma geral. Nas organizações, é algo que pode permitir melhorar a visão sobre os processos e as suas informações ao dar oportunidade de conhecimento sobre a mesma, reduzir a possibilidade de omissão entre os dados dos processos, possibilitar o controle sobre os produtos e serviços prestados, facilitar a investigação, e aumentar a confiança entre as organizações e a sociedade [CAPPELLI e LEITE, 2009]. E aplicada a *softwares*, transparência pode permitir informações completas, objetivas, confiáveis, de qualidade, melhorar o acesso à informação, auxiliar na compreensão da informação e permitir ainda que canais de comunicação estejam abertos para acesso livre às informações.

O presente trabalho encontra-se organizado da seguinte forma: a presente seção apresenta a motivação deste trabalho; a seção 2 detalha aspectos sobre a aplicação da transparência e demonstra a relação entre os diferentes conceitos que devem ser operacionalizados para aplicá-la a um dado domínio; a seção 3 aborda sobre transparência nos processo de negócio, onde apresenta modelos desenvolvidos sem a preocupação de ser transparente, os mesmos modelos alterados para contemplar maior nível de transparência permitindo compará-los e, por último, os modelos sob a ótica de aspectos; na seção 4 são apresentadas as operacionalizações de transparência sob o viés de código fonte; e por fim, na seção 5 são discutidas as considerações finais.

2. Operacionalização da transparência

A transparência, por ser um requisito não funcional, torna-se uma característica não mensurável, podendo somente ser definida como "suficiente" pelo usuário final, baseado em seu ponto de vista, ou seja, um produto pode ser considerado transparente na opinião de um indivíduo enquanto de outro não.

Portanto, cada esforço realizado em um dado produto com o objetivo de torná-lo transparente, apenas pode ser considerado como uma atividade que auxiliará no incremento do nível de transparência, tornando-o mais transparente, uma vez que não é possível definir se um produto é de fato totalmente transparente devido às suas características não funcionais.

Neste contexto, [CAPPELLI, 2008] define um conjunto de características que se fazem necessárias na aplicação da transparência, conceituadas pela autora como Graus de Transparência, tais como:

– Acessibilidade: A transparência é realizada através da capacidade de acesso. Esta capacidade é identificada através da aferição de práticas que implementam características de portabilidade, operabilidade, disponibilidade, divulgação e desempenho;

– Usabilidade: A transparência é realizada através das facilidades de uso. Esta capacidade é identificada através da aferição de práticas que implementam características de uniformidade, intuitividade, simplicidade, amigabilidade e compreensibilidade;

– Informativo: A transparência é realizada através da qualidade da informação. Esta capacidade é identificada através da aferição de práticas que implementam características de clareza, acurácia, completeza, corretude, consistência e integridade;

– Entendimento: A transparência é realizada através do entendimento. Esta capacidade é identificada através da aferição de práticas que implementam características de composição, concisão, divisibilidade, dependência, adaptabilidade e extensibilidade;

- Auditabilidade: A transparência é realizada através da auditabilidade. Esta capacidade é identificada através da aferição de práticas que implementam características de explicação, rastreabilidade, verificabilidade, validade e controlabilidade [CAPPELLI, 2009].

Cada Grau de Transparência possui um conjunto de atributos que são necessários para que aquele grau tenha um bom nível de transparência. A **Tabela 1** apresenta a relação discriminada desses atributos.

Tabela 1: Detalhamento dos atributos de Graus de Transparência. Fonte: [CAPPELLI, 2009]

Graus de Transparência	Atributos	Descrição do Atributo
Acessibilidade	Portabilidade	Capacidade de ser usado em diferentes ambientes.
Acessibilidade	Disponibilidade	Capacidade de ser utilizado no momento em que se fizer necessário.
Acessibilidade	Divulgação	Capacidade de ser apresentado.
Usabilidade	Uniformidade	Capacidade de manter uma única forma.
Usabilidade	Simplicidade	Capacidade de não apresentar dificuldades ou

		obstáculos.
Usabilidade	Operabilidade	Capacidade de estar operacional.
Usabilidade	Intuitividade	Capacidade de ser utilizado sem aprendizado prévio.
Usabilidade	Desempenho	Capacidade de operar adequadamente.
Usabilidade	Adaptabilidade	Capacidade de mudar de acordo com as circunstâncias e necessidades.
Usabilidade	Amigabilidade	Capacidade de utilização sem esforço
Informativo	Clareza	Capacidade de nitidez e compreensão.
Informativo	Completeza	Capacidade de não faltar nada do que pode ou deve ter.
Informativo	Corretude	Capacidade de ser isento de erros.
Informativo	Atualidade	Capacidade de estar no estado atual.
Informativo	Comparabilidade	Capacidade de ser comparado.
Informativo	Consistência	Capacidade de resultado aproximado de várias medições de um mesmo item.
Informativo	Integridade	Capacidade de correto e imparcial.
Informativo	Acurácia	Capacidade de execução isenta de erros sistemáticos.
Entendimento	Concisão	Capacidade de ser resumido.
Entendimento	Compositividade	Capacidade de construir ou formar a partir de diferentes pares.
Entendimento	Divisibilidade	Capacidade de ser particionado.
Entendimento	Detalhamento	Capacidade de descrever em minúcias.
Entendimento	Dependência	Capacidade de identificar a relação entre as partes de um todo.
Auditabilidade	Validável	Capacidade de ser testado por experimento ou observação para identificar se o que está sendo feito é correto.
Auditabilidade	Controlabilidade	Capacidade de domínio.
Auditabilidade	Verificabilidade	Capacidade de identificar se o que está sendo feito é o que deve ser feito.
Auditabilidade	Rastreabilidade	Capacidade de seguir o desenvolvimento de um processo ou a construção de uma informação, suas mudanças e justificativas.
Auditabilidade	Explicável	Capacidade de informar a razão de algo.

Algumas dessas características possuem a capacidade de interação com outras, sendo capaz de influenciar de forma positiva ou negativa. [CAPPELLI e LEITE, 2009] apresentam um grafo (**Figura 5**) desenvolvido a partir do NFR (*Non-Funcional Requirements*) Framework, proposto por [Chung, 2000]. Este framework define uma forma sistemática para decompor requisitos não funcionais (características de qualidade), priorizar, operacionalizar e tratar interdependências entre elas, independentemente de quais sejam.

Nesta estrutura, também são representados os tipos de contribuição entre os elementos se influenciam. Estes tipos de contribuição podem ser de "BREAK", "HURT", "UNKNOWN", "HELP" e "MAKE". Cada um destes tipos representa respectivamente:

a) BREAK - Provê contribuição negativa suficiente para que a característica superior não seja atendida;

b) HURT - Provê contribuição negativa parcial para não atendimento da característica superior;

c) UNKNOWN - Provê contribuição porém não se sabe se negativa ou positiva;

d) HELP - Provê contribuição positiva parcial para atendimento da característica superior;

e) MAKE - Provê contribuição positiva suficiente para que a característica 26 superior seja atendida;

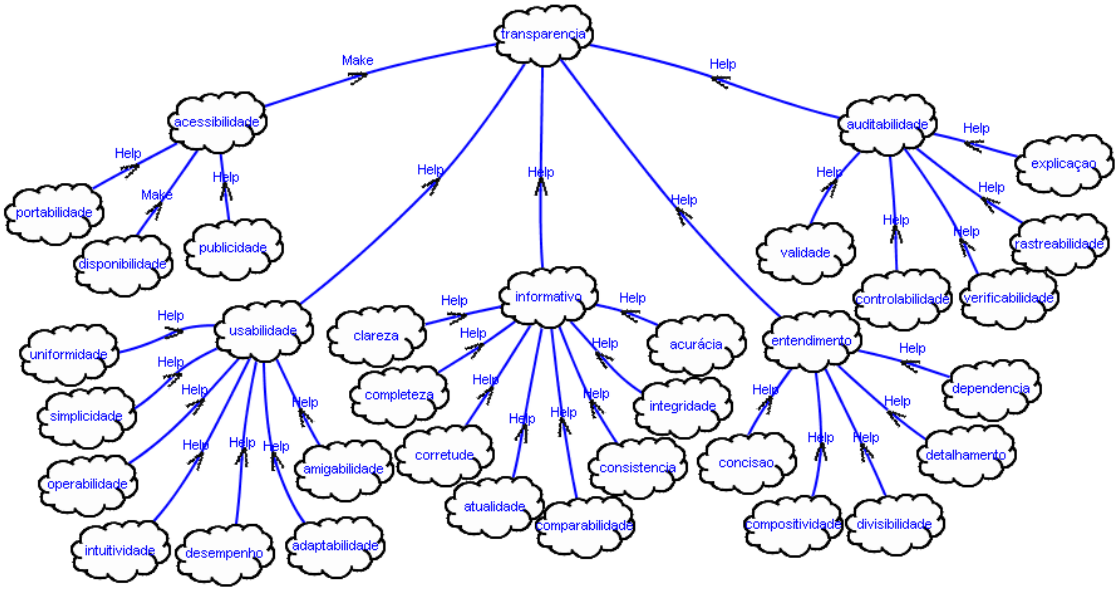

Figura 5: Diagrama de relacionamento de transparência com outros requisitos não funcionais
Fonte: [CAPPELLI e LEITE, 2009]

3. Transparência nos Processos Organizacionais

3.1. Transparência Aplicada aos Processos Organizacionais

Organizações modernas buscam gerir seus negócios de forma eficiente amparadas em processos organizados, otimizados e controlados. Nesse sentido as organizações possuem processos primários e secundários [CRUZ, 2000, 2003].

Processos primários são todos aqueles que estão diretamente ligados à fabricação do produto que a organização tem por objetivo disponibilizar para seus clientes [CAPPELLI e LEITE, 2009]. Processos secundários, também chamados de processos de suporte, são todos que, como o próprio nome diz, suportam os processos primários e os próprios secundários, dando-lhes apoio para que possam existir [CAPPELLI e LEITE, 2009].

Para Carvalho [1989], os processos devem ser classificados segundo características de atividades que executam. Carvalho [1989] classifica as atividades como de tratamento da informação e substantivas. Nas atividades de tratamento de informação há consumo e/ou geração de informação, já as substantivas tem como objetivo principal a geração de produtos físicos e que, apesar de gerarem informação de controle, são executadas com fim produtivo [CAPPELLI e LEITE, 2009].

Apesar da existência de diversos tipos de classificação para processos organizacionais, todos podem ser representados através de modelos de processo, construídos através de linguagens de modelagem que contêm elementos como atores, atividades, fluxos de informação, regras de negócio, eventos, documentos, entre outros.

Diante disso, há dois conceitos importantes que devem ser esclarecidos: *Business Process Management* (BPMa) - Gerenciamento de Processos de Negócio e *Business Process Modeling* (BPMo) - Modelagem de Processos de Negócio. Ambos os conceitos nos submetem à sigla BPM, por isso a necessidade de conceituação [SOUSA, 2010].

Bloomergschmelzer [2006] conceitua BPMa em referência aos meios tecnológicos para a empresa adquirir a visibilidade e o controle sobre os processos de vida longa, processos de múltiplos passos que abrangem um intervalo grande de sistemas e pessoas em uma ou mais organizações [SOUSA, 2010]. Enquanto BPMo é conceituado pelo autor como o conjunto de práticas ou tarefas que as empresas podem executar para descrever visualmente todos os aspectos de um processo de negócio, incluindo o seu curso, controle e pontos de decisão, gatilhos e condições para execução das atividades, o contexto em que uma atividade executa e os recursos associados [SOUSA, 2010].

No contexto desse trabalho, iremos utilizar a sigla BPM para representar os conceitos abordados para BPMo. BPM pode ser vista como auxilio para a compreensão do negócio, uma vez que seu mapeamento deverá ser bem detalhado a fim de representar fielmente todo o fluxo de trabalho realizado que permite atingir os objetivos do negócio [SOUSA, 2010]. Portanto, BPM permite à organização obter conhecimento sobre si e sobre a forma como seus produtos e serviços são gerados. Além disso, permite a explicitação da tecnologia e das aplicações que apoiam a execução dos processos e dos dados que trafegam entre estes sistemas, dando visibilidade sobre as aplicações e sobre os controles exercidos por estes [CAPPELLI e LEITE, 2009].

Apesar dos modelos representarem os processos de negócio, estes nem sempre contemplam níveis de transparência suficientes para seus usuários. É importante que algumas características sejam adicionadas ao modelo do processo para que o objetivo de transparência seja alcançado. Transparência, por se tratar de um requisito de qualidade, geralmente pode estar entrelaçada ou espalhada nas diversas atividades do modelo, podendo ser a transparência dita como transversal. Cappelli [2010] sugere que esta característica de espalhamento e entrelaçamento de um conceito em um determinado domínio foi estruturada na Engenharia de *Software* (ES) pelo paradigma de aspectos [Kiczales et al., 1997]. Portanto, incorporar características de transparência ao modelo de processo pode estar diretamente relacionada a aspectos.

No contexto de inserção de características de transparência no modelo do processo é importante esclarecer que as inserções devem vir com o objetivo de tornar clara a parte tácita, ou seja, as atividades conhecidas e não representadas do processo. É importante também apontar que a correta inclusão da transparência preserva a eficácia do processo e não necessariamente o altera, mas pode melhorá-lo proporcionando pequenas alterações sob o viés da eficiência.

3.2. Modelo de Processo de Negócio sem Inserção de Elementos de Transparência

Um exemplo de modelo de processo de negócio sem a aplicação de elementos de transparência pode ser visto conforme apresentado na **Figura 2**. Os modelos apresentados neste trabalho foram desenvolvidos na ferramenta *Cross-Oryx*[1], em BPM 1.1. O *Cross-Oryx* é uma extensão da ferramenta *Oryx*[2] que possui elementos extras para a representação de objetos transversais ao modelo. A ferramenta *ORYX*[2], por sua vez, é um *framewok Open Source* para modelagem gráfica de processos que não necessita de instalação do *software*, pois é baseada em tecnologia Web.

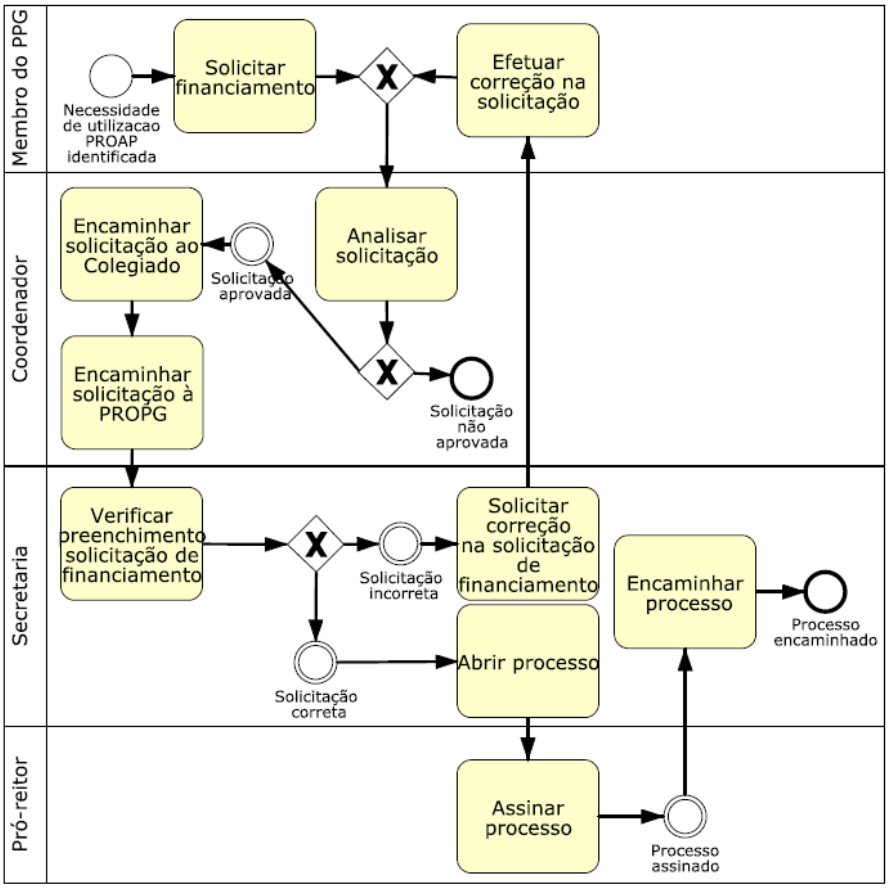

Figura 6: Modelo de processo de negócio de solicitação de financiamento.
Fonte: Elaborado pelos autores.

No modelo, estão representadas as atividades necessárias para a aplicação do financiamento do Programa de Apoio à Pós-graduação (PROAP-CAPES) em uma instituição federal de ensino superior no Brasil. Tal financiamento é utilizado pelo Programa de Pós- graduação (PPG) para qualquer necessidade do PPG, seja em aquisição de materiais, construção de benfeitorias, atualização de infra-estrutura e também de bolsas de mestrado e doutorado por exemplo.

[1] http://www.ppgsc.ufrn.br/~analuisa/crossoryxeditor .

[2] http://bpt.hpi.uni-potsdam.de/Oryx/LearnAboutOryx e http://oryx-editor.org/.

Normalmente, os modelos elaborados a partir de BPM levam em consideração as atividades mais evidentes no processo de negócio. Assim, para os envolvidos no processo, a leitura do modelo pode ser suficiente, embora muitos elementos possam não ser representados podendo gerar algum tipo de controvérsia. Por exemplo, no modelo apresentado não está claro se é necessário o registro de alguma informação na atividade *Solicitação de Financiamento*, nem quais informações devem ser registradas. Esse fato está correlacionado a conhecimento tácito que está sob o domínio dos envolvidos nas regras de negócio do modelo representado. Outras pessoas podem ter dúvidas na leitura e interpretação das atividades relacionadas.

Outro fluxo que pode gerar má interpretação é visto entre as atividades da *Secretaria* da Pró-reitoria de Pós-graduação (PROGP), quando esta solicita a correção na solicitação do financiamento. Entre essa atividade e a atividade *Efetuar Correção na Solicitação* existem elementos que não estão claros, como por exemplo: o que deve ser registrado entre uma atividade e outra, se a solicitação de correção é simplesmente um aviso de solicitação, ou requer, novas informações e se existe um porquê dessa solicitação de correção.

Diante dessas evidências, é possível apresentar um novo modelo do processo de negócio com a inserção de elementos que irão auxiliar na transparência e no entendimento do processo em questão.

3.3. Modelo de Processo de Negócio com Inserção de Elementos de Transparência

Para se obter a transparência, os elementos que podem ser inseridos no novo modelo são desde novas atividades, anotações, artefatos ou documentos, até mesmo novos fluxos de atividades. Dessa forma, há na nova representação um maior detalhamento do processo real representado no modelo.

A **Figura 3** apresenta o modelo de negócio de solicitação de financiamento com a inserção de elementos com o objetivo de tornar o processo mais transparente, ou seja, evidenciar elementos intrínsecos do conhecimento tácito que não foram representados no modelo original. O novo modelo foi construído levando-se em consideração a inserção de elementos que satisfizessem os atributos de Rastreabilidade, Detalhamento e Verificabilidade. O modelo também foi desenvolvido na ferramenta Cross-Oryx, utilizando a versão BPM 1.1.

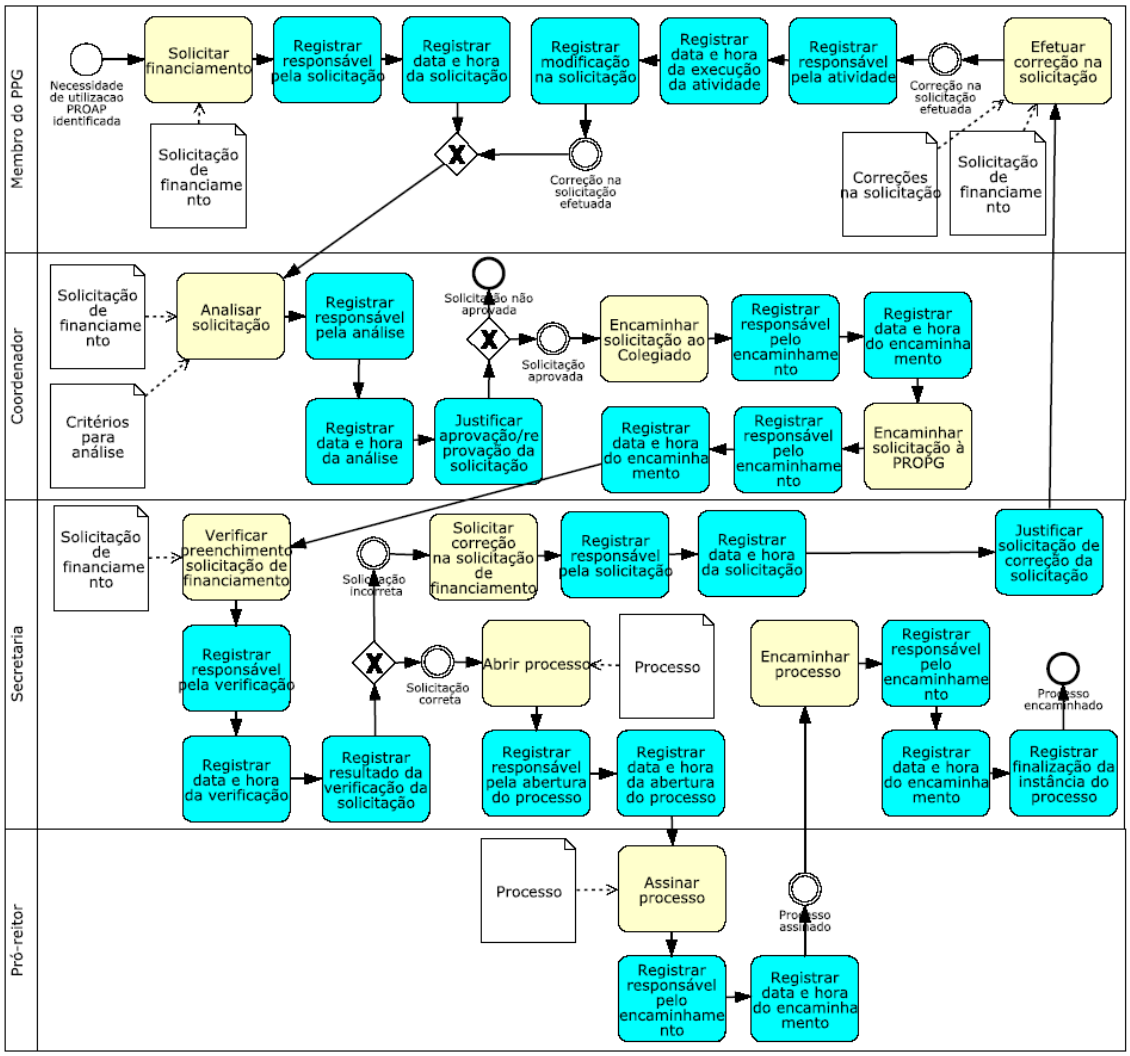

Figura 7: Modelo de processo de negócio com inserção de elementos de transparência. Fonte: Elaborado pelos autores.

Com a inserção de novos elementos no modelo, há um maior detalhamento e conseqüentemente auxilia a sua transparência. Esse detalhamento nem sempre irá afetar o processo original, uma vez que o que está sendo feito é tornar clara as atividades intrínsecas no processo e que não foram representadas no modelo. O detalhamento, portanto, não irá afetar a eficácia do processo, pois deverá garantir o seu funcionamento conforme a realidade, mas por outro lado a eficiência do processo será beneficiada com a inserção dos elementos de transparência.

Apesar de tornarmos o processo mais transparente, o detalhamento do modelo poderá gerar uma representação menos transparente, ou seja, a representação poderá estar tão carregada que sua leitura se torna complexa e pouco clara, com maior número de níveis e maior passagem de bastão entre os papeis envolvidos no processo. No entanto, dizer que a representação de um processo não está transparente, não significa que o processo perdeu sua transparência, isso porque o processo pode continuar fácil de ser executado, no entanto, sua representação carregada de tarefas e eventos pode passar uma imagem de um processo complicado e confuso. O inverso também é verdadeiro, quando há um processo complexo, ou seja, o processo é extremamente difícil de ser implementado e executado, mas por outro lado, sua representação depois de aplicada transparência passe uma impressão de simplicidade.

Portanto, ao se aplicar transparência, um requisito de qualidade, em processos percebe-se a perda de simplicidade e clareza na sua representação, mas não no processo. Uma solução para resolver a legibilidade do modelo com elementos de transparência é o uso da notação de aspectos na representação gráfica.

A próxima seção apresenta o modelo com a implementação de características de aspectos.

3.4. Modularização da Transparência no Modelo: implementação sob a ótica de aspectos

A implementação de aspectos no modelo com elementos de transparência deixa evidente que transparência tem características transversais, ou seja, uma característica de qualidade, que pode ter duplicidade de elementos que são comuns em várias atividades do modelo original. Para Cappelli [2009], esse conceito pode potencializar a reestruturação do modelo com elementos de transparência em um modelo com características de aspectos, onde os elementos são dispostos isoladamente do modelo original e os elementos de transparência ficam em raias verticais, sendo cada um deles ligados às atividades do processo original.

A **Figura 4** apresenta o processo em um modelo com características de aspectos, onde a legibilidade e clareza da representação aumenta a transparência do modelo. Portanto, há agora um detalhamento maior do processo, onde este se torna mais transparente e também a sua representação gráfica passa a ter maior simplicidade e conseqüentemente maior transparência. O modelo está desenvolvimento no editor *Cross-Oryx*[3] utilizando seus elementos de representação transversal ao modelo BPM.

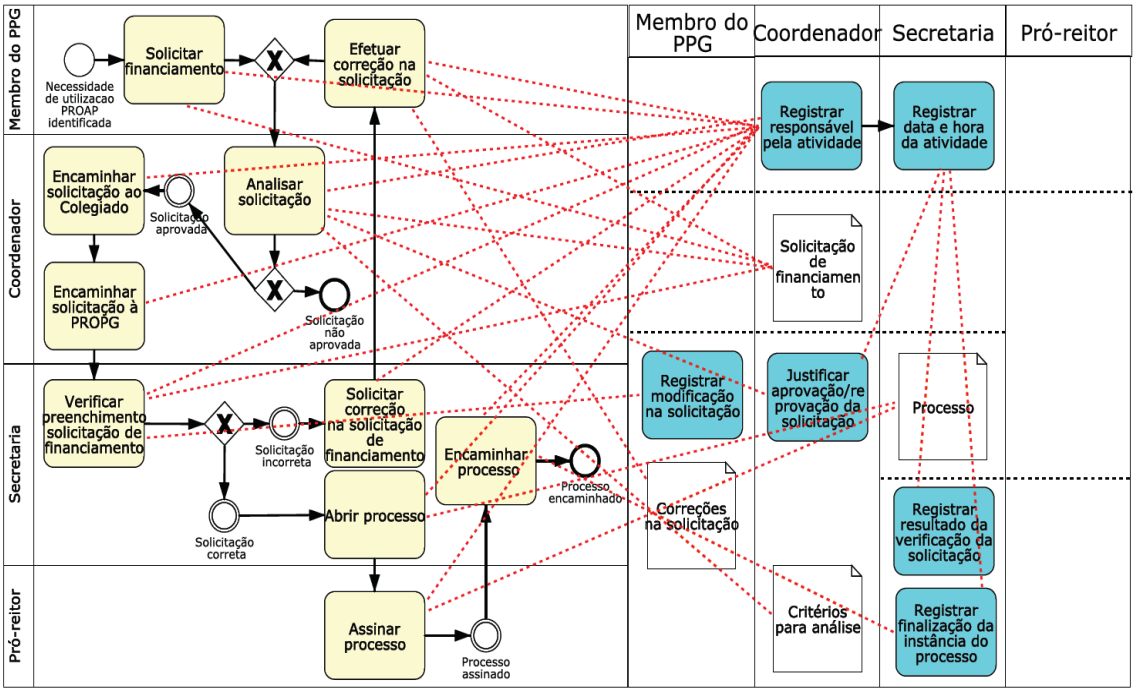

Figura 8: Modelo de processo de negócio com transparência evidenciando as características transversais.
Fonte: Elaborado pelos autores.

[3] http://www.ppgsc.ufrn.br/~analuisa/crossoryxeditor/

4. Contexto de Transparência em *Softwares*

Softwares têm cada vez mais se tornado parte do cotidiano das pessoas. É inevitável constatar que o crescimento do uso do software nas rotinas pessoais e profissionais tornaram-se mais freqüentes e em algumas situações, imperceptível, como no caso dos softwares embarcados[4].

Outro fator importante é o advento da internet e sua popularização, resultando em muitos usuários que não possuem o conhecimento de que, por trás de suas páginas pessoais, freqüentemente utilizadas, está implementada uma estrutura de software. E a internet passa a ter um papel fundamental na vida dessas pessoas, uma vez que permite o acesso aos principais assuntos de seu interesse, realiza pesquisas de notícias e principalmente, hoje em dia, o crescente uso por necessidades como cidadão, seja para o envio de informações de imposto de renda, a verificação de condições legais de seu veículo, condições como eleitor, como cliente de uma instituição financeira, como aluno de um centro de ensino ou pai verificando o andamento pedagógico de seu filho na escola.

Portanto, a internet passa a estar totalmente vinculada às principais necessidades como cidadão e é sabido que esse uso tende a aumentar. Dessa forma, entende-se que o contexto de transparência está também aderente a software. Como citado em parágrafos anteriores nesse texto, Fung (2007) acredita que a transparência vem sendo impulsionada pela transformação e poder dos computadores e internet, além de poder auxiliar na criação de uma nova geração mais eficiente de políticas de transparência colaborativas.

Diante disso, é importante que exista uma caracterização formal do que deva ser transparência em software para que empresas e órgãos públicos possam direcionar seus esforços de construção já alinhados com essa nova e crescente demanda, pois o cidadão passa a ter maior interesse de como os resultados de software ou mesmo sua execução são apresentados.

Um exemplo disso pode estar no acompanhamento da atuação de determinado governo estadual que divulga seus resultados na internet. Recentemente foi lançado pelo Contas Abertas o Projeto de Fomento à Transparência nas Contas Públicas Brasileiras, onde foi criado o Índice de Transparência[5]. Esse fato demonstra a proporção que está recebendo o assunto transparência e a importância da internet, ou seja, software na sua divulgação, uso e apresentação de resultados.

O entendimento do que é Transparência de Software é pouco trivial. A tarefa de mapear a ontologia de Transparência é trabalhosa uma vez que pode necessitar de questões sob a ótica de ontologias e reutilização de software e sob a ótica de operação envolve desde trabalhos de programação (operacionalização) a trabalhos que lidam

[4] Um sistema embarcado é um sistema microprocessado no qual o computador é completamente encapsulado ou dedicado ao dispositivo ou sistema que ele controla. Diferente de computadores de propósito geral, como o computador pessoal, um sistema embarcado realiza um conjunto de tarefas predefinidas, geralmente com requisitos específicos. Já que o sistema é dedicado a tarefas específicas, através de engenharia pode-se otimizar o projeto reduzindo tamanho, recursos computacionais e custo do produto.
Fonte: Wikipedia < http://pt.wikipedia.org/wiki/Software_embarcado>.

[5] Disponível em: http://contasabertas.uol.com.br/website/informativo/detalheinformativo.aspx?id=91 e http://www.indicedetransparencia.org.br/.

com o conceito de variabilidade de software, construção de arquiteturas e requisitos intencional entre outros.

Apesar do conceito de transparência em software parecer óbvio ele não é, pois requer decisões sobre o foco a ser abordado sobre o tema. É necessário abordar transparência sob diferentes aspectos e deixar claro onde está sendo aplicado o conceito. Por exemplo, ainda são pouco claras as questões de transparência da informação ou transparência do processo sob a ótica de quem a usa. Podemos falar de software transparente, quando estamos na verdade querendo saber se o serviço prestado pelo software é transparente (nesse caso queremos que as informações sobre o serviço sejam transparentes). Como é constatado há de se contextualizar o foco da aplicação da transparência para que não haja má interpretação do conceito a que se deseja.

Na seção subsequente é analisada a operacionalização da transparência em software, sob o viés de código fonte.

4.1. Operacionalização de *Software* com Viés de Código Fonte

Para o estudo de transparência de software sob o viés de código fonte foi analisado o software Lattesscholar[6], software que utiliza os serviços do Google, CNPQ Lattes[7] e Google Scholar[8] para contagem de citações e cálculo do H-index de pesquisadores.

Foram escolhidos de forma aleatória as seguintes atributos: Validade, Controlabilidade, Verificabilidade e Rastreabilidade. Todos esses itens fazem parte da categoria Auditabilidade.

As categorias foram extraídas do trabalho de Cappelli (2009) e foram também listadas as principais seções agrupadoras das operacionalizações. Entende-se por operacionalização, as atividades necessárias para atender ao atributo de transparência em questão.

CATEGORIA: AUDITABILIDADE

ATRIBUTO: VALIDADE

Definição: capacidade de ser testado por experimento ou observação para identificar se o que está sendo feito é correto. Capacidade de ser avaliado por experimento ou observação para identificar se está de acordo com as expectativas dos usuários Cappelli (2009).

validade: s. f. Qualidade de válido.

válido: adj. 1. Valioso. 2. Que tem saúde; são, vigoroso. 3. Legítimo, legal. 4. Certo, correto [Ferreira, 1986].

SEÇÃO AGRUPADORA

1 - Possuir mecanismos para validar o atendimento das condições iniciais e finais
2 - Definir mecanismos de validação
3 - Usar técnicas de simulação de uso do *software*

[6] http://www.er.les.inf.puc-rio.br/~wiki/index.php/Lattesscholar
[7] http://lattes.cnpq.br/
[8] http://scholar.google.com.br/

ATRIBUTO: CONTROLABILIDADE

Definição: Capacidade de ter domínio Cappelli (2009).

Controlado: adj. 1. Submetido a controle. 2. Que tem controle; comedido Ferreira, 1986].

SEÇÃO AGRUPADORA

1 - Definir os pontos de controle
2 - Definir políticas de acompanhamento
3 - Permitir execução com acompanhamento
4 - Indicar desvios (realizado X planejado)
5 - Manter o *software* associado com informações de versionamento/configuração

ATRIBUTO: VERIFICABILIDADE

Definição: Capacidade de identificar se o que está sendo feito é o que deve ser feito Cappelli (2009).

SEÇÃO AGRUPADORA

1 - Usar as normas de programação e documentação de código
2 - Anexar requisito ao código
3 - Usar ambiente que permita verificação automática
4 - Usar especificação sistemática

ATRIBUTO: RASTREABILIDADE

Definição: Capacidade de seguir o desenvolvimento de uma ação ou a construção de uma informação, suas mudanças e justificativas Cappelli (2009).

SEÇÃO AGRUPADORA

1 - Manter o *software* associado com informações de versionamento/configuração
2 - Identificar claramente o relacionamento entre as diferentes partes do *software*
3 - Utilizar especificações de pré e por condição
4 - Possibilitar a navegação entre documentos de origem e documentos resultantes
5 - Possibilitar a identificação de fontes de informação geradoras de documentos
6 - Possibilitar a navegação entre versões de um mesmo documento

Após a escolha dos atributos e a listagem das seções agrupadoras das operacionalizações, foram pesquisadas e identificadas os itens de operacionalização e listadas abaixo das suas respectivas seções. O resultado final das operacionalizações para os atributos ficou representado como segue:

CATEGORIA: AUDITABILIDADE

ATRIBUTO: VALIDADE

OPERACIONALIZAÇÕES

1 - Possuir mecanismos para validar o atendimento das condições iniciais e finais
   1.1 O código fonte está identado?
   1.2 O código fonte está sem erros de sintaxe ou gramaticais em comandos predefinidos na linguagem como por exemplo: if, while, case, for, funções da linguagem, etc?
   1.3 Os tipos de variáveis definidas estão criadas em conformidade com seu uso?
   1.4 A estrutura dos comandos e corpo do programa estão criados de acordo com a formalização da linguagem, por exemplo, com variáveis declaradas no início do programa, corpo do programa delineado de acordo com exigência da linguagem, seja por Begin/End, chaves {} ou outros?
   1.5 As estruturas condicionais permitem uma fácil análise do conjunto de condições, ou seja, existem poucos testes lógicos em um só comando condicional?
   1.6 As estruturas de repetição estão criadas, sempre que possível, com o mínimo de laços internos?
2 - Definir mecanismos de validação
   2.1 As variáveis estão criadas com nomes que possam identificar seu uso?

2.2 As funções, classes, métodos e outros estão criados com nomes que possam identificar seu uso?

3 - Usar técnicas de simulação de uso do *software*

3.1 O código fonte é compilado sem erros?

3.2 O código fonte é interpretado sem erros?

ATRIBUTO: CONTROLABILIDADE

OPERACIONALIZAÇÕES

1 - Definir os pontos de controle

1.1 Existe agenda definida para a inspeção de código?

2 - Definir políticas de acompanhamento

2.1 Existem processos definidos para inspeção de código?

2.2 Existem responsáveis com papéis bem definidos para as inspeções de código?

2.3 Existem política para uso de *software* de controle de versões, como por exemplo, registros de *branch, tags*, bloqueio de arquivo, checkout, checkin, commit, release, update, entre outros?

2.4 Existe política definida para desenvolvedores atualizarem seus *software* de controle de versão?

3 - Permitir execução com acompanhamento

Não abordado

4 - Indicar desvios (realizado X planejado)

4.1 Os problemas identificados nas inspeções são catalogados?

4.2 Existe uma planilha ou um método de comparação entre realizado e planejado?

4.3 A planilha é utilizada de forma eficaz?

5 - Manter o *software* associado com informações de versionamento/configuração

5.1 Existe *software* de controle de versão sendo utilizado para registro de evolução corretiva ou preventiva do *software*?

5.2 Os códigos novos são atualizados no *software* de controle de versionamento?

5.3 As atualizações dos arquivos de códigos fontes foram feitas no *software* de controle de versão de forma a permitir a visualização de seu versionamento, desde a origem até o documento atual?

ATRIBUTO: VERIFICABILIDADE

OPERACIONALIZAÇÕES

1 - Usar as normas de programação e documentação de código

1.1 Existe um documento de arquitetura para padronização de código?

1.2 O código desenvolvido esta condizente com o documento de arquitetura?

1.3 Existem regras de nomeação de classes, funções, procedimentos e outros?

2 - Anexar requisito ao código

2.1 O código contém comentários com as discriminações das regras dos requisitos dos usuários?

2.2 As regras de requisitos dos usuários condizem com os comentários no código?

3 - Usar ambiente que permita verificação automática

Não abordado

4 - Usar especificação sistemática

4.1 As especificações dos usuários estão discriminadas em casos de uso?

4.2 Existem planos de teste?

4.3 As classes a serem criadas estão evidenciadas?

ATRIBUTO: RASTREABILIDADE

OPERACIONALIZAÇÕES

1 - Manter o *software* associado com informações de versionamento/configuração

1.1 Os arquivos armazenados no *software* de controle de versão possuem tags de versão com comentários do porquê foram atualizados, inseridos ou excluídos?

1.2 Os arquivos armazenados no *software* de controle de versão indicam por branchs qual a versão está em produção?

2 - Identificar claramente o relacionamento entre as diferentes partes do *software*

2.1 Existe uma matriz de rastreabilidade entre as requisitos funcionais e as classes de objetos codificadas?

2.2 Existe uma matriz de rastreabilidade entre as requisitos funcionais e as classes de objetos codificadas?

2.3 Existe a indicação por comentários de em qual arquivo foi declarada uma função utilizada nas diversas partes do *software*?
2.4 Existe a indicação por comentários em que partes do sistema a função é utilizada?
2.5 Existe nos artefatos de requisitos a identificação dos diferentes clientes ou usuários que solicitaram mudanças nos requisitos?
2.6 Existe discriminado a relação entre os casos de testes e requisitos?

3 - Utilizar especificações de pré e por condição
3.1 Os artefatos de casos de uso contém seções de pré e pós-condições?

4 - Possibilitar a navegação entre documentos de origem e documentos resultantes
4.1 As atualizações dos artefatos do projeto do sistema foram feitas no *software* de controle de versão de forma a permitir a visualização de seu versionamento, desde o documento de origem (*baseline*) até o documento atual?

5 - Possibilitar a identificação de fontes de informação geradoras de documentos
5.1 Os artefatos possuem discriminação de documentos da origem da pesquisa que deram suporte a elicitação dos requisitos?

6 - Possibilitar a navegação entre versões de um mesmo documento
6.1 Os artefatos do projeto do sistema estão armazenados em *softwares* de controle de versão?
6.2 Estão atualizados de forma a se criar todas as versões dos artefatos e códigos fontes do sistema?

Cada item de operacionalização recebeu uma classificação como: Não Atende, Quase Não Atende, Não se Aplica, Atende Parcialmente que já estavam sendo sugeridos pelo Grupo de Engenharia de Requisitos da PUC-Rio[9] e mais um item foi sugerido, o Não Observado. Esse item foi inserido na lista, pois, algumas operacionalizações foram propostas para o contexto geral de software, mas como a análise do Lattesscholar foi feito apenas sob o viés de código fonte, algumas operacionalizações não puderam ser avaliadas. Além disso foram introduzidas: Pré-condições necessárias para a Operacionalização; Observações inerentes a avaliação do Latesscholar; e a Instância de Operacionalização, que reflete qual o nível ou técnica utilizada para a avaliação da operacionalização. A **Tabela 2** apresenta os resultados das avaliações do validade.

Tabela 2: Detalhamento do atributo Validade.
Fonte: elaborada pelos autores.

2.2. Auditabilidade - 2.2.1. Validade [Código Fonte]					
Nº	**RNF e Operacionalizações:**	**Pré-condição**	**Avaliação Lattesscholar**	**Observação**	**Instância da operacionalizaçã o**
1	Possuir mecanismos para validar o atendimento das condições iniciais e finais				
1.1	O código fonte está identado?	Código produzido.	**Atende**		Através de leitura.
1.2	O código fonte está sem erros de sintaxe ou gramaticais em comandos predefinidos na linguagem como por exemplo: if, while, case, for, funções da linguagem, etc?	Código produzido.	**Atende**	Variável conforme linguagem utilizada.	Através de leitura.
1.3	Os tipos de variáveis definidas estão criadas em conformidade com seu uso?	Código produzido.	**Atende**		Através de leitura.

[9] http://transparencia.les.inf.puc-rio.br/

1.4	A estrutura dos comandos e corpo do programa estão criados de acordo com a formalização da linguagem, por exemplo, com variáveis declaradas no início do programa, corpo do programa delineado de acordo com exigência da linguagem, seja por Begin/End, chaves {} ou outros?	Código produzido.	**Atende**	Variável conforme linguagem utilizada.	Através de leitura.
1.5	As estruturas condicionais permitem uma fácil análise do conjunto de condições, ou seja, existem poucos testes lógicos em um só comando condicional?	Código produzido.	**Atende**	Intenção de não permitir estruturas condicionais em demasia numa mesma linha de teste de condição.	Através de leitura.
1.6	As estruturas de repetição estão criadas, sempre que possível, com o mínimo de laços internos?		**Atende**		Através de leitura.
2	Definir mecanismos de validação				
2.1	As variáveis estão criadas com nomes que possam identificar seu uso?	Código produzido.	**Atende**		Através de leitura.
2.2	As funções, classes, métodos e outros estão criados com nomes que possam identificar seu uso?	Código produzido.	**Atende**		Através de leitura.
3	Usar técnicas de simulação de uso do software				
3.1	O código fonte é compilado sem erros?	Código finalizado.	**Não se aplica**		Através de observação da execução de compilação.
3.2	O código fonte é interpretado sem erros?	Código finalizado.	**Não testado**		Através de observação da interpretação pelo browser.

5. Considerações Finais

O presente trabalho traz os resultados de transparência aplicada a modelos de negócio e a softwares, onde foram aplicados elementos de transparência em um processo de negócio de solicitação de financiamento para aquisições em um departamento de pós-graduação de uma instituição federal de ensino e também a avaliação do software Lattesscholar sob o viés de código fonte.

Algumas dificuldades foram encontradas pelos autores no que diz respeito à modelagem com o Cross-Oryx e sua compatibilidade com as versões mais novas geradas pelo Oryx. Até a data desse documento, o Cross-Oryx não tinha compatibilidade com o BPM 2.0 gerado no Oryx. Dessa forma, todos os modelos de processos de negócio, que inicialmente foram construídos na versão BPM 2.0, tiveram que ser reconstruídos na versão BPM 1.1. Falta de compatibilidade do Cross-Oryx e pouca flexibilidade do Oryx na conversão de versões.

A versão utilizada do Cross-Oryx apresentou algumas falhas, o elemento Crosscutting, respectivo aos elementos transversais, não funcionou (apenas o elemento de conexão entre objetos transversais funcionou). O principal elemento que seria a raia na vertical simplesmente desenha o objeto de uma atividade comum no modelo.

Outros elementos de verticalização também desenham uma atividade no modelo, ao invés de desenhar o que é proposto. O elemento Element Affected desenha o círculo esperado no modelo, mas parece não criar a ligação correta com os outros objetos. Um pool é arrastado e o software cria uma atividade.

Dessa forma, o modelo apresentado com transparência e com aspectos atende as necessidades de uso de transparência e suas características de transversalidade, mas o projeto desenvolvido no Cross-Oryx não pode ser continuado até que nova versão do software seja disponibilizada.

Com relação a software, nem todos os resultados da pesquisa de transparência foram transcritos para esse documento, uma vez que alguns atributos avaliados não continham resultados de operacionalizações com foco em código fonte ou então não puderam ser observadas pelos autores, uma vez que esses não tiveram acesso ao projeto, mas somente ao fonte do Latesscholar.

Referências

[BLOOMBERGSCHMELZER, 2006] BLOOMBERGSCHMELZER, BLOOMBERG, J. , RONALD, S.. *Service Orient or Be Doomed!: How Service Orientation Will Change Your Business*. Hoboken; New Jersey: John Wiley & Sons, 2006, 249 p. Bibliografia: ISBN-10, 0-471-76858-8 / ISBN-13, 978-0471-76858-6.

[CAPPELLI, 2008] CAPPELLI, C., LEITE, J. C. S. P. *Transparência de Processos Organizacionais*. Universidade Federal Fluminense, LATEC. II Simpósio Internacional de Transparência os Negócios. 2008.

[CAPPELLI e LEITE, 2009] CAPPELLI, C.; LEITE, J. C. S. do P.. *Uma Abordagem para Transparência em Processos Organizacionais Utilizando Aspectos*. Rio de Janeiro, 2009. 328 p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

[CARVALHO, 1989] CARVALHO L. C. S.; FREIRE, A. C.. *Restrições e Desdobramentos na Implantação de uma Metodologia Estruturada de Análise de Sistemas: Um Estudo de Caso*. In: XIII Encontro Nacional da ANPAD, 1989, São Paulo. Anais do XIII Encontro Nacional da ANPAD. São Paulo: ANPAD, 1989. v. 2. p. 693-696.

[CHUNG, 2000] CHUNG, L.; NIXON, B.; YU, E.; MYLOPOULOS, J. *Non-Functional Requirements in Software Engineering* – Kluwer Academic Publishers – Massachusetts, USA, 2000.

[CRUZ, 2000] CRUZ, T. Workflow: *A Tecnologia que vai Revolucionar Processos*. São Paulo: Atlas. 2000. 2ª ed. ISBN: 85-224-2618-0. 226 p.

[CRUZ, 2003] CRUZ, T. *Sistemas, Métodos & Processos - Administrando Organizações por Meio de Processos de Negócios*. Atlas. 2003. 2ª Ed. ISBN: 85-224-3329-1. 274 p.

[FERREIRA, 1986] FERREIRA, A. B. de H.. *Novo dicionário da língua portuguesa*. Rio de Janeiro: Nova Fronteira, 1986. 1838 p.

[FUNG, 2007] FUNG A.; GRAHAM M.; WEIL D. *Full Disclosure, the Perils and Promise of Transparency*, Cambridge University Press, 2007.

[HOLZNER, 2006] HOLZNER B.; HOLZNER L. *Transparency in Global Change: The Vanguard of the Open Society*. University of Pittsburgh Press; 1 edition, 2006.

[KICZALES et al., 1997] KICZALES, G., LAMPING, J., MENDHEKAR, A., MAEDA, C., LOPES, C., LOINGTIER, J., IRWIN, I. *Aspect-Oriented Programming. European Conference on Object-Oriented Programming (ECOOP)*, LNCS 1241, Springer, Finland, pp. 220-242. June 1997.

[SOUSA, 2010] SOUSA, H. P.. *Identificação automática de serviços em uma abordagem SOA*. *Undergraduate project*. Universidade Federal do Estado do Rio De Janeiro, Escola de Informática Aplicada. Projeto Final de Graduação. 2010.