Leitura Baseada em Perspectiva: A Visão do Projetista Orientada a Objetos

Sômulo Nogueira Mafra, Guilherme Horta Travassos

COPPE/UFRJ - Programa de Engenharia de Sistemas e Computação, Cx. Postal 68.511, CEP 21945-970, Rio de Janeiro - RJ - Brasil

{somulo, ght}@cos.ufrj.br

Abstract. *This paper presents a new reading technique called OO-PBR. OO-PBR has been defined in order to allow identifying critical defects in software requirements written in natural language. OO-PBR supports requirements review by exploring object oriented modeling. The steps for defining OO-PBR, its assumptions and experimental results are discussed along the paper.*

Resumo. *Este artigo apresenta uma nova técnica de leitura, denominada OO-PBR. OO-PBR tem como principal objetivo a detecção de defeitos críticos em documentos de requisitos de software descritos em linguagem natural. OO-PBR apóia a revisão dos requisitos ao explorar a construção de modelo de projeto orientado a objetos. As etapas para a definição de OO-PBR são discutidas, destacando-se as premissas e os resultados experimentais obtidos.*

1. Introdução

O documento de requisitos desempenha um papel fundamental no desenvolvimento de software. Ao descrever as características e funcionalidades que o sistema deve possuir, o documento de requisitos representa o interesse de diferentes *stakeholders*, incluindo usuários, clientes, engenheiros de software, gerentes, entre outros (Sommerville, 2004).

Tais *stakeholders* podem possuir diferentes pontos de vista a respeito das funcionalidades e das restrições envolvidas no sistema a ser desenvolvido. Dessa forma, é fundamental que o documento de requisitos passe constantemente por um processo de atualização e de revisão. Diante desse contexto, uma importante questão identificada é:

- ✓ **Como garantir um nível adequado de qualidade no documento de requisitos?**

A condução de inspeções de software tem se demonstrado uma alternativa atraente para este propósito. A possibilidade de identificar defeitos ainda na fase inicial do projeto tem levado organizações a se preocuparem cada vez mais com a condução de inspeção em requisitos. Essa preocupação tem sido reforçada por modelos de maturidade como o MR mps (Weber *et al.*, 2004) e o CMMI (Chrissis *et al.*, 2003).

Além disso, o documento de requisitos representa também um importante insumo para a produção de artefatos em fases posteriores. Sua descrição constitui uma base para a arquitetura do sistema, para o modelo de projeto, para o desenvolvimento propriamente dito, para a validação do sistema, entre outros (Sayão e Breitman, 2005).

Entretanto, para executar satisfatoriamente uma determinada tarefa com base no documento de requisitos, o engenheiro de software necessita, primeiramente, obter deste

documento o entendimento necessário à resolução da tarefa em questão. Porém, muitas vezes as informações necessárias para a obtenção desse entendimento encontram-se dispersas e mascaradas por outras informações não relevantes no contexto da tarefa. Como conseqüência, o engenheiro de software pode desperdiçar valioso esforço na compreensão de aspectos não relevantes, caso não seja apoiado durante esta leitura (Basili *et al*., 1996). Em vista disso, um importante desafio é:

- ✓ **Como orientar o engenheiro de software na obtenção do entendimento necessário para a condução de uma tarefa com base nos requisitos?**

A aplicação de técnicas de leitura (Basili *et al*., 1996) poderia contribuir satisfatoriamente para a resolução dessas questões. Além de apoiar a revisão do documento de requisitos, a aplicação de técnicas de leitura pode apoiar o engenheiro de software também na construção de artefatos com base nos requisitos.

Como exemplo desse apoio pode ser citada a construção de modelos de projeto orientado a objetos (OO). Durante a leitura dos requisitos, a técnica de leitura pode orientar o engenheiro de software na verificação das informações necessárias para a construção de tal modelo. Dessa forma, caso o documento de requisitos não contenha essas informações significa que também não apóia satisfatoriamente a construção do modelo, caracterizando a presença de um potencial defeito no documento.

A justificativa para explorar a construção de modelos OO durante a revisão de requisitos é motivada principalmente pelo aumento na complexidade do desenvolvimento de sistemas OO observado nos últimos anos. Segundo Deligiannis *et al*. (2002), a adoção do paradigma OO não tem contribuído de forma significativa para o aumento de produtividade e qualidade no desenvolvimento de software.

O aclamado benefício do paradigma OO, que supostamente diminui o custo de manutenção evolutiva, não foi obtido em Hatton (1998). Além disso, alguns aspectos do paradigma OO, como polimorfismo, dificultam a predição do caminho de execução do software, prejudicando o planejamento de testes de integração (Lima e Travassos, 2004).

Portanto, o objetivo deste trabalho é apresentar uma nova técnica de leitura, denominada OO-PBR (***O**bject **O**riented - **P**erspective **B**ased **R**eading*). A aplicação de tal técnica durante a revisão de requisitos pode aumentar a qualidade no desenvolvimento OO ao permitir ao engenheiro de software o aprendizado sobre o problema descrito nos requisitos. Através desse aprendizado, o engenheiro de software pode: (a) identificar defeitos críticos e (b) construir modelos iniciais de projeto OO.

Entretanto, a implantação de uma tecnologia em uma organização pode trazer conseqüências indesejáveis, caso a tecnologia contenha falhas, não esteja adequada ao contexto industrial ou mesmo não seja aplicada corretamente (Silva, 2004). Nessas situações, melhorias esperadas pela implantação da tecnologia dificilmente ocorreriam, frustrando interesses envolvidos, e abalando a credibilidade do fornecedor da tecnologia.

Para minimizar a possibilidade de ocorrência desses riscos, durante seu processo de definição, OO-PBR teve a sua aplicação prática avaliada em dois estudos experimentais de observação precedidos por uma avaliação piloto (Mafra, 2006). Cada estudo possibilitou a identificação de oportunidades de melhoria, resultando numa nova versão da técnica. Através do refinamento de OO-PBR, buscou-se caracterizar o grau de

maturidade da técnica visando a uma posterior utilização no contexto industrial.

O trabalho encontra-se dividido em seis seções, incluindo esta que introduz o artigo. Na seção 2, são discutidas algumas características das técnicas de leitura. A seção 3 descreve a técnica OO-PBR. A seção 4 apresenta os resultados experimentais decorrentes de estudos de observação envolvendo OO-PBR. Na seção 5 a aplicação de OO-PBR é ilustrada através de um exemplo. Por fim, a seção 6 conclui o artigo.

2. Técnicas de Leitura de Software

Segundo Basili *et al*. (1996), uma técnica de leitura pode ser caracterizada como uma série de passos para a análise individual de um artefato de software de forma a permitir a obtenção do entendimento necessário para a execução de uma determinada tarefa.

Nesse sentido, uma técnica de leitura deve prover orientação ao engenheiro de software na execução de suas tarefas. O nível dessa orientação depende do objetivo da tarefa a ser executada, podendo variar desde um procedimento passo a passo, até a um conjunto de questões às quais o engenheiro de software deve manter a sua atenção.

Dessa forma, um importante desafio durante a definição de uma técnica de leitura está em investigar como as pessoas adquirem uma compreensão satisfatória sobre um artefato (Shull, 1998). Essa investigação é tratada na próxima subseção.

2.1. Foco na Compreensão do Artefato

Uma técnica de leitura visa a apoiar a obtenção de uma compreensão sobre um determinado artefato. Essa compreensão deve ser suficiente para que a pessoa possa entender: (a) quais partes do artefato contêm informações importantes para a tarefa, e (b) como a compreensão dessas informações contribui para a satisfação dessa tarefa.

Nesse sentido, Shull (1998) sugere observar a forma como um determinado artefato é utilizado no contexto de um ciclo de desenvolvimento. O resultado dessa observação é encapsulado em dois procedimentos: abstração e utilização (Figura 1).

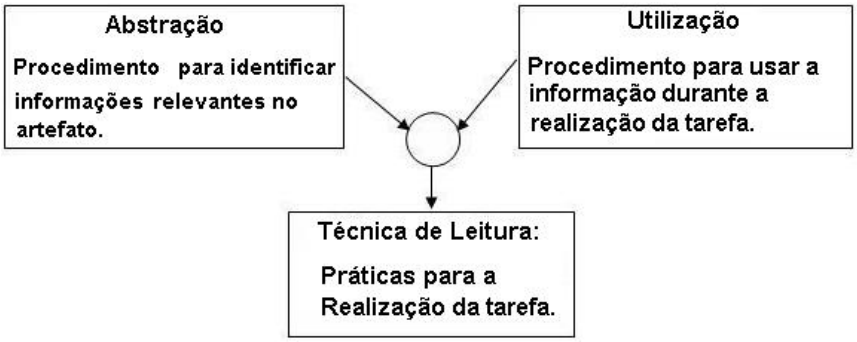

Figura 1 - Construção de uma técnica de leitura - adaptada de Shull (1998).

Enquanto o procedimento de abstração orienta o engenheiro de software na compreensão da abstração, ou seja, na compreensão das informações necessárias para executar uma determinada tarefa, o procedimento de utilização orienta o revisor a, uma vez tendo compreendido as informações necessárias para a realização da tarefa, como proceder para desempenhar a tarefa. Nesse sentido, cada um desses procedimentos aborda uma parte da tarefa e, quando combinados em uma técnica de leitura apóiam a execução da tarefa por completo, conforme ilustra a Figura 1. Na próxima subseção, é

descrito como a pesquisa sobre técnicas de leitura tem evoluído ao longo dos anos.

2.2. Pesquisa Experimental sobre Técnicas de Leitura

A pesquisa envolvendo técnicas de leitura tem sido direcionada por resultados de estudos experimentais executados ao longo dos últimos anos (Basili *et al*. 1996; Shull 1998; Travassos *et al*. 999; Denger *et al*. 2004; Thelin *et al*. 2004). À medida que novos estudos são executados, as técnicas são refinadas e novas hipóteses geradas, proporcionando assim um maior corpo de conhecimento (Ciolkowski *et al*., 2002).

De forma a contribuir para esse corpo de conhecimento experimental, optou-se pela utilização de uma metodologia rigorosa de pesquisa baseada em evidência, representada pela condução de uma revisão sistemática (Kitchenham, 2004). A condução de tal revisão permitiu, entre outros, caracterizar a aplicação prática das técnicas identificadas na literatura permitindo apontar suas limitações e seus benefícios. O resultado dessa caracterização é apresentado em Mafra e Travassos (2005).

Além disso, foi possível identificar na literatura a carência de técnicas de leitura de requisitos que explorem a construção de modelos de projeto OO. Estudos envolvendo a perspectiva de projetista adotada por PBR (Shull, 1998) utilizaram técnicas de análise estruturada para a construção de DFD's. Por sua vez, as técnicas UBR (Thelin *et al*., 2004) e OORT's (Travassos *et al*., 1999) apóiam a inspeção de modelos de projeto OO, não explorando a construção de tais modelos durante sua aplicação.

Dessa forma, concluiu-se que a definição de uma nova técnica PBR relacionada à inspeção de requisitos que explorasse a construção de modelos de projeto OO poderia ser interessante para o desenvolvimento de software. A justificativa pela definição de uma nova técnica PBR deve-se ao fato de estudos experimentais terem demonstrado que a construção de modelos durante a aplicação de PBR permitiu uma maior compreensão do problema descrito nos requisitos (Denger *et al*., 2004), (Belgamo e Fabbri, 2005).

Além disso, a motivação para a definição de uma técnica que levasse em consideração aspectos intrínsecos do paradigma OO foi reforçada pelos resultados promissores obtidos da avaliação experimental das OORT's, que envolvem a inspeção de modelos de projeto OO (Travassos *et al*., 1999), (Marucci *et al*., 2002). A definição de OO-PBR é detalhada na próxima seção.

3. OO-PBR: A Visão do Projetista Orientada a Objetos

Nesta seção, o propósito de OO-PBR e a justificativa para a definição de uma nova técnica PBR são apresentados. Além disso, o processo de definição de tal técnica é discutido em detalhes.

3.1. Propósito de OO-PBR

A técnica OO-PBR visa a apoiar uma tarefa de análise que tenha como objetivo a identificação de defeitos em um documento de requisitos descrito em linguagem natural. Nesse sentido, OO-PBR assegura que cada revisor avaliará o documento segundo a perspectiva de projetista ao fornecer um procedimento orientando-o a criar um modelo de classes com base nos requisitos.

A construção do modelo de classes ajudaria o revisor a enfocar sua revisão. O objetivo não é duplicar o trabalho realizado em outros pontos do ciclo de

desenvolvimento do software, mas criar representações que possam ser usadas como base para a criação futura de artefatos mais específicos e que possam revelar quão bem os requisitos conseguem apoiar as tarefas seguintes.

Durante a criação do modelo de classes, revisores tentariam identificar possíveis defeitos existentes. Para facilitar esse trabalho, OO-PBR fornece um conjunto de questões que são respondidas à medida que o revisor percorre as etapas da construção do modelo. Quando os requisitos não fornecem informações suficientes para responder às questões, isso significa que também não fornecem informações suficientes para apoiar o usuário na construção de um modelo de classes, caracterizando a presença de um potencial defeito. Na próxima subseção, a justificativa para a definição de uma nova técnica PBR é discutida.

3.2. Justificativa para uma nova PBR

Segundo a UML (Pender, 2003), um sistema OO pode ser projetado focando-se sua estrutura (modelagem estrutural) e sua dinâmica (modelagem dinâmica).

No que se refere à modelagem estrutural do sistema, ou seja, como os conceitos estão representados e como eles estão associados, podem ser citados o modelo de classes e o diagrama de objetos, por exemplo. Em relação à modelagem dinâmica do sistema, citam-se os modelos de caso de uso, diagramas de atividades, máquinas de estado, e diagramas de interação.

Na proposta original de PBR foram definidas três perspectivas: usuário, projetista e testador (Shull, 1998). Para cada perspectiva foi definida uma abstração específica resultando em diferentes técnicas. Para a perspectiva de usuário foi estabelecida a modelagem de casos de uso, para a perspectiva de testador foi estabelecida a construção de casos de testes utilizando partição por equivalência e análise de valor limite, e para a perspectiva de projetista foi definido o uso de diagramas de fluxo de dados como abstração para projeto estruturado. Nenhuma dessas abstrações leva em consideração a modelagem estrutural de um sistema OO.

Portanto, uma das motivações para abordar a construção de modelos de projeto OO na aplicação de PBR veio da possibilidade de se enfocar aspectos não contemplados pelas abstrações das outras perspectivas de PBR.

A opção de se utilizar modelo de classes como abstração específica de OO-PBR foi motivada principalmente pela alta popularidade e a ampla utilização desses modelos na indústria de software (Pender, 2003). Além disso, a utilização de modelos de classes pode ser uma alternativa para a modelagem do domínio sobre o qual o sistema irá atuar. Nesse sentido, o modelo de classes encontra-se no centro do processo de modelagem, ao permitir a modelagem de definições de recursos essenciais para o funcionamento do sistema. Na próxima subseção é apresentado como o procedimento de abstração pode apoiar a construção de modelos de classes a partir dos requisitos.

3.3. Procedimento de Abstração

Durante a elaboração do procedimento de abstração de OO-PBR procurou-se identificar: (a) quais tipos de informação no documento de requisitos devem ser observadas por um projetista ao elaborar um modelo de classes, e (b) como essas informações podem ser extraídas do documento de requisitos.

Informações a serem observadas. No que se refere às informações a serem observadas por um projetista de software, Beck e Cunningham (1989) apontam três aspectos:

- **Classes**. Os nomes das classes criam um vocabulário para a discussão do projeto. Portanto, um desafio durante a modelagem OO é encontrar o conjunto adequado de palavras que descrevam as classes. Os autores argumentam que o nome de uma classe deve ser internamente consistente e auto-explicativo no contexto de um projeto. Os autores sugerem a utilização de um tempo considerável para a escolha de um conjunto de nomes adequados para as classes.
- **Responsabilidade**. Responsabilidades identificam problemas a serem resolvidos. Uma responsabilidade serve como uma forma de discussão de soluções em potencial. Responsabilidades são expressas por um conjunto de orações verbais, cada uma contendo um verbo na voz ativa. Quanto maior o poder de expressão dessas orações, maior o grau de concisão e poder do projeto.
- **Colaboração**. Uma importante característica do projeto OO é que nenhum objeto (instância de uma classe) seja uma "ilha". Todos os objetos possuem relações com outros objetos, os quais demandam colaboração de serviços ou controle. Objetos são colaboradores no sentido de que podem enviar ou receber mensagens de modo a satisfazer suas responsabilidades.

Kaindl (1994) aponta ainda um quarto aspecto a ser observado:

- **Função/Propósito**. Uma função pode ter vários propósitos. Num domínio bancário, a função "saque" tem o propósito de que o cliente tenha dinheiro disponível. Um outro propósito é que a quantia sacada pelo cliente seja deduzida de sua conta. Assim, uma função pode demandar várias responsabilidades.

Dessa forma, o procedimento de abstração foi definido em três passos, a saber:

1. Identificar classes e seus atributos;
2. Identificar relacionamentos entre classes;
3. Identificar as principais funcionalidades do sistema.

Entretanto, a identificação de responsabilidades não foi levada em consideração. Durante a definição da técnica, observou-se que esse passo adicional, apesar de contribuir para a construção de um modelo de classes mais completo, não contribuiu para a detecção de defeitos, que é o objetivo principal de OO-PBR.

Extração de informações. A pesquisa sobre diretrizes que orientem o engenheiro de software na construção de um modelo de classes a partir de um documento de requisitos descrito em linguagem natural não é recente na Engenharia de Software. Diversos pesquisadores sugeriram a utilização de informações lingüísticas presentes em documentos de requisitos para a modelagem de classes (Moreno, 1997).

Entretanto, conforme observou Larman (2001), não é possível um mapeamento automático entre informações lingüísticas e classes conceituais, uma vez que palavras em uma linguagem natural tendem a ser ambíguas.

Dessa forma, Moreno (1997) propôs um mapeamento entre um subconjunto de estruturas no mundo lingüístico, chamado padrões lingüísticos, e um subconjunto de componentes do mundo conceitual, chamado padrões conceituais. A solução encontrada

pela autora foi transformar uma linguagem natural, que é potencialmente ilimitada, em uma linguagem natural mais restrita com sintaxe e semântica bem definidas. Moreno (1997) desenvolveu sua idéia levando em consideração a língua espanhola.

Entretanto, apesar do reconhecimento do trabalho de Moreno (1997), optou-se pela definição de diretrizes que apoiassem a elaboração de modelos de classes a partir da definição de requisitos descritos informalmente através de uma linguagem natural. Dessa forma, eventuais ambigüidades, imprecisões e inconsistências decorrentes da utilização de uma linguagem natural que prejudicassem a elaboração do modelo de classes poderiam ser exploradas como potenciais indicadores da presença de defeitos no documento de requisitos. Na próxima subseção é apresentado como o procedimento de utilização de OO-PBR foi definido, de forma a apoiar a detecção de defeitos.

3.4. Procedimento de Utilização

Durante a elaboração do procedimento de utilização de OO-PBR, buscou-se definir um conjunto de questões que apoiassem a detecção de defeitos tipicamente encontrados em documentos de requisitos de software.

Dessa forma optou-se por dividir tais defeitos em duas categorias: (a) defeitos relacionados à abstração e (b) defeitos relacionados a aspectos gerais.

Por defeitos relacionados à abstração, entende-se como quaisquer defeitos presentes nos requisitos que dificultem (ou impossibilitem) a construção de um modelo de classes, como a omissão de informações sobre tipos de atributos, estados etc.

Por defeitos relacionados a aspectos genéricos, entende-se como defeitos que não estejam diretamente relacionados à abstração utilizada (no caso, um modelo de classes), mas que afetem a qualidade dos requisitos dificultando o entendimento, por exemplo.

Essa opção pode ser considerada uma nova proposta em relação à definição original das técnicas PBR, que limitavam-se a apontar defeitos relacionados apenas às abstrações sendo elaboradas. As características de qualidade verificadas originalmente eram associadas somente à verificação das informações necessárias para a construção da abstração associada à perspectiva; a verificação de aspectos que pudessem afetar o entendimento do documento não era contemplada.

De forma a categorizar eventuais defeitos detectados por OO-PBR, decidiu-se utilizar a taxonomia de defeitos proposta em Shull (1998). As categorias são:

- **Omissão**: um defeito do tipo omissão é caracterizado quando: (1) algum requisito importante relacionado à funcionalidade, ao desempenho, às restrições de projeto, ao atributo, ou à interface externa não foi incluído; (2) não está definida a resposta do software para todas as possíveis situações de entrada de dados; (3) faltam seções no documento de requisitos; (4) faltam referências de figuras, tabelas, e diagramas; (5) falta definição de termos e unidades de medidas.
- **Ambigüidade**: um requisito tem várias interpretações devido a diferentes termos utilizados para uma mesma característica ou vários significados de um termo para um contexto em particular.
- **Inconsistência**: dois ou mais requisitos são conflitantes e o conhecimento sobre o domínio não permite identificar qual requisito é verdadeiro.

- **Fato Incorreto**: um requisito descreve um fato que não é verdadeiro, considerando-se as condições solicitadas para o sistema.
- **Informação Estranha**: as informações fornecidas no requisito não são necessárias ou mesmo usadas.
- **Outros**: outros defeitos como a inclusão de um requisito numa seção errada do documento.

Vale ressaltar que a seguinte categorização de defeitos proposta nesta taxonomia não é ortogonal, ou seja, um determinado defeito pode ser classificado em uma ou mais categorias. Além disso, a proposta de tal taxonomia não é definitiva, e cada organização pode adaptá-la à sua realidade, incluindo ou excluindo as categorias identificadas.

Nesse sentido, o procedimento de utilização de OO-PBR, ou seja, o procedimento definido para a identificação de defeitos em documentos de requisitos, foi composto por uma série de questões que visavam a verificar os seguintes critérios: completeza, clareza, consistência, corretude, concisão e organização adequada. Além disso, optou-se, em cada questão, por orientar explicitamente o revisor sobre uma possível categoria de defeito a ser reportada de acordo com a taxonomia de defeitos.

3.5. Mapeamento dos Procedimentos de OO-PBR

Tendo definido-se os procedimentos de abstração e de utilização, o desafio seguinte foi o de definir uma forma adequada para o mapeamento desses procedimentos. Decidiu-se dividir OO-PBR em três seções, a saber: Instruções, Passos e Questões.

Na seção de instruções são determinados os procedimentos os quais o revisor deve seguir para aplicar a técnica de forma adequada. A seção de passos contém orientações para a construção do modelo de classes. Finalmente, na seção de questões, encontram-se listadas as questões para a identificação de defeitos. Para auxiliar a construção do modelo de classes, e o relato de discrepâncias, foram desenvolvidos alguns formulários descritos na próxima subseção.

3.6. Formulários de Apoio à OO-PBR

No que se refere à construção do modelo de classes, foram desenvolvidos o Formulário de Classes e o Formulário de Funcionalidades. O objetivo desses formulários está em permitir ao revisor preencher informações extraídas do documento de requisitos que o auxiliem na construção do modelo de classes.

Portanto, uma observação importante sobre OO-PBR, é que a instrumentação proporcionada pela técnica não visa a apoiar a construção de modelo de classes, mas sim a extração de informações que apóiem a construção de um modelo de classes (sempre mantendo o foco na identificação de defeitos nos requisitos, que é o objetivo da técnica). Dessa forma, o modelo de classes poderia ser construído baseado nas informações presentes em tais formulários, seja através de um aplicativo de software ou através de lápis e papel, por exemplo. A idéia foi manter a aplicação da técnica independente de tecnologias que possam restringir a sua aplicação.

Em relação à identificação de discrepâncias, foi desenvolvido o Formulário de Discrepâncias. Tal formulário tem como objetivo permitir o relato de informações como a descrição da discrepância, o tipo da discrepância, a localização da discrepância no

documento de requisitos etc. Na Figura 2, um extrato de OO-PBR é apresentado. A técnica OO-PBR pode ser obtida mediante contato com os autores.

OO-PBR

Instruções

Por favor, siga os **passos** para construir o esboço de um modelo de classes baseado no documento de requisitos a ser inspecionado. À medida que for lendo cada requisito, construa o modelo, e responda as **questões**, visando a identificar eventuais discrepâncias no documento de requisitos. **Lembre-se:**

- O seu objetivo é identificar discrepâncias no documento de requisitos sendo inspecionado; dentro do contexto da técnica, a construção do modelo de classes tem importância secundária, atuando apenas como um instrumento para identificar discrepâncias. Portanto, não seja **exaustivo** na construção do modelo.
- Não faça **suposições**; caso não consiga extrair dos requisitos as informações necessárias para a construção do modelo, relate essa omissão de informações como discrepâncias, ao invés de tentar deduzi-las.
- Não tente corrigir discrepâncias; apenas descreva-as de forma a permitir ao autor do documento corrigi-las.

Passos

1) Leia um requisito por vez de forma a extrair potenciais nomes de **classes** e de **atributos**. Extraia nomes que representem conceitos que você considere pertinentes ao problema.
   - Preencha cada nome extraído no Formulário de Classes. Verifique se o nome representa uma classe ou um atributo. Caso o nome represente um atributo, indique a qual classe ele pertence.
   - Para cada atributo, verifique se foi especificada uma unidade de valor. Ex.: "tempo decorrido (atributo) em número de dias (unidade de valor)". De acordo com a unidade de valor, identifique seu provável **tipo**.
   - Caso algum atributo represente uma faixa de valores (como o **estado** de um objeto, por exemplo), extraia todos os possíveis valores descritos no documento, e descreva-os no Formulário de Classes.
2) Leia novamente o requisito de forma a extrair **relacionamentos** entre as classes identificadas. Possíveis relacionamentos vêm dos verbos de orações envolvendo os nomes das classes.
   - Represente potenciais estruturas *"é um"* com **herança**, caso pertinente. Identifique classes que possuam algum atributo em comum, e verifique se essas classes podem ser representadas em hierarquias de herança.
   - Represente os relacionamentos no modelo. Extraia as quantidades descritas no requisito para determinar a **multiplicidade** dos relacionamentos.
3) Caso o requisito lido seja requisito funcional, extraia as **funcionalidades** descritas, representando-as através de orações verbais. Ao extrair uma funcionalidade, preencha-a no Formulário de Funcionalidades

Figura 2 - Extrato da técnica OO-PBR.

Na próxima seção é descrito como a definição de OO-PBR foi influenciada pela condução de estudos experimentais de observação.

4. Avaliação de OO-PBR Através de Experimentação

Durante seu processo de definição, a técnica OO-PBR foi avaliada em dois estudos experimentais de observação precedidos por uma avaliação piloto (Mafra, 2006). Esses estudos foram conduzidos em ambiente acadêmico, com estudantes de graduação e de pós-graduação. Os participantes envolvidos possuíam diferentes níveis de experiências e habilidades referentes a inspeções de software e à modelagem OO. 90% (19 de 21) dos participantes possuíam experiência em desenvolvimento de software na indústria.

Tais estudos não tiveram como objetivo o teste de hipóteses; portanto, a comparação com outras abordagens de inspeção adotadas na indústria foi descartada nesse primeiro momento. A formulação de quaisquer hipóteses durante essa etapa inicial de elaboração da técnica poderia ser vista como mera especulação, dado o grau de imaturidade de uma tecnologia recém-definida.

Portanto, os estudos tiveram como principal objetivo proporcionar aos pesquisadores uma compreensão refinada a respeito da aplicação prática de OO-PBR. Através dessa compreensão seria possível apontar com níveis razoáveis de segurança eventuais pontos fortes e fracos da tecnologia. Dessa forma, tais aspectos poderiam ser explorados em futuros estudos a serem conduzidos em contexto industrial.

Durante esses estudos foi registrada a forma como os participantes aplicavam OO-PBR. O registro dessas informações foi possível através da observação dos pesquisadores, que tomaram nota de cada ação desempenhada pelos participantes, do preenchimento pelos participantes de formulários pós-experimento, e de sessões de entrevistas individuais. Nesse sentido, foi possível avaliar como a técnica influenciava (positiva ou negativamente) o desempenho dos participantes.

Como conseqüência, a técnica OO-PBR foi refinada visando a aprimorar seu processo de leitura. Os principais resultados experimentais são apresentados em tópicos.

Objetivo de OO-PBR. Originalmente, OO-PBR visava a apoiar duas tarefas: a) detecção de defeitos em requisitos e b) construção de um modelo de classes. Entretanto, foi observado que a construção de um modelo de classes completo tomava parte considerável do tempo de inspeção. Como conseqüência, o desempenho do revisor era prejudicado, resultando numa baixa quantidade de defeitos detectados. Diante disso, optou-se pela construção de um esboço de modelo de classes, ao invés de tentar explorar todos os aspectos envolvidos na construção de um modelo de classes completo.

Modelagem Estrutural *versus* Modelagem Dinâmica. O fornecimento de diretrizes para a identificação de responsabilidades das classes aumentou o grau de dificuldade de aplicação de OO-PBR. Além disso, a aplicação de tais diretrizes não contribuiu satisfatoriamente para a detecção de defeitos. Como resultado, decidiu-se por não contemplar esse passo em OO-PBR. A modelagem de tais aspectos dinâmicos seria recomendável para outras representações, como diagramas de seqüência.

Presença de um Processo Sistemático. A presença de um processo sistemático foi apontada como um aspecto positivo de OO-PBR. A utilização de tal processo proporcionou a revisores inexperientes a aprendizagem de boas práticas relacionadas a inspeções de requisitos e à construção de modelos de classes.

Utilidade de OO-PBR. Outro efeito positivo relatado pelos participantes sobre a utilidade de OO-PBR foi a possibilidade de identificar-se defeitos mesmo após os documentos inspecionados terem sido definidos e aprovados por *stakeholders* experientes no domínio. Essa percepção da utilidade de OO-PBR pelos participantes é um fator fundamental para uma futura transferência da técnica para a indústria.

Efetividade das Questões de OO-PBR. As questões propostas por OO-PBR foram analisadas em relação ao apoio para a detecção de defeitos. Assim, foi possível identificar quais questões contribuem para a detecção de defeitos, e quais são inócuas para esse fim. Como conseqüência dessa análise, novas questões foram incluídas, algumas foram alteradas e outras, que demonstraram ser inócuas, foram excluídas.

Como resultado, as questões de OO-PBR foram apontadas como um dos principais pontos positivos da técnica. Participantes ressaltaram principalmente a simplicidade e a objetividade das questões, além da presença de exemplos que ilustram defeitos típicos de serem capturados por cada questão.

Efetividade dos passos para o esboço de um modelo de classes. Os passos para a construção do esboço de um modelo de classes receberam comentários positivos dos participantes. Além disso, participantes de ambos os estudos afirmaram que o esboço do modelo de classes teve papel fundamental no entendimento do problema e dos conceitos descritos nos requisitos. Na próxima seção, OO-PBR tem sua aplicação ilustrada.

5. Aplicando OO-PBR

Nesta seção, a aplicação de OO-PBR é ilustrada através de um exemplo. A Figura 3 apresenta o trecho de um documento de requisitos a ser utilizado nesse exemplo.

Funções do Produto (Pág. 14)	Legenda:
Os **vôos** podem ser **comerciais** ou **fretados**. Cada vôo é descrito por um **número**, e por um local de **origem** e um local de **destino**. Além disso, por questões referentes ao **DAC** (Departamento de Aviação Civil), cada número deve ser composto por 4 **algarismos**. Um vôo pode ser **suspenso**, **replanejado**, ou estar **ativo**, em um determinado **momento**.	**Negrito**: Potenciais Classes e Atributos Sublinhado: Potenciais Relacionamentos

Figura 3 - Trecho de um documento de requisitos utilizado no exemplo.

5.1. Passo 1: Identificando Classes e Atributos

Durante a aplicação do Passo 1, o revisor é orientado a ler um requisito por vez de forma a extrair potenciais nomes de classes e de atributos. Para isso, o revisor deve:

- Preencher cada nome extraído no Formulário de Classes. Verificar se o nome representa uma classe ou um atributo. Caso o nome represente um atributo, o revisor deve indicar a qual classe o atributo pertence.
- Para cada atributo, o revisor deve verificar se foi especificada alguma unidade de valor. Ex.: "tempo decorrido (atributo) em número de dias (unidade de valor)". De acordo com a unidade de valor, o revisor deve identificar o seu provável tipo.
- Caso algum atributo represente uma faixa de valores (como o estado de um objeto, por exemplo), o revisor deve extrair todos os possíveis valores descritos no documento, e descrevê-los no Formulário de Classes.

De acordo com o trecho do documento de requisitos ilustrado na Figura 3, uma classe em potencial identificada seria o conceito "vôo". O preenchimento das informações relacionadas a vôo é ilustrado na Figura 4. Vale ressaltar que nem todos os nomes identificados na descrição do requisito lido ocasionarão na identificação de classes ou atributos, como no caso de "DAC", "algarismo" ou "momento", por exemplo. Um provável modelo de classes resultante da leitura do trecho do documento de requisitos é ilustrado na Figura 5.

Classe:	Vôo				
Atributos					
Nome	Situação	**Tipo**	Estado	**Valores**	Replanejado, suspenso, ativo.
Nome	Número	**Tipo**	Numérico	**Valores**	N/D
Extraído de:	Funções do Produto, Pág. 14.				

Figura 4 - Preenchimento do Formulário de Classes.

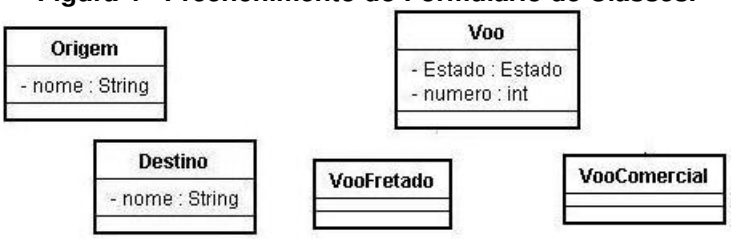

Figura 5 - Modelo de classes após a aplicação do Passo 1.

5.2. Passo 2: Identificando Relacionamentos

Nesse passo, o revisor é orientado a reler o requisito de forma a extrair relacionamentos entre as classes identificadas. Nesse sentido, possíveis relacionamentos vêm dos verbos de orações envolvendo os nomes das classes. Para isso, o revisor deverá representar:

- Potenciais estruturas "é um" com herança, caso pertinente. Além disso, o revisor deverá identificar classes que possuam algum atributo em comum, e verificar se essas classes podem ser representadas em hierarquias de herança.
- Os relacionamentos no modelo. Para isso, o revisor deverá extrair as quantidades descritas no requisito para determinar a multiplicidade dos relacionamentos.

O modelo de classes resultante do passo 2 é apresentado na Figura 6.

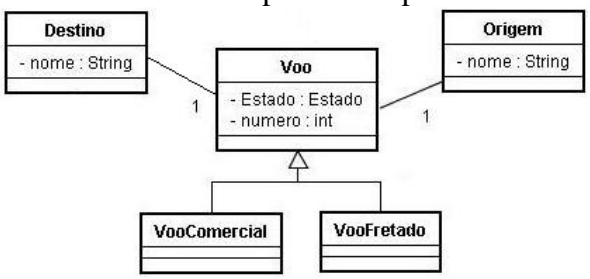

Figura 6 - Modelo de classes após a aplicação do Passo 2.

5.3. Passo 3: Identificando Funcionalidades

Caso o requisito lido represente uma funcionalidade, o revisor deverá, durante a aplicação do Passo 3, extrair as funcionalidades descritas, e representá-las através de orações verbais. Ao extrair uma funcionalidade, o revisor deverá preenchê-la no Formulário de Funcionalidades determinando quais classes, presentes no Formulário de Classes, são candidatas a satisfazerem-na. Além disso, para cada funcionalidade identificada, o revisor deve extrair (caso pertinente) os dados de entrada e de saída requeridos para sua execução, descrevendo-os no Formulário de Funcionalidades, além dos casos de sucesso e de erro. No que se refere ao trecho do documento de requisitos ilustrado na Figura 3, uma provável funcionalidade extraída pode ser representada pela Figura 7.

Funcionalidade:	Gerenciar Pouso de Avião		
Dados de Entrada:	N/D		
Dados de Saída:	N/D		
Casos de Sucesso:	Pista Livre	**Casos de Erro:**	Cabine replanejar pouso
Classes Candidatas:	Avião, Cabine de Controle e Pista de Pouso.		
Extraído de:	Funções do Produto, Pág. 14.		

Figura 7 - Preenchimento do Formulário de Funcionalidades.

5.4. Detectando Defeitos

Após ter aplicado todos os passos de OO-PBR ao requisito sendo lido, o revisor é orientado a responder a uma série de questões visando a detectar defeitos no documento de requisitos. Vale ressaltar que em conjunto com OO-PBR é fornecido um anexo contendo exemplos ilustrando como cada questão de OO-PBR pode ser aplicada.

Ao responder as questões para detecção de defeitos de OO-PBR, o revisor é orientado a manter o foco em um conjunto delimitado de informações. Entre o conjunto

delimitado de informações, encontram-se a descrição de um requisito em particular, os formulários de classes e de funcionalidades, e o modelo de classes construído. À medida que ler um requisito, o revisor deverá responder questões relacionadas:

- À descrição do requisito:

Q1.1 O requisito está descrito com termos vagos? Caso positivo, reporte *omissão*.

Q1.2 O requisito está descrito de forma ambígua? Caso positivo, reporte *ambigüidade*.

Q1.3 O requisito está descrito numa seção adequada? Caso negativo, reporte *localização incorreta*.

Q1.4 Há sentenças onde o ponto referenciado não é claro? Caso positivo, reporte *ambigüidade*.

Q1.5 Há presença de sinônimos (mais de um nome representando um mesmo conceito) ou homônimos (um nome representando mais de um conceito)? Caso positivo, reporte *ambigüidade*.

Q1.6 O requisito está descrito de forma a permitir testá-lo? Caso negativo, reporte *omissão*.

Q1.7 O requisito representa uma funcionalidade que não deve ser implementada na solução final? Caso positivo, reporte *informação estranha*.

Q1.8 O requisito contradiz outros trechos do documento? Caso positivo, reporte *inconsistência*.

Q1.9 O requisito faz sentido sobre o que você conhece sobre o domínio? Caso negativo, reporte *fato incorreto*.

- Ao Formulário de Classes:

Analise as classes e seus atributos descritos no **Formulário de Classes**. As informações presentes no requisito lido (ou em alguma seção de definição de dados ou similar) permitem:

Q2.1 Estipular os tipos para todos os atributos dessas classes? Caso contrário, reporte *omissão*.

Q2.2 Estipular todos os possíveis valores para os atributos que representam faixa de valores (Ex.: estados de um objeto)? Caso contrário, reporte uma *omissão*.

- Ao Formulário de Funcionalidades:

Analise as funcionalidades descritas no **Formulário de Funcionalidades**, e responda:

Q3.1 São descritos os casos de sucesso e de erro para cada funcionalidade? Caso negativo, reporte *omissão*.

Q3.2 É descrito passo a passo como o sistema deve proceder para executar as funcionalidades? Caso as funcionalidades não estejam devidamente especificadas, reporte *omissão*.

Q3.3 São descritas as entradas para cada funcionalidade? Caso negativo, reporte *omissão*.

Q3.4 São descritas as saídas para cada funcionalidade? Caso contrário, reporte *omissão*.

Após responder a essas questões, o revisor é orientado a ler o requisito seguinte, aplicando novamente os passos 1, 2 e 3, além das questões acima relatadas.

Caso tenha lido todos os requisitos, finalizando dessa forma a construção do modelo de classes, o revisor é orientado a responder às seguintes questões:

Analise as classes e seus atributos descritos no **Formulário de Classes**, e responda:

Q4.1 Os conceitos pertinentes ao domínio da aplicação estão descritos numa seção de Glossário ou similar? Caso negativo, importantes definições podem não estar detalhadas adequadamente. Reporte *omissão*.

Q4.2 Existem classes que devem ser excluídas, uma vez que: (a) não são responsáveis pela execução de uma funcionalidade, (b) não armazenam informações relevantes ao sistema ou (c) não representam conceitos relevantes? Caso positivo, reporte *informação estranha*.

Q4.3 Conceitos que você julgue pertinentes ao problema estão representados nas classes e em

seus respectivos atributos? Caso contrário, reporte *omissão*.

Analise as funcionalidades descritas no **Formulário de Funcionalidades**, e responda:

Q4.4 As funcionalidades extraídas são suficientes para a resolução do problema descrito nos requisitos? Caso contrário, reporte *omissão*.

Analise o esboço do **modelo de classes**, e responda as seguintes questões:

Q4.5 As informações presentes nos requisitos permitem a você estipular a multiplicidade dos relacionamentos envolvidos? Caso contrário, reporte *omissão*.

Q4.6 Os requisitos descrevem explicitamente todos os relacionamentos entre as classes extraídas no Formulário de Classes? Caso contrário, reporte *omissão*.

À medida que for identificando discrepâncias no documento de requisitos, o revisor deve preencher o Formulário de Discrepâncias, conforme ilustrado na Figura 8.

Número	Tipo	Local.	Descrição da discrepância
1	Omissão	RF12	Não é possível identificar o tipo do atributo de acordo com a descrição dos requisitos.
2	Omissão	Pág. 5	A descrição do requisito não permite identificar a multiplicidade entre o relacionamento das classes

Figura 8 - Preenchimento do Formulário de Discrepâncias.

6. Conclusão

Este artigo apresentou uma nova técnica de leitura denominada OO-PBR. OO-PBR visa a apoiar a revisão de documentos de requisitos de software descritos em linguagem natural através da exploração de aspectos intrínsecos do paradigma OO.

OO-PBR pode contribuir consideravelmente para a melhoria no desenvolvimento do software OO ao permitir a redução de custos e o aumento na qualidade dos modelos desenvolvidos. A remoção eficiente de defeitos críticos na fase onde eles são menos custosos de serem corrigidos diminui consideravelmente o custo total do projeto. Por outro lado, a aplicação de OO-PBR pode fornecer ao engenheiro de software orientação sobre como modelar o sistema OO a partir dos requisitos, permitindo uma visão inicial da estrutura do projeto que poderia ser utilizada no plano inicial de testes de integração, aumentando a qualidade do produto.

OO-PBR teve sua aplicação prática avaliada em dois estudos de observação que permitiram o refinamento da técnica. Através desse processo de refinamento, houve a preocupação em minimizar a possibilidade de ocorrência de eventuais riscos relacionados à utilização de uma técnica relativamente imatura no contexto industrial.

Agradecimentos

Os autores agradecem o apoio financeiro do CNPq para a realização deste trabalho.

Referências

Basili, V. *et al*. (1996) “The Empirical Investigation of Perspective-Based Reading”, Empirical Software Engineering: An International Journal, vol. 1, n. 2, pp. 133-164, 1999.

Beck, K., Cunnigham, W. (1989) “A Laboratory For Teaching Object-Oriented Thinking”. Proceedings of the 4th ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications, OOPSLA’89, Volume 24 Issue 10, October 1999.

Belgamo, A., Fabbri, S. (2005) "GUCCRA: Técnica de Leitura para apoiar a Construção Modelos de Casos de Uso e a Análise de Documentos de Requisitos" In: XIX SBES, Uberlândia, MG, Brasil.

Chrissis, M., Konrad, M., Shrum, S. (2003) "CMMI: Guidelines for Process Integration and Product Improvement", Addison Wesley.

Ciolkowski, M., Laitenberger, O., Rombach, D., Shull, F., Perry, D. (2002) "Software Inspections, Reviews & Walkthroughs", 24th ICSE, May 19 - 25, 2002, Orlando, Florida.

Deligiannis, I, Shepperd, M., Webster, S., Roumeliotis, M. (2002) "Review of Experimental Investigations into Object-Oriented Technology", Empirical Software Engineering, vol.7, n.3, Kluwer Academic Publishers.

Denger, C., Ciolkowski, M., Lanubile, F. (2004) "Investigating the Active Guidance Factor in Reading Techniques for Defect Detection", Proc. of ISESE'04, Redondo Beach, California.

Hatton, L. (1998) "Does OO really match the way we think?" IEEE Software, 15(3):46-54, May/June.

Kaindl, H. (1994) "Object-Oriented Approaches in Software Engineering and Artificial Intelligence", Journal of Object-Oriented Programming 6, 8, 38-45.

Kitchenham, B. (2004) "Procedures for Performing Systematic Reviews", Joint Technical Report Software Engineering Group, Department of Computer Science Keele University, United King and Empirical Software Engineering, National ICT Australia Ltd, Australia.

Larman, C. (2001) "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process". Prentice Hall PTR, Upper Saddle River, NJ, USA.

Lima, G., Travassos, G. (2004) "Testes de Integração Aplicados a Software Orientado a Objetos: Heurísticas para Ordenação de Classes". III SBQS, Brasília, DF.

Mafra, S. N., Travassos, G. H. (2005) "Técnicas de Leitura de Software: Uma Revisão Sistemática". In: XIX Simpósio Brasileiro de Engenharia de Software, SBES'05, Uberlândia, MG, Brasil.

Mafra, S. N. (2006) "Leitura Baseada em Perspectiva: A Visão do Projetista Orientada a Objetos". Dissertação de Mestrado, PESC, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

Marucci, R. *et al*. (2002) "OORTs/ProDeS: Definição de Técnicas de Leitura para um Processo de Software Orientado a Objetos". In: I SBQS, Gramado, RS, Brasil.

Moreno, A. (1997) "Object-Oriented Analysis from Textual Specifications". In Proc. of 9th international conference on SEKE, Madrid, Spain, 1997.

Pender, T. (2003) "UML Bible". Wiley Publishing, Inc, Indianapolis, IN, USA.

Sayão, M., Breitman, K. (2005) "Gerência de Requisitos". Mini-curso apresentado no XIX Simpósio Brasileiro de Engenharia de Software, Uberlândia, MG, Brasil.

Shull, F. (1998) "Developing Techniques for Using Software Documents: A Series of Empirical Studies", PhD Thesis, Department of Computer Science, University of Maryland, USA.

Silva, L. (2004) "Uma abordagem com apoio ferramental para aplicação de técnicas de leitura baseada em perspectiva". Diss. de Mestrado, PESC, COPPE/UFRJ, Rio de Janeiro, Brasil.

Sommerville, I. (2004) "Software Engineering", Pearson, 2004.

Thelin, T., Runeson, P., Wöhlin, C., Olsson, T., Andersson, C. (2004) "Evaluation of Usage-Based Reading-Conclusions after Three Experiments", Emp. Soft. Eng., V.9, 1-2 (77-110).

Travassos, G., Shull, F., Fredericks, M., Basili, V. (1999) "Detecting defects in object-oriented designs: using reading techniques to increase software quality". Proceedings of the 14th ACM SIGPLAN conference on OOPSLA, Vol. 34 Issue 10.

Weber, K., Rocha, A. R., *et al.* (2004) "Modelo de Referência para Melhoria de Processo de Software: uma abordagem brasileira". XXX Conf. Latino-americana de Informática, Arequipa - Peru, 2004.