O Uso de Pontos de Vista na Elicitação de Requisitos

Julio Cesar Sampaio do Prado Leite

Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
R. Marquês de S. Vicente 225 Rio de Janeiro 22453

1989

Resumo

Recentemente a comunidade de engenharia de software vem mostrando cada vez mais seu interesse pela parte inicial da construção de software. Esta parte é conhecida como a análise de requisitos. Em particular, cresce o interesse pela tarefa de elicitação dos requisitos, ou seja, de tornar explícito os desejos, as intenções e necessidades dos clientes em relação ao software a ser construído. Este artigo apresenta os conceitos usados e resultados obtidos na aplicação de pontos de vista na análise de requisitos. A análise de pontos de vista é apresentada como uma alternativa na tarefa de validação da elicitação de requisitos. Com isto pretende-se dispor de uma validação antecipada do entendimento, pelos engenheiros de software, do software a ser construído. Nosso trabalho propõe um formalismo para a representação de pontos de vista, bem como procedimentos automáticos para a análise de diferentes pontos de vista sobre um mesmo problema. Estes procedimentos automáticos produzem uma agenda em que discrepâncias entre visões são identificadas e classificadas.

Abstract

Requirements modeling demands that the knowledge of what should be modeled be available. Acquiring this knowledge or eliciting the necessary requirements is recognized to be a very hard problem. Different suggestions have been made to alleviate this problem. We present an approach centered on the very early validation of requirements, exploring the existence of multi viewpoints in describing a given situation.

1 Introdução

A validação de software, ou seja, a confirmação de que o produto é aquele desejado pelo usuário, ocorre, normalmente, no fim do ciclo de vida. O teste do sistema, como é

comumente conhecida esta validação, é o teste integrado dos programas do sistema pelo usuário.

Recentemente, vários pesquisadores [Agresti 87] têm proposto linguagens de especificação executáveis, de forma a trazer o processo de validação para mais perto da fase de definição.

Nosso trabalho parte de uma proposta em que a validação é feita antecipadamente à especificação. Na nossa proposta a validação é feita no próprio processo de elicitação de requisitos. Outros pesquisadores [Rich 87] [Fickas 87] partilham da mesma idéia, mas propõem uma validação baseada num domínio. Para estes autores, um requisito de software seria validado contra um domínio já codificado. Portanto a validação seria feita contra o domínio da aplicação, que seria a mais fiel possível representação do mundo real.

O uso de domínios para a validação possibilita que seja feita uma diferenciação entre problemas de correteza e completeza, facilidade não presente nas atuais técnicas para validação. O problema com o uso de domínios é, não só o seu alto custo, como também a complexidade de sua construção [Arango 89].

A resolução de pontos de vista como meio para a validação de requisitos, oferece uma alternativa ao uso de domínios, já que pode diferenciar problemas de correteza e completeza e não depende da construção de um domínio. Esta capacidade de diferenciar entre correteza e completeza possibilita atacar um ponto crucial do desenvolvimento de software, isto é, será que estamos construindo o software desejado, ou não? Atacando o problema da validação logo no início do processo de construção estamos evitando ocorrer em possíveis erros de projeto, que mais tarde serão, não só de difícil correção, como de alto custo.

2 O Processo da Análise de Requisitos

A hipótese estudada em nossa pesquisa, de que a resolução de pontos de vista auxilia a validação de requisitos, foi enunciada e aplicada usando-se uma visão da análise de requisitos orientada a processos [Leite 87]. Nesta visão orientada a processo, o processo de elicitação é diferenciado do processo de modelagem dos requisitos.

O processo de elicitação compreende os processos de: coleta de fatos, validação dos fatos e comunicação. O processo de modelagem compreende os processos de: organização e representação. Apesar da dificuldade de separá-los, já que a análise de requisitos é um emaranhado desses processos, esta classificação nos auxilia na tentativa de melhor compreender esta tão difícil tarefa. A Figura 1 nos dá a classificação utilizada.

Resolução de pontos de vista é entendido como um processo composto de quatros sub-processos: identificação, classificação, avaliação e integração (Figura 2). De acordo com nossa definição de elicitação, identificação e classificação fazem parte da validação de fatos, enquanto avaliação e integração fazem parte da comunicação.

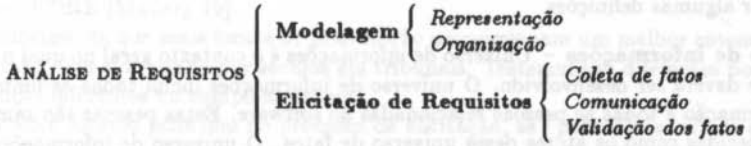

Figura 1: O Processo da Análise de Requisistos

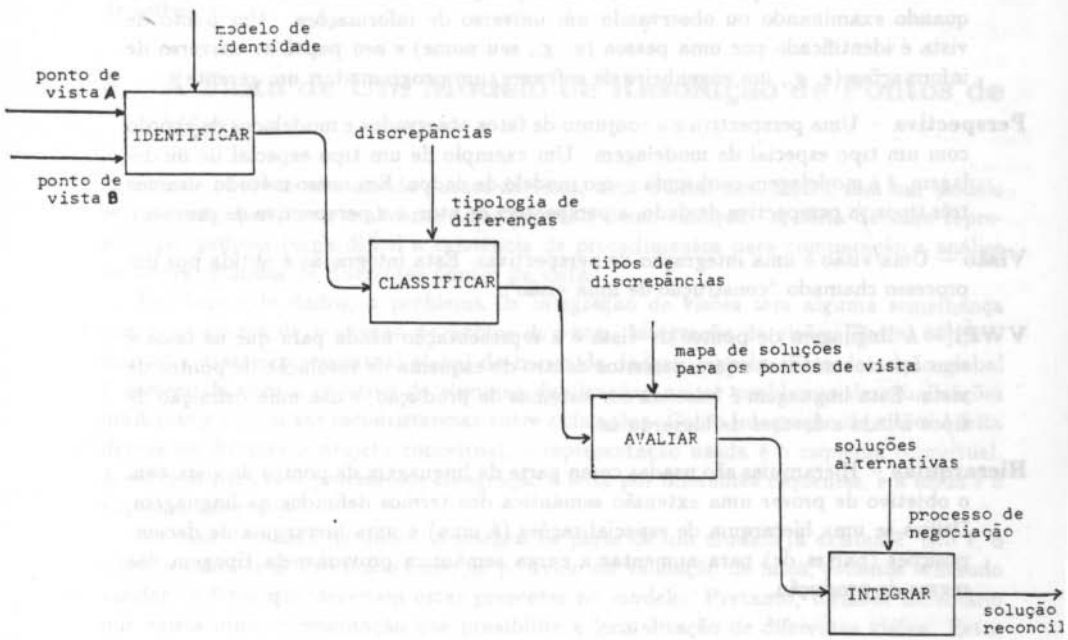

Figura 2: Resolução de Pontos de Vista

3 Definições

A resolução de pontos de vista é utilizada no processo de validação dos fatos. A aquisição desses fatos supõe uma orientação baseada na busca de palavras chaves da aplicação e numa representação própria. Portanto, um dos problemas básicos abordados pela nossa pesquisa é: como ter uma representação que possibilite a aplicação de um modelo de resolução de pontos de vista (Figura 2). Antes de melhor definirmos o problema convêm estabelecer algumas definições.

Universo de informações – Universo de informações é o contexto geral no qual o software deverá ser desenvolvido. O universo de informações inclui todas as fontes de informação e todas as pessoas relacionadas ao software. Estas pessoas são também conhecidas como os atores desse universo de fatos. O universo de informações é a realidade circunstanciada pelo conjunto de objetivos definidos pelos que demandam o software.

Pontos de Vista – Um ponto de vista é uma posição mental usada por uma pessoa quando examinando ou observando um universo de informações. Um ponto de vista é identificado por uma pessoa (e. g., seu nome) e seu papel no universo de informações (e. g., um engenheiro de software, um programador, um gerente).

Perspectiva – Uma perspectiva é o conjunto de fatos observados e modelados de acordo com um tipo especial de modelagem. Um exemplo de um tipo especial de modelagem, é a modelagem conhecida como modelo de dados. Em nosso método, usamos três tipos: a perspectiva de dado, a perspectiva de ator, e a perspectiva de processo.

Visão – Uma visão é uma integração de perspectivas. Esta integração é obtida por um processo chamado "construção de uma visão".

VWPl – A linguagem de pontos de vista é a representação usada para que os fatos e seus relacionamentos sejam descritos dentro do esquema de resolução de pontos de vista. Esta linguagem é baseada em sistemas de produção, e usa uma definição de tipos aliada a árvores de hierarquias.

Hierarquias – Hierarquias são usadas como parte da linguagem de pontos de vista com o objetivo de prover uma extensão semântica dos têrmos definidos na linguagem. Utiliza-se uma hierarquia de especializações (é uma) e uma hierarquia de decomposições (partes de) para aumentar a carga semântica provida pela tipagem das regras de produção.

4 A Existência de Pontos de Vista

Na tarefa de modelar as expectativas dos usuários dentro de um universo de informações, o engenheiro de software pode encontrar, e usualmente encontra, opiniões diferentes sobre

o problema em questão. Diferentes engenheiros de software, quando modelando as expectativas de usuários num **mesmo** universo de informações, produzem diferentes modelos. O mesmo engenheiro de software, quando modelando o mesmo universo de informações pode fazê-lo usando diferentes perspectivas (por exemplo, um modelo de dados *versus* um modelo de processos).

Tudo isso é conhecido. O ponto importante é que alguns métodos de engenharia de software usam pontos de vista com o objetivo de produzir um modelo que "melhor" espelhe as expectativas dos usuários no universo de informações. Um exemplo de tal método é CORE [Mullery 79].

O princípio de que mais fontes de informação proporcionam um melhor entendimento de um caso, tem sido usado por séculos em tribunais. Testemunhas diferentes podem ter lembranças diferentes ou complementares.

Usando o mesmo princípio no processo de elicitação, as "possibilidades" de detectar problemas de correteza e completeza serão maiores. No entanto, para se lucrar com este princípio é necessário comparar e analizar diferentes pontos de vista.

A análise e comparação de pontos de vista como proposto por Ross (SADT) e Mullery (CORE) são tarefas informais, e por isso dependem basicamente de um "bom" engenheiro de software.

5 A Falta de Um Modelo de Resolução de Pontos de Vista

Apesar de advocarem o uso de pontos de vista, nem CORE nem SADT têm um modêlo estruturado que possa realmente tirar proveito desse enfoque. A falta de uma representação própria torna difícil a existência de procedimentos para comparação e análise de visões oriundas de diferentes pontos de vista.

Em banco de dados, o problema de integração de visões tem alguma semelhança com o problema de resolução de pontos de vista. Integração de visões [Batini 86] produz uma descrição conceitual global do banco de dados proposto. Esta descrição global é perseguida com o objetivo de eliminar duplicações, evitar problemas de atualizações múltiplas e minimizar inconsistências entre aplicações. Como integração de visões é feita depois ou durante o projeto conceitual, a representação usada é o esquema conceitual. A entrada para este processo de integração é feita por diferentes esquemas, e a saída é o esquema integrado.[1]

No caso de banco de dados, a integração parte de um modelo já existente, isto é, o schema conceitual. No nosso caso, no processo da validação de fatos, estamos tentando validar os fatos que deveriam estar presentes no modelo. Portanto, torna-se necessário que exista uma representação que possibilite a formalização de diferentes visões. Estas

[1]Batini e outros escrevem: "The form in which the inputs and outputs exist in an integration system (which may be partly automated) is not stated explicitly by any of the authors considered."

visões serão usadas para validar os fatos de tal forma, que no fim do processo de elicitação, o modelo possa "melhor" refletir o universo de informações.

Validação é um processo que é intrinsicamente ligado ao processo de construir um modelo. Resolução de pontos de vista pode ser usada como um meio de validação, de forma que só se modele fatos já resolvidos anteriormente. Convém ressaltar que a representação usada no processo de validação por pontos de vista, **não** pretende ser a mesma usada para o modelo final produzido pela análise de requisitos.

6 Resolução de Pontos de Vista e seus Problemas

Em seguida, detalharemos quais os problemas que foram atacados em nossa pesquisa no que se refere a identificação e classificação de pontos de vista. Na conclusão deste artigo descrevemos os resultados de experimentação obtidos por nossa pesquisa.

Nosso trabalho não lida com os problemas de comunicção na resolução de pontos de vista. Nossa pesquisa se dedicou aos problemas de identificação de discrepâncias e a classificação dessas discrepâncias. Identificando e classisficando discrepâncias entre diferentes pontos de vista estamos criando uma **agenda** que possa orientar a avaliação e a integração desses pontos de vista.

6.1 Problemas Abordados

Para que o modelo citado (Figura 2) seja usado, é indispensável que **haja** uma estruturação para pontos de vista. Este fato caracteriza o primeiro problema:

Como formalizar pontos de vista?

Na Figura 2, retângulo 1, é claramente dependente do formalismo proposto e levanta a seguinte questão.

Como formalmente comparar pontos de vista?

O retângulo 2 da citada Figura coloca em questão o seguinte problema.

O quanto se pode diferenciar entre problemas de conflito e falta de fatos entre pontos de vista, e como classificar os tipos de diferenças entre estes pontos de vista?

O principal problema abordado por nossa pesquisa é o seguinte:

> No processo de análise de requisitos, o problema reside em como diferenciar entre informações incorretas e falta de informações, sem contar com um domínio já previamente codificado.

Para que este problema possa ser estudado é necessário que as primeiras duas perguntas sejam respondidas. A próxima seção dá uma explicação global sobre a representação proposta bem como sua utilização no contexto da validação de fatos.

7 A Estratégia Proposta

A estratégia proposta é composta de um método e uma linguagem, VWPl, de representação de pontos de vista. O método tem procedimentos para a formalização de pontos de vista, bem como procedimentos para a análise desse formalismo. A linguagem é o meio que codifica o formalismo e torna possível sua análise.

A linguagem é derivada de PRISM [Langley 86], uma arquitetura de sistemas de regras. Portanto, nossa estratégia é basicamente um processo que compara duas bases de regras, cada uma representando um ponto de vista diferente.

Conforme observado na Figura 2, o processo de validação de fatos é dependente do processo da coleta de fatos. Supõe-se que os fatos (palavras-chave) estão à nossa disposição antes da aplicação da resolução de pontos de vista. Maior detalhe sobre coleta de fatos, bem como a codificação de pontos de vista estão descritos em [Leite 87c] e [Leite 87m].

Em seguida, apresentamos uma sucinta descrição do método para a produção de pontos de vista, uma descrição da linguagem VWPl e uma descrição dos procedimentos que analizam diferentes pontos de vista.

7.1 Método

Na Figura 3 há um descrição geral da estratégia. José e Maria, ambos engenheiros de software, modelam as intenções dos usuários. Ambos usam VWPl para expressar o universo de informações. Eles usam diferentes perspectivas (processo, dado e ator) e diferentes hierarquias (partes-de é-uma) com o objetivo de criar sua própria visão. Uma vez de posse de críticas, cada analista resolve os conflitos e integra sua percepção final em uma visão. Esta visão é expressa usando a perspectiva processo em conjunto com as hierarquias. Depois disso, os pontos de vista de José e Maria são comparados e analisados.

Portanto, de modo a identificar e classificar discrepâncias entre pontos de vista diferentes, visões devem ser extraídas de cada ponto de vista. Visões são produzidas por um processo chamado construção de visões. A construção de uma visão é baseada no seguinte:

- a disponibilidade de um método de coleta de fatos,
- uma pessoa com um ponto de vista desempenhando um papel no universo de informações, usa perspectivas diferentes e hierarquias para modelar seu ponto de vista,
- perspectivas e hierarquias são analisadas por um analisador estático, e
- uma visão é um modelo integrando as diferentes perspectivas e hierarquias derivadas de um mesmo ponto de vista.

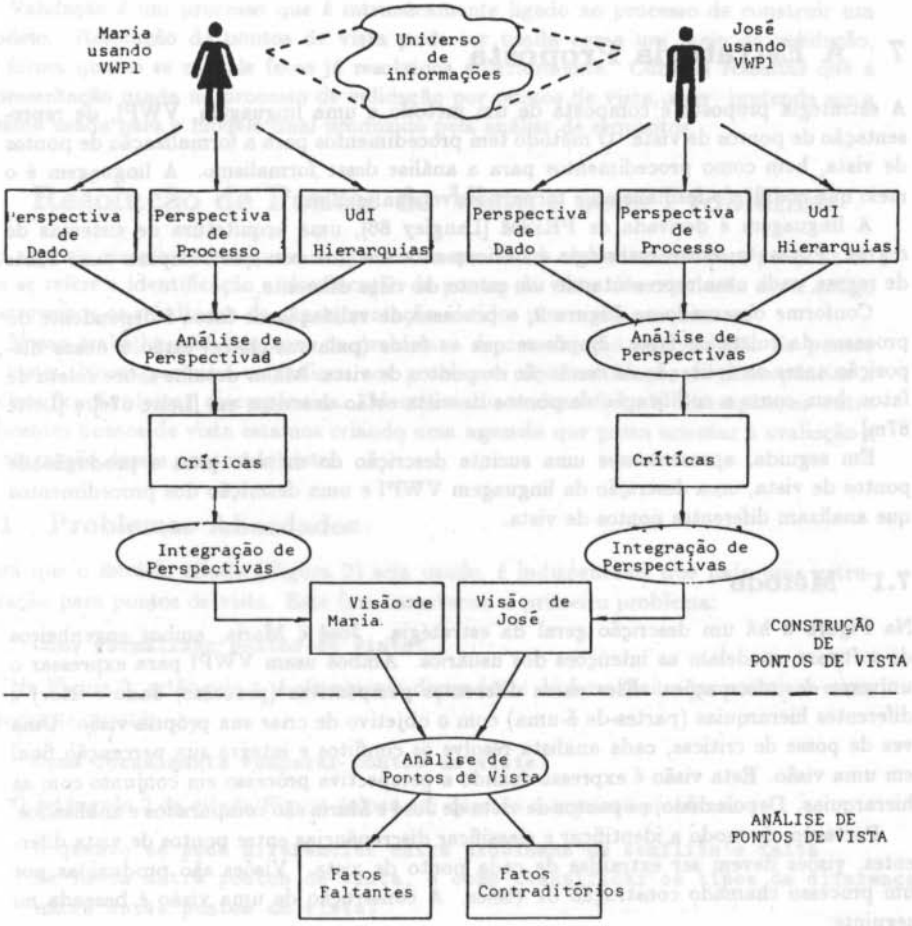

Figura 3: A Estratégia Proposta

Com a disponibilidade de duas visões, torna-se possível comparar diferentes pontos de vista.

Conforme observado antes, é de conhecimento geral que engenheiros de software, quando modelando o universo de informações, o fazem usando diferentes perspectivas. Um exemplo disso é o uso de modelos de dados e modelos de processos. Nossa pesquisa, além desses usuais modelos põe em evidência os atores envolvidos no universo de informações [Leite 87b]. A idéia de se explicitar os atores é de usar a perspectiva daqueles que são responsávies pelos processos, isto é, agentes humanos e agentes físicos. O objetivo do uso de hierarquias [Tanimoto 87] é o de tentar imbutir alguma descrição semântica na informação codificada em VWPl. As hierarquias são compostas de relações de especialização entre palavras chave e de relações de decomposição entre palavras chave.

Para construir uma visão, o engenheiro de software descreve o problema usando as três perspectivas e as duas hierarquias.

As perspectivas e hierarquias são comparadas e são produzidas, uma "lista de discrepâncias" e os "tipos de discrepâncias". Uma visão é a integração de perspectivas e de hierarquias conforme o ponto de vista de uma pessoa e com o auxílio de uma "agenda", produzida pela análise de perspectivas.

A construção de uma perspectiva é um processo no qual se assume o uso do conceito de "vocabulário da aplicação", de tal forma que as palavras chave do universo de informações são utilizadas na representação de pontos de vista.

A idéia principal é a de que o engenheiro de software primeiro analisa o problema e anota suas observações usando a perspectiva de ator em VWPl. As observações descritas usando as outras perspectivas são feitas independentemente e em tempos diferentes. Do mesmo modo, as observações relativas às hierarquias são codificadas.

Supõem-se que a comparação dessas perspectivas e hierarquias produzidas por um mesmo engenheiro de software possibilitará que este produza uma perspectiva (processo) e hierarquias finais de melhor qualidade. Esta perspectiva e hierarquias são então consideradas a representação do ponto de vista do referido engenheiro de software.

São as seguintes as diretrizes para a aquisição de perspectivas e hierarquias.

- A produção das hierarquias "é-uma" e "partes-de" constantes do universo de informações.
- Para cada perspectiva:
  - achar os fatos;
  - expressar os fatos usando as palavras chave do domínio de aplicação;
  - classificar os fatos em: fatos objeto, fatos ação e fatos agente e
  - definir funcionalmente os fatos usando o formalismo de produções.
- Objetos representam tanto os objetos como os estados desses objetos.

- Não há imposições para a produção de hierarquias; o detentor do ponto de vista é quem decide o que deve ou não estar presente nas hierarquias.
- O intuito é o de deixar estas linhas gerais um pouco frouxas, de modo a facilitar ao engenheiro de software, a expressão de seu ponto de vista.

É importante ressaltar o meta uso da estratégia de resolução de pontos de vista. Ele é aplicado na própria construção de um ponto de vista. Isto é, a estratégia para validação de pontos de vista é usada para checar os modelos de perspectiva (dado, processo, ator).

Para que o método alinhavado acima possa ser usado, é necessário que haja uma representação para este tipo de informação. A seguir, lidaremos com tal representação.

7.2 A Linguagem de Pontos de Vista

VWPl é uma linguagem derivada de PRISM [Langley 86], uma arquitetura para sistemas de produção. Não sendo uma linguagem de requisitos, sua utilidade é restrita ao processo de validação de fatos.

A idéia básica atrás de VWPl, é a de ter uma estrutura prè-definida para a construção de regras. A imposição de maior restrição no schema usual de lado direito (RHS) e lado esquerdo (LHS) de uma regra, torna possível que tenhamos alguma informação de ordem semântica na estrutura das regras. O enfoque usado em VWPl é semelhante aos usados em *case grammars* [Rich 83], onde as estruturas produzidas pelas regras da gramática correspondem a relações semânticas, ao invés de somente relações sintáticas.

No caso de VWPl as relações semânticas são expressas pelo uso do que chamamos restrições de *tipos* e de *classes*. Tipos são os diferentes fatos usados, isto é, fatos **objeto**, fatos **ação** e fatos **agente**. Classes são os diferentes papéis que cada fato pode ter numa regra. Numa regra um fato pode:

- ser retirado da memória de trabalho (nós chamamos a isto de *classe de entrada*),
- ser adicionado à memória de trabalho (nós chamamos a isto de *classe de saída*), ou
- permanecer na memória de trabalho (isto é chamado de *classe invariante*).

Um fato é uma relação entre palavras chave. As palavras chave, para expressar fatos em VWPl, são checadas contra uma lista de tipos antes de serem analisadas (parsed). Em outras palavras, um teste de pertinência semântica é efetuado nas palavras chave disponíveis para a descrição de uma visão.[2] Um fato é composto de uma palavra-chave-fato e um atributo-fato. Um exemplo é "(livro =ident-livro =autor =titulo)", onde `livro` é a palavra-chave-fato, e *ident-livro, autor e titulo* são os atributos-fato.

Para cada perspectiva há uma combinação especial de tipos e de classes. Neste artigo somente usamos a perspectiva de dado e de agente. Para a perspectiva de dado usamos a seguinte estrutura de regras.

[2] A gramática completa de VWPl pode ser encontrada em [Leite 88].

- LHS (lado esquerdo) – a *entrada* é um **objeto**, e a *invariante* pode ser um **agente** ou um **objeto**.
- RHS (lado direito) – a *saída* é uma **ação**.

Para a perspectiva de ator usamos a seguinte estrutura.

- LHS (lado esquerdo) – a *entrada* é um **agente**, e a *invariante* pode ser um **agente** ou um **objeto**.
- RHS (lado direito) – a *saída* é uma **ação**.

Hierarquias são codificadas como listas. Estas por sua vez são organizadas pelo tipo de hierarquia (é uma, ou partes-de). Para cada tipo a raiz da hierarquia é a cabeça de uma lista seguida dos ramos daquela hierarquia.

Na Figura 5 damos um exemplo do uso da linguagem VWPl. Este exemplo mostra como uma simples sentença foi codificada de acordo com a perspectiva usada.[3]

7.3 O Analisador Estático

A análise de diferentes perspectivas e de diferentes visões é efetuada por uma série de procedimentos, que fazem uma análise de dois conjuntos de perspectivas ou visões. Conforme já observado esta análise é realmente uma análise entre dois conjuntos de regras.

Comparar dois grupos de regras só faz sentido quando há similaridade entre eles. No nosso caso, existe uma série de fatores que nos levam a esta similaridade; abaixo listamos os mais importantes.

- Os detentores de pontos de vista estão se baseando no mesmo universo de informações.
- O uso de um método em que o uso do conceito de "vocabulário de aplicação" norteia a coleta de fatos.
- O uso de uma linguagem especial que restrige como as regras são expressas.

O analisador estático proposto e implementado em **Scheme** tem duas importantes tarefas: achar quais as regras quem tem similaridade entre si, e uma vez que regras dos diferentes conjunto de regras são identificadas como similares, identificar e classificar as discrepâncias existentes entre elas. Regras que não encontram similares são classificadas como falta de informação. Apesar do analisador ser centrado em

[3]Este exemplo "de brinquedo" tem somente o objetivo de dar uma idéia da linguagem VWPl. Em nosso estudo de casos [Leite 88], exemplos com até 20 regras para cada perspectiva ou ponto de vista foram utilizados.

```
        "Um oficial do navio tem, como um de seus deveres,
         a tarefa de limpar o deck do navio, e ele
         pode usar diferentes tipos de vassouras"

REGRAS:
                    Perspectiva de Ator

    (10 ((oficial =patente =nome) (iate =numero-de-registro))
     -->
     (($delete-from wm (oficial =patente =nome))
      ($add-to wm (limpar-o-deck =numero-de-registro))))

; O agente ''oficial'' (entrada) executa a ação ''limpar-o-deck'' (saída)
; no objeto ''iate'' (invariante).

                    Perspectiva de Dado

    (20 ((oficial =nome) (barco =numero-de-registro)
         (vassoura =tipo))
     -->
     (($delete-from wm (vassoura =tipo))
      ($add-to wm (limpar-o-deck-com-vassoura =tipo =numero-de-registro))))

; O objeto ''vassoura'' (entrada) é usado pelo agente ''oficial''
; (invariante) para  executar a ação ''limpar-o-deck-com-vassoura'' (saída)
; no objeto ''barco'' (invariante)

                    Hierarquias
    (is-a (2 (navio iate barco)))
    (parts-of (2 (navio deck)))
```

Figura 4: Exemplo

uma análise sintática há o uso de descritores de semântica tais como: hierarquias e "case grammars".

Sendo baseado na representação sintática de termos, o analisador utiliza-se de "pattern matchers" e "partial matchers". Estes procedimentos de "matching", que são aplicados entre fatos de dois conjuntos de regras diferentes, têm diferentes algorítmos de score dependendo da informação semântica disponível sobre tipos e classes em cada fato.

A classificação de discrepâncias, isto é, determinar quais as discrepâncias relativas a falta de informação e quais as discrepâncias relativas a informações contraditórias, é feita baseada em scores e nas hierarquias. É óbvio que o analisador estático não pode afirmar nada sobre duas regras que não têm discrepâncias, mas na realidade não estão corretas com respeito ao universo de informações.

Se tomarmos o exemplo dado anteriormente, o analisador estático nos fornecerá as seguintes mensagens:

```
Fatos Contraditórios:

  (20 (1020 (((30 limpar-o-deck =numero-de-registro)
              30 limpar-o-deck-com-vassoura-
                =tipo =numero-de-registro . 0.5))))

mensagem 20: "as regras 10 e 20 têm
         diferentes fatos para representar
         uma ação considerada similar"

   (1 (1020 1 ((oficial) oficial) (=patente =nome) =nome)))

mensagem 1: "as regras 10 e 20 têm atributos diferentes para o mesmo
             agente ''oficial''"

       (4 (1020 (((22 iate =numero-de-registro) 22 barco
                                    =numero-de-registro) . 0.66)))

mensagem 4:  "de acordo com a hierarquia
         é-uma os respectivos objetos
         estao em contradição"

Fatos Faltantes:

       (16 (1020((vassoura) .21)))

mensagem 16:  "o objeto ''vassoura''
         esta faltando na regra 10"
```

8 Conclusão

Nossa tese [Leite 88] demonstrou a possibilidade do uso de um formalismo para pontos de vista, bem como de procedimentos automáticos para a análise de diferentes pontos de vista. Esta demonstração não só se realizou pela apresentação de uma linguagem e de um analisador de pontos de vista, como pelo estudo de casos com quatro pessoas distintas que comprovaram a nossa tese de que a resolução de pontos de vista pode ser usada na validação de requisitos.

Mais recentemente, [Leite 89], demonstramos a aplicação da análise de pontos de vista ao problema da biblioteca, conforme proposto no "International Workshop of Software Specification and Design". Neste trabalho comparamos, automaticamente, diferentes visões de vários autores sobre o problema da biblioteca e obtivemos resultados bastante semelhantes aos observados por Wing [Wing 88] que usou uma comparação manual. Este estudo de caso mais uma vez demonstra a potencialidade do uso de pontos de vista na elicitação de requisitos.

Lembrando que no processo de desenvolvimento de software, quão mais tardiamente são os erros descobertos, maior será o custo para corrigí-los, ressalta-se a importância de detectar estes erros o mais cedo possível. A validação de requisitos é justamente o processo que cuida de apontar erros na fase inicial de entendimento e definição do software. Neste contexto, ressaltamos que as experiências de nossa tese e o trabalho realizado para a quinta IWSSD demonstram, como já referido, a eficácia do uso da análise de pontos de vista na validação de requisitos.

Futuros trabalhos nesta área devem aprofundar os experimentos empíricos iniciados na tese e melhorar a interface homem-programa conforme presente atualmente em VWPl. Atualmente, estamos atacando duas frentes: o desenvolvimento de um *prettyprinter* para os resultados do analisador estático e o uso do analisador de pontos de vista na validação da elicitação de linguagens da aplicação [Leite 89b].

Referências

[Agresti 87] Agresti, W.; *In New Paradigms for Software Development*, W. Agresti, Ed., IEEE Computer Society, Long Beach, CA 1987

[Arango 89] Arango, G.; Domain Analysis: From Art Form to Engineering Discipline. *In 5th International Workshop on Software Specification and Design* (Pittsburgh, PA, 1989), IEEE Computer Society Press, pp. 152-159.

[Batini 86] Batini, C., Lenzerini, M., and Navathe, S.; A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys 18.4* (Dec.1986), 323-364.

[Fickas 87] Fickas, S.; Automating the Analysis Process: An Example. *In 4th International Workshop on Software Specification and Design* (Monterey, CA, 1987), IEEE Computer Society Press, pp. 58-67

[Langley 86] S. Ohlsson and P. Langley; *PRISM Tutorial and Manual.* Tech. Rep. 86-02, University of California, Irvine - Dept. of Computer Science, Feb. 1986.

[Leite 87b] Leite, J.; *The Agent Viewpoint.* Tech. Rep. RTP 070, Dept. of Comp. Science, Univ. of Calif., Irvine, Mar. 1987.

[Leite 87c] Leite, J.; *A Proposal for Applied Research on Requirements Elicitation.* Tech. Rep. RTP 072, Dept. of Comp. Science, Univ of Calif., Irvine, Jul. 1987.

[Leite 87m] Leite, J.; *VWPI Manual.* Tech. Rep. RTP 082, Dept. of Comp. Science, Univ of Calif., Irvine, Oct. 1987.

[Leite 88] Leite, J.; *Viewpoint Resolution in Requirements Elicitation.* PHD thesis, Dept. of Comp. Science, Univ. of Calif., Irvine, 1988.

[Leite 89] Leite, J.C.S.P.; Viewpoint Analysis: A Case Study. *In 5th International Workshop on Software Specification and Design* (Pittsburgh, PA, 1989), IEEE Computer Society Press, pp. 111-119.

[Leite 89b] Leite, J.C.S.P.; Produção de Linguages da Aplicação, Primeiros Resultados; *Departamento de Informática PUC/RJ*, Rio de Janeiro – Brasil, 1989.

[Mullery 79] Mullery, G. CORE - A Method for Controlled Requirement Specification. In *Proc. 4th Int. Conf. on Softw. Eng.* (1979), IEEE Computer Society Press, pp. 126-135.

[Rich 83] Rich E.; *Artificial Intelligence.* Mc Graw-Hill, New York, 1983.

[Rich 87] Rich, C., Waters, R., and Reubenstein, H.; Toward a Requirements Apprentice. *In 4th International Workshop on Software Specification and Design* (Monterey, CA, 1987), IEEE Computer Society Press, pp. 79-86

[Ross 77] Ross, D. Structured Analysis (SA): A Language for Communicating Ideas. In *Tutorial on Design Techniques*, Freeman and Wasserman, Eds., IEEE Computer Society Press, Long Beach, CA 1980, pp. 107-125.

[Tanimoto 87] Tanimoto, S.; *The Elements of Artificial Intelligence.* Computer Science Press, Potomac, Maryland, 1987.

[Wing 88] Wing, J.; A Study of 12 Specification of the Library Problem. *in IEEE Software 5, 4* (Jul 1988), 66-76.