O USO DE UMA LINGUAGEM SEMI-FORMAL NO PROCESSO DE FORMALIZAÇÃO DE ESPECIFICAÇÕES DE SOFTWARE

STANLEY LOH
JOSÉ MAURO VOLKMER DE CASTILHO

Universidade Federal do Rio Grande do Sul
Instituto de Informática
Curso de Pós-Graduação em Ciência da Computação
Av. Osvaldo Aranha, 99
CEP 90210 Porto Alegre - RS - Brasil
Fones: (0512) 28-1633 R.3115
Telex: (051) 2680 Caixa Postal: 1501
FAX: (0512) 24-4164
E-Mail: CASTILHO@SBU.UFRGS.ANRS.BR

Apoio: FINEP, RHAE/CNPq

RESUMO

Neste artigo é discutido o uso de uma linguagem semi-formal, semi-natural, a ser utilizada no processo de tradução de especificações informais para formais de sistemas de software.

O objetivo desta linguagem é permitir a divisão do processo de formalização de especificações (transformação entre especificações informais e formais), facilitando o processo e permitindo um maior controle da qualidade de seu resultado.

Para as transformações entre as linguagens (informal, semi-formal e formal), foram identificadas regras heurísticas que facilitam esta tarefa e é recomendado o uso de um dicionário de termos-chave para auxílio na identificação de partes da especificação que descrevem a mesma informação.

ABSTRACT

The use of a semi-formal specification language in the process of formal specification of software systems is discussed.

The use of such a language divides the process of formalization in two steps, which could be performed with less effort, achieving better quality final results.

A set of heuristic rules is proposed, to guide the two translation steps (from informal to semi-formal specification, and from semi-formal to formal specification).

A dictionary of terms that appear in the specification plays an important role in the process, helping in the identification of "synonymous" parts of the specification.

1. INTRODUÇÃO

No desenvolvimento de sistemas de software, a etapa de Análise e Especificação de Requisitos do Sistema é uma das mais importantes, pela influência que o seu produto tem sobre o resultado final do desenvolvimento.

Intuitivamente, depois de construída a especificação dos requisitos, o produto das outras etapas é, de certa forma, uma conseqüência dela. Se a especificação for formal, tem-se as vantagens adicionais de: entendimento não-ambíguo do que está especificado; possibilidade de provar formalmente propriedades da especificação; possibilidade de validação experimental do sistema especificado, no caso de especificações executáveis (a especificação funciona como protótipo do sistema).

Entende-se por Especificação Formal aquela que faz uso de uma Linguagem Formal ou Formalismo (conforme [KLE67], **formalismo** é um sistema ou linguagem que possui um conjunto de regras de formação bem-definidas e um conjunto de regras de derivação que permitem desconsiderar o significado do que está descrito no sistema e trabalhar apenas com a sua forma, com o intuito de verificar novos significados ou propriedades).

Apesar de ser preferível utilizar uma linguagem formal a uma informal (**linguagem informal** é toda aquela que não é formal), durante a especificação de software, alguns autores ([TEO82], [BAL78] e [BAL85]) defendem a idéia de que devem ser utilizados os dois tipos de linguagens, isto é, a especificação deve ser feita nas duas linguagens. Nesta estratégia, cada linguagem desempenha o seu papel, não devendo ser substituído ou negligenciado. Assim, as linguagens informais serviriam para a coleta inicial dos dados e para a validação pelo Usuário das informações documentadas, enquanto que as linguagens formais seriam úteis para a definição precisa do software e para comunicação entre o pessoal do desenvolvimento.

Construída a Especificação Formal do sistema, há um conjunto de regras e procedimentos, automáticos ou manuais, para se completar o desenvolvimento do sistema.

Entretanto, não há regras ou procedimentos precisos, bem definidos, para realizar a construção das especificações formais. Esta atividade tem muito de arte, e seu produto depende da experiência e sensibilidade de seus executores.

Este artigo apresenta e discute uma proposta de procedimento de construção de especificações formais para sistemas de software, utilizando uma linguagem semi-natural e semi-formal (apresenta-se a linguagem como **semi-natural** por ser um sub-conjunto da Linguagem Natural e como **semi-formal** por ser mais precisa que a Linguagem Informal e por não apresentar o grau de formalismo exigido para as linguagens formais).

Esta linguagem pretende ser intermediária na transformação de especificações informais para formais. Para auxiliar nas

transformações, foram identificadas regras heurísticas, cuja função será ilustrada através de exemplos (**regras heurísticas** são regras informais de julgamento, que proporcionam uma melhor tomada de decisão sem, entretanto, garantirem a solução do problema, como é o caso dos algoritmos [LOH91]).

A seção 2, a seguir, discute alguns problemas do processo de formalização de sistemas de software e apresenta algumas idéias encontradas na literatura para solucionar tais problemas.

Na seção 3, são apresentadas a função de uma linguagem semi-formal no contexto do Projeto SILOG (em desenvolvimento na Universidade Federal do Rio Grande do Sul) e a proposta para os procedimentos de formalização. Nesta seção, são também apresentadas as linguagens informal e formal utilizadas no projeto.

A linguagem utilizada no projeto em questão é apresentada na seção 4 e, por fim, na seção 5, são dados exemplos de procedimentos de formalização segundo a proposta aqui defendida.

2. PROCESSO DE FORMALIZAÇÃO DE ESPECIFICAÇÕES

Entende-se por um documento de especificação formal aquele que possui as seguintes qualidades [DAV80]:

- precisão: permitindo apenas uma interpretação para as informações descritas;
- consistência: não contendo duas ou mais informações contraditórias;
- objetividade: contendo apenas informações relevantes para o desenvolvimento do software;
- inteireza: contendo todas as informações relevantes.

Muitas técnicas foram propostas, na literatura, para a especificação formal de software ([DAV82], [BAB85], etc). Entretanto, poucos trabalhos tratam do processo de aquisição destas especificações, ou seja, do trabalho de transformar especificações informais para a representaçção usada nestas técnicas ([BAB85]).

Segundo [BAR85], a **formalização** começa com alguma idéia informal sobre o que o software deve fazer e termina com uma descrição formal de o que o software deve fazer, mas não como isto deve ser feito.

Este processo de formalização pode ser considerado como a tarefa mais difícil da fase de especificação de software.

Interpretar corretamente as informações escritas em uma linguagem informal é um problema para a formalização, pois requer o entendimento de informações implícitas (contexto) e a solução de ambigüidades. A dificuldade também se dá por falta

de ferramentas que auxiliem estas atividades, tornando o processo bastante dependente de quem o faz (a análise da especificação informal é feita na mente da pessoa).

Some-se a isto o fato de que as linguagens formais (linguagens-objeto da formalização) são cansativas e de difícil escrita e leitura, exigindo muita atenção a detalhes quando se trabalha com grandes volumes de informação ([BAL83], [BAL85]).

Isto tudo permite, obviamente, o surgimento de erros no documento de especificação.

A fim de solucionar parte destes problemas, [BAL78] sugere que os aspectos informais do desenvolvimento de software sejam tratados com o auxílio de ferramentas automatizadas. Desta forma, tarefas rotineiras e que exigem maior atenção seriam desempenhadas por estas ferramentas, liberando os Analistas de Sistemas para se preocuparem com tarefas de mais alto nível, que exigem raciocínio e criatividade ([BAL83]).

Outra sugestão (em [BAL85]) é que este processo de formalização seja dividido em etapas menores, com o intuito de controlar cada etapa em particular. Assim, ter-se-ia um desenvolvimento formal, com controle e documentação das transformações e demais tarefas. Com esta estratégia, a qualidade do produto fica garantida não pela verificação deste produto mas pela qualidade do processo ([BAL81], [BAL83]).

Uma idéia para dividir o processo de formalização é a de se usar uma linguagem intermediária entre as linguagens informais e formais, a fim de aproximá-las (sem, entretanto, substituí-las), simplificando assim o trabalho de transformação entre estas linguagens ([BAL85], [DAV82]).

Esta linguagem permitiria a elaboração de especificações semi-formais, isto é, menos formais que as especificações formais (e também menos informais ou naturais que as especificações informais).

A diferença fundamental entre as especificações informais e formais é que as primeiras são mais concisas, fazendo uso de contexto (informações implícitas), enquanto que as últimas explicitam e detalham mais as informações ([BAL78]).

A função da linguagem semi-formal é a de explicitar mais informações que as linguagens informais, eliminando ambigüidades e dando maior precisão à especificação, sem, entretanto, usar tantos detalhes como nas linguagens formais.

Desta forma, à medida que se progride pelos diferentes níveis de linguagens (da mais informal para a mais formal), a descrição do software vai sendo alterada, a fim de se melhorar a sua qualidade (precisão, objetividade, etc). Assim, são descartadas algumas informações (aquelas irrelevantes para o processo de desenvolvimento do software), outras são acrescidas (aquelas implícitas no contexto) e outras são mantidas ou

somente alteradas na sua forma de representação.

O uso de uma linguagem semi-formal (intermediária) permite um maior controle das transformações entre as linguagens informais e formais, acrescentando-se assim qualidade ao processo de formalização e, ao mesmo tempo, facilitando-o.

3. O USO DE UMA LINGUAGEM SEMI-FORMAL NO PROJETO SILOG/UFRGS

No projeto SILOG/UFRGS (Ferramentas para Desenvolvimento Automatizado de Sistemas de Informação Especificados em Lógica), foi definida e está em teste uma linguagem semi-formal e semi-natural, intermediária às duas linguagens usadas para especificação de software (uma informal e outra formal).

Esta linguagem semi-formal pretende facilitar a identificação dos elementos (conceitos) que aparecem na especificação informal (linguagem informal) e que serão utilizados na especificação formal (linguagem formal), diminuindo assim as ambigüidades e imprecisões da linguagem informal durante a interpretação dos fatos a serem formalizados.

Para tanto, faz-se uso de estruturas simples (frases simples e padronizadas) na linguagem semi-formal, a fim de tornar mais explícitas as informações e de eliminar certas informações consideradas irrelevantes.

Também, ao se definir a linguagem semi-formal, procurou-se uma proximidade com a linguagem formal utilizada, com a pretensão de facilitar as transformações daquela para esta.

Neste projeto, também foram identificadas regras heurísticas para as transformações entre as linguagens (informal, semi-formal e formal). Estas heurísticas são regras informais para auxílio à atividade de formalização (transformação entre as linguagens) e, como tal, não asseguram a solução do problema. Contudo, orientam a atividade, numa tentativa de controlá-la.

Juntamente com as heurísticas, deverá ser utilizado um Dicionário de Termos, cuja função é permitir a identificação de partes da especificação que tratam de uma mesma informação, mas em notação diferente.

Neste dicionário, deverão ser definidos os termos sinônimos (ou expressões sinônimas), os verbos e sua forma substantivada, as formas ativa e passiva dos verbos (ou sinônimos na falta de uma das formas) e os adjetivos e as orações adjetivas correspondentes.

Exemplos:
alunos = corpo discente;
consultar = consulta;
consultar = ser consultado;

devedor = que tem débito.

No projeto, a linguagem informal é representada pela Linguagem Natural em sua forma escrita. Supõe-se que esta seja uma maneira fácil de introduzir informações no processo de desenvolvimento de software.

A linguagem formal utilizada é baseada em lógica, sendo que numa especificação formal, o software é descrito em 3 partes: parte de Dados e Relações (estrutura), parte de Operações (manipulação dos dados) e parte de Restrições de Integridade (condições a serem respeitadas durante a utilização do software).

Na primeira parte, os dados e suas relações são representados por símbolos predicativos ou funcionais com ou sem argumentos (representados por variáveis).

Exemplos:
usuário(x), material_bibliográfico(y), retirou(x,y).

Já na parte de operações, cada função do software (de manipulação ou de consulta ou mista) é descrita através de suas pré e pós-condições, sendo que as pré-condições devem ser verdadeiras para a execução bem sucedida da operação e as pós-condições caracterizam as alterações feitas nos dados pela operação correspondente (ou por sua execução).

Exemplos:
Empréstimo(x,y):
Pré-condições: usuário(x) e material_bibliográfico(y).
Pós-condições: retirou(x,y).

Por fim, as restrições são descritas conforme o seguinte padrão:

fatos_1 --> fatos_2,

sendo que a restrição será violada sempre que "fatos_1" for verdadeiro e "fatos_2" não.

Exemplos:
retirou(x,y) e periódico(y) --> professor(x);
retirou(x,y) e material_bibliográfico(y) --> usuário(x).

Além disso, deverá haver coerência entre a parte de dados e a parte de operações, já que a segunda utiliza os símbolos da primeira (idem para a parte de dados e a parte de restrições).

4. DEFINIÇÃO DA LINGUAGEM SEMI-FORMAL UTILIZADA

A especificação do software na Linguagem Semi-Formal é dividida em três partes, como a divisão feita na Linguagem Formal (partes de Dados, Operações e Restrições).

4.1 Parte de Dados

As informações são descritas na forma de orações simples (um verbo só) do tipo padrão:

SUJEITO VERBO COMPLEMENTO_VERBAL,

sendo que SUJEITO e COMPLEMENTO_VERBAL podem ser substantivos ligados por preposição e ainda cada substantivo pode vir qualificado por adjetivos.

Exemplos:
usuários retiram materiais bibliográficos;
professores retiram periódicos;
usuários têm nomes;
materiais bibliográficos têm títulos.

Em alguns casos, poderá ser usada a forma:
SUJEITO VERBO_INTRANSITIVO ADJUNTO_ADVERBIAL.

Exemplos:
empréstimos são feitos em datas;

Deverá ser dada atenção especial a pronomes (devem ser substituídos pelos nomes correspondentes), nomes referentes à Organização (as frases deverão ser reescritas) e informações irrelevantes (devem ser desconsideradas).

4.2 Parte de Operações

Cada operação deverá ter um nome e um conjunto de ações que descreve o que deve ser feito, sendo que os formatos devem ser como a seguir:

Nome = VERBO COMPLEMENTO_VERBAL (ou "SUBSTANTIVO + PREPOSIÇÃO" no lugar do VERBO);

Ação = VERBO COMPLEMENTO_VERBAL.

As ações deverão ser de um dos 5 tipos a seguir e deverão seguir esta mesma ordem:
- solicitação ou recebimento de informação;
- verificação de uma condição (neste caso, a palavra "se" poderá ser usada após o verbo);
- cálculo de uma informação;
- modificação do conjunto de informações;
- fornecimento de informação.

Eventualmente, poderão ser utilizadas as expressões "para cada" e "de cada", denotando iterações a serem feitas. Porém, elas serão desconsideradas quando da transformação para a Linguagem Formal, pois, nesta última, não há referência explícita a iterações.

Exemplos:
Contratar funcionário:

Recebe nome e endereço do funcionário
Verifica se o funcionário não está cadastrado
Cadastra o funcionário.

Marcar consulta:
Recebe nome do paciente e nome do médico
Recebe horário da consulta
Verifica se paciente está cadastrado
Verifica se médico existe
Verifica se horário da consulta coincide com algum horário que o médico tem disponível
Anota consulta.

Cálculo da idade do paciente
Recebe nome do paciente
Calcula idade, que é igual ao ano atual menos o ano de nascimento do paciente
Informa a idade do paciente.

4.3 Parte de Restrições

As restrições de integridade deverão ser descritas segundo as formas-padrão identificadas em [LOH91] (em número de 15, aproximadamente).

A seguir, são apresentadas algumas destas formas-padrão (os termos em letras minúsculas serão substituídos por valores adequados):

TODO sujeito DEVE verbo complemento_verbal;
SOMENTE sujeito PODE verbo complemento_verbal;
sujeito QUE verbo_1 complemento_verbal_1 NÃO PODE verbo_2 complemento_verbal_2;
sujeito SÓ PODE verbo complemento_verbal_1 SE é complemento_verbal_2.

Exemplos:
Toda pessoa deve ter 1 nome, no máximo.
Somente professores podem retirar periódicos.
Usuários que têm débito não podem retirar material.
Usuário só pode retirar periódico se é professor.

5. PROCEDIMENTOS DE FORMALIZAÇÃO

Para os procedimentos de transformação entre as linguagens (informal, semi-formal e formal), deverão ser utilizadas as heurísticas identificadas e definidas. Ainda que não permitam transformações automáticas, estas regras heurísticas servem para orientar e facilitar aquelas tarefas.

Há dois conjuntos de regras heurísticas, conforme a finalidade (e ainda cada conjunto apresenta heurísticas para as partes de Dados, Operações e Restrições):
(i) heurísticas para transformação da linguagem informal

para a linguagem semi-formal;
(ii) heurísticas para transformação da linguagem semi-formal para a linguagem formal.

A seguir são apresentados alguns exemplos de transformações e as heurísticas utilizadas em cada caso (os exemplos se referem a parte de um sistema para uma biblioteca, e é apresentada apenas uma parte das heurísticas identificadas em [LOH91]).

5.1 Transformação Linguagem Informal para Linguagem Semi-Formal

Parte de Dados

Linguagem Informal:

Os usuários retiram da Biblioteca, por empréstimo, materiais. Os usuários devem fornecer seu nome para o cadastramento na Biblioteca.

Linguagem Semi-formal:

A) usuários retiram materiais
B) usuários têm nomes

Heurísticas usadas:

A) - cada período da especificação na Linguagem Informal deve ser analisado em separado;
- devem ser analisados os verbos;
- só são permitidos verbos bitransitivos;
- nomes referentes à Organização devem ser desconsiderados;
- podem ser utilizados sinônimos ou verbos na voz passiva adequar as informações aos formatos; no caso, "retirar" é sinônimo de "retirar por empréstimo".

B) - pronomes devem ser substituídos pelos nomes correspon-dentes; no caso, "seu nome" é trocado por "nome do usuário";
- as preposições devem ser analisadas, bem como a relação que indicam; no caso, a preposição "do" está indicando que "usuário tem nome".

Parte de Operações

Linguagem Informal:

Na função de empréstimo de materiais da Biblioteca, o funcionário responsável solicita ao usuário seu nome e o título do material que deseja. Após, o funcionário verifica se o usuário está cadastrado na Biblioteca e se o material está disponível. Se positivo, o funcionário registra em uma ficha que o usuário retirou o material.

Linguagem Semi-formal:

Cadastrar usuário:
Recebe nome do usuário
Recebe título do material
Verifica se usuário está cadastrado
Verifica se material está disponível
Registra que o usuário retirou o material.

Heurísticas usadas:

- cada verbo deve ser analisado, pois pode representar uma ação;
- cada ação deve respeitar um dos 5 tipos estabelecidos nesta linguagem;
- a ordem das ações deve ser observada;
- devem ser desconsideradas informações sobre meios físicos de armazenamento e sobre quem faz a ação;
- informações redundantes devem ser eliminadas; na expressão "material que deseja", "que deseja" é redundante e deve ser desconsiderado.

Parte de Restrições

Linguagem Informal:

Quando do empréstimo de periódicos, deve ser verificado se o usuário é professor. Usuários só podem retirar materiais disponíveis.

Linguagem Semi-formal:

A) Usuário só pode pode retirar periódico se é professor.
B) Somente materiais disponíveis podem ser retirados por usuários.

Heurísticas usadas:

A)
- atentar para expressões do tipo "deve", "não pode", "somente", "se", "nunca", "sempre", as quais podem identificar uma restrição;
- tentar encaixar a restrição em um dos formatos pré-definidos.

B)
- idem anterior;
- lembrar que "ser retirado por" é a forma passiva de "retirar".

5.2 Transformação Linguagem Semi-Formal para Linguagem Formal

Parte de Dados

Linguagem Semi-formal:

A) usuários retiram materiais.
B) usuários têm nomes.

Linguagem Formal:

A) usuário(x), retira(x,y), material(y).
B) usuário(x), tem(x,y), nome(y).

Heurísticas usadas:

A) - transformar o formato "SUJEITO VERBO COMPLEMENTO" para "sujeito(x), verbo(x,y), complemento(y)";
   - usar os nomes no singular.
B) - idem anterior.

Parte de Operações

Linguagem Semi-formal:

Cadastrar usuário:
Recebe nome do usuário
Recebe título do material
Verifica se usuário está cadastrado
Verifica se material está disponível
Registra que o usuário retirou o material.

Linguagem Formal:

Cadastrar_usuário(n,t):
∃ u,m
Pré-condições:
nome(n), título(t), usuário(u), tem(u,n),
material(m), tem(m,t), disponível(m).
Pós-condições:
retira(u,m).

Heurísticas usadas:

- analisar cada ação da operação;
- todas as variáveis devem ser definidas por um predicado;
- nomes solicitados (por verbos tipo "recebe", "solicita") devem ser argumentos de entrada da operação;
- a preposição "de" pode indicar uma relação do tipo "tem(x,y)";
- fatos ocorrendo após "verifica se" são pré-condições;
- expressões do tipo "sujeito ESTÁ CADASTRADO" devem ser transformadas para "sujeito(x)" nas pré-condições, onde "x" é argumento de entrada ou está ligado a um argumento de entrada;
- o verbo "ser" pode indicar uma especialização de um conceito; no caso, "disponível" é uma especialização de "material";
- a expressão "REGISTRA QUE fato" indica que o fato deve ser uma pós-condição;

- devem ser eliminadas predicados redundantes, isto é, que aparecem mais de uma vez;
- variáveis que aparecem nas pré-condições sem prévia negação devem ser definidas pelo quantificador existencial.

Parte de Restrições

Linguagem Semi-formal:

1) Usuário só pode pode retirar periódico se é professor.
2) Somente materiais disponíveis podem ser retirados por usuários.

Linguagem Formal:

1) ∀ x,y
usuário(x) e periódico(y) e retira(x,y) --> professor(x)
2) ∀ x,y
usuário(y) e retira(y,x) --> material(x) e disponível(x)

Heurísticas usadas:

1) - o formato "sujeito Só PODE verbo complemento_verbal_1 SE é complemento_verbal_2" é traduzido diretamente para "∀ x,y sujeito(x) e complemento_verbal_1(y) e verbo(x,y) --> complemento_verbal_2(x)").

2) - o formato "SOMENTE sujeito PODE verbo complemento_verbal" é traduzido para "∀ x,y complemento_verbal(y) e verbo(x,y) --> sujeito(x);
- expressões com adjetivos, como no tipo "nome adjetivo" devem ser transformadas para "nome(x) e adjetivo(x)";
- devem ser usados os termos-padrão definidos no Dicionário de Termos; no caso, "retira(x,y)" é o padrão para "é_retirado_por(y,x)".

6. CONCLUSÃO

Neste artigo, foi apresentada uma linguagem semi-formal, desenvolvida no projeto SILOG/UFRGS, a ser utilizada no processo de formalização de especificações de software, como meio intermediário às linguagens informais e formais.

Esta linguagem foi definida a partir de estudos feitos com o processo de formalização. Também foram identificadas regras heurísticas para auxiliar nas transformações entre as linguagens informal, semi-formal e formal.

Com base nos experimentos realizados com esta linguagem e com as heurísticas, notou-se que o processo de formalização é facilitado, uma vez que os elementos a serem utilizados na linguagem formal são mais facilmente identificados.

Notou-se, também, que a linguagem semi-formal elimina certas ambigüidades e imprecisões que ocorrem na linguagem informal (natural). Para tanto, foram úteis a simplicidade das estruturas da linguagem semi-formal e o uso do Dicionário de Termos.

Quanto às heurísticas, acredita-se que elas facilitam o trabalho de quem faz as transformações entre as linguagens (identificando padrões e sugerindo ações).

Com as heurísticas também, tem-se um passo inicial para a automatização de algumas tarefas de formalização e/ou para o desenvolvimento de ferramentas de apoio (ainda que, no momento, seja difícil automatizar todo o processo).

Entretanto, o conjunto de regras heurísticas não está completo. Ainda que úteis e suficientes para os casos testados as heurísticas identificadas não garantem a sua abrangência a todos os casos possíveis.

A linguagem semi-formal também se mostrou útil no processo de validação da especificação por parte do Usuário do software. Por ser semi-natural, ela permite uma maior participação, na fase de Especificação, de pessoas não-familiarizadas com formalismos (linguagens formais).

Apesar de suas vantagens, há ainda certos problemas que deverão ser pesquisados a seguir.

A abordagem proposta aqui depende ainda, em muito, da participação do Analista que faz a formalização.

Também, o sucesso da abordagem depende de que a especificação informal esteja completa.

Quanto ao grande volume de informações geradas, espera-se que a tarefa de analisá-las seja facilitada com o advento de ferramentas de apoio.

É bom salientar aqui que este trabalho considerou apenas parte dos problemas envolvendo a especificação de software. Por isto, não foram tratados aspectos ligados a tempo na especificação, nem se ateve a especificar requisitos de desempenho do sistema e volume de dados.

No momento, estão sendo feitos testes com a linguagem e as heurísticas, a fim de precisar as vantagens desta abordagem (adequação da linguagem, clareza, inteireza e eficiência das heurísticas, etc).

Como trabalho certo a ser realizado a seguir, está o desenvolvimento de ferramentas de apoio à formalização segundo esta abordagem (documentação das especificações, verificação de consistência, controle de sinônimos, identificação de padrões e sugestão de ações conforme a definição das heurísticas).

Finalizando, é necessário lembrar que a linguagem semi-formal aqui apresentada só foi testada para a Linguagem Formal utilizada no projeto em questão. Pretende-se, futuramente, avaliar o uso daquela com outras linguagens de especificação (diagramas E-R, DFD, VDM, etc).

BIBLIOGRAFIA

[BAB85] BABB II, R. G. et al. Workshop on models and languages for software specification and design. Computer, Los Angeles, v.18, n.3, Mar. 1985.

[BAL78] BALZER, R.; GOLDMAN, N.; WILE, D. Informality in program specifications. IEEE Transactions on Software Engineering, New York, v.SE-4, n.2, Mar. 1978.

[BAL83] BALZER, R.; CHEATHAM, T. E.; GREEN, C. Software technology in the 1990's: using a new paradigm. Computer, Los Angeles, v.16, n.11, Nov. 1983.

[BAL85] BALZER, R. A 15 year perspective on automatic programming. IEEE Transactions on Software Engineering, New York, v.SE-11, n.11, Nov. 1985.

[BAR85] BARSTOW, D. R. Domain-specific automatic programming. IEEE Transactions on Software Engineering, New York, v.SE-11, n.11, Nov. 1985.

[DAV80] DAVIS, A. M. Automating the requirements phase: benefits to later phases of the software life-cicle. In: INTERNATIONAL COMPUTER SOFTWARE & APPLICATIONS CONFERENCE, 4., Chicago, Oct. 27-31, 1980. Proceedings. New York, IEEE, 1980. COMPSAC 80.

[DAV82] DAVIS, A. M. The design of a family of application-oriented requirements languages. Computer, Los Angeles, v.15, n.5, May 1982.

[KLE67] KLEENE, S. C. Mathematical Logic. John Wiley & Sons, 1967.

[LOH91] LOH, S. Uma Linguagem comum entre usuários e analistas para definição de requisitos de sistemas de informação. Porto Alegre, CPGCC/UFRGS, 1991. Dissertação de Mestrado.

[MAA89] MAAREK, Y. S.; BERRY, D. M. The use of lexical affinities in requirements extraction. In: INTERNATIONAL WORKSHOP ON SOFTWARE SPECIFICATION AND DESIGN , 5., Pittsburgh May 19-20. 1989. Proceedings. Publicado em Software Engineering Notes, New York, v.14, n.3, May 1989.

[MEY85] MEYER, B. On formalism in specifications. IEEE Software, Los Alamitos, v.2, n.1, Jan. 1985.

[TEO82] TEOREY, T. J. & FRY, J. P. Design of database structures. Englewood Cliffs, Prentice-Hall, 1982.