

31988-7

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL  
INSTITUTO DE INFORMÁTICA  
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Uma Linguagem Comum entre Usuários  
e Analistas para Definição de  
Requisitos de Sistemas de Informação

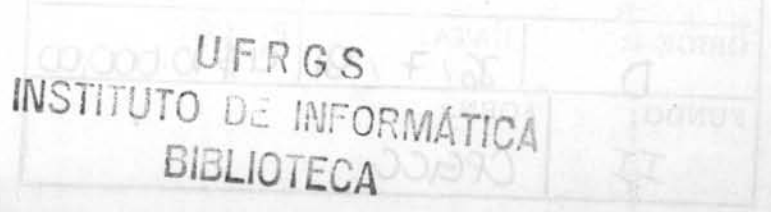
por  
Stanley Loh



Dissertação submetida como requisito  
parcial para a obtenção do grau  
de Mestre em Ciência da Computação

orientador  
Prof. José Mauro Volkmer de Castilho

Porto Alegre, Janeiro de 1991.



ESTE LIVRO DEVE SER DEVOLVIDO NA  
ÚLTIMA DATA ARMAZENADA

### CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Loh, Stanley

Uma linguagem comum entre usuários e analistas para definição de requisitos de sistemas de informação. / Loh Stanley. - Porto Alegre: CPGCC da UFRGS, 1991. 180p.

Dissertação (mestrado) - Universidade Federal do Rio Grande do Sul, Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, 1991. Orientador: Castilho, José Mauro Volkmer de.

Dissertação: análise de requisitos, comunicação usuário-analista, informalidade, especificação de software, linguagem formal, formalização.

Dedico este trabalho  
à memória  
do meu pai

## SUMARIO

RESUMO.....	p.7
ABSTRACT.....	p.8
1 INTRODUÇÃO.....	p.10
PARTE I: ANALISE BIBLIOGRAFICA.....	p.13
2 DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO: ETAPAS E CONCEITOS.....	p.14
2.1 Fase de Análise de Requisitos.....	p.18
2.2 Fase de Especificação.....	p.26
3 PROBLEMAS E SUGESTÕES DE SOLUÇÕES.....	p.29
3.1 Fase de Análise de Requisitos.....	p.29
3.2 Comunicação entre Usuários e Analistas.....	p.36
3.3 Fase de Especificação.....	p.43
3.4 Transformação de Informal para Formal.....	p.47
3.5 Necessidade de uma Linguagem Comum.....	p.55
PARTE II: PROPOSTA DE LINGUAGEM COMUM.....	p.62
4 FUNDAMENTOS DA PROPOSTA.....	p.63
4.1 Linguagem Comum.....	p.63
4.2 Dicionário de Termos.....	p.66
4.3 Linguagem Informal.....	p.68
4.4 Linguagem Formal.....	p.68
4.4.1 Parte de Dados e Estrutura.....	p.70
4.4.2 Parte de Operações.....	p.73
4.4.3 Parte de Restrições.....	p.77
4.5 Paradigma Transformacional.....	p.79
4.6 Heurísticas.....	p.80
4.6.1 Heurísticas para Coleta de Informações.....	p.84



4.6.2	Heurísticas para Transformações entre linguagens.....	p.85
5	<b>APRESENTAÇÃO DA PROPOSTA.....</b>	<b>p.87</b>
5.1	<b>Definição da Linguagem Comum.....</b>	<b>p.87</b>
5.1.1	Parte de Dados e Estrutura.....	p.87
5.1.2	Dicionário de Dados.....	p.92
5.1.3	Parte de Operações.....	p.95
5.1.4	Parte de Restrições.....	p.99
5.2	<b>Definição das Heurísticas para Coleta.....</b>	<b>p.104</b>
5.3	<b>Definição das Heurísticas para Transformação     da Linguagem Informal para a Linguagem Comum.....</b>	<b>p.109</b>
5.3.1	Parte de Dados e Estrutura.....	p.110
5.3.2	Parte de Operações.....	p.112
5.3.3	Parte de Restrições.....	p.114
5.4	<b>Definição das Heurísticas para Transformação     da Linguagem Comum para a Linguagem Formal.....</b>	<b>p.116</b>
5.4.1	Parte de Dados e Estrutura.....	p.116
5.4.2	Parte de Operações.....	p.120
5.4.3	Parte de Restrições.....	p.127
5.5	<b>Definição das Heurísticas para Transformação     da Linguagem Formal para a Linguagem Comum.....</b>	<b>p.134</b>
5.5.1	Parte de Dados e Estrutura.....	p.134
5.5.2	Parte de Operações.....	p.134
5.5.3	Parte de Restrições.....	p.139
6	<b>ESTUDO DE CASO.....</b>	<b>p.141</b>
6.1	<b>Primeira Etapa: coleta de dados (obtenção     da Linguagem Informal).....</b>	<b>p.141</b>

6.2	Segunda Etapa: transformação para a Linguagem Comum.....	p.146
6.3	Terceira Etapa: transformação para a Linguagem Formal.....	p.149
6.4	Quarta Etapa: paráfrase da especificação.....	p.154
7	PROPOSTA DE FERRAMENTAS.....	p.164
8	CONCLUSÃO.....	p.165
9	BIBLIOGRAFIA.....	p.173

## RESUMO

O presente trabalho tem por objetivo apresentar uma linguagem comum entre Usuários e Analistas para definição de requisitos, a ser utilizada durante as fases de Análise de Requisitos e Especificação, realizadas durante o desenvolvimento de Sistemas de Informação.

A motivação para o trabalho surgiu da busca de uma solução para o problema de compatibilizar as diferenças entre as linguagens usadas por aqueles. Normalmente, são utilizados dois tipos de linguagens.

O primeiro tipo tem, por principal característica, a informalidade, sendo as linguagens deste tipo, portanto, naturais mas pouco precisas. Já as linguagens do segundo tipo apresentam grande precisão, mas pouca legibilidade.

Considerando que as linguagens informais são melhores para a participação dos Usuários no desenvolvimento de Sistemas de Informação e que as linguagens formais são úteis e necessárias para que Analistas elaborem a especificação do sistema e projetistas a interpretem, fez-se necessário o estudo de uma linguagem intermediária que busque um meio termo entre legibilidade (ou naturalidade) e precisão e que, ao mesmo tempo, seja próxima das linguagens informais e formais já em uso.

São também apresentadas, neste trabalho, heurísticas (regras informais) para as transformações entre as linguagens, para justificar a referida proximidade, e um estudo de caso para avaliação dos graus de precisão e legibilidade da linguagem comum proposta.

**Palavras-chave:** análise de requisitos, comunicação usuário-analista, informalidade, especificação de software, linguagem formal, formalização.

## ABSTRACT

The objective of this work is to present a common language for users and analysts, for requirements definition during Information Systems development.

The motivation for this work arose from the study of the communication problem that users and analysts have, working with different languages of at least two kinds (natural and formal).

Natural languages have informality as their main characteristic, hence are not precise. On the other side, formal languages are precise, but sometimes not readable.

Informal or natural languages are better for user participation in information system development, and formal languages are useful and necessary to analysts when they create a system specification for implementors. It is necessary to search for an intermediate language, that could play a middle role between readableness and precision, and that, at the same time, is relatively close to informal and formal languages.

In this work, heuristics (informal and common sense rules) for requirements elicitation and for transformations between languages are defined too. A case study is detailed, for illustrate the degree of precision and readableness of the common language proposed here.

**Keywords:** requirement analysis, user-analyst communication, informality, software specification, formal language, informal-formal transformation.

## 1 INTRODUÇÃO

Este trabalho reflete a preocupação que a comunidade de pesquisa vem apresentando com relação às primeiras fases do desenvolvimento de Sistemas de Informação (SI).

As fases de Análise de Requisitos e Especificação procuram definir o Sistema de Informação, de forma que projetistas e implementadores não precisem recorrer novamente à coleta de dados sobre o que deve fazer o sistema.

Ambas podem ser consideradas as mais importantes, justamente, por serem as primeiras. Daí que as fases seguintes serão realizadas com base em decisões tomadas nestas primeiras fases. Além disto, é nestas fases que os objetivos do SI são definidos. Portanto, corrigir erros ou defeitos nestas fases é crucial para o sucesso do SI.

Grande parte das insatisfações dos Usuários para com os Sistemas de Informação se dá porque os objetivos (e conseqüentemente a finalidade) destes sistemas não estão voltados, de fato, para os problemas e anseios dos Usuários.

Assim, a maioria dos sistemas desenvolvidos e implantados acabam por serem usados de forma precária, já que não solucionam todos os problemas para os quais foram criados.

Pesquisas foram e ainda estão sendo feitas para melhorar a Especificação dos Sistemas de Informação. Foram propostas linguagens, técnicas e ferramentas que permitem especificar de modo preciso aqueles sistemas.

Acontece, porém, que ainda não se pode garantir que um sistema especificado realmente atenderá às necessidades do seu

futuro Usuário. Problemas de comunicação entre as pessoas envolvidas (como interpretações erradas, informações fornecidas de modo incompleto ou impreciso, etc) dificultam a coleta, definição e validação dos requisitos do SI (atividades estas que constituem a Análise de Requisitos).

Muitos destes problemas de comunicação ocorrem pelas diferenças que há entre as linguagens utilizadas por Usuários e Analistas quando coletando, definindo, validando e especificando Sistemas de Informação.

Os primeiros estão acostumados com linguagens naturais (informais por natureza), que são mais legíveis, porém menos precisas.

Já os Analistas (e outros profissionais da área envolvidos no desenvolvimento do sistema) precisam utilizar linguagens formais, que não devem permitir ambigüidades ou imprecisões (pois o SI será projetado e implementado com base nas informações contidas no seu documento de Especificação). Estas linguagens, entretanto, não são acessíveis aos Usuários, uma vez que, para serem lidas, requerem treinamento.

Surge então a necessidade de uma linguagem comum entre Usuários e Analistas, a fim de facilitar a comunicação entre estes durante a definição dos requisitos do SI. Esta linguagem também deverá apresentar graus intermediários de precisão e legibilidade (ou naturalidade), para diminuir a distância entre linguagens informais e formais usadas.

O presente trabalho discute de forma introdutória o processo de desenvolvimento de Sistemas de Informação, definindo etapas e conceitos, no capítulo 2. Dar-se-á maior atenção para as fases de

Análise de Requisitos e Especificação.

No capítulo 3, serão discutidos alguns problemas específicos, encontrados naquelas duas fases, e algumas sugestões de soluções, bem como será melhor detalhada a necessidade de uma linguagem comum, como já introduzido.

O capítulo 4 apresentará os fundamentos teóricos da Linguagem Comum proposta neste trabalho, defendendo seus objetivos, estratégias e características. Serão apresentadas também, como premissas, uma linguagem informal e uma linguagem formal, especificamente escolhidas para o teste da Linguagem Comum. Será fundamentado também o uso das heurísticas (para coleta das informações da Linguagem Informal e para as transformações entre as linguagens).

No capítulo 5, são apresentadas e definidas a Linguagem Comum, propriamente, e as heurísticas a ela associadas.

Um estudo de caso é apresentado no capítulo 6, para avaliação prática das idéias contidas na proposta.

Por fim, discutem-se a possibilidade de desenvolvimento de ferramentas automatizadas, para auxílio no uso da Linguagem Comum (capítulo 7), e as conclusões tiradas deste trabalho (capítulo 8).



## PARTE I: ANÁLISE BIBLIOGRÁFICA

A seguir, é feita uma análise de trabalhos e idéias, encontrados na literatura, sobre os assuntos em questão neste trabalho, tentando-se assim firmar um consenso.

## 2 DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO: ETAPAS E CONCEITOS

O processo de desenvolvimento de Sistemas de Informação (SI) tem por objetivo primeiro elaborar um sistema que satisfaça as necessidades e solucione os problemas de uma determinada Organização e das pessoas que desta fazem parte.

Através da racionalização de tarefas manuais, intelectuais e automatizadas e do planejamento das informações manipuladas, pode-se obter um Sistema de Informação que cumpra a função a que foi destinado.

Mas tal processo não é trivial, mesmo em se tratando de Organizações pequenas. Envolve pessoas, máquinas, informações, fluxos de recursos dos mais variados tipos, e os mais variados problemas.

Então, para desenvolver um Sistema de Informação adequado a determinada Organização, é necessário, antes de tudo, estudar e planejar como realizar este desenvolvimento. Que fatores influenciam este processo? Quais os melhores métodos? Que ferramentas de auxílio são necessárias? Que tipos de pessoas devem desempenhar que funções? Que tipos de cuidados devem ser tomados para garantir que o produto final será o desejado? Como aumentar a produtividade do processo de desenvolvimento de Sistemas de Informação?

Muitos pesquisadores têm-se lançado à procura de respostas para estas e outras perguntas. Há anos o desenvolvimento de SIS tem sido estudado, e resultados significativos foram obtidos. Porém, está-se ainda longe de um paradigma sem problemas e muito há que se fazer para alcançar tal perfeição.

Um dos grandes progressos, neste sentido, foi a definição do processo em fases, estruturando as tarefas desempenhadas, as pessoas envolvidas, as informações necessárias e os produtos resultantes em cada fase. Apesar de haver alguns pensamentos discordantes, há um certo consenso sobre esta maneira de proceder.

O desenvolvimento de SIs pode ser dividido em 4 fases principais, assim ordenadas: Análise de Requisitos, Especificação, Projeto e Implementação. As vezes, as duas primeiras são consideradas como uma só (ver [Fai86]). Aqui será feita a divisão para melhor entendimento.

Os nomes das fases variam de acordo com a área do pesquisador e o enfoque dado, mas basicamente as características são as mesmas. Então, por exemplo, a primeira fase, às vezes, é denominada de Análise e Definição de Necessidades; a segunda, de Projeto Conceitual ou Modelagem; a terceira, de Projeto Lógico/Físico e a última, de Codificação ou Programação [Teo82] [Lei87] [Fai86] [Loh88b].

É bom salientar que há algumas variações para esta divisão, sendo incorporadas, às vezes, outras fases, tais como Teste, Verificação e Validação, que podem fazer parte de alguma das fases principais (note-se a preferência pelo termo "fases principais", dando a entender que há margem para subfases).

Há também quem veja o processo de desenvolvimento, ou parte dele, como um ciclo, o que não contraria a idéia aqui apresentada, uma vez que refinamentos ou retornos a fases anteriores podem ser feitos quando constatados erros ou imperfeições (apesar de isto evidenciar uma fraqueza nos métodos

de desenvolvimento).

O paradigma de Prototipação também pode ser considerado uma variação ao ciclo em fases apresentado aqui, mas aquele não deixa de ter pontos em comum com este (para maiores detalhes sobre Prototipação, ver [IEEE SW Eng. Notes, Dezembro de 1982]).

A primeira fase, Análise de Requisitos (AR), até o momento, é a menos estudada, não por ser menos importante que as demais, mas por apresentar os problemas mais difíceis e menos tratáveis sistematicamente. Nesta fase, são coletadas as informações sobre os requisitos do sistema, ou seja, o que deverá ser produzido. O Analista de Sistemas é o responsável por esta etapa, da qual o Usuário (proprietário do sistema) participa ativamente. O produto da fase de Análise de Requisitos é um documento descrevendo informalmente os serviços que o Sistema proverá ao Usuário (ainda não há um consenso geral sobre como deve ser este produto) [Loh88b] [Lei87]. Outras informações sobre necessidades de desempenho do sistema (tempo de resposta, volume de dados, etc) também devem ser determinadas nesta fase.

A fase de Especificação é a responsável pela definição formal (precisa, clara, não-ambígua, completa) do sistema, isto é, os requisitos coletados anteriormente deverão agora ser formalizados para que não haja erros devidos a má interpretação durante as fases seguintes. Entretanto, a Especificação não deve se preocupar com como será concretizado (projetado e implementado) o sistema, mas tão-somente com quais funções serão desempenhadas e que informações serão manipuladas. Quem desempenha esta tarefa é também o Analista de Sistemas com apoio

do Usuário. O documento final desta fase é a Especificação do Sistema (ou documento de Especificação) e é elaborado a partir do Documento de Requisitos, produto da Análise de Requisitos.

Já a fase de Projeto busca criar uma alternativa de implementação para o sistema especificado, conforme o ambiente e os recursos físicos e computacionais disponíveis. As estruturas de dados, os modos como as operações sobre os dados (programas) serão implementadas, os mecanismos de controle, o uso dos recursos físicos, tudo isto deverá ser planejado e descrito num documento de Projeto. Este deve respeitar os requisitos necessários para o sistema, coletados e descritos nas fases anteriores. A tarefa de projeto está a cargo do Projetista (ou Projetistas) do Sistema.

Por fim, a fase de implementação transforma o Projeto do sistema num sistema concreto, dependente de máquina. Nesta etapa, os programas serão codificados segundo uma linguagem executável em alguma máquina real. O(s) programador(es) é(são) quem realiza a tarefa de implementação.

Cabe ainda esclarecer que as informações sobre o SI devem estar coerentes entre si ao longo do desenvolvimento e em cada fase. Isto é, os requisitos (objetivos) do sistema não poderão ser alterados conforme vão sendo desenvolvidas as etapas. Assim, ao final do desenvolvimento, as características do sistema implementado atenderão exatamente (nem mais, nem menos) aos requisitos propostos no Documento de Requisitos e formalmente definidos no Documento de Especificação.

Da mesma forma, a implementação do sistema deverá

corresponder aos planos de implementação estabelecidos na fase de Projeto.

Entretanto, se houver algum tipo de mudança (e isto pode ocorrer quando são detectados erros ou imprecisões) em alguma fase, as informações correspondentes a esta mudança também deverão ser propagadas para as demais fases.

Para controlar a coerência do sistema durante o desenvolvimento, faz-se necessária uma boa metodologia que abranja todas as fases. Porém, existem poucas metodologias (ou quase nenhuma) que abranjam todo o desenvolvimento de SI. O que realmente acontece é a integração de métodos e técnicas, desenvolvidos separadamente para partes daquele processo, de modo a formarem uma metodologia de desenvolvimento (metodologia é uma coleção de métodos e técnicas; método é um processo ordenado e sistemático para realizar determinada tarefa e técnica é uma coleção de ferramentas e instruções sobre como utilizá-las, segundo [Lei87]).

Neste trabalho, o estudo será limitado às fases de Análise de Requisitos, Especificação e à fase de Projeto quando se tratar da "interface" desta com a fase de Especificação.

## **2.1 Fase de Análise de Requisitos**

A fase de Análise de Requisitos (AR), por ser a primeira, tem uma importância crítica no desenvolvimento de SIs. As fases seguintes a esta tomarão como base as informações coletadas e documentadas durante a AR. Grande parte das decisões tomadas durante o desenvolvimento levarão em conta o que foi estabelecido

como requisitos do sistema [Teo82 p.15], e as pessoas responsáveis tentarão tornar o produto final do desenvolvimento, isto é, o SI gerado, o mais fiel possível ao sistema proposto como objetivo na AR.

A importância da AR se dá também em razão de serem os erros cometidos na definição e especificação de requisitos os mais caros e difíceis de corrigir, uma vez que tais erros ocasionam desvio do objetivo (o que fazer), enquanto que os erros decorrentes de projeto e/ou implementação afetam apenas os planos para atingir os objetivos (como fazer) [Gom81] [Bos89] [Haa82].

O objetivo da AR é coletar, analisar, organizar e documentar as informações que serão utilizadas nas fases seguintes. Entre estas informações, deverão constar os dados necessários, a estrutura daqueles e as operações e restrições sobre estes dados, independentemente de como será implementado o sistema. Entre os dados, estarão as informações recebidas pelo sistema e as geradas por este.

A estrutura dos dados define as associações entre os dados e as composições destes. Como operações, entende-se todas as funções e procedimentos, sejam manuais, intelectuais (decisões) ou automatizáveis, que são ou serão desempenhados dentro da Organização e que utilizam os dados descritos anteriormente. Como restrições sobre os dados, têm-se informações sobre as propriedades dos dados que não podem ser alteradas, informações sobre o controle de acesso a determinados dados por pessoas ou departamentos e informações sobre a necessidade de uso dos dados (volume e taxa de crescimento de conjuntos de dados, volume de execução de operações, tempo de execução de uma operação, etc).



Estas formam a perspectiva de uso dos dados (Usage Perspective) como definido em [Teo82].

Para a atividade de coleta, podem ser usadas uma das seguintes técnicas: entrevistas, questionários, análise de formulários, análise de sistemas automatizados já existentes e observação do ambiente [Lei87] [Dav82b]. Todas estas devem ser utilizadas, independentemente ou em conjunto, pelo Analista de Sistemas.

A técnica de entrevistas talvez seja a mais usada, porém ela requer do Analista uma certa habilidade para se alcançarem bons resultados. O Analista deve saber fazer as perguntas certas, no momento certo e às pessoas certas. Deve também saber conduzir a entrevista para seus objetivos, evitando desvios indesejáveis. Outra habilidade necessária é a de saber ouvir: de nada adiantará entrevistar uma pessoa se não se sabe extrair informações do que é dito pelo entrevistado. Por fim, entrevistar também tem a ver com relações humanas e aí entra em jogo uma série de outros fatores. Devido ao contato pessoal entre entrevistador e entrevistado, esta técnica permite deduzir informações "escondidas" quando são notadas atitudes que aparentam omissão de informações (por exemplo, quando o entrevistado foge de um assunto, dando a entender que não quer falar sobre este assunto). Para maiores detalhes sobre relações humanas, ver [Loh89c].

Os questionários, por sua vez, são bons para obter pequenas informações de um grande número de pessoas em pontos afastados [Dan74]. Favorecem também a aquisição de informação por darem tempo para que as pessoas reúnam os dados para as respostas



[Dan74]. As desvantagens de tal técnica são que as pessoas, às vezes, interpretam mal as perguntas e recusam-se a responder questionários, ou o fazem com desinteresse, principalmente no caso de questionários enfadonhos e que tomam muito tempo (outros detalhes em [Loh89c]).

Tanto as entrevistas quanto os questionários fazem uso de perguntas para a obtenção de informações, por isto faz-se necessário estudá-las um pouco mais.

Há dois tipos de perguntas: não-estruturadas e estruturadas.

As perguntas do primeiro tipo são amplas e necessitam mais tempo para serem respondidas. São usadas para se obterem idéias gerais, conhecerem-se intenções e sentimentos e para atrair para debates. São melhores porque incitam a pessoa que responde a raciocinar sobre o assunto [Nir81].

Já as perguntas estruturadas, aquelas que requerem simplesmente "sim" ou "não" como resposta, são mais fáceis de responder e ajudam na obtenção de fatos específicos. Também são usadas para comparar pensamentos e para pedir posições definidas [Nir81].

Em [Nir81], sugere-se o seguinte para formular perguntas não-estruturadas:

- formar perguntas que não podem ser respondidas com "sim" ou "não";
- preceder as palavras-chave com perguntas do tipo "o que é", "o que me diz sobre", etc;
- revidar com resumos, cuja não-concordância levará à coleta de mais informações.

Estes dois tipos de perguntas estão bem presentes no

comportamento do Analista, quando este entrevista alguém ou elabora um questionário, apesar de, às vezes, não ficarem explícitas. No começo da coleta de dados, o Analista geralmente usa perguntas não-estruturadas como forma de conhecer os assuntos envolvidos e limitá-los. Feito isto, mais perguntas não-estruturadas são colocadas a fim de conhecer e limitar cada assunto em especial. Após sucessivos refinamentos, são utilizadas perguntas estruturadas para confirmar o que se coletou. As perguntas padronizadas envolvendo palavras-chave ocorrem com frequência, principalmente quando o entrevistado cita um novo assunto, fato ou palavra.

Ainda no caso de uso de questionários, requer-se do Analista uma habilidade especial para leitura. Semelhante ao caso da entrevista, onde era necessário saber ouvir, aqui se faz necessário saber extrair informações de textos escritos.

Outra técnica de coleta é a observação do ambiente da Organização. O Analista deve passar algum tempo na Organização, observando as funções desempenhadas nela e até mesmo desempenhando algumas. A atenção deverá concentrar-se nas funções que há na Organização, como são desempenhadas, que informações e materiais utilizam. Esta técnica é boa no sentido que coleta diretamente as informações, sem que alguém as forneça ao Analista, porém falha quando as pessoas sendo observadas se inibem ou se superam, modificando o comportamento habitual no desempenho das funções.

Para a técnica de análise de formulários, é necessário abstrair as funções a partir das informações contidas nos

formulários. É uma técnica de coleta parcial e serve mais como complemento de outra(s) técnica(s).

A última técnica sugerida, análise de sistemas já existentes, pressupõe a automatização de algumas ou todas funções de uma Organização. É útil para coleta parcial de informações, porém recomenda-se especial atenção neste caso, uma vez que o sistema existente pode estar ultrapassado.

Em todas as técnicas de coleta, há que se preocupar com o problema de amostragens. Deve-se certificar que todos os segmentos da Organização (departamentos, gerência, funcionários dos escalões mais baixos, diversas funções, etc) estejam representados significativamente, pois uma pessoa não possui conhecimento completo sobre determinado assunto e, se mesmo assim o fosse, poderia esquecer algo.

Por fim, como dito em [Loh88b], a atividade de coleta deve ser rápida (levar o menor tempo possível), completa (coletar todas as informações necessárias), objetiva (não se preocupar em obter informações irrelevantes) e confortável (para a Organização e seus funcionários, não lhes prejudicando o desempenho).

Como já visto antes, o Analista de Sistemas é responsável pela atividade de coleta de dados. Porém, ele não a faz sozinho, sendo necessária a cooperação dos Usuários (funcionários da Organização), já que são estes quem melhor conhece a Organização e, portanto, quem melhor pode fornecer as informações desejadas [Chr87].

A participação do Usuário é um fator chave para o sucesso da AR [Par87], mas, para que esta participação seja efetiva, uma boa comunicação entre Usuários e Analistas deverá ser estabelecida.

Em [Bos89], são citados estudos sobre a correspondência entre boa comunicação e produtividade (de desenvolvimento de SI's), e [Vit83] declara que a facilidade na resolução de problemas depende da qualidade no relacionamento entre Usuários e Analistas. Em [Bos89] também, é comentada a associação entre comunicações pobres e insatisfações dos Usuários (insatisfação com o SI desenvolvido).

Uma boa comunicação permite que os Analistas colem e definam melhor os requisitos e que os Usuários forneçam e avaliem melhor as informações necessárias. Portanto, as habilidades comunicativas devem ser aprimoradas, principalmente nos Analistas de Sistemas (não que estas sozinhas garantam o sucesso do desenvolvimento, mas é certo que complementarão substancialmente os métodos e metodologias de desenvolvimento) [Bos89 p.2, 3, 5].

A atividade seguinte à coleta dos dados é a análise dos dados. Nesta, o Analista de Sistemas verifica as informações coletadas, eliminando aquelas que não interessam nesta fase (por exemplo, informações sobre a implementação do sistema e informações sobre recursos físicos necessários), as quais podem ter sido coletadas por acaso. Também deverão ser verificadas possíveis omissões de informações (frases incompletas, falta de complementos adnominais, etc), e, sempre que necessário, deve-se completar o conjunto de informações, refazendo-se parte da atividade de coleta.

Após a análise, os dados deverão ser organizados segundo a área ou assunto a que pertençam. O objetivo desta atividade é classificar os dados para que possam ser melhor entendidos. Esta classificação dos dados é principalmente importante no caso de

sistemas médios e grandes (requeridos para organizações de porte) e deve ser estabelecida naturalmente, isto é, conforme as divisões da Organização (por exemplo, classificar por departamentos, seções, áreas, assuntos) e conforme o Usuário a vê [Dav82 p.24].

Por fim, as informações devem ser documentadas. Supõe-se que entre aquelas estarão os dados a serem usados, sua estrutura, as operações de manipulação, as restrições e os requisitos de uso. Este documento, por ser o produto final da AR, será usado como entrada na fase de Especificação e, portanto, deverá ter como qualidades [Loh88b]:

- clareza: deve ser fácil de ser entendido;
- precisão: nenhuma de suas partes deverá ter dupla interpretação;
- objetividade: descrever somente o que é relevante para a fase de Especificação;
- consistência: sem contradições;
- inteireza: deve ser completa, isto é, conter todos os dados relevantes.

Apesar de desejável que o documento seja completo, alguns autores advogam que os requisitos são incompletos por característica [Lei87]. Esta discordância, entretanto, não deve desestimular a procura de métodos e técnicas que tornem completo o documento de requisitos de um SI.

Quanto à precisão do documento, esta não deve ser alcançada em detrimento da clareza do documento. Formalismos não devem ser empregados, e o aconselhável é usar a linguagem natural, uma vez que o documento, geralmente, serve como contrato entre a

Organização (representada pelos Usuários) e o pessoal que desenvolve o SI. O documento deve ser passível de entendimento pelos Usuários, e os formalismos empregados no desenvolvimento de SIs confundem aqueles além de forçá-los a aprender linguagens que não lhes cabem.

Como última etapa da fase de AR, o documento de requisitos deve ser validado pelo(s) Usuário(s) (Validação é o processo pelo qual o Usuário julga se o documento de requisitos corresponde exatamente ao seu problema [Nos88]). O fato de se validarem os requisitos, isto é, validar o SI, o mais cedo possível permite que alguns erros sejam detectados e corrigidos também mais cedo possível, principalmente aqueles erros mais caros e mais difíceis de corrigir: os desvios do objetivo.

## **2.2 Fase de Especificação**

O objetivo da fase de Especificação de Requisitos (ou Especificação do Sistema) é formalizar as informações contidas no documento de requisitos (produto da AR), eliminando as ambigüidades e imprecisões. Desta forma, a descrição dos requisitos do SI será corretamente compreendida pelo pessoal de projeto.

Basicamente, a fase de Especificação só se limita a transformar as informações documentadas pela AR. Novas informações só devem ser acrescentadas a modo de resolver ambigüidades ou imprecisões. Caso seja notada a falta de algum requisito, este deverá primeiramente ser acrescentado ao documento de requisitos, através de um retorno à fase de AR.

Também aqui nesta fase, não deverão ser tratados assuntos referentes à implementação do sistema, os quais pertencem às fases de Projeto e Implementação.

Para a atividade de formalização das informações, deverá ser utilizado um formalismo, ou seja, uma linguagem formal. A característica principal de uma linguagem formal é que ela permite somente uma interpretação dos fatos através dela comunicados, contrariamente às linguagens naturais, que são informais e, por isto, carregam imprecisões e ambigüidades.

Para fazer esta transformação (informações em uma linguagem informal para uma linguagem formal), o Analista deverá ter experiência nesta atividade (analisando textos em linguagens informais e escrevendo em linguagens formais) e, sempre que possível e necessário, deverá solicitar ajuda ao Usuário, pois mesmo experiente, o Analista poderá se confundir com as imprecisões inerentes às linguagens naturais.

O produto desta fase é o documento de especificação do SI. Nele estarão os requisitos do sistema escritos em um linguagem formal.

Este documento deverá ter como qualidades (em parte, conforme [Dav80]):

- precisão (permitir apenas uma interpretação para o que descreve);
- consistência (não conter contradições);
- objetividade (só conter informações relevantes);
- inteireza (descrever todas as informações relevantes);
- fidelidade (ao documento de requisitos).



Apesar da linguagem formal ser mais rigorosa que a linguagem informal, aquela deve ser legível, de modo a permitir que o pessoal que participa da fase de Projeto compreenda as informações descritas, isto é, o uso de formalismos não deve dificultar a legibilidade do documento de especificação.

Além de não possuir imprecisões, uma descrição em linguagem formal também tem a vantagem de poder ser verificada automaticamente.

A verificação de uma descrição formal consiste em analisar as propriedades e características desta descrição, procurando-se descobrir falhas (como falta de informações, informações inconsistentes ou contraditórias, etc). O ideal é que esta verificação seja feita automaticamente por uma ferramenta automatizada.

Idealmente também, a Especificação (ou Documento de Especificação) do Sistema também deve ser executável, isto é, deve ser descrita em uma linguagem formal que possa ser interpretada em uma simulação de execução do Sistema de Informação, mesmo que ineficientemente. Assim, a especificação do sistema permite que a prototipação do sistema final possa ser validada pelo Usuário e que os erros possam ser corrigidos antes que se parta para uma implementação [Bal78] [Bal83].

Uma estratégia deste tipo sugere o uso de técnicas ou até mesmo de ferramentas automatizadas que otimizem esta especificação (que passa a ser vista como um programa abstrato em uma linguagem de programação de alto nível [Bal78]) e que extraiam uma implementação (do sistema) executável em alguma máquina real [Bal83].



### 3 PROBLEMAS E SUGESTÕES DE SOLUÇÕES

A seguir, são discutidos os principais problemas que afligem as fases de Análise de Requisitos e Especificação do desenvolvimento de Sistemas de Informação, bem como sugestões de soluções encontradas na literatura.

#### 3.1 Fase de Análise de Requisitos

O primeiro problema da AR é o fato de muitos desenvolvedores de Sistemas de Informação a ignorarem. É comum SI's serem desenvolvidos sem a devida atenção a esta primeira etapa, ou seja, os dados são coletados, analisados e organizados de modo rápido e, após isto, passa-se para a especificação do sistema, sem prévias organização, documentação e validação do que foi coletado.

Uma das razões de tal problema é a falta de métodos e técnicas para a realização da AR.

[Bos89] afirma que há relativamente poucos trabalhos de pesquisa sobre o assunto e poucas sugestões práticas e bem-formuladas sobre como coletar informações. [Maa89] acrescenta que os métodos existentes (como os de Jackson, Parnas, Orr, etc) não explicam realmente como os requisitos são obtidos.

A razão desta lacuna se dá porque estes métodos foram desenvolvidos para as etapas de especificação e/ou projeto, o que exclui a atividade de coleta dos dados.

Esta falta de métodos e técnicas, ou mesmo de uma abordagem sistemática para a realização da AR, tem sobrecarregado

o Analista de Sistemas, que deve detalhar seu próprio método, avaliando e melhorando-o através de testes e julgamentos pessoais.

[Mat87 p. 461] resume: as habilidades e a experiência do Engenheiro de Software têm desempenhado importantes funções nas primeiras fases do desenvolvimento de SI's.

Esta dependência da atividade à experiência do Analista torna-o responsável direto pelo sucesso ou insucesso da AR.

Em geral, os Analistas não têm se saído mal, pois realmente desenvolveram seus próprios métodos (muitas vezes intuitivamente) e têm conseguido bons resultados. Porém as dificuldades características da AR, aliadas à falta de ferramentas de auxílio, conseguem conturbar ainda mais o trabalho do Analista, provocando erros durante a AR.

Podem ser sugeridas algumas formas como realizar a coleta dos dados, analisando-se a literatura ou o modo como os Analistas agem.

Primeiro, o uso de perguntas é talvez a técnica mais difundida e abrange os questionários e as entrevistas. Começar com perguntas genéricas padrões permite que se conheçam os assuntos de interesse para a Organização e os seus limites.

[Bos89 p.7] cita a técnica de "brainstorming", que também se aplica bem para um começo de AR. Nesta técnica, todas as idéias são úteis, devendo ser pouco avaliadas (a quantidade sobrepõe a qualidade).

Após, faz-se uso de perguntas específicas para explorar e detalhar cada assunto em particular. Como [Bos89 p.6] declara, cada pergunta cria um mini-quadro (um ponto de vista),

direcionando o pensamento.

As perguntas específicas aparecem em formas padrões com lacunas que devem ser preenchidas com palavras-chave. Para tanto, é necessário identificar as palavras-chave da Organização (podem ser objetos, informações, cargos, atividades, documentos, etc).

Exemplificando, poderia haver a pergunta "como é feito(a) o(a) \_\_\_\_\_ ?", a ser completada com uma atividade (por exemplo, contratação de funcionários) e a pergunta "o que é feito com o(a) \_\_\_\_\_ ?", devendo ser completada com um tipo de informação ou documento (por exemplo, pedido de compra).

[Bos89] propõe alguns tipos de perguntas específicas que servem para alcançar certos objetivos ("outcome frame"), para rever o que foi dito ("backtracking frame"), para estimular a criatividade e alterar o ponto de vista ("as-if frame"), para guiar a direção da comunicação ("procedures") e para obter informações mais detalhadas ("pointers").

A seguir, têm-se exemplos de cada tipo:

- "outcome frame": qual o motivo deste encontro ?; o que se quer alcançar neste encontro ?;
- "backtrack frame": vamos resumir o que foi decidido ...; você está dizendo que ...;
- "as-if frame": o que você faria se ... ?; imagine que ...; vamos supor que ...;
- "procedures": que evidência nos leva a concluir que <preencher com um objetivo> foi alcançado ?;
- "pointers": você pode ser mais específico ?; o que exatamente você quer dizer ?; o que você quer dizer com \_\_\_\_\_ ?

Em [Loh89c p.31], é sugerido também o uso de perguntas de reflexão (onde, como, por que, etc).

As perguntas funcionam também como estímulos, induzindo o Usuário a fornecer as informações necessárias.

As vezes, durante uma entrevista, surgem comentários desvinculados do contexto, que, aparentemente, não acrescentam qualquer novidade. Em [Loh89c p.30], aconselha-se explorar a finalidade de tais comentários, pois, às vezes, eles escondem informações ainda não tratadas.

Por fim, tudo deve ser anotado para que não se perca nenhuma informação.

Apesar do progresso feito nesta área, alguns problemas ainda ocorrem durante a AR.

Por exemplo, o Usuário tem dificuldade em raciocinar sobre o que quer. Na maioria das vezes, ele sabe o que quer, mas não consegue colocar em palavras [Toe83].

Some-se a isto o fato ainda de que estes raciocínios se dão com base em apenas informações disponíveis, somente fatos recentes, análises estatísticas intuitivas e julgamentos em torno de um ponto único [Dav82b].

Por outro lado, o Analista também tem suas dificuldades, usando métodos próprios, não-sistematizados e sem auxílio de ferramentas: há esquecimento de perguntas a serem feitas ao Usuário e de informações já coletadas (se não forem anotadas e aparecerem em grandes quantidades).

Além disto tudo, a coleta de informações pode ainda ser prejudicada pelo entrelaçamento de opiniões e fatos, isto é, emoções e desejos facilmente se confundem com idéias num processo

de comunicação [Loh89c p.25, 37], o que é algo natural e comum em se tratando de comunicação de pessoas.

Para o caso de a coleta ser feita por observação, [Ken84] propõe um método, considerando analogias com a crítica de cinema. Por exemplo, luz e cor corresponderiam à imagem da Organização; o vestuário (figurino) seria a imagem do gerente; cenário, os equipamentos e ambientes; o foco e a profundidade do campo, a atenção a múltiplos objetivos; e etc. Este método deve ser usado em conjunto com outras técnicas, como entrevistas e questionários, tentando coletar o que estes não coletaram.

Quanto às atividades de Análise e Organização das informações coletadas, os problemas são de omissão. Isto é, as informações são coletadas e, logo em seguida, documentadas, não se analisando seu conteúdo e estrutura. Tal procedimento pode conduzir a documentos que contêm informações não-relevantes e que são não-estruturados.

A atividade de documentação, por sua vez, também possui suas anomalias. Se o documento de requisitos for escrito em uma linguagem formal, apropriada aos métodos computacionais (como acontece nos casos em que o documento de especificação é utilizado como documento de requisitos), ele se torna inacessível, por ser ilegível, ao Usuário.

É importante a escolha de uma boa técnica de documentação, que aumente a compreensão e o entendimento dos dados pelo Usuário (quanto melhor o Usuário entender a técnica de representação e descrição das informações, melhor ele pode validar as informações coletadas) [Nos88].

Porém, as poucas técnicas de documentação de requisitos

propostas, como PSL/PSA e CASCADE (sugeridas em [Teo82]), são mais técnicas para especificação de requisitos (documento formal) do que para documentação de requisitos.

Técnicas assim dificultam a participação do Usuário no processo de validação dos requisitos. Desta forma, o Analista fica inclinado a construir sozinho o documento e completar lacunas de informações com base em seus próprios conhecimentos, ao invés de coletar mais informações junto ao Usuário. O resultado é que o documento final é, em boa parte, uma visão do Analista, como afirma [Bos89 p. 4].

Os Dicionários de Dados (sugeridos em [Teo82]) funcionam como uma alternativa para a documentação de requisitos, já que descrevem os dados e suas associações, as operações desempenhadas sobre os dados e as restrições do sistema. Porém, ainda assim representam uma técnica que elabora documentos difíceis de ler e compreender, além de usar uma linguagem estranha ao Usuário.

[Teo82] sugere que a validação seja feita de modo compreensível ao Usuário, como, por exemplo, com uso de frases simples em linguagem natural (LN) (no caso, em Inglês) para descrever as implicações das informações que o Usuário forneceu.

[Nos88] acrescenta que não é necessário educar os Usuários com novas técnicas de documentação, bastando usar textos em LN e modelando o sistema também em LN.

Em [Cer83], é proposta uma metodologia que documenta requisitos através de frases em LN, e [Bou85] apresenta uma ferramenta cuja entrada são frases simples em LN (no formato "sujeito / verbo / complemento").

Outro problema da AR é a possibilidade de efetiva

participação do Usuário, a qual é essencial para o sucesso do desenvolvimento do SI. Ocorre, muitas vezes, que o Usuário não dá a devida importância a este fato, recusando-se a participar do processo ou participando desinteressadamente. É preciso alertá-lo de tal importância.

Outras vezes, esta participação fraca se dá por preconceitos entre Usuários e Analistas. Os Usuários acreditam que os Analistas não entendem o negócio e as necessidades da Organização, não conseguem transformar uma definição em um sistema bem sucedido, põem muita ênfase nas technicalidades, não falam uma linguagem familiar ao Usuário, não fazem o sistema como esperado pelo Usuário, etc. Já os Analistas acham que os Usuários não sabem o que querem, não sabem comunicar com precisão suas necessidades, não entendem o que o Analista explica, e assim por diante (ver [Loh89c]).

Resumindo, os principais problemas da AR são:

- desatenção com a importância da AR;
- falta de métodos, técnicas e ferramentas (para coleta, análise, organização, documentação e validação das informações);
- dificuldade para o Usuário fornecer as informações necessárias;
- falta de uma técnica de documentação que seja acessível a Usuários e adequada para Analistas;
- participação efetiva do Usuário.



### 3.2 Comunicação entre Usuários e Analistas

A comunicação é o principal problema envolvendo Usuários e Analistas [Gue83]. Uma má comunicação gera produtos com defeitos, discordâncias e intrigas. Uma vez que a AR não pode ser feita sem interação entre Usuários e Analistas, como visto anteriormente, ela está fadada a sofrer com os problemas decorrentes de má comunicação que possa ocorrer entre aqueles.

Por definição, a Comunicação se dá por uma "interface" comum aos seus participantes, permitindo que informação e entendimento sejam trocados entre as partes [Loh89c]. "A verdadeira comunicação cria verdadeiro entendimento precisa, clara e rapidamente" [Bla87]. Porém, a Comunicação é apenas uma ferramenta para o entendimento comum, sendo necessária, mas não suficiente, como explica [Pen76]: "o objetivo da Comunicação não é, necessariamente, concordância (de idéias), e sim compreensão."

A importância de uma boa comunicação se faz mais que óbvia nos dias atuais, notando-se o grande número de reuniões e encontros que acontecem, os quais consomem de forma excessiva o tempo das pessoas. É preciso saber aproveitá-lo bem, como um recurso físico de uma empresa, tornando produtivos estes encontros. [Bos89 p.2] cita estudos sobre a produtividade de encontros.

No caso de desenvolvimento de SIs, a Comunicação (entre Usuários, Analistas, Projetistas, etc) tem-se apresentado como fator-chave no sucesso do desenvolvimento. [Bos89 p.2] afirma que uma boa comunicação é a solução para muitos dos problemas no desenvolvimento de sistemas, principalmente na fase de Análise de

Requisitos, e cita alguns trabalhos que provam que a produtividade e o sucesso no desenvolvimento de SIs dependem de uma boa comunicação [Bos89 p. 2, 5].

Além disto, o processo de comunicação complementa o valor das metodologias de desenvolvimento [Bos89 p.3].

Mas o problema de Comunicação entre Usuários e Analistas não é fácil de ser resolvido.

Para começar, nota-se que há 2 tipos muito diferentes de pessoas envolvidas.

Usuários e Analistas são pessoas com funções, objetivos e percepções sobre o SI muito diferentes [Teo82]. Possuem também diferentes ideologias e interesses [Lan82]. Além disto, um não sabe o que o outro precisa ou quer saber (o que não é importante para um pode ser para o outro) [Lan82].

E ainda, os Usuários possuem conhecimento específico do domínio (ambiente da Organização) e usam o vocabulário do domínio, enquanto os Analistas estão familiarizados com as metodologias de desenvolvimento e usam um vocabulário próprio. Analistas e Usuários, além de utilizarem linguagens diferentes, possuem diferentes visões (pontos de vista) sobre o mundo [Bos89 p.3].

Some-se a isto o fato de que os Usuários apresentam dificuldades em raciocinar sobre suas necessidades e suas visões do mundo [Bos89 p.4] [Toe83].

Por outro lado, os Analistas, apesar de experientes em alguma metodologia de desenvolvimento, nem sempre apresentam habilidades comunicativas [Bos89 p.5] [Wei83] [Bar86].

Afora as diferenças, a Comunicação é problemática por

natureza, uma vez que faz uso de linguagens potencialmente ambíguas.

Apesar de ser o principal instrumento da Comunicação informativa (segundo [Jak] e [Ste72]), o uso de linguagens proporciona alguns equívocos.

Primeiramente, uma linguagem faz uso de símbolos, que são concretizações de idéias que estão na mente das pessoas. Porém, conforme [Nir81], esta concretização não é total, sendo que apenas alguns aspectos da imagem mental são simbolizados.

Além disto, não há certeza de que estes símbolos serão corretamente entendidos pelo receptor da mensagem, uma vez que os significados dos símbolos não estão neles próprios, mas na mente de quem os usa (segundo Hayakawa citado em [Dav72]).

Continuando, a Comunicação está fadada a ser invadida por emoções. Idéias e sentimentos se confundem tanto quando se emite uma mensagem, quando se a interpreta [Loh89c], e isto deturpa o conteúdo da mensagem.

Quanto à compreensão dos símbolos na linguagem, muitos problemas surgem.

Grupos diferentes de pessoas usam convenções lingüísticas próprias e, portanto, diferentes. Com isto, uma mesma palavra ou expressão pode ter vários significados dependendo de quem a usa [Loh89c p.18]. A razão disto é que os signos verbais possuem um caráter arbitrário [Pnn76]. Os conceitos são determinados pelas pessoas que os usam, segundo uma concordância [Ken78].

O uso e a difusão de linguagens proporcionaram um aumento de símbolos e conotações para estes, o que se tornou um entrave, pois o vocabulário é muito vasto para ser conhecido plenamente

[Buy74].

[DeB84] fala em um abismo semântico ("semantic gap"), caracterizado quando pessoas diferentes associam diferentes significados às palavras. Este abismo tende a confundir as pessoas, como ocorre entre Usuários e Analistas quando aqueles não entendem os jargões destes [Kos87].

A fim de melhorar a Comunicação entre Usuários e Analistas, [Bos89] propõe um protocolo para conduzir entrevistas ou encontros, modelando padrões de Comunicação. Sugere também que se usem, para melhorar a comunicação, meta-comentários, metáforas e técnicas para estruturar o conteúdo da Comunicação.

Para diminuir a imprecisão das palavras, [Bos89] sugere que os termos sejam definidos precisamente. Para tanto, na mesma referência, são propostas perguntas padrões ("pointers") que buscam definições mais precisas para os termos. Da mesma forma, em [Loh89c p.21,27], aconselha-se precisar o significado de um termo através de termos já definidos, e em [Bar86], tem-se a advertência de que o não-entendimento de algumas palavras pode levar ao não-entendimento de um assunto todo.

Sobre as palavras ainda, [Wal76] sugere: atenção à escolha dos símbolos a serem utilizados, o uso de um dicionário de termos para defini-los e explicá-los e aconselha que se evitem palavras abstratas, que podem conduzir a uma compreensão errada. Também devem ser evitadas palavras técnicas, e o ideal é o uso da terminologia local, aceita pelos membros envolvidos na Comunicação [Loh89c pg.31].

Já [Dan74] também tem seus conselhos para melhorar a Comunicação: usar palavras e frases simples, palavras curtas e

familiares, economizar em adjetivos e floreios, arranjar os pensamentos de forma lógica e evitar palavras inúteis, que não ajudam no entendimento da mensagem.

Por outro lado, a repetição é também um fator atenuante das más interpretações da linguagem, Porém a repetição deve ser feita de modos diferentes, ou seja, repetir as idéias mas com palavras diferentes [Loh 89c p.9,31].

Justamente para tornar a Comunicação mais precisa é que aconteceu a evolução da linguagem como declara [Hae]: "a necessidade de se entenderem (os seres humanos entre si) com mais clareza e maior rapidez foi um dos fatores que determinou a evolução da linguagem". E isto pode ser notado observando-se o uso de linguagens específicas para grupos específicos, como no caso de funções especializadas e como aconteceu na Idade Média, quando o latim era a língua comum da ciência [Buy74].

A análise do contexto (contexto ou conotação é aquela mensagem escondida, aquilo que é dito informalmente, ou seja, as mensagens implícitas [Buy74]) também se faz necessária para um bom processo de Comunicação, como pode ser visto pelas seguintes citações:

"o comunicador efetivo é aquele que concerne as idéias e, não, as palavras" [Dav72],

"quanto mais rico for o contexto de uma mensagem, mais limitada será a perda de informações" [Jak] e

"a ambigüidade das palavras é resolvida pelos humanos, pelo entendimento do contexto" [Ken78].

Em [Bal78], é discutido o uso da análise do contexto para completar descrições em linguagem natural, a fim de se obter a

especificação formal de um sistema. Neste artigo, é apresentado um protótipo que analisa textos em linguagem natural (previamente, estruturados sintaticamente) através de regras pré-estabelecidas, indicando frases ou partes do texto possivelmente ambíguas. Após, as partes ambíguas são completadas (para eliminar as ambigüidades) com informações contidas em outras partes do texto.

Ainda, para melhorar a Comunicação entre Usuários e Analistas, ambos deveriam ser treinados para adquirirem habilidades comunicativas (por exemplo, treinar a escrita, fala, audição, leitura e pensamento) [Bos89 p.5] [Loh89c] [Wei83] [Bar86]. Tais habilidades dificilmente são desenvolvidas na formação daqueles profissionais, o que dificulta uma efetiva Comunicação entre eles.

O treinamento também serviria para solucionar parcialmente o problema de "abismo semântico" entre Usuários e Analistas, na opinião de [DeB84].

Para o Analista, seria aconselhável ainda saber distinguir fatos de opiniões e desejos, examinando criticamente as informações que vão sendo coletadas, verificando se há outras interpretações possíveis para a situação, quantificando descrições (tendo cuidado especial com adjetivos como "perto", "rápido", "bom", "necessário", etc) e tornando sempre a conversação objetiva (por exemplo, utilizando as perguntas "em que isto se baseia?", "quantas vezes?", "como?", "o que se quer dizer com isto?", "os fatos realmente justificam esta conclusão?") [Nir81].

Ainda, quando informações forem fornecidas, deve-se levar em



conta o que os ouvintes já sabem, pois "as mesmas palavras estimulam diferentes imagens mentais" [Nir81], o que pode provocar interpretações erradas do que foi dito. [Wal76] sugere que se conheça a audiência e se selecionem, assim, os símbolos mais adequados para utilizar no discurso.

[Buy74] complementa declarando que, "para se comunicar normalmente, é preciso saber abstrair e concretizar".

Por fim, o uso de exemplos clareia o conteúdo da Comunicação [Nir81], tornando esta mais efetiva, pois, mesmo que Analistas e Usuários concordem sobre algo, ainda assim não se pode ter certeza de que um realmente entendeu o outro [Lan82]. [Sch81] confirma esta suspeita ao escrever que "o entendimento do Analista é melhor evidenciado pelo que fazem e não pelo que dizem". Os protótipos, neste caso, funcionam como exemplos daquilo que o Analista entendeu sobre o ambiente do Usuário.

Em resumo, os principais problemas de comunicação entre Usuários e Analistas são:

- baixa produtividade da comunicação;
- conflito de interesses (objetivos e funções) entre Usuários e Analistas;
- diferença de linguagens e conhecimentos entre Usuários e Analistas;
- falta de habilidades comunicativas entre os participantes;
- imprecisões das linguagens naturais (uso e compreensão de símbolos e mensagens).



### 3.3 Fase de Especificação

Especificações são programas escritos em uma linguagem de programação de alto nível, com alto grau de abstração [Bal78 p.94], e, ao mesmo tempo, representam a realidade em um modo altamente formal [Bal78 p.95].

Desta forma, uma especificação pode ser vista como um nível intermediário entre a realidade e a implementação executável da realidade, podendo ser a especificação transformada em ambas através de sucessivas abstrações ou refinamentos [Bal78 p.95].

Para [Bal78 p.94], uma especificação deve: definir completamente a funcionalidade do sistema, representar uma classe ampla de possíveis implementações e ser executável.

Para [Dav82 p.23], as especificações devem poder ser processáveis por outros programas para fins de:

- prova automática de programa (verificar se a implementação corresponde à especificação);
- síntese automática de programa (transformar a especificação em uma implementação processável em alguma máquina);
- simulação automática do sistema (verifica a funcionalidade e o tempo de desempenho);
- geração automática de teste de programa (testar se o sistema implementado se comporta de modo apropriado em certas situações).

Alguns autores (entre eles [Bor85], [Bal78 p.95]), acreditam que as linguagens de especificação de requisitos deviam ser passíveis de compreensão pelo Usuário, até mesmo permitindo a

este especificar sistemas. Há, porém, três problemas, pelo menos:

- linguagens de especificação devem ser formais, e estas, por natureza, são difíceis de serem entendidas e usadas;
- Usuários têm aversão a formalismos;
- as linguagens de fácil compreensão (como as linguagens naturais) são informais, não se prestando portanto para especificar requisitos, devido às suas ambigüidades, e por não poderem ser executadas ou verificadas automaticamente [Dav80].

Defende-se aqui a separação entre linguagem para documentação de requisitos (voltada para o Usuário) e linguagem para especificação de requisitos (voltada para Analistas e Projetistas), idéia que será explicada mais adiante.

Algumas linguagens e formalismos foram propostos para especificar requisitos, alegando-se possuírem as qualidades já discutidas em capítulo anterior.

[Dav82 p.22] fala em linguagens baseadas na abordagem axiomática, entre elas Alphard [Sha77], Gypsy [Amb77], CLU [Lis77], Special [Rob77] e Simula [Dah66], mais os trabalhos de Guttag [Gut77] e Hoare [Hoa69]. Esta abordagem inclui os conceitos de abstração de dados, abstração procedural e asserção de programa, sendo considerada boa para programadores e projetistas.

O mesmo autor cita também as linguagens baseadas no modelo operacional, onde o comportamento do sistema é descrito em termos de uma máquina hipotética bem-definida, como os autômatos e as máquinas de estado (entre as linguagens, VDL [Weg72] e RTRL [Tay80]).

Estas linguagens, segundo [Dav82 p.23], podem ser analisadas automaticamente para prova, síntese, simulação e geração de teste do sistema.

Na mesma referência ainda, são analisadas as linguagens PSL [Tei77], IORL [Eve80] e RSL [Alf77]. As duas primeiras provêm meios de expressar a decomposição das funções do sistema em sub-funções, podendo ser precisamente especificadas as entradas e saídas de cada função. Os respectivos analisadores verificam a consistência entre as funções, no tocante ao fluxo dos dados. A linguagem RSL, por sua vez, permite a modelagem operacional do sistema como uma máquina de estados finitos. Ela também trata de aspectos de tempo, o que as anteriores não faziam, sendo assim útil para especificar requisitos de sistemas "real-time".

Em [Bal85], é analisada a linguagem de especificação de alto nível chamada Gist, defendendo-se que a mesma formaliza construtores usados na linguagem natural e se aproxima mais do modo como pensamos sobre processos, que é diferente do modo como escrevemos sobre processos.

Porém, especificações em Gist não são fáceis de serem lidas. Por isto, foi desenvolvido um parafraseador, o qual ajuda na leitura da especificação. Um problema de Gist ainda não solucionado é que esta linguagem ainda está muito longe de uma linguagem de especificação de baixo nível, o que dificulta a execução e a síntese automática de programas [Bal85 p.1265].

Em [Bos89 p.4], os diagramas de fluxo de dados (DFDs), como o proposto em [Gan79] são recomendados para documentar e modelar requisitos, no tocante a operações, e [Set86] e [Teo82] aconselham o modelo E-R de [Che76] para as estruturas de dados.

[Bal78 p.95] discute três formalismos para especificação de sistemas: conjuntos, especificação axiomática e bancos de dados relacionais.

Conjuntos são estruturas simples e, por isso, possíveis de serem implementados de várias formas. Na especificação axiomática, o comportamento é definido através de equações algébricas que se encadeiam, formando expressões complexas. Já as linguagens da abordagem relacional definem estados do comportamento do sistema como asserções sobre o banco de dados relacional, sendo as ações expressas como inclusões ou exclusões de tuplas sobre o banco de dados. Esta última é considerada mais apropriada pelo referido autor, por permitir a modelagem de objetos e operações em uma maneira mais natural, facilitando assim a participação do Usuário no processo de especificação.

Por fim, mas sem esgotar o tema, há as linguagens em lógica, que permitem especificar o sistema, partindo da idéia de que a especificação é uma "base de conhecimento" sobre o domínio ou realidade que é modelada pelo SI. Uma linguagem de lógica usada com esta finalidade [Cas87] permite, pelo seu grande poder de representação de conhecimento, especificar com muita precisão os aspectos estáticos do sistema, e, se for usada uma linguagem adequada [Cas87], especificar também os aspectos dinâmicos do mesmo (ver ainda [Loh89a]).

Quanto a ferramentas para validação e verificação de especificações, [Bal85] apresenta o parafraseador, para validação estática, e o avaliador simbólico, para validação dinâmica. Esta última ferramenta permite que testes parciais sejam gerados e realizados automaticamente, enquanto os comportamentos são

explicados em linguagem natural (pelo "natural language behavior explainer"). As possibilidades desta superam as dos provadores de teoremas, que só verificam os comportamentos esperados, e as dos interpretadores, que só provêem uma execução ineficiente e podem tornar-se cansativos quando se tenta testar todo o sistema (segundo [Bal85 p.1260]).

Resumindo, os principais problemas da fase de Especificação são:

- difícil validação do documento de Especificação pelos Usuários;
- difícil tarefa de formalização;
- falta de ferramentas de auxílio.

### 3.4 Transformação de Informal para Formal (formalização)

Um dos maiores problemas, talvez o maior, da fase de Especificação é o próprio processo de Especificação, ou seja, a transformação das informações de uma descrição (linguagem) informal para uma representação formal.

A "formalização começa com alguma idéia informal sobre o que o programa (ou sistema) deve fazer e termina com uma descrição formal (precisa) de o que o programa deve fazer, mas não como isto deve ser feito" [Bar85].

Apesar de muitas linguagens formais terem sido propostas, não há muitos trabalhos (ou métodos) sobre o processo de formalização. [Bab85] cita o problema de transformação da Especificação Informal para a forma de entrada apropriada para as ferramentas de Especificação Formal propostas na literatura.



Para [Bal78 p.97], o principal problema da formalização é interpretar corretamente uma Especificação Informal, pois, uma vez que o entendimento da Especificação é obtido, basta somente transformar a informação de uma forma para outra.

A análise do contexto, isto é, das informações implícitas é um dos fatores-chave para a interpretação correta de uma Especificação Informal. [Ken78] declara: a ambigüidade das palavras é resolvida pelos humanos pelo entendimento do contexto.

Para [Bal78 p.96], o uso de contexto simplifica o processo de especificação. Os documentos se tornam mais concisos, e apenas parte da Especificação fica explícita; "o resto está implícito e deve ser extraído do contexto". Isto permite direcionar a atenção para apenas algumas informações (aquelas explicitadas), o que torna o documento de Especificação mais legível e mais compreensível.

O problema é que o contexto, contendo informações adicionais para a Especificação e também as regras para a interpretação de linguagem informal, nem sempre é claro ou inteiramente conhecido.

Num processo de formalização, as informações implícitas do documento (ou Especificação) informal devem ser explicitadas no documento (ou Especificação) formal, e as regras para interpretação da linguagem formal devem ser precisamente conhecidas.

Para interpretar uma Especificação Informal, faz-se necessário analisar sua informalidade. Atualmente, e geralmente, esta análise é feita mentalmente pelo Analista de Sistemas, quando este elabora a Especificação Formal do Sistema. Mas já há esforços em busca de ferramentas e métodos que ajudem,

primeiramente, o Analista e que, depois talvez, desempenhem aquela tarefa.

Uma das maneiras de melhor tratar a informalidade é defini-la mais claramente, estudando seus padrões e o modo como os Analistas a tratam.

[Bal78 p.96] propõe que a informalidade seja introduzida (colocada) no computador e, com a ajuda deste, aquela possa ser melhor tratada para a obtenção da Especificação Formal, bem como, possam ser definidos os procedimentos pelos quais o Analista transforma as informações em modo informal para a Especificação Formal.

Uma das formas em que se apresenta a informalidade são os textos em linguagem natural. Esta é talvez a forma mais natural e simples de introduzir a informalidade no desenvolvimento do sistema e é utilizada por [Cer83], [Abb83], [Bal78], [Bou85].

Uma vez que a informalidade tenha sido capturada (Especificação Informal), pode-se passar ao processo de formalização.

Assim como o processo de programação (codificação), a formalização é uma atividade difícil, informal, cansativa, sujeita a erros e não-documentada [Bal83 p.39] [Bal85 p.1258], necessitando, portanto, mais estudos para melhorá-la.

Já que o entendimento da Especificação Informal possui um papel importante no processo de formalização, uma das idéias é aumentar a participação do Usuário, tentando introduzir mais e melhores conhecimentos [Bal78 p.97]. Também desta forma poderão ser documentados e, talvez, repetidos os mecanismos utilizados pelo Usuário ao interpretar uma Especificação Informal e



desenvolver uma Especificação Formal [Bal85 p.1265].

Como na programação [Bal85 p.1257], é útil dividir a formalização em fases ou etapas, sendo que algumas destas poderão ser parcialmente formalizadas, de modo que cada atividade individual possa ser controlada, e as decisões, documentadas [Bal83 p.43]. Através deste rígido controle, ter-se-á maior confiabilidade no processo de formalização, sendo válido o resultado mais pelo desenvolvimento (formal) do que pela examinação do produto final [Bal81] [Bal83 p.41].

Um dos caminhos pelo qual pode ser obtida a granularização da formalização é o uso de níveis de linguagens. [Bal85] utiliza esta idéia, propondo duas linguagens de especificação (uma de alto e outra de baixo nível) e estudando a possibilidade de uma conexão entre elas através de transformações. O problema é que foi necessário "baixar" o nível da linguagem de alto-nível, e ainda assim é grande o trabalho a ser feito na formalização [Bal85 p.1266].

Quanto a analisar as decisões tomadas e os procedimentos utilizados por quem faz a formalização, esta idéia tem sido muito difundida. Através disto, pode-se firmar o conhecimento empregado no processo de formalização, para ser usado em outras ocasiões [Bal85 pg.1265].

A seguir são descritos alguns comportamento encontrados na literatura, os quais procuram identificar padrões na Especificação Informal (textos em linguagem natural). Uma vez identificados, estes padrões são transformados, quase que univocamente, para elementos da linguagem formal. Este comportamento deve ter sido verificado repetidas vezes, o que

explica sua citação pelos referidos autores.

Em [Bal78 p.100,103], é reconhecida a existência de frases com informações sobre processos e frases sobre a estrutura dos dados.

[Jac83] declara que cada verbo da Especificação Informal (texto em linguagem natural) é uma possível ação e que cada nome é uma possível entidade na Especificação Formal.

No trabalho de [Abb83], a solução informal (em Linguagem Natural) para um problema é implementada, mapeando-se verbos para "procedures", nomes comuns para tipos de dados, nomes próprios para objetos, atributos para funções e assim por diante.

Já [Bal85 p.1258] afirma que as pessoas vêem o mundo em termos de objetos, relacionamentos entre estes e operações que podem ser feitas sobre aqueles. Estas informações devem ser identificadas na Especificação Informal e transpostas para a Especificação Formal.

Em [Mat87], são apresentados 3 métodos para o que foi chamado de modelagem estrutural, a qual serve para extrair conceitos de requisitos confusos:

- determinação de termos funcionais (que identificam funções);
- obtenção de módulos pela análise do fluxo de dados;
- montagem do esqueleto do programa pela análise do fluxo de controle.

Para tanto, deverão ser enumerados os termos, conceitos deverão ser extraídos deste e associações entre os termos deverão ser definidas.

[Lud82] aconselha o reconhecimento, em um texto, de

ocorrências de nomes, os quais poderão dar origem a objetos na Especificação Formal. É necessário também fixar conexões entre os nomes, caracterizando relacionamentos lógicos em um nível ainda informal.

Bustin (citado em [Maa89] sugere que verbos de um texto representam procedimentos da Especificação Formal.

[Abb83], em sua ferramenta, mapeia um verbo em um texto para um procedimento na Especificação Formal, e um atributo para uma função.

[Sim83] cita um programa que reconhece objetos, classes de objetos, relacionamentos entre eles e operadores que podem ser aplicados sobre eles em um texto em linguagem natural que serve como entrada para o programa.

[Set86] aconselha que se aprenda a ler a estrutura do mundo real nele próprio, deixando que este revele suas características. Desta forma, um verbo caracteriza um relacionamento entre entidades e um adjetivo um atributo de uma entidade; já um advérbio temporal seria um atributo do relacionamento, pois qualifica o verbo.

Por sua vez, [Cer83] utiliza as associações sintáticas como informações úteis para definir associações lógicas entre entidades.

No trabalho de [Bal78], podem ser encontradas boas idéias para interpretação do contexto e, conseqüentemente, para completar e formalizar uma especificação. Os textos em linguagem natural são analisados, procurando-se situações em que se caracterizem omissões de operandos, nomes, informações sobre seqüência de operações, parâmetros, limites em cláusulas

condicionais, etc. Também é feita uma verificação de fluxo de informações entre as operações, ainda descritas informalmente, para se descobrir a seqüência destas e a possível omissão de informação (análise produtor/consumidor). São também utilizadas algumas idéias mais específicas como, por exemplo, o reconhecimento de um nome plural indicando que uma operação deve ser repetida várias vezes. Para melhorar a análise do texto, são introduzidos parênteses, caracterizando os papéis e as relações sintáticos. As tarefas da chamada fase Lingüística da ferramenta apresentada no artigo não estão muito claras.

Ainda que o processo de formalização seja melhorado através de divisão em fases, identificação de padrões, etc, ele não deixará de estar sujeito a erros, uma vez que é uma tarefa trabalhosa e desempenhada por pessoas (que não são perfeitas) [Mir89] [Bal83 p.39] [Bal85 p.1258] [Bal78 p.95]. Os erros humanos são principalmente freqüentes em tarefas repetitivas e automáticas (que não envolvem criação, raciocínio ou decisão) ou que envolvem grandes volumes de informações.

Para tanto, há a necessidade de ferramentas que automatizem o processo de formalização ou partes deste, o que torna mais que desejável introduzir a informalidade no desenvolvimento de SIs, para que seja tratada de forma adequada.

Uma ferramenta (automatizada) de tal tipo agiria interativamente com as pessoas envolvidas no processo, desempenhando as funções metódicas, questionando o Usuário em busca de entendimento da Especificação Informal e documentando as decisões tomadas pelo Analista na elaboração da Especificação Formal [Bal78]. Isto permitiria ao Analista se concentrar mais

nos detalhes de alto nível, enquanto a ferramenta seria responsável pelas tarefas de baixo nível [Bal83 p.43]. Assim, o processo se tornaria mais confiável e mais simples [Bal78 p.97]. A viabilidade de tal ferramenta é discutida em [Bal78].

Uma ferramenta como a anteriormente idealizada funcionaria, a princípio, como um assistente na formalização, mas, à medida que conhecimentos fossem incorporados, ela também tomaria decisões. Para a concretização desta idéia, as atividades devem ser formalizadas, pelo menos em parte [Bal83 pg.42,43].

[Maa89] sugere que uma função específica para ferramentas deste tipo seria a de reconhecer quando frases falam sobre um mesmo assunto (por exemplo, avião, vôo, passageiros, etc) e encontrar os sujeitos, verbos e complementos destas frases. Exemplos de ferramentas deste tipo são encontrados em [Bal78], [Bou85] e [Abb83].

Cabe salientar que uma ferramenta que automatize completamente o processo de formalização ainda está longe de ser alcançada [Bal83] [Bal85], pois a informação não pode ser formalizada simplesmente por processos objetivos e determinísticos em um computador [Ken78].

Apesar da grande utilidade de ferramentas no processo de formalização, erros não deixarão de ocorrer [Bal78 p.99], uma vez que a informação, em sua essência real, é ambigua e subjetiva [Ken78], e interpretar corretamente uma Especificação Informal é um problema que também aflige os seres humanos [Bal78 p.97]. [Nis89] complementa: "quando alguém se move de uma descrição informal de uma aplicação para uma representação formal, erros são freqüentemente introduzidos devido ao entendimento incorreto

ou mal-formado das propriedades desejadas do sistema".

Uma solução para este problema é a validação feita pelo Usuário [Bal78].

Apesar deste obstáculo crucial para formalização, não se devem negligenciar as propostas aqui citadas, pois elas representam ganhos pequenos mas significativos.

Em resumo, os principais problemas da tarefa de formalização são:

- distância entre linguagens informais e linguagens formais;
- falta de ferramentas;
- dificuldades para interpretar linguagens naturais.

### 3.5 Necessidade de uma Linguagem Comum

O maior problema para quem escreve ou documenta requisitos é saber que linguagem usar [Dav82 p.21]. Atualmente, o documento de Especificação de Requisitos é lido por Usuários (para validação) e por Projetistas (para projetar o sistema) [Dav82 p.21], além, é claro, do próprio Analista, que o escreve. Há, porém, diferenças significativas entre as partes envolvidas.

O Usuário é um especialista numa aplicação particular, ou seja, no ambiente da Organização, enquanto o pessoal que desenvolve o SI é especialista em Engenharia de Sistema, Software e Hardware [Dav82 p.21].

O Usuário utiliza uma linguagem natural técnica, mais fácil para ele ler e expressar idéias. Já Analistas e Projetistas trabalham com linguagens formais, mais precisas e menos ambíguas que as linguagens naturais. Estes já estão familiarizados com a



sintaxe e semântica das linguagens formais utilizadas, o que facilita o uso destas linguagens [Dav82].

Se a Especificação de Requisitos fosse escrita em uma linguagem natural (inglês ou português, por exemplo), ela não poderia ser processada automaticamente, uma vez que ainda não há tecnologia para isto. Além disso, as linguagens naturais são ambíguas, o que pode provocar erros de interpretação e, conseqüentemente, de implementação do SI. A vantagem é que as linguagens informais são de fato um modo normal e familiar de comunicação, exigindo menos treino para uso [Bal78 p.96].

Por outro lado, se uma linguagem formal fosse escolhida para a escrita da Especificação de Requisitos, o Usuário teria dificuldades em ler e compreender tal documento, durante a validação do sistema, uma vez que estas linguagens são mais legíveis por pessoal treinado [Dav82]. Ao se preocuparem em ser formais, estas linguagens descartam os mecanismos que tornam as linguagens informais (como as linguagens naturais) compreensíveis [Bal85 p.1259]. O parafraseador da linguagem Gist [Bal85 p.1259] é uma ferramenta para tornar as linguagens formais mais legíveis.

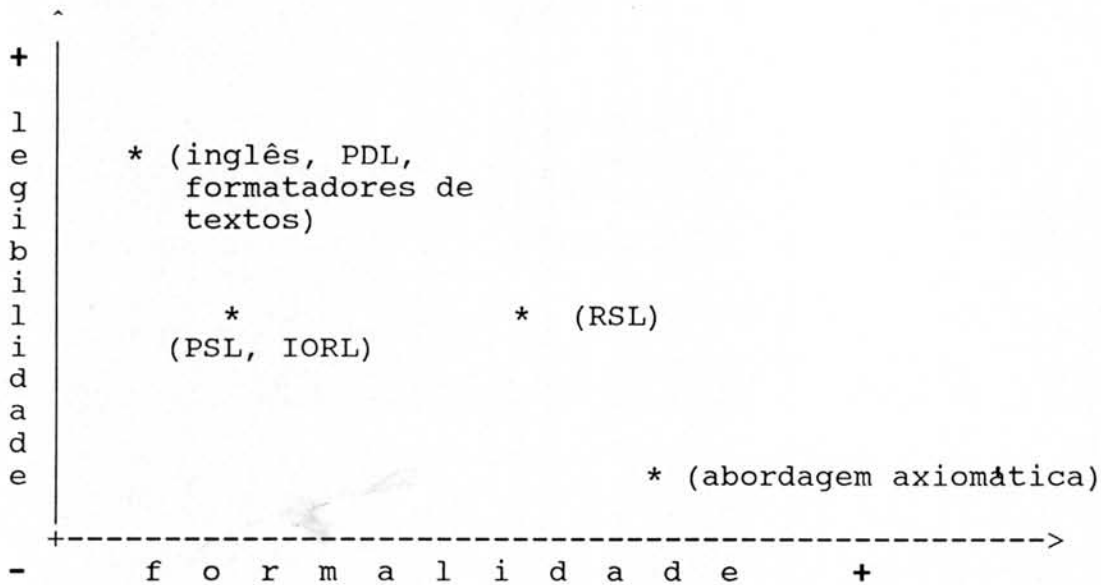
Outra diferença de documentos escritos em linguagens informais para aqueles escritos em linguagens formais é que os primeiros são mais concisos e necessitam de maior entendimento do seu contexto, enquanto que os últimos explicitam mais informações [Bal78 p.96], sendo, portanto, mais extensos, detalhados, complexos e cansativos.

Surge então a necessidade de uma linguagem intermediária entre linguagens informais e linguagens formais, ou como diz [Yeh84], de um domínio comum entre Usuários e Analistas.



[Lev82] sugere que uma linguagem para especificação de requisitos deve ser precisa e natural, ao mesmo tempo, para atender a Usuários e Analistas-Projetistas, e [Bos89 p.3] acrescenta que tal linguagem deve integrar as visões de ambos.

[Dav82] explica com um gráfico:



O ideal seria uma linguagem que se encontrasse no canto superior direito do gráfico.

Algumas tentativas foram feitas para alcançar tal propósito.

[Dav82 p.22] cita alguns trabalhos que usaram um subconjunto bem-definido e relativamente não-ambíguo do Inglês, buscando alcançar uma linguagem bastante legível e com o rigor matemático das linguagens formais. Porém, documentos assim escritos ainda não poderiam ser verificados automaticamente.

[Dav82] propõe um linguagem para sistemas interativos ou de tempo real, na qual alguns construtores sintáticos da linguagem natural foram empregados de modo formal. A estes construtores eram acrescentados termos-padrão (nomes e verbos) da aplicação.

Assim o Usuário vê um texto em Inglês técnico, enquanto o pessoal de computação vê um programa. Para o projeto desta linguagem foi usada a seguinte filosofia geral: suprir tão ricamente quanto necessário um conjunto de construtores, sem alterar o conceito de o que é "natural" (isto é, de acordo com as regras geralmente aceitas de uso do Inglês) [Dav82 p.25].

[Eps85] adverte que as restrições feitas às linguagens naturais podem trazer problemas para que o Usuário formule expressões. Por isto, devem ser feitas somente restrições que os Usuários possam aprender rapidamente e lembrar facilmente.

[Eps85] lembra que a linguagem restrita é uma linguagem artificial. Porém há as defesas de:

[Buy74] "qualquer língua é artificial, já que devemos aprendê-la" e [Rob] "as novas linguagens, em vez de destruírem as antigas, lhes servem de estimulante e fator de esclarecimento".

As novas linguagens e palavras e as extensões sobre a linguagem natural têm o propósito de tornar a comunicação mais clara e mais precisa [Mil76]. Esta necessidade de uma comunicação mais clara e mais rápida foi um dos fatores que determinou a evolução da própria linguagem natural [Hae].

A invenção de linguagens (como as linguagens lógicas, matemáticas e de computador) facilita a comunicação de idéias em certas situações ou problemas [Mil76]. Por esta razão, grupos específicos passaram a adotar linguagens próprias, adequadas para seus interesses [Loh89c].

Atualmente os modelos gráficos (diagramas) são utilizados como linguagem comum entre Usuários e o pessoal que desenvolve o SI, sendo o modelo E-R de [Che76] o mais aceito [Lev82] [Set86]

[Chr87] para representar a estrutura dos dados e o DFD (diagrama de fluxo de dados) de [Gan79] o mais empregado para descrever os processos (operações) do SI.

Apesar de bastante utilizados (talvez por falta de algo melhor), os modelos apresentados não são bons para Comunicação entre Usuários e Analistas (apesar de úteis para documentação), pois são somente voltados para Analistas e Projetistas [Lei87] [Bos89 p.4].

Além disto, modelos gráficos muito complexos dificultam ao Usuário compreender o que está representado (é o caso de um modelo E-R estendido contendo várias interconexões) [Teo89].

[Bar86] afirma que o Analista deve expressar as idéias de modo compatível com o estilo do Usuário, e [Sen83] complementa declarando que o Usuário não precisa entender os símbolos dos modelos, mas apenas o conteúdo destes. [Lei87] sugere, então, a descrição do modelo em um sub-conjunto da linguagem natural, e [Nos88] vai mais longe ao aconselhar que os Usuários escrevam textos em linguagem natural e os Analistas modelem o SI também em linguagem natural.

[Ken78] finaliza: os modelos conceituais devem ser baseados tão perto quanto possível do modo humano de perceber a informação.

A busca de uma linguagem comum precisa e natural, ao mesmo tempo, resulta em linguagens que não são nem precisas nem naturais, ou tem parcialmente uma ou outra destas propriedades.

Uma das alternativas é buscar uma linguagem comum que não desconsidere o uso de linguagens informais e linguagens formais.

[Mey85] apóia dizendo que as Especificações de Requisitos em

linguagens formais não são substitutas de Especificações em linguagem natural, mas sim um complemento para melhorar a qualidade destas especificações.

[Dav82] fala em vários graus (ou níveis) de formalidade e legibilidade, [Bal85 p.1266] concretiza em parte esta idéia ao propor a aproximação de suas duas linguagens de especificação (uma de alto e outra de baixo nível), a fim de obter um meio termo entre precisão e naturalidade.

Assim ter-se-iam vários níveis de linguagens, cada nível com certo grau de formalidade e legibilidade. A medida que se avançam pelos níveis, as linguagens perdem em um grau e ganham em outro, como no seguinte desenho:

Níveis	1	2	...	N-1	N
Linguagens	natural, imprecisa	um pouco natural que a anterior; um pouco mais precisa que a anterior	menos natural que a anterior	menos natural mais precisa	formal precisa

A função destas linguagens intermediárias seria a de explicitar mais informações do que a linguagem do nível anterior, dando maior precisão e formalidade (diminuindo as ambigüidades). Isto, naturalmente, implicaria em perda de legibilidade.

Neste trabalho, propõe-se uma linguagem que agrega os níveis 2 a N-1 num único nível, intermediário.

Em resumo, fica clara a necessidade de uma linguagem (ou várias) intermediária entre as linguagens informais e formais, que aproxime estas, controlando perdas e ganhos em legibilidade (ou naturalidade) e formalidade. Com isto, melhora-se o processo

de formalização de especificações e fica mais fácil a comunicação entre Usuários e Analistas de Sistemas para coleta e validação de informações. Esta linguagem comum ou intermediária não deve ser gráfica, pois modelos gráficos muito complexos dificultam o entendimento das informações descritas.

## PARTE II: PROPOSTA DE LINGUAGEM COMUM

Para tentar solucionar os problemas vistos anteriormente e devido à necessidade já referida, foi desenvolvida uma proposta de Linguagem Comum que busca conciliar as visões e pontos-de-vista de Usuários e Analistas e que, ao mesmo tempo, atinja um meio-termo entre naturalidade e precisão, ou seja, que esteja próxima, por um lado, de linguagens informais, usadas para documentar os requisitos do SI (fase de AR), e, por outro, de linguagens formais, usadas para especificar os requisitos do SI (fase de Especificação).

## 4 FUNDAMENTOS DA PROPOSTA

Pretende-se, a seguir, fundamentar a proposta da Linguagem Comum.

### 4.1 Linguagem Comum

Deseja-se que a Linguagem Comum proposta aqui facilite a comunicação Usuário-Analista, pois se supõe que ela seja conhecida de ambos, já que utiliza os mesmos conceitos das linguagens naturais, não requerendo esforço para ser aprendida. Supõe-se também que ela não possui as muitas ambigüidades e imprecisões das linguagens naturais, nem é complexa, não-familiar e cansativa como as linguagens formais.

Isto facilita em muito o entendimento entre Usuários e Analistas nos seus encontros e reuniões, sendo assim instrumento de produtividade no desenvolvimento de SI's.

Por ser simples, utilizando-se de frases simples e curtas, espera-se que a Linguagem Comum torne clara a interpretação do que foi documentado (escrito) e diminua a ambigüidade, uma vez que "quebra" as frases complexas (comuns nas linguagens naturais), explicitando as relações entre os termos.

Este processo de trazer à luz informações antes implícitas ou discretas (pouco visíveis) simplifica o conteúdo do texto e segue o conselho de [Nir81] para que não se expressem idéias complexas em poucas palavras, mas que se procure dividi-las em idéias mais simples (usando frases mais simples e dividindo o discurso em unidades).



Apesar de diferente das linguagens naturais, supõe-se que a Linguagem Comum esteja próxima e seja semelhante a estas, o que se tentará mostrar com as heurísticas de transformação entre as linguagens.

Na verdade, a Linguagem Comum é uma simplificação, um subconjunto da linguagem natural. Portanto, não deixa de apresentar uma certa naturalidade, o que pode ser notado ao verificar-se que a comunicação natural entre as pessoas se utiliza, em parte, da própria Linguagem Comum, apesar das restrições desta (a comunicação por linguagens naturais utiliza, é óbvio, muito mais elementos que a simples Linguagem Comum).

Outra grande vantagem nisto é que a Linguagem Comum proposta permite a participação do Usuário. Seja validando ou completando (ou até mesmo escrevendo) a documentação de requisitos, o Usuário não necessita grande esforço ou gasto de tempo para entender e aprender a usar a Linguagem Comum. É claro que as restrições da Linguagem Comum precisam ser aprendidas, mas a própria linguagem se utiliza de construções da Linguagem Natural.

Por outro lado, a Linguagem Comum também está próxima da linguagem formal (no caso deste trabalho, a Linguagem Formal assumida).

Tal qualidade permite que os requisitos do SI possam ser documentados formalmente, ou seja, o processo de formalização é facilitado com o uso da Linguagem Comum.

Através de suas relações sintáticas, a Linguagem Comum explicita as relações entre os elementos, além, é claro, de facilitar a identificação dos próprios elementos.

O uso de padrões (frases-padrão), de construtores da

linguagem natural e de frases estruturais e sobre processos também permite uma rápida e melhor identificação dos elementos a serem formalizados na linguagem formal.

Além disso, a Linguagem Comum segue o conceito de granularização [Bal83 p.43], isto é, dividir as atividades em passos menores e mais formais, de forma a controlar melhor o seu desempenho.

E isto permite que algumas tarefas sejam automatizadas ou, pelo menos, auxiliadas por ferramentas automatizadas.

Uma estratégia assim permite a verificação das atividades e de seus resultados parciais, tornando quase formal (grande parte, pelo menos) o desenvolvimento de especificações de SI's.

Assim, portanto, o Analista pode ser ajudado nesta difícil tarefa, e a probabilidade de erros diminui, pois apenas alguns passos não são formais, sendo ainda dependentes da experiência do Analista.

Entretanto, cabe salientar que a Linguagem Comum proposta aqui não substitui as linguagens informais, nem as linguagens formais, mas sim lhes complementa seu valor e função. É errado, portanto, tentar usá-la sozinha como linguagem para captar a informalidade e, ao mesmo tempo, para especificar formalmente, SI's. A Linguagem Comum não é natural o suficiente como as linguagens informais, nem precisa o necessário como as linguagens formais.

A Linguagem Comum é, outrossim, um meio intermediário entre o modo como pensamos sobre e o modo como escrevemos sobre sistemas, como quer [Bal85 p.1258], não sendo, portanto, nem um, nem outro.

Apesar de ser um sub-conjunto da linguagem natural, a Linguagem Comum não pode ser considerada natural. Ela é uma convenção (uma linguagem específica) própria para uma determinada situação-problema. No caso, ela é uma simplificação da linguagem natural (o português), justamente para diminuir as imprecisões desta e permitir uma melhor comunicação entre pessoas envolvidas neste determinado ambiente e situação.

Nesta mesma linha de raciocínio, a Linguagem Comum proposta foi desenvolvida tendo em vista uma classe de linguagens informais e linguagens formais e foi testada somente com as linguagens informal e formal assumidas neste trabalho.

Por estar no meio do difícil caminho entre informalidade e formalidade e por representar um meio-termo entre naturalidade e precisão, a Linguagem Comum proposta não segue algum modelo formal e não foi testada para muitos sistemas, não podendo assim ser considerada completa ou perfeita. Por sua vez, a mesma se mostrou adequada e de grande valia nos sistemas em que foi avaliada. E como declara [Sim83 p.26]: "o fato de que os modelos representam o mundo real imperfeitamente e incompletamente não os faz inúteis".

Além disto, espera-se que a Linguagem Comum substitua com melhorias os modelos até então utilizados para comunicação entre Usuários e Analistas e para documentar requisitos.

#### **4.2 Dicionário de Termos**

Um Dicionário de Termos é proposto para complementar a Linguagem Comum. Nele deverão ser definidos e explicados os

termos utilizados por Usuários e Analistas e, conseqüentemente, presentes na Linguagem Comum.

Também deverão ser registrados os termos (ou palavras ou expressões) que são sinônimos, bem como aqueles considerados de mesmo contexto (formas ativa e passiva dos verbos, verbos e seus respectivos substantivos, adjetivos e as orações adjetivas que os substituem).

O objetivo primeiro de tal Dicionário é diminuir o "abismo semântico" [Bos89] que há entre Usuários e Analistas. Pela definição clara dos termos e desmistificação dos jargões, a comunicação se tornará mais clara, precisa e efetiva.

Com este instrumento, os termos poderão ser padronizados ou relacionados de forma que possam ser identificadas as partes do texto que "falam" a mesma coisa mas de modos diferentes. Isto permite a unificação ou conciliamento de pontos-de-vista de pessoas diferentes, como aconselha [Bos89], e pode ser conseguido pelo registro de sinônimos ou pelo uso de paráfrases (definição de um conceito utilizando outros já definidos [San]).

Também poderão ser definidos no Dicionário (e utilizados na Linguagem Comum) termos novos quando se fizer necessário significar conceitos mais específicos, característicos e próprios da situação em questão.

É objetivo também do Dicionário precisar e padronizar termos que possuem múltiplos significados. Isto é muito comum [Dav82 p.22], uma vez que o significado de um termo é um conceito arbitrário [Loh89c] e depende de quem o usa, onde, quando e por que (segundo Wittgenstein citado em [Eps86]).

Supõe-se que o uso de um Dicionário de Termos facilite a

comunicação e aumente a produtividade no desenvolvimento de especificações de SI's, porém ele não deve ser muito grande e deve ser facilmente consultado.

#### **4.3 Linguagem Informal**

Para o trabalho em questão, assumiu-se como Linguagem Informal a Linguagem Natural (no caso, o português) em sua forma escrita.

A razão para tal escolha é que se presume serem os textos em Linguagem Natural a forma mais simples de introduzir a informalidade (informações em estado informal) no desenvolvimento de SI's.

Além disto, esta é a maneira mais fácil de conseguir que os Usuários participem no processo e forneçam informações.

Já a forma escrita é particularmente útil porque obriga quem escreve a organizar seus pensamentos e fornecer informações de modo que outras pessoas possam entendê-los mais facilmente.

Portanto, sempre que se referenciar Linguagem Informal, assume-se a Linguagem Natural (português) escrita como representante de tal. Isto não quer dizer que a Linguagem Comum proposta não possa ser utilizada com outras linguagens informais. Porém, os resultados obtidos talvez não sejam os mesmos conseguidos com a Linguagem Informal assumida neste trabalho.

#### **4.4 Linguagem Formal**

Como a Linguagem Comum proposta aqui pretende estar próxima

às linguagens formais, é preciso que ela abranja várias classes e tipos de linguagens formais. Entretanto, esta é uma tarefa difícil e tal objetivo bastante improvável de ser atingido no momento.

Por isto, procurou-se restringir a classe de linguagens formais a uma linguagem específica que pudesse ser testada com a proposta.

Assumiu-se, então, como Linguagem Formal, uma linguagem baseada em lógica, representando dados, sua estrutura e relações, procedimentos (operações) e restrições de integridade.

Assim, tem-se uma Linguagem Comum "base", definida e testada, para ser expandida e melhorada, mais tarde, para abranger também outros tipos de linguagens formais.

Esta estratégia foi escolhida devido à outra alternativa (pesquisar uma Linguagem Comum que fosse universal, isto é, que abrangesse todos os tipos de linguagens formais) ter sido avaliada como de difícil realização no momento, apesar de uma linguagem comum universal ser de grande valia. Isto fica como objetivo para estudos posteriores.

Partir de uma classe restrita de linguagens formais é uma alternativa viável a curto prazo e permite que se chegue logo a resultados que possam medir o andamento do trabalho.

A Linguagem Formal assumida tem como principais características proporcionar apenas uma interpretação de seus conceitos e ser independente de implementação.

Dividiu-se a Linguagem Formal em 3 partes para que cada uma pudesse especificar melhor um dos 3 aspectos de um SI, a saber:

- dados, sua estrutura e relações;



- operações (ou procedimentos) sobre os dados;
- restrições sobre os dados.

Porém, esta divisão não isola as partes. Pelo contrário, há comunicação e redundância entre elas, de forma que as informações documentadas possam sempre ter sua consistência verificada sobre as 3 partes, ao mesmo tempo.

Cada parte é detalhada a seguir.

#### 4.4.1 Parte de Dados e Estrutura

Foi escolhida como linguagem para especificação dos dados e suas estruturas, uma linguagem em lógica, definida como a seguir:

A linguagem assumida possui como alfabeto o seguinte conjunto de símbolos:

- símbolos funcionais e predicativos (nomes que começam com letras minúsculas);
- variáveis individuais (nomes que começam com minúsculas);
- constantes individuais (começando com maiúsculas);
- conectivos lógicos: e, ou, não, --> (se-então); observação: "vírgulas" equivalem a "e";
- quantificadores:  $\exists$  e  $\forall$ ;
- parênteses e colchetes.

Os elementos ou conceitos do mundo real são referenciados por constantes ou variáveis e, geralmente, qualificados por um predicado.

Exemplo: predicado "vendedor"; referência ao elemento "João": vendedor(João), cuja interpretação é a afirmação "João é vendedor".

As associações entre os elementos são representadas por símbolos predicativos ou por funções (de dois argumentos), dependendo do contexto em que a associação aparece.

Se uma associação aparece como argumento de outra associação, ela deve ser interpretada como uma função. De outra forma, é interpretada por um símbolo de predicado.

Exemplo: associação "vendeu" entre vendedores e produtos: `vendeu(João, Carro)`, cuja interpretação é a afirmação "João vendeu um Carro", sendo que "João" deve ser um "vendedor" e "Carro" deve ser um "produto".

Outro exemplo: associação "em" definindo um adjunto adverbial de tempo para indicar a data da venda: `em( vendeu(João, Carro), 20/11/90)`, cuja interpretação é "João vendeu um Carro em 20/11/90", sendo "`vendeu(João,Carro)`" interpretada como um evento.

Neste último caso, a associação "vendeu" é tratada como função, tendo como resultado ou resposta (dos argumentos João e Carro) o evento "venda de um Carro feita por João".

A definição dos nomes depende de quem os escolhe. Porém, ela deve ser coerente ao longo da especificação do sistema. Por exemplo, a associação entre "vendedores" e "produtos" poderia ter sido, de outro modo, denominada de "venda" e a já referida associação designando o adjunto adverbial denominada "ocorrida\_em". O uso dos nomes deve ser padronizado mesmo sabendo-se que palavras ou expressões são sinônimas.

Esta parte da linguagem deve, portanto, definir os conceitos ("sortes") do ambiente sendo modelado e a estrutura deste (associações entre os elementos). É bom que estas definições

sejam feitas em linhas separadas, indicando assim as correspondências com as frases da Linguagem Comum que originaram estas linhas. Isto será útil para se fazer a paráfrase da especificação, como será vista adiante.

Um exemplo de linguagem para um Sistema de Vendas é dado a seguir:

- tipos ("sortes"), definindo os elementos do mundo real:

vendedor(x), produto(y), data(z), preço(w);

- associações:

vendeu(x,y) com "x" sendo do tipo vendedor e "y" produto;

tem(x,y) com "x" sendo do tipo produto e "y" preço;

em(x,y) com "x" sendo uma função do tipo "vendeu" e "y" do tipo "data".

Observação: nesta linguagem, só são permitidas associações binárias, isto é, entre dois conceitos ou argumentos, devido às restrições da própria Linguagem Comum. Portanto, relações com mais de dois argumentos devem ser transformadas.

Exemplo:

vende\_a(x,y,z), vendedor(x), produto(y), cliente(z) ==>

**Alternativa 1:**

vendedor(x), produto(y), cliente(z),

venda(w), participa\_em\_venda(x,w),

participa\_em\_venda(y,w),

participa\_em\_venda(z,w)

**Alternativa 2:**

vendedor(x), produto(y), cliente(z),

vende(x,y), a(vende(x,y), z)

ou considerando a relação "vende" como uma "venda"

venda(x,y), venda\_feita\_a(venda(x,y), z)

**Alternativa 3:**

Considerando como "venda" o par produto, cliente:

vendedor(x), produto(y), cliente(z),

venda(y,z), feita\_por(venda(y,z), x).

#### 4.4.2 Parte de Operações

A parte de operações da Linguagem Formal assumida é baseada na linguagem em lógica e é definida como a seguir.

Existem três tipos de operações: de manipulação, de consulta e mistas.

O primeiro tipo utiliza o enfoque de especificação de operações em termos de pré e pós-condições de [Hoa69].

Os fatos descritos nas pré-condições devem ser verdadeiros para que a operação seja bem sucedida. Como consequência da operação, deverão ser verdadeiros os fatos definidos nas pós-condições. Estas, na verdade, descrevem as alterações que as informações sofrem com a operação. Assume-se que os fatos não alterados pelas pós-condições continuam valendo (sendo verdadeiros), o que exclui a necessidade de reafirmá-los nas pós-condições.

É necessário, nas operações de manipulação, listar os argumentos de entrada (aqueles obviamente necessários para que a operação seja efetuada), como nas "procedures" das linguagens de programação. Estes argumentos caracterizam as informações que devem ser fornecidas pelo Usuário do SI. Assume-se que as demais informações utilizadas ou manipuladas pela operação e que não são

definidas como argumentos de entrada estejam prontamente disponíveis ao SI, isto é, o Sistema de Informação tem conhecimento de tal informação em algum meio (provavelmente um Banco de Dados ou de Conhecimentos).

Entretanto, é possível definir alguns argumentos da lista de entrada como opcionais, não precisando ser fornecidos pelo Usuário (deve-se colocar o símbolo "opc", após o nome do argumento). Neste caso, os termos das pré e pós-condições que se referirem àqueles argumentos opcionais serão simplesmente desconsiderados.

Na parte de operações da Linguagem Formal, é permitido também o uso de expressões matemáticas (por exemplo,  $x=(y+z).(w-s)/k$ ), de algumas constantes básicas (como, um\_mês, uma\_hora, duas\_semanas, três\_dias, etc) e de operadores especiais (total\_de, média\_de, menos\_de, maior\_de, etc).

Forma geral das operações de manipulação:

nome\_da\_operação(lista\_de\_argumentos):

pré-condições: fatos.

pós-condições: fatos.

Sendo que "fatos" são um conjunto de predicados (ou um só) conectados logicamente com "e" ou "ou". Porém, não é permitido o uso do conector "-->" (se-então).

Observação: as variáveis são consideradas sempre globais, isto é, abrangendo tanto pré como pós-condições, quando não especificado.

Exemplo:

cadastrar\_nome(x):

pré-condição: não[ nome(x) ].

pós-condição: nome(x).

Observação: deve-se considerar que esta especificação da operação é uma forma mais simples de se dizer:

pré-condição:  $\forall y$  não [ nome(y) e  $y=x$  ].

pós-condição:  $\exists y$  nome(y) e  $y=x$ .

No caso, assume-se que a variável individual "x" será instanciada com um valor fornecido pelo Usuário do Sistema de Informação, quando do início da operação, e passará, então, a ser tratada como uma constante individual (razão por que ela não vem quantificada).

Outros exemplos:

cadastrar\_pessoa(n):

$\forall p1 \exists p2$

pré-condição: não [ pessoa(p1) e nome(n) e tem(p1,n) ].

pós-condição: pessoa(p2) e nome(n) e tem(p2,n).

mudar\_endereço\_de\_pessoa(n,e):

$\exists p$

pré-condição: pessoa(p) e nome(n) e tem(p,n).

pós-condição: endereço(e) e tem(p,e).

O segundo tipo de operações diferencia-se do primeiro por possuir argumentos de saída, ou seja, necessariamente fornece uma resposta, e também por não possuir a parte de pós-condições, justamente por não interferir nos dados. No mais, possui as mesmas características que o primeiro tipo.

Forma geral das operações de consulta:

nome\_da\_operação(lista\_de\_argumentos\_de\_entrada):

lista\_de\_argumentos\_de\_saída.

pré-condições: fatos.



Exemplo:

consultar\_nomes\_cadastrados\_em\_tal\_dia(d): x.

pré-condições:

$\exists x, d$  nome(x) e dia(d) e cadastrado\_em(x,d).

Observação: a representação dos dados nas operações deve estar coerente com a representação dos mesmos dados na primeira parte da linguagem. Assim no exemplo anterior, o conceito "cadastrado\_em" deve ter sido estabelecido como uma associação entre dois argumentos, sendo o primeiro deles uma entidade do tipo "nome" e o segundo do tipo "dia".

Já as operações mistas (do terceiro tipo), combinam características de operações de manipulação e de operações de consulta.

Forma geral das operações mistas:

nome\_da\_operação(lista\_de\_argumentos\_de\_entrada):

lista\_de\_argumentos\_de\_saida.

Pré-condições: fatos.

Pós-condições: fatos.

Nas operações ainda, foi estabelecido um conceito novo de "identificador". Ele se comporta como uma generalização dos conceitos que podem identificar elementos distintos do mundo real. Por exemplo, considerando o conceito Pessoa com atributos Nome, Endereço, Telefone, Número da Carteira de Identidade, CPF e Data do Nascimento, tem-se que CPF e Número da Carteira de Identidade são identificadores do conceito Pessoa (são únicos para cada pessoa), uma vez que não há dois ou mais elementos deste tipo associados a um mesmo valor para os conceitos identificadores.

Este novo conceito é bastante útil (e simplifica o trabalho de especificação de uma operação) nos casos em que é necessário utilizar um dos argumentos de entrada como identificador de uma ocorrência.

Exemplo:

consulta\_endereço\_de\_pessoa(identificador): y.

† y

pré-condição: pessoa(identificador) e  
endereço(y) e tem(identificador, y).

No exemplo, o Usuário poderia fornecer o CPF ou o número da carteira de identidade da pessoa cujo endereço se quer conhecer ou, até mesmo, o seu "surrogate", se este for conhecido.

A tradução do exemplo seria como a seguir:

consulta\_endereço\_de\_pessoa(id): y.

† y, z

pré-condições:

endereço(y) e  
( [ pessoa(id) e tem(id,y) ] ou  
[ cpf(id) e pessoa(z) e tem(z,id) e tem(z,y) ] ou  
[ n\_cart\_ident(id) e pessoa(z) e tem(z,id) e  
tem(z,y) ] ).

#### 4.4.3 Parte de Restrições

Só há uma forma para representar restrições na Linguagem Formal assumida:

fatos\_1 --> fatos\_2

(interpretação: se fatos\_1 então fatos\_2),

sendo que fatos\_1 e fatos\_2 devem ser um conjunto de um ou mais predicados (conectados e quantificados logicamente), e fatos\_2 ainda pode vir negado (fatos\_1 -> não fatos\_2).

Nas restrições, devem ser descritas as informações sobre os mapeamentos das associações (1:1, 1:N, N:M) e as informações sobre obrigatoriedade de participação dos elementos nas associações.

Exemplos:

Restrição de tipos:

$\forall x,y \text{ tem}(x,y) \rightarrow \text{empregado}(x) \text{ e } \text{dependente}(y)$ , cuja interpretação é "a associação 'tem' associa um empregado a um dependente".

Restrição de mapeamento:

$\forall x,y,z \text{ tem}(x,y) \rightarrow \text{não } [ \text{tem}(z,y) \text{ e } z \neq x ]$ , cuja interpretação é "um dependente só pode estar associado a um empregado".

Restrição de obrigatoriedade de participação na associação:

$\forall y \exists x \text{ dependente}(y) \rightarrow \text{tem}(x,y) \text{ e } \text{empregado}(x)$ , cuja interpretação é "um dependente deve estar associado a um empregado".

Outras restrições também podem e devem ser especificadas nesta linguagem.

Exemplo: usuário da biblioteca que retirou periódico não pode retirar documento.

$\forall x,y,z$

$\text{usuário}(x) \text{ e } \text{periódico}(y) \text{ e } \text{retirou}(x,y) \rightarrow$

$\text{não } [ \text{retirou}(x,z) \text{ e } \text{documento}(z) ]$ .

Observação: deve-se ter um cuidado especial com os nomes

"tem" e "em" designando associações (relações), pois podem associar vários tipos de conceitos. Exemplos: pessoa e nome, pessoa e endereço, venda e data, venda e local, etc.

#### **4.5 Paradigma Transformacional**

A Linguagem Comum aqui proposta segue uma linha de pensamento e pesquisa que procura ver o desenvolvimento de SI como sucessivas transformações de informações de um estado para outro, desde a Análise de Requisitos até que seja alcançada uma implementação [Bal83] [Bal85].

Esta nova visão prevê a divisão do processo de desenvolvimento em etapas menores, que possam ser controladas e desempenhadas formalmente, com a ajuda do computador ou por este mesmo.

Para tanto, deverão ser desenvolvidas especificações formais que sejam executáveis e, como tal, funcionem como protótipos para o Usuário validar.

Após, as especificações são otimizadas, gerando uma implementação do sistema em uma linguagem de programação. As decisões tomadas seguiriam os conceitos de eficiência utilizados pelos programadores atuais [Bal83].

Estas transformações, de uma forma para outra, preservam algumas características (semântica) e alteram outras (performance), o que avaliza o processo [Bal81].

Além de melhorar a produtividade do desenvolvimento, estas idéias contribuem para aliviar alguns problemas de manutenção de sistemas, a qual passaria a ser feita diretamente na

especificação. Esta, por sua vez, seria recompilada gerando um novo código, mais otimizado que o anterior. [Bal85 p.1263] [Bal78 p.97].

Supõe-se que o tempo de desenvolvimento e o número de erros sejam menores neste novo paradigma. Porém, esta vantagem está condicionada à utilização de ferramentas automatizadas que auxiliem no processo de desenvolvimento.

#### 4.6 Heurísticas

Para as transformações entre as linguagens Informal, Comum e Formal, são utilizadas heurísticas para guiar e facilitar aquelas tarefas.

O uso de heurísticas se faz necessário e útil em atividades pouco estruturadas, que requerem inteligência e que, portanto, não podem se automatizadas (modeladas por algoritmos).

Para Bruner (citado em [Puc69] existem 2 tipos de raciocínio:

- analítico: aquele que possui etapas isoladas, nitidamente concebidas e objetivadas pelo homem, podendo ser expressas por palavras;
- intuitivo: aquele que busca soluções sem que haja consciência do processo.

Para o primeiro tipo, são utilizados algoritmos, os quais "asseguram a resolução do problema, desde que obedecidas todas as suas regras e etapas".

No segundo caso, os algoritmos não se prestam devido à complexidade do problema e de sua resolução. Assim são utilizados

processos heurísticos, que são "meios pouco exatos de solução de problema, com cujo auxílio poder-se-á ou não lograr o resultado almejado".

Para [Len82 p.192,195,223], "heurística é um pedaço de conhecimento capaz de sugerir ações plausíveis a seguir ou ações implausíveis a evitar". Heurísticas nada mais são que "regras informais de julgamento que guiam as pessoas numa rápida tomada de decisão". Desta forma, as heurísticas conduzem ao objetivo em menos tempo, através de melhores escolhas.

Já [Puc69] define Heurística como a "ciência que estuda as constantes da atividade do pensamento criador". Para o mesmo autor, a atividade heurística é a "geratriz de novos sistemas de ação, a qual desvenda as constantes dos objetos à volta do homem e que até então permaneciam desconhecidos".

Para Poia (citado em [Puc69], "o objetivo da Heurística é a pesquisa de regras e métodos que levem às descobertas e às invenções; a experiência pessoal mais a observação do modo pelo qual outras pessoas os resolvem devem ser a base do método para a solução de problemas".

A partir deste ponto, a fim de limitar o campo de estudo e uso da palavra "heurística", passarão a ser chamadas simplesmente de **heurísticas** as regras informais utilizadas nos processos heurísticos e que ajudam na solução de um determinado problema, porém não garantindo a sua resolução (conforme uso feito por [Len82]).

O estudo de heurísticas ou atividades heurísticas é viável e já começa a dar resultados. [New76 p.119,121] explica que as pesquisas sobre a psicologia de processamento de informação



envolvem 2 tipos de atividade empírica:

- conduzir observações e experimentos com comportamento humano em tarefas que requerem inteligência;
- programar o sistema de símbolos para modelar o comportamento humano observado.

[New76] exemplifica com o programa GPS (General Problem Solver) que utiliza "idéias derivadas de cuidadosa análise de protocolos que os humanos produzem enquanto pensando durante a tarefa de solucionar problemas".

[Len82] também cita um programa que coleta dados empíricos, nota regularidades neles e define novos conceitos.

[Len82] dá um exemplo de atividade heurística: analisar o jogo ótimo de Blackjack, construir uma tabela de ações apropriadas (heurísticas) e tomá-la como estratégia básica; após, avaliar a sua performance. O mesmo autor conclui: heurísticas podem ser construídas através de uma exaustiva e sistemática pesquisa dos procedimentos de ação.

A força de uma heurística está na sua capacidade de generalização: a heurística é boa se a generalização feita foi boa [Len82 p.190]. Por exemplo, se uma heurística H foi útil numa situação S, então é provável que heurísticas similares a H sejam úteis em situações similares a S [Len82 p.190].

Disto conclui-se que uma determinada heurística é específica para um determinado domínio, sendo que sua força pode diminuir em outros domínios [Len82 p.210,211].

Portanto, o uso e a pesquisa de heurísticas são adequadas para modelar mundos que são: [Len82 p.246]

- observáveis;

- estáveis (as heurísticas funcionam independente do tempo - passado, futuro, presente);
- contínuos (onde certas ações são boas em uma situação S e ações similares àquelas também são boas em situações similares a S).

[New76] complementa afirmando que são necessários graus mínimos de ordem e padrão e que estes sejam detectíveis.

Desta forma, o presente trabalho procurou identificar heurísticas nas tarefas de transformação entre linguagens e também na tarefa de coletar as informações iniciais, introduzindo assim a informalidade no processo de desenvolvimento para que seja adequadamente tratada.

A razão de tal escolha (por um processo heurístico) é que algoritmos não podem ser definidos para estas tarefas, uma vez que estas são pouco estruturadas e pouco tratáveis mecanicamente, além de apresentarem dificuldades para a identificação de seus padrões e ordem.

Assim, utilizando-se dos conselhos já citados, a referida atividade heurística procurou analisar a forma como as pessoas desempenham tais tarefas, identificando padrões, definindo-os e os avaliando em novas situações. A experiência e intuição do autor vieram complementar este conjunto inicial de heurísticas, mas os experimentos é que foram cruciais para tal trabalho.

Por fim, uma coisa é clara: as heurísticas descobertas e definidas não estão completas nem asseguram a solução do problema, devendo, portanto, serem aperfeiçoadas e complementadas com mais atividades heurísticas.

Outrossim, para a classe de situações-problema considerada

aqui, e salvas as restrições feitas, espera-se que as heurísticas a serem apresentadas sejam úteis, melhorando a execução de tarefas ao orientá-las, controlá-las e sistematizá-las.

#### **4.6.1 Heurísticas para Coleta de Informações**

Estas heurísticas têm por objetivo principal ajudar na captação das informações em estado informal e representá-las de forma que possam ser tratadas depois.

Analisando-se o modo como os Analistas coletam informações e identificando naquele alguns padrões, chegou-se a um conjunto inicial de heurísticas.

Tais heurísticas procuram modelar o comportamento de um Analista na referida tarefa, mas sabe-se que não de modo completo. Elas diminuem a dependência da tarefa à experiência do Analista. Direcionando a atividade e sugerindo ações, as heurísticas vão auxiliando o Analista e, às vezes, realizando tarefas sem a interferência deste. Isto libera o Analista de tarefas repetitivas e permite que este se preocupe mais com tarefas que exigem inteligência ou habilidades humanas que as máquinas não podem, até agora, simular, tais como aspectos psicológicos, comunicativos, emotivos, etc.

Supõe-se que a coleta de informações seja feita em menos tempo (através de modos mais objetivos) e com maior segurança e confiabilidade (por evitar erros de esquecimento), se as heurísticas forem usadas.

#### 4.6.2 Heurísticas para Transformações entre Linguagens

Foram também identificadas e definidas heurísticas para os processos de transformação entre as linguagens (Linguagem Informal assumida para a Linguagem Comum, Linguagem Comum para a Linguagem Formal assumida e da Linguagem Formal para a Linguagem Comum). Este último caso de transformações (Linguagem Formal para Linguagem Comum) permite que a Especificação Formal seja parafraseada, isto é, seja traduzida (de volta) para a Linguagem Comum. Com isto, o Usuário pode validar a Especificação Formal mesmo não conhecendo a Linguagem Formal utilizada.

Através da análise do comportamento do Analista nestas atividades, pode-se identificar certos padrões nas ações. Verificou-se que certas formas-padrão na linguagem fonte sugeriam certas ações de transformação para a linguagem objeto, e isto foi tomado como base para a definição das heurísticas.

Ao se testarem as heurísticas em alguns experimentos, notou-se que elas não cobrem todas as tarefas e isto exige uma forte participação do Analista.

Por outro lado, estas heurísticas ajudam consideravelmente na realização das tarefas, o que torna o processo mais produtivo (mais rápido e com menos erros) e menos cansativo para o Analista.

Isto se deve ao fato de as heurísticas ajudarem naquelas tarefas mais repetitivas e que exigem maior atenção a detalhes e controle de grande volume de informações.

A identificação de heurísticas para estes processos também permite que sejam desenvolvidas ferramentas automatizadas para

auxílio ao Analista e que até mesmo algumas tarefas sejam feitas automaticamente.

Ao sistematizar os processos, as heurísticas procuram direcionar as transformações, controlando as perdas de informações entre as linguagens e, conseqüentemente, os graus de naturalidade e precisão.

Além disto, as heurísticas procuram comprovar a proximidade da Linguagem Comum proposta com a Linguagem Informal assumida e com a Linguagem Formal assumida, uma vez que aquelas são simples e de fácil entendimento e facilitam as transformações.

## 5 APRESENTAÇÃO DA PROPOSTA

Neste capítulo, são apresentadas e definidas a Linguagem Comum e as heurísticas (para coleta das informações e para as transformações entre as linguagens).

### 5.1 Definição da Linguagem Comum

A seguir é feita a definição da Linguagem Comum proposta para ser intermediária e próxima às linguagens Informal e Formal assumidas neste trabalho.

A Linguagem Comum possui 3 partes: uma para representar os dados e suas relações, outra para modelar as operações sobre os dados e outra para definir restrições de integridade.

Cada parte pode ainda ser sub-dividida conforme áreas, divisões ou assuntos da Organização.

É definido também como deverá ser o Dicionário de Termos.

#### 5.1.1 Parte de Dados e Estrutura

Devem ser utilizadas frases-padrão para representar os dados e suas relações. Informações que não se encaixarem nos padrões ou não puderem ser transformadas para as frases-padrão serão consideradas irrelevantes para esta etapa.

As frases-padrão deverão ser orações simples (com um verbo principal somente) do tipo:

SUJEITO VERBO COMPLEMENTO\_VERBAL,

sendo que SUJEITO e COMPLEMENTO\_VERBAL podem ser compostos de vários nomes ligados por preposição, e cada nome ainda pode ter



adjetivos associados a ele.

Opcionalmente poderão ser utilizados Adjuntos Adverbiais nas frases-padrão, para descrever tempo, local ou outra qualidade das ações representadas pelos verbos.

Os verbos podem ser regidos por preposição ou não, podem estar na forma ativa ou passiva e ainda podem ser expressões, se assim for definido no Dicionário de Termos.

Só são permitidos verbos transitivos diretos ou indiretos (definindo associações binárias). Portanto, não podem ser usados verbos bitransitivos (associações ternárias), em razão de ser impossível a quantificação de Sujeitos e Complementos, como será visto na parte de restrições da Linguagem Comum (mapeamento das associações segundo os tipos "para todo  $x \rightarrow 1 y$ " ou "para todo  $x \rightarrow N u$ "). Os verbos bitransitivos deverão ser transformados ou o objeto indireto deverá ser considerado como adjunto adverbial.

Exemplo:

"vendedores vendem produtos a clientes"  $\Rightarrow$  "vendedores vendem produtos" e "vendas são feitas a clientes".

Observação: o símbolo " $\Rightarrow$ " indica que a(s) expressão(ões) que o precede(m) é(são) transformada(s) na(s) ou gera(m) a(s) expressão(ões) que o sucede(m).

Já os verbos intransitivos somente poderão ser usados se seguidos por um adjunto adverbial (introduzido por preposição), uma vez que só assim caracterizam associações entre dados.

As orações compostas escritas na Linguagem Informal devem ser transformadas em orações simples na Linguagem Comum.

Exemplo:

Usuário pode retirar periódico somente se é professor  $\Rightarrow$

usuários retiram periódicos; professores são usuários.

A seguir são dados exemplos de representação dos dados e suas relações na Linguagem Comum.

Observação: convencionou-se usar o plural dos nomes centrais de Sujeitos e Complementos Verbais e o verbo na terceira pessoa do plural do presente do indicativo, além de serem suprimidos os artigos.

SUJEITO	VERBO	COMPLEMENTO VERBAL
vendedores	vendem	casas
alunos de mestrado	solicitam	bolsas
professores de cursos de pós- graduação	ministram aulas a	turmas de alunos
livros	são emprestados a	usuários da biblioteca
alunos especiais	matriculam-se em	disciplinas oferecidas
usuários da biblioteca	fazem reserva de	livros
usuários da biblioteca	retiram por empréstimo	livros
vendas	ocorrem em	datas

Exemplos de frases-padrão com uso adicional do adjunto adverbial:

Professores dão aulas a turmas em salas e horários;

Mecânicos operam máquinas em datas;

Profissionais ministram cursos em cidades;

Clientes pagam valores por produtos;

Clientes pagam produtos com valores;

Pacientes pagam consultas com cheques.

Frases que contenham nomes que referenciam a Organização ou parte desta ou seus funcionários devem ser transformadas, uma vez que interessam somente novas informações.

Exemplo 1:

A biblioteca empresta livros: é uma informação que, se armazenada no Sistema de Informação, não traria nada de novo; tal frase seria útil se houvesse mais de uma biblioteca e se quisesse guardar informações sobre que biblioteca emprestou que livros (então a frase correta seria: bibliotecas emprestam livros).

Exemplo 2:

A balconista vende produtos: supondo-se que só exista uma balconista (daí a palavra no singular), o conceito "balconista" não acrescenta nada de novo, pois somente ela é que vende produtos.

Exemplo 3:

A Organização doa livros a instituições: a frase correta a ser formulada é "livros são doados a instituições".

No mesmo raciocínio, deverão ser documentadas (nas frases-padrão) somente informações relevantes para a Organização, e esta decisão cabe ao Usuário em conjunto com o Analista. Assim, por exemplo, a frase "usuários da biblioteca devolvem livros" muito provavelmente não é de interesse, pois não se quer (supondo-se) guardar este tipo de informação.

Outro cuidado a ser tomado é com pronomes. Estes deverão ser substituídos pelos nomes correspondentes.

Exemplo 1: Usuários retiram livros. Eles também podem reservar livros ==> Usuários retiram livros; Usuários reservam livros.

Exemplo 2: Pacientes desmarcam suas consultas ==> Pacientes desmarcam consultas de pacientes.

Observação: geralmente, o nome correspondente aparece em frases anteriores no documento da Linguagem Informal.

É sempre interessante analisar as orações adjetivas, pois, além de qualificarem um nome, geralmente, elas guardam em si novas informações.

Exemplo: Alunos que têm débito não podem retirar livros ==> isto indica que "alunos têm débito" é uma informação que também precisa ser definida através das frases-padrão.

Quanto às orações adjetivas ainda, é necessário transformá-las em adjetivos simples ou expressões adjetivas, já que não são permitidas duas orações na mesma frase da Linguagem Comum.

Exemplo: alunos que têm débito = alunos com débito = alunos em débito = alunos devedores.

Estas expressões com mesmo significado deverão ser anotadas no Dicionário de Termos.

Na Linguagem Comum ainda, deve-se ter especial atenção com o verbo "ser". O sujeito de tal verbo deve ser o conceito mais abrangente.

Exemplo: pessoas recebem salário se são empregados.

É errado dizer que: pessoas são empregados.

O correto, na Linguagem Comum, é: empregados são pessoas, pois todo empregado é também uma pessoa, o que não ocorre no inverso.

Frases deste tipo também deverão ser documentadas quando encontrados Sujeitos ou Complementos Verbais com nomes comuns.

Exemplo: pessoas jurídicas têm contas bancárias; pessoas

físicas têm contas bancárias ==> pessoas têm contas bancárias;  
pessoas físicas são pessoas; pessoas jurídicas são pessoas.

As duas últimas frases são óbvias devido ao nome "pessoas" ser comum, mas tais informações devem ser explicitadas para melhorar a precisão.

Todas as frases deverão ser documentadas nas formas (vozes) ativa e passiva. Quando o verbo não possuir forma passiva (verbos transitivos indiretos), um sinônimo deverá ser usado (ver o Dicionário de Termos a seguir). Tal estratégia permite a "navegação" de informações e consultas nos dois sentidos das associações, permite a identificação de frases que falam a mesma coisa e permite a definição de restrições (de obrigatoriedade de participação dos elementos nas associações e mapeamentos destas).

### **5.1.2 Dicionário de Termos**

Os termos usados na Linguagem Comum (sujeitos, verbos e complementos verbais), que apresentam dúvida quanto ao seu significado, deverão ser definidos no Dicionário de Termos.

Esta definição deverá ser feita (pelo Usuário com assistência do Analista) para precisar o significado de cada termo e, desta forma, forçar uma utilização padrão. Isto melhorará, em parte, a comunicação entre Usuários e Analistas, diminuindo o "abismo semântico" entre eles.

Exemplo: Vendas são caracterizadas pelos vendedores que as fazem e pelos produtos vendidos; informações sobre vendas devem conter ainda dados sobre o cliente e a data de realização.

Deverão também ser anotados, no Dicionário, os termos

sinônimos (com mesmo significado), podendo serem um único nome ou uma expressão.

Exemplo: alunos = corpo discente; fazer reserva = reservar; alunos que têm débito = alunos com débito.

Tal estratégia permite uma unificação de pontos-de-vista diferentes, além de ajudar na identificação de frases que dizem a mesma coisa.

Serão também considerados como sinônimos os verbos e seus respectivos substantivos.

Exemplo: vender = venda.

Isto é particularmente útil para identificar características de associações, através de uma padronização na forma de representação das informações.

Exemplo: vendedores vendem produtos em datas; vendas são feitas em lojas ==> isto informa que "vendas são feitas em lojas e em datas"; portanto, lojas e datas são informações sobre vendas.

No Dicionário, ainda, deverão ser relacionadas as formas passiva e ativa dos verbos utilizados na Linguagem Comum.

Exemplo: emprestar (voz ativa) <--> ser emprestado (voz passiva); bibliotecas emprestam livros, e livros são emprestados por bibliotecas.

Caso um verbo não possua a forma passiva, um sinônimo deve ser achado para tal. Neste caso, os termos só serão considerados sinônimos se estiverem associando os mesmos tipos de elementos (sujeito e complemento verbal).

Exemplos:

alunos matriculam-se em disciplinas ==> disciplinas

têm alunos ("ter" é a forma passiva de "matricular-se em" para alunos-disciplinas);

professores dão aulas a turmas ==> turmas recebem aulas de professores ("receber aulas de" é a forma passiva de "dar aulas a" para professores e turmas);

vendas ocorrem em datas ==> datas caracterizam vendas ("caracterizar" é a forma passiva de "ocorrer em" para vendas e datas).

A razão para isto é que são necessários os dois modos (representando os dois sentidos da relação) para a definição do mapeamento das associações como será visto na parte de restrições da Linguagem Comum.

O Dicionário de Termos também servirá para que sejam feitas referências cruzadas entre as partes da Linguagem Comum (dados, operações e restrições).

Outros exemplos de sinônimos:

fazer matrícula em = matricular-se em;

ter vínculo com = vincular-se a;

fazer consulta a = consultar (a).

A identificação dos sinônimos deve ser feita de forma explícita pelo Usuário (respondendo a perguntas do tipo "quais os sinônimos de \_\_\_\_\_?") ou através de dedução feita pelo Analista, ao analisar os textos (o que deverá ser confirmado pelo Usuário).

Neste último caso, o Analista deve atentar para frases semelhantes e para nomes (ou expressões) utilizados com mesma finalidade ou função.

Exemplos:



pacientes fazem consulta aos médicos; pacientes consultam os médicos ==> de onde se deduz que "fazer consulta" é sinônimo de "consultar";

antes da matrícula, os alunos preenchem uma solicitação de matrícula; para fazer a matrícula, é necessário receber o pedido de matrícula do aluno ==> de onde se deduz que "solicitação de matrícula" é sinônimo de "pedido de matrícula".

### 5.1.3 Parte de Operações

As operações, na Linguagem Comum, seguem uma das seguintes formas:

1. Verbo objeto\_ou\_complemento\_verbal:  
Ações
2. Substantivo\_do\_verbo preposição objeto\_ou\_complemento\_verbal:  
Ações

Sendo que cada ação pode ser simples (com o seguinte formato: Verbo complemento\_verbal) ou ser outra operação (neste caso, a ação é definida por outras ações).

Observação: o complemento verbal pode vir introduzido pela partícula "se" no caso de operações de consulta (isto indica que será verificada uma condição) e pode incluir formações mais complicadas (orações, comparações, etc).

Eventualmente, as expressões "de cada" e "para cada" poderão ser utilizadas para indicar que uma operação (ou ação ou tratamento) será repetida para um conjunto de elementos. Entretanto, este tipo de informação é desconsiderado na Linguagem Formal.

Exemplo 1:

Contratar funcionário:

Recebe nome e dados pessoais do funcionário  
Verifica se o funcionário não está cadastrado  
Cadastra o funcionário

Exemplo 2:

Matrícula de aluno:

Recebe pedido de matrícula do aluno  
Para cada disciplina no pedido de matrícula  
Verifica se o aluno tem os pré-requisitos da  
disciplina requerida  
Registra matrícula do aluno na disciplina

Exemplo 3:

Verifica se aluno tem pré-requisitos de disciplina:

Recebe disciplina requerida  
Recebe aluno  
Verifica se o aluno cursou com conceito de aprovação as  
disciplinas pré-requisitos da disciplina requerida  
Verifica se o aluno tem, no mínimo, o número de  
créditos exigidos pela disciplina requerida  
Coloca "sim" na lista exterior

Exemplo 4:

Aumentar salário do funcionário em 10%

Recebe identificação do funcionário  
Verifica se funcionário está cadastrado  
Calcula novo salário, que é igual a salário do  
funcionário + 10%  
Substitui velho salário do funcionário pelo novo  
salário do funcionário

Algumas restrições são impostas:

1) as ações devem ser de um dos 5 tipos:

- solicitar ou receber uma informação: esta informação deverá ser fornecida por um Usuário do Sistema de Informação; é a maneira de introduzir informações no Sistema;
- verificar uma condição: consultando ou processando as informações disponíveis ao Sistema de Informação;
- calcular uma informação, utilizando uma expressão matemática;

- modificar o conjunto de informações: podendo o Sistema consultar ou processar informações antes de fazer a modificação;
- fornecer informações ao meio exterior: como forma de resposta a operações solicitadas por Usuários;

2) a ordem das ações deve ser como a descrita acima.

São assumidas as seguintes interpretações :

1) as ações de modificação só são efetuadas se as condições forem verificadas (se as condições não forem verdadeiras, a operação não é efetuada);

2) assume-se que as informações que são utilizadas pelas operações e não são solicitadas pelo Sistema de Informação estão disponíveis ao Sistema; assim não são necessárias, na operação do exemplo 3, as ações "consulta pré-requisitos da disciplina requerida" e "consulta o número de créditos do aluno";

3) assume-se que, quando efetuadas sem problemas, as operações não necessitam informar nada ao Usuário do Sistema; caso contrário, algum aviso deve ser dado.

4) as operações de consulta sempre fornecem uma resposta, seja esta uma informação do Sistema de Informação ou simplesmente um "sim" ou "não", confirmando ou não a validade de um fato.

As operações que envolvem decisões que não puderem ser definidas por um algoritmo servirão apenas para registrar as informações resultantes da decisão (tomada por alguma pessoa ou pessoas em contato com o Sistema de Informação). Assume-se que estas informações serão fornecidas ao Sistema de Informação quando for desempenhada a operação.

Exemplo:

Seleciona bolsistas:

Recebe identificação de bolsistas a selecionar

Verifica se bolsista é candidato

Registra bolsista selecionado

Deve-se tomar como base para a definição de operações que todas as informações devem estar disponíveis ao Sistema de Informação. Assim, tomando-se o exemplo anterior, é necessário registrar quem são os bolsistas selecionados, mesmo antes de associá-los a uma bolsa individual, apesar de esta informação existir fora do Sistema de Informação, provavelmente na mente de alguém ou em um documento informal (uma anotação particular, talvez).

Foi definido ainda um novo conceito para a parte de operações da Linguagem Comum: o conceito de lista exterior. O objetivo de tal é fornecer um canal de comunicação no sentido Sistema de Informação --> meio exterior.

Exemplo: Divulga bolsistas selecionados (sabe-se informalmente que esta operação irá criar uma lista dos bolsistas selecionados em algum meio físico, provavelmente um relatório).

Ações: Para cada bolsista selecionado

Coloque bolsista na lista exterior.

Assim, portanto, tem-se que o Sistema de Informação funciona como uma caixa-preta, recebendo informações, processando-as internamente, guardando-as, recuperando-as e fornecendo respostas.

Por fim, é possível analisar o uso das informações de que o Sistema de Informação tem conhecimento, verificando se as informações recuperadas em uma operação são registradas em outra.

Por exemplo, no caso da operação do exemplo anterior, "bolsista selecionado" é interpretado como um tipo de informação

de conhecimento do Sistema de Informação (pois não um dado fornecido pelo Usuário); portanto, esta informação deve ter sido fornecida ao (ou registrada no) Sistema de Informação, em outra operação (no caso, a operação "seleciona bolsistas").

Observação final: a Linguagem Comum permite certas variações (se pequenas) dos seus padrões para a definição dos complementos das ações das operações (como o uso de comparações, orações adjetivas, outras orações, etc); isto é bastante útil para permitir que um maior número de informações possam ser documentadas, mas o uso de variações muito complicadas pode dificultar o trabalho de transformação para a Linguagem Formal, podendo até inutilizar as heurísticas de transformação.

#### **5.1.4 Parte de Restrições de Integridade**

"Integridade é uma propriedade de um banco de dados (ou Sistema de Informação), que se refere à validade do seu conteúdo e da forma pela qual foi alcançada". "Um estado (visto como uma configuração estantânea do banco de dados) é dito integro se formado por uma combinação válida de objetos e alcançado por meio de transições válidas a partir de um outro estado integro" [San80].

"Uma restrição de integridade é uma regra que restringe o conjunto de estados integros e/ou o conjunto de transições válidas em um banco de dados" [San80].

A definição dos dados e das operações cria um conjunto de estados ou ações permitidos. No sentido inverso, as restrições procuram definir o que não é permitido ou o que,

obrigatoriamente, deve ocorrer (estados ou ações). Muitas vezes, as restrições criam redundâncias de informações (já descritas de outra forma nas primeiras partes da linguagem); porém, há inúmeras informações que não podem ser especificadas a não ser através de restrições de integridade.

Ocorre é que não há uma maneira precisa, correta e amplamente aceita como ideal ou completa para se definirem restrições. Esta definição varia com o tipo da restrição e o tipo do sistema sendo especificado.

[San80] apresenta um ótimo estudo sobre classificação de restrições de integridade, no âmbito de Banco de Dados. Entre outras coisas, as restrições são analisadas pelas suas características e pela forma como são tratadas e especificadas.

Os padrões propostos aqui na Linguagem Comum (identificados em experimentos realizados), e usados para a definição de restrições, não abrangem todas as possibilidades, mas procuram ser uma base para permitir a definição de restrições, pelo menos, os tipos mais comuns. Outras restrições, com pouco ou algum esforço, poderão ser encaixadas em ou transformadas para um dos padrões ou poderão ser definidas através de combinações dos padrões.

Por outro lado, não é intenção da Linguagem Comum restringir totalmente a definição de restrições aos padrões aqui propostos ou a combinações destes. Se for necessário especificar uma restrição de integridade que não siga os padrões da Linguagem Comum, isto deverá ser feito. Porém, não é garantido que as heurísticas para transformação da Linguagem Comum para a Linguagem Formal se apliquem nestes casos.

Observação: a representação dos dados nas restrições deve estar de acordo com a representação dos mesmos na primeira parte da Linguagem Comum.

Padrões para a representação de restrições:

Observações:

- nomes em letras minúsculas caracterizam elementos a serem instanciandos;
- os parêntesis indicam nomes opcionais;
- os colchetes informam alternativas válidas;
- "n" é um número inteiro.

1. TODO sujeito DEVE verbo complemento\_verbal.

Exemplo:

Toda pessoa deve ter 1 nome.

2. sujeito NÃO [DEVE / PODE] verbo complemento\_verbal.

Exemplo:

Alunos não pode retirar periódicos.

5. TODO sujeito (PODE) verbo n complemento\_verbal, NO MAXIMO.

Exemplo:

Toda pessoa pode ter 1 nome, no máximo.

6. TODO sujeito DEVE verbo n complemento\_verbal, NO MINIMO.

Exemplo:

Toda pessoa deve ter 1 telefone para recados, no mínimo.

7. a) SOMENTE sujeito PODE verbo complemento\_verbal.

Exemplo:

Somente professores podem retirar periódicos.

b) SOMENTE sujeito\_1 e sujeito\_2 e ... e sujeito\_n PODEM verbo complemento\_verbal.

Exemplo:



Somente professores e funcionários podem reservar periódicos.

8. a) sujeito\_1 QUE sujeito\_2 verbo\_1 (NÃO) [PODE / DEVE] verbo\_2 complemento\_verbal.

Exemplo:

Materiais que usuários retiram não podem ser de consulta local

b) sujeito QUE verbo\_1 complemento\_verbal\_1 (NÃO) [PODE / DEVE] verbo\_2 complemento\_verbal\_2.

Exemplo:

Usuários que têm débito não podem retirar material.

Alunos que têm bolsa devem estar em 1 projeto de pesquisa.

9. sujeito NÃO PODE SER complemento\_verbal\_1 (E complemento\_verbal\_2) AO MESMO TEMPO.

Exemplo:

Funcionário não pode ser candidato ao mesmo tempo.

Usuário não pode ser professor e aluno ao mesmo tempo.

10. sujeito (NÃO) [PODE/DEVE] verbo\_1 complemento\_1 SE (NÃO) verbo\_2 complemento\_2.

Exemplo:

Usuários não podem retirar material se têm débito.

Alunos devem estar em projetos de pesquisa se têm bolsa.

11. sujeito DEVE verbo SOMENTE complemento\_verbal.

Exemplo:

Consultas devem ocorrer somente nos horários para consulta que os médicos têm (pois médicos têm horários para consulta).

12. sujeito (NÃO) (DEVE / PODE) SER [MAIOR / MENOR] QUE complemento\_verbal.

Exemplo:

O número total de alunos que se matriculam em uma disciplina

não pode ser maior que o número de vagas da disciplina.

Salário não pode ser menor que salário-mínimo.

13. NÃO [DEVE / PODE] HAVER [MAIS / MENOS] DE n complemento\_verbal.

Exemplo:

Não pode haver mais de uma disciplina oferecida com mesmo local e com mesmo horário (pois disciplina oferecida tem local e horário).

Não pode haver mais de 1 diretor.

16. sujeito NÃO [DEVE / PODE] [AUMENTAR / DIMINUIR].

Exemplo:

Custo da produção não pode aumentar.

Salário do funcionário não pode diminuir.

17. sujeito SÔ [DEVE / PODE] verbo complemento\_verbal\_1 SE [E / FOI] complemento\_2.

Exemplo:

Candidato só pode ser bolsista se foi selecionado.

Usuário só pode retirar periódico se é professor.

18. SÔ [DEVE / PODE] verbo complemento\_verbal\_1 QUEM [E / FOI] complemento\_2.

Exemplo:

Só pode ser funcionário quem foi candidato.

Só pode receber salário quem é funcionário.

Observações:

- sujeitos e complementos verbais podem ser um nome mais adjetivos (nome adjetivo\_1 E adjetivo\_2);
- sujeitos e complementos também podem ser um nome mais uma oração adjetiva (nome QUE sujeito\_2 verbo; nome QUE verbo complemento\_verbal).

Outros exemplos de restrições de integridade:

Alunos não podem se matricular em mais de uma disciplina oferecida com mesmo horário (mistura os padrões 2 e 13).

Não pode haver mais de um paciente que faz consulta ao mesmo médico, no mesmo horário (mistura o padrão de número 13 com parte do padrão 8b).

## **5.2 Definição das Heurísticas para Coleta**

As heurísticas identificadas e aqui propostas têm o objetivo de orientar o processo de coleta das informações necessárias ao desenvolvimento do Sistema de Informação, captando assim a informalidade.

Neste trabalho, considera-se somente uma técnica de coleta de dados (entrevista), apesar de outras serem bastante usadas e úteis (observação, análise de documentos, etc).

A entrevista caracteriza-se pela interação Usuário-Analista, sendo que o segundo procura induzir o primeiro a fornecer informações sobre a Organização.

Foram definidas algumas heurísticas para orientar este processo de interação e coleta de dados. Elas funcionam como guias de um roteiro, evitando assim que o Analista que faz a entrevista esqueça perguntas.

Desta forma, as heurísticas vêm auxiliar Analistas não muito experientes e se constituem numa base para um método de coleta baseado em entrevistas.

São formadas por perguntas padrões genéricas ou específicas, podendo as últimas apresentar lacunas para serem preenchidas.

Ao final das perguntas, é feita uma simulação do ambiente (suas funções) com participação do Usuário, onde os tipos de informação serão instanciados com exemplos reais do ambiente.

O objetivo desta simulação é confirmar os fatos coletados ou completá-los. Isto permite que o Usuário use exemplos para explicar algo que não pode definir bem de imediato.

Ter-se-á também uma melhor avaliação do que é fato e o que é opinião ou sentimento (conforme problema citado na primeira parte deste trabalho).

Além disto, as simulações permitem coletar informações que possam ter passado despercebidas durante a entrevista, uma vez que as análises são feitas, geralmente, apenas com informações disponíveis e mais recentes e com estatísticas intuitivas e julgamentos não muito abrangentes (conforme problema já citado e discutido em [Dav82b]).

Na verdade, as simulações se assemelham à técnica de observação do ambiente para a coleta de dados. O Usuário é induzido a fazer ao invés de descrever o que faz. Entretanto, aquelas supõem situações enquanto que esta analisa as situações reais.

As heurísticas assumem que os Usuários a serem entrevistados conhecem bastante a realidade da Organização (ou de parte desta) e que estes fornecerão as informações com boa vontade.

Deverão ser entrevistados vários Usuários e de forma a representarem relativamente bem o todo da Organização (os vários setores, funções, escalões, etc).

Talvez por isto, uma mesma informação possa vir a ser coletada mais de uma vez e até fornecida de maneiras diferentes.

Isto é previsto pelas heurísticas e é desejado, já que se quer coletar o máximo de informações possíveis, e a coleta por técnicas diferentes procura corrigir defeitos de uma com as qualidades de outra (técnica).

As respostas às perguntas, bem como tudo o que é dito pelo Usuário na entrevista, deverão ser gravadas para que não haja esquecimento.

Não é necessário que o Usuário responda a todas as perguntas, e é bom que ele deixe "em branco" aquelas questões que não conhece bem.

As informações coletadas não deverão ser deturpadas, mas o Analista deve procurar completar frases onde se note a falta de informações (como, por exemplo, falta de sujeitos, complementos verbais, complementos nominais e adjuntos adnominais). Isto será extremamente útil para a definição das informações na Linguagem Comum e para a posterior especificação do Sistema.

A seguir são apresentadas as heurísticas para a coleta.

**Perguntas Genéricas:**

1. Quais os objetivos da Organização ?
2. Quais as divisões da Organização ?
3. Qual a missão ou colaboração de cada divisão dentro da Organização ?
4. Quais os objetivos particulares de cada divisão ?
5. Quais os (tipos de) Usuários ou Clientes da Organização ?
6. Quais os produtos ou serviços fornecidos pela Organização a seus Usuários/Clientes ?
7. Que funções a Organização desempenha para atender seus Clientes/Usuários ?

8. Que outras funções são necessárias para manter a Organização ?
9. Que melhoras nos serviços ou produtos da Organização são solicitadas pelos Clientes/Usuários ?
10. Listar as diversas funções desempenhadas pelos funcionários da Organização ?
11. Quais as decisões tomadas dentro da Organização ?
12. Que seria necessário para melhorar o trabalho dos funcionários da Organização ?
13. Que cadastro, arquivos, registros, formulários e outras informações são mantidos pela Organização ?
14. Quais os relatórios internos ou externos da Organização ?
15. Quais são os fornecedores ou mantenedores da Organização ?
16. Que mudanças estão previstas no funcionamento da Organização ?

**Perguntas Específicas:**

- a) Para cada tipo identificado na pergunta 5:

Como são atendidos (que funções são desempenhadas para tal) ?

Que informações são guardadas a seu respeito ?

- b) Para cada tipo identificado pergunta 15:

Qual o fluxo de informações entre este e a Organização ?

Que serviços ou produtos são trocados entre este e a Org. ?

Que informações são necessárias sobre o tipo ?

- c) Para cada tipo identificado em 13 e 14:

Quais as informações que constituem este (que são guardadas nele) ?

Qual a finalidade deste (para que é utilizado) ?

Quando este é atualizado ou consultado ?

d) Para cada decisão identificada em 11:

Como é tomada (que critérios, informações, ferramentas e técnicas são utilizados) ?

Quais são e como são apresentados os resultados desta ?

Quando esta é feita ?

Quais as conseqüências e riscos ?

Quais os fatores críticos para uma boa decisão ?

e) Para cada função identificada nas demais respostas:

Quais as informações que devem estar disponíveis para que a função seja desempenhada ?

Quais as ações a serem tomadas ?

Que condições devem ser avaliadas antes de se desempenhar a função ?

Quais os resultados da função ?

Que exceções podem ocorrer e com que se deve tomar cuidado especial (algo raro mas que pode acontecer; casos anormais) ?

Observação: algumas ações ainda poderão ser tratadas como funções devido a sua complexidade; então, as mesmas perguntas específicas deverão ser feitas para cada uma delas.

Deverão ser identificadas palavras-chave nas respostas às perguntas. Podem ser substantivos, expressões, nomes adjetivados, verbos, etc.

Para cada palavra-chave identificada (exceto verbos), deve-se fazer as seguintes perguntas quando apropriado (o Analista decide):

O que é \_\_\_\_\_ ?

Para que serve \_\_\_\_\_ ?



Onde é utilizado \_\_\_\_\_ ?

Fale mais sobre \_\_\_\_\_ ?

Há sinônimos para \_\_\_\_\_ ?

Quem fornece \_\_\_\_\_ ?

Para cada verbo identificado, deverão ser feitas as seguintes perguntas quando apropriado (decidido pelo Analista):

O que é \_\_\_\_\_ ?

Quais os complementos verbais mais utilizados com \_\_\_\_\_ ?

Fale mais sobre \_\_\_\_\_ ?

Há sinônimos para \_\_\_\_\_ ?

Para que \_\_\_\_\_ ?

Como \_\_\_\_\_ ?

Qual o substantivo apropriado para \_\_\_\_\_ ?

Observação: as lacunas deverão ser preenchidas com as palavras-chave.

Deverá ser verificado se cada palavra-chave aparece, no mínimo, em uma função. Se isto não ocorrer, fica caracterizada a falta de uma função que trate tal elemento ou a omissão deste em uma função.

### **5.3 Definição das Heurísticas para Transformação da Linguagem Informal para a Linguagem Comum**

Nestas transformações, as informações descritas na Linguagem Informal (em forma de respostas ao questionário) deverão ser traduzidas para a Linguagem Comum segundo os padrões desta.

### 5.3.1 Parte de Dados e Estrutura

Os períodos compostos (aqueles com mais de uma oração) deverão ser decompostos em orações simples. As conjunções deverão ser desconsideradas, bem como as informações que descrevem, já que não interessam para definir os dados e suas estruturas.

Deve-se procurar completar as frases-padrão da Linguagem Comum com as correspondentes (e corretas) informações.

Isto também deve ser feito quando o texto na Linguagem Informal apresenta omissões (sujeitos, complementos, adjuntos, etc). Primeiro, deve-se completar o texto na Linguagem Informal e, então, transpor as frases para a Linguagem Comum.

Exemplos: livros são doados (por quem, a quem); usuários lêem na biblioteca (o que); pessoas votam (em quem, onde, quando).

As orações com verbos intransitivos (sem adjunto adverbial) também devem ser desconsideradas. Porém, deve-se verificar se o sujeito da frase aparece em outras frases. Se não, algo estará errado, e a coleta deverá ser refeita (pois o sujeito pode ser informação importante).

Exemplo: livros estragam (porém, mais adiante aparece: usuários da biblioteca lêem livros, etc).

Deve-se procurar sempre generalizar as informações (identificar o tipo dos exemplos citados).

Exemplos:

reuniões acontecem no final da semana ==> reuniões acontecem em datas;

vendas de carros devem ter a permissão do diretor; vendas de

motos devem ter a permissão do supervisor ==> vendas de produtos têm permissões de funcionários.

Observação: as informações das duas frases originais caracterizam bem restrições e, portanto, devem ser documentadas também na parte de restrições da Linguagem Comum.

Os adjuntos adverbiais, por sua vez, devem, necessariamente, ser introduzidos por preposição. Caso contrário (e isto ocorre, geralmente, com advérbios de modo), deverão ser desconsiderados ou substituídos por informações mais precisas.

Exemplos:

Visitas são feitas anualmente ==> em anos;

Visitas são feitas regularmente ==> desconsiderar.

As preposições "DE" e "COM" devem ser analisadas com certa atenção, pois podem esconder informações importantes.

Supondo a forma "nome\_1 DE nome\_2" para preposição "DE", deve-se verificar se "nome\_1" é propriedade de "nome\_2" ou se "nome\_1" é formado por "nome\_2". No primeiro caso, uma frase-padrão do tipo "nome\_2 TEM nome\_1" deverá ser acrescentada à Linguagem Comum (exemplo: nome de aluno ==> alunos têm nomes).

Já no segundo caso, acrescenta-se, à Linguagem Comum, uma frase-padrão do tipo "nome\_1 TEM nome\_2" (exemplo: receita de medicamentos ==> receita têm medicamentos).

Com a preposição "com", deve-se verificar se "nome\_1" (na forma "nome\_1 COM nome\_2") tem "nome\_2". Em caso afirmativo, acrescenta-se, à Linguagem Comum, um frase-padrão do tipo "nome\_1 TEM nome\_2" (exemplo: aluno com débito ==> alunos têm débitos).

A seguir é apresentado um exemplo de transformação:

Quais os serviços oferecidos pela Organização (uma

Biblioteca) aos seus Usuários ?

A Biblioteca permite que os Usuários retirem por empréstimo materiais bibliográficos, sendo exigido o nome do Usuário. Para os professores, há o serviço de empréstimo de periódicos.

Linguagem Comum:

usuários retiram materiais bibliográficos ("retirar" é sinônimo de "retirar por empréstimo");

periódicos são emprestados a professores;

usuários têm nomes (deduzido de "nome do Usuário").

### 5.3.2 Parte de Operações

Cada função identificada na coleta (e descrita na Linguagem Informal) deve ser tratada como uma operação na Linguagem Comum. Deverão ser identificadas as ações efetuadas durante a função (cada verbo, em potencial).

Então, verifica-se o tipo de cada ação (solicitação de informação, verificação de condição, cálculo de uma informação, modificação do conjunto de informações, fornecimento de informações ao meio exterior), e estas são colocadas na ordem apropriada (proposta na Linguagem Comum).

Devem ser utilizados verbos semelhantes a: solicita, recebe, verifica (se fato), substitui, altera, registra, anota, coloca, fornece, lista, calcula, etc.

Deve-se também procurar completar as informações quando necessário e desconsiderar meios físicos de armazenamento da informação.

Exemplo: verifica se há horário disponível no arquivo de

consultas ==> completar "horário" com "para consulta ao médico" e desconsiderar "no arquivo de consultas".

A seguir é apresentado um exemplo de transformação.

Descrição em Linguagem Informal:

Modificar o endereço do paciente (algun funcionário responde como é desempenhada esta função):

Eu **solicito** ao paciente o seu nome. **Vou** até o cadastro e **pego** sua ficha. **Vejo** o seu endereço antigo e **pergunto** se **quer** mesmo **muda**-lo. Ele me **dá** seu novo endereço e eu **substituo** o velho na ficha. Então **recoloco** a ficha no cadastro.

Descrição em Linguagem Comum:

Nome: Modificar endereço de paciente.

Ações:

1. Solicita nome do paciente.

Observações:

- desconsiderar "ao paciente", pois indica interação com o Usuário do Sistema de Informação;
- "seu nome" significa "nome do paciente";
- desconsiderar "vou" e "pego", pois são ações físicas que não envolvem informação;
- "vejo" deve ser desconsiderado, pois assume-se que todas as informações estão disponíveis ao Sistema de Informação, se não solicitadas ao Usuário;
- desconsiderar "pergunto", "quer", "mudar", pois assume-se que a operação não necessita confirmação.

2. Recebe novo endereço do paciente.

Observações:

- as ações devem ser consideradas do ponto de vista do

Sistema de Informação, isto é, o agente da ação é sempre o Sistema, por isto a inversão do verbo "dá";

- "ele" e "seu" se referem a "paciente".

Substitui endereço do paciente pelo novo endereço.

Observações:

- "substituo" implica em substituir o antigo pelo novo;
- "velho endereço" é na verdade o endereço do paciente para o Sistema de Informação, assim como "novo endereço" passará a ser considerado somente "endereço";
- "na ficha" não interessa pois é meio físico de armazenamento de informação.

As ações também poderão ser consideradas como operações (e, como tais, poderão ser descritas por outras ações) toda vez que for necessário divid-las para melhor explicá-las (isto deve ser verificado na Linguagem Informal e ocorre, geralmente, em respostas às perguntas específicas sobre palavras-chave, no caso, verbos).

### 5.3.3 Parte de Restrições

A identificação de restrições deve ser feita segundo uma das formas (não-exclusivas) apresentadas a seguir.

- 1) verificação do mapeamento das associações ("para todo x --> 1 y" ou "para todo x --> N y":

deve-se fazer a seguinte pergunta ao Usuário (para todas as frases-padrão da parte de dados da Linguagem Comum, inclusive as frases inversas):

UM sujeito PODE verbo QUANTOS complementos\_verbais ?

Exemplos:

**Frase-padrão:** pacientes consultam médicos;

**Pergunta:** Um paciente pode consultar quantos médicos ?;

**Resposta:** vários (equivalente ao símbolo "N");

**Mapeamento** da associação "consulta" entre "paciente" e "médico":  
"para todo x --> N y", sendo "x" o sujeito da frase e "y" o complemento verbal.

**Observação:** não é necessário especificar tal informação, pois não se trata de uma restrição (o número "N" não foi definido).

**Frase-padrão:** pacientes têm nomes;

**Pergunta:** Um paciente pode ter quantos nomes ?;

**Resposta:** um, no máximo;

**Mapeamento** da associação "tem" entre "paciente" e "nome": "para todo x --> 1 y";

**Definição** da restrição de integridade na Linguagem Comum: Todo paciente pode ter 1 nome, no máximo (padrão número 5).

2) verificação de obrigatoriedade de participação de um elemento em uma associação:

fazer a seguinte pergunta ao Usuário (para todas as frases-padrão da parte de dados);

UM sujeito DEVE OU PODE verbo complemento\_verbal ?

Exemplo:

**Frase-padrão:** consultas ocorrem em datas;

**Pergunta:** uma consulta deve ou pode ocorrer em data ?;

**Resposta:** deve;

**Caráter da participação** de "consulta" na associação "ocorrer\_em" com o elemento "data": obrigatório (utilizar o padrão número 1 para a especificação desta restrição);



**Definição** da restrição de integridade na Linguagem Comum: "toda consulta deve ocorrer em data".

3) verificação de frases escritas na Linguagem Informal que se encaixem em um dos padrões de restrições (diretamente ou com alguma modificação na forma, mas não no conteúdo):

Exemplo:

"Somente vendedores podem vender produtos" (padrão número 7a);

"Funcionário não pode ter salário se não está contratado" (padrão número 10).

Especial atenção deve ser dada às palavras ou expressões que identificam restrições: dever (ou conjugações deste verbo como "deve" e "devem"), somente (ou "só), nunca, sempre, se ... então, não pode, somente ... se, etc.

#### **5.4 Definição das Heurísticas para Transformação da Linguagem Comum para a Linguagem Formal**

##### **5.4.1 Parte de Dados**

Primeiramente, deverão ser identificados os elementos sintáticos de cada frase-padrão da Linguagem Comum (parte de dados), isto é, o sujeito, o verbo, o complemento verbal e o adjunto adverbial (se houver) da frase. Também deverão ser identificadas as preposições e os adjetivos.

Esta tarefa deve ser feita pelo Analista. Porém, ela pode ser agilizada se estes elementos já forem identificados e marcados na formação das frases-padrão da Linguagem Comum.

O problema de sinônimos deverá ser solucionado com o uso de

um dos nomes como padrão.

Heurísticas:

1. sujeito(x), complemento\_verbal(y), verbo(x,y).

Exemplo:

Pacientes consultam médicos ==> paciente(x), médico(y),  
consulta(x,y).

(Observação: consulta = consultou = recebeu consulta de).

2. preposição( verbo(x,y), w), adjunto\_adverbial(w).

Observação: a "preposição" pode vir representada por um verbo intransitivo mais um preposição introduzindo adjunto adverbial.

Exemplo:

Usuários da biblioteca retiram livros em datas ==> em(  
retirou(u,l), d), usuário(u), livro(l), data(d).

(Observação: "retirou" é sinônimo de "empréstimo"; "em" é sinônimo de "ocorreu\_em").

3. substantivo(x) e adjetivo(x).

Exemplos:

candidato selecionado ==> candidato(x) e selecionado(x);

aluno novo ==> aluno(x) e novo(x).

4. tratamento de preposições:

As preposições definem relações entre os nomes que são ligados por elas.

(Observação: os tipos de relações aqui tratados são aqueles mais utilizados no contexto deste trabalho).

**Preposição "A":** formato = nome\_1 A nome\_2.

**Tipo de Relação**                      **Tratamento**

4.1 destino, finalidade    nome\_1(x) e a\_nome\_2(x)

Exemplo: candidato a bolsa ==> candidato(x) e a\_bolsa(x).

**Preposição "DE":** formato = nome\_1 DE nome\_2.

**Tipo de Relação**                      **Tratamento**

4.2 restrição, tipo                      nome\_1(x) e de\_nome\_2(x)

Exemplo: aluno de graduação ==> aluno(x) e de\_graduação(x).

4.3 proveniência                      nome\_1(x) e de\_nome\_2(x)

Exemplo: aluno do interior ==> aluno(x) e do\_interior(x).

4.4 propriedade                      nome\_1(x), de\_nome\_2(x), nome\_2(y) e tem(y,x)

Exemplos: nome de aluno ==> nome(x), de\_aluno(x), aluno(y) e tem(y,x);

débito de aluno ==> débito(x), de\_aluno(x), aluno(y) e tem(y,x).

4.5 formação                      nome\_1(x), de\_nome\_2(x), nome\_2(y) e tem(x,y)

Exemplos: receita de medicamentos ==> receita(x), de\_medicamentos(x), medicamento(y) e tem(x,y);

agenda de consultas ==> agenda(x), de\_consultas(x), consulta(w,z), paciente (w), médico(z), tem(x, consulta(w,z));

pedido de matrícula ==> pedido(p), de\_matricula(p), matricula(w,z), aluno(w), disciplina(z), tem(p, matricula(w,z)).

4.6 outros tipos                      nome\_1(x) e de\_nome\_2(x)

**Preposição "PARA":** formato = nome\_1 PARA nome\_2.

**Tipo de Relação**                      **Tratamento**

4.7 destino                      nome\_1(x) e para\_nome\_2(x)

Exemplos: verba para evento ==> verba(x) e para\_evento(x);

material para aquisição ==> material(x) e para\_aquisição(x);

horário para consulta ==> horário(x) e para\_consulta(x).

**Preposição "COM":** formato = nome\_1 COM nome\_2.

**Tipo de Relação**                      **Tratamento**



adjetivo "feitos" e não o substantivo "produtos") ==>  
produto(x) e feito\_em\_casa(x);

livros marcados com consulta local ("com" está  
complementando o participio "marcados") ==>  
livro(x) e marcado\_com\_consulta\_local(x).

#### 5.4.2 Parte de Operações

Regras Gerais:

- analisar cada ação da operação descrita na Linguagem Comum e identificar padrões;
- a representação dos dados deve estar de acordo com a mesma na parte de dados e estrutura da Linguagem Comum;
- todas as variáveis deverão ser definidas por um predicado;

Exemplo: tem(x,y) e nome\_1(x) ==> deve-se definir "y" na  
forma nome\_2(y).

<b>Padrões</b>	<b>Tratamento</b>
1. SOLICITA nome (ou sinônimos como RECEBE)	"x" é argumento de entrada da operação
2. a) VERIFICA SE fato (ou CONSULTA SE, etc.)	Pré: fato
b) VERIFICA SE NÃO fato	Pré: não fato
3. SUBSTITUI nome_1 POR nome_2	Pós-condição: não [nome_1] e nome_2
4. a) REGISTRA nome_1 (ou ANOTA, MARCA, etc)	Pós: nome_1
b) REGISTRA QUE fato	Pós: fato
5. RETIRA nome_1 (ou DESMARCA, etc)	Pós: não [nome_1]
6. NOVO/VELHO nome	velho --> nome(x2) novo --> nome(x1) x1 ≠ x2 e x1 é argumento de entrada

7. a) NÃO fato                      não [ fato ]
- b) sujeito NÃO                      não [ verbo(x,y) e sujeito(x) e  
verbo compl. verbal                      compl\_verbal(y) ]
8. a) nome ESTA                      nome(x), onde "x" é argumento de entrada  
CADASTRADO                      ou tem ligação com este
- b) nome EXISTE                      idem
9. nome\_1 verbo nome\_2              nome\_1(x), nome\_2(y) e verbo(x,y)
11. nome\_2 DE nome\_1              se existe predicado tem(x,y) tal que  
nome\_1(x) e nome\_2(y) então tem(x,y),  
nome\_1(x) e nome\_2(y)  
senão nome\_2(x) é de\_nome\_1(x)
12. MESMO, E                      idéia de igualdade (x=y)  
COINCIDE, COINCIDEM,  
etc.
13. OUTRO nome,                      nome(x), nome(y) e  $x \neq y$   
DOIS nome,  
MAIS DE UM nome, etc.
14. nome\_de\_ação                      procurar e relacionar o sujeito e o  
(substantivo                      complemento verbal  
de um verbo)
15. HÁ nome                      nome(x)
- 16 COLOCA nome EM                      nome(y) e "y" é argumento de saída  
(ou NA, NO, PARA)  
LISTA EXTERIOR
17. nome adjetivo                      nome(x) e adjetivo(x)
18. REGISTRA nome\_1                      nome\_1(x) e nome\_2(x)  
COMO nome\_2
19. nome\_1 QUE verbo                      nome\_1(x), nome\_2(y) e verbo(x,y)  
nome\_2
20. nome\_1 EM                      ver o tipo de ligação entre nome\_1  
(ou NA, NO) nome\_2                      e nome\_2
21. nome\_1 verbo\_ser\_                      nome\_1(x) e nome\_2(x)  
\_ou\_estar nome\_2
22. eliminar redundâncias
23. nome\_2 COM nome\_1              se existe predicado tem(x,y) tal que  
nome\_1(x) e nome\_2(y) então tem(x,y),  
nome\_1(x) e nome\_2(y)  
senão nome\_2(x) é com\_nome\_1(x)

24. preposições introduzindo adjunto adverbial      prep( verbo(x,y), adjunto),
25. NO MINIMO      x >= y
26. NO MAXIMO      x <= y
27. nome\_1 QUE sujeito verbo      nome\_1(x), sujeito(y), verbo(y,x)
28. nome\_1 preposição nome\_2      ver ligação de nome\_1 com nome\_2
29. nas pré e pós-condições: ligação com argumentos (de entrada e saída) e definição de seus tipos
30. CALCULA x, QUE É IGUAL A expressão (é colocado nas pré ou pós-condições, dependendo onde será usado "x")      x = expressão

Quantificação das Variáveis:

- todas devem ser quantificadas exceto as que identificam argumentos de entrada (pois serão instanciadas);
  - quantificar uma vez só;
  - a quantificação deve ser feita após a especificação das pré e pós-condições e dos argumentos de entrada;
  - a prioridade de avaliação é: primeiro as pré-condições, depois as pós-condições; quantificar as variáveis por ordem de aparição na operação.
31. se a variável aparece nas pré-condições e dentro de um "fato", sendo que "fato" está negado ( não [ fato ] ) ==> usar  $\forall$
32. se a variável aparece nas pré-condições e dentro de um "fato" não-negado ==> usar  $\exists$
33. se a variável aparece nas pós-condições e dentro de um "fato" não-negado ==> usar  $\exists$
34. variáveis que caracterizam argumentos de saída ==> usar  $\exists$



35. se a variável aparece nas pós-condições e dentro de um "fato", sendo que "fato" está negado ( não [ fato ] ) ==> usar  $\forall$

Exemplos:

A)

**Linguagem Comum**

1. Matrícula de aluno:
2. Recebe pedido de matrícula
3. Verifica se aluno está cadastrado
4. Verifica se aluno não tem débito
5. Verifica se não há duas disciplinas com mesmo horário no pedido de matrícula
6. Para cada disciplina do pedido
7. Verifica se aluno tem pré-requisitos da disciplina
8. Verifica se a disciplina foi oferecida
9. Registra matrícula

**Linguagem Formal**

**Heurísticas Utilizadas**

- |   |            |
|---|------------|
| 1. 2. Matrícula_de_aluno(p):                  | 1, 9, 11   |
| $\forall$ db, d1, d2, h1, h2                  | 30, 32, 31 |
| $\exists$ nd, na, d, a, dpr, c,<br>ncr1, ncr2 |            |

**Pré-condições:**

- pedido(p), tem(p,nd),
- tem(p,na), nome(nd), nome(na),
- disc(d), aluno(a),
- tem(d,nd), tem(a,na),

**Obs:** pedido é argumento de entrada; pedido tem nome do aluno e nomes das disciplinas requeridas para matrícula.

- |              |          |
|--------------|----------|
| 3. aluno(a), | 2, 8, 22 |
|--------------|----------|

**Obs:** não é necessária.

- |  |         |
|--|---------|
| 4. não [ tem(a,db) e<br>débito(db) ],  | 2, 7, 9 |
| 5. não [ disc(d1),<br>disc(d2), d1 $\neq$ d2, hor(h1),<br>hor(h2), h1=h2, tem(d1,h1), tem(d2,h2),<br>tem(p,d1), tem(p,d2) ], |         |

6. **Obs:** a expressão "para cada" é desconsiderada.

- |   |          |
|---|----------|
| 7.a) pré_req(dpr), tem(d,dpr),<br>disc(dpr), cursou(a,dpr),<br>com(cursou(a,dpr),c),<br>conceito_de_aprovação(c), | 2, 9, 11 |
|---|----------|

- |   |              |
|---|--------------|
| b) tem(a,ncr1), n_cred(ncr1),<br>n_cred(ncr2), exige(d,ncr2), ncr1 $\geq$ ncr2, | 2, 9, 25, 12 |
|---|--------------|

**Obs:** esta ação foi definida por duas outras ações; a) verifica se aluno cursou com conceito de aprovação as disciplinas pré-requisito da disciplina requerida; b) verifica se o aluno tem, no mínimo, o mesmo número de créditos exigidos pela disciplina requerida; os adjetivos "exigidos" e "requerida" servem para distinguir de "créditos da disciplina" e de "disciplina pré-requisito", respectivamente.

8. oferecida(d). 2, 17, 21  
 9.  
**Pós-condições:** matricula(a,d) 4, 14  
**Obs:** foi necessário verificar as relações do nome (sujeito e complemento verbal).

B)

**Linguagem Comum**

1. Alterar endereço do paciente
2. Recebe identificação do paciente
3. Recebe novo endereço
4. Verifica se paciente está cadastrado
5. Substitui o velho endereço do paciente pelo novo endereço do paciente

**Linguagem Formal**

**Heurísticas Utilizadas**

- |  |              |
|--|--------------|
| 1.2.3. Alteração_de_end_de_pac(e1, id):  | 1            |
| ∀ e2   | 35           |
| 4. <b>Prê:</b><br>paciente(id).  | 2, 8         |
| 5. <b>Pós:</b><br>end(e1), tem(id, e1),<br>não [ end(e2), tem(id,e2), e2≠e1 ]. | 3, 6, 11, 29 |

C)

**Linguagem Comum**

1. Marcar consulta
2. Recebe nome do paciente e nome do médico
3. Recebe horário da consulta
4. Verifica se paciente está cadastrado
5. Verifica se médico existe
6. Verifica se horário da consulta coincide com algum horário que o médico tem disponível
7. Anota consulta

**Linguagem Formal**

**Heurísticas Utilizadas**

- |   |               |
|---|---------------|
| 1.2.3. Marcar_consulta(np, nm, h):                                      | 1             |
| ∃ p, m, h2  | 32            |
| 4. <b>Prê:</b><br>pac(p), nome(np), tem(p,np),                          | 2, 8          |
| 5. med(m), nome(nm), tem(m,nm),   | 2, 8          |
| 6. horário(h), horário(h2), h=h2,<br>tem_disp(m,h2).                    | 2, 12, 28, 27 |
| <b>Obs:</b> "da consulta" e "algum" são usados para distinção de nomes. |               |
| 7. <b>Pós:</b><br>consulta(p,m), em(consulta(p,m),h).                   | 4, 14         |

D)

**Linguagem Comum**

1. Desmarcar consulta

2. Recebe nome do paciente, nome do médico e horário da consulta para desmarcar
3. Desmarca esta consulta

**Linguagem Formal**

**Heurísticas Utilizadas**

- |   |           |
|---|-----------|
| 1.2. Desmarcar_consulta(np, nm, h):   | 1         |
| Obs: "esta" e "da consulta para desmarcar" são adjetivos para distinção de nomes.             |           |
| $\forall p, m$  | 35        |
| 3. Pós:   | 5, 14, 29 |
| não [ consulta(p,m), em(consulta(p,m),h), pac(p), med(m), horário(h), tem(p,np), tem(m,nm) ]. |           |

E)

**Linguagem Comum**

1. Listar médicos por especialidade
2. Recebe especialidade requisitada
3. Coloca na lista exterior nomes dos médicos que têm esta especialidade

**Linguagem Formal**

**Heurísticas Utilizadas**

- |  |            |
|--|------------|
| 1.2. Lista_médicos(e):nm.                                  | 1          |
| $\dagger nm, m$  | 34, 32     |
| 3. Pré:  |            |
| med(m), tem(m,e), especialidade(e), nome(nm), tem(m,nm).   | 16, 19, 11 |
| Obs: "esta" é adjetivo temporário para distinção de nomes. |            |

F)

**Linguagem Comum**

1. Empréstimo de materiais
2. Recebe identificação do usuário e identificação do material
3. Recebe data atual
4. Verifica se material está cadastrado
5. Verifica se usuário existe
6. Verifica se usuário não tem débito
7. Verifica se material não foi emprestado a outro usuário
8. Verifica se material não é de consulta local
9. Calcula a data de devolução do empréstimo, que é igual a data atual mais uma semana
10. Registra empréstimo e a devolução

**Linguagem Formal**

**Heurísticas Utilizadas**

- |                                       |     |
|---------------------------------------|-----|
| 1.2.3.                                | 1   |
| Empréstimo_de_materiais(idu, idm, d): |     |
| $\forall dv, idu2$                    | 31  |
| 4. Pré:                               |     |
| material(idm),                        | 2,8 |
| 5. usuário(idu),                      | 2,8 |
| 6. não [débito(db), tem(idu,db) ]     | 2,9 |

7. não [ usuário(idu2), retirou(idu2,idm), idu2≠idu ], 2, 9, 13  
 Obs: "ser emprestado" é a forma passiva de "retirar".
8. não [ de\_consulta\_local(idm) ]. 2, 21, 28, 17  
 Obs: considerar "de consulta local" como adjetivo.
9. Pós: 30  
 data(dd), de\_devolução(dd), data(d),  
 dd = d + uma\_semana
10. retirou(idu,idm), data(d), 4,14  
 em(retirou(idu,idm),d),  
 é\_devolvido\_em(retirou(idu,idm),dd).
- Obs: "empréstimo" é sinônimo de "retirou" e "devolução" é sinônimo de "é\_devolvido\_em".

## G)

### Linguagem Comum

1. Reservar material
2. Recebe identificação do usuário e do material
3. Verifica se usuário e material estão cadastrados
4. Verifica se material está emprestado a outro usuário
5. Verifica se não há reserva do material a outro usuário
6. Registra reserva

### Linguagem Formal

### Heurísticas Utilizadas

- 1.2. Reservar\_material(idu, idm): 1  
 † idu2 ∀ idu3 32, 31
3. Pré: 2,8  
 usuário(idu), material(idm),
4. retirou(idu2,idm), usuário(idu2), 2, 8, 13  
 idu≠idu2,
- Obs: "está emprestado" é a passiva de "retirou".
5. não [ reservou(idu3,idm), 2, 9, 13, 15, 7, 28  
 usuário(idu3), idu≠idu3 ].
6. Pós: 4, 14  
 reservou(idu,idm).
- Obs: é necessário ver as ligações das preposições "do" e "a"; "reserva" é sinônimo de "reservou".

## H)

### Linguagem Comum

1. Devolução de material
2. Recebe identificação do material
3. Retira empréstimo

### Linguagem Formal

### Heurísticas Utilizadas

- 1.2. Devolução\_de\_material(idm): 1  
 † u 35
3. Pós: 5, 14  
 não [ retirou(u,idm), material(idm),  
 usuário(u) ].
- Obs: "empréstimo" é sinônimo de "retirou".

I)

#### Linguagem Comum

1. Registra conceitos finais das disciplinas
2. Recebe identificação do aluno, identificação da disciplina e conceito
3. Verifica se aluno se matriculou na disciplina
4. Registra que aluno cursou a disciplina com conceito

#### Linguagem Formal

#### Heurísticas Utilizadas

- 1.2. Registra\_conceitos\_finais(ida,idd,c): 1
3. **Prê:** 2,9  
aluno(ida), disc(idd),  
matricula(ida,idd).
- Obs:** "matricular-se em" é sinônimo de "matricula".
4. **Pôs:** 4, 9, 24  
cursou(ida,idd), conceito(c),  
com(cursou(ida,idd),c).

#### 5.4.3 Parte de Restrições

É apresentado, a seguir, o conjunto de heurísticas para a transformação de restrições da Linguagem Comum para a Linguagem Formal.

Este conjunto foi dividido em heurísticas básicas e específicas. A diferença é que as específicas, que procuram identificar os padrões de restrições como definido na Linguagem Comum, devem fazer uso das básicas sempre que necessário.

#### Heurísticas básicas:

- a) nome adjetivo ==> nome(x) e adjetivo(x).
- b) nome\_1 SEM nome\_2 ("sem" = "que não tem") ==> NÃO [ tem(x,y) e nome\_1(x) e nome\_2(y) ].
- c) MESMO nome ==> nome(x), nome(y) e x=y.
- d) nome\_1 QUE verbo nome\_2 ==> verbo(x,y) e nome\_1(x) e nome\_2(y).
- e) MAIS DE UM/UMA nome; OUTRO nome ==> nome(x) e nome(y) e x≠y.
- f) nome\_1 COM nome\_2; nome\_1 QUE TEM nome\_2 ==> tem(x,y) e nome\_1(x) e nome\_2(y).

g) (se o verbo é "ser") verbo(x,y) e complemento(y) ==> complemento(x).

h) quantificação de variáveis:

variáveis antes do sinal "-->" ==> usar  $\forall$

variáveis após "-->" e dentro de "fatos" negados ==>  $\forall$

variáveis após "-->", não negadas ==>  $\exists$

i) nome\_1 preposição nome\_2 ==> ver a ligação entre "nome\_1" e "nome\_2" (se há verbo associando-os, se um deles é adjetivo ou adjunto adverbial do outro).

j) nome de ação ==> ver as ligações (sujeito, complemento verbal e adjunto adverbial).

### Heurísticas Específicas:

Os números informam o padrão identificado, conforme numeração de padrões das restrições em Linguagem Comum. Seguindo os números, é apresentada a forma como deve ser especificada a restrição.

1.  $\forall x, \exists y$   
sujeito(x) --> complemento(y) e verbo(x,y).

2.  $\forall x, y$   
complemento(y) e verbo(x,y) --> não [ sujeito(x) ].

5. a) (para casos em que "n" = 1)

$\forall x, y, z$   
sujeito(x) e complemento(y) e verbo(x,y) -->  
não [ complemento(z) e verbo(x,z) e  $z \neq y$  ].

b) (para casos em que "n" > 1)

$\forall x, y \exists y_1, \dots, y_n$   
sujeito(x) e complemento(y) e verbo(x,y) -->  
não [ complemento(y<sub>1</sub>) e verbo(x,y<sub>1</sub>) e  $y \neq y_1$  e  
complemento(y<sub>2</sub>) e verbo(x,y<sub>2</sub>) e  $y \neq y_1$  e  $y \neq y_2$  e  $y_1 \neq y_2$  e  
... complemento(y<sub>n</sub>) e verbo(x,y<sub>n</sub>) e  $y \neq y_1$  e ... e  
 $y \neq y_n$  e  $y_1 \neq y_2$  e ... e  $y_{n-1} \neq y_n$  ].

6.  $\forall x, \exists y$   
sujeito(x) --> complemento(y) e verbo(x,y).

7. a)  $\forall x, y$   
complemento(y) e verbo(x,y) --> sujeito(x)

b)  $\forall x, y$   
complemento(y) e verbo(x,y) --> sujeito\_1(x) ou  
sujeito\_2(x) ou ...



8. a)  $\forall x, y, z$   
sujeito\_1(x) e sujeito\_2(y) e verbo\_1(y,x) -->  
(não) [ complemento(z) e verbo\_2(x,z) ].

b)  $\forall x, y, z$   
sujeito(x) e complemento\_1(y) e verbo\_1(x,y) -->  
(não) [ complemento\_2(z) e verbo\_2(x,z) ].

**Observação:** usar  $\dagger z$ , se for usado o "não".

9. a)  $\forall x$   
sujeito(x) --> não [ complemento\_1(x) ].

b)  $\forall x$   
sujeito(x) --> não [ complemento\_1(x) e complemento\_2(x) ].

10. a)  $\forall x, y, z$   
sujeito(x) e complemento\_2(y) e verbo\_2(x,y) -->  
não [ complemento\_1(z) e verbo\_1(x,z) ].

b)  $\forall x, y \dagger z$   
sujeito(x) e complemento\_2(y) e verbo\_2(x,y) -->  
complemento\_1(z) e verbo\_1(x,z).

11.  $\forall x, y$   
sujeito(x) e verbo(x,y) --> complemento(y).

12. a)  $\forall x, y$   
sujeito(x) --> não [ complemento(y) e  $x > y$  ].

b)  $\forall x, y$   
sujeito(x) --> não [ complemento(y) e  $x < y$  ].

13.  $\forall y_1, \dots, y_{n+1}$   
complemento(y1) --> não [ complemento(y2) e  $y_1 \neq y_2$  e ...  
e complemento( $y_{n+1}$ ) e  $y_{n+1} \neq y_1$  e ...  
e  $y_{n+1} \neq y_n$  ].

17. a)  $\forall x, y$   
sujeito(x) e complemento\_1(y) e verbo(x,y) -->  
complemento\_2(x).

18. a)  $\forall x, y$   
complemento\_1(y) e verbo(x,y) --> complemento\_2(x).

**Observação:** os padrões 16, 17b e 18b definem restrições dinâmicas, que só podem ser verificadas se as informações do Sistema de Informação fossem marcadas com sinais de tempos (ou momentos) nos quais valem (são verdadeiras).

A seguir são apresentados exemplos de transformações para os exemplos de restrições apresentados anteriormente:



Toda pessoa deve ter um nome.

Número do padrão de restrição	Resultado	Heurísticas usadas
1	$\forall x \exists y \text{ pessoa}(x) \rightarrow \text{nome}(y) \text{ e } \text{tem}(x,y)$	1

Toda pessoa deve ter um nome, no máximo.

Número do padrão de restrição	Resultado	Heurísticas usadas
1 e 5	$\forall x,y,z \text{ pessoa}(x) \text{ e } \text{nome}(y) \text{ e } \text{tem}(x,y) \rightarrow \text{não} [ \text{nome}(z) \text{ e } \text{tem}(x,z) \text{ e } z \neq y ]$	5, A

Toda pessoa deve ter 1 telefone para recados, no mínimo.

Número do padrão de restrição	Resultado	Heurísticas usadas
6	$\forall x \exists y \text{ pessoa}(x) \rightarrow \text{telefone}(y) \text{ e } \text{para\_recados}(y) \text{ e } \text{tem}(x,y)$	6

Todo aluno não pode retirar periódicos.

Número do padrão de restrição	Resultado	Heurísticas usadas
2	$\forall x,y \text{ periódico}(y) \text{ e } \text{retirou}(x,y) \rightarrow \text{não} [ \text{aluno}(x) ]$	2

Somente professores podem retirar periódicos.

Número do padrão de restrição	Resultado	Heurísticas usadas
7a	$\forall x,y \text{ periódico}(y) \text{ e } \text{retirou}(x,y) \rightarrow \text{professor}(x)$	7a

Somente professores e funcionários podem reservar periódicos.

Número do padrão de restrição	Resultado	Heurísticas usadas
7b	$\forall x,y \text{ periódico}(y) \text{ e } \text{reservou}(x,y) \rightarrow \text{professor}(x) \text{ ou } \text{funcionário}(x)$	7b

Materiais que usuários retiram não podem ser de consulta local.

Número do padrão de restrição	Resultado	Heurísticas usadas
-------------------------------	-----------	--------------------

8a  $\forall x,y$  material(x) e usuário(y) e retirou(y,x) --> não [ de\_consulta\_local(x) ] 8a, G

Usuários que tem débito não podem retirar material.

<b>Número do padrão de restrição</b>	<b>Resultado</b>	<b>Heurísticas usadas</b>
8b	$\forall x,y,z$ usuário(x) e débito(y) e tem(x,y) --> não [ material(z) e retirou(x,z) ]	8b

Aluno com bolsa deve estar em um projeto de pesquisa.

<b>Número do padrão de restrição</b>	<b>Resultado</b>	<b>Heurísticas usadas</b>
8c	$\forall x,y \exists z$ aluno(x) e bolsa(y) e tem(x,y) --> projeto(z) e tem(z,x)	8b, F

**Observação:** "está em" é a forma passiva de "tem", com este associando projetos e alunos.

Funcionário não pode ser candidato ao mesmo tempo.

<b>Número do padrão de restrição</b>	<b>Resultado</b>	<b>Heurísticas usadas</b>
9a	$\forall x$ funcionário(x) --> não [ candidato(x) ]	9a

Usuário não pode ser professor e aluno ao mesmo tempo.

<b>Número do padrão de restrição</b>	<b>Resultado</b>	<b>Heurísticas usadas</b>
9b	$\forall x$ usuário(x) --> não [ professor(x) e aluno(x) ]	9b

Usuários não podem retirar material se têm débito.

<b>Número do padrão de restrição</b>	<b>Resultado</b>	<b>Heurísticas usadas</b>
10a	$\forall x,y,z$ usuário(x) e débito(y) e tem(x,y) --> não [ material(z) e retirou(x,z) ]	10a

Alunos devem estar em projetos de pesquisa se têm bolsa

<b>Número do padrão de restrição</b>	<b>Resultado</b>	<b>Heurísticas usadas</b>
--------------------------------------	------------------	---------------------------

10b  $\forall x,y \exists z$  aluno(x) e bolsa(y) e tem(x,y) --> projeto(z) e tem(z,x) 10b

**Observação:** "está em" é a forma passiva de "tem", com este associando projetos e alunos.

Consultas devem ocorrer somente nos horários para consulta que os médicos têm.

Número do padrão de restrição	Resultado	Heurísticas usadas
11	$\forall x,y,z \exists w$ consulta(x,y) e em(consulta(x,y),z) --> horário(z) e médico(w) e tem(w,z)	11, D

O número total de alunos que se matriculam em uma disciplina não pode ser maior que o número de vagas da disciplina.

Número do padrão de restrição	Resultado	Heurísticas usadas
12a	$\forall x,y,z,w$ número_total(w) e de_alunos(w) e aluno(x) e matricula(x,y) e disciplina(y) --> não [ número(z) e de_vagas(z) e tem(y,z) e $w > z$ ]	12a, A, D

**Observação:** "matricular-se em" é sinônimo de "matricula".

Salários não podem ser menor que salário-mínimo.

Número do padrão de restrição	Resultado	Heurísticas usadas
12b	$\forall x,y$ salário(x) --> não [ salário_mínimo(y) e $x < y$ ]	12b, C

Não pode haver mais de uma disciplina oferecida com mesmo local e com mesmo horário em um pedido de matrícula.

Número do padrão de restrição	Resultado	Heurísticas usadas
13a	$\forall y_1,y_2,x_1,z_1,x_2,z_2,w$ disciplina(y1), local(x1), oferecida(y1), horário(z1), tem(y1,x1), tem(y1,z1), pedido(w), tem(,y1) --> não [ tem(w,y2), disciplina(y2), local(x2), oferecida(y2), horário(z2), tem(y2,x2), tem(y2,z2), $y_1 \neq y_2, x_1 = x_2, z_1 = z_2$ ]	13, F, I, A

**Observação:** considera-se "n=1".

Não pode haver mais de um diretor.

<b>Número do padrão de restrição</b>	<b>Resultado</b>	<b>Heurísticas usadas</b>
13b	$\forall y_1, y_2$ diretor( $y_1$ ) $\rightarrow$ não [ diretor( $y_2$ ) e $y_1 \neq y_2$ ]	13

Usuário só pode retirar periódico se é professor.

<b>Número do padrão de restrição</b>	<b>Resultado</b>	<b>Heurísticas usadas</b>
17a	$\forall x, y$ usuário( $x$ ) e periódico( $y$ ) e retirou( $x, y$ ) $\rightarrow$ professor( $x$ )	17a

Só pode receber salário quem é funcionário.

<b>Número do padrão de restrição</b>	<b>Resultado</b>	<b>Heurísticas usadas</b>
18a	$\forall x, y$ recebe( $x, y$ ) e salário( $y$ ) $\rightarrow$ funcionário( $x$ )	18a

Outros exemplos:

Alunos não podem se matricular em mais de uma disciplina oferecida com mesmo horário.

<b>Número do padrão de restrição</b>	<b>Resultado</b>	<b>Heurísticas usadas</b>
	$\forall x, y_1, y_2, z_1, z_2$ disciplina( $y_1$ ), matricula( $x, y_1$ ), disciplina( $y_2$ ), matricula( $x, y_2$ ), $y_1 \neq y_2$ , oferecida( $y_1$ ), oferecida( $y_2$ ), horário( $z_1$ ), horário( $z_2$ ), $z_1 = z_2$ , tem( $y_1, z_1$ ), tem( $y_2, z_2$ ) $\rightarrow$ não [ aluno( $x$ ) ]	2, E, A, F, C

**Observação:** "matricular-se em" = "matricula".

Não pode haver mais de um paciente que faz consulta ao mesmo médico, no mesmo horário.

<b>Número do padrão de restrição</b>	<b>Resultado</b>	<b>Heurísticas usadas</b>
2	$\forall y_1, y_2, x_1, z_1, x_2, z_2$ paciente( $y_1$ ), consulta( $y_1, x_1$ ), médico( $x_1$ ), horário( $z_1$ ), em(consulta( $y_1, x_1$ ), $z_1$ ) $\rightarrow$ não [ paciente( $y_2$ ), $y_1 \neq y_2$ ,	13, D, C, I

consulta(y2,x2), x1=x2, horário(z2),  
z1=z2, em(consulta(y2,x2),z2) ]

**Observação:** "fazer consulta" é sinônimo de "consulta".

## 5.5 Definição das Heurísticas para Transformação da Linguagem Formal para a Linguagem Comum (parafraseador)

Seguindo a idéia de [Bal85], estas heurísticas funcionam como um "parafraseador" para as especificações formais (documentos escritos na Linguagem Formal assumida).

### 5.5.1 Parte de Dados e Estrutura

1. verbo(x,y) e sujeito(x) e complemento\_verbal(y) ==>  
sujeito verbo complemento\_verbal.

Exemplo: paciente(x) e médico(y) e consulta(x,y) ==>  
pacientes consultam médicos

**Observação:** deve-se usar o plural como na Linguagem Comum.

2. preposição( verbo(x,y), z) e sujeito(x) e  
complemento\_verbal(y) e adjunto\_adverbial(z) ==> sujeito verbo  
complemento\_verbal preposição adjunto\_adverbial.

Exemplo: em( consulta(x,y), z) e paciente(x) e médico(y) e  
data(z) ==> pacientes consultam médicos em datas.

3. sujeito(x), verbo\_intransitivo\_mais\_preposição\_introduzindo\_  
adjunto\_adverbial( verbo(x,y), z), adjunto\_adverbial(y) ==>  
sujeito verbo preposição adjunto

### 5.5.2 Parte de Operações

São apresentados os padrões a serem identificados e os possíveis tratamentos (transformações para Linguagem Comum):

1. nome(x) com "x" não aparecendo em associação do tipo  
nome\_2(x,y) ou nome\_2(y,x) ==> nome ESTA CADASTRADO.

2. nome\_1(x,y), nome\_2(x), nome\_3(y) ==> nome\_2 nome\_1 nome\_3.
3. nome(x1) e nome(x2) e x1≠x2 ==> OUTRO nome.
4. NÃO fato ==> NÃO ACONTECE QUE fato; NÃO HA fato; se fato = "sujeito verbo complemento", então usar "sujeito NÃO verbo complemento".
5. nome\_2(nome\_1(x,y),z) ==> x nome\_1 y nome\_2 z.
6. constantes ou variáveis não-quantificadas, representando argumentos de entrada ==> procurar o tipo de entidade numa forma "nome(variável)".
7. usar "verifica se" nas pré-condições (para cada linha originada por uma ação da Linguagem Comum).
8. argumentos de entrada ==> RECEBE argumentos.
9. x verbo y (ou uma referência a y) e z TEM y ==> x verbo y DE z; referência a y DE z.
10. x≠y e nome(x) e nome(y) ==> DOIS/DUAS nomes QUE; OUTRO nome.
11. nome(x) e nome(y) e x=y ==> MESMO nome.
12. x verbo\_1 y (ou referência a y) e y verbo\_2 z ==> x verbo\_1 y QUE verbo\_2 z; referência a y QUE verbo\_2 z.
13. nome\_1(x) e nome\_2(x), sendo um dos nomes adjetivos ==> nome\_1 nome\_2 (ou nome\_2 nome\_1, o que ficar melhor).
14. x>y (ou x<=y) ==> x >= (ou x<=) QUE y.
15. x verbo\_1 y (ou referência a y) e z verbo\_2 y, sendo verbo\_2 diferente do verbo "ter" ==> x verbo\_1 y QUE z verbo\_2; referência a y QUE z verbo\_2.
16. x verbo\_1 y e x verbo\_2 z ==> x QUE verbo\_1 y verbo\_2 z; x QUE verbo\_2 z verbo\_1 y.
17. adjetivo(x), estando sozinho em uma linha gerada por uma ação da Linguagem Comum ==> x E/FOI adjetivo.
18. pós-condição: NÃO fato ==> EXCLUI/RETIRA/DESMARCA (INFORMAÇÃO) QUE fato.
19. pós-condição: fato ==> INCLUI/REGISTRA (INFORMAÇÃO) QUE fato; INCLUI/REGISTRA fato.
20. REGISTRA INFORMAÇÃO QUE x TEM y e RETIRA INFORMAÇÃO QUE x TEM DOIS y ==> SUBSTITUI VELHO y DE x PELO NOVO y DE x.
21. operações de consulta ==> COLOCA NA LISTA EXTERIOR argumentos\_de\_saída.

22.  $x = \text{expressão\_matemática}$  (deve conter um operador matemático, no mínimo)  $\implies$  CALCULA  $x$ , QUE É IGUAL A expressão\\_matemática.

Observações:

- explicar linha por linha (conforme foram gerados os grupos de linhas);
- "referência a" pode ser  $>$ ,  $=$ ,  $\neq$ , etc ou "solicita", "recebe", "registra", etc;
- marcar as partes já traduzidas e não usá-las mais, a não ser que definam tipos de argumentos em uma associação;
- as palavras com letras maiúsculas definem palavras-padrão a serem procuradas;
- as palavras com letras minúsculas definem tipos de palavras (ex: adjetivo = palavra com função sintática de adjetivo);
- os parêntesis caracterizam explicações ou palavras opcionais;
- "fato" é uma frase-padrão ou nome, como definido na Linguagem Comum, ou predicados (conectados logicamente por "e" ou "ou"), como definido na Linguagem Formal.

São apresentados a seguir exemplos de transformações feitas sobre os exemplos anteriores de operações em lógica.

**Exemplo A):**

<b>Resultado</b>	<b>Heurísticas Usadas</b>
Matricula de aluno:	
Recebe pedido	8
Verifica se pedido tem nome do aluno e nome da disciplina	7, 2, 9
Verifica se aluno não tem débito	7, 4, 2
Verifica se pedido não tem duas disciplinas que tem mesmo horário	7, 4, 2, 10, 12, 11
Verifica se aluno cursou disciplina pré-requisito da disciplina com conceito de aprovação	7, 2, 9, 5, 13
Verifica se aluno tem número de	7, 2, 14, 15



créditos >= que número de créditos que disciplina exige	
Verifica se disciplina é/foi oferecida	7, 17
Registra que aluno se matriculou em/na disciplina	19, 2

**Observação:** "matricula" é sinônimo de "se matriculou em".

**Exemplo B):**

<b>Resultado</b>	<b>Heurísticas Usadas</b>
Alteração de endereço de paciente	
Recebe endereço	8
Recebe (identificação de) paciente	8
Verifica se paciente está cadastrado	7, 1
Registra que paciente tem endereço	19, 2
Retira informação que paciente tem dois endereços	18, 10

**Observação:** pela heurística de número 20, as duas últimas ações poderiam ser substituídas por "Substitui velho endereço do paciente pelo novo endereço".

**Exemplo C):**

<b>Resultado</b>	<b>Heurísticas Usadas</b>
Marcar consulta	
Recebe nome do paciente	8, 9
Recebe nome do médico	8, 9
Recebe horário	8
Verifica se horário é mesmo que horário que médico tem disponível	7, 11, 15
Registra consulta	19

**Exemplo D):**

<b>Resultado</b>	<b>Heurísticas Usadas</b>
Desmarcar consulta	
Recebe nome do paciente	8, 9
Recebe nome do médico	8, 9
Recebe horário	8
Retira informação que paciente consulta médico em horário	18, 2, 5

**Exemplo E):**

<b>Resultado</b>	<b>Heurísticas Usadas</b>
Listar médicos	
Recebe especialidade	8
Coloca na lista exterior nome do médico que tem especialidade	21, 9, 12

**Exemplo F):**

<b>Resultado</b>	<b>Heurísticas Usadas</b>
Empréstimo de material	
Recebe usuário, material e data	8
Verifica se material está cadastrado	7, 1
Verifica se usuário está cadastrado	7, 1
Verifica se usuário não tem débito	7, 4, 2
Verifica se outro usuário não retirou material	7, 4, 10, 2
Verifica se não acontece que material é de consulta local	7, 4, 17
Calcula data de devolução, que é igual a data atual mais uma semana	22
Registra que usuário retirou material em data e devolve empréstimo em data de devolução	19, 5

**Observação:** utilizar "empréstimo" no lugar de "retirou" (alguns sinônimos ficam melhor que os nomes-padrão correspondentes, em certas frases).

**Exemplo G):**

<b>Resultado</b>	<b>Heurísticas Usadas</b>
Reservar material	
Recebe usuário e material	8
Verifica se usuário está cadastrado	7, 1
Verifica se material está cadastrado	7, 1
Verifica se outro usuário retirou material	7, 10, 2
Verifica se outro usuário não reservou material	7, 10, 4, 2
Registra que usuário reservou material	19, 2

**Exemplo H):**

<b>Resultado</b>	<b>Heurísticas Usadas</b>
------------------	---------------------------

Devolução de material	
Recebe material	8
Retira que usuário retirou material	18, 2

**Exemplo I):**

<b>Resultado</b>	<b>Heurísticas Usadas</b>
Registra conceitos finais	
Recebe aluno, disciplina e conceito	8
Verifica se aluno está cadastrado	7, 1
Verifica se disciplina está cadastrada	7, 1
Verifica se aluno se matriculou em disciplina	7, 2
Registra que aluno cursou disciplina com conceito	19, 5

**Observação:** "se matriculou em" é usado como sinônimo de matrícula.

### 5.5.3 Parte de Restrições de Integridade

Deve-se identificar os formatos das restrições em Linguagem Formal conforme os formatos apresentados nas heurísticas de transformação da Linguagem Comum para a Linguagem Formal (lado direito).

Após devem ser identificados os elementos sintáticos e estes devem ser formatados como indicado nos padrões de restrições na Linguagem Comum.

Exemplo:

$\forall x \exists y$   
paciente(x) --> nome(y), tem(x,y)

(note-se que esta restrição se encaixa no formato 1 da Linguagem Formal).

**Elementos identificados:**

sujeito = "paciente"

complemento = "nome"

verbo = "tem" (usar a forma primitiva "ter").

**Paráfrase (formato 1 da Linguagem Comum):**

Todo paciente deve ter nome.

## 6 ESTUDO DE CASO

Foi realizado um estudo com a Linguagem Comum e as heurísticas, propostas neste trabalho, para apresentação do funcionamento global das idéias aqui discutidas.

Tomou-se como Organização o ambiente de "caixa" de um restaurante, o qual é controlado por um único funcionário. Este funcionário fez o papel de Usuário neste estudo de caso.

### 6.1 Primeira Etapa: Coleta de Dados (obtenção da Linguagem Informal)

Segundo as heurísticas já apresentadas anteriormente, esta etapa é realizada através de entrevistas ao Usuário, com o intuito de coletar informações sobre a Organização.

A entrevista se deu através de perguntas e respostas, como definido nas heurísticas para coleta.

Algumas perguntas não tiveram respostas, por isto não aparecem neste documento.

A seguir, são transcritas as perguntas feitas ao Usuário e suas respectivas respostas.

Qual o objetivo da Organização ?  
Cuidar do caixa do restaurante.

Quais são os Usuários/Clientes da Organização ?  
Garçons; o dono do restaurante; clientes do restaurante.

Quais os serviços prestados pela Organização ?  
Cuidar do caixa, anotar pedidos, fechar contas, fechar o caixa no final do dia.

Que outras funções são necessárias para manter a Organização ?  
Cobrar contas.

Que cadastros, arquivos, registros, formulários e outras informações são mantidos pela Org. ?  
Cardápio de pratos e bebidas, caderno especial com contas penduradas e clientes devedores.

Quais os relatórios internos ou externos da Org. ?  
Lucro do dia.

Quais as informações que constituem o cardápio ?  
Pratos, bebidas e preços.

Quais as informações que constituem o caderno especial ?  
Nome do cliente, valor da conta pendurada.

Qual a finalidade do cardápio ?  
Serve para se verificar o valor de cada prato e bebida.

Qual a finalidade do Caderno Especial ?  
Serve para registrar as "penduras" de contas e verificar os devedores.

Quando é atualizado ou consultado o cardápio ?  
Quando vou fechar uma conta (consulta); quando há inclusão de novos pratos ou bebidas ou exclusão (atualização, porém, não é responsabilidade do caixa).

Quando é atualizado ou consultado o Caderno Especial ?  
Quando há pendura de contas (atualização) ou pagamento de contas penduradas (atualização) e quando há novas penduras, para verificar se o cliente já tem conta pendurada.

Funções identificadas:

1. Cuidar do caixa (função geral)
2. Anotar pedidos
3. Fechar contas
4. Cobrar contas
5. Fechar caixa
6. Pendurar conta
7. Pagamento de contas penduradas
8. Inclusão de novos pratos e bebidas no cardápio (não é de responsabilidade do caixa)
9. Exclusão de novos pratos e bebidas no cardápio (não é de responsabilidade do caixa)

Para cada função identificada, quais as informações que devem estar disponíveis para que a função seja desempenhada ?

2. Anotar pedido: prato ou bebida, valor, número da mesa.
3. Fechar conta: valor dos pratos e bebidas (no cardápio).
4. Cobrar conta: valor pago pelo cliente, valor da conta.
6. Pendurar conta: nome do cliente, valor da conta.
7. Pagamento de pendura: nome do cliente, valor da conta pendurada.

Para cada função identificada, quais as ações a serem tomadas ?

2. Anotar pedido: o garçon me pede para (eu) anotar os pedidos de uma mesa, então anoto (o pedido).
3. Fechar conta: o garçon me pede para (eu) fechar a conta de uma mesa; vejo (no cardápio) o preço dos pratos e bebidas pedidos (e anotados); somo os valores e dou o total (da conta) ao garçon.

4. Cobrar conta: recebo o dinheiro ou cheque (para pagamento da conta); vejo se há troco.
5. Fechar caixa: conto o dinheiro no caixa; diminuo o total do valor de início (do caixa) e forneço o lucro ao patrão.
6. Pendurar conta: recebo o nome do cliente, vejo se ele não é devedor e anoto o nome do cliente, o valor da conta e a data da pendura no caderno especial.
7. Pagamento de pendura: o cliente vem e diz que quer pagar uma conta que deve; peço o seu nome e vejo o valor (da conta) no caderno especial (com penduras); dou o valor ao cliente e recebo o pagamento; tiro o nome do cliente devedor do caderno (especial).

Para cada função identificada, que condições devem ser avaliadas antes de fazê-la ?

3. verificar se tudo foi anotado (se está correto).
4. se o pagamento for em cheque, devo ver se está preenchido corretamente e pergunto ao patrão se aceita o cheque.
5. todas as mesas devem estar fechadas.
6. o cliente não pode ser devedor.

Para cada função identificada, que exceções podem ocorrer e com que se deve tomar cuidado especial (casos anormais) ?

3. quando o valor de um prato não está no cardápio, recorro ao patrão.
4. quando não há troco, peço ao dono do restaurante; se o cliente não tem dinheiro, penduro a conta dele.

Exemplos de funções:

2. Garçon pede ao funcionário para anotar uma pizza calabresa e um cerveja para a mesa 2.

Funcionário anota uma cerveja e uma pizza calabresa na conta da mesa 2.

3. Garçon pede para fechar a mesa 2.

Funcionário vê na conta o que foi pedido (uma cerveja e uma pizza calabresa) e procura no cardápio os valores (20 e 100 cruzeiros). Funcionário soma os valores e dá ao garçon a nota com o valor total da conta (140 cruzeiros).

4. Garçon dá 150 cruzeiros e a nota de mesa 2.

Funcionário calcula o troco e o dá para o garçon (10 cruzeiros), dizendo de que mesa é.

Palavras-chave identificadas:

caixa, cliente, garçon, dono do restaurante, bebida, prato, conta, valor do pedido, troco, lucro do caixa, mesa fechada, pessoa que deve ao restaurante, valor de um prato.

Verbos identificados:

cuidar, anotar, fechar, cobrar, contar, verificar, pendurar, pagar.

Há sinônimos para **cliente** ?



Freguês.

Há sinônimos para **dono do restaurante** ?  
Patrão.

Há sinônimos para **conta** ?  
Nota.

Há sinônimos para **valor do pedido** ?  
Preço do pedido, total do pedido.

Há sinônimos para **prato** ?  
Comida.

Há sinônimos para **pessoa que deve ao restaurante** ?  
Devedor, cliente devedor, aquele que "pendura" ou tem conta pendurada.

Há sinônimos para **valor de um prato** ?  
Preço de um prato.

Há sinônimos para **mesa fechada** ?  
Mesa encerrada.

O que é **pedido** ?  
É a relação de pratos e bebidas que o garçon pede, que o cliente quer.

O que é **conta** ?  
É a relação de pratos e bebidas com o preço e o total ou só o total de uma mesa.

O que é **lucro do caixa** ?  
É o total de dinheiro no caixa no fim do dia menos o troco que havia no caixa no início do dia.

O que é **mesa fechada** ?  
Mesa com a conta encerrada, isto é, com o total somado e pago.

O que é **valor do pedido** ?  
É o total dos valores dos pratos e bebidas do pedido.

O que é **troco** ?  
É a diferença entre o dinheiro que o cliente dá (valor pago) e o valor da conta.

Para que serve **valor de um prato** ?  
Para somar o total da conta.

Onde é utilizado **valor de um prato** ?  
Para somar o total da conta.

Quem fornece **pedido** ?  
O garçon.

Quem fornece **valor de um prato** ?  
Cardápio ou dono do restaurante.

Quais os complementos verbais mais utilizados com **anotar** ?  
Pedidos.

Quais os complementos verbais mais utilizados com **fechar** ?  
Caixa, conta, mesa.

Quais os complementos verbais mais utilizados com **cobrar** ?  
Conta.

Quais os complementos verbais mais utilizados com **contar** ?  
Dinheiro do caixa.

Quais os complementos verbais mais utilizados com **verificar** ?  
Troco, valor dos pedidos, lucro.

Quais os complementos verbais mais utilizados com **pendurar** ?  
Conta.

O que é **anotar** (pedidos) ?  
Anotar a relação de pratos e bebidas que o cliente quer.

O que é **fechar conta** ?  
Somar os valores dos pedidos.

O que é **fechar caixa** ?  
Verificar o lucro.

O que é **cobrar** ?  
Receber pagamento (valor ou dinheiro).

O que é **pendurar** ?  
Não pagar a conta.

Qual o substantivo apropriado para **anotar** ?  
Registro de.

Qual o substantivo apropriado para **fechar** ?  
Encerramento de.

Qual o substantivo apropriado para **cobrar** ?  
Cobrança de.

Qual o substantivo apropriado para **contar** ?  
Contagem.

Qual o substantivo apropriado para **verificar** ?  
Verificação.

Qual o substantivo apropriado para **pendurar** ?  
Pendura.

## 6.2 Segunda Etapa: Transformação para a Linguagem Comum

Deve ser realizada com base nas heurísticas já apresentadas.

Parte de Dados e Estrutura:

Mesas têm contas.

Contas pertencem a mesas.

Cardápios têm pratos.

Pratos estão em cardápios.

Cardápios têm bebidas.

Bebidas estão em cardápio.

Pratos têm preços de pratos.

Preços de pratos definem pratos.

Bebidas têm preços de bebidas.

Preços de bebidas definem bebidas.

Caixas têm valores em dinheiro.

Valores em dinheiro pertencem a caixas.

Caixas têm valores de início.

Valores de início pertencem a caixas.

Mesas têm números.

Números identificam mesas.

Pedidos têm valores de pedidos.

Valores de pedidos definem pedidos.

Contas têm valores de contas.

Valores de contas definem contas.

Clientes penduram contas.

Contas são penduradas por clientes.

Pedidos têm pratos.

Pratos são anotados em pedidos.

Pedidos têm bebidas.

Bebidas são anotadas em pedidos.

Caixas têm lucros.

Lucros pertencem a caixas.

Cadernos Especiais têm contas penduradas.

Contas penduradas pertencem a cadernos especiais.

Penduras acontecem em datas.

Datas marcam penduras.

Contas têm pedidos.

Pedidos pertencem a contas.

Clientes têm nomes.

Nomes identificam clientes.

Dicionário de Termos: (parte de sinônimos)

freguês = cliente = pessoa.

patrão = dono do restaurante.

valor = preço (para pratos, bebidas, contas e pedidos).

valor da conta = valor total da conta = preço total da conta.

valor do pedido = valor total do pedido = preço total do pedido.

cliente que pendura conta = devedor = cliente devedor = pessoa  
que deve ao restaurante = cliente que tem conta pendurada.

prato = comida.

conta fechada = conta encerrada = mesa fechada.

anotar = registro de.

fechar = encerrar = encerramento de.

cobrar = cobrança de.

contar = contagem de.

verificar = verificação de.

pendurar = pendura de.

acontecer em = em

ter conta pendurada = pendurar conta.

pagar = pagamento de.

(mais os verbos e suas formas passivas ou sinônimos, como visto  
acima na parte de dados)

Parte de Operações:

Anotar pedido

Recebe pedido (com pratos e bebidas)

Recebe número da mesa

Anota pedido da mesa

Fechar conta

Recebe número da mesa

Calcula valor total da conta da mesa, que é igual à  
soma dos valores dos pedidos da mesa (valor do pedido  
da mesa é igual à soma do valor de cada prato do pedido  
mais o valor de cada bebida do pedido)

Coloca na lista exterior valor total da conta

Registra mesa fechada

Cobrar conta

Recebe número da mesa

Recebe valor pago

Calcula troco, que é igual a valor pago menos valor total da  
conta da mesa

Coloca na lista exterior troco

#### Fechar caixa

Calcula lucro, que é igual a valor em dinheiro do caixa  
menos valor de início do caixa  
Coloca na lista exterior lucro

#### Registrar pendura de conta

Recebe nome do cliente, valor da conta e data  
Verifica se cliente não é devedor (não pendurou conta)  
Registra pendura

#### Pagar conta pendurada

Recebe nome do cliente  
Retira pendura do cliente

#### Parte de Restrições de Integridade:

Toda mesa pode ter 1 conta, no máximo  
Toda conta deve pertencer a 1 mesa, no máximo.  
Todo prato deve estar em 1 cardápio.  
Todo prato deve estar no cardápio.  
Toda bebida deve estar no cardápio.  
Todo prato deve ter 1 preço, no máximo.  
Todo preço de prato deve definir 1 prato, no mínimo.  
Toda bebida deve ter 1 preço, no máximo.  
Todo preço de bebida deve definir 1 bebida, no mínimo.  
Valor em dinheiro deve pertencer ao caixa.  
Valor de início deve pertencer ao caixa.  
Toda mesa deve ter 1 número, no máximo.  
Todo número deve identificar 1 mesa, no máximo.  
Todo pedido deve ter 1 valor, no máximo.  
Todo valor de pedido deve definir 1 pedido, no mínimo.  
Toda conta deve ter 1 valor de conta, no máximo.  
Todo valor de conta deve definir 1 conta, no mínimo.  
Todo cliente pode pendurar 1 conta, no máximo.  
Toda conta pode ser pendurada por 1 cliente, no máximo.  
Lucro deve pertencer ao caixa.  
Toda conta pendurada deve pertencer ao caderno especial.  
Toda pendura deve acontecer em 1 data, no máximo.  
Todo pedido deve pertencer a 1 conta, no máximo.  
Todo cliente deve ter 1 nome, no máximo.  
Todo nome deve identificar 1 cliente, no máximo.  
Não pode haver mais de 1 cardápio.  
Não pode haver mais de 1 caderno especial.  
Não pode haver mais de 1 lucro.  
Não pode haver mais de 1 caixa.  
Não pode haver mais de 1 valor em dinheiro.  
Não pode haver mais de 1 valor de início.  
Cliente não pode pendurar conta se pendurou outra conta.

#### Verificação da Linguagem Comum:

Conceitos usados e não gerados ou fornecidos:  
valor de prato e bebida (cardápio)

valor em dinheiro do caixa  
valor de início do caixa

### 6.3 Terceira Etapa: Transformação para a Linguagem Formal

Esta etapa deve ser realizada pelo Analista com ajuda das heurísticas já apresentadas.

Observação: os números entre colchetes informam as heurísticas utilizadas.

Parte de Dados:

mesa(x), conta(y), tem(x,y) [1]  
cardápio(x), prato(y), tem(x,y) [1]  
cardápio(x), bebida(y), tem(x,y) [1]  
prato(x), preço(y), tem(x,y) [1]  
bebida(x), preço(y), tem(x,y) [1]  
caixa(x), valor(y), em\_dinheiro(y), tem(x,y) [1, 4.12]  
caixa(x), valor(y), de\_início(y), tem(x,y) [1, 4.5]  
mesa(x), número(y), tem(x,y) [1]  
pedido(x), valor(y), tem(x,y) [1]  
conta(x), valor(y), tem(x,y), de\_conta(y) [1]  
cliente(x), conta(y), pendura(x,y) [1]  
pedido(x), prato(y), tem(x,y) [1]  
pedido(x), bebida(y), tem(x,y) [1]  
caixa(x), lucro(y), tem(x,y) [1]  
caderno(x), especial(x), conta(y), pendurada(y), tem(x,y) [1]  
pendura(x,y), data(y), acontece\_em(pendura(x,y),z) [2]  
conta(x), pedido(y), tem(x,y) [1]  
cliente(x), nome(y), tem(x,y) [1]

Parte de Operações:

Anotar\_pedidos(p,n): [1]  
‡ p, pr, b, m, c  
**Pós:** pedido(p), prato(pr), bebida(b),  
tem(p,pr), tem(p,b),  
número(n), mesa(m), tem(m,n),  
conta(c), tem(m,c), tem(c,p). [4, 11, 23, 33]

Fechar\_conta(n):v1. [1]  
‡ v1, v2, c, m, v3, p, v4, pr, b  
**Pré:** v1=soma(v2), valor(v1), total(v1),  
número(n), conta(c), mesa(m), tem(m,c), tem(m,n),  
tem(c,v1), valor(v2), pedido(p), tem(p,v2), tem(c,p),  
v2=v3+v4, valor(v3), prato(pr), tem(pr,v3),  
bebida(b), valor(v4), tem(b,v4).  
**Pós:** fechada(m). [30, 17, 11, 16, 4a, 33, 32]



Cobrar\_conta(n,v):t. [1]  
 $\exists m, t, c, vt$   
**Prê:** mesa(m), número(n), tem(m,n), valor(v),  
 pago(v), troco(t),  $t=v-vt$ , valor(vt), total(vt),  
 conta(c), tem(c,vt), tem(m,c). [30, 11, 17, 16, 32]

Fechar\_caixa:l.  
 $\exists l, vd, vi, c$   
**Prê:** lucro(l),  $l=vd-vi$ , valor(vd), em\_dinheiro(vd),  
 caixa(c), tem(c,vd), valor(vi), de\_inicio(vi),  
 tem(c,vi). [30, 11, 20, 17, 16, 32]

Pendurar\_conta(n,v,d): [1]  
 $\forall c, cc2, \exists cc1$   
**Prê:** não [ nome(n), cliente(c), tem(c,n), conta(cc2),  
 pendura(c,cc2) ].  
**Pós:** pendura(c,cc1), valor(v), tem(cc1,v),  
 em(pendura(c,cc1),d), data(d), pendurada(cc1).  
 [2, 7, 4, 14, 31, 33]

Pagar\_conta\_pendurada(n): [1]  
 $\forall c, cc$   
**Pós:** não [ pendura(c,cc), cliente(c), nome(n),  
 tem(c,n), conta(cc), pendurada(cc) ].  
 [5, 14, 35]

Parte de Restrições:

Toda mesa pode ter 1 conta, no máximo.  
 $\forall x, y, z$   
 mesa(x), conta(y), tem(x,y) --> não [ conta(z), tem(x,z),  
 $z \neq y$  ]. [5]

Toda conta deve pertencer a 1 mesa, no máximo.  
 $\forall x \exists y$   
 conta(x) --> mesa(y), tem(y,x). [1]

**Obs:** "pertencer a" é o inverso de "ter".

$\forall x, y, z$   
 conta(x), mesa(y), pertence\_a(x,y) --> não [ mesa(z),  
 pertence\_a(x,z),  $z \neq y$  ]. [5]

Todo prato deve estar em 1 cardápio.

**Obs:** é desnecessária pois só há um cardápio.

Todo prato deve estar no cardápio.  
 $\forall x \exists y$



prato(x) --> cardápio(y), tem(y,x). [1]

Toda bebida deve estar no cardápio.

$\forall x \dagger y$   
bebida(x) --> cardápio(y), tem(y,x). [1]

Todo prato deve ter 1 preço, no máximo.

$\forall x \dagger y$   
prato(y) --> preço(y), tem(x,y). [1]

$\forall x, y, z$   
prato(x), preço(y), tem(x,y) --> não [ preço(z),  
tem(x,z),  $z \neq y$  ]. [5]

Todo preço de prato deve definir 1 prato, no mínimo.

$\forall x \dagger y$   
preço(x), de\_prato(x) --> prato(y), tem(y,x). [6, A, I]

Toda bebida deve ter 1 preço, no máximo.

$\forall x \dagger y$   
bebida(y) --> preço(y), tem(x,y). [1]

$\forall x, y, z$   
bebida(x), preço(y), tem(x,y) --> não [ preço(z),  
tem(x,z),  $z \neq y$  ]. [5]

Todo preço de bebida deve definir 1 bebida, no mínimo.

$\forall x \dagger y$   
preço(x), de\_bebida(x) --> bebida(y), tem(y,x). [6, A, I]

Valor em dinheiro deve pertencer ao caixa.

$\forall x \dagger y$   
valor(x), em\_dinheiro(x) --> caixa(y), tem(y,x). [1, A, I]

Valor de início deve pertencer ao caixa.

$\forall x \dagger y$   
valor(x), de\_início(x) --> caixa(y), tem(y,x). [1, A, I]

Toda mesa deve ter 1 número, no máximo.

$\forall x \dagger y$   
mesa(x) --> número(y), tem(x,y). [1]

$\forall x, y, z$   
mesa(x), número(y), tem(x,y) --> não [ número(z),  
tem(x,z),  $z \neq y$  ]. [5]

Todo número deve identificar 1 mesa, no máximo.

$\forall x \dagger y$   
número(x) --> mesa(y), tem(y,x). [1]

$\forall x, y, z$   
número(x), mesa(y), tem(y,x) --> não [ mesa(z),  
tem(z,x),  $z \neq y$  ]. [5]

Todo pedido deve ter 1 valor de pedido, no máximo.

$\forall x \dagger y$   
pedido(x) --> valor(y), tem(x,y), de\_pedido(y). [1]

$\forall x, y, z$   
pedido(x), valor(y), de\_pedido(y), tem(x,y) -->  
não [ valor(z), tem(x,z),  $z \neq y$  ]. [5]

Todo valor de pedido deve definir 1 pedido, no mínimo.

$\forall x \dagger y$   
valor(x), de\_pedido(x) --> pedido(y), tem(y,x). [6, I, A]

Toda conta deve ter 1 valor de conta, no máximo.

$\forall x \dagger y$   
conta(x) --> valor(y), tem(x,y), de\_conta(y). [1]

$\forall x, y, z$   
conta(x), valor(y), de\_conta(y), tem(x,y) -->  
não [ valor(z), tem(x,z),  $z \neq y$  ]. [5]

Todo valor de conta deve definir 1 conta, no mínimo.

$\forall x \dagger y$   
valor(x), de\_conta(x) --> conta(y), tem(y,x). [6, I, A]

Todo cliente pode pendurar 1 conta, no máximo.

$\forall x, y, z$   
cliente(x), conta(y), pendura(x,y) --> não [ conta(z),  
pendura(x,z),  $z \neq y$  ]. [5]

Toda conta pode ser pendurada por 1 cliente, no máximo.

$\forall x, y, z$   
conta(x), cliente(y), pendura(y,x) --> não [ cliente(z),  
pendura(z,x),  $z \neq y$  ]. [5]

Lucro deve pertencer ao caixa.

$\forall x \dagger y$   
lucro(x) --> caixa(y), tem(y,x). [1]

Toda conta pendurada deve pertencer ao caderno especial.

$\forall x \dagger y$   
conta(x), pendurad(x) --> caderno(y), especial(y),  
tem(y,x). [1, A]

Toda pendura deve acontecer em 1 data, no máximo.

$\forall x, w \dagger y$   
pendura(x,w), cliente(x), conta(w) --> data(y),  
em(pendura(x,w), y). [1, J]

**Obs:** "acontecer\_em" é sinônimo de "em".

$\forall x, w, y, z$   
pendura(x,w), cliente(x), conta(w), data(y),  
em(pendura(x,w), y) --> não [ data(z), em(pendura(x,w), z),  
 $z \neq y$  ]. [5, J]

Todo pedido deve pertencer a 1 conta, no máximo.

$\forall x \exists y$   
pedido(x) --> conta(y), tem(y,x). [1]

$\forall x, y, z$   
pedido(x), conta(y), tem(y,x) --> não [ conta(z),  
tem(z,x),  $z \neq y$  ]. [5]

Todo cliente deve ter 1 nome, no máximo.

$\forall x \exists y$   
cliente(x) --> nome(y), tem(x,y). [1]

$\forall x, y, z$   
cliente(x), nome(y), tem(x,y) --> não [ nome(z),  
tem(x,z),  $z \neq y$  ]. [5]

Todo nome deve identificar 1 cliente, no máximo.

$\forall x \exists y$   
nome(x) --> cliente(y), tem(y,x). [1]

$\forall x, y, z$   
nome(x), cliente(y), tem(y,x) --> não [ cliente(z),  
tem(z,x),  $z \neq y$  ]. [5]

Não pode haver mais de 1 cardápio.

$\forall y1, y2$   
cardápio(y1) --> não [ cardápio(y2),  $y1 \neq y2$  ]. [13]

Não pode haver mais de 1 caderno especial.

$\forall y1, y2$   
caderno(y1), especial(y1) --> não [ caderno(y2),  
especial(y2),  $y1 \neq y2$  ]. [13, A]

Não pode haver mais de 1 lucro.

$\forall y1, y2$   
lucro(y1) --> não [ lucro(y2),  $y1 \neq y2$  ]. [13]

Não pode haver mais de 1 caixa.

$\forall y1, y2$   
caixa(y1) --> não [ caixa(y2),  $y1 \neq y2$  ]. [13]

Não pode haver mais de 1 valor em dinheiro.

$\forall y1, y2$   
valor(y1), em\_dinheiro(y1) --> não [ valor(y2),  
em\_dinheiro(y2),  $y1 \neq y2$  ]. [13, A]

Não pode haver mais de 1 valor de início.

$\forall y1, y2$   
valor(y1), de\_início(y1) --> não [ valor(y2),  
de\_início(y2),  $y1 \neq y2$  ]. [13, A]

Cliente não pode pendurar conta se pendurou outra conta.

$\forall x, y, z$   
cliente(x), conta(z), pendura(x,z) --> não [ conta(y),  
pendura(x,y),  $y \neq z$  ]. [10a, E]

#### 6.4 Quarta Etapa: paráfrase da especificação

São as transformações da Linguagem Formal para a Linguagem Comum e devem ser feitas com base nas heurísticas apresentadas.

Parte de dados:

São apresentadas os conceitos representados na Linguagem Formal, a respectiva transformação para a Linguagem Comum e as heurísticas utilizadas (entre colchetes).

mesa(x), conta(y), tem(x,y)  
Mesas têm contas [1]

cardápio(x), prato(y), tem(x,y)  
Cardápios têm pratos [1]

cardápio(x), bebida(y), tem(x,y)  
Cardápios têm bebidas [1]

prato(x), preço(y), tem(x,y)  
Pratos têm preços [1]

bebida(x), preço(y), tem(x,y)  
Bebidas têm preços [1]

caixa(x), valor(y), em\_dinheiro(y), tem(x,y)  
Caixas têm valores em dinheiro [1]

caixa(x), valor(y), de\_inicio(y), tem(x,y)  
Caixas têm valores de início [1]

mesa(x), número(y), tem(x,y)  
Mesas têm números [1]

pedido(x), valor(y), tem(x,y)  
Pedidos têm valores [1]

conta(x), valor(y), tem(x,y), de\_conta(y)  
Contas têm valores de conta [1]

cliente(x), conta(y), pendura(x,y)  
Clientes penduram contas [1]

pedido(x), prato(y), tem(x,y)  
Pedidos têm pratos [1]

pedido(x), bebida(y), tem(x,y)  
Pedidos têm bebidas [1]

caixa(x), lucro(y), tem(x,y)

Caixas têm lucros [1]

caderno(x), especial(x), conta(y), pendurada(y), tem(x,y)  
Cadernos especiais têm contas penduradas [1]

pendura(x,y), data(y), acontece\_em(pendura(x,y),z)  
Penduras acontecem em datas [3]

conta(x), pedido(y), tem(x,y)  
Contas têm pedidos [1]

cliente(x), nome(y), tem(x,y)  
Clientes têm nomes [1]

Parte de operações:

São apresentadas as operações na Linguagem Formal, as respectivas traduções para a Linguagem Comum e as heurísticas utilizadas (entre colchetes).

Anotar\_pedidos(p,n):

‡ p, pr, b, m, c

**Pós:** pedido(p), prato(pr), bebida(b),  
tem(p,pr), tem(p,b),  
número(n), mesa(m), tem(m,n),  
conta(c), tem(m,c), tem(c,p).

Anotar pedido

Recebe pedido e número da mesa

Registra que mesa tem conta, que conta tem pedido,  
que pedido tem prato, que pedido tem bebida.

[8, 6, 9, 18]

Fechar\_conta(n):v1.

‡ v1, v2, c, m, v3, p, v4, pr, b

**Pré:** v1=soma(v2), valor(v1), total(v1),  
número(n), conta(c), mesa(m), tem(m,c), tem(m,n),  
tem(c,v1), valor(v2), pedido(p), tem(p,v2), tem(c,p),  
v2=v3+v4, valor(v3), prato(pr), tem(pr,v3),  
bebida(b), valor(v4), tem(b,v4).

**Pós:** fechada(m).

Fechar conta

Recebe número da mesa

Calcula valor total da conta da mesa, que é igual à soma  
do valor do pedido da conta da mesa, que é igual à soma  
do valor do prato e valor da bebida

Registra que mesa foi fechada

Coloca na lista exterior valor total da conta da mesa.

[8, 6, 9, 22, 13]

Cobrar\_conta(n,v):t.

‡ m, t, c, vt

**Pré:** mesa(m), número(n), tem(m,n), valor(v),  
pago(v), troco(t), t=v-vt, valor(vt), total(vt),  
conta(c), tem(c,vt), tem(m,c).

Cobrar conta

Recebe número da mesa e valor pago

Calcula troco, que é igual a valor pago menos valor total  
da conta

Coloca na lista exterior troco.

[8, 6, 9, 17, 22, 9]

Fechar\_caixa:l.

‡ l, vd, vi, c

**Pré:** lucro(l), l=vd-vi, valor(vd), em\_dinheiro(vd),  
caixa(c), tem(c,vd), valor(vi), de\_início(vi),  
tem(c,vi).

Fechar caixa

Calcula lucro, que é igual a valor em dinheiro do caixa  
menos valor de início do caixa

Coloca na lista exterior lucro.

[22, 17, 9, 21]

Pendurar\_conta(n,v,d):

∀ c, cc2, ‡ cc1

**Pré:** não [ nome(n), cliente(c), tem(c,n), conta(cc2),  
pendura(c,cc2) ].

**Pós:** pendura(c,cc1), valor(v), tem(cc1,v),  
em(pendura(c,cc1),d), data(d), pendurada(cc1).

Pendurar conta

Recebe nome do cliente, valor da conta, data da pendura

Verifica se cliente não pendura outra conta

Registra que pendura ocorre em data.

[8, 9, 7, 4, 3, 19, 2]

Pagar\_conta\_pendurada(n):

∀ c, cc

**Pós:** não [ pendura(c,cc), cliente(c), nome(n),  
tem(c,n), conta(cc), pendurada(cc) ].

Pagar conta pendurada

Recebe nome do cliente

Retira que cliente pendura conta pendurada.

[18, 2, 13, 8, 9]

Parte de restrições:



São apresentadas as restrições na Linguagem Formal, as respectivas transformações para a Linguagem Comum e as heurísticas utilizadas (entre colchetes).

$\forall x, y, z$   
mesa(x), conta(y), tem(x,y) --> não [ conta(z), tem(x,z),  
z≠y ].  
Toda mesa pode ter 1 conta, no máximo. [5]

$\forall x \dagger y$   
conta(x) --> mesa(y), tem(y,x).  
Toda conta deve pertencer a mesa. [1]

Obs: "pertencer a" é o inverso de "ter".

$\forall x, y, z$   
conta(x), mesa(y), pertence\_a(x,y) --> não [ mesa(z),  
pertence\_a(x,z), z≠y ].  
Toda conta pode pertencer a 1 mesa, no máximo. [5]

Obs: "pertencer a" é o inverso de "ter".

$\forall x \dagger y$   
prato(x) --> cardápio(y), tem(y,x).  
Todo prato deve estar no cardápio. [1]

$\forall x \dagger y$   
bebida(x) --> cardápio(y), tem(y,x).  
Toda bebida deve estar no cardápio. [1]

$\forall x \dagger y$   
prato(y) --> preço(y), tem(x,y).  
Todo prato deve ter preço. [1]

$\forall x, y, z$   
prato(x), preço(y), tem(x,y) --> não [ preço(z),  
tem(x,z), z≠y ].  
Todo prato pode ter 1 preço, no máximo. [5]

$\forall x \dagger y$   
preço(x), de\_prato(x) --> prato(y), tem(y,x).  
Todo preço de prato deve definir 1 prato, no mínimo. [6, A, I]

$\forall x \dagger y$   
bebida(y) --> preço(y), tem(x,y).  
Toda bebida deve ter preço. [1]

$\forall x, y, z$   
bebida(x), preço(y), tem(x,y) --> não [ preço(z),  
tem(x,z), z≠y ].  
Toda bebida pode ter 1 preço, no máximo. [5]



$\forall x \exists y$   
preço(x), de\_bebida(x) --> bebida(y), tem(y,x).  
Todo preço de bebida deve definir 1 bebida, no mínimo. [6, A, I]

$\forall x \exists y$   
valor(x), em\_dinheiro(x) --> caixa(y), tem(y,x).  
Valor em dinheiro deve pertencer ao caixa. [1, A, I]

$\forall x \exists y$   
valor(x), de\_inicio(x) --> caixa(y), tem(y,x).  
Valor de início deve pertencer ao caixa. [1, A, I]

$\forall x \exists y$   
mesa(x) --> número(y), tem(x,y).  
Toda mesa deve ter número. [1]

$\forall x, y, z$   
mesa(x), número(y), tem(x,y) --> não [ número(z),  
tem(x,z),  $z \neq y$  ].  
Toda mesa pode ter 1 número, no máximo. [5]

$\forall x \exists y$   
número(x) --> mesa(y), tem(y,x).  
Todo número deve identificar mesa. [1]

$\forall x, y, z$   
número(x), mesa(y), tem(y,x) --> não [ mesa(z),  
tem(z,x),  $z \neq y$  ].  
Todo número pode identificar 1 mesa, no máximo. [5]

$\forall x \exists y$   
pedido(x) --> valor(y), tem(x,y), de\_pedido(y).  
Todo pedido deve ter valor de pedido. [1]

$\forall x, y, z$   
pedido(x), valor(y), de\_pedido(y), tem(x,y) -->  
não [ valor(z), tem(x,z),  $z \neq y$  ].  
Todo pedido pode ter 1 valor de pedido, no máximo. [5]

$\forall x \exists y$   
valor(x), de\_pedido(x) --> pedido(y), tem(y,x).  
Todo valor de pedido deve definir 1 pedido, no mínimo. [6, I, A]

$\forall x \exists y$   
conta(x) --> valor(y), tem(x,y), de\_conta(y).  
Toda conta deve ter valor de conta. [1]

$\forall x, y, z$   
conta(x), valor(y), de\_conta(y), tem(x,y) -->  
não [ valor(z), tem(x,z),  $z \neq y$  ].  
Toda conta pode ter 1 valor de conta, no máximo. [5]

$\forall x \exists y$   
valor(x), de\_conta(x) --> conta(y), tem(y,x).  
Todo valor de conta deve definir 1 conta, no mínimo. [6, I, A]

$\forall x, y, z$   
cliente(x), conta(y), pendura(x,y) --> não [ conta(z),  
pendura(x,z),  $z \neq y$  ].  
**Todo cliente pode pendurar 1 conta, no máximo. [5]**

$\forall x, y, z$   
conta(x), cliente(y), pendura(y,x) --> não [ cliente(z),  
pendura(z,x),  $z \neq y$  ].  
**Toda conta pode ser pendurada por 1 cliente, no máximo. [5]**

$\forall x \dagger y$   
lucro(x) --> caixa(y), tem(y,x).  
**Lucro deve pertencer ao caixa. [1]**

$\forall x \dagger y$   
conta(x), pendurad(x) --> caderno(y), especial(y),  
tem(y,x).  
**Toda conta pendurada deve pertencer ao caderno especial. [1, A]**

$\forall x, w \dagger y$   
pendura(x,w), cliente(x), conta(w) --> data(y),  
em(pendura(x,w),y).  
**Toda pendura deve acontecer em data. [1, J]**

**Obs: "acontecer\_em" é sinônimo de "em".**

$\forall x, w, y, z$   
pendura(x,w), cliente(x), conta(w), data(y),  
em(pendura(x,w),y) --> não [ data(z), em(pendura(x,w),z),  
 $z \neq y$  ].  
**Toda pendura pode acontecer em 1 data, no máximo. [5, J]**

**Obs: "acontecer\_em" é sinônimo de "em".**

$\forall x \dagger y$   
pedido(x) --> conta(y), tem(y,x).  
**Todo pedido deve pertencer a conta. [1]**

$\forall x, y, z$   
pedido(x), conta(y), tem(y,x) --> não [ conta(z),  
tem(z,x),  $z \neq y$  ].  
**Todo pedido pode pertencer a 1 conta, no máximo. [5]**

$\forall x \dagger y$   
cliente(x) --> nome(y), tem(x,y).  
**Todo cliente deve ter nome. [1]**

$\forall x, y, z$   
cliente(x), nome(y), tem(x,y) --> não [ nome(z),  
tem(x,z),  $z \neq y$  ].  
**Todo cliente pode ter 1 nome, no máximo. [5]**

$\forall x \dagger y$   
nome(x) --> cliente(y), tem(y,x).

Todo nome deve identificar cliente. [1]

$\forall x, y, z$

nome(x), cliente(y), tem(y,x) --> não [ cliente(z),  
tem(z,x),  $z \neq y$  ].

Todo nome pode identificar 1 cliente, no máximo. [5]

$\forall y1, y2$

cardápio(y1) --> não [ cardápio(y2),  $y1 \neq y2$  ].

Não pode haver mais de 1 cardápio. [13]

$\forall y1, y2$

caderno(y1), especial(y1) --> não [ caderno(y2),  
especial(y2),  $y1 \neq y2$  ].

Não pode haver mais de 1 caderno especial. [13, A]

$\forall y1, y2$

lucro(y1) --> não [ lucro(y2),  $y1 \neq y2$  ].

Não pode haver mais de 1 lucro. [13]

$\forall y1, y2$

caixa(y1) --> não [ caixa(y2),  $y1 \neq y2$  ].

Não pode haver mais de 1 caixa. [13]

$\forall y1, y2$

valor(y1), em\_dinheiro(y1) --> não [ valor(y2),  
em\_dinheiro(y2),  $y1 \neq y2$  ].

Não pode haver mais de 1 valor em dinheiro. [13, A]

$\forall y1, y2$

valor(y1), de\_início(y1) --> não [ valor(y2),  
de\_início(y2),  $y1 \neq y2$  ].

Não pode haver mais de 1 valor de início. [13, A]

$\forall x, y, z$

cliente(x), conta(z), pendura(x,z) --> não [ conta(y),  
pendura(x,y),  $y \neq z$  ].

Cliente não pode pendurar conta se pendura outra conta. [10a, E]

## 7 PROPOSTA DE FERRAMENTAS

Algumas ferramentas automatizadas podem ser desenvolvidas para auxiliar o Analista de Sistemas nas atividades de Análise de Requisitos e Especificação, com base nas heurísticas aqui apresentadas.

Primeiramente, pode-se construir uma ferramenta para auxílio à coleta de informações. Esta ferramenta lembraria ao Analista as perguntas que devem ser feitas e documentaria as respostas fornecidas pelo Usuário.

Com um pouco mais de ambição e estudo, poderia ser incorporado conhecimento a esta ferramenta de tal modo que ela mesma fizesse as perguntas ao Usuário.

Neste caso, para as perguntas específicas e para as perguntas sobre palavras-chave (aquelas com lacunas a serem preenchidas), seria necessário um mecanismo para identificar funções e tipos de informações (representados por verbos e substantivos). De modo mais simples, o Usuário mesmo se encarregaria desta tarefa, fornecendo as respostas com as palavras-chave já marcadas no texto. De outra forma, faz-se necessário um sistema de inteligência que analise textos em linguagem natural e identifique seus elementos sintáticos.

De qualquer forma, uma vez identificadas as palavras-chave (dentro das respostas às perguntas genéricas), esta ferramenta faria as perguntas específicas para cada tipo.

Outra ferramenta poderia ser construída para auxiliar na transformação das informações em Linguagem Informal (respostas ao questionário fornecidas pelo Usuário) para a Linguagem Comum.

A primeira função dela seria documentar as informações nos formatos da Linguagem Comum, procurando lembrar ao Analista os formatos-padrão.

Outra função seria a de registrar os sinônimos e procurar padronizar o seu uso. Assim, o Usuário escolheria um dos sinônimos como o termo-padrão, a ferramenta procuraria, nos textos em Linguagem Comum, os sinônimos e os trocaria pelo termo-padrão.

Esta ferramenta também poderia procurar frases semelhantes ou que utilizassem alguns termos em comum, indicando assim ao Analista partes do texto que possivelmente falassem a mesma coisa mas de modos diferentes. Isto facilitaria a identificação de sinônimos que não fossem fornecidos pelo Usuário.

Já a tarefa de analisar o texto em Linguagem Informal e traduzir suas informações para os formatos da Linguagem Comum seria mais complicada e dependeria de análises sintáticas sobre o primeiro texto.

Porém esta tarefa poderia ser auxiliada por uma ferramenta que, ao final da definição de requisitos em Linguagem Comum, faria a análise dos termos (substantivos) usados nas operações e não gerados em nenhuma outra operação, sugerindo assim a definição de outras operações possivelmente relevantes, mas que, por alguma razão foram esquecidas.

Esta ferramenta também verificaria se cada substantivo utilizado nas partes de operações e restrições da Linguagem Comum foi definido na parte de dados e estrutura da mesma linguagem.

A seguir, seria útil desenvolver uma ferramenta para auxiliar o Analista nas transformações da Linguagem Comum para

a Linguagem Formal assumida. Ela se encarregaria de identificar os padrões definidos nas heurísticas (pelas palavras constantes, representadas em letras maiúsculas) e sugeriria as correspondentes ações (ou formatos para a Linguagem Formal), além de fazer a documentação da especificação.

De novo, seria necessária uma análise sintática antes de intervenção da ferramenta. Esta análise poderia ser feita pelo Analista, que marcaria os verbos, os substantivos, os adjetivos e as preposições (indicando o tipo das relações e os nomes associados). A possibilidade de uma ferramenta automatizada fazer tal análise não é descartada; porém, julga-se isto algo bastante complicado e que necessita maiores conhecimentos e estudos para ser sugerido.

Nestas transformações ainda, a quantificação das variáveis poderia ser feita automaticamente por esta ferramenta.

Por fim, há a idéia de uma ferramenta automatizada para auxiliar nas paráfrases (transformações da Linguagem Formal para a Linguagem Comum). Obviamente, só com base nas heurísticas apresentadas aqui uma ferramenta tal não poderia fazer as transformações sozinhas; ela seria um assistente do Analista. As funções dela seriam identificar padrões na Linguagem Formal, sugerir formatos para a tradução na Linguagem Comum, documentar as decisões tomadas pelo Analista (as paráfrases) e fazer a padronização no uso de sinônimos (conforme a definição no Dicionário de Termos).

Quanto à especificação em Linguagem Formal, é útil pesquisar ferramentas automatizadas para: verificação de suas propriedades; geração de código executável; execução da especificação (a qual

funcionaria como um protótipo). Mas isto já é outro assunto, o qual foge ao escopo deste trabalho.



## 8 CONCLUSÃO

Para definir a Linguagem Comum e as heurísticas associadas a ela, foram realizados experimentos com os seguintes casos: um consultório médico, uma biblioteca e o ambiente de uma secretaria de um curso de pós-graduação.

Em todos estes casos, partiu-se de um conjunto inicial de heurísticas para a coleta das informações, definido por experiências anteriores nesta tarefa e sugestões da literatura.

Após, procurou-se identificar, nos textos em Linguagem Natural (que continham as informações coletadas), alguns padrões na definição de dados, suas estruturas, operações e restrições.

Com isto, pode-se formular um sub-conjunto da Linguagem Natural como base para a Linguagem Comum que se queria. Este sub-conjunto foi também definido com vistas à Linguagem Formal assumida, ou seja, procurou-se abranger apenas os conceitos necessários na Linguagem Formal. A medida que os casos iam sendo estudados (um de cada vez), aquele sub-conjunto era testado e melhorado. Refez-se, então, os experimentos utilizando-se a Linguagem Comum assim definida, e foram identificados padrões nas transformações entre as linguagens (Informal, Comum e Formal), os quais vieram a compor o elenco de heurísticas, como já visto.

Por fim, foi feito, de maneira completa, o estudo de um caso (o qual foi apresentado neste trabalho): foram utilizadas as heurísticas para coleta das informações (em Linguagem Informal); estas informações foram transformadas para os formatos da Linguagem Comum (com a ajuda das correspondentes heurísticas); após, fez-se a tradução das informações em Linguagem Comum para a

Linguagem Formal assumida (também com a ajuda das heurísticas); por fim, utilizou-se das heurísticas para transformação da Linguagem Formal para a Linguagem Comum, para a obtenção da paráfrase da especificação do sistema.

Neste último estudo de caso, procurou-se avaliar a Linguagem Comum e as heurísticas. Notaram-se poucos defeitos, os quais, em tempo, foram corrigidos. Apesar dos poucos experimentos feitos (poucos casos estudados e poucas pessoas envolvidas) e da maneira acadêmica como foram realizados, algumas conclusões podem ser tiradas deste trabalho.

A Linguagem Comum proposta aqui mostrou-se útil para a identificação dos conceitos a serem modelados na Linguagem Formal, o que fica claro com as heurísticas para a transformação da Linguagem Comum para a Linguagem Formal (não são muitas e conseguem abranger grande parte das transformações, pelo menos nos casos estudados).

A Linguagem Comum também facilitou a validação da Especificação pelo Usuário (seja através das paráfrases ou mesmo por se tratar de um sub-conjunto da Linguagem Natural). Desta forma, o Usuário não precisa aprender formalismos para validar a Especificação.

Já as heurísticas de transformação entre as linguagens (Informal, Comum e Formal), apesar de serem apenas "conselhos" de como o Analista pode desempenhar as tarefas de transformações (não sugerindo, portanto, transformações automáticas), facilitaram este trabalho, nos estudos realizados, e mostraram-se um início para futuros estudos de automatização de tarefas (total ou de partes ou simplesmente ferramentas de auxílio).

Da mesma forma, as heurísticas para coleta das informações (e conseqüentemente introdução da informalidade no desenvolvimento de SI's) também se mostraram úteis, facilitando o trabalho do Analista.

É bom que fique claro, entretanto, que ainda há dependência destas atividades ao Analista, o qual deve fazer ainda boa parte do trabalho. As heurísticas identificadas e apresentadas aqui não são completas, não garantindo a solução (por natureza), nem garantindo sua validade para todos os casos.

A Linguagem Comum também permitiu, com base nos experimentos feitos, que Usuários e Analistas utilizassem uma mesma linguagem para definir e validar, em conjunto, requisitos de Sistemas de Informação. Também o Dicionário de Termos ajudou no entendimento entre eles, uma vez que termos duvidosos foram melhor definidos, que sinônimos foram identificados e padronizados e uma vez que se fez uso de mecanismos para facilitar a identificação de partes dos textos que falavam a mesma coisa ou sobre a mesma coisa.

Notou-se também que a Linguagem Comum apresenta graus de legibilidade e precisão intermediários às linguagens Informal e Formal (no caso, apenas para as linguagens assumidas).

Por ser um sub-conjunto da Linguagem Natural, ela se mostrou mais legível que a Linguagem Formal. E, através de suas limitações, ela conseguiu diminuir algumas imprecisões da Linguagem Natural, conforme constatado nos experimentos. Neste sentido, o uso de frases simples, a padronização de sinônimos e a definição explícita das ações nas operações foram de grande valia.

Sabe-se que as limitações da Linguagem Comum (em relação à Linguagem Natural) ocasionaram perda de informações. Por exemplo, nas parte de dados, ao serem eliminadas as conjunções (que relacionam orações), não se tornou mais possível identificar relações de tempo, causa, efeito, etc. Acredita-se que estas informações não são relevantes para a parte de dados e estrutura (a não ser que se trate de sistemas que incorporam tempo, o que não é objetivo aqui).

Já na parte de operações, a Linguagem Comum limitou a definição das operações. Notou-se, nos experimentos, que estas limitações facilitaram a transformação das operações, da Linguagem Comum para a Linguagem Formal assumida, e que foi possível especificar as operações sem grandes perdas (pelo menos, considerando-se a Linguagem Formal assumida, as operações na Linguagem Comum continham todas as informações relevantes).

No caso da parte de restrições de integridade, a Linguagem Comum se mostrou flexível (conforme a apresentação desta em capítulos anteriores), permitindo a especificação de quaisquer tipos de restrições. Os padrões (formatos) apresentados não são para restringir esta especificação, mas apenas para facilitar a transformação para a Linguagem Formal (assumida). Estes padrões foram identificados em experimentos e mostraram-se bastante abrangentes (muitos dos padrões permitiram a especificação de mais de uma restrição de integridade, nos experimentos realizados). Outros formatos podem ser utilizados, mas talvez não possam ser traduzidos para a Linguagem Formal (assumida) com a ajuda das heurísticas.

Verificou-se alguns pontos vulneráveis neste trabalho, ao se

realizarem os experimentos finais. Há ainda uma grande dependência à habilidade do Usuário em fornecer informações. Apesar de as heurísticas de coleta tentarem alcançar todas as informações relevantes, ainda assim há possibilidades de falta de informações porque o Usuário não as forneceu (ou por não ter sido induzido a fornecê-las ou por outros motivos mais complicados, como desinteresse, má-vontade, pressa, etc).

O problema é que estas omissões se refletirão na especificação do SI (tanto na Linguagem Comum, como na Linguagem Formal) e, posteriormente, na implementação do sistema.

Notou-se também que é crucial para o sucesso das idéias aqui expostas que as informações na Linguagem Informal estejam descritas de modo completo, ou seja, sem omissões de sujeitos, complementos, adjuntos, verbos, etc (deve estar explícito o maior número de informações possíveis). Por exemplo, na seguinte ação da operação de matrícula:

Verifica se aluno tem o número de créditos exigidos pela disciplina,

está implícito que o aluno deve ter, **no mínimo**, o número de créditos e não **exatamente** o número de créditos exigidos pela disciplina à qual ele quer se matricular.

Verificou-se também a impossibilidade de definir ações opcionais para as operações. Tal problema só pode ser resolvido através de especializações de operações. Por exemplo, a operação "devolução de empréstimo" deve exigir que o Usuário pague uma multa (ou debitá-la na conta do Usuário), se o empréstimo estiver em atraso. A solução é definir duas operações:

devolução de empréstimo sem multa (sem atraso) e



devolução de empréstimo com multa.

Outra fraqueza assumida é com relação a sub-tipos de conceitos que não são identificados na parte de dados da Linguagem Comum. Por exemplo, nas frases-padrão:

pacientes fazem consultas e

pacientes marcam consultas,

há o conceito "consultas feitas" (referente à primeira frase) e o conceito "consultas marcadas" (referente à segunda frase), ambos sub-tipos de consultas (uma "consulta marcada", provavelmente, se tornará, mais tarde, uma "consulta feita").

Ainda quanto às operações, é claro que existem muitos outros pontos questionáveis, como operações não-estruturadas, interativas, etc, que estão incluídos entre os problemas sendo pesquisados pela comunidade científica. Não é objetivo aqui estudar todos os problemas de especificações, por isto a razão de este trabalho não procurar soluções muito abrangentes.

O grande volume de informações é também um entrave ao uso de Linguagem Comum, como é entrave em qualquer situação. Espera-se que, com o advento de ferramentas automatizadas, Usuários e Analistas possam ser auxiliados na leitura, escrita e gerência de grandes volumes de informações.

Outra fraqueza da Linguagem Comum é que ela permite a definição de qualquer frase-padrão, operação ou restrição, desde que obedecidos seus formatos (menos para as restrições, que são livres). Deste modo, poderá haver partes da especificação que não tem nexos ou não fazem sentido (apesar de corretas sintaticamente). Somente a validação feita pelo Usuário pode detectar tal inconsistência.

Deve-se levar em conta também que este trabalho não se preocupou com Sistemas de Informação temporais (onde as informações estão sempre associadas a instantes de tempo em que são verdadeiras) ou interativos (com grande volume de interações máquina-Usuário), nem tratou de todos os problemas que afligem a fase de Análise de Requisitos, como a coleta de informações sobre uso dos dados ("usage perspective").

Como trabalhos futuros, seria interessante estudar um mecanismo que guarde correspondências entre as informações nas diferentes linguagens. Assim poderia haver um controle sobre a origem de uma informação ou seu destino depois das transformações para outras linguagens (que informações gerou ou como foi representada em outra linguagem). Este controle seria útil na manutenção da consistência das informações ao longo do desenvolvimento do SI (se alguma mudança ocorresse em uma parte do desenvolvimento, ela deveria propagar-se para as outras partes).

Um trabalho certo, a ser feito posteriormente, é o teste da Linguagem Comum com outros tipos de linguagens formais, a fim de verificar o grau de abrangência da Linguagem Comum, já que esta foi desenvolvida com vistas à Linguagem Formal assumida e somente foi testada com esta (as heurísticas foram propostas para o uso somente da linguagem assumida).

Também a identificação de novas heurísticas, nestes testes, permitiria compará-las com as já apresentadas aqui e, talvez, formar um conjunto de heurísticas genéricas.

O desenvolvimento de ferramentas automatizadas que auxiliem no uso da Linguagem Comum e das heurísticas é importante para a



aceitação da Linguagem Comum e vital para o seu sucesso.

Também deverão ser feitos estudos mais profundos quanto ao tratamento de linguagens naturais, uma vez que, neste trabalho, foram considerados apenas aspectos primitivos e simples de tal assunto.

Por fim, pode-se dizer que os objetivos deste trabalho foram alcançados.

Conseguiu-se desenvolver e testar (ainda que em poucos casos) uma Linguagem Comum que facilitasse a comunicação entre Usuários e Analistas para a definição de requisitos de Sistemas de Informação, que apresentasse um certo grau de legibilidade e um certo grau de precisão e que fosse próxima das linguagens informais e formais (pelo menos, as linguagens assumidas para teste). Para este último caso, foram identificadas e testadas heurísticas que facilitam o trabalho de transformação entre as linguagens feito pelo Analista de Sistemas.

## 9 BIBLIOGRAFIA

- [ABB83] ABBOTT, R. J. Program design by informal English descriptions. Communications of the ACM, New York, v.26, n.11, p.882-94, Nov. 1983.
- [ALF77] ALFORD, M. W. A requirements engineering methodology for real-time processing requirements. IEEE Transactions on Software Engineering, New York, v.SE-3, n.1, p.89-107, Jan. 1977 apud: [Dav82].
- [AMB77] AMBLER, A. L. et al. GYPSY: a language for specification and implementation of verifiable programs. SIGPLAN Notices, New York, v.12, n.3, p.1-10, Mar. 1977 apud: [Dav82].
- [AZE72] AZEVEDO, M. C. (coordenador) Pensamento - código - informação. Porto Alegre, UFRGS, 1972. Série Cadernos Universitários, n.3.
- [AZE73] AZEVEDO, M. C. (coordenador) Atenção - signos - graus de informação. Porto Alegre, UFRGS, 1973. Série Cadernos Universitários, n.4.
- [BAB85] BABB II, R. G. et al. Workshop on models and languages for software specification and design. Computer, Los Angeles, v.18, n.3, p.103-8, 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, p.94-103, Mar. 1978.
- [BAL81] BALZER, R. ; CHEATHAM Jr., T. E. Program transformations. IEEE Transactions on Software Engineering, New York, v.SE-7, n.1, p.1-2, Jan. 1981.
- [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, p.39-46, Nov. 1983.
- [BAL85] BALZER, R. A 15 year perspective on automatic programming. IEEE Transactions on Software Engineering, New York, v.SE-11, n.11, p.1257-67, Nov. 1985.
- [BAR85] BARSTOW, D. R. Domain-specific automatic programming. IEEE Transactions on Software Engineering, New York, v.SE-11, n.11, p.1321-36, Nov. 1985.

- [BAR86] BARRET, R. A.; DAVIS, B. C. Successful systems analysts hone their communication skills. Data Management, Illinois, v.24, n.4, p.18-21, Apr. 1986.
- [BLA87] BLACKMAN, J. Open communication improves business circulation. Data Management, Illinois, v.25, n.2, p.12-5, Feb. 1987.
- [BOR85] BORGIDA, A.; GREENSPAN, S.; MYLOPOULOS, J. Knowledge representation as the basis for requirements specifications. Computer, Los Angeles, v.18, n.4, p.82-91, Apr. 1985.
- [BOS89] BOSTROM, R. P. Successful application of communication techniques to improve the system development process. Information & Management, Amsterdam, v.16, n.5, p.279-96, May 1989.
- [BOU85] BOUZEGHOUB, M.; GARDARIN, G.; METAIS, E. Database design tools: an expert system approach. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASE, Stockolm, Aug. 21-23. Proceedings. s.l., s.n., 1985. p.82-95.
- [BUY74] BUYSENS, E. Semiologia e comunicação lingüística. Trad: I. Blikstein. 2.ed. São Paulo, Cultrix. 1974.
- [CAI75] CAINE, S. H. ; GORDON, E. K. PDL - a tool for software design. In: NATIONAL COMPUTER CONFERENCE, Anaheim, May 19-22, 1975. Proceedings. Montvale, AFIPS Press, 1975. AFIPS n.44. apud: [Dav82]. p.134-45.
- [CAS87] CASTILHO, J. M. V. Especificações formais e sistemas de bancos de dados. Buenos Aires, Kapelusz. 1987.
- [CER83] CERI, S. (ed.) Methodologies and tools for database design. Amsterdam, North-Holland. 1983.
- [CHE76] CHEN, P. P. S. The E-R model: toward a unified view of data. ACM Transactions on Database Design, New York, v.1, n.1, p.9-36, Mar. 1976.
- [CHR87] CHRISMAN, C.; BECCUE, B. Effective communication with users improves data base design. Data Management, Illinois, v.25, n.1, p.17-22, Jan. 1987.
- [DAH66] DAHL, O. Simula - an ALGOL-based simulation language. Communications of the ACM, New York, v.9, n.9, p.671-78, Sept. 1966. apud: [Dav82].
- [DAN74] DANIELS, A. ; YEATES, D. Formação básica em análise de sistemas. São Paulo, Livros Técnicos e Científicos, 1974.

- [DAV72] DAVIS, K. Human behavior at work: human relations and organizational behavior. 4.ed. New York, McGraw-Hill. 1972.
- [DAV80] DAVIS, A. M. Automating the requirements phase: benefits to later phases of the software life-cycle. In: INTERNATIONAL COMPUTER SOFTWARE & APPLICATIONS CONFERENCE, 4., Chicago, Oct. 27-31, 1980. Proceedings. New York, IEEE, 1980. COMPSAC 80. p.42-8.
- [DAV82] DAVIS, A. M. The design of a family of application-oriented requirements languages. Computer, Los Angeles, v.15, n.5, p.21-8, May 1982.
- [DAV82b] DAVIS, G. B. Strategies for information requirements determination. IBM Systems Journal, Armonk, v.21, n.1, p.4-30, Jan. 1982.
- [DeB77] De BRABANDER, B. ; EDSTRÖM, A. Successful information system development projects. Management Science, Providence, v.24, n.2, p.32-43, Oct. 1977.
- [DeB84] De BRABANDER, B. ; T G. Successful information system development in relation to situational factors which affect effective communication between MIS-users and EDP-specialists. Management Science, Providence, v.30, n.2, p.12-24, Feb. 1984.
- [EPS85] EPSTEIN, S. S. Transportable natural language processing through simplicity - the PRE system. ACM Transactions on Office Information Systems, New York, v.3, n.2, p.107-20, Apr. 1985.
- [EPS86] EPSTEIN, I. O signo. 2.ed. São Paulo, Atica, 1986.
- [EVE80] EVERHART, C. R. A unified approach to software system engineering. In: INTERNATIONAL COMPUTER SOFTWARE & APPLICATIONS CONFERENCE, 4., Chicago, Oct. 27-31, 1980. Proceedings. New York, IEEE, 1980. COMPSAC 80. p.49-55.
- [FAI86] FAIRLEY, R. Software engineering concepts. 2.ed. New York, McGraw-Hill. 1986.
- [GAN79] GANE, C. ; SARSON, T. Structured systems analysis: tools and techniques. Englewood Cliffs, Prentice-Hall, 1979.
- [GOM81] GOMAA, H.; SCOTT, D. B. H. Prototyping as a tool in the specification of user requirements. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 5., San Diego, Mar. 9-12, 1981. Proceedings. New York, IEEE/ACM, 1981. p.333-42.

- [GUE83] GUERRIERI Jr., J. A. Establishing user requirements. In: AUERBACH systems development management. Philadelphia, 1983.
- [GUT77] GUTTAG, J. G. Abstract data types and the development of data structures. Communications of the ACM, New York, v.20, n.6, p.396-404, June 1977.
- [HAA82] HAASE, V. H. ; KOCH, G. R. Application-oriented specifications: developing the connection between user and code. Computer, Los Angeles, v.15, n.5, p.10-1, May 1982.
- [HAE] HAESBAERT, D. et al. Linguagem e Comunicação. In AZEVEDO, M. C. (coordenador) Pensamento - código - informação. Série Cadernos Universitários, n.3. UFRGS. 1972.
- [HOA69] HOARE, C. A. R. An axiomatic basis for computer programming. Communications of the ACM, New York, v.12, n.10, p.576-80, Oct. 1969.
- [JAC83] JACKSON, M. System development. Englewood Cliffs, Prentice-Hall, 1983 apud: [Lei87].
- [JAK] JAKOBSON, R. Linguística e Comunicação. São Paulo, Cultrix. s.d.
- [KEN78] KENT, W. Data and reality. Amsterdam, North-Holland, 1978.
- [KEN84] KENDALL, K. E.; KENDALL, J. E. STROBE: a structural approach to observation of the decision-making environment. Information & Management, Amsterdam, v.7, n.1, p.1-12, Feb. 1984.
- [KOM81] KOMODA, N.; HARUNA, K.; KAJI, H.; SHINOZAWA, H. An innovative approach to system requirements analysis by using structural modeling method. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 5., San Diego, Mar. 9-12, 1981. Proceedings. New York, IEEE/ACM, 1981. p.305-13.
- [KOS87] KOST, G. A importância da comunicação entre o departamento de sistemas e a alta administração. MIS, São Paulo, n.4, v.1, p.13-26, Maio 1987.
- [LAN82] LAND, F. Adapting to changing user requirements. Information & Management, Amsterdam, v.5, n.2, p.59-76, June 1982.
- [LEI87] LEITE, J. C. S. P. A survey on requirements analysis. University of California at Irvine, June 1987. ASE Project RTP 070.



- [LEN82] LENAT, D. B. The nature of Heuristics. Artificial Intelligence, Amsterdam, v.19, n.2, p.189-249, Oct. 1982.
- [LEV82] LEVENE, A. A.; MULLERY, G. P. An investigation of requirement specification languages: theory and practice. Computer, Los Angeles, v.15, n.5, p.50-9, May 1982.
- [LIS77] LISKOV, B. et al. Abstractions mechanisms in CLU. Communications of the ACM, New York, v.20, n.8, p.564-76, Aug. 1977.
- [LOH88a] LOH, S. & CASTILHO, J. M. V. Um Estudo sobre projeto conceitual de bancos de dados: banco de dados para a biblioteca CPD-CPGCC/UFRGS. Porto Alegre, CPGCC/UFRGS. 1988. RP n.89.
- [LOH88b] LOH, S. Método de coleta e documentação de dados para a modelagem de sistemas com estudo de casos. Porto Alegre, Curso de Bacharelado em Ciências de Computação/UFRGS. 1988. Trabalho de Conclusão.
- [LOH89a] LOH, S. Estudo de caso para a metodologia do projeto "Ferramentas para especificações formais de sistemas de banco de dados". Porto Alegre, CPGCC/UFRGS, 1989. Trabalho Individual.
- [LOH89b] LOH, S.; CASTILHO, J. M. V. Um Método para coleta e documentação de dados para a modelagem de sistemas. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 4., Campinas 5-7 Abr. 1989. Anais. São Paulo, SBC, 1989. p.90-9.
- [LOH89c] LOH, S. Análise dos aspectos que influenciam a comunicação entre usuários e analistas de sistemas. Porto Alegre. CPGCC/UFRGS, 1989. TI-130.
- [LUD82] LUDEWIG, J. Computer-aided specification of process control systems. Computer, Los Angeles, v.15, n.5, p.12-20, May 1982.
- [MAA89] MAAREK, Y. S.; BERRY, D. M. The use of lexical affinities in requirements extraction. In: INTERNATIONAL SOFTWARE ENGINEERING SPECIFICATION AND DESIGN, 5., Pittsburgh May 19-20. 1989. Proceedings. Publicado em Software Engineering Notes, New York, v.14, n.3, May 1989. p.196-202.
- [MAN87] MANTHA, R. W. Data flow and data structure modeling for database requirements determination: a comparative study. Management Information Systems Quarterly, Minneapolis, v.11, n.4, p.78-85, Dec. 1987.



- [MAT87] MATSUMURA, K.; MIZUTANI, H.; ARAI, M. An Application of structural modeling to software requirements analysis and design. IEEE Transactions on Software Engineering, New York, v.SE-13, n.4, p.461-71, Apr. 1987.
- [MEY85] MEYER, B. On formalism in specifications. IEEE Software, Los Alamitos, v.2, n.1, p.6-27, Jan. 1985.
- [MIL76] MILLER, G. A. (organizador) Linguagem, psicologia e comunicação. São Paulo, Cultrix. 1976.
- [MIR89] MIRIYALA, K; HARANDI, M. T. Analogical approach to specification derivation. In: INTERNATIONAL WORKSHOP ON SOFTWARE SPECIFICATION AND DESIGN. 5., New York, v.14, n.3, May 1989. Publicado em Software Engineering Notes, New York, v.14, n.3, p.203-10, May 1989.
- [NEW76] NEWELL, A. & SIMON, H. A. Computer Science as empirical inquiry: symbols and search. Communications of the ACM, New York, v.19, n.3, p.113-26, Mar. 1976.
- [NIR81] NIRENBERG, J. S. A Psicologia da comunicação. São Paulo, IBRASA, 1981.
- [NIS89] NISKIER, C.; MAIBAUM, T.; SCHWABE, D. A look through PRISMA: towards pluralistic knowledge-based environments for software specification acquisition. 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, p.197-203, May 1989.
- [NOS88] NOSEK, J. T.; SCHWARTZ, R. B. User validation of information system requirements: some empirical results. IEEE Transactions on Software Engineering, New York, v.14, n.9, p.1372-5, Sept. 1988.
- [PAR87] PARMAR, L. Success factors in managing systems projects. Data Management, Park Ridge, v.25, n.3, p.27-30, Mar. 1987.
- [PEN76] PENTEADO, J. R. W. A Técnica da comunicação humana. 5.ed. São Paulo, Biblioteca Pioneira de Administração e Negócios, 1976.
- [PER76] PEREIRA, L. M. A Compreensão da linguagem natural em IA. In: ENCONTRO DE ENSINO E INVESTIGAÇÃO DO PORTUGUÊS, Lisboa, Nov. 1976.
- [PNN76] PENNA, A. G. Comunicação e Linguagem. São Paulo. Eldorado Tijuca,. 1976.

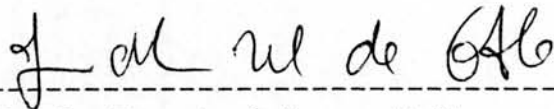
- [PUC69] PUCHKIN, V. N. Heurística: a ciência do pensamento criador. São Paulo. Zahar, 1969.
- [ROB] ROBALINO, H. et al. Código e linguagem. In AZEVEDO, M. C. (coordenador) Pensamento - código - informação. Série Cadernos Universitários, n.3. UFRGS. 1972.
- [ROB77] ROBINSON, L.; ROUBINE, O. Special - a specification and assertion language. SRI TR CSL-46, Jan. 1977 apud: [Dav82].
- [SAN] SANTOS, E. T. Semiologia - semiótica - estruturalismo In: AZEVEDO, M. C. (coordenador) Atenção - signos - graus de informação. Série Cadernos Universitários, n.4. UFRGS. 1973.
- [SAN80] SANTOS, C. S. Caracterização sistemática de restrições de integridade em bancos de dados. Rio de Janeiro, PUC/RJ, 1980. Tese de Doutorado.
- [SAV83] SAVIOLI, F. P. Gramática em 44 lições: com mais de 1700 exercícios. 6.ed. São Paulo, Atica, 1983.
- [SCH81] SCHARER, L. Pinpointing requirements. Datamation, New York, v.24, n.4, p.139-54, Apr. 1981.
- [SEN83] SENN, J. A. A comparison of structured methodologies. In: AUERBACH systems development management, Phyladelphia, 1983.
- [SET86] SETZER, V. W. Projeto lógico e projeto físico de bancos de dados. Belo Horizonte, UFMG, 1986. V Escola de Computação.
- [SHA77] SHAW, M. et al. Abstraction and verification in ALPHARD: defining and specification of iteration and generators. Communications of the ACM, New York, v.20, n.8, p.553-63, Aug. 1977.
- [SIM83] SIMON, H. A. Search and reasoning in problem solving. Artificial Intelligence, Amsterdam, v.21, n.1,2, p.87-98, Mar. 1983.
- [STE72] STEWART, D. K. A Psicologia da comunicação. São Paulo, Forense, 1972.
- [TAY80] TAYLOR, B. J. A method for expressing the functional requirements of real-time systems. In: IFAC/IFIP WORKSHOP ON REAL-TIME PROGRAMMING, Apr. 1980 apud: [Dav82].

- [TEI77] TEICHROEW, D.; HERSHEY III, E. A. PSL/PSA: a computer-aided technique for structured documentation and analysis of information processing systems. IEEE Transactions on Software Engineering, New York, v.SE-3, n.1, p.23-36, Jan. 1977.
- [TEO82] TEOREY, T. J. & FRY, J. P. Design of database structures. Englewood Cliffs, Prentice-Hall, 1982.
- [TEO89] TEOREY, T. J. et al. E-R model clustering as an aid for user communication and documentation in database design. Communications of the ACM, New York, v.32, n.8, p.975-87, Aug. 1989.
- [TOE83] TOELLNER, J. D. The Spectrum project management / systems development methodology package. In: AUERBACH system development management. Philadelphia, 1983.
- [VIT83] VITALARI, N. P.; DICKSON, G. W. Problem solving for effective systems analysis: an experimental exploration. Communications of the ACM, New York, v.26, n.11, p.948-56, Nov. 1983.
- [WAL76] WALTON, T. F. Communications and data management. New York. John Wiley & Sons, 1976.
- [WEG72] WEGNER, P. The Vienna Definition Language. ACM Computing Surveys, New York, v.4, n.1, p.5-63, Mar. 1972.
- [WEI83] WEISS, M. The human side of systems: an experiential approach. Information & Management, Amsterdam, v.6, n.2, p.103-8, Apr. 1983.
- [YEH84] YEH, R. T. et al. Software requirements: new directions and perspectives. In: HANDBOOK of software engineering. New York, Van Nostrand Reinhold. 1984.

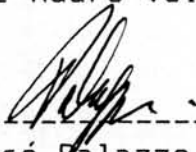
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL  
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Uma linguagem comum entre usuários e analistas para a definição  
de requisitos de sistemas de informação.

Dissertação apresentada aos Srs.



Prof. Dr. José Mauro Volkmer de Castilho



Prof. Dr. José Palazzo M. de Oliveira



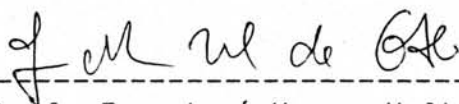
Prof. Dr. Júlio Cesar Sampaio do P. Leite



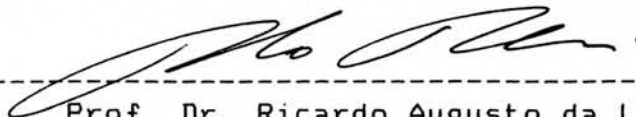
Prof. Dr. José Valdeni de Lima

Visto e permitida a impressão

Porto Alegre, 08. / 05. / 91.



Prof. Dr. José Mauro Volkmer de Castilho  
Orientador



Prof. Dr. Ricardo Augusto da L. Reis  
Coordenador do Curso de Pós-Graduação  
em Ciência da Computação