SOBRE A UTILIZAÇÃO DE CONCEITOS SUBJACENTES ÀS LINGUAGENS FORMAIS ORIENTADAS A OBJETOS NA MODELAGEM DE ASPECTOS DO APARATO COGNITIVO.

.

Diego Munk London

COPPE/UFRJ

Programa: História das Ciências e das Técnicas e epistemologia

Mestrado em Ciências (M. Sc)

Professor Luís Alfredo Vidal de Carvalho, D.Sc.
Professor Sérgio Exel Gonçalves, D.Sc.
Orientadores

RIO DE JANEIRO, RJ – BRASIL
JULHO DE 2006

SOBRE A UTILIZAÇÃO DE CONCEITOS SUBJACENTES ÀS LINGUAGENS FORMAIS ORIENTADAS A OBJETOS NA MODELAGEM DE ASPECTOS DO APARATO COGNITIVO.

Diego Munk London

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM HISTÓRIA DAS CIÊNCIAS E DAS TECNICAS E EPISTEMOLOGIA.

Aprovada por:

_____
Prof. Luis Alfredo Vidal de Carvalho, D.Sc.

_____
Prof. Sérgio Exel Gonçalves, D.Sc.

_____
Prof. Ricardo Silva Kubrusly, D.Sc.

_____
Prof. Geraldo Xexéo, D.Sc.

_____
Profa. Roseli Suzi Wedemann. D.Sc.

RIO DE JANEIRO, RJ – BRASIL
JULHO DE 2006

LONDON, DIEGO MUNK

Sobre a utilização de conceitos subjacentes às linguagens formais orientadas a objetos na modelagem de aspectos do aparato cognitivo. [Rio de Janeiro] 2006

VII, 146 p. 29,7 cm (COPPE/UFRJ, M.Sc., História das Ciências e das Técnicas e Epistemologia, 2006)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1. Limitações de Sistemas Formais

2.Inteligência Artificial

3.Orientação a objetos

4.Pensamento e Linguagem

I. COPPE/UFRJ II.Título(Série)

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

SOBRE A UTILIZAÇÃO DE CONCEITOS SUBJACENTES ÀS LINGUAGENS FORMAIS ORIENTADAS A OBJETOS NA MODELAGEM DE ASPECTOS DO APARATO COGNITIVO

Diego Munk London

Julho/2006

Orientadores: Luis Alfredo Vidal de Carvalho
Sérgio Exel Gonçalves

Programa: História das Ciências e das Técnicas e Epistemologia

Este trabalho possui como principal objetivo investigar a utilização dos conceitos subjacentes à programação orientada a objetos como possíveis instrumentos para o desenvolvimento de modelos que reproduzam alguns aspectos específicos das habilidades cognitivas humanas.

O ponto de partida é a exploração dos relacionamentos entre linguagens naturais e processos cognitivos genéricos e a correspondente relação entre linguagens formais especializadas e processos cognitivos específicos.

Em seguida serão apresentadas algumas considerações sobre as linguagens formais utilizadas para a programação de computadores e os principais conceitos do paradigma de programação orientada a objetos, além de possíveis relacionamentos entre estes conceitos e algumas idéias presentes no Tractatus Lógico-Philosophicus de Ludwig Wittgenstein.

Serão também analisadas certas características especificas a respeito dos conceitos presentes na programação orientada a objetos com o intuito de revelar algumas de suas vantagens e fraquezas e definir sua elegibilidade enquanto conjunto de critérios para a construção de modelos que representem aspectos específicos de nosso aparato cognitivo.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

ON THE SUBJACENT CONCEPTS OF THE OBJECT ORIENTED MODELING AND PROGRAMMING AS POSSIBLE INSTRUMENTS FOR THE DEVELOPMENT OF MODELS OF ASPECTS OF HUMAN COGNITIVE SKILLS

Diego Munk London

July/2006

Advisors: Luis Alfredo Vidal de Carvalho
Sérgio Exel Gonçalves

Department: History of Science

The main objective of this paper is to investigate the utilization of the subjacent concepts of the Object Oriented modeling and programming as possible instruments for the development of models of aspects of human cognitive skills.

The starting point is the exploration of the relationship between natural languages and generic cognitive process and the correspondent relationship between specialized formal languages and specific cognitive process.

After that , will be presented some considerations about the formal languages used for computer programming and the principal concepts of the object oriented programming paradigm and some relations between the object oriented concepts and some ideas presented by LUDWIG WITTGENSTEIN in his Tractatus Logico-Philosophicus.

The paper will then analyze some relevant facts and characteristics of the object oriented paradigm with the objective of reveal some of its strong contributions and some of its weakness and define its eligibility as a set of criteria to the construction of models which represent specific aspects of our cognitive structure.

Sumário

Introdução

Desde a antiguidade clássica, o desejo de conhecer a mente é uma das grandes questões concernentes a condição humana, questão que divide e entremeia os esforços individuais e conjuntos de diversas disciplinas que dedicaram seus estudos ao intuito de responder as questões fundamentais a que somos submetidos devido ao conjunto de nossas experiências.

A busca pela natureza da mente faz parte das tentativas de resposta a questão referente à nossa própria natureza. As teorias e modelos a respeito da consciência e dos processos cognitivos fazem parte das tentativas da humanidade de obedecer à premissa máxima do oráculo de Delfos, “Conhece-te a ti mesmo”. A partir da revolução racional-humanista-helênica operada através da unificação das cidades estado gregas, as teorias sobre a mente dividem-se em dois grandes grupos, subdivididos em uma quantidade interminável de subcategorias. Estes grupos são genericamente representados de um lado pelos defensores da idéia de que os processos mentais possam ser reduzidos a processos físicos de alguma espécie e os que acreditam que os processos mentais, ou parte deles sejam irredutíveis, mantendo-se além de nossas capacidades investigativas cientificas e racionais.

Este trabalho situa-se em uma das subdivisões do segundo grupo, ao afirmar, baseado nas obras existentes e em minha própria intuição, a hipótese de que nem todos os aspectos da mente possam ser reduzidos a processos físicos.

A partir deste ponto surgem as seguintes questões:

1 - Quais processos mentais podem ser compreendidos através de sua redução a processos físicos.

2 – Como esta compreensão pode ser alcançada ou pelo menos perseguida.

3 – O que entendemos como compreensão de um processo e qual é o peso epistemológico deste termo no contexto do presente trabalho.

Este trabalho oferece algumas possibilidades de respostas para estas questões:

1 – Os processos mentais passiveis de compreensão através de processos reducionistas são os processos referentes às atividades puramente cognitivas.

2 – Esta compreensão pode ser perseguida e alcançada (dentro dos limites epistemológicos do conhecimento descritos abaixo) através da analise estrutural da linguagem, uma vez assumida a linguagem como representante de processos cognitivos específicos e aceita a premissa de que este representante deve possuir algum nível de identidade estrutural com o representado.

3 – O significado do termo conhecer no contexto do presente trabalho é a noção de que o conhecimento é a capacidade de gerar modelos implementáveis dos processos cognitivos estudados.

Ao longo da formulação destas questões e possíveis respostas, serão apresentados alguns pontos de vista a respeito da validade epistemológica das teorias cognitivas existentes e do relacionamento entre pensamento e linguagem.

Após esta etapa, o foco do trabalho será radicalmente estreitado, focalizando um subconjunto da linguagem e buscando semelhanças e diferenças entre este e alguns processos cognitivos humanos.

O subconjunto das linguagens naturais ao qual me refiro é, de forma geral, o conjunto das linguagens de programação de computadores e de forma mais especifica, as linguagens pertencentes ao paradigma de programação orientada a objetos, tema bastante em voga na engenharia de software desde os anos 70 e que possui hoje uma boa base teórica estabelecida além de interessantes variantes conceituais como a programação orientada a aspectos e a programação orientada a protótipos.

Um dos objetivos do trabalho é apresentar a idéia de que os conceitos subjacentes à programação orientada a objetos possuem correlatos em outras tentativas de formalizar a linguagem natural e a visão humana do mundo.

Dentro deste espírito será traçada uma comparação entre alguns conceitos do Tractatus Lógico-Philosophicus de Ludwig Wittgenstein (obra de 1921 onde Wittgenstein apresenta sua teoria de um mundo e uma linguagem fundados em

combinações de objetos) e alguns conceitos pertencentes ao atual paradigma de modelagem e programação orientados a objetos.

Esta comparação conta também com a hipótese de que conceitos que se reapresentam diversas vezes, ao longo do tempo, do espaço e das disciplinas possuam um valor epistemológico aumentado em relação a outros conceitos.

Além disto, como não possuímos instrumentos definitivos de reconhecimento da verdade, podemos utilizar a recorrência e migração dos conceitos como um instrumento para determinar possíveis verdades vigentes, ou pelo menos como uma forma de identificar conceitos e conjuntos de conceitos que nos levem a formulação de modelos interessantes.

Após esta etapa, será apresentado um capitulo de discussão no qual discutiremos diversos aspectos a favor e contra o paradigma orientado a objetos assim como algumas soluções presentes neste paradigma para problemas antigos e algumas soluções presentes em variações da orientação a objetos com o intuito de sanar seus próprios problemas, novos e antigos.

Ao final do capitulo de discussão será apresentado um rascunho de uma proposta de utilização dos conceitos da orientação a objetos no intuito de modelar um aspecto especifico do conjunto das capacidades cognitivas humanas, a capacidade de discernir semanticamente entre proposições possíveis e impossíveis.

Esta proposta contará com os conceitos da orientação a objetos para ampliar drasticamente o conteúdo dos elementos proposicionais permitindo a geração de modelos avançados de compreensão e análise texto.

I – Definição do campo investigativo

1.1 – Os limites do conhecimento. Uma tentativa de definição do espaço investigativo em função do problema epistemológico.

O objetivo deste capítulo é definir, com a maior clareza possível, o espaço investigativo das proposições teóricas apresentadas nos capítulos seguintes.

A definição deste espaço investigativo é uma expressão da crença de que determinadas questões não se encontrem sujeitas a determinados tipos de soluções e que determinadas soluções não parecem estar sujeitas a determinados critérios de aceitação e validação.

Nem sempre é possível encontrar respostas rigorosas para problemas complexos. Para alguns, esta ausência pode ser anulada com a aplicação indeterminada de tempo, esforço e engenho, para outros, entretanto, parece haver algum tipo de impossibilidade, algum tipo de bloqueio cognitivo quando tentamos descrever através da linguagem algo que escapa a estrutura e as capacidades da linguagem, quando o referente encontra-se acima das capacidades representacionais do sistema simbólico que pretende representa-lo.

> "De forma introdutória, eu gostaria de chamar a atenção para o conhecido fato de que é impossível provar um teorema matemático completamente, por que quando alguém pensa que outra pessoa conseguiu, ainda é necessário provar que a prova inicial é livre de falhas, e assim por diante, ad infinitum, o que parece ser uma tarefa acima da falibilidade humana.
>
> Um individuo não pode jamais garantir que uma prova é correta, o melhor que se pode afirmar é (não foram encontradas falhas)."[1]
>
> (Dijkstra, 1961)

[1] "By way of introduction, i should like to draw attention to the not unknown fact that it is impossible to prove a mathematical theorem completely, because when one thinks that one has done so, one still has the duty to prove that the first proof was flawless, and so on, ad infinitum. So much for human fallibility. One can never guarantee that a proof is correct, the best one can say is (I have not discovered any mistakes)"
(Minha Tradução)

Existem numerosos trabalhos na literatura disponível acerca da capacidade humana de conhecer a realidade através da linguagem, que aparentemente constitui sua principal ferramenta de captura da realidade e construção de conhecimento.

Alguns destes trabalhos serão citados neste capítulo com o intuito de ilustrar os conceitos listados abaixo, que por enquanto podem ser compreendidos com as seguintes definições[2].

> a) Conhecer: Capturar um determinado fenômeno através de uma teoria.
>
> b) Fenômeno: Qualquer ocorrência ou conjunto de ocorrências passível de experimentação sensível ou inteligível por um agente consciente.
>
> c) Teoria: Construção lingüística que pretende expor o comportamento intimo (estrutura causal) de um determinado fenômeno.

O objetivo de uma teoria é descrever de forma mais completa possível o conjunto de relações causais de um fenômeno ou conjunto de fenômenos.

Infelizmente, a própria natureza da substancia que expressa as teorias (a linguagem), parece ser adequada para a descrição de processos finitos, ou pelo menos, apenas potencialmente infinitos. O fluxo de relações causais descritíveis pela linguagem pode ser infinito no tempo, ou seja, sob circunstancias favoráveis, o encadeamento destes fatos pode prolongar-se indefinidamente. Mas os fatos não poderão ocorrer todos ao mesmo tempo, assim como uma única causa não poderá ser causa de todas as conseqüências incluindo si própria.

A realidade, em contrapartida, parece ser infinita tanto na sucessão de instantes (infinito potencial) quanto na largura de cada instante (infinito atual), o que resulta em uma incapacidade da linguagem para descrever simultaneamente todas as relações causais contidas em um instante real.

[2] Estas são definições arbitrarias que solicito ao leitor que faça a gentileza de aceitar dentro do escopo do trabalho com a finalidade expressa de esclarecer os pontos vindouros.

Esta incapacidade dos sistemas simbólicos de capturar a totalidade das relações causais da realidade encontra-se bem explorada ao longo da história do pensamento, principalmente nas disciplinas da filosofia, da física, da química, da matemática e da inteligência artificial (como veremos com os exemplos a seguir), no entanto, possuímos outras formas "rigorosas" de satisfazer nossa necessidade de anulação do mal estar da ignorância em relação ao mundo que nos cerca.

Antes de prosseguir, considero válida uma rápida olhada na distinção entre sistema lógico e linguagem formal proposta por ALONZO CHURCH[3].

> "Nós distinguimos entre um sistema lógico e uma linguagem formal apontando o fato de que o primeiro é a formulação de um calculo abstrato para o qual nenhuma interpretação é determinada, possuindo sintaxe mas não semântica; mas o segundo é um sistema lógico acrescido de uma conexão com seus referente, aquilo que ele expressa."
> (Church 1951)

A distinção proposta por CHURCH é particularmente interessante se a deslocarmos um pouco em direção à filosofia da linguagem. Segundo CHURCH, os sistemas lógicos são compostos de uma lista de símbolos primitivos do vocabulário (o léxico, geralmente classificados em categorias que serão utilizadas no desenvolvimento das regras de formação e das regras de derivação), das regras de formação (regras que permitem ou proíbem a formação de uma proposição determinada.), e das regras de derivação (através das quais podemos gerar novas proposições a partir de proposições pré-existentes).

CHURCH afirma que a diferença entre um sistema lógico e uma linguagem formal é a ausência, no primeiro, do referente extralingüístico das

[3] "Matemático norte americano que em 1936 publicou a primeira definição precisa de uma função calculável, contribuindo enormemente para o desenvolvimento sistemático da teoria dos algoritmos. Church foi um aluno de Princeton e manteve-se lá por 40 anos, tornado-se professor de matemática e filosofia. Em 1967 mudou-se para a Universidade da Califórnia em Los Angeles. A solução dos problemas algorítmicos envolve a construção de um algoritmo capaz de responder pelo comportamento de outros algoritmos, e se este meta algoritmo não pode ser construído, isto significa que o problema é insolúvel. Teoremas estabelecendo a insolubilidade destes problemas encontram-se entre os mais importantes dentro da teoria dos algoritmos, e o teorema de Church é o primeiro de seu gênero. Partindo da tese do matemático inglês Alan Turing, Church provou que não existem algoritmos para uma classe de questões aritméticas essenciais." (Minha Tradução)

proposições, um sistema lógico formal possuiria apenas significantes sem significado (símbolos sem referentes), enquanto as linguagens formais poderiam se beneficiar dos elementos do sistema lógico e possuir um referente extralingüístico.

No entanto, me parece que, apesar de possuir um referente de uma natureza muito diferente, o sistema lógico também possui um referente, o referente da linguagem formal são estados de coisas possíveis, e o referente dos sistemas lógicos são as regras de possibilidades combinatórias dos elementos lexicais.

Desta forma, um sistema lógico poderia ser também considerado uma linguagem formal, que tentaria referir-se a regras abstratas, que nem sempre podem ser expostas com ou sem clareza. Pensando desta forma, podemos concluir que a sintaxe, enquanto funciona como representante de uma ordem, também possui semântica.

O desejo e a crença na possibilidade de exibir ao menos parte desta "regra de encaixe do mundo" (que inclui os primitivos e as regras) é uma possível resposta para a pergunta "Qual é a motivação da lógica?".

Uma conexão interessante, que pode auxiliar a compreensão do que seja este referente dos sistemas lógicos pode ser encontrado na obra de GÉRARD LEGRAND.

> "O que é então o *Logos*? Enumerou-se uma quinzena de traduções: Norma do mundo, Relação (ou proporção Pitagórica), Explicação discursiva, Razão, Lei do Devir, Definição, Fórmula, Narração, Lição, Coleção, Dizer ou a versão cristã, que assimilou o *Lógos* grego ao do Evangelho segundo São João, via neoplatonismo! Clémence Ramnoux, que nos fornece o essencial dessa enumeração menciona, com razão, o plural *Logoi* no sentido de "coleção de frases e fórmulas"."
> (Legrand, 1987)

Para o bem ou para o mal, este referente da sintaxe parece não ser suscetível a uma descrição completa e um grande numero de indivíduos busca uma solução menos rigorosa porém mais flexível e abrangente para lidar com os aspectos da realidade menos suscetíveis a uma descrição formal.

Existem sem dúvida, formas alternativas de lidar com a realidade, a mais poderosa delas parece ser a aceitação do dogma religioso, onde a incapacidade do discurso lingüístico (ainda presente) se completa através da aceitação de uma verdade não formulável, mas aceita, de alguma forma, pessoal e não comunicável.

Não possuo qualquer intenção de hierarquizar estas formas de conhecimento, mas declaro minha intenção de, neste trabalho, manter-me atado às possibilidades de resposta que possam, através da linguagem, expressar um sistema de relações causais compreensível e descritível através de algum sistema simbólico.

WITTGENSTEIN expõe de forma brilhante a distinção epistemológica entre os saberes em sua Conferencia Sobre Ética.

> "Toda a minha tendência, e creio que a de todos aqueles que tentaram alguma vez escrever ou falar de Ética ou Religião, é correr contra os limites da linguagem. Esta corrida contra as paredes de nossa jaula é perfeita e absolutamente desesperançada. A Ética, na medida em que brota do desejo de dizer algo sobre o sentido último da vida, sobre o absolutamente bom, o absolutamente valioso, não pode ser uma ciência. O que ela diz nada acrescenta, em nenhum sentido, ao nosso conhecimento, mas é um testemunho de uma tendência do espírito humano que eu pessoalmente não posso senão respeitar profundamente e que por nada neste mundo ridicularizaria." (Wittgenstein,1995)

Possuindo em mente esta distinção, proponho uma delimitação possível dos limites do espaço investigativo, excluindo-se deste espaço tudo aquilo que não aparenta ser passível de captura através da linguagem, tudo aquilo que não podemos explicar com a poderosa, mas limitada ferramenta da linguagem.

É importante ressaltar que acredito que o avanço da linguagem em relação à realidade possua uma natureza assintótica, desta forma, ele não é estático, crescendo sempre, aumentando os limites da "jaula" sem no entanto esgotar as possibilidades do real.

Creio que para realizar esta delimitação seja necessário abrir mão, ao menos temporariamente, do infinito atual, do infinito do instante e contentar-se

com teorias que possuam como referente, objetos e objetivos que se encontrem dentro dos limites do dizível.

Certamente, existem numerosos e valorosos pensadores que defendem a controverso argumento de que TUDO possa ser descrito (com rigor) pela linguagem, valendo-se do fato de que a impossibilidade de testar e invalidar cada uma das proposições/teorias possíveis dificulta a refutação desta tentadora via.

No entanto, até os mais radicais reducionistas tendem a concordar com o fato de que, até agora, todas as tentativas de descrever a realidade através de teorias foram felizes enquanto mantiveram-se dentro de um espaço investigativo restrito.

Uma eloqüente exposição de um reducionismo consciente pode ser encontrada em "A Perigosa Idéia de Darwin" (DENNETT,1998).

Dennett utiliza como uma metáfora para referenciar-se a tudo que se encontre fora do espaço investigativo da linguagem o termo "Skyhook" retirado da aeronáutica, que significa um ponto suspenso no céu, uma causa incausada que serviria de fundamento para todo resto e compara as teorias que buscam "Skyhooks" às teorias que buscam "gruas", causas que são também conseqüências e que não explicam a totalidade dos fenômenos, mas apenas parte deles.

> "Quem suspira pelos SkyHooks chama de reducionistas os que saem ansiosamente em defesa das gruas, e são capazes muitas vezes de fazer o reducionismo parecer filisteu e insensível, se não totalmente nocivo..."
> (Dennett, 1998)

A posição especifica deste trabalho é a de que o uso reducionista de "gruas" conduz a resultados importantes. No entanto, DENNETT parece acreditar que esta técnica possa dar conta de todos os problemas propostos, que o espaço investigativo da ciência seja todo o espaço disponível, que não existam fenômenos ou conjuntos de fenômenos cuja natureza não seja capturável através de um sistema simbólico rigoroso. DENNETT acredita na inexistência total de fenômenos não algoritmizaveis enquanto seus opositores defendem a impossibilidade de algoritmização de parte ou de todos os fenômenos presentes na experiência vital.

1.2 – A tensão geradora de teorias.

Coloco então, como ponto de partida, o seguinte encontro de certezas conflitantes:

> a) Possuímos acesso a fenômenos complexos que não podem, como um conjunto, serem destrinchados por uma descrição de cadeias causais expressa pela linguagem.
>
> b) Possuímos a necessidade de sanar convincentemente nossa ignorância a respeito destes fenômenos. Percebemos que qualquer resposta aceitável deve provir da linguagem, que por sua vez não possui capacidade para a geração desta resposta.

A presença incontestável da experiência vital nos coloca entre duas certezas contraditórias, de um lado, a indiscutível presença de algo que É, de outro uma aparente permanência da incapacidade de finalizar nossa tarefa explicativa, aparentemente devido a um desencontro estrutural entre a ferramenta disponível para geração de teorias explicativas, a linguagem, e o alvo de todas as teorias, a realidade ou aspectos da realidade.

> "É sintomático, que quando nos encontramos com um destes problemas que parecem insolúveis, o problema esteja revestindo uma determinada estrutura lógica. Por um lado, possuímos uma crença ou conjunto de crenças as quais não podemos renunciar, mas por outro lado, possuímos outra crença ou conjunto de crenças contraditório com o primeiro conjunto e que parece tão irrenunciável quanto este."
> (Searle, 1997)

O fato de percebermos as limitações da linguagem não nos exime da tarefa de responder com teorias às perguntas geradas pela apresentação dos fenômenos.

Creio que o caminho, apontado por diversos autores, como por exemplo KANT[4] (Kant, 1974) e CHALMERS (Chalmers, 1995), é prosseguir na geração de teorias, elucidando processos isolados cuja enunciação através de um sistema formal seja possível e liberar aquilo que aparentemente não se sujeita ao jugo da linguagem para o domínio do misticismo e da metafísica.

A motivação primária deste trabalho, não difere, portanto da motivação de todas as outras teorias geradas pelo homem com o intuito de saciar sua sede de conhecimento.

> "É o espanto que impele o cientista à "dissipar a ignorância" e que fez Einstein dizer: "O eterno mistério do mundo [isto é, do universo] é sua compreensibilidade" portanto, todo o "desenvolvimento" subseqüente de teorias que correspondam à compreensibilidade do universo "é, numa certa medida, uma fuga constante do assombro.""
> (Arendt, 1971)

Seu desdobramento, no entanto, parece apontar para a necessidade de criação de teorias especificas, que respondam a aspectos isolados do fenômeno da experiência vital.

1.2.1 - Primeira percepção irrenunciável, a existência da Experiência Vital.

O principal elemento motivador deste esforço investigativo não é diferente da motivação geral de todas as outras teorias que visam explicar, com maior ou menor abrangência, aspectos da realidade na qual nos encontramos inseridos cotidianamente, independentemente de nossa vontade.

Este elemento motivador é a presença não negociável da realidade propriamente dita, que nos instiga a explica-la, desafiando-nos a compreende-la inteira ou parcialmente.

A negação da existência objetiva do real ou o deslocamento de sua origem para o âmbito subjetivo da consciência não elimina o problema, mas apenas o desloca conservando integralmente seu mistério, independentemente

[4] Immanuel Kant (1724-1804)

do ***lócus*** de emanação da realidade (mundo, mente ou ambos). Esta emanação requer uma explicação que deve dar conta do conjunto de aspectos da Experiência Vital à qual somos submetidos durante o período compreendido entre o nascimento e a morte.

A existência da Experiência Vital, independentemente da **verdade** a respeito desta experiência (seja esta experiência objetiva, subjetiva, real, ilusória, provocada por um gênio maligno ou um bom deus), nos leva a concluir que algo existe, algo funda nossas experiências e que, em nenhum período da história humana dominamos, de forma satisfatória a natureza exata deste algo.

Esta experiência vital compartilhada, em um espaço comum, pode ser definida como uma amalgama entre o contato involuntário com o conjunto dos entes que povoam o mundo (entes de todos os tipos, animados e inanimados) e o componente pessoal que faz com que cada um de nós desconfie de que apesar da bem sucedida comparação entre descrições de objetos, algo pareça manter-se pessoal e não comparável.

Temos então, uma experiência única, simultaneamente objetiva (por que possui diversos aspectos compartilhados) e subjetiva (por que um dos aspectos compartilhados é, paradoxalmente, aquilo que torna as experiências pessoais e diferencia um ente do outro).

> "O que nos garante a objetividade do mundo no qual vivemos é que este mundo é comum a nós e a outros seres pensantes. Mediante as comunicações que estabelecemos com os outros homens, recebemos deles raciocínios prontos; sabemos que esses raciocínios não vem de nós e, ao mesmo tempo, reconhecemos neles a obra de seres racionais como nós. E como esses raciocínios parecem aplicar-se ao mundo de nossas sensações, cremos poder concluir que esses seres racionais viram a mesma coisa que nós; é assim que sabemos que não estávamos sonhando.
>
> Esta é, portanto a primeira condição de objetividade: o que é objetivo deve ser comum a vários espíritos, e por conseguinte poder ser transmitido de um a outro."
>
> (Poincaré[5],1998)

[5] Henri Poincaré (1854-1912)

O conjunto dos objetos que compõe o grande estado de coisas da realidade exerce pressão sobre todos os seus elementos levando-os a uma ou varias reações que visam geralmente preservar a persistência de sua forma no conjunto. A maioria dos componentes deste conjunto parece reagir a esta pressão dando continuidade à sua existência através de ações de primeira pessoa, ações não planejadas que garantam sua adaptação a mudanças no meio.

O ser humano, entretanto difere-se dos outros elementos deste vasto conjunto ao executar um salto sobre si mesmo e tornar o mundo e a si mesmo em objetos de um conhecimento de terceira pessoa, objetos e combinações de objetos que podem e devem ser compreendidos independentemente da necessidade básica de adaptação para sobrevivência.

O ser humano possui, portanto, além da motivação instintiva de manutenção de sua existência uma necessidade de gerar teorias explicativas que o retirem do horror gerado pela consciência da existência do mundo e de sua simultânea ignorância em relação e este mundo (que inclui ele próprio). Esta necessidade sofisticada, gerada por algo mais sutil do que fatos isolados passiveis de captura através de sistemas simbólicos (Linguagens) como o frio e a fome, leva o ser humano a tomar atitudes atípicas para os outros entes que povoam a realidade. A geração de teorias explicativas da realidade é a atividade que busca anular o mal estar de saber-se no mundo e simultaneamente saber-se ignorante em relação às causas e meios através dos quais esta experiência ocorre.

> "Os que comparecem como espectadores ao festival da vida são tomados por pensamentos de admiração, que são, então, postos em palavras."
> (Arendt,1971)

O assombro provocado por esta percepção aumenta ao longo do tempo, de vida do homem e da humanidade, à medida que constatamos a insuficiência das teorias disponíveis em estabelecer uma resposta definitiva para os questionamentos gerais a respeito da natureza.

Vale ressaltar que a experiência que motiva a geração de teorias não se reduz à presença do mundo exterior. A experiência envolve a presença do mundo exterior (objeto) e a presença, tão ou mais instigante e misteriosa, da consciência que o percebe (sujeito).

Todos os elementos do conjunto da realidade sofrem o efeito do contato com o conjunto do qual fazem parte, mas o grau de percepção deste fenômeno e as reações automáticas ou intencionais a ele variam grandemente.

Um objeto inanimado (como uma pedra ou uma mesa) não possui (até onde os conhecemos) nenhum grau de conhecimento do que se passa, mas sua estrutura reage às condições físicas aumentando ou reduzindo sua persistência, o valor de seus atributos se alteram passivamente de acordo com as variações do conjunto, mas sem sombra de dúvida existe uma variação, o que faz com que possamos considera-los como sensíveis, em algum nível, à existência.

Uma estrutura biológica muito simples como uma Ameba ou um vegetal já possui, além das características dos seres inanimados uma capacidade (aparentemente não intencional) de reagir ativamente às variações do conjunto, melhorando suas chances de aumentar seu tempo de persistência.

Os seres humanos por sua vez parecem possuir cada um destes diversos níveis de relacionamento com o conjunto além de suas características únicas.

A uma parte destas características únicas dos seres humanos é dado o nome de **mente**, um conceito genérico para um eclético conjunto de propriedades, portanto, a motivação para desvendar o sujeito é tão grande quanto a motivação pelo objeto, e os investigadores devem se lançar com igual afinco em ambas as direções[6].

1.2.2 – Segunda percepção irrenunciável, a realidade é completa, as teorias incompletas.

Antes de prosseguir na exposição do segundo ponto impulsionador das teorias explicativas da realidade, cabe definir o sentido da palavra conhecer dentro do escopo desejado e necessário para este trabalho.

[6] Mantendo a desmedida esperança de que a resposta, se é que existe uma resposta nos moldes buscados, seja uma só.

Na Grécia antiga, no período Helênico que se inicia em torno do século V AC começa a formar-se um determinado enquadramento do mundo como algo explicável não através do mito, relacionado diretamente com as sensações, mas através da razão, representada pelo *Logos.* A partir deste momento, conhecer passa a ser conhecer através da linguagem, expor através de um sistema simbólico predeterminado a natureza exata de cada um dos elementos do conjunto da realidade.

A ciência ganha então, a tarefa de trazer todos os mistérios para a luz da linguagem, munida inicialmente da esperança de que esta tarefa seja finita e realizável.

Segundo POINCARRÉ (Poincarré,1998), a ciência pode ser descrita como "Conhecimento racional estruturado e acumulado", a análise hermenêutica total dos objetos de estudo de forma que o objeto torne-se totalmente conhecido, desprovido de mistério.

Esta captura se daria através da construção de um sistema formal capaz de explicar e, na maioria dos casos, reproduzir o objeto estudado quando necessário ou quando solicitado.

Segundo HOFSTADTER (2001), o objetivo do desenvolvimento da lógica como disciplina independente foi o desejo de capturar os processos mentais em um sistema formal.

> "...Nos últimos dois séculos, o método axiomático veio a ser explorado com poder e vigor crescentes...Gerou-se assim uma opinião em que fica tacitamente pressuposto que todo setor do pensamento matemático pode ser dotado de um conjunto de axiomas suficiente para desenvolver sistematicamente a totalidade infinita das verdadeiras proposições...".
> (Nagel e Newman, 1998)

Aparentemente, este ponto na história diferencia definitivamente as formas de busca e valorização do conhecimento em voga no oriente, onde continua existindo uma concepção de que conhecer não precisa corresponder ao enquadramento de todos os aspectos do objeto da observação através de uma descrição formal.

Apesar de nossas tentativas de descrever a realidade como um todo permanecerem em um ponto de eqüidistância epistemológica entre si, ou seja, não conseguimos ainda gerar uma teoria que seja absolutamente completa e sem falhas, podemos descrever processos isolados com razoável sucesso, e utilizar estas descrições para manipular objetos do mundo real.

> "Uma das mais incomodas conseqüências dos recentes avanços da ciência é o fato de que estes avanços nos revelam que sabemos menos do que podíamos supor. Quando eu era jovem, todos sabiam, ou pensavam que sabiam, que um homem consiste de uma alma e um corpo; que o corpo encontra-se no tempo e no espaço e que a alma se encontra apenas no tempo.
>
> Se a alma sobrevive ou não a morte, é um problema sobre o qual as opiniões sempre divergiram, mas o fato de haver uma alma era considerado indubitável.
>
> Em relação ao corpo, o senso comum com certeza considera sua existência auto-evidente, e da mesma forma procede o cientista, mas o filósofo possui meios de questionar esta existência de uma forma ou de outra, reduzindo a existência do corpo a idéias na mente do individuo que possui o corpo.
>
> O filosofo no entanto, não é levado a sério, e a ciência permanece confortavelmente materialista.[7]"
>
> (Russel,1928)

O ser humano parece possuir uma grande capacidade de compreender processos isolados e uma pequena capacidade de compreender a existência como um todo.

Na busca pelas verdades constantes na Experiência Vital, dois determinados ramos investigativos parecem ser particularmente suscetíveis à

[7] "One of the most painful circumstances of recent advances in science is that each one makes us know less than we thought we did. When I was young we all knew, or thought we knew, that a man consists of a soul and a body; that the body is in time and space, but the soul is in time only. Whether the soul survives death was a matter as to which opinions might differ, but that there is a soul was thought to be indubitable. As for the body, the plain man of course considered its existence self-evident, and so did the man of science, but the philosopher was apt to analyse it away after one fashion or another, reducing it usually to ideas in the mind of the man who had the body and anybody else who happened to notice him. The philosopher, however, was not taken seriously, and science remained comfortably materialistic, even in the hands of quite orthodox scientists." (Minha Tradução)

incapacidade da linguagem em abordar aspectos simultâneos de sistemas complexos.

Estes ramos seriam a filosofia da matemática e a Inteligência Artificial. Ambos experimentam problemas semelhantes cujas metáforas podem ser utilizadas para compreender o outro.

Na matemática, o problema da incapturabilidade do real através da linguagem parece ter sido mantido, ao longo de uma duradoura tradição, imerso sob o poder dos axiomas e de sua difícil e subjetiva interpretação.

A principio, um axioma é uma proposição cuja verdade é auto-evidente, definição inerentemente vaga e subjetiva.

No entanto, um dos mais famosos axiomas da matemática, o quinto postulado de Euclides, responsável pela afirmação de que por um ponto externo em uma reta passa uma e apenas uma paralela a esta reta, permanece como um incomodo para gerações de matemáticos que pressentiam que sua asserção não possuía a auto-evidencia de um axioma, e simultaneamente não se encontrava fundada, não derivava através de regras lógicas, dos outros axiomas existentes.

O quinto postulado não era, portanto, nem um axioma nem um teorema o que fazia com que o sistema Euclidiano não fosse considerado um modelo perfeito de completude e consistência.

> "A principal razão para esta alegada falta de auto-evidência parece ter sido o fato de que o axioma das paralelas faz uma afirmação sobre regiões inteiramente remotas do espaço. Euclides define as linhas paralelas como retas em um plano que, "sendo estendidas indefinidamente em ambas as direções", não se encontram. Conseqüentemente, afirmar que duas linhas são paralelas é pretender que as duas linhas não se encontrarão sequer "no infinito". Mas os antigos estavam familiarizados com linhas que, embora não se cortem umas as outras em qualquer região finita do plano, se encontram "no infinito". Tais linhas são denominadas "assintóticas". Assim, uma hipérbole é assintótica aos seus eixos. Não era portanto intuitivamente evidente para os antigos geômetras que de um ponto fora de uma reta dada, apenas uma reta pudesse ser traçada que não fosse encontrar a reta dada, mesmo no infinito"
> (Nagel e Newman, 1998)

Um modelo completo é aquele no qual todos os teoremas (proposições) possíveis podem ser geradas através da aplicação das regras lógicas de derivação aos axiomas preexistentes do sistema.

Um modelo consistente é aquele no qual a aplicação de uma determinada regra lógica de derivação não gera, sob hipótese alguma, verdades contraditórias, paradoxais.

Durante um período de aproximadamente 2000 anos, o postulado das paralelas incomodou os mais rigorosos, mas ninguém se atreveu a questiona-lo seriamente, até que, após Gauss, percebe-se que seria possível criar um sistema consistente utilizando outro postulado que não o quinto postulado de Euclides, o postulado das paralelas.

Esta descoberta abriu os olhos dos matemáticos do século 19 para a necessidade de axiomatização de todos os campos da matemática, necessidade esta que conduziu a barreiras aparentemente intransponíveis para nosso principal método de construção de conhecimento, a linguagem, e este problema persiste, ainda hoje, não apenas na matemática mas em todos os campos do conhecimento onde exista a ambição de criar respostas que atendam simultaneamente a dois requisitos: ser capaz de gerar respostas para todas as questões que possam ser propostas e evitar incômodos paradoxos, evitando a geração de respostas contraditórias.

1.3 – Como o problema epistemológico se aplica ao estudo da mente?

O estudo dos fenômenos que recaem sob o conceito “mentais” parece recair no mesmo problema epistemológico descrito acima, parecem existir diversos aspectos passiveis de descrição e compreensão através de teorias rigorosas construídas com sistemas simbólicos (linguagem) e diversos aspectos que resistem a este tipo de investigação.

No conjunto das disciplinas que buscam a compreensão dos processos mentais, a principal divisão se dá entre aqueles que acreditam que estes processos possam ser capturados através de um sistema de regras formais e aqueles que não acreditam na possibilidade deste feito.

As demais subdivisões parecem ser fruto de divergências metodológicas entre os que defendem a possibilidade de algoritmização total do conjunto dos

fenômenos mentais e divergências argumentativas entre os partidários da metafísica total que defendem a impossibilidade de algoritmização de qualquer processo mental, alem dos inúmeros tons de cinza existentes entre eles.

Este trabalho busca expor um destes tons de cinza, território onde pode ser admitido simultaneamente a existência de fenômenos inefáveis e fenômenos algoritmizáveis.

Em outras palavras, de acordo a taxonomia proposta por JOHN SEARLE (Searle,1998), possuímos três grandes grupos: Dualistas de Substancia, Dualistas de Propriedade e Materialistas.

Dualistas de substancia acreditam que a alma (ou o que quer que este símbolo represente) possua uma existência própria e não física, que existam de fato duas substancias, uma física e outra metafísica, além da física.

> "E se quer uma definição da alma, e saber o que ela é, respondo facilmente: É substancia dotada de razão, apta a reger um corpo (*substantia quaedam rationis particeps, regendo corpori acommodata*)."
> (Santo Agostinho, 1997)

Dualistas de propriedade não defendem a hoje impopular hipótese citada acima, mas acreditam que uma mesma (e única) substancia, física, possa possuir propriedades não físicas, uma subdivisão dos dualistas de propriedade são os epifenomenalistas, que acreditam que a propriedade metafísica seja causada ou emergida a partir de uma determinada configuração das propriedades físicas. Desta forma, a substancia única possuiria propriedades distintas sendo a propriedade metafísica uma conseqüência da complexidade atingida pela propriedade física, cérebros não são a consciência, mas são responsáveis por ela.

Materialistas acreditam, de uma forma geral, que só exista uma substancia e uma propriedade, purgando o estudo da mente de qualquer traço de metafísica, podem também ser divididos em inúmeros grupos dentre os quais cabe destacar o funcionalismo, subdivisão do materialismo segundo o qual um determinado mecanismo deve ser capaz de executar todas as funções de outro mecanismo caso o primeiro possua uma estrutura causal interna e um conjunto de instruções capazes de implementar as funções do segundo.

As três hipóteses possuem vantagens e desvantagens epistemológicas:

Algumas características interessante de teorias que situam a mente em áreas do conhecimento mais ou menos passiveis de investigação científica (excluindo o dualismo de substancia) pode ser obtida através de uma breve comparação entre os seguintes autores: Daniel Dennett, John Searle, Roger Penrose e David Chalmers.

Segundo **Daniel Dennett** , todos os fenômenos mentais podem ser reduzidos a operações simples em um nível algorítmico, para DENNETT toda complexidade pode ser decomposta e conhecida através da descrição de suas partes por algum sistema simbólico.

> "O termo algoritmo descende do latim (algorismus) passando pelo inglês antigo (algorisme e, erroneamente, a partir daí, algorithm), do nome de um matemático persa, Mûusâ al-Khowârizm, cujo livro sobre procedimentos matemáticos, escrito por volta de 835 d.C., foi traduzido para o latim no século XII por Abelardo de Bath ou Roberto de Chester. A idéia de que o algoritmo é um procedimento infalível de comprovação e de certa forma mecânico existe a séculos, mas foi o trabalho pioneiro de Alan Turing, Kurt Godel e Alonzo Church, na década de 1930, que de certa forma fixou a nossa atual compreensão do termo."
> (Dennett, 1998)

Dennett propõe três características básicas que subjazem sob o termo algoritmo.

> "1 – **Neutralidade do substrato**: O procedimento para divisões longas funciona igualmente bem com lápis ou caneta, papel ou pergaminho, luzes néon ou a fumaça expelida dos aviões, usando o sistema simbólico que você preferir. A eficiência do procedimento deve-se à sua estrutura lógica, não às eficiências causais dos materiais usados na comprovação, desde que as eficiências causais permitam seguir exatamente as etapas prescritas
>
> 2 – **Irracionalidade subjacente**: Embora o desenho do procedimento em geral seja brilhante, ou produza resultados brilhantes, cada uma

> de suas etapas, assim como a transição entre elas, é impressionantemente simples. Simples o bastante para que possa ser executado por um dispositivo totalmente mecânico.
>
> 3 – **Resultados garantidos**: Seja o que for que um algoritmo fizer, ele o faz sempre, se for executado sem erros. Um algoritmo é uma receita infalível."
> (Dennett, 1998)

DENNETT expõe claramente uma de suas opiniões mais controversas, a idéia de que qualquer processo pode ser reduzido a uma receita composta de passos simples, que pode ser executada sobre uma grande variedade de materiais indiferentemente e que se devidamente executada produz sempre o mesmo resultado.

> "Voltando a questão levantada anteriormente, existem limites para o que se possa considerar um processo algorítmico?
> Acho que a resposta é não. Se você quisesse, poderia tratar qualquer processo no nível abstrato como um processo algorítmico."

Desta forma, DENNETT se coloca como um defensor da hipótese de que é possível re-implementar todos os processos que compõem a mente em um artefato cujo substrato físico seja diferente do cérebro humano.

DENNETT sustenta a corrente, rotulada de Inteligência Artificial Forte, de que todos os processos que compõem a mente humana sejam suscetíveis a uma decomposição lógica em passos simples que executados resultariam no processo original. Desta forma abre-se a possibilidade para, entre outras coisas, mentes de silício (ou qualquer outro material) com capacidades cognitivas semelhantes ou idênticas às nossas.

A crença de DENNETT no reducionismo científico encontra interessantes refutações nas idéias de **JOHN SEARLE.**

O principal ponto de discordância é o fato de SEARLE afirmar a impossibilidade de algoritmização da totalidade dos processos mentais, SEARLE não concorda com a possibilidade de que se possa copiar os atributos da mente humana e manter-se de acordo com o principio na neutralidade do substrato.

> "Vou resumir minha posição sobre como a pesquisa do cérebro pode continuar respondendo às perguntas que nos instigam, a saber, se o cérebro é um órgão como qualquer outro; se é uma máquina orgânica. A consciência é causada por processos mentais de nível inferior no cérebro e é, por si só, uma propriedade do cérebro. Por ser uma propriedade que surge através de certas atividades dos neurônios, podemos vê-la como uma "propriedade emergente" do cérebro. Uma propriedade emergente de um sistema é aquela que é causalmente explicada pelo comportamento dos elementos do sistema; mas, não é uma propriedade de quaisquer elementos individuais e não pode ser explicado simplesmente como a soma das propriedades destes elementos."
> (Searle,1998)

Segundo SEARLE a reprodução física das estruturas que causam os fenômenos mentais pode (caso seja perfeita em detalhes) ocasionar a emergência da totalidade destes fenômenos, no entanto, Searle acredita que não se possa isolar o processo das características físicas inerentes aos elementos que compõe o todo complexo. Segundo este pensamento, a emergência dos fenômenos mentais poderia ser ao menos teoricamente reproduzido, mas de forma alguma descrito através de um sistema simbólico que isole o processo de sua base material.

Um antagonista de DENNETT ainda mais radical do que SEARLE é o físico e matemático inglês **ROGER PENROSE.**

Segundo PENROSE, assim como para Searle, os fenômenos mentais podem ser compreendidos não por sua descrição algorítmica, mas pelas características físicas de seus componentes, no entanto, PENROSE dá um passo além quando propõe uma natureza especifica e inusitada para os componentes físicos mínimos da mente.

Para SEARLE, os processos mentais mínimos são aqueles que se dão no nível das atividades neuronais, para PENROSE estes processos ainda são demasiadamente grosseiros (grandes) para dar conta da emergência de fenômenos que, segundo ele, não poderiam ser causados por causas que estejam vinculadas à física clássica.

PENROSE propõe que os comportamentos gerados pela união de materiais que obedecem às regras da física clássica seria insuficiente para reproduzir as características não algorítmicas da mente, e aventa a hipótese de que, em um nível sub neuronal a matéria da qual emerge a consciência possua um comportamento superposicionista, regido pelas leis da física quântica.

Uma das inspirações para busca de subconjuntos de fenômenos que possam ser conhecidos através da linguagem é a obra do filósofo australiano **DAVID CHALMERS**.

Segundo CHALMERS, o rótulo geral consciência utilizado para referenciar o conjunto dos fenômenos mentais é bastante amplo, permitindo sua subdivisão em uma série de grupos de problemas com características diferentes, CHALMERS assume então um comportamento para cada grupo de problemas, agindo de forma reducionista, como DENNETT, em relação ao que convencionou chamar de “Easy Problem” e de forma Metafísica em relação ao que chama de “Hard Problem”

> "Não existe apenas um problema da consciência. "Consciência" é um termo vago e ambíguo, que se refere a vários fenômenos diferentes. Cada um destes fenômenos deve ser investigado, mas algumas são mais fáceis de explicar do que outros. Para começar, uma medida útil é dividir os problemas abrigados sob o rótulo de consciência em dois grupos, os problemas "fáceis" e os problemas "difíceis". Os problemas fáceis são aqueles que podem ser explicados em termos de mecanismos computacionais ou neurais. Os problemas difíceis são aqueles que parecem resistir a estes métodos.[8]"
>
> (Chalmers, 1995)

Para CHALMERS, os fenômenos mentais referentes à nossa individualidade, nossos gostos, emoções e percepções subjetivas qualitativas,

[8] “There is not just one problem of consciousness. "Consciousness" is an ambiguous term, referring to many different phenomena. Each of these phenomena needs to be explained, but some are easier to explain than others. At the start, it is useful to divide the associated problems of consciousness into "hard" and "easy" problems. The easy problems of consciousness are those that seem directly susceptible to the standard methods of cognitive science, whereby a phenomenon is explained in terms of computational or neural mechanisms. The hard problems are those that seem to resist those methods.” (M.T.)

fazem parte do “Hard Problem”, parte do conjunto de características mentais sobre as quais a descrição através da linguagem, e sua conseqüente geração de conhecimento não alcançam um nível de descrição adequado.

CHALMERS oferece uma concisa lista de habilidades consideradas por ele como integrantes do ”Easy Problem”:

> "Os problemas "fáceis" da consciência incluem aqueles que explicam os seguintes fenômenos:
>
> a) A habilidade de discriminar, categorizar e reagir a estímulos do meio ambiente;
>
> b) A integração da informação por um sistema cognitivo;
>
> c) A capacidade de reportar estados mentais;
>
> d) A habilidade de um sistema de acessar seus próprios estados internos;
>
> e) A atenção;
>
> f) O controle deliberado do comportamento;
>
> g) A capacidade de distinguir entre a vigília e o sono;[9]"
>
> (Chalmers, 1995)

Considerando esta lista, podemos localizar o processo cognitivo alvo deste trabalho nos itens A e B, a integração de informação por um sistema cognitivo. Mais especificamente, este item pode ser compreendido através do questionamento acerca das habilidades através das quais reconhecemos os

[9] “The easy problems of consciousness include those of explaining the following phenomena:
- the ability to discriminate, categorize, and react to environmental stimuli;
- the integration of information by a cognitive system;
- the reportability of mental states;
- the ability of a system to access its own internal states;
- the focus of attention;
- the deliberate control of behavior;
- the difference between wakefulness and sleep.” (Minha Tradução)

objetos como instancias de classes, acrescentamos classes a um modelo e checamos a compatibilidade entre os objetos componentes de um sistema.

> "Todos estes fenômenos estão associados com a noção de consciência. Por exemplo, em algumas situações podemos afirmar que um estado mental é consciente quando ele pode ser verbalmente reportado, ou quando é internamente acessível. Algumas vezes um sistema é considerado consciente de alguma informação quando ele possui a habilidade de reagir baseado nesta informação, ou, utiliza esta informação, ou quando pode integrar esta informação e explora-la com um controle sofisticado de seu comportamento. Algumas vezes dizemos que uma informação é consciente precisamente quando é deliberada. Em outras, dizemos que um organismo é consciente como uma outra forma de dizer que o organismo esta acordado.[10]"
>
> (Chalmers, 1995)

Segundo CHALMERS, as características integrantes do conjunto nomeado como "Easy Problem" podem ser reduzidas a processos algorítmicos.

> "Não existe uma verdadeira questão a respeito da possibilidade de se explicar estes fenômenos cientificamente. Todos eles são de alguma forma vulneráveis a explicações em termos de mecanismos computacionais ou neurais. Por exemplo, para explicar as capacidades de acesso e reportabilidade de estados internos, nós precisamos apenas especificar os mecanismos através dos quais informações sobre estados internos são recolhidas e tornadas disponíveis para relatórios verbais.

[10] "All of these phenomena are associated with the notion of consciousness. For example, one sometimes says that a mental state is conscious when it is verbally reportable, or when it is internally accessible. Sometimes a system is said to be conscious of some information when it has the ability to react on the basis of that information, or, more strongly, when it attends to that information, or when it can integrate that information and exploit it in the sophisticated control of behavior. We sometimes say that an action is conscious precisely when it is deliberate. Often, we say that an organism is conscious as another way of saying that it is awake." (Minha Tradução)

Para explicar a integração de informação, nós devemos apenas exibir mecanismos através dos quais informações são armazenadas em conjunto e explorada por processos posteriores.

Em relação à questão da vigília e do sono, seria suficiente uma resposta neurobiológica apropriada, responsável pelo comportamento contrastante dos indivíduos nos diferentes estados.

Em cada um destes casos, uma abordagem cognitiva ou neurofisiológica pode claramente fornecer respostas satisfatórias.

Se estes fenômenos fossem tudo o que há a respeito da consciência, então a consciência não seria um problema tão grande.

Embora nós ainda não possuímos nada nem remotamente próximo de uma resposta completa destes fenômenos, nós possuímos uma clara idéia de como nós devemos explica-los.

Este é o motivo pelo qual eu chamo estes problemas de "fáceis". Obviamente "fácil" é um termo relativo. Considerando o árduo caminho e a grande quantidade de detalhes, deve levar um século ou dois de árduo trabalho empírico. Ainda assim, existem todas as razões para acreditar que os métodos das ciência cognitivas e da neurociência irão obter sucesso.[11]"

(Chalmers, 1995)

[11] "There is no real issue about whether these phenomena can be explained scientifically. All of them are straightforwardly vulnerable to explanation in terms of computational or neural mechanisms. To explain access and reportability, for example, we need only specify the mechanism by which information about internal states is retrieved and made available for verbal report. To explain the integration of information, we need only exhibit mechanisms by which information is brought together and exploited by later processes. For an account of sleep and wakefulness, an appropriate neurophysiological account of the processes responsible for organisms' contrasting behavior in those states will suffice. In each case, an appropriate cognitive or neurophysiological model can clearly do the explanatory work.

If these phenomena were all there was to consciousness, then consciousness would not be much of a problem. Although we do not yet have anything close to a complete explanation of these phenomena, we have a clear idea of how we might go about explaining them. This is why I call these problems the easy problems. Of course, "easy" is a relative term. Getting the details right will probably take a century or two of difficult empirical work. Still, there is every reason to believe that the methods of cognitive science and neuroscience will succeed " (M.T.)

No entanto, ao contrário de DENNETT, CHALMERS não se atreve a reduzir a processos algoritmizaveis as características que fazem parte do “Hard Problem”.

> "O verdadeiro problema "difícil" da consciência é o problema da experiência. Quando nós pensamos e percebemos, parte do fenômeno pode ser atribuído a processos de informação, mas existe também um aspecto subjetivo. Como colocado por Nagel (1974), existe algo que é ser um organismo consciente, este algo é o aspecto subjetivo da experiência.
>
> Quando vemos, por exemplo, nós experimentamos sensações visuais: a sensação de vermelho, a experiência do escuro e do claro, a qualidade de definição da imagem em um campo visual. Outras experiências também possuem percepções de diferentes modalidades: o som de um clarinete, o cheiro das flores. Existem portanto, sensações corporais, de dores a orgasmos; imagens mentais que são conjuradas internamente, o sentimento qualitativo da emoção, e a experiência de uma seqüência de pensamentos conscientes. Aquilo que une todos estes estados é o fato de que existe algo que faz parte de todos eles. Todos eles são estados da experiência.[12]"

Uma forma possível de estabelecer os limites entre o pensamento destes autores, DENNETT, SEARLE, PENROSE e CHALMERS é enxerga-los através da ótica da função epistemológica da linguagem, se fizermos isto, poderemos concluir que cada um deles concede à linguagem poderes diferentes, movendo, de acordo com sua crença, a fronteira entre o que pode e o que não pode ser dito.

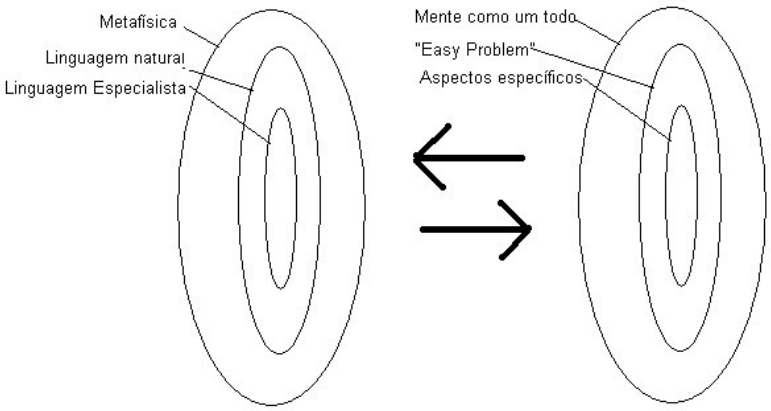
[12] “The really hard problem of consciousness is the problem of experience. When we think and perceive, there is a whir of information-processing, but there is also a subjective aspect. As Nagel (1974) has put it, there is something it is like to be a conscious organism. This subjective aspect is experience. When we see, for example, we experience visual sensations: the felt quality of redness, the experience of dark and light, the quality of depth in a visual field. Other experiences go along with perception in different modalities: the sound of a clarinet, the smell of mothballs. Then there are bodily sensations, from pains to orgasms; mental images that are conjured up internally; the felt quality of emotion, and the experience of a stream of conscious thought. What unites all of these states is that there is something it is like to be in them. All of them are states of experience.” (M.T.)

Visto desta forma, poderíamos compreender a posição de DENNETT como uma crença inquebrantável na capacidade da linguagem em capturar as regras que geram a consciência, para SEARLE e PENROSE as causas da consciência estaria além dos limites do dizível, dos limites da linguagem e para CHALMERS, parte do problema estaria ao alcance de um tratamento lingüístico e outra parte estaria fora deste alcance.

Acredito que a divisão do problema da mente proposto por CHALMERS possa ser utilizada na construção de subdivisões no grupo dos fenômenos mentais cujo entendimento supõe-se possível através da linguagem.

1.3.1 – Linguagem e pensamento x Linguagens e pensamentos.

Surge neste ponto, pela primeira vez neste trabalho, a hipótese de que a linguagens naturais possuam uma grande amplitude para descrever de forma pouco rigorosa um grande número de fenômenos e de que linguagens específicas possuam uma pequena amplitude para descrever com mais rigor a um número reduzido de fenômenos.



Aparentemente, a linguagem natural sofre, devido a sua amplitude, do mal da dubiedade e da fraca delimitação dos contornos de seus referentes.

Proponho a hipótese de que exista uma relação com algum grau de proporcionalidade inversa entre a capacidade de abrangência de um sistema simbólico e sua capacidade para expressar rigorosamente o referente.

Este seria um motivo possível para a existência de linguagens especialistas. A linguagem natural permite que se fale com pouco rigor dos objetivos dos especialistas, mas se o objetivo é lançar luz sobre os mínimos aspectos de um determinado ponto de vista da realidade, a linguagem natural torna-se pouco eficiente por ser demasiadamente frouxa, demasiadamente dúbia.

O segundo aspecto da hipótese proposta é a de que para cada sistema simbólico especialista, possui um conjunto correspondente de referentes que consiste em um subconjunto da estrutura cognitiva.

Surgem então duas perguntas fundamentais ao desenvolvimento deste trabalho.

A) - Podemos utilizar os conceitos e as estruturas das linguagens para construir modelos de seus processos cognitivos correspondentes?

B) - A que subconjunto da estrutura cognitiva corresponderia um possível modelo gerado a partir de um sistema simbólico utilizado na composição das seqüências de instruções executadas por um computador, isto é, qual é a contrapartida cognitiva das linguagens de programação, e mais especificamente, qual é a contrapartida cognitiva das técnicas de analise e desenvolvimento de sistemas orientados a objetos e qual sua relação com nossa capacidade de agrupar e reconhecer objetos e classes?

No próximo capítulo exploraremos algumas diferentes relações entre linguagem e pensamento de forma a apoiar e esclarecer as afirmações acima.

II – Linguagem e Pensamento.

2.1 – Algumas posições a respeito do relacionamento entre linguagem e pensamento.

A opinião dos teóricos acerca do relacionamento entre a linguagem e pensamento encontra-se longe de um consenso.

As teses a este respeito variam grandemente entre uma relação "forte", onde a linguagem possua um papel fundamental na estrutura do pensamento. E uma relação "fraca", onde o papel da linguagem varia entre a visão de que a linguagem é um mero meio para externalização do pensamento, não participando de seus processos internos, até a corrente que afirma que a linguagem seja um auxiliar para o desenvolvimento dos processos mentais servindo entre outras coisas para a aquisição de conceitos e fortalecimento da memória.

CARRUTHERS propõe a seguinte taxonomia[13]:

> "Formas "fortes" incluem a visão de que a linguagem é conceitualmente necessária para o pensamento (aceita por muitos filósofos) e a visão de que a linguagem é de fato o meio de todo pensamento conceitual humano.[14]"
>
> (Carruthers,2001)

Este recorte possível das teorias a respeito do relacionamento entre linguagem e pensamento encontra apoio nas idéias de VIGOTSKI.

[13] Taxonomia refere-se à classificação das coisas, e aos princípios subjacentes da classificação. Quase tudo - objetos animados, inanimados, lugares e eventos - pode ser classificado de acordo com algum esquema taxonômico. (http://pt.wikipedia.org/wiki/Taxonomia)

[14] "Strong forms include the view that language is conceptually necessary for thought (endorsed by many philosophers) and the view that language is de facto the medium of all human conceptual thinking" (Minha Tradução)

> "O exame dos resultados das investigações anteriores sobre o pensamento e a linguagem mostrará que, desde a antiguidade até hoje, todas as teorias oscilam entre a identificação, ou fusão, do pensamento e da fala, por um lado, e sua disjunção e segregação igualmente absolutas, quase metafísicas, por outro. Seja expressando um desses extremos em sua forma pura, seja combinando-os, isto é, assumindo uma posição intermediaria – mas sempre em algum ponto ao longo do eixo que une os dois pólos – todas as diferentes teorias sobre o pensamento e a linguagem ficam restritas a esse circulo."
>
> (Vigotski, 2005)

Nas páginas que seguem pretendo expor em primeiro lugar uma versão da concepção popular do relacionamento entre linguagem e pensamento, em seguida algumas versões de teorias "fracas" da linguagem, sendo elas: a linguagem como mero instrumento de comunicação, a linguagem como instrumento de desenvolvimento diacrônico, a linguagem como modeladora do aparato cognitivo e a linguagem como instrumento de apoio do aparato cognitivo.

Mostrarei também algumas teorias "Fortes" da linguagem: A linguagem como elemento essencial do pensamento, o pensamento como linguagem e a linguagem como uma maquina virtual seqüencial sobre uma base conexionista.

2.1.1 – A resposta do senso comum, a linguagem como elemento essencial do pensamento.

Para o senso comum, não parece existir uma distinção clara entre o pensamento e a linguagem natural.

> "Muitas pessoas parecem gastar boa parte de sua atividade mental envolvidos em um "dialogo de si consigo", com proposições em linguagem natural ocupando uma proporção significativa do fluxo de suas consciências[15]"
>
> (Carruters, 2001)

Este elemento da sabedoria popular foi corroborado por uma pesquisa realizada entre 1990 e 1993 pelo professor R. Hulburt, que utilizou um método bastante simples para testar esta hipótese.

Um grupo de indivíduos foi instado a utilizar fones de ouvido durante períodos pré-estabelecidos de tempo, ocasionalmente, o fone produzia uma série de ruídos.

Quando um individuo ouvisse um ruído deveria "congelar" o que estivesse passando por sua cabeça naquele exato momento e realizar uma anotação explicativa a respeito deste pensamento.

Embora as incidências de dialogo interno variem bastante, todos os indivíduos considerados normais (em oposição aos esquizofrênicos), relataram a ocorrência de dialogo interno variando entre 70% e 80% do tempo.

Obviamente os resultados desta experiência podem ser questionados de uma série de formas, principalmente através da crítica ao tamanho reduzido do grupo de testes utilizado por HULBURT, no entanto, dentro do contexto deste trabalho, a pesquisa externaliza um interessante fato, de que para o pensamento comum, a resposta à questão acerca do relacionamento entre linguagem e pensamento é de que a linguagem é um elemento constitutivo do pensamento.

Esta resposta é integrante do grupo das "teorias fortes sobre o papel da linguagem no pensamento" que consideram a hipótese de que a linguagem possua, alem de um obvio papel comunicativo na intersubjetivação de pensamentos, uma participação ativa na cognição humana.

[15] "Many people seem to spend a good deal of their waking activity engaged in 'inner speech', with imaged natural language sentences occupying a significant proportion of the stream of their conscious mentality" (Minha Tradução)

2.1.2 - As respostas das ciências cognitivas, algumas Teorias "fracas" do relacionamento entre linguagem e pensamento.

2.1.2.1 - A linguagem como instrumento comunicativo.

Uma resposta diametralmente oposta, a forma mais radical das "teorias fracas sobre o papel da linguagem no pensamento" pode ser encontrada nas teses defendidas por diversos membros da comunidade das ciências cognitivas.

Alguns cientistas apóiam a posição oposta à do pensamento como diálogo de si consigo, confinando a linguagem a um papel meramente comunicativo, definindo-a como um canal, um meio para a entrada e saída de informação (pensamentos) que depois (uma vez dentro) deverá ser processada através de outras formas de representação.

Um dos argumentos para a validação da idéia de que a linguagem possua um papel meramente comunicativo é o fato de que a maioria dos cientistas cognitivos acredita hoje que a linguagem se encontre em um módulo distinto do cérebro, responsável pelo fluxo de entrada e saída de informações e que parece implausível que a linguagem possa exercer seu papel principal (entrada e saída) e também algum tipo de participação nos processos cognitivos.

Segundo CARRUTHERS, este argumento revela-se pouco consistente se considerarmos exemplos como o da imaginação visual.

> "Atualmente, praticamente todos pensam que o sistema visual encontra-se em um módulo distinto da mente, contendo uma boa parte de estruturas inatas. No entanto, a maioria dos cientistas cognitivos aceita também o fato de que a imaginação visual instancia e utiliza os recursos do módulo visual para propósitos cognitivos, por exemplo, muitas áreas

> comuns do cortex visual são ativadas tanto na atividade visual quanto na atividade imaginativa.[16]"
>
> (Carruters, 2001)

O que se torna aparente é a hipótese de que o módulo responsável pela cognição central pode cooptar os recursos dos módulos periféricos, ativando algumas de sua características para que sirvam de apoio às operações centrais da estrutura cognitiva.

O mesmo pode então ocorrer com a linguagem. Se considerarmos que a linguagem emerge de uma área específica do cérebro, parece uma hipótese bastante consistente que a estrutura cognitiva central possa acessar e utilizar seus recursos quando estes se mostrarem necessários em certos processos de pensamento.

Uma defesa radical das teorias fracas sobre a participação da linguagem no pensamento parece ser dificultada pela presença de algumas atividades cognitivas onde o uso de imagens e da linguagem é mais evidente, impedindo uma defesa radical da teoria comunicativa da linguagem através do argumento de que as características oriundas de outros centros de controle, distintos do centro responsável pela cognição central, não possam ser instanciadas quando necessário.

> "Nota-se também, que dificilmente alguém irá sustentar que a imaginação visual seja um mero epifenomeno dos processos cognitivos centrais, não possuindo nenhuma influencia no desenvolvimento e no decorrer destes processos. Pelo contrario, parece mais provável que existem várias tarefas que nós não podemos resolver facilmente sem desenvolver alguma forma de imagem visual. Por exemplo, suponha que lhe seja pedido (oralmente)

[16] "Almost everyone now thinks that the visual system is a distinct input-module of the mind, containing a good deal of innate structure. But equally, most cognitive scientists now accept that visual imagination re-deploys the resources of the visual module for purposes of reasoning – for example, many of the same areas of the visual cortex are active when imagining as when seeing." (Minha tradução)

> que descreva o formato da letra 'A'. Parece inteiramente plausível que o sucesso nesta tarefa deve depender da geração de uma imagem visual da letra, a partir da qual a resposta ('um triangulo') pode ser gerada. Portanto, aparentemente a cognição central opera, ao menos em parte, através da cooptação de recursos do sistema visual para gerar representações visuais, que podem ser úteis para solucionar uma variedade de tarefas cognitivo-espaciais[17]"
>
> (Carruters, 2001)

O exemplo acima remete a uma utilização de funções imagéticas oriundas do córtex visual (Kosslyn, 1994), mas o esquema utilizado pode ser igualmente eficiente se aplicado à utilização da linguagem em processos cognitivos.

2.1.2.2 – A linguagem como instrumento de desenvolvimento diacrônico.

Além disto, a observação dos fatos aponta para insistentes resultados que parecem validar versões menos radicais da "Teoria Fraca da Linguagem", como por exemplo, o fato de que a faculdade da linguagem parece ser essencial para que os seres humanos possam efetuar certos tipos de processos cognitivos.

Aparentemente, a linguagem é o canal através do qual nós adquirimos e absorvemos muitos de nossas crenças e conceitos, e em muitos destes casos, estes conceitos e crenças dificilmente poderiam ter sido obtidos de outra forma, como por exemplo, o conhecimento cientifico e histórico acumulado, que é transmitido através da linguagem de geração em geração.

[17] "Note, too, that hardly anyone is likely to maintain that visual imagery is a mere epiphenomenon of central cognitive reasoning processes, playing no real role in those processes in its own right. On the contrary, it seems likely that there are many tasks which we cannot easily solve without deploying a visual (or other) image. For example, suppose you are asked (orally) to describe the shape which is enclosed within the capital letter 'A'. It seems entirely plausible that success in this task should require the generation of a visual image of that letter, from which the answer ('a triangle') can then be read off. So it appears that central cognition operates, in part, by co-opting the resources of the visual system to generate visual representations, which can be of use in solving a variety of spatial-reasoning tasks." (Minha Tradução)

Outro aspecto interessante deste relacionamento encontra-se na questão das falhas e atrasos do desenvolvimento cognitivo de crianças com deficiências no desenvolvimento lingüístico, é considerado ponto pacífico o fato de que o desenvolvimento cognitivo e lingüístico das crianças possuem algum grau de simultaneidade, desenvolvendo-se eqüitativamente.

> "Se a linguagem na criança é avançada, então igualmente avançadas serão suas habilidades em um grupo de tarefas; e se a linguagem na criança é atrasada, então igualmente atrasadas serão sua capacidades cognitivas.[18]"
>
> (Carruters, 2001)

Um dos exemplos mais famosos é o do "menino lobo", criado em condições selvagens e incapaz de recuperar certas habilidades cognitivas que deveriam ter sido desenvolvidas anteriormente, junto com o desenvolvimento da linguagem ausente.

Esta constatação reforça a tese de uma participação mais intensa da linguagem nos processos cognitivos, mas esbarra em uma segunda constatação empírica que nos conduz à conclusão contrária, de que os processos cognitivos possuam algum grau de independência em relação à linguagem.

Este segundo fato é observável nos relatos clínicos sobre pacientes com afasia, danos em áreas específicas do cérebro onde áreas responsáveis pela linguagem são atingidas, mas diversas capacidades cognitivas resistem a esta perda.

Aparentemente, a relação entre linguagem e pensamento parece diferir em natureza e intensidade quando se trata de crianças ou adultos.

Se considerarmos a combinação destes dois fatos, podemos nos encaminhar para uma possibilidade híbrida de resposta na qual a participação efetiva da

[18] "If children's language is advanced, then so will be their abilities across a range of tasks; and if children's language is delayed, then so will be their cognitive capacities." (Minha tradução)

linguagem nos processos cognitivos varie de acordo com a fase de desenvolvimento do ser humano.

> "Outra forma de ver esta questão é considerar que esta função cognitiva da linguagem possui significância apenas durante o período de desenvolvimento - ou diacrônico - ao invés de sincrônico. Nada é dito a respeito do papel da linguagem nos processos cognitivos de adultos, depois que um determinado conjunto de crenças e conceitos já foram adquiridos.[19]"
>
> (Carruters, 2001)

Uma outra possível razão para a radicalização das teorias fracas da linguagem é a radicalização oposta proposta por alguns ramos da filosofia e das ciências sociais que afirmam que a linguagem é responsável por toda a atividade cognitiva, o que soa excessivamente forte e pouco plausível.

> "Um movimento crucial de ajuste é a percepção de que a concepção cognitiva da linguagem pode ser formulada com diversos níveis de intensidade, e que cada versão deve ter seus méritos analisados separadamente.[20]"
>
> (Carruters, 2001)

Abordagens interessantes surgem da perspectiva de se construir teorias onde o papel da linguagem na atividade cognitiva não seja exagerado, nem para um lado, nem para o outro.

---

[19] "Another way of putting the point is that this proposed cognitive function of language is purely developmental - or diachronic - rather than synchronic. Nothing is said about the role of language in the cognition of adults, once a normal set of beliefs and concepts has been acquired."

[20] "A crucial liberalizing move, therefore, is to realize that the cognitive conception of language can come in many different strengths, each one of which needs to be considered separately on its own merits." (Minha tradução)

2.1.2.3 - A linguagem como modeladora do aparato cognitivo.

Uma versão mais forte e mais controversa do papel da linguagem no pensamento vem sendo proposta e defendida por alguns pesquisadores nas ultimas décadas.

Estes pesquisadores defendem a hipótese de que a linguagem é mais do que uma ferramenta de externalização dos pensamentos e também é mais do que uma forma de "carregar" a mente com um conjunto de informações necessárias a vida individual e em sociedade.

Nesta hipótese a linguagem possuiria a capacidade de modelar a estrutura do aparato cognitivo de acordo com as suas características próprias, desta forma, seres humanos criados em diferentes culturas, que utilizem diferentes linguagens, possuiriam percepções e capacidades diversas, análogas as características das linguagens nas quais foram desenvolvidas.

> "Por exemplo, a aquisição do idioma Yucatec - no qual os plurais são raramente marcados e muitos mais substantivos são tratados gramaticalmente como termos que designam substancias como água ou lama - conduz os indivíduos a perceberem as similaridades entre objetos com base em suas composições materiais em detrimento de seus formatos.
>
> E crianças criadas falando coreano - no qual os verbos possuem poucas formas de flexão e combinações massivas de substantivos são permitidos no discurso informal – o que parece conduzir as crianças a possuírem uma menor capacidade para tarefas de categorização, mas uma maior capacidade para outros tipos de tarefas onde as habilidades requeridas espelham a estrutura da língua.[21]"

[21] "For example, acquisition of Yucatec (as opposed to English) – in which plurals are rarely marked and many more nouns are treated grammatically as substance-terms like 'mud' and 'water' – leads subjects to see similarities amongst objects on the basis of material composition rather than shape. And children brought up speaking Korean (as opposed to English) – in which verbs are highly inflected and massive noun ellipsis is permissible in informal speech – leads children to be much weaker at categorization tasks, but much better at means–ends tasks such as using a rake to pull a distant object towards them." (Minha Tradução)

(Carruters, 2001)

Outro exemplo bastante conhecido para a teoria da linguagem como moduladora do aparato cognitivo é o fato conhecido dos esquimós possuírem diferentes nomenclaturas para diversas matizes de branco e serem conseqüentemente capazes de reconhecer estas variações da cor, invisíveis para indivíduos cujos aparatos cognitivos não tenham, em algum momento de seu desenvolvimento, sido esculpidos para este fim pela linguagem na qual se encontraram imersos.

No entanto, apesar de demonstrar uma participação bastante ativa da linguagem no pensamento, esta hipótese ainda se localiza entre as teorias "fracas" da linguagem pelo fato de defender o caráter diacrônico da influencia da linguagem no pensamento, ou seja, a linguagem possui um papel fundamental na modulação do aparato cognitivo mas uma vez modulado a linguagem retorna a seu papel meramente comunicativo.

2.1.2.4 - A linguagem como instrumento de apoio do aparato cognitivo.

Uma outra visão possível, que também permitirá a existência de um subconjunto de leituras mais fracas ou fortes é a hipótese de que a linguagem seja um instrumento de apoio a determinados processos cognitivos.

Uma linha possível de investigação empírica desta idéia é a observação dos solilóquios (diálogos de si consigo) executados por crianças quando precisam executar uma tarefa mais difícil, que exija mais concentração, o fato observado é de que as crianças parecem intensificar o uso da verbalização de acordo com o nível de dificuldade da tarefa executada.

Em sua forma mais fraca, a teoria da linguagem como elemento de apoio à cognição não se difere muito do que já foi descrito anteriormente, ou seja, a linguagem seria um meio de impulsionar as capacidades do aparato cognitivo e uma vez que este impulso fosse tomado tornaria-se dispensável, retornando a seu papel comunicativo.

Em sua forma mais forte, esta hipótese pode ser vista como uma afirmação de que a linguagem faz parte do processo cognitivo provendo auxilio em todas as etapas do desenvolvimento e na solução de um grande número de

diferentes tarefas.

Uma posição intermediaria possível seria a visão da linguagem como uma forma de aumentar as capacidades cognitivas humanas de forma constante mas operando como um módulo à parte que poderia ser invocado quando necessário pelo aparato cognitivo central.

Um exemplo desta visão seria a possibilidade de realizar cálculos aritméticos em um pedaço de papel utilizando o suporte da linguagem como uma forma de aumentar minha memória e permitindo que sejam realizadas operações que seriam impossíveis sem algum tipo de extensão das capacidades humanas nativas, neste caso, a linguagem possuiria a capacidade e a função de aumentar nossos capacidades cognitivas e não apenas uma mera função comunicativa.

> "Para expressar uma idéia é possível descarregar as responsabilidades sobre a persistência dos objetos na memória, concedendo ao intelecto um objeto que estará armazenado em outro lugar mas que mantêm-se disponível para reflexões futuras.
>
> Ao executar cálculos aritméticos em um pedaço de papel, nos tornamos aptos a realizar tarefas computacionais que sem este artifício estaria alem de nossas capacidades (e das capacidades de nossa memória). De uma forma semelhante, podemos considerar a possibilidade de que o diálogo de si consigo sirva para estender as capacidades da memória, uma vez que atualmente encontra-se bem estabelecida a tese de que as capacidades de memória humanas podem ser grandemente fortalecidas por associações. O dialogo de si consigo pode portanto ajudar na solução de atividades cognitivas complexas.[22]"
>
> (Carruters, 2001)

[22] "Thus by writing an idea down, for example, I can off-load the demands on memory, presenting myself with an object of further leisured reflection; and by performing arithmetic calculations on a piece of paper, I may be able to handle computational tasks which would otherwise be too much for me (and my short-term memory). In similar fashion, it may be that inner speech serves to enhance memory, since it is now well-

2.1.3- Teorias "fortes" da linguagem.

As teorias fortes da linguagem irão variar em forma e em intensidade mas permanecem possuindo em comum a crença de que a linguagem possui um papel fundamental nas atividades cognitivas.

2.1.3.1 – A linguagem como meio obrigatório para o pensamento.

Uma versão particularmente forte para o relacionamento entre linguagem e pensamento é proposta por alguns ramos da filosofia que acreditam que a linguagem seja uma condição obrigatória para a ocorrência de qualquer tipo de ação cognitiva.

> "O problema, é que todos estes argumentos dependem de uma forma ou de outra, de uma concepção anti realista da mente - requerendo, entre outras coisas, que a partir do momento que não podemos perceber nenhum pensamento na ausência de um comportamento lingüístico, estes pensamentos não podem sequer existir na ausência deste comportamento.[23]"
>
> (Carruters, 2001)

Esta hipótese, considerando como uma só diversas sub variações, se baseia na idéia de que só existe aquilo que pode ser percebido, e que não podemos

established that the powers of human memory systems can be greatly extended by association. Inner speech may thus facilitate complex trains of reasoning." (Minha Tradução)

[23] "But these arguments all depend, in one way or another, upon an anti-realist conception of the mind – claiming, for instance, that since we cannot interpret anyone as entertaining any given fine-grained thought in the absence of linguistic behavior, such thoughts cannot even exist in the absence of such behavior" (Minha Tradução)

perceber nada que não seja expresso através da linguagem, e que, portanto, só existiriam pensamentos exprimíveis pela linguagem e fundados na linguagem.

Em geral, os partidários desta hipótese negam a possibilidade de que animais não humanos possam possuir pensamentos, uma vez que não possuem a capacidade de utilizar convenientemente a linguagem, esta conclusão entra em conflito não apenas com o senso comum mas também com o que podemos descobrir empiricamente a respeito dos processos cognitivos dos animais o que nos fornece fortes evidencias de que esta versão exageradamente forte não possua fundamentos aceitáveis na realidade.

2.1.3.2 – A máquina virtual seqüencial.

A idéia da máquina virtual proposicional é de que todos os pensamentos conceituais de que é capaz o ser humano sejam desenvolvidos através da linguagem, mas que a linguagem, por sua vez não poderia existir sem uma base não lingüística pré-existente.

> "Dennett oferece uma das mais claras versões deste ponto de vista. Ele argumenta que as capacidades cognitivas humanas foram radicalmente transformadas em conseqüência do surgimento da linguagem natural, permitindo que a mente fosse colonizada por memes (idéias ou conceitos que são transmitidos, retidos e selecionados de uma maneira supostamente análoga aos genes; Ver Dawkins, 1976). De acordo com esta visão, a mente anterior ao surgimento da linguagem consistia em um conjunto de processos conexionistas que conferiam aos antigos hominídeos algum grau de flexibilidade e inteligência, mas que eram extremamente limitados em suas capacidades computacionais. O surgimento da linguagem significa que uma

> nova arquitetura cognitiva seqüencial e estruturada poderia ser acrescentada ao sistema.[24]"
>
> (Carruters, 2001)

A idéia é a presença de um processamento de alto nível que se baseia inteiramente em processamento lingüístico, segundo DENNETT os animais seriam capazes de efetuar pensamentos pré-linguisticos por possuírem uma versão da maquina conexionista que sustenta a maquina virtual seqüencial da linguagem.

Esta visão parece extremamente plausível se levarmos em conta a visão do senso comum a respeito dos processos utilizados pelos seres humanos para resolução de diversos tipos de problemas. Em geral, não utilizamos uma solução não lingüística para resolver problemas propostos através da linguagem como por exemplo um calculo ou a analise de um texto. Em contrapartida não parecemos utilizar soluções baseadas na linguagem quando construímos soluções para problemas propostos de forma não lingüística, como por exemplo o reconhecimento visual de padrões.

Esta dupla forma de resolução de problemas com naturezas distintas reforça a tese Dennettiana de que possuímos dois aparatos distintos, um conexionista e real, compartilhado com outros animais e o outro virtual e seqüencial, exclusivo dos seres humanos.

[24] "Perhaps Dennett (1991) provides one of the clearest exponents of this view. He argues that human cognitive powers were utterly transformed following the appearance of natural language, as the mind became colonized by memes (ideas, or concepts, which are transmitted, retained and selected in a manner supposedly analogous to genes; see Dawkins, 1976). Prior to the evolution of language, on this picture, the mind was a bundle of distributed connectionist processors – which conferred on early hominids some degree of flexibility and intelligence, but which were quite limited in their computational powers. The arrival of language then meant that a whole new – serial and compositionally structured – cognitive architecture could be programmed into the system." (Minha Tradução)

2.2 – Considerações sobre o relacionamento entre linguagem e pensamento.

Do ponto de vista do presente trabalho, todas as formas vistas acima, da mais forte até a mais fraca confirmam de certa forma a intuição original de que exista um elemento estrutural em comum entre as atividades cognitivas e a linguagem.

Mesmo na versão mais fraca na qual a linguagem é considerada apenas um meio para tornar comuns (comunicar) os pensamentos, é preciso que exista alguma coisa em comum entre o representante e o representado, que confira ao representante a capacidade de representar.

Portanto, proponho que, independente do posicionamento adotado em relação a natureza do relacionamento entre linguagem e pensamento, utilizemos o principio de que a análise da linguagem pode nos revelar dados sobre a natureza de nossos processos cognitivos.

2.2.1 – Linguagem natural e linguagens especialistas.

Proponho também assumir as hipóteses, oriundas da teoria dos conjuntos de que seja possível mensurar a igualdade entre conjuntos através de sua extensão.

> "Uma possível relação entre conjuntos, mais elementar do que a de pertinência, é a de **igualdade**. A igualdade de dois conjuntos A e B é universalmente denotada pelo familiar símbolo:
>
> a = b
>
> A mais básica propriedade da pertinência é sua relação com a igualdade, e que pode ser assim formulada.

> Axioma da Extensão – Dois conjuntos são iguais se e somente se eles tem os mesmos elementos."
>
> (Halmos, 2001)

A igualdade entre conjuntos seria mensurável, portanto, através da comparação entre a quantidade e a natureza de seus componentes, desta forma, me parece que o conjunto das capacidades cognitivas possa ser comparado, por extensão, com o conjunto dos referentes que podem ser expressos pela linguagem e que através desta comparação seja possível lançar a hipótese de que o conjunto dos referentes que podem ser expressos pela linguagem não corresponde em extensão ao conjunto da totalidade dos processos cognitivos mas talvez corresponda a um de seus subconjuntos.

> "Axioma da especificação - Para todo conjunto A e toda condição S(x) corresponde um conjunto B cujos elementos são exatamente aqueles elementos x de A para os quais S(x) é válida."
>
> (Halmos, 2001)

Proponho também a idéia de que para subconjuntos específicos do conjunto dos processos cognitivos correspondam subconjuntos específicos do conjunto das linguagens, e que através do estudo de cada subconjunto especifico das linguagens possamos inferir fatos sobre os subconjuntos específicos dos processos cognitivos.

Chego portanto, ao ponto principal deste trabalho, a idéia de que possamos, através da analise da história das linguagens de programação e do recorte de alguns tipos escolhidos de linguagem e modelagem de sistemas, lançar luz sobre parte dos processos cognitivos correspondentes.

III - História e características das linguagens de programação orientadas a objetos.

3.1 - Considerações sobre computações.

Segundo o filósofo DAVID CHALMERS, podemos definir computação como uma relação de identidade estrutural entre uma descrição formal de uma ação ou seqüência de ações (o programa) e a estrutura causal interna de um objeto físico que possui a capacidade de manipular sua estrutura de forma a seguir as instruções para implementar a estrutura formal descrita.

> "...Compreender o significado da computação requer a analise do conceito de implementação, o nexo entre computações abstratas e sistemas físicos concretos. Proponho uma analise, baseada na idéia de que um sistema implementa uma computação se a estrutura causal do sistema espelha a estrutura formal da computação...[25]"
> (Chalmers, 1999)

Esta receita pode adequar-se corretamente a computações muito simples, como uma gangorra que alterna sistematicamente seus lados variando com poucas opções o valor de uma única variável, ou muito complexas como um modelo de simulação de vôo que envolve inúmeras combinações de estados.

Em ambos os casos, parece ser possível isolar a descrição da execução desejada de sua implementação, e a maior parte das descrições formais parece poder ser implementada em qualquer artefato que possua as relações causais corretas, independente de sua natureza material.

A medida que o conjunto de regras descrito começa a se tornar mais complexo, torna-se menor o número de estruturas físicas capazes de implementa-lo.

[25] "...Justifying the role of computation requires analysis of implementation, the nexus between abstract computations and concrete physical systems. I give such an analysis, based on the idea that a system implements a computation if the causal structure of the system mirrors the formal structure of the computation... " (Minha Tradução)

3.2 - Primórdios da Computação

> "Um subproduto necessário da existência humana é a informação. Todas as possíveis facetas de nossas vidas produzem dados, e algumas vezes nós desejamos guardar estes dados para usos futuros.[26]"
> (Poage, 2002)

Após a peste negra que dizimou um terço da população européia entre 1347 e 1351 (Burke,1984) a concentração de riquezas se alterou de forma drástica, Muitos haviam morrido e os remanescentes, estimulados pelo duplo jubilo de terem sobrevivido e de terem herdado os bens dos que pereceram possuíam o capital necessário para uma serie de novos negócios e investimentos. Quando a prensa de GUTEMBERG foi inventada, no século XV, os clássicos gregos e romanos que haviam sobrevivido ao tempo e a idade média tornaram-se subitamente disponíveis a uma quantidade maior de leitores, e através desta popularização os europeus foram expostos a obras que relatavam e detalhavam vários tipos de autômatos pneumáticos e Hidráulicos (WOODCROFT, 1851). Esta exposição, somada a fartura de recursos e ao efeito classicista do fim da idade media, gerou uma onda de fascinação com autômatos que automatizavam, de alguma forma operações que antes eram privilegio dos seres vivos.

Aliados a estes fatores encontravam-se a crescente influencia do mecanicismo na filosofia e nas ciências, naquela época reunidas sob o rotulo de "Filosofia natural", manifestado através das obras de GALILEU, KEPLER e NEWTON, que pretendiam despir a realidade de atributos que não pudessem ser modelados através de um sistema suficientemente poderoso, como a álgebra, por exemplo.

Em 1642, no tempo das maquinas calculadoras de PASCAL e LEIBNITZ, a descrição da ação desejada (o calculo) era implementada diretamente na máquina, através do ajuste das rodas dentadas que a compunham. Já estava viva, no entanto, a idéia de que um determinado artefato

[26] "A necessary byproduct of human existence is information. Every possible facet of our lives produces data, and sometimes we wish to store this data for future use" (Minha Tradução)

físico pudesse implementar um determinado conjunto de ações previamente descrito.

> "Quando, há alguns anos atrás, vi pela primeira vez um instrumento que, quando transportado, gravava automaticamente o número de passos dados por um pedestre, ocorreu-me de imediato que toda a aritmética podia ser sujeita a uma maquinaria similar, não só para que a contagem, mas também a adição e a subtração, a multiplicação e divisão, pudessem, através de uma máquina adequada e eficaz, ser fácil e rapidamente feitas e com resultados certos."
> (Leibnitz, 1998)

Quase cem anos depois, em 1725, o engenheiro francês BALISSE BOUCHON, filho de um construtor de órgãos automáticos, decidiu combinar idéias oriundas dos autômatos mecânicos e uma utilidade pratica representada na forma de teares de seda, muito comuns na região de Lyon (Baird, 2001). A forma típica de "programar" um órgão automático era través de pinos acoplados sobre um cilindro que gira e que abrem válvulas de ar ao passarem e a forma tradicional de construir um cilindro era enrolar um pedaço de papel em torno do cilindro e marcar as localizações corretas dos pinos, que eram depois, fixados por um marceneiro.

BOUCHON percebeu que o papel marcado era uma forma de armazenar informação e transmiti-la para uma maquina que poderia implementa-la seguindo uma seqüência de ordens pré-determindas (algoritmo) e as utilizou para construir um tear, que podia tecer desenhos de seda, de acordo com instruções cifradas em uma folha giratória de papel perfurado, onde somente trabalhavam agulhas coincidentes com os furos (Poage, 2002).

A grande inovação da máquina de BOUCHON é a separação da descrição da ação do artefato implementador, o que permite a percepção de que uma mesma maquina poderia implementar mais de uma computação (conjunto de regras) diferentes, desta forma, não era mais necessário criar uma maquina para cada tarefa, bastava descrever as tarefas como um conjunto de ações que pudesse ser refletido pela estrutura causal da maquina. É interessante notar que os princípios básicos do tear de JACQARD (Descendente do tear de

BOUCHON), assemelha-se bastante ao conceito da máquina de TURING, concebida cerca de 150 anos mais tarde, Os cartões perfurados foram largamente utilizados como método de persistência de dados até 1980, quando novos meios de armazenamento foram introduzidos.

3.3 – O desencontro estrutural entre a descrição e a ação e seu reencontro através da introdução de camadas intermediárias de tradução.

O fator mais interessante do ponto de vista do escopo deste trabalho é o fato de que a forma de armazenamento das instruções e a forma como ela se relaciona com o artefato implementador ira determinar as possibilidades estruturais desta lista de ações (descrição formal).

Os frutos das subseqüentes revoluções cientificas (eletricidade, eletromagnetismo, semicondutores entre outros) levaram a descobertas que permitiram a criação de maquinas com capacidade para espelhar causalmente descrições formais cada vez mais sofisticadas, acompanhando esta evolução do hardware, as descrições formais da ação ou conjunto de ações (programa) podiam descrever a estrutura lógica do objeto ou conjunto de objetos (MiniMundo[27]) que se propunha a mimetizar de forma mais fiel, completa e consistente.

Desta forma, podemos compreender as linguagens de programação como sistemas simbólicos através dos quais pode-se gerar uma descrição formal dos aspectos lógicos de uma determinada estrutura do "mundo real". Para que esta descrição possa ser considerada uma linguagem de programação, deve existir algum artefato capaz de executar (através de algum tipo de relação causal) estas instruções.

> "Todas as linguagens de programação, assim como as linguagens naturais, podem ser vistas como modelos do mundo.[28]"
> (Personen, 2001)

[27] Utilizarei a expressão "MiniMundo" para me referir ao domínio especifico do problema a ser resolvido .
[28] "All programming languages, like natural languages, can be seen as models of the world" (M. T.)

Gradualmente, a estrutura da descrição foi se afastando da estrutura do artefato implementador em direção ao que convencionamos chamar de linguagens de alto nível (mais próximas do homem do que da máquina).

Esta evolução se deu através da criação de programas intermediários que possuem como objetivo intermediar a estrutura formal principal (programa) com a estrutural causal da maquina.

A estrutura formal das linguagens de desenvolvimento se afastaram da estrutura causal da maquina sem que fosse realizada nenhuma alteração na estrutura causal da maquina. Elas passaram a se comunicar através de programas intermediários, compiladores, interpretadores, Maquinas Virtuais ou sistemas operacionais. Cada um destes programas possui a função de nos permitir uma interface com a maquina através de estruturas formais diferentes das estruturas causais da maquina. Estes intermediários mapeiam uma estrutura na outra.

A medida que os programas intermediários (responsáveis pela capacidade de mapeamento) vão se sofisticando, mais completas e naturais se tornam as descrições dos minimundos que serão implementadas com sucesso graças ao esforço de tradução dos intermediários.

> "Outra forma de ver como programas modelam o mundo é percebendo que originalmente programas são escritos em linguagem de maquina, o que logo se mostrou ser impraticável para humanos. Surgem então, linguagens humanas de alto nível com seus respectivos compiladores. Esta estratégia faz com que a quantidade de linguagem de maquina referente a poucas instruções de alto nível aumente cada vez mais. Até certo ponto, as instruções de alto nível são atalhos que ainda apontam para a subjacente estrutura da maquina, mas a medida que as linguagens evoluem, o sistema descrito passa a apontar não mais para a maquina, mas para o mundo do lado de fora do computador. Objetos não são atalhos para instruções de máquina - eles são simulações computacionais de objetos do mundo real.[29]"

[29] "Another way of seeing how programs model the world is noticing that originally programs were written in a machine language, which soon proved to be inhumane. Higher-level human readable languages with compilers were invented. This way more and more machine language instruction could be achieved with fewer and fewer higher-level language instructions. The higher-level instructions were just shortcuts that still

(Personen, 2001)

3.4 – Linguagens de Programação.

As camadas intermediárias (compiladores e interpretadores) permitem a possibilidade de criar linguagens com estruturas muito diferentes entre si, alem de serem diferentes em relação à estrutura causal da máquina.

> "Existem hoje mais de 2400 linguagens de programação."
> (Richards, 2001)

Porque este número é tão expressivo e como hierarquizar estas linguagens, através de suas semelhanças de classes e qualitativamente?

A criação de linguagens de programação parece obedecer às estruturas mentais requeridas para domínios específicos do conhecimento[30], desta forma, cada grupo de linguagens procura atender melhor a um determinado objetivo e cada linguagem dentro do grupo procura apresentar melhorias em relação a seus pares, na execução de suas tarefas.

> "Para se implementar um algoritmo em um computador, é necessário descrevê-lo de uma forma que o computador esteja apto a executá-lo.
>
> Essa descrição é feita por intermédio de uma "linguagem de programação". O próprio conjunto de instruções de um processador pode ser entendido como uma "linguagem de programação". Entretanto, essa linguagem normalmente não é a mais adequada para a descrição de um programa, uma vez que os algoritmos necessários podem ser sofisticados, e essa linguagem primitiva, também chamada de "linguagem de máquina" não é nem um pouco amigável ao programador, demandando um esforço muito grande na elaboração de programas mais complexos. Sendo assim, foram desenvolvidas, ao

---

pointed to the underlying machine architecture, but as the languages evolved, the language constructs no longer pointed to the machine, but instead to the world outside the computer. Objects are no shortcuts of machines instructions – they are computer simulations of real objects." (M.T.)

[30] Neste ponto me utilizarei novamente do axioma da especificação (capítulo III) para supor a geração de subconjuntos especializados das linguagens de programação e de seus equivalentes cognitivos.

> longo da história da computação, diversas "linguagens de programação", cada qual, a seu tempo, introduzindo facilidades e recursos que foram tornando a tarefa de programar mais fácil e menos suscetível a erros. "
> (Gudwin, 1997)

A divisão de famílias que se segue é parte da apresentada em (Gudwin, 1997)[31] e obviamente não esgota o vastíssimo universo das linguagens de programação, mas ilustra o caminho percorrido até o surgimento das linguagens OO.

Segundo GUDWIN, as linguagens de programação podem ser divididas dentro da seguinte Taxonomia, linguagens de baixo Nível, linguagens não estruturadas, linguagens procedurais, linguagens funcionalistas e linguagens orientadas a objetos.

3.4.1 · Linguagens de Baixo Nível

Linguagens de baixo nível são linguagens cujas instruções correspondem quase que diretamente ao código de máquina que será enviado ao processador para execução.

Na verdade, existem tantas linguagens de baixo nível quantos são os conjuntos de instruções dos diferentes processadores. Essas linguagens são conhecidas de uma maneira unificada como "Linguagem Assembly", sendo que na verdade deve existir uma linguagem Assembly para cada processador. Sendo assim, deve haver um Assembly 8086, um Assembly 68000, um Assembly 80386, um Assembly Pentium e assim por diante.

[31] Cada grupo possui apenas uma descrição muito breve de suas características seguida de alguns exemplos. O principal objetivo deste trabalho é o exame detalhado dos conceitos da programação Orientada a Objetos e sua recorrência na história do pensamento. Os demais paradigmas são utilizados como auxiliares históricos para a compreensão destes conceitos, não sendo cabível, dentro do escopo deste trabalho uma exposição mais detalhada sobre os componentes da cena de fundo dos conceitos alvo.

3.4.2 · Linguagens Não-Estruturadas.

Linguagens não-estruturadas são linguagens mais sofisticadas que as linguagens de baixo nível. São linguagens mais flexíveis, pois seus comandos não estão tão vinculados ao processador e sistema utilizados, tornando seu uso possível em diferentes plataformas.

Do mesmo modo, a semântica de seus termos torna-se mais genérica, estando desassociada do conjunto de instruções que irão efetivamente implementá-la durante a execução do programa. Isso permite que operações mais sofisticadas sejam emuladas por sequências de instruções mais simples em linguagem de máquina.

As linguagens não estruturadas corresponderam a um grande salto qualitativo em termos de programação, quando comparadas com as linguagens de baixo nível. Entretanto, tornaram-se gradualmente obsoletas, com a introdução de linguagens estruturadas mais sofisticadas, tais como as linguagens procedurais, as linguagens funcionais e as linguagens orientadas a objetos.

3.4.2.1 – Cobol.

O COBOL (COmmon Business Oriented Language), foi designado especificamente para a construção de aplicações comerciais, tais como folhas de pagamento, inventário de produtos, contabilidade, operando tipicamente em grandes volumes de dados. Sua estrutura não é adequada, portanto, para problemas científicos, onde cálculos complexos são necessários.

Encontra-se disponível em várias plataformas, desde máquinas PC até mainframes. O mesmo programa COBOL pode ser compilado e executar em uma grande variedade de máquinas, com apenas pequenas mudanças no código.

A linguagem COBOL foi desenvolvida em 1959 por um grupo chamado Comitê CODASYL (Conference on Data Systems Languages), que inclui representantes de origem acadêmica, bem como fabricantes de computadores. O objetivo desse grupo foi o de desenvolver uma linguagem padrão para o desenvolvimento de programas comerciais, de tal forma que todos os fabricantes

pudessem oferecer compiladores para tal linguagem. A conferência foi patrocinada pelo Departamento de Defesa dos Estados Unidos (DOD).

Como resultado do CODASYL, o primeiro compilador COBOL veio ao mercado em 1960. Entretanto, devido à diversidade entre os compiladores COBOL que surgiram, decidiu-se elaborar uma norma para regulamentar a linguagem. Em 1968, a primeira versão do ANSI-COBOL foi desenvolvida e aprovada.

A maioria dos grandes fabricantes de computadores e distribuidores de software começaram a distribuir compiladores utilizando o ANSI-COBOL. Em 1974, uma segunda versão foi elaborada e em 1985, uma terceira, conhecida como COBOL-85.

Esforços foram empregados na elaboração de uma versão (COBOL 2002) do COBOL que possui-se algumas características da orientação a objeto e programação visual.

3.4.2.2 – Basic.

A linguagem BASIC (Beginners All-purpose Symbolic Instruction Code) foi desenvolvida por Thomas E. Kurtz e John G. Kemeny, membros do departamento de matemática de Dartmouth, em 1963.

O BASIC foi desenvolvido com o intuito de ser uma linguagem fácil e interativa, de modo a possibilitar seu uso por cientistas da computação, que rapidamente pudessem criar programas e executá-los.

Ela se tornou popular nos primeiros microcomputadores. Uma de suas características mais marcantes é o fato de ser uma linguagem interativa, ou seja, dá facilidades ao programador para ir programando ao mesmo tempo em que executa.

É uma linguagem freqüentemente interpretada, havendo entretanto, compiladores para ela.

Não é uma linguagem padrão entre computadores, apesar de existir uma norma ANSI que define a linguagem, existindo mais de 80 diferentes variantes, com diferenças às vezes significativas entre elas. Algumas implementações mais recentes incorporam recursos de programação visual e orientação a objeto (Visual Basic).

3.4.3 – Linguagens Estruturadas

Essas linguagens se diferenciam das linguagens não estruturadas, pela existência de estruturas de controle, que entre outras coisas promovem o teste de condições (if-then-else), controlam a repetição de blocos de código (for, while, do), fazem a seleção de alternativas (switch, case), e dividem o código do programa em módulos chamados de funções ou procedimentos.

3.4.3.1 - Linguagens Procedurais.

As linguagens procedurais são um dos sub-tipos das linguagens chamadas "estruturadas".

As linguagens procedurais são caracterizadas pela existência de algoritmos que determinam uma seqüência de chamadas de procedimentos, que constituem o programa. Essa característica é colocada em contraposição às linguagens funcionais, onde se descrevem expressões que caracterizam um certo tipo de conhecimento.

As linguagens procedurais mais comuns são o C, o Pascal e o Fortran.

Algumas linguagens mais sofisticadas, tais como o Ada, Modula-2 e Modula-3 foram desenvolvidas para corrigir deficiências e explorar novos contextos onde o C, Pascal e Fortran não eram totalmente adequados.

3.4.3.1.1 – C.

O desenvolvimento da linguagem C está intimamente conectado ao desenvolvimento do sistema operacional Unix. Durante o desenvolvimento do Unix, foi necessária a criação de uma linguagem de alto nível para a programação do sistema, uma vez que o esforço de programação em linguagem assembly era muito grande. Sendo assim, um grupo de pesquisadores do Bell Labs, incluindo Ken Thompson e Dennis Ritchie, desenvolveram uma linguagem de alto nível a qual batizaram de "linguagem B".

A linguagem B foi posteriormente substituída por outra que a sucedeu, a "linguagem C". Com o desenvolvimento de um compilador C portátil (por Steve

Johnson, do Bell Labs), o Unix pôde ser transportado facilmente para diversas plataformas, tornando tanto a linguagem C como o sistema operacional Unix um sucesso para diferentes plataformas computacionais.

O grande sucesso da linguagem C está intimamente ligado ao sucesso do sistema operacional Unix, sendo que quase todas as implementações do Unix (para diferentes arquiteturas) possuem um compilador C nativo.

Posteriormente, devido a esse sucesso, a linguagem C passou a ser utilizada também fora do ambiente Unix, como por exemplo em PC´s (DOS, Windows e OS/2), com a introdução dos compiladores da Borland e Microsoft.

Atualmente, a linguagem C é descrita pela norma ISO - ISO/IEC 9899:1990.

3.4.3.1.2 – Pascal

O Pascal é uma linguagem que foi originada no ALGOL[32], tendo sido desenvolvida por Niklaus Wirth para o CDC 6600, por volta de 1967-68, como uma ferramenta de instrução para programação elementar. Apesar de ter sido desenvolvida inicialmente apenas com propósitos educacionais, tornou-se posteriormente uma linguagem de propósitos gerais, tendo-se tornado inclusive o ancestral de diversas outras linguagens, tais como o Modula-2 e o Ada.

É uma linguagem que enfatiza bastante sua sintaxe, evitando artifícios encontrados em outras linguagens, tais como “casts”, para efetuar conversão de tipos, ou outros artifícios que poderiam tornar a linguagem mais flexível. Neste sentido, o Pascal foi perdendo lugar para o C, que tem uma estrutura semelhante, e ao mesmo tempo dá ao programador a flexibilidade necessária para resolver problemas onde a sintaxe rígida do Pascal não permite muita elaboração.

[32] ALGOL (ALGOrithmic Language) is one of several high level languages designed specifically for programming scientific computations. It started out in the late 1950's, first formalized in a report titled ALGOL 58, and then progressed through reports ALGOL 60, and ALGOL 68. It was designed by an international committee to be a universal language. Their original conference, which took place in Zurich, was one of the first formal attempts to address the issue of software portability. ALGOL's machine independence permitted the designers to be more creative, but it made implementation much more difficult. Although ALGOL never reached the level of commercial popularity of FORTRAN and COBOL, it is considered the most important language of its era in terms of its influence on later language development. ALGOL’s lexical and syntactic structures became so popular that virtually all languages designed since have been referred to as "ALGOL - like"; that is they have been hierarchical in structure with nesting of both environments and control structures.

3.4.3.1.3 – Fortran

O “IBM Mathematical FORmula TRANslating system”, ou FORTRAN, foi a primeira linguagem de alto nível (ou seja, superior ao assembly).

Desenvolvida por Backus, dentre outros, a linguagem foi introduzida em 1958, direcionada primeiramente para cálculos científicos, de tal forma que facilidades como manipulações de “strings” eram quase inexistentes, e a única estrutura de dados existente era o “array”. Entretanto, tratou-se de um grande salto qualitativo, quando comparado ao Assembly. Tornou-se uma linguagem de grande aceitação, sendo utilizada até hoje em aplicações numéricas.

A primeira norma regulamentando a linguagem veio entretanto somente em 1966, quando a “American National Standards Institute” (ANSI) publicou uma norma que unificou a linguagem, sendo referenciada como FORTRAN´66.

Depois disso, a linguagem passou por diversas revisões, de modo a incorporar idéias da programação estruturada. As mais notáveis revisões, foram o FORTRAN 77 e o FORTRAN 90, introduzidas em 1978 e 1991 , respectivamente.

O FORTRAN´90 incorpora diversas características das linguagens modernas de programação, dentre estas o uso de “arrays” dinâmicos, “records”, módulos e apontadores, derivação de tipos e “overload” de operadores, um melhor sistema de declarações e o uso de protótipos e modernas estruturas de controle (SELECT CASE, EXIT, ...).

O FORTRAN´90 é descrito pela norma internacional ISO/IEC 1539:1991 (E). O comitê X3J3 da ISO, que é dedicado exclusivamente ao desenvolvimento do FORTRAN, está, no momento, preparando uma nova versão da linguagem, que deverá incluir, dentre outros, suporte para processamento paralelo e programação orientada a objetos.

3.4.5 – Linguagens Funcionais.

Linguagens funcionais são linguagens que evidenciam um estilo de programação diferente das linguagens procedurais, chamado de programação funcional. A programação funcional enfatiza a avaliação de expressões, ao invés da execução de comandos. As expressões nessas linguagens são formadas utilizando-se funções para combinar valores básicos.

As linguagens funcionais mais conhecidas são o LISP e o Prolog. A linguagem Scheme também é frequentemente citada, por ser uma variante simplificada do LISP. Diversas outras linguagens funcionais são encontradas na literatura, por exemplo, ASpecT, Caml, Clean, Erlang, FP, Gofer, Haskell, Hope, Hugs, Id, IFP, J, Miranda, ML, NESL, OPAL,e Sisal.

3.4.5.1 – Prolog

O Prolog (Programação em Lógica) é uma linguagem de programação que visa implementar uma estrutura similar a da lógica de predicados de FREGE, estruturando o mundo em termos de funções/relações e objetos.

Introduzida em 1973 por Alain Colmerauer e seus associados na Universidade de Marseille, com o propósito inicial de traduzir linguagens naturais.

Em 1977, David Warren da Universidade de Edimburgo, implementou uma eficiente versão do Prolog, batizada de Prolog-10.

A partir daí, tornou-se uma escolha natural para a resolução de problemas que envolvem a representação simbólica de objetos e relações entre objetos. O fundamento básico por trás do Prolog é a noção de programação em lógica, onde o processo de computação pode ser visto como uma sequência lógica de inferências.

Esta noção desenvolve-se então de modo a resultar em um mecanismo automático de prova de teoremas. Atualmente, o Prolog é utilizado em diversas aplicações na área de computação simbólica, incluindo-se aí: bases de dados relacionais, sistemas especialistas, lógica matemática, prova automática de teoremas, resolução de problemas abstratos e geração de planos, processamento de linguagem natural, projeto de arquiteturas, logística, resolução

de equações simbólicas, construção de compiladores, análise bioquímica e projeto de fármacos.

Sua estrutura conceitual está intimamente ligada com a lógica matemática, o que a torna uma ferramenta indispensável para o estudo de lógica.

Versões diferentes do Prolog podem ser encontradas nas mais diversas plataformas, tanto na forma de compiladores como de interpretadores.

3.4.5.2 – LISP.

O LISP (LISt Processor) é uma linguagem de programação que foi desenvolvida utilizando-se idéias oriundas do estudo da inteligência artificial.

Sua constituição é tal que programas escritos em LISP tendem a ser um modelo que emula as habilidades cognitivas do ser humano. O elemento básico utilizado pelo LISP são os símbolos, eminentemente símbolos alfanuméricos.

A partir de 1984, o LISP começou a desenvolver-se em diversos dialetos, sem que um deles dominasse marcadamente os outros. Em um esforço conjunto de um seleto grupo de pesquisadores em linguagens de programação, representando diversas instituições, desenvolveu-se o Common Lisp, integrando com sucesso diversas características positivas das diferentes versões do Lisp que circulavam na época.

O Common Lisp é um dialeto do LISP, sucessor do MacLisp e influenciado marcadamente pelo Zetalisp e de certa forma também pelo Scheme e pelo Interlisp.

O Common Lisp é hoje utilizado como um padrão comercial para a linguagem Lisp, sendo suportado por um grande número de companhias.

Posteriormente, o Common Lisp foi enriquecido pela adição de funções genéricas, dando origem a uma variante sua com capacidade de programação orientada a objetos chamada CLOS (Common Lisp Object System).

3.4.6 – Observações gerais.

Dentre a miríade de características exibidas por estas poucas e exageradamente resumidas descrições de linguagens de programação, duas chamam especial atenção, a primeira é o fato de que praticamente todas as descrições terminam com o encaminhamento da linguagem para algum nível de proximidade do paradigma orientado a objetos.

É claro que isto pode ser apenas efeito de uma ilusão coletiva (onde ninguém percebe que o rei esta nu), não servindo de garantias de que determinada forma de programar seja melhor ou pior do que outras, mas, por outro lado, aponta para uma sobrevivência dos conceitos mais fortes, uma espécie de seleção natural das linguagens de programação.

Quando se utiliza o carregado conceito Darwiniano de Seleção Natural, surge a dúvida a respeito da preexistência dos conceitos da programação OO. Estes conceitos já existiam e disputavam espaço com os demais pontos de vista ou eles passam a existir a medida que os preexistentes se alteram?

Uma resposta tentadora é a de que estes conceitos já existissem, mas não eram ainda aplicáveis ao desenvolvimento de software devido a pouca complexidade das camadas tradutoras intermediárias (compiladores e interpretadores). É claro que ao migrarem para o “mundo” da engenharia de software estes conceitos sofreram alterações, mas a quantidade de características remanescentes não parece ser suficiente para que os consideremos uma nova espécie.

A outra observação que considero pertinente é a de que tradicionalmente, o desenvolvedor de software é instado a escolher entre uma forma de programar, uma forma de modelar e uma forma de armazenar (persistir) de dados. No paradigma Orientado a Objetos estas três atividades assumem um conjunto comum de conceitos, mantendo forte semelhança entre si. Apesar deste fato também não constituir um argumento definitivo a favor ou contra este Meta Modelo (de modelos, programas e dados), o fato é que, até agora, não havíamos vivido um momento do desenvolvimento de software onde podemos utilizar simultaneamente a UML (para modelar), uma das muitas linguagens de programação que suportam a implementação OO (Java, C++, SmallTalk etc...) e

uma das muitas camadas de persistência OO disponíveis como por exemplo o Prevayler[33]).

3.5 - Programação Orientada a Objetos.

O paradigma da programação orientada a objetos se solidifica no final dos anos setenta. Nos paradigmas de programação tradicionais, um programa consiste em um conjunto de variáveis e funções que utilizam estas variáveis como parâmetros de entrada e retornam novos valores, alterando estas variáveis. No paradigma Orientado a Objetos, programas consistem em objetos independentes e suas interações.

Os objetos incorporam as variáveis e as funções que as manipulam, funções que antes pertenciam ao programa como um todo e que passam então a pertencer ao domínio de um objeto específico.

Como resultado desta alteração, todo o comportamento de um objeto passa a ser encapsulado e manipulado internamente, a comunicação e interação entre objetos passa a ser realizada inteiramente através de mensagens, chamadas[34] aos métodos públicos de um objeto, que se encarregará internamente da implementação de suas ações.

Esta afirmativa reforça a tese dos defensores das linguagens OO de que estas representariam de forma mais fidedigna a estrutura do MiniMundo que pretendemos representar. Aparentemente, no mundo real, não existem variáveis ou métodos “soltos”, que não pertencem a nenhum objeto, e os objetos parecem, na maioria das vezes, possuir acesso exclusivo a suas variáveis.

Este ponto de vista é ratificado pela afirmativa abaixo, que expõe outro aspecto do realismo da Orientação a Objetos.

> "Uma grande diferença entre programas modelados com objetos e programas modelados de acordo com paradigmas tradicionais é o fato de que existe uma expectativa de que os primeiros consigam se assemelhar mais com o domínio do problema que esta sendo

[33] http://www.prevayler.org/wiki.jsp?topic=PrevalentManifesto

[34] Para que objetos troquem mensagens, é preciso que um objeto A execute uma chamada a um dos métodos públicos de outro objeto B.

> analisado: os componentes do programa são iguais aos componentes do mundo real.[35]"
> (Abadi e Cardelli 1996)

Em uma linguagem de programação tradicional, o domínio do problema é inicialmente analisado, sendo a responsabilidade de representa-lo em um modelo computacional transferida para o programador/analista.

Segundo os defensores do paradigma de programação OO, este pode possuir uma relação mais isomórfica com os componentes e seus comportamentos no mundo real, impedindo uma grande variação de implementações derivadas de analises pessoais que variam de pessoa para pessoa de acordo com o perfil de cada um. Uma vantagem derivada é o aumento da comunicabilidade do sistema entre programadores e usuários e entre programadores e outros programadores, devido ao fato de que, supostamente, os componentes do programa e os componentes do mundo real possuiriam grande semelhança.

Um exemplo destas características pode ser obtido se imaginarmos que um determinado programador deve modelar o comportamento de animais e plantas em uma floresta. Em uma linguagem procedural a forma mais natural de executar esta tarefa seria através da criação de variáveis para armazenar os estados dos animais e plantas e funções para manipulação destas variáveis. Seria função do programador impedir que a função rastejar seja passada para uma coruja (ou suas variáveis) ou que a função Voar Seja passada para um cachorro.

Um código como este tende a gerar um nível de complexidade crescente que pode inviabilizar o projeto ou sua manutenção, gerando o famoso "código spagetti", se uma alteração é solicitada, o programador deverá localizar o trecho de código onde as mudanças precisam ser feitas e estar certo, ou rezar para que esteja, de que as alterações realizadas não interfiram com outras partes do programa, além disto, este modelo tira pouco, ou nenhum proveito do fato de que diversos objetos da floresta compartilham características comuns. O modelo Orientado a Objetos faz uso das características comuns, por exemplo, todos os animais podem ser derivados de uma Superclasse Animais, que possuiria todos

[35] "A Great Difference Between programs designed with objects as compared with traditional paradigms, is that the programs are Said to resemble conceptually the actual problem domain that is being analyzed: the components in the program are equal to the components in the real world" (M.T.)

as características comuns compartilhadas por animais, desta forma, uma coruja herdaria as características comuns a todos os animais tornando desnecessária uma implementação redundante, em uma linguagem procedural, o programador necessita tradicionalmente realizar repetidamente operações de copiar e colar para **reutilizar** um determinado método. A herança é completada por comportamentos e atributos próprios, isto garante que todos os comportamentos da coruja estejam internalizados no objeto coruja, Desta maneira, torna-se pouco provável que alguém consiga fazer a coruja rastejar ou o cachorro voar.

3.5.1 – História da Programação Orientada a Objetos.

3.5.1.1 – Simula.

Nos primeiros tempos da atividade de resolver problemas com computadores, havia uma sensação de que cada nova linguagem (maquina, assembly, Fortran) representavam grandes avanços em relação as anteriores, e comparativamente, isto pode ser considerado um fato, embora nem todos concordassem com isto como podemos ver no artigo do Professor EDGESTER W. DYKSTRA (Dyskstra, 1983).

> "O fato brutal é que uma grande parte da história da computação pode ser escrita em termos de projetos milionários que falharam. Isto pode valer também para outras disciplinas, mas o que parece único na computação é que, na maioria dos casos, a futura falha já se mostrava obvia no momento em que o projeto foi concebido. Foi uma longa história de castelos no ar, um depois do outro, e o fenômeno é tão persistente que requer uma explicação. Como podemos explicar o persistente erro de julgamento das questões técnicas envolvidas?[36]"
> (Dyskstra, 1983)

[36] "The Brutal fact is that a large part of the history of computing can be written in terms of multi-million-dollar projects that failed. That may hold for other disciplines as well, but what seems unique to computing is that in most cases the future failure was already obvious at the moment the project was conceived. It was a long history of castles in the air, the one after the other, and the phenomenon is so persistent that it needs an explanation. How do we explain the persistent misjudgement of the technical issues involved?" (Minha Tradução)

Com a possibilidade técnica de descrever o mundo e criar modelagens de forma mais natural, surge na década de 60 em Oslo, SIMULA, considerada a primeira linguagem de programação orientada a objetos.

SIMULA é uma criação da equipe dos professores Ole-Johan Dahl e Kristen Nygaard (Dæhlen, 2001), do departamento de informática da universidade de Oslo assim como os conceitos fundamentais que subjazem a modelagem e o desenvolvimento de software orientado a objetos se inicia com o surgimento da linguagem SIMULA.

SIMULA I, surgida entre 1962 e 1965 e SIMULA 67 (1967) foram as duas primeiras linguagens orientadas a objetos.

A sintaxe da SIMULA é baseada na linguagem pré-existente ALGOL 60, e é fortemente padronizada o que permite grande portabilidade entre plataformas.

Durante os anos 60, linguagens de uso especifico entram em declínio, surgindo à necessidade de adaptar a SIMULA I para transforma-la em uma linguagem de uso geral.

Surge então SIMULA 67 que introduz a maioria dos conceitos chave da programação orientada a objetos: classes, objetos, subclasses (Conceito posteriormente conhecido como herança) entre outros.

Nygaard vinha trabalhando com pesquisa operacional desde 1950 e em 1960 percebeu e expressou a necessidade de formas acuradas de descrição do comportamento de sistemas complexos (sistemas que envolvem as estruturas causais de mais de um objeto alem de suas relações), “em 1961 à idéia evoluiu para o desenvolvimento de uma linguagem que poderia igualmente ser utilizada para realizar a descrição do sistema (para pessoas) e a prescrição do sistema (como um programa de computador gerado através de um compilador).” Neste ponto a experiência e o conhecimento de Dahl sobre compiladores foi essencial para tornar real o ideal de Nygaard.

O compilador do SIMULA I foi parcialmente financiado pelo projeto UNIVAC e foi concluído em janeiro de 1965. Simula I se tornou conhecido como uma linguagem de programação ideal para realizar simulações do mundo real (não computacional), mas rapidamente transcendeu a esfera da modelagem ao exibir a presença de diversas propriedades que faziam dela uma linguagem de programação de uso geral.

Com o acréscimo de diversas funções, entre elas o mecanismo de herança, SIMULA 67 é desenvolvido como uma ferramenta de desenvolvimento geral que também poderia ser especializada para diferentes domínios, inclusive simulações de sistemas.

Compiladores do SIMULA 67 começam a surgir para os principais hardwares existentes na época, UNIVAC, IBM, Control DATA, Burroughs, DEC e outros computadores existentes no inicio dos anos 70.

SIMULA 67 ainda é utilizado em diversas partes do mundo, mas sua principal contribuição ao mundo do desenvolvimento de software é ter introduzido uma das principais categorias de desenvolvimento, posteriormente rotulada como programação orientada a objetos, os conceitos do SIMULA foram importantes na discussão e desenvolvimento da idéia de dados abstratos agrupados (Structs) e na idéia de reunir, em uma só entidade computacional (Classe) dados e métodos, informações e comportamentos.

No final dos anos 70, Alan Kay[37] e seu grupo de pesquisadores no laboratório (PARC) da xerox utilizaram o SIMULA 67 para como uma plataforma para o desenvolvimento de uma nova linguagem chamada SMALLTALK que estendeu as capacidades e os conceitos da orientação a objetos através da inclusão de interfaces gráficas com o usuário e execução interativa do programa (substituindo o padrão batch em voga até então).

Na década de 1980, Bjarne Stroustrup inicia seu desenvolvimento da Linguagem C++ inserindo os principais conceitos da SIMULA na já existente linguagem C.

SIMULA inspirou também grande parte dos trabalhos desenvolvidos na área de construção de componentes, de re-usabilidade de código e a construção de bibliotecas de código reutilizável (Uma obsessão dos tempos modernos).

Em 1980 um caminho diferente da trilha proposta por SIMULA foi vislumbrado quando grandes recursos foram investidos no desenvolvimento das linguagens de programação ADA e PROLOG (ambas não orientadas a objetos), apesar disto, a orientação a objetos é hoje, o estilo dominante para a implementação de programas complexos com um grande número de componentes interagindo.

[37] Alan Kay, born Springfield, MA, May 17th 1940; Kay is one of the inventors of the Smalltalk programming language and one of the fathers of the idea of Object Oriented Programming. He is the conceiver of the laptop computer and the architect of the modern windowing GUI.

Entre o grande número de linguagens orientadas a objetos existentes hoje podemos citar Eiffel (B. Meyer), CLOS (D. Bobrow and G. Kiczales), SELF (D. Ungar and others).

Em particular podemos destacar o fenomenal crescimento da linguagem JAVA desenvolvida pela SUN, que devido a sua grande flexibilidade e usabilidade para aplicações Web popularizou rapidamente o uso e os conceitos da orientação a objetos.

3.5.1.2 - SmallTalk

> "A chave para a compreensão da mudança de paradigma da computação pessoal, interfaces gráficas e modelagem orientada a objetos encontra-se em não ver o trabalho dos anos sessenta como uma coisa velha melhorada. O que ocorreu era mais do que uma forma melhorada de se trabalhar com MainFrames, um mero aumento de funcionalidades para o usuário final ou a transformação das estruturas de dados em algo mais abstrato.[38]"
> (Kay, 1993)

Surgida no final dos anos 60, SmallTalk é considerada uma linguagem arquetípica do paradigma da orientação a objetos, por utilizar, de forma radical, uma grande quantidade de seus conceitos básicos.

Segundo ALAN KAY, o principal idealizador da linguagem, SmallTalk foi à expressão natural do encontro de diversas mentes, idéias e tendências oriundas principalmente das descobertas norueguesas como o SIMULA e dos grandes centros de pesquisa Norte Americanos, principalmente o ARPA (Advanced Research Program Agency) e do PARC (Palo Alto research Center), da Xerox.

SmallTalk fazia parte do conjunto de descobertas que nos levou ao próximo paradigma computacional que incluía entre outras coisas, a computação pessoal, o fim dos clássicos e gigantescos MainFrames e o acesso dos não especialistas ao universo da computação (ou pelo menos o aumento substancial do número de

[38] "The soon to follow paradigm shift of modern personal computing, overlapping window interfaces, and object-oriented design came from seeing the work of the sixties as something more than a "better old thing". That is, more than a better way to do mainframe computing, for end-users to invoke functionality, to make data structures more abstract." (Minha Tradução)

especialistas e a redução (ou alteração) dos requisitos necessários para construir programas de computador).

KAY descreve o surgimento do SmallTalk como uma longa acumulação de influencias onde novas idéias são gradualmente formadas e aceitas através do vislumbre sucessivo de aspectos que vão surgindo ao longo do tempo a partir de diferentes fontes.

> "As novas idéias surgem denunciadas como o trabalho de um insano, e em poucos anos são consideradas obvias e mundanas e finalmente, os detratores originais se apresentam como pais da idéia".
> (Shopenhauer, 1994)

Segundo kay os conceitos da OOP foram vislumbrados por ele algumas vezes ao longo dos anos 60, a primeira aparição significativa se deu em 1961 quando Kay trabalhava como programador para a força aérea americana e surgiu o problema de como transportar arquivos de uma instalação militar para a outra, ambas as instalações utilizavam o computador Burroughs 220 que não possuía um sistema operacional padrão ou formatos definidos para as estruturas e extensões de arquivos. Esta indefinição de padrões obrigou e permitiu que fosse criada (por um analista anônimo) uma elegante solução para a estruturação dos dados a serem transportados.

Os dados foram armazenados em um arquivo dividido em três partes, na terceira parte haviam o dados propriamente ditos, como "diego", "32", "casado", na segunda parte haviam os métodos responsáveis pela manipulação destes dados e na primeira parte haviam registros de ponteiros que referenciavam os métodos da segunda parte.

Esta estrutura revelou-se muito eficiente e manteve-se em uso até ser soterrada pelo uso do COBOL.

Com exceção da primeira parte (a dos ponteiros), esta estrutura se assemelha bastante a atual estrutura de classes e objetos de nossas contemporâneas linguagens orientadas a objeto. Esta classes encapsulam os dados tornando-os acessíveis apenas através dos métodos públicos correspondentes, da mesma maneira que a pouco usual estrutura de intercambio de informações desenvolvida para o Burroughs 220.

Esta inovação persistiu por alguns anos, até que uma linguagem procedural, o ALGOL impôs-se como padrão sobrescrevendo técnicas menos ortodoxas.

Um dos grandes problemas para a ocorrência de uma mudança radical na estrutura das linguagens de programação e da mentalidade do programador era a limitação de capacidade computacional.

> "Em 1968 eu vi duas ou três coisas que mudaram minhas noções de computação.
>
> A forma na qual pensávamos até no momento era baseada na visão de Doug Engelbart, segundo a qual um MainFrame era como uma estrada de ferro, pertencente a uma instituição que decide o que pode e quando pode ser feito.
>
> Englebert estava tentando ser como Henry Ford. Um computador pessoal, da forma como era visto nos anos sessenta, era como um automóvel.
>
> Em 1968 eu vi o primeiro trabalho de Seymour Papert com crianças utilizando LOGO e vi o primeiro realmente eficiente sistema de reconhecimento de padrões de texto escrito a mão no RAND. Era um sistema fabuloso. E teve um grande efeito sobre mim por que gerava certas intuições. Quando misturava as idéias que ele provocava com a idéia de que crianças deveriam utiliza-lo, o conceito de um computador se tornou algo mais parecido com uma super mídia. Algo como um super papel.[39]"
>
> (Kay, 1991)

A medida que surgiam estes avanços nas linguagens de programação e nas estruturas de dados, novos avanços na área de hardware permitiam e provocavam a aceleração deste avanço.

Em 65, KAY lê o artigo de Gordon Moore (Moore, 1965) (Dono e fundador da Intel) onde ele formula a popularmente conhecida “lei de Moore”, que prevê que a

[39] “In 1968 I saw two or three things that sort of changed my whole notion of computing. The way we had been thinking about it was sort of Doug Englebart's view that the mainframe was like a railroad, owned by an institution that decided what you could do and when you could do it.
Englebart was trying to be like Henry Ford. A personal computer as it was thought of in the sixties was like an automobile. In 1968 I saw Seymour Papert's first work with kids and LOGO, and I saw the first really great handwriting-character-recognition system at Rand. It's a fabulous system. And that had a huge influence on me because it had an intimate feel. When I combined that with the idea that kids had to use it, the concept of a computer because something much more like a supermedium. Something more like a superpaper." (M.T.)

quantidade de transistores em um circuito integrado de silicone (chip) deve aumentar exponencialmente a cada ano.

> "Existe um custo mínimo em qualquer período da evolução tecnológica, no presente, ele é atingido quando 50 componente são utilizados por circuito. Mas o custo mínimo esta subindo rapidamente enquanto a curva de custos como um todo tende a diminuir. Se olharmos 5 anos a frente, uma previsão dos custos sugere que o custo mínimo por componentes devera se encontrar em circuitos com mais ou menos 1,000 componentes.
>
> Em 1970, o custo de manufatura dos componentes pode ser previsto em um décimo do custo atual.
>
> A complexidade para elementos de custo mínimo vem duplicando a cada ano. Certamente, com o passar do tempo, a taxa de crescimento é um pouco mais incerta, entretanto, não existe razão para acreditar que ela não se mantenha constante por pelo menos 10 anos.
>
> Isto significa que em 1975, o número de componentes por circuito integrado com custo mínimo será de 65,000.[40]"
>
> (Moore, 1965)

O Artigo de MOORE leva KAY a conceber a possibilidade de maquinas pequenas e potentes o suficiente para que possam implementar com razoável performance programas estruturados no novo paradigma emergente da orientação a objetos, programas que exigiam um grande esforço da maquina devido principalmente ao esforço resultante do mapeamento da descrição formal (programa) na estrutura causal (máquina).

---

[40] "Thus there is a minimum cost at any given time in the evolution of the technology.
At present, it is reached when 50 components are used per circuit. But the minimum is rising rapidly while the entire cost curve is falling (see graph below). If we look ahead five years, a plot of costs suggests that the minimum cost per component might be expected in circuits with about 1,000 components per circuit (providing such circuit functions can be produced in moderate quantities.) In 1970, the manufacturing cost per component can be expected to be only a tenth of the present cost.
The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years.
That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000. I believe that such a large circuit can be built on a single wafer." (Minha Tradução)

> "Dentro do contexto de simbiose homem - máquina do ARPA e na presença do reduzido computador de Ed Cheadle, a "lei" de Moore me veio novamente à mente, desta vez com grande impacto. Pela primeira vez eu pude imaginar a possibilidade de acomodar uma máquina do tamanho de um quarto em cima de uma mesa.[41]"
> (KAY,1993)

Um processo semelhante, fruto da opção entre funcionalidades do software e performance do hardware pode ser observado na estrutura do JAVA, onde a adição de uma maquina virtual permite a distribuição multi plataforma (vários sistemas operacionais) do software ao custo de uma quantidade razoável de processamento extra.

Em 1967 KAY é encaminhado por DAVE EVANS (um dos contratantes do programa de pesquisa avançada do governo dos Estados Unidos conhecido como ARPA (Advanced Research Project Agency) para um trabalho de consultoria em um projeto chefiado por ED CHEADLE, no laboratório de uma companhia aérea local.

O projeto de ED CHEADLE consistia em um pequeno computador pessoal (a Flex-Machine), A maquina de ED CHEADLE não é o primeiro computador pessoal, pois já existia o LINK de WES CLARK, mas o projeto Flex-Machine era o primeiro a possuir como principal objetivo a popularização de seu uso entre os usuários não especialistas.

Enquanto KAY e CHEADLE pesquisavam soluções para seu projeto , Utah recebeu a visita do mítico projetista DOUG ENGELBART.

> "Doug Elgelbert era um profeta de dimensões bíblicas, ele é sem dúvida, um dos pais do que, na Flex-Machine, comecei a chamar de computação pessoal.[42]"
> (KAY,1993)

[41] "In the midst of the ARPA context of human-computer symbiosis and in the presence of Ed´s "little machine", Gordon Moore´s "law" again came to mind, this time with great impact. For the first time i made the leap of putting the room-sized interactive TX-2 or even a 10MIP 6600 on a desk..." (M.T.)

[42] "[...] Doug Engelbart. A prophet of biblical dimensions, he was very much one of the fathers of what on the FLEX-MACHINE i started to call "personal computing"" (M.T.)

Englebart divulgava sua crença de que o resultado do sonho tecnológico fomentado pelo projeto ARPA seria o aumento de potência do intelecto humano através de um veiculo interativo que navegaria através de vetores em um espaço conceitual.

O que seu sistema podia fazer, até mesmo para os padrões atuais era inacreditável, não apenas hipertexto, mas gráficos, navegação eficiente e formas totalmente novas de estabelecer o relacionamento entre maquina e usuário.

> "O impacto desta visão produziu, nas mentes daqueles que estavam ávidos para ver o novo, uma instigante metáfora do que exatamente a computação pessoal deveria ser, e eu imediatamente adotei várias destas idéias na Flex-Machine[43]"
> (Kay, 1993)

No entanto, o sistema apresentado por Doug Engelbert era grande demais para ser implementado na Flex-Machine e apesar de apresentar diversas idéias interessantes não apresentava uma solução definitiva para o advento do modelo de linguagem de programação ideal para o ainda não nascido computador pessoal.

> "A colisão da FLEX-Machine, do monitor de tela plana, dos seminários sobre comunicações de Barton e Grail e do trabalho de mcLuhan e Papert com crianças, ajudou a formar uma imagem de como um computador pessoal realmente deveria ser. Eu recordava Aldus Manutius, que 40 anos depois da prensa, ajustou o livro em suas dimensões atuais, permitindo que pudessem ser transportados em sacos.[44]"
> (Kay, 1993)

No início de 1971, Bob Taylor, um dos contratantes do PARC reuniu um grande grupo, na maioria oriundo de Berkley, não muito tempo depois vários

[43] "the impact of this vision was to produce in the minds of those who were "eager to be augmented" a compelling metaphor of what interactive computing should be like, and i immediately adopted many of the ideas for the Flex-Machine." (M.T)

[44] "Now the collision of the FLEX Machine, the Flat-Screen display, Grail, Barton´s "communications " talk, McLuhan, and Papert's work with children all came together to form an image of what a personal computer

integrantes do grupo de Doug Engelbert se juntaram ao grupo insatisfeitos com a decisão de Engelbert de não apostar em uma versão distribuída de sua NLS, mantendo-se fiel ao sistema de processador único otimizado pelo método de "Time-Sharing".

Kay continuava em busca de um modelo de linguagem de programação que conjugasse as necessidades de leveza, consistência, facilidade de aprendizagem, e grande poder de modelagem.

O conceito de modelagem através de objetos oriundo do SIMULA permanecia forte apesar de gerar algumas perplexidades lógicas devido a algumas inversões hierárquicas nas noções de funções (métodos) e objetos.

> "Eu levei um tempo consideravelmente longo para perceber isto, acredito que isto se deva parcialmente ao fato de que invertem-se as noções tradicionais de operadores e funções.[45]"
> (Kay, 1993)

No verão de 71, as idéias foram refinadas e surgem pela primeira vez o rótulo SmallTalk e o produto correspondente, uma linguagem de programação leve, robusta, portátil e totalmente voltada para a visão de um mundo onde todas as coisas são objetos.

> "O nome foi também uma reação contra a tradição deísta indo européia onde os sistemas recebiam nomes como ZEUS, ODIN e THOR, e dificilmente realizavam alguma coisa. Eu imaginei que SmallTalk fosse um nome tão inócuo que se por acaso ele fizesse alguma coisa boa as pessoas iriam ficar agradavelmente surpresas[46]"
> (Kay, 1993)

---

really should be. I remembered Aldus Manutius who 40 years after the printing press put the book into its modern dimensions by making it fit into saddlebags" (M.T.)

[45] "It take me a remarkably long time to see this, partly I think because one has to invert the traditional notion of operators and functions"

[46] "The name was also a reaction against the "IndoEuropean God Theory" where systems were named ZEUS, ODIN and THOR, and hardly did anything. I figured that SmallTalk was so innocuous a label that if it ever did anything nice people would be pleasantly surprised" (M.T.)

Quando me refiro a uma linguagem totalmente orientada a objetos quero dizer que não existe no Smalltalk nenhum tipo primitivo, nada existe a priori, todos os elementos com os quais se constrói um sistema são classes e instancias de classes, não existem funções soltas, todas as funções fazem parte do escopo de uma classe e para utiliza-las é preciso instanciar objetos pertencentes as classes correspondentes.

3.5.2 – Principais Conceitos do paradigma da Orientação a Objetos.

> "Nós (humanos) desenvolvemos uma técnica particularmente poderosa para lidar com a complexidade. Este é o motivo pelo qual realizamos abstrações. Incapazes de lidar com a totalidade de um objeto complexo, nós optamos por ignorar seus detalhes não essenciais, passando a lidar com uma versão generalizada, um modelo idealizado do objeto.[47]"
> (Wulf e Shaw 1981)

3.5.2.1 - Classes.

> "Uma classe é um esquema ou protótipo que define as variáveis e os métodos comuns a todos os objetos de um certo tipo.[48]"
> (The JavaTM Tutorial)

Em nossa experiência cotidiana, podemos com freqüência encontrar muitos objetos do mesmo tipo, por exemplo, a bicicleta do vizinho é apenas uma das bicicletas do mundo.

Utilizando a terminologia da Orientação a Objeto podemos dizer que uma bicicleta é uma instância da classe de objetos (re)conhecidos como bicicletas.

Bicicletas possuem atributos (quadro, rodas e pedais por exemplo) e comportamentos (andar, quebrar, empinar) em comum.

---

[47] "We (human) have developed an exceptionally powerful technique for dealing with complexity. We abstract for it. Unable to master the entirety of a complex object, we choose to ignore its inessential details, dealing instead with the generalized, idealized model of the object." (M.T)

[48] "A class is a blueprint or prototype that defines the variables and the methods common to all objects of a certain kind." (M.T)

Na fabricação de bicicletas, o fabricante utiliza a seu favor o fato de que as bicicletas partilham características (atributos e métodos) e fabrica muitos tipos de bicicletas a partir do mesmo esquema básico, pois obviamente seria muito ineficiente projetar cada uma das bicicletas a partir do zero.

Na programação orientada a objetos é também possível a existência de muitos objetos que partilhem características, como no caso das bicicletas, é possível tirar vantagem da similaridade destes objetos e criar uma definição genérica dos atributos e métodos comuns a estes objetos, esta definição comum é chamada de classe.

Uma classe é a generalização de um conjunto de características comuns a um grupo de objetos.

A partir da descrição do conjunto de atributos (dados) e métodos (ações) comuns ao grupo podemos a qualquer momento criar uma nova instancia de uma determinada classe.

3.5.2.2– Objetos.

> "Um objeto de um sistema é uma coleção de variáveis e métodos relacionados. Objetos em sistemas são geralmente utilizados para modelar objetos do mundo real que podem ser encontrados na vida cotidiana[49]"
>
> (The JavaTM Tutorial)
>
> "Nós permanecemos limitados pelo número de elementos que podemos compreender simultaneamente, mas através do mecanismo de abstração, nós utilizamos partes da informação com enorme conteúdo semântico.
> Isto é particularmente verdadeiro se nós observarmos uma visão Orientada a Objetos do mundo, por que objetos, como abstrações ou entidades do mundo real, representam um conjunto de informações particularmente denso.[50]"

[49] "An object is a software bundle of related variables and methods. Software objects are often used to model real-world objects you find in everyday life." (M.T.)
[50] "We are still constrained by the number of things that we can comprehend at one time, but through abstraction, we use chunks of information with increasingly greater semantic content. This is especially true if

(**Booch ,1994)**

Um objeto é uma instancia de uma determinada classe, podemos olhar ao redor e ver muitos exemplos de objetos do mundo-real: Um cachorro, uma mesa, uma bicicleta, pessoa ou televisão.

Todos os objetos do mundo real possuem algumas regras estruturais comuns, todos possuem estados e comportamentos, por exemplo, um cachorro possui estados (Definidos pelo valor de seus atributos como cor, nome, estomago (cheio ou vazio) etc...) e comportamentos (babar, correr, comer, abanar o rabo).

Um objeto em um programa também possui estados e comportamentos, um objeto de um programa mantém seus estado através do valor de suas variáveis, por exemplo, se criarmos um objeto cachorro e atribuirmos o valor zero a variável QuantidadeDeComidaNoEstomago, o estado do cachorro será “com fome”, se alterarmos o valor da variável para 10 o estado do cachorro será “satisfeito ou sem fome”.

3.5.2.3– Encapsulamento.

Dependendo do escopo de controle das variáveis e dos métodos, um objeto pode ser visto como uma caixa preta, o valor das variáveis só pode ser alterado de dentro do próprio objeto através da ação de seus métodos públicos.

Por exemplo, se criarmos uma classe Prefeito e criarmos uma instancia desta classe chamada oPrefeitoDeTeresopolis.

A classe prefeito possui um atributo chamado salário, caso este atributo seja de acesso público, ou seja, não esteja encapsulado, poderíamos através da instancia da classe atribuir qualquer valor para a variável salário, caso a variável esteja devidamente encapsulada, qualquer alteração em seu valor só poderá ser realizada através do método AlterarSálario, que por sua vez possuirá uma implementação própria que pode impor regras a alterações que estejam fora da legislação brasileira.

Classe Prefeito

we take an object-oriented view of the world, because objects, as abstractions of entities in the real world, represent a particularly dense and cohesive clustering of information.” (M.T.)

Private doble Salário -> Atributo
Alterarsalario (double valor) -> Método

Utilizando a sintaxe do Java, podemos criar um objeto oPrefeitoDeTeresopolis que é uma instancia da classe Prefeito:

```
Prefeito oPrefeitoDeTeresópolis = new Prefeito()
```

Se a variável salário não fosse encapsulada poderíamos fazer a seguinte ação:

```
OPrefeitoDeTeresópolis.Salário=60.000,00
```

Com a variável encapsulada precisamos fazer:

```
OprefeitodeTeresopolis.AlteraSalario(60.000,00)
```

A alteração do valor da variável dependerá então da implementação do método **AlteraSalario()**, por exemplo, no caso em que o salário Maximo permitido a um prefeito seja 4.000,00:

```
AlteraSalario(double  osalario){
    If (osalário <= (4.000,00)){
            Salário=osalario;
    }
}
```

Sem encapsulamento, o estado dos objetos, definido sempre pelo valor de suas variáveis, fica vulnerável a intervenção direta de outros objetos.

3.5.2.4– Herança.

> "Uma classe herda seu estado e seus comportamentos de sua superclasse. A Herança fornece um mecanismo poderoso e natural para organizar e estruturar sistemas computacionais[51]"
> (**The Java™ Tutorial)**

De uma forma geral, objetos são definidos em termos de classes, podemos saber bastante a respeito de um objeto conhecendo sua classe.

Mesmo que não saibamos o que é uma "caloi", se soubermos que é uma bicicleta também saberemos que possui duas rodas, pedais e guidão.

Software orientado a objetos vai um passo alem deste conceito e permite que classes sejam definidas por outras classes.

Por exemplo, bicicletas de corrida e bicicletas de crianças são ambas tipos de bicicletas, na programação orientada a objetos elas são subclasses da classe bicicleta, que por sua vez é a superclasse das classes que herdam suas características.

Cada SubClasse herda atributos e métodos de sua SuperClasse, no entanto as SubClasses não estão restritas aos atributos e métodos herdados de sua SuperClasse, SubClasses podem adicionar novas características às herdadas, adquirindo características próprias alem das características da SuperClasse.

[51] "A class inherits state and behavior from its superclass. Inheritance provides a powerful and natural mechanism for organizing and structuring software programs." (M.T.)

3.5.2.5 – Polimorfismo.

> "Polimorfismo é a capacidade de um objeto para assumir diferentes formas. Em uma linguagem orientada a objetos, polimorfismo é a propriedade de que uma mensagem pode significar coisas diferentes dependendo do objeto que irá recebe-la.[52]"
> (Cohoon e Davidson, 2002)

SubClasses podem sobrescrever métodos herdados fornecendo implementações especializadas destes métodos, geralmente esta escolha se da através da análise dos parâmetros de entrada de um determinado método, cada combinação específica de parâmetros de entrada ativa uma diferente implementação do método em questão, gerando um comportamento especifico.

3.5.2.6 - Mensagens.

A idéia de modelar um sistema através da compreensão da estrutura dos objetos envolvidos deriva naturalmente da necessidade de modelar sistemas que possuem mais de um objeto envolvidos e onde estes objetos interajam entre si.

No entanto, os atributos dos objetos são, em geral, encapsulados (como visto acima), e desta forma, os objetos devem interagir entre si através de mensagens, trocadas através da API (Public Advanced Interface) de cada classe.

A API consiste em uma lista de métodos públicos que podem ser invocados a partir de uma instancia da classe (objeto), a implementação destes métodos é interna e vedada ao objeto, mas através deles os outros objetos do sistema podem passar mensagens, na forma de parâmetros do método, e fazer com que os objetos de um sistema interajam entre si.

Um objeto sozinho não costuma ser muito útil, uma bicicleta largada na calçada não passa de metal e plástico, a bicicleta é ***útil*** apenas quando outro objeto (pessoa) interage com ela, passando um parâmetro força para o método

[52] "Polymorphism is the capability of something to assume different forms. In an object-oriented language, polymorphism is the property that a message can mean different things depending on the object receiving it." (M.T.)

Pedalar(força){} que por sua vez será responsável pela alteração do estado do objeto (resultante do valor de seus atributos).

3.5.3 – UML (Modelagem Orientada a Objetos).

Como já foi dito anteriormente, durante a descrição da história do SIMULA, o paradigma de desenvolvimento Orientado a Objetos consiste tanto em uma descrição do sistema para computadores, mapeada através dos softwares intermediários, quanto de uma descrição do sistema para seres humanos.

Devemos lembrar que a medida que aumenta a complexidade do sistema, aumenta também a necessidade de envolvimento de vários indivíduos (analistas, programadores etc...) no processo, e estes indivíduos devem, de alguma forma, tornar comuns suas idéias e impressões sobre um sistema e seus objetos.

3.5.3.1– História da UML.

Na década de oitenta, os objetos começaram a sair dos laboratórios de pesquisa e dar seus primeiros passos em direção ao mundo “real”. SmallTalk estabilizou uma plataforma que as pessoas podiam usar e C++ surgiu.

Neste período surge uma preocupação crescente a respeito dos métodos de projeto se encaixariam em um mundo orientado a objetos.

Métodos de projetos tornaram-se muito populares nas décadas de setenta e oitenta e havia uma crença generalizada de que formas especificas de modelagem deveriam surgir par auxiliar no desenvolvimento de sistemas orientados a objetos.

Até 1994, não existia ainda um consenso a respeito do método ideal para o desenvolvimento OO, cada um dos principais metodologistas liderava informalmente um grupo de seguidores.

De uma forma geral, os métodos eram bastante semelhantes, mas eles possuíam uma série de pequenas diferenças que poderiam causar, entre outros malefícios, interpretações dúbias de uma determinada notação.

A padronização veio a partir do anuncio da união de dois dos principais metodologistas, Grady Booch e Jim Rumbaugh que haviam largado suas antigas ocupações (GE e IBM) para fundarem a Rational Software (empresa voltada para

a construção de ferramentas CASE (computer aided software enginiering), aplicativos de auxilio ao desenvolvimento de sistemas).

Em 1995 Booch e Rumbaugh apresentaram a primeira descrição pública de seu método unificado, Versão 0.8 da documentação do *Método unificado*.

UML (Unified Modeling language) é a sucessora da onda de métodos de análise e projeto orientado a objetos (OOA&D) que surgiu no final dos anos oitenta e início dos anos noventa. Mais especificamente, ela unifica os métodos de Booch, Rumbaught e Jacobson (que se juntou a eles um pouco depois).

UML é uma linguagem de modelagem, não um método, a maioria dos métodos consiste, em principio, de uma linguagem de modelagem e de um processo. A linguagem de modelagem é a notação, utilizada por métodos para expressar um sistema. O processo é a sugestão dos passos a serem seguidos na elaboração de um projeto.

> "...acredito que a maioria das pessoas quando diz que está utilizando um método usa a linguagem de modelagem, mas raramente segue o processo. Portanto, em vários aspectos, a linguagem de modelagem é a parte mais importante do método. Ela é a parte chave para a comunicação. Se você quer discutir o seu projeto com alguém, vocês tem que compreender a linguagem de modelagem, não o processo que vocês utilizaram para desenvolver o projeto. "
> (Fowler, 2000)

A versão final da UML conta com cerca de doze tipos diferentes de diagramas, mas cinco deles compõem seu corpo principal, estes cinco diagramas pretendem estabelecer cinco pontos de vista diferentes do sistema e de seus componentes.

3.5.3.2 – Conceitos Básicos da UML.

3.5.3.2.1 – Diagrama de casos de uso: Ponto de vista externo.

> "Um dos maiores desafios no desenvolvimento de sistemas é o da construção do sistema certo (aquele que o usuário espera). Isso se torna difícil devido aos ruídos de comunicação, a diferença de jargões e linguagens utilizadas por desenvolvedores e clientes, atingir uma boa comunicação entre os habitantes de universos profissionais diversos é um dos grandes desafios de um desenvolvedor de software.
>
> Por muito tempo, em desenvolvimento tradicional e orientado a objetos, as pessoas usavam interações típicas para ajuda-las a compreender os requisitos do sistema. Entretanto, estes cenários eram tratados muito informalmente – sempre feitos, mas raramente documentados."
> (Fowler, 2000).

Tradicionalmente, o levantamento de requisitos para a correta construção de um sistema (levando-se em conta que um bom sistema é aquele que atende a seus requisitos da melhor forma possível, o que compreende diversos critérios como praticidade, eficiência, completude e consistência).

O diagrama de casos de uso representa uma visão externa do sistema, no diagrama de caso de uso não existe a preocupação com a estrutura interna ou com a implementação das funcionalidades.

A principal preocupação do diagrama de casos de uso é estabelecer junto ao especialista do domínio, usuário, todas as funcionalidades que compõe os requisitos de um determinado sistema.

No D.C.U. são definidos os atores que irão interagir com o sistema e as funcionalidades as quais eles terão acesso.

Do ponto de vista conceitual, o D.C.U. efetua um recorte, diérese, do minimundo que queremos capturar, utilizando como pontos de recorte os aspectos separadores externamente visíveis.

O D.C.U é composto de figuras e diagramas extremamente simples, obedece a um conjunto de regras exíguo e representa a visibilidade externa das funcionalidades do sistema, de seus atores e de sua interação.

Olhando sob um outro aspecto, o D.C.U representa os efeitos reais que serão esperados dos objetos em jogo, e a visibilidade de seus métodos de acesso público, sua A.P.I (*Advanced Public Interface*).

3.5.3.2.2 – Diagrama de Classes.

O diagrama de classes é o mais utilizado e o receptáculo para o maior escopo de conceitos de modelagem OO.

Um diagrama de classes descreve os tipos de objetos no sistema os vários tipos de relacionamento estático que existem entre eles.

Segundo fowler (2000) , há dois tipos principais de relacionamento estático:

3.5.3.2.2.1 - Associações (Por exemplo, uma biblioteca que possuí livros).

As associações são os diversos relacionamentos necessários e possíveis entre objetos, estes relacionamento se estabelece quando um objeto invoca, através de uma mensagem, um método público de outra mensagem.

3.5.3.2.2.2 - Subtipos (Uma biblioteca pública é um tipo de biblioteca).

Subtipos são a expressão no diagrama da UML da característica das linguagens OO conhecida como Herança, a capacidade de uma subclasse de herdar atributos e métodos de suas superclasses.

Algumas linguagens de programação OO admitem herança múltipla, outras não, no entanto isto não constitui uma limitação para uma linguagem de notação como a UML, onde qualquer uma das alternativas é viável.

Segundo Cook e Daniels (1994) existem três principais perspectivas nas quais um diagrama de classes pode ser utilizado:

3.5.3.2.2.3 - Perspectiva conceitual.

> "Se tomar a perspectiva conceitual, você projeta um diagrama que representa os conceitos no domínio que está sendo estudado. Estes conceitos serão naturalmente relacionados às classes que vão executa-los, mas freqüentemente não existe um mapeamento direto. Um modelo conceitual deve ser implementado com pouca ou nenhuma preocupação com o *software* que poderá implementa-lo, portanto ele pode ser considerado independente de linguagem."

A perspectiva conceitual procura identificar os macro aspectos de cada um dos objetos que compõe o sistema sem detalhar, através de seus métodos e atributos, o funcionamento e a estrutura interna destes objetos.

3.5.3.2.2.4 - Perspectiva de Especificação.

> "Agora estamos analisando o *software*, mas estamos analisando suas interfaces, não sua implementação. O desenvolvimento orientado a objeto põe muita ênfase na diferença entre interface e implementação"

A perspectiva de especificação possui uma quantidade de detalhes maior do que a perspectiva conceitual, pois além de explicitar as classes dos sistema explicitam também as ações que os objetos instanciados a partir destas classes devem ser capazes de executar e como os outros objetos devem fazer para passar as devidas mensagens para cada uma destas ações.

3.5.3.2.2.5 – Perspectiva de implementação.

> "Nesta visão, realmente, temos classes e estamos pondo a implementação às claras. Esta é, provavelmente, a perspectiva usada com mais freqüência, mas, de várias formas, a perspectiva de especificação é freqüentemente a melhor para ser usada".
> (Fowler, 2000)

Na perspectiva de implementação somam-se todas as características da perspectiva conceitual e da perspectiva de especificação, alem do detalhamento da implementação de cada um dos métodos da classe, esta perspectiva é

freqüentemente dependente da tecnologia (linguagem) escolhida para o desenvolvimento, é uma perspectiva de baixo nível com pouca expressividade das características gerais do sistema mas muito útil para alguém que, de fato, precise efetuar manutenções ou alterações em um sistema desconhecido.

> "A compreensão das diversas perspectivas é crucial tanto para desenhar como para ler diagramas de classes. Infelizmente, as linhas entre as perspectivas não são rígidas, e a maioria dos modeladores não se preocupa em ter suas perspectivas classificadas quando eles estão modelando. Embora acredite que isto não afeta muito a perspectiva conceitual e a perspectiva de especificação, é muito importante separar a perspectiva de especificação e a perspectiva de implementação."
> (Fowler, 2000)

Apesar de não constarem na constituição formal da UML, as perspectivas fornecem um valioso auxilio a comunicação entre os indivíduos envolvidos e a compreensão das diversas visibilidades de um objeto, que segundo esta definição pode ser visto de uma distancia longa (conceitual), onde apenas as grandes linhas são discerníveis, média (Especificação), onde podemos visualizar as interfaces para os métodos da classe e pequena, onde podemos visualizar as características gerais, as interfaces para os métodos e os detalhes de sua implementação.

3.5.3.2.3 - Diagrama de seqüência: Visão do sistema em movimento.

Diagramas de seqüência são modelos que descrevem como grupos de objetos colaboram em algum comportamento que dependa de todos eles.

> "Tipicamente um diagrama de seqüência captura o comportamento de um único Caso de Uso. O diagrama mostra vários objetos e as mensagens que serão passadas entre este objetos em um Caso de Uso."
> (Fowler, 2000)

O diagrama de seqüência representa a interação entre objetos, através de mensagens, com o intuito de testar o funcionamento de um determinado aspecto do sistema.

3.5.3.2.4 – Diagrama de atividade: Ponto de Vista estrutural sem detalhes.

Diagramas de atividade são uma das partes mais inesperadas da UML.

Ao contrário da maioria das outras técnicas da UML, diagramas de atividades não têm origens claras junto com os demais diagramas, os diagramas de atividades parecem ser um componente herdado dos paradigmas de modelagem anteriores e assemelha-se muito a um tradicional fluxograma.

O diagrama de atividades é particularmente útil para modelar processos de "workflow" ou qualquer outro tipo de processos que conte com diversas atividades paralelas.

O diagrama de atividades descreve a seqüência de atividades com suporte para comportamento condicional e paralelo.

3.5.3.3 – Observações.

Procurei construir neste capitulo minha visão pessoal da natureza e do papel das linguagens de programação em particular e das linguagens de programação e técnicas de análise e modelagem orientadas a objetos.

A intuição principal desta dissertação é a de que os conceitos que subjazem o desenvolvimento orientado a objetos possuam algum tipo de relação com estruturas cognitivas especificas.

IV - Exemplos da recorrência dos conceitos da Orientação a Objetos no Tractatus Logico-Philosophicus de Ludwig Wittgenstein.

4.1 – Migração de conceitos.

O objetivo do presente capítulo é tentar fortalecer a hipótese de que os conceitos do paradigma de desenvolvimento de software Orientado a Objetos possuam algum nível de relacionamento estrutural com aspectos específicos do aparato cognitivo.

Isto será realizado através da observação de idéias e teorias escolhidos de acordo com a meta-idéia[53], recorrente na epistemologia e na teoria do conhecimento, de que determinados padrões de idéias e pensamentos podem emergir e submergir ao longo da história do pensamento.

A existência de conceitos e termos com algum grau de semelhança em lugares, tempos e disciplinas diferentes suscita o aparecimento de padrões, que podem reforçar ou enfraquecer uma determinada idéia ou ponto de vista.

A percepção de que determinadas idéias podem ocorrer em tempos, lugares e disciplinas diferentes já foi amplamente explorada pela epistemologia e recebe o nome genérico de migração de conceitos.

Existem diversos bons exemplos de migrações de conceito, um deles, mais literário do que cientifico pode ser encontrado em um ensaio de JORGE LUIS BORGES **(Borges, 1999)** chamado “A flor de Coleridge”, neste ensaio, BORGES defende a idéia de que o ressurgimento de determinadas imagens na história da literatura mundial não derive de cópias ou influencias, mas do fato puro e simples de que humanos com a mesma estrutura cognitiva encaminhem de formas semelhantes a resolução de determinados problemas e situações.

> “Por volta de 1938, Paul Valéry escreveu: “A história da literatura não deveria ser a história dos autores e dos acidentes de sua carreira ou da carreira de suas obras, e sim a história do Espírito como produtor ou consumidor de literatura. Essa história poderia ser levada a termo sem mencionar um único escritor”. Não era a primeira vez que o espírito formulava essa observação; em 1844, no povoado de Concord, outro de seus amanuenses anotara: “Dir-se-ia que uma

[53] Idéia que se aplica a outras idéias.

> única pessoa redigiu quantos livros há no mundo; há neles tal unidade central que é inegável serem obra de um único cavalheiro onisciente" (Emerson: Essays,2 V III).
>
> Vinte anos antes, Shelley sentenciou que todos os poemas do passado, do presente e do porvir são episódios de um único poema infinito, construído por todos os poetas do orbe (A Defense of Poetry, 1821)
>
> Estas considerações (implícitas, sem dúvida, no panteísmo) permitiram um infindável debate; eu, agora, invoco-as para executar um modesto propósito: a história da evolução de uma idéia, por meio dos textos heterogêneos de três autores"
>
> (Borges, 1999)

Mesmo que as circunstancias não sejam absolutamente iguais, a semelhança na estrutura cognitiva do sujeito e na estrutura do problema apresentado bastam para prover uma igualdade de circunstancias.

BORGES apresenta, a titulo de exemplo, uma situação excepcionalmente comum na literatura, a situação na qual o protagonista da trama passa por uma experiência antinatural como viagem no tempo, ou um sonho anormalmente vívido por exemplo, e ao retornar a esfera do cotidiano e questionar-se sobre a veracidade das experiências vividas, encontra alguma prova material irrefutável da existência real da experiência sofrida.

Este é exatamente o caso da "Flor de Coleridge", onde o protagonista acorda de um sonho particularmente realista e encontra em sua mão uma flor oriunda do jardim do sonho, que não poderia de outra forma ter se materializado naquele lugar.

A idéia da prova material que dissipa a sensação do sonho é um conceito migrante na literatura.

Um outro exemplo interessante pode ser encontrado no artigo "Proofs are Programs: 19th century Logic and 21st Century Computing" de Philip Wadler, onde o autor efetua uma comparação entre os resultados obtidos por lógicos e por cientistas de computação, que chegam, por diferentes caminhos a resultados semelhantes.

> "Em épocas diferentes, um lógico motivado por preocupações lógicas e um cientista de computação motivado por preocupações práticas descobriram o mesmo sistema de tipos.
>
> Muitas linguagens de programação funcionais modernas utilizam o sistema de tipos Hindley-Milner. Descoberto pelo lógico Hindley em 1969 e redescoberto pelo cientista de computação Reynolds em 1974.[54]"
>
> (Wadler, 2000)

Estes exemplos ilustram minha intenção de propor que os conceitos que subjazem o desenvolvimento de sistemas Orientados a Objetos sejam conceitos recorrentes com expressões idênticas ou muito semelhantes em outras obras que possuam como objetivo modelar o mundo exterior ou o processo interior de percepção do mundo, o aparato cognitivo.

Diante da farta quantidade de material disponível (a história da filosofia é rica em tentativas de modelagem através da generalização de objetos em classes) e da necessidade de uma escolha única (devido a limitações de espaço e em prol da concisão e clareza do texto), optei por uma breve análise de partes do Tractatus Lógico-Philosophicus de Ludwig Wittgenstein e da localização nesta obra de conceitos com algum grau de semelhança com os encontrados nos métodos de desenvolvimento Orientados a Objetos descritos no capítulo anterior.

4.2 – O Tractatus e Seus Objetivos.

Para compreender as possíveis relações entre o Tractatus Lógico-Philosophicus (referenciado a partir deste ponto como TLF) e os conceitos da programação orientada a objetos, precisamos nos focar, primeiramente, nos **objetivos gerais** de Wittgenstein ao escrever sua primeira e principal obra.

Segundo Bertrand Russel em sua introdução a obra de Wittgenstein (Russel,1922), um dos objetivos de Wittgenstein era **limpar a linguagem de tudo o que fosse dúbio e duvidoso**, criando então uma linguagem logicamente perfeita através da qual fosse possível expressar os fatos e estados de coisas

[54] "Time and again, a logician motivated by logical concerns and a computer scientist motivated by practical concerns have discovered exactly the same type system, usually with the logician getting there first. Most modern functional languages use the Hindley-Milner type system. Discovered by the logician Hindley in 1969 and re-discovered by the computer scientist Reynolds in 1974."

(combinações de fatos) do mundo com a mínima ambigüidade possível. Este sistema simbólico formalizado permitiria entre outras coisas uma eficiente comunicação intersubjetiva, que não deixaria margens para dúvidas ou interpretações, sendo ideal para a transmissão de conhecimentos gerais ou específicos.

Outro dos objetivos seria compreender e expor o elemento comum entre o referente extralingüístico (o mundo) e seu representante (a linguagem), aquilo que permite que o representante execute sua função de representar.

Wittgenstein pretendeu atacar estas questões através do exame criterioso das estruturas lingüísticas e da conseqüente geração de uma linguagem (sistema simbólico) rigorosa (logicamente perfeita). Seu método é o de drenar da linguagem tudo aquilo que é dúbio e permite o surgimento de paradoxos (isto inclui os símbolos não essenciais da linguagem natural).

Dito desta forma, não parece existir muita diferença de objetivos (embora as diferenças metodológicas sejam imediatamente visíveis) entre o Principia Matemática (1964) de Russel e o TLF, no entanto esta impressão é rapidamente sucedida por uma das mais fortes conclusões do TLF: a idéia de que este algo em comum existente entre o mundo e a linguagem não poderia ser dito, mas apenas mostrado, caso contrario a linguagem possuiria a capacidade de representar algo que esta fora de sua capacidade de representar (conceber o inconcebível).

> "Mais interessante que estas questões relativamente particulares é a atitude do Sr. Wittgenstein diante do místico. Sua atitude a este respeito provém naturalmente de sua doutrina lógica pura, segundo a qual a proposição lógica é uma figuração (verdadeira ou falsa) do fato, e tem em comum com o fato uma certa estrutura. É essa estrutura comum que a torna capaz de ser uma figuração do fato, mas a própria estrutura não pode ser posta em palavras, já que é uma estrutura de palavras, tanto quanto dos fatos aos quais elas se referem. Portanto, tudo que está envolvido na própria idéia da expressividade da linguagem não deve admitir expressão na linguagem e é, portanto, num sentido perfeitamente preciso, inexprimível."
> (Russel,1922)

Curiosamente, esta afirmação, a afirmação de que existem coisas que não podem ser ditas coloca Wittgenstein em oposição com a corrente da qual é inspiração máxima e paradoxal representante, o positivismo lógico do Circulo de Viena.

O Circulo de Viena consistia em um grupo de intelectuais de diversas disciplinas cujo principal objetivo era determinar as condições de verdade de uma determinada proposição cientifica ou filosófica. Uma imagem mais viva do Circulo de Viena pode ser obtida através da enumeração de alguns de seus integrantes.

> "Moritz Shlick (Fundador do Circulo de Viena), Otto Neurath (Economista e sociólogo), Herbert Feigl (Fisico), Rudolf Carnap (lógico e especialista em notação matemática), Kurt Godel (Lógico e matemático), Viktor Kraft, Felix Kaufmann, Phillip Frank, Friedrich Waissman (Filosofo e principal colaborador de Wittgenstein durante o período que sucedeu a primeira grande guerra) e Hans hahn (matemático) (e a irmã cega de Hans que fumava charutos, Olga, especialista em lógica booleana)"
> (Edmonds e Eidinow, 2003).

Este grupo buscava, assim como Wittgenstein, o melhor esclarecimento possível do obscuro relacionamento entre linguagem e realidade.

O circulo de Viena investigava a possibilidade de se afirmar a verdade ou falsidade de uma proposição com base em processos formais que oferecessem rigor suficiente para diluir as duvidas obscurantistas e niilistas dos que afirmavam que esta possibilidade estava alem do alcance da linguagem.

Para os integrantes do Circulo de Viena, assim como para o RUSSEL dos PRINCIPIA, não haveria, até que surgisse uma evidencia contraria, limites para a capacidade de abrangência dos sistemas formais, todos os limites poderiam ser superados através de melhorias e avanços nos próprios sistemas, mas não existiria uma barreira categoricamente intransponível.

A prova matemática da impossibilidade da construção de um sistema formal completo e consistente viria em 1931 (NAGEL, 1998), através da prova de KURT GODEL de que qualquer sistema formal deveria optar entre a completude (capacidade de gerar todas as proposições / teoremas possíveis) ou consistência (capacidade de NÃO gerar proposições / teoremas contraditórios).

O impacto das demonstrações de GODEL são expressos por CHAITIN de um ponto de vista pessoal.

O TLF aponta para uma conclusão bastante semelhante à de GODEL, mas realiza a demonstração através da expressão da incapacidade de qualquer linguagem em falar a respeito daquilo que confere sentido a si própria, aquilo que faz com que sua estrutura possua uma determinada quantidade de formas corretas (espaço lógico).

Para Wittgenstein a semântica seria fruto de atributos ocultos da sintaxe, o elemento que concederia sentido a uma proposição seria o mesmo que concede sentido ao mundo, objeto da proposição.

No entanto, Wittgenstein identifica uma série de estruturas comuns às diversas linguagens e que deveriam fazer parte de uma linguagem logicamente perfeita, ou próxima disto.

Estas estruturas comuns são *templates* sobre os quais a grande variedade de relações poderia ser derivada e são alguns destes padrões estruturais que proponho analisar na seção abaixo.

Existem vários pontos de apoio nas duas principais obras de Wittgenstein, o Tractatus Lógico-Philosophicus e as Investigações Filosóficas, que podem ser utilizados para estabelecer um relacionamento e um diálogo com os conceitos constitutivos das linguagens de modelagem e desenvolvimento Orientadas a Objetos, irei utilizar principalmente alguns aforismos e trechos escolhidos no “Tractatus Logico-Philosophicus” e nas “Investigações Filosóficas” que no meu julgamento ilustram a existência de conceitos com algum grau de semelhança com os principais conceitos da Orientação a Objetos, mais especificamente os conceitos de classes, objetos, métodos, atributos, encapsulamento, polimorfismo e troca de mensagens.

Wittgenstein pretende no TLF descrever as condições que devem ser preenchidas por uma linguagem logicamente perfeita. Seu principal compromisso no desenrolar desta tarefa não é com o aspecto psicológico de utilização da linguagem para expressar sentidos específicos e pessoais. Nem com o aspecto epistemológico da relação entre os nomes e seus referentes, o que constituía a grande questão lingüística da antiguidade.

> "Platão (429-347 a.C.) consagrou um de seus diálogos, o Crátilo, a este problema. Dos três interlocutores que ele retrata, Crátilo sustenta que a língua espelha exatamente o mundo; Hermógenes defende a posição contrária, a de que a língua é arbitrária; e Sócrates representa instância intermediária..."
> (Weedwood, 1995)

A grande diferença parece residir no foco da preocupação, enquanto a antiguidade preocupava-se com a relação entre nomes e coisas, a atenção de Wittgenstein parece voltar-se para a natureza e os resultados das conexões entre os nomes e para a relação entre proposições e estados de coisas.

O questionamento a respeito do poder da linguagem de referenciar-se a algo fora dela permanece forte como nunca.

> "o que um fato deve possuir em comum com outro para que possa ser um símbolo do primeiro ?"
> (Russel, 1922)

Entretanto, a questão principal do Crátilo parece ser a definição da linguagem como elemento do conjunto das coisas que são naturais (Phisys) ou forjadas pelos homens (Nómos), e esta questão, para Wittgenstein parece encontrar-se em um nível inferior, se é que ainda existia.

Para Wittgenstein, a questão principal, que ofuscava todas as outras era "Qual o elemento comum entre a linguagem e a realidade, que permite que uma seja representada pela outra?".

Todas as outras questões tornam-se secundárias, não faz diferença se a origem da linguagem é a natureza ou a sociedade, independente de sua origem o poder representador se mantém, e o fato deste poder existir é inegável e a busca por sua natureza e seus limites a única e verdadeira questão da filosofia.

Para Wittgenstein, este algo poderia ser encontrado nas estruturas e não nos elementos estruturados, e a origem da estrutura (sujeito/objeto, natural/artificial) importava menos do que a compreensão de sua natureza.

> "Na figuração e no afigurado deve haver algo de idêntico, a fim de que um possa ser, de modo geral, uma figuração do outro."
> (TLF, 2.161)

O objeto pode possuir múltiplos significados, a proposição não, pois aquilo que significa na proposição encontra-se mais em sua forma do que em seus componentes.

> "O que uma proposição deve ter em comum com a realidade para poder afigura-la à sua maneira – correta ou falsamente – é sua forma de afiguração."
> (TLF, 2.17)

Este tipo de simbolismo pressupõe a idéia de que os objetos podem possuir referencias múltiplas e indistintas, mas a combinação de objetos referencia um e apenas um estado de coisas possível.

O problema da significação dos símbolos parece centrar-se no fato de que os componentes da linguagem não são, sozinhos um "*auto logos*[55]*",* mas recebem sua significação definitiva no momento em que são contextualizados junto com os outros componentes da proposição (estado de coisas proposicional) que serve de símbolo para um estado de coisas real (combinação de objetos do mundo). No entanto, a forma de arrumação dos objetos não pode ser totalmente arbitraria, devendo seguir regras de derivação e combinação previamente definidas para que a linguagem seja consistente (não gere estados de coisas impossíveis ou proposições dúbias, proposições que signifiquem mais de um estado de coisas). Um objeto determinado não pode assumir qualquer papel em uma proposição. Para que esta condição (de adequação) seja satisfeita, é necessário que exista a possibilidade de troca de mensagens[56] entre os objetos da proposição.

[55] Um Auto Logos é uma unidade da linguagem que possui uma relação biunívoca com um referente extralingüístico. Existem diversas teorias sobre a passagem do léxico (pedaço) para o Auto Logos. No TLF um nome não é considerado um Auto Logos pois pode possuir múltiplos relacionamentos extralinguisticos, a proposição entretanto, possui o poder de retratar a realidade de forma biunívoca, pois os nomes definem seus significados quando colocados no contexto da proposição.

[56] Neste caso, a possibilidade da troca de mensagens equivale a possibilidade de combinação/encaixe entre os objetos que compõe um estado de coisas possível.

A quantidade de possibilidades combinatórias de um objeto é potencialmente infinita, visto que podem existir infinitas instancias de objetos que podem se combinar com ele (mas a quantidade de combinações impossíveis é também, pelo mesmo motivo, potencialmente infinita).

> "A figuração pode afigurar toda a realidade cuja forma ela tenha."
> (TLF, 2.171)

Dentro das combinações possíveis, a significação do objeto dentro da proposição irá variar de acordo com a forma com a qual irá se acoplar aos outros componentes da proposição. Após o acoplamento os componentes da proposição adquirem um significado único, passam a representar um determinado estado de coisas, portanto, uma linguagem perfeita possui uma relação biunívoca única entre uma determinada proposição e um determinado estado de coisas, esta relação se torna única quando os nomes são estruturados em uma proposição, nomes podem ser dúbios, mas nomes estruturados não.

> "Agora parece possível especificar a forma proposicional mais geral: ou seja, dar uma descrição das proposições de uma notação qualquer, de modo que cada sentido possível seja exprimível por um símbolo a que a descrição convenha e cada símbolo a que a descrição convenha possa exprimir um sentido, desde que os significados dos nomes sejam convenientemente escolhidos"
> (TLF, 4.5)

No aforismo 1.2 do TLF Wittgenstein define o mundo como um conjunto de fatos.

> "O mundo resolve-se em fatos"
> (TLF, 1.2)

Os fatos podem ocorrer em grupos, estados de coisas, e estes podem ser representados por pensamentos e proposições que possuem algo (uma forma lógica) em comum com o real estado de coisas.

> “A totalidade dos estados existentes de coisas é o mundo”
> (TLF, 2.04)

O mundo consistiria, portanto de todos os estados de coisas (combinações de fatos) possíveis e de fato atualizados (ocorridos).

A realidade consistiria de um espaço lógico que abrange todos os estados de coisas possíveis (inclusive os não ocorridos) e possui como subconjunto o mundo, o conjunto dos estados de coisas que são, alem de possíveis, de fato ocorridos.

> “A existência ou não existência de um estado de coisas é a realidade”
> (TLF, 2.06)

No pensamento haveria um espelho da realidade, uma recriação do espaço lógico de combinações possíveis.

> “A figuração lógica dos fatos é o pensamento”
> (TLF, 3)

Através da constatação empírica poderíamos separar no pensamento os estados de coisas apenas possíveis (pertencentes somente à realidade) e os estados de coisas possíveis e ocorridos (pertencentes à realidade e ao mundo).

A linguagem seria uma ferramenta de externalização da forma lógica do pensamento que por sua vez possuiria um mapeamento estrutural com a forma lógica dos estados de coisas da realidade e do mundo.

> “Na proposição o pensamento exprime-se sensível e perceptivelmente”
> (TLF 3.1)

4.3. O TLF e a Modelagem Orientada a Objetos.

Podemos estabelecer, aparentemente sem grandes danos, um interessante relacionamento entre os conceitos da modelagem OO e as idéias do TLF.

Alguns destes possíveis relacionamentos encontram-se expostos nas paginas subseqüentes.

4.3.1. Diagrama de Classes.

> "Um diagrama que exibe uma coleção de elementos declarativos (estáticos) de um modelo, em geral, classes e seus conteúdos e relacionamentos.[57]"
> OMG, 2001

Como já foi dito no capítulo anterior, o diagrama de classes pretende fornecer uma descrição estática de uma porção específica do mundo, de forma que, esta descrição seja aplicável em uma implementação em uma linguagem de programação orientada a objetos.

Os diferentes objetos deste MiniMundo encontram-se agrupados em classes que por sua vez possuem atributos e operações próprias e que se conectam umas as outras de forma a ilustrar as diferentes relações existentes entre estas classes.

> "Um nome toma o lugar de uma coisa, um outro, o de uma outra coisa, e estão ligados entre si, e assim o todo representa – como um quadro vivo – o estado de coisas"
> TLF 4.0311

O Diagrama de classes representa uma imagem de uma parte do mundo (MiniMundo) da mesma forma que Wiitgenstein sugere que o mundo consista de fatos que podem ser acoplados para formar estados de coisas e que proposições na linguagem são imagens destes fatos e estados de coisas. O diagrama de

[57] "A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships."

classes é um modelo estruturado que representa a estrutura do fenômeno correspondente de seu referente (neste caso, é indiferente se o referente é o mundo ou o pensamento, as representações mentais do mundo).

Para Wittgenstein, este escopo (a combinação dos fatos em estados de coisas e dos estados de coisas em outros estados de coisas) compõe o mundo, ou pelo menos à parte do mundo sobre a qual é possível falar e pensar, já que sobre o impensável nós não podemos falar.

> "A totalidade das proposições é a linguagem"
> (TLF, 4.001)

Para Wittgenstein, tudo o que pode ser pensado pode ser dito e apenas o que pode ser dito pode ser pensado.

> "A proposição é uma figuração da realidade. A proposição é um modelo da realidade tal como pensamos que seja."
> (TLF, 4.01)

Existe algo em comum que faz com que a linguagem adquira esta capacidade de espelhar o mundo.

> "Os limites da minha linguagem significam os limites do meu mundo"
> (TLF, 5.6)

Este aforismo sugere que só poderemos captar do mundo aquilo que nosso espelho proposicional (linguagem) estiver apto a representar, da mesma forma, um diagrama de classes possui uma capacidade limitada de representação de um MiniMundo, expansível através da expansão da notação do modelo.

4.3.2 - Classes.

> "Uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, métodos, relacionamentos e semântica.[58]"
> (OMG, 2001)

> "Modelagem Orientada a Objetos envolve mecanismos de abstração utilizados para descrever a estrutura comum de fenômenos similares. A declaração de uma classe representa uma abstração de substancia.[59]"
> (Nygaard, 1986)

Como já foi razoavelmente demonstrado no capítulo anterior e recolocado através das duas citações acima, uma classe consiste na abstração de uma série de atributos e operações comuns a um determinado grupo de objetos, que a partir desta abstração podem ser considerados instancias de sua classe (a classe é a generalização do objeto e o objeto é a instancia da classe).

> "O principal elemento de um modelo Orientado a Objetos é a classe, que representa uma coleção abstrata de objetos com algum conjunto em comum de propriedades. Os componentes de um sistema são modelados em objetos que são classificados como membros de uma classe. A partir de cada classe nós podemos instanciar objetos que representam uma "simulação" no sistema dos componentes que foram modelados na classe.[60]"
> (Andersen,1997)

---

[58] "A description of a set of objects that share the same attributes, operations, methods, relationships, and semantics." (M.T.)

[59] "Object-oriented modeling involves abstraction mechanisms used to describe the common structure of similar phenomena. The declaration of a class represents an abstraction of substance" (M.T.)

[60] "The main element of an objected-oriented model is the class, which represents an abstract collection of objects with some (for the purpose suitable) common set of properties. Components of a system (i.e., part of the world) are modeled into objects that are in turn classified as members of a class. Of each class we can instantiate objects that represent a "simulation" of the components in the system that we have modeled " (M.T.)

Quando o desenvolvedor de sistemas recebe a incumbência de modelar um determinado sistema em um diagrama de classes, a tarefa inicial será procurar por objetos com algum conjunto de características comuns que juntas correspondam a uma descrição generalizada dos objetos em questão.

Aquilo de faz de uma classe uma classe, dentro de um determinado sistema, parece estar claramente expresso no seguinte aforismo do TLF.

> "Ou uma coisa possui propriedades que nenhuma outra possui, podendo-se então, sem mais, destacá-las das outras por meio de uma descrição e indicá-la; ou, pelo contrário, há várias coisas que possuem todas as suas propriedades em comum, sendo então impossível apontar para uma delas.
>
> Pois se uma coisa não é distinguida por nada, não posso distingui-la, pois, caso contrario, ela passaria a estar distinguida."
> (TLF 2.02331)

O conjunto de coisas com um conjunto comum de atributos constitui uma classe, uma coisa que não possua atributos em comum com outras constitui por si só uma classe onde o mecanismo de herança ainda não gerou variações.

Uma outra característica das classes e suas instancias (objetos) visível no corpo do TLF é a de que um símbolo pode ser substituído por outro que seja uma instancia da mesma classe que o símbolo original, isto faria com que fossem respeitadas as regras de possibilidades combinatórias dos elementos da linguagem.

> "O que designa no símbolo é aquilo que é comum a todos os símbolos pelos quais ele pode ser substituído de acordo com as regras da sintaxe lógica."
> (TLF, 3.344)

Esta asserção, sobre o agrupamento dos objetos poder ser definido de acordo com a possibilidade de substituição de uns pelos outros, irá constituir um interessante problema da engenharia de software orientada a objetos, conhecido como Liskov Substitution Principle que é devidamente explicitada no próximo capítulo.

4.3.3 - Atributos.

> "Uma característica de uma classe que descreve um conjunto de valores possíveis que instancias desta classe podem possuir.[61]"
> (OMG, 2001)

Qualquer objeto possui atributos, os atributos são o que diferenciam uma instancia de uma classe de outra, por exemplo, a classe mesa pode possuir os atributos material e número de pés:

Exemplo simplificado de uma classe mesa[62]:

```
public class Mesa
{
Material material;
int NumPes;
public Mesa(Material material, int NumPes)     //construtor
        {
        this.material=material;
        this.NumPes=NumPes;
        }
}
```

Exemplos de duas instancias da classe mesa:

```
Mesa AmesadaCozinha = new Mesa(madeira, 4);

Mesa AmesadaSala = new Mesa(metal, 6);
```

Wittgenstein descreve a questão dos atributos através dos atributos posicionais de um objeto no espaço alem de fornecer outros exemplos ilustrativos como o do som e o da cor.

> "O objeto espacial deve estar no espaço infinito. (O ponto do espaço é um lugar de argumento).

[61] "A feature within a classifier that describes a range of values that instances of the classifier may hold" (M.T.)

> Não é preciso, por certo, que a mancha no campo visual seja vermelha, mas uma cor ela deve ter: tem a sua volta, por assim dizer, o espaço das cores. O som deve ter uma altura, o objeto do tato, uma dureza, etc."
> (TLF 2.0131)

O estado do objeto pode variar de acordo com o valor de seus atributos, por exemplo, se possuo uma instancia da classe humano que possui um atributo estomago, e altero o valor do atributo estomago (provavelmente através do método **comer()**), de vazio para cheio, ocorreu uma alteração no estado do objeto, o valor dos atributos pode mudar ao longo de nossas vidas, como o valor de nosso atributo idade, que define nossos estados em quanto jovens, velhos, crianças etc...

> "O objeto é o fixo, subsistente; a configuração é o variável, instável."
> (TLF 2.0271)

A configuração, no aforismo acima representa os atributos e seus estados, a estrutura do objeto subsiste, sua configuração varia através da alteração do valor de seus atributos.

Em sistemas orientados a objeto, devido ao encapsulamento dos atributos, que impede sua manipulação externa, a alteração do estado objeto deve ser solicitada ou sugerida através de mensagens enviadas aos métodos correspondentes do objeto.

4.3.4 - Operações.

> "Um serviço que pode ser requisitado à um objeto para ativar um comportamento. Uma operação possui uma assinatura que deve restringir os parâmetros possíveis.[63]"
> (OMG, 2001)

---

[62] A sintaxe utilizada para os exemplos em código é a da linguagem Java, unicamente por ser a linguagem Orientada a Objetos com a qual possuo maior familiaridade.

[63] "A service that can be requested from an object to effect behavior. An operation has a signature which may restrict the actual parameters that are possible." (M.T.)

Todos os objetos do mundo possuem relacionamentos, expressos através de sua capacidade de agir sobre outros objetos ou ser utilizado por eles, como visto no capítulo anterior, no paradigma OO este agir sobre se dá através da chamada aos métodos públicos do objeto, estes métodos (públicos ou não), constituem o conjunto de operações do objeto.

> "A operação mostra-se numa variável; ela mostra como, a partir de uma forma de proposições, se pode chegar a uma outra."
> (TLF, 5.24)

Como afirmado há pouco, a correta significação de uma proposição parece ser determinada no momento em que seus objetos componentes se encontram através da troca de mensagens entre suas respectivas operações (métodos públicos), neste momento a proposição passa a representar um fato possível.

> "A ocorrência da operação não caracteriza o sentido da proposição
> Pois a operação não enuncia nada, apenas seu resultado o faz, e este depende das bases da operação.
> (Não se pode confundir operação e função)."
> (TLF, 5.25).

O aforismo o 5.25 do TLF permite duas aproximações interessantes, a primeira diz respeito ao fato de que a ocorrência da operação, por si só não caracteriza o sentido da proposição. O fato de um objeto possuir uma determinada possibilidade de acoplamento em uma proposição não faz com que o comportamento (função) derivado do envio de algum tipo de mensagem para esta proposição ocorra sem que a mensagem seja efetuada. A possibilidade não garante a ocorrência, é preciso uma causa para qualquer conseqüência, e mesmo que esta conseqüência seja possível, a causa é necessária.

A segunda aproximação diz respeito a distinção entre operação e função, a operação parece dizer respeito as possibilidades públicas de acoplamento de um objeto, enquanto a função esta relacionada ao efeito do acoplamento, ao efeito provocado por uma determinada chamada a uma determinada operação.

É também curioso observar a relação destes termos do TLF com as diferentes perspectivas do diagrama de classes descritas por FOWLER em seu livro e citadas no capítulo anterior.

Na perspectiva de especificação, ficam claros os métodos públicos existentes, as operações do objeto e a forma de ativá-las através de mensagens. No entanto, apenas na perspectiva de implementação, surgem com clareza os comportamentos que serão implementados em cada forma de acoplamento possível.

Uma operação pode possuir muitas funções, que irão variar de acordo com a mensagem passada a esta operação.

4.3.5 - Associações.

> "O relacionamento semântico entre dois ou mais classes que especificam conexões entre suas instancias"
> (OMG, 2001)

A associação entre duas ou mais classes indica que os objetos instanciados a partir destas classes possuem uma certa relação entre si.

> “A configuração dos objetos constitui o estado de coisas”
> (TLF, 2.0272)

Como foi dito acima, apenas a definição dos acoplamentos pode definir qual das possíveis funções irá emergir das operações disponíveis.

> “No estado de coisas os objetos se concatenam, como os elos de uma corrente.”
> (TLF, 2.03)

> “Nos estados de coisas os objetos estão uns para os outros de uma determinada maneira.”
> (TLF, 2.031)

No espaço lógico total (a realidade no sentido do TLF), os objetos podem se acoplar de diversas maneiras, no mundo (estado de coisas atualizado), ou em uma proposição já proferida, esta multiplicidade se resume a uma única maneira pela qual os objetos encontram-se relacionados.

4.4 – Uma possível critica ao modelo Orientado a Objetos através das "Investigações filosóficas" de Wittgenstein.

O ataque às pretensões da modelagem Orientada a Objetos através do "segundo" Wittgenstein encontra-se bem representada no artigo do professor HOLMBOE[64] (Holmboe, 2004).

A hipótese central defendida por seu artigo é oposta à deste trabalho, pois segundo seu ponto de vista, o paradigma de desenvolvimento de software Orientado a Objetos não estaria mais bem relacionado com nossa estrutura cognitiva nativa (*Natural Thinking*) do que qualquer outra forma de projeto, modelagem e implementação de sistemas.

Para validar esta hipótese, o professor HOLMBOE utiliza a comparação entre o TLF e a segunda grande obra de Wittgenstein, As Investigações Filosóficas, onde, segundo seu artigo, Wittgenstein renunciaria a uma linguagem logicamente perfeita reconhecendo sua forçosa artificialidade e sua incapacidade em abstrair a estrutura do mundo e da linguagem natural.

> "O Tractatus Lógico Philosophicus, foi originalmente publicado em 1921. Nesta obra, Wittgenstein expõe um conjunto de postulados sobre o uso da linguagem para descrever o mundo. De um ponto de vista matemático, ele prove uma descrição idealizada de uma linguagem logicamente perfeita.
>
> Neste artigo eu demonstro como a modelagem orientada a objetos pode ser vista como um exemplo desta linguagem logicamente perfeita.

[64] Christian Holmboe is Senior Knowledge Engineer and Training Manager at Computas A/S. He holds degrees in Pedagogy and Computer Science, and specializes in the cognitive aspects of visual data modeling. He has recently spent several years as a Research Fellow and Associate Professor at the University of Oslo, Norway, teaching and leading research projects in Computer Science Education. Holmboe is the author of several research papers on the learning of software engineering as a collaborative activity and the understanding of object oriented programming. He still holds a position at the University of Oslo as visiting professor.

> Sua segunda maior publicação foi a obra Investigações Filosóficas (1958). A visão mais pragmática da linguagem apresentada no segundo livro ajuda a explicar a aparente contradição entre a suposta naturalidade do pensamento Orientado a Objetos e os problemas documentados nos processos de aprendizagem.[65]"
> (Holmboe, 2004)

O argumento principal do artigo do professor HOLMBOE é a tese de que a inflexibilidade da modelagem orientada a objetos a distanciaria inevitavelmente da estrutura cognitiva "natural" dos seres humanos.

Segundo seu artigo, Wittgenstein teria percebido isto em sua segunda grande obra e reparado o erro cometido no TLF, reconhecendo que uma linguagem logicamente perfeita não poderia possuir vínculos estruturais com uma realidade plural, fluida e prenhe de paradoxos (imperfeita portanto, do ponto de vista da lógica), mas retratável através de uma linguagem também plural, fluida e infestada de paradoxos como nossa linguagem natural.

Creio que a tese do professor HOLMBOE esteja correta se o alvo da modelagem da linguagem formal for o mundo como um todo; neste caso, a exclusão de dubiedades e paradoxos criaria um distanciamento insuportável entre representante e representado.

No entanto, sob um diferente ponto de vista, defendido por esta dissertação, o professor HOLMBOE não levou em conta o fato de que linguagens especificas não pretendem efetuar modelos genéricos, mas abstrair do todo (mundo), determinados grupos de aspectos que possam ser representados, isolados do resto, através de linguagens formais especificas.

Além disto, creio que foi subestimada uma das mais importantes características da OO, o polimorfismo, característica que permite que um objeto somente assuma uma determinada identidade durante o estabelecimento do contexto no qual se encontra.

---

[65] "Tractatus Lógico philosophicus, was originally published in 1921. In this book, Wittgenstein outlines a set of postulates about the use use of language to describe the world. From a mathematical point of view, he gives an idealized description of a logically perfect language. In this paper I demonstrate how object oriented modeling can be seen as an example of such logically perfect language.
Wittgensteins´s second major publication is the Philosophical Investigations (1958). The more pragmatic view of language presented in the latter book helps explain the apparent contradiction between the assumed naturalness of OO-thinking and problems documented in the learning process..." (M.T.)

Através deste recurso seria possível conjugar a complexidade da realidade ou de um aspecto dela e sua posterior representação pela linguagem formal que pretende representa-la.

Apesar da discordância no que diz respeito à tese central do artigo, os argumentos do professor HOLMBOE e os vínculos estabelecidos entre o TLF e os conceitos da orientação a objeto são precisos e preciosos para que observemos a força de suas ocorrências na historia do pensamento.

Capítulo V – Discussão

O objetivo deste capítulo é apresentar algumas questões diretamente relacionadas com a programação e o desenvolvimento orientados a objetos e a importância destas questões para a elegibilidade dos conceitos da OOP como instrumentos para modelar o comportamento de processos cognitivos específicos.

5.1 – Programando no Grande x Programando no Pequeno.

Os termos “Programming in the large” (PIL) e “programming in the Small” (PIS) surgem no início da década de 80 com um artigo homônimo de DeRemer e Kron (1975).

Com o tempo, as idéias deste artigo tornaram-se conceitos chave para a compreensão da necessidade da existência de diversas técnicas diferentes de analise, modelagem e implementação.

O termo “Programando no pequeno” refere-se a um tipo de programação que é executada por uma ou poucas pessoas e que produz um código que uma pessoa (geralmente aquela que programou) pode entender.

O sentido de “entender” no parágrafo acima refere-se a capacidade de realizar modificações sem gerar conseqüências inesperadas e sem a necessidade de buscar auxilio em uma documentação externa. O programador pode utilizar algum tipo de documentação para compreender melhor os requisitos do sistema, saber melhor aquilo que o usuário deseja, mas não precisa de documentação para compreender como o código implementa estes requisitos, a leitura do código é suficiente para sua compreensão.

O termo “Programando no Grande” refere-se a um tipo de programação executado por grandes grupos de pessoas ou por pequenos grupos durante um grande período de tempo. Esta forma de programar produz um tipo código que não pode ser compreendido sem uma estratégia do tipo “dividir para conquistar”.

Quando a meta é “Programar no Pequeno”, o principal objetivo é gerar um código limpo que possa ser facilmente compreendido, esta é uma forma de suportar mudanças.

Na “Programação no Grande” a ênfase se encontra em dividir o trabalho em módulos cujas interações serão precisamente especificadas. Este objetivo requer planejamento e documentação cuidadosos.

Uma vez que os módulos e suas interações parecem estar bem definidos, o trabalho pode começar nos módulos individuais, este trabalho pode ser “Programação no Pequeno” ou “Programação no Grande”, dependendo da complexidade de cada módulo.

Na “Programação no Grande”, mudanças podem ser complicadas. Se a mudança atravessar as fronteiras entre os módulos, o trabalho de vários profissionais poderá ter que ser refeito. Por este motivo, um dos objetivos da “Programação no Grande” é criar módulos que não precisem ser alterados quando prováveis mudanças forem executadas.

> "...Estruturar uma grande coleção de módulos para formar um sistema é uma atividade essencialmente distinta e intelectualmente diferente da atividade de construção dos módulos individuais.[66]"
> (DeRemer e Kron, 1975)

Programar no Pequeno requer a habilidade de escrever código que irá efetuar as ações desejadas, que outros possam ler e que não seja muito complicado para realizar alterações.

Programar no Grande requer uma grande capacidade de abstração. Até que seja implementado, uma abstração é tudo o que um módulo é. Vista como um todo, uma abstração de um sistema deve criar uma arquitetura que pareça atender os requisitos do sistema com uma quantidade mínima ou nula de alterações, alem de definir interações entre os módulos que sejam precisas e demonstravelmente corretas.

Programar no Grande requer habilidades de gerenciamento. A questão da abstração não consiste apenas em descrever um sistema funcional, mas inclui também o esforço e a capacidade de dirigir corretamente os esforços dos indivíduos envolvidos no desenvolvimento do trabalho.

---

[66] “...Structuring a large collection of modules to form a system is an essentially distinct and different intellectual activity from that of constructing the individual modules.”
(DeRemer e Kron, 1975)

O surgimento das técnicas de desenvolvimento Orientadas a Objetos confluem para a idéia de integrar estes dois aspectos do desenvolvimento de sistemas, através da UML[67] e das técnicas de modelagem OO é possível criar abstrações de diversos níveis, variando por exemplo de uma perspectiva conceitual (Completamente abstrata) para uma perspectiva de implementação (Híbrida).

Conectando as técnicas de modelagem da UML com as técnicas análogas de programação OO (disponível em inúmeras linguagens que suportam este modelo de programação) é possível acoplar fortemente os dois mundos.

Através das ferramentas CASE (Computer Aided Software Enginiering) disponíveis, como por exemplo o Rose da Rational Systems, empresa dos três idealizadores da UML (Booch, Rumbaught e Jacobson) é possível alterar simultaneamente o código e o modelo tanto a partir de um como a partir do outro, desta forma os mundos da "Programação no Grande" e da Programação no Pequeno parecem conectar-se de uma forma inédita na história do desenvolvimento de sistemas e da modelagem lógica de aspectos do mundo real[68].

Além da grande vantagem do gerenciamento simultâneo das duas perspectivas (Grande e Pequeno), o paradigma da programação Orientada a Objetos fornece também outras vantagens para o desenvolvimento de sistemas que possuam simultaneamente aspectos "Grandes" e "Pequenos", a primeira delas é um modelo definido de taxonomia, uma forma de efetuar de forma rigorosa a divisão do minimundo em módulos e a divisão dos módulos em submódulos e assim por diante.

Naturalmente, a OOP surge como uma técnica de modelagem, e diante disto se caracteriza nitidamente como um instrumento de "programming in the large", no entanto, nunca existiu a pretensão de abandonar as técnicas algorítmicas de "programming in the small", apenas demonstrar a existência da necessidade das duas perspectivas e da comunicação entre elas.

[67] U.M.L, Unified Modeling Language ou Linguagem de Modelagem Unificada.

[68] Através das ferramentas de modelagem UML disponíveis no mercado, é possível derivar código (Java, c++, SmallTalk etc...) a partir de um diagrama de classes, no entanto, não é possível gerar o código interno a cada método, este "recheio" dos métodos deve ser inserido a moda antiga, mantendo nítida a separação entre "programming in the large" e "programming in the small".

5.1.2 – Continuo x Discreto

Atualmente, entre os integrantes da comunidade de desenvolvimento de software existe um consenso tácito de que as técnicas de programação e modelagem orientadas a objetos possuem uma utilidade e um direcionamento voltados para as macroestruturas do sistema, enquanto o funcionamento interno dos métodos pertencentes aos objetos seria descrito através de métodos algorítmicos tradicionais.

Possuiríamos portanto, de acordo com este ponto de vista, uma visão não contínua do processo de modelagem e implementação de um modelo computacional da realidade. Este modelo seria discreto devido ao fato de que sua estrutura se altera à medida que nos movemos das estruturas gerais (macro) para as estruturas especificas (micro).

Um exemplo prático desta mudança encontra-se nas ferramentas de apoio ao desenvolvimento de sistemas que oferecem uma visão integrada da camada de modelagem e do código propriamente dito, como por exemplo o ROSE da empresa RATIONAL fundada pelos criadores da forma definitiva da UML.

O ROSE é uma ferramenta visual de modelagem onde o desenvolvedor cria os diagramas de seu sistema, uma das características desta ferramenta (e de outras ferramentas análogas), é permitir que o diagrama de classes seja automaticamente transformado em um código equivalente, composto em uma linguagem predefinida que ofereça suporte à programação orientada a objetos como SIMULA, SmallTalk, JAVA ou C++ entre outras.

No entanto, a ferramenta só automatiza a transformação de modelo em código no que se refere as estruturas gerais do sistema, as classes, os atributos e os limites externos dos métodos. A implemantação da ação interna dos métodos, aquilo que eles de fato fazem, é deixada para que o programador implemente manualmente, sem o auxilio da ferramenta de modelagem.

Este funcionamento reflete a noção comum de que algoritmos devem ser procedurais e de que a redução de um sistema aos objetos que o compõe possui um limite definido pela descrição das ações implementadas pelos métodos dos objetos.

Uma visão contrária a esta é apresentada por Wittgenstein em seu TLF (Wittgenstein, 1993), Wittgenstein afirma que os objetos aos quais temos acesso

através da observação de um determinado estado de coisas, são objetos complexos, formados pela união de diversos outros objetos menos complexos e assim por diante, até que se chegue na unidade mínima de composição da realidade nomeada por Wittgenstein de objetos simples.

Segundo Wittgenstein os objetos simples não seriam cognitivamente acessíveis, mas existiriam como uma necessidade lógica de oferecer um fundamento à realidade.

Segundo Wittgenstein, a estrutura do mundo seria continua e invariante até seu fundamento ultimo, o objeto simples, mantendo a estrutura de objetos tanto no macro (estado de coisas) quanto no micro (objetos simples).

> "A proposição mais simples, a proposição elementar, assegura a existência de um estado de coisas"
> (TLF, 4.21)

Neste aforismo, Wittgentein afirma que os elementos mínimos dos objetos complexos são também objetos, inexistindo portanto, atributos ou métodos que não sejam, eles mesmos objetos.

Acredito que as duas visões, a visão do TLF e a visão da engenharia de software atual possam ser complementares e não auto-excludentes no sentido em que podemos considerar o próprio algoritmo como um objeto complexo, composto por sinais proposicionais (objetos simples) e componentes de um objeto ainda mais complexo.

Desta forma estaríamos ampliando o significado do termo objeto para abrigar formas distintas do objeto padrão da engenharia de software, que teoricamente deveria conter atributos e métodos.

Se aceitarmos os termos da proposição como objetos simples e os métodos dos objetos tradicionais como sub objetos, poderíamos conciliar as duas visões e afirmar que o OOP se aplicaria tanto a "programação no pequeno" quanto à programação no grande.

Por outro lado, se nos mantivermos presos a concepção atual de objetos, os sinais proposicionais simples e os métodos não poderia ser considerados objetos e desta forma observaríamos uma descontinuidade estrutural na descida das

estruturas maiores para as estruturas menores, passando do paradigma Orientado a Objetos para uma estrutura algorítmica seqüencial tradicional.

5.2 - Os objetos absorvem as operações.

A distinção das perspectivas (grande e pequena) consiste em uma importante conseqüência do surgimento da OOP, mas outras questões derivadas apresentam-se com igual força e importância, por exemplo a premissa de que se pode modelar o mundo apenas com objetos. E quais seriam as conseqüências que isto poderia trazer.

Tradicionalmente, de forma mais intuitiva nas lógicas Aristotélica e clássicas e formalmente, a partir da lógica de predicados de GOTTLOB FREGE (Frege, 1978) é estabelecida uma distinção entre funções e objetos.

Cabe neste ponto detalhar um pouco o que pretendo designar com estes termos de difícil definição.

> "Ainda não está fora de dúvida qual seja a referência que tem, em Análise, a palavra "função", apesar de seu uso freqüente há muito tempo. Encontramos, nas explicações deste termo, ocorrerem sempre uma ou outra das seguintes expressões, ou ambas conjuntamente: expressão do cálculo e variável"
> (Frege,1978)

Dentro da disciplina da engenharia de software e do desenvolvimento de sistemas, uma função pode ser definida em termos de uma expressão de um cálculo, que pode conter variáveis locais e em alguns casos acessar variáveis globais; recebe um conjunto de 0 a N parâmetros e retorna, de acordo com o resultado dos cálculos que a compõem aplicados aos parâmetros de entrada, um valor de saída, geralmente booleano ou numérico.

As linguagens de programação tradicionais, como o VB Script ou o Pascal possuem bibliotecas de funções onde agrupam-se funções de vários gêneros, quando se deseja realizar uma operação simplesmente "chama-se" a função passando a ela parâmetros, geralmente variáveis já existentes no programa.

Por exemplo, em VB Script, se desejamos extrair a raiz quadrada de uma soma procederíamos da seguinte maneira:

S = 3 + 2
R = Sqrt(S)

Ou seja, a variável S recebe o valor da soma de dois tipos primitivos[69] (3 e 2) e este valor resultante (armazenado em S) é utilizado como parâmetro para uma função pré existente, que se agrupo geralmente em uma biblioteca do sistema, junto com outras funções pouco ou nada relacionadas a ela.

A OOP propõe que todas as funções sejam internalizadas pelos objetos, passando a estarem agrupadas de acordo com as características e possibilidades especificas do objeto, por exemplo, a operação acima, em um pseudo-código Orientado a Objetos ficaria da seguinte maneira:

Int S = New Int(2 + 3)
R= S.sqrt()

No segundo exemplo, a função Sqrt não se encontra mais em uma biblioteca de funções que recebe uma variável primitiva como parâmetro (um objeto do tipo inteiro com valor 5), neste caso, o número não é mais uma variável primitiva, é uma instancia da classe número e a função Sqrt é um método da classe número assim como voar é um método da classe coruja.

Voar(Coruja) X Coruja.Voar()

A idéia, contrária às definições da lógica de predicados tradicional, possui importantes conseqüências e funda o paradigma da Orientação a Objetos no momento em que afirma que qualquer função deve pertencer a um objeto, possuindo assim, um domínio, uma identidade com os outros métodos (funções) que também façam parte do objeto, em uma linguagem orientada a objetos não devem existir tipos primitivos, tudo são classes.

Críticos da linguagem JAVA, utilizam freqüentemente o argumento de que JAVA não é uma linguagem totalmente orientada a objetos pelo fato de possuir alguns traços característicos das linguagens procedurais como por exemplo, a

[69] Tipos primitivos são elementos nativos de linguagens de programação que não possuem uma estrutura de objetos (N.A.).

existência da classe Math, que segundo os detratores não é realmente uma classe, mas uma biblioteca de funções disfarçada.

A razão da existência desta biblioteca é a existência de tipos primitivos em JAVA, um deles seria o tipo referente aos números inteiros, como um tipo primitivo não é uma classe e não pode possuir métodos, surge a necessidade de agrupar estes métodos que deveriam estar na classe correspondente em uma biblioteca "genérica" de funções. Desta forma, o código correspondente em JAVA ao exemplo acima ficaria da seguinte maneira:

```
Int S = (2+3);
Math.Sqrt(S);
```

O que se assemelha mais ao paradigma procedural do que ao paradigma Orientado a Objetos, apesar da chamada nominal a classe biblioteca Math.

Segundo os defensores do JAVA que defendem na integra a OOP, o motivo desta "falha" é uma troca de rigor por performance, uma vez que tipos primitivos são processados mais rapidamente do que objetos instanciados a partir de classes.

Uma referencia direta a esta questão pode ser encontrada no artigo "OOP Meets Álgebra, Suggesting New "Ways of Looking"" de KIRBY URNER (Urner, 1999).

> "Como a forma de pensar Orientada a Objetos afeta os já bem definidos conceitos algébricos, como por exemplo, os quaternions[70]? Afirmar que um quaternion é um tipo de objeto não soa muito controverso, mas nós devemos nos lembrar que "objetos" não são apenas dados, mas também definem operações. O que os quaternions podem fazer, são parte de sua natureza.
>
> Esta forma de pensar é um pouco diferente de escrever um quaternion como [s,v] e explicar que s refere-se a parte escalar e V refere-se a parte do vetor. Fazer isto seria dar conta apenas dos atributos do quaternion ignorando seus dados, suas operações, e dando uma idéia semelhante à exposta abaixo:

q.s = 1.0

q.v = 3.0

Onde q é uma instancia de um objeto quaternion com atributos s e v. Mas e quanto aos métodos?

Nos livros tradicionais, os operadores flutuam de forma semi-independente de seus argumentos. q1 * q2 (dois quaternions multiplicados) implica em uma operação (multiplicação) que aceita dois quaternions como entrada. Em uma linguagem de programação pre-OOP nós escreveríamos:

q3 = mult(q1,q2);

Mas no modelo Orientado a Objetos, nós possuímos a opção de pensar nas operações como parte da definição dos objetos. Ao invés de escrevermos q3 = mult(q1,q2) nós escreveríamos:

q3 = q1.mult(q2)[71]"

(Urner,1999)

Esta inversão da álgebra tradicional , internalizando os métodos nos objetos correspondentes resolve e simplifica diversos problemas de modelagem e representação de cenários (estados de coisas) e modelos simples ou complexos. A própria noção de que os operadores fazem parte dos objetos, exclui temporariamente a difícil questão a respeito da origem dos operadores, de seu lócus e da natureza de sua relação com os objetos.

---

[71] quaternions? To say "a quaternion is a kind of object" oes´t sound very controversial, but we need to remenber that "objects" aren´t "just data" but also define operations. What quaternions do is part of what they are.This way of thinking is a little different from writing a quaternion as [s,v] and explain s is the "scalar part" and v the "vector part". So far, that´s just explaining a data format, giving an idea like:
q.s = 1.0;
q.v = 3.0;
where q is some quaternion object with properties (fields) s and v. But what about methods?
In standart text books, operators float semi-independly of their arguments. q1*q2 (two quaternions multiplied) implies an operation (multiplication) that accepts two quaternions as input. In pre-OOP computer language, we might have written: q3 = mult(q1,q2); But in the OOP model, we have the option to think of operations as part of the definition of the objects themselves. Instead of writing q3 = mult(q1,q2), we instead write: q3 = q1.mult(q2); "

5.2.1 – Um efeito colateral da modelagem OO.

Este artifício, da inclusão dos operadores nos objetos, possui no entanto alguns efeitos colaterais não muito agradáveis com os quais a engenharia de software e os defensores do modelo OOP vem lutando com o objetivo de minimizar os problemas e maximizar as vantagens.

Um dos problemas apresentados é a existência de funções ou métodos que, ou não possuam um domínio definido ou não se encaixem automaticamente em um objeto e a existência de funções ou métodos que se encaixem simultaneamente no domínio de diversos objetos, gerando uma desnecessária redundância.

É interessante lembrar que nas linguagens pré-OOP estes problemas não existiam, pois, o comportamento padrão era agrupar todas as funções em uma única biblioteca cuja identificação entre os métodos agrupados era o fato de serem todos métodos, independente de suas outras características.

Desta forma, surgiram os novos problemas descritos acima, simultaneamente com a solução de outros problemas que foram resolvidos quando os objetos absorvem seus métodos.

Aparentemente, a anterior inexistência destes problemas ocorria pelos seguintes motivos:

a) Um método não precisava possuir uma identificação nítida com o comportamento de um objeto especifico.

b) O agrupamento comum de todos os métodos evitava a redundância de métodos semelhantes ou idênticos internalizados por objetos diferentes.

Visto de outro ponto de vista, mais próximo da filosofia da linguagem e da mente, o problema parece soar semelhante a critica do professor HOLMBOE ao modelo OOP já visto no capítulo anterior.

Resumidamente, o argumento utilizado é que um modelo OOP ortodoxo, onde predomina a crença de que TODOS os elementos de um MiniMundo possam ser modelados como objetos, atrapalharia, devido a suas próprias regras, a geração de modelos consistentes, pois o representado (o mundo) não parece ser composto apenas de objetos, seus atributos e seus métodos. Por isto, segundo HOLMBOE, WITTGENSTEIN constataria nas INVESTIGAÇÔES FILOSÓFICAS (ano), que o mundo não seria redutível a um modelo gerado por uma linguagem formal, apenas pela linguagem natural, que compartilharia do caráter informal do representado (o mundo).

Podemos pensar nesta argumentação como um contraponto a idéia de URNER descrita acima, que propõe que os operadores não "flutuem de forma semi independente", mas sejam absorvidos pelos objetos aos quais pertençam.

Quando trocamos:

A=Sqrt(x,y) por A=x.sqrt(y)

Com a mudança, ganhamos clareza, pois passamos a conhecer melhor a origem do método Sqrt() e não geramos nenhum tipo de redundância, pois o mecanismo de Herança garante que não precisemos implementar o método mais de uma vez para que seja herdado por todas as subclasses.

No entanto, existem métodos ainda mais genéricos que o Sqrt(), e a não ser que utilizemos um instrumento de herança comum a todas as classes, como a classe *OBJECT* do JAVA, o que por sua vez causa outros problemas, será necessário que o método seja implementado mais de uma vez gerando redundância e ferindo o principio da simplicidade. Um exemplo muito utilizado na literatura especializada em desenvolvimento de software é o do método Log[72]() que ou se encontra em uma superclasse comum a todas as outras (como no mecanismo do JAVA), ou possui diversas implementações idênticas (gerando

redundância), ou precisa de uma terceira solução para que o problema possa ser resolvido da melhor forma possível.

A herança comum do JAVA não é aceitável como solução por que o que ela faz é reunir todos os métodos que podem ser utilizados por todos os objetos em uma grande biblioteca de funções sem um domínio definido, típica das linguagens anteriores a orientação a objetos. Esta crítica é bem parecida com a critica a classe *math* feita acima, mas a classe *math* pretende justificar-se devido a melhor performance dos tipos primitivos e a herança comum devido a necessidade de evitar a redundância.

Cria-se portanto um paradoxo entre as normas da OOP e as necessidades de modelagem impostas pelo próprio referente dos modelos (o mundo), ou agrupamos funções em bibliotecas disfarçadas de classes, sem um domínio definido, ou somos obrigados a duplicar código desnecessariamente.

5.2.2– Algumas soluções possíveis para o problema das operações aritméticas entre objetos.

Estabelece-se então, a seguinte questão, o que são e a que domínio pertencem os operadores?

Esta questão é largamente discutida tanto na filosofia analítica quanto na engenharia de software, e as soluções empregadas por diferentes linguagens de programação divergem grandemente mesmo quando se trata de duas linguagens que suportam e implementam o paradigma orientado a objetos.

Uma discussão recorrente é levantada pelos defensores da linguagem C++ em detrimento da linguagem JAVA.

[72] Do Inglês LOGGED, Lançar no Diário de Bordo (Dicionário Oxford Inglês Português,1996, Oxford University Press), significa, dentro do jargão da engenharia de software, manter registros atualizados sobre as ações do sistema. No caso da OOP, manter controle sobre as ações de cada objeto.

> "Programadores experientes que aprendem Java costumam ficar consternados ao descobrir que eles não podem aplicar operadores aritméticos e relacionais a operandos que forem instancias de classes criadas pelo programador. Java não permite que o programador utilize simultaneamente o paradigma orientado a objetos e expressões ordinárias da sintaxe.
> Alguns acreditam que esta limitação seja um sério impedimento, enquanto outros acreditam que seja apenas uma mera inconveniência notacional. Alguns poucos chegam a defender que este fato seja uma positiva proteção contra o mau uso das operações! Independente de nossas opiniões, programadores continuam precisando realizar operações aritméticas.[73]"
> (Weisert, 1997)

Por exemplo, em C++ seria possível escrever o seguinte trecho de código:

```
Date aDataInicial, aDataFinal;
ADataFinal = aDataInicial + 14;
```

Este trecho de código não poderia ser escrito em JAVA, pois contraria algumas regras implícitas da Orientação a Objetos, a primeira destas regras nos diz que, por definição, nada existe fora do domínio dos objetos. E a segunda diz que apenas o próprio objeto deve manipular seus atributos, um operador que não faça parte do objeto iria ferir o conceito de encapsulamento.

O principal argumento a favor do encapsulamento é o fato de que sem ele torna-se muito difícil controlar coerentemente o estado dos objetos e impedir que operações indesejadas ou até mesmo impossíveis (devido a desencontros de tipos) ocorram.

Qual será o objeto resultante da soma de uma maçã e um mosquito? As operações entre números funcionam sempre bem, porque são operações entre membros da mesma classe. Quando realizamos mentalmente operações entre

[73] "Experienced programmers learning Java are dismayed to discover that they can't apply arithmetic and relational operators to operands that are instances of the classes they define. Java won't allow a programmer to use both the object paradigm and ordinary expression syntax on the same data items.
Some of us regard that limitation as a serious impediment, while many others view it as a mere notational inconvenience. A few even consider it a positive protection against misuse of operations! Whatever our opinion, application programmers still need to do arithmetic on application-domain data." (Minha tradução)

objetos utilizamos mecanismos cognitivos de controle destas operações, quando passamos a implementar objetos e classes em um programa de computador, precisamos criar mecanismos que conjuguem a necessidade de efetuar as operações entre objetos e a necessidade de impedir que a ação de operadores que não conhecem a estrutura dos objetos provoquem efeitos indesejados.

a) A solução dos tipos primitivos:

A solução mais simples aventada em JAVA para este problema foi a inserção de tipos primitivos na linguagem.

Tipos primitivos são elementos do programa que não são objetos instanciados a partir de classes, como por exemplo os inteiros ou as datas.

> "Por que não utilizar apenas tipos de dados primitivos da linguagem Java para dados numéricos simples, i.e. esqueça os benefícios da orientação a objetos para dados de domínios específicos como dinheiro, datas, distancias e pesos?[74]"
> (Weisert, 1997)

O problema desta solução é que ela fere gravemente a primeira regra formulada acima, a regra que afirma que TUDO em um sistema OO deve estar dentro do domínio dos objetos, o que não é o caso dos tipos primitivos, esta solução preserva a integridade dos objetos de fato ao custo da integridade da linguagem que torna-se um híbrido procedural+OO.

b) A solução da exposição do atributo.

Uma variação mais interessante desta solução é converter, em tempo de execução, objetos em tipos primitivos e vice-versa, desta forma trabalharíamos com objetos que se comportariam como tipos primitivos durante uma operação aritmética.

[74] "Why not use only Java's built-in primitive data types for elementary numeric data, i.e. forgo the benefits of object-orientation for application-domain data2 like amounts of money, dates, distances, and weights?" (Minha Tradução)

Esta solução é superior a anterior porque nela, menos elementos estão fora do domínio dos objetos (pelo menos durante a maior parte do tempo). Nesta solução, os operadores continuam não fazendo parte do domínio de nenhum objeto, mais os operandos são instancias de classes que se transformam em tipos primitivos no momento da operação voltando em seguida a seu estado anterior.

O procedimento principal desta solução é isolar temporariamente o valor de um atributo do objeto em si, e isto fere igualmente o conceito de encapsulamento.

> "Esta técnica repudia a idéia central da orientação a objetos. Na verdadeira OOP nós não consideramos o valor de um objeto como algo distinto do próprio objeto.[75]"
> (Weisert, 1997)

c) A solução da substituição dos operadores universais por métodos locais.

A terceira solução consiste em abrir mão dos operadores aritméticos tradicionais em favor de métodos pertencentes aos objetos, desta forma poderíamos possuir a seguinte classe cadeira:

Classe Cadeira
Encosto -> Atributo
Pés -> Atributo
Sentar() -> método
Add() - > Método (Equivalente ao operador "+")

Onde o método Add() substitui o operador +, desta forma podemos realizar a operação de soma entre um objeto do tipo Cadeira e outro objeto.

Cadeira aCadeira = new Cadeira(); // Criação do primeiro objeto

[75] "this technique repudiates what object-oriented programming is about. In true OOP we don't consider the value of an object as something distinct from the object itself." (Minha Tradução)

```
Cadeira aSegundaCadeira = New Cadeira; // Criação do segundo objeto
ACadeira.Add(aSegundaCadeira); // Soma dos dois objetos sem a utilização
 //do operador aritmético "+"
```

Esta solução preserva o encapsulamento dos atributos dos objetos e não fere a regra da inexistência de funções fora dos objetos, mas gera uma obvia redundância de operadores, uma vez que muitos objetos deverão possuir um método Add(). Podemos considerar, como uma forma de amenizar esta redundância, o fato de que as implementações destes métodos nos objetos seriam diferentes e que a redundância estaria apenas no fato de que todos podem ser somados, mas que a operação em si, seria diferente, possuiria peculiaridades em cada um deles (objetos).

Acredito que do ponto de vista de uma compreensão cognitivista dos conceitos do paradigma Orientado a Objetos, esta solução seja extremamente válida, uma vez que diferentes objetos executam as operações aritméticas de forma peculiar, como a água e a areia. Outro ponto interessante é que a implementação dos métodos responsáveis pelas operações aritméticas parecem variar de acordo com o valor dos atributos do objeto, como por exemplo o metal sólido e o metal derretido. Se levarmos esta variação em consideração cada objeto possuiria não uma, mas muitas implementações dos métodos correspondentes as tradicionais operações aritméticas.

5.3 – A modelagem Orientada a Aspectos.

A modelagem Orientada a Aspectos visa resolver um outro aspecto indesejável da modelagem Orientada a Objetos pura, a redundância de métodos universais, utilizados por instancias de varias classes que não possuem, obrigatoriamente, uma superclasse em comum (O que nos forneceria uma solução para o problema através do mecanismo de herança) .

A modelagem orientada a aspectos procura resolver a questão através do "desencapsulamento" de métodos cuja utilização seja comum a um grande número de classes.

> "Podemos encontrar diversos problemas para os quais nem as técnicas de programação procedurais nem as técnicas Orientadas a Objeto são suficientes para capturar, com clareza, algumas das importantes descrições de modelagem que o programa deve implementar. Isto força a tomada de algumas decisões de modelagem que ficaram espalhadas pelo código, resultando em um código emaranhado cuja manutenção é, geralmente, bastante difícil para desenvolver e manter. Nós apresentamos uma analise da razão pela qual certas decisões de modelagem tem se mostrado tão difíceis de capturar com clareza utilizando o código atual. Chamaremos as propriedades às quais estas decisões se referem de aspectos, e mostraremos que a razão pela qual elas se mostram tão difíceis de capturar é o fato de que elas atravessam horizontalmente as funcionalidades básicas do sistema.
>
> Nós apresentamos as bases para uma nova técnica de programação, chamada de Programação Orientada a Aspectos, que torna possível expressar com clareza programas que envolvem tais aspectos, incluindo o devido isolamento, composição e reuso do código dos aspectos.[76]"
> (Kiczales, Lamping et al, 1997)

A proposta da Programação Orientada a Aspectos é isolar do diagrama de classes tradicional todos os métodos que fizerem parte de todos, ou pelo menos de um grande número de classes do sistema.

Estes métodos (como o método Log() descrito acima por exemplo) seriam então declarados aspectos e separados das classes propriamente ditas, o modelo contemplaria três mundos diversos, os aspectos, as classes e os aspectos nas classes, que seria gerados por uma camada intermediaria que possui a função de "costurar" aspectos e classes em um único produto final.

Apesar do produto final (costurado) conter tanto as classes quanto os aspectos, tanto a modelagem quanto a implementação e a manutenção do sistema ganham em simplicidade e expressividade com a separação.

Do ponto de vista teórico, a programação Orientada a Aspectos pode ser considerada uma autocrítica a radicalidade da inversão da lógica de predicados Fregeana, assumindo que nem todos os métodos (funções) presentes no mundo podem ser internalizados em um objeto à cujo domínio pertença.

---

[76] We have found many programming problems for which neither procedural nor object-oriented programming techniques are sufficient to clearly capture some of the important design decisions the program must implement. This forces the implementation of those design decisions to be scattered throughout the code, resulting in "tangled" code that is excessively difficult to develop and maintain. We present an analysis of why certain design decisions have been so difficult to clearly capture in actual code. We call the properties these decisions address aspects, and show that the reason they have been hard to capture is that they cross-cut the system's basic functionality. We present the basis for a new programming technique, called aspectoriented programming, that makes it possible to clearly express programs involving such aspects, including appropriate isolation, composition and reuse of the aspect code. The discussion is rooted in systems we have built using aspect-oriented programming." (Minha Tradução)

Sob um outro ponto de vista, podemos considerar outra forma de resolver o problema atacado pela Programação Orientada a Aspectos.

Se considerarmos que todos os objetos possuam uma herança comum, como por exemplo a super classe “Object” em JAVA, e possuirmos a possibilidade de manipular esta superclasse para podermos incluir atributos e métodos que desejarmos tornar globais, podemos, através do mecanismo de herança, resolver o problema da redundância de métodos globais.

No entanto, a utilização de uma superclasse comum a todos os objetos não é, necessariamente mais correta do que a criação de estruturas de objetos com origens distintas e aparentemente as duas soluções se equivalem do ponto de vista técnico, embora a solução da superclasse possa ser considerada mais elegante do ponto de vista teórico, que possui como objetivo principal criar modelos do mundo com o máximo de possível de semelhança estrutural entre o modelo e seu referente.

5.4 – A modelagem através de protótipos.

Outra crítica interessante a programação orientada a objetos deriva do fato de que parecem existir famílias de objetos que não possuem uma estrutura única comum, desta forma não seria possível construir uma única superclasse para todos os membros da família.

> "De acordo com a visão clássica, nenhum membro de uma classe deveria possuir um status especial em relação aos outros, uma vez que todos compartilham as mesmas propriedades. Rosch imaginou diversas experiências que mostram claramente que a realidade não corresponde à visão clássica - Alguns membros ou subcategorias são melhores exemplos de uma categoria e são chamados de protótipos.[77]"

[77] “According to the classical view no member of a category should have any special status, since they all share the same properties. Rosch designed a number of experiments that clearly showed that this is not the

(Personen, 2001)

Este problema, de que nem sempre se pode atribuir com certeza uma determinada categoria a um objeto, ou de que determinados objetos possam ser classificados sem problemas em duas ou mais categorias diferentes parece bastante semelhante aos problemas apresentados pela LÓGICA FUZZY, apresentada introduzida pelo professor LOFTI A. ZADEH[78] em seu artigo sobre FUZZY SETS ou conjuntos difusos publicado em 1965.

> "Um conjunto difuso é uma classe de objetos com um continuo de graduações de pertinência. Este conjunto é caracterizado por uma função de pertinência que retorna, para cada objeto, um nível de pertinência entre 0 e 1 ...[79]"
>
> (Zadeh, 1965)

Os conjuntos difusos ou confusos do professor ZADEH, são conjuntos cuja definição não possui valores absolutos mas apenas valores relativos como nos mostram seus próprios exemplos.

> "Mais freqüentemente do que seu oposto, as classes encontradas no mundo físico real não possuem um critério de pertinência definido precisamente.

---

case – some members or subcategories are better examples of a category and are hence called prototypes." (Minha Tradução)

[78] Until 1965, Dr. Zadeh's work had been centered on system theory and decision analysis. Since then, his research interests have shifted to the theory of fuzzy sets and its applications to artificial intelligence, linguistics, logic, decision analysis, control theory, expert systems and neural networks. Currently, his research is focused on fuzzy logic, soft computing, computing with words, and the newly developed computational theory of perceptions and precisiated natural language. (The Berkeley Initiative in Soft Computing)

[79] "A fuzzy set is a class of objects with a continuum of grades of membership. Such a set is characterized by a membership (characteristic) function which assigns to each object a grade of membership ranging between zero and one..." (Minha Tradução)

> Por exemplo, a classe de todos os animais claramente inclui cachorros, cavalos, pássaros etc..., e claramente exclui objetos como pedras, fluidos, plantas etc... Entretanto, objetos como estrelas do mar, bactérias etc... possuem um status ambíguo no que diz respeito à classe dos animais. O mesmo tipo de ambigüidade surge no caso de um numero como 10 em relação à "classe" dos números reais que são muito maiores do que 1.
>
> Claramente, a "classe de todos os números reais maiores que 1" ou a "classe das mulheres bonitas", ou a "classe dos homens altos", não constituem classes e conjuntos no sentido matemático usual destes termos.
>
> Ainda assim, permanece o fato de que estas "classes" definidas imprecisamente possuem um importante papel no pensamento humano, particularmente no domínio do reconhecimento de padrões, comunicação da informação e abstrações.[80]"
>
> (Zadeh, 1965)

Como podemos observar nos exemplos acima, os FUZZY SETS são conjuntos cuja função de pertinência varia de acordo com referenciais que não estão fixos, podendo ser redefinidos subjetivamente, permitindo que diferentes objetos possuam diferentes níveis de pertinência a estes conjuntos.

A existência de conjuntos definidos de forma relativa e não absoluta parece ser uma característica essencial da linguagem natural, que permite sua fantástica elasticidade, e de outros processos cognitivos humanos, como o reconhecimento de padrões. No entanto, a principal diferença entre a lógica dos conjuntos difusos

[80] "More often than not, the classes of objects encountered in the real physical world do not have precisely defined criteria of membership.
For example, the class of animals clearly includes dogs, horses, birds, etc. as its members, and clearly excludes such objects as rocks, fluids, plants, etc. However, such objects as starfish, bacteria, etc. have an ambiguous status with respect to the class of animals. The same kind of ambiguity arises in the case of a number such as 10 in relation to the "class" of all real numbers which are much greater than 1.
Clearly, the "class of all real numbers which are much greater than 1" or "the class of beautiful women", or "the class of tall men", do not constitute classes or sets in the usual mathematical sense of this terms.
Yet, the fact remains that such imprecisely defined "classes" play an important role in human thinking, particularly in the domain of pattern recognition, communication of information and abstractions." (Minha Tradução)

e a programação orientada a protótipos é o fato de que na programação orientada a protótipos até mesmo um conjunto com uma função de pertinência bem definida pode possuir membros que possuam status variados de pertinência, como veremos a seguir com o exemplo da classe avó, a definição básica da classe avó é a do conjunto avó tal que avó={x : x mãe y e y mãe/pai z}, ou seja, o conjunto avó contêm com igual grau de pertinência todas as mulheres cujas filhas ou filhos possuam filhos.

No entanto, existem outras características que definem os membros da classe das avós, como por exemplo, a existência de cabelos brancos e outras características comuns, desta forma, um membro da classe pode ser um representante mais adequado da classe do que outro, permitindo a existência de diversos níveis de pertinência, assim como ocorre na lógica fuzzy.

Apesar deste possível complemento, a programação orientada a protótipos também lida com conjuntos cujos elementos podem não possuir nenhuma intercessão comum, sendo o conjunto definido pelas intercessões entre membros isolados.

Personen (2001), aponta algumas das principais diferenças entre a orientação a objetos e a modelagem através de protótipos:

a) A separação entre condições suficientes e condições necessárias para ser membro de uma classe.

b) A existência de diferentes níveis de participação em uma classe.

c) A categoria pode ser representada por um membro ideal, ou protótipo, mas nem sempre um membro da categoria possuirá todos os atributos necessários para ser considerado um membro ideal.

d) A possibilidade de existência de casos dúbios e pouco claros, objetos que podem ou não fazer parte de uma determinada categoria

e) A modelagem através de protótipos geralmente apresenta uma estrutura de semelhança de família.

f) Membros de uma categoria podem possuir características indefinidas dentro daquela categoria.

> "Exemplos dos efeitos dos protótipos podem ser encontrados em diversas categorias. Pombos são entes mais representativos da categoria PASSAROS do que galinhas, ou pingüins, cadeiras de mesa são melhores exemplos da categoria CADEIRA do que cadeira de balanço ou cadeira elétrica. Cabelo grisalho e estilo de vida domestico e pacifico caracterizam muitas avós, mas Elizabeth Taylor pode ser uma avó, embora não possua nenhuma destas características.[81]" Personen(2001)

Ao contrario da modelagem Orientada a Aspectos, o problema levantado pela modelagem orientada a protótipos parece resistir a outras formas de abordagem fundamentadas no mecanismo de herança e na existência de uma superclasse comum.

Em relação ao problema levantado pela orientação a protótipos, a existência de uma superclasse comum parece ser, justamente a essência do problema, ou seja, o fato de que nem sempre, todos os membros de uma determinada classe possuem uma interseção estrutural (atributos e métodos) que permitam que ambos herdem características de uma mesma superclasse.

Desta maneira, a classe seria definida pela semelhança entre seus membros que possuiriam identidades estruturais com outros membros da mesma classe mas não com todos eles.

[81] "Examples of prototype effects are found in many categories. Robins are more representative of the category BIRD than are chickens, or penguins, desk chairs are better examples of a CHAIR than rocking chairs or electric chairs. Gray hair, domestic and peaceful lifestyles characterize many grandmothers, but Elizabeth Taylor can still be a grandmother, although she does not have these properties." (Minha Tradução)

O problema central da programação orientada a aspectos se assemelha muito com a objeção levanta por HOLMBOE a identificação entre a OOP os processos cognitivos humanos. HOLMBOE utiliza como fundamento de sua argumentação sua leitura das Investigações Filosóficas (Wittgenstein, 1994), onde Wittgenstein expõe sua opinião em relação aos integrantes da categoria "Jogos", demonstrando que, aparentemente, não existem um conjunto de características comuns a todos os jogos, mas que uns se assemelham com outros e assim por diante criando uma estrutura de semelhança de família.

5.5 – Possíveis contribuições dos conceitos do paradigma da orientação a objetos para a construção de modelos cognitivos.

Após este breve passeio pelos conceitos da programação Orientada a Objetos resta esclarecer os possíveis vínculos entre estes conceitos e aspectos do aparato cognitivo.

Que aspectos são estes?

Acredito que estes aspectos representem a fração de nossas capacidades cognitivas responsável pela taxonomia da realidade, pela divisão e estruturação da realidade em unidades mínimas e combinações destas unidades.

A maturidade da dúvida epistemológica elimina a esperança ingênua de que um determinado critério taxonômico possa capturar todos os entes da realidade sem gerar dúvidas ou contradições, no entanto, percebemos também que determinadas divisões de categorias respondem melhor do que outras a determinados pontos de vista em relação à realidade. Aparentemente, diferentes necessidades requerem diferentes divisões e encaixes e diferentes divisões e encaixes requerem diferentes formas de representação.

Baseado na investigação realizada acima acredito que a analise da modelagem orienta a objetos possa servir como base para compreendermos a forma como estruturamos uma de nossas taxonomias mentais da realidade,

admitindo a pluralidade de pensamentos e linguagens ao invés de pensamento e linguagem como no titulo do livro de Vigotski (Vigotski, 2005).

Uma aplicação prática destes conceitos pode ser imaginada como um analisador semântico de estruturas de objetos, um processo que possa processar uma proposição como um recorte tradicional e seja capaz de instanciar objetos a partir de suas unidades lexicais, aumentando drasticamente seu campo semântico e permitindo que analisemos a compatibilidade dos elementos da proposição sob um ponto de vista privilegiado em relação à analise sintática tradicional.

Um analisador sintático tradicional seria capaz de aferir valores de possibilidade (P) ou impossibilidade (I) a frases com as seguintes estruturas:

Ivo viu a uva (P)

Ivo viu a viu (I)

A + B (P)

A + * (I)

Um analisador sintático tradicional pode recorrer a um catalogo de funções sintáticas e regras de formação de um determinado sistema formal para determinar que um operador deve estar sempre ladeado por operandos, ou que um verbo não pode ser o predicado de uma proposição.

No entanto, um analisador sintático tradicional não poderia distinguir entre:

Ivo viu a uva (P) e

Ivo ouviu a uva (P)

De acordo com as regras sintáticas de formação de proposições não existe nenhum impedimento na formação da segunda proposição, no entanto, um ser humano, de posse de uma capacidade de analise semântica da proposição saberia que não é possível (a não ser em sentido figurado), ouvir a uva, uma vez que produzir sons não é uma função da classe uva, de sua superclasse fruta ou de suas instancias, as uvas. E desta forma, uma instância da classe uva não poderia ser utilizada como parâmetro para o método ouvir pertencente a Ivo, uma instancia da classe pessoa, que possui entre seus métodos a possibilidade de ouvir objetos que produzam algum tipo de som.

Acredito que, se formos capazes de instanciar objetos a partir dos elementos da proposição, poderemos analisar a possibilidade ou impossibilidade da proposição a luz da analise das possibilidades combinatórias dos objetos.

Uma analise orientada a objetos da proposição Ivo viu a uva procederia da seguinte maneira:

Ivo -> Instancia da classe Pessoa.

Viu -> Método da classe Pessoa. (A operação é internalizada no objeto de acordo com a já discutida inversão da álgebra)

A uva -> Instancia da classe Uva

Exemplo de implementação utilizando a sintaxe da linguagem JAVA[82].

```
Pessoa Ivo = new Pessoa();

Uva aUva = New Uva();

Ivo.Ouvir(aUva) - (I) O objeto aUva não pode ser utilizado como parâmetro para o método ouvir, uma vez que não possui os atributos sonoros necessários.
```

[82] Por comodidade e clareza.

> Ivo.Ver(aUva) – (P) O objeto aUva pode ser utilizado como parâmetro para o método Ver, uma vez que possui os atributos espaciais necessários.

Em um analisador de proposições que utilize como critério de possibilidade da proposição as possibilidades combinatórias dos objetos envolvidos, uma proposição impossível (resultado da interação de objetos incompatíveis), resultaria em algum tipo de erro de formação antes imperceptível pela mera analise sintática, proporcionando um aumento na capacidade semântica de processos automatizados de percepção de sentido em uma proposição.

A definição da possibilidade de um estado de coisas ou da proposição que o representa através da analise das possibilidades combinatórias de seus elementos encontra-se exposta também no Tractatus Lógico-Philosophicus (Wittgenstein, 1993).

> "O estado de coisas é uma ligação de objetos (coisas)."
>
> (TLF, 2.01)

Assim como, o próprio objeto é também um estado de coisas composto por outros objetos mais simples.

> "É essencial para a coisa poder ser parte constituinte de um estado de coisas"
>
> (TLF, 2.011)

Todo objeto deve possuir em si a possibilidade de combinação com outros objetos, caso contrário não poderá fazer parte do espaço lógico (realidade) ou do mundo.

> "Na lógica nada é casual: se a coisa pode aparecer no estado de coisas, a possibilidade do estado de coisas já deve estar prejulgado na coisa."
>
> (TLF, 2.012)

> "...Se as coisas podem aparecer em estados de coisas, isto já deve estar nelas.
>
> (O que é lógico não pode ser meramente possível. A lógica trata de cada possibilidade e todas as possibilidades são fatos seus.)
>
> Assim como não podemos de modo algum pensar em objetos espaciais fora do espaço, em objetos temporais fora do tempo, também não podemos pensar em nenhum objeto fora de sua possibilidade de ligação com outros.
>
> Se posso pensar no objeto na ligado estado de coisas, não posso pensar nele fora da possibilidade desta liga."
>
> (TLF, 2.0121)

As possibilidades de conexão de um determinado objeto com outros objetos em um estado de coisas ou na proposição que o representa, já se encontra pré estabelecido nas definições internas do próprio objeto, desta forma, se conhecermos o objeto conheceremos também suas possibilidades combinatórias.

> "Se conheço o objeto, conheço também todas as possibilidades de seu aparecimento em estados de coisas.

> (Cada uma destas possibilidades deve estar na natureza do objeto.)
>
> Não se pode encontrar depois uma nova possibilidade."
>
> (TLF, 2.0123)

Uma nova possibilidade combinatória iria requerer uma alteração estrutural no objeto o que geraria um novo objeto.

> "Ou uma coisa possui propriedades que nenhuma outra possui, podendo-se então, sem mais, destacá-las das outras por meio de uma descrição e indicá-la; ou, pelo contrário, há várias coisas que possuem todas as suas propriedades em comum, sendo então impossível apontar para uma delas.
>
> Pois se uma coisa não é distinguida por nada, não posso distingui-la, pois, caso contrário, ela passaria a estar distinguida."
>
> (TLF, 2.02331)

Caso dois objetos diferentes possuam a mesma configuração estrutural, estes objetos são instancias da mesma classe, instancias da mesma classe podem possuir estados diferentes, mas suas possibilidades combinatórias são as mesmas, uma vez que são definidas por sua estrutura e não por seus estados.

Ao cogitar a hipótese de instanciar objetos a partir dos elementos proposicionais reforçamos a idéia já discutida anteriormente de que a divisão do mundo em objetos seja contínua e não discreta, ou seja, ao fracionarmos um determinado objeto complexo obteríamos outros objetos e assim por diante, desta forma, os componentes da proposição não seriam meros atributos de objetos como por exemplo nomes, mas objetos completos contendo todos os seus atributos e métodos, inclusive o nome.

Ao criarmos instancias de objetos a partir dos elementos da proposição passaríamos a lidar com símbolos complexos ao invés dos símbolos simples utilizados tradicionalmente. A partir de símbolos simples podemos inferir, através de tabelas de classificação, suas classes gramaticais e funções sintáticas e através destes atributos podemos verificar se a proposição obedece as regras de formação proposicionais vigentes, ou seja, podemos verificar se os elementos lexicais encontram-se dispostos de acordo com regras de formação válidas. No entanto, as regras de formação proposicionais sintáticas tradicionais permitem a criação de proposições com combinações entre objetos que não possuem, nos dizeres de Wittgenstein, a possibilidade de aparição em determinado estado de coisas.

Ao transformarmos nomes em objetos aumentando seu campo semântico podemos criar melhores regras de formação e de verificação de possibilidades combinatórias entre os elementos da proposição.

Ainda estaríamos trabalhando com a interpretação de símbolos, mas nossos novos símbolos objetos nos diriam muito mais coisas a respeito de si próprios do que nossos antigos símbolos atributos.

Conclusão

Após o caminho percorrido apresentam-se não apenas uma, mas diversas conclusões referentes aos diferentes capítulos que fundamentam a hipótese geral formulada e sua posterior conclusão obtida.

A hipótese principal consiste na idéia de que, através da utilização dos conceitos subjacentes a programação orientada a objetos, possamos modelar aspectos do aparato cognitivo humano que correspondam a estes aspectos.

A conclusão referente a esta hipótese é a de que os acima citados conceitos podem ser utilizados para que possamos criar modelos que aumentem a capacidade analítica de sistemas artificiais em referencia às possibilidades combinatórias dos elementos proposicionais e a consistência interna de uma proposição.

Esta capacidade é obtida através de um drástico aumento do campo semântico dos elementos da proposição (Como aumento do campo semântico entendo um drástico aumento da quantidade de informações disponíveis à respeito dos elementos proposicionais). Este aumento não deve ser apenas quantitativo, devido às ações de uma série de mecanismos inerentes ao paradigma orientado a objetos, como por exemplo, o encapsulamento dos atributos, a internalização dos métodos e sua divisão e distribuição determinados de acordo com os domínios de cada classe de objetos.

Esta conclusão geral é obtida através da hipótese de que o estudo da linguagem pode nos fornecer um precioso repositório de informações a respeito dos processos cognitivos por ela representados, e que este potencial independe de uma série de crenças, opções e decisões a serem tomados ao longo do caminho.

Algumas destas possíveis bifurcações estão expressas nos diferentes capítulos do trabalho:

No primeiro capitulo apresento uma visão dos aspectos epistemológicos referentes as teorias da mente e a inteligência artificial. A principal questão apresentada é a dúvida em relação a capacidade dos sistemas formais em abarcarem de forma completa e consistente as características mentais humanas.

No inicio do capitulo o termo mente é utilizado indistintamente como um rótulo que engloba características distintas e variadas, e são apresentadas as posições antagônicas do ponto de vista conhecido como inteligência artificial forte, que acredita na capacidade dos sistemas formais de descreverem todos os atributos e detalhes dos processos mentais e na possibilidade de reproduzir estes processos através desta captura e o ponto de vista dos dualistas de substancia, que consideram a linguagem insuficiente para a descrição e reprodução de aspectos metafísicos da mente.

Partimos então para uma posição intermediaria cujo principal representante hoje é o filosofo DAVID CHALMERS, e que afirma que o termo mente engloba características metafísicas inefáveis e características passiveis de uma descrição algorítmica através de linguagens formais.

Este trabalho se situa em uma posição intermediária que afirma que independente das diferenças existentes entre as naturezas materiais e estruturais dos aspectos mentais e de suas representações lingüísticas, o mero fato da linguagem possuir a capacidade de comunicar e representar estes processos a capacita como alvo de investigações que pretendam, indiretamente, definir regras e critérios que nos possibilitem criar modelos de processos cognitivos específicos.

O segundo capítulo explora alguns pontos de vista possíveis a respeito do relacionamento entre linguagem e pensamento, variando entre a identificação e a disjunção total entre eles, o objetivo deste capitulo é reforçar a hipótese de que independentemente do nível de relacionamento entre linguagem e pensamento é preciso que exista, no mínimo, uma identidade estrutural que permita que o representante represente o representado. Outra conclusão do segundo capítulo é a de que diferentes visões e perspectivas do mundo possuem representações mais ou menos acuradas em relação a linguagens especializadas em representa-las.

Esta investigação conduz à conclusão de que a linguagem natural possui uma grande abrangência em sua capacidade de representação mas possui pouco rigor nas representações permitindo interpretações dúbias e textura aberta, enquanto as linguagens formais possuem âmbitos estreitos de representação mas possuem a capacidade de representar o mundo permitindo uma mínima quantidade de dubiedades e interpretações.

O terceiro capítulo apresenta as linguagens de programação de computadores como linguagens formais e estabelece um recorte critico de sua historia que conduz ao paradigma de analise, modelagem e programação orientada a objetos. O terceiro capitulo apresenta também os principais conceitos da programação orientada a objetos e da principal linguagem de modelagem e representação do conhecimento orientada a objetos a UML.

O quarto capitulo busca subsídios na história do pensamento que ajudem a mensurar a força dos conceitos do paradigma orientado a objetos, com este objetivo, recorremos a uma das obras de LUDWIG WITTGENSTEIN, o Tractatus Logico-Philosophicus, obra cujo objetivo principal é mensurar as capacidade da linguagem natural e das linguagens formais para capturar e retratar o mundo.

Surpreendentemente, os pontos de contatos entre a TLF e os conceitos da orientação a objetos são sólidos e numerosos, o que reforça a hipótese da familiaridade dos conceitos OOP com algumas de nossas estruturas mentais.

A análise de outras recorrências históricas destes conceitos não foi possível no âmbito deste trabalho, mas fica como referencia para futuras incursões intelectuais, dentro do recorte proposto. O próximo autor a ser investigado seria o Aristóteles de As Categorias, onde Aristóteles descreve uma serie de critérios para a realização de taxonomias mentais cuja estrutura também parece possuir fortes vínculos com o paradigma da orientação a objetos.

O quinto capitulo, denominado “Discussão”, apresenta uma série de questões, observações e críticas ao paradigma orientado a objetos:

Por exemplo, a questão da continuidade versus descontinuidade, objetos dividem-se sempre em objetos, ou em algum momento abandonamos a estrutura de objetos e retornamos a manipulação de símbolos simples tradicionais?

A tradição e o senso comum da engenharia de software afirmam que em determinado momento, todos os programas retornam à estrutura algorítmica tradicional, lidando com símbolos simples e não objetos. Wittgenstein afirma o contrário defendendo a hipótese de que objetos complexos sejam sempre compostos por outros objetos também compostos, e assim por diante, até os objetos simples, fundamentos da substancia, mas ainda objetos.

Esta questão acabou revelando-se de grande relevância em relação a conclusão final e geral deste trabalho, a qual chegaremos dentro em breve.

Outra questão relevante presente no capitulo de discussão refere-se à inversão da álgebra proposta pelo modelo orientado a objetos. Esta idéia baseia-se em uma inversão do padrão estabelecido por classificações e visões de mundo mais tradicionais, onde o mundo encontra-se dividido entre objetos e funções, ou operações.

Do ponto de vista da aritmética, isto equivale à idéia de que os operadores possuem existência independente ou semi-indepenedente dos operandos que são, portanto, manipulados por funções externas a eles.

De um ponto de vista mais genérico, podemos depreender que, na visão tradicional, as operações existem de forma independente dos objetos que lhes servem de parâmetros.

Na engenharia de software, encontramos este pensamento tradicional na forma das bibliotecas de funções, onde as funções eram armazenadas independentemente do domínio dos objetos que lhes servem de parâmetros.

No paradigma orientado a objetos, todas as operações são internalizadas nos objetos, gerando uma divisão de mundo onde existem apenas objetos e suas respectivas configurações internas.

Esta mudança fornece uma série de novas vantagens, questões e problemas que são devidamente discutidos ao longo do capitulo.

A conclusão geral, do capitulo e da dissertação como um todo é a geração da hipótese de que todos os elementos lexicais tradicionais de uma

proposição podem ser interpretados como objetos ou partes de objetos, tornando a proposição uma combinação de objetos e não dos símbolos simples anteriores.

Os símbolos anteriores representavam partes do objeto ao qual pertencem, por exemplo, o substantivo João corresponde ao atributo nome (do tipo substantivo), pertencente ao objeto oJoão, que por sua vez é uma instancia da classe pessoa que encontra-se inserida em uma longa e detalhada taxonomia de objetos, possivelmente exprimível através de um diagrama de classes da UML.

A idéia de trabalhar com os objetos complexos inteiros e não só com um de seus atributos faz com que possamos mensurar de forma muito mais acurada as possibilidades combinatórias dos elementos da proposição, transformando uma habilidade sintática de verificação da consistência proposicional em uma analise com capacidades semânticas.

Após percorrer as críticas a OOP, muitas delas justas e pertinentes como a critica realizada pela Programação Orientada a Aspectos (AOP) e pela Programação Orientada a Protótipos (POP), não creio que os conceitos da OOP possam modelar completamente nossa capacidade de discernir entre uma combinação de objetos possível e uma combinação de objetos impossível, mas acredito que possamos através dela e de seus desdobramentos (como a AOP), nos aproximarmos de nosso objetivo capturando e modelando pontos de vistas e aspectos de nossa forma humana de captura da realidade.

Bibliografia:

ARENDT, Hanna, 1991, "**A Vida do Espírito**", Rio de Janeiro, RJ, Editora Relume Dumará.

BACKUS, John, 1978, "**Can Programming Be Liberated from the Von Neumann Style? A Functional Style and Its Algebra of Programs**", Communications of the ACM, August 1978,Volume 2, Number 8

BAIRD, Rodney, 2001, **The Saone-Rhone River Valley.** http://www.ancientroute.com/water/SaoneRhone.htm

Booch, G. (1994). **Object-Oriented Analysis and Design.** Benjamin/Cummings, second edition.

BORGES, Jorge Luis, 1999, "**Obras Completas Volume II**", Rio de Janeiro, RJ, Editora Globo.

BURKE, james, 1984, "**Faith in Numbers".** San Antonio, KLNR.

CARRUTHERS, Peter, 2001, "**The cognitive functions of language**", Department of Philosophy, University of Maryland.

CARVALHO, Luis Alfredo Vidal de, 2005, "**Datamining**", Rio de Janeiro, RJ, Editora Ciência Moderna.

CHAITIN, Gregory J., 2005, "**Meta Math! The quest for omega**", NY, Random House.

CHALMERS, D.J., 1996, **Does a rock implement every finite-state automaton?** Department of Philosophy University of Arizona.

CHALMERS, D.J., 1999, **A Computational Foundation for the Study of Cognition.** Department of Philosophy University of Arizona.

CHALMERS, David, 1995, "**Facing Up to the Problem of Consciousness",** *Journal of Consciousness Studies* 2(3):200-19.

CHURCH, Alonso, 1951, "**The Need For Abstract Entities**", American Academy of Arts and Sciences Proceedings n 80, pg. 100-113.

COHOON , J.P., DAVIDSON, J. W., 2002, "**C++ Program Design : An Introduction to Programming and Object- Oriented Design**", New York, McGraw - Hill 3 ed.
.

COOK, Steve e DANIELS, John, 1994, "**Designing Object Systems: Object-Oriented Modeling with Syntropy**," Prentice Hall.

COOPER, Richard, **Towards an Object Oriented Language for Cognitive Modeling.**

DæHLEN, Morten,2001, **Simula Research Laboratory Early days!** Simula Research Laboratory, Oslo.

DAWKINS, R., 1976, "**The Selfish Gene**". Oxford University Press.

DENNET, Daniel C., 1994, "**The Role of Language in Intelligence"**. Em What is Intelligence?, The Darwin College Lectures, Ed. Jean Khalfa, Cambridge, Cambridge Univ. Press.

DENNETT, Daniel, C., 1998, "**A Perigosa Idéia de Darwin**", Rocco

DENNETT, Daniel, C. 1991. "**Consciousness Explained**". Penguin Press.

DeREMER, Frank e KRON, Hans, 1975, "**Programming-in-the large versus programming-in-the-small",** Los Angeles, Califórnia, Proceedings of the international conference on Reliable software, pg: 114 - 121

DOURISH, Paul, 2004**, Software as an Embodied Phenomenon: Cognitive, Social and Cultural Aspects of Programs and Programming**. Proceedings of the 2004 IEEE Symposium on Visual Languages and Human Centric Computing

DIJKSTRA, E.W.,1961, "**On the Design of Machine Independent Programming Languages**", AMSTERDAM, STICHTING MATHEMATISCH CENTRUM.

DYKSTRA, W, 1983, "**The fruits of misunderstanding",**

EDMONDS, David e Eidnow, John, 2003, **O Atiçador de Wittgenstein: a história de uma discussão de dez minutos entre dois grandes filósofos**, Trad. Pedro Jorgenses Jr. Rio de janeiro, Editora Diffel.
FODOR, J.A. 1975. **The Language of Thought.** Harvard University Press.

FODOR, J.A. 1992. **The big idea: Can there be a science of mind?** Times Literary Supplement 4567:5-7 (July 3, 1992).

FOWLER, Martin, 2000, **UML Essencial**, São Paulo, SP, Editora BookMan.

FREGE, Gottlob, 1978, "Lógica e Filosofia da Linguagem", São Paulo, SP, Ed. Cultrix.

FUKS, Saul e LEGEY, F.L. Luiz, 1997, "**Algumas Considerações Sobre os Paradigmas Analítico e Sistêmico**", Rio de Janeiro, RJ, Editora Relume Dumará – COPPE/UFRJ.

GAMMA, Erich, HELM, Richard, JOHNSON, Ralph, VLISSIDES, J, 2000, **"Padrões de Projeto: Soluções Reutilizáveis de Software orientado a Objetos**". São Paulo, SP, Editora Bookman.

GUDWIN, Ricardo, R., 1997, **Linguagens de Programação**, Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação

HALMOS, Paul, R, 2001, "**Teoria Ingênua dos Conjuntos**", Rio de Janeiro, RJ, Editora Ciência Moderna"

HOFSTADTER, Douglas, 2001, "**Gödel, Escher, Bach: um Entrelaçamento de Gênios Brilhantes**.", São Paulo, SP, Editora Imprensa Oficial SP

HOLMBOE, Christian, 2004, **A Wittgenstein Aproach to the Learning of OO-Modeling**, Computer Science Education , Vol 14, N 4, PP 277 – 296, Taylor and Francis Group.

HOPCROFT, John E., ULLNAN, Jeffrey D., MOTWANI, Rajeev, 2002, "**Introdução à teoria de Autômatos, Linguagens e Computação**", Rio de Janeiro, RJ, Editora Campus.

KANT, Immanuel, 1974, "**Crítica da Razão Pura e outros escritos filosóficos**", Rio de Janeiro, RJ, Editora Abril Cultural, Coleção "Os Pensadores" n XXV

KAY, Alan, 1993, "**The early days of Smalltalk"**. ACM.

KICZALES, Gregor, LAMPING, John et al, 1997, "**Aspect-Oriented Programming**", in proceedings of the European Conference on Object-Oriented Programming (ECOOP), Finland. Springer-Verlag.

KOSSLYN, S, 1994, "**Image and Brain**", MIT Press.

LEGRAND, Gérard, 1987, "**Os Pré-Socráticos**", Rio de janeiro, RJ, Ed. Jorge Zahar.

LEIBNIZ, Gottfried Wilhelm, 1998, **Machina arithmetica in qua non additio tantum et subtractio sed et multiplicatio nullo, divisio vero paene nullo animi labore peragantur.** Tradução de Cleonice Narciso e Magda Lourenço realizada a partir da versão inglêsa do texto latino feita pelo Dr. Mark Holmes, David Eugene Smith (ed.), A Source Book in Mathematics, New York: Dover, 1959, pp. 173-181. Revisão de Olga Pombo.

LISKOV, Barbara e WING, Jeanette, 1994, "**A Behavioral Notion of Subtyping**", ACM Transactions on Programming Languages and Systems, Vol 16, No 6, November, 1994, 1811-1841.

LOFTING, C. J., 1999, "**Object Oriented Thinking : Pros and Cons"., ** Copyright © 1999 C. J. Lofting.

LOPES, Cristina Videira, 2003, "**Beyond AOP: Toward Naturalistic Programming**", proceedings of the OOPSLA Onward! track, 2003.

MACKEY, Karen, 1996, **What Is Object-Oriented Technology Telling Us About Ourselves**. Lockheed Martin Advanced Concepts Center

MARTIN Abadi e LUCA Cardelli, 1996 , **A theory of Objects**. Springer-Verlag New York
McGram-Hill, New York, NY, 2002.
MIAH ,Shajan , 1997, "***Critique of the Object Oriented Paradigm: Beyond Object-Orientation***".

MLODINOW, Leonard, 2004, "**A Janela de Euclides**", São Paulo SP, Ed. Geração Editorial.

MOORE, E. Gordon, 1965, "**Cramming more components onto integrated circuits**",

Eletronics, Volume 38, Number 8, April 19, 1965

NAGEL, Ernest James e NEWMAN, R. , 1998, "**A prova de Godel"**., Rio de janeiro, RJ, Editora Perspectiva

NYGAARD, K, 1986, "**Basic concepts in object oriented programming"**. SIGPLAN-Notices 128 – 132.

OMG, 2001, "**Unified Modeling Language Specification v1.4"**, Needham, MA, Object Management Group.

OTHERO, Gabriel de Ávila, MENUZZI, Sérgio de Moura, 2005, "**Lingüística Computacional Teoria & Prática**", São Paulo, SP, Parábola Editorial.

PENROSE, Roger, 1993, "**A mente nova do rei: computadores, mentes e as leis da física**", Rio de Janeiro, RJ, Editora Campus.

PERSONEN, Juha Petteri, 2001, "**Psychological Criticism of The Prototype-Based Object-Oriented Languages"**. University of Helsinki

PIAGET, Jean, 2002, "**Epistemologia Genética**", São Paulo, SP, Ed. Martins Fontes.

POAGE, Josh, 2002, "**The Development of the Jacquard Loom: Early History of Computer Data Storage".**

POINCARRÉ, Henri, 1998, **O Valor da Ciência**. Editora ContraPonto.

REED, Adam, 2003, **Object-Oriented Programming and Objectivist Epistemology: Parallels and Implications**, The Journal of Ayn Rand Studies 4, no. 2

RICHARDS, Martin, 2001, **Comparative Programming Languages**, Cambridge, University Computer Laboratory.

ROTMAN, Brian, 1993, "**The Ghost in Turing's Machine**", Stanford, California, Stanford University Press.

RYAN, Bob., 1991, "**Dynabook Revisited with Alan Kay".** Byte. vol 16.

Russell, B. (1964). *"**The Principles of Mathematics**".* Northampton: John Dickens & Co LTDA

SAMPAIO, Luiz Sergio Coelho de, 2001, "A lógica da diferença", Rio de Janeiro, RJ, Ed-UERJ.

SANTO AGOSTINHO, 1997, "**Sobre a pontencialidade da Alma**", Petrópolis, RJ, Editora Vozes.

SCHOPENHAUER, Artur, 1994, "**Sobre Livros e Sobre Leitura"**, Rio de Janeiro, RJ, Editora Paraula.

SEARLE, John, R. 1997, "**Libertad e Neurobiologia",** Éditions Grasset & Frasquelle.

SEARLE, John, R., 1998, "**O mistério da consciência**", Rio de Janeiro, RJ, Editora Paz e Terra.

SHAY David, 2005, "**On The Object-Oriented Version of The Zombie Argument and Its Implications on The Mind-Body Problem**" Cornell.

SOUZA, João Nunes de, 2002, "**Lógica para Ciência da Computação**", Rio de Janeiro, RJ, Ed. Campus.

URNER, Kirby, 1999, **OOP Meets Algebra,Suggesting New "Ways of Looking"**
W. Wulf, M.Shaw, L. Flon, and P. Hilfinger**, Fundamental structures of Computer Science**, Addison-Wesley, Reading, MA, 1980.

WADLER, Philip, 2000, "**Proofs are Programs: 19th century Logic and 21st Century Computing**", Avaya Labs.

WEEDWOOD, Barbara, 1995, **História Concisa da Lingüística**, Editora Parábola.

WEISERT, Conrad, 1997, "**Conventions for Arithmetic Operations in Java**"

WITTGENSTEIN, Ludwig, 1993, **Tractatus Logico-Philosophicus**, Tradução, apresentação e estudo introdutório de Luiz Henrique Lopes dos Santos, Introdução de BERTRAND RUSSEL, Ed. Universidade de São Paulo.

WITTGENSTEIN, Ludwig, 1995, "**Conferencia sobre Ética**", UFSC

WITTGENSTEIN, Ludwig, 1994, "**Investigações Filosóficas**", Petrópolis, RJ, Editora Vozes.

WOLFF, J Gerard, 1993, **COMPUTING, COGNITION AND INFORMATION COMPRESSION**. School of Electronic Engineering and Computer Science, University of Wales

WOODCROFT, Bennet, 1851, “**The Pneumatics of Hero of Alexandria”**. Taylor, Walter and Maberly, Londres.

VARELA, Francisco J., THOMPSON, Evan, ROSCH, Eleanor, 2003, “**A Mente Incorporada**”, Porto Alegre, RS, Ed. ArtMed.

VIGOTSKI, L.S., 2005, “**Pensamento e Linguagem**”. Rio de Janeiro, RJ, Ed. Martins Fontes.

ZADEH, L. A., 1965, “**Fuzzy Sets**”, Information And Control, 8, 338-353 (1965)