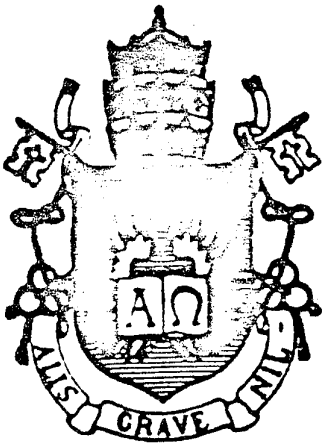


PUC

Série: Monografias em Ciência da Computação,
No.17/89

UMA TAXONOMIA PARA AMBIENTES DE SOFTWARE

Clovis T. Fernandes

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 – CEP-22453
RIO DE JANEIRO – BRASIL

Responsável por publicações:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC RIO, Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453 - Rio de Janeiro, RJ
BRASIL

Tel.: (021) 529-9386 TELEX: 31078 FAX: (021) 274-4546
BITNET: userrtlc@lncc.bitnet

PUC/RIO - DEPARTAMENTO DE INFORMÁTICA

Série: Monografias em Ciência da Computação, 17/89

Editor: Paulo Augusto Silva Veloso

julho, 1989

UMA TAXONOMIA PARA AMBIENTES DE SOFTWARE[1]

CLOVIS TORRES FERNANDES[2]

[1]Apresentado pelo Prof. Carlos José Pereira de Lucena

[2]De licença da Divisão de Ciências de Computação do ITA – Instituto Tecnológico de Aeronáutica; trabalho parcialmente financiado por CAPES/PICD.

RESUMO

Uma nova taxonomia para ambientes de software é apresentada: o modelo tridimensional. Este modelo é composto de três níveis. O primeiro refere-se ao nível dimensional representando três pontos de vista: interação entre Inteligência Artificial e Engenharia de Software, cobertura de paradigmas metodológicos e escala (com relação ao número de usuários e tamanho do sistema). Cada ponto de vista possui um conjunto de valores ou estados em que pode variar. A tripla <estado$_j$, estado$_k$, estado$_l$ >, onde estado$_j$ pertence à primeira dimensão, estado$_k$ à segunda e estado$_l$ à terceira, constitui um dentre os possíveis cubos que podem modelar ambientes de software. O segundo nível refere- se aos mecanismos de instanciação, que consiste em dividir o grupo de ambientes pertencendo a cada cubo em classes menores. O terceiro nível refere-se ao modelo geral de ambientes, que consiste em estabelecer os componentes políticas, mecanismos e estruturas para cada instância. O modelo é estendido para um modelo n-dimensional e a estrutura dos níveis é abstraída em uma meta–taxonomia para taxonomias de ambientes de software. O modelo tridimensional proposto pode ser obtido desta meta–taxonomia por instanciação.

Palavras-chaves:

taxonomia, ambientes de software, ambientes, meta–taxonomia, engenharia de software

ABSTRACT

A new taxonomy for software environments is presented: the 3–dimensional model. This model is composed of three levels. The first one refers to the dimensional level representing three points of view: interaction between Artificial Intelligence and Software Engineering, covering of methodological paradigms and scale (concerning number of users and system's size). Each point of view presents a value or state set in which it can vary. The triple <state$_j$, state$_k$, state$_l$ >, where state$_j$ belongs to the first dimension, state$_k$ to the second dimension and state$_l$ to the third dimension, constitutes one out of many possible cubes which can model software environments. The second level refers to the instantiation mechanisms, which consists of dividing the group of environments belonging to each cube in smaller classes. The third level refers to the general model of environments, which consists of establishing the components politics, mechanisms and structures for each instance. The model is extended to a n– dimensional model and the structure of the levels is abstracted to a meta–taxonomy for software environment taxonomies. The proposed 3–dimensional model may be obtained from this meta– taxonomy by instantiation.

Keywords:

taxonomy, software environments, environments, meta–taxonomy, software engineering.

SUMÁRIO

1 Introdução

O objetivo deste trabalho é propor uma nova taxonomia para a área de ambientes de software. Não é um objetivo apresentar uma classificação extensiva dos ambientes atuais usando esta nova taxonomia. Ambientes representativos serão usados apenas para exemplificar o uso das classes e das categorias taxonômicas apresentadas.

Inicialmente comparou-se algumas taxonomias encontradas na literatura, usando-se alguns princípios que foram derivados exclusivamente para tal tarefa. Com base nesse trabalho de análise, uma nova taxonomia é proposta: o modelo tridimensional. Como o próprio nome diz, têm-se três dimensões que procuram enfocar três pontos de vista: escala, interação de IA (Inteligência Artificial) e ES (Engenharia de Software) e cobertura dos paradigmas metodológicos. Cada ponto de vista apresenta um conjunto de estados ou valores em que pode variar. A tripla <estado$_j$, estado$_k$, estado$_l$ >, onde estado$_j$ pertence à primeira dimensão, estado$_k$ à segunda e estado$_l$ à terceira, constitui um dos possíveis cubos que podem modelar (ou classificar) ambientes de software.

O modelo tridimensional apresenta três níveis. O primeiro deles, refere-se aos cubos. O segundo deles refere-se ao mecanismo de instanciação, que consiste em dividir o grupo de ambientes de cada cubo em classes menores. O terceiro deles refere-se ao modelo geral de ambientes que consiste em estabelecer os componentes políticas, mecanismos e estruturas para cada instância.

A estruturação do modelo tridimensional pode ser abstraída numa meta-taxonomia. O modelo geral desta meta- taxonomia consiste de três níveis, onde o primeiro pode constar de 1 ou mais dimensões e sempre deve ser referenciado; o segundo é o mecanismo de instanciação , que é opcional, e o terceiro é o do modelo geral para cada instância, que também é opcional mas, se aparecer, deve vir acompanhado do segundo nível.

Com o intuito de evitar confusão, apresento a definição de **ambiente de software** empregada neste trabalho: ***coleção de ferramentas de hardware e de software colocadas à disposição do usuário para construção de sistemas de software.*** A amplitude desta definição tem uma explicação. O objetivo aqui é analisar taxonomias existentes e apresentar uma nova, de forma que qualquer definição mais restritiva poderia conflitar com as taxonomias examinadas. Sendo assim, não será necessário apresentar definições do tipo seguinte: ambientes de programação, ambientes de desenvolvimento de software, assistentes de programação baseados em conhecimento, ambientes 'CASE' e software assemelhado. Todos serão tratados como ambientes de software ou simplesmente **ambientes.**

Na segunda seção, examina-se a área de taxonomia em geral e alguns princípios taxonômicos são levantados. Nas seções seguintes, analisam-se sete taxonomias para ambientes à luz dos princípios taxonômicos apresentados. Na décima seção apresenta-se a proposta da nova taxonomia, o modelo tridimensional. E finalmente, apresenta-se a proposta da meta-taxonomia para ambientes, da qual o modelo tridimensional é apenas uma instância.

2 Por que usar uma taxonomia?

Em todas as áreas de pesquisa, inicialmente produz-se uma grande quantidade de informação nos mais variados setores, sem se importar muito com o seu interrelacionamento. Posteriormente, começa-se a se pesquisar e verificar o relacionamento de um setor com o outro. Fazendo uma analogia com programação, essa fase pode ser considerada como pesquisa 'bottom-up'. Logo percebe-se que o avanço pode ser muito lento seguindo apenas esse caminho. Então, surgem pessoas que tomam o caminho inverso, 'top-down', e analisam e estruturam o campo de pesquisa em foco, de modo a orientar e indicar os caminhos de pesquisa mais promissores. Esse caminho inverso começa pela apresentação de uma taxonomia (classificação) da área em foco. Isso pode dar um novo alento à pesquisa 'bottom-up', até que se torna necessário uma nova revisão da área de pesquisa e esse processo continua, alternada ou simultaneamente.

Taxonomia é o estudo da classificação, incluindo suas bases, princípios, procedimentos e regras. Ou seja, a taxonomia é a disciplina que trata de explicar como se classifica (Crisci e Armengol, 1983). Uma taxonomia de ambientes apresenta procedimentos que auxiliam a classificar os ambientes, de modo que se tenha condições de se dizer que o ambiente A pertence à determinada classe ou que os ambientes B e C pertencem à mesma classe ou não. Além disso, devem ser salientados dois aspectos que considero importantes:

- A taxonomia deve ser encarada apenas como uma baliza e não como reflexão de verdades fundamentais, visto que, dependendo do *objetivo* e do *ponto de vista*, pode-se apresentar uma outra classificação para o mesmo campo de pesquisa.

- Em áreas de pesquisa novas, os caminhos 'bottom-up' e 'top-down', alternados ou simultâneos, podem implicar na criação de novos setores ou de novas maneiras de encarar o campo de pesquisa.

Quais os objetivos que norteiam o estabelecimento de uma taxonomia para a área de ambientes? Por ser uma área efervescente, com novas técnicas e enfoques sendo criados e testados numa quantidade incrível, conforme se pode verificar pelos anais dos congressos sobre ambientes de desenvolvimento de programas mais recentes (Henderson, 1988; Brereton, 1988), o primeiro objetivo é estruturar e consolidar o conhecimento acerca dos ambientes e procurar ter uma compreensão mais profunda de suas propriedades, semelhanças, diferenças e interrelações. Uma boa taxonomia de ambientes seria fecunda como princípio organizador do nosso conhecimento do assunto.

O segundo objetivo é poder analisar e contrastar ambientes existentes, analisar ambientes propostos e até mesmo propor novos ambientes com base nas análises e comparações feitas. E, finalmente, o terceiro objetivo é poder estabelecer uma classificação que ajude a identificar direções de pesquisa na área e especificar aonde mais trabalho necessita ser feito.

Sumarizando, os objetivos de uma taxonomia para ambientes são os seguintes: organizar o conhecimento da área fornecendo uma linguagem comum, classificar ambientes com finalidade de comparação e identificar direções de pesquisa.

Como foi salientado acima, uma taxonomia depende muito do ponto de vista empregado. No caso de ambientes de programação, pode-se estabelecer uma taxonomia voltada ou para

questões arquiteturais (comunicação entre as ferramentas, armazenamento e recuperação de informação etc.), ou para questões ambientais (interface do usuário, se o ambiente é adequado a programadores iniciantes ou não etc.) ou para uma mistura de ambos os tipos. Pode-se também estabelecer uma taxonomia considerando o ponto de vista do projetista de ambientes (questões arquiteturais e ambientais), do pondo de vista do gerador ou adaptador de ambientes (questões arquiteturais e ambientais) e do ponto de vista do usuário final do ambiente (questões ambientais). Além disso, pontos de vistas relevantes a uma particular situação também podem ser considerados. Dentre estes, podem ser incluídos: complexidade do software produto, número de usuários, interação de IA com ES, tipos de software endereçados etc.

Um outro aspecto importante diz respeito à forma de se obter a classificação. Basicamente existem três tipos: explícita, implícita ou mista. O tipo explícito ocorre quando a taxonomia apresenta classes explícitas, hierárquicas ou não, onde um ambiente pertence a uma determinada classe se satisfizer os seus requisitos. Ou seja, cada classe apresenta um conjunto de requisitos de pertinência específico.

O tipo implícito ocorre quando a taxonomia apresenta uma lista geral de requisitos, hierárquica ou não, e as classes são formadas implicitamente pelos conjuntos de ambientes que satifazem idênticos requisitos. O tipo misto ocorre quando a taxonomia apresenta tanto classes explícitas quanto lista geral de requisitos.

O trabalho consiste em analisar os modelos taxonômicos atualmente empregados, além de analisar os ambientes existentes e propostos na literatura, e propor, baseado em alguns princípios, uma nova estrutura taxonômica para ambientes. A lista de princípios aqui apresentada não pretende ser considerada como expressão da verdade absoluta. Deve ser considerada mais como uma maneira conveniente de análise dos modelos taxonômicos aqui apresentados.

Os princípios que deverão ser usados para comparar as taxonomias são os seguintes:

- **Robustes** — quanta modificação o avanço na pesquisa poderá impor na taxonomia empregada; **pouca**, **média** e **muita** robustez são os estados considerados aqui; pouca robustes implica que pode haver muita modificação na taxonomia; média robustez que pode haver um número médio de modificações e muita robustez que pouca ou nenhuma modificação ocorrerá.

- **Granularidade** — qual o grau de estruturação em classes distintas que a taxonomia apresenta; **baixa**, **média** e **alta** granularidade são os estados considerados aqui; baixa granularidade implica que a taxonomia apresenta classes explícitas com apenas uma categoria ou uma lista geral de requisitos muita pequena (2 ou 3 ítens); média granularidade quando a taxonomia apresenta um número médio de categorias (em torno de 4 níveis) ou um número médio de ítens na lista geral de requisitos (em torno de 10); alta granularidade quando valores superiores aos apresentados na média granularidade ocorrem; categoria designa um nível em uma classificação hierárquica.

- **Eficácia** — qual o grau de facilidade na aplicação e análise da classificação obtida; **baixa**, **média** e **alta** eficácia são considerados aqui; baixa eficácia ocorre quando é

difícil aplicar e/ou analisar a classificação obtida; média quando tal tarefa é medianamente dificil de realizar e alta quando é muito fácil.

- **Ortogonalidade** — cada ambiente de qualquer classe (ou conjunto) deve diferir, ao menos, em um requisito de cada ambiente de qualquer outra classe (ou conjunto); ou seja, ambientes com todos requisitos comuns satisfeitos não podem ser distribuídos em classes, explícitas ou implícitas, diferentes; os dois estados possíveis são: **é ortogonal** ou **não é**.

Um problema comum a análises desse tipo é o da subjetividade. A subjetividade ocorre em três níveis. O primeiro deles refere-se à escolha dos aspectos que caracterizam uma taxonomia: objetivo, ponto de vista, tipo da classificação, robustez, granularidade, eficácia e ortogonalidade. Por que estes aspectos e não outros? Porque que estes aspectos são relevantes para a análise e comparação de taxonomias de ambientes e cada um evidencia um dado importante para tal tarefa.

O segundo deles refere-se ao estabelecimento dos estados dos princípios. Isto está baseado apenas na experiência obtida em lidar com taxonomias e ambientes. Não obstante, variações parecem não alterar substancialmente a análise.

O terceiro deles refere-se à aplicação dos princípios em tempo de análise das taxonomias. Apenas a aplicação dos princípios granularidade e ortogonalidade é objetiva. Ao passo que a determinação dos estados de robustez e eficácia de uma taxonomia encerra muita subjetividade, que pode variar de acordo com a experiência do analista com relação a taxonomias de ambientes e com relação à área de ambientes, com objetivo e com o ponto de vista.

Uma boa taxonomia deve:

- possuir **muita robustez**, porque a área é muito dinâmica e deve fazer frente ao avanço nas pesquisas com pouca ou nenhuma modificação;
- possuir **média granularidade**, para estruturar adequadamente a classificação dos ambientes;
- possuir **alta eficácia**, de modo a facilitar o trabalho de aplicação e análise da classificação obtida;
- ser **totalmente ortogonal**, de modo a não apresentar ambiguidade na classificação dos ambientes.

3 A taxonomia de Lucena

Lucena (1987) ao estabelecer subsídios para a arquitetura da máquina ETHOS, apresenta uma taxonomia para ambientes de desenvolvimento de software (também em Lucena e Soares, 1985). Esta taxonomia apresenta três classes de ambientes:

1. **ambientes centrados em sistemas operacionais.**

2. **ambientes centrados em linguagens.**

3. **ambientes centrados em métodos ou metodologias.**

Aqui entende-se metodologias como o conjunto de métodos utilizados para a solução de uma classe de problemas, juntamente com as heurísticas utilizadas na seleção dos métodos e em suas aplicações.

A classe (1) é bem caracterizada por *caixas de ferramentas* ('toolkits') ou *bancadas do programador* ('workbenches') construídos sobre o sistema operacional UNIX (Kernighan and Mashey, 1984; Dolotta et al., 1984).

A classe (2) é constituída por conjunto de ferramentas organizadas em torno de uma ou mais linguagens (ou formalismo). As linguagens são, em geral, linguagens de programação (por exemplo Ada) e os ambientes são, na maioria, monolinguísticos. Servem de exemplo sistemas como Arcturus (Standish et al., 1984), para Ada, e sistemas como o Cornell Program Synthesizer (CPS), para PL/CS (Teiltelbaum et al., 1981), além dos conhecidos sistemas da Xerox: Interlisp (Teiltelman et al., 1981), Smalltalk (Goldberg et al., 1983), Mesa (Sweet, 1985) e Cedar (Teiltelman, 1984).

A classe (3) é formada por sistemas que têm como aspecto mais característico a metodologia ou método usado para o projeto e derivação de programas. Como exemplo têm-se sistemas baseados em cálculos de transformações como os de Bauer (1985) e Darlington (1981). Recentemente, diversos ambientes foram construídos para o apoio a metodologias consagradas, como as de Jackson, Yourdon e Constantine, Entidades e Relacionamentos, Petri Nets e outras. Como exemplo, têm-se os ambientes Mosaico (Staa, 1986) e Talisman (Staa, 1988) desenvolvidos na PUC/RJ. Um enfoque que vem sendo adotado é o de construir esses sistemas parcialmente como sistemas especialistas nessas metodologias, procurando representar, naturalmente, o grande conteúdo heurísticos que elas encerram (ex. Schwabe et al., 1987). Talvez os mais expressivos projetos dessa área sejam os projetos do ISI/USC (Balzer, 1985) e do Instituto Kestrel (Smith et al., 1985) que procuram associar as técnicas de especificação formal, programação heurística e transformação de programas.

O objetivo da taxonomia de Lucena é classificar os ambientes do ponto de vista do usuário (ambiental) dentro das três classes. É pouca robusta (onde colocar ambientes baseados em SGBD's?), tem baixa granularidade (apenas uma categoria) e média eficácia (permite uma classificação muito fácil, mas a análise do resultado tem utilização limitada).

Na classe (2), necessita-se diferenciar os ambientes verdadeiramente centrados em linguagens, como os desenvolvidos para apoiar o desenvolvimento de programas Lisp, dos ambientes que oferecem uma estrutura de interface com o usuário flexível de forma a permitir o uso de diversas linguagens, muito embora eles possam permanecer monolinguísticos. Este problema é devido à baixa granularidade.

Na classe (3), também devido à baixa granularidade, convivem ambientes que apoiam apenas fragmentos do processo de desenvolvimento de software (quanto ao ciclo de vida do paradigma empregado) e ambientes que apoiam o processo todo. Além disso, convivem ambientes cujas arquiteturas estão baseadas em técnicas de IA e ambientes cujas arquiteturas são convencionais. Ou seja, sente-se a falta de uma melhor caracterização dos ambientes dentro desta categoria.

Um aspecto importante que a taxonomia não leva em conta é o do ponto de vista da escala, ou seja, quanto ao tamanho dos projetos, considerando-se o número de pessoas envolvidas bem como o tamanho e complexidade dos sistemas.

Por tudo isso (e de acordo com nossa definição de boa taxonomia), esta não é uma boa taxonomia de ambientes, apesar de ter servido aos seus propósitos muito bem na época de sua formulação. Além disso, o fato de ser pouca robusta implica em obsolescência . De fato, esta taxonomia pode ser considerada como obsoleta.

4 O modelo de Meyer

Meyer (1986) assevera que questões arquiteturais em ambientes são aquelas que tem a ver com a estrutura dos ambientes e não com seus conteúdos, com os relacionamentos entre componentes e não com as funcões individuais destes componentes. Ele distinguiu quatro dimensões — cultura, abstração, comunicação e evolução:

- **Cultura** — inclui todos aspectos que não são facilmente mensuráveis e não resultam necessariamente em uma característica imediatamente identificável do ambiente mas que jogam um papel fundamental em dar forma ao projeto global do ambiente.
- **Abstração e estrutura** — inclui os mecanismos que possibilitam estruturar um ambiente e descrever seus componentes e sub-sistemas.
- **Comunicação** — caracteriza os mecanismos de comunicação relevantes.
- **Evolução** — relaciona os aspectos de tempo.

A seguir, apresento uma estruturação hierarquizada destes quatro grupos. Ressalto que a apresentação é sumária, com o objetivo de ilustrar o alcance da taxonomia. Leitores interessados em mais detalhes podem dirigir-se ao trabalho de Meyer (1986).

cultura

- — tipo de usuários
  - — iniciantes
  - — experientes
  - — especialistas
- — hardware
  - — hardware para suportar o software
  - — o software deve se adaptar ao hardware
- — linguagens
  - — linguagem única
  - — linguagem base + outras linguagens disponíveis
  - — independente de linguagem
- — método

- método único
- conjunto fixo de métodos
- adaptabilidade de métodos
- independente de método

abstração e estrutura

- especificação de ferramentas
  - abstração funcional
  - tipos abstratos de dados
  - nenhuma ênfase em abstração
- composição de ferramentas
  - técnicas de combinação
    - composição funcional
    - empacotamento
    - herança
  - tempo de composição
    - amarração estática
    - amarração dinâmica
- seleção de componentes (invocação)
  - critérios de seleção
    - sistemas com base em modos
    - sistemas sem modos
  - generalidade de aplicação de componentes
    - aplicabilidade específica
    - aplicabilidade geral
- abertura ao resto do mundo
  - relação com o resto do mundo
    - ambiente isolado
    - ambiente embutido
  - extensibilidade pelo usuário
    - sistemas controlados
    - sistemas livres

comunicação

- ambiente para máquina virtual
  - do ambiente para a máquina virtual
  - ambientes opacos
  - ambientes transparentes
  - da máquina virtual para o ambiente

— ferramentas para ferramentas

- — arquivos
- — mecanismo tipo 'pipe' do Unix
- — base de dados
- — facilidades públicas
- — sinais tipo interrupção, semáforos etc.

— ambiente para usuário humano

- — forma
  - — apenas texto
  - — janelas
  - — sistemas gráficos
- — grau de concorrência do usuário
  - — processo único
  - — processos múltiplos
- — número de canais
  - — canal único
  - — múltiplos canais

— hardware para hardware

- — redes locais
- — sistemas de arquivos distribuidos
- — etc.

evolução

— tempo de projeto

- — conservação do objeto
  - — sistema de arquivos
  - — DBMS
- — controle de trocas e de configuração
  - — esquema de nomes
  - — relacões inter-entidades
  - — recriação do sistema
- — verificações de integridade
  - — verificação explícita
  - — verificação dirigida por eventos
- — apoio ao ciclo de vida
  - — modelo específico
  - — modelo adaptável
  - — não apóia nenhum modelo em particular

— tempo de ambiente

— facilidade de trocar componentes
— facilidade de integrar novos componentes

O objetivo desta taxonomia é descrever os ambientes e apresentar direções de pesquisa, tanto do ponto de vista arquitetural quanto do ambiental, nas quatro dimensões apresentadas. Ao se analisar essas dimensões, verifica-se que Meyer tinha como alvo uma classe restrita de ambientes, a saber, os ambientes usualmente conhecidos como ambientes de programação. Constata-se que tem média robustez, alta granularidade (lista geral hierárquica de requisitos), baixa eficácia (esse aspecto um tanto detalhista de descrever ambientes, torna difícil comparar ambientes ou mesmo classificar um ambiente em particular) e é ortogonal.

O fato de possuir alta granularidade implicou em baixa eficácia. Além disso, não foram consideradas características de conexão com IA. No entanto, esta taxonomia considera algumas questões quanto ao problema do ponto de vista da escala.

Apesar de tocar em quase todos aspectos ambientais e arquiteturais relevantes, a taxonomia de Meyer não é boa, principalmente devido ao tipo de classificação, implícita através de lista hierárquica de requisitos, e ao fato de restringir o escopo dos ambientes apenas aos ambientes de programação.

5 O modelo de Weiser et al.

O objetivo do modelo de Weiser et al. (1986) é descrever arquiteturas subjacentes aos ambientes. As características levantadas são, segundo os seus autores, pontos de discussão para o futuro, identificando direções de pesquisa em arquiteturas de ambientes.

Foram compiladas as seguintes características, julgadas relevantes ao projeto de arquiteturas de ambientes:

1. **Depósitos**

   - Todo armazenamento de objetos toma lugar em depósitos, por definição. Arquivos são depósitos, memória principal é depósito, programas podem estar em depósitos etc. Um ambiente pode contar com um ou mais depósitos.
   - Algumas dimensões de variação: duração, estabilidade, gerência, estrutura, atividade, base de dados, segurança, leitura/escrita, distribuição.

2. **Interfaces de programas a depósitos**

   - Distintos dos depósitos são os meios de acessá- los. A um determinado estilo de depósito podem ser dadas muitas interfaces de acesso diferentes.
   - Alguns possíveis estilos: tradicional, linguagem de consulta, concorrência, 'daemons', gerência de versões, mecanismos de tipos, proteção.

3. **Comunicação entre as ferramentas**

   - A interconexão das ferramentas é um aspecto muito importante de qualquer ambiente. A arquitetura subjacente pode suportar interconexões simples ou complexas.
   - Alguns tipos de conexão: explícita/implícita, memória principal, 'pipes', chamada de procedimentos, mensagens e depósitos.

4. **Composição e empacotamento de ferramentas**

   - O empacotamento de ferramentas depois que elas estejam interconectadas é um problema distinto da interconexão das ferramentas. Por exemplo, um editor de ligação é um empacotador tradicional de ferramentas, usando chamada de procedimentos como interconexão entre elas.
   - Alguns mecanismos de empacotamento: especialização (ex.: um editor de ligação), linguagem, nível da linguagem, tempo de ligação, herança, ocultação da estrutura interna e transparência.

5. **Nomes**

   - Os objetos dos depósitos necessitam de algum tipo de nome ou senão eles não podem ser acessados ou usados.
   - Alguns tipos de relacionamento entre os nomes: hierarquia, relatividade quanto aos depósitos, associatividade, tempo de vida, questão das versões e quanto aos tipos.

6. **Eval**

   - Eval é o termo (emprestado do Lisp) que os autores do modelo usam para descrever o processo de iniciar trabalho computacional. Eles se concentraram nos diferentes modos de iniciar e terminar Eval.
   - Algumas possibilidades: chamada de procedimentos, assincronismo, 'exec' do Unix, 'fork' da linguagem C, tempo de vida e modo (interpretado ou compilado).

7. **Co-existência**

   - Co-existência refere-se aos diferentes tipos de relacionamentos ambientes/sistemas operacionais e ambientes/outros ambientes.
   - Algumas das possibilidades: abaixo, ao lado e acima.

8. **Interface do usuário**

   - Não apresentam um tratamento adequado desta questão. Algumas das características levantadas: 'window systems', múltiplos meios de interação ('mouse', teclado, visual, toque na tela, som), troca ou reprogramação da interface e interação com ou sem modos.

O modelo de Weiser et al. não constitui verdadeiramente uma taxonomia visto não apresentar classes nem lista geral de características que auxiliem na classificação. Os próprios autores do modelo não a consideram uma taxonomia, porque facilidade de classificação não era o mais importante objetivo deles. O objetivo deles era identificar direções de pesquisa em ambientes, e a apresentação das características acima procura enfatizar isso, do ponto de vista arquitetural. Justamente por esse motivo, além da tentativa de sumarizar todos trabalhos relevantes na área, é que o modelo de Weiser et al. foi apresentado.

Embora não constitua uma taxonomia propriamente dita, a análise quanto aos princípios taxonômicos revela que é pouca robusta (porque as características levantadas podem ser contestadas por outros pesquisadores), tem baixa granularidade, baixa eficácia (segundo os próprios autores do modelo) e claramente não é ortogonal.

Devido a esses problemas, fica-se com a impressão de que, após a aplicação deste modelo aos ambientes, não adianta muito ter tanto trabalho, principalmente quando se quer compará-los. Além disso, não ajuda no dimensionamento do problema da escala, não trata do problemas relacionados com arquiteturas baseadas em IA nem trata do escopo dos ambientes quanto ao ciclo de vida do paradigma empregado. Portanto, fica evidente que o uso deste modelo para análise de ambientes é muito limitado.

6 A taxonomia de Dart et al.

A taxonomia de Dart et al. (1987) apresenta quatro classes para enquadrar os ambientes:

- **Ambientes centrados em linguagem** fornecem ferramentas integradas apoiando uma linguagem específica. Como exemplo, têm-se os ambientes Interlisp e Smalltalk.
- **Ambientes caixa de ferramentas** consistem de uma coleção de ferramentas menores, que podem ou não compartilhar informação uma com as outras. Como exemplo, tem-se o Unix/PWB (Dollota et al., 1984).
- **Ambientes orientados à estrutura** apoiam manipulação direta de estruturas dos programas e também diferentes visões das estruturas dos programas em diferentes níveis de abstração. Como exemplo, têm-se os ambientes Cornell Program Synthesizer (Reps e Teiltelbaum, 1981) e Pecan (Reiss, 1984).
- **Ambientes baseados em metodologia** apoiam metodologias de desenvolvimento de software específicas via ferramentas automatizadas de engenharia de software. Como exemplo, têm-se os ambientes Talisman (Staa, 1988) e IStar (Dowson, 1987).

O objetivo desta taxonomia é classificar os ambientes do ponto de vista do usuário (ambiental) dentro das quatro classes. Devido à semelhança com a taxonomia de Lucena, é de se esperar que apresente os mesmos problemas: pouca robustez, baixa granularidade e média eficácia.

Estas quatro classes podem ser identificadas com as apresentadas na taxonomia de Lucena. Os ambientes caixa de ferramentas podem ser identificados com os ambientes centrados

em sistemas operacionais e os ambientes baseados em metodologias com os ambientes centrados em métodos ou metodologias. Os ambientes centrados em linguagem e os orientados à estruturas podem ser identificados com os ambientes centrados em linguagem de Lucena, eliminando-se assim um dos problemas que ocorria na taxonomia de Lucena. Nesta taxonomia pode-se diferenciar ambientes tipo Interlisp de ambientes tipo CPS.

Contudo, a falta de uma melhor caracterização dos ambientes dentro da categoria dos ambientes baseados em metodologias continua devido à baixa granularidade bem como os problemas derivados da não consideração tanto do ponto de vista da escala quanto do escopo dos ambientes quanto ao ciclo de vida do paradigma empregado. Por conseguinte, esta taxonomia também pode ser considerada como obsoleta.

7 A taxonomia de Perry e Kaiser

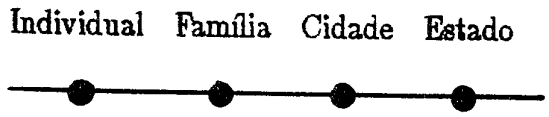
Perry e Kaiser (1987) apresentam um modelo geral para ambientes composto de três componentes interrelacionados: políticas, mecanismos e estruturas. Políticas são regras, direcionamentos e estratégias impostas ao programador pelo ambiente, durante o processo de desenvolvimento de software. Na maioria das vezes, o projeto das ferramentas e das estruturas de apoio definem ou impõem as políticas do ambiente. Estas são as políticas implícitas. Por exemplo, o requisito que somente objetos ligados e carregados possam ser executados induz uma política de sempre compilar os módulos antes de ligá-los. Algumas arquiteturas permitem a especificação explícita de políticas. O ambiente Arcadia (Taylor et al., 1988) permite programar as políticas desejadas com respeito às várias estruturas e mecanismos disponíveis.

Faz-se, ainda, uma distinção entre apoiar e obrigatoriamente impor políticas. Se uma política é apoiada, então os mecanismos e estruturas fornecem um meio de satisfazer esta política, podendo, no entanto, satisfazer outras políticas. Por exemplo, suponha que desenvolvimento "top-down" seja uma política apoiada. Então, seria razoável encontrar ferramentas e estruturas que habilitem o usuário a construir o sistema em uma forma top-down. Além disso, se encontrariam ferramentas e estruturas para construir sistemas de outras maneiras também. Se uma política é imposta, então , não somente é apoiada como é impossível fazê-lo de qualquer outro modo dentro do ambiente.

Mecanismos são as linguagens, ferramentas e fragmentos de ferramentas que operam sobre as estruturas fornecidas pelo ambiente e que implementam, junto com as estruturas, as políticas apoiadas e impostas pelo ambiente.

Estruturas são aqueles objetos ou agregados de objetos sobre os quais os mecanismos operam. As estruturas básicas mais simples são os arquivos. No entanto, a tendência é ter-se objetos mais complexos: representação interna complexa, base de dados, base de objetos.

A seguir, Perry e Kaiser apresentam quatro classes de modelos do ponto de vista da escala: quanto ao número de usuários e quanto ao tamanho do sistema. A classificação é feita em termos de uma metáfora sociológica que realça as distinções com respeito aos problemas de escala. A taxonomia distingue quatro classes de modelos quanto à escala:



As quatro classes seguem a metáfora de que uma família é uma coleção de indivíduos, uma cidade é uma coleção de famílias, e um estado é uma coleção de cidades; cada classe incorpora aquelas classes à sua esquerda.

A classe **modelo individual** representa aqueles ambientes que fornecem o conjunto mínimo de ferramentas de implementação necessário para construir software. Estes ambientes são frequentemente referidos como ambientes de programação. Os mecanismos oferecidos são: editores, compiladores, linker/loaders e depuradores. A estrutura é simples e é compartilhada entre os mecanismos: arquivos e árvores decoradas. As políticas, em geral, são frouxas com relação às questões metodológicas. O esquema geral é o seguinte:

Modelo individual =

({políticas induzidas pelas ferramentas}, {ferramentas de implementação}, {estrutura simples e compartilhada})

Estes ambientes são dominados pela questão da construção de software, com ênfase nos mecanismos e fazendo com que políticas e estruturas assumam um papel secundário. As políticas são induzidas pelos mecanismos, enquanto as estruturas satisfazem o requisito de fazer as ferramentas trabalharem juntas.

São apresentados, como exemplo do processo, quatro grupos de ambientes que são instanciações do modelo individual: ambientes caixa de ferramentas, interpretativos, orientados a linguagem e transformacionais. Os ambientes caixa de ferramentas são exemplificados pelo UNIX; os interpretativos pelo Smalltalk e Interlisp; orientados a linguagem pelo CPS; e transformacionais pelo CHI/Refine(Smith et al., 1985). A seguir, mostram-se os seus respectivos esquemas. A notação "..." no fim da lista indica que facilidades adicionais podem estar disponíveis. A notação classe::instanciação é usada para indicar a particular instanciação para a classe de modelo.

Modelo individual::caixa de ferramentas =

({políticas induzidas pelas ferramentas e pelas estruturas tipo shell do UNIX, ...}, {editores, compiladores, loaders, depuradores, ...}, {sistema de arquivos})

Modelo individual::interpretativo =

({virtualmente nenhuma política restritiva}, {interpretador, ferramentas de suporte subjacentes}, {representação intermediária})

Modelo individual::orientado a linguagem =

({prevenção contra erros, descoberta e notificação de erros o mais cedo possível, ...}, {editor, compilador, depurador, ...}, {árvores decoradas})

Modelo individual::transformacional =

({construção transformacional, ...}, {interpretador, máquina transformacional, ...}, {representação intermediária-árvores decoradas, ...})

Note-se a compatibilização da taxonomia de Dart et al. com estas quatro instanciações. A diferença é que o 'modelo transformacional' é apenas um grupo dos possíveis ambientes da classe 'ambientes baseados em metodologia'. Isto serve para realçar os problemas encontrados naquela taxonomia. Aqui outros grupos do modelo individual podem ser instanciados de forma clara e concisa.

A classe **modelo família** representa aqueles ambientes que fornecem, além de um conjunto de ferramentas de construção de programas, facilidades que apoiam a interação de um pequeno grupo de usuários, digamos em torno de 10 pessoas. No modelo individual, nenhuma política de coordenação é necessária, enquanto que neste modelo algumas regras são necessárias para regular certas interações críticas entre os programadores. O esquema geral é o mostrado a seguir, onde a notação "..." no início da lista indica que as políticas, mecanismos e estruturas do nível anterior estão incluídas.

Modelo família =

({..., políticas de coordenação}, {..., mecanismos de coordenação}, {..., bases de dados não convencionais})

A característica que distingue o modelo família do individual é a coordenação imposta. Porque as estruturas do modelo individual não são ricas o suficiente para coordenar atividades simultâneas, estruturas mais complexas são requeridas. São estas estruturas que dominam o projeto do ambiente.

Quatro grupos de ambientes são apresentados como instanciações do modelo família:

Modelo família::caixa de ferramentas estendido
Modelo família::integrado
Modelo família::distribuído
Modelo família::gerência de projetos.

Ambientes caixa de ferramentas estendidos são exemplificados pelo UNIX junto com um sistema de controle de versões/configurações SCSS ou RCS (Tichy, 1985); ambientes integrados que, por analogia, estendem os ambientes orientados a linguagem do modelo individual, são exemplificados por Gandalf Prototype (Habermann e Notkin, 1986), Toolpack (Osterweil, 1983) e R^n (Cooper et al., 1986); ambientes distribuídos expandem o modelo integrado através de um número de máquinas conectadas por uma rede local, são representados por DSEE (Leblang and Chase, Jr., 1984) e Mercury (Kaiser et al., 1987); e ambientes de gerência de projetos são representados por CMS. Esta classe de modelos representa o estado da arte corrente em ambientes.

O modelo cidade é introduzido para fazer frente aos problemas de escala e de complexidade de interações num projeto de um grande sistema e com aproximadamente 20 pessoas trabalhando juntas. O esquema geral é o seguinte:

Modelo cidade =

({..., políticas de cooperação}, {mecanismos de cooperação}, {estruturas para cooperação})

Cooperação imposta é a principal característica deste modelo, visto que a noção de coordenação imposta do modelo família é insuficiente quando aplicado aos problemas de escala representado pelo modelo cidade. Neste modelo, são as políticas que dominam o projeto do ambiente. Pouco trabalho de pesquisa tem sido feito em ambientes que implementam um modelo cidade. Na verdade, Perry e Kaiser argumentam que o modelo família está atualmente sendo usado onde se necessita de um modelo cidade, e que o modelo família não é adequado para a tarefa. Inscape/Infuse (Perry e Kaiser, 1987) e IStar representam modelos desta classe.

A última classe de modelos é a de **estado**. A noção de um estado como uma coleção de cidades é análoga à de uma companhia com um coleção de projetos. Os problemas que surgem nesta escala podem ser os seguintes: padronizar o uso de uma linguagem particular para todos os projetos; estabelecer o uso de um ambiente uniforme para todos os projetos; estabelecer uma metodologia comum e um conjunto de padrões para uso em todos projetos. Com estes problemas resolvidos, têm-se redução nos custos e melhoria na produtividade. O esquema geral é o seguinte:

Modelo estado =

({..., políticas padronizadoras}, {...mecanismos de apoio}, {...estruturas de apoio})

Neste modelo, o interesse em padrões é dominante. Esta política de padronização tende a induzir políticas nos projetos específicos, isto é, em seus ambientes modelo cidade. Os autores não apresentam nenhuma instância deste modelo.

O objetivo da taxonomia de Perry e Kaiser é classificar ambientes e identificar direções de pesquisa do ponto de vista do projetista de ambientes e da escala. Constata-se que ela é muito robusta, tem média granularidade, média eficácia e é ortogonal.

O que a torna muito robusta é o mecanismo de instanciação, que não fica limitado aos modelos instanciados aqui. O avanço na pesquisa pode dar origem a um modelo não coberto pelos apresentados, mas o impacto na taxonomia é mínimo dentro desta estrutura. Basta criar uma nova instância de um modelo que satisfaça a nova necessidade.

A eficácia é média devido a granularidade apresentada, que é boa mas insuficiente. Devido a isso, a taxonomia apresenta os seguintes problemas:

- Em cada categoria convivem ambientes que apoiam apenas fragmentos do processo de desenvolvimento de software, ambientes que apoiam o processo todo mas cuja arquitetura está baseada em técnicas de IA, ambientes com arquitetura convencional.
- Apesar de toda facilidade de instanciação de cada modelo, sente-se a falta de uma melhor caracterisação dos ambientes dentro de cada categoria:
  - Por exemplo, é necessário diferenciar ambientes que apoiam completamente um paradigma mas tem arquitetura convencional de ambientes que também apoiam um paradigma completo mas tem arquitetura baseada em IA.

- Outro exemplo: é necessário diferenciar ambientes que apoiam completamente um paradigma dos que apoiam apenas fragmentos de paradigmas.

Por tudo isso e mais a nossa definição de boa taxonomia, pode-se dizer que esta se aproxima muito do ideal, precisando apenas receber um polimento para se tornar uma boa taxonomia.

8 A taxonomia de Penedo e Riddle

Penedo e Riddle (1988) não estavam preocupados em estabelecer uma nova taxonomia sobre ambientes quando esta surgiu. O trabalho deles consistiu em escrever uma introdução a uma edição especial da revista IEEE Transactions on Software Engineering sobre arquiteturas de ambientes de engenharia de software, como editores convidados. No entanto, com o intuito de melhor situar o leitor quanto ao estado da arte, eles produziram um extensivo levantamento das características dos ambientes. Com base nesse levantamento, eles procuraram enquadrar os ambientes até então conhecidos bem como os trabalhos apresentados naquela edição.

Similaridades e diferenças entre ambientes atuais podem ser agrupadas em quatro áreas: tecnologia subjacente, características desejáveis, apoio para o desenvolvimento do software alvo e a estrutura arquitetural básica.

Várias tecnologias subjacentes são usadas como base para muitos dos ambientes correntes. Penedo e Riddle identificaram as seguintes:

- **Tecnologia baseada em sistemas operacionais** — Nestes ambientes, as ferramentas são invocadas como programas, os dados são organizados e acessados através de um sistema de arquivos e o controle do usuário é efetuado através de uma linguagem de comandos. Exemplos: Unix da AT&T Bell Laboratories e VMS da DEC.

- **Ambientes orientados à estrutura** — Exemplos: CPS, Gandalf, Mentor (Donzeau-Gouge et al., 1984) e Cépage (Meyer, 1988).

- **Ambientes centrados em linguagem** — Exemplos: Interlisp, Smalltalk, Cedar e Mesa.

- **Tecnologia baseada em SGBD** — Os dados são estruturados e organizados através do uso de esquemas de banco de dados, as ferramentas comportam-se como transformadores de dados, o controle dos dados é efetuado através do fluxo implícito ou explícito entre as ferramentas, com o banco de dados servindo como um depósito de informação logicamente centralizado, e o controle do usuário pode ser efetuado através de uma linguagem de comando ou uma interface de usuário mais complexa. Exemplos: PCTE e APSE's.

- **Tecnologia de sistemas especialistas** — Bases de conhecimento são usadas para manter informação tanto sobre os produtos sendo produzidos quanto sobre os processos usados para produzí-los, as ferramentas são vistas como interpretadores de regras

especificando características e restrições sobre produtos e processos e o controle é indiretamente efetuado através da especificação das características e restrições. Exemplos: CHI/Refine, KBemacs (Waters, 1985), projeto da ISI/USC, Microscope (Ambras and O'Day, 1988), Triad (Ramanathan and Rose, 1985) e ETHOS.

- **Tecnologia baseada em objetos** — Situa-se entre as tecnologias baseadas em SGBD e em sistemas especialistas. Toda informação é modelada como relações entre objetos mantidos em um sistema de base de objetos persistentes. Os objetos são uniformemente acessíveis e genericamente manipuláveis tanto pelos programas quanto pelos usuários. Regras declarativas expressam relacionamentos de consistência automaticamente mantidos para garantir a integridade da base de objetos. O processamento é invocado automaticamente em resposta a trocas na base de objetos, realizando tarefas de rotina sem intervenção do usuário. Exemplo: Arcadia (Taylor et al. 1988).

Penedo e Riddle examinaram as tecnologias subjacentes através dos seguintes três componentes: controle do usuário, ferramentas e estruturas. Basicamente, são equivalentes aos componentes do modelo geral de ambientes de Perry e Kaiser: políticas, mecanismos e estruturas.

Ambientes podem ter um conjunto de características desejáveis dentre as quais Penedo e Riddle identificam as seguintes:

- **Utilidade**: quanto das atividades relativas ao ciclo de vida são oferecidas (por exemplo, projeto arquitetural, teste de aceitação etc.).
- **Interação**: interfaces amigáveis e consistentes, facilidade de aprendizagem, proximidade da prática existente etc.
- **Adaptabilidade**: grau para o qual o ambiente pode ser adaptado, feito sob medida ou estendido.
- **Automação**: quanta assistência automatizada e/ou inteligente o usuário recebe no ciclo do desenvolvimento de software.
- **Integração**: quanto dos recursos do ambiente são integrados e co-operativamente compartilham código e objetos de dados persistentes.
- **Valor**: taxa de benefícios (aumento de produtividade, aumento da qualidade do produto etc.) e custos (investimento de aquisição, tempo de desenvolvimento etc.).

Segundo Penedo e Riddle, a comparação de ambientes com relação ao que é visível no apoio ao desenvolvimento do software alvo pode ser vista como uma elaboração da característica de utilidade deles. Utilidade, como definida anteriormente, diz respeito às tarefas relacionadas com as fases (projeto, especificação etc.) e tarefas que cruzam as fronteiras das fases (validação, monitoração do progresso etc.), à cobertura de papéis do usuário relacionados ou não com o ciclo de vida (projetista, gerente do projeto, criador do ambiente etc.), bem

como com os tipos de sistemas de software da aplicação fim que podem ser criados (sistemas de contabilização, sistemas de controle de processos etc.

Quanto à estrutura arquitetural básica, eles identificam os seguintes atributos chaves:

- **Arquitetura de máquina virtual** — Exemplos: arquitetura de alguns APSE's.
- **Arquitetura de rede** — Exemplo: Caede.
- **Arquitetura centrada em dados** — Esta coloca o depósito de informação do ambiente no núcleo e organiza os componentes em termos do fluxo de dados entrando e saindo da base de dados. Exemplos: DREAM (Riddle et al., 1977) e a maioria dos ambientes que contam com tecnologia SGBD exibem este tipo de arquitetura.
- **Arquitetura centrada no controle** — Esta coloca o subsistema supervisor interno do ambiente no núcleo e organiza os componentes em termos do fluxo de controle entre eles. A maioria dos ambientes baseados na tecnologia dos sistemas especialistas exibem este tipo de arquitetura.

Os ambientes podem exibir um destes atributos arquiteturais exclusivamente, como o ambiente DREAM acima citado. No entanto, o mais comum é exibir uma mistura destes atributos de arquitetura. Toolpack é um exemplo disto, sobrepondo uma arquitetura centrada em dados em uma arquitetura de máquina virtual.

O objetivo da taxonomia de Penedo e Riddle é classificar ambientes e identificar direções de pesquisa do ponto de vista do projetista de ambientes. Constata-se que ela é muito robusta, tem média granularidade, baixa eficácia e não é totalmente ortogonal.

Um dos motivos da baixa eficácia é o problema de apresentar uma classificação mista, apresentando dois grupos de classes explícitas e duas listas gerais de requisitos, mas o fator agravante é a não ortogonalidade, visto que o primeiro grupo classifica os ambientes quanto à tecnologia subjacente e o último grupo classifica os mesmos ambientes quanto à arquitetura básica, sendo que nesta última área a não ortogonalidade é flagrante. O que está conflitando mesmo é que eles reuniram quatro classificações numa só, mas sem apresentar nenhuma integração. Com isso fica relativamente fácil classificar os ambientes mas extremamente difícil extrair alguma coisa do resultado da análise. Além disso, há uma identificação muito grande entre a classe 'tecnologia baseada em SGBD' e a classe 'arquitetura centrada em dados' e entre a classe 'tecnologia baseada em sistemas especialistas' e a classe 'arquitetura centrada no controle'.

Um aspecto importante que a taxonomia tenta apresentar uma solução é o do ponto de vista da escala, através das duas listas gerais de requisitos. No entanto, o resultado é fraco por não conseguir um modelo adequado para políticas de coordenação e cooperação junto com os respectivos mecanismos e estruturas associadas.

Devido a esses problemas, fica-se com a impressão de que se terá muito trabalho para comparar ambientes e muito pouco resultado prático, apesar da taxonomia tocar em quase todos aspectos ambientais e arquiteturais relevantes.

9 A taxonomia de Partridge

Partridge (1988) apresenta uma revisão da área de interação entre IA e ES, argumentando que não só a ES pode se beneficiar com os avanços da IA, mas que também a IA pode se beneficiar dos avanços da ES em construir sistemas práticos, usáveis e confiáveis. Uma taxonomia desta área de interação é desenvolvida, constando de três grupos: ambientes de apoio baseados em IA, mecanismos e técnicas de IA em software prático e ferramentas e técnicas de ES em sistemas de IA. A seguir, tem-se um breve sumário de cada grupo:

- **Ambientes de apoio baseados em IA**

  - Programação automática/Implementação transformacional.
  - Assistentes de software baseados em conhecimento.
  - Exemplos: ETHOS, SAFO (Vieira e Lins, 1985), CHI/Refine e PLUMber's Apprentice (Partridge, 1988).

- **Mecanismos e técnicas de IA em software prático**

  - Tecnologia de sistemas especialistas.
  - Aprendizagem mecânica.
  - Interfaces homem-computador.
  - Interfaces em linguagem natural para sistemas de banco de dados.
  - Exemplos: os trabalhos de Wilenski (1984) e Harris (1984) além dos ambientes que usam tais técnicas para aumentar a eficiência das ferramentas empregadas no processo de software (Ramamoorthy et al., 1986).

  Observação: deve-se notar que este grupo difere totalmente do anterior, visto tratar-se de recurso usado para ajudar os usuários em cada estágio do ciclo de vida envolvido ou em outras atividades de apoio ao processo de software convencional. O grupo anterior, por sua vez, enxerga o processo de software de maneira diferente do convencional, estabelecendo novos paradigmas com ciclos de vida próprios.

- **Ferramentas e técnicas de ES em sistemas de IA**

  Esta categoria trata do uso inovativo de ES para gerar sistemas de IA robustos e confiáveis. Consideram-se os seguintes tópicos: ambientes de apoio aos sistemas, novas linguagens de implementação (possivelmente com multi-paradigmas), modelamento entidade-relacionamento, tecnologia de dicionário de dados, especificação formal e técnicas de provas.

Finalmente, Partridge observa que a área de interação entre IA e ES não é estática e assim qualquer tentativa corrente para mapear todas as possibilidades deve ser vista apenas como mera especulação (Simon, 1986; Mostow, 1985; Waltz, 1985). Com isso, ele pretende dizer que tal tarefa é importante mas não deve ser encarada como uma verdade fundamental. No

entanto, Arango et al. (1988) propõem uma estrutura conceitual para futuras discussões de aplicações de IA em ES através de uma abordagem incremental e sistemática.

O objetivo da taxonomia de Partridge está limitada a mapear as possibilidades de interação entre IA e ES do ponto de vista teórico, metodológico e de arquitetura básica de ambientes, através de três classes apenas. Constata-se que ela é uma boa taxonomia, levando-se em consideração o escopo reduzido de sua atuação. Ela tem muita robustez, média granularidade, alta eficácia e é ortogonal.

10 A proposta de uma nova taxonomia

Nesta seção, a proposta de um novo modelo taxonômico é apresentada. Esta nova taxonomia procura englobar diversos aspectos positivos verificados nos modelos da literatura, de acordo com a visão do projetista de ambientes e da escala, além de procurar sanar todos os problemas encontrados nas taxonomias acima analisadas. Os objetivos desta taxonomia são: organizar o conhecimento da área, classificar ambientes com finalidade de comparação e identificar direções de pesquisa.

Comparou-se algumas taxonomias encontradas na literatura, usando- se alguns princípios que foram derivados exclusivamente para tal tarefa. Dentre as quais, a de Perry e Kaiser e a de Partridge são as que tem mais aspectos positivos e menos problemas, além de possuirem um tipo de classificação explícita que julgo ser mais eficaz. Um passo natural é procurar um modelo taxonômico que englobe de forma integrada as duas citadas taxonomias. Um dos problemas com a taxonomia de Perry e Kaiser é não instanciar adequadamente modelos quanto à interação de IA e ES. Assim, a taxonomia de Partridge pode servir de modelo para as instanciações nessa dimensão. Um outro problema, refere- se a não diferenciação de ambientes que apoiam completamente um paradigma de ambientes que apoiam apenas fragmentos de paradigmas. Assim, torna-se necessário incorporar este aspecto ao novo modelo taxonômico. Com base nesse trabalho de análise, uma nova taxonomia é proposta: o modelo tridimensional.

10.1 O modelo tridimensional

A taxonomia de Partridge estende-se em uma dimensão e refere-se à interação de IA e ES. Aqui aproveitamos para fazer uma alteração. Eliminamos a classe de 'ferramentas e técnicas de ES em sistemas de IA' e acrescentamos a classe 'ambientes que não fazem uso de IA' (ambientes convencionais). Portanto, a primeira dimensão fica composta das seguintes classes: sem uso de IA, mecanismos e técnicas de IA (como o emprego de sistemas especialistas em ambientes) e arquitetura baseada em IA (como programação automática/implementação transformacional , assistentes de software baseados em conhecimento etc.). A segunda dimensão refere-se aos paradigmas metodológicos (corresponde à dimensão de atividade de Henderson and Notkin, 1987): ambientes que suportam fragmentos de paradigmas (como o Mosaico), que suportam um paradigma completo (como o ETHOS) e que suportam multi-paradigmas (como os ambientes que serão gerados pelo ETHOS). A dimensão de atividade de Henderson

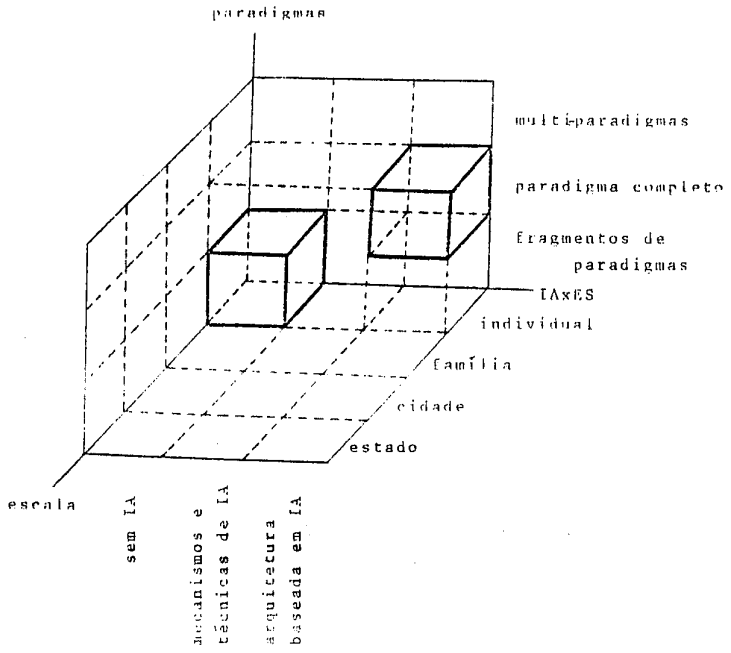
e Notkin define o escopo das atividades de especificação, desenvolvimento e manutenção que uma dada metodologia endereça. A taxonomia de Perry and Kaiser estende-se na terceira dimensão dimensão e refere-se ao ponto de vista da escala: nível individual, de família, de cidade e de estado.

Sumarizando, têm-se três dimensões que procuram enfocar três pontos de vista: interação de IA e ES, cobertura dos paradigmas metodológicos e escala. Cada ponto de vista apresenta um conjunto de estados ou valores em que pode variar. A tripla <estado$_j$, estado$_k$, estado$_l$ >, onde estado$_j$ pertence à primeira dimensão, estado$_k$ à segunda e estado$_l$ à terceira, constitui um dos possíveis cubos que podem modelar (ou classificar) ambientes.

O modelo tridimensional apresenta três níveis. O primeiro deles, refere-se aos cubos. O segundo deles refere-se ao mecanismo de instanciação, que consiste em dividir o grupo de ambientes de cada cubo em classes menores. O terceiro deles refere-se ao modelo geral de ambientes que consiste em estabelecer os componentes políticas, mecanismos e estruturas para cada instância.

A figura abaixo esquematiza o modelo tridimensional. Como exemplo, dois dos 36 diferentes cubos são realçados. O na origem dos eixos identifica os ambientes de programação e ambientes 'CASE' atuais (como por exemplo o Cépage e o Mosaico), que não possuem arquitetura baseada em técnicas de IA, tal como uma simples base de conhecimentos, suportam apenas fragmentos de paradigmas, tal como as fases de codificação, teste e depuração de determinado ciclo de desenvolvimento de software e são usados individualmente.

O outro cubo caracteriza ambientes tais como o ETHOS e o SAFO, em que toda a arquitetura baseia-se em técnicas avançadas de IA, suportam um paradigma completo de desenvolvimento de software e que ainda são usados individualmente. Note-se que no caso específico do ETHOS, por ser um meta-ambiente, pode-se adicionalmente identificar o tipo de ambientes por ele gerado: usados individualmente, arquitetura apoiada em técnicas de IA, podendo suportar mais de um paradigma completo de desenvolvimento de software.



10.2 O mecanismo de instanciação

O modelo geral de ambientes consistindo dos componentes políticas, mecanismos e estruturas proposto por Perry e Kaiser (também usado por Penedo e Riddle) continua valendo aqui. Vamos alterar um pouco o mecanismo de instanciação, pois lá existia apenas uma dimensão, ao passo que aqui temos três dimensões. Tomemos como exemplo o modelo caixa de ferramentas que é uma instanciação do modelo individual:

Modelo individual::caixa de ferramentas =

({políticas induzidas pelas ferramentas e pelas estruturas tipo shell do Unix, ...},
{editores, compiladores, loaders, depuradores, ...}, {sistema de arquivos})

Na taxonomia tridimensional, o modelo caixa de ferramentas pode ser uma instanciação do cubo representado por <fragmentos de paradigma, individual, sem IA>. Assim temos:

Modelo <sem IA, fragmentos de paradigma, individual>::caixa de ferramentas =

({políticas induzidas pelas ferramentas e pelas estruturas tipo shell do Unix, ...},
{editores, compiladores, loaders, depuradores, ...}, {sistema de arquivos})

Assim, os modelos individual::interpretativo e individual::orientado a linguagem de Perry e Kaiser são instanciações do mesmo cubo acima:

Modelo <sem IA, fragmentos de paradigma, individual>::interpretativo

Modelo <sem IA, fragmentos de paradigma, individual>::orientado a linguagem

Já o modelo individual::transformacional de Perry e Kaiser tem que ser instanciação de um outro cubo, visto que sua arquitetura subjacente é fortemente baseada em IA:

Modelo <arquitetura baseada em IA, paradigma completo, individual>::transformacional =

({construção transformacional,...}, {interpretador, máquina transformacional, ...},
{representação intermediária-árvores decoradas, ...})

Assim, com os 36 diferentes tipos de cubos obtidos com os três pontos de vista e através desse mecanismo poderoso de instanciação, pode-se classificar ambientes existentes ou em pesquisa.

10.3 O modelo n-dimensional

A taxonomia é tridimensional apenas porque quis reunir três enfoques relevantes para a classificação e análise de ambientes e porque fica fácil trabalhar com três dimensões. No entanto, esta taxonomia apresenta uma característica importante que falta às outras: flexibilidade. Flexibilidade está intimamente relacionada com robustez. Na verdade, flexibilidade significa robustez no mais alto grau.

Isto implica que, se o usuário da taxonomia quiser, pode adaptá-la às características que ele julgar relevantes para o tipo de trabalho desejado. Assim, pode ser interessante usar

apenas uma das dimensões (Perry e Kaiser, 1987), ou apenas duas delas(Hermann, Fernandes e Rangel Netto, 1989) ou as três dimensões como foi apresentado.

Além disso, nada impede o usuário da taxonomia de trocar as dimensões apresentadas por outras novas. Nada o impede, também, de utilizar mais do que três dimensões. Ou seja, a taxonomia pode ser n-dimensional, com o mecanismo de instanciação funcionando de forma similar ao apresentado. Considere que se deseja classificar ambientes de acordo com n diferentes pontos de vista. Com isso, passamos a ter n dimensões, uma para cada ponto de vista:

$<dim_1, dim_2, \ldots, dim_n>$

Cada dimensão pode apresentar um número de estados ou valores em que pode variar. $<estado\text{-}dim_1$, estado- $dim_2, \ldots,$ estado-$dim_n >$ constitui um n-cubo, onde estado-dim_1 pertence à primeira dimensão, ..., e estado-dim_n à enésima dimensão. Para cada n-cubo pode-se fazer adequadas instanciações, de acordo com as necessidades do aplicador da taxonomia:

$<estado\text{-}dim_1$, estado-$dim_2, \ldots,$ estado- $dim_n >$ :: $instância_1$
$<estado\text{-}dim_1$, estado-$dim_2, \ldots,$ estado- $dim_n >$ :: $instância_2$
⋮
$<estado\text{-}dim_1$, estado-$dim_2, \ldots,$ estado- $dim_n >$ :: $instância_m$

Para n = 1 ponto de vista, pode-se ter como exemplo o ponto de vista <escala>, instanciando-se assim a taxonomia de Perry e Kaiser.

Para n = 2 pontos de vista, pode-se ter como exemplo os pontos de vista <cobertura dos paradigmas metodológicos, escala>.

Para n = 3 pontos de vista, pode-se ter como exemplo os pontos de vista <interação de IA e ES, cobertura dos paradigmas metodológicos, escala>, instanciando-se assim a taxonomia tridimensional.

Em outro exemplo, a dimensão **tipo da geração do ambiente** pode ser de grande importância num processo particular de classificação e análise de ambientes. Os possíveis estados dessa nova dimensão são (Lucena, 1987):

- construção 'à mão' (a maioria dos ambientes atuais).
- adaptação de infraestrutura (Arcadia).
- configuração e/ou extensão de kernel (ambientes caixa de ferramentas).
- geração baseada em especificação (ambientes gerados pelo Gandalf).
- geração inteligente (ambientes gerados pelo ETHOS).

Por exemplo, o CPS pertence ao seguinte modelo 4-dimensional, que se constitui no modelo tridimensional estendido por esta nova dimensão:

Modelo <sem IA, fragmentos de paradigma, individual, geração baseada em especificação>::orientado a linguagem

Se por outro lado, está-se interessado apenas em classificar os meta-ambientes com relação ao tipo da geração dos ambientes, a dimensão **tipo do meta-ambiente** deve ser considerado, possuindo os seguintes estados:

- adaptador de infraestrutura (Arcadia).
- configuração e/ou extensão de kernel (Hoffnagle e Beregi,1985).
- gerador baseado em especificação (Gandalf).
- gerador inteligente (ETHOS).

Por exemplo, o ETHOS pertence ao seguinte modelo estendido por esta nova dimensão:

Modelo <arquitetura baseada em IA, paradigma completo, individual, gerador inteligente>

10.4 Análise das taxonomias

O objetivo da taxonomia tridimensional é organizar o conhecimento da área, classificar os ambientes e identificar direções de pesquisa dos pontos de vista da escala, de cobertura de paradigmas metodológicos e de interação entre IA e ES. É muito robusta, tem média granularidade, alta eficácia e é ortogonal, o que satisfaz os requisitos declarados para uma boa taxonomia.

É muito robusta, pois pode-se acrescentar facilmente pontos de vista como novas dimensões (constituindo-se no modelo n- dimensional), além de já apresentar o poderoso esquema de instanciação.

Tem média granularidade porque o número de categorias é sempre pequeno devido ao tipo misto de classificação. No primeiro nível, tem-se uma lista geral de requisitos composta de três classes. Cada classe tem três ou quatro estados possíveis. Tendo-se determinado um ponto tridimensional, o próximo nível consiste em estabelecer as instanciações, sempre um número pequeno, onde cada instanciação pode ser descrita por três componentes, caracterizando um terceiro nível.

Quanto à eficácia, a base é quanto ao grande poder que ela tem de classificar adequadamente o que as outras taxonomias tem problemas para classificar. No que se refere à análise, acredito que depende de mais experiência no uso da taxonomia. No entanto, considerando-se a experiência em classificar e analisar ambientes através desta taxonomia, pode-se dizer que ela promete ser muito boa, principalmente por apresentar uma perspectiva gráfica hierárquica, visto ser possível identificar os cubos e as instanciações dentro de cada cubo.

Uma análise similar aplica-se ao modelo n-dimensional, ressaltando-se o aspecto de flexibilidade encontrada no modelo.

11 Uma Meta-taxonomia para ambientes

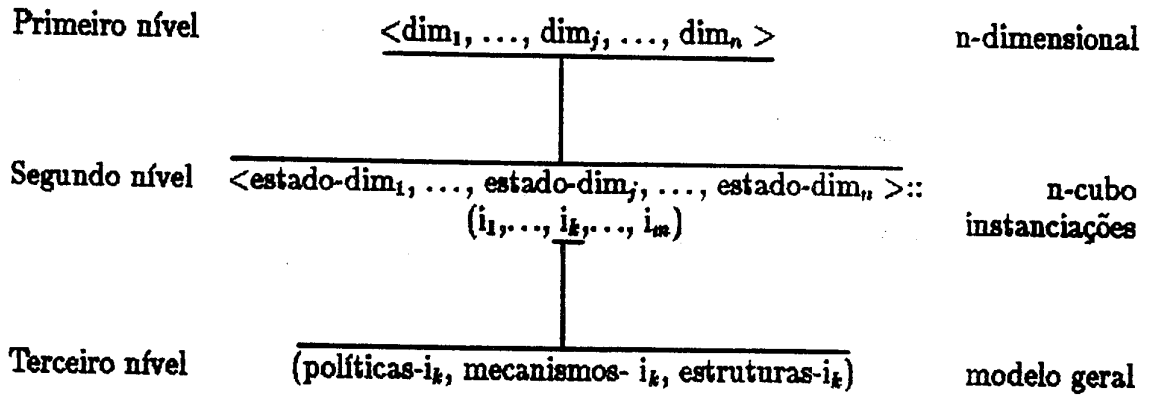
A estruturação do modelo tridimensional pode ser abstraída numa meta-taxonomia. Assim, nesta seção, uma generalização do modelo tridimensional é apresentada, juntamente com

uma metodologia de aplicação da mesma. Pesquisadores do campo da Biologia tem usado taxonomias estruturadas hierarquicamente há muito tempo. Um dos problemas com esse tipo de taxonomia é a falta de flexibilidade como definida anteriormente. Uma das maneiras mais eficientes de resolver tal problema é através do número flexível de dimensões e do mecanismo de instanciação.

Assim, o modelo meta-taxonômico apresenta uma estrutura hierárquica, composta de três níveis:

1. nível n-dimensional, onde cada dimensão espelha um ponto de vista ou um aspecto relevante;
2. nível da instanciação, onde para cada n-cubo pode-se ter m instanciações, $m \leq 0$;
3. nível do modelo geral de ambientes (Perry e Kaiser, 1987) especificado para cada instanciação, onde descrevem- se os grupos de ambientes através dos componentes políticas, mecanismos e estruturas.

A figura abaixo exemplifica a estrutura em três níveis:



O esquema para a derivação de um modelo taxonômico específico apresenta a seguinte sintaxe:

Modelo <n dimensões>::[instanciação = [(modelo de ambientes)]]

onde [...] indica opcionalidade. Assim, apenas o primeiro nível pode aparecer sozinho, enquanto o segundo e o terceiro devem vir sempre acompanhados do primeiro e o terceiro apenas se o segundo aparecer.

O modelo permite caracterizar ambientes de acordo com a granularidade desejada pelo analista. A cada nível incorporado ao modelo refinam-se os grupos de ambientes modelados. Isso se refere apenas aos dois primeiros níveis, pois o terceiro tem um papel apenas descritivo. Uma metodologia para o analista instanciar uma taxonomia particular é a seguinte:

- Estabelecer o objetivo do modelo taxonômico.
- Estabelecer o número de níveis necessários, de acordo com o objetivo estabelecido.
- Estabelecer quantas e quais dimensões são necessárias, de acordo com os pontos de vista visados. Além disso, estabelecer quais os estados ou valores para cada dimensão.
- Se necessário o segundo nível, estabelecer quantas e quais instanciações são necessárias para cada n- cubo. Um aspecto importante aqui é procurar manter o máximo de ortogonalidade.
- Se necessário o terceiro nível, estabelecer as políticas, mecanismos e estruturas para cada instanciação ou cada ambiente.

Por exemplo, o modelo para a taxonomia tridimensional apresentada, com o objetivo de comparar os ambientes quanto aos pontos de vista de escala, interação de IA e ES e cobertura dos paradigmas metodológicos, é esquematizado da seguinte forma, quando se deseja empregar apenas o primeiro nível:

Modelo <interação de IA e ES, cobertura de paradigmas metodológicos, escala>

onde o modelo

Modelo <sem IA, fragmentos de paradigmas, individual>

é uma das 36 instâncias do modelo acima, onde as suas dimensões tiveram os valores fixados, determinando um cubo em particular. Todos ambientes que satisfizerem estes valores são representantes deste cubo.

Se o objetivo é ter-se uma granularidade maior, então para cada cubo faz-se, sempre que possível e desejável, um trabalho de instanciação. Por exemplo, para o cubo acima, têm-se as seguintes instanciações (são como que fatias do cubo):

Modelo <sem IA, fragmentos de paradigmas, individual>::caixa de ferramentas

Modelo <sem IA, fragmentos de paradigmas, individual>::interpretativo

Modelo <sem IA, fragmentos de paradigmas, individual>::orientado a linguagem

Quanto ao terceiro nível, para cada instanciação descrevem-se as suas principais características através das componentes políticas, mecanismos e estruturas, de forma a caracterizar bem o grupo de ambientes que se deseja atingir. Assim temos, por exemplo, o modelo abaixo, que tem como representantes UNIX/PWB, Turbo Pascal 5.0 (Borland, 1988) e alguns ambientes para Modula-2:

Modelo <sem IA, fragmentos de paradigma, individual>::caixa de ferramentas =

({políticas induzidas pelas ferramentas e pelas estruturas tipo shell do Unix, ...}, {editores, compiladores, loaders, depuradores, ...}, {sistema de arquivos})

Esta estrutura taxonômica é muito flexível, como se pode atestar pelo exemplo seguinte. O objetivo é comparar Assistentes de Software Baseados em Conhecimento, sem se importar com a arquitetura subjacente. Esta última restrição implica que não se necessita usar o terceiro nível. Assim, pode-se modelar este grupo de ambientes de duas maneiras pelo menos, usando-se 1 ou 2 níveis. Mostow (1985) mostra que tais sistemas podem variar ao longo de muitas dimensões, incluindo:

- **Escopo**: que tipos de software são endereçados?
- **Poder**: quanto é automatizado?
- **Nível**: que parte da rota de requisitos informais até linguagem de máquina é endereçado?
- **finalidade**: que parte do ciclo de vida são endereçados?
- **Conhecimento**: que tipos de conhecimento são explicitamente usados pela máquina?

Com 2 níveis, pode-se, por exemplo, utilizar as cinco dimensões acima mais a de escala e de interação de IA e ES. Assim tem-se:

Modelo <escopo, poder, nível, finalidade, conhecimento, interação de IA e ES, escala>

Para todos os cubos onde escala=individual e interação de IA e ES=arquitetura baseada em IA, obtém-se uma instanciação para Assistentes de Software Baseados em Conhecimento.

Com 1 nível, pode-se deixar implícito que se deseja analisar Assistentes de Software (área de interação IA e ES) e não se importar com o problema de escala. Ou seja, ao se fixar valores para essas dimensões, é como se estes pontos de vista se incorporassem ao objetivo. Assim, pode- se ter o seguinte modelo:

Modelo <escopo, poder, nível, finalidade, conhecimento>

Com base nesse modelo, o projeto ISI/USC é modelado por

<escopo: {finalidade geral}, poder: {decisões feitas manualmente e realizadas automaticamente}, nível: {da especificação comportamental até programa de alto nível compilável}, finalidade: {especifica, implementa, otimiza, mantém}, conhecimento: {comportamento do programa visado, transformacões, derivação} >

enquanto o sistema de demonstração KBemacs é modelado por

<escopo: {finalidade geral}, poder: {escrita de código interativo}, nível: {código Lisp e planos de codificação}, finalidade: {implementação}, conhecimento: { idiomas de programação} >

e o CHI/Refine por

<escopo: {finalidade geral}, poder: {automático ou interativo}, nível: {de especificação até o código}, finalidade: {implementa, otimiza}, conhecimento: { transformações, axiomas de domínio, lógica} >

12 Conclusão

Apresentou-se a proposta de uma nova taxonomia, o modelo tridimensional, cujos objetivos são organizar o conhecimento da área, classificar os ambientes e identificar direções de pesquisa dos pontos de vista da escala, de cobertura de paradigmas metodológicos e de interação com IA. Uma extensão óbvia para n-dimensões foi também apresentada.

Apresentou-se adicionalmente uma meta-taxonomia para ambientes, com a qual é possível derivar facilmente um modelo particular de acordo com os objetivos e pontos de vista de uma análise em particular. Como exemplo, mostrou-se a derivação do modelo tridimensional. O autor espera refinar o modelo através de uma maior experimentação e troca de idéias com outros pesquisadores da área.

Como foi dito anteriormente, Arango et al. (1988) propõem uma estrutura conceitual para futuras discussões de aplicações de IA em ES através de uma abordagem incremental e sistemática. Um dos problemas que se tem é como particionar o universo dos problemas de ES em domínios restritos de estudo. A solução apresentada por eles consiste em introduzir a noção de um cubo n-dimensional para a tarefa de decomposição. Cada eixo corresponde a um ponto de vista de como o universo pode ser decomposto. Isto é muito semelhante ao que é feito pela taxonomia n-dimensional. Então, parece natural, como linha de pesquisa futura, tentar empregar o método taxonômico apresentado aqui como coadjuvante na determinação desses domínios restritos.

Reconhecimento: Agradeço aos Profs. Bruno Maffeo e Arndt von Staa que leram uma versão anterior deste trabalho e deram sugestões muito úteis. Em especial, agradeço ao apoio e colaboração dos Profs. Júlio César do Prado Sampaio Leite, José Lucas Mourão Rangel Netto e Carlos José Pereira de Lucena, cujas discussões valiosas ajudaram a dar forma ao trabalho; no entanto, a responsabilidade pelas opiniões aqui emitidas é inteiramente do autor.

Bibliografia

AMBRAS, J.; O'DAY, V. MicroScope: a knowledge-based programming environment. *IEEE Software*, May 1988. pp.50-58.

ARANGO, G.; BAXTER, I.; FREEMAN, P. A framework for incremental progress in the application of Artificial Intelligence to Software Enginneering. *Software Engineering Notes, 13*(1):46-50, Jan 1988.

BALZER, R. A 15 year perspective on automatic programming. *IEEE Trans. on Soft. Engineering, SE- 11*(11):1257-1268, Nov. 1985.

BAUER, . et al. *The Munchen Project CIP*. Springer Verlag, Berlin, 1985.

BRERETON, P. (ed.) Software Engineering Environments. Ellis Horwood, Chchister, UK, 1988.

COOPER, K.D.; KENNEDY, K.; TORCZON, L. The impact of interprocedure analysis and optimization in the R^n programming environment. *ACM Trans. on Programming Languages and Systems, 8*(4):491-523, Oct. 1986.

CRISCI, J.V.; ARMENGOL, M.F.L. *Introduccion a la teoria practica de la tazonomia numerica*. OEA, Washington, 1983.

DARLINGTONN, J. *The structured description of algorithm derivations in algorithmic languages*. North Holland, New York, NY, 1981.

DART, S.; ELLISON, R.J.; FEILER, P.H.; HABERMANN, N. Software Development Environmennts. *IEEE Computer*, Nov. 1987.

DOLOTTA, T.A.; HAIGHT, R.C.; MASHEY, J.R. UNIX Time- sharing Systems: The Programmer's Workbench. In: *Interactive Programming Environments*, D.R. Barstow et alii(eds.), McGraw Hill, New York, NY, 1984. pp. 353-369.

DONZEAU-GOUGE, V.; HUET, G.; KAHN, G.; LANG, B. Programming environments based on structure editors: the Mentor experience. In: *Interactive Programming Environments*, D.R. Barstow et al. (eds.). McGraw-Hill, New York, NY, 1984. pp. 128-140.

DOWSON, M. IStar — An integrated project support environment. *Sigplan Notices, 22*(1):27-33, Jan. 1987. Proceedings of the Second ACM SIGSOFT/SIGPLAN'87 Software Engineering Symposium on Practical Software Development Environments.

GOLDBERG, A. *Smalltalk-80: the language and its implementation*. Reading, MA, Addison-Wesley, 1983.

TEITELBAUM, T.; REPS, T.W. The Cornell program synthesizer: a syntax-directed programming environment. *Tech. Rep. TR 80-421*, Department of Computer Science, Cornnell University, May 1980. Communications of ACM, 24(9): 563-573, Sept. 1981.

TEITELMAN, W. A tour through Cedar. *IEEE Software, 1*:44-73, April 1984.

TEITELMAN, W.; MASINTER, L. The Interlisp programming ennvironment. *IEEE Computer*, April 1981. pp. 25-33.

TICHY, W.F. RCS — A system for version control. *Software — Practice and Experience, 15*(7):25-34, July 1985.

VIEIRA, N.J.; CARVALHO, R.L. SAFO — Um ambiente para desenvolvimento de protótipos de sistemas especialistas baseado em lógica. *Segundo Simpósio Brasileiro de Inteligência Artificial*, São José dos Campos, nov., 1985. pp. 11-14

WALTZ, D.L. Scientific DataLink's Artificial Intelligence Classification Scheme. *The AI Magazine*, Spring, 1985. pp. 58-63.

WEISER, M. et al. Panel Report on Landscaping for programming environments. In: *Proceedings of the University of Maryland Workshop on Requirements for a Software Engineering Environment*, M. Zelkowitz et al. (eds.), May 5-8, 1986. pp. 21- 49.

WILENSKY, R. Talking to Unix in English: an overview of an on-line consultant. *AI Magazine, 5*(1):29-39, 1984.

PENEDO, M.H.; RIDDLE, W.E Software engineering environment architectures. *IEEE Trans. on Software Engineering, SE-14*(6):689-696, June 1988.

PERRY, D.E.; KAISER, G.E. Models of Software Development Environments. *Proceedings of the 10th International Conference on Software Enginneering*, Singapore, 1988.

RAMAMOORTHY, C.V.; GARG, V.; PRAKASH, A. Programming in the large. *IEEE Trans. on Software Engineering, SE-12*(7):769-783, July 1986.

RAMANATHAN, J.; ROSE, J.M. TRIAD — beyond isolated systems for development-in-the-large and programming-in-the- small. *Software Engineering Notes, 10*(5):6272, Oct. 1985.

REISS, S.P. Graphical program development with PECAN Program Development Systems. *Software Engineering Notes, 9*(3):30-41, May 1984. Proceedings of the ACM SIGSOFT/SIGPLAN'84 Software Enngineering Symposium on Practical Software Development Environments.

RIDDLE, W.E. An assessment of DREAM. In: Software Engineering Environments. H. Hunke (ed.). North-Holland, Amsterdan, 1981.

SCHWABE, D.; MARTINS, R.C.B.; PESSOA, T.E. An intelligent tool for program development: an expert assistant in Jackson's JSP. Proceedings of AVIGNON 87 - Expert Systems and their Applications - E.C.C.A.J., Avignon, France, May 1987.

SIMON, H.A. Whether Software Engineering needs to be Artificially Intelligent. *IEEE Trans. on Software Engineering, SE-12*(7):726-732, July 1986.

SMITH, D.R.; KOTIK, G.B.; WESTFOLD, S.J. Research on knowledge-based software environments at Kestrel Institute. *IEEE Trans. on Soft. Engineerin, SE-11*(11):1278-1295, Nov. 1985.

STAA, A. *MOSAICO: manual do usuário.* IESA – Tecnologia de Sistemas, 1986.

STAA, A. *Ambiente de engenharia de software assistido por computador: TALISMAN.* STAA Informática, 1988.

STANDISH, . et al. Arcturus: a prototype advanced Ada programming environment. *Software Engineering Notes, 9*(3):57-64, May 1984. Proceedings of the ACM SIGSOFT/SIGPLAN'84 Software Engineering Simposium on Practical Software Development Environments.

SWEET, R.E. The Mesa programming environment. *Sigplan Notices, 20*(7):216-229, July 1985. Proceedings of the ACM SIGPLAN'85 Symposium on Language Issues in Programming Environments.

HABERMANN, A.N.; NOTKIN, D. Gandalf: software development environmennts. *IEEE Trans. on Software Engineering, SE-12*(12):1117-1127, Dec. 1986.

HARRIS, L.R. Experience with Intellect. *AI Magazine,* 5(2):43-50, 1984.

HENDERSON, P.B. (ed.)*Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments.* Boston, MA, Nov. 28-30, 1988. Software Engineering Notes, 13(5), Nov. 1988.

HENDERSON, P.B.; NOTKIN, D. Integrated design and programming environments. *IEEE Computer*, Nov. 1987.

KAISER, G.E.; KAPLAN, S.M.; MICALLEF, J. Multiuser, distributed language-based environments. *IEEE Software*, Nov. 1987. pp 58-67.

KERNIGHAN, B.W.; MASHEY, J.R. The UNIX Programming Environments. In: *Interactive Programming Environments*, D.R. Barstow et alii(eds.), McGraw Hill, 1984.

LAMB, D.A. IDL: sharing intermediate representations. *ACM Trans. on Programming Languages and Systems, 9*(3):297-318, July 1987.

LEBLANG, D.B.; CHASE,Jr., R.P. Computer-aided software engineering in a distributed workstation environment. *Software Engineering Notes, 9*(3):104-112, May 1984. Proceedings of the ACM SIGSOFT/SIGPLAN'84 Software Engineering Symposium on Practical Software Development Environments.

LUCENA, C.J.P. Subsídios para a arquitetura de uma estação de trabalho ETHOS. *Reunião de Trabalho do Projeto ETHOS*, Petrópolis, julho, 1987.

LUCENA, C.J.P.; SOARES, J. Bases de conhecimentos sobre métodos de desenvolvimento de software. *2o Simpósio Brasileiro de Inteligência Artificial*, São José dos Campos, nov., 1985. pp.61-67.

MEYER, B. Towards a taxonomy of architectural issues for programming environments. In: *Proceedings of the University of Maryland Workshop on Requirements for a Software Engineering Environment*, M. Zelkowitz et al. (eds.), May 5-8, 1986. pp. 42-49.

MEYER, B. Cépage: toward computer-aided design of software. *The Journal of Systems and Software, 8*, 1988. pp. 419-429

MOSTOW, J. What is AI? And What does it have to do with Software Engineering? *IEEE Trans. on Software Engineering, SE-11*(11):1253-1256, Nov. 1985.

OSTERWEIL, L.J. Toolpack — an experimental software development environment research project. *IEEE Trans. on Software Engineering, se-9*(6):673-685, Nov. 1983.

PARTRIDGE, D. Artificial intelligence and software engineering: a survey of possibilities. *Information and Software Technology, 30*(3):146-152, April 1988.