UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Pythonissa: uma linguagem visual para elaboração da previsão do tempo

por
Carlos Augusto Moreira dos Santos

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

Carla Maria Dal Sasso Freitas
Orientadora

Porto Alegre, fevereiro de 2001

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Santos, Carlos Augusto Moreira dos

Pythonissa: uma linguagem visual para elaboração da previsão do tempo / por Carlos Augusto Moreira dos Santos. — Porto Alegre: PPGC da UFRGS, 2001.
89 p.: il.

Dissertação (mestrado)— Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2004. Orientadora: Freitas, Carla Maria Dal Sasso.

1. Meteorologia 2. Visualização Científica 3. Interface Homem-Máquina 4. Linguagens Visuais. I. Freitas, Carla Maria Dal Sasso. II. Título.

Dedicado a Paulo Moreira dos Santos, meu pai, que infelizmente não pôde ver concluída esta tarefa para a qual tanto me incentivou.

Agradecimentos

Muitas pessoas colaboraram, direta ou indiretamente, para a conclusão deste trabalho. Agradeço inicialmente à Teresa, minha esposa, e a nossos filhos, Fernando e Pedro, que suportaram dois anos de afastamento de tudo e todos a que estavam ligados para me acompanhar na "grande aventura". Obrigado também aos familiares e amigos que nos deram tanto apoio (moral, técnico e até mesmo financeiro). Dentre estes, destacam-se Lílian e Luiz Vargas, sem os quais nada teria sido possível, e o primo Vítor Tavares e sua esposa, Margareth, que me acolheram em sua casa nos primeiros meses em que estive em Porto Alegre.

Na UFRGS, agradeço inicialmente à professora Carla Freitas pela confiança depositada ao me selecionar para o mestrado, pela orientação e pelo exemplo de vida, como pessoa dedicada à ciência e ao ensino. Pessoas como ela nos fazem manter a esperança de que nem tudo está perdido na educação deste país, apesar da patológica insensatez de muitos dos que nos governam. Alfredo Kojima contribuiu na fase inicial da implementação, acrescentando suporte à linguagem Python ao Vis5D. Não é sempre que se pode contar com o apoio de uma celebridade, e mais raro ainda é encontrar uma celebridade tão humilde e desinteressada quanto o Kojima demonstrou ser. Seria impossível também esquecer o constante apoio dos servidores do setor administrativo, da biblioteca e dos laboratórios do Instituto de Informática, sem cuja ajuda nós, pós-graduandos, jamais conseguiríamos concluir nossos trabalhos. Sou grato, em particular, às administradoras da rede, Margareth Shaefer e Cristina Nunes, pelo apoio técnico, e à Eliane, nossa fornecedora de incentivo, docinhos e café (litros dele).

Devo também lembrar os colegas da Faculdade de Meteorologia e do Centro de Pesquisas Meteorológicas da UFPEL. A professora Roseli Gomes, então chefe do CPMet, e o Conselho Departamental da Faculdade autorizaram meu afastamento pelo período necessário à realização do curso. Meu colega Rogério de Souza e Silva se dispôs a preencher a lacuna deixada por meu afastamento durante esse tempo. Alguns dos meteorologistas e pesquisadores esclareceram dúvidas sobre a elaboração da previsão do tempo, forneceram material de consulta e revisaram partes do texto. Agradeço em especial a Júlio Marques e à Dra. Natalia Fedorova pela revisão da seção 2.1. Obrigado também à professora Simone de Assis, que ocupando a chefia do CPMet me apoiou para que o trabalho fosse concluído.

Agradeço também à CAPES, pela concessão da bolsa de estudos, e aos colegas da Pró-Reitoria de Pesquisa e Pós-Graduação da UFPEL que lutaram por ela quando fomos ameaçados com sua perda.

Por fim, lembro que tanto este documento quanto o software cuja construção ele relata foram produzidos usando apenas *software livre*. Sou grato às pessoas que criaram os programas que usei. Uma lista completa de créditos encontra-se na página 89, mas quero destacar Bill Hibbard e Johan Kellum pelo apoio que nos deram quando decidimos modificar o Vis5D.

Sumário

Lista de Abreviaturas

API	Application-Program Interface
AWIPS	Advanced Weather Interface Processing System
BUFR	Binary Universal Form for the Representation of meteorological data
COLA	Center for Ocean–Land–Atmosphere Studies
CPTEC	Centro de Previsão do Tempo e Estudos Climáticos
ECMWF	European Centre for Medium–Range Weather Forecasts
EMBRAPA	Empresa Brasileira de Pesquisa Agropecuária
EMBRATEL	Embresa Brasileira de Telecomunicações
EUA	Estados Unidos da América
FTP	File Transfer Protocol
GrADS	Grid Analysis and Display System
GRIB	Gridded Binary
GTS	Global Telecommunications System
HDF	Hierarchical Data Format
INMET	Instituto Nacional de Meteorologia
INPE	Instituto Nacional de Pesquisas Espaciais
MARS	Meteorological Archiving and Retrieving System
NCAR	National Center for Atmospheric Research
NCEP	National Center for Environmental Prediction
NCSA	National Center for Supercomputing Applications
NEONS	Naval Environmental Operational Nowcasting System
NetCDF	Network Common Data Format
NOAA	National Oceanic and Atmospheric Administration
OMM	Organização Meteorológica Mundial
RADAR	Radio Detection and Ranging
RENPAC	Rede Nacional de Comunicação de Dados por Comutação de Pacotes
UFPEL	Universidade Federal de Pelotas
VC	Visualização Científica
WYSIWYG	What You See Is What You Get
WWW	World-Wide-Web
X	X Window System

Lista de Figuras

Lista de Tabelas

Resumo

Uma das maiores dificuldades encontradas pelos técnicos envolvidos na elaboração da previsão do tempo é a falta de integração entre o software de visualização usado por eles e os programas usados para escrever os boletins. Os previsores necessitam de um meio rápido e fácil de gerar previsões com outras formas de apresentação, além do formato de texto em que ela normalmente é produzida.

A partir do estudo dessas dificuldades, formulou-se a hipótese de que seria benéfico criar uma linguagem visual para a criação da previsão do tempo, que permitisse gerar tanto o texto de um boletim meteorológico quanto as imagens correspondentes. Este trabalho descreve a especificação dessa linguagem, à qual se deu o nome de *Pythonissa*. Ela foi definida usando o formalismo de grafos e se constitui de um modelo da estrutura de um boletim de previsão do tempo.

Em Pythonissa, cada região geográfica para a qual é feita a previsão é representada por um vértice em um grafo. Os fenômenos presentes na região também são representados por vértices, de outros tipos, ligados à região por arestas que denotam sua presença. Cada tipo de vértice e aresta tem mapeamentos para representações gráficas e para elementos de controle em uma interface com o usuário.

A partir da linguagem, foi implementado um protótipo preliminar, no qual é possível criar um boletim de por meio de uma interface visual e gerar o texto e a imagem correspondentes. Foi dado início, também, à construção de um *framework* para integração da linguagem a um ambiente de visualização de dados, de modo a produzir uma aplicação utilizável em um ambiente de trabalho real. Para isto foram usados o software de visualização Vis5D e a linguagem de *scripts* Python. A este framework, se deu o nome de *Py5D*.

Palavras-chave: Meteorologia, Visualização Científica, Interface Homem-Máquina, Linguagens Visuais

TITLE: “PYTHONISSA: UMA LINGUAGEM VISUAL PARA ELABORAÇÃO DA PREVISÃO DO TEMPO”

Abstract

One of the major difficulties in preparing weather forecast bulletins is the lack of integration of visualization software and tools used for writing and formating them for different media. Fast and easy-to-use tools are needed in order to help forecasters to build textual and graphical representations of such bulletins.

Based on the study of weather forecast preparation and the inherent difficulties of this task, due to lack of tools, we suggest that a visual language is a valuable tool for building both textual and graphical weather forecasts bulletins. This work proposes the visual language Pythonissa using graph grammars, as a model to specify the structure of these bulletins.

In Pythonissa different types of vertexes are used to represent each geographical region as well as the phenomena observed/to occurr there. A region vertex is linked to a phenomenum vertex by an edge to indicate the occurrence of the phenomenum in that region. Graphical symbols and control elements in the user interface are used to represent and build vertexes and edges.

A prototype was implemented to validate the process of building weather forecast bulletins using Pythonissa and generating its textual and graphical forms. A framework integrating Pythonissa into the Vis5D visualization environment, based on the scripting language Phyton, started to be designed in order to build an application suitable for use at CPMet, the Meteorological Research Center of Federal University of Pelotas, in Rio Grande do Sul, Brazil. This framework was named *Py5D*.

Keywords: Meteorology, Scientific Visualization, Human-Computer Interface, Visual Languages

1 Introdução

1.1 Motivação e objetivos deste trabalho

Ao longo de centenas anos de pesquisas os meteorologistas ampliaram muito o seu conhecimento sobre os fenômenos que regem o tempo e o clima. Hoje se fazem previsões cada vez mais antecipadas, precisas e confiáveis, beneficiando diversas atividades humanas, tais como serviços de saúde e segurança públicas, agricultura, construção civil, geração e transmissão de energia elétrica, comunicações, transportes, turismo e lazer.

A Meteorologia moderna faz uso intenso dos computadores como ferramentas de trabalho. O uso de computadores caracteriza, segundo Moura [MOU86], o início da "era científica" da Meteorologia. Os modelos matemáticos assumiram uma importância tão grande que algumas das maiores instalações de supercomputadores no mundo estão hoje em centros de previsão do tempo [DRA95].

Observando os trabalhos apresentados nas últimas edições do Congresso Brasileiro de Meteorologia [SBM96, SBM98] constata-se que há, também no Brasil, uma intensificação da pesquisa no campo da modelagem numérica. Um passo importante neste sentido foi dado com a criação, pelo Instituto Nacional de Pesquisas Espaciais (INPE), do Centro de Previsão do Tempo e Estudos Climáticos (CPTEC), onde supercomputadores estão sendo usados para executar modelos adequados às características específicas de nosso país. Graças à redução dos custos dos sistemas computacionais de alto desempenho e aos investimentos feitos em recursos materiais e humanos tornou-se possível que instituições regionais também trabalhem com modelagem numérica.

Muitas instituições elaboram previsão do tempo no Brasil. As de maior porte são o Instituto Nacional de Meteorologia (INMET) e o INPE. Outras organizações, empresas e universidades atuam no setor, a maioria delas em âmbito regional. O Centro de Pesquisas Meteorológicas da Universidade Federal de Pelotas (CPMet/UFPEL), ao qual está vinculado o autor desta dissertação, é uma dessas organizações. O CPMet é uma unidade de ensino, pesquisa e extensão, subordinada à Faculdade de Meteorologia da UFPEL. Dentre as atividades lá desenvolvidas está a elaboração de previsão do tempo para o estado do Rio Grande do Sul.

O principal motivo da execução deste trabalho foi a constatação, pelo convívio com os técnicos envolvidos na elaboração da previsão do tempo, da necessidade de um meio rápido e fácil de gerar previsões com outras formas de apresentação, além do formato de texto em que ela normalmente é produzida.

Um estudo anterior [SAN99] já analisara as ferramentas usadas na preparação da previsão por instituições dos Estados Unidos, Europa e Brasil, bem como o software que era desenvolvido por algumas delas. Constatou-se que já existiam modelos adequados para armazenamento, recuperação e representação visual de dados meteorológicos, mas que ainda não havia — e continua não havendo — um modelo canônico de interface. Alguns paradigmas de diálogo com o usuário haviam sido propostos em 1995 [REP95], entre eles a adoção de técnicas de *manipulação direta* para tornar as interfaces com o usuário mais simples e eficientes.

O trabalho aqui descrito abordou, portanto, em caráter mais abrangente, o estudo de

técnicas que permitissem construir software de apoio à elaboração da previsão do tempo. Especificamente, tratou-se das formas de representação não-textual da previsão, culminando com a especificação de uma linguagem visual. O objetivo do trabalho foi integrar a visualização de dados com a elaboração, por meio dessa linguagem, de uma representação gráfica da previsão do tempo, a partir da qual seria gerado o texto final.

1.2 Organização do texto

Além deste capítulo introdutório, o texto está subdividido da seguinte maneira:

- o capítulo 2 apresenta fundamentos teóricos e práticos sobre Meteorologia, e temas relacionados com visualização e linguagens visuais;
- o capítulo 3 discute o software usado pelos previsores, e define os requisitos de uma linguagem visual para a representação da previsão;
- o capítulo 4 apresenta a especificação da linguagem Pythonissa, de acordo com os requisitos definidos no capítulo 3;
- o capítulo 5 descreve a implementação do protótipo de um sistema de elaboração de previsões do tempo baseado na linguagem definida no capítulo 4;
- os resultados obtidos e as conclusões a que se chegou são apresentados, finalmente, no capítulo 6.

As seções 2.1, sobre meteorologia e previsão do tempo, e 4.2, sobre gramática de grafos, seriam desnecessárias em um trabalho voltado à área de conhecimento de que cada uma delas trata. Sua inclusão no texto, entretanto, facilita a compreensão do documento sem que se precise recorrer constantemente à bibliografia referenciada.

2 Fundamentos teórico-práticos

Este capítulo revisa alguns conceitos de meteorologia, visualização científica, interfaces com o usuário e linguagens visuais, considerados relevantes no contexto do trabalho. Mostra-se como cada tema se enquadra dentro do que tem sido pesquisado pela comunidade científica e como o tema se relaciona com os objetivos do trabalho.

2.1 Meteorologia, computação e previsão do tempo

A Meteorologia é o ramo da Física que estuda os fenômenos atmosféricos [TUB80]. Devido à ampla gama de aplicações dos conhecimentos meteorológicos, atualmente esta ciência se divide em vários ramos especializados: Meteorologias Física, Dinâmica, Sinótica, Aeronáutica, Marítima e Agrícola; Climatologia, Aerologia, Biometeorologia e Hidrologia (há controvérsias quanto à Hidrologia ser um ramo da Meteorologia ou da Engenharia, mas isso é irrelevante para os fins deste trabalho).

A Meteorologia Sinótica é o ramo que se dedica ao estudo dos fenômenos e processos atmosféricos a partir da observações simultâneas em uma região, com finalidade de previsão. O adjetivo *sinótico*, ou *sinóptico*, deriva do grego *synoptikós* (relativo à sinopse; que permite ver concisamente as diversas partes de um conjunto; resumido, sintético [LAV92]).

O termo "previsão do tempo" diz respeito normalmente a estimativas de curto prazo, com antecedência de até cinco dias ou um pouco mais, Já a "previsão climática" está dedica-se a prognósticos de prazo mais longo.

2.1.1 Fenômenos meteorológicos e situação sinótica

A radiação solar é a principal fonte de energia da superfície de nosso planeta. São as variações de intensidade da radiação recebida, decorrentes dos movimentos da terra — de rotação em torno de seu próprio eixo e de translação em torno do Sol — que provocam a maior parte das modificações nas condições atmosféricas, resultando em fenômenos como a chuva, o vento e outros mais.

Os fenômenos presentes em um determinado instante caracterizam a *situação sinótica* do local em que são observados (ou para o qual são previstos, tratando-se de um prognóstico). Os dados que caracterizam a situação sinótica podem ser plotados sobre um mapa, constituindo a *carta sinótica*. Existe uma simbologia padrão para essa plotagem [VIA91, p.291], que permite representar todas as informações de forma sucinta, mas para sua compreensão é necessário ter conhecimento técnico prévio. Uma carta contendo dados previstos (resultantes de um modelo matemático, por exemplo) é chamada de *mapa prognóstico*.

Os processos físicos que geram os fenômenos sinóticos são bastante complexos. Apresenta-se a seguir uma descrição breve dos fenômenos cuja previsão é o objeto de interesse deste trabalho. Este resumo foi elaborado com base na bibliografia [TUB80, VIA91, FED99] e em orientações fornecidas verbalmente pela Dra. Natalia Fedorova e pelos meteorologistas Júlio Marques e Vladair Oliveira, do CPMet.

Temperatura

A superfície terrestre é o principal receptor da energia solar que incide sobre a Terra, porque o ar tem uma capacidade muito pequena de absorver diretamente a radiação. O aquecimento do resulta, principalmente, do balanço de radiação junto à superfície. O solo transfere calor para o ar por condução. Como o ar aquecido é menos denso do que o ar frio, criam-se correntes de convecção ascendentes que levam o calor para as partes mais altas da atmosfera e contribuem também para o transporte da umidade.

Durante o dia, quando o balanço de calor na superfície é positivo, o solo armazena parte da energia absorvida, devolvendo-a à atmosfera no período noturno. Os oceanos e grandes espelhos d'água também absorvem a energia solar durante o dia, mais até do que o solo, e a devolvem à noite. Isto contribui para suavizar a diferença de temperatura entre os períodos diurno e noturno.

A temperatura do ar varia ciclicamente ao longo do dia. O valor mais baixo ocorre um pouco antes do amanhecer e o mais alto no meio da tarde. A temperatura média diária também varia ao longo do ano, tendo seu mínimo no inverno e o máximo no verão.

Um boletim de previsão sempre deve indicar a tendência da temperatura relativa ao dia anterior, no mesmo horário. Pode ser de permanecer estável, em declínio ou em elevação.

Umidade do ar e cobertura de nuvens

Ao incidir sobre os oceanos, rios e superfícies úmidas não-vegetadas, a radiação solar provoca a evaporação; incidindo sobre as plantas, provoca a evapo-transpiração. Parte do vapor d'água resultante desses processos dissolve-se no ar, gerando a umidade e transferindo energia para a atmosfera na forma de calor latente; o restante ascende às camadas mais altas, onde condensa-se ou congela-se, em partículas minúsculas, formando as nuvens, das quais se origina a precipitação.

As nuvens e a umidade do ar desempenham dois papéis muito importantes para a vida em nosso planeta: por meio da reflexão e da filtragem limitam a intensidade e o espectro da radiação solar incidente na superfície; e através do *efeito estufa* regulam as variações de temperatura da atmosfera. Sem isto, a intensidade da radiação e as variações de temperatura seriam insuportáveis para os seres vivos, porque durante o dia toda a energia recebida do sol atingiria a superfície, e à noite todo o calor absorvido durante o dia seria perdido para o espaço.

Pressão atmosférica e vento

A pressão atmosférica em um determinado local resulta do peso da atmosfera posicionada verticalmente acima dele. A pressão diminui com a altitude, porque à medida que se sobe a altura da coluna de ar até o topo da atmosfera diminui, e com ela o peso. Como o ar é muito compressível, sua densidade também diminui à medida que aumenta a altitude.

Ao ser aquecido, o ar se expande, diminuindo a densidade e, por conseqüência, a pressão. O aquecimento diferenciado do ar, em locais próximos ou distantes, gera gradientes de pressão que provocam os ventos. As variações de pressão também alteraam certas propriedades do ar, entre elas a capacidade de dissolver água.

Se a terra fosse uma esfera homogênea, lisa e sem rotação, a circulação geral das massas de ar no planeta se daria de acordo com o esquema mostrado na figura 2.1. A causa da circulação é a diferença entre as temperaturas do ar em torno do equador (zona mais quente) e nos pólos (zona mais fria), que faz com que em pequenas altitudes haja

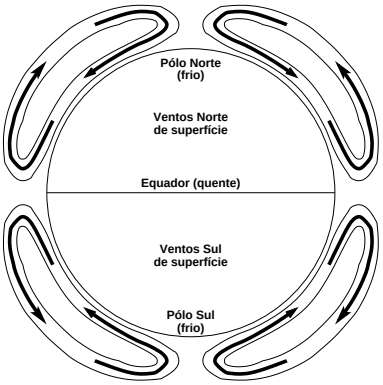


FIGURA 2.1 – Circulação geral da atmosfera, supondo um globo terrestre homogêneo e sem rotação (baseado em [TUB80]). Este modelo é apenas hipotético, pois se a terra não tivesse rotação apenas a face voltada para o Sol receberia energia.

movimento do ar no sentido dos pólos para o equador e, em grandes altitudes, no sentido contrário. A rotação do planeta, aliada à influência da orografia e outros fatores locais, entretanto, faz com que haja muitos desvios nesses movimentos.

As zonas de baixa pressão atmosférica são chamadas de *ciclones*, e as de alta pressão são chamadas de *anticiclones*. A formação, evolução e deslocamento dos ciclones e anticiclones são os principais processos sinóticos, responsáveis pela mudança do tempo em grandes regiões. Esses processos dependem do aparecimento, movimento e modificação das massas de ar e das zonas frontais entre essas massas [FED99].

Precipitação

A precipitação é o retorno à superfície terrestre da água armazenada nas nuvens, pela ação da gravidade. Este fenômeno ocorre quando as gotículas que formam as nuvens atingem um tamanho tal que o seu peso supera a ação do empuxo e das correntes de ar ascendentes.

Dependendo da temperatura em que se encontra a nuvem, pode haver apenas a condensação da água ou também o seu congelamento. No primeiro caso ocorre precipitação na fase líquida, como **chuva**; no segundo caso há precipitação também na fase sólida, como **granizo** ou **neve**. A neve, entretanto, só se forma quando a nuvem está numa temperatura tão baixa que o vapor congela diretamente, sem passar pela fase líquida (processo de sublimação), o que resulta na formação de flocos ao invés de pedras de gelo.

Descargas elétricas

Dentro da nuvem, o ar ascendente arranca elétrons livres das gotículas que caem, transportando-os para a parte superior. Com isto, a parte superior da nuvem fica carregada negativamente e a inferior positivamente.

Quando a diferença de potencial entre duas nuvens próximas — ou entre a parte inferior de uma nuvem e o solo — atinge um valor muito alto, rompe-se a rigidez dielétrica do ar. Ocorre então uma descarga elétrica, o **raio**. A descarga provoca um aquecimento brusco e uma rápida expansão do ar circundante, o que resulta na produção de uma luminosidade, o **relâmpago**, e de uma onda sonora, o **trovão**.

As trovoadas estão na maioria das vezes associadas à ocorrência de chuva, mas há um fenômeno chamado *trovoada seca*, observado quando há forte instabilidade e baixa umidade.

Nevoeiro

O nevoeiro resulta da condensação da umidade do ar, formando partículas microscópicas de água em suspensão na atmosfera. Quando as partículas são de maior tamanho, capazes de molhar os objetos com os quais entram em contato, o nevoeiro é chamado de **neblina**.

Embora existam muitos tipos de nevoeiro, seus mecanismos físicos de formação podem ser reduzidos a três processos primários [OLI98]:

Nevoeiro de radiação. Resulta do resfriamento do ar a seu ponto de orvalho. O ponto de orvalho é o teor de umidade a partir do qual o ar não é capaz de dissolver mais água do que já contém. A partir desse ponto, qualquer acréscimo de água resultará em condensação.

Nevoeiro frontal. Deve-se à adição de vapor d'água ao ar, aumentando o ponto de orvalho.

Nevoeiro de advecção. Resulta da mistura vertical de parcelas de ar úmido de diferentes temperaturas.

A formação de nevoeiro está normalmente associada à orografia, ao balanço de radiação do solo, ao ingresso de massas de ar úmido e à nebulosidade. Isto torna difícil relacionar o nevoeiro com os outros fenômenos presentes. Como regra geral, entretanto, podemos dizer que ele não se forma quando há precipitação, temperatura elevada ou ventos com velocidade maior do que 2 m/s (ventos moderados ou fortes).

Orvalho e geada

O orvalho é um fenômeno tipicamente noturno, bastante comum, resultante da condensação do vapor d'água sobre as superfícies sólidas. Já a geada, também noturna, é a ocorrência de temperaturas do ar abaixo de 0 °C, com ou sem formação de gelo. A *geada branca* caracteriza-se pelo congelamento do orvalho que se forma sobre as superfícies; a *geada negra* é aquela em que a temperatura do ar cai abaixo de 0 °C mas não se forma gelo porque o ar está muito seco.

Dependendo da combinação de fatores causadores, a geada é classificada em duas categorias:

Geada de irradiação. Normalmente associada a uma grande perda de calor pelo solo durante a noite, devido à nebulosidade baixa, ou ausente, e vento muito fraco ou inexistente.

Geada de advecção. Associada ao ingresso das massas de ar polar, de temperatura muito baixa, quando a cobertura de nuvens é inferior a 50%.

Não é possível estabelecer com rigidez as características das situações sinóticas propícias à formação da geada. Além disto, é preciso analisar também os dias anteriores ao da formação (por exemplo, [PED98]). Como regra geral, entretanto, pode-se afirmar que ela não ocorre quando a temperatura do ar está acima de 3 °C, nem quando o céu está nublado, nem quando há precipitação ou ventos de moderados a fortes.

2.1.2 Análise de dados para elaboração da previsão

Para fazer a previsão é necessário obter a maior quantidade possível de dados sobre o tempo real, não apenas no local para onde a previsão é feita, mas em uma grande parte do planeta. Para isto, a Organização Meteorológica Mundial (OMM) coordena um sistema internacional de observação, composto por estações de superfície e equipamentos de sensoriamento remoto, como radiosondas, radares e satélites.

Para facilitar o intercâmbio de dados, a OMM padronizou os equipamentos, unidades de medição e métodos de leitura a serem usados em todas as estações, bem como os horários em que devem ser feitas as observações em todo o mundo: 06:00 h, 12:00 h, 18:00 h e 24:00 h GMT (Greenwich Meridian Time). Também foram padronizados diversos formatos para armazenamento e transmissão de mensagens: SYNOP [BRA8?b], METAR/SPECI [BRA93] e TEMP [BRA8?c] são formatos de texto, do tempo em que se usava Telex e terminais do tipo TTY para transmissão; BUFR [THO94] é um formato "binário", mais recente, que não é legível pelo ser humano. Para campos (grades de dados) foi padronizado o formato GRIB [DEY98], para o qual existe uma extensão que permite incluir imagens de satélite.

Todas as instituições filiadas à OMM seguem esses padrões. Os dados coletados por elas são remetidos à OMM, que os redistribui através de uma rede de comunicações chamada *Global Telecommunications System* (GTS). Centros de pesquisa em muitos países, inclusive no Brasil, usam os dados recebidos via GTS para alimentar modelos matemáticos cujos resultados servem de base para a elaboração da previsão. O advento da Internet tornou ainda maior e mais fácil o intercâmbio desses dados.

Apesar da crescente melhoria da qualidade dos modelos matemáticos, a elaboração final da previsão continua a depender da experiência de meteorologistas treinados nessa atividade, os *previsores*. Como parte de seu trabalho, eles analisam um grande volume de dados, tanto resultantes de modelos quanto observados. Diversos tipos de software são usados para isso, principalmente os de visualização científica (alguns deles são discutidos no capítulo 3).

A previsão é feita analisando a situação sinótica corrente, comparando-a com as anteriores e com mapas prognósticos, e analisando as causas e conseqüências dessas diferenças à luz do conhecimento do previsor sobre o comportamento do tempo na área específica à qual se destina a previsão. As conclusões dessa análise são então expressas na forma de um boletim, redigido em termos adequados à compreensão pelo público leigo.

2.1.3 Previsão do tempo do CPMet/UFPEL

O CPMet divulga diariamente dois boletins de previsão, o primeiro às 10 h, com a previsão para o dia, e o segundo às 16 h, com a previsão para o dia seguinte. Uma equipe de seis meteorologistas trabalha em revezamento. Cada um fica responsável pela previsão durante uma semana. Para elaborar a previsão usam-se os resultados do modelo global do NCEP/NOAA, resultados dos modelos global e regional do CPTEC/INPE, imagens dos satélites GOES e TIROS produzidas por duas estações de rastreamento do próprio CPMet, sondagens realizadas pelo radar meteorológico e os dados observados na Estação Agroclimatológica existente no campus da UFPEL (mantida através de um convênio entre a Universidade e a EMBRAPA). A figura 2.2 mostra um esquema simplificado da estrutura existente no Centro.

Com exceção das leituras do radar, todas as informações são coletadas automaticamente. As imagens de satélite são armazenadas, via rede, em uma das estações de tra-

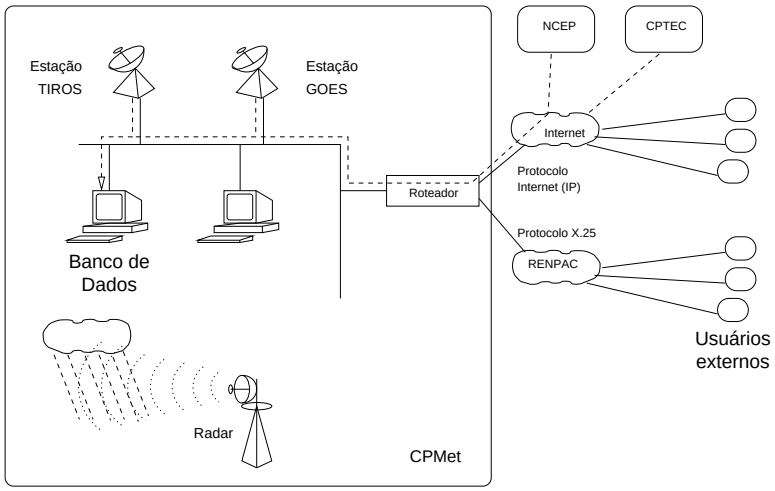


FIGURA 2.2 – Fluxo de informações no CPMet

balho usadas pelos meteorologistas. Os resultados de modelos são obtidos via Internet por processos agendados que também fazem um pré-processamento, deixando-os prontos para a visualização.

Para a visualização dos resultados de modelos os previsores têm à disposição um conjunto de *scripts* para o software de visualização GrADS (seção 3.1.1), que implementam uma interface gráfica, com *menus* que facilitam a observação de cada um dos campos disponíveis. Para visualização e tratamento das imagens de satélite usa-se o ImageMagick [IMA98], além de um programa de conversão de formatos desenvolvido no próprio CPMet.

O boletim é redigido usando um editor de textos comum. O texto é dividido em cinco partes:

Identificação: data e hora da emissão do boletim, período que ele abrange e nome do previsor que o elaborou.

Análise sinótica: uma breve descrição da situação sinótica sobre o Rio Grande do Sul em geral. Nesta seção são mencionados os fenômenos de grande abrangência geográfica observados, tais como massas de ar (quente ou frio, seco ou úmido) frentes e linhas de instabilidade.

Extremos observados: os valores máximos e mínimos de temperatura e umidade relativa do ar observados desde a véspera.

Previsão para as regiões: uma descrição da situação sinótica esperada em cada uma das regiões na qual o Estado foi dividido (figura 2.3), para diversos fenômenos meteorológicos. Esta parte do boletim é a que mais interessa para os fins deste trabalho. Ela é descrita com mais detalhes a partir da seção 4.3.2, na página 56.

Tendência para o período seguinte: as condições para as quais deve evoluir o tempo nos dias seguintes àquele coberto pelo boletim, considerando todo o Estado.

O texto da previsão, assim como uma parte dos dados usados em sua elaboração, são postos à disposição do público na Internet, no endereço `http://visimet.cpmet.`

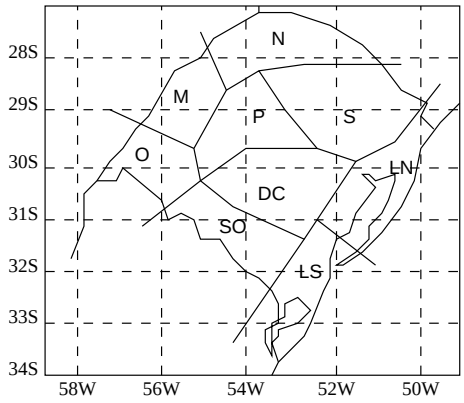


FIGURA 2.3 – Regiões do Rio Grande do Sul para as quais é feita a previsão: DC–Depressão Central, LN–Litoral Norte, LS–Litoral Sul, M–Missões, N–Norte, O–Oeste, P–Planalto, S–Serra e SO–Sudoeste.

`ufpel.tche.br`. Um grande número de usuários (mais de 700, cadastrados atualmente) obtém a previsão através da Rede Nacional de Comunicação de Dados por Comutação de Pacotes (RENPAC) [EMB9?b, EMB9?a] usando o sistema VisiMet [YAM96]. A previsão também é fornecida para jornais, emissoras de rádio e televisão, e diretamente à comunidade através de um serviço de atendimento por telefone chamado Disque Previsão.

Um exemplo completo de boletim é apresentado no anexo (página 77).

2.1.4 Problemas enfrentados pelos previsores

O maiore problema para a elaboração da previsão do tempo é a falta de integração entre o software de visualização e o editor usado para redigir o boletim. Isso é um grande entrave para os previsores, pois os obriga a fazer toda a interpretação dos dados e depois digitar a previsão.

Em sua forma atual (texto simples) o boletim é de uso limitado: não se presta à divulgação em meios “visuais”, como WWW ou televisão, e mesmo para a mídia impressa (jornal) uma representação gráfica seria mais atraente do que um simples texto.

Outro fator que dificulta a elaboração do boletim é a terminologia. Não deve haver ambigüidades ou mal-entendidos na comunicação entre o previsor e o usuário final, mas termos que possuem um significado para o técnico podem ser interpretados de forma diferente pelo leigo. Por exemplo, a divulgação pela imprensa da presença de um *ciclone* (zona de baixa pressão atmosférica), causou pânico entre a população do Rio Grande do Sul [MED99]. Para evitar problemas desse tipo é preciso uniformizar a terminologia e a organização do texto, evitando que haja variações significativas entre os boletins elaborados por membros diferentes de uma mesma equipe.

Ao ter que se preocupar com detalhes de redação os previsores acabam se envolvendo demais com a forma do boletim, quando deveriam estar concentrados no seu conteúdo. Isto é indesejável em um centro operacional, porque a previsão precisa ser distribuída o mais rapidamente possível. Para facilitar o trabalho dos previsores deveria existir um software único que permitisse, simultaneamente, visualizar os dados e elaborar a previsão. Desta forma, os previsores trabalhariam com uma única aplicação, com a qual ficariam habituados, aumentando a produtividade e eliminando os contratempos decorrentes de usar diversos programas diferentes para fazer seu trabalho.

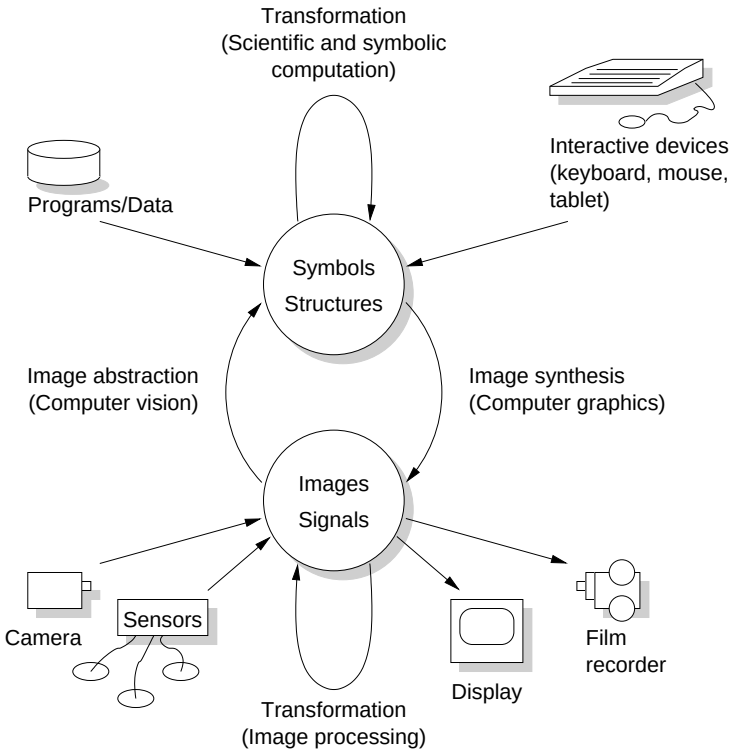


FIGURA 2.4 – Taxonomia da Visualização Científica [MCC87]

2.2 Visualização Científica

2.2.1 O que é Visualização

Visualização Científica (VC) é uma atividade multidisciplinar, que engloba a síntese e o entendimento de imagens que representem algum tipo de informação. O objetivo da VC é compreender a informação, não imitá-la. O conceito de *realismo* não se aplica à visualização com o mesmo sentido que se usa em computação gráfica. Normalmente se faz exatamente o contrário, conferindo aos fenômenos estudados atributos visuais que não existem na natureza. Usam-se, por exemplo, cores para representar valores de temperatura e setas para representar grandezas vetoriais, como a velocidade do vento.

Earnshaw e Wiseman [EAR92] fazem uma distinção entre Visualização Científica e apresentação gráfica: a primeira envolve uma atividade exploratória, muitas vezes interativa, com o objetivo de investigar as características da informação e extrair dela conclusões; a segunda destina-se apenas a comunicar resultados obtidos.

A primeira grande iniciativa em prol da disseminação da VC entre os cientistas foi o *Workshop on Visualization in Scientific Computing* (ViSC) [MCC87], no qual a visualização foi proposta como um tipo de "alternativa aos números", uma solução para o problema de se ter dados demais para analisar. A figura 2.4 ilustra o relacionamento entre diversos campos de pesquisa independentes que são unificados pela VC.

Incontáveis trabalhos têm sido publicados na literatura especializada, tanto relatando aplicações quanto discutindo as técnicas básicas necessárias à visualização. Analisando-os, pode-se identificar cinco linhas de pesquisa principais, embora alguns estudos abranjam mais de uma delas:

1. Técnicas ou ferramentas para representação visual de algum tipo particular de informação como, por exemplo, resultados de modelos climáticos ou de Dinâmica de Fluidos Computacional [MAX93, RIB94, BRY99].

2. Bibliotecas ou *toolkits* para o desenvolvimento de aplicações de visualização, tais como como VisAD [HIB92] e Vtk [SCH96].

3. Técnicas para a construção de interfaces com suporte à navegação e exploração [FUR86, BIE93, PER93, JER98, WOO93].

4. Linguagens visuais de programação, como as encontradas em AVS [UPS89, AVS99], Visual Data Explorer [IBM99], Tioga [AIK96] e Khoros [KHO99].

5. Estudos sobre os aspectos psico-cognitivos da interpretação da informação representada graficamente. Shneiderman [SHN98] revisa longamente este tema.

As representações visuais precisam ser combinadas com interfaces com o usuário, de que trata a seção 2.3.1, para conferir às aplicações o caráter interativo e exploratório mencionado. A visualização, portanto, não se restringe apenas a encontrar maneiras adequadas de representar informações graficamente, embora este seja, obviamente, o ponto chave da atividade.

2.2.2 Classificação das representações visuais

Para os fins deste trabalho, representação visual é uma imagem gerada a partir de uma certa informação que não é, necessariamente, visível na natureza. Num sistema de visualização científica, essa imagem é resultado de um processo de síntese, a partir dos símbolos e estruturas que compõem um modelo de dados, conforme mostrado na figura 2.4.

Lohse *et al.* [LOH94] consideram que representações visuais são "estruturas de dados para expressão do conhecimento". Segundo eles, há duas taxonomias para classificação de imagens (ou representações visuais: a *funcional*, baseada no uso pretendido e finalidade do material gráfico; e a *estrutural*, derivada da forma e não do conteúdo da imagem.

Como resultado de um experimento conduzido por aqueles autores, foram identificados onze agrupamentos de representações, segundo uma taxonomia estrutural. Algumas das conclusões obtidas por eles influenciaram bastante os rumos deste trabalho:

1. Formatos inusitados de representação foram considerados mais difíceis de classificar, ou seja, associar a representações conhecidas.

2. Diagramas de estrutura carregam muito mais informação espacial, não-numérica e concreta, enquanto mapas e cartogramas expressam mais informação numérica.

3. Cartogramas sobrepõem informação numérica à geográfica, sendo por isso mais difíceis de compreender do que os mapas.

4. Ícones são atraentes, mas carregam pequenas quantidades informação, de tipo muito variado, por isso podem ser difíceis de compreender.

5. Imagens fotorealistas carregam muita informação, mas para decodificá-las é preciso prestar atenção aos detalhes (uma abordagem sugerida é tratar as imagens para realçar os detalhes importantes).

6. Gráficos e tabelas foram considerados semelhantes em termos de representação visual.

7. Representações tridimensionais têm mais sentido para pessoas com conhecimento prévio sobre o assunto representado.

8. Dados temporais são mais difíceis de mostrar em gráficos estáticos do que dados cíclicos.

Uma carta sinótica, que se constitui da plotagem de símbolos e valores numéricos sobre um mapa, é um tipo de cartograma. A interpretação da carta é parte da formação dos meteorologistas. O ítem 3 mencionado acima, portanto, não deveria ser motivo de grande preocupação, pelo menos no que diz respeito a usuários técnicos. Ainda assim, por sugestão dos previsores do CPMet, deveria ser possível ter dois tipos de representação da previsão: uma detalhada, destinada a usuários técnicos, e outra simplificada, para uso do público em geral. Isto sugere a existência de diferentes níveis de mapeamento do ambiente dimensional dos dados para o do dispositivo de exibição.

2.2.3 Dimensões dos dados e de suas representações

A restrição mais severa à representação visual de dados é o fato de que o dispositivo de exibição mais comum, a tela do computador, possui apenas três dimensões:

1. posição horizontal, normalmente associada ao eixo x num sistema de coordenadas cartesianas;

2. posição vertical, normalmente associada ao eixo y;

3. cor do *pixel*.

As duas primeiras são de caráter geométrico, mas limitadas a um conjunto pré-definido de valores discretos de posições x–y, dependentes do hardware utilizado. Já a terceira dimensão, cor, não é de caráter espacial, e os valores possíveis para ela são limitados tanto pela capacidade de memória do *frame buffer* quanto pela gama de cores suportada pelo monitor.

Usando de forma muito particular alguns conceitos emprestados da matemática, as representações visuais têm um *domínio de definição* discreto, composto por um espaço bidimensional e uma grandeza escalar. O uso de diferentes símbolos e formas geométricas, construídas pelo adequado arranjo de cores dos pontos na tela, permite-nos ampliar a gama de tipos de dados e valores representáveis.

Nos boletins de previsão do tempo (ver exemplo na página 77) as posições geográficas não são descritas em termos de coordenadas, mas por nomes de regiões. Isto facilita a compreensão pelos usuários não-técnicos, pois os nomes lhes são familiares e podem ser compreendidos observando as subdivisões em um mapa. Os valores dos atributos de cada fenômeno são expressos usando termos que mais *qualificam* a situação sinótica em cada região do que *quantificam*. De fato, a temperatura é o único fenômeno expresso numericamente, uma vez que os usuários estão, com certeza, muito mais acostumados a observar valores em um termômetro do que, por exemplo, nos instrumentos usados para medir velocidade do vento (anemômetro) e volume de precipitação (pluviômetro). Alguns fenômenos são expressos quantitativamente, mas usando um sistema de classificação (vento

de intensidade *fraca*, *moderada* ou *forte*, por exemplo). Para outros fenômenos, finalmente, é apenas mencionada a sua presença ou não na região, sem quantificação (ver seção 4.3.3).

Passamos, então, de um sistema de coordenadas baseado em latitude, longitude, altitude, variável e valor, como o usado no Vis5D (ver seção 3.1.4), para um outro mais abstrato, baseado em região, fenômeno e valor, este último nem sempre numérico.

2.3 Interfaces com o usuário

2.3.1 Necessidade e importância das interfaces

O termo *interface* significa "superfície de contato entre dois meios com densidades diferentes" [LAV92]. No caso da relação software-usuário, densidades de detalhamento ou proximidade da organização do computador. Quanto mais alto o *nível* de uma interface, mais próxima ela está do ser humano e mais distante da máquina. Os elementos que compõem a interface funcionam, então, como a linguagem por meio da qual se dá essa comunicação.

O usuário não interage com a máquina, mas com o software que está sendo executado nela. O hardware de entrada/saída serve então como o mecanismo por meio do qual o software recebe/envia informação de/para o usuário. Diferentes tipos de software executados na mesma máquina, mesmo simultaneamente, podem estabelecer políticas de interface também diferentes. Por esses motivos, os termos "interface homem-máquina" e "interface humano-computador" não identificam com precisão o processo. Entretanto, como ressalta Bos [BOS93], "o usuário normalmente não é capaz de distinguir entre o remetente da mensagem e seu portador".

As primeiras interfaces gráficas introduziram a técnica de facilitar a associação de idéias por meio de *metáforas*, alusões a objetos do mundo real aos quais estariam relacionadas operações conhecidas. Uma das metáforas mais bem sucedidas foi a da mesa de trabalho (*desktop*), originária do Xerox Alto e popularizada com o Macintosh, da Apple. Ela está presente até hoje, tanto no próprio Mac em outros sistemas de janelas.

O que se espera ao prover uma interface visual é que a existência de um modelo de diálogo comum, usado por todos os programas, facilite o trabalho do usuário. Pelo menos uma parte do conhecimento adquirido sobre como usar uma aplicação possa ser reaproveitado para as demais. O modelo de diálogo provido por falta deve ter, portanto, um conjunto razoável de atributos que atenda às necessidades do usuário médio. Definir "razoável" e "usuário médio" é um desafio a ser vencido por quem projeta e implementa a interface.

Outro aspecto a considerar na construção de aplicações interativas é a integração de muitas delas dentro de um ambiente de trabalho de modo a prover um todo harmonioso. Para isto as aplicações devem possuir duas características:

Consistência. Um software tem modelo de diálogo consistente quando todos os elementos de sua interface são desenvolvidos em concordância com as mesmas regras de estilo, aliadas à uniformização de aparência provida por um *toolkit* (ver seção 2.3.5). Esta é uma característica individual.

Coerência (entre aplicações). Decorre da adesão de todas as aplicações ao mesmo modelo de diálogo.

Coerência é difícil de se obter quando se desenvolve software para o ambiente de janelas disponível na maioria das estações de trabalho científicas, o *X Window System*. X foi concebido para prover total configurabilidade e por isto não padronizou uma política de interface com o usuário [SCH86]. Isto resultou na criação de inúmeras interfaces, cada uma com seu próprio modelo de diálogo. James Gettys, um dos criadores do sistema, admite que apesar de tanta flexibilidade ser uma das maiores forças do X, por vezes ela acaba se tornando também sua maior fraqueza [GET98].

2.3.2 Por que as interfaces são difíceis de construir

Uma questão que sempre se levanta ao tratar da construção de interfaces com o usuário é "Por que é tão difícil?". Shneiderman [SHN98] discute longamente o problema, apresentando os resultados de muitas pesquisas em torno do tema, mas essa discussão é bastante antiga: já em 1993, Myers [MYE93] enumerava diversas razões pelas quais tanto o projeto quanto a implementação de interfaces eram inerentemente difíceis, concluindo que a pesquisa sobre o tema continuaria a prover melhores teorias, metodologias e ferramentas, mas que os problemas dificilmente seriam resolvidos. Interfaces "homem-computador" continuariam sendo difíceis de projetar e implementar. Advertia ainda que com novos estilos de interação, baseados em fala, reconhecimento de gestos, agentes inteligentes e visualização 3D, o esforço envolvido só poderia aumentar.

O ponto chave da concepção de uma interface está em saber que mecanismos de interação são os mais adequados ao tipo de tarefa a ser executada com auxílio do computador, de acordo com o perfil do usuário. Como exemplo prático, Grote e Bullock, descrevendo a interface criada para o sistema meteorológico WFO-Advanced [GRO97], relatam a dificuldade de incluir nas especificações do software "o modo como o usuário utiliza as várias funções da estação de trabalho no contexto do desempenho de suas tarefas".

Satisfazer o usuário não é tarefa simples quando o desenvolvedor não sabe exatamente quem é esse usuário e quais são as suas necessidades e preferências. O mesmo software pode ser usado por pessoas com diferentes níveis de conhecimento e com opiniões divergentes, possivelmente conflitantes, a respeito de o que torna um programa mais fácil de usar. Outro exemplo: Shneiderman, já citado, afirma que usuários inexperientes podem ser beneficiados por mecanismos visuais, como *menus* e botões, mas usuários experientes, que sejam bons datilógrafos, são mais produtivos usando comandos de teclado. Isto fica evidente ao se ver a opinião de um *megahacker* como Andrew Tanenbaum sobre as interfaces gráficas e sistemas baseados no paradigma *What You See Is What You Get* (WYSIWYG) [TAN2000]:

> "Digitar '`rm x y z`' é muito mais rápido do que clicar cinco vezes e então ter de convencer o sistema de que você realmente, de verdade, quer dizer isso e que não é um erro e que você é um adulto emancipado maior de 18 anos que compreende as conseqüências e ainda assim quer fazê-lo."

Exageros à parte, para atender a qualquer tipo de usuário a solução seria prover ambos os mecanismos, interface gráfica e de linha de comando. O acréscimo de mais funcionalidade, entretanto, não torna o software necessariamente mais fácil de usar. Além disto, prover muitas maneiras diferentes de realizar cada tarefa pode aumentar consideravelmente o tamanho do programa se o número de tarefas também é grande. E quanto maior o programa, mais difícil ele se torna de escrever, documentar e manter.

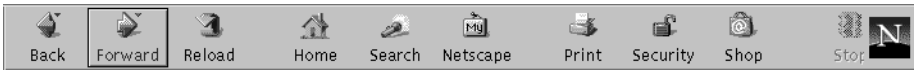


FIGURA 2.5 – Barra de ferramentas do Netscape Navigator. O comando correspondente a cada ícone deveria ser inferido pelo usuário a partir da associação da figura a uma operação conhecida, entretanto cada um deles é acompanhado de uma legenda. Os ícones ajudam a localizar os botões mas as legendas são necessárias para explicar seu significado.

Não há como garantir que aquilo que o desenvolvedor do software entende como sendo o melhor (por exemplo, a representação visual de um certo tipo de informação) será interpretado de modo idêntico pelo usuário. Caso não haja essa identidade, o elemento visual passa a desempenhar um papel secundário na interface, ou meramente decorativo. A figura 2.5 mostra um exemplo disto.

Uma vez que não há como saber por antecipação como o usuário reagirá a uma determinada interface, torna-se necessário muitas vezes recorrer a dispendiosos testes de usabilidade. Estes, segundo Shneiderman, nem sempre produzem resultados totalmente fiéis ao que ocorreria em uma situação de uso real do software.

Testar a interface de um sistema apenas depois de ter seu desenvolvimento concluído também pode ser muito arriscado, pois quanto mais adiantada a construção do software mais dispendioso se torna alterá-lo. Por isto se recorre constantemente a protótipos da interface, permitindo que ela seja testada antes de o restante da aplicação ser implementado. A prototipação permite aos projetistas e desenvolvedores ter uma visão mais realista da reação que o usuário terá diante do software. Isto é muito importante para evitar a ocorrência do que Hobart [HOB95] chama de “esquecer o usuário”.

Muitos programadores são *hackers* que programam usando editores de texto como VI e EMACS. Podemos avaliar por isto a distância que os separa dos usuários do software escrito por eles. Ferramentas de programação precisam ser tão configuráveis quanto possível, para tornar produtivo o trabalho do desenvolvedor, mas não se pode querer que não-hackers, ainda que técnicos, estejam aptos — ou sejam obrigados — a configurar cada detalhe da interface de um programa.

Por outro lado, é ilusão imaginar que o usuário não terá de se adaptar aos limites do software: a relação entre os dois é sempre limitada a um “máximo denominador comum”, e o computador ainda é o mais limitado dos dois em termos de capacidade interativa. Ainda assim, é indispensável que o usuário mantenha o controle — ou tenha a ilusão de mantê-lo — mesmo em software de uso científico, no qual é aceitável um nível de complexidade maior do que o encontrado em software “de escritório”. Para isto a interface deve prover, tanto quanto possível, uma sensação de resposta visual imediata, como a que se obtém usando manipulação direta.

2.3.3 Manipulação direta

Para que uma interface com o usuário seja bem sucedida, a semântica dos comandos tem de ser dedutível do comportamento visual. Esse relacionamento entre representação visual (sintaxe) e resultados obtidos (semântica) baseia-se num conjunto de metáforas que ajudam as pessoas a inferir o resultado de um comando pela associação entre a imagem que lhes é apresentada e conceitos aos quais elas já estejam acostumadas. Por exemplo, arrastar um objeto de um ponto para outro deve resultar na transferência do objeto da posição que ele originalmente ocupava no ambiente para a posição correspondente às

coordenadas do ponto na tela onde o objeto foi solto. De modo semelhante, arrastar um objeto para um ícone no formato de uma lata de lixo deve resultar em sua eliminação (o objeto foi jogado fora).

O termo "manipulação direta" foi cunhado por Shneiderman, em 1983 [SHN83]. Segundo ele "a chave na criação de um sistema de manipulação direta é conseguir uma representação adequada ou *modelo da realidade*" [SHN98]. Os elementos componentes deste modelo são então representados visualmente. A técnica de interação consiste em fazer com que os dados da aplicação sejam manipulados *através* de suas representações gráficas, usando-se os dispositivos de hardware disponíveis, tais como teclado e *mouse*, comumente encontrados nas estações de trabalho.

Segundo Bos [BOS93], "o raciocínio por trás da manipulação direta é que é mais fácil usar uma ferramenta que mostra seu efeito imediatamente do que programar uma ação para ocorrer mais tarde. Por outro lado, se a ação precisa de muitos ajustes, pode ser útil anotá-la para referência posterior". Muitos programas seguem este princípio por meio de opções de salvamento de estado. Metview (seção 3.1.2), por exemplo, permite salvar os parâmetros de uma visualização na forma de um ícone para ser repetida posteriormante.

O princípio de funcionamento da manipulação direta sugere, portanto, a existência de pelo menos dois tipos de componentes na aplicação: o modelo (dados) e sua representação visual, através da qual se dá a interação com o usuário. Como saber, entretanto, que funções deve desempenhar cada componente? E como projetar e implementar a aplicação da forma mais eficiente e fácil de manter? Para atender a esses requisitos foi definida uma arquitetura para as aplicações, descrita a seguir.

2.3.4 O paradigma Modelo-Vista-Controlador

A arquitetura Modelo-Vista-Controlador (*Model-View-Controller* — MVC) foi introduzida e popularizada através da linguagem de programação Smalltalk, a partir da versão Smalltalk-80 [LAL91]. Esta versão teve um importante papel para a arte da programação por, entre outras coisas, implementar uma interface interativa multijanelas totalmente baseada no conceito de orientação a objetos.

MVC tornou-se uma abordagem muito aceita para construção de interfaces visuais, uma espécie de *framework*, cujos componentes tiveram suas características incorporadas, mais tarde, a diversos padrões de projeto (*design pattern*s) [GAM95]. A arquitetura se baseia na divisão de papéis entre três tipos de componentes:

Modelo. Representa e gerencia o comportamento dos dados da aplicação, provendo informações sobre eles (geralmente para a vista) e realizando mudanças (na maioria das vezes, requisitadas pelo controlador).

Vista. Encarregada da apresentação dos dados representados pelo modelo. A vista produz e mostra ao usuário uma representação gráfica do estado do modelo, que é atualizada sempre que ocorre uma mudança nesse estado.

Controlador. Responsável pela interação do usuário com o modelo. Em uma interface gráfica, o controlador interpreta a entrada de dados, através do teclado ou *mouse*, indicando ao modelo ou à vista as alterações necessárias.

Cada um desses componentes é encarregado de uma tarefa específica e é funcionalmente independente dos demais. A vista conhece explicitamente o modelo e o controlador; o controlador conhece explicitamente o modelo e a vista; entretanto, não há conexão

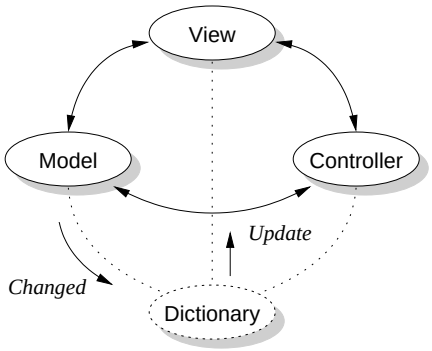


FIGURA 2.6 – Componentes de uma aplicação baseada no paradigma MVC [LAL91]

explícita entre o modelo e os outros dois. A comunicação entre os três se dá segundo as convenções estabelecidas na interface (conjunto de métodos e seus argumentos) de cada uma das classes.

Na implementação existente no Smalltalk-80 o modelo não possui referências explícitas nem ao controlador nem à vista. Um quarto objeto, chamado *dicionário de identidade*, implementa a dependência entre vista e modelo (figura 2.6). O modelo sinaliza mudanças em seu estado usando o método *changed*, resultando no envio pelo dicionário de uma mensagem *update* a cada vista.

A definição de uma clara separação de papéis entre os três componentes é a principal característica da abordagem MVC como estratégia de projeto. Normalmente, cada componente é desenvolvido como um objeto separado, de acordo com a sua função dentro do paradigma. Ao final, todos são reunidos, com a possibilidade de ser substituído aquele que não se apresentar adequado, sem que tal alteração cause impacto nos outros, desde que preservadas as interfaces entre eles.

Para combinar manipulação direta e MVC é preciso que, na implementação, o objeto que desempenha o papel de vista seja capaz de realizar operações que modificam o estado do modelo (*drag-and-drop*, por exemplo). Esse comportamento normalmente pertencente ao controle. Lalonde e Pugh [LAL91] chamam esta combinação de papéis em um único objeto de *relaxamento do paradigma*, e identificam duas razões para fazê-lo:

- quando a janela é muito simples, pode-se fundir vista e controlador;
- quando um modelo é distribuído ou carece de funcionalidade, pode-se criar um modelo virtual que contém o modelo real e a funcionalidade.

A essas duas, acrescenta-se uma terceira: quando a vista incorpora funções de controle, como é necessário para implementar a técnica de manipulação direta, também é necessário fundí-la ao controlador. Os papéis de vista e controlador continuam a existir, mas são desempenhados pelo mesmo “ator”.

Definida uma arquitetura para o sistema, é preciso decidir de que modo cada um dos componentes será implementado. A parte mais complexa dessa implementação reside na gerência dos dispositivos de entrada-e-saída e no tratamento das ações dos usuários. Ao invés de reescrever, do zero, uma interface diferente para cada aplicação, é normalmente mais vantajoso usar um *pacote* preexistente para construção de interfaces.

2.3.5 Construtores de interfaces e de aplicações

Conforme já se discutiu na seção 2.3.1, a interface de uma aplicação com o usuário tem de ser consistente e é recomendável também que tenha coerência com as das outras aplicações que compõem o ambiente de trabalho. A padronização das interfaces é obtida quando os desenvolvedores de cada aplicação aderem ao uso comum de duas ferramentas de projeto e implementação:

- um conjunto de regras que define como serão a aparência e o comportamento das aplicações. Essas regras constituem um *manual de estilo*;
- um conjunto de bibliotecas que ofereçem tipos de dados e funções de uso comum entre todas as aplicações. Essas bibliotecas constituem o que se convencionou chamar de *toolkit*.

Toolkits e manuais de estilo fazem parte de uma classe específica de software: a dos Sistemas de Gerenciamento de Interface com o Usuário (*User Interface Management Systems* — UIMS), que ajudam a projetar e implementar as interfaces. Em conjunto, eles definem qual será o *look-and-feel* (aparência e sensação) da aplicação. Como exemplos, pode-se citar o toolkit Motif e seu manual de estilo correspondente [OPE97a, OPE97b].

Indo um passo além dos UIMS, os Ambientes de Desenvolvimento de Interfaces com o Usuário (*User Interface Development Environments* — UIDE) são ferramentas usadas para automatizar a construção e/ou geração de interfaces para aplicações.

Uma discussão detalhada sobre UIMS/UIDE é feita por Bos [BOS93] e não cabe aqui repeti-la. É importante, entretanto, ressaltar que a combinação das técnicas de construção de interfaces gráficas com o paradigma de manipulação direta deu origem a uma outra classe de software, que é de grande interesse para os fins deste trabalho: a dos sistemas de Programação para Usuários Finais (*End-User Programming* — EUP). Esses sistemas de "programação para não-programadores" assemelham-se aos UIDE, mas destinam-se a permitir que pessoas sem formação ou conhecimento específico em programação possam programar também, normalmente por meio de uma linguagem visual.

2.4 Linguagens visuais e programação para não-programadores

Em um estudo ancestral sobre linguagens visuais, Shu [SHU86] afirmava que, tradicionalmente, as estruturas das linguagens de programação eram baseadas em representações textuais (unidimensionais, comando após comando) em decorrência da organização interna dos computadores. Para ele, as linguagens convencionais continuariam a ser usadas no futuro, mas para encorajar o desenvolvimento da "computação por usuários finais" (*end-user computing*), deveriam existir representações mais amenas para a compreensão humana. Shu mencionava três premissas com base nas quais linguagens de programação visuais cumpririam esse objetivo, embora ele mesmo admitisse não saber até que ponto elas se aplicavam no ambiente computacional:

1. Pessoas, em geral, preferem figuras ao invés de palavras.
2. Figuras são mais poderosas do que palavras como meio de comunicação. Elas podem conter mais significado em uma unidade de expressão mais concisa.
3. Figuras não têm as barreiras de idioma que as linguagens naturais têm. Elas são entendidas pelas pessoas a despeito de que idioma elas falam.

TABELA 2.1 – Classificação das linguagens visuais [FRE92]

Segundo o aspecto funcional (aplicação a que se destina a linguagem)	Segundo a abordagem usada para construçãodas sentenças da linguagem
Especificação de software e programação	Icônicas, baseadas na manipulação de ícones
Especificação de interfaces com o usuário	Diagramáticas, quando a linguagem é baseadaem diagramas
Visualização de estruturas de dados, algoritmos e programas	Tabulares, baseadas no preenchimento de tabelase formulários
Consulta a bancos de dados	Demonstracionais, quando as construções sãoexemplificadas pelo usuário, através detécnicas interativas

Um estudo mais recente [LOH94] mostra que há um grande número de aspectos culturais dos quais depende a interpretação que se faz das imagens, uma vez que a associação do elemento visual com outras informações depende de um conhecimento prévio.

A estrutura de uma linguagem de programação (elementos sintáticos e semântica de sua organização) depende de dois fatores: a finalidade a que ela se destina e os mecanismos usados para construir os programas. Freitas [FRE92] propôs a classificação das linguagens visuais esses fatores como critérios (tabela 2.1). A segunda classificação, entretanto, é válida apenas para sistemas de programação visual implementados na forma de aplicações interativas. Linguagens destinadas à construção de interfaces com o usuário, sejam elas declarativas, como UIL [OPE97a], ou procedurais, como Tk [OUS94], permitem que a programação seja feita usando apenas um editor de textos. O uso de um UIDE apenas torna o trabalho mais fácil.

Nos “bons tempos”, quase todos os microcomputadores vinham acompanhados de um interpretador BASIC. Aprender a usar o computador incluia também o aprendizado dessa linguagem, de modo que cada novo usuário era, potencialmente, um novo programador. HyperCard [APP95], lançado em 1987, foi o primeiro ambiente popular de programação baseado inteiramente em uma interface gráfica, destinado a substituir uma linguagem de programação convencional por uma linguagem visual. Antes disso, porém, VisiCalc já havia propiciado aos usuários um sistema para programação de cálculos, baseado em uma metáfora de planilha e na técnica de *programação por exemplo* (*Programming By Example — PBE*) [BRI99].

A importância histórica do VisiCalc transcende a simples definição de uma nova classe de aplicativos: ele tornou viável a pessoas comuns o uso do computador para a solução de problemas para os quais não existiam programas específicos, sem que tivessem de escrever uma única linha de código em BASIC ou outra linguagem convencional. Em uma entrevista concedida à revista Byte [LIC89], os criadores do programa mencionam que os compradores iam às lojas comprar “um VisiCalc”. Só então ficavam sabendo que para poder usá-lo teriam de comprar também um Apple II. Isto ajudou a mudar completamente a noção que as pessoas tinham do computador, passando a encará-lo primeiro como instrumento pessoal de trabalho e depois como mais um eletrodoméstico.

O que caracteriza um sistema de programação para usuários finais, fundamentalmente, é o fato de ele permitir que a lógica do programa gerado seja deduzida a partir de uma estrutura (um *modelo*) montada através de uma interface visual. No caso da folha de cálculos, por exemplo, a ordem de avaliação das expressões contidas nas células é determinada com base na sua disposição ao longo de linhas e colunas e/ou nas dependências

existentes entre células; uma dependência cíclica é resolvida por um laço de execução de cálculos.

A combinação das técnicas de visualização com interfaces gráficas e manipulação direta propiciou a criação de linguagens visuais destinadas à "condução de computações científicas" [BUR94], tais como AVS [AVS99], DX [IBM99] e Khoros [KHO99]. Essas três ferramentas popularizaram o paradigma *caixas e setas*, usado posteriormente em Tioga [AIK96], e constituem sistemas de programação voltados para usuários técnicos.

3 Software usado em pesquisa e previsão do tempo

O software "gráfico" usado usado em meteorologia pode ser enquadrado em duas categorias: a primeira é a dos programas de visualização aplicados em pesquisa, com finalidade exploratória; a segunda é a dos sistemas integrados, voltados especificamente para a elaboração da previsão, destinados a prover um ambiente de trabalho completo para o previsor, inserido no fluxo de informações da instituição.

Este capítulo apresenta um breve resumo das características do software que, segundo os relatos encontrados na literatura, estava sendo usado e/ou desenvolvido por instituições meteorológicas da Europa, Estados Unidos e Brasil.

3.1 Software de exploração

3.1.1 GrADS

Grid Analysis and Display System (GrADS) [DOT95] é um software de visualização e tratamento de dados desenvolvido por Brian Doty, no Center for Ocean–Land–Atmosphere Studies (COLA), e mantido pelo esforço individual dele e outros voluntários. O código-fonte é de acesso público, o que facilita seu porte para diversas plataformas de hardware e sistemas operacionais (há versões para UNIX, com X Window System, MS-DOS, Windows NT e Macintosh). Não é permitido distribuir versões modificadas do GrADS. Doty argumenta que isto poderia levar a uma fragmentação do desenvolvimento do programa, além de dificultar o apoio aos usuários.

GrADS implementa um modelo de dados de quatro dimensões: latitude, longitude, nível de pressão atmosférica e tempo. Suporta tanto dados em reticulados (grades, igualmente espaçadas ou não) quanto pontuais (de estações). Os formatos de arquivos de dados suportados são binário (formato próprio), GRIB e NetCDF [UNI99]. GrADS não reconhece o formato BUFR para observações nem imagens em formato *raster*, como as de satélite. Não há nenhum suporte para comunicação ou acesso a bancos de dados, apenas para arquivos.

Os arquivos de dados são compostos por um descritor, também chamado "arquivo de controle", no formato texto, e um arquivo de dados (binário). Existem comandos de baixo nível para tratamento de arquivos GRIB e arquivos auto-descritivos (*Self-Describing Files*, SDF, que possuem *metadados* descritores do conteúdo) ou descritos externamente (*eXternally-Described Files*, XDF) baseados nos formatos HDF [NCS99] e NetCDF. Há também um script para conversão entre esses formatos.

Imagens podem ser exportadas para um *metafile* e posteriormente visualizadas ou convertidas para PostScript ou GIF por utilitários externos. As versões mais recentes do GrADS podem salvar a imagem da tela de visualização diretamente, nos formatos GIF, JPEG, PostScript, MIFF, PCX, BMP e XPM, entre outros, usando a biblioteca do ImageMagick [IMA98].

A interface com o usuário do GrADS é por linha de comando. Há uma linguagem própria, com sintaxe semelhante à do C, dotada de comandos para tratamento de arquivos, seleção dos dados dentro do ambiente dimensional, criação de títulos, legendas, botões e *menus* dentro da janela de visualização, *plotagem* de dados e gravação de resultados.

Formas gráficas primitivas (texto, linhas, retângulos, retângulos cheios, polígonos, textos e marcadores) também podem ser desenhadas. Há um recurso muito primitivo de criação de caixas de diálogo e outros *widgets*, baseado no *toolkit* Xaw [PET94], mas não há uma linguagem de programação visual.

GrADS é implementado como um programa monolítico e não define uma API para extensão (não é possível incorporar funções externas escritas em C ou FORTRAN, por exemplo). A linguagem de script, entretanto, é bastante poderosa e provê funções pré-definidas para tratamento dos dados. Novas funções podem ser definidas pelo usuário em scripts. Um script pode invocar a execução de outro script ou de programas externos.

3.1.2 Metview

Metview [ECM97] é uma ferramenta interativa para acesso, manipulação e visualização de informações meteorológicas, desenvolvida em conjunto pelo INPE/CPTEC e o European Centre for Medium–Range Weather Forecasts (ECMWF). O software foi concebido para ser integrado à rotina de trabalho do ECMWF e baseia-se em dois padrões de software daquela instituição: a biblioteca gráfica MAGICS (para visualização 2D) e o sistema de recuperação de dados meteorológicos MARS (Meteorological Archiving and Retrieving System).

Metview pode operar em um ambiente totalmente distribuído, com módulos sendo executados de forma coordenada em estações e servidores UNIX diferentes. O programa incorpora uma interface de comunicação cliente-servidor para obter dados a partir de servidores MARS. Na falta de um servidor, pode-se ler diretamente arquivos em disco, nos formatos GRIB (para campos, incluindo o formato GRIB estendido para imagens de satélite) e BUFR (para observações), além de um formato de matriz específico do Metview. Dados recuperados em uma requisição ao MARS podem ser salvos nos formatos GRIB, BUFR ou texto. Pode-se realizar algumas operações especiais com os dados: filtragem de observações, conversão de matrizes em isolinhas e interpolação de dados de observação pontuais para conversão em campos numéricos.

A interface visual, orientada a objetos, é desenvolvida sobre X Window System e Motif. Tanto os dados quanto as funções são representados como ícones, manipulados através do *mouse*. Os dados podem ser apresentados como linhas e áreas de contorno, linhas de corrente, vetores e símbolos, e pode-se incluir na imagem linhas costeiras e divisões políticas. Seqüências de imagens podem ser animadas como um “filme” ou dispostas lado-a-lado, em diversos quadros. Uma limitação do Metview é que o único formato em que as imagens podem ser salvas é PostScript.

O sistema possui uma linguagem de programação de *macros*, com sintaxe semelhante à da linguagem C, que inclui funções de alto nível para tratamento dos tipos de dados suportados e funções específicas para scripts, que não estão disponíveis em modo interativo. Scripts possibilitam que Metview implemente o recurso salvamento de estado, mencionado na seção 2.3.3: o resultado de uma visualização pode ser salvo como uma macro, identificada por um ícone na janela de trabalho que ao ser *clicado* provoca a execução do script.

Em um script do Metview, pode-se chamar funções externas escritas em FORTRAN, C e C++, ou scripts do *shell* do UNIX. A API permite que funções recebam e passem argumentos de/para o Metview. Para a criação de novas macros há um editor integrado à interface gráfica. Há também um temporizador que executa macros em horários agendados pelo usuário. Combinados, macros e agendamento de operações permitem construir

aplicações que operem sem intervenção do usuário [COR98].

Karhila [KAR95] classifica Metview como um *wrapper* (empacotador) de MARS e MAGICS, aumentado com uma poderosa linguagem meteorológica de macros. Considerando as características apresentadas, entretanto, ele poderia ser visto não só como uma ferramenta para visualização e manipulação de dados, mas também como um *framework* para construção de aplicações por meio da programação de macros. A principal limitação do Metview como framework, é que sua extensibilidade é restrita pelos recursos da API e pelas características da linguagem de macros (eminentemente procedural; sem suporte à definição de novos tipos de dados; nenhum acesso às funções primitivas do sistema operacional ou do sistema de janelas). Isso torna o Metview inadequado para implementação de sistemas que exijam acesso a recursos de programação baixo nível.

3.1.3 NCAR Graphics

NCAR Graphics [NCA95] é um software de visualização científica desenvolvido pelo National Center for Atmospheric Reasearch (NCAR), a partir do início da década de 80. Até a versão 3 ele se constituía apenas de uma biblioteca de funções para uso com FORTRAN ou C e um conjunto de programas utilitários. A partir da versão 4 foi incluída uma linguagem de comandos e um modelo de dados baseado em NetCDF. O software roda em UNIX e X Window, baseando seu suporte a gráficos no padrão GKS [ANS85]. O sistema se divide em três componentes principais:

Low Level Utilities (LLU). Biblioteca contendo aproximadamente 500 funções gráficas de baixo nível, para C e FORTRAN, incluindo a implementação de GKS;

High Level Utilities (HLU). Biblioteca de funções de alto nível, orientada a objetos, que usa as funções da LLU mas esconde grande parte da complexidade dos dados e facilita a construção de aplicações interativas;

NCAR Command Language (NCL). Linguagem de scripts que pode ser usada tanto em modo interativo quanto *batch*.

O componente LLU é de livre distribuição, sob os termos da licença GPL [STA2000]. HLU e NCL são disponíveis sob uma licença que provê acesso ao código-fonte e permite ao licenciado utilizá-los em quantas estações desejar, inclusive fazendo modificações e acréscimos. Não é permitido, porém, redistribuir ou revender programas que contenham a biblioteca HLU ou o interpretador NCL, modificados ou não.

Dados podem ser lidos ou gravados nos formatos NetCDF, HDF, GRIB, ASCII e nos padrões binários de ponto flutuante IEEE e Cray. Não há suporte para imagens de satélite nem dados de radar. O formato gráfico padrão é NCGM, uma extensão do CGM [ANS86]. Existem comandos em NCL e funções na HLU para tratamento transparente de dados meteorológicos em todos esses formatos de armazenamento.

A HLU possui vários formatos para plotagem de dados, mas não suporta objetos tridimensionais nem *rendering* volumétrico. Todavia, pode-se usar as funções da LLU para qualquer tipo de desenho. Os resultados das visualizações podem ser salvos nos formatos NCGM e todas as variantes de PostScript, com controle total de tamanho e posicionamento na página. Para outros formatos, como GIF e JPEG é preciso usar um utilitário externo ou escrever funções de conversão que operem através da API do NCAR Graphics.

NCAR Graphics não tem interface gráfica com o usuário. O interpretador da linguagem de scripts (NCL) pode ser usado em modo interativo, mas a operação é via linha de

comando, como ocorre no GrADS. A sintaxe de NCL é semelhante à do C. Uma facilidade encontrada em NCL é a criação de *blocos de visualização*, seqüências de comandos que podem ser salvas como uma espécie de macro e executadas invocando seu nome na linha de comando (conceitualmente semelhantes às *caixas* de Tioga [AIK96], Data Explorer [IBM99] e Khoros [KHO99], mas sem a interface visual). O sistema conta com uma coleção de utilitários para filtragem de arquivos, edição interativa, visualização e animação de NCGMs, conversão de formatos e auxílio ao usuário.

Comparado aos demais pacotes, NCAR Graphics pode ser considerado o de mais baixo nível. A interface de programação é seu ponto forte. Tanto as bibliotecas HLU e LLU quanto a linguagem NCL possuem interfaces para FORTRAN e C. NCL tem recursos para processamento de matrizes e vetores semelhantes às do FORTRAN 90. Subrotinas externas escritas em C ou FORTRAN podem ser incorporadas a NCL, sendo que, para FORTRAN, o processo é parcialmente automatizado.

3.1.4 Vis5D

Vis5D [HIB90] é um software para visualização que adota um modelo de dados em 5 dimensões: linhas (latitude), colunas (longitude), níveis de pressão (altitude), tempo e número da variável física (temperatura, pressão, etc.). O sistema consiste de um programa de visualização, que usa um formato próprio de arquivos de dados, e diversos utilitários para importação, exportação e edição de arquivos ".v5d". Há também uma biblioteca de funções, para C e FORTRAN, para que o usuário escreva o seu próprio conversor. O software distribuído livremente, sob licença GPL, e roda nas plataformas UNIX, Windows NT e OS/2.

Vis5D aproveita os recursos do OpenGL para gerar imagens de grande apelo visual, incluindo além de mapas com divisões políticas e costeiras também a topografia do terreno. Os modos de visualização incluem isosuperfícies, fatias (de linhas de contorno, de cores, de vetores de vento e de linhas de corrente), *rendering* volumétrico (para representar uma das variáveis usando transparência) e linhas de corrente (trajetórias de vento). Legendas e títulos podem ser incluídos como rótulos de texto. Todos esses modos de operação podem ser combinados com animação.

Apesar de ter excelente exibição, Vis5D tem recursos pobres para exportação de gráficos. Os únicos formatos suportados são XWD (X Window Dump) e PPM (Portable PixMap). Formatos GIF, SGI RGB e PostScript só podem ser gerados por conversão.

A maioria dos utilitários para conversão e edição de arquivos tem interface de linha de comando. O programa de visualização possui uma interface gráfica, dividida em duas janelas: o *painel de controle* (dos parâmetros da visualização) e uma janela de visualização com imagens 3D, que pode ser dividida em duas ou mais subjanelas.

Existe uma API para programas em C que permite aos desenvolvedores incluir Vis5D como um subsistema de visualização de outros sistemas. Os programas podem então invocar o núcleo do Vis5D através das funções integrantes da API. A linguagem de scripts adotada é Tcl [OUS94] (pode-se compilar o programa sem ligá-lo à biblioteca *run-time* do Tcl, mas neste caso a linguagem de scripts se resume aos comandos e funções do Vis5D).

Tcl amplia as possibilidades de aplicação do programa. Pode-se usar o suporte da linguagem para gráficos, comunicação via rede e ligação a módulos externos escritos em C. Infelizmente o toolkit que acompanha a linguagem (Tk) não é usado e sim uma biblioteca de *widgets* muito primitiva chamada "LUI".

Devido à disponibilidade do código-fonte, Vis5D foi usado como base para diversos

outros sistemas [HIB98b]. Isto originou versões do programa incompatíveis entre si devido á fragmentação do desenvolvimento. Muitos recursos novos criados em sistemas derivados não foram incorporadas ao programa original. Steven Johnson deu início a um projeto destinado a melhorar essa situação [JOH2000].

3.1.5 VisAD

Apesar de não ter sido objeto de estudo no trabalho anterior nem neste, vale a pena mencionar VisAD, originário do mesmo grupo de pesquisa que criou o Vis5D. VisAD é um *famework* destinado à construção de aplicações de visualização em geral. De início, era uma biblioteca de funções para ser usada com linguagens C e FORTRAN [HIB92]. Posteriormente, foi reescrito como uma biblioteca de classes para a linguagem de programação Java [HIB98a].

3.2 Sistemas integrados

Na bibliografia pesquisada, muitos artigos relatam a criação de soluções *ad hoc*, baseadas em scripts para os programas de visualização mencionados na seção anterior [ALM98a, ALM98b, COR98, PES98]. Foram encontrados apenas dois ambientes de trabalho completos, com interface visual, ambos ainda em desenvolvimento: metAP e AFPS. Os dois adotam métodos semelhantes para seleção de dados e estabelecimento dos parâmetros de visualização, mas apenas o AFPS permite compor o boletim usando a interface gráfica, ainda que o metAP também ofereça um ambiente de trabalho totalmente voltado à atividade de previsão.

3.2.1 metAP

Pauli, Matter e Ambrosetti [PAU95] descrevem o Swiss Meteorological Workstation Project (meteorologischer Arbeitsplatz — metAP) desenvolvido para ser usado pelo Instituto Meteorológico da Suíça, que integra cinco centros de pesquisa responsáveis por tarefas de previsão numérica do tempo, previsão de avalanches e disseminação de resultados.

O armazenamento de dados no metAP fica a cargo de um banco de dados NEONS (Naval Environmental Operational Nowcasting System), um sistema de armazenamento de informações ambientais desenvolvido pela Marinha dos Estados Unidos e usado em instituições de pesquisa dos Estados Unidos, Canadá, Austrália e diversos países europeus (o CPTEC/INPE também o está adaptando às suas necessidades). NEONS provê acesso aos dados em um nível mais abstrato que o de um banco de dados comum e oferece as rotinas de acesso para o software de visualização.

Há cinco tipos de visualizadores no metAP, destinados a gráficos 3D (*voxels*), 2D, símbolos e texto. Os autores identificam quatro grandes classes de dados: *pontuais*, *imagens*, *grades* e *volumes*, cujo caminho desde a aquisição na natureza até a visualização é mostrado na figura 3.1.

O projeto do metAP leva em consideração requisitos ergonômicos tanto do software quanto do próprio local de trabalho de modo que a interação mais eficiente entre o software e o usuário contribua para um melhor desempenho das tarefas dos previsores. A interface se divide em três partes principais ou “passos de trabalho”:

1. seleção principal da área de interesse temático;

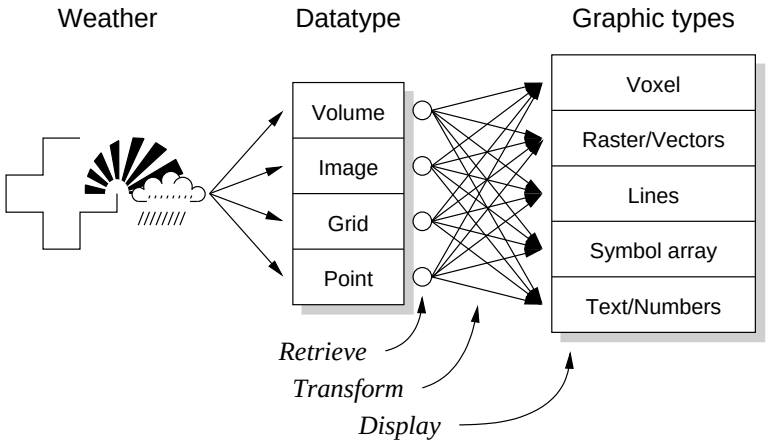


FIGURA 3.1 – Tipos de dados no banco de dados meteorológico e sua transformação em tipos gráficos no software metAP, segundo Pauli *et al*. [PAU95]. Para cada tipo de dado existe um processo de recuperação, representado pelos círculos, e um conjunto de processos de transformação, representados pelas setas, que convertem dados em tipos gráficos.

2. seleção e apresentação de parâmetros da visualização;

3. visualização e edição do conjunto de dados ou produto selecionado.

Cada passo tem uma janela associada, como mostra a figura 3.2. Há apenas uma seleção principal e uma janela de parâmetros associados à janela de visualização selecionada no momento, entre as muitas janelas de visualização que podem existir simultaneamente. Segundo Pauli, a restrição de apenas uma janela de parâmetros é para preservar a clareza, e "não transformar amontoados de papel nas mesas em amontoados de janelas na tela do computador".

A seleção principal (figura 3.2a) é composta por duas barras de botões e uma caixa de texto para digitação de comandos. A primeira é a barra de funções, que ativa operações como "Fim", "Trocar Usuário", "Informação de Status", "Alarmes" (um botão cuja imagem muda com a chegada de um alerta) e "Ajuda". A segunda barra, de aplicações, ativa os módulos de tratamento de dados, que podem ser de modelos numéricos, imagens de satélite, radiosondas, relatórios (textos), radar, dados sinóticos e meteogramas, entre outros. Também pode haver botões para agendamento de operações, correio eletrônico, um editor para produzir previsões e execução de *macros*. Finalmente, na caixa de texto, o operador pode digitar linhas de comando, conforme descrito mais adiante.

A janela de parâmetros (figura 3.2b) é uma caixa de diálogo cujo conteúdo depende da área de interesse escolhida na seleção principal. Esta caixa está dividida em quatro seções, que determinam o *ambiente dimensional* da visualização:

O quê: parâmetro físico que se quer observar (temperatura, pressão, visibilidade, etc.).

Onde: áreas de onde os dados devem provir ou onde se quer vê-los (global, continente, região, etc).

Quando: Data/hora dos dados (fixas ou inicial, final e intervalo para animações). Também pode incluir combinações de níveis verticais (altitudes padrão) com intervalos de tempo.

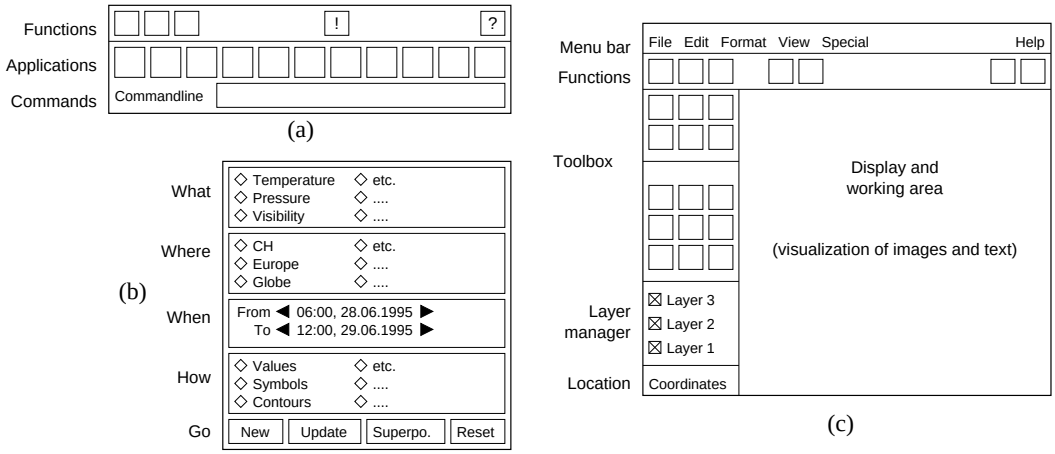


FIGURA 3.2 – Visão esquemática da interface janelas do metAP (baseado em [PAU95]). Em (a) a janela principal, com as barras de funções e de aplicações e a linha de comando. Em (b) a janela de parâmetros. Em (c) a janela de visualização contendo as barras superior (*menu*) e de funções; a parte inferior contém, à esquerda, as caixas de ferramentas, o gerenciador de camadas e o indicador da localização, além da área de trabalho, à direita.

Como: formato de exibição (valores, símbolos, linhas de contorno, contornos preenchidos, etc.).

Além dessas quatro seções há mais uma área contendo botões de comando tais como "New" e "Reset".

O terceiro passo consiste da janela de visualização (figura 3.2c). Ela segue um modelo bastante adotado em aplicações de desenho e editoração, com uma barra de *menu* horizontal a partir da qual são ativados menus do tipo suspenso (*pull-down*). Abaixo da barra de menu há uma barra de botões de função. Há uma caixa com ferramentas para manipulação da vista e abaixo dela uma para gerência das camadas (campos) que permite ativar, desativar ou trocar suas ordens.

Além da operação em modo interativo o sistema possui um mecanismo para execução de *macros*, com pré-cálculo das visualizações via módulo de agendamento, e integrado à interface com o usuário. Uma visualização (consulta ao banco de dados mais opções gráficas) pode ser gravada como uma macro. Na janela principal os botões de aplicações também atuam como menus suspensos, que podem ser configurados para conter as macros. Macros também podem ser ativadas pela linha de comando na janela principal. As visualizações podem ser pré-calculadas na própria estação de trabalho do previsor ou em servidores dedicados.

O desenvolvimento do metAP foi dividido em duas partes: a primeira foi a construção de um *framework* com as funcionalidades básicas, tais como auxílio ao usuário, agendamento, etc.; sobre ele, vão sendo desenvolvidos os módulos para recuperação, transformação, visualização e edição de dados.

3.2.2 AFPS

O National Weather Service (EUA) desenvolve um projeto para instalar em seus escritórios de previsão do tempo o "Advanced Weather Interactive Processing System" (AWIPS), composto por estações de trabalho que permitirão aos previsores examinar múltiplos tipos de dados e criar produtos para disseminação aos usuários [MAT96].

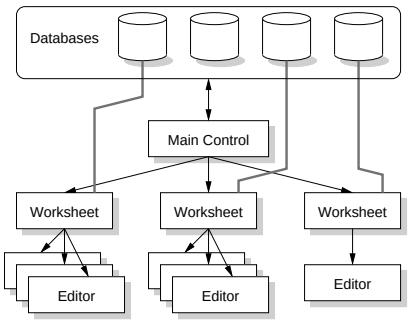


FIGURA 3.3 – Componentes do AFPS [MAT96]

O ambiente de trabalho do previsor no AWIPS é o "AWIPS Forecast Preparation System" (AFPS), cujo desenvolvimento iniciou em 1990. Os testes com o primeiro protótipo iniciaram-se em 1996 e a partir de 1999 ele se tornou operacional [NOA99]. O método de preparação das previsões centra-se em um banco de grades de dados para previsão. Os previsores podem visualizar e editar esses dados intuitivamente e desenhar sobre eles o prognóstico. São providos algoritmos para inicializar os bancos com dados de modelos, observações e, opcionalmente, fontes locais. Os benefícios decorrentes desse método são:

- como todos os produtos são derivados de um único banco de dados, há consistência entre as previsões;
- é mais fácil verificar e monitorar as previsões, porque são constituídas de grades de campos e não de textos em formato livre;
- as grades de previsão podem ser compartilhadas para conciliar as diferenças nas fronteiras entre os escritórios vizinhos e também para operações de backup.

O sistema é composto por três módulos (figura 3.3). O módulo *Main Control* provê um inventário dos bancos de dados disponíveis. Há um banco de dados de previsão "oficial" com base no qual todas as previsões devem ser feitas. O módulo *Worksheet* apresenta ao usuário um banco de dados como uma planilha na qual são selecionados os dados que serão vistos. O módulo *Editor* é a ferramenta com a qual o previsor desenha o prognóstico sobre os campos vindos do banco de dados e selecionados na planilha. Há dois tipos de edição: espacial e temporal. Na primeira os campos são projetados em uma superfície plana; na segunda apresenta-se o gráfico de uma série temporal de valores em um único ponto da grade.

Como plataforma operacional para o AFPS, foi projetada a estação de trabalho Advanced [MAC96]. Esta estação conta com software e hardware para aquisição de dados e opera como um sistema autônomo, capaz de receber dados nacionais e locais de diversas fontes, em vários formatos, incluindo observações, dados de radar e imagens de satélite [EDW96]. Também conta com software para decodificação de dados e armazenamento nos formatos NetCDF, binário e texto. A estação roda sistema operacional UNIX e X Window System e pode ser configurada com um ou mais *displays*.

A filosofia de projeto da interface com o usuário da WFO-Advanced foi estabelecida com base em estudos e protótipos, de modo que ela não seja apenas intuitiva e fácil de usar, mas que realmente simplifique o trabalho do usuário [GRO97]. A linguagem Tcl e

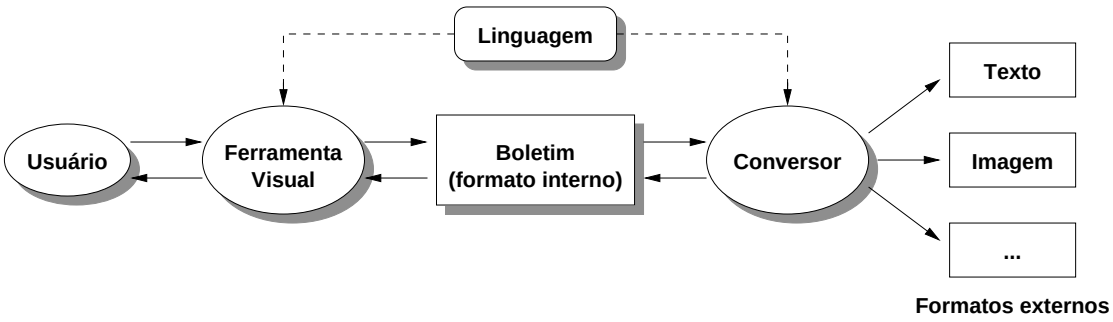


FIGURA 3.4 – Arquitetura da alternativa proposta.

o *toolkit* Tk [OUS94] foram usados tanto para prototipação como para a construção de vários componentes da interface.

A aplicação com a qual o previsor trabalha é o Gridded Forecast Editor (GFE) [NOA99], cujo objetivo é fornecer aos previsores uma ferramenta através da qual eles desenhem o prognóstico sobre uma grade de dados, ao invés de digitar o texto de uma previsão. Feito isto, são gerados automaticamente produtos textuais, gráficos e tabulares. Deste modo, ao invés de escrever "a temperatura prevista para a região é de 20°C", por exemplo, o meteorologista pinta a área correspondente com uma cor que corresponde a 20°C, usando uma ferramenta "aerógrafo". À medida que vai pintando, os elementos da grade de valores de temperatura vão sendo preenchidos com o valor 20.

3.3 Abordagem proposta

Um ambiente de trabalho para elaboração da previsão é um caso particular de sistema de programação para usuários finais, no qual o "programa" é o boletim do tempo. A filosofia adotada no AFPS, de permitir que o previsor pinte o prognóstico sobre os dados visualizados, mostra-se bastante adequada para um sistema desse tipo.

Mesmo existindo uma forma gráfica do boletim, a forma de texto, na qual ele é normalmente publicado, não deve ser suprimida. Entretanto, como o tempo de que o previsor dispõe para elaborar o boletim é pequeno, ambas devem ser produzidas simultaneamente.

Em teoria, o previsor poderia pintar o prognóstico sobre o mapa, com uma ferramenta de desenho à mão livre, para depois programa analisar o desenho e a partir dele gerar o texto correspondente. É mais prático, porém, gerar ambas as formas a partir de uma mesma representação interna estruturada.

O segue o paradigma de ferramenta da desenho/pintura (o GFE). A estruturação foi obtida dividindo a área da previsão em uma grade cujas células são "pintadas" de acordo com os valores dos fenômenos previstos. Essa abordagem não é adequada a um boletim como o do CPMet, no qual há uma regionalização do prognóstico. Além disso, o produto gerado com o GFE assemelha-se muito a uma carta sinótica, de fácil compreensão pelo meteorologista mas difícil de entender pelo leigo, pelos motivos discutidos na seção 2.2.2.

A arquitetura proposta possui se divide em três componentes (figura 3.4):

1. Uma linguagem que define, de forma não ambígua, as frases que o meteorologista pode usar em um boletim de previsão do tempo. Cada componente das frases usadas no boletim tem uma representação gráfica correspondente nessa linguagem.

2. Uma ferramenta visual, na qual o boletim é composto acrescentando sobre um mapa instâncias das representações dos elementos e editando seus atributos. Esta ferramenta permite que o previsor "desenhe" o prognóstico sobre os campos visualizados, ao invés de tentar descrevê-lo textualmente.

3. Um conversor, capaz de gerar diferentes formatos de representação externa do prognóstico, a partir da representação interna do boletim.

Adotando a arquitetura MVC (seção 2.3.4), temos, num primeiro momento, o boletim desempenhando o papel de modelo e a interface visual os papéis de vista e controlador. Esta subdivisão é discutida mais detalhadamente no capítulo 5. Antes disto, no capítulo 4, tratar-se-á da especificação formal da linguagem na qual o conteúdo do boletim deve ser escrito.

Embora o ideal fosse produzir uma solução genérica, as informações colhidas para a realização do trabalho anterior demonstram que há muitas diferenças entre as necessidades de cada instituição, bem como entre as rotinas de trabalho adotadas por elas. Decidiu-se, então, limitar o escopo deste trabalho às necessidades do Centro de Pesquisas Meteorológicas da UFPEL, cujas atividades foram descritas na seção 2.1.3.

4 A linguagem Pythonissa

Em Português, “pitonisa” significa “Sacerdotisa que lia o oráculo do templo de Apolo em Delfos” e “Adivinha, profetisa” [LAV92]. Pitonisa vem do latim *Pytonissa*, derivado do grego *Pyhōn*, “espírito da adivinhação”, a partir de *Pthō*, “assento no oráculo de Delfos”. Trata-se, portanto, de um nome adequado para uma linguagem destinada à escrita de previsões, ainda que resultantes da aplicação de métodos científicos e não da inspiração dos deuses do Olimpo. O acréscimo ds letra “h”, gerando o nome, *Pythonissa* é uma alusão a “Python”, o nome da linguagem de programação usada na implementação.

4.1 Necessidade de uma especificação formal

Existem três abordagens distintas para especificação de software: a *informal*, que não força o uso de uma linguagem formal para a especificação, a *semiformal*, que força o uso de uma linguagem formal no nível sintático, e a *formal*, que usa uma linguagem formal tanto para a especificação da sintaxe quanto da semântica [NUN97]. Uma especificação informal pode ser feita usando uma linguagem natural, como o Português, seguindo critérios adotados pelo seu autor. Uma especificação formal, por outro lado, utiliza uma linguagem própria para essa finalidade, de modo que o resultado pode ser mais compacto, completo e livre de ambigüidades, possibilitando verificar formalmente que a implementação corresponde à especificação.

A especificação formal no nível sintático permite que a correção da sintaxe de um programa seja verificada na fase de compilação (ou durante a execução, tratando-se de uma linguagem interpretada). A correção da lógica, por sua vez, pode ser verificada executando o programa com dados de entrada conhecidos e comparando os resultados obtidos com os esperados. Entretanto, uma vez qua a semântica de cada instrução seja definida formalmente, é possível determinar se um programa atende ou não à especificação prevendo o resultado de sua execução a partir dos dados de entrada.

Para linguagens de programação convencionais, existem técnicas de especificação consagradas, aplicadas à construção de compiladores e interpretadores. Essas técnicas baseiam-se em representações textuais das instruções que devem ser executadas pelo computador. Mesmo linguagens expressam a concorrência (paralelismo) normalmente usam representações essencialmente textuais. Linguagens visuais, por outro lado, são difíceis de especificar formalmente porque lidam com relações espaço-temporais (localização, forma, e movimentos) e sensoriais (diferenças de cor, brilho, etc.) dos elementos de uma imagem. Rekers e Schürr [REK96] observam que embora exista uma grande variedade de linguagens visuais, apenas umas poucas são dotadas de uma especificação formal da sintaxe.

Quando a linguagem visual é implementada por meio de uma aplicação com interface gráfica interativa, torna-se difícil fazer verificações de sintaxe e semântica. Nesse tipo de implementação, os comandos são ações realizadas sobre imagens, usando mecanismos como clicar e arrastar-e-soltar (*drag-and-drop*). O modelo de programação adotado comumente é o de aplicação guiada por eventos, que constituem a entrada, e provocam a execução de comandos ao ativar os elementos de controle da interface (*widgets*). É pra-

ticamente impossível verificar a correção dessa entrada em ambientes de janelas como o X Window System, nos quais os eventos são gerados de maneira assíncrona. Pode-se simular a entrada enviando à aplicação uma seqüência de eventos sintéticos, mas não há como saber quais são todas as seqüências possíveis.

No caso do sistema proposto — de composição da previsão do tempo — seria preciso conferir tanto se os elementos desenhados pelo previsor representam fenômenos meteorológicos válidos quanto se eles estão presentes nos locais e horários corretos. Fazer isto a partir da entrada limitaria demais a liberdade do usuário. É preferível deixá-lo compor livremente o prognóstico para depois verificar sua correção. Por este motivo optou-se por concentrar o trabalho na especificação de um modelo para o boletim, não para a interface da aplicação por meio da qual ele é composto.

Tomada a decisão de prover uma especificação formal para o boletim, o passo seguinte foi escolher o formalismo a usar. Optou-se por grafos porque modelos baseados em transformações de grafos têm sido usados com sucesso para representação de programas e em ferramentas de programação visual [AND99]. Como vantagens da Gramática de Grafos para a especificação de software destacam-se o alto nível da descrição e a possibilidade de representar visualmente tanto a sintaxe quanto a semântica [NUN97].

4.2 Conceitos de Gramática de Grafos

Há um grande número de artigos publicados na literatura sobre Gramática de Grafos. Alguns tratam o tema de modo abrangente, outros relatam o uso para especificação de software em geral e linguagens gráficas e visuais em particular [REK94, BAR97]. O objetivo do resumo a seguir é homogeneizar a terminologia e a notação usadas no restante do capítulo, visto que aquelas usadas pelos diferentes autores nem sempre são as mesmas. Tomou-se como base três artigos [EHR79, AND99, REK96]. Exceto por algumas diferenças tipográficas e comentários adicionais, é quase uma transcrição, sem pretensão alguma de originalidade.

4.2.1 Grafo, produção e morfismo

Definição 4.1 (Grafo). $G := (V, E, \mathrm{l}_V, \mathrm{l}_E, \mathrm{s}, \mathrm{t})$ é um grafo sobre dois conjuntos de rótulos L_V, L_E, sendo que:

- $V(G) := V$ e $E(G) := E$ são conjuntos finitos de vértices e arestas;
- $\mathrm{l}_V(G) : V \to L_V$ e $\mathrm{l}_E(G) : E \to L_E$ são funções de rotulação, que associam um identificador (rótulo) a cada vértice e aresta de G;
- $\mathrm{s}(G) : E \to V$ e $\mathrm{t}(G) : E \to V$ são funções que atribuem a cada aresta um vértice de origem e um de destino;
- $\mathrm{l}(G) := \mathrm{l}_V(V) \cup \mathrm{l}_E(E)$ será usada como abreviação para o conjunto de todos os rótulos de vértices e arestas em um grafo G.

Os sufixos V ou E das funções de rotulação podem ser omitidos quando estiverem claros, pelo contexto. Da mesma maneira, $x \in G$ é usado como uma abreviação de $x \in V(G) \vee x \in E(G)$. Os conjuntos imagem L_V e L_E são chamados de *alfabetos* de rótulos, de vértices e arestas respectivamente. □

Grafos são desenhados, em sua forma mais simples, com os vértices representados por pontos, círculos, etc., e as arestas representadas por arcos ou setas. O termo *arco* é usado como sinônimo de aresta. Os vértices e arestas de um grafo são também chamados de *ítens*.

Conforme mostrado adiante (seção 4.3.3), este modelo genérico precisa ser estendido para suportar tipos, atributos e *rótulos-curinga*, bem como as diversas operações que se farão com os grafos.

Definição 4.2 (Morfismo). Um par de funções $\mathrm{h} := (\mathrm{h}_V, \mathrm{h}_E)$ é um morfismo $\mathrm{h} : G \to G'$ de um grafo G para um grafo G' com $G := (V, E, \mathrm{l}_V, \mathrm{l}_E, \mathrm{s}, \mathrm{t})$ e $G' := (V', E', \mathrm{l}'_V, \mathrm{l}'_E, \mathrm{s}', \mathrm{t}')$ se

- $\mathrm{h}_V : V \to V'$ e $\mathrm{h}_E : E \to E'$ são mapeamentos totais;
- $\forall v \in V : \mathrm{l}'_V(\mathrm{h}_V(v)) = \mathrm{l}_V(v) \wedge \forall e \in E : \mathrm{l}'_E(\mathrm{h}_E(e)) = \mathrm{l}_E(e)$;
- $\forall e \in E : \mathrm{s}'(\mathrm{h}_E(e)) = \mathrm{h}_V(\mathrm{s}(e)) \wedge \forall e \in E : \mathrm{t}'(\mathrm{h}_E(e)) = \mathrm{h}_V(\mathrm{t}(e))$.

A forma abreviada $\mathrm{h}(x)$ pode ser usada ao invés de $\mathrm{h}_V(x)$ ou $\mathrm{h}_E(x)$ se o subscrito omitido puder ser depreendido a partir do contexto. □

Definição 4.3 (Subgrafo). Se um grafo $G' := (V', E', \mathrm{l}'_V, \mathrm{l}'_E, \mathrm{s}', \mathrm{t}')$ é a imagem de um grafo $G := (V, E, \mathrm{l}_V, \mathrm{l}_E, \mathrm{s}, \mathrm{t})$ em um morfismo $\mathrm{h} := (\mathrm{h}_V, \mathrm{h}_E)$, denotada $G' := \mathrm{h}(G)$, então G' é um subgrafo de G, denotado por $G' \subseteq G$. □

Esta é uma definição genérica de subgrafo, e bastante flexível, mas casos particulares podem impor restrições mais severas, como veremos a seguir.

Definição 4.4 (Produção). Uma produção genérica $p := (L, R)$ de uma gramática de grafos é uma *tupla* de grafos sobre os mesmos alfabetos de vértices e arestas. Os seus lados esquerdo e direito, $\mathrm{lhs}(p) := L$ e $\mathrm{rhs}(p) := R$ têm um subgrafo comum K, chamado *contexto*, que deve satisfazer plenamente às seguintes restrições:

- $\forall e \in E(K) \Rightarrow \mathrm{s}(e) \in V(K) \wedge \mathrm{t}(e) \in V(K)$
  com $E(K) := E(L) \cap E(R)$ e $V(K) := V(L) \cap V(R)$, isto é, origens e destinos das arestas comuns de L e R também são vértices comuns;
- $\forall x \in L \cap R \Rightarrow \mathrm{l}_L(x) = \mathrm{l}_R(x)$
  isto é, elementos comuns a L e R não diferem com respeito a seus rótulos em L e R.

Podemos definir também as formas abreviadas $\mathrm{Xlhs}(p)$ e $\mathrm{Xrhs}(p)$ para representar todos os elementos de $\mathrm{lhs}(p)$ e $\mathrm{rhs}(p)$, respectivamente, que não pertencem ao subgrafo comum K. Usando para isto a operação $\setminus$ (diferença), temos que:

- $\mathrm{Xlhs}(p) := L \setminus K$
- $\mathrm{Xrhs}(p) := R \setminus K$.

Observe-se que tanto $\mathrm{Xlhs}(p)$ quanto $\mathrm{Xrhs}(p)$ podem não resultar em grafos.

A forma genérica de aplicar uma produção $p := (L.R)$ a um grafo G consiste em encontrar uma ocorrência de L em G e removê-la, inserindo R em seu lugar, mas nem sempre se pode operar de maneira tão direta. A maior dificuldade para definir a *aplicação* de uma produção está em decidir quais as ocorrências (ver definição 4.5) de L em G são permitidas e quais não são, o que pode pode exigir um modelo mais complexo do que o apresentado. □

Definição 4.5 (Ocorrência). Um morfismo $\mathrm{h} := (\mathrm{h}_V, \mathrm{h}_E : H \rightarrow G)$ é uma ocorrência do grafo H no grafo G. Como conseqüência disto,
$\forall v \in V(H) : \mathrm{s}(\mathrm{h}_V(v)) = \mathrm{h}_E(\mathrm{s}(v)) \vee \mathrm{t}(\mathrm{h}_V(v)) = \mathrm{h}_V(\mathrm{t}(v))$.
Se h_V e h_E forem funções bijetoras, então H e G são grafos **isomorfos** e a ocorrência de H em G é dita uma **imagem isomorfa de H em G**. □

Ocorrências permitem identificar o casamento entre o lado esquerdo de uma produção e o grafo sobre o qual ela será aplicada. Mas a aplicação de uma produção exige que se estabeleçam algumas condições adicionais, caracterizando um tipo especial de ocorrência, chamado ***redex***.

Definição 4.6 (Redex). Um morfismo $\mathrm{h} := (\mathrm{h}_V, \mathrm{h}_E : L \rightarrow G)$ identifica um redex do grafo L no grafo G com respeito a outro grafo R se, e somente se, forem satisfeitas duas condições:

- Condição de soltura:
  $\forall v \in V(L) \setminus V(R), e \in E(G) :$
  $(\mathrm{s}(e) = \mathrm{h}_V(v) \vee \mathrm{t}(e) = \mathrm{h}_V(v)) \Rightarrow \exists e' \in E(L) \setminus E(R) : \mathrm{h}_E(e') = e$
  Esta condição impede que a aplicação de uma produção gere um grafo inválido, no qual uma ou mais arestas fiquem soltas (sem origem e/ou destino) porque um de seus vértices terminais, ou ambos, foi removido.

- Condição de identificação:
  $\forall x \in L \setminus R, x' \in L : \mathrm{h}(x) = \mathrm{h}(x') \rightarrow x = x'$
  Esta condição impede que dois elementos do lado esquerdo de uma produção, um dos quais será deletado e o outro não, sejam mapeados para o mesmo elemento do grafo a que ela é aplicada.

Um morfismo é chamado de **redex potencial** se a condição de identificação é plenamente satisfeita, mas talvez não a condição de soltura. □

Ehrig [EHR79] mostra que as condições de soltura e identificação, juntas, garantem que a aplicação de uma produção seja *reversível*, o que simplifica consideravelmente o desenvolvimento de algoritmos de análise sintática (*parsing*). Basta inverter os papéis dos lados esquerdo e direito, isto é, remover do grafo todos os elementos que casem com elementos de Xrhs e adicionar uma cópia de cada elemento de Xlhs.

A condição de soltura assegura que não temos que adicionar arestas que não sejam cópias de elementos de Xlhs. A condição de identificação assegura que temos que adicionar uma cópia separada de cada elemento de Xlhs. Isto é muito importante para os objetivos deste trabalho, uma vez que, durante a análise das diversas situações sinóticas componentes de um boletim, é necessário identificar que produções foram aplicadas na transição entre duas delas.

Definição 4.7 (Derivação). Uma derivação $p := (L, R)$ é uma tupla $pi := (p, \mathrm{h}, \mathrm{h}')$ tal que $\mathrm{h} : L \rightarrow G$ e $\mathrm{h}' : R \rightarrow G'$ definem a aplicação de p a um grafo G com resultado G', onde:

- h é um redex de L em G com respeito a R;
- h' é um redex de R em G' com respeito a L;
- $\mathrm{h}|_K = \mathrm{h}'|_K$, sendo K o contexto de L e R, e

- $G \setminus (\mathrm{h}(L \setminus R)) = G' \setminus (\mathrm{h}'(R \setminus L))$.

A aplicação da produção p ao grafo G com resultado G' é denotada por $G \stackrel{p}{\Longrightarrow} G'$. Uma derivação $(p, \mathrm{h}, \mathrm{h}')$ para a qual h e h' são redex potenciais é uma **derivação potencial de produção**. □

4.2.2 Gramática de grafos e suas linguagens

Definição 4.8 (Gramática de grafos). Uma gramática de grafos gg é uma tupla $(A, \mathcal{P})$, sendo A um grafo inicial não-vazio (o axioma) e $\mathcal{P}$ um conjunto de produções de gramáticas de grafos. □

Para simplificar as definições seguintes o grafo inicial A será tratado como um caso especial de produção cujo lado esquerdo é vazio λ. O conjunto de todas as derivações potenciais de produções de gg é abreviado por $\mathcal{PI}(gg)$.

Definição 4.9 (Linguagem de grafos). Sejam G e G' grafos. Sendo $gg := (A, \mathcal{P})$ uma gramática de grafos a sua linguagem, denotada $\mathcal{L}(gg)$ é definida como:
$G \in \mathcal{L}(gg) \Leftrightarrow A \Longrightarrow^* G$
com $G \Longrightarrow G' \Leftrightarrow \exists p \in \mathcal{P} : G \stackrel{p}{\Longrightarrow} G'$
e sendo $\Longrightarrow^*$ o fecho transitivo e reflexivo de $\Longrightarrow$.
Esta definição formal nos permite então verificar, por meio de um algoritmo de análise sintática, se um grafo pertence à linguagem, isto é, se ele pode ser obtido aplicando a A uma ou mais das produções pertencentes a $\mathcal{P}$. □

4.2.3 Gramática de grafos em camadas e suas linguagens

O conceito de divisão em camadas (*layering*) é apresentado por Rekers e Scürr em [REK95] e, de forma um pouco diferente em [REK96]. Ao invés da decomposição usual dos alfabetos de rótulos em duas camadas, terminais e não-terminais, faz-se uma decomposição de granularidade fina em um número de camadas.

Definição 4.10 (Conjunto de rótulos em camadas). A decomposição $L_V \oplus L_E = L_0 \oplus \ldots \oplus L_n$ do alfabeto de rótulos de vértices e arestas em n subconjuntos é um conjunto de rótulos em camadas. O operador $\oplus$ representa a união disjunta dos conjuntos, ou seja sendo A e B dois conjuntos, $A \oplus B = (A - A \cap B) \cup (B - A \cap B)$.

A função layer retorna, para qualquer elemento de um grafo G o índice da camada à qual pertence o rótulo:

$$\forall x \in G : \mathrm{layer}(x) = i \Leftrightarrow \mathrm{l}(x) \in L_i \,.$$

□

Definição 4.11 (Sublinguagem). Dada uma decomposição $L_0 \oplus \ldots \oplus L_n$ do alfabeto de rótulos L_V e L_E, a linguagem de uma gramática de grafos gg pode ser decomposta em um número de sublinguagens $L_0(gg), \ldots L_n(gg)$, tal que

$$L_i(gg) := \left\{ G \in L(gg) | l(G) \subseteq \bigcup_{j \leq i} L_j \right\} .$$

□

Definição 4.12 (Gramática de grafos em camadas). Uma gramática de grafos $gg := (A, \mathcal{P})$ é chamada de gramática de grafos em camadas com respeito a uma atribuição global de camadas $L_0 \ldots L_n$ a seus rótulos se $\forall p := (L, R) \in \mathcal{P}$: □

- R é um grafo conexo;
- o lado esquerdo L é não-vazio;
- o lado direito R sem os elementos de K é não-vazio;
- $L < R$ com respeito à seguinte ordem:

$$G < G' \Leftrightarrow \exists i : |G|_i < |G'|_i \wedge \forall j < i : |G|_i = |G'|_j$$

  com $|G|_k$ definido como $|\{x \in G | \mathrm{layer}(x) = k\}|$, isto é, o número de elementos de G que têm um rótulo de camada L_k.

As restrições existentes na definição anterior conferem propriedades desejáveis para o *parsing* da gramática, para o qual Rekers e Schürr criaram um algoritmo[REK95].

4.2.4 Variações e extensões do conceito de grafo

Andries *et al.* [AND99] enumeram diversas extensões do conceito genérico de grafo que são de interesse para os fins deste trabalho:

Tipos. Um grafo é dito *tipado* quando seus rótulos são divididos em *classes*, ou *tipos*, e arestas de um certo tipo só podem incidir sobre certos tipos de vértices de origem e destino. O conceito foi estendido neste trabalho para incluir *subclasses* de rótulos (seção 4.3.3).

Identificadores. Os vértices podem receber *identificadores*, que são úteis quando um grafo contém mais de um vértice do mesmo tipo, para permitir que sejam feitas referências individuais a cada um deles. Dentro das convenções adotadas neste trabalho um identificador é um *título* dado a um vértice, não devendo ser confundido com o rótulo, que identifica seu *tipo*.

Atributos. Vértices e arestas podem também possuir atributos. Um atributo pode ser um número, uma expressão, uma lista ou mesmo um grafo. Neste trabalho, os valores dos fenômenos que compões as condições sinóticas são atributos.

Grafos hierárquicos. Um grafo é hierárquico quando um vértice pode ser abstraido para um subgrafo e um conjunto de arestas ligando dois subgrafos pode ser abstraido para uma única aresta. Neste trabalho, o boletim de previsão do tempo é modelado como um grafo hierárquico.

Grafo não-dirigido. Na noção clássica de *grafo dirigido* não há arestas paralelas, mas segundo Andries [AND99], grafos *não-dirigidos* podem ser representados por grafos dirigidos substituindo cada aresta dirigida por um par de arestas em sentidos opostos. No modelo de boletim todas as relações são dirigidas, exceto a vizinhança entre regiões, mas optou-se por definir um tipo específico de aresta ao invés de usar arestas paralelas.

Andries define “grafo dirigido” como aquele em que “o conjunto de arestas é dado por uma relação binária sobre o conjunto de vértices”. Isto não corresponde à definição 4.1, que é muito mais genérica. O que importa aqui é o conceito de relação dirigida. Além dessas extensões, outras duas são dignas de menção, embora não sejam usados neste trabalho: *multigrafos*, que contém *arestas paralelas*, e *hipergrafos*, que contém arcos com mais de uma origem e/ou destino (*hiperarcos*).

4.3 Definição da linguagem

4.3.1 Terminologia

Define-se aqui os significados com os quais alguns termos são usados no texto a seguir, de acordo com o formalismo adotado. Estes significados não são necessariamente os mesmos que se empregam ao tratar de orientação a objetos (e que variam entre os autores, às vezes de forma conflitante).

Classe (Tipo). Categoria taxonômica a que pertence um ente, dentro de uma classificação pré-estabelecida, baseada em critérios morfológicos e/ou comportamentais. No que diz respeito a vértices e arestas, *classe* e *tipo* são usados com o mesmo significado neste texto. Para cada tipo, define-se uma marca (*rótulo*) usada para representar um objeto (em nosso caso particular, um vértice ou aresta de um grafo) de modo a caracterizá-lo dentre os demais.

Subclasse (Subtipo). Uma categoria taxonômica que é semelhante a uma outra, sua ancestral (superclasse), da qual se diferencia por pequenas alterações morfológicas e/ou comportamentais.

Objeto. Um ente concreto; uma instância (*strictu sensu*) de uma classe. Um objeto delimita um *escopo*, dentro do qual são visíveis os seus *atributos*.

Relação. Modo como dois objetos estão vinculados um ao outro.

Identificador. Nome que distingue, unicamente, um objeto entre os demais. O conjunto de identificadores usados em um sistema caracteriza um *espaço de nomes*. Existe obrigatoriamente uma relação biunívoca entre objetos e identificadores, mas não entre objetos e tipos, isto é, pode existir mais de um objeto com o mesmo tipo, mas não com o mesmo identificador.

Atributo (de um objeto). Um segundo objeto, que tem para com o primeiro uma relação de *pertinência*, estabelecida quando da definição da classe da qual o primeiro é uma instância. O acesso a um atributo só é possível dentro do escopo do objeto a que ele pertence.

Além de definir esses termos, tomou-se cuidado com o uso da palavra “tempo”, que possui tanto um sentido cronológico quanto um meteorológico. Para evitar confusões de interpretação, ao invés de “condições do tempo” e “situação do tempo”, usa-se sempre “condições sinóticas” e “situação sinótica”. Por motivos semelhantes, usa-se “presença” de um fenômeno, ao invés de “ocorrência”, evitando a confusão com a definição anterior (definição 4.5).

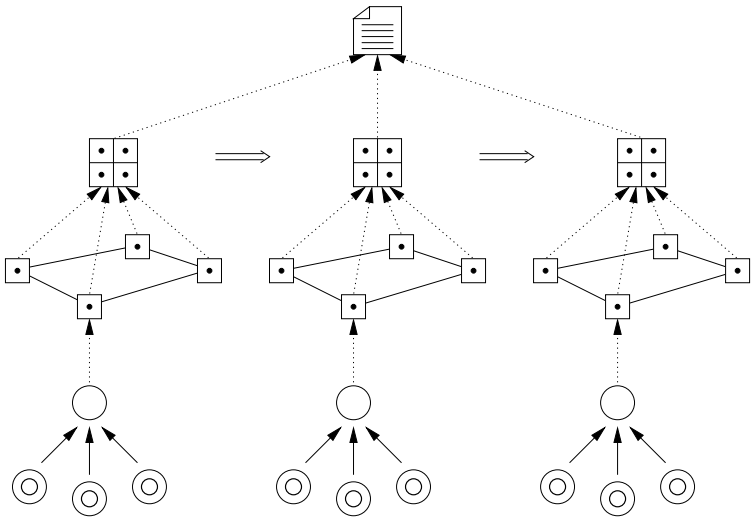


FIGURA 4.1 – Estrutura do boletim de previsão do tempo. Um boletim (🗐) é composto (⋯▸) por situações sinóticas (⊞) sucessivas, cada uma delas composta, por sua vez, por situações sinóticas regionais (⊡) formadas por regiões (◯) vizinhas (——) nas quais estão presentes (⟶) os fenômenos meteorológicos (◎).

4.3.2 Modelo do boletim

Para a modelagem no domínio *espacial* (territorial), dividimos a área de cobertura da previsão em *subdomínios*, as regiões, distribuindo sobre elas os elementos do texto. As regiões formam um grafo de vértices fixos, rotulados, ligados por arcos que denotam vizinhança. A cada vértice de região são conectados outros, de *fenômenos presentes*, caracterizando a situação sinótica. Assim, cada situação sinótica é composta por diversas situações regionais.

Para a modelagem no domínio *temporal*, é necessário dividir o intervalo abrangido pelo boletim em períodos, tais como manhã, tarde e noite, o que nos leva a ter diversas situações sinóticas consecutivas em um boletim.

No domínio *valor*, um fenômeno cuja intensidade seja nula (ausente) simplesmente não é incluído no modelo. Alguns fenômenos estão sempre presentes (temperatura, cobertura de nuvens e vento) enquanto os demais são eventuais. Quando um fenômeno está presente, os seus atributos, quando há, são representados pelo formato do ícone correspondente ao vértice.

O boletim é modelado como um grafo hierárquico com estrutura de árvore (figura 4.1). O boletim é sempre um grafo *conexo* (não há vértices isolados).

4.3.3 Tipos de vértices e seus atributos

Os tipos de vértices são simbolizados por ícones. Inicialmente se pretendia definir esses ícones a partir dos símbolos padrão da OMM, adotados na “Simbologia para boletim diário do tempo” do INMET [BRA8?a, p. 62]. Muitos deles, entretanto, tiveram de ser adaptados para ter formato mais atraente e tornar mais fácil a futura manipulação através de uma interface gráfica (os fenômenos foram descritos na seção 2.1.1).

Boletim (🗐). O boletim possui os seguintes atributos (ver seção 2.1.3):

- nome do meteorologista que fez a previsão;
- data e hora da emissão;
- período de tempo abrangido;
- análise sinótica;
- extremos observados;
- tendência para o período seguinte.

As situações sinóticas sucessivas que compõem o boletim não são tratadas como atributos, mas como vértices ligados ao boletim por arestas do tipo *Composição*.

Situação sinótica (⊞). A situação sinótica possui um único atributo: o intervalo de tempo, dentro do período coberto pelo boletim, a que ela corresponde (manhã, tarde ou noite, por exemplo). À semelhança do boletim, as regiões que compõem uma situação sinótica são vértices ligados a ela por arestas do tipo *Composição*.

Região (◯). Cada região recebe um nome que a distingue unicamente entre as demais. Os nomes das regiões são considerados identificadores dos vértices, não atributos.

Fenômeno (◎). Um dos fenômenos meteorológicos cuja manifestação caracteriza uma situação sinótica. Instâncias de Fenômeno jamais serão criadas, apenas das suas subclasses (Fenômeno é uma *classe abstrata*); entretanto este rótulo pode ser usado como *curinga* quando for necessário representar um fenômeno genérico. Os vértices do tipo Fenômeno são sempre *pendentes*, isto é, estão ligados ao restante do grafo por uma única aresta. De Fenômeno são derivadas nove subclasses, descritas a seguir.

4.3.4 Tipos de fenômenos

Temperatura (🌡). A temperatura do ar ambiente. Os *valores* previstos são o $_{mín}$ Mínimo e o ^{máx} Máximo. Caso se deseje apresentar um valor único, observado ou esperado, deve-se usar "$_{obs}$".

A *tendência* da temperatura pode ser de permanecer Estável, em Declínio ou em Elevação.

Esses atributos são representados todos juntos, em um mesmo ícone. Por exemplo, "$^{25}_{5}$" significa "temperatura estável, com mínima de 5°C e máxima de 25°C".

A presença da temperatura é obrigatória em todas as situações sinóticas.

Cobertura de Nuvens (☁). Conforme a porção da abóbada celeste que está coberta por nuvens, classifica-se o céu como Claro, Parcialmente nublado, Nublado ou Encoberto.

A cobertura de nuvens também é obrigatória em todas as situações sinóticas.

Vento (🌬). De acordo com a *velocidade*, o vento pode ser Fraco, Moderado ou Forte. A ausência de vento é comumente chamada de *calmaria*.

A *direção* de onde vem o vento é sempre um dos octantes Norte, Sul, Leste, Oeste, Nordeste, Noroeste, Sudeste Sudoeste. Os ícones desconsideram convenção de que no hemisfério sul as aletas (também chamadas "rebarbas") devem sempre ser desenhadas voltadas para cima.

Além da velocidade e direção, também pode-se informar se o vento é em Rajadas.

Assim como na temperatura, os atributos são representados todos juntos, em um mesmo ícone. Por exemplo, "" significa "ventos de sudoeste, fracos, em rajadas".

Chuva (). A *intensidade* da chuva pode ser Fraca (chuvisco), Moderada ou Forte. A chuva também pode ser em Pancadas.

A frase "Chuva de intensidade moderada, em pancadas" é representada por "".

Granizo (). Granizo não tem atributo e sua presença está sempre associada à presença de chuva.

Neve (). Assim como o nevoeiro, apenas a presença ou não de neve é assinalada no boletim do CPMet, por isso vértices do tipo Neve também não têm nenhum atributo.

Trovoadas (). Trovoadas não têm nenhum atributo. Sua presença está normalmente associada à presença de chuva.

Geada (). A *intensidade* da geada pode ser Fraca, Moderada ou Forte.

Nevoeiro (). Embora a simbologia padrão preveja a representação de névoa e nevoeiro em diversos tipos e intensidades, o boletim do CPMet assinala apenas a presença ou não de nevoeiro é assinalada. Por isto vértices do tipo Nevoeiro não têm nenhum atributo.

Além desses fenômenos, há outros que são comumente mencionados pelos meteorologistas dentro da análise sinótica: frentes, linhas e zonas. Tais fenômenos têm sua manifestação distribuída por uma área ampla, que abrange mais de uma região, e portanto exigem um outro tipo de tratamento em termos de modelagem, o que não foi feito neste trabalho.

4.3.5 Tipos de relações entre vértices

Os tipos de arestas estabelecem o caráter da relação entre os vértices, o que nos permite considerar o modelo do boletim como um diagrama de entidades e relacionamentos. Os tipos de arestas existentes são:

Composição (). Representa que os vértices origem e destino fazem parte de um objeto composto. A relação é dirigida e o vértice destino ocupa sempre uma posição mais elevada na hierarquia (mais próxima da raiz).

Vizinhança (). Representa a adjacência física entre duas regiões representadas pelos vértices interligados. A relação é não-dirigida, mas ao invés de representá-la por dois arcos paralelos (ver seção 4.2.4) deu-se preferência a uma forma simplificada.

Presença (). Significa que um determinado fenômeno certamente estará presente em uma região. Este prognóstico é feito quando o previsor tem dados suficientes para fazer tal afirmação. Caso o fenômeno não venha a se manifestar, pode-se considerar que houve um erro na previsão.

Possibilidade de presença (). Significa que existem condições para a presença de um fenômeno em uma região, mas que não é certo que ele se manifeste.

Embora possa parecer contraditório que se inclua esse tipo de prognóstico, ele muito útil para decidir sobre a realização ou não de muitas atividades que dependem da situação sinótica. Um exemplo é a aplicação de defensivos agrícolas, que nunca deve ser feita quando há possibilidade de chuva, pois caso venha mesmo a chover haverá graves prejuízos tanto financeiros, pela perda do produto aplicado, quanto ambientais, porque a chuva poderá levar o produto tóxico diretamente para os cursos d'água.

As relações de presença são dirigidas, uma vez que o fenômeno sempre se faz presente na região, e nunca o contrário. A frase "$\vdots \dashrightarrow \bigcirc_i$" significa "possibilidade de pancadas de chuva de intensidade moderada na região i". Para tornar mais compacta a representação, convencionamos que

$$\begin{aligned} \circledcirc \longrightarrow \bigcirc_i &= \circledcirc_i \\ \circledcirc \dashrightarrow \bigcirc_i &= \overline{\circledcirc}_i \end{aligned}$$

E que a não-presença e não-possibilidade de presença de um fenômeno na região i será representada por

$$\widetilde{\circledcirc}_i$$

sendo o índice i desnecessário quando não se tratar de uma região, ou quando esta for identificável pelo contexto.

4.3.6 Elementos sintáticos

Começamos por definir o alfabeto da linguagem, isto é, os conjuntos de vértices e arestas válidos, conforme a definição de grafo (Definição 4.1, na página 50).

Definição 4.13 (Alfabeto). Os conjuntos de rótulos válidos são:

- $L_V := \{$, , , , , , , , , , , $\}$ é o conjunto de rótulos de vértices;
- $L_F := \{$, , , , , , , , $\}$ é o subconjunto de rótulos de fenômenos;
- $L_{Fp} := \{$, , $\}$ é o subconjunto de rótulos de fenômenos de precipitação;
- $L_{Ff} := \{$, $\}$ é o subconjunto de rótulos de fenômenos de formação;
- $L_E := \{$, , , $\}$ é o conjunto rótulos de arestas;
- $L_{Efr} := \{$, $\}$ é o subconjunto de rótulos de arestas que relacionam fenômenos com regiões;
- $L := L_V \cup L_E$.∴
  $L := \{$, , , , , , , , , , , , , , , , , $\}$

□

Doravante, um rótulo qualquer, pertencente a L_F será representado por "$\circledcirc$" e um rótulo qualquer pertencente a L_{Efr} será representado por "$\longrightarrow$".

Apesar hever ícones definidos para todas as combinações de valores de atributos de fenômenos, os rótulos contidos em L_F restringem-se apenas ao *tipo* e portanto não consideram atributos. Nas definições a seguir, quando for necessário especificar um determinado valor de atributos de um fenômeno, será usado o ícone correspondente. Fica implícito, também, que qualquer grafo $G := (V, E, \mathrm{l}_V, \mathrm{l}_E, \mathrm{s}, \mathrm{t})$ satisfaz a duas condições:

$$\forall v \in V : \mathrm{l}(v) \in L_V$$
$$\forall e \in E : \mathrm{l}(e) \in L_E$$

Definição 4.14 (Intervalos de tempo). $T := \{t_1, \ldots t_n\}$ é uma seqüência de n instantes sucessivos no tempo e t um instante de tempo qualquer, tal que $t \in T$. □

Definição 4.15 (Identificadores de região). $I := \{i_1, \ldots i_m\}$ é o conjunto de m identificadores de região. Sendo V o conjunto de vértices de um grafo, e $\mathrm{fv}(I) : I \to V$ uma função que associa a cada identificador um vértice v, estabelece-se três restrições:

1. $i \in I \Rightarrow \mathrm{l}(\mathrm{fv}(i)) = \bigcirc$;
2. $\forall i \in I, i' \in I : \mathrm{fv}(i) = \mathrm{fv}(i') \Rightarrow i = i'$;
3. $\forall i \in I, i' \in I, i \neq i' : \mathrm{fv}(i) \neq \mathrm{fv}(i')$.

Ou seja, cada identificador dá o nome a uma e apenas uma região. A região que tem o identificador i é denotada como $\bigcirc_i$. □

A situação sinótica num instante é estabelecida pela presença ou não dos fenômenos em cada região e pelos atributos desses fenômenos. O grafo, portanto, é adequado para a modelagem de frases descrevendo entidades (vértices) e os relacionamentos estáticos (arestas) entre elas. Este formalismo permite também que o modelo tenha uma estrutura rígida e que dela se derive um *layout* físico, a partir do qual se pode construir uma imagem onde os relacionamentos entre as entidades sejam visíveis e, por meio de uma interface interativa, manipuláveis. Isto só vem confirmar o que a bibliografia consultada apresenta.

Um sistema de transformação de grafos poderia modelar as transições entre situações sinóticas consecutivas (mudanças no estado do modelo), se fosse possível estabelecer relacionamentos de causa e efeito entre cada situação e sua sucessora, isto é, uma *lógica temporal*, expressável como um conjunto de produções. Para a especificação da linguagem segundo a definição formal adotada (definição 4.9) seria preciso, portanto:

1. enumerar todos os elementos dos conjuntos de rótulos de vértices e arestas;
2. definir a situação sinótica inicial (grafo axiomático);
3. enumerar as produções aplicáveis, que constituiriam a gramática da linguagem.

Esta era, erroneamente, a intenção original quando se optou pela Gramática de Grafos: as transições entre as situações sinóticas (apenas as das situações regionais, uma vez que o modelo desconsidera interações entre regiões) seriam modeladas como transformações. A correção sintática do boletim seria garantida pela definição da situação sinótica, mas a correção semântica deveria ser garantida por um conjunto de produções aplicáveis e pelas regras de aplicação impostas. A prática, porém, demonstrou ser impossível, dentro dos limites do modelo adotado, estabelecer relações diretas entre as situações sinóticas consecutivas em cada região, pelas seguintes razões:

1. As relações entre os fenômenos e seus fatores causadores são muito complexas, conforme descrito na seção 2.1.1.

2. Nem todas as causas dos fenômenos fazem parte do modelo. Tais causas incluem frentes, linhas de instabilidades, massas de ar, ciclones e anticiclones, entre outros fatores, cuja presença não pode ser regionalizada.

3. Mesmo que todos os fatores causadores fizessem parte do modelo, verificar a correção de uma situação sinótica em função da situação anterior exigiria a construção de um tipo de sistema especialista, o que fugiria do escopo original do trabalho.

4. O modelo desconsidera a relação entre as situações sinóticas de regiões vizinhas. Nada impede que o previsor inclua no prognóstico situações sinóticas totalmente diferentes para regiões adjacentes, que seriam impossíveis de ocorrer na realidade.

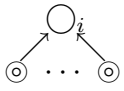
5. O intervalo de tempo entre a emissão do boletim e o início do período para o qual ele é válido é muito grande, principalmente quando se trata do boletim da tarde, que tem validade para o dia seguinte. No Rio Grande do Sul, assim como em todo o sul do continente, as situações sinóticas são muito influenciadas pela entrada de massas de ar originárias da região polar, que entram muito rapidamente no Estado.

Assim sendo, dentro das limitações práticas existentes, podemos usar o formalismo de grafos para definir completamente a sintaxe da linguagem, mas apenas parcialmente a semântica. Cabe ao previsor verificar a correção. Ainda assim, podem ser tratadas algumas restrições decorrentes de a presença de um fenômeno estar associada à presença e/ou ausência de outros.

Definição 4.16 (Situação sinótica regional). O grafo $\boxdot_{i,t} := (V, E, \mathrm{l}_V, \mathrm{l}_\mathrm{E}, \mathrm{s}, \mathrm{t})$ é a situação sinótica na região $\bigcirc_i$ no instante t, devendo ser satisfeitas as seguintes condições:

1. $\forall v \in V, v' \in V : \mathrm{l}(v) \neq \mathrm{l}(v')$;
2. $\forall e \in E : \mathrm{l}(e) \in L_{Efr} \Rightarrow \mathrm{l}(\mathrm{s}(e)) \in L_F \wedge \mathrm{l}(\mathrm{t}(e)) = \bigcirc$;
3. $\forall e \in E : \mathrm{l}(e) = \cdots\!\blacktriangleright \Rightarrow \mathrm{l}(\mathrm{s}(e)) = \bigcirc \wedge \mathrm{l}(\mathrm{t}(e)) = \boxdot$;
4. $\forall e \in E : \mathrm{l}(e) = \text{——} \Rightarrow \mathrm{l}(\mathrm{s}(e)) = \bigcirc \wedge \mathrm{l}(\mathrm{t}(e)) = \bigcirc$;

A condição nº 1 garante que só exista um vértice de cada tipo em uma situação sinótica regional. A condição nº 2 implica que os vértices de fenômenos sempre são terminais e que os de região nunca o são, uma vez que sempre há fenômenos presentes. Esse grafo pode ser desenhado como



mas para uma representação mais sucinta usamos um conjunto de tríadas ordenadas fenômeno-presença-região:

$$\boxdot_{i,t} := \{(\circledcirc, \longrightarrow, \bigcirc_i), \ldots\}_t$$

Como $\bigcirc_i$ é comum a todas as tríadas, podemos simplificar a representação para

$$\boxdot_{i,t} := \bigcirc_i, \{(\circledcirc, \longrightarrow), \ldots\}_t$$

e sabendo que i identifica unicamente uma região,

$$\boxdot_{i,t} := \{(\circledcirc, \longrightarrow), \ldots\}_{i,t}$$

TABELA 4.1 – Condições para a presença dos fenômenos.

Fenômeno presente	Fenômenos associados			
				Precipitação
Chuva (⁞⁞⁞)				
Granizo (◇)	> 20°C			⁞⁞⁞
Neve (✳)	< 3°C			
Trovoada (ϟ)				⁞⁞⁞[1]
Geada (⌊¯¯⌋)	< 3°C	[2]		⁞⁞⁞
	< 3°C	∨ [3]	∧	⁞⁞⁞
Nevoeiro (≡)		[2]		⁞⁞⁞
		∨ [3]	∧	⁞⁞⁞ ∧ ▒

[1]Desconsiderando trovoada seca [2]Geada de radiação [3]Geada de advecção

Restrições Conforme já foi discutido, existe um conjunto de restrições que se pode impor ao conteúdo do boletim. Essas restrições, apresentadas na tabela 4.1, são definidas formalmente a seguir:

1. $\forall v \in V : \mathrm{l}(v) \in L_{Fp} \Rightarrow \nexists v' \in V : \mathrm{l}(v') =$ ☀;
2. $\forall v \in V : \mathrm{l}(v) =$ ◇ $\Rightarrow \exists v', v'' \in V : \mathrm{l}(v') =$ 🌡$^{>20} \wedge \mathrm{l}(v'') =$ ⁞⁞⁞;
3. $\forall v \in V : \mathrm{l}(v) =$ ✳ $\Rightarrow \exists v' \in V : \mathrm{l}(v') =$ 🌡$^{<3}$;
4. $\forall v \in V : \mathrm{l}(v) =$ ϟ $\Rightarrow \exists v, v'' \in V : (\mathrm{l}(v') =$ ⛅ $\vee \mathrm{l}(v') =$ ☁$) \wedge \mathrm{l}(v'') =$ ⁞⁞⁞;
5. $\forall v \in V; \mathrm{l}(v) =$ ⌊¯¯⌋ $\Rightarrow$
$$\left\{ \begin{array}{c} \exists v' \in V : \mathrm{l}(v') = \text{🌡}^{<3} \\ \wedge \\ \left(\begin{array}{c} (\exists v'' \in V : \mathrm{l}(v'') = \text{☀} \wedge \nexists v''' \in V : \mathrm{l}(v''') = \text{↓}) \\ \vee \\ \left(\begin{array}{c} (\exists v'' \in V : \mathrm{l}(v'') = \text{☀} \vee \mathrm{l}(v'') = \text{⛅}) \wedge \\ (\nexists v''' \in V : \mathrm{l}(v''') = \text{🌡} \wedge \mathrm{l}(v''') = \text{🌡}) \end{array} \right) \end{array} \right) \\ \wedge \\ \nexists \mathrm{l}v'''' \in V : \mathrm{l}(v'''') = \text{⁞⁞⁞} \end{array} \right.$$
6. $\forall v \in V; \mathrm{l}(v) =$ ≡ $\Rightarrow$
$$\left\{ \begin{array}{c} \left(\begin{array}{l} \exists v' \in V : \mathrm{l}(v') = \text{☀} \wedge \\ \nexists v'' \in V : \mathrm{l}(v'') = \text{↓} \wedge \\ \nexists v''' \in V : v'''\mathrm{l}(v''') = \text{⁞⁞⁞} \end{array} \right) \\ \vee \\ \left(\begin{array}{l} (\exists v' \in V : \mathrm{l}(v') = \text{☀} \vee \mathrm{l}(v') = \text{⛅}) \wedge \\ (\nexists v'' \in V : \mathrm{l}(v'') = \text{🌡} \wedge \mathrm{l}(v'') = \text{🌡}) \wedge \\ (\nexists v''' \in V : \mathrm{l}(v''') = \text{⁞⁞⁞} \wedge \mathrm{l}(v''') = \text{▒}) \end{array} \right) \end{array} \right.$$

□

Doravante, o termo “situação regional” será usado como sinônimo de “situação sinótica regional”.

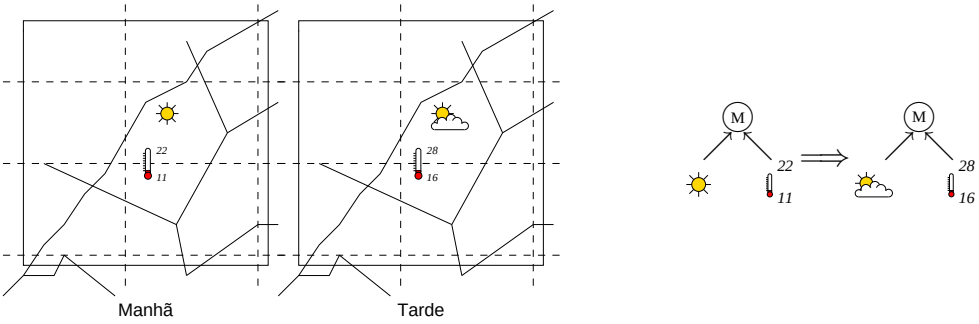


FIGURA 4.2 – Representação de duas situações sinóticas consecutivas para a região das Missões (à esquerda, como símbolos sobre o mapa de regiões; à direita, como transformação de grafos). As sentenças correspondentes seriam “Céu claro, passando a nublado” e “Temperatura em elevação”.

Exemplo 1. Considerando que as regiões para as quais se fará a previsão são aquelas constantes na figura 2.3 e que os períodos de tempo são manhã, tarde e noite, isto é,

$$I := \{\mathrm{DC}, \mathrm{LN}, \mathrm{LS}, \mathrm{M}, \mathrm{N}, \mathrm{O}, \mathrm{P}, \mathrm{S}, \mathrm{SO}\}$$
$$T := \{\mathrm{manhã}, \mathrm{tarde}, \mathrm{noite}\}$$

e a situação regional

> “Litoral sul: no período da manhã céu nublado; possibilidade de pancadas de chuva fraca; ventos do quadrante norte, fracos; temperatura estável.”

seria representada por

$$\{\text{⛅}, \text{🌧}, \text{🌡}, \text{🌡}\}_{\mathrm{LS,manhã}}$$

Exemplo 2. Sendo $\boxdot_{i,t_j}$ e $\boxdot_{i,t_{j+1}}$ as situações sinóticas na região i nos intervalos de tempo j e $j+1$, respectivamente, a transição entre estas situações pode ser simbolizada por $\boxdot_{i,t_j} \Longrightarrow \boxdot_{i,t_{j+1}}$. A figura 4.2 mostra como duas frases do texto do boletim podem ser representadas como transformações de grafos, expressando a evolução no domínio *temporal*.

Definição 4.17 (Situação sinótica). Uma situação sinótica, no tempo t, é um conjunto de m situações sinóticas regionais (das regiões $\bigcirc_i \ldots \bigcirc_m$), definido como

$$\boxplus_t := \{\boxdot_{1,t}, \ldots, \boxdot_{m,t}\}$$

que é representado, de forma mais simplificada, como

$$\boxplus_t := \{\boxdot_1, \ldots, \boxdot_m\}_t$$

□

Definição 4.18 (Boletim meteorológico). Um boletim meteorológico é uma sucessão de situações sinóticas em n diferentes instantes, definido como

$$\text{🗐} := \{\boxplus_{t_1}, \ldots, \boxplus_{t_n}\}$$

□

5 Implementação

5.1 Protótipo da interface com o usuário

Não existe um padrão para interfaces de aplicações voltadas à previsão do tempo. Como pôde ser visto no capítulo 3, cada sistema adota a interface que seus desenvolvedores consideram mais adequada. Dentre as diversas alternativas, a metáfora da aplicação de desenho/pintura, adotada no AFPS [NOA99], foi a escolhida. Ela tem a vantagem de já ser conhecida pelos usuários, por ser usada em muitas aplicações.

5.1.1 Diretrizes básicas

Algumas decisões importantes foram tomadas com respeito à interface com o usuário em função da experiência prática, no convívio diário com os previsores, e também pelo que se pode constatar a partir do estudo da bibliografia:

1. As representações usadas para as informações deveriam ser semelhantes àquelas já existentes no software com o qual os previsores estão habituados a trabalhar, de modo a facilitar a compreensão.

2. O boletim precisa ser representado como um cartograma. Não há outra alternativa, já que é preciso sobrepor a um mapa — e por conseqüência para localizar geograficamente — informações tanto numéricas quanto qualitativas.

3. Grandezas de caráter qualitativo, como *tempo bom* e *tempo instável*, seriam representadas como ícones.

4. Para facilitar a compreensão seria importante manter à mostra uma legenda com o significado dos símbolos e ícones usados.

5. Representação tridimensional não seria prioridade, uma vez que o boletim contém a previsão de acordo com regiões do território.

6. As modificações das grandezas representadas ao longo do tempo deveriam, preferencialmente, ser apresentadas na forma de uma animação, dada a sua natureza temporal, que torna inadequado o uso de gráficos estáticos.

5.1.2 Implementação

Conforme foi discutido na seção 2.3, o desenvolvimento da interface é um processo iterativo, envolvendo ciclos sucessivos de implementação e teste. Por isto, decidiu-se implementar inicialmente uma *maquete* da aplicação, por meio da qual se pudesse fazer experiências sobre a forma de construir a versão final. Deste processo, resultou um projeto de interface, descrito a seguir.

A maquete foi implementada em Python, usando o *toolkit* Tk. A linguagem já fora escolhida para a construção do sistema e a opção por Tk deveu-se à integração com a linguagem, por meio do módulo *TkInter*, e à da grande flexibilidade do *widget* "canvas" [OUS94, Cap.19], que facilitou muito a programação.

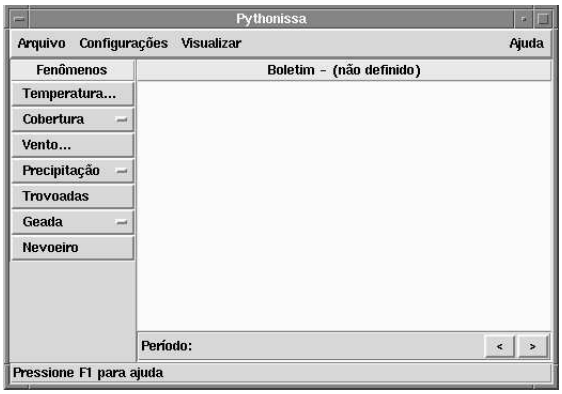


FIGURA 5.1 – Janela principal da interface

Visão geral

Ao dar início ao trabalho, o previsor tem à sua frente a janela principal do programa, mostrada na figura 5.1. Esta janela está dividida em quatro áreas:

- Na parte superior, ocupando toda a extensão da tela está a barra de *menu*, contendo as funções gerais do sistema, que são criar, salvar e carregar arquivos de prognósticos, configurar o sistema, escolher opções de visualização. A barra provê também um botão de acesso a uma função de auxílio ao usuário (atualmente este auxílio se resume a uma caixa de diálogo “Sobre Pythonissa”.

- O lado esquerdo da janela é ocupado por uma barra de botões na qual o usuário seleciona os fenômenos cuja presença ele deseja incluir no boletim.

- Ocupando a maior parte da janela, à direita da barra de fenômenos, está a área de trabalho, onde será desenhado o boletim. No seu topo há um título, indicando que tipo de boletim será desenhado, conforme descrito a seguir.

- A parte inferior da janela contém a barra de *status*, que informa a data inicial e o período para o qual o boletim será válido. À sua direita há os botões de avanço e retrocesso, que permitem passar aos períodos seguinte e anterior ao que está na tela, respectivamente.

Criação de um boletim

Para criar um novo boletim o previsor seleciona a opção “Arquivo/Novo boletim”, no cardápio principal, conforme mostra a figura 5.2. Ao se fazer isto, surge uma janela auxiliar na qual o previsor escolhe o tipo de boletim que deseja criar. Esse tipo é na verdade um conjunto pré-definido de parâmetros que incluem a área coberta pelo mapa, os nomes das regiões e as coordenadas dos pontos que definem seus limites, o valor do incremento de tempo entre cada situação sinótica consecutiva (em horas) e um nome para cada um desses intervalos (madrugada, manhã, tarde, e noite, por exemplo).

A janela de seleção contém, além da seleção do tipo de boletim, apenas um outro campo para que o previsor informe a data inicial a partir da qual o prognóstico será válido. Ao completar as informações, o usuário pressionará o botão “OK”, o que fará com que a

FIGURA 5.2 – Criação de um novo boletim de previsão do tempo

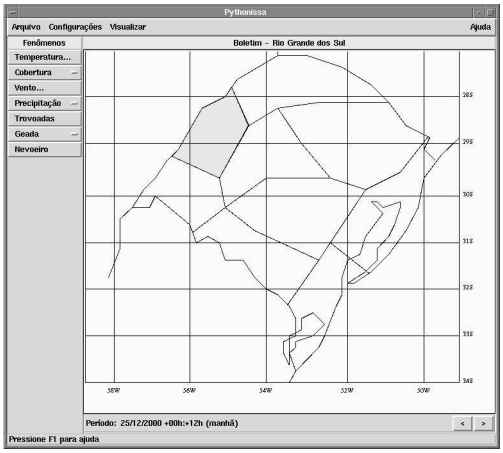


FIGURA 5.3 – Área de trabalho com o mapa das regiões

janela principal passe a ter a aparência mostrada na figura 5.3. Para editar o prognóstico de uma das regiões, basta “clicar” no seu interior, o que fará com que ela passe a ter uma cor destacada.

Inclusão de fenômenos

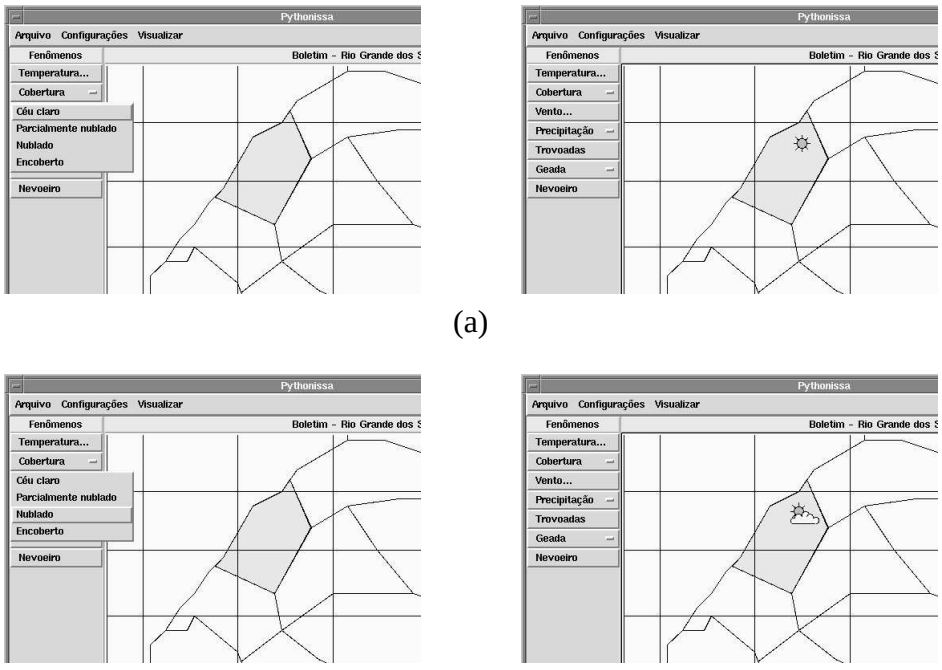
Tendo selecionado a região de interesse, basta selecionar na barra à esquerda os fenômenos a fazer presentes. A figura 5.4 ilustra a inclusão da informação sobre cobertura de nuvens: tendo a região selecionada, o operador ativa o botão desejado na barra de fenômenos e, a partir dele, escolhe a opção desejada. Feito isto, o ícone correspondente será desenhado sobre a região.

O tipo de botão usado para cada fenômeno depende dos atributos que este tem:

- trovoadas e nevoeiro são caracterizadas apenas por sua presença ou não, por isto são botões simples;

- para temperatura e vento, que têm diversos atributos, o pressionamento dos botões ativa caixas de diálogo nas quais os valores são preenchidos, o que é indicado pela existência de retucências no rótulo dos botões;

- para precipitação há um submenu onde o ítem “Chuva” ativa uma caixa de diálogo, a exemplo da temperatura, e os ítens “Granizo” e “Neve” são botões simples.



(a)

(b)

FIGURA 5.4 – Inclusão no boletim das frases “céu claro” (a) e “céu nublado” (b).

Para avançar no tempo, isto é, editar o período seguinte do boletim, o previsor pressiona o botão de avanço, na parte inferior da janela. Feito isto, pode-se incluir os fenômenos presentes neste período, do mesmo modo como foi descrito anteriormente.

No seu estágio atual, a maquete não trata a diferença entre presença e possibilidade de presença dos fenômenos.

Salvar e exportar

Para gravar o boletim em arquivo de disco o previsor seleciona a opção “Arquivo/Salvar” no cardápio de comandos da barra superior. O nome do arquivo é determinado automaticamente pelo programa, em função da data inicial.

Para *exportar* o boletim (convertê-lo para formatos externos), deve ser usado o comando “Arquivo/Salvar como...”. Surge então uma caixa de diálogo na qual o previsor pode selecionar o nome do arquivo no qual deve ser salvo o boletim e o formato. A maquete provê apenas os formatos de texto e EPS (*Encapsulated PostScript*). Este último é mérito exclusivo do criador do *widget* canvas, não do autor deste trabalho. O texto gerado, entretanto, é bastante simplificado. No caso dos exemplos anteriores, ele seria :

```
Previsão do tempo para Rio Grande do Sul
Válida para o dia 25 de dezembro de 2000
Região das Missões
     manhã: céu claro
     tarde: céu nublado
```

5.2 Razões para construir um *framework*

A maquete da interface, descrita na seção anterior, não tem utilidade em um ambiente "de produção", uma vez que, conforme descrito na seção 2.1.4, a maior dificuldade enfrentada pelos previsores é exatamente o fato de terem de usar diversos tipos diferentes de software para fazer seu trabalho. Como forma de verificar a aplicabilidade da linguagem em uma situação de trabalho real, decidiu-se tentar mesclá-la com uma aplicação de visualização.

5.2.1 Alternativas existentes

Considerando as estratégias adotadas pelos desenvolvedores do metAP e do AFPS (seção 3.2), concluiu-se que seria vantajoso dispor um *framework* para desenvolvimento de aplicações, que propiciasse extensibilidade, suporte à prototipação rápida e portabilidade. Isto não correspondia às características de nenhum software disponível.

Para a implementação desse *framework* havia duas alternativas: construí-lo a partir de um toolkit para visualização já existente, como Vtk [SCH96], ou acrescentar a um software de visualização de dados as funções necessárias para implementar da linguagem do boletim e a interface com o usuário. A segunda opção foi considerada a melhor. A alternativa de desenvolver o software "do zero" usando uma biblioteca gráfica de mais baixo nível, como OpenGL, não foi sequer considerada. Reimplementar os recursos de um software de visualização seria excessivamente trabalhoso e, como contribuição científica, de pouco valor.

Buscou-se, então, o software de visualização de código aberto que melhor atendesse às necessidades que se tinha, ponderando vários fatores, alguns de caráter tórico, outros de natureza prática:

1. NCAR Graphics oferecia uma biblioteca muito rica, mas apenas de funções de baixo nível. Não se tinha experiência com seu uso e temeu-se uma sobrecarga de programação, considerando o tempo disponível para a conclusão do trabalho. Além disto, havia o problema de a UFRGS não dispor de uma licença de uso, embora o CPMet a tivesse. Posteriormente o NCAR adotou uma licença mais liberal para o software, mas quando isto ocorreu o trabalho já estava em andamento.

2. Metview era um sistema de nível muito mais alto do que o necessário. Com base na bibliografia disponível, constatou-se que não seria possível apenas a utilização de macros. Ele também não estava disponível para uso, nem na UFRGS nem no CPMet. O pouco conhecimento sobre o sistema, restrito à consulta de bibliografia, também pesou contra sua adoção.

3. GrADS não oferecia uma API para programação de baixo nível e os recursos para construção de interfaces disponíveis em sua linguagem de scripts eram limitados demais. Essas limitações poderiam ter sido contornadas alterando o programa, mas quando se deu início à implementação o código-fonte do GrADS ainda não estava disponível publicamente (e mesmo depois disto as restrições quanto à redistribuição de versões modificadas do GrADS limitariam demais a utilidade do software criado).

4. A nova versão do framework VisAD, já estava em desenvolvimento pelo mesmo grupo de pesquisa de onde se originou Vis5D. Ela tinha como principal desvanta-

gem depender de uma extensão para a linguagem Java (Java3D), o que comprometia tanto o desempenho quanto a portabilidade.

5. Vis5D oferecia uma API para programação em C e Tcl e a licença (GPL) permitia a distribuição de versões modificadas. Já havia relatos de seu uso para outras finalidades [HIB98b]. Contra ele pesava a interface gráfica ser baseada em um *toolkit* muito limitado, mas este problema era contornável.

5.2.2 Estratégia adotada

Pesando as diversas alternativas existentes, optou-se por estender Vis5D. Desta forma, a implementação do protótipo ficou dividida em duas partes (a figura 5.5 mostra como essas partes se relacionam):

1. Um *framework* de visualização, que incorporasse o modelo de dados do Vis5D e seus recursos de exibição, mas no qual a interface com o usuário pudesse ser modificada dinamicamente, gerando diferentes tipos de aplicação.

2. Uma aplicação, implementada sobre o framework, que incorporasse a este o modelo de dados do boletim, uma interface para sua criação/edição e um conversor para formatos “externos”.

Para tornar o framework mais flexível, decidiu-se dotá-lo de uma linguagem de *scripts*, que deveria incorporar o modelo de dados da aplicação e propiciar formas naturais e de alto nível para tratá-lo. A existência de uma linguagem de programação poderosa, de propósito geral, por trás do ambiente de trabalho viabilizaria a construção de aplicações sofisticadas. Ao invés de se desenvolver uma nova linguagem, optou-se por adotar Python, por três razões:

1. baseia-se no paradigma de orientação a objetos (embora não tenha suporte completo para encapsulamento);

2. é facilmente extensível, graças à interface para programas em C;

3. já tem uma grande biblioteca de módulos prontos para as mais diversas finalidades, inclusive interfaces gráficas com o usuário.

Embora Tcl já fosse suportada pelo Vis5D, isto não incluia o toolkit Tk, conforme foi discutido na seção 3.1.4. Além disto, Tcl não se baseia no paradigma de objetos e sua sintaxe foi considerada pouco clara, o que dificultaria a manutenção de um sistema grande.

Para o módulo de interface com o usuário, há várias possibilidades, uma vez que ele é carregado dinamicamente. Inicialmente se pensou em adotar a combinação do módulo X-extension com o toolkit Motif. Entretanto, devido à incompatibilidade do módulo com versões mais recentes do interpretador Python (o autor do módulo não se interessou mais em atualizá-lo), aliada ao sucesso obtido com Tk na construção da maquete, optou-se por adotar este último.

A arquitetura mostrada na figura 5.5 prevê módulos diferentes para o boletim e para dados meteorológicos porque, devido ao tipo de representação interna do boletim (grafo), não seria prático tentar estender o modelo de dados do Vis5D para armazená-lo. Optou-se por tratar internamente o boletim como listas de vértices e arestas e arquivá-lo em formato de texto, o que requer apenas um interpretador para a leitura/gravação do boletim.

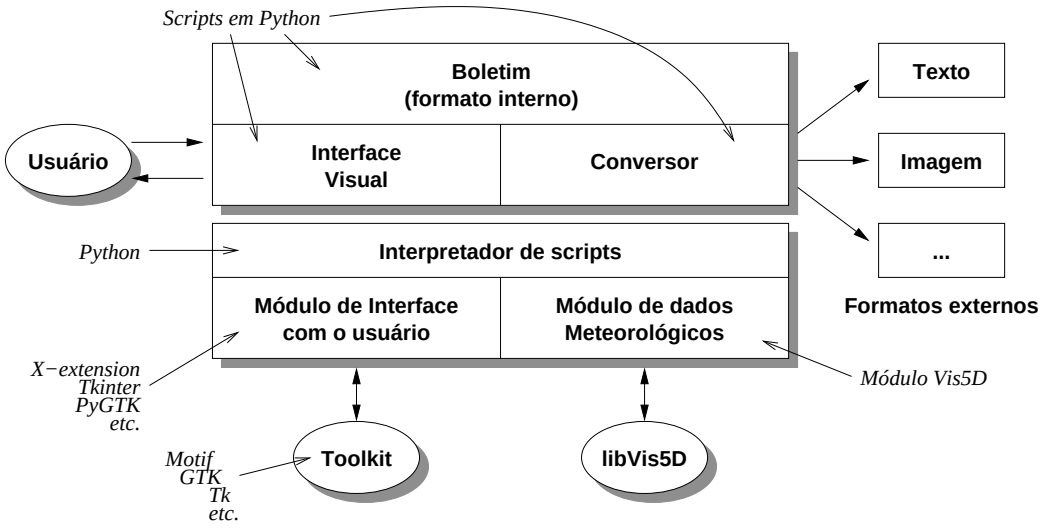


FIGURA 5.5 – Sistema construído sobre o *framework* Py5D

O framework se destina a prover a funcionalidade básica e uma API para acesso a tipos de dados (classes), dados e funções específicas enquadradas nas seguintes categorias:

- abstração do modelo de dados;
- interface gráfica (elementos de controle);
- representação visual dos dados (vistas).

Além dessas funções, específicas do framework, outras que seriam "herdadas" do software usado em sua construção: da linguagem Python, módulos para acesso a rede e comunicação de dados, acesso a banco de dados, serviços do sistema operacional, multimídia e muitas outras mais [VAN98b]; de Vis5D, se um modelo de representação de dados e um conjunto de funções destinadas à sua manipulação e visualização [HIB2000a, HIB2000b].

A forma de construção adotada foi a mesma utilizada normalmente para estender o interpretador Python [VAN98a]: módulos independentes, carregados dinamicamente. O uso do interpretador permitiria também a extensão do framework em tempo de execução, pela carga de novos módulos.

Assim como no caso da linguagem Pythonissa, foi escolhido para o framework um nome adequado: Py5D, significando "mescla de Pythonissa com Vis5D".

5.2.3 Considerações sobre a portabilidade

Um fator levado em consideração no desenvolvimento do sistema foi a portabilidade. Por isto se decidiu que o software produzido seria o menos dependente possível de algum sistema operacional em particular, podendo ser usado em qualquer plataforma compatível com UNIX e POSIX que tivesse o compilador C e as bibliotecas de funções necessárias para rodar Python e Vis5D.

O trabalho começou a ser feito em arquiteturas SPARC, com sistema operacional SunOS, e Intel x86, com sistemas GNU/Linux. Posteriormente seu desenvolvimento passou a ser feito em plataforma Intel com sistema operacional FreeBSD. Pretende-se tam-

bém que ele venha a rodar em ambiente Alpha com DEC UNIX, uma vez que o CPMet dispõe também desse tipo de equipamento.

5.3 Estágio atual do desenvolvimento

Das tarefas necessárias para a implementação do framework, apenas uma parte delas foi concluída. São elas:

- criação de um módulo, em Python, para tratamento de boletins de previsão do tempo, de acordo com o modelo definido na seção 4.3.2;
- criação de um módulo "vis5d" para Python, permitindo acesso à funções definidas na API do programa (o módulo foi desenvolvido por Alfredo Kojima);
- definição de representações gráficas para as classes (rótulos) de vértices e arestas;
- definição de um mapeamento entre rótulos de vértices e arestas para elementos da interface gráfica (forma visual da linguagem Pythonissa), implementada na maquete;
- definição do modelo de interface com o usuário (maquete);

Em seu estado atual, entretanto o framework não está completo e ainda carece de muitas melhorias. Isto se deve principalmente a problemas encontrados no trato com o Vis5D, a saber:

- Apesar de haver uma documentação detalhada da API, não há referências sobre a implementação. Exceto pelo próprio código-fonte, e por um artigo bastante antigo que descreve a estrutura original de programa [HIB90], as únicas fontes de consulta foram informações prestadas pelos autores do programa via e-mail.
- Tamanho e falta de organização do código. Vis5D é composto por aproximadamente 120 mil linhas de código-fonte em C (e alguns programas opcionais, em FORTRAN). Algumas partes do código sofreram sucessivas alterações e remendos ao longo do tempo de vida do programa, o que dificulta sobremaneira a sua compreensão.
- Dificuldade de separação entre a interface com o usuário e restante do programa. Existem muitas funções internas, que não fazem parte daquelas constituintes da API mas são chamadas por elas, que contém vínculos com elementos da interface com o usuário. Isto pode ser atribuído em parte à pobreza do toolkit LUI, usado pelo programa, e em parte a falhas na implementação.

Assim sendo, o framework é ainda um trabalho em andamento, que demandará mais tempo e trabalho de programação, para que seja concluído.

6 Conclusões

Ao término deste trabalho foram obtidos diversos resultados, alguns deles esperados desde o início, outros não. A análise desses resultados é o objeto deste capítulo.

6.1 Adequação ao plano de estudo e pesquisa

Deve-se analisar, inicialmente, se o que foi feito está de acordo com a proposta original, isto é, se foi seguido o rumo traçado inicialmente ou se houve desvios; e se houve, por que razões.

Comparando o que foi descrito nos capítulos 4 e 5 com o plano de estudo e pesquisa original, observar-se-á que a proposta inicial era estudar a aplicação de técnicas de manipulação direta em interfaces visuais, tendo como foco ferramentas para a elaboração da previsão.

O objetivo do trabalho era "especificar uma linguagem visual que permitisse editar, por meio de uma interface gráfica, as sentenças componentes do texto de uma previsão do tempo". Comparando-se isto com a especificação da linguagem, no capítulo 4, se conclui que o objetivo foi atingido parcialmente, por duas razões:

1. O modelo de boletim proposto, conquanto descreva satisfatoriamente a sintaxe, falha em parte no que diz respeito à semântica, o que o limita a uma especificação apenas semiformal, não formal.

2. Existem partes do boletim real que não estão previstas no modelo. Para isto já foram apresentadas justificativas no final da seção 4.3.3.

Ambas as deficiências do modelo se devem à complexidade nas relações dos diversos fenômenos entre si e com outros fatores que são seus causadores, o que foi discutido, ainda que brevemente, na seção 2.1. Apesar dessas limitações, o trabalho pode ser considerado bem-sucedido por três motivos:

1. Ainda que de maneira incompleta, o modelo de boletim provê uma representação adequada para os principais fenômenos cuja presença caracteriza uma situação sinótica.

2. Fez-se um mapeamento completo da representação interna, como grafos, tanto para uma representação como figura (ícones sobre um mapa) quanto para elementos de uma interface visual.

3. O modelo de boletim adota um formalismo conhecido, o que torna seu tratamento computacional mais fácil do que seria se fosse usada uma especificação informal.

Cabe lembrar que a proposta original não era a criação de um sistema capaz de fazer, por si só, a previsão. O fato de não se ter podido incluir no modelo a lógica temporal, de que se fala na seção 4.3.6, não o invalida, apenas reduz sua abrangência.

Análisado o objetivo principal do trabalho, resta avaliar o objetivo secundário: construir um protótipo para a avaliação da linguagem especificada. Sob este aspecto, cabe considerar que:

1. A maquete construída implementa todos os elementos da linguagem e permitiu comprovar também que é possível usar o mecanismo de manipulação direta para compor o boletim.

2. Embora se tenha decidido tentar integrar a linguagem a um software de visualização existente, de modo a poder testar a linguagem em um ambiente operacional, este não era o objetivo inicial.

6.2 Orientações para trabalhos futuros

Dentre as possibilidades de continuação do trabalho iniciado, destaca-se inicialmente a conclusão da implementação do framework Py5D. Existem bons motivos para isto:

- Do ponto de vista de engenharia de software, permitiria aprofundar o estudo da construção de aplicações modulares.

- Em termos de visualização científica, Py5D transcenderia o sistema no qual foi baseada sua concepção: Vis5D é um software de visualização dotado de uma linguagem de scripts. Py5D seria uma linguagem de scripts dotada de um módulo de visualização. É uma abordagem diferente, mais flexível.

- Sob a ótica do previsor do tempo, estudos adicionais seriam necessários para que se chegasse a um software em condições de ser usado no dia-a-dia de um centro operacional.

Existem razões, também para aprofundar o estudo sobre a formalização da linguagem e sobre a modelagem do boletim:

- O modelo proposto considerou apenas a realidade do CPMet/UFPEL. Ele é fortemente influenciado pela cultura do Centro, é bastante simplificado e provê apenas os recursos mínimos necessário aos objetivos deste trabalho.

- Embora se tenha concluído da impossibilidade da aplicação da lógica temporal, seria possível aplicar uma *lógica espacial*: os arcos de vizinhança poderiam ter um atributo correspondendo à distância entre as regiões (ou seus pontos centrais, melhor dizendo) que seria usado por regras de validação de uma situação sinótica. Isto impediria que se criasse, por exemplo, um boletim contendo previsão de temperatura elevada em uma região e baixa em outra região vizinha.

- O formalismo de camadas foi pouco explorado. Existe uma relação direta entre a camada em que se encontra um rótulo e o nível de seu controlador dentro da hierarquia de controladores da interface, conforme se pode ver na barra de fenômenos da maquete. Isto poderia ser usado para gerar dinamicamente a interface, ou pelo menos partes dela.

- A combinação do modelo formal com processamento de linguagem natural permitiria que a aplicação gerasse um texto mais suave. Na sua forma atual, o texto gerado pela maquete tem uma aparência artificial.

- A presença simultânea de alguns fenômenos poderia ser combinada em um único ícone (representando chuva, céu encoberto e trovoada, por exemplo), bem como em uma única frase. Há também expressões que aparecem comumente nos boletins que estão relacionadas à ausência ou presença de chuva: “tempo bom” (quando não chove) e “tempo instável” (com chuva ou com possibilidade de chuva).
- Não se tratou, no modelo do boletim, de fenômenos que são normalmente mencionados na análise sinótica, como frentes e massas de ar (quente/frio, seco/úmido, etc.). A presença desses fenômenos não é regionalizada e axige uma abordagem diferente.

Espera-se que no futuro seja dada continuidade ao estudo iniciado. Como se pode observar pelos ítens citados acima, há um grande número de possibilidades para que novos trabalhos sejam feitos a partir dos resultados obtidos.

Anexo Exemplo de boletim do tempo do CPMet

```
                 UNIVERSIDADE FEDERAL DE PELOTAS
                CENTRO DE PESQUISAS METEOROLOGICAS
                           C P M e t
=====================================================================
PREVISAO DO TEMPO PARA O RIO GRANDE DO SUL
TEXTO ELABORADO AS 16:14 HS DO DIA 14/07/99.
=====================================================================
PREVISAO VALIDA PARA O DIA 15 DE JULHO DE 1999 (QUINTA).
=====================================================================
ANALISE(16:00): A FRENTE FRIA ENTRA NO ESTADO CAUSANDO TEMPO INSTAVEL
                NAS REGIOES CENTRO/SUL, NAS DEMAIS REGIOES DO ESTADO
                TEMPO BOM.
                TEMPERATURAS EM ELEVACAO EM  TODAS AS REGIOES.
=====================================================================
DADOS EXTREMOS OBSERVADOS - 14/07/1999
FONTE: ESTACAO AGROCLIMATOLOGICA (C0NVENIO EMBRAPA/UFPEL)

        TEMPERATURA:              MINIMA : 11.0 GRAUS (AS 06:00 H)
                                  MAXIMA : 24.6 GRAUS (AS 14:00 H)
        UMIDADE RELATIVA:         MAXIMA : 100 % (AS 01:00 HS)
                                  MINIMA : 48 % (AS 13:30 HS)

        PRECIPITACAO (MEDIA) PARA O MES DE JULHO:         121.7 MM
        TOTAL ACUMULADO MENSAL (ATE 09:00 HS DIA 14/07/99): 5.4 MM
        DIARIA ACUMULADA (ATE AS 09:00 hs DO DIA 14/07/99): 0.0 MM
=====================================================================
CAPITAL:
        TEMPO INSTAVEL COM CHUVAS.
        CEU  NUBLADO.
        VENTOS DO QUADRANTE NORTE FRACOS.
        TEMPERATURA ESTAVEL.
        MAXIMA: 22/24 GRAUS
        MINIMA: 11/13 GRAUS
---------------------------------------------------------------------
LITORAL NORTE:
        TEMPO INSTAVEL COM CHUVAS.
        CEU  NUBLADO.
        VENTOS DO QUADRANTE NORTE FRACOS.
        TEMPERATURA ESTAVEL.

        TRAMANDAI                  MAXIMA: 22/24 GRAUS
                                   MINIMA: 11/13 GRAUS
---------------------------------------------------------------------
LITORAL SUL:
        TEMPO  INSTAVEL COM CHUVAS
        CEU  NUBLADO.
        VENTOS DO QUADRANTE NORTE FRACOS.
        TEMPERATURA ESTAVEL.

        PELOTAS                    MAXIMA: 20/26 GRAUS
                                   MINIMA: 10/12 GRAUS
---------------------------------------------------------------------
```

REGIAO NORTE:
TEMPO BOM.
CEU COM AUMENTO DE NUBULOSIDADE.
VENTOS DO QUADRANTE NORTE FRACOS.
TEMPERATURA EM ELEVACAO.

ERECHIM MAXIMA: 22/24 GRAUS
MINIMA: 14/16 GRAUS

REGIAO SUDOESTE:
TEMPO INSTAVEL COM CHUVAS.
CEU NUBLADO.
VENTOS DO QUADRANTE NORTE FRACOS.
TEMPERATURA ESTAVEL.

BAGE MAXIMA: 22/24 GRAUS
MINIMA: 13/15 GRAUS

REGIAO OESTE:
TEMPO INSTAVEL COM CHUVAS.
CEU NUBLADO.
VENTOS DO QUADRANTE NORTE FRACOS.
TEMPERATURA EM DECLINIO.

URUGUAIANA MAXIMA: 20/22 GRAUS
MINIMA: 10/12 GRAUS

REGIAO DAS MISSOES:
TEMPO BOM PASSANDO A INSTAVEL SUJEITO A CHUVAS.
CEU PARCIALMENTE NUBLADO A NUBLADO.
VENTOS DE NORDESTE FRACOS.
TEMPERATURA ESTAVEL.

SANTO ANGELO MAXIMA: 21/23 GRAUS
MINIMA: 13/15 GRAUS

REGIAO SERRANA:
TEMPO BOM PASSANDO A INSTAVEL SUJEITO A CHUVAS.
CEU PARCIALMENTE NUBLADO A NUBLADO.
VENTOS DO QUADRANTE NORTE FRACOS.
TEMPERATURA ESTAVEL.

CAXIAS MAXIMA: 20/22 GRAUS
MINIMA: 10/12 GRAUS

REGIAO DA DEPRESSAO CENTRAL:
TEMPO INSTAVEL SUJEITO A CHUVAS.
CEU NUBLADO.
VENTOS DO QUADRANTE NORTE FRACOS.
TEMPERATURA EM ELEVACAO.

SANTA MARIA MAXIMA: 22/24 GRAUS
MINIMA: 14/16 GRAUS

REGIAO DO PLANALTO:
TEMPO BOM.
CEU COM AUMENTO DE NEBULOSIDADE.
VENTOS DO QUADRANTE NORDESTE FRACOS.

TEMPERATURA EM ELEVACAO.

PASSO FUNDO MAXIMA: 24/26 GRAUS
MINIMA: 14/16 GRAUS

==========

TENDENCIA:

SEXTA (16/07): TEMPO INSTAVEL NAS REGIOES CENTRO/NORTE COM CHUVAS.
NAS DEMAIS REGIOES TEMPO BOM.
A TEMPERATURA EM DECLINIO NAS REGIOES CENTRO/SUL.

==========

METEOROLOGISTA: ELIANE ALVES.

==========

Bibliografia

[AIK96] AIKEN, A.; CHEN, J.; STONEBRAKER, M.; WOODRUFF, A. Tioga-2: a direct manipulation database visualization environment. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 12., 1996, New Orleans. **Proceedings**... [S.l.: s.n.], 1996. p.208-217.

[ANS85] AMERICAN NATIONAL STANDARDS INSTITUTE. **ANSI X3.124-1985**: American National Standard for Information Systems: Computer Graphics: Graphical Kernel System (GKS) Functional Description. New York, 1985.

[ANS86] AMERICAN NATIONAL STANDARDS INSTITUTE. **ANSI X3.122-1986**: American National Standard: Computer Graphics: Metafile for the Storage and Transfer of Picture Description Information. New York, 1986.

[AND99] ANDRIES, M. et al. Graph transformation for specification and programming. **Science of Computer Programming**, [S.l.], v.34, n.1, p.1–54, Jan. 1999.

[ALM98a] ALMEIDA, E. S. et al. Extração e manipulação dos dados de reanálise do ECMWF utilizando Metview. In: CONGRESSO BRASILEIRO DE METEOROLOGIA, 10., 1998. **Anais**... Rio de Janeiro: Sociedade Brasileira de Meteorologia, 1998. 1 CD-ROM.

[ALM98b] ALMEIDA, E. S. et al. Visualização dos modelos ETA e global do CPTEC/INPE através do sistema Vis5D. In: CONGRESSO BRASILEIRO DE METEOROLOGIA, 10., 1998. **Anais**... Rio de Janeiro: Sociedade Brasileira de Meteorologia, 1998. 1 CD-ROM.

[APP95] APPLE COMPUTER. **Getting Started with HyperCard 2.3**. Cupertino, 1995. Disponível em: <`ftp://manuals.info.apple.com/Apple_Support_Area/Manuals/software/0306857AHC2.3GS.PDF`>. Acesso em: 20 dez. 2000.

[AVS99] ADVANCED VISUAL SYSTEMS. **AVS5**: The Original Visual Programming Data Visualization Computing Environment. Disponível em: <`http://www.avs.com/products/AVS5/avs5.htm`>. Acesso em: 20 dez. 2000.

[BAR97] BARDOHL, R. **Application of Graph Transformations to visual languages**. Berlin: Technical University Berlin, 1997. 27p. (Technical report 97–10)

[BIE93] BIER, E. A.; STONE, M. C.; PIER, K.; BUXTON, W.; DEROSE, T. D. Toolglass and Magic Lenses: The see-through interface. In: SIGGRAPH, 1993, Anaheim. **Proceedings...** New York: ACM Press, 1993. p.73–80.

[BOS93] BOS, G. **Rapid user interface development with the scripting language Gist**. 1993. Tese. (Doutorado) – Groningen. Disponível em: <`http://www.let.rug.nl/~bert/dissertation.ps.gz`>. Acesso em: 20 dez. 2000.

[BRA8?a] BRASIL. Instituto Nacional de Meteorologia. **Manual de Operação**. [S.l.,

198?].

[BRA8?b] BRASIL. Instituto Nacional de Meteorologia. **Código SYNOP FM 12**. [S.l., 198?].

[BRA8?c] BRASIL. Ministério da Aeronáutica, Comando Geral de Apoio, Diretoria de Eletrônica e Proteção ao Vôo. **Mensagem Meteorológica de Altitude — TEMP**. Brasília, [198?].

[BRA93] BRASIL, Ministério da Aeronáutica, Comando Geral de Apoio, Diretoria de Eletrônica e proteção ao Vôo. **FMA 105-6 — Códigos METAR e SPECI**. Brasília: DEPV, 1993.

[BRI99] BRICKLIN, D. **Was VisiCalc the "first" spreadsheet?** Disponível em: <http://www.bricklin.com/firstspreadsheetquestion.htm>. Acesso em: 20 dez. 2000.

[BRY99] BRYSON, S. et al. Visually exploring gigabyte data sets in real time. **Communications of the ACM**, New York, v.42, n.8, p.83–90, Aug. 1999.

[BUR94] BURNETT M.; HOSSLI, R.; PULLIAM, T.; VANVOORST, B.; YANG X. Toward Visual Programming Languages for Steering Scientific Computation. **IEEE Computational Science & Engineering**, Los Alamitos, v.1, n.4, p. 44-62, Winter 1994.

[COR98] CORRÊA, Patrícia C. P. et al. Animação automática de dados meteorológicos utilizando o sistema Metview. In: CONGRESSO BRASILEIRO DE METEOROLOGIA, 10., 1998. **Anais**... Rio de Janeiro: Sociedade Brasileira de Meteorologia, 1998. 1 CD–ROM.

[DEY98] DEY, C. H. **GRIB (Edition 1)**. [S.l.]: NOAA/NCEP, 1998. Disponível em: <ftp://nic.fb4.noaa.gov/pub/nws/nmc/docs/gribed1/>. Acesso em: 11 jan. 1999.

[DOT95] DOTY, B.; HOLT, T.; FIORINO, M. **Grid Analysis and Display System**. 1995. Disponível em: <ftp://grads.iges.org/grads/sprite/doc/>. Acesso em: 11 jan. 1999.

[DRA95] DRAKE, J.; FOSTER, I. Introduction on the special issue on parallel computing in climate and weather modeling. **Parallel computing**, [S.l.], v.21, n.10, Oct. 1995.

[EAR92] EARNSHAW, R. A.; WISEMAN, N. **An Introductory Guide to Scientific Visualization**. Berlin: Springer-Verlag, 1992. 156p.

[ECM97] ECMWF; BRASIL. Instituto Nacional de Pesquisas Espaciais. **Metview Users Manual**. 1997. Disponínel em: <http://www.cptec.inpe.br/products/metview/>. Acesso em 11 jan. 1999.

[EDW96] EDWARDS, G. J. Implementation of the data acquisition component of the WFO–Advanced 2d display (D2D). In: INTERNATIONAL CONFERENCE ON INTERACTIVE INFORMATION AND PROCESSING SYSTEMS FOR METEOROLOGY, OCEANOGRAPHY AND HIDROLOGY, 12., 1996, Atlanta. **Proceedings**... Atlanta: American Meteorology Society, 1996.

[EHR79] EHRIG, H. Introduction to the algebric theory of Graph Grammars. In: INTERNATIONAL WORKSHOP ON GRAPH-GRAMMARS AND THEIR APPLICATION TO COMPUTER SCIENCE AND BIOLOGY, 1979. **Proceedings**... Berlin: Springer-Verlag, 1979. p.1–27.

[EMB9?a] EMPRESA BRASILEIRA DE TELECOMUNICAÇÕES. **RENPAC 3025 e 2032**: descrição funcional. Rio de Janeiro, [199?]

[EMB9?b] EMPRESA BRASILEIRA DE TELECOMUNICAÇÕES. **RENPAC 3028 e 2028**: descrição funcional. Rio de Janeiro, [199?]

[FED99] FEDOROVA, N. **Meteorologia Sinótica**. Pelotas: UFPEL: Ed. Universitária, 1999.

[FRE92] FREITAS, C. M. D. S. **Um estudo sistemático sobre linguagens visuais**. Porto Alegre: CPGCC/UFRGS, 1992. 86p. Relatório de Pesquisa.

[FUR86] FURNAS, G. W. Generalized fisheye views. In: ACM CHI CONFERENCE, 1986, Boston. **Proceedings**... New York: ACM Press, 1986. p.16–23.

[GAM95] GAMMA, E. et al. **Design Patterns**: elements of reusable object-oriented software. Reading: Addison-Wesley, 1995.

[GET96] GETTYS, J.; SCHEIFLER, R. W. **Xlib:** C language X interface. Cambridge: X Consortium, Inc., 1996.

[GET98] GETTYS, J. **The two edged sword**. Disponível em: <`http://freshmeat.net/editorials/`>. Acesso em 11 jan. 1999.

[GRO97] GROTE, U. H.; BULLOCK, C. S. User interface design of the WFO–Advanced workstation. In: INTERNATIONAL CONFERENCE ON INTERACTIVE INFORMATION AND PROCESSING SYSTEMS FOR METEOROLOGY, OCEANOGRAPHY AND HIDROLOGY, 13., 1997, Long Beach. **Proceedings**... Long Beach: American Meteorology Society, 1996. p.190–193.

[HIB90] HIBBARD, W.; SANTEK, D. The Vis5D system for easy interactive visualization. In: VISUALIZATION, 1990, San Francisco. **Proceedings**... Los Alamitos: IEEE Computer Society Press, 1990. p.28–35.

[HIB92] HIBBARD, W.; DYER, C.; PAUL, B. Display of scientific data structures for algorithm visualization. In: VISUALIZATION, 1992, Boston. **Proceedings**... Los Alamitos: IEEE Computer Society Press, 1992. p.139–146.

[HIB98a] HIBBARD, W. VisAD: Connecting people to computing and people to people. **ACM SIGGRAPH Newsletter**, New York, v.32, n.3, Aug. 1998. Disponível em: `http://www.siggraph.org/publications/newsletter/v32n3/columns/elvins.html`. Acesso em 18 dez. 1998.

[HIB98b] HIBBARD, W. **Vis5D Home Page**. Disponível em: <`http://www.ssec.wisc.edu/~billh/vis5d.html`>. Acesso em 18 dez. 1998.

[HIB2000a] HIBBARD, W. **Vis5D API**. Disponível em: <`http://www.ssec.wisc.edu/~billh/api52.html`>. Acesso em 29 nov. 2000.

[HIB2000b] HIBBARD, W. **Vis5D Scripting**. Disponível em: <`http://www.ssec.wisc.edu/~billh/script52.html`>. Acesso em 29 nov. 2000.

[HOB95] HOBART, J. Principles of Good GUI Design. **Unix Review**, San Francisco, v. 13, n. 10, p.37–46, Sept. 1995. Disponível em <`http://axp16.iie.org.mx/Monitor/v01n03/ar_ihc2.htm`>. Acesso em: 20 dez. 2000.

[IBM99] INTERNATIONAL BUSINESS MACHINES. **Open Visualization Data Explorer**. Disponível em: <`http://www.research.ibm.com/`

dx/>. Acesso em: 20 dez. 2000.

[IMA98] **IMAGEMAGICK**: X11 Image Processing and Display Package. Disponível em: <http://www.wizards.dupont.com/cristy/ImageMagick.html>. Acesso em: 4 dez. 1998.

[JER98] JERDING, D.; STASKO, J. The Information Mural: a technique for displaying and navigating large information spaces. **IEEE Transactions on Visualization and Computer Graphics**, Los Alamitos, v.4, n.3, p.257–271, July 1998.

[JOH2000] JOHNSON, Steven G. **Vis5D+**. Disponível em: <http://vis5d.sourceforge.net/>. Acesso em 04 dez. 2000.

[KAR95] KARHILA, V. E. Metview in a member state: a dream or a reality? In: WORKSHOP ON METEOROLOGICAL OPERATIONAL SYSTEMS, 5., 1995, Reading. **Proceedings**... Reading: ECMWF, 1995. p.172–174.

[KHO99] KHORAL INC. **Khoros**. Disponível em: <http://www.khoral.com/khoros/>. Acesso em 04 dez. 2000.

[LAL91] LALONDE, W. R.; PUGH, J. R. **Inside Smalltalk**. Englewood Cliffs: Prentice-Hall, 1991.

[LAU91] LAU-KEE, D. et al. VPL: an active, declarative visual programming system. In: IEEE WORKSHOP ON VISUAL LANGUAGES, 1991. **Proceedings**... Los Alamitos: IEEE Computer Society Press, 1991. p.40-46.

[LAV92] LAVISOLO, E. (Ed.). **Larousse Cultural — Dicionário da Língua Portuguesa**. São Paulo: Nova Cultural, 1992. 1176p.

[LIC89] LICKLIDER, T. R. Ten years of rows and columns (spread sheet programs). **Byte Magazine**, New York, v.14, n.13, p.324–331, Dec. 1989

[LOH94] LOHSE, G. L. et al. A classification of visual representations. **Communications of the ACM**, New York, v.37, n.12, p.36–49, Dec. 1994.

[MAC96] MACDONALD, A. E.; WAKEFIELD, J. S. WFO–Advanced: an AWIPS–like prototype forecaster workstation. In: INTERNATIONAL CONFERENCE ON INTERACTIVE INFORMATION AND PROCESSING SYSTEMS FOR METEOROLOGY, OCEANOGRAPHY AND HIDROLOGY, 12., 1996, Atlanta. **Proceedings**... Atlanta: American Meteorology Society, 1996. p.190–193.

[MAT96] MATHEWSON, M. A. Using the AWIPS Forecast Preparation System (AFPS). In: INTERNATIONAL CONFERENCE ON INTERACTIVE INFORMATION AND PROCESSING SYSTEMS FOR METEOROLOGY, OCEANOGRAPHY AND HIDROLOGY, 12., 1996, Atlanta. **Proceedings**... Atlanta: American Meteorology Society, 1996.

[MAX93] MAX, N.; CRAWFIS, R.; WILLIAMS, D. Visualizations for Climate Modeling. **IEEE Computer Graphics and Applications**, Los Alamitos, v.13, n.4, p.34–40, July 1993.

[MCC87] MCCORMICK, B. H.; DEFANTI, T. A.; BROWN, M. D. Special report: Visualization in Scientific Computing – a synopsis. **IEEE Computer Graphics and Applications**, Los Alamitos, v.21, n.6, p.61–70, July 1987.

[MED99] MEDO de ciclone provoca confusão. **Correio do Povo**, Porto Alegre, 29 maio 1999. Disponível em: <http://www.cpovo.net/jornal/

a104/n241/html/07medo9d.htm>. Acesso em: 20 dez. 2000.

[MOU86] MOURA, A. D. Evolução da meteorologia: da Babilônia aos nossos dias. **Revista Brasileira de Tecnologia**, Brasília, v.17, n.1, p.5–14, 1986.

[MYE93] MYERS, B. A. **Why are Human-Computer Interfaces Difficult to Design and Implement?** Pittsburgh: Carnegie Mellon University, School of Computer Science, July 1993. (Technical Report n.CMU-CS-93-183) Disponível em: <http://www.cs.cmu.edu/afs/cs.cmu.edu/project/garnet/doc/papers/whyhardTR.ps>. Acesso em: 20 dez. 2000.

[NCA95] NATIONAL CENTER FOR ATMOSPHERIC RESEARCH. **NCAR Graphics Fundamentals**. Boulder, 1995. 1 CD–ROM. Disponível em <http://ngwww.ucar.edu/ngdoc/ng/fund/fundhome.html>. Acesso em: 20 dez. 2000.

[NCS99] NATIONAL CENTER FOR SUPERCOMPUTING APPLICATIONS. **NCSA HDF Home Page**. Disponível em: <http://hdf.ncsa.uiuc.edu>. Acesso em 14 jan. 1999.

[NOA99] NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION. **AFPS Quick Start Guide**. Boulder, 1999.

[NUN97] NUNES, D. J.; RIBEIRO, L.; KORFF, M. **Métodos formais para especificações:** gramática de Grafos. Pelotas: UFPEL, 1997. 44p. Minicurso apresentado na 1. Escola de métodos formais para qualidade de software, 1997, Pelotas.

[OLI98] OLIVEIRA, V. M. **Condições para formação de nevoeiro na região de Pelotas**. 1998. Dissertação (Mestrado) — Universidade Federal de Pelotas, Pelotas.

[OPE97a] OPEN GROUP. **Motif 2.1 — Programmer's Guide**. Cambridge, MA, 1997.

[OPE97b] OPEN GROUP. **CDE 2.1/Motif 2.1 — Style Guide and Glossary**. Cambridge, MA, 1997.

[OUS94] OUSTERHOUT, J. K. **Tcl and the Tk toolkit**. Reading: Addison-Wesley, 1994. 458p.

[PAU95] PAULI, C.; MATTER, D.; AMBROSETTI, P. Swiss Meteorological Workstation Project. In: WORKSHOP ON METEOROLOGICAL OPERATIONAL SYSTEMS, 5., 1995, Reading. **Proceedings**... Reading: ECMWF, 1995. 321p. p.273–282.

[PED98] PEDROTTI, C. B.; FEDOROVA, N. Processos de formação de geada em pelotas no ano de 1996. In: CONGRESSO BRASILEIRO DE METEOROLOGIA, 10., 1998. **Anais**... Rio de Janeiro: Sociedade Brasileira de Meteorologia, 1998. 1 CD–ROM.

[PER93] PERLIN, K.; FOX, D. Pad: An alternative apprach to the computer interface. In: SIGGRAPH, 1993, Anaheim. **Proceedings**... New York: ACM Press, 1993. p.57–64.

[PER98] PEROUTKA, M. R.; MEIGGS, R. K.; ROMBERG, M. B. The Generation of Products In Interactive Forecast Preparation. In: INTERNATIONAL CONFERENCE ON INTERACTIVE INFORMATION AND PROCESSING SYSTEMS FOR METEOROLOGY, OCEANOGRAPHY AND

HIDROLOGY, 14., 1998, Phoenix. **Proceedings**... Phoenix: American Meteorology Society, 1998. Disponível em: <http://www.nws.noaa.gov/msm/ifp/prdppr.htm>. Acesso em 05 maio 1999.

[PES98] PESQUERO, J. F.; SATYARMURTY, P. Um sistema operacional de visualização de produtos meteorológicos. In: CONGRESSO BRASILEIRO DE METEOROLOGIA, 10., 1998. **Anais**... Rio de Janeiro: Sociedade Brasileira de Meteorologia, 1998. 1 CD–ROM.

[PET94] PETERSON, C. D. **Athena Widget Set — C Language Interface**. Maynard: Digital Equipment Corporation/X Consortium, 1994.

[REK94] REKERS, J. **On the use of Graph Grammars for defining the syntax of graphical languages**. Leiden: Leiden University, 1994. (Technical Report 94-11). Disponível em: <ftp://ftp.wi.leidenuniv.nl/pub/CS/TechnicalReports/1994/tr94-11.ps.gz>. Acesso em: 20 dez. 2000.

[REK95] REKERS, J.; SCHÜRR, A. **A parsing algorithm for context-sensitive Graph Grammars**. Leiden: Leiden University, 1995. (Technical Report 95-05) Disponível em: <ftp://ftp.wi.leidenuniv.nl/pub/CS/TechnicalReports/1995/tr95-05.ps.gz>. Acesso em: 20 dez. 2000.

[REK96] REKERS, J.; SCHÜRR, A. **Defining and parsing visual languages with layered Graph Grammars**. Leiden: Leiden University, 1996. (Technical Report 96-09). Disponível em: <ftp://ftp.wi.leidenuniv.nl/pub/CS/TechnicalReports/1996/tr96-09.ps.gz>. Acesso em: 20 dez. 2000.

[REP95] REPORT of the Working Group on Meteorological Workstations. In: WORKSHOP ON METEOROLOGICAL OPERATIONAL SYSTEMS, 5., 1995, Reading. **Proceedings**... Reading: ECMWF, 1995. p.16–17.

[RIB94] RIBARSKI, W.; AYERS, E.; EBLE, J.; SOUGATA, M. Glyphmaker: creating customized visualizations of complex data. **IEEE Computer**, Los Alamitos, v.27, n.7, p.57–64, July 1994.

[SAN99] SANTOS, C. A. M. **Recursos de visualização científica necessários para um sistema meteorológico operacional**. 1999. 81p. Trabalho Individual (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

[SBM96] CONGRESSO BRASILEIRO DE METEOROLOGIA, 9., 1996, Campos do Jordão. **Anais**... Rio de Janeiro: Sociedade Brasileira de Meteorologia, 1996.

[SBM98] CONGRESSO BRASILEIRO DE METEOROLOGIA, 10., 1998. **Anais**... Rio de Janeiro: Sociedade Brasileira de Meteorologia, 1998. 1 CD–ROM.

[SCH86] SCHEIFLER, R.; GETTYS, J. The X Window System. **ACM Transactions on Graphics**, New York, v.5, n.2, p.79–109, Apr. 1986.

[SCH96] SCHOEREDER, W.; MARTIN, K.; LORENSEN, W. **The visualization toolkit**: an object-oriented approach to 3D graphics. Upper Saddle River: Prentice Hall PTR, 1996. 826p.

[SHN83] SHNEIDERMAN, B. Direct Manipulation: A Step Beyond Programming Languages. **IEEE Computer**, Los Alamitos, v.6, n.8, p.57–69, Aug. 1983.

[SHN98] SHNEIDERMAN, B. **Designing the user interface**. Reading: Addison-Wesley, 1998. Chap.6, p.185–234

[SHU86] SHU, N. C. Visual programming languages: a perspective and a dimensional analysis. **IBM Systems Journal**, [S.l.], v.28, n.4, p.525–547, 1989.

[STA2000] STALLMAN, Richard. **The GNU General Public Licence**. Disponível em: <http://www.gnu.org/copyleft/gpl.html>. Acesso em: 17 ago. 2000.

[TAN2000] TANENBAUM, A. S. **Andrew S. Tanenbaum's Personal FAQ (Frequently Asked Questions)**. Disponível em: <http://www.cs.vu.nl/~ast/home/faq.html>. Acesso em: 17 ago. 2000.

[THO94] THORPE, W. **A Guide to the WMO code form FM 94 BUFR**. Monterey: NOAA/Fleet Numerical Meteorology and Oceanography Center, 1994. Disponível em: <ftp://nic.fb4.noaa.gov/pub/nws/nmc/docs/bufrguide/>. Acesso em 11 jan. 1999.

[TUB80] TUBELIS, A.; NASCIMENTO, F. **Meteorologia descritiva**: fundamentos e aplicações brasileiras. São Paulo: Nobel, 1980.

[UNI99] UNIVERSITY CORPORATION FOR ATMOSPHERIC RESEARCH. **Unidata NetCDF**. Disponível em: <http://www.unidata.ucar.edu/packages/netcdf/>. Acesso em 11 jan. 1999.

[UPS89] UPSON, C. et al. The Application Visualization System: a computational environment for Scientfic Visualization. **IEEE Computer Graphics and Applications**, Los Alamitos, v.9, n.7, p.30–42, July 1989.

[VAN98a] VAN ROSSUM, G. **Extending and embeding the Python interpreter**. Amsterdam: Stichting Mathematitisch Centrum, 1998. 60p.

[VAN98b] VAN ROSSUM, G. **Python library reference**. Amsterdam: Stichting Mathematitisch Centrum, 1998. 60p.

[VIA91] VIANELLO, R. L.; ALVES, A. R. **Meteorologia Básica e Aplicações**. Viçosa: UFV: Imprensa Universitária, 1991. 449p.

[WOO93] WOODRUFF, A.; LANDAY, J.; STONEBRAKER, M. Goal-directed zoom. In: SIGCHI, 1998, Los Angeles. **Proceedings**... New York: ACM Press, 1993. p.305–306.

[YAM96] YAMAZAKI, Y.; BAKST, L.; SANTOS, C. A. M.; SILVA, R. S. Sistema Visimet de distribuição e visualização de informações meteorológicas. In: CONGRESSO BRASILEIRO DE METEOROLOGIA, 9., 1996, Campos do Jordão. **Anais**... Rio de Janeiro: Sociedade Brasileira de Meteorologia, 1996. p.26-29. .

Créditos

Este texto foi redigido usando o editor de documentos **LyX**, criado por MATTIAS ETTRICH e mantido pelo LyX Team. LyX é uma interface visual para o **LaTeX**, criado por LESLIE LAMPORT com base no sistema de *tipografia eletrônica* **TeX**, de DONALD KNUTH. A distribuição de LaTeX usada foi **teTeX**, de THOMAS ESSER. Foram criados uma classe de documento para o LaTeX e um *layout* correspondente para o LyX, para os quais contribuíram GUSTAVO TORRES, FÁBIO OLIVÉ LEITE, e RAFAEL BOHRER ÁVILA. Também foram de grande valia as sugestões dadas por membros da lista brasileira de usuários de TeX, com destaque para KLAUS STEDING-JESSEN, autor do "LaTeX demo", do qual copiei muitas idéias.

As figuras foram desenhadas com **XFig**, de KEN YAP, e convertidas para Encapsulated PostScript (EPS) com **TransFig**, de MICAH BECK, ambos mantidos atualmente por BRIAN SMITH.

Imagens em PostScript foram convertidas para EPS usando **ps2epsi**, de GEORGE CAMERON, incluído no pacote **PSutils**, mantido por ANGUS DUGGAN. Imagens em formato JPEG foram encapsuladas em EPS com **jpeg2ps**, de THOMAS MERZ. Capturas de tela e conversões de formato de imagens foram feitas com **Xv**, de JOHN BRADLEY, usando o servidor X virtual **Xnest**, de DAVOR MATIC (MIT X Consortium).

Para a pré-visualização usou-se **xdvi**, de ERIC COOPER (modificado por ROBERT SCHEIFLER, MARK EICHIN e muitos outros) e **GV**, de JOHANNES PLASS (baseado no **GhostView**, de TIM THEISEN). GV é uma interface visual para o **Ghostscript**, de L. PETER DEUTSCH.

As fontes usadas na impressão foram: **Times** (nos estilos Upright, *Slanted*, e *Italic*), **Courier**, Helvetica e Computer Modern (no estilo *CALLIGRAPHIC*).