ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

EDSON MURAKAMI

**Uma infra-estrutura de desenvolvimento de sistemas de informação
orientados a serviços distribuídos para agricultura de precisão**

São Paulo
2006

EDSON MURAKAMI

**Uma infra-estrutura de desenvolvimento de sistemas de informação
orientados a serviços distribuídos para agricultura de precisão**

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para a obtenção do
título de Doutor em Engenharia.

Área de concentração: Sistemas Digitais
Orientador: Professor Livre Docente Antonio
Mauro Saraiva.

São Paulo
2006

AUTORIZO A REPRODUÇÃO E DIVULGAÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Murakami, Edson

**Uma infra-estrutura de desenvolvimento de sistemas de informação orientados a serviços distribuídos para agricultura de precisão / E. Murakami. -- São Paulo, 2006.
192 p.**

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Sistemas de informação 2.Sistemas distribuídos 3.Arquitetura de software 4.Agricultura de precisão I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

Dedico este trabalho à minha querida esposa
Lúcia e aos nossos filhos Bia e Lucas.

AGRADECIMENTOS

A Deus em primeiro lugar, que me colocou com as pessoas que me ajudaram nesse trabalho.

À minha querida família, Lúcia, Bia e Lucas, que compreenderam a minha ausência e sempre me deram apoio, amor e carinho, fortalecendo-me sempre.

Ao professor Livre Docente Antonio Mauro Saraiva, meu orientador e amigo, que nos anos de convivência, muito me incentivou e ensinou, visando meu amadurecimento como pesquisador.

Ao prof. Dr. Carlos Eduardo Cugnasca, com quem comecei este trabalho de pesquisa, por ter me dado a oportunidade de realizar este curso.

Ao prof. Dr. José Paulo Molin e ao Leonardo Afonso Angeli Menegatti, ambos da Escola Superior de Agricultura “Luiz de Queiroz”, pela colaboração no desenvolvimento do protótipo de filtragem de dados de produtividade.

Aos meus pais que sempre me apoiaram e acreditaram em mim.

RESUMO

MURAKAMI, EDSON. **Uma infra-estrutura de desenvolvimento de sistemas de informação orientados a serviços distribuídos para agricultura de precisão**. 2006. 192 f. Tese (doutorado) - Escola Politécnica da Universidade de São Paulo, São Paulo, 2006.

Interpretar a enorme quantidade de dados coletados, entender as causas e propor estratégias para gerenciar a variabilidade do campo, freqüentemente são apontados como alguns dos principais problemas para o avanço da agricultura de precisão, AP. Os sistemas de informação tornam-se fundamentais na solução desses problemas, mas apesar de existirem muitos pacotes de software disponíveis no mercado, variando de muito simples a muito sofisticados e diversos sistemas originados de experiências de pesquisas, a natureza proprietária e monolítica das soluções impedem o uso em larga escala. A AP envolve uma grande variedade de técnicas de análise, fontes e formatos de dados, perfis de usuário, e muitos outros aspectos que tornam uma aplicação muito complexa do ponto de vista da engenharia de software. Com o objetivo de fornecer a base para o desenvolvimento de sistemas de informação para AP baseados em padrões abertos e plataformas de software livre, uma infra-estrutura de desenvolvimento de sistemas de informação para a agricultura de precisão é proposta. Com base nas idéias seminais dessa proposta, são desenvolvidos protótipos para a condução de experimentos, os quais exploram caminhos de evolução para a infra-estrutura, com especial atenção sobre aspectos de arquitetura de software. Como estudo de caso, uma aplicação web que realiza filtragem de dados de monitores de produtividade é apresentada. Usando a metodologia de desenvolvimento em espiral, sucessivas versões dessa aplicação foram criadas e os resultados usados para propor melhoramentos à infra-estrutura. A infra-estrutura final contém cinco componentes: uma arquitetura de referência para sistemas de informação orientados a serviços para AP, uma linguagem padrão para troca de dados entre serviços agrícolas, um barramento para conexão de serviços agrícolas, um provedor de serviços geoespaciais e um portal para serviços agrícolas. Ela se mostrou adequada para a criação de sistemas de informação para AP interoperáveis, de baixo custo e com capacidade de evolução, mudando o paradigma dos sistemas para AP preponderantemente proprietários e monolíticos para abertos e orientados a serviços distribuídos.

Palavras-chave: Padrões abertos. Arquitetura orientada a serviço. Sistemas de informação. Agricultura de precisão.

ABSTRACT

MURAKAMI, EDSON. **An infrastructure for developing distributed service-oriented information systems for precision agriculture**. 2006. 192 f. Thesis (doctoral) - Escola Politécnica da Universidade de São Paulo, São Paulo, 2006.

Interpreting the huge amount of data collected, understanding the causes and being able to propose sound strategies for the field variability management are frequently pointed out as major issues for the advance of precision agriculture. Therefore, the information systems become fundamental for the solution of these problems. Although there are many available software packages in the market, varying from simple to very sophisticated and diverse systems deriving from experiences of research, the monolithic and proprietary nature of the solutions hinder their use in wide scale. Precision agriculture involves a great variety of techniques of data analysis, data sources, data formats, user profiles, and many other aspects that make it a complex application from the software engineering point of view. Aiming to supply the base for the development of open standards-based precision agriculture information systems and free software platforms, an infrastructure for developing precision agriculture information systems is proposed. Based on the fundamentals of that proposal, prototypes are developed which explore different evolutionary paths for the infrastructure, with special attention to software architecture aspects. As a case study, a yield monitor data filtering web application is presented. Using the spiral development methodology, successive versions of this application were created and the results used to improve the infrastructure. The final infrastructure contains five components: a service-oriented reference architecture for precision agriculture information systems, a standard language for data exchange between agricultural services, a service bus for connecting agricultural services, a geospatial services provider, and an agricultural services portal. It revealed to be adequate for the creation of precision agriculture interoperable systems, of low cost and with capacity for evolution, moving the paradigm of systems for AP preponderantly monolithic and proprietary to open and distributed service-oriented.

Keywords: Open standards. Service-oriented architecture. Information systems. Precision agriculture.

LISTA DE FIGURAS

FIGURA 2.1 – FLUXO PARA CRIAÇÃO DE PÁGINAS WEB EM TEMPO REAL.....	29
FIGURA 2.2- ESTRUTURA DO CROP MODELS ANALIST	32
FIGURA 2.3 – ESQUEMA DA INTERFACE DO PCYIELD E SUAS LIGAÇÕES COM DADOS NA INTERNET, MODELO CROPGRO-SOYBEAN, ARQUIVOS DE DADOS E USUÁRIOS	34
FIGURA 2.4 - ARQUITETURA DO SISTEMA DE SIMULAÇÃO BASEADO NA WEB PARA TRANSPORTE E RETENÇÃO DE CONTAMINANTES NO SOLO	37
FIGURA 2.5 – INTERFACE GRÁFICA DO VESPER	40
FIGURA 2.6 – RESULTADO DO PROCESSAMENTO DO VESPER.....	41
FIGURA 2.7 – INTERFACE DO IRMDSS	44
FIGURA 2.8 – ABORDAGEM DO FMS	55
FIGURA 2.9 – PROCESSO DE TRANSFORMAÇÃO DOS REQUISITOS	55
FIGURA 2.10 – INTERFACE DO LACTUS v1.01	56
FIGURA 2.11 – ARQUITETURA EM CAMADAS	57
FIGURA 2.12 – FERRAMENTA PARA CONSTRUÇÃO DE SISTEMAS DE APOIO A DECISÃO PARA CONTROLE DE PRAGAS	58
FIGURA 2.13 – ESTRUTURA PRINCIPAL E PARCEIROS DENTRO DE UMA REDE AGRÍCOLA	61
FIGURA 2.14 – INTERFACE GRÁFICA DA PLATAFORMA DE SOFTWARE.....	61
FIGURA 2.15 – UM REGISTRO DO ARQUIVO RDS	64
FIGURA 2.16 – UM REGISTRO DO ARQUIVO DO AGROMAP, DELIMITADO POR TABULAÇÃO, CRIADO A PARTIR DO RDS	64
FIGURA 2.17 – MODELO DE RELEVO TRI-DIMENSIONAL USANDO DADOS OBTIDOS PELO RDS	65
FIGURA 2.18 – FLUXO DE CONVERSÃO DE UM CONJUNTO DE DADOS PARA O AGRIS AP	73
FIGURA 2.19 – ARQUITETURA PARA AS APLICAÇÕES	74
FIGURA 2.20 – ARQUITETURA DO SDI PARA AGRICULTURA DE PRECISÃO.....	77
FIGURA 2.21 – INTERFACE GRÁFICA PARA VISUALIZAÇÃO DE MAPAS.....	79
FIGURA 2.22 – SELEÇÃO DOS MAIS RECENTES E PRINCIPAIS MOTIVAÇÕES PARA O USO DAS TECNOLOGIAS DA GEOINFORMAÇÃO NA AGRICULTURA	81
FIGURA 2.23 – MÚLTIPLOS ASPECTOS DOS REQUISITOS DA GEOINFORMAÇÃO	82
FIGURA 2.24 - DIAGRAMA DE CASOS DE USO DO MOSAICO	84
FIGURA 2.25 - CASOS DE USO DO MOSAICO	85
FIGURA 2.26 – DIAGRAMA DE SUBSISTEMAS DO MOSAICO.....	85
FIGURA 3.1 – RELACIONAMENTO ENTRE MODELO DE REFERÊNCIA, ESTILO DE ARQUITETURA E ARQUITETURA DE REFERÊNCIA IMPLEMENTANDO UMA ARQUITETURA DE SOFTWARE.....	104
FIGURA 3.2 – PILHA DE TECNOLOGIAS W3C	106
FIGURA 3.3 - MODELO ARQUITETURAL DOS WEB SERVICES.....	109
FIGURA 3.4 – EMPILHAMENTO CONCEITUAL DAS TECNOLOGIAS WEB SERVICES.....	114
FIGURA 3.5 – FRAMEWORK OPENGIS WEB SERVICES	117
FIGURA 3.6 – APLICAÇÕES E SERVIÇOS OPENGIS	118
FIGURA 3.7 – DIAGRAMA DE PACOTES EM UML. OS PACOTES REPRESENTAM DE FORMA ABSTRATA OS ESQUEMAS BASE DA GML	122
FIGURA 3.8 - REPRESENTAÇÃO EM UML DO ESQUEMA FEATURE	123
FIGURA 3.9 – DOMÍNIO POINT-TO-POINT.....	125
FIGURA 3.10 – DOMÍNIO PUBLISH/SUBSCRIBE	126
FIGURA 3.11 – ELEMENTOS DE UMA PÁGINA DE PORTAL	128
FIGURA 3.12 – FLUXO DE CRIAÇÃO DA PÁGINA DE PORTAL	129
FIGURA 3.13 – ARQUITETURA LÓGICA DO J2EE	132

FIGURA 3.14 – FACILIDADES DE INTEROPERABILIDADE DA PLATAFORMA J2EE	133
FIGURA 4.1 - RELACIONAMENTO DO MODELO DE REFERÊNCIA, ESTILO ARQUITETURAL E ARQUITETURA DE REFERÊNCIA, PARA DEFINIÇÃO DA ARQUITETURA DE UM SISTEMA DE INFORMAÇÃO PARA AP	138
FIGURA 4.2 – VISÃO GERAL DA ARQUITETURA DE REFERÊNCIA. O BARRAMENTO DE SERVIÇOS RECEBE AS REQUISIÇÕES REALIZADAS PELAS APLICAÇÕES E INVOCA OS SERVIÇOS APROPRIADOS, AGRÍCOLAS OU GEOESPACIAIS. DEPOIS QUE O PROCESSAMENTO É ENCERRADO ARMAZENA AS RESPOSTAS NO REPOSITÓRIO DE RESULTADOS E NOTIFICA O CLIENTE	138
FIGURA 4.3 – ARQUITETURA EM CAMADAS PROPOSTA PARA SISTEMAS DE INFORMAÇÃO PARA AP	139
FIGURA 4.4 – DIAGRAMA DE DEPENDÊNCIA DE PACOTES EM UML DOS ESQUEMAS PAML E GML.....	143
FIGURA 4.5 – DIAGRAMA DE CLASSES DO ESQUEMA <code>FIELDData</code> , UM FRAMEWORK DE APLICAÇÃO BASEADO NO FRAMEWORK GML	144
FIGURA 4.6 – ESQUEMA <code>FIELDData.xsd</code>	146
FIGURA 4.7 – A FEIÇÃO <code>YIELDSample</code> EM UM DOCUMENTO XML VÁLIDO, CONFORME ESQUEMA <code>FIELDData.xsd</code> (<code>YIELDSAMPLES.xml</code>).....	147
FIGURA 4.8 – CAMADAS DE UM ESB	148
FIGURA 4.9 - ARQUITETURA FORTEMENTE ACOPLADA	150
FIGURA 4.10 - ARQUITETURA FRACAMENTE ACOPLADA.....	150
FIGURA 4.11 – ARQUITETURA DO AGRIBUS	151
FIGURA 4.12 – COMPONENTES DO BARRAMENTO DE SERVIÇOS DO AGRIBUS.....	152
FIGURA 4.13 - <code>AGRIBUSCONFIGURATION.xml</code>	153
FIGURA 4.14 – ESTRUTURA GERADA A PARTIR DO <code>AGRIBUSCONFIGURATION.xml</code>	154
FIGURA 4.15 – DIAGRAMA DE COMPONENTES DO AGRIBUS.....	155
FIGURA 4.16 – ARQUITETURA DO PROVEDOR DE SERVIÇOS GEOESPACIAIS.....	161
FIGURA 4.17 – PÁGINA PRINCIPAL DO PAWSP	166
FIGURA 4.18 – PÁGINA DE PERSONALIZAÇÃO DO PORTAL	167
FIGURA 5.1 – VISÃO LÓGICA DA ARQUITETURA HÍBRIDA DO FILTERING COM WEB SERVICES E EJB.....	171
FIGURA 5.2 - WSDL DO SERVIÇO DE FILTRAGEM DE DADOS DE MONITORES DE PRODUTIVIDADE	172
FIGURA 5.3 – INTERFACE GRÁFICA DO FILTERING CONSTRUÍDA COM O FRAMEWORK STRUTS	173
FIGURA 5.4 – VISÃO GERAL DA ARQUITETURA DA APLICAÇÃO WEB “FILTERING”	175
FIGURA 5.5 – DIAGRAMA DE IMPLANTAÇÃO EM UML DA APLICAÇÃO WEB “FILTERING”	176

LISTA DE QUADROS

QUADRO 2.1 – CARACTERÍSTICAS GERAIS DOS SISTEMAS.	49
QUADRO 2.2 - CAPACIDADES DE MANIPULAÇÃO DE MAPAS.	50
QUADRO 2.3 – CAPACIDADES DE INTERCONEXÃO E INTEROPERABILIDADE.	51
QUADRO 2.4 - CAPACIDADES ANALÍTICA E SUPORTE À DECISÃO.	52

LISTA DE SIGLAS

AGLS	Australian Government Locator Service Metadata
AgMES	Agricultural Metadata Element Set
AGRIS	International Information System for the Agricultural Sciences and Technology
AIC	Agricultural Information Component
AP	Agricultura de Precisão
API	Application Programming Interface
ASP	Active Server Pages
B2B	Business-to-Business
B2C	Business-to-Consumer
B2E	Business-to-Enterprise
BLOB	Binary Large Object
CASE	Computer-Aided Software Engineering
CGI	Common Gateway Interface
CORBA	Common Object Request Broker Architecture
CSDGM	Content Standard for Digital Geospatial Metadata
DAAC	Danish Agricultural Advisory Centre
DCMES	Dublin Core Metadata Element Set
DCOM	Microsoft Distributed Component Object Model
DIAS	Danish Institute of Agricultural Sciences
DLIOs	Document-Like Information Objects
DLL	Dynamic-link library
DSS	Decision Support System
DTD	Document Type Definition
EAI	Enterprise Application Integration
EJB	Enterprise JavaBeans
ESB	Enterprise Service Bus
ESRI	Environmental Systems Research Institute
FAO	Food and Agricultural Organization of the United Nations
FGDC	Federal Geographic Data Committee
FMS	Farm Management Systems
FTP	File Transfer Protocol
GIS	Geographic Information System
GML	Geographic Markup Language
GNU	General Public Licence Model
GPS	Global Positioning System

HTTP	Hypertext Transfer Protocol
IACS	Integrated Administration and Controlling System
ISO	International Organization for Standardization
J2EE	Java 2 Enterprise Edition
J2SE	Java 2 Platform, Standard Edition
JCP	Java Community Process
JDBC	Java Database Connectivity
JMS	Java Message Service
JMX	Java Management Extension
JNI	Java Native Interface
JORAM	Java Open Reliable Asynchronous Messaging
JSP	Java ServerPages
JSR	Java Specification Requests
JVM	Java Virtual Machine
LAA	Laboratório de Automação Agrícola
LPIS	Land Parcel Information Systems
MOM	Message-Oriented Middleware
MOSAICo	Modelo de Objetos para Sistemas Abertos de Informação de Campo
NAICS	North American Industry Classification System
NRW	North Rhine-Westfalia
OASIS	Organization for the Advancement for Structured Information Standards
ODBC	Open DataBase Connectivity
ODBMS	Object Database Management System
OGC	OpenGIS Consortium
ORM	OGC Reference Model
PAML	Precision Agriculture Markup Language
PAWSP	Precision Agriculture Web Services Provider
PCMCIA	Personal Computer Memory Card International Association
PTP	Point-to-Point
RDF	Resource Description Framework
RMI	Remote Method Invocation
RSS	Really Simple Syndication
SAS	Statistical Analysis System
SDI	Spatial Data Infrastructures
SDSS	Spatial Decision Support System
SGBD	Sistema Gerenciador de Bancos de Dados
SIG	Sistemas de Informação Geográfica
SLD	Styled Layer Descriptors
SMIL	Synchronized Multimedia Integration Language

SML	Sensor Model Language
SMTP	Simple Mail Transfer Protocol
SOA	Services-Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SRE	Sistema de Referência Espacial
SVG	Scalable Vector Graphics
TCP/IP	Transmission Control Protocol/Internet Protocol
TI	Tecnologia da Informação
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
UNSPSC	Universal Standards Products and Services Classification
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTMS	Universal Mobile Telecommunications System
VIN	Vehicle Identification Number
VRA	Variable Rate Application
W3C	World Wide Web
WAICENT	World Agricultural Information Centre Portal
WAP	Wireless Application Protocol
WCAS	Web Catalog Service
WCS	Web Coverage Service
WCTS	Web Coordinate Transformation Service
WFS	Web Feature Service
WFS-G	Web Gazetter Service
WML	Wireless Markup Language
WMO	World Meteorological Organisation
WMS	Web Map Service
WSDL	Web Service Description Language
WTS	Web Terrain Service
XML	eXtensible Markup Language
XSLT	Extensible Stylesheet Language Transformation

SUMÁRIO

1	INTRODUÇÃO.....	17
1.1	MOTIVAÇÃO	17
1.2	OBJETIVO.....	22
1.3	ORGANIZAÇÃO DO TRABALHO	23
2	SISTEMAS DE INFORMAÇÃO PARA AGRICULTURA DE PRECISÃO.....	25
2.1	INTRODUÇÃO	25
2.2	EXPERIÊNCIAS E PROPOSTAS	26
2.2.1	<i>Pl@nteInfo</i>	27
2.2.2	<i>SORTINFO</i>	30
2.2.3	<i>Crop Models Analist</i>	31
2.2.4	<i>PCYield</i>	33
2.2.5	<i>Sistema de simulação para transporte e retenção de contaminantes no solo</i> ..	37
2.2.6	<i>VESPER - Variogram Estimation and Spatial Prediction plus Error</i>	38
2.2.7	<i>IRMDSS - Integrated Range Management Decision Support System</i>	42
2.3	ANÁLISE DO MERCADO	45
2.3.1	<i>Características gerais dos sistemas</i>	45
2.3.2	<i>Capacidades de manipulação de mapas</i>	46
2.3.3	<i>Capacidades de interconexão e interoperabilidade</i>	46
2.3.4	<i>Capacidades analíticas e suporte a decisão</i>	47
2.4	TRABALHOS RELACIONADOS À INFRA-ESTRUTURA DE DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO PARA AGRICULTURA DE PRECISÃO	54
2.4.1	<i>FMS - Farm Management Systems</i>	54
2.4.2	<i>Sistema de apoio a decisão para controle de pragas</i>	57
2.4.3	<i>TeleFarm</i>	59

2.4.4	<i>Experiências em transferência de dados entre sistemas para AP</i>	63
2.4.5	<i>Preagro-Profile</i>	65
2.4.6	<i>AgXML Standard</i>	67
2.4.7	<i>agroXML</i>	69
2.4.8	<i>AGRIS AP</i>	71
2.4.9	<i>XML schema para dados agrometeorológicos</i>	73
2.4.10	<i>Uma infra-estrutura para dados geoespaciais interoperáveis para AP</i>	75
2.4.11	<i>Geo-Services para agricultura</i>	79
2.4.12	<i>MOSAICo</i>	83
2.4.13	<i>Uma proposta para desenvolvimento de SI para AP com Web Services</i>	86
2.5	CONSIDERAÇÕES FINAIS.....	86

3 TECNOLOGIA DA INFORMAÇÃO: ALGUNS CONCEITOS, PADRÕES E TECNOLOGIAS.....88

3.1	INTRODUÇÃO	88
3.2	ARQUITETURA DE SOFTWARE.....	89
3.2.1	<i>Abstração</i>	90
3.2.2	<i>Elementos</i>	92
3.2.3	<i>Propriedades arquiteturais</i>	94
3.2.4	<i>Visões arquiteturais</i>	97
3.2.5	<i>Estilos arquiteturais</i>	98
3.2.6	<i>Arquiteturas de referência</i>	103
3.3	PADRÕES ABERTOS E TECNOLOGIAS RELACIONADAS	105
3.3.1	<i>W3C e OASIS</i>	105
3.3.2	<i>OpenGIS</i>	116
3.3.3	<i>JCP</i>	123

3.4	CONSIDERAÇÕES FINAIS	134
4	INFRA-ESTRUTURA DE DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO ORIENTADOS A SERVIÇOS DISTRIBUÍDOS PARA AGRICULTURA DE PRECISÃO	136
4.1	INTRODUÇÃO	136
4.2	A ARQUITETURA DE REFERÊNCIA PARA SISTEMAS DE INFORMAÇÃO PARA AP	137
4.3	A LINGUAGEM PADRÃO PARA TROCA DE DADOS ENTRE SERVIÇOS AGRÍCOLAS	141
4.4	O BARRAMENTO PARA CONEXÃO DE SERVIÇOS AGRÍCOLAS	148
4.5	O PROVEDOR DE SERVIÇOS GEOESPACIAIS	159
4.6	O PORTAL PARA SERVIÇOS AGRÍCOLAS	164
5	ESTUDO DE CASO: SERVIÇO DE FILTRAGEM DE DADOS “FILTERING”	168
5.1	INTRODUÇÃO	168
5.2	A METODOLOGIA PARA IDENTIFICAÇÃO, CARACTERIZAÇÃO E REMOÇÃO DE ERROS DE MAPAS DE PRODUTIVIDADE	169
5.3	A INFRA-ESTRUTURA INICIAL	170
5.4	O FILTERING	171
5.5	AS VERSÕES FINAIS DA INFRA-ESTRUTURA E DO FILTERING	175
5.6	AMBIENTES DE EXECUÇÃO DOS EXPERIMENTOS COM O PROTÓTIPO	177
6	CONSIDERAÇÕES FINAIS	179
6.1	CONCLUSÕES	179
6.2	PRINCIPAIS CONTRIBUIÇÕES	182
6.3	TRABALHOS FUTUROS	183
	REFERÊNCIAS	185

1 INTRODUÇÃO

1.1 Motivação

A Agricultura de Precisão (AP) tem como alicerce um conjunto de recursos que permitem fazer, em áreas extensas, o que os pequenos agricultores sempre fizeram, o tratamento das diferenças existentes dentro de um talhão, agregando o conhecimento acumulado pelas ciências agrárias. A idéia fundamental é possibilitar que o agricultor possa identificar as áreas de altas e baixas produtividades dos talhões e possa administrar essas diferenças com os mesmos critérios agronômicos já dominados, porém com maior grau de detalhamento.

O conceito de gerenciamento da AP é relativamente novo. Foi introduzido em meados dos anos 80 e, além de atrair grande interesse, iniciou uma grande revolução no gerenciamento de recursos (ROBERT, 1999). Desde então, recebeu diferentes nomes (*site-specific management*, *site-specific farming*, *precision farming* e *precision agriculture*), mas uma característica comum para todas essas denominações é a sua natureza dirigida a dados e uso intensivo de tecnologia.

A AP tem recebido grande atenção, e uma quantidade significativa de pesquisas têm sido feitas cobrindo vários aspectos, atividades e tecnologias, como visto em Plant (2001); Zhang et al. (2002) e Korduan, Bill e Bölling (2004), entre outros autores. Um grande número de produtos, sistemas e dispositivos têm sido lançados no mercado com a promessa de ajudar a alcançar os objetivos básicos da AP, aumentar a rentabilidade da produção, melhorar a qualidade do produto e proteger o ambiente.

Embora o potencial e benefícios teóricos da adoção da AP sejam evidentes e apesar dos avanços tecnológicos, alguns aspectos críticos como a justificativa econômica e ambiental, ainda não foram totalmente resolvidos. É difícil demonstrar o impacto econômico positivo da AP na fazenda (PEDERSEN et al., 2003), portanto, mesmo sabendo que resultados positivos provavelmente serão conseguidos, mais casos de sucesso são necessários. O impacto ambiental da AP também é difícil de ser avaliado (STAFFORD, 2000) e grandes esforços serão necessários para ativar seu potencial para o benefício do ambiente (AUERNHAMMER, 2001). Uma análise conduzida entre produtores na Dinamarca mostrou que, embora eles estejam otimistas sobre a AP, o maior problema é a dificuldade para verificar os ganhos econômicos e ambientais (PEDERSEN et al., 2003).

Uma das barreiras que precisam ser superadas antes que as tecnologias da AP possam ser largamente implementadas, é o excesso de dados para o gerenciamento da fazenda (ZHANG; WANG, M; WANG, N, 2002). Grandes quantidades de dados já podem ser adquiridas eletronicamente no campo. É possível também praticar a aplicação a taxas variáveis (VRA – *Variable Rate Application*) de calcário, fertilizantes, herbicidas e outros agro-químicos. Embora existam muitos pontos a serem melhorados na tecnologia eletrônica existente para aquisição de dados e VRA, essas duas tarefas não parecem ser os principais problemas no momento. Os principais problemas estão entre elas: interpretar a grande quantidade de dados coletados, compreender as causas da variabilidade e capacidade de propor estratégias para o gerenciamento da variabilidade para serem usadas em VRA.

O desafio continua sendo transformar os dados e informações em conhecimento útil aplicável para tomada de decisões (SØRENSEN et al., 2002). Sistemas de informação têm um papel vital no tratamento e manipulação de dados para tomada de decisões.

Existem muitos sistemas de software que têm sido usados pelos praticantes da AP como pode ser observado no capítulo 2 desse trabalho. Os sistemas disponíveis, na sua

maioria são monolíticos e possuem tarefas específicas como mapeamento de produtividade, amostras de solo e aplicação de fertilizantes. Muitos desses sistemas são fornecidos como pacotes de software pelos fabricantes de equipamentos como monitores de produtividade, VRA e coletores de amostras de solo, todos com funcionalidades limitadas. Como a prática da AP exige a integração dessas tarefas, o usuário tem que aprender a trabalhar com muitos pacotes de software, com diferentes interfaces de usuário e diferentes formatos de dados de entrada e saída, que podem ser incompatíveis ou podem necessitar de conversão externa.

Pacotes de software mais sofisticados também estão disponíveis. Esses pacotes incorporam muitas funcionalidades úteis para a AP como conectividade com banco de dados, funcionalidades GIS (*Geographic Information System*) para processamento, análise e visualização, conectividade com equipamentos de campo, funções geoestatísticas, entre outros.

No entanto, nesses pacotes de software ainda faltam requisitos importantes, identificados por muitos autores (ADRIAN; NORWOOD; MASK, 2005; BACKES; DÖRSCHLAG; PLÜMER, 2003; COX, 2002; FOUNTAS; BLACKMORE; PETERSEN, 2002; KORDUAN, 2003; KORDUAN; BILL; BÖLLING, 2004; LÜTTICKEN, 2000; NÖLLE, 2004; PEDERSEN et al., 2003; SARAIVA, 1998; SARAIVA; MASSOLA; CUGNASCA, 1998; SARAIVA; MASSOLA; PAZ, 1997; SØRENSEN et al., 2002):

- Sistemas de suporte a decisão e gerenciamento deveriam ser projetados para atender as necessidades específicas dos produtores;
- Os sistemas deveriam ter uma interface de usuário simples que permitisse customizações para diferentes perfis de usuário. Uma interface amigável é importante para todos, especialmente para usuários com pouca experiência com software;

- Métodos simples de usar e métodos automatizados para processamento são necessários. Os sistemas deveriam permitir a inclusão e programação de novos métodos de acordo com as regras definidas pelos usuários;
- O usuário deveria ter permissão para controle completo e ter acesso a parâmetros de processamento e funções de análise. Usuários experientes deveriam poder controlar e experimentar novas soluções;
- Uso de conhecimento especialista como conhecimento baseado em regras, deveria ser possível. Isso pode oferecer a oportunidade de refinar o sistema às condições locais e incluir os conhecimentos, práticas e preferências do usuário;
- Sistemas de software padronizados e integrados são necessários. Isso pode reduzir o investimento técnico, a curva de aprendizado e a necessidade de suporte técnico;
- Suporte à integração e à interoperabilidade com outros pacotes de softwares, locais ou remotos, via Internet usando padrões abertos. Essas características são fundamentais para integração com sistemas legados e sistemas distribuídos;
- Escalabilidade, para atender a diferentes necessidades;
- Suporte a metadados;
- Baixo custo.

Nas pesquisas realizadas neste trabalho, não foi encontrado nenhum sistema que preencha esses requisitos básicos. É improvável que um único sistema proprietário possa atendê-los por causa da complexidade e amplitude, o que reforça a tese de que o uso de componentes especialistas distribuídos e construídos sobre uma plataforma aberta, é a solução mais apropriada para a construção de sistemas de informação para AP.

Um fator importante que favorece a solução baseada em componentes distribuídos bem estruturados usando padrões abertos é a percepção de que a prática da AP ainda tem

muitas incertezas, que são ou serão objeto de pesquisa. Como consequência, novos métodos, componentes e técnicas de processamento, terão de ser incorporados aos sistemas num futuro próximo, depois de terem sido provados experimentalmente e disponibilizados para uso.

Embora a maioria dos sistemas considere os métodos da AP separados do gerenciamento da fazenda, com forte foco na variabilidade espacial, pode-se prever que estarão integrados em breve, numa visão mais holística da produção agrícola. Um conceito expandido da AP pode ser uma produção agrícola gerenciada de forma mais precisa, envolvendo não somente a planta, mas também a produção animal. Uma expansão ainda maior envolve a visão do agronegócio, que ainda é pouco explorado (KORDUAN; BILL; BÖLLING, 2004; LÜTTICKEN, 2000). Problemas como a rastreabilidade e segurança do alimento estão inerentemente ligadas à informação sobre o processo de produção. Os métodos e tecnologias da AP serão de grande importância para fornecer informações sobre cada produto até o mercado. O mercado, por sua vez, fornecerá parâmetros de volta para a fazenda realimentando a cadeia produtiva do agronegócio.

Essa expansão demandará outra escala de integração dos sistemas de informação para AP, que somente será econômica e tecnicamente possível através do uso de técnicas e conceitos computacionais modernos e de sistemas bem projetados, orientados por uma arquitetura capaz de fornecer propriedades como extensibilidade, reusabilidade e portabilidade.

A Arquitetura Orientada a Serviços (SOA) tem sido usada para criação de componentes de negócio extensíveis e reusáveis com alta capacidade de integração. Nessa arquitetura, componentes são disponibilizados como serviços, com contratos e interfaces bem definidas. Aplicações projetadas usando SOA podem ser incorporadas em uma aplicação composta mais facilmente que em uma aplicação monoliticamente projetada. Dois dos principais princípios dos componentes são facilmente obtidos com SOA, modularidade

(divisão de grandes tarefas em tarefas menores) e encapsulamento (uso de interfaces claramente definidas para esconder as partes internas de um componente).

A abordagem orientada a serviços pode trazer diversos benefícios para os sistemas de informação para AP, dada a necessidade de integração com diversas fontes de dados e aplicações de negócios diferentes. Os benefícios dessa abordagem incluem:

- Desenvolvimento de componentes de negócio simplificados;
- Facilidade de montar soluções de negócios construídos como redes de serviços;
- Facilidade de integração de aplicações heterogêneas;
- Flexibilidade e agilidade nas mudanças;
- Proteção dos componentes de negócio devido a mudanças na tecnologia.

Essas considerações e a possibilidade de continuidade dos trabalhos iniciados por Saraiva (1998, 2003) motivaram a proposta deste trabalho de pesquisa.

1.2 Objetivo

Baseado no contexto apresentado, o objetivo desse trabalho é propor uma infra-estrutura de desenvolvimento de sistemas de informação orientados a serviços distribuídos para agricultura de precisão.

O propósito dessa infra-estrutura é fornecer elementos básicos para apoiar e facilitar o desenvolvimento de sistemas de informação para agricultura de precisão mais abrangentes e flexíveis, capazes de acompanhar a rápida evolução da AP.

Para atender esse objetivo, a infra-estrutura proposta especifica uma arquitetura de referência para sistemas de informação para agricultura de precisão, uma linguagem padrão para troca de dados entre serviços agrícolas, um barramento para conexão de serviços agrícolas, um provedor de serviços geoespaciais e um portal para serviços agrícolas.

A especificação da infra-estrutura se apóia nos estudos realizados no domínio da arquitetura de software e nos padrões abertos para desenvolvimento de sistemas distribuídos orientados a serviços, no modelo de objetos para sistemas abertos de informação de campo para agricultura de precisão (MOSAICo) (Saraiva, 1998) e na proposta de Saraiva (2003) em utilizar *Web Services* para desenvolvimento e integração de sistemas.

Essa infra-estrutura apoiará os engenheiros de software no desenvolvimento de sistemas de informação orientados a serviços distribuídos utilizando padrões abertos para a Internet. As funcionalidades de diferentes domínios de negócio poderão ser construídas como serviços reutilizáveis e através da composição desses serviços poderão ser criados outros, que implementam processos de negócio completos.

Dessa forma, pretende-se que a infra-estrutura venha a ser uma base robusta para futuros desenvolvimentos de sistemas de informação que atendam os requisitos básicos e sejam extensíveis o suficiente para acompanhar a evolução da agricultura de precisão e da tecnologia da informação.

1.3 Organização do trabalho

O Capítulo 2 apresenta uma análise dos sistemas de informação para AP e áreas correlatas usados tanto em experimentos científicos quanto em aplicações de mercado. A análise procurou identificar problemas e boas práticas que pudessem influenciar nas decisões da proposta.

O Capítulo 3 apresenta alguns conceitos da arquitetura de software e da computação orientada a serviços e padrões abertos e tecnologias relacionadas a software livre, com o objetivo de subsidiar a escolha das tecnologias da informação para a infra-estrutura proposta.

O Capítulo 4 apresenta a especificação dos componentes da infra-estrutura proposta, que é constituída de:

- Uma arquitetura de referência para sistemas de informação para agricultura de precisão;
- Uma linguagem padrão para troca de dados entre serviços agrícolas;
- Um barramento para conexão de serviços agrícolas;
- Um provedor de serviços geoespaciais;
- Um portal para serviços agrícolas.

O Capítulo 5 apresenta o estudo de caso usado para validar as idéias deste trabalho. Os resultados dos experimentos com o protótipo para filtragem de dados de monitores de produtividade “Filtering” são apresentados e discutidos.

O Capítulo 6 apresenta as considerações finais: conclusões, contribuições e trabalhos futuros.

Finalmente, são apresentadas as referências bibliográficas utilizadas na elaboração deste trabalho.

2 SISTEMAS DE INFORMAÇÃO PARA AGRICULTURA DE PRECISÃO

2.1 Introdução

Os desafios e oportunidades da agricultura de precisão são, atualmente, bem mais amplos e se inserem nas questões globais relacionadas a sustentabilidade da produção e conservação ambiental: o correto uso da água para evitar a ameaça de escassez e a poluição dos mananciais; a manutenção da fertilidade do solo; o controle das pragas e doenças que afetam as plantas; a necessidade de alimentar um crescente contingente de pessoas; a maior rigidez dos padrões para a qualidade e segurança dos alimentos e produtos.

O tratamento dessas questões demanda muita informação e conhecimento. A informação é a chave para o sucesso de qualquer atividade e uma atividade pode ser aperfeiçoada cada vez mais se a informação sobre ela obedecer ao ciclo: obtenção de novas informações seguida da interpretação e utilização dessas informações para melhorar a atividade e gerar conhecimento.

Há várias iniciativas heterogêneas documentadas sobre manipulação e tratamento de dados coletados na AP. A maioria concentra-se em aspectos particulares do problema de tratamento dos dados. Grande parte dessas iniciativas constitui-se de sistemas experimentais, desenvolvidos como apoio à pesquisa.

Tem-se observado também um crescente empenho das empresas na tentativa de fornecerem sistemas de informação para gerenciamento e tratamento das informações geradas pela AP. No entanto, a visão imediatista do mercado tem gerado produtos que não atendem plenamente a demanda dos agricultores em termos de funcionalidades, custo e principalmente em termos de capacidade de extensão e integração.

Este capítulo é uma atualização do abrangente levantamento realizado por Saraiva (1998) em sua tese de doutoramento. O objetivo foi: identificar o caminho que os experimentos e o mercado tomaram desde o último levantamento feito por Saraiva; identificar as tecnologias utilizadas; identificar os principais problemas dos atuais sistemas de informação para agricultura de precisão e áreas correlatas; levantar requisitos para entender os problemas de negócio; identificar tipos de usuários; e qual o nível de utilização da Internet nas áreas rurais.

Este capítulo está organizado da seguinte forma: primeiro são apresentadas experiências e propostas de implementação de sistemas de informação para AP encontrados na literatura; em seguida é apresentada uma análise dos principais sistemas de informação para agricultura de precisão do mercado, que também é uma atualização de um levantamento feito por Sawyer, Bowen e McNeil (1999); e finalmente são apresentados alguns trabalhos mais abrangentes relacionados à infra-estrutura de desenvolvimento e integração de sistemas de informação para AP.

O objetivo destas atualizações foi entender as necessidades, identificar os principais problemas e aproveitar as boas práticas existentes para então, após definir as tecnologias mais apropriadas para solucionar ou minimizar os problemas encontrados, propor uma infra-estrutura capaz de apoiar o engenheiro de software no desenvolvimento desses tipos de sistemas.

2.2 Experiências e propostas

Esta seção apresenta um resumo das experiências e propostas de sistemas de informação para AP e áreas correlatas realizadas por diversos centros de pesquisa.

2.2.1 Pl@nteInfo

Pl@nteInfo (*Danish Web-Based Information and Decision Support System*) (JENSEN et al., 2000) é um sistema de suporte a decisão, que usa a *World Wide Web* para fornecer aos produtores e consultores agrícolas, informações e suporte a decisão para gerenciamento agrícola, no momento que precisarem. Os dados são coletados de diferentes fontes, processados por modelos de suporte a decisão e os resultados são integrados dentro de páginas personalizadas com gráficos, informações técnicas e *links* para informações adicionais.

Pl@nteInfo foi desenvolvido em colaboração entre o *Danish Institute of Agricultural Sciences* (DIAS) e o *Danish Agricultural Advisory Centre* (DAAC). Os objetivos para desenvolver o Pl@nteInfo incluem:

- Conduzir pesquisas com possibilidades de usar a Internet para consultas em tempo real e ilustrar os resultados dessas pesquisas através de exemplos práticos (principal objetivo);
- Coletar e visualizar o conhecimento agrícola formalizado que é usado para consultas práticas;
- Construir um sistema amigável para suporte a decisões para produtores e consultores.

O projeto arquitetural do sistema Pl@nteInfo envolve um servidor web principal do DIAS e um servidor de colaboração do DAAC. Os dados, modelos e informações do Pl@nteInfo são distribuídos entre esses dois servidores web. O servidor web principal DIAS contém um *framework* lógico para extrair e compilar informações das organizações participantes. A maioria da lógica da aplicação é executada no lado do servidor e somente documentos HTML estáticos com gráficos e códigos *JavaScript* são enviados aos clientes.

Os documentos HTML dinâmicos são montados em tempo real por programas SAS (*Statistical Analysis System*) e scripts CGI (*Common Gateway Interface*) em Perl no servidor principal DIAS. O código dos modelos de suporte à decisão e os procedimentos para apresentação gráfica das saídas dos modelos são escritos por programas SAS e armazenados em um servidor de aplicação SAS. Os servidores web se comunicam com o servidor de aplicações SAS usando o componente SAS/IntrNet (*SAS Institute Inc.*).

A maioria das informações dinâmicas é baseada em dados meteorológicos que são armazenados em bancos de dados, podendo ser dados derivados e originais. Esses bancos de dados são atualizados toda manhã por FTP (*File Transfer Protocol*) do *Danish Meteorological Institute* (DMI). O banco de dados contém uma seleção de parâmetros de diferentes estações climáticas com resolução de 1, 3 ou 24h, dependendo do parâmetro. Imagens meteorológicas são transferidas a cada 10 minutos, para animação e previsão de precipitação local.

A maioria das funcionalidades e informações do Pl@nteInfo é restrita aos usuários cadastrados, porém uma grande parte das informações no Pl@nteInfo é pública. A informação pública é obtida através de preenchimento de formulários e cliques em mapas, o que permite aos usuários especializarem a informação. As informações dos usuários previamente cadastradas são armazenadas no servidor Pl@nteInfo e incluem nome, endereço, coordenadas de localização geográfica, preferências, talhões, culturas, medidas de precipitações locais e irrigações aplicadas. O sistema tem três tipos de usuários: produtor, consultor e visitante. Em 1998, os valores anuais para subscrição de produtores, consultores e visitantes eram de aproximadamente 53, 212 e 0 Euros.

O modelo de decisão é implementado no servidor de aplicações SAS e usa dados climáticos do banco de dados meteorológico para calcular valores de respostas e um componente gráfico para apresentar esses valores graficamente em páginas web. As novas

preferências de serviços do subscritor são lidas do banco de dados de subscritores e usadas para extrair interpretações especializadas relevantes para o assunto do banco de dados de comentários especializados. Essas informações também são apresentadas em páginas web. A Figura 2.1 mostra o fluxo para criar páginas web em tempo real.

Figura 2.1 – Fluxo para criação de páginas web em tempo real

Fonte: (JENSEN et al., 2000)

Pl@nteInfo é um DSS (*Decision Support System*) baseado na web em contínuo desenvolvimento. A abordagem de informações distribuídas e a interface amigável tem tornado o Pl@nteInfo popular entre os produtores e consultores agrícolas. Segundo Jensen et al. (2000) no futuro os modelos de decisão do Pl@nteInfo serão distribuídos através das organizações.

Os mantenedores do Pl@nteInfo acreditam que essa é a abordagem certa para desenvolver modelos e ferramentas de suporte a decisão para sistemas agrícolas, e que a compreensão e a adoção de aplicações baseadas na web tornar-se-ão imperativos tanto no nível das propriedades quanto nas premissas científicas.

Apesar do Pl@nteInfo ser baseado na web, uma característica fundamental atualmente, o seu motor é uma implementação proprietária que se comunica com os clientes através de uma estrutura de dados definida pelas interfaces do componente SAS/IntrNet, o que limita a capacidade de integração com outras aplicações. Além disso, SAS é uma ferramenta

estatística de mercado com custo bastante elevado, o que pode limitar a adoção em massa do sistema.

2.2.2 SORTINFO

SORTINFO (*Danish Web-Based Information System*) (JENSEN, 2001) é um sistema de informação baseado na web, uma parte operacional do Pl@nteInfo (*Danish Web-Based Information and Decision Support System*). O objetivo principal do sistema é adicionar valor na informação de maneira que usuários sem nenhum conhecimento da base de dados especializada possam recuperar informações correspondentes aos requisitos desejados.

SORTINFO permite que usuários tomem melhores decisões sobre a seleção de variedades: por tornar o acesso aos dados de variedades de forma rápida e fácil; por processar e combinar dados de diferentes fontes e localidades para simples visualização; e por permitir que os usuários recuperem informações no detalhe que corresponde às suas necessidades e capacidades.

SORTINFO foi desenvolvido como uma aplicação em três camadas onde as tarefas da aplicação estão divididas entre três camadas distribuídas:

A camada de acesso a dados consiste do banco de dados. As tarefas desta camada são armazenar os dados em uma maneira organizada e estruturada e permitir a recuperação de dados especificados por múltiplos e simultâneos usuários;

A camada de lógica de negócio incorpora os métodos para recuperar dados do banco de dados, processar e aplicar regras de negócio;

A camada de apresentação é a interface do usuário. Esta camada permite ao usuário requisitar informações e através da interação com a camada de lógica de negócio apresentar os resultados da requisição ao usuário.

A camada de apresentação é isolada da camada de acesso a dados pela camada de lógica de negócio, desse modo ela é independente da estrutura de dados. Diferentes camadas de apresentação podem ser desenvolvidas com diferentes apresentações dos dados do banco de dados.

Cada camada do SORTINFO reside em um servidor localizado fisicamente em local diferente na rede: no *servidor de banco de dados* reside o sistema gerenciador de banco de dados *Microsoft SQL Server*; no *servidor web*, são hospedados os scripts escritos em ASP (*Active Server Pages*). Scripts ASP combinados com código HTML permitem a geração de páginas web dinâmicas, pois podem obter conexão com o banco de dados e recuperar dados através de requisições SQL; e no computador do usuário o *web browser* permite a entrada de parâmetros para diversos tipos de consulta, através do preenchimento de formulários ou cliques do mouse, e a exibição dos resultados das requisições que podem ser através de gráficos ou *download* de arquivos do tipo ASCII.

A abordagem do SORTINFO de organizar a aplicação em camadas é bastante interessante, no entanto essa arquitetura utilizada isola apenas a camada de acesso a dados da camada de lógica de negócio. A camada de lógica de negócio está fortemente acoplada à camada de apresentação, não permitindo o reuso das lógicas de negócio. Além da propriedade de reusabilidade, fica prejudicada a propriedade de manutenibilidade, pois os scripts ASP misturam lógica de negócio com lógica de apresentação.

2.2.3 Crop Models Analist

Este sistema reúne os modelos de simulação *CROPGRO-Soybean* e *CERES-Maize* ao *ArcView 3.1 GIS* e permite: (1) criar mapas para qualquer das 200 variáveis prognosticadas pelos modelos para cada dia da simulação; (2) interativamente executar o modelo do SIG

(Sistemas de Informação Geográfica) para testar hipóteses e fazer comparações com os dados medidos; e (3) avaliar prescrições sobre múltiplos anos de simulação (SEIDL et al., 2001).

A estrutura do sistema consiste de três componentes: modelos, base de dados e SIG e foi desenvolvido para usar modelos de crescimento para analisar causas da variabilidade no talhão e avaliar as prescrições. A Figura 2.2 mostra a organização deste sistema.

Figura 2.2- Estrutura do Crop Models Analyst

Fonte: (SEIDL et al., 2001)

O componente Crop Models Analyst é uma extensão do *ArcView 3.1* que visualmente mostra a informação do talhão, com dados medidos e entradas e saídas dos modelos. Neste módulo, o usuário pode gerar mapas de prescrição, avaliar cenários clicando nos talhões do mapa ou no menu e criar mapas de erros com os dados medidos e prognosticados. Todas as operações listadas no menu foram escritas na linguagem de programação *Avenue*, a linguagem de scripts do *ArcView*.

O componente central do sistema é o Crop Models Database, uma base de dados em *Microsoft Access'97*, que armazena e manipula os dados de entrada e saída dos modelos de simulação. Este componente cria arquivos de entrada, importa os arquivos de saída dos modelos e passa informações de volta para o Crop Model Analyst.

O Crop Growth Models corresponde aos dois modelos de simulação orientados a processo *CERES-Maize* e *CROPGRO-Soybean*. Esses modelos são escritos na linguagem de programação FORTRAN.

Os modelos Crop Growth Models fazem o prognóstico de crescimento de milho e soja baseado nos dados de gerenciamento do talhão como clima, solo, etc. O banco de dados, Crop Models Database, gerencia a maioria dos dados requeridos para a simulação incluindo os arquivos de entrada e saída dos modelos. O SIG Crop Models Analyst conecta ao banco de dados para a criação e visualização de mapas.

Este sistema é monolítico, baseado em soluções proprietárias e não disponibiliza interface web.

2.2.4 PCYield

PCYield é um sistema de suporte à decisão baseado no modelo para produção de soja *CROPGRO-Soybean* (WELCH et al., 2002). Esse sistema combina o modelo *CROPGRO-Soybean* com uma interface baseada em *Windows*. As principais características são: 1) gerenciamento localizado de dados, 2) acesso via Internet de dados pluviométricos, 3) indicadores de risco e 4) relatórios gráficos.

A partir das entradas: variedade (cultivar) e data de plantio, e considerando o clima e data esperada para colheita, o PCYield pode responder a duas perguntas específicas: Que rendimento pode-se esperar? O que acontece se não houver irrigação por um determinado período e o clima permanecer seco? As respostas a estas duas perguntas, se usadas

corretamente, podem ajudar: na seleção do cultivar; na determinação da data de plantio; nas decisões de replantio; na programação da irrigação; no *marketing*; e na estimativa dos efeitos do clima anormal, especialmente a seca. Além da informação atual do clima, PCYield requer uma entrada que pode ser obtida de uma simples caracterização do talhão.

A Figura 2.3 apresenta a estrutura do PCYield e o fluxo para análise de: 1) como os usuários executam análises de rendimento e de irrigação e 2) como os dados de entrada são controlados.

Figura 2.3 – Esquema da interface do PCYield e suas ligações com dados na Internet, modelo CROPGRO-Soybean, arquivos de dados e usuários

Fonte: (WELCH et al., 2002)

O *CROPGRO-Soybean* normalmente funciona dentro de DSSAT (*Decision Support System for Agrotechnology Transfer*). Desse modo, todas as entradas e saídas do modelo *CROPGRO-Soybean* são lidas ou escritas em arquivos texto do tipo ASCII, com formatos e

padrões internacionalmente aceitos. A interface do usuário é um programa escrito em *Microsoft Visual Basic*, que: 1) recebe entradas do usuário; 2) obtém dados do clima pela Internet; 3) gerencia todos os dados persistentes entre as sessões; 4) gera arquivos de dados formatados para o modelo a ser usado; 5) invoca o *CROPGRO-Soybean* de forma transparente para o usuário; e 6) converte os arquivos de saída do modelo DSSAT correspondente, para tabelas e gráficos.

O PCYield compara duas alternativas de cenários, por exemplo, “Normal...” e “Este ano...”. Essa estratégia foi adotada por duas razões. Primeiro, por solicitação dos produtores, pois normalmente eles comparam novas práticas ou situações, que pode acontecer no ano corrente, com aquelas que eles consideram normais. Segundo, porque *CROPGRO-Soybean* é melhor nas estimativas de produtividade relativas que nas absolutas.

Para as análises o PCYield gera dois arquivos: FILEX, com dados para gerenciamento da produção e WTH com dados do clima. O FILEX contém dados (cultivar, data de plantio, tipo de solo e histórico de irrigação) específicos dos dois cenários a serem comparados. O WTH varia com o tipo de análise. Para análises de produtividade do cenário “este ano...”, o WTH é gerado com dados coletados em tempo real até a data da projeção, dados projetados de sete dias a partir da data da análise e 10 conjuntos de dados históricos das estações climáticas. E para o cenário “normal...” o WTH é gerado com dados históricos de 10 estações climáticas completas.

Uma vez que os arquivos são gerados, o PCYield invoca o *CROPGRO-Soybean*, localizado numa máquina *Windows*. O modelo codificado em *Microsoft FORTRAN* para MS-DOS é executado em *background*. As saídas são simultâneas em modo caractere na tela e em um arquivo. PCYield usa uma barra de progressão para mostrar a execução ao usuário. Depois de terminada a execução, o PCYield lê os arquivos de saída e mostra os resultados em tabelas e gráficos.

O PCYield gerencia dois tipos de dados: descrições do talhão pelo usuário e dados obtidos em tempo real sobre informações do clima. Todos os dias, de meia-noite a meia-noite a WSI Corp, empresa privada parceira, coleta dados do clima de diversas fontes e os interpola numa grade de dois quilômetros quadrados. A WSI Corp, envia dados reais do dia anterior e previsões de sete dias para um computador da Universidade de Illinois. Esses dados são verificados, catalogados por estado, gravados em arquivos WTH e disponibilizados no *StratSoy*, um sítio específico sobre soja.

O PCYield já possui um conjunto de arquivos com esses dados e permite que sejam atualizados através da opção *Update Files*. Esse comando realiza uma interação cliente/servidor entre o PCYield e o *StratSoy*, através de um protocolo para sistemas abertos, permitindo a expansão de acesso a diversos tipos de clientes. Esse protocolo não faz nenhuma suposição sobre formatos de dados, e toda comunicação é realizada através de conexões no padrão *Winsock* (QUINN; SHUTE, 1995).

Os resultados deste trabalho trouxeram muitas contribuições. A experiência mostrou que a abordagem de aumentar gradativamente o número de funcionalidades e conseqüentemente a complexidade é fundamental para a aceitação do sistema pelos produtores. Simplicidade e modularização são características chave para um sistema de informação para AP. Mostrou também que a tendência é visualizar modelos como aplicações do lado do servidor. Uma vez que a execução centralizada de modelos sobre a web tem muitas vantagens em termos de suporte, manutenibilidade e fornece uma interface mais comum e conhecida dos usuários. No entanto, também está construído sobre plataformas de software proprietárias e não é baseado na web.

2.2.5 Sistema de simulação para transporte e retenção de contaminantes no solo

Este sistema é um simulador baseado na web para transporte e retenção de compostos inorgânicos e orgânicos dissolvidos no solo (ZENG et al., 2002). O sistema foi desenvolvido em Java e é baseado no modelo de transporte multi-reação de metais pesados no solo MRTM. O núcleo do sistema foi escrito em C e FORTRAN com o objetivo de conseguir melhor desempenho. Para que essas linguagens interagissem com Java foi usada a API JNI (*Java Native Interface*). A interface do usuário foi construída com Swing, uma API para desenvolvimento de interfaces gráficas em Java. A Figura 2.4 mostra a arquitetura do sistema de simulação, que é dividida basicamente em 3 partes: Pacote de computação MRTM, Pacote de visualização gráfica e Applets Swing.

Figura 2.4 - Arquitetura do sistema de simulação baseado na web para transporte e retenção de contaminantes no solo

Fonte: (ZENG et al., 2002)

O Pacote de computação MRTM está escrito em FORTRAN e lida com todos os cálculos de reação. O Pacote de visualização gráfica ajuda a desenhar as curvas bi-dimensionais em *applets* escritos em Java. Os Applets Swing controlam todo o

sistema, recebem parâmetros do Interface do usuário, invocam o pacote computacional e executa o pacote gráfico para animar as reações e desenhar as curvas.

O código legado em FORTRAN executa cálculos complexos com matrizes, integrais, e outros cálculos numéricos. Este código foi mantido intacto em vez de ser portado para Java por questões de desempenho. FORTRAN ainda é mais eficiente que Java e o custo de reescrever em Java seria muito alto, uma vez que existem milhares de linhas de código já testadas em FORTRAN.

Para permitir que código Java chame rotinas em FORTRAN, foram construídas rotinas em C/C++ para encapsular o código FORTRAN, através de bibliotecas compartilhadas ou dinâmicas que estendem a interface C-FORTRAN da *Sun Microsystem Inc.* As rotinas em C/C++ são acessíveis por Java através da API JNI, que é uma boa solução para reuso de código legado.

Um dos principais problemas dessa solução foi a restrição de segurança implícita dos *Applets*. Para que o *Applet* acesse os códigos *Java Native Interface* (JNI) foi necessário realizar configurações de segurança na máquina cliente, o que não é desejável. O ideal é que essas configurações fiquem escondidas do usuário, ou seja, as configurações devem ser feitas no lado do servidor.

2.2.6 VESPER - Variogram Estimation and Spatial Prediction plus ERror

VESPER é um software para a plataforma *PC-Windows* desenvolvido pelo *Australian Centre for Precision Agriculture* (ACPA) para predição espacial geoestatística capaz de executar “krigagem” com variogramas locais (WHELAN; MCBRATNEY; MINASNY, 2001, 2002). O uso do VESPER inclui geração de mapas de produtividade, interpolação de modelos de elevação digital e avaliação de problemas de salinidade do solo. O software também

permite krigagem convencional, com variograma inteiro da área com opções para ajustes manuais, criação dos limites do talhão e grade de interpolação.

VESPER é um software *shareware*¹, escrito para fornecer técnicas de predição espacial para a indústria da agricultura de precisão. Ele oferece uma variedade de opções para lidar com conjuntos de dados variando densidade de dados, distribuição espacial, e observação de incertezas. Tais conjuntos de dados são obtidos de vários sensores de tempo real, produtividade, solo e cultura, e também através de amostragens manuais.

Arquivos de entrada e saída são selecionados através de uma interface gráfica. Para realizar a análise espacial é necessário que os dados dos arquivos de entrada estejam associados a localizações com coordenadas Cartesianas. Os arquivos de saída registram os detalhes das configurações da sessão específica, parâmetros do modelo do variograma, localizações da predição, valores e variância da predição. A Figura 2.5 mostra a interface gráfica onde o arquivo de entrada é selecionado e os nomes dos arquivos de saída são especificados.

¹ *Shareware* é um método de propaganda de software, através da distribuição de uma versão para experimentação sem custo, muito comum para software proprietário.

Figura 2.5 – Interface gráfica do VESPER
Fonte: (VESPER, 2006)

Para executar o VESPER os seguintes parâmetros devem ser especificados:

- Arquivos de entrada e saída;
- Grade de interpolação;
- Parâmetros de krigagem;
- Parâmetros do variograma

O arquivo de entrada deve ter o formato texto com múltiplas colunas com ou sem cabeçalho separados por delimitadores (tabulação, espaço ou vírgula). Por exemplo:

```
x, y, z
50.16571, 38.60503, 6.73180134
```


51.81264,38.41478,5.38031385

53.48674,37.77423,4.83549547

As colunas X e Y correspondem a *Easting & Northing* (coordenadas com orientação leste e norte) e a coluna Z corresponde ao valor do dado. A grade de interpolação pode ser gerada ou informada através de arquivo externo. Os parâmetros de krigagem e os parâmetros do variograma são informados pelo usuário através da seleção de modelos pré-definidos já existentes na ferramenta.

Depois do processamento o VESPER mostra graficamente o mapa da interpolação e o mapa de incertezas da predição (erro padrão) (Figura 2.6).

Figura 2.6 – Resultado do processamento do VESPER
Fonte: (VESPER, 2006)

O arquivo de saída gerado pelo VESPER tem formato texto ASCII, que consiste de cinco colunas contendo: número de ordem da grade, coordenada X, coordenada Y, valor predito e derivação padrão do valor predito.

O VESPER possui uma ferramenta para conversão desse arquivo em outras formas de arquivos texto ou grade ASCII para permitir integração com outros sistemas. O arquivo pode ser convertido para o padrão ASCII com delimitadores (tabulação, espaço ou vírgula) ou em um arquivo *ASCII raster grid*. O formato *ASCII raster grid* pode ser importado pelos softwares Surfer e *ArcView GIS*.

2.2.7 IRMDSS - Integrated Range Management Decision Support System

O IRMDSS (SUGUMARAN, 2002) é um sistema integrado para gerenciamento e planejamento sustentável de recursos florestais. Gerenciamento de recursos florestais, assim como a AP, requer a integração de diferentes fontes de informações com grandes volumes de dados. Esse sistema é uma ferramenta que fornece suporte a decisão para planejadores de florestas ocidentais da Índia, a qual, ecologicamente, é uma importante área da Terra.

O IRMDSS combina sensoriamento remoto, SIG, sistemas baseados em conhecimento e modelos analíticos, oferecendo aos usuários uma ferramenta interativa e flexível para suporte a decisões.

O IRMDSS foi desenvolvido pela customização do *ArcView GIS* que foi acoplado em outros softwares usando bibliotecas de link dinâmico (DLL). O IRMDSS fornece aos planejadores e gerentes uma forma de organizar, acessar e avaliar melhor grandes quantidades de informações e estratégias alternativas para o efetivo gerenciamento e planejamento das áreas florestais.

O *ArcView GIS* foi usado como a plataforma base para o projeto e desenvolvimento do IRMDSS porque abrange a maioria dos requisitos para o desenvolvimento de um *Spatial*

Decision Support System (SDSS), tais como: um sistema de gerenciamento de banco de dados; processamento de imagem com o módulo *Image analysis*; simples modelagem através do *Spatial analyst*; desenvolvimento de interfaces gráficas do usuário com o módulo *Dialog designer*, gerador de relatórios com o módulo *Report writer*. Abrange também porque oferece diferentes meios para a comunicação com outras aplicações. Entretanto, faltam as capacidades de manipulação de bases de conhecimento e modelagem. Para criar o SDSS foram acoplados ao *ArcView GIS*, o *MS-Access*, o *Sictus PROLOG* e capacidades de modelagem.

Para conectar os diferentes componentes foram usados vários métodos. A conexão do *ArcView* com *MS-Access* foi realizada através do *Open DataBase Connectivity* (ODBC), que fornece uma maneira padrão para definição e acesso a fontes de dados. Os modelos e interfaces gráficas do usuário foram desenvolvidos usando o módulo *Dialog Designer*, a linguagem de programação *Avenue* e o módulo *Spatial analyst*. A conexão entre o *ArcView* e o sistema baseado em conhecimento (*Sictus PROLOG*), foi obtido através de *Dynamic-link Library* (DLL) criada com a linguagem de programação C++. E baseado nos requisitos dos usuários, o *ArcView GIS* foi customizado e os quatro principais módulos, *Data Manager*, *Protection Manager* e *Display Manager* (*Mapmaker* e *Report generator*) foram adicionados à janela principal do *ArcView* (Figura 2.7).

Figura 2.7 – Interface do IRMDSS

Fonte: (SUGUMARAN, 2002)

O IRMDSS fornece uma interface gráfica amigável para o usuário para manipulação de informações e conhecimentos técnicos complexos para tomada de decisão. Permite organizar informações baseadas em conhecimento específico e dados existentes, projetar alternativas, avaliar consequências de novos planos de gerenciamento ou políticas para avaliar e comparar com esquemas de gerenciamento alternativos. No entanto, o sistema tem algumas limitações, por exemplo: suporta poucos formatos de dados, é monolítico, a integração com tecnologias diferentes das existentes no *ArcView* devem ter implementações específicas, possui apenas modelos estáticos simples e não permite que o usuário mude as variáveis. Os modelos deveriam ser genéricos porque as áreas florestais mudam rapidamente através de fatores muito dinâmicos, como na AP.

2.3 Análise do mercado

Atualmente há no mercado sistemas que reúnem várias funcionalidades requeridas pela AP. A seguir é apresentada uma análise dos mais conhecidos e utilizados sistemas de informação para AP do mercado. Os sistemas são comparados por características como: capacidade de manipulação de mapas, interoperabilidade, capacidades analíticas e suporte a decisão. Essas informações foram baseadas no relatório técnico *Precision Agriculture Software for Physical Record-keeping and Field Mapping* elaborado por Sawyer, Bowen e McNeil (1999), e nas pesquisas realizadas na Internet com o objetivo de atualizar o relatório acrescentando novos sistemas e novas características.

Os quadros de 2.1 a 2.4 apresentam os resultados das análises dos sistemas. Respectivamente, correspondem a: características gerais; capacidade de manipulação de mapas; interconexão e interoperabilidade; e capacidades analíticas e suporte à decisão.

2.3.1 Características gerais dos sistemas

O Quadro 2.1 apresenta as características gerais através de uma breve descrição dos sistemas e o enquadramento em três categorias:

- *Introdutório*: para usuários com experiência em sistemas computacionais limitados, que desejam manter dados (espaciais ou não espaciais) para suporte ao gerenciamento de dados da AP;
- *Avançado*: para usuários com experiência em sistemas computacionais, ou para aqueles que querem recursos avançados como suporte a dados de GPS (*Global Positioning System*), importação de dados geo-referenciados como solo e imagem ou algumas funcionalidades de SIG como capacidade de manipulação de multicamadas;

- *Professional*: para usuários com experiência em sistemas computacionais, que requerem completa capacidade de SIGs, incluindo análise espacial, imagem, métodos de interpolação e superfície 3D.

2.3.2 Capacidades de manipulação de mapas

O Quadro 2.2 apresenta as capacidades de manipulação de mapas:

- *Mapa ligado a banco de dados*: capacidade de referenciar dados de polígonos ou pontos no banco de dados;
- *Capacidades de Sistemas de Informações Geográficas*: capacidade de capturar, armazenar, atualizar, analisar e mostrar informações referenciadas geograficamente (pontos, linhas e polígonos, múltiplas camadas de dados e variáveis espacialmente referenciadas). Topologia, capacidade de reconhecer adjacências de características espaciais;
- *Importação e exportação de dados*: Tipos de formatos aceitos, incluindo ASCII, DBF, DXF, TIF, BMP, *ArcView shape* e arquivos somente para leitura;
- *Suporte a web*: capacidade de manipular dados de campo e mapas na web.

2.3.3 Capacidades de interconexão e interoperabilidade

O Quadro 2.3 apresenta as capacidades de interconexão e interoperabilidade entre sistemas:

- *Sensoriamento Remoto*: Alguma capacidade de sensoriamento remoto: importar imagens geo-referenciadas; capacidade de manipular e processar imagens e mostrar imagens incluindo, infravermelho, radar e multi-espectral e capacidade de fornecer coordenada a uma imagem.

- *Monitores de produtividade/GPS*: capacidade do sistema de ser independente de formatos de dados de monitores de produtividade específicos ou de receptores GPS.
- *Gerenciamento de dados de registro*: capacidades de manter dados de propriedades, produção, máquinas e produtos; gerar relatórios, gráficos; e capacidade de importar e exportar dados do tipo ASCII, DBF, etc.

2.3.4 Capacidades analíticas e suporte a decisão

O Quadro 2.4 apresenta as capacidades analíticas e suporte a decisão:

- *Análises Estatísticas*: capacidade de realizar consultas em dados espaciais; funções matemáticas como desvio padrão, máximo e mínimo, etc;
- *Métodos de Interpolação*: fórmulas matemáticas para predizer valores desconhecidos usando valores conhecidos de localizações vizinhas (kriging, inverso da distância, etc.) para produzir mapas de superfície contínuos.
- *Análises Econômicas*: capacidade de fornecer análises de custo/benefício;
- *Cálculos de Misturas de Fertilizantes*: capacidade de produzir cálculos de misturas de fertilizantes;
- *Mapas de Aplicação*: capacidade de produzir mapas a partir de dados analisados para aplicação em campo, incluindo fertilização a taxas variáveis e capacidade de incluir fórmulas definidas pelo usuário.

Nome da Empresa/ Instituição	Nome do Software	Descrição do Software	Open Source /Custo
Farm Works Software http://www.farmworks.com	Farm Trac	<i>Introdutório</i> , mantém registros e mapas da propriedade. Integra com outros softwares da Farm Works para aumentar a sua capacidade.	Não/ U\$300.00
Farm Works Software http://www.farmworks.com	Quick Yields	<i>Introdutório</i> , software monolítico para visualizar e imprimir mapas com dados de campo.	Não/Gratuito
SIGA Farm Software http://www.siga.net	SigaField for Windows	<i>Introdutório</i> , software para gerenciamento de talhões com módulo para análise de custo/benefício, geração de mapas de recomendações de fertilizantes e registro de informações de campo.	Não/ U\$395.00
BCL LandView Systems Inc. http://www.landview.com	WinCrop	<i>Introdutório</i> , mantém registros das produções. O WinCrop é utilizado para gerenciar propriedades e talhões.	Não/ U\$249.00
BCL LandView Systems Inc. http://www.landview.com	FarmView Record Keeper	<i>Introdutório</i> , mantém registros de dados de produção e planejamento. É um sistema de informações para gerenciamento da propriedade. Pode ser integrado ao Land View Mapper Lite.	Não/ U\$599.00
John Deere http://www.deere.com	JDMap	<i>Introdutório</i> , software de mapeamento de campo usado com o sistema GreenStar. Fornece funcionalidades básicas de mapeamento de talhões.	Não/ U\$560.00
Farm Works Software http://www.farmworks.com	Farm Site	<i>Avançado</i> , software para mapear a propriedade que inclui as ferramentas Quick Yield e FarmSite Mobile. Requer o módulo FarmTrac.	Não/ U\$450.00
Farm Works Software http://www.farmworks.com	Farm Site Pro	<i>Avançado</i> , software para registro e mapeamento de talhões usando coordenadas reais. Possui funções multi-layers e é multi-usuário. Inclui as ferramentas Quick Yield e FarmSite Mobile.	Não/ U\$1,500.00
BCL LandView Systems Inc. http://www.landview.com	LandView Mapper Pro	<i>Avançado</i> , pacote para geração de mapas multi-usuário, multi-layers e GPS. Pode ser integrado ao LandView DSS Pro.	Não/ U\$899.00
BCL LandView Systems Inc. http://www.landview.com	LandView DSS Pro	<i>Avançado</i> , mantém registros de propriedade, é multi-usuário e faz análise de mapas. Pode ser integrado ao LandView Mapper Pro.	Não/ U\$1,999.00
SST Development Group, Inc. http://www.sstdevgroup.com	SSToolkit	<i>Avançado</i> , sistema para gerenciamento de dados espaciais agrícolas na plataforma ARCVIEW GIS. Módulos adicionais podem ser adicionados.	Não/ U\$3,000.00

SST Development Group, Inc. http://www.sstdevgroup.com	SSToolbox	<i>Professional</i> , sistema para gerenciamento de dados espaciais agrícola, na plataforma ARCVIEW GIS e Surfer, para usuários avançados.	Não/US\$9,000.00
Linnet Geomatics http://www.linnet.ca	Croplands-The System	<i>Professional</i> , software com recursos de SIG que cobre grande parte das operações de gerenciamento dos serviços envolvidos na produção agrícola.	Não/US\$22,500.00
ESRI http://www.esri.com	ARCVIEW GIS	<i>GIS Professional</i> , software usado em várias áreas para mapeamento e sistemas de informações geográficas. É uma ferramenta para mostrar, consultar e analisar informações espaciais. <i>Não possui customização para a agricultura.</i>	Não/US\$1,195.00
MapInfo Corporation http://www.mapinfo.com	MapInfo Professional	<i>GIS Professional</i> , software usado em várias áreas para mapeamento e sistemas de informações geográficas. MapInfo Corporation fornece módulos adicionais para necessidades específicas de mapeamento. <i>Não possui customização para a agricultura.</i>	Não/US\$1,950.00

Quadro 2.1 – Características gerais dos sistemas.

Nome da Instituição	Nome do Software	Mapa Ligado a Banco de Dados	SIG	Importação/ Exportação de Dados	Suporte à Web
Farm Works Software http://www.farmworks.com	Farm Site	sim	sim	ARCVIEW shape files, ASCII, DLG, BMP, TIFF, PCX, GIF, JPEG	Não
Farm Works Software http://www.farmworks.com	Farm Site Pro	sim	sim	ARCVIEW shape files, ASCII, DLG, BMP, TIFF, PCX, GIF, JPEG	Não
BCL LandView Systems Inc. http://www.landview.com	LandView Mapper Pro	Requer o LandView Farm Setup Utility	sim	BMP, DXF	Não

BCL LandView Systems Inc. http://www.landview.com	LandView DSS Pro	Requer LandView Farm Setup Utility	sim	BMP, DXF	Fornecer serviço de mapeamento e gerenciamento da produção através do LANDVIEW ON-LINE™
SST Development Group, Inc. http://www.sstdevgroup.com	SSToolkit	sim	sim	DXF, MapInfo formats, AgNavigator (.fif, .pts, .pdt), FieldLink, Atlas, SGIS, AgView (muitos outros formatos)	Não
SST Development Group, Inc. http://www.sstdevgroup.com	SSToolbox	sim	sim	DXF, MapInfo formats, AgNavigator (.fif, .pts, .pdt), FieldLink, Atlas (.bna), SGIS, AgView (muitos outros formatos)	Não
Linnet Geomatics http://www.linnet.ca	Croplands-The System	sim	sim	A maioria dos formatos	Não
ESRI http://www.esri.com	ARCVIEW GIS	sim	sim	DXF, DWG, DGN, MIF, E00, BMP, TIFF, shape, WMP, JPEG, EPS	Internet Mapping: Internet Mapping Server (IMS), ArcIMS e ArcWeb Services, o pacote de SIG Web Services.
MapInfo Corporation http://www.mapinfo.com	MapInfo Professional	sim	sim	DXF, MIF, GIF, TIFF, BMP, ASCII, DBF, XLS, MDB	Fornecer o MapXtreme® for Windows e o MapXtreme® Java Edition, permite criar aplicações servidoras que permitem manipular mapas através de browsers.

Quadro 2.2 - Capacidades de manipulação de mapas.

Nome da Instituição	Nome do Software	Sensor. Remoto / Imagens	Monitores de Produtividade / GPS	Ger. de Dados de Registro
Farm Works Software http://www.farmworks.com	Farm Site	Sim	AgLeader (all formats), MicroTrak (all formats), RDS, Springhill Eng. Ag Navigator, JDMMap (yield)	Não
Farm Works Software http://www.farmworks.com	Farm Site Pro	Sim	AgLeader (all formats), MicroTrak (all formats), RDS, Springhill Eng. Ag Navigator, JDMMap (yield)	Sim
BCL LandView Systems Inc. http://www.landview.com	LandView Mapper Pro	Sim	Não	Não
BCL LandView Systems Inc. http://www.landview.com	LandView Pro	Sim	AgLeader, MicroTrak, JD Greenstar, AGCO and ASCII	Sim
SST Development Group, Inc. http://www.sstdevgroup.com	SSToolkit	Sim	Ag Leader Advanced (.txt), JDMMap (.txt), MicroTrak (.txt, .mty, .dbf), Ag Navigator (.yif), HarvestMaster (.txt)	Requer configurar tabelas do banco de dados
SST Development Group, Inc. http://www.sstdevgroup.com	SSToolbox	Sim	Ag Leader Advanced (.txt), JDMMap (.txt), MicroTrak (.txt, .mty, .dbf), Ag Navigator (.yif), HarvestMaster (.txt) RDS, AGCO (txt)	Requer configurar tabelas do banco de dados
Linnet Geomatics http://www.linnet.ca	Croplands-The System	Sim	Qualquer monitor	Sim
ESRI http://www.esri.com	ARCVIEW GIS	Sim	ASCII	Requer configurar tabelas do banco de dados
MapInfo Corporation http://www.mapinfo.com	MapInfo Professional	Sim	ASCII	Requer configurar tabelas do banco de dados

Quadro 2.3 – Capacidades de interconexão e interoperabilidade.

Nome da Instituição	Empresa/	Nome do Software	Análises Estatísticas	Métodos de Interpolação	Análises Econômicas	Cálculos de Misturas de Fertilizantes	Mapas de aplicação
Farm Works Software http://www.farmworks.com		Farm Site	Sim	Sim	Não	Não	Sim
Farm Works Software http://www.farmworks.com		Farm Site Pro	Sim	Sim	Não	Não	Sim
BCL LandView Systems Inc. http://www.landview.com		LandView Mapper Pro	Sim	Não	Não	Não	Não
BCL LandView Systems Inc. http://www.landview.com		LandView DSS Pro	Sim	Não	Sim	Sim	Sim
SST Development Group, Inc. http://www.sstdevgroup.com		SSToolkit	Sim	Sim	Sim	Não	Sim
SST Development Group, Inc. http://www.sstdevgroup.com		SSToolbox	Sim	Sim	Sim	Sim	Sim
Linnet Geomatics http://www.linnet.ca		Croplands-The System	Sim	Sim	Sim	Sim	Sim
ESRI http://www.esri.com		ARCVIEW GIS	Sim	Não	Requer escrever código	Requer escrever código	Requer escrever código
MapInfo Corporation http://www.mapinfo.com		MapInfo Professional	Sim	Não	Não	Não	Não

Quadro 2.4 - Capacidades analítica e suporte à decisão.

Dessa análise pôde-se concluir que os mais completos sistemas de informação para AP comerciais são: *Croplands-The System* e *SSToolbox*, nesta ordem. São também os de custo mais elevado, pois são pacotes de uso profissional com recursos de GIS e com capacidades de gerenciar a maioria das atividades de uma propriedade. Esses sistemas são bastante complexos e necessitam de pessoas especializadas para manipulá-los. Realizam integração com pacotes externos através importação e exportação de dados, o que exige atualização do sistema sempre que um novo formato surge. A maioria dos sistemas analisados são proprietários e não permitem distribuição gratuita. Poucos oferecem recursos web, o *LandView DSS Pro* é o que mais se preocupou em fornecer esse tipo de recurso. Ambos os softwares para SIG analisados são bastante completos, inclusive com suporte para acesso e manipulação de mapas através da Internet, porém, não são customizados para AP.

As características: facilidade de integração de sistemas, suporte para web e baixo custo, são requisitos fundamentais para esses tipos de sistema. Nos últimos anos o software livre tem sido uma alternativa econômica e financeiramente viável ao modelo atual de licenciamento de software, além de ser uma boa forma de receber colaboração sem custo. O governo de países como China, Índia e Brasil e alguns blocos econômicos estão sinalizando para o mundo que pretendem adotar em suas políticas de Tecnologia da Informação (TI) soluções baseadas em software livre e que pretendem fomentar suas indústrias de software para desenvolverem sobre plataformas livres.

Uma das propostas deste trabalho é fomentar soluções para AP baseadas em padrões abertos e plataformas de software livre, como poderá ser observado nos próximos capítulos.

2.4 Trabalhos relacionados à infra-estrutura de desenvolvimento de sistemas de informação para agricultura de precisão

Esta seção apresenta algumas iniciativas de desenvolvimento nas áreas: ambientes e plataformas de software para desenvolvimento de sistemas de informação para AP e áreas correlatas; soluções para integração de aplicações usando padrões abertos; e infra-estruturas para dados espaciais sobre plataformas de software livre. A maioria dessas iniciativas está fortemente relacionada à proposta desta tese.

2.4.1 FMS - *Farm Management Systems*

O FMS (MASSRUHÁ; MEIRA; FILETO, 1998) é um ambiente de desenvolvimento de sistemas para gerenciamento de fazendas. Segundo Massruhá, Meira e Fileto (1998) a experiência adquirida no desenvolvimento de aplicações usando este ambiente, permitiu a criação de um *framework* para apoiar o processo de desenvolvimento de sistemas para gerenciamento de fazendas, das atividades de especificação até a codificação. O *framework* aplica a metodologia e as ferramentas do ambiente de desenvolvimento FMS combinado com ferramentas *Computer-Aided Software Engineering* (CASE), técnicas de programação visual e orientação a objetos.

O processo de desenvolvimento de software suportado pelo ambiente FMS, que inclui um analista de sistemas e um especialista do negócio, consiste do uso de um Assistente de Especificação (AEsp), que gera uma especificação na Linguagem de Composição de Aplicações FMS (LC-FMS) e um gerador de código fonte (GFMS), que gera o código da aplicação na linguagem “C” a partir das especificações LC-FMS, o qual é compilado e transformado em um programa executável. A Figura 2.8 apresenta a abordagem de desenvolvimento proposto para o FMS e a Figura 2.9 mostra o processo de transformação.

Figura 2.8 – Abordagem do FMS

Fonte: (MASSRUHÁ, 1997)

Figura 2.9 – Processo de transformação dos requisitos

Fonte: (MASSRUHÁ, 1997)

O LC-FMS é composto por várias linguagens: formulário (LC-FORM), ajuda (LC-HELP), consistência (LC-CONSIST), aplicação (LC-APPLY), eventos (LC-EVENT), ações (LC-ACTION) e relatórios (LC-REPORT). Todas são textuais e cada uma tem sua própria sintaxe e semântica. A LC-EVENT é a linguagem que torna possível especificar os mecanismos de eventos, a principal característica do FMS.

O FMS integra ferramentas como gerador de interface gráfica, sistemas gerenciadores de banco de dados, linguagens de consulta e geradores de relatórios, além de ferramentas CASE para ajudar no projeto e especificação, que fornece métodos apropriados para documentação e desenvolvimento. As ferramentas suportadas pelo ambiente FMS na geração de código incluem: o *Delphi* da *Borland* e o *Visual C++* da *Microsoft*. Uma ferramenta para auxiliar a programação dos eventos foi construída em *Delphi* e nomeada de *FMS Form Expert*. Essa ferramenta é um recurso de programação visual adicionado ao LC-EVENT.

Alguns sistemas foram desenvolvidos usando o *framework* para validar o ambiente FMS e o processo proposto por Massruhá, Meira e Fileto (1998), como o sistema de gerenciamento de gado leiteiro – *Lactus* (Figura 2.10), disponível gratuitamente na rede de software livre para agropecuária do Ministério da Agricultura, Pecuária e Abastecimento (<http://www.agrolivre.gov.br/>), o sistema de gerenciamento de produção vegetal e o sistema de gerenciamento de gado de corte.

A desvantagem desse ambiente é que as soluções geradas são monolíticas e baseadas em soluções proprietárias e desse modo, as aplicações geradas têm problemas de interoperabilidade e portabilidade, propriedades fundamentais para os sistemas atuais.

Figura 2.10 – Interface do Lactus v1.01

2.4.2 Sistema de apoio a decisão para controle de pragas

Beck e Xin (1998) propõem uma arquitetura em três camadas para desenvolvimento de aplicações na Internet (Figura 2.11). A camada principal é composta por um sistema gerenciador de banco de dados orientados a objetos (ODBMS) que reside em um servidor remoto na Internet e atua como o repositório de todas as informações e conhecimento. A segunda camada é composta pelas aplicações *Common Object Request Broker Architecture* (CORBA) construídas na linguagem de programação Java (CORBA *servlet*), e a terceira camada é composta por *applets*² que são obtidos através de *downloads* e executados nas máquinas clientes (Java Thin Client).

O *applet* se conecta ao CORBA *servlet* localizado em um servidor que possui os métodos tais como *get(Object_ID)* e *send(Object)* que recupera e envia um objeto entre o *applet* e o *servlet*. O CORBA *servlet* se comunica com o banco de dados através de uma interface.

Figura 2.11 – Arquitetura em camadas

Fonte: (BECK; XIN, 1998)

² **Applets** são pequenos aplicativos escritos em Java que se utilizam da JVM (Java Virtual Machine) existente na máquina cliente ou embutida no próprio browser do cliente para interpretar seu bytecode.

Na Universidade da Flórida, trabalhos experimentais usando OODBMS para construção de bancos de dados agrícolas têm sido realizados há algum tempo. Alguns sistemas foram construídos usando OODBMS como *ObjectStore*, *Poet* e *Versant*, por exemplo, a ferramenta para construção de Sistemas de Apoio a Decisão para Controle de Pragas (Figura 2.12).

Figura 2.12 – Ferramenta para construção de sistemas de apoio a decisão para controle de pragas

Fonte: (BECK; XIN, 1998)

Essa ferramenta, que permite a aquisição do conhecimento, foi desenvolvida em Java e pode ser acessada por especialistas a partir de estações experimentais localizadas em todo o estado da Flórida. Os especialistas podem usar essas ferramentas a qualquer hora e suas informações são registradas e disponibilizadas imediatamente para outros especialistas e usuários. A arquitetura desta solução vai ao encontro das idéias da arquitetura para sistemas

de informação para agricultura de precisão proposta nesta tese, com exceção do *middleware* orientado a objetos distribuídos baseado em CORBA.

2.4.3 *TeleFarm*

TeleFarm é uma plataforma de software para desenvolvimento de software para agricultura de precisão proposta por Lütticken (2000). Lütticken, com base na avaliação da percepção dos produtores e consultores sobre a AP e nos estudos das necessidades de informação e comunicação para a implantação da AP, propõe uma rede de comunicação sobre uma plataforma de software aberta, que possa ser operada pelo próprio agricultor e seus parceiros de negócio, no escritório ou no campo. Lütticken acredita que essa plataforma é fundamental para o progresso da AP e para o agronegócio como um todo e deverá superar as diversas barreiras ainda existentes como necessidade de dados básicos atualizados e sistemas incompatíveis e monolíticos. O conceito dessa plataforma de software é baseado na integração de modernos softwares (Java, GIS) e tecnologias de hardware (GPS, Internet) de uma maneira amigável, com o objetivo de obter a aceitação e melhorar a percepção da AP, por parte dos agricultores e parceiros no gerenciamento agrícola como um todo.

Na definição da plataforma, quanto a requisitos de software dos produtores, foram consideradas algumas suposições:

- Usuários são geralmente inexperientes com software;
- Requisitos educacionais têm que ser considerados;
- Softwares somente serão aceitos a um baixo custo;
- Integração de sistemas existentes é importante;
- Problemas no gerenciamento do dia a dia devem ser cobertos primeiro;
- Segurança dos dados é crucial.

Essa plataforma está sendo desenvolvida por uma corporação de fazendeiros da Alemanha, fabricantes de máquinas, consultores e desenvolvedores de software, que iniciaram com a máxima: “software de agronegócio para agronegócio”. A plataforma de software oferece uma interface gráfica do usuário comum, uma estrutura de banco de dados, um mecanismo de atualização e capacidade de comunicação interna e externa entre aplicações. Desse modo, fornece uma estrutura geral para uma rede de informação e comunicação e um ambiente padronizado onde os fazendeiros e seus parceiros são conectados. A estrutura da plataforma também permite adicionar desenvolvedores de software e integrar módulos existentes, o que permite agilizar o desenvolvimento e manter o custo baixo.

A plataforma foi chamada de *TeleFarm*, claramente expressando a inseparável integração de técnicas de telecomunicações dentro de modernas práticas agrícolas (Figura 2.13). A aparência da interface do usuário nasceu do *look and feel* da *Microsoft* e foi baseado nos conceitos do *browser HotJavaTM* da *Sun*. A interface segue os princípios do acesso intuitivo “aperte o botão” simples de aprender e usar. A aparência da interface do usuário pode ser alterada de fazenda para fazenda dependendo do tipo da fazenda, da estrutura ou equipamento e não há módulos separados para os diferentes pacotes de software: agrícola, animais, produção ou financeiro (Figura 2.14). A idéia é que a plataforma forneça um sistema de informação e gerenciamento para o agronegócio como um todo, com uma estrutura flexível e altamente comunicativa para a produção agrícola, marketing e finanças.

Figura 2.13 – Estrutura principal e parceiros dentro de uma rede agrícola
Fonte: (LÜTTICKEN, 2000)

Figura 2.14 – Interface gráfica da plataforma de software
Fonte: (LÜTTICKEN, 2000)

Para reduzir a redundância de dados um modelo de banco de dados orientado a objetos foi escolhido. Desse modo, os dados serão entrados uma única vez e irão então ser transferidos automaticamente entre os módulos relacionados. Essa é uma abordagem comum, não muito eficiente do ponto de vista do conceito de baixo acoplamento. O uso da base de dados como *middleware* torna a aplicação acoplada pela camada de dados.

Lütticken (2000) considera importante que a transferência de informações permita a integração completa de qualquer tipo de informação externa através de um formato padrão de dados em XML (*eXtensible Markup Language*) e que a plataforma de software deveria gerar documentos XML para troca de dados de forma padronizada. Por exemplo, ao baixar informações de pesticidas da Internet num documento XML, o sistema deveria integrar esse documento automaticamente, de maneira que, assim que o produtor vinculasse o pesticida a um talhão o sistema mostrasse as possíveis restrições na cultura ou talhão.

A plataforma usa a Internet como meio de comunicação devido a sua maneira eficiente de comunicar dados e pela familiaridade que os produtores já têm pelo fato de usarem a Internet das formas mais comuns como *home banking* e correio eletrônico. A integração da Internet dentro da plataforma é considerada uma forma de facilitar o seu uso para busca de informações e transferência de dados entre escritório, campo e parceiros de negócio.

A plataforma propõe o uso da terceira geração de sistemas móveis com o padrão de transmissão de dados *Universal Mobile Telecommunications System* (UTMS), que é capaz de receber imagens e mapas em telefones móveis. Acredita-se que essas tecnologias irão aumentar e melhorar o uso de telefones móveis em máquinas agrícolas para transferir e adquirir informações pela Internet.

Segundo Lütticken (2000), os agricultores querem primeiramente registrar e administrar os dados de campo mais precisamente antes de adotarem o gerenciamento localizado. Com a plataforma de software proposta, os produtores poderão organizar o

trabalho diário simplesmente atribuindo tarefas aos empregados, máquinas e talhões. Essas tarefas serão armazenadas num servidor e poderão ser acessadas do campo por operadores de máquinas agrícolas.

O cenário dessa proposta é o seguinte: As máquinas serão equipadas com computadores de bordo, equipados com a mesma plataforma de software do escritório. Para a comunicação de dados é usado um cartão de telefone PCMCIA (*Personal Computer Memory Card International Association*) e para registro automático de dados é usado um GPS. O computador de bordo fornecerá comunicação com o escritório ou outro parceiro móvel, para recuperar a lista de tarefas preparada pelo produtor a qualquer hora do dia via Internet. Outras informações como talhão, máquina ou cultura, também podem ser recuperadas pelo operador da máquina. Assim que uma tarefa é iniciada o GPS e o *Timer* documentam o trabalho realizado. Num intervalo definido, os dados são transferidos para o escritório a fim de mostrar o estado atual da tarefa, o que permite ao produtor, parceiro ou cliente, acompanhar o trabalho sendo realizado, uma vez que todos utilizam a mesma plataforma de software.

A plataforma de software proposta *TeleFarm* com comunicação via Internet e telefones móveis, pode ser usada pelos produtores agrícolas e seus parceiros de negócio, dentro e fora do campo. A plataforma de software preconiza a implementação de um ambiente padronizado para comunicação e gerenciamento de informação, pois uma das barreiras identificadas para a adoção da AP foi a falta de um sistema de software aberto para gerenciamento e comunicação dentro de uma rede agrícola.

2.4.4 Experiências em transferência de dados entre sistemas para AP

O Instituto de Agricultura, Alimento e Engenharia Ambiental da Universidade do Oeste da Hungria tem conduzido experiências em AP desde 1998. Em um dos seus experimentos foram usados dois sistemas: o RDS para mapeamento de produtividade e o

Agrocom ACT para amostragem de solo, aplicação localizada de nutrientes e monitoramento de produtividade (MESTERHÁZI et al., 2002).

Pelo fato de o RDS possuir uma base de dados mais valiosa que a do *Agrocom* e este ser mais completo em termos de funcionalidades, houve a necessidade da integração desses sistemas. O RDS possui informações importantes que não existem no *Agrocom*, por exemplo, a altura, uma informação importante para descrever as condições do relevo que pode ajudar a detectar pontos com risco de erosão. O *Agrocom* inclui o software *AgroMap Professional* baseado no *ArcView*, que torna possível manipular os objetos básicos do *AgroMap*, como limites do talhão, mapas de produtividade, mapas de aplicação localizada de nutrientes, etc., num Sistema de Informação Geográfica (SIG). Com a integração, os dados importantes do RDS podem ser processados pelo *AgroMap*.

A solução de integração foi a importação do arquivo gerado pelo RDS dentro do *AgroMap Professional*. No entanto, o formato do arquivo RDS com extensão “.gpc” (Figura 2.15) não é entendido pelo módulo de importação do *AgroMap*. Foi necessário transformar o arquivo “.gpc” no formato “.aft” entendido pelo *AgroMap* (Figura 2.16).

```
Job no. 2 @450 cm @8,14 t/ha @13,5% @16,79kg @47,683820° @17,959026° @124m @00
VN000002@0450@000814@01350@001679@+47683820@+017959026@001240@4@00
```

Figura 2.15 – Um registro do arquivo RDS

Yield	Moist.%	Y	X	Height (m)
8,14	13,5	47,68382	17,959026	124

Figura 2.16 – Um registro do arquivo do AgroMap, delimitado por tabulação, criado a partir do RDS

Além das diversas possibilidades de um SIG, com essa transformação foi possível, por exemplo, apresentar um modelo de relevo tri-dimensional (Figura 2.17).

Figura 2.17 – Modelo de relevo tri-dimensional usando dados obtidos pelo RDS

Fonte: (MESTERHÁZI et al., 2002)

A solução de integração através de arquivos texto ainda é bastante utilizada, no entanto, há diversas restrições como falta de semântica, estrutura pouco flexível para representação das informações, etc. Essas restrições não permitem que as informações sejam entendidas e manipuladas de forma automática pelas máquinas.

Essa solução foi muito utilizada antes do advento do padrão aberto XML. As novas soluções têm usado documentos XML para troca de dados entre sistemas heterogêneos.

2.4.5 Preagro-Profile

De 1999 a 2003 diversos pesquisadores trabalharam na pesquisa e desenvolvimento do projeto *Preagro* (<http://www.preagro.de>) para estabelecer um sistema de informação para agricultura de precisão (KORDUAN, 2003). Neste projeto muitos conjuntos de dados heterogêneos foram coletados. Esta base de dados foi construída a partir da prática dos fazendeiros produtores e foi ideal para o desenvolvimento dos modelos de dados e um sistema de armazenamento de dados com características reais para agricultura de precisão. A base de dados foi também usada para a extração de importantes metadados relacionados a AP, para

descrever os dados e realizar o armazenamento e recuperação em um sistema de catálogo. Um conjunto de metadados baseados no *Content Standard for Digital Geospatial Metadata* (CSDGM) recomendado pelo *Federal Geographic Data Committee* (FGDC) foi criado.

O desenvolvimento de elementos de metadados padronizados e fáceis de usar, melhora a interoperabilidade entre produtos de software e o suporte a mecanismos de troca e recuperação de dados automatizados. Metadados incluem informações sobre disponibilidade, finalidade de uso, formas de acesso para aquisição e transferência de dados. Metadados são necessários para uso de fontes de dados, acesso rápido à informação, diminuição de redundância, independência de aplicação, compartilhamento de dados entre aplicações e combinação de múltiplos conjuntos de dados. O projeto *Preagro* desenvolveu mais de 60 tipos de conjuntos de dados em quase 40 formatos diferentes.

Metadados pressupõe processamento de dados automatizados, além de permitir a interoperabilidade entre produtos de software e atores na cadeia de suprimento da produção agrícola e apoiar na rastreabilidade dos produtos. Baseado nas pesquisas realizadas no projeto *Preagro*, um modelo de metadados para um meta-sistema de informação baseado na Internet foi desenvolvido. Mais de 50 cientistas, 9 fazendeiros e provedores de serviços de diferentes regiões da Alemanha trabalharam juntos para obter informações na AP. Os conjuntos de dados coletados foram descritos por metadados, armazenados em um banco de dados e fornecidos em um meta-sistema de informação. Os elementos dos metadados foram adaptados para o padrão de metadados CSDGM. A versão estendida é chamada de *Preagro-Profile*.

No projeto *Preagro*, mais de 6500 conjuntos de dados espaciais de diferentes categorias e várias expansões espaciais e temporais foram coletados. Para compartilhar esses dados, um sistema de informação baseado na Internet foi projetado e implementado. Os dados geoespaciais das categorias de solo, planta, produtividade, econômico, ecológico, mapas básicos geográficos e sensoriamento remoto foram descritos como um conjunto de metadados

uniformes. Esses metadados foram armazenados em um banco de dados relacional. O modelo do banco de dados é um modelo de metadados que tem como entidade central um conjunto de dados. Esse conjunto de dados é descrito com metadados. Um conjunto de dados consiste de um ou mais arquivos dependentes de software, Por exemplo, *.shp, *.shx, *.dbf para *ArcView*, um formato comumente usado do *Environmental Systems Research Institute* (ESRI). Os conjuntos de dados podem ser consultados através de *web browsers* por categoria, formato, tipo e criador do conjunto de dados, região, fazenda, projeção de sistemas de coordenadas, data, etc.

Para permitir o acesso aos conjuntos de dados, as aplicações devem conhecer e usar o padrão de metadados CSDGM e suas extensões padronizadas criadas para elementos de dados que não possuem equivalência no padrão.

Mais de 6500 arquivos XML foram gerados, para diferentes tipos e conjuntos de dados, e armazenados em um diretório, prontos para serem consultados ou recuperados como exemplos. Para exportar os arquivos XML foi desenvolvido um programa em PHP script e disponibilizado num servidor web. Os elementos dos metadados e sua estrutura foram armazenados em um banco de dados no SGBD MySQL. Esses recursos podem ser acessados através de serviços disponíveis nas páginas da Internet do projeto *Preagro* (<http://www.preagro.de/Netzwerk>).

2.4.6 AgXML Standard

AgXML Standard (AgXML, 2004) são padrões para troca de dados entre companhias de processamento de grãos e óleo de sementes. Esses padrões são fornecidos sem custo para aqueles que utilizam os padrões de acordo com a licença da *AgXML L.L.C.* *AgXML L.L.C.* é uma Companhia de Responsabilidade Limitada (L.L.C) formada por um grupo de organizações com o compromisso de desenvolver padrões para o eficiente e efetivo uso da

comunicação de informações eletrônicas dentro da cadeia de suprimento do agronegócio como um todo. Entre outras organizações que participam da padronização estão a *Adams Net*, *Agris* (uma divisão da *John Deere*), *Archer Daniels Midland*, *Bunge* e *Cargill*, com a participação de organizações e agências como Departamento de Agricultura dos EUA, Associação Nacional de Sementes e Grãos e *RAPID Inc*, uma organização para padronização de negócios eletrônicos que serve a indústria agrícola.

O desenvolvimento dos padrões AgXML é um esforço aberto e a abordagem geral para desenvolvimento é a seguinte:

- Identificar os processos do negócio que, se fossem eletrônicos, melhorariam a eficiência e a eficácia do processo de negócio;
- Identificar os requisitos dos processos de negócio;
- Definir os *schemas* XML e as diretrizes relacionadas que suportam os requisitos;
- Obter compromisso dos participantes na integração dos seus processos de negócio através da comunicação com mensagens baseadas em XML e na formação de um fórum para entendimento desses processos.

Os *schemas* XML são o núcleo dos padrões AgXML. Esses *schemas* seguem as recomendações W3C do *XML Schema* e são divididos em cinco tipos:

- *Top-Level schemas*: são *schemas* que definem a estrutura de dados das mensagens que serão trocadas entre parceiros. Um *top-level schema* pode incluir componentes ou *schemas* de tipos de dados. Um exemplo é o *Contract.xsd*, que define a estrutura dos contratos de commodities;
- *Data-Type schemas* (dt*.xsd): são *schemas* compostos de um simples tipo de dados que contém valores enumerados. *Data-type schemas* nunca incluem outros *schemas*. *Data-type schema* sozinho nunca define mensagens entre parceiros de

negócio. Um exemplo é o *dtCountryRegionCode.xsd*, que contém os valores enumerados representando os estados e províncias;

- *Integrated Top-Level schemas* (*~.xsd): são os *Top-level schemas* que têm as referências dos *schemas* incluídos, substituídos pelos respectivos conteúdos. O resultado é um único arquivo, melhor que um conjunto de arquivos usados para representar a estrutura de dados de uma única mensagem de negócio. *Integrated top-level schemas* são fornecidos para conveniência dos desenvolvedores e não são para serem usados em produção!
- *Glossary schema* (*Glossary.xsd*): é um único *schema* que inclui todos os *schemas* de tipos de dados (*dt*.xsd*). O *glossary schema* sozinho nunca define uma mensagem entre parceiros de negócio.

Atualmente os padrões AgXML Standard disponíveis são:

- *Commodity-Contract* (disponível desde 2001)
- *PriceAuthority* (disponível desde 2004)

A documentação do AgXML inclui casos de uso e diagramas de atividades em UML (*Unified Modeling Language*), além de exemplos de instâncias de documentos XML e representações em HTML desses exemplos, o que facilita bastante o entendimento e uso dos padrões.

2.4.7 agroXML

Em uma iniciativa para a questão de integração e interoperabilidade de aplicações, no contexto agrícola, Doluschitz et al. (2005) propõe o agroXML, uma linguagem para troca de dados padronizada, baseada no conceito do padrão *eXtensible Markup Language* (XML). O

agroXML é composto de um *XML schema* e um dicionário. Os termos específicos sobre agricultura são armazenados no *schema* e descritos no dicionário. O ponto forte dessa proposta é o fato do *schema* e o dicionário do agroXML, estarem sendo submetidos a um grupo para serem discutidos e coordenados em uma força tarefa, na tentativa de obter o máximo de aceitação. Os resultados desse trabalho serão mantidos no repositório *agroXML-repository* e disponibilizados pela Internet.

O agroXML é uma linguagem para troca de dados compreensivos que ao mesmo tempo são independentes de fabricante, especialmente para o setor agrícola. O conceito do agroXML é baseado no XML, um padrão mundial para descrever documentos na web. O *schema* agroXML é livremente acessível e pode ser implementado em diferentes programas. Esse *schema* pode ser estendido com conceitos dos setores público e privado. Os potenciais usuários do agroXML são os envolvidos na produção ou cadeia de suprimentos do setor agrícola.

Os principais objetivos do desenvolvimento do agroXML são:

- Implementar um formato de dados amplamente conhecido para ser usado por todos os membros da cadeia do agro-alimento;
- Evitar múltiplas entradas de dados em diferentes níveis da cadeia do agro-alimento para minimizar a redundância;
- Harmonizar a discussão sobre o conteúdo e extensão do repositório do agroXML por envolver especialistas de diferentes origens em uma plataforma neutra; e
- Melhorar e acelerar a adoção.

2.4.8 AGRIS AP

AGRIS Application Profile (AGRIS AP, 2005, 2006) define um conjunto de metadados para uso na troca de informações agrícolas entre os países participantes do *International Information System for the Agricultural Sciences and Technology* (AGRIS). Este sistema foi criado pela *Food and Agricultural Organization of the United Nations* (FAO) em 1974, para facilitar a troca de informações e para relacionar a literatura de todo o mundo que tratam de quaisquer aspectos da agricultura. O AGRIS AP atualmente é mantido pelo grupo de coordenação AGRIS/CARIS da FAO.

AGRIS AP é um conjunto de metadados padronizados criado especificamente para melhorar a descrição, troca e subsequente recuperação de *Document-Like Information Objects* (DLIOs) agrícolas. AGRIS AP possui um formato que permite o compartilhamento de informações através de sistemas bibliográficos dispersos e é baseado em padrões de metadados bem conhecidos e aceitos como *Dublin Core Metadata Element Set* (DCMES), *Australian Government Locator Service Metadata* (AGLS) e *Agricultural Metadata Element Set* (AgMES).

O uso de *Application Profile* permite que implementadores compartilhem informações sobre o modelo de dados de suas aplicações. Isto promove uma terminologia comum e boas práticas no compartilhamento de informações. O *Application Profile* usa elementos dos *XML namespaces* que o compõe e não permite a criação de novos elementos. Essa é a diferença básica entre o *XML namespace* e o *Application Profile*. O *namespace* permite a criação de novos termos com URI (*Uniform Resource Identifier*) únicos. O uso de *namespaces* permite que outros implementadores, no mesmo domínio ou em domínio diferente, veja se um termo já foi criado, o que reduz a reinvenção de termos similares com significados diferentes ou diferentes termos com significados similares.

O *namespace* AgMES:

- Identifica a autoridade de gerenciamento para todos os elementos e *schemas* que neste caso é o WAICENT³;
- Fornece identificadores únicos para elementos; e
- Define unicamente *schemas* e listas controladas.

E o AGRIS AP:

- Usa um ou mais *namespaces* existentes (DCMES, AGLS e AgMES);
- Não introduz nenhum elemento ou refinamento novo;
- Especifica informação de cardinalidade e tipo de dado
- Especifica os valores e *schemas* permitidos e listas de valores controladas; e
- Refina, quando necessário, definições padrões de elementos ou refinamentos fornecidos pelos *namespaces*.

O principal objetivo do AGRIS AP é servir como um formato flexível para troca de informação, independente de plataforma, aderente aos atuais padrões de metadados e interoperável entre diferentes sistemas de informação com diferentes tipos de informações, dentro da rede AGRIS. A rede AGRIS é composta por mais de 200 centros de pesquisas.

O processo de compartilhamento de dados entre os participantes da rede usando o AGRIS é mostrado na Figura 2.18.

³ WAICENT - World Agricultural Information Centre Portal. Portal da FAO que provê conhecimento acumulado a respeito de informações agrícolas.

Figura 2.18 – Fluxo de conversão de um conjunto de dados para o AGRIS AP

Fonte: (AGRIS AP, 2006)

O primeiro passo envolve o mapeamento do modelo de dados local para o *Application Profile*. Em seguida é criado um script de conversão dos dados ou é realizada a exportação direta dos dados em um documento XML (alguns bancos de dados permitem isso). Sobre esses dados são aplicadas transformações para converter o documento XML com os dados exportados em outro documento XML no formato AGRIS AP. Sobre esse XML são aplicados processos de normalização de dados, que irão garantir não só a validação do arquivo XML pelo DTD (*Document Type Definition*), mas também que todos os requisitos tais como uso de vocabulário controlado e declarações de *schemas*, foram validados. Os documentos XML resultantes dessa conversão, representam a camada de troca de dados comum do AGRIS AP. Muitos centros de pesquisa AGRIS usam a ferramenta WebAGRIS (WebAGRIS, 2006) como um sistema gerenciador de banco de dados. O WebAGRIS permite a exportação dos registros para o formato AGRIS AP XML.

2.4.9 XML schema para dados agrometeorológicos

Os países nórdicos, Dinamarca, Noruega e Suécia, juntos com o Japão e Nova Zelândia desenvolveram um formato de dados comum para troca de informações

agrometeorológicos na forma de um *XML schema*. A principal intenção de uso desse padrão proposto é apoiar o desenvolvimento de aplicações cliente-servidor para advertência de pragas, de maneira que as informações e processamentos possam ser compartilhados (RAFOSS et al., 2005). Segundo os autores, essa proposta será submetida para revisão à divisão de agrometeorologia da *World Meteorological Organisation* (WMO).

A arquitetura proposta para o desenvolvimento das aplicações será baseada em *Service-Oriented Architecture* (SOA). A arquitetura assume quatro tipos de aplicações de software. O primeiro são aplicações clientes que requisitam dados e metadados, tais como uma lista de estações meteorológicas que têm dados disponíveis. O segundo são aplicações que encapsulam bancos de dados, que permitem que os dados do banco de dados sejam acessados por usuários autorizados sobre a Internet de uma maneira padronizada, através da interface *BDInterfaceParaClientes* (Figura 2.19). O terceiro tipo de aplicação de software é um provedor de registros de metadados, onde metadados de banco de dados e estações são registradas através de aplicações clientes que usam a interface *RegistroInterfaceParaClientes*.

Figura 2.19 – Arquitetura para as aplicações

Fonte: (RAFOSS et al., 2005)

O quarto tipo são encapsuladores de banco de dados que tornam os metadados de banco de dados disponíveis para o provedor de registros de metadados através da interface `RegistroInterfaceParaBD`.

2.4.10 Uma infra-estrutura para dados geoespaciais interoperáveis para AP

Korduan, Bill e Bölling (2004) propõem o estabelecimento de infra-estruturas de dados espaciais, *Spatial Data Infrastructures* (SDI) como solução para superar problemas como a falta de informações geográficas em todas as áreas da sociedade. Informações geográficas são componentes essenciais das vidas privada, pública e econômica, e o mercado não tem atendido a demanda, talvez por limitações técnicas, judiciais ou organizacionais.

Na agricultura e especialmente na agricultura de precisão, a informação geográfica é muito importante. A AP lida com variabilidade espacial que requer dados espaciais sobre solo, cultura, máquinas agrícolas e aplicações. Análises espaciais devem ser executadas com SIG e muitos conjuntos de dados diferentes têm que ser coletados e gerenciados. Por usarem diferentes dispositivos, que suportam diferentes formatos de dados e diferentes sistemas de referência espacial os produtores têm que transformar os dados para que possam ser usados entre SIGs proprietários, o que consome muito tempo e dinheiro. Além disso, as demandas em rastreabilidade, sustentabilidade e transparência da produção tem aumentado bastante, o que da perspectiva do produtor, seria melhor transferir as tarefas de gerenciamento de dados para provedores de serviços, os quais têm melhores condições para estabelecer e manter um SDI. Korduan, Bill e Bölling (2004) acreditam que se a informação geográfica da agricultura se tornar largamente acessível através das SDI ela pode facilmente ser usada em outras comunidades e para outros propósitos, como gerenciamento de desastre para avaliar doenças ou danos na produção devido a desastres naturais como seca e inundações. Na AP

informações como dados de solo e fluxo de água também são freqüentemente usados e essa informação pode também ser de interesse para outros propósitos ambientais.

Com o desenvolvimento de um SDI a interoperabilidade será melhorada, pois exige a construção de modelos de dados semânticos homogêneos e metadados padronizados para permitir o desenvolvimento de serviços geo-espaciais interoperáveis. A implementação de serviços espaciais como *Web Services* vai ao encontro das idéias do *OpenGIS Consortium* (OGC).

O SDI proposto por Korduan, Bill e Bölling (2004) sugere o uso do *XML namespaces* para a padronização de nomes, *Geographic Markup Language* (GML) para descrição de geometrias de forma uniforme e *eXtensible Markup Language* (XML) como forma de conversação.

Atualmente o termo interoperabilidade não está restrito a troca de dados, ele é estendido também para transferidores de funcionalidades entre diferentes sistemas, que pode ser realizado por interfaces. A comunicação através de interfaces proprietárias é um problema comum que tem sido resolvido através de interfaces independentes de fabricantes, por exemplo, a SQL (*Structured Query Language*), os *middlewares* para comunicação entre sistemas distribuídos e os *OGC Web Services*. O desenvolvimento de interfaces independentes de fabricantes para informações geográficas é o objetivo primário da OGC.

Os principais componentes do SDI proposto por Korduan, Bill e Bölling (2004) são: os metadados padrões para AP, os dados, o armazenamento dos dados, os *OGC Web Services* e o *OGC Client*. A Figura 2.20 mostra a arquitetura do SDI para agricultura de precisão.

Figura 2.20 – Arquitetura do SDI para agricultura de precisão

Fonte: (KORDUAN; BILL; BÖLLING, 2004)

A infra-estrutura permite que servidores *OGC Web Feature Service/OGC Web Map Service* (WFS/WMS) sejam acessados por outros servidores WFS/WMS, o que denota a sua capacidade de integração. Para criar um mapa, um *Cliente OGC* acessa serviços WMS/WFS em um servidor que acessa outros serviços WMS/WFS em servidores remotos.

A implementação da infra-estrutura de dados geográficos espaciais foi realizada com o *framework Deegree* (DEEGREE, 2006). O *Deegree* é um software livre sob a licença GNU (*General Public Licence Model*). O *framework* suporta as especificações do *OpenGIS Consortium* (OGC) e ISO/TC 211OGC. A arquitetura do *Deegree* é totalmente baseada nas

especificações e conceitos da OGC, desse modo não há problemas para integrar produtos padronizados de outros fabricantes como *ArcIMS* do *Environmental Systems Research Institute* (ESRI).

O *framework Deegree* suporta as especificações OGC, *Web Map Service* (WMS), *Web Feature Service* (WFS), *Web Coverage Service* (WCS), *Web Catalog Service* (WCAS) baseado no *Web Services Stateless Catalog Profile* da OGC, *Web Gazetteer Service* (WFS-G), *Web Terrain Service* (WTS), *Web Coordinate Transformation Service* (WCTS), *Geography Markup Language* (GML), *Filter Encoding* (FE) e *Styled Layer Descriptor* (SLD). O *framework* pode ser perfeitamente usado para implementações de soluções para agricultura de precisão porque todos os serviços requeridos são suportados, o *framework* é livre de código aberto, os serviços oferecidos são bem documentados e estão sendo usados por outros SDIs.

No protótipo do SDI o WFS foi construído sobre o *Jakarta Apache Tomcat Server* como uma aplicação web. Além do WFS foram realizadas as configurações para o *Filter Service* (FS), uma interface de filtragem de dados para o componente de armazenamento de dados e o *Display Element Generator* (DEG), que combina as regras de visualização do FS com gerador de dados *raster Render Service* (RS). Os serviços são visualizados pelo usuário através de um *Mozilla Scalable Vector Graphic Browser*. A interface gráfica foi construída com XHTML e disponibilizada num servidor HTTP Apache. A interface consiste de um visualizador de mapas, uma lista de camadas usadas, tipos de características e funções para navegação temporal e espacial, conforme mostrado na Figura 2.21.

Figura 2.21 – Interface gráfica para visualização de mapas

Fonte: (KORDUAN; BILL; BÖLLING, 2004)

2.4.11 Geo-Services para agricultura

Segundo Nölle (2004), a agricultura está experimentando novamente o uso intensivo das Tecnologias da Geoinformação e Sistemas de Informação Geográfica. O exemplo mais proeminente são as demandas para o *Integrated Administration and Controlling System* (IACS) como o *framework* de controle e gerenciamento dos subsídios concedidos às atividades agrícolas, que a partir de 2005, as tecnologias da geoinformação deverão ser

amplamente integradas na Europa em função da construção do *Land Parcel Information Systems* (LPIS), que deverá ser utilizado por todos os estados membros.

Aplicações tradicionais como:

- Agricultura de Precisão;
- Planejamento estrutural agrícola;
- Gerenciamento de Nascentes e Erosões;
- Consultoria Agrícola em Geral;

E temas como:

- Diretivas que estabelecem uma nova estrutura legal para a proteção sustentável da água (*The Water Framework Directive*);
- Rastreabilidade na Agricultura;
- Monitoramento de organismos manipulados geneticamente (GMO);
- Uso de padrões pelos fazendeiros para que possam receber o pagamento do subsídio (*Cross Compliance*),

colocam as tecnologias da geoinformação numa importante posição no mundo da TI aplicada à agricultura. A Figura 2.22 mostra os mais recentes e principais direcionamentos para o uso das tecnologias da geoinformação na agricultura.

Figura 2.22 – Seleção dos mais recentes e principais motivações para o uso das tecnologias da geoinformação na agricultura

Fonte: (NÖLLE, 2004)

Nölle acredita que as tecnologias da geoinformação deverão se tornar tecnologias chaves na agricultura como um todo e deverá afetar todos os níveis do agronegócio: a fazenda e os níveis administrativo e industrial. Devido a sua complexidade, o uso das tecnologias da geoinformação na agricultura precisa de muita informação vinda de diversas fontes (Figura 2.23).

Figura 2.23 – Múltiplos aspectos dos requisitos da geoinformação

Fonte: (NÖLLE, 2004)

O eficiente e efetivo uso da geoinformação na agricultura, assim como em outros domínios, depende principalmente do acesso fácil, rápido, seguro e livre de redundância a todos os dados e informações requeridos e distribuídos.

Desse modo, NÖLLE propõe a utilização de serviços geoespaciais (*geo-services*) padronizados operando dentro de infra-estruturas de dados espaciais (*SDI - Spatial Data Infrastructures*).

A abordagem proposta é o acesso às fontes de dados externas de forma mais direta possível, que é o objetivo central do SDI do estado alemão *North Rhine-Westfalia* (NRW) GDI NRW (BRÜGGEMANN, 2004).

O GDI NRW é baseado na especificação do *OpenGIS Consortium* (OGC) e no *Technical Committee* (TC) 211 da *International Organization for Standardization* (ISO) (<http://www.gdi-nrw.org/>).

Os usuários acessam os *geo-services* através de um Cliente *OGC-compliant* chamado de *Agricultural Information Component* (AIC). Os *geo-services* são serviços disponíveis em máquinas servidoras como *WebFeatureServices* (WFS) e *WebMapServices* (WMS). Essa arquitetura é semelhante à proposta por Korduan, Bill e Bölling (2004).

NÖLLE alerta que, embora haja um sentimento de que a recente idéia dos *geo-services* seja uma grande idéia, não existem exemplos suficientes, sob condições reais, para convencer a comunidade inteira a seguir a abordagem de serviços geoespaciais e SDI. Para garantir o sucesso dessa abordagem, mais iniciativas como o GDI NRW devem ser iniciadas. A infraestrutura descrita nesta tese propõe um provedor de serviços geoespaciais para agricultura de precisão implementado como um SDI.

2.4.12 MOSAICo

O MOSAICo (Modelo de Objetos para Sistemas Abertos de Informação de Campo para Agricultura de Precisão) (SARAIVA, 1998) é um meta-modelo de sistemas para AP, que não está vinculado a uma implementação específica, mas que serve de referência para o desenvolvimento de implementações posteriores, fornecendo a base conceitual dos requisitos dessa classe de sistemas. A Figura 2.24 apresenta o diagrama de casos de uso de alto nível, resumizando os usos e os atores do sistema. Nesse diagrama quatro atores podem ser identificados: operador, gerente, equipamento de aquisição de dados, equipamento de controle de aplicação. Três casos de uso envolvem os atores e o sistema: Inserir dados de campo, Gerar Mapas e Configurar o Sistema.

Figura 2.24 - Diagrama de casos de uso do MOSAICo

O caso de uso *Inserir dados de campo* envolve a entrada de diversos tipos de dados, de amostragem, da propriedade, dos equipamentos, etc. O caso de uso *Gerar mapas* refere-se a todo o processamento e à obtenção final dos mapas de aplicação. O caso de uso *Configurar o sistema* refere-se à programação do sistema para um determinado usuário, propriedade, cultura ou outras condições particulares, incluindo inserção de dados e conhecimento sobre culturas, máquinas, mercados, sobre o usuário, etc.

Na descrição de casos de uso, estes são subdivididos e detalhados, especificando-se a sequência de interações atores-sistema para cada caso, enfatizando os conteúdos trocados nas interações. As subdivisões dos três casos de uso primários resultaram nos sub-casos listados na Figura 2.25.

Caso de uso: **Inserir dados de campo**
 Registro de novo proprietário
 Registro de nova propriedade agrícola
 Registro de novo talhão
 Leitura de dados amostrados a partir de dispositivos externos
 Entrada manual de dados amostrados

Caso de uso: **Configurar o sistema**
 Configuração de regras de manejo
 Configuração da base de dados de insumos e equipamentos
 Configuração de receitas de tarefas
 Configuração de atalhos para automação de tarefas
 Registro de novo operador

Caso de uso: **Gerar mapas**
 Correção de arquivos de dados amostrados
 Criação de camadas de dados primárias
 Criação de camadas de dados básicas
 Edição manual de camadas de dados
 Criação de camadas de dados secundárias
 Realização de simulações de cenários
 Geração de mapas de aplicação.
 Geração de arquivos de saída para controladores de aplicação
 Impressão de mapa

Figura 2.25 - Casos de uso do MOSAICo

A Figura 2.26 mostra o diagrama de subsistemas do MOSAICo onde é apresentado o possível agrupamento dos objetos em subsistemas, enfatizando sua coesão em termos de conteúdo, ocorrência, e macro-funções às quais se relacionam.

Figura 2.26 – Diagrama de Subsistemas do MOSAICo

O MOSAICo é importante neste trabalho de pesquisa porque fornece requisitos a serem atendidos pelos sistemas de informação para AP. Ele proporciona uma forma estruturada de apresentar informações sobre as funções necessárias, e os componentes básicos dos sistemas. Por ser um modelo independente de implementação ele é um ponto de partida para o detalhamento de sistemas a serem implementados em qualquer plataforma computacional. Como será visto no capítulo 4, esse modelo foi usado em vários experimentos: na definição da linguagem de troca de dados PAML; na criação de um framework espaço-temporal (RIBEIRO et al., 2003; RIBEIRO; SARAIVA; MURAKAMI, 2003, 2004) para criação de componentes de negócio, que são construídos como serviços; e no protótipo criado para validação de algumas idéias deste trabalho.

2.4.13 Uma proposta para desenvolvimento de SI para AP com *Web Services*

Saraiva (2003) propôs o uso de *Web Services* para desenvolvimento e integração de sistemas para AP e biodiversidade. A motivação desse trabalho foi a necessidade de integração de sistemas de informação independentes que são desenvolvidos ou adquiridos por diferentes departamentos dentro de uma organização ou, ainda, a necessidade de integração de sistemas e dados entre diferentes organizações.

2.5 Considerações finais

Neste capítulo, foram analisadas diversas iniciativas acadêmicas e comerciais em termos de sistemas de informação para AP e algumas propostas voltadas para o apoio ao desenvolvimento de sistemas de informação para AP como modelos abstratos do negócio, plataformas de software, ferramentas para construção de sistemas e padrões para comunicação de dados entre aplicações.

Os sistemas criados em experimentos acadêmicos na sua maioria são pouco abrangentes e resolvem problemas específicos. São pouco flexíveis e a extensibilidade está amarrada a arquiteturas de software definidas pelos desenvolvedores, as quais não incorporam conceitos atuais que permitem a evolução do software. A capacidade de evolução dos sistemas é essencial para acompanhar a evolução das pesquisas em AP.

Os sistemas comerciais são mais abrangentes e permitem gerenciamento e tratamento dos dados gerados pela AP. No entanto, o mercado tem gerado produtos que não atendem plenamente a demanda dos agricultores em termos de funcionalidades, custo e principalmente em termos de capacidade de integração com outros sistemas.

Independente da origem dos sistemas, comerciais ou acadêmicos, o problema parece estar na capacidade de integração e nas soluções proprietárias, que dificultam a evolução dos sistemas e a comunicação com pacotes de terceiros.

O próximo capítulo apresenta alguns conceitos sobre arquitetura de software, padrões abertos e plataformas de software livre, os quais aplicados e usados de forma padronizada podem resolver ou pelo menos minimizar os problemas apresentados neste capítulo.

3 TECNOLOGIA DA INFORMAÇÃO: ALGUNS CONCEITOS, PADRÕES E TECNOLOGIAS

3.1 Introdução

Inicialmente a computação era tida como um mecanismo que tornava possível a automatização de determinadas tarefas. Com o avanço tecnológico, as grandes máquinas começaram a perder espaço para equipamentos cada vez menores e mais poderosos. A evolução das telecomunicações permitiu que, aos poucos, os computadores passassem a se comunicar. Conseqüentemente, tais máquinas deixaram de simplesmente automatizar tarefas e passaram a lidar com “Informação”.

Segundo Rezende e Abreu (2000) o termo “Tecnologia da Informação” (TI) serve para designar o conjunto de recursos tecnológicos e computacionais para a geração e uso da informação e está fundamentada nos seguintes componentes: hardware e seus dispositivos e periféricos; software e seus recursos; sistemas de telecomunicações; e gestão de dados e informações. As razões que levaram à disseminação do uso da TI incluem: melhoria nos processos de negócio; melhores controles; redução de custo; e agregação de valores aos serviços e produtos afetados.

Na agricultura de precisão o papel da TI de uma maneira geral está em prover ferramentas e sistemas de informação que auxiliem o gerenciamento da informação, integrando dados, provendo ferramentas de análise e tomada de decisão e gerando mapas de prescrição, produtividade, etc. O papel da TI na agricultura de precisão foi abordado em profundidade por Saraiva (1998) em sua tese de doutoramento, intitulada “Um Modelo de

Objetos para Sistemas Abertos de Informações de Campo para Agricultura de Precisão, MOSAICo”.

Este capítulo apresenta conceitos, padrões e tecnologias da informação usadas na proposta deste trabalho. Especial ênfase em arquitetura de software é dada devido a sua importância na construção de sistemas de informação e na proposta desse trabalho. Neste estudo houve a preocupação em selecionar e utilizar o máximo possível de padrões abertos e softwares livres em contrapartida à maioria das soluções dos sistemas de informação e infra-estruturas apresentadas no capítulo anterior, que mostra que os principais e mais utilizados sistemas de informação para AP são proprietários, complexos, com capacidade de integração limitada e principalmente de alto custo, um dos fatores que influencia diretamente a adoção da agricultura de precisão pelos pequenos e médios produtores.

3.2 Arquitetura de software

A arquitetura de software é uma importante disciplina para engenheiros de software. Essa disciplina tem evoluído sobre o tempo como evolução natural das abstrações de projeto, de como os engenheiros têm buscado melhorar a maneira de entender seu software e novas maneiras para construir maiores e mais complexos sistemas de software.

Embora haja diversas definições de “arquitetura” na literatura, pode-se observar muita similaridade entre elas. Por exemplo, a maioria das definições indica que uma arquitetura está relacionada à estrutura e ao comportamento, às decisões significativas, a um estilo arquitetural e é influenciada por seus interessados e seu ambiente.

Para Jazayeri, Ran e Linden (2000), a arquitetura de software é colocada como uma ferramenta para lidar com a complexidade do software e enfatizam que arquitetura deve satisfazer os requisitos funcionais e não funcionais do sistema, incrementando a definição de que arquitetura de software é o conjunto de componentes e seus relacionamentos. Portanto, é

possível notar que a arquitetura de software é mais do que a descrição dos componentes que a compõem e do relacionamento entre eles.

Kruchten (2003) diz que uma arquitetura é um conjunto de decisões significativas sobre a organização de um sistema de software, a seleção dos elementos estruturais e suas interfaces pelos quais o sistema é composto, o seu comportamento como especificado nas colaborações entre os elementos, a composição destes elementos dentro dos subsistemas e o estilo arquitetural que guia a organização dos elementos, suas interfaces, colaborações e composição.

Bass, Clements e Kazman (2003) definem arquitetura de software de um programa ou sistema de computação, como estrutura ou estruturas do sistema, que compreende elementos de software, propriedades desses elementos visíveis externamente e seus relacionamentos.

3.2.1 Abstração

O princípio da abstração é o coração da arquitetura de software, que consiste de esconder alguns detalhes do sistema através do encapsulamento para melhor identificar e sustentar suas propriedades. Um sistema complexo pode ter muitos níveis de abstração, cada qual com sua própria arquitetura. Uma arquitetura representa uma abstração do comportamento do sistema com elementos arquiteturais que fornecem interfaces abstratas a outros elementos num determinado nível. Dentro de cada elemento pode ser encontrada outra arquitetura, definindo o sistema de sub-elementos que implementam o comportamento representado pelas interfaces abstratas dos elementos pais. Esta recursão de arquiteturas continua descendo para os mais básicos elementos do sistema: aqueles que não podem ser decompostos em elementos abstratos menores.

Além dos níveis de arquitetura, um sistema de software terá freqüentemente múltiplas fases operacionais, tais como inicialização, processamento normal, re-inicialização e parada.

Cada fase operacional tem sua própria arquitetura. Por exemplo, um arquivo de configuração será tratado como um elemento de dado durante a fase de inicialização, mas não será considerado um elemento arquitetural durante o processamento normal, desde que naquele ponto a informação que ele continha já tenha sido distribuída no sistema. Uma descrição completa de uma arquitetura de software deve ser capaz de descrever não apenas o comportamento operacional da arquitetura durante cada fase, mas também a arquitetura de transição entre as fases.

Perry e Wolf (1992) definem elementos de processamento como “transformadores de dados”, enquanto Shaw e Clements (1997) descrevem componentes como uma unidade de software que executa alguma função em tempo de execução. Exemplos incluem programas, objetos, processos e filtros. Essas definições aumentam a distinção entre arquitetura de software e o que é tipicamente referenciado como estrutura de software. Arquitetura de software é uma abstração de um comportamento de um sistema de software e estrutura de software é uma propriedade do código fonte. Embora existam vantagens em ter uma estrutura modular do código combinado com a decomposição do comportamento dentro de um sistema, existem também vantagens em ter componentes de software independentes implementados usando partes do mesmo código (por exemplo, bibliotecas compartilhadas). Fielding (2000) separa a visão da arquitetura de software do código fonte para focar nas características do software independente da implementação do componente. Embora o projeto arquitetural e projeto estrutural de código fonte sejam fortemente relacionados, as atividades de projeto são separadas.

3.2.2 Elementos

Fielding (2000) diz que a arquitetura de software é definida pela configuração dos elementos arquiteturais, *componentes*, *conectores* e *dados*, restringidos em seus relacionamentos para obter um conjunto desejado de propriedades arquiteturais.

Perry e Wolf (1992) apresentam um modelo que define uma arquitetura de software como um conjunto de *elementos* arquiteturais que tem uma *forma* particular, explicados por um conjunto de *razões*. Os *elementos* arquiteturais incluem, elementos de processamento, de dados e de conexão. A *forma* é definida por propriedades de elementos e relacionamentos entre eles, isto é, restrições nos elementos. A *razão* fornece a base para a arquitetura por motivar a escolha do estilo arquitetural, dos elementos e da forma.

Uma característica chave do modelo em (PERRY; WOLF 1992) é a distinção de vários tipos de elementos. *Elementos de processamento* são aqueles que executam transformações em dados, *elementos de dados* são aqueles que contêm a informação que é usada e transformada, e *elementos de conexão* são elementos que mantêm os diferentes pedaços da arquitetura juntos. Fielding (2000) usa os termos *componentes* e *conectores* para referenciar os elementos de *processamento* e *conexão* respectivamente.

Garlan e Shaw (1993) descrevem uma arquitetura de um sistema como uma coleção de componentes computacionais com uma descrição de interações entre eles, os conectores. A arquitetura de um sistema de software define esse sistema em termos de componentes e de interações entre esses componentes. Para especificar a estrutura e topologia do sistema, a arquitetura deve mostrar a correspondência entre os requisitos e os elementos do sistema construído.

3.2.2.1 Componentes

Segundo Fielding (2000) um componente é uma unidade abstrata de instruções e estado interno de software que fornece uma transformação dos dados através de sua interface.

Os componentes são os aspectos mais facilmente reconhecidos da arquitetura de software. Os elementos de processamento (PERRY; WOLF, 1992) são definidos como aqueles componentes que realizam a transformação nos elementos de dados.

Garlan e Shaw (1993) descrevem componentes simplesmente como os elementos que executam computação. Fielding (2000) é mais preciso e define um componente como uma unidade abstrata de instruções e estado interno do software que fornece uma transformação dos dados através de suas interfaces. Exemplo de transformação inclui carga da memória a partir de armazenamento secundário, execução de algum cálculo, tradução para um formato diferente, encapsulamento de outros dados, etc. Um componente é melhor definido por sua interface e serviços que fornece a outros componentes do que pela implementação de suas interfaces.

3.2.2.2 Conectores

Fielding (2000) define conector como um mecanismo abstrato que media a comunicação, coordenação ou cooperação entre componentes. Shaw e Clements (1997) dizem que um conector é um mecanismo que media a comunicação, coordenação ou cooperação entre componentes. Exemplos incluem: chamada a procedimentos remotos, protocolos de passagem de mensagens e fluxo de dados.

Conectores permitem a comunicação entre componentes pela transferência de elementos de dados de uma interface para outra sem mudar os dados. Internamente, um conector pode consistir de um subsistema de componentes para transformar os dados,

executar a transferência e então reverter a transformação para entrega. Entretanto, a abstração comportamental externa capturada pela arquitetura ignora esses detalhes.

3.2.2.3 Dados

Um dado é um elemento de informação que é transferido ou recebido de um componente, através de um conector. Exemplos incluem seqüência de bytes, mensagens, parâmetros empacotados e objetos serializados, mas não inclui informação que é permanentemente residente ou escondida dentro de componentes. Da perspectiva arquitetural, um “arquivo” é uma transformação, para uma seqüência de bytes, que um componente sistema de arquivo pode fazer a partir de um dado “nome de arquivo” recebido em sua interface. A natureza dos elementos de dados numa arquitetura de aplicação baseada em rede irá freqüentemente determinar se um estilo arquitetural é ou não apropriado.

3.2.3 Propriedades arquiteturais

O conjunto de propriedades arquiteturais de uma arquitetura de software inclui todas as propriedades que derivam da seleção e organização de componentes, conectores e dados dentro do sistema. Exemplos incluem, propriedades funcionais e não funcionais obtidas do sistema tais como facilidade de evolução, reusabilidade de componentes, eficiência e extensibilidade, freqüentemente referenciadas como atributos de qualidade (BASS; CLEMENTS; KAZMAN, 2003).

A seguir são apresentadas algumas propriedades arquiteturais. A intenção não é fornecer uma lista ampla e, sim, apenas aquelas propriedades que influenciam os estilos arquiteturais analisados neste trabalho. As propriedades arquiteturais são cobertas pela maioria dos livros de engenharia de software.

3.2.3.1 Escalabilidade

Escalabilidade refere-se a habilidade da arquitetura suportar grande número de componentes ou interações entre componentes dentro de uma configuração ativa. A escalabilidade pode ser melhorada pela simplificação dos componentes, pela distribuição dos serviços entre muitos componentes (descentralização de interações) e pelo controle das interações e configurações como resultado de monitoramento. Estilos influenciam esse fatores por determinar a localização dos estados da aplicação, o grau de distribuição e o acoplamento entre os componentes.

3.2.3.2 Simplicidade

O princípio da separação de interesses para alocação de funcionalidades dentro de componentes é uma maneira de induzir a simplicidade. Se a funcionalidade pode ser alocada tal que os componentes fiquem substancialmente menos complexos, então eles serão fáceis de entender e de implementar.

Aplicar o princípio da generalidade para elementos arquiteturais também melhora a simplicidade.

3.2.3.3 Modificabilidade

A modificabilidade diz respeito à facilidade com que uma mudança pode ser feita em uma arquitetura de aplicação ou componente. Um sistema deve estar preparado para mudanças fragmentadas e graduais, onde as novas e antigas implementações coexistam sem impedir as novas implementações de fazerem uso das suas capacidades de extensão.

A Modificabilidade pode ser quebrada em partes menores: extensibilidade, configurabilidade, customizabilidade e reusabilidade, como descrito a seguir.

3.2.3.3.1 Extensibilidade

A extensibilidade é definida como a capacidade de adicionar funcionalidade a um sistema. Extensibilidade dinâmica implica em adicionar uma funcionalidade a um sistema implantado sem impactar no resto dele.

3.2.3.3.2 Configurabilidade

A configurabilidade está relacionada a extensibilidade e a reusabilidade, no que se refere à modificação de componentes após a implantação, tal que eles sejam capazes de usar um novo serviço ou tipo de elemento de dados.

3.2.3.3.3 Customizabilidade

A customizabilidade refere-se a capacidade para especializar temporariamente o comportamento de um elemento arquitetural, tal que ele possa executar um serviço incomum. Um componente é customizável se ele pode ser estendido por um cliente dos serviços do componente sem causar impacto em outros clientes daquele componente.

3.2.3.3.4 Reusabilidade

A reusabilidade é uma propriedade de uma arquitetura de aplicação, se seus componentes, conectores ou elementos de dados podem ser reusados, sem modificação, em outras aplicações.

3.2.3.4 Portabilidade

Um software é portátil se ele pode ser executado em diferentes ambientes.

3.2.3.5 Confiabilidade

Confiabilidade, na perspectiva de arquiteturas de aplicação, podem ser visualizadas como o grau que uma arquitetura é suscetível a falhas, no nível de sistema, na presença de falhas parciais dentro dos componentes, conectores ou dados. Estilos podem melhorar a confiabilidade por evitar pontos de falhas, permitir redundância, monitoramento ou reduzir o escopo da falha para uma ação recuperável.

3.2.4 Visões arquiteturais

Além das muitas arquiteturas dentro de um sistema, e dos muitos estilos arquiteturais, das quais as arquiteturas são compostas, é também possível ver uma arquitetura de muitas perspectivas diferentes. Perry e Wolf (1992) descrevem três importantes visões em arquitetura de software: visões de processos, dados e conexões. A visão de processo enfatiza o fluxo de dados através dos componentes e alguns aspectos das conexões entre os componentes com respeito a dados. A visão de dados enfatiza o fluxo de processamento, com menos ênfase nos conectores. A visão de conexão enfatiza o relacionamento entre os componentes e o estado da comunicação.

Múltiplas visões arquiteturais são comuns dentro de estudos de caso de arquiteturas específicas (BASS; CLEMENTS; KAZMAN, 2003). Uma metodologia de projeto arquitetural, o Modelo de Visão 4+1 (KRUCHTEN, 1995), organiza a descrição de um software usando cinco visões concorrentes, cada qual objetivando um específico conjunto de interesses: visão lógica, enfatiza o modelo de objetos lógico do design (quando um método de design orientado a objetos é usado); visão de processo, captura os aspectos de concorrência e sincronização do design; visão física, descreve o mapeamento do software dentro do hardware e reflete seus aspectos distribuídos; visão de desenvolvimento, descreve a organização estática

do software em seu ambiente de desenvolvimento; e a visão de cenários ilustra as 4 visões através de um conjunto de cenários ou casos de uso.

3.2.5 Estilos arquiteturais

Uma vez que uma arquitetura engloba propriedades funcionais e não funcionais, é difícil comparar arquiteturas diretamente para diferentes tipos de sistemas ou comparar os mesmos tipos de sistema em ambientes diferentes. Os estilos são um mecanismo para categorizar arquiteturas e para definir suas características comuns (DI NITTO; ROSENBLUM, 1999). Cada estilo fornece uma abstração para as interações de componentes, capturando a essência de um padrão de interação ignorando os detalhes do resto da arquitetura.

Perry e Wolf (1992) definem estilo arquitetural como uma abstração de tipos de elementos e aspectos formais das várias arquiteturas específicas, talvez concentrando apenas em certos aspectos de uma arquitetura. Um estilo arquitetural encapsula decisões importantes sobre os elementos arquiteturais e enfatiza restrições importantes nos elementos e seus relacionamentos. Essa definição permite aos estilos focarem nos conectores de uma arquitetura ou em aspectos específicos das interfaces dos componentes.

Garlan e Shaw (1993) e Shaw e Clements (1997), definem estilo em termos de um padrão de interações entre componentes tipados. Especificamente, um estilo arquitetural determina o vocabulário de componentes e conectores que podem ser usados em instâncias daquele estilo, junto com um conjunto de restrições de como eles podem ser combinados.

Novas arquiteturas podem ser definidas como instâncias de estilos específicos (DI NITTO; ROSENBLUM, 1999). Desde que estilos arquiteturais podem comunicar diferentes aspectos de arquitetura de software, uma dada arquitetura pode ser composta de múltiplos

estilos. Do mesmo modo, um estilo híbrido pode ser formado pela combinação de múltiplos estilos básicos dentro de um único estilo coordenado.

Alguns estilos arquiteturais são freqüentemente usados para soluções para todas as formas de sistemas. Entretanto, um bom projetista deveria selecionar um estilo que atenda as necessidades de um problema particular a ser resolvido. Escolher o estilo arquitetural certo para uma aplicação baseada em rede requer um entendimento do domínio do problema.

Os estilos arquiteturais escolhidos para projeto de sistemas devem atender ou exceder as necessidades desse sistema. Com base nos estudos dos sistemas de informação para AP apresentado no capítulo anterior e na análise realizada, foram selecionados alguns estilos arquiteturais considerados potencialmente interessantes para minimizar os problemas encontrados nos atuais sistemas para AP, os quais são brevemente descritos abaixo.

3.2.5.1 Cliente-servidor

O estilo cliente-servidor é o estilo arquitetural mais freqüentemente encontrado para aplicações baseadas em rede. Um componente servidor oferece um conjunto de serviços e fica aguardando requisições a esses serviços. Um componente cliente, desejando que um serviço seja executado, envia uma requisição ao servidor por um conector. O servidor rejeita ou executa a requisição e envia uma resposta de volta ao cliente. Um servidor é usualmente um processo que não termina e serve a mais de um cliente. Uma variedade de sistemas cliente-servidor são analisados por Sinha (1992).

A separação dos interesses é o princípio por trás das restrições cliente-servidor. Uma separação das funcionalidades deveria simplificar o componente servidor e melhorar a escalabilidade. Frequentemente, as funcionalidades são referenciadas pelos mecanismos usados pela implementação do conector, tal como chamada a procedimento remoto ou *middleware* orientado a mensagem (UMAR, 1997).

3.2.5.2 Sistema em camadas e cliente-servidor em camadas

Um sistema em camadas é organizado hierarquicamente, cada camada fornecendo serviços para a camada acima e usando serviços da camada abaixo (GARLAN; SHAW, 1993). Embora o sistema em camadas seja considerado um estilo “puro”, seu uso em sistemas baseados em rede é limitado à sua combinação com o estilo cliente-servidor para fornecer o cliente-servidor em camadas.

Os sistemas em camadas reduzem o acoplamento através de múltiplas camadas por esconder as camadas internas das outras, com exceção da adjacente e desse modo, melhoram a reusabilidade. A desvantagem é que eles adicionam um *overhead* e latência, ao processamento dos dados, reduzindo o desempenho.

Os sistemas cliente-servidor em camadas adicionam componentes *proxy* e *gateway* ao estilo cliente-servidor. Um *proxy* (SHAPIRO, 1986) atua como um servidor compartilhado para um ou mais componentes clientes, recebendo requisições e encaminhando-as, com possível tradução, aos componentes servidores. Um componente *gateway* aparece como um servidor normal para os clientes ou *proxies* que requisitam seus serviços, mas está de fato encaminhando as requisições, com possível tradução, aos seus servidores. Esses componentes mediadores podem ser adicionados em múltiplas camadas para adicionarem características como balanceamento de carga e checagem de segurança para o sistema.

As arquiteturas baseadas em cliente-servidor em camadas são referenciadas como arquiteturas duas camadas, três camadas ou multicamadas na literatura de sistemas de informação (UMAR, 1997).

3.2.5.3 Objetos distribuídos

O estilo objetos distribuídos organiza um sistema como um conjunto de componentes interagindo como pares. Um objeto é uma entidade que encapsula alguma informação de estado ou dado privado, um conjunto de operações ou procedimentos que manipulam dados e possivelmente um canal de controle, de maneira que coletivamente eles podem ser considerados uma única unidade (CHIN; CHANSON, 1991). Em geral, um estado de objeto é completamente escondido e protegido de todos os outros objetos. A única maneira de examiná-lo ou modificá-lo é através de requisição ou invocação das suas operações acessíveis publicamente. Isso cria uma interface bem definida para cada objeto, permitindo que as especificações das operações dos objetos sejam públicas e ao mesmo tempo mantendo a implementação de suas operações e a representação de suas informações privadas, aumentando assim a sua capacidade de evolução.

Uma operação pode invocar outras operações, possivelmente outros objetos. Essas operações podem sucessivamente criar invocações em outros, e assim por diante. Uma cadeia de invocações é referenciada como uma ação (CHIN; CHANSON, 1991). O estado é distribuído entre os objetos. Isto pode ser vantajoso em termos de manter o estado onde é mais provável ser atual, mas tem a desvantagem de ser mais difícil de obter uma visão completa das atividades do sistema (pouca visibilidade).

Para um objeto interagir com outro, ele deve conhecer a identidade do outro objeto. Quando a identidade de um objeto muda, é necessário modificar todos os outros objetos que explicitamente o invocam (GARLAN; SHAW, 1993). Deve existir algum objeto controlador que é responsável em manter o estado do sistema a fim de completar as exigências da aplicação. As responsabilidades centrais dos sistemas de objetos distribuídos incluem:

gerenciamento de objetos, gerenciamento de interações de objetos e gerenciamento de recursos (CHIN; CHANSON, 1991).

3.2.5.4 Orientado a serviço

Service-Oriented Architecture (SOA) é um estilo com restrições para induzir um número de características desejáveis em uma arquitetura (KOROTKIY, 2005). Na literatura, SOA é freqüentemente caracterizada como um estilo que suporta fraco acoplamento, alinhamento com o negócio, serviço baseado em rede, para permitir flexibilidade e interoperabilidade, independentemente de tecnologia. Huhns e Singh (2005) apresentam os conceitos chaves da computação orientada a serviços e o modelo arquitetural dos *Web Services* como a base para uma arquitetura orientada a serviços.

Os serviços em SOA representam o conhecimento do domínio do negócio numa granularidade alta e permite que esse conhecimento esteja disponível para um consumidor. SOA permite obter baixo acoplamento entre seus componentes de processamento, consumidor de serviço e serviço, porque requer que os conectores sejam simples, genéricos e independentes de aplicação, e as mensagens descritivas, definidas num formato padrão, sejam entregues através desses conectores.

Uma vez que os conectores são genéricos, a semântica específica da aplicação deve ser expressa em mensagens descritivas que comunicam a descrição do problema, de um consumidor para um fornecedor de serviço. Essas mensagens especificam o que deve ser resolvido, mas não como deve ser resolvido. A razão disso é porque o fornecedor de serviço é quem possui a capacidade de resolver o problema que falta no consumidor do serviço.

A fim de permitir a "compreensão", na comunicação entre componentes, os dados devem ter estrutura e sintaxe unificada e devem ser expressas por um vocabulário

compartilhado. O vocabulário é definido em um *schema*, permitindo extensibilidade, uma propriedade muito importante devido ao grande número de domínios de aplicação possíveis.

Em SOA, o baixo acoplamento é conseguido forçando os elementos conectores serem independentes de aplicação. Toda a semântica específica da aplicação deve ser contida em elementos de dados (mensagens). Um serviço pode ser unicamente caracterizado por um conjunto de mensagens que ele pode interpretar corretamente. Essas mensagens são restringidas por um determinado *schema* que pode ser visto como uma definição abstrata do serviço. Do ponto de vista do consumidor de um serviço, o serviço é conhecido unicamente através dessa definição abstrata. Uma vez que um serviço representa a habilidade para resolver um problema num domínio de aplicação, o *schema* efetivamente tem o papel de especificação dessa habilidade.

3.2.6 Arquiteturas de referência

Uma arquitetura de referência serve de base para a elaboração de arquiteturas de software. As arquiteturas de referência são definidas a partir de modelos de referência e estilos arquiteturais, que são aplicados a um domínio específico. A maturidade dos domínios facilita o trabalho dos arquitetos de software, pois são padronizados, documentados e bem conhecidos. Por exemplo, os domínios de compiladores, sistemas gerenciadores de base de dados e sistemas operacionais. Desse modo, os arquitetos não precisam investir na definição da arquitetura e sim na projeção das propriedades arquiteturais das arquiteturas de referência que mais se assemelham a sua necessidade.

Muitas vezes os modelos de referência, os estilos arquiteturais e as arquiteturas de referência são generalizados e confundidos como arquitetura de software. No entanto, nenhum deles é arquitetura de software (BASS; CLEMENTS; KAZMAN, 2003). São elementos úteis para a definição da arquitetura de um sistema, que indicam decisões importantes.

Um *modelo de referência* consiste na decomposição padronizada do problema em partes conhecidas que cooperam entre si em prol de uma solução. Geralmente, estes problemas são de domínio bastante amadurecido e trazem a experiência de analistas de negócio em conjunto com desenvolvedores (BASS; CLEMENTS; KAZMAN, 2003).

Os *estilos arquiteturais* expressam esquemas de organização estrutural de sistemas, fornecendo um conjunto de componentes do sistema, suas responsabilidades e a forma de interação entre eles, estabelecendo um padrão de utilização.

Uma *arquitetura de referência* consiste em componentes de software e os relacionamentos entre eles que implementam funcionalidades relativas às partes definidas no modelo de referência. Cada uma destas partes pode ser implementada em apenas um ou vários componentes de software (BASS; CLEMENTS; KAZMAN, 2003). As arquiteturas de referência são aplicáveis a um domínio particular.

Os modelos de referência agregam solução aos problemas do ponto de vista de negócio e quais arquiteturas de referência apresentam uma solução do ponto de vista técnico para o domínio específico. Um estilo arquitetural é usado juntamente com o modelo de referência para a definição da arquitetura de referência, a qual irá apoiar a definição da arquitetura de software, como mostra a Figura 3.1.

Figura 3.1 – Relacionamento entre modelo de referência, estilo de arquitetura e arquitetura de referência implementando uma arquitetura de software

Fonte: (BASS; CLEMENTS; KAZMAN, 2003)

Concluindo a definição, pode-se dizer que modelos de referência, estilos de arquitetura e arquiteturas de referência não são arquiteturas de software. São passos úteis voltados para a definição da arquitetura de um software. Cada qual constitui-se de um conjunto de decisões antecipadas de projeto e direciona o fundamento básico para os próximos passos (BASS; CLEMENTS; KAZMAN, 2003).

Tais técnicas podem ser usadas para definir a arquitetura para a construção de um sistema simples bem como para a construção de grandes sistemas compostos por subsistemas independentes, os quais possuem características particulares que diferenciam suas arquiteturas específicas, mas as características genéricas são compartilhadas pela arquitetura de referência do sistema. As arquiteturas específicas de cada subsistema não são definidas isoladamente. Elas são derivações da arquitetura definida para o sistema todo.

3.3 Padrões abertos e tecnologias relacionadas

Esta seção apresenta os principais padrões abertos e tecnologias relacionadas usadas na proposta da infra-estrutura.

3.3.1 W3C e OASIS

O consórcio da *World Wide Web* (W3C) (W3C, 2006) é um consórcio internacional formado por mais de 400 organizações de mais de 40 países, onde uma equipe com dedicação exclusiva e o público trabalham juntos para desenvolver padrões web. A missão do W3C é conduzir a *World Wide Web* ao seu potencial máximo, desenvolvendo protocolos e diretrizes que garantam o crescimento da web por longo tempo. O W3C já publicou mais de noventa padrões chamados recomendações W3C, por exemplo, XML, *XML Schema*, *Namespace*, SVG (*Scalable Vector Graphics*), XHTML, SOAP (*Simple Object Access Protocol*), SMIL (*Synchronized Multimedia Integration Language*), MathML, etc. Na perseguição de sua

missão, a W3C tem objetivos a longo prazo para criar uma grande rede *World Wide Web* que permita uma web para todos e em todo lugar.

Para alcançar o objetivo de criar uma web para todos e em todo lugar, a W3C cria e promove formatos e protocolos abertos interoperáveis (não-proprietários) evitando a fragmentação do mercado do passado. A pilha de tecnologias do W3C é mostrada na Figura 3.2.

Figura 3.2 – Pilha de tecnologias W3C

Fonte: (W3C, 2006)

A pilha de tecnologias W3C mostra uma visão da infra-estrutura da web, o maior foco dos trabalhos do W3C. A base fundamental, URIs, HTTP, XML e RDF (*Resource Description Framework*) suporta os objetivos de cinco áreas: acessibilidade, internacionalização, independência de dispositivo, acesso móvel e garantia da qualidade, as quais permeiam as tecnologias W3C.

3.3.1.1 Web Services

Freqüentemente os *Web Services* e a Arquitetura Orientada a Serviços (SOA) são vistos como uma combinação, mas eles são distintos. SOA representa um conceito arquitetural abstrato, uma abordagem para construir sistemas de software baseados em componentes com baixo acoplamento. Os *Web Services* representam uma importante abordagem para realizar SOA.

O *World Wide Web Consortium* (W3C) define *Web Services* como um sistema de software projetado para suportar interação máquina a máquina sobre uma rede. Possui uma interface descrita em um formato capaz de ser processado por máquinas, especificamente *Web Service Description Language* (WSDL) (BOOTH et al., 2004). Outros sistemas podem interagir com *Web Services* da maneira prescrita pela sua descrição, usando mensagens *Simple Object Access Protocol* (SOAP), normalmente usando *Hypertext Transfer Protocol* (HTTP) com *eXtensible Markup Language* (XML) em conjunto com outros padrões relacionados à web.

Embora a tecnologia *Web Services* não seja a única abordagem para realizar SOA, ela é uma tecnologia que a indústria de tecnologia da informação como um todo tem aceitado (HUHNS; SINGH, 2005; KOROTKIY, 2005; PASLEY, 2005; WANG et al., 2005). Com *Web Services*, a indústria está novamente buscando o desafio que a computação distribuída forneceu por um longo período: fornecer uma maneira uniforme de descrever componentes ou serviços dentro de uma rede, localizando e acessando-os. A diferença entre a abordagem *Web Services* e as abordagens tradicionais como as tecnologias de objetos distribuídos da *Object Management Group* (OMG), *Common Object Request Broker Architecture* (CORBA) ou o *Microsoft Distributed Component Object Model* (DCOM), está no aspecto de baixo acoplamento da arquitetura. Em vez de construir aplicações que resultam em coleções de

objetos ou componentes firmemente integrados, que são bem conhecidos e entendidos em tempo de desenvolvimento, a abordagem de serviços é muito mais dinâmica e adaptável a mudanças. Outra diferença chave é que através dos *Web Services*, a indústria está resolvendo os problemas usando tecnologia e especificações que estão sendo desenvolvidas de uma maneira aberta utilizando parcerias e grandes consórcios como *World Wide Web Consortium* (W3C) e *Organization for the Advancement for Structured Information Standards* (OASIS) e baseado em padrões e tecnologias da Internet.

Web Services teve seu início em meados de 2000 com a introdução das primeiras versões XML do protocolo *Simple Object Access Protocol* (SOAP), da linguagem de descrição *Web Service Description Language* (WSDL) e do serviço de registro *Universal Description, Discovery and Integration* (UDDI). Esse conjunto básico de padrões forneceu à indústria uma base para interoperabilidade entre componentes de software (*Web Services*) independente da localização na rede, para especificar detalhes de implementação tanto dos serviços quanto da infra-estrutura onde está implantado.

Web Services são inerentemente neutros de tecnologia de transporte. Além dos protocolos comuns da web como *Hypertext Transfer Protocol* (HTTP) e *Hypertext Transfer Protocol Secure* (HTTPS) é possível usar protocolos proprietários.

A Figura 3.3 apresenta modelo arquitetural dos *Web Services*. Como uma base para uma arquitetura orientada a serviços, o modelo de *Web Services* define como os *Web Services* são anunciados, descobertos, selecionados e usados.

Figura 3.3 - Modelo arquitetural dos Web Services

O modelo arquitetural dos *Web Services* consiste na descrição e publicação do serviço pelo Provedor de serviços no Provedor de registro de serviços e na busca da descrição e uso dos serviços pelo Consumidor de serviços.

Os *Web Services* são fundamentados basicamente em três tecnologias: *Web Services Description Language* (WSDL); *Universal Description, Discovery and Integration* (UDDI), e *Simple Object Access Protocol* (SOAP).

3.3.1.2 WSDL

Web Services Description Language (WSDL) é talvez o conjunto de metadados mais maduro para descrever *Web Services*. Metadados são importantes e fundamentais para obter baixo acoplamento. WSDL fornece uma definição abstrata da informação necessária para implantação e interação de serviços e é uma recomendação candidata da W3C (BOOTH; LIU, 2006). A W3C publica as recomendações candidatas para obter experiência de implementação.

WSDL é um formato XML para descrever serviços como um conjunto de pontos finais (*endpoints*) que operam em mensagens contendo informações orientadas a documento ou orientadas a procedimento. Para definir um *endpoint*, as operações e mensagens são descritas abstratamente e vinculadas a um protocolo de rede e a um formato de mensagem. O WSDL permite a descrição de *endpoints*, mensagens e protocolo de rede, que são usados para comunicação com o serviço.

Um documento WSDL tem duas partes: definição abstrata e descrição concreta. A definição abstrata define mensagens SOAP de uma maneira independente de plataforma e de linguagem. A descrição concreta define assuntos específicos, tal como serialização de objetos (operação de transformar um objeto em uma sequência de bytes).

WSDL fornece suporte para uma variedade de padrões de interação de mensagens. Por exemplo, ele suporta envio de mensagens com ou sem respostas e permite que serviços especifiquem outros serviços.

3.3.1.3 UDDI

As camadas de mensagem, descrição e transporte são fundamentais para permitir a comunicação de *Web Services* de maneira interoperável usando mensagens. Entretanto, para facilitar isso é necessário coletar e armazenar importantes metadados que descrevem esses serviços, por exemplo o WSDL. Os metadados devem estar de uma forma que possam ser descobertos e recuperados pelos usuários que procuram por serviços que resolvam um determinado problema de negócio. Tais metadados devem estar disponíveis em repositórios, nos quais diferentes organizações possam publicar os serviços que hospedam.

O *Universal Description, Discovery and Integration* (UDDI) é uma especificação amplamente conhecida para registro de *Web Services*. Ele define um serviço de agregação de metadados e protocolos específicos para consulta e atualização de repositórios de informações

de *Web Services (UDDI registry)*. Desenvolvedores de soluções podem consultar esses repositórios em tempo de projeto, para escolherem serviços compatíveis com seus requisitos. Depois de localizar um diretório em um *UDDI registry*, eles podem enviar uma série de requisições de consultas para obter informações detalhadas sobre *Web Services*, como localização dos provedores de serviços e referências das suas implementações. Com essas informações os desenvolvedores podem gerar aplicações que invocam os serviços escolhidos. Uma outra solução são as consultas dinâmicas a repositórios UDDI, que consiste da recuperação e invocação dos serviços escolhidos em tempo de execução.

Repositórios UDDI podem ser fornecidos de três maneiras:

- UDDI público – repositório que serve como recurso para *Web Services* baseados na Internet;
- UDDI intra-empresa – repositório UDDI interno e privado de uma determinada empresa;
- UDDI inter-empresa – repositório UDDI compartilhado entre parceiros de negócio específicos.

Em fevereiro de 2005 a *Organization for the Advancement of Structured Information Standards* (OASIS) anunciou a versão 3.0.2 do UDDI como um padrão (CLEMENT at al., 2004). OASIS é um consórcio internacional sem fins lucrativos que dirige o desenvolvimento, convergência e adoção de padrões para *e-business*. Este consórcio foi fundado em 1993 e tem mais de cinco mil participantes representando mais de 600 organizações e membros individuais em 100 países.

3.3.1.4 SOAP

Simple Object Access Protocol (SOAP) fornece um mecanismo simples e relativamente leve para troca de informação estruturada entre serviços. SOAP é projetado para reduzir o custo e a complexidade da integração de aplicações que são construídas em diferentes plataformas. SOAP tem sido revisado desde sua criação, foi recomendado em 2003 pela W3C, que tem gerenciado sua evolução (MITRA et al., 2003). A recomendação W3C é uma especificação ou conjunto de regras que, depois de ter obtido consenso, recebe o endosso do diretor e dos membros da W3C.

SOAP define um mecanismo de envelopamento extensível que estrutura a troca de mensagens entre *Web Services*. Uma mensagem SOAP é um documento XML que contém três elementos distintos: um envelope, um cabeçalho e um corpo. O envelope é o elemento raiz da mensagem SOAP. Ele contém um elemento cabeçalho opcional e um elemento corpo obrigatório. O elemento cabeçalho é um mecanismo genérico para adicionar características extensíveis ao SOAP. Cada elemento filho do cabeçalho é chamado de bloco de cabeçalho. SOAP define diversos atributos bem conhecidos que podem ser usados para indicar quem deveria lidar com um bloco de cabeçalho e se o processamento dele é opcional ou obrigatório. O elemento corpo é sempre o último elemento filho do envelope e é o contêiner para o conteúdo atual da mensagem (*payload*) que é destinado para o receptor final que processará a mensagem. SOAP não define nenhum bloco de cabeçalho interno e define somente um *payload*, que é o elemento *Fault* para reportar erros.

As mensagens SOAP são transmitidas de maneira unidirecional, do remetente ao receptor. Entretanto, múltiplas mensagens unidirecionais podem ser combinadas dentro de padrões mais sofisticados para troca de mensagens. Por exemplo, o padrão mais comum é o par de mensagens síncronas requisição/resposta. A flexibilidade das mensagens fornecidas por

SOAP permite que serviços se comuniquem usando uma variedade de padrões de troca de mensagens, para satisfazer a grande variedade de aplicações distribuídas. Diversos padrões têm provado sua utilidade em sistemas distribuídos. O uso de chamadas a procedimentos remotos, por exemplo, popularizou o padrão de troca de mensagens síncronas requisição/resposta. Quando o padrão requisição/resposta assíncrono é usado, a explícita correlação de mensagens torna-se obrigatória.

As mensagens podem ser distribuídas com base no conteúdo dos cabeçalhos e nos dados dentro do corpo da mensagem. Ferramentas desenvolvidas para o modelo de dados XML podem inspecionar e construir mensagens completas. Tais benefícios não são disponíveis em arquiteturas como *Common Object Request Broker Architecture* (CORBA), *Distributed Component Object Model* (DCOM) e *Remote Method Invocation* (RMI), onde os cabeçalhos dos protocolos possuem detalhes da infra-estrutura que não são vistos pela aplicação. Qualquer agente de software que envia ou recebe mensagens é chamado de nó SOAP. O nó que executa a transmissão inicial de uma mensagem é chamado de “remetente original”. O nó final que consome e processa a mensagem é chamado de “receptor final”. Qualquer nó que processa a mensagem entre o remetente original e o receptor final é chamado de “intermediário”. A coleção de nós intermediários atravessados pela mensagem e o receptor final são coletivamente referenciados como “caminho da mensagem”.

Para permitir que partes do caminho da mensagem sejam identificadas, cada nó participa em um ou mais papéis. A especificação do SOAP define dois papéis: “próximo” e “receptor final”. O “próximo” é um papel universal no qual todo nó SOAP assume, exceto o remetente original. O “receptor final” é o papel que o nó terminal assume, que tipicamente é a aplicação, ou em alguns casos, a infra-estrutura que executa o trabalho por trás da aplicação. O corpo de um envelope é sempre destinado ao “receptor final”. Os cabeçalhos podem ser destinados a intermediários ou ao “receptor final”.

A criação e suporte de padrões para *Web Services* é um componente crítico para sua efetiva funcionalidade e sucesso. Vários padrões estão sendo especificados e gerenciados pela W3C e OASIS. A seguir é apresentada uma pilha conceitual de padrões para *Web Services*.

3.3.1.5 Padrões para Web Services

A figura 3.4 mostra a pilha conceitual das tecnologias *Web Services*, que devem ser padronizadas para que a arquitetura orientada a serviços seja efetivamente implementada (KREGER, 2003). As três seções da pilha são fundamentais para os papéis da arquitetura orientada a serviços: interação, descrição e recuperação.

Figura 3.4 – Empilhamento conceitual das tecnologias Web Services

Fonte: (KREGER, 2003)

A base é a seção *Wire* que engloba as tecnologias requeridas para transporte de mensagens, de um serviço requisitante para o provedor de serviços, sobre a rede. A camada *Transport* é responsável pela conectividade de rede via TCP/IP (*Transmission Control*

Protocol/Internet Protocol). A camada *Packaging* define como a mensagem é codificada para ser transportada. A camada *Extensions* define o conjunto extensível de características expressas no cabeçalho das mensagens. Essas camadas devem suportar o conjunto de informações XML. SOAP e HTTP são os padrões mais amplamente suportados por essas camadas, apesar de outros padrões serem aceitos como FTP (*File Transfer Protocol*), por exemplo.

A próxima seção a ser padronizada é a *Description*. Todos os tipos de descrição nessa camada são especificados e expressos na linguagem XML Schema. As descrições *Interface* e *Implementation Description* definem os mecanismos de interação com um *Web Service*, o que inclui as operações e mensagens suportadas, como serializar e enviar as mensagens pela rede. A camada *Policy* será usada para descrever informações específicas de serviços, além de mecanismos, tais como taxonomia, requisitos de segurança, *timeouts*, custo e parâmetros de qualidade de serviço. A camada *Presentation* descreve como uma interface do usuário é gerada para esse serviço. Essas quatro camadas descrevem totalmente um serviço. As próximas duas camadas descrevem os relacionamentos e interações entre serviços. Serviços relacionados podem ser expressos na camada *Composition*. Esta camada inclui agrupamentos, composições, dependências e relacionamentos pai-filho. A camada *Orchestration* abrange pedidos de operações, fluxo de trabalho e processos de negócio. As duas camadas finais descrevem acordos entre o serviço requisitante e o provedor. A camada *Service Level Agreement* define o desempenho específico, uso, custo, métricas e pontos iniciais, esperados do serviço. A camada *Business Level Agreement* descreve um acordo contratual entre os dois parceiros de negócio que irão realizar negócios usando *Web Services*.

Discovery agencies, a próxima seção da pilha abrange as tecnologias que permitem que as descrições dos serviços sejam publicadas e descobertas (*Publication and*

Discovery), além de prover a inspeção de *sites* em busca de descrições de serviços já publicados (Inspection). A publicação pode ser definida como algum meio de tornar a descrição de um serviço disponível a um requisitante, por exemplo, um e-mail para o *registry*. O descobrimento pode variar de um simples acesso a um sistema de arquivos a um sofisticado mecanismo de busca de registros de serviços em tempo de desenvolvimento ou em tempo de execução.

As páginas por trás da pilha representam a infra-estrutura que interessa a cada camada da pilha. As páginas são, Security, Management e Quality of Service. As soluções para esses interesses podem ser diferentes para cada camada da pilha. Por exemplo, a segurança é a mais preocupante, no entanto, é a mais madura.

As tecnologias dessas camadas combinam para criar uma infra-estrutura completa para ser usada pelas aplicações *Business-to-Business* (B2B), soluções *Enterprise Application Integration* (EAI), bem como a emergente infra-estrutura de *grid* (WANG et al., 2005).

Kreger (2003), mostra o quão distante da padronização estão essas tecnologias. A padronização ocorre em quatro estágios: especificação, submissão, grupo de trabalho e aprovação. Atualmente diversos grupos de trabalhos do W3C e diversos comitês técnicos da OASIS vêm trabalhando nessas padronizações. Kreger (2003), apresenta uma excelente visão do atual estado dos trabalhos de padronização das tecnologias da pilha conceitual de *Web Services*.

3.3.2 OpenGIS

O OpenGIS ou OGC é um consórcio industrial com mais de 250 companhias, agências governamentais e universidades cujos membros trabalham num processo de consenso colaborativo para prover e aumentar interoperabilidade entre tecnologias envolvendo informações e localizações espaciais. A motivação do OGC é um mundo onde todos

disponibilizem informações e serviços geoespaciais em qualquer rede, aplicação ou plataforma. O OGC tem como missão prover uma interface de compartilhamento de informações espaciais e codificar especificações abertas e publicá-las para que fiquem disponíveis para uso global.

O OGC possui um Modelo de Referência (*OGC Reference Model - ORM*) que provê uma arquitetura de *framework* para o desenvolvimento dos trabalhos no OGC (PERCIVALL, 2003). Nesse modelo existe uma camada dedicada exclusivamente à *Web Services: OpenGIS Web Service Framework*.

3.3.2.1 OpenGIS Web Services Framework

O *OpenGIS Web Service Framework* identifica serviços, interfaces e protocolos de troca de dados que podem ser utilizados por qualquer aplicação (Figura 3.5).

Figura 3.5 – Framework OpenGIS Web Services

Fonte: (PERCIVALL, 2003)

Serviços OpenGIS são implementações de serviços que seguem as especificações de implementação do OpenGIS. Essas aplicações, chamadas aplicações OpenGIS, podem ser “plugadas” no *framework* e então serem executadas no mesmo ambiente operacional. Por implementarem interfaces comuns, as aplicações e serviços podem ser adicionados, modificados ou substituídos sem gerar impacto em outras aplicações.

Os diferentes tipos de serviços (Data Services, Portrayal Services e Processing Services) implementam as operações que manipulam informações geográficas propriamente ditas. Estas operações retornam informações nos formatos representados na Figura 3.5 pela caixa Encodings. Os serviços podem ser publicados nos Registry Services. Assim, os clientes podem procurar por serviços nos Registry Services e, uma vez encontrados, estabelecer uma conexão com estes serviços e usá-los.

A arquitetura que compreende os serviços e as aplicações OpenGIS (Application Services) pode ser melhor visualizada na Figura 3.6.

Figura 3.6 – Aplicações e Serviços OpenGIS

Fonte: (PERCIVALL, 2003)

Os serviços OpenGIS mais difundidos e implementados são:

- *Web Map Service* (WMS);
- *Web Feature Service* (WFS).

3.3.2.1.1 Web Map Service

O *Web Map Service* (WMS) é uma forma de produzir mapas a partir de dados georeferenciados e realizar consultas aos seus atributos. O mapa é a representação visual das informações geoespaciais, e não a informação propriamente dita. No WMS, a aplicação cliente não necessita implementar muitas operações. Um navegador web padrão pode fazer o papel de aplicação cliente. Basta que ele envie os parâmetros corretamente para o servidor, via protocolo HTTP. As informações trafegadas pela rede consistem basicamente de arquivos XML e imagens gráficas nos formatos GIF, JPEG, PNG, entre outros.

3.3.2.1.2 Web Feature Service

Assim como o WMS, o *Web Feature Service* (WFS) é uma forma de permitir a aplicações cliente visualizar informações geográficas na Internet. A grande diferença é a forma na qual estes dados geoespaciais são transportados do servidor para o cliente. Enquanto que, no WMS, os dados geoespaciais são transportados como uma imagem gráfica, no WFS os dados são transportados na forma de GML. O cliente WFS não é tão simples quanto o cliente WMS, pois precisa decodificar GML e transformar estes documentos XML em informações geográficas.

3.3.2.2 OpenGIS Geography Markup Language

O Modelo de Referência do OGC define diversas linguagens e métodos de codificação para representar semântica, sintaxe e esquemas de recursos de informações relacionadas a geoprocessamento e dados espaciais como *Sensor Model Language* (SML), linguagem para descrever e codificar sensores, *Styled Layer Descriptors* (SLD), linguagem para produzir mapas com estilos definidos pelo usuário, e *Geography Markup Language* (GML), linguagem para descrever e codificar informações geo-espaciais. Apenas a GML será apresentada em mais detalhes, pois foi a linguagem escolhida e usada para definir a PAML, uma linguagem para troca de dados entre aplicações para AP proposta neste trabalho de pesquisa.

A *Geography Markup Language* (GML) (COX et al, 2002), é uma especificação de implementação codificada em *XML Schema* para transporte e armazenamento de informações geográficas, incluindo aspectos geométricos e propriedades de objetos de interesse do domínio.

A expectativa é que ela tenha um grande impacto na habilidade de compartilhar e interligar conjuntos de dados geoespaciais, dadas suas características decorrentes de uma implementação em XML: independência de plataforma, separação de conteúdo e apresentação, extensibilidade, facilidade de entendimento, etc. A especificação da GML define o XML schema, os mecanismos e as convenções que objetivam:

- Prover uma estrutura aberta para a definição de esquemas de aplicações e objetos geoespaciais que suportem subconjuntos das capacidades declarativas de GML;
- Suportar a descrição de esquemas de aplicações geoespaciais para domínios específicos e comunidades de informação;

- Permitir a criação e manutenção de esquemas e dados geoespaciais distribuídos e relacionados;
- Aumentar a habilidade das organizações para compartilhar dados geoespaciais;
- Servir como ferramenta para a interoperabilidade de Sistemas de Informação Geográfica (SIG) de uma maneira incremental.

GML baseia-se no modelo abstrato da OpenGIS que define um objeto geográfico como sendo uma abstração de um fenômeno do mundo real, e que possui associada uma localização relativa sobre a superfície da Terra. Desta maneira, uma representação digital do mundo real pode ser entendida como um conjunto de objetos. O estado de um objeto é definido por um conjunto de atributos, onde cada propriedade é uma tripla (nome, tipo, valor). O número de propriedades de um objeto, com seu nome e tipo, é determinado pelo seu tipo. Objetos geoespaciais são aqueles que possuem propriedades do tipo geometria. Eles podem ser agrupados em uma coleção de objetos, que pode ser entendida, por si mesma, como um objeto, a qual pode possuir propriedades próprias.

GML segue o modelo de geometria definido no Modelo Abstrato do OpenGIS. Por exemplo, as tradicionais geometrias de 0,1 e 2 dimensões devem ser definidas com base em um Sistema de Referência Espacial (SRE) e são representadas por pontos, linhas e polígonos. A geometria de um objeto pode, ainda, ser representada por coleções de outros tipos de geometria: coleções homogêneas, como multipontos, multilinhas e multipolígonos ou heterogêneas.

A versão 2.1.2 da GML está de acordo com a recomendação para esquemas XML publicado pelo W3C (*World Wide Web Consortium*). GML também foi desenvolvido para ser consistente com a recomendação de espaços de nomes (*XML Namespaces*), que são usados

para distinguir as definições de feições e propriedades definidas para diferentes aplicações de domínios específicos.

GML 2.1.2 utiliza três esquemas base para codificar informações espaciais. O esquema *Geometry* (*geometry.xsd*), o esquema *Feature* (*features.xsd*) que inclui propriedades de feições comuns, e o esquema *Xlink* (*xlink.xsd*) que fornecem atributos Xlink para suportar funcionalidades de *links* para Internet. A Figura 3.7 apresenta um diagrama de pacotes em UML para mostrar os esquemas base da GML.

Figura 3.7 – Diagrama de pacotes em UML. Os pacotes representam de forma abstrata os esquemas base da GML

Fonte: (COX et al, 2002)

Um esquema XML define como um arquivo em XML pode ser construído, quais os elementos podem ser utilizados e quais as propriedades desses elementos. Assim, GML serve para descrever como documentos devem codificar dados espaciais.

Com o esquema *Geometry*, uma geometria pode ser descrita por pontos (*Point*), sequência de linhas (*LineString*), um anel fechado de linhas (*LinearRing*), e polígonos (*Polygon*). Geometrias são agrupadas em coleções (*GeometryCollection*). Estas coleções podem ser especializadas em coleções de polígonos (*MultiPolygon*), de linhas (*MultiLineString*) e de pontos (*MultiPoint*). Geometrias podem, ainda, ser associadas livremente, de acordo com a semântica da aplicação (*GeometryAssociation*). A elas acrescenta-se coordenadas e retângulos, para melhor definir um objeto geográfico. A Figura 3.8 mostra uma representação em UML do esquema *Feature*.

Figura 3.8 - Representação em UML do esquema *Feature*

Fonte: (COX et al, 2002)

Tanto o esquema *Feature* quanto o *Geometry* definem tipos e elementos abstratos e concretos. Esquemas escritos por desenvolvedores podem definir elementos e/ou tipos para nomear e distinguir feições e coleções de feições. No capítulo quatro será mostrado como a PAML criou novos elementos sem perder a compatibilidade com a GML.

3.3.3 JCP

O *Java Community Process* (JCP) (JCP, 2006), foi estabelecido em 1998 e é um processo formalizado que permite às partes interessadas serem envolvidas na definição de futuras versões e características da plataforma Java.

O processo JCP envolve o uso de *Java Specification Requests* (JSR), que são documentos formais que descrevem propostas de especificações e tecnologias a serem adicionadas à plataforma Java. Revisões formais de JSRs são conduzidas antes que se tornem versões finais e são votadas em um comitê executivo chamado *JCP Executive Committee*. A JSR final provê uma implementação de referência que é uma implementação livre na forma de código livre e um *kit* de compatibilidade de tecnologia para verificar a especificação. JSRs finais incluem, J2EE, J2ME, *Servlet*, *Java ServerPages* (JSP), *Enterprise JavaBeans* (EJB), *Portlets* e *Java Message Service* (JMS) (JMS, 2005).

3.3.3.1 Middleware orientados a mensagens

O termo *middleware* representa uma camada intermediária entre o sistema operacional e as aplicações distribuídas, tendo como objetivo abstrair a heterogeneidade existente da comunicação distribuída.

Message-Oriented Middleware (MOM) é um *middleware* orientado a mensagem que funciona com base na troca de mensagens entre programas de maneira assíncrona, promovendo a integração e possibilitando que elementos de negócio sejam trocados de forma confiável. Algumas das soluções MOM *open source* incluem: *Java Open Reliable Asynchronous Messaging* (JORAM), OpenJMS e JBossMQ. Essas soluções implementam o conjunto de interfaces JMS.

Serviços de mensagens permitem computação distribuída fracamente acoplada. Um componente envia uma mensagem para uma destinação e o receptor pode recuperar a mensagem dessa destinação. O remetente e o receptor não precisam estar disponíveis no mesmo instante para se comunicarem e nem precisam conhecer qualquer coisa um do outro. Os dois precisam conhecer somente qual o formato da mensagem e qual destinação usar. Essa é a característica que distingue o serviço de mensagens das tecnologias de forte acoplamento,

como *Remote Method Invocation* (RMI), que requer que uma aplicação conheça os métodos da aplicação remota.

Um sistema de mensagens deve fornecer as seguintes partes:

- **Provedor**, sistema de mensagens que fornece características de controle e administração;
- **Cliente**, programa ou componente que produz ou consome mensagens (produtor ou consumidor);
- **Mensagem**, objeto que comunica informações entre clientes.

A maioria dos sistemas de mensagens suporta duas formas de uso, chamadas de domínios *Point-to-Point* e *Publish/Subscribe*.

No domínio *Point-to-Point*, também chamado de PTP, existe basicamente a presença de três elementos: fila (*Queue*), produtor e consumidor. Cada mensagem é enviada para uma fila, de onde é consumida. O provedor, de acordo com os parâmetros passados pela mensagem, pode armazená-la até que sejam consumidas ou seu tempo expire. A Figura 3.9 mostra o domínio PTP.

As características desse domínio são:

- Cada mensagem tem apenas um consumidor;
- O produtor e o consumidor não mantêm relação de tempo entre eles;
- O consumidor reconhece quando a mensagem é processada corretamente.

Figura 3.9 – Domínio Point-to-Point

No modelo *Publish/Subscribe* (também chamado *pub/sub*) mostrado na Figura 3.10, as mensagens são direcionadas para um tópico (Topic) e então entregues para os vários clientes inscritos naquele tópico. Por *default* as mensagens só são entregues aos clientes que estão ativos naquele instante. Para resolver esse problema devem ser criadas mensagens do tipo *Durable*.

Figura 3.10 – Domínio Publish/Subscribe

A leitura dessas mensagens pode ser feita de duas formas: síncrona e assíncrona. Na forma síncrona a aplicação espera pela mensagem. Na forma assíncrona um *listener* (ouvinte) fica aguardando o aviso da chegada da mensagem para então ativar o processo de leitura.

3.3.3.2 Portlet

A complexidade e o alto custo dos modernos sistemas de informação têm motivado muitas empresas a investirem em portais como mecanismo de gerenciamento de informações de maneira coesa e estruturada.

Os portais oferecem muitas vantagens sobre outros tipos de aplicações, por exemplo:

- fornecem um único ponto de entrada para os usuários, produtores, consultores e parceiros;
- podem *acessar Web Services* direta e transparentemente;
- são altamente flexíveis, podendo atuar na forma de *B2E intranets*, *B2B extranets* ou *B2C internets*; e diversos portais podem ser combinados para formar uma rede de portais, estendendo as capacidades de negócio.

Com o crescimento do número de portais, várias interfaces de programação de aplicações (API) para componentes de portais foram criadas. Esta variedade de APIs gerou problemas de incompatibilidade. Para resolver esses problemas foi criada a especificação para *portlets*, *Portlet Specification e API 1.0* (ABDELNUR; HEPPER, 2003).

A API para Portlet 1.0 é baseada na plataforma *Java 2 Platform, Enterprise Edition, version 1.3* (J2EE). Essa especificação garante que os *portlets* construídos com essa API possam ser executados em qualquer ambiente de execução em conformidade com a especificação J2EE.

Segundo a especificação, portal é uma aplicação baseada na web que fornece personalização, *single sign-on*, agregação de conteúdo de diferentes fontes e hospedagem da camada de apresentação dos sistemas de informação. Agregação é o processo de integrar conteúdos de diferentes fontes dentro de uma página web. Um portal pode ter características sofisticadas de personalização para fornecer conteúdo personalizado para cada usuário. A página de um portal agrega diversos *portlets* e pode conter também áreas de navegação e *banners*.

Portlet é um componente web baseado em Java que processa requisições e gera conteúdo dinâmico. O conteúdo gerado por um *portlet* é chamado de fragmento, um pedaço de marcação, por exemplo, de HTML, XHTML, ou WML (*Wireless Markup Language*). Um

fragmento pode ser agregado a outros fragmentos para formar uma página completa. Um *portlet* pode agregar conteúdo de outros *portlets* para formar a página do portal. Um *portlet* contêiner gerencia o ciclo de vida do *portlet*.

O *portlet* contêiner provê o ambiente de execução de *portlets* e gerencia seu ciclo de vida. O contêiner recebe as requisições do portal e delega as requisições aos respectivos *portlets*. A Figura 3.11 mostra os elementos de uma página de portal.

Figura 3.11 – Elementos de uma página de portal

Fonte: (ABDELNUR; HEPPER, 2003)

A Figura 3.12 mostra o fluxo de criação de uma página no portal.

Figura 3.12 – Fluxo de criação da página de portal

Fonte: (ABDELNUR; HEPPEL, 2003)

Os *portlet* são executados dentro de um *Portlet Container*. O *Portlet Container* recebe o conteúdo gerado pelos *portlets* e manipula o conteúdo para criar a página do portal. O *Portal Server* cria a página do portal com o conteúdo gerado pelos *portlets* e envia para o *Dispositivo cliente*, por exemplo, um *web browser*, onde é mostrado ao usuário.

Os *portlets* compartilham muitas similaridades com *servlets* (COWARD, 2001):

- *Portlets* são componentes web baseados na tecnologia Java;
- *Portlets* são gerenciados por um contêiner especializado;
- *Portlets* geram conteúdos dinâmicos;
- O ciclo de vida dos *portlets* é gerenciado por um contêiner;
- *Portlets* interagem com clientes web através do paradigma *request/response*;

Os *portlets* diferem dos *servlets* nos seguintes aspectos:

- *Portlets* geram apenas fragmentos de marcação, não documentos inteiros. O portal agrega fragmentos de marcação *portlet* dentro de uma página completa do portal;
- *Portlets* não são diretamente vinculados a uma URL (*Uniform Resource Locator*);
- Clientes web interagem com *portlets* através de um sistema de portal;
- *Portlets* têm modos predefinidos e estados de janelas que indicam a função que o *portlet* está executando;
- *Portlets* podem existir muitas vezes no portal;

Os *portlets* têm funcionalidades extras não fornecidas pelos *servlets*:

- *Portlets* têm meios de acessar e armazenar configurações persistentes e customização de dados;
- *Portlets* têm acesso a informação de perfis de usuários;
- *Portlets* têm funções de reescrita de URLs para criar *hyperlinks* dentro de seus conteúdos, que permite que o *Portal Server* crie *links* e ações em fragmentos de páginas.

Os *portlets* não têm acesso a algumas funcionalidades fornecidas pelos *servlets*:

- Configurar a codificação do conjunto de caracteres do *response*;
- Configurar o cabeçalho HTTP no *response*;
- Acesso à URL da requisição do cliente ao portal.

Por causa dessas diferenças, foi decidido que os *portlets* precisariam ser um novo componente. Desse modo, um *portlet* não é um *servlet*. Isso permite definir uma interface clara e comportamento para os *portlets*.

Para permitir o uso tanto quanto possível da infra-estrutura existente dos *servlets*, a especificação dos *Portlets* tentou aproveitar o máximo possível das funcionalidades providas pelo *servlets*. Muitos conceitos e partes da API do *Portlet* foram modelados depois da API do *Servlet*.

Portlets, *servlets* e JSPs são empacotados em uma aplicação web estendida chamada *portlet application*. *Portlets*, *servlets* e JSPs dentro do mesmo *portlet application* compartilham, *classloader*, *application context* e *session*.

Portlet application é uma aplicação web, como definido na *Servlet Specification 2.3* (COWARD, 2001), contendo *portlets* e um descritor para implantação, além de *servlets*, JSP, páginas HTML, classes e outros recursos normalmente encontrados em uma aplicação web. Um *portlet application* é portátil, podendo ser executado em múltiplas implementações de *portlet containers*.

3.3.3.3 J2EE

Java 2 Enterprise Edition (J2EE) é um padrão dinâmico para a produção de aplicações corporativas seguras, escaláveis e altamente disponíveis. O padrão define quais serviços devem ser fornecidos pelos servidores que suportam J2EE, os quais são aprovados pelo JCP através de JSR. A especificação J2EE não define como organizar a arquitetura lógica em máquinas físicas, ambientes, etc. No entanto, ela fornece diversos tipos de componentes e para cada um deve haver um contêiner, que devem ser implementados pelos fornecedores de servidores de aplicação. A Figura 3.13 mostra a arquitetura lógica do J2EE com seus elementos arquiteturais.

Figura 3.13 – Arquitetura lógica do J2EE

Fonte: (SHANNON, 2003)

Os relacionamentos dos elementos arquiteturais mostrados na Figura 3.13 são relacionamentos lógicos e não necessariamente implicam na distribuição física em máquinas separadas, processos ou máquinas virtuais.

Os contêineres, denotados pelos retângulos separados, são ambientes de execução J2EE que fornecem serviços necessários aos componentes representados na metade superior do retângulo. Os serviços fornecidos são representados pelas caixas na metade inferior do retângulo. Por exemplo, o Application Client Container provê a API *Java Message Service* (JMS) para o Application Client. Todos os serviços são detalhados na especificação J2EE (SHANNON, 2003).

As setas representam acessos requeridos às outras partes da plataforma J2EE. O Application Client Container provê aos Application Clients acesso

direto ao Database através da API *Java Database Connectivity* (JDBC). Acessos similares ao DataBase são fornecidos pelo Web Container através de páginas JSP e Servlets e pelo EJB Container através dos *Enterprise JavaBeans* (EJB). As APIs da *Java 2 Platform, Standard Edition* (J2SE), são suportadas pelos ambientes de execução J2SE de cada um dos quatro tipos de componentes de aplicação, Application Client, Applet, JSP e servlet e EJB.

A Figura 3.14 mostra as possibilidades de integração, através das APIs J2EE, em cada contêiner.

Figura 3.14 – Facilidades de interoperabilidade da plataforma J2EE

Fonte: (SHANNON, 2003)

Muitas das APIs descritas acima fornecem interoperabilidade com componentes que não são parte da plataforma J2EE, tal como serviços web ou CORBA. As direções das setas indicam os relacionamentos dos componentes.

Uma boa prática seria desenvolver as aplicações, componentes e serviços usando os padrões da especificação J2EE, dessa forma pode-se conseguir reusabilidade, segurança, escalabilidade, disponibilidade e portabilidade, entre outras qualidades de software necessárias para a evolução de sistemas de informação para AP.

3.4 Considerações finais

Apesar do fato de algumas importantes camadas da pilha de *Web Services* ainda precisarem ser padronizadas e produzidas, os *Web Services* no seu estado atual têm se mostrado como uma excelente tecnologia para integração de aplicações. *Web Services* podem integrar aplicações que são executadas em plataformas diferentes, permite que informações de um banco de dados de uma aplicação possam ser disponibilizados para outras aplicações, permite que serviços internos sejam disponibilizados sobre a Internet e podem ser usados entre dois parceiros de negócio através da definição de contratos de negócio com uma linguagem padrão para troca de dados.

Tecnologias inovadoras como *Web Map Service*, *Web Feature Service* e *Web Service Provisioning* já estão sendo trabalhados e padronizados pelo OGC e OASIS. *Web Map Service* e *Web Feature Service* são *Web Services* com funcionalidades para executarem tarefas de geoprocessamento como geração de mapas, conversão de coordenadas, buscas espaciais, etc. Diversas implementações *open source* desses serviços como Deegree (DEEGREE, 2006) e MapServer (MAPSERVER, 2006) estão disponíveis livremente na web. *Web Service Provisioning* é uma peça chave para operacionalizar os *Web Services* “pague pelo uso”. Ele é uma mistura complexa de serviços de autenticação, registro, medição, faturamento e

operações de gerenciamento que controla o comportamento de um *Web Service* durante seu uso interno ou entre parceiros. Esses serviços são desenvolvidos e utilizados como utilitários ou componentes do tipo “pague pelo uso”, um exemplo é o serviço de histórico de veículos VIN (*Vehicle Identification Number*) que retorna o registro e a história de um veículo em uso nos EUA e na Europa.

Apesar das tecnologias apresentadas neste capítulo tratarem especificamente do uso de Java, uma das plataformas de software livre mais utilizadas no desenvolvimento de sistemas multiplataforma, o uso de outras tecnologias que implementam os mesmos conceitos é perfeitamente possível.

Desse modo, o uso dessas tecnologias agregada de um bom projeto de arquitetura pode trazer grandes contribuições para a solução dos atuais problemas dos sistemas de informação para agricultura de precisão como pouca flexibilidade e interoperabilidade e alto custo.

O próximo capítulo apresenta a proposta deste trabalho, que consiste da especificação de uma infra-estrutura de desenvolvimento de sistemas de informação para agricultura de precisão, fundamentada nos requisitos levantados no capítulo 2 e nos padrões abertos e tecnologias relacionadas estudadas neste capítulo.

4 INFRA-ESTRUTURA DE DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO ORIENTADOS A SERVIÇOS DISTRIBUÍDOS PARA AGRICULTURA DE PRECISÃO

4.1 Introdução

As experiências têm mostrado que a prática da AP exige integração de diferentes sistemas de informação, o que obriga o usuário a aprender a trabalhar com muitos pacotes de software, cada um com diferentes funcionalidades, diferentes interfaces de usuário e diferentes formatos de dados, que normalmente são incompatíveis ou necessitam de conversão. Esses, na sua maioria, são sistemas monolíticos que possuem tarefas específicas como mapeamento de produtividade, amostragem de solo e recomendação de aplicação de fertilizantes. Muitos são fornecidos como pacotes de software por fabricantes de equipamentos como monitores de produtividade e coletores de amostras de solo, todos com funcionalidades limitadas e capacidade de integração restrita. Alguns pacotes de software possuem SIGs proprietários integrados para análises espaciais, enquanto outros necessitam de muitas transformações dos dados de entrada e saída, para que possam ser processados por SIGs externos.

Com objetivo de resolver ou minimizar esses problemas e atender as atuais necessidades dos sistemas de informação para agricultura de precisão este trabalho de pesquisa foi realizado. Com base nos resultados apresentados no capítulo 2 (sistemas de informação para agricultura de precisão), no capítulo 3 (padrões abertos e tecnologias relacionadas) e no capítulo 5 (resultado dos experimentos com o protótipo), uma infraestrutura de desenvolvimento de sistemas de informação foi proposta. Essa infra-estrutura

pretende apoiar o engenheiro de software no desenvolvimento de sistemas de informação para agricultura de precisão e áreas correlatas, através do uso de padrões abertos e plataformas de software livre. A infra-estrutura é constituída basicamente dos seguintes componentes:

- Uma arquitetura de referência para sistemas de informação para agricultura de precisão;
- Uma linguagem padrão para troca de dados entre serviços agrícolas;
- Um barramento para conexão de serviços agrícolas;
- Um provedor de serviços geoespaciais;
- Um portal para serviços agrícolas;

Em seguida são apresentados em detalhes cada componente da infra-estrutura. A descrição dos componentes é seguida de uma implementação que serviu como experimento para validação das propostas. Apesar das implementações terem sido realizadas com base na linguagem Java, nada impede que os conceitos apresentados sejam implementados em outras linguagens.

4.2 A arquitetura de referência para sistemas de informação para AP

A *arquitetura de referência* para sistemas de informação para AP proposta neste trabalho de pesquisa está fundamentada no MOSAICo (Modelo de Objetos para Sistemas Abertos de Informação de Campo para agricultura de precisão) (*modelo de referência*) e nos estilos arquiteturais cliente-servidor em camadas e orientado a serviços (*estilo arquitetural*).

A Figura 4.1 apresenta o relacionamento do modelo de referência, estilo arquitetural e arquitetura de referência, para definição de arquiteturas de software para sistemas de informação para AP.

Figura 4.1 - Relacionamento do modelo de referência, estilo arquitetural e arquitetura de referência, para definição da arquitetura de um sistema de informação para AP

A visão geral da arquitetura de referência é apresentada na Figura 4.2.

Figura 4.2 – Visão geral da arquitetura de referência. O barramento de serviços recebe as requisições realizadas pelas aplicações e invoca os serviços apropriados, agrícolas ou geoespaciais. Depois que o processamento é encerrado armazena as respostas no repositório de resultados e notifica o cliente

Clientes como *web browsers*, *WAP-phones*, *paggers*, acessam aplicações através do Portal que centraliza e disponibiliza uma interface simples e padronizada aos usuários. As aplicações interagem com o Barramento de serviços, o qual é responsável pela invocação dos serviços geograficamente distribuídos. O barramento deve garantir a entrega das requisições e quando necessário pode aplicar transformações nos dados antes de se comunicar com os serviços. O Barramento de serviços usa um repositório (Repositório de resultados) para armazenar resultados de serviços que exijam grande esforço de processamento e desse modo são invocados de forma assíncrona. Tanto os Serviços Agrícolas quanto os Serviços Geoespaciais são *Web Services* hospedados em servidores de aplicação distribuídos pela Internet.

A visão lógica da arquitetura em camadas proposta pode ser visualizada na Figura 4.3.

Figura 4.3 – Arquitetura em camadas proposta para sistemas de informação para AP

A organização em camadas aumenta a reusabilidade e a modificabilidade dos sistemas, porque as camadas conhecem somente as camadas adjacentes através de suas interfaces. A

dependência é sempre de cima para baixo, sendo a camada superior sempre cliente da camada inferior.

Camada Cliente: Nessa camada estão os dispositivos de acesso às aplicações como *web browsers*, *WAP phones*, *paggers*, etc. Através desses dispositivos clientes o usuário fornece dados para processamento e recebe respostas.

Camada de Apresentação: Essa camada disponibiliza diversos tipos de interfaces do usuário, para entrada de dados, através de um portal. Este portal é um ambiente para disponibilização e gerenciamento de aplicações web de forma centralizada, capaz de apresentar uma quantidade muito grande de informação de forma concisa e centralizada criando um único ponto de entrada para os produtores, fornecedores e parceiros e facilitando o acesso às diversas aplicações de um processo de negócio.

Camada de Integração: Essa camada é responsável pela integração das aplicações com serviços, internos ou externos, necessários para um processo de negócio completo. A camada consiste de um barramento de serviços e um repositório de resultados.

O barramento de serviços contém um roteador que identifica o serviço especificado na requisição do cliente e invoca os serviços requisitados passando os dados de entrada. Este roteador usa de manipuladores de transporte para invocar serviços através de conexões síncronas ou assíncronas que suportam protocolos eficientes para transferência de arquivos pela rede, porque alguns serviços podem requerer grandes movimentações de dados. Esses manipuladores podem aplicar transformações nos dados antes ou depois da invocação dos serviços e são responsáveis em avisar o cliente do fim do processamento. O repositório de resultados contém mecanismos de armazenamento persistente de dados para manter os resultados das tarefas realizadas pelos serviços para reuso em consultas futuras ou um armazenamento temporário que mantém os resultados até a sua recuperação pelo cliente.

Camada de Negócio: Nessa camada residem os serviços, agrícolas, geoespaciais e outros necessários para a execução de um processo de negócio completo. Alguns serviços podem ser criados para acesso a sistemas legados ou para integração de bases heterogêneas.

Os serviços agrícolas são serviços que realizam tarefas completas de um processo de negócio, por exemplo, simulação de crescimento de planta, processamento de filtragem de dados de produtividade, captura de dados meteorológicos, etc. Os serviços agrícolas são construídos sobre o paradigma da computação orientada a serviços (HUHNS; SINGH, 2005), que implementam desde funcionalidades específicas até processos de negócios completos.

Os serviços geoespaciais são serviços de manipulação de dados espaciais básicos de um Sistema de Informação Geográfica (SIG). Os dados geométricos e alfanuméricos são codificados em GML e armazenados em banco de dados com extensão para tipos de dados geoespaciais. Esses serviços são baseados nos serviços da OGC, *Web Feature Service* (WFS) e *Web Map Service* (WMS) (OGC, 2006).

A comunicação, tanto dos serviços agrícolas quanto dos serviços geoespaciais, é baseada na troca de documentos XML. Esses documentos XML são criados e validados pelo esquema PAML (*Precision Agriculture Markup Language*).

Camada de Recursos: Essa camada consiste de recursos como bancos de dados, sistemas legados, sistemas especialistas e outros dispositivos necessários para um sistema de informação para agricultura de precisão.

4.3 A linguagem padrão para troca de dados entre serviços agrícolas

Precision Agriculture Markup Language (PAML) é uma linguagem padrão para troca de dados estruturados entre aplicações baseada na *eXtensible Markup Language* (XML) (BRAY et al., 2004). A definição do vocabulário dessa linguagem foi baseada no modelo de

objetos para sistemas abertos de informação de campo (MOSAICo) (SARAIVA, 1998), no vocabulário controlado de terminologias da agricultura e domínios relacionados, AGROVOC (AGROVOC, 2006) e no modelo da *Geography Markup Language* (GML) (COX et al., 2002). GML é uma especificação em XML para transporte e armazenamento de informação geográfica, incluindo tanto propriedades espaciais quanto não espaciais de fenômenos geográficos. XML permite declarações estruturadas de dados mais precisas com conteúdo mais significativo, que permitem buscas através de múltiplas plataformas e manipulação e visualização de dados via Internet.

A estrutura da PAML é definida por um esquema XML. *XML Schemas* (FALLSIDE; WALMSLEY, 2004), descrevem estruturas de documentos XML, definindo, por exemplo, os elementos e atributos que podem aparecer em um documento, os tipos de dados e valores padrão para os elementos e atributos, quais elementos são elementos filhos, etc. Com *XML Schema* é fácil descrever conteúdo de documentos, definir restrições e formatos de dados, validar dados e converter dados entre diferentes tipos. O esquema da PAML é uma extensão do esquema da GML.

A especificação GML (COX et al., 2002) define a sintaxe do seu esquema. A GML fornece uma estrutura aberta para a definição de esquemas de aplicações e objetos geográficos, suporta a descrição de esquemas de aplicações geográficas para domínios específicos, permite a criação e manutenção de esquemas e dados geográficos distribuídos e relacionados e fornece interoperabilidade entre Sistemas de Informação Geográfica (SIG).

A Figura 4.4 mostra os diagrama de pacotes em UML dos esquemas que compõem os *frameworks* PAML e GML. Os pacotes destacados são os pacotes utilizados e testados neste trabalho.

Figura 4.4 – Diagrama de dependência de pacotes em UML dos esquemas PAML e GML

Os esquemas base da GML, *Geometry*, *Feature* e *Xlink* podem ser vistos como os componentes de um *framework* de aplicação (*Framework GML*) para desenvolvimento de esquemas que pertencem a um determinado domínio, por exemplo, agricultura de precisão (*Framework PAML*). Os esquemas *FieldData*, *AgriculturalData*, *Specialist*, *TaskAssistent*, *GeoInformation* e *SimulationModel*, que correspondem aos

subsistemas do MOSAICo, mais o `CustomType`, que são tipos de objetos e restrições reutilizáveis, formam o `Framework PAML`.

A Figura 4.5 mostra o diagrama de classes do esquema `FieldData`, que estende as classes abstratas do esquema `Feature` da GML. Esta relação de herança garante compatibilidade com a GML.

Figura 4.5 – Diagrama de classes do esquema `FieldData`, um framework de aplicação baseado no framework GML

Todas as classes que estendem de `AbstractFeature` são feições da PAML. Essas classes fazem parte do módulo Dados de Campo do MOSAICo. A classe `FieldDataModel` e `FieldDataMember` representam as coleções de feições da PAML. As classes destacadas em cinza são do esquema `Feature` da GML. A Figura 4.6 mostra um trecho do código do esquema `FieldData` (`FieldData.xsd`). Esse esquema corresponde à

implementação do subsistema “Dados de Campo” do MOSAICo, representado pelo pacote FieldData mostrado na Figura 4.4.

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  targetNamespace="http://www.padis.org"
  xmlns="http://www.padis.org"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  version="1.0.0" xml:lang="en">

...
<!-- =====
      SEÇÃO 1 - INCLUSÃO DE ESQUEMAS
===== -->

<!-- Include constructs from the FieldData CustomType schema -->
<xsd:include schemaLocation="CustomType.xsd"/>
<!-- Import constructs from the GML Feature e Geometry schemas -->
<xsd:import namespace="http://www.opengis.net/gml"
  schemaLocation="../../../schema/GML-2.1.2/feature.xsd"/>
<xsd:import namespace="http://www.w3.org/1999/xlink" />

<!-- =====
      SEÇÃO 2 - DECLARAÇÃO DE ELEMENTOS GLOBAIS
===== -->

...
<!-- Abstract FieldData Members -->
<xsd:element name="Farm" type="FarmType"
  substitutionGroup="_FieldDataFeature"/>

...
<!-- Abstract Sample Features -->
<xsd:element name="YieldSample" type="YieldSampleType"
  substitutionGroup="_SampleFeature"/>

...
<!-- =====
      SEÇÃO 3 - DEFINIÇÃO DE TIPOS
===== -->

...
<xsd:complexType name="FarmType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence>
        <xsd:element ref="id"/>
        <xsd:element ref="name"/>
        <xsd:element name="phoneNumber"
          type="PhoneNumberType"
          minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element ref="address"/>
        <xsd:element ref="Farmer"/>
        <xsd:element ref="boundary"/>
        <xsd:element ref="Equipment"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

```

        <xsd:element ref="Field"
                    minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

...
<xsd:complexType name="YieldSampleType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence>
        <xsd:element ref="id"/>
        <xsd:element ref="collectDate"/>
        <xsd:element name="flow" type="xsd:decimal"/>
        <xsd:element name="cycles" type="xsd:decimal"/>
        <xsd:element name="distance" type="xsd:decimal"/>
        <xsd:element name="swath" type="xsd:decimal"/>
        <xsd:element name="moisture" type="xsd:decimal"/>
        <xsd:element name="field" type="xsd:decimal"/>
        <xsd:element name="crop" type="xsd:decimal"/>
        <xsd:element name="altitude" type="xsd:decimal"/>
        <xsd:element name="dryYield" type="xsd:decimal"/>
        <xsd:element name="easting" type="xsd:decimal"/>
        <xsd:element name="northing" type="xsd:decimal"/>
        <xsd:element name="pointDistance" type="xsd:decimal"/>
        <xsd:element name="northDistance" type="xsd:decimal"/>
        <xsd:element name="eastDistance" type="xsd:decimal"/>
        <xsd:element ref="gml:Point"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

...
</xsd:schema>

```

Figura 4.6 – Esquema FieldData.xsd

Na SEÇÃO 1 são declarados os esquemas importados como o Geometry e o Feature da GML e o CustomType da PAML. Na SEÇÃO 2 são declarados os elementos e seus respectivos tipos, por exemplo, os elementos Farm e YieldSample dos tipos FarmType e YieldSampleType respectivamente. Os tipos FarmType e YieldSampleType são definidos na SEÇÃO 3. Esses tipos estendem o tipo AbstractFeatureType da GML, portanto, são compatíveis com essa linguagem e podem usar seus elementos. Por exemplo, o tipo YieldSampleType usa o elemento Point.

A Figura 4.7 mostra um trecho do documento XML construído conforme definido no esquema `FieldData.xsd` apresentado na Figura 4.6, que corresponde a uma instância desse esquema com dados do elemento `YieldSample`.

```
<?xml version="1.0" encoding="UTF-8"?>
<FieldDataModel xmlns="http://www.padis.org"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.padis.org FieldData.xsd">

  <!-- AbstractFeatureType -->
  <gml:name>FieldDataModel-YieldSamples</gml:name>
  <fieldDataMember>
  ...
    <YieldSample>
      <id>1</id>
      <collectDate>2005-11-27</collectDate>
      <flow>60.0</flow>
      <cycles>60.0</cycles>
      <distance>60.0</distance>
      <swath>60.0</swath>
      <moisture>60.0</moisture>
      <field>60.0</field>
      <crop>60.0</crop>
      <altitude>60.0</altitude>
      <dryYield>60.0</dryYield>
      <easting>60.0</easting>
      <northing>60.0</northing>
      <pointDistance>60.0</pointDistance>
      <northDistance>60.0</northDistance>
      <eastDistance>60.0</eastDistance>
      <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
        <gml:coord>
          <gml:X>20.0</gml:X>
          <gml:Y>5.0</gml:Y>
        </gml:coord>
      </gml:Point>
    </YieldSample>
  ...
    <dateCreated>2005-11</dateCreated>
</FieldDataModel>
```

Figura 4.7 – A feição `YieldSample` em um documento XML válido, conforme esquema `FieldData.xsd` (`yieldsamples.xml`)

A PAML define a estrutura padrão para representação, transporte e armazenamento de dados na infra-estrutura proposta. Por estender o *framework GML*, a PAML torna-se um *framework* de aplicação, que também pode ser estendido, sem perder a compatibilidade com a

especificação GML, o que permite sua evolução sem impactar as aplicações que a utilizam como padrão para troca de dados.

4.4 O barramento para conexão de serviços agrícolas

O AgriBUS é um barramento que possibilita a conexão e interoperabilidade de componentes e aplicações. O AgriBUS fornece uma infra-estrutura de conectividade baseada em *Web Services* e *Message Services* que objetiva principalmente servir de *middleware* entre as aplicações e os serviços agrícolas e geoespaciais, provendo segurança, confiabilidade, desempenho e escalabilidade. O AgriBUS é baseado no padrão de integração *Enterprise Service Bus* (ESB) (PASLEY, 2005). ESB é uma nova tecnologia de *middleware* que fornece as características requeridas por SOA. ESB fornece um ambiente de hospedagem para *Web Services*, permitindo a conexão e a exposição de serviços sobre protocolos de transporte baseados em padrões da Internet, como HTTP e FTP. O ESB fornece características que são essenciais para a implantação de serviços como gerenciamento, transformação e validação de mensagens. A Figura 4.8 mostra as camadas de um ESB.

Figura 4.8 – Camadas de um ESB

Na camada *Infra-estrutura*, estão os ativos de software existentes, incluindo, sistemas legados, banco de dados e outras fontes de informações, os quais são conectados

através de adaptadores de serviços. Na camada Barramento de Serviços estão os adaptadores para expor os componentes existentes e os serviços que fornecem conectividade, transformação e transporte. Na camada de Serviços de Negócio estão os *Web Services* e composições desses, orquestrados pelas linguagens de descrição de Processos de Negócio. As descrições de processos de negócio formam a quarta camada. Os serviços são conectados ao barramento através das descrições de suas interfaces. Às conexões dos serviços ao barramento, estão associadas políticas de segurança, disponibilidade, manipulação de erros, mensagens, *logging*, etc.

Os objetivos do AgriBUS incluem:

- Permitir que aplicações heterogêneas, desenvolvidas por equipes diferentes, interajam independentemente de terem sido construídas com a mesma tecnologia;
- Possuir uma camada de conectividade que otimize as interações entre consumidores e provedores de serviços;
- Ser capaz de responder aos contextos orientados a serviço, mensagem e eventos.

O AgriBUS combina *Web Services* com serviços de mensagens para obter algumas propriedades arquiteturais importantes para aplicações críticas como escalabilidade, confiabilidade e reusabilidade. Serviços de mensagem permitem computação distribuída fracamente acoplada. Por exemplo, uma mensagem enviada para uma destinação (fila de mensagens) pode ser recuperada pelo receptor, sem que o remetente e o receptor estejam disponíveis no mesmo instante e se conheçam. Os dois precisam conhecer somente qual o formato da mensagem e qual destinação usar. Essa é a característica que distingue o serviço

de mensagens das tecnologias de forte acoplamento, como *Remote Method Invocation* (RMI), que requer que uma aplicação conheça os métodos da aplicação remota.

A primeira geração de aplicações orientadas a serviços referenciava diretamente os clientes dos serviços, conforme Figura 4.9. Esta arquitetura permite que a aplicação se comunique com o serviço remoto, mas não implementa confiabilidade de dados, escalabilidade e reuso da lógica do cliente do serviço. A adição de serviços de mensagens cria uma segunda geração de arquitetura de aplicações orientadas a serviço, como mostra a Figura 4.10. O serviço de mensagens retira da aplicação, a tarefa de se comunicar com o *Web Service* diretamente, comunicando-se através de um adaptador (CARBONE, 2005).

Figura 4.9 - Arquitetura fortemente acoplada

Figura 4.10 - Arquitetura fracamente acoplada

Nessa arquitetura híbrida as informações são encaminhadas a um servidor de serviços de mensagens, que de forma nativa fornece tolerância a falhas, balanceamento de carga e garantia de entrega da mensagem. O desacoplamento dos clientes de serviço da aplicação permite reutilização desses clientes e simplifica o processo de atualização dos *Web Services*. Além disso, quando uma aplicação ficar ocupada os dados do serviço são enfileirados no servidor de mensagens até que a aplicação se desocupe e possa recuperá-los.

A Figura 4.11 mostra a arquitetura do AgriBUS.

Figura 4.11 – Arquitetura do AgriBUS

O AgriBUS é um ESB extensível baseado em Java, formado pelas Aplicações de Negócio, Aplicações da Plataforma Tecnológica e pelo Barramento de Serviços, que suporta os requisitos funcionais mais comuns de um ESB, que inclui:

- **Roteamento:** fornecer um mecanismo de roteamento flexível e eficiente;
- **Transformação:** com base no requisitante e no destinatário, ser capaz de aplicar transformações nos dados de forma que o destinatário possa entender;

- **Multiprotocolos de transporte:** ser extensível o suficiente para suportar múltiplos protocolos de mensagens dependendo das necessidades;
- **Segurança:** fornecer mecanismos de autenticação e autorização para acessar diferentes serviços.

Os principais componentes do Barramento de Serviços do AgriBUS são mostrados na Figura 4.12.

Figura 4.12 – Componentes do barramento de serviços do AgriBUS

Fonte: (SODHI, 2005)

Os Receptores possuem dois tipos de interfaces, que são expostas para permitir que aplicações clientes possam enviar mensagens ao barramento, um *servlet* para receber requisições HTTP e uma destinação *Java Message Service* (JMS) (JMS, 2005). O Núcleo é responsável pelo roteamento, transformação e aplicação de segurança. É composto de um roteador que recebe as requisições e baseado no contexto da mensagem, aplica as transformações (Transformer), roteamento (Router) e segurança (Security). As

informações de transformação, roteamento e segurança são especificadas nas propriedades da mensagem. Os Despachantes são manipuladores dos diversos tipos de mecanismos de transporte do AgriBUS como email, HTTP e serviços de mensagens.

Todas as partes do AgriBUS são interconectadas através do processamento do documento XML `AgriBUSConfiguration.xml`, que relaciona todas as “rotas” nas quais o AgriBUS opera. No `AgriBUSConfiguration.xml` um elemento `<Route>` pode ter um ou mais elementos filhos do tipo `TransportHandler`. Este elemento basicamente aponta para um *bean* que representa um `TransportHandler` que deveria ser usado por esta rota. Opcionalmente o elemento `<Route>` pode ter um filho `DeadLetterDestination` que aponta para outra rota representando a destinação e um filho `AuthConstraint` que define um usuário ou grupo com autorização para usar esta rota. Trechos do `AgriBUSConfiguration.xml` são mostrados na Figura 4.13.

```
<?xml version="1.0" encoding="UTF-8"?>
<AgriBUSConfiguration xmlns="http://www.agribus.org.br/agribus/xml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
...
<Route name="utmConversorService" retryPolicyRef="100-sec-retry"
transformerRef="utmConversorServiceTransform">
  <TransportHandler beanName="utmConversorJMSTransport"/>
  <DeadLetterDestination routeName="DeadLetter"/>
  <AuthConstraint principals="murakami"/>
</Route>
...
<Route name="RedeliveryRequest" retryPolicyRef="500-sec-retry">
  <TransportHandler beanName="redeliveryRequestJMSTransport"/>
  <DeadLetterDestination routeName="DeadLetter"/>
</Route>
...
</AgriBUSConfiguration>
```

Figura 4.13 - AgriBUSConfiguration.xml

O `AgriBUSConfiguration.xml` é quem define o comportamento do AgriBUS. O AgriBUS carrega este XML e com as informações contidas nele gera uma estrutura de dados conforme a Figura 4.14.

Figura 4.14 – Estrutura gerada a partir do AgriBUSConfiguration.xml

O AgriBUS usa as informações desta estrutura, através do `EsbConfigManager`, para decifrar a rota, a transformação, e a checagem de autorização.

O diagrama de classes da Figura 4.14 mostra duas importantes interfaces, `TransformHandler` e `TransportHandler`. Essas interfaces permitem que diferentes implementações de transformadores e transportadores sejam criadas para rotear e transformar mensagens.

A Figura 4.15 mostra um diagrama de componentes que descreve a estrutura estática e dinâmica do AgriBUS.

Figura 4.15 – Diagrama de componentes do AgriBUS

O AgriBUS ao receber uma mensagem destinada para uma rota específica, serviço ou aplicação, executa uma seqüência de processamentos. A seqüência de processamento é mostrada abaixo.

1. O servidor de aplicações cria uma instância do EsbRouter;
2. O método `ejbCreate()` do EsbRouter é chamado, o qual executa as seguintes tarefas:
 - a. Carrega o arquivo `EnvConfig.xml` da URL especificada no *deployment descriptor* `ejb-jar.xml` do AgriBUS;
 - b. Lê a localização do `AgriBUSConfiguration.xml` contido no `EnvConfig.xml` e o carrega dentro de um *XmlBean* (XMLBEANS, 2005);

- c. Cria e inicializa a instância do `EsbConfigManager`, que cria e inicializa todos os *beans* configurados no `AgriBUSConfiguration.xml`, os quais formam a estrutura de dados mostrada na Figura 4.14;
- d. Inicializa a instância do `EsbRouterMonitorMBean`. Com esse *bean* é possível gerenciar os objetos do AgriBUS usando as interfaces do servidor de aplicação que suporta *Java Management Extension* (JMX) (JMX, 2005);
- e. Quando um componente ou serviço precisa enviar uma mensagem ele a envia para a fila “esb-in-queue” de entrada de dados do AgriBUS. Isso resulta na invocação do método `onMessage()` do `EsbRouter`. Vários processamentos ocorrem através deste método:
 - i. A destinação é colocada numa propriedade da mensagem, essa propriedade e outras são definidas no `envConfig.xml`);
 - ii. A verificação se o requerente tem autorização para enviar a mensagem é realizada, caso não tenha a mensagem é descartada e um *log* é gerado;
 - iii. Se um `TransformHandler` estiver configurado então ele é recuperado pela instância do `EsbConfigManager` e seu método `transformMessage()` é invocado passando a mensagem como parâmetro;
 - iv. Para cada `TransportHandler` obtido para a destinação da mensagem, o `EsbRouter` invoca o `transportMessage()` passando a mensagem como parâmetro;
 - v. Se a destinação não é encontrada no `EsbConfigManager`, então a mensagem é enviada para o `TransportHandler` da rota `DeadLetter`;

vi. Se o manipulador da rota é encontrado, mas o `TransportHandler` falha, a mensagem é enviada para a fila `RedeliveryQueue`. O enfileiramento para re-entrega envolve os seguintes passos:

1. A política de tentativas de entrega para a referida rota é recuperada do `EsbConfigManager`;
2. Baseado no intervalo da política de tentativas, contida no `ArgiBUSConfiguration.xml`, a próxima tentativa é calculada;
3. A hora é colocada na propriedade `MessageRedeliveryTime` da mensagem e enviada para a fila “`Redelivery.Queue`”;
4. Uma outra mensagem é enviada para o tópico “`Redelivery.Request.Topic`”. Esta mensagem tem um objeto *Long* com o tempo calculado para a nova tentativa de envio.

O tratamento de re-entrega de mensagens inclui agendamento e processamento. Ao receber uma mensagem de requisição de re-entrega, o `RedeliveryRequestProcessor` cria uma instância do `RedeliveryTask` com o tempo especificado na mensagem e o agenda através do `RedeliveryScheduler`. O `RedeliveryScheduler` é uma classe *singleton* que delega o trabalho para o `Timer`. O método `run()` do `RedeliveryTask` é chamado na hora agendada o qual tenta receber a mensagem da fila “`Redelivery.Queue`”. O receptor de mensagens que o `RedeliveryTask` usa tem um *selector* para selecionar somente as mensagens que têm o tempo agendado definido na propriedade `MessageRedeliveryTime`. Esta mensagem é então, enviada para a fila de entrada do ArgiBUS “`esb-in-queue`”.

Se o número máximo de tentativas é excedido então a mensagem é enviada para a rota `DeadLetter`. Se um erro ocorre durante uma tentativa de re-entrega, a mensagem é enviada para a rota `DeadLetter` associada à destinação. Se o envio para a rota `DeadLetter` também falha por qualquer razão, a mensagem é enviada para a rota `Error`. Finalmente se o envio para a rota `Error` também falhar, a transação toda do método `onMessage()` do `EsbRouter` é desfeita com um processamento *roll back*, que resulta na colocação da mensagem novamente na fila de entrada do AgriBUS.

O AgriBUS utiliza a tecnologia JMX (JMX, 2005), o que permite o gerenciamento e monitoramento dos *beans* em tempo de execução. O `EsbRouter` e o `RedeliveryRequestProcessor` são gerenciados pelos *Mbeans* `EsbRouterMonitor` e `RedeliverySchedulerMonitor` respectivamente. As operações de gerenciamento e monitoramento através de console web incluem:

- Dinamicamente atualizar a configuração carregada pelo `EsbConfigManager`;
- Controlar as re-entregas agendadas;
- Monitorar as filas de mensagens;

No futuro outras operações como atualizar e adicionar novas rotas, transformadores e transportadores e aplicar transformações e restrições de segurança em tempo de execução, deverão ser implementadas no AgriBUS, através da tecnologia JMX.

Um grande benefício do AgriBUS é que ele é puramente baseado nos padrões J2EE e não utiliza qualquer característica J2EE específica de um fornecedor. Ele pode ser implantado em qualquer servidor de aplicação J2EE. O AgriBUS foi testado sobre o servidor de aplicações JBoss (JBoss, 2005), um servidor de aplicações J2EE *open source*.

4.5 O provedor de serviços geoespaciais

O provedor de serviços geoespaciais proposto, GSPdegree (*Deegree-based Geospatial Services Provider*) tem como objetivo prover serviços geoespaciais, comumente existentes em SIGs baseados na web. Este provedor implementa os serviços geoespaciais com *Web Services*, conforme as idéias do *OpenGIS Consortium* (OGC).

Este componente da infra-estrutura é fundamental para os sistemas de informação para AP, pois a informação geográfica é muito importante para a agricultura de precisão. A AP lida com variabilidade espacial que requer dados espaciais sobre solo, cultura, máquinas agrícolas e aplicações. Análises espaciais devem ser executadas com SIG e muitos conjuntos de dados diferentes têm que ser coletados e gerenciados. Muito tempo e dinheiro é gasto com transformações de dados espaciais com diferentes sistemas de referência, para que esses dados possam ser interpretados e usados em SIGs proprietários. Além da AP, se a informação geográfica for padronizada e acessível, ela poderá ser útil em outras áreas como rastreabilidade, preservação de mananciais e outros propósitos ambientais.

Este provedor é baseado nos conceitos de *Spatial Data Infrastructures* (SDI) (GSDI, 2004; KORDUAN; BILL; BÖLLING, 2004). SDI melhora a interoperabilidade porque permite a construção de modelos de dados semânticos homogêneos e metadados padronizados, requisitos fundamentais para a interoperabilidade de informações geográficas entre SIGs.

Os serviços oferecidos pelo provedor de serviços geoespaciais são os principais serviços do *OGC Web Services Framework* do Modelo de Referência (*OGC Reference Model - ORM*) (PERCIVALL, 2003): *Web Map Service* (WMS) e *Web Feature Service* (WFS). Diversas implementações *open source* desses serviços como Deegree (DEEGREE, 2006) e MapServer (MAPSERVER, 2006) estão disponíveis livremente na web. O Deegree foi

escolhido para implementar o provedor de serviços geoespaciais por ser totalmente escrito em Java. O *framework* Deegree (DEEGREE, 2006) é um software livre sob a licença GNU (*General Public Licence Model*) escrito em Java destinado a implementar aplicações SIGs baseadas na web. Sua arquitetura e interfaces garantem interoperabilidade porque são totalmente baseadas nas especificações e conceitos da OGC, o que permite integrar produtos padronizados de outros fabricantes como *ArcIMS* do *Environmental Systems Research Institute* (ESRI), por exemplo.

O *framework* Deegree suporta as especificações do OGC e ISO/TC 211OGC, *Web Map Service* (WMS), *Web Feature Service* (WFS), *Web Coverage Service* (WCS), *Web Catalog Service* (WCAS) baseado no *Web Services Stateless Catalog Profile* da OGC, *Web Gazetter Service* (WFS-G), *Web Terrain Service* (WTS) e *Web Coordinate Transformation Service* (WCTS). Deegree suporta ainda, a GML, o *Filter Encoding Specification* e o *Styled Layer Descriptor Specification* (OGC, 2006).

A Figura 4.16 mostra a arquitetura do provedor de serviços geoespaciais baseado no *framework* Deegree (GSPdegree).

Figura 4.16 – Arquitetura do provedor de serviços geoespaciais

Para criar um mapa, o servidor GSPdegree utiliza dados acessados remotamente via REMOTEFWS, REMOTEWCS, REMOTEWMS ou localmente via LOCALWFS e LOCALWCS na mesma máquina virtual. O LOCALWFS e LOCALWCS podem estar em máquinas virtuais distintas também. Essa característica torna o servidor de serviços geoespaciais GSPdegree um excelente mecanismo de integração de SIGs distribuídos, pois os serviços são disponibilizados como *Web Services*, que são independentes de tecnologia e plataforma.

O provedor de serviços geoespaciais GSPdegree é um servidor WMS, que cria os mapas utilizando dados acessados através de *Web Services* locais e remotos. O WMS do Degree permite integrar dinamicamente dados geoespaciais (pontos, linhas, polígonos, imagens) de diferentes fontes em um único mapa e entregar este mapa para computadores pessoais como *desktop* e *palmtop*. O WMS do Degree é a implementação de referência oficial da especificação *OGC WMS 1.1.1*, que compreende um *Web Coverage Service* (WCS) e um *Web Feature Service* (WFS).

O WMS do Degree é muito flexível no que diz respeito às possibilidades de configuração, a adaptação para diferentes fontes de dados e formatos, leiautes e ambientes servidores. A configuração do WMS, WCS e WFS, é feita através da configuração de diferentes arquivos XML, que controlam a funcionalidade do servidor Degree. Os *Web Services* do Degree são realizados como *servlets* Java, no caso do WMS é um *servlet* genérico e um *servlet* especial para requisição de legenda gráfica. Estes *servlets* devem ser integrados no respectivo servidor web.

O WFS do Degree é baseado no *OGC Web Feature Service Implementation Specification 1.0.0* e serve as requisições *GetCapabilities*, *GetFeature*, *DescribeFeatureType* e *Transaction*. Possui interfaces para *Oracle Spatial*, *ESRI Shapefiles* e Bancos de Dados Relacionais, que são acessíveis através de interfaces *Java DataBase Connectivity* (JDBC). Uma das capacidades especiais do WFS consiste do armazenamento de dados geográficos em bases de dados convencionais. Além disso, WFS do Degree é capaz de usar diferentes esquemas de base de dados relacional sem adicionar ou alterar código. Por exemplo, a geração de dados para um esquema diferente como outra base de dados ou um visualizador gráfico, pode ser produzido pela integração de um *XSLT Stylesheet* no processamento de uma requisição *GetFeature*. Essa capacidade permite diferentes visões de um mesmo esquema de base de dados.

Os meios de armazenamento e funções atualmente disponíveis incluem:

- **Oracle Spatial:** *query* (simples e complexos tipos de objetos geográficos), *insert*, *update* e *delete*;
- **Postgis:** *query*, *insert*, *update* e *delete* (simples e complexos tipos de objetos geográficos);
- **MySQL (com extensão espacial):** *query* (simples e complexos tipos de objetos geográficos);
- **ESRI ArcSDE:** *query*, *insert*, *update* e *delete* (simples tipos de objetos geográficos);
- **Features com pontos de geometria armazenados em bancos de dados convencionais (colunas X e Y):** *query*, *insert*, *update* e *delete* (simples e complexos tipos de objetos geográficos);
- **Features em banco de dados convencional com geometrias como GML em campos texto:** *query* (simples e complexos tipos de objetos geográficos), *insert*, *update* e *delete* (simples tipos de objetos geográficos);
- **Shapefiles:** *query* (simples tipos de objetos geográficos);
- **MapInfo MIF:** *query* (simples tipos de objetos geográficos);
- **BNA files:** *query* (simples tipos de objetos geográficos);

O WFS do Deegree oferece várias opções de configuração, que permitem o ajuste para diferentes fontes de dados, tipos de dados e ambientes servidores, através da configuração dos diversos arquivos XML.

O WCS do Deegree é baseado na especificação *OGC Web Coverage Service 1.0.0* e serve as requisições *GetCapabilities* e *GetCoverage*. O WCS suporta TIFF, GIF, BMP, PNG, JPEG e o formato binário IDRISI. Para realizar acessos rápidos a imagens *raster* muito

grandes o WCS do Deegree usa particionamento de imagens *raster* (*tiles*). Os *tiles* podem ser indexados ou acessados através de um *quadtree* ou através de seu nome construído a partir das coordenadas iniciais. Para a criação de *tiles* dois utilitários são oferecidos juntamente com o *framework* Deegree (*TileImageTree* e *TileImageTree2*) que criam índices para os *tiles* de grandes imagens *raster*. A implementação da especificação *OGC GridCoverage* contida no Deegree oferece a oportunidade para organizar dados *raster* em uma maneira que a busca dos dados possam ser otimizadas. Ambas as variantes requerem que o *raster* esteja dividido em *tiles* em um tamanho entre 400 e 1000 *pixels*. Os *tiles* serão armazenados como arquivos de imagem em um sistema de arquivos, pois o *Deegree* ainda não suporta armazenamento em campos BLOB (*Binary Large Object*) de bancos de dados.

4.6 O portal para serviços agrícolas

A complexidade e o alto custo dos modernos sistemas de informação têm motivado muitas empresas a investirem em portais como mecanismo de gerenciamento de informações de maneira coesa e estruturada. Os portais oferecem muitas vantagens sobre outros tipos de aplicações, por exemplo: fornecem um único ponto de entrada para os usuários; podem acessar *Web Services* direta e transparentemente; são altamente flexíveis, podendo atuar na forma de *B2E (intranet)*, *B2B (extranet)* ou *B2C (internet)*; e podem acessar outros portais para formar uma rede de portais, estendendo as capacidades de negócio.

Um portal é uma aplicação web que fornece personalização, autenticação única, agregação de conteúdo e hospedagem da camada de apresentação de um sistema de informação. Agregação é a ação de integrar conteúdos de diferentes fontes em uma página web. Um portal pode ter características de personalização sofisticadas para fornecer conteúdo personalizado aos seus usuários.

Os benefícios potenciais de um portal vão de encontro a vários problemas identificados por diversos pesquisadores, citados no capítulo 1, como:

- “Os sistemas deveriam ter uma interface de usuário simples que permita customizações para diferentes perfis de usuário. Uma interface amigável é importante para todos, especialmente para usuários com pouca experiência com software”. *Portais permitem a personalização de forma padronizada e simples;*
- “O usuário deve ter permissão para controle completo e ter acesso a parâmetros de processamento e funções de análise. Usuários experientes devem poder controlar e experimentar novas soluções”. *Portais permitem single sign-on, de maneira que o controle de permissões fica centralizado e elimina a necessidade do usuário se conectar a diversos sistemas diferentes para realizar uma tarefa;*
- “Sistemas de software padronizados e integrados são necessários. Isso pode reduzir o investimento técnico, a curva de aprendizado e a necessidade de suporte técnico”. *Portais permitem integração de outros portais formando uma rede de portais, o que é uma forma de integração de aplicações.*
- “Suporte a integração e interoperabilidade com outros pacotes de softwares, locais ou remotos, via Internet usando padrões abertos. Essas características são fundamentais para integração com sistemas legados e sistemas distribuídos”. *Além da integração entre portais, um portal pode invocar diretamente um Web Service, que é uma tecnologia usada para integração de aplicações.*

Desse modo, este trabalho propõe o uso da tecnologia de portais para a centralização de acesso e personalização das aplicações para AP. O portal proposto neste trabalho, denominado PASWP (*Precision Agriculture Web Services Portal*) foi criado usando o

framework *JetSpeed* da *Apache Software Foundation* (JETSPEED, 2005), que implementa a especificação de *portlets* (ABDELNUR; HEPPEL, 2003).

JetSpeed é uma implementação *open source* de um *Enterprise Information Portal* com Java e XML, que permite que os usuários acessem recursos da rede como banco de dados, aplicações e *Web Services*. O usuário pode acessar o portal através de *web browsers*, *WAP-phones*, *paggers* ou qualquer outro dispositivo. JetSpeed atua como um concentrador, onde informações de múltiplas fontes são disponibilizadas de maneira simples e fácil. Os dados apresentados pelo JetSpeed é independente do tipo do conteúdo, por exemplo XML, RSS (*Really Simple Syndication*) ou SMTP (*Simple Mail Transfer Protocol*) podem ser integrados com JetSpeed. A Figura 4.17 mostra a página principal do protótipo do PAWSP (*Precision Agriculture Web Services Provider*).

Figura 4.17 – Página principal do PAWSP

A Figura 4.18 mostra a página de personalização do portal.

Figura 4.18 – Página de personalização do portal

5 ESTUDO DE CASO: SERVIÇO DE FILTRAGEM DE DADOS “FILTERING”

5.1 Introdução

Este capítulo apresenta e discute um estudo de caso utilizado para validar as idéias desse trabalho. A abordagem adotada foi a experimental, na qual foram desenvolvidos protótipos que permitiram a condução de experimentos com as idéias que foram sendo elaboradas. Fundamentando-se em estudos realizados sobre experiências e propostas de sistemas de informação para agricultura de precisão e áreas correlatas, arquitetura de software, padrões abertos e plataformas de software livre, diversas versões do protótipo foram sendo geradas e experimentadas, e baseando-se nos resultados, novas idéias foram sendo elaboradas e implementadas, conforme o modelo de desenvolvimento de sistema denominado Espiral. A cada versão do protótipo eram testados novos recursos da infra-estrutura.

O protótipo é uma aplicação web que realiza a filtragem de dados de monitores de produtividade. A metodologia de filtragem consiste na identificação, caracterização e remoção de erros de mapas de rendimento gerados por monitores de produtividade, instalados em colhedeiras equipadas com receptores de GPS (Global Positioning System) e sensores de produtividade. A metodologia foi desenvolvida por Menegatti (2002) (MOLIN; MENEGATTI, 2002) na Escola Superior de Agricultura “Luiz de Queiroz” da Universidade de São Paulo (ESALQ-USP). Esta aplicação foi escolhida para ser o protótipo porque envolve diversas funcionalidades importantes para a AP, no que diz respeito a qualidade dos dados manipulados para tomada de decisão e porque envolve características não funcionais como transferência de arquivos através da Internet, desempenho, garantia da execução do serviço, comunicação assíncrona, etc. Outras razões motivaram a escolha dessa aplicação: a

disponibilidade de usuários especialistas (Molin e Menegatti da ESALQ-USP), a grande quantidade de dados coletados de diversos monitores de produtividade e a oportunidade de criação de um serviço reutilizável independente de tecnologia.

5.2 A metodologia para identificação, caracterização e remoção de erros de mapas de produtividade

Diversos monitores de produtividade comerciais estão disponíveis no mercado para a coleta de informações de forma bastante adensada. Segundo Menegatti (2002), devido à automação e à alta quantidade de informações ocorrem diversos erros no processo de coleta e a retirada desses erros é um fator importante que interfere na qualidade da informação. Portanto, é necessário que os erros sejam retirados para a geração dos mapas de produtividade. A metodologia para isso consiste em uma rotina de filtragem de dados brutos, com oito etapas, que elimina erros presentes em mapas de produtividade. Originalmente essas etapas eram executadas com a ajuda, apenas, de uma planilha de cálculos:

Etapa 0 - Preparo dos dados para a aplicação do filtro.

Etapa 1 - Retirada de erros grosseiros de posicionamento.

Etapa 2 - Retirada de pontos com produtividade nula ou ausente.

Etapa 3 - Retirada de pontos com largura parcial de plataforma.

Etapa 4 - Remoção de dados com umidade nula ou ausente.

Etapa 5 - Remoção de pontos com distância nula.

Etapa 6 - Remoção de pontos gravados durante o intervalo de preenchimento.

Etapa 7 - Remoção de pontos com valores de produtividade discrepantes.

Com o objetivo de automatizar a metodologia de filtragem e ao mesmo tempo testar as idéias sobre a infra-estrutura de desenvolvimento de sistemas para AP, um protótipo foi desenvolvido e experimentos conduzidos sobre ele.

A seguir são apresentadas as diversas fases da evolução desse protótipo, com os respectivos comentários dos avanços ou retrocessos sobre as idéias iniciais.

5.3 A infra-estrutura inicial

A primeira proposta de criação da infra-estrutura foi elaborada e submetida à comunidade científica da área, no *6th International Conference on Precision Agriculture and Others Precision Resources Management* em 2002 (ICPA'02), em Minneapolis nos Estados Unidos da América (MURAKAMI et al., 2002). Com a aceitação da proposta, iniciou-se um trabalho de refinamento da infra-estrutura.

A infra-estrutura inicial consistia de um conjunto de *frameworks* baseados em componentes para desenvolvimento de sistemas de informação para AP. A idéia era que o desenvolvimento baseado em componentes pudesse permitir que os sistemas acompanhassem a rápida evolução da AP, por permitir a substituição ou adição de novos componentes aos sistemas sem impactá-los. A infra-estrutura era composta basicamente de três *frameworks*:

- eMOSAICo - para desenvolvimento de componentes com funcionalidades básicas para Sistemas de Informação para AP, uma implementação do modelo conceitual MOSAICo (Modelo de Objetos para Sistemas Abertos de Informação de Campo) para Agricultura de Precisão (SARAIVA, 1998);
- *Struts* - para desenvolvimento de aplicações baseadas na web, um *framework* de código aberto da *Apache Software Foundation* que implementa o padrão MVC (*Model, View, Controller*);
- *Mapper* - para permitir interoperabilidade entre os componentes através de transformações, um *framework* baseado em XML que mapeia dados de uma entidade para qualquer outra. Por exemplo, ler dados de um arquivo texto e escrever em uma tabela de banco de dados ou ler dados de banco de dados e escrever em XML.

5.4 O Filtering

O protótipo operacional da aplicação de filtragem de dados de monitores de produtividade foi implementado sobre a infra-estrutura proposta e utilizou os três *frameworks*, eMOSAICo, *Struts* e *Mapper* (MURAKAMI, E. et al., 2003a, 2003b). O eMOSAICo foi usado para criar os objetos de negócio, o *Struts* para implementar a camada de apresentação e o *Mapper* para transformar o formato proprietário dos arquivos texto, gerados pelos monitores de produtividade, para o formato interno padronizado (PAML) e vice-versa. A Figura 5.1 mostra a visão lógica da arquitetura do *Filtering*. A arquitetura era baseada na arquitetura orientada a serviços e nos *Enterprise JavaBeans* (EJB).

Figura 5.1 – Visão lógica da arquitetura híbrida do Filtering com Web Services e EJB

A idéia de usar EJB e *Web Services* foi garantir a interoperabilidade entre aplicações homogêneas e heterogêneas respectivamente, com a possibilidade de clientes acessarem tanto o *Web service* como o EJB. A razão de usar EJB foi a facilidade de transformar um EJB *SessionBean Stateless* em um *Web Services*, além dos controles de transação e de segurança nativos da tecnologia. A estratégia de construir *Web Services* a partir de interfaces de EJBs

mostrou-se uma boa e flexível estratégia para os componentes de negócio, uma vez que permite a definição do uso do componente, local ou remoto, em tempo de implantação.

A metodologia de filtragem foi implementada em Java como componente EJB que posteriormente foi convertido em *Web Service*. O acesso ao *Web Service* foi realizado através de uma aplicação web construída com o *framework Struts*. Os parâmetros, tipo de retorno e localização do *Web Service* foram descritos em WSDL (*Web Service Description Language*). A partir da descrição do serviço foi gerado o cliente que pode acessá-lo independentemente de plataforma ou localização física que estiver implantado. A Figura 5.2 mostra parte do serviço de filtragem descrito em WSDL.

```

...
<complexType name="MonitorInternalFormat">
  <sequence>
    <element name="altitude" nillable="true" type="xsd:string"/>
    <element name="crop" nillable="true" type="xsd:string"/>
    <element name="cycles" nillable="true" type="xsd:string"/>
    <element name="distance" nillable="true" type="xsd:string"/>
    <element name="dryYield" nillable="true" type="xsd:string"/>
    <element name="field" nillable="true" type="xsd:string"/>
    <element name="flow" nillable="true" type="xsd:string"/>
    <element name="id" type="xsd:int"/>
    <element name="latitude" nillable="true" type="xsd:string"/>
    <element name="longitude" nillable="true" type="xsd:string"/>
    <element name="moisture" nillable="true" type="xsd:string"/>
    <element name="swath" nillable="true" type="xsd:string"/>
  /sequence>
</complexType>
<wsdl:message name="productionDataFilteringResponse">
  <wsdl:part name="productionDataFilteringReturn"
    type="impl:ArrayOfMonitorInternalFormat"/>
</wsdl:message>...
<wsdlsoap:address
  location="http://localhost:8080/axis/services/FilteringWS"/>
...
  
```

Figura 5.2 - WSDL do serviço de filtragem de dados de monitores de produtividade

Os trechos numerados correspondem a: (1) formato dos parâmetros de entrada do serviço; (2) retorno do serviço; e (3) localização do serviço.

Dois arquivos textos são enviados para a aplicação. Um possui os limites do talhão e o outro os dados coletados pelo monitor de produtividade. Esses arquivos são submetidos ao

componente *Mapper* que transforma esses dados numa estrutura interna de representação dos dados. Essa estrutura é então submetida ao componente de filtragem, onde é aplicado o algoritmo da metodologia para identificação, caracterização e remoção de erros. Ao terminar o processamento essa estrutura é devolvida ao componente *Mapper* que a converte novamente ao formato inicial. A Figura 5.3 mostra a interface gráfica do *Filtering*.

Figura 5.3 – Interface gráfica do Filtering construída com o framework Struts

O protótipo se mostrou um valioso instrumento para a validação e aperfeiçoamento da infra-estrutura. Depois de alguns experimentos foram detectados alguns problemas como: falta de funcionalidades geoespaciais para gerar mapas, limitações de interoperabilidade devido aos diferentes formatos de dados gerados pelos diferentes monitores de produtividade, manipulação de grandes quantidades de dados usando comunicação síncrona e diferentes

caminhos de acesso às aplicações. Algumas propostas de melhoria da infra-estrutura foram então elaboradas:

- Evoluir o eMOSAICo para fornecer além de funcionalidades básicas para AP, fornecer também características espaciais e temporais. Esta proposta tornou-se tema de doutorado e encontra-se em desenvolvimento no LAA/EPUSP (RIBEIRO; SARAIVA; MURAKAMI, 2003, 2004);
- Criar uma linguagem padronizada para comunicação entre aplicações. Essa proposta resultou na criação da linguagem PAML (*Precision Agriculture Markup Language*), que codifica o eMOSAICo em XML para servir de linguagem padrão para troca de dados entre aplicações;
- Incluir um serviço de mensagens para comunicação assíncrona para garantir a disponibilidade da aplicação, quando da manipulação de grandes quantidades de dados. Essa proposta resultou na criação do AgriBUS, um barramento de serviços baseado em *Web Services* e serviços de mensagem;
- Padronizar as interfaces do usuário e formas de acesso às aplicações. Essa proposta resultou na definição do uso da tecnologia de portais para centralizar o acesso às aplicações e o uso de padrões web (W3C, 2006) para padronizar as interfaces do usuário. A necessidade dessa padronização se deu pelas necessidades dos sistemas para AP apresentadas no capítulo 1 e pela constatação durante os experimentos, onde vários componentes (serviços) foram sendo implementados para apoiar o desenvolvimento do protótipo como o *Localizador de Serviços* e o *Conversor de Coordenadas Geográficas*, cada qual com sua interface gráfica e sua URL, acessados de forma individualizada e descentralizada.

5.5 As versões finais da infra-estrutura e do Filtering

A última versão do protótipo testou os padrões e as tecnologias adotadas na infra-estrutura. Nessa fase foram testadas tanto plataformas de software livre quanto proprietárias, para garantir algumas qualidades de software desejadas para as aplicações para AP como portabilidade, flexibilidade, extensibilidade e reusabilidade. A Figura 5.4 mostra a visão geral da arquitetura e a Figura 5.5 mostra como o protótipo foi implantado.

Figura 5.4 – Visão geral da arquitetura da aplicação web "Filtering"

Figura 5.5 – Diagrama de implantação em UML da aplicação web “Filtering”

Os componentes destacados na cor cinza formam a plataforma de software livre usada no protótipo.

Como mostrado no diagrama, os diversos componentes de um sistema de informação construído com a infra-estrutura podem ser distribuídos em múltiplas plataformas de hardware e software. Tipicamente os componentes das camadas de apresentação e integração são implantados em um servidor e os componentes de negócio (serviços) em diversos servidores distribuídos na Internet. No Application Server foram usados dois servidores de aplicação, (JBoss Application Server e Tomcat Web Server) com serviços agrícolas e serviços geoespaciais respectivamente, para demonstrar a capacidade de

distribuição. O AgriBUS também poderia ser implantado separadamente da aplicação *Filtering*. Para os clientes a localização dos componentes do sistema é transparente.

A solução da última versão do protótipo usa basicamente softwares livres, com exceção do *Web Browser* e dos Sistemas Operacionais. O objetivo foi mostrar a independência de plataforma de hardware e software dos sistemas construídos com a infraestrutura proposta.

As experiências mostraram que a infra-estrutura tem recursos suficientes para construir aplicações robustas, escaláveis, extensíveis e de baixo custo para atenderem aos requisitos tecnológicos dos sistemas de informação para AP.

5.6 Ambientes de execução dos experimentos com o protótipo

Os experimentos com o protótipo foram conduzidos em dois ambientes. No primeiro, foram executados os experimentos com o protótipo em rede local a 100 Mbps (Intranet). No segundo foi usada a rede pública através de Internet rápida a 300Kbps. O volume de dados variou entre 10.000 a 50.000 registros em cada arquivo de tamanos 1 e 5 Mb respectivamente, sendo cada registro um ponto de amostra de produtividade georeferenciado. O tempo de resposta na Intranet variou entre 3 e 7 segundos e na Internet de 4 a 10 minutos, dependendo do horário de uso. A maior parte desse tempo é devido à transferência de arquivo (*upload*). Por se tratar de um processamento assíncrono, podemos considerar esses tempos aceitáveis. Como trabalho futuro foi proposta a evolução da infra-estrutura para suportar computação em grid, o que permitirá o processamento de volumes de dados bem maiores que os usados neste trabalho.

Ambientes de execução utilizados nos experimentos:

Cliente

- Pentium III 800Mhz com 248Mb de memória RAM e 20Gb de disco rígido;

- Microsoft Windows XP Professional 2002;
- Microsoft Internet Explorer 6.0 e Netscape 7.02.
- JVM – Java Virtual Machine 1.4.2.

Servidor

- AMD Sempron 1,50 GHz com 512 Mb de memória RAM e 80Gb de disco rígido;
- Microsoft Windows XP Professional 2002;
- Servidor de Aplicações:
 - JBoss Application Server 3.2.7 para o servidor de aplicações para AP;
 - Jetspeed 1.4 para o Portal PAWSP;
 - Sun Java System Application Server Platform Edition 8 (usado às vezes no lugar do JBoss);
 - Tomcat 4.1 para o Provedor de Serviços Geoespaciais GSPdegree;
 - JVM – Java Virtual Machine 1.4.2.

6 CONSIDERAÇÕES FINAIS

Este capítulo contém os comentários finais e reflexões a respeito do trabalho desenvolvido. São apresentadas as conclusões sobre o trabalho, com base nos estudos e nos resultados alcançados com a infra-estrutura e com os exemplos de implementação. São destacadas as contribuições e são sugeridos caminhos para que as idéias e a infra-estrutura possam continuar evoluindo em novos trabalhos de pesquisa.

6.1 Conclusões

Como atividade inicial deste trabalho foram realizados estudos sobre sistemas de informação e infra-estruturas de desenvolvimento de sistemas para agricultura de precisão. Esses estudos mostraram problemas diversos que vão desde interfaces do usuário não-padronizadas e pouco amigáveis até problemas de integração e capacidade de evolução. Os resultados do estudo sobre sistemas de informação para AP permitem concluir que independentemente da origem dos sistemas, comerciais ou experimentais desenvolvidos pela pesquisa, o problema em grande parte está no caráter proprietário da maioria das iniciativas e em como as arquiteturas desses sistemas são projetadas. Em razão disso e da necessidade de evolução dos sistemas para AP, a sua complexidade e a sua abrangência demandam capacidade de integração com outros sistemas e recursos.

Com base nessa conclusão, foi realizado então, um estudo sobre a alternativa que parece ser a mais viável quanto à interoperabilidade, capacidade de evolução e baixo custo, o uso de padrões abertos e plataformas de software livre. O estudo realizado focalizou também

arquitetura de software, e dele se concluiu que a combinação dos estilos arquiteturais cliente-servidor em camadas e orientado a serviços pode suportar a maioria dos sistemas para AP.

Baseado nas pesquisas realizadas sobre sistemas de informação para AP e tecnologias da informação, uma infra-estrutura de desenvolvimento de sistemas orientados a serviços distribuídos para agricultura de precisão foi definida, a qual é composta de cinco componentes: uma arquitetura de referência para sistemas de informação orientados a serviços para AP, uma linguagem padrão para troca de dados entre serviços agrícolas, um barramento para conexão de serviços agrícolas, um provedor de serviços geoespaciais e um portal para serviços agrícolas.

A arquitetura de referência formada pelo modelo de referência MOSAICo e estilos arquiteturais em camadas e orientado a serviços se mostrou bastante interessante como ponto de partida para desenvolvimento de novos sistemas de informação para AP, uma vez que fornece um modelo de negócio consistente e uma estrutura padronizada e bem definida para dirigir o projeto e a implementação. Este modelo de negócio foi materializado em uma linguagem codificada em XML (PAML) que estende o padrão aberto para transporte e armazenamento de informações geográficas (GML). Essas características facilitam o entendimento e uso da linguagem e permitem a sua evolução sem causar impacto nas aplicações que utilizam suas versões anteriores. Desse modo, a linguagem pode avançar na modelagem do domínio da AP mantendo a compatibilidade e a interoperabilidade das aplicações.

O barramento de serviços se mostrou muito importante no papel de integrador de aplicações heterogêneas. Ele otimiza as interações entre consumidores e provedores de serviços, fornecendo segurança, escalabilidade, confiabilidade e reuso de clientes de serviços. O barramento permite a conexão de sistemas legados, banco de dados, sistemas especialistas,

etc. Em alguns casos é necessário implementar componentes de tradução no barramento, para que a integração desses sistemas seja possível.

As infra-estruturas de dados espaciais (SDI – *Spatial Data Infrastructure*) baseadas nos padrões abertos da OGC são a grande promessa de substituição e/ou compatibilização de SIGs proprietários e abertos. As implementações abertas existentes permitem muitas das funcionalidades de SIGs na web. O provedor de serviços geoespaciais é um componente da infra-estrutura fundamental para desenvolvimento de sistemas para AP. No entanto, mais testes devem ser realizados e melhorias implementadas no provedor proposto, para que possa efetivamente cumprir o seu papel dentro da proposta de sistemas de informação orientados a serviços distribuídos para agricultura de precisão.

O portal foi especificado para atender a demanda dos usuários por uma interface padronizada e centralizada que facilitasse o acesso aos diversos sistemas necessários para a realização da AP. O portal proposto fornece personalização, autenticação única, integra conteúdos de diferentes fontes em uma página web e é baseado em padrões abertos. Apesar de não ter sido testado com muitas aplicações, ele atende à maioria das necessidades identificadas neste trabalho. Assim como foi recomendado para o provedor de serviços geoespaciais, mais experimentos devem ser realizados com este componente.

A fim de apoiar os experimentos com a infra-estrutura um protótipo foi criado e testado. O protótipo, apesar de simples, mostrou como um sistema de informação para AP pode ser construído com serviços agrícolas e geoespaciais distribuídos usando a infra-estrutura. Mostrou também, que essa pode ser uma alternativa interessante para publicação e disponibilização de algoritmos especializados desenvolvidos por pesquisadores. Implementados como serviços, esses algoritmos podem ficar disponíveis a um público mais amplo, acessados como uma aplicação web ou como um serviço invocado diretamente de programas.

Em suma, a infra-estrutura se mostrou adequada para a criação de sistemas de informação para AP interoperáveis, de baixo custo e com capacidade de evolução, mudando o paradigma dos sistemas para AP preponderantemente proprietários e monolíticos para abertos e orientados a serviços distribuídos.

6.2 Principais contribuições

Este trabalho de tese demonstrou como construir sistemas de informação para agricultura de precisão interoperáveis, de baixo custo e com capacidade de evolução usando uma infra-estrutura baseada em padrões abertos e plataformas de software livre. A demonstração de como construir sistemas para agricultura de precisão foi operacional, através da construção das regras negociais como serviços distribuídos (agrícolas e geoespaciais) interligados através de um barramento de serviços e acessados por interfaces de usuário baseadas em padrões web disponíveis em um portal. Ainda que não tenham sido realizados experimentos intensos, a arquitetura definida na infra-estrutura para os sistemas de informação permite que serviços sejam distribuídos de maneira a formar uma grande rede para processamento de alto desempenho.

No domínio da agricultura de precisão, a infra-estrutura inova por definir uma solução aos problemas de negócio através do uso de *Web Services*, componentes independentes de plataforma de software e hardware que podem ser implantados, encontrados e invocados via Internet. Inova também por permitir a construção de sistemas de informação para AP através da composição de serviços agrícolas e geoespaciais distribuídos e abertos, criando uma nova geração de sistemas de informação para AP, antes monolíticos e proprietários.

O modelo de referência MOSAICo e os estilos arquiteturais cliente-servidor em camadas combinado com o estilo orientado a serviços, definem a arquitetura de referência para sistemas de informação para AP. Esta arquitetura de referência contribui por dirigir os

projetos das arquiteturas dos novos sistemas de informação para AP, permitindo a evolução das soluções dos pontos de vista de negócio e técnico.

Apesar de existirem iniciativas semelhantes como AgXML e AgroXML, a PAML inova no sentido de ser uma linguagem padrão para troca de dados entre serviços com a mesma estrutura do modelo de objetos com recurso espaço-temporal implementado como um *framework* (*Spatio-Temporal Framework* - jSTF), para implementação de soluções de negócio para AP. Este *framework* é uma evolução das idéias seminais do eMOSAICo (*extended MOSAICo*), que atualmente é conduzido como um outro trabalho de doutorado no LAA (Laboratório de Automação Agrícola da Escola Politécnica da Universidade de São Paulo).

O barramento de serviços AgriBUS é uma inovação para os sistemas de informação para AP por fornecer uma infra-estrutura de conectividade baseada em *Web Services* e *Message Services* que objetiva principalmente servir de *middleware* entre aplicações clientes e serviços agrícolas e geoespaciais, provendo segurança, confiabilidade, desempenho e escalabilidade. Através deste *middleware* é possível também conectar outros recursos como base de dados, sistemas especialistas e legados.

6.3 Trabalhos futuros

Trabalhos futuros motivados pela possibilidade de consolidação das idéias apresentadas neste trabalho são apresentados numa ordem de prioridade considerada importante para que seja alcançado um nível de usabilidade razoável.

O primeiro passo para a consolidação das idéias apresentadas é a entrega da infra-estrutura a diversos experimentadores para que seja empregada na construção de sistemas de informação para AP nos diversos tipos como modelos de simulação, suporte a decisão e SIG para web. A implementação de serviços isolados para tratamento e análise de dados,

resultantes de pesquisa e não disponíveis em softwares proprietários e que viesse a complementá-los é uma possibilidade concreta, em oposição ao desenvolvimento de uma ferramenta completa para AP. A implementação desses serviços deve priorizar o uso do *Spatio-Temporal Framework* (jSTF). Os resultados devem ser recolhidos e analisados para que possam ser definidos pontos primordiais de evolução dos conceitos expostos nesta tese.

O segundo passo é evoluir a PAML de maneira a abranger todo o MOSAICo, de forma que permita a representação do modelo em documentos XML capazes de serem interpretados pelos sistemas de informação mais abrangentes que deverão surgir com o uso da infra-estrutura. Essa evolução deve acompanhar a evolução do *framework* jSTF.

O terceiro passo é criar um servidor de registros para publicar e manter um catálogo de serviços especificados em um ou mais sistemas usados para classificar categorias de negócio, indústria e produtos como NAICS (*North American Industry Classification System*) e UNSPSC (*Universal Standards Products and Services Classification*). Um serviço para localização de serviços deve ser implementado e disponibilizado no portal.

Finalmente, em paralelo aos demais passos devem ser continuados os estudos para evolução dos conceitos apresentados nesta tese, por exemplo: *Grid Computing*, *Web Services Provisioning*, comunicação sem fio, computação móvel e computação pervasiva.

Um problema identificado nos experimentos foi a dificuldade de transportar os dados do computador de bordo para o computador do escritório. Desse modo, uma pesquisa mais abrangente deve ser feita sobre essa necessidade. Por exemplo, os sistemas móveis com o padrão de transmissão de dados *Universal Mobile Telecommunications System* (UTMS) são capazes de receber imagens e mapas em telefones móveis. Acredita-se que essas tecnologias poderão aumentar e melhorar o uso de telefones móveis em máquinas agrícolas para transferir e adquirir informações pela Internet.

REFERÊNCIAS*

- ABDELNUR, A.; HEPPER, S. JSR-000168 Portlet Specification, final release, 2003. Disponível em: <<http://www.jcp.org/aboutJava/communityprocess/final/jsr168/>>. Acesso em: 27 de fevereiro de 2006.
- ADRIAN, A.M; NORWOOD, S.H.; MASK, P.L. Producers' perceptions and attitudes toward precision agriculture technologies. **Computers and Electronics in Agriculture**, v.48, p.256–271, 2005.
- AGRIS AP. AGRIS Application Profile: Methodology. Disponível em: <http://www.fao.org/agris/agmes/Documents/ap_methodology.doc>. Acesso em: 27 de fevereiro de 2006.
- AGRIS AP. The AGRIS Application Profile for the International Information System on Agricultural Sciences and Technology: Guidelines on Best Practices for Information Object Description. Food and Agriculture Organization of the United Nations, 2005. 81p. Disponível em: <http://www.fao.org/documents/show_cdr.asp?url_file=/docrep/008/ae909e/ae909e00.htm>. Acesso em: 27 de fevereiro de 2006.
- AGROVOC. Agrovoc Thesaurus. Food and Agriculture Organization of the United Nations. Disponível em: <http://www.fao.org/aims/ag_intro.htm>. Acesso em: 27 de fevereiro de 2006.
- AgXML. AgXML Standard. AgXML L.L.C. 2004. Disponível em: <<http://www.agxml.org/Standard.html>>. Acesso em: 27 de fevereiro de 2006.
- AUERNHAMMER, H. Precision farming – the environmental challenge. **Computers and Electronics in Agriculture**, v.30, p.31-43, 2001.
- BACKES, M.; DÖRSCHLAG, D.; PLÜMER, L. A metadata profile for precision agriculture based on ISO 19115. In: EUROPEAN CONFERENCE ON PRECISION AGRICULTURE, 4., 2003, Berlin. **Proceedings**. Berlin, 2003. p.41-46.
- BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software Architecture in Practice**, Second Edition. Addison-Wesley Professional, 2003.
- BECK, H.W.; XIN, J. Using Java, CORBA, and OODBMS to Develop Agricultural Databases. In: INTERNATIONAL CONFERENCE ON COMPUTERS IN AGRICULTURE, 7., 1998, Orlando. **Proceedings**. St. Joseph, ASAE, 1998.

* De acordo com:

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6023**: informação e documentação: referências: elaboração. Rio de Janeiro, 2002.

- BOOTH, D. et al. Web Services Architecture. W3C Working Group Note 11 February 2004. Disponível em: <<http://www.w3.org/TR/ws-arch>>. Acesso em: 27 de fevereiro de 2006.
- BOOTH, D.; LIU, C.K. Web Services Description Language (WSDL) Version 2.0 Part 0: Primer. W3C Candidate Recommendation 6 January 2006. Disponível em: <<http://www.w3.org/TR/wsdl20-primer/>>. Acesso em: 27 de fevereiro de 2006.
- BRAY, T. et al. Extensible Markup Language (XML) 1.0 (Third Edition), 2004. W3C Recommendation. Disponível em: <<http://www.w3.org/TR/REC-xml>>. Acesso em: 27 de fevereiro de 2006.
- BRÜGGEMANN, H. The GDI NRW as a component of the German, European and Global Spatial Data Infrastructure. In: GLOBAL SPATIAL DATA INFRASTRUCTURE INITIATIVE, 7., 2004, India. **Proceedings**. Bangalore, 2004. 9p.
- CARBONE, G. Enhancing Web Services Infrastructures with JMS, Disponível em: <<http://www.onjava.com/pub/a/onjava/2002/06/19/jms.html>>. Acesso em: 01 de outubro de 2005.
- CHIN, R. S.; CHANSON, S. T. Distributed object-based programming systems. **ACM Computing Surveys**, v.23, n.1, p.91–124. 1991.
- CLEMENT, L. et al. UDDI Version 3.0.2. OASIS UDDI Spec Technical Committee Draft 2004. Disponível em: <http://uddi.org/pubs/uddi_v3.htm>. Acesso em: 27 de fevereiro de 2006.
- COWARD, D. JSR-000053 Java Servlet 2.3 and JavaServer Pages 1.2 Specifications, 2001. Disponível em: <<http://www.jcp.org/aboutJava/communityprocess/final/jsr053/>>. Acesso em: 27 de fevereiro de 2006.
- COX, S. et al. GML Implementation Specification, version. 2.1.2. OpenGIS Implementation Specification, 2002. Disponível em: <<http://www.opengeospatial.org/docs/02-069.pdf>>. Acesso em: 27 de fevereiro de 2006.
- COX, S. Information technology: the global key to precision agriculture and sustainability. **Computers and Electronics in Agriculture**, v.36, p.93-111, 2002.
- DEEGREE. Framework para Construção de Infra-Estrutura para Dados Espaciais. Disponível em: <<http://deegree.sourceforge.net/>>. Acesso em: 01 de março de 2006.
- DI NITTO, E.; ROSENBLUM, D. Exploiting ADLs to specify architectural styles induced by middleware infrastructures. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 1999, Los Angeles. **Proceedings**. Los Angeles, 1999. p.13–22.
- DOLUSCHITZ, R. et al. agroXML - A Standardized Data Format for Information Flow in Agriculture. In: EFITA/WCCA JOINT CONGRESS ON IT IN AGRICULTURE, 2005, Portugal. **Proceedings**. Vila Real, 2005. p.439-443.
- FALLSIDE, D.C.; WALMSLEY, P. XML Schema Part 0: Primer Second Edition, 2004. W3C Recommendation. Disponível em: <<http://www.w3.org/TR/xmlschema-0>>. Acesso em: 27 de fevereiro de 2006.

- FIELDING, R.T. **Architectural Styles and the Design of Network-based Software Architectures**. 2000. 162p. Ph.D. Thesis, University of California, Irvine. Irvine, California.
- FOUNTAS, S.; BLACKMORE, S.B.; PETERSEN, S.M. A new methodology for decision analysis on precision farming based on user's experience. In: INTERNATIONAL CONFERENCE ON PRECISION AGRICULTURE, 6., 2002, Wisconsin. **Proceedings**. Madison, ASA/CSSA/SSSA, 2002. p.1640-1654.
- GARLAN, D.; SHAW, M. **An introduction to software architecture**. Ambriola & Tortola (eds.), Advances in Software Engineering & Knowledge Engineering, v. II, World Scientific Pub Co., Singapore, 1993, p.1-39.
- GSDI. GSDI Cookbook version 2.0. Developing Spatial Data Infrastructures: The SDI Cookbook. 2004. Disponível em: <<http://www.gsdi.org/gsdicookbookindex.asp>>. Acesso em: 27 de março de 2006.
- HUHNS, M.; SINGH, M.P. Service-Oriented Computing: Key Concepts and Principles. **IEEE Internet Computing**, v.9, n.1, p.75-81. 2005.
- JAZAYERI, M.; RAN, A.; LINDEN, F. **Software Architecture for Product Families**. Addison Wesley, 2000
- JBOSS. JBoss Application Server. Disponível em: <<http://www.jboss.com/products/jbossas/>>. Acesso em: 08 de outubro de 2005.
- JCP. Java Community Process. Community Development of Java Technology Specifications. Disponível em: <<http://jcp.org/en/home/index>>. Acesso em: 25 de abril de 2006.
- JENSEN, A.L. Building a web-based information system for variety selection in field crops - objectives and results. **Computers and Electronics in Agriculture**. v.32, p.195-211. 2001.
- JENSEN, A.L. et al. Pl@nteInfo - a web-based system for personalised decision support in crop management. **Computers and Electronics in Agriculture**. v.25, p.271-293. 2000.
- JETSPEED. Apache Portals - The Apache Software Foundation, 2005. Disponível em: <<http://portals.apache.org/>>. Acesso em: 27 de fevereiro de 2006.
- JMS. Java Message Service Tutorial. Sun Developer Network, Disponível em: <<http://java.sun.com/products/jms/tutorial/>>. Acesso em: 01 de outubro de 2005.
- JMX. Java Management Extensions. Sun Developer Network, Disponível em: <<http://java.sun.com/products/JavaManagement/>>. Acesso em: 08 out. 2005.
- KEEN, M. et al. **Patterns: Implementing an SOA Using an Enterprise Service Bus**. IBM redbooks. 2004.
- KORDUAN, P. Standardization in data management to increase interoperability of spatial precision agriculture data. In: EUROPEAN CONFERENCE ON PRECISION AGRICULTURE, 4., 2003, Berlin. **Proceedings**. Berlin, 2003. p.323 - 328.

- KORDUAN, P.; BILL, R.; BÖLLING, S. An Interoperable Geodata Infrastructure for Precision Agriculture. In: AGILE CONFERENCE ON GEOGRAPHIC INFORMATION SCIENCE, 7., 2004, Berlin. **Proceedings**. Greece, 2004. p.747-751.
- KOROTKIY, M. 2005. Towards an Ontology-enabled Service Oriented Architecture. In: IBM PHD STUDENT SYMPOSIUM IN INTERNATIONAL CONFERENCE ON SERVICE ORIENTED COMPUTING, 3., 2005, Netherlands. **Proceedings**. Amsterdam, 2005. p. 1-6.
- KREGER, H. Fulfilling the Web Services Promise. **Communications of the ACM**, v.46, n.6, p.29-34. 2003.
- KRUCHTEN, P. The 4+1 View Model of architecture. **IEEE Software**, v.12, n.6, p.42-50. 1995.
- KRUCHTEN, P. **The Rational Unified Process: An Introduction**, Third Edition. Addison-Wesley Professional, 2003.
- LÜTTICKEN, R.E. Development of Internet-Based Communication and Information Network to Progress the Implementation of Precision Agriculture. In: INTERNATIONAL CONFERENCE ON PRECISION AGRICULTURE, 5., 2000, Minnesota. **Proceedings**. Bloomington, ASA/CSSA/SSSA, 2000. 14p. 1 CD-ROM.
- MAPSERVER. Ambiente para desenvolvimento de aplicações para manipulação de dados espaciais para Internet. Disponível em: <<http://mapserver.gis.umn.edu>>. Acesso em: 01 de março de 2006.
- MASSRUHÁ, S.M.F.S., et al. FMS: A software development environment for farm management systems. In: EUROPEAN CONFERENCE FOR INFORMATION TECHNOLOGY IN AGRICULTURE, 1., 1997, Copenhagen. **Proceedings**. Copenhagen :The Royal Veterinary and Agricultural University, 1997. p.477-482.
- MASSRUHÁ, S.M.F.S; MEIRA, C.A.A.; FILETO, R. A framework for the development of the farm management systems. In: INTERNATIONAL CONFERENCE ON COMPUTERS IN AGRICULTURE, 7., 1998, Michigan. **Proceedings**. St. Joseph, ASAE, 1998. p.284-291.
- MENEGATTI, L.A.A. **Metodologia para Identificação, Caracterização e Remoção de Erros de Mapas de Produtividade**. 2002. 84p. Dissertação (Mestrado). Escola Superior de Agricultura "Luiz de Queiroz", Universidade de São Paulo, São Paulo.
- MESTERHÁZI, P.Á. et al. Data Transfer among Precision Farming Systems. In: 2002 ASAE ANNUAL INTERNATIONAL MEETING/CIGR XVTH WORLD CONGRESS, 2002. Chicago. **Proceedings**. Illinois. Paper 021047. 2002. 11p.
- MITRA, N. et al. SOAP Version 1.2 Part 0: Primer. W3C Recommendation 24 June 2003. Disponível em: <<http://www.w3.org/TR/soap12-part0/>>. Acesso em: 27 de fevereiro de 2006.

- MOLIN, J.P.; MENEGATTI, L.A.A. Methodology for identification, characterization and removal of errors on yield maps. In: 2002 ASAE ANNUAL INTERNATIONAL MEETING/CIGR XVTH WORLD CONGRESS, 2002, Chicago. **Proceedings**. Illinois. 2002.
- MURAKAMI, E. et al. A Solution for Interoperability of Precision Agriculture Applications using Web Services. In: EUROPEAN CONFERENCE ON PRECISION AGRICULTURE, 4., 2003, Berlin. **Proceedings**. Berlin, 2003a. p.511-512.
- MURAKAMI, E. et al. An Infrastructure for Development of Information System for Precision Agriculture. In: INTERNATIONAL CONFERENCE ON PRECISION AGRICULTURE AND OTHERS PRECISION RESOURCES MANAGEMENT, 6., 2002, Minneapolis. **Proceedings**. USA, 2002. p.1712-1722. 1 CD-ROM.
- MURAKAMI, E. et al. Desenvolvimento de Software para Agricultura de Precisão Baseado em Componentes Interoperáveis. In: CONGRESSO BRASILEIRO DA SOCIEDADE BRASILEIRA DE INFORMÁTICA APLICADA À AGROPECUÁRIA E À AGROINDÚSTRIA, 4., 2003, Porto Seguro. **Anais**. Porto Seguro, 2003b. 5p. 1 CD-ROM.
- NÖLLE, O. Geo-Services: Optimizing Agricultural Business Processes. In: EC GI & GIS WORKSHOP, ESDI STATE OF THE ART, 10., 2004, Poland. **Proceedings**. Warsaw, 2004. 8p.
- OGC. Open Geospatial Consotium. OpenGIS Specifications. Disponível em: <<http://www.opengeospatial.org/specs/>>. Acesso em: 03 de outubro de 2006.
- PASLEY, J. How BPEL and SOA are changing Web Services development. **IEEE Internet Computing**, v.9, n.3, p.60-67. 2005.
- PEDERSEN, S.M., et al. Adoption of precision farming in Denmark. In: EUROPEAN CONFERENCE ON PRECISION AGRICULTURE, 4., 2003, Berlin. **Proceedings**. Berlin, 2003. p.533-538.
- PERCIVALL, G. OGC Reference Model, version 0.1.3, 2003. Open Geospatial Consortium Inc. Disponível em: <<http://www.opengeospatial.org/specs/?page=orm>>. Acesso em: 27 de fevereiro de 2006.
- PERRY, D.E.; WOLF, A.L. Foundations for the study of software architecture. **ACM SIGSOFT Software Engineering Notes**, v.17, n.4, p.40-52. 1992.
- PLANT, R.E. Site-specific management: the application of information technology to crop production. **Computers and Electronics in Agriculture**, v.30, p.9-29, 2001.
- QUINN, B.; SHUTE, D. **Windows Sockets Network Programming**. Addison-Wesley. 1995.
- RAFOSS, T. et al. Draft proposal standard: XML-schema for agrometeorological data. In: EFITA/WCCA JOINT CONGRESS ON IT IN AGRICULTURE, 2005, Portugal. **Proceedings**. Vila Real, 2005. p.490-497.

- REZENDE, D.A.; ABREU, A.F. **Tecnologia da Informação Aplicada a Sistemas de Informação Empresariais**. Atlas, 2000.
- RIBEIRO, L.C.M. et al. A geo-spatial database for precision agriculture information systems. In: EUROPEAN CONFERENCE ON PRECISION AGRICULTURE, 4., 2003, Berlin. **Proceedings**. Germany, 2003. p.543-544. 1 CD-ROM.
- RIBEIRO, L.C.M.; SARAIVA, A.M.; MURAKAMI, E. Componentes Espaço-Temporais para Agricultura de Precisão. In: CONGRESSO BRASILEIRO DE AGRICULTURA DE PRECISÃO, 2004, Piracicaba. **Anais**. São Paulo, 2004. 1 CD-ROM.
- RIBEIRO, L.C.M.; SARAIVA, A.M.; MURAKAMI, E. Modelagem de Banco de Dados Espaço-Temporal Para Agricultura De Precisão. In: CONGRESSO BRASILEIRO DA SOCIEDADE BRASILEIRA DE INFORMÁTICA APLICADA À AGROPECUÁRIA E AGROINDÚSTRIA, 4., 2003, Porto Seguro. **Anais**. Bahia, 2003. 1 CD-ROM.
- ROBERT, P.C. Precision Agriculture: research needs and status in the USA. In: EUROPEAN CONFERENCE ON PRECISION AGRICULTURE, 2., 1999, Danemark. **Proceedings**. SCI/Sheffield Academic Press, 1999. p.19-33.
- SARAIVA, A.M. **Tecnologia da Informação na Agricultura de Precisão e Biodiversidade: Estudos e proposta de Utilização de Web Services para Desenvolvimento e Integração de Sistemas**. 2003. Tese (Livre Docência) – Escola Politécnica, Universidade de São Paulo. São Paulo, 2003. 187p.
- SARAIVA, A.M. **Um Modelo de Objetos para Sistemas Abertos de Informações de Campo para Agricultura de Precisão – MOSAICO**. 1998. 235p. Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo, São Paulo, 1998.
- SARAIVA, A.M.; MASSOLA, A.M.A.; CUGNASCA, C.E. An object model for field information systems. In: INTERNATIONAL CONFERENCE ON PRECISION AGRICULTURE, 4., 1998, St. Paul. **Proceedings**. Minnesota, ASA/CSSA/SSSA, 1998. p.1355-1366.
- SARAIVA, A.M.; MASSOLA, A.M.A.; PAZ, S.M. Object oriented approach to the development of a field information system. In: ANNUAL INTERNATIONAL MEETING OF THE AMERICAN SOCIETY OF AGRICULTURAL ENGINEERS, 1997, Minneapolis. **Proceedings**. Minnesota, ASAE, 1997. paper 97-3015
- SAWYER, B.J.; BOWEN, C.M.; MCNEIL, R.L. Precision Agriculture Software for Physical Record-keeping and Field Mapping 1999 Update. Report Prepared for the Canada-Alberta Farm Business Management Program. LandWise Inc. March 31, 1999. Disponível em: <<http://www.landwise.ca/SoftComp.shtml>>. Acesso em: 30 de junho de 2003.
- SEIDL, M.S., et al. GIS-Crop Model Decision Support System to Evaluate Corn and Soybean Prescriptions. **American Society of Agricultural Engineers**. v.17, p.721-728. 2001.

- SHANNON, B. Java™ 2 Platform, Enterprise Edition Specification, v1.4, 2003. Disponível em: <<http://java.sun.com/j2ee/1.4/docs/index.html>>. Acesso em: 27 de fevereiro de 2006.
- SHAPIRO, M. Structure and encapsulation in distributed systems: The proxy principle. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS, 6., 1986, Cambridge. **Proceedings**. Cambridge, 1986. p.198–204.
- SHAW, M. et al. Abstractions for software architecture and tools to support them. **IEEE Transactions on Software Engineering**, v.21, n.4, p.314–335. 1995.
- SHAW, M., CLEMENTS, P. A field guide to boxology: Preliminary classification of architectural styles for software systems. In: INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE, 21., 1997, Washington. **Proceedings**. Washington, 2005. p.6–13.
- SINHA, A. Client-server computing. **Communications of the ACM**, v.35 n. 7, p.77–98. 1992.
- SODHI, B. Implement a customizable ESB with Java. Disponível em: <<http://www.javaworld.com/javaworld/jw-08-2005/jw-0808-esb.html>>. Acesso em: 27 de fevereiro de 2006.
- SØRENSEN, C.G., et al. Information sources and decision making on precision farming. In: INTERNATIONAL CONFERENCE ON PRECISION AGRICULTURE, 6., 2002, Wisconsin. **Proceedings**. Madison, ASA/CSSA/SSSA, 2002. p.1683-1695.
- STAFFORD, J.V. Implementing Precision Agriculture in the 21st Century. **J. Agric. Engng. Res.**, v.76, p.267-275, 2000.
- SUGUMARAN, R. Development of an integrated range management decision support system. **Computers and Electronics in Agriculture**. v.37, p.199-205, 2002.
- UMAR, A. **Object-Oriented Client/Server Internet Environments**. Prentice Hall PTR, 1997.
- VESPER. Australian Centre Precision Agriculture. User Manual Vesper 1.6. Disponível em: <<http://www.usyd.edu.au/su/agric/acpa/vesper/vesper.pdf>>. Acesso em: 10 de janeiro de 2006.
- W3C. World Wide Web Consortium. Disponível em: <<http://www.w3.org/Consortium/about-w3c>>. Acesso em: 25 de abril de 2006.
- WANG, D., et al. A Grid Service Module for Natural-Resource Managers. **IEEE Internet Computing**, v.9, n.1, p.35-41. 2005.
- WebAGRIIS. Ferramenta para criação de AGRIS AP XML. Disponível em: <http://www.fao.org/agris/tools/WebAGRIIS/WebAGRIIS_app.htm>. Acesso em: 27 de fevereiro de 2006.

- WELCH, S.M., et al. PCYield: Model-based support for soybean production. **Agricultural Systems**. v.74, p.79-98. 2002.
- WHELAN, B.M.; MCBRATNEY, A.B.; MINASNY, B. Vesper – Spatial prediction software for precision agriculture. In: EUROPEAN CONFERENCE ON PRECISION AGRICULTURE, 3., 2001, Montpellier. **Proceedings**. Montpellier: Ecole Nationale Agronomique de Montpellier, 2001. p. 139-144.
- WHELAN, B.M.; MCBRATNEY, A.B.; MINASNY, B. Vesper 1.5 - spatial prediction software for precision agriculture. In: INTERNATIONAL CONFERENCE ON PRECISION AGRICULTURE, 6., 2002, Wisconsin. **Proceedings**. Madison, ASA/CSSA/SSSA, 2002. p. 14. 1 CD-ROM.
- XMLBEANS. Apache XMLBeans. Disponível em: <<http://xmlbeans.apache.org/>>. Acesso em: 01 de outubro de 2005.
- ZENG, H. et al. A Web-based Simulation System for Transport and Retention of Dissolved Contaminants in Soil. **Computers and Electronics in Agriculture**, v.33, p. 105-120, 2002.
- ZHANG, N.; WANG, M.; WANG, N. Precision agriculture - a worldwide overview. **Computers and Electronics in Agriculture**, v.36, p. 113-132, 2002.