Serviços Baseados em Localização e Publish-Subscribe no Domínio da Pecuária de Precisão Computacional

Márcio Aparecido Inacio da Silva

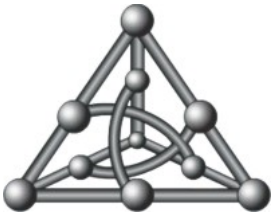
Dissertação de Mestrado

Orientação:Prof. Dr. Marcelo Augusto Santos Turine

Co-orientação:Prof. Dra. Hana Karina Salles Rubinsztejn

Área de Concentração: Engenharia de Software

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, Curso de Pós-Graduação em Ciência da Computação, Faculdade de Computação da Fundação Universidade Federal de Mato Grosso do Sul.



Faculdade de Computação
Universidade Federal de Mato Grosso do Sul
29 de Julho de 2011

Aos heróis da minha vida,
os meus pais

Agradecimentos

À minha amada família por seu apoio e presença em todos os momentos. Em especial, aos meus pais, Manoel e Severina, que sempre foram o exemplo a ser seguido e ao meu irmão Marcelo pelas lições de todos os dias.

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

Ao meu orientador professor Marcelo Turine, pela atenção e confiança no decorrer e conclusão deste trabalho, sem as quais ele não seria realidade.

À minha co-orientadora professora Hana Karina, pela atenção, confiança e por ter contribuído no desenvolvimento deste trabalho.

Aos meus amigos, sempre presentes nos momentos em que mais precisei.

A todos os membros do Laboratório de Engenharia de Software (LEDES) que colaboraram com o desenvolvimento deste projeto e aos amigos que lá fiz.

Aos amigos do curso de Ciência da Computação com os quais criei fortes laços de amizade.

A todos os professores e funcionários da Faculdade de Computação da Universidade Federal de Mato Grosso do Sul (FACOM/UFMS), que me acolheram sempre que precisei.

À Capes pelo apoio financeiro para realização deste mestrado.

Resumo

A abordagem de sistemas sensíveis ao contexto com integração de Serviços Baseados em Localização (LBS) e *Publish-Subscribe* (Publicar-Assinar) é uma das oportunidades para atender as necessidades de acesso a ambientes computacionais úbiquos. No entanto, o projeto e a implementação desses sistemas são complexos, principalmente, pela falta de suporte de ferramentas automatizadas da Engenharia de Software, que se preocupa em propor técnicas e métodos para auxiliar o desenvolvimento ágil de softwares de maior qualidade e em menor tempo, reutilizando artefatos já especificados e implementados. O objetivo principal desta pesquisa é especificar e implementar um componente de software genérico intitulado Orion utilizando uma abordagem de componentes, que tem como objetivo gerenciar e distribuir, de forma síncrona ou assíncrona, notificações ou avisos sobre quaisquer tipos de eventos. Para ser possível localizar os usuários em tempo real, será especificado, implementado e integrado ao Orion o software intitulado mobile Eros, que tem como objetivo capturar dados de contexto do usuário. Para testar e validar os componentes e a arquitetura proposta, será desenvolvida um estudo de caso aplicado à plataforma e-SAPI *bovis* a fim de auxiliar o processo de rastreabilidade bovina e Pecuária de Precisão no Brasil. A aplicação de Tecnologias de Informação e Comunicação (TIC) no sistema produtivo de carne bovina e a adoção das Boas Práticas Agropecuárias (BPAs) objetivam garantir a produção de alimentos seguros e com atributos de qualidade que atendam aos interesses dos grandes mercados. Com os constantes embargos europeus à carne brasileira, é relevante que a adoção e o uso de um sistema de gestão de eventos sanitários e riscos contribui para a tomada de decisões em busca de uma melhoria da vigilância sanitária no Brasil. Ao descobrir um foco de doença, será possível propagar este evento como um alerta para as autoridades responsáveis, como as agências reguladoras, institutos de defesa, o MAPA (Ministério da Agricultura, Pecuária e Abastecimento) e demais interessados no problema da rastreabilidade bovina. Assim, como resultado final deste projeto, conclui-se que o software Orion é um mecanismo eficiente e eficaz de disseminação de eventos originados pela plataforma e-SAPI *bovis*, podendo ser extensível e aplicável a diversos outros domínios.

Palavras-chave: Reuso de Software, Serviços Baseados em Localização, Rastreabilidade, *Publish-Subscribe*, Computação Ubíqua, Carne Bovina.

Abstract

The context-aware system approach integrated with Location-Based Services (LBS) and Publish-Subscribe is becoming an opportunity to attend the need to access computational environment, anytime and anywhere. However, this construction is struggling for lack of support from software engineering that is concerned to propose tools and techniques to help design and agile software development, higher quality in shortest time. Many products are developed on the basis of artifacts already specified and implemented using techniques for reuse. The main objective of this research is to specify and implement a software component for the platform e-SAPI *bovis* called Orion, to assist the process of tracing cattle in Brazil. The beef production system becomes more structured and consolidated due to the motivation given by the economy of scale to the intensive use and application of Information and Communication Technologies (ICT) in food traceability. The adoption of Good Manufacturing Practices (BPP) aims to ensure safe food production and quality attributes that meet the interests of large markets. With those european embargoes on Brazilian beef, it is important that the system discovers an outbreak of disease and spread it as a warning to the responsible authorities, as regulatory agencies, institutes of defense, the MAPA (Ministry of Agriculture, Livestock and Supply) and others interested in the problem of bovine traceability. Such notice must be delivered to stakeholders wherever they are, speeding the process of decision-making. Orion will be a mechanism for dissemination of events generated by the platform e-SAPI *bovis* as caused by third-party software, using Web Services as communication mechanism.

Keywords: Reuse, Location-Based Services, Traceability, Publish-Subscribe, Ubiquitous Computing, beef.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Siglas

3G Terceira Geração da Telefonia Móvel

ABIEC Associação Brasileira das Indústrias Exportadoras de Carnes

A-GNSS *Assisted Global Navigation Satellite System*

A-GPS *Assited Global Positioning System*

CETIC Centro de Estudos sobre as Tecnologias da Informação e da Comunicação

CORBA *Common Object Request Broker Architecture*

DDS *Data Distribution Service*

EAD Educação a distância

e-GTA Guia de Trânsito Animal Eletrônica

e-SAPI Sistema Eletrônico Agropecuário de Produção Integrada

FACOM Faculdade de Computação da UFMS

FIFO *First In First Out*

GNSS *Global Navigation Satellite System*

GPS *Global Positioning System*

GTA Guia de Trânsito Animal

HTTP *Hypertext Transfer Protocol*

JMS *Java Message Service*

Kbps Kilobits por segundo

Koala *C[K]omponent Organizer and Linking Assistent*

LBS *Location-Based Services*

MAPA Ministério da Agricultura, Pecuária e Abastecimento

OLTP *Online Transaction Processing*

OMG *Object Management Group*

PDA Personal Digital Assistant

pub-sub *Publish-Subscribe*

QR code *Quick Response code*

REST *Representational State Transfer*

RUP *Rational Unified Process*

RTD *Round-Trip Delay*

RTT *Round-Trip Time*

SAPI Sistema Agropecuário de Produção Integrada

SHA-1 *Secure Hash Algorithm*

SISBOV *Serviço de Rastreabilidade da Cadeia Produtiva de Bovinos e Bubalinos*

SMS *Short Message Service*

SOA *Service Oriented Architecture*

SOAP *Simple Object Access Protocol*

TI Tecnologia da Informação

UbiComp Computação Ubíqua

UFMS Universidade Federal de Mato Grosso do Sul

UML *Unified Management Language*

URI *Uniform Resource Identifier*

XML *Extensible Markup Language*

WADL *Web Application Description Language*

Wi-Fi *Wireless Fidelity*

WSN Wireless Sensor Network

WSDL *Web Service Description Language*

Capítulo 1

Introdução

1.1 Considerações Iniciais

O crescimento do mercado de software nas mais diversas áreas de aplicação impõe a necessidade de desenvolvimento de software de alta qualidade por meio de métodos, técnicas e ferramentas de Engenharia de Software. Alto custo, alta complexidade, rápida evolução das tecnologias de informação e comunicação, dificuldade de manutenção, disparidade entre os requisitos dos usuários e o produto desenvolvido, falta de planejamento e gerência de projetos de software têm motivado o surgimento de novas técnicas para o processo desenvolvimento de software.

Há um tempo atrás, não era previsível levar um computador para qualquer lugar e ainda com acesso a Internet. O avanço das novas tecnologias, técnicas e linguagens para o desenvolvimento de novos produtos de software, proporcionaram o acesso rápido à informação, revolucionando a comunicação entre as pessoas. Em paralelo, com o avanço da telefonia celular surgiu uma maior mobilidade aos usuários, possibilitando em qualquer lugar receber, realizar chamadas, enviar imagens e dados.

Assim, novas perspectivas começaram a surgir da união de mobilidade com acesso à Internet. As redes de telefonia móvel também evoluíram, fornecendo o acesso à Internet para os celulares. Porém, existe a necessidade de prover serviços e softwares para toda esta potencialidade de comunicação.

A heterogeneidade entre os atuais dispositivos móveis[1], no que se refere a configuração de hardware, ainda é um desafio para os sistemas que provêem conteúdo. Todo conteúdo antes de ser enviado aos usuários deve ser customizado, levando-se em consideração o contexto[2] do dispositivo utilizado [59].

Heterogeneidade, segundo Costa[19], é apenas um dos desafios inerentes à *Computação Ubíqua* ou UbiComp. Weiser [78] afirma que na *Computação Ubíqua*, a tecnologia da

[1]Dispositivos de baixa capacidade de processamento e armazenamento comparados a computadores comuns, como por exemplo: celulares, PDAs e *smartphones*.

[2]Contexto é qualquer informação que possa ser usada para caracterizar a situação de uma pessoa, lugar ou objeto relevante à interação entre usuário e software.

informação deve ser mais integrada ao cotidiano, tornando-se transparente e auxiliando os usuários a lidar com a sobrecarga diária de informações.

As aplicações ubíquas precisam adaptar-se ao ambiente, compreendendo o contexto em que estão inseridas. Essa classe de sistemas computacionais, adaptativos ao contexto, proporciona o desenvolvimento de aplicações mais ricas, elaboradas e complexas, que exploram a natureza dinâmica e a mobilidade do usuário [69]. Entretanto, o desenvolvimento de aplicações continuamente adaptadas ao ambiente que permaneçam funcionando mesmo quando o indivíduo se movimentar ou trocar de dispositivo [19], é tema de várias pesquisas científicas.

Segundo Weiser [78], a *Computação Ubíqua* é a terceira geração da computação. A primeira geração foi marcada pelos *mainframes* (um computador para muitos), a segunda geração é a dos computadores pessoais (um computador para uma pessoa). Já na terceira geração, são muitos computadores para uma pessoa que interagem entre si para prover serviços ao usuário.

Eugster [25, 26] afirma que com o aumento do número de computadores/dispositivos interconectados relatado por Weiser, faz-se necessário a adoção de abstrações de programação para uma comunicação remota muito mais precisa, ou seja, que minimize os erros de comunicação. O paradigma defendido é o *Publish-Subscribe* (Publicar-Assinar), ou simplesmente pub-sub[3], onde a comunicação é anônima, assíncrona e *multicast*[4] por natureza. Além disso, o sistema é capaz de adaptar-se rapidamente em um ambiente dinâmico [35], que é proporcionado pelas mudanças das informações de contexto do usuário, detectada pelo ambiente ubíquo readequando o serviço de acordo com o novo contexto.

A mobilidade é uma característica relevante na *Computação Ubíqua*, pois a computação deve ocorrer em qualquer lugar (computação onipresente). Desta forma, existe a necessidade de conhecer a localização ou posição do usuário para que os serviços providos a ele sejam adaptados de acordo com seu posicionamento. Tais serviços são denominados Serviços Baseados em Localização (*Location-Based Services* - LBS)[5] [58, 63, 11, 37].

Com pub-sub e LBS é possível desenvolver diversas soluções computacionais relevantes para a sociedade. Um exemplo de grande utilidade pública é a utilização de uma Rede de Sensores sem Fio [34](WSN - *Wireless Sensor Network*) para monitorar o clima e prever a formação de furacões, terremotos e *tsunamis* [46, 27]. Estes sensores podem enviar dados como temperatura, umidade do ar e velocidade do vento, e os pesquisadores podem investigar e prever tópicos onde: temperatura maior que 30°C, umidade do ar igual a 20% e velocidade do vento maior que 50 KM/h é definido como perigo iminente de furacão. Todos os usuários que se inscreverem para receber notícias neste tópico receberão o comunicado "Perigo iminente de furacão"assim que os valores descritos forem constatados. Outro exemplo, é um supermercado X que deseja enviar notificações aos celulares de clientes toda vez que estes se encontrarem nas imediações do estabelecimento. Estes são alguns exemplos de aplicações em redes WSN, mas não é foco deste trabalho.

[3]Neste trabalho será utilizado o termo pub-sub em todas as citações.

[4]Forma de comunicação onde há o envio de mensagens de um para muitos.

[5]O termo LBS será empregado em todas as citações neste trabalho como abreviação de Serviços Baseado em Localização.

Portanto, muitas vezes cada organização cria seu próprio sistema de envio de notificações aos usuários interessados em determinado assunto, porém o retrabalho tem um alto custo [30]. Atualmente, muitos produtos de software são desenvolvidos em função de artefatos já especificados e implementados, utilizando técnicas de reutilização de software para melhorar a produtividade [74, 73, 8, 20], a manutenibilidade e a qualidade tanto do software quanto do processo de desenvolvimento [66, 5], foco do presente projeto de pesquisa.

Diante deste panorama e das tendências tecnológicas, várias pesquisas estão sendo realizadas a fim de propor LBS, Pub-Sub e tecnologias de reuso eficientes e eficazes para facilitar o processo de desenvolvimento de software no contexto da computação ubíqua. Surge assim, uma oportunidade para propor e explorar técnicas, arquitetura de software e ferramentas para especificar, projetar e implementar produtos de software reutilizáveis e com qualidade utilizando LBS e Pub-Sub e em vários domínios. Neste trabalho, o cenário de estudo de caso será a Cadeia Produtiva da Carne Bovina, área importante da economia brasileira que necessita de soluções de Tecnologia da Informação (TI) para manter o país como o maior exportador de carne bovina do mundo [61, 14, 1].

O problema de interesse neste trabalho é especificar e prototipar um mecanismo customizável e reutilizável de disparos de notificações, sem a complexidade ou rigidez presentes nos softwares comerciais disponíveis no mercado.

1.2 Motivações

Nos últimos anos ocorreram inúmeros avanços tecnológicos no que se refere à comunicação entre dispositivos móveis e redes de alta velocidade (redes 3G[6]) [23]. Segundo o Centro de Estudos sobre as Tecnologias da Informação e da Comunicação - (CETIC.br)[17], na última pesquisa realizada em 2008, o consumo de Internet em dispositivos móveis é predominante nas Classes A e B da população brasileira devido aos altos custos. No entanto, tanto a posse quanto o uso do celular cresceu quatro anos consecutivos e, 56% dos entrevistados têm celular na data da pesquisa em 2008.

Embora no Brasil o uso das redes de alta velocidade ainda seja para poucos, Dekleva [23] afirma que a difusão destas tecnologias na população acontecerá inevitavelmente em um futuro próximo. O Japão, por exemplo, introduziu redes 3G em 2001, em 2006 50% dos assinantes de banda larga para dispositivos móveis já possuíam 3G.

Vislumbrando todo o potencial dessas novas tecnologias, softwares e serviços devem ser desenvolvidos e oferecidos para a sociedade a fim de suprir a necessidade de informação, que é um insumo cada vez mais importante em áreas como o Agronegócio, seja na produção propriamente dita, ou para a coordenação das cadeias produtivas. A inserção competitiva do Agronegócio brasileiro nos mercados mundiais exige, cada vez mais, que a informação ao longo de toda a cadeia esteja disponível, seja por questões de rastreabilidade dos produtos, seja para a coordenação das ações pelas várias esferas, inclusive as governamentais

[6] É a terceira geração de celulares com acesso à altas velocidades - acima de 300 kbps - para transmissão de dados (ex.: para teleconferências). A primeira geração foi analógica e a segunda, digital.

[62].

O desenvolvimento de novas ferramentas que auxiliem o processo de rastreabilidade bovina e Vigilância Sanitária no Brasil torna-se necessário para que o país se mantenha como grande exportador de carne bovina [1]. Neste contexto, desde de 2008 a Faculdade de Computação (FACOM) e a Universidade Federal de Mato Grosso do Sul (UFMS) em parceria com a Embrapa Gado de Corte desenvolveu a plataforma e-SAPI *bovis*[61]. Trata-se de um sistema Web de planejamento, gestão e avaliação de informações relativas à produção pecuária bovina nacional, com a proposta de assegurar confiabilidade e agilidade no acompanhamento de informações relativas à produção por parte de todos os atores envolvidos na cadeia produtiva da carne bovina.

A criação desta plataforma motivou a elaboração deste projeto de mestrado, que objetiva agregar mais funcionalidades e estender seu uso para dispositivos móveis. A proposta da plataforma e-SAPI *bovis* prover conteúdo para este novo tipo de dispositivo motivou o estudo do trabalho de Rubinsztejn [59]. Rubinsztejn implementa um mecanismo de customização de conteúdo, que leva em consideração o contexto do usuário (geoposicionamento, bateria do dispositivo, resolução de tela, memória, etc). O estudo deste trabalho levou à compreensão de como pode ser implementado para o problema da Pecuária de Precisão Computacional. O suporte à adaptação de conteúdo sensível a contexto para dispositivos móveis em sistemas Pub-Sub viabiliza o uso da plataforma, tanto para celulares com poucos recursos de processamento e armazenamento quanto para poderosos *smartphones*.

1.3 Objetivos

O objetivo principal desse trabalho é especificar e desenvolver um componente Pub-Sub integrado com LBS denominado **Orion**[7] utilizando uma abordagem baseada em componentes de software e genérica, ou seja, possível de ser customizada e independentemente de qualquer domínio ou área de aplicação. Orion deverá ser capaz de interagir com outros softwares, por meio do padrão XML[8] (*Extensible Markup Language*), como forma de troca de dados entre as aplicações. O componente será de fácil extensão e adaptação para diferentes domínios. Para captação de dados de contexto do usuário será desenvolvido um software para dispositivos móveis denominado *mobile* **Eros**[9].

Orion será desenvolvido para ser utilizado na Web, onde será possível escolher qual o assunto fornecido para o usuário receberá notificações, além de definir regras para recebimento destas notificações. Eros, será desenvolvido para ser utilizado em dispositivos móveis, fornecendo dados importantes de contexto do usuário, como localização, memória, bateria, capacidade de processamento, entre outros dados de hardware. Através do software Eros o usuário visualizará as notificações que recebeu oriundas de Orion, além

[7] Orion é uma constelação reconhecida em todo o mundo, por incluir estrelas brilhantes e visíveis de ambos os hemisférios.

[8] Linguagem de marcação especialmente criada para armazenar dados. É muito utilizada para transferir dados entre sistemas distintos.

[9] Deus da mitologia grega, deus do Amor.

de realizar tarefas de propósito específico do domínio abordado neste trabalho, a Pecuária.

Os objetivos específicos do projeto são:

- Desenvolver um *Web Service* para ser um componente de interface entre a arquitetura proposta e aplicações externas que se interessem em prover conteúdo à Plataforma e-SAPI. Será possível que tanto usuários (*subscribers* ou assinantes) quanto empresas (*publishers*) se cadastrem no componente Orion;
- Desenvolver o software para *mobile* **Eros**, responsável por realizar um subconjunto de operações providas por Orion e obtenção de informações de contexto;
- Implementar no software Eros métodos de localização baseados em GPS (*Global Positioning System*) e pela triangulação de antenas de telefonia;
- Especificar padrão XML para transferência de dados entre softwares de terceiros e o componente Orion;
- Especificar e implementar método de filtragem de notificações para o componente Orion;
- Testar, avaliar e validar o software *mobile* Eros;
- Integrar o componente Orion com a Plataforma e-SAPI *bovis*; e
- Testar, avaliar e validar o componente Orion com a Plataforma e-SAPI *bovis*.

1.4 Organização do Texto

Este texto está dividido em 5 capítulos. Neste Capítulo foram apresentados o problema a ser investigado, o contexto, as motivações e os objetivos principais do presente trabalho. No Capítulo 2 são apresentados conceitos e definições acerca dos paradigmas *Publish-Subscribe*, LBS e Computação Ubíqua. No Capítulo 3 são apresentadas as vantagens do desenvolvimento de software baseado em componentes focando na reusabilidade e um estudo dos principais modelos de componentes de software existentes na literatura. No Capítulo 4 é apresentada a Plataforma e-SAPI *bovis*, que será utilizada como estudo de caso deste projeto, mostrando seu histórico, objetivos, arquitetura e funcionamento. No Capítulo 5 é apresentada a arquitetura e tecnologias do componente *Publish-Subscriber* e LBS. No Capítulo 6 é descrito a aplicação do software *mobile* Eros e Orion aplicado no estudo de caso da Plataforma e-SAPI *bovis*. As considerações finais e resultados do trabalho são apresentados no Capítulo 7. Por fim, são apresentadas as principais referências bibliográficas consultadas.

Capítulo 2

Sistemas Pub-Sub e LBS

2.1 Considerações Iniciais

Em 2008, o Centro de Estudos sobre as Tecnologias da Informação e da Comunicação - (CETIC.br)[17], publicou uma matéria relatando o crescimento no número de pessoas que fazem a aquisição de dispositivos móveis. Aliado a este fato, a expansão das redes de alta velocidade para dispositivos móveis como 3G e *Wi-Fi* forneceu uma infraestrutura importante para o desenvolvimento de sistemas que auxiliam no cotidiano das pessoas.

A união de sistemas pub-sub e LBS podem fornecer serviços que vão desde notícias a avisos de catástrofes da natureza [27, 46]. Neste capítulo são apresentados conceitos de Pub-Sub e LBS, bem como a relação entre esses paradigmas e a Computação Ubíqua.

2.2 Computação Ubíqua

Segundo Oliveira [48], um dos grandes tópicos de pesquisa e de desenvolvimento atual é a *Computação Ubíqua*. Esta tecnologia abrange essencialmente a forma pela qual os usuários percebem e utilizam seus dispositivos móveis para realizar as tarefas, a forma de criar e implementar estas aplicações assim como do "aumento"do ambiente físico pela disponibilidade e ubiquidade das informações e serviços.

Em uma visão ideal espera-se que os usuários não percebam sequer a existência de dispositivos, sendo o próprio ambiente modificado por equipamentos embarcados. Um dos pontos centrais da ubiquidade consiste na obtenção de informações sobre o contexto e sobre o usuário. Entre os elementos desta adaptação encontram-se a normalização e o intercâmbio de informações pessoais; os formalismos que permitam exprimir as regras de adaptação dos serviços em função do contexto e do usuário; a definição de contextos e de preferências dos usuários.

O conceito *Computação Ubíqua* é inerente aos sistemas pub-sub integrados com LBS, pois a aplicação responde às mudanças de localização de acordo com as preferências do

usuário. Sua definição muitas vezes pode ser confundida conceitualmente com a de *computação móvel* e *computação pervasiva*[7], como explicado a seguir e ilustrado na Figura 2.1:

- *Computação Móvel*: Os dispositivos computacionais possuem a capacidade de serem móveis, ou seja, o usuário pode levá-los para qualquer lugar sem que o dispositivo perca a capacidade de realizar atividades computacionais durante o período em que está se movendo. [7];
- *Computação Pervasiva:* O conceito de computação pervasiva implica na computação de forma invisível ao usuário, o dispositivo computacional é capaz de obter informações do ambiente no qual está inserido e ajustar a aplicação para melhor atender as necessidades do usuário [10];
- *Computação Ubíqua*: A computação Ubíqua une a mobilidade da *Computação Móvel* com a onipresença da *Computação Pervasiva*, ou seja, qualquer dispositivo computacional, enquanto em movimento, pode construir dinamicamente modelos computacionais dos ambientes nos quais o usuário se move, configurando seus serviços dependendo da necessidade [7, 10, 78]. Como por exemplo,s receber notícias de uma cidade que o usuário está visitando.

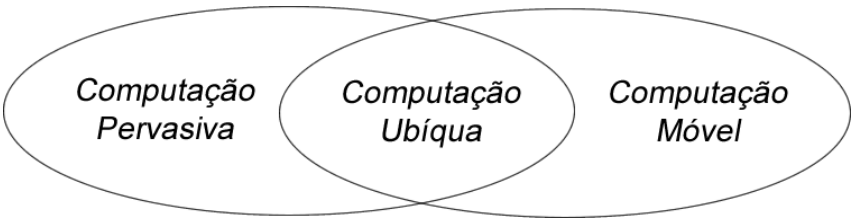


Figura 2.1: Relação entre os conceitos [7].

Da definição anterior nota-se que algo que é móvel não necessariamente é pervasivo, e o contrário também é verdadeiro. Os dispositivos que se aplicam o conceito de *Computação Ubíqua* por sua vez possuem um alto grau de mobilidade apresentando um alto grau de *Computação Pervasiva*.

2.3 Sistemas *Publish-Subscribe*

2.3.1 Conceitos

Os sistemas distribuídos necessitam de características efetivas para serem escaláveis, e dentre essas características destaca-se a necessidade de que a produção e consumo de informação seja assíncrono. Tal fato motivou a criação de um paradigma em que o consumidor é informado de um evento que é de seu interesse [43, 35, 26] disparado por um produtor. Esse paradigma recebeu o nome de *Publish-Subscribe*. Para facilitar a compreensão dos conteúdos subsequentes, os conceitos a seguir são fundamentais:

- ***Publisher*** é uma entidade que cria mensagens de notificação baseada em situações que ela pode detectar e traduzir para mensagens. Ex.: sistema de detecção de eventos sísmicos notifica usuários ou outros softwares;
- ***Subscriber*** é uma entidade que recebe notificações produzidas pelos *Publishers*;
- **Tópico** é um conceito usado para categorizar notificações, é o assunto no qual o *Subscriber* está interessado, também pode ser chamado de *Feed*, canal ou grupo. Ex.: notícias sobre esporte;
- **Assinar** é a ação do *Subscriber* manifestar interesse em receber notificações de um determinado **Tópico**;
- **Publicar** é ação de criar notificações/eventos realizada pelo *Publisher*;
- **Notificação/Evento** é o ato de transmissão de uma mensagem; e
- **Filtro** é uma consulta ou expressão lógica aplicada ao conteúdo das notificações. Este tipo de abordagem é mais específica que a utilização de tópicos, pois aplica um refinamento no recebimento de notificações. Ex.: Um usuário deseja receber notícias de esporte somente quando for de futebol e de seu time.

O pub-sub é um paradigma que consiste em três componentes principais: os que assinam (*subscribers*), os que publicam conteúdo (*publishers*) e uma infraestrutura para as notificações. Os *subscribers* expressam interesse em algum evento ou padrão de evento (como a publicação de notícias de esportes). *Publishers* são os que geram os eventos publicando seus conteúdos. A infraestrutura é responsável por capturar o evento e enviá-lo ao *subscriber* [43].

2.3.2 Modelo de Comunicação Básico

Segundo Eugster [26], o modelo de comunicação básico em sistemas pub-sub pode ser representado como ilustrado a seguir. Na Figura 2.2 está representada uma característica presente em serviços de eventos ou *Event Brokers*.

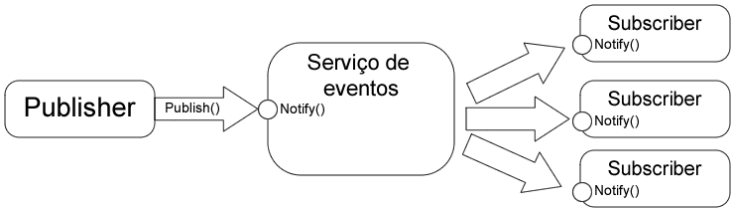


Figura 2.2: Desacoplamento de Espaço [26].

O Desacoplamento de Espaço indica que o *publisher* não necessita saber da existência dos *subscribers*, e o contrário também é verdadeiro. Portanto, são independentes.

Na Figura 2.3 é representada a interação entre o *publisher* e o *subscriber* chamada de Desacoplamento de Tempo. Neste caso, o *publisher* não necessita que o *subscriber* esteja

conectado para que ele possa publicar algum conteúdo. De forma análoga, o *subscriber* pode receber notificações que foram publicadas por um *publisher* que não está mais conectado.

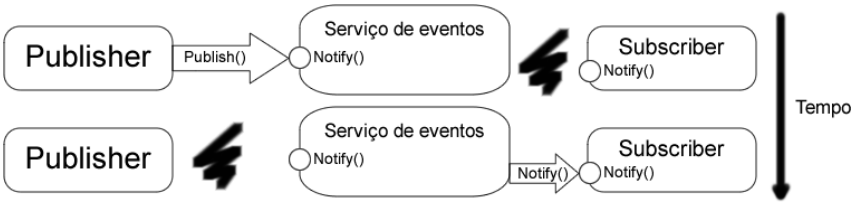


Figura 2.3: Desacoplamento de Tempo [26].

O Desacoplamento de Sincronismo, representado na Figura 2.4, indica que tanto a publicação quanto o recebimento de notificação acontecem de forma assíncrona, isto é, tanto quem publica e quem recebe não necessitam interromper o seu fluxo principal de execução para atender às solicitações.

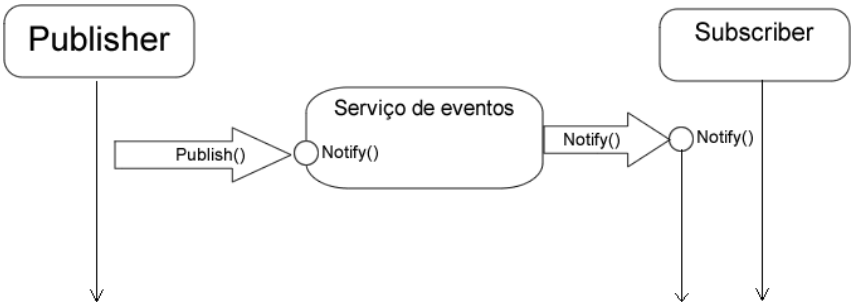


Figura 2.4: Desacoplamento de Sincronismo [26].

2.3.3 Sistemas *Topic-based* e *Content-based*

Os sistemas pub-sub podem ser classificados em baseado em tópicos (*Topic-based*) e/ou baseado em conteúdo (*Content-based*). No baseado em tópicos, os *subscribers* manifestam seu interesse assinando um tópico. As notificações geradas em um tópico serão enviadas a todos que o assinam [43, 26, 25].

Nos sistemas baseado em conteúdo, os *subscribers* especificam interesse por meio de filtros, que são consultas sobre o conteúdo dos eventos. Quando um evento é disparado um filtro é aplicado, se resultar em verdadeiro, este evento é encaminhado ao *subscriber* [43, 35, 25].

Das duas abordagens, a baseada em conteúdo é a mais complexa. Apesar de oferecer uma maior flexibilidade, demanda um maior esforço de projeto e implementação, pois existe uma variedade de filtros possíveis e a filtragem pode ser trabalhosa.

2.3.4 Exemplos de Softwares

Existe uma grande quantidade de softwares que implementam o paradigma Publish-Subscribe, cada um com suas especificidades e desenvolvido para uma ou mais linguagens de programação. Na Tabela 2.1 são destacados cinco dos principais softwares pub-sub do mercado, identificando os dados básicos referentes à linguagem de programação utilizada para implementação de softwares, as empresas desenvolvedoras e em qual categoria de pub-sub se enquadra.

Tabela 2.1: Sistemas Publish-Subscribe [28]

Nome	**Linguagem**	**Desenvolvedor**	**Expressividade**
Biztalk	.NET	Microsoft	*Content-based*
ActiveMQ	Java	Apache Software Foundation	(indefinido)
Tuxedo	C,C++ e COBOL	Oracle Corporation	*Content-based*
WebSphere MQ	C,COBOL,Perl e Java	IBM	(indefinido)
DDS	C,C++,C# e Java	Object Management Group	*Topic-based*

A partir da análise destes sistemas serão elencadas as melhores caracterísitcas para que sejam aplicadas ao componente **Orion**, e ao fim uma breve descrição de suas desvantagens com relação a aplicação no projeto de Orion.

BizTalk

O BizTalk, de propriedade da Microsoft, é um servidor de gerenciamento de processos de negócios que permite automatizar e otimizar os processos de negócios. Inclui poderosas e familiares ferramentas para o design, desenvolvimento e gerenciamento destes processos [40]. As ferramentas de desenvolvimento do BizTalk abrangem o paradigma SOA (*Service Oriented Architecture* [39]) e a arquitetura de estrutura básica .NET. Ferramenta poderosa, mas proprietária, não possui envio de avisos via Serviços de Mensagens Curtas (SMS), sem integração com LBS.

ActiveMQ

O ActiveMQ é desenvolvido pela *Apache Software Foundation* e basicamente é um *Middleware* para o JMS (*Java Message Service*), destinado a fazer o envio de mensagens a dois ou mais clientes. Também não possui integração com LBS e não é possível definir variáveis, como temperatura em um tópico que alertará em caso de furacão [6].

Tuxedo

O *Tuxedo* [53] foi originalmente desenvolvido pela AT&T em 1983 com o objetivo de criar e administrar sistemas de suporte a operações e processamento de transações online (OLTP). E em seguida adquirida pela Oracle em 2008 [28]. Este servidor não possui ferramentas para integração com dispositivos móveis assim como o BizTalk, e também como todos os anteriores não possui integração com LBS.

WebSphere MQ

O *WebSphere MQ* (WMQ) representa a evolução do trabalho desenvolvido pela IBM desde 1992 na primeira versão do seu MQSeries[1]. Caracteriza-se pela segurança em robustez do tráfego de mensagens. O WMQ possui uma estrutura chamada *Queue Manager* onde as mensagens que não foram entregues são armazenada até que o destinatário a requisite novamente. A ordem das mensagens segue a ordenação de um FIFO (*First In First Out*) [28]. Este software também não possui integração com dispositivos móveis e LBS.

DDS - *Data Distribution Service*

O DDS surgiu do consórcio entre o *American Real-Time Innovations* e o *French Thales Group*. Ambas as instituições juntaram esforços no sentido de criar uma especificação única que, foi mais tarde aprovada pela OMG [52](*Object Management Group*), resultando em 2003 no aparecimento da primeira versão do DDS [28]. Focado em aplicações de tempo real não fornece suporte a dispositivos móveis, nem *Web Services*. Possui uma ótima qualidade de serviço pelo fato de trabalhar com sistemas de tempo real, como equipamentos de telecomunicação e simulação.

Os softwares descritos anteriormente, apesar de serem difundidos e robustos, possuem algumas deficiências ou rigidez quanto a customização. O Biztalk é uma ferramenta proprietária; O ActiveMQ é apenas um serviço mensagens para JMS; Tuxedo, WebSphere MQ e DDS não permitem integração de dispositivos móveis. Portanto, Orion será implementado para surprir as necessidades que os Softwares pub-sub anteriormente descritos não suprem.

Além dos softwares citados anteriormente, podem ser citados tambem OpenCom[18], que utiliza os principais recursos do Microsoft COM para realizar sua implementação. Logo, é ligada a uma plataforma proprietária, indo de encontro a um dos principais objetivos do projeto: utilizar software livre.

E por fim, o ReMMoC[32] desenvolvido pela Universidade de Lancaster em parceria com a Lucent Technologies. Trata-se de um middleware reflexivo configurável, que suporta o desenvolvimento de aplicações móveis e elimina as propriedades heterogêneas de um

[1]MQ é a sigla para *Message Queue*. *MQSeries* é uma família de softwares da IBM cujos componentes são utilizados para interligar aplicações para que possam trabalhar em conjunto. Este tipo de aplicação é chamada de Software de Integração de Negócio ou *Middleware*.

ambiente computacional através do uso de reflexão, da tecnologia de componentes, e da utilização de abstrações em *Web Services*. Apesar de ser proposto para dispositivos móveis, não possui ciência de contexto, ou seja, não fornece informações sobre o hardware do dispositivo do usuário ou geoposicionamento. Portanto, não supre as necessidades de Orion de obter informações de contexto do usuário.

2.4 Serviços Baseados em Localização

Os serviços providos por sistemas que consideram o posicionamento/localização do usuário para fornecê-los são denominados Serviços Baseados em localização, *Location-based Services* (LBS)[2, 36, 9]. Segundo um estudo da *The Insight Research Corporation* [57] intitulado "Mercado dos Serviços Baseados em Localização 2008-2013", os consumidores de serviços para dispositivos móveis estão adquirindo LBS juntamente com outros serviços como vídeo sob chamadas, mensagens multimídia e compartilhamento de arquivos. O valor gerado com LBS em 2008 foi estimado em mais de 1,6 bilhões de dólares.

Segundo a *Insight Research*, em 1990 os serviços de localização sem fio eram restrito às aplicações militares. Atualmente, é comum em softwares de navegação para automóveis, para os pais que utilizam a tecnologia para saber onde os filhos estão através da localização do celular, entre outras aplicações.

Os serviços oferecidos podem fornecer meios de localizar pessoas, máquinas, veículos, recursos e serviços sensíveis à localização, bem como usuários determinarem sua própria localização [2]. Basicamente, os sistemas LBS devem se concentrar em duas ações:

1. Obter a localização do usuário; e
2. Utilizar esta informação para prover um serviço.

Estas ações são necessárias para ser possível responder quatro questões básicas [2]:

1. Onde eu estou?
2. Onde está o mais próximo...?
3. Onde está meu...?
4. Como eu posso chegar lá?

Para que tudo funcione como o esperado, o software LBS deve determinar a localização do usuário. As formas mais conhecidas para determinar o posicionamento de usuários, bem como uma breve comparação entre elas, são apresentadas a seguir.

Baseado em Rede - UMTS/GSM

O sistemas de localização baseado em rede tem seu funcionamento baseado no *round-trip delay* (RTD)[2] e triangulação entre os dispositivos móveis e as estações base[3] [36, 54, 2]. Esta abordagem funciona muito bem em áreas urbanas ou internas, mas tem baixa performance em áreas rurais [54], devido ao fato da alta densidade de estações base na zona urbana.

GPS

O Sistema de Posicionamento Global (GPS) é baseado nos dados GPS recebidos via satélite. Possui uma baixa performance em áreas urbanas/internas, mas possui uma boa performance em áreas rurais ou suburbanas [54]. A disponibilidade deste serviço está ligada a três fatores: a precisão do horário do GPS, o tráfego de dados do satélite e na medição de distâncias. Caso ocorra a indisponibilidade de um desses três fatores, o serviço fica completamente paralisado.

A-GPS

Para melhorar o serviço GPS nas áreas urbanas, foi desenvolvido o A-GPS (*Assisted* GPS). Nada mais é do que a combinação do GPS com o sistema **Baseado em Rede**, propiciando uma maior rapidez e confiabilidade na localização de satélites. Na prática, esta união possibilita, por exemplo, o usuário utilizar o GPS enquanto dirige.

GNSS

O GNSS (Sistema Global de Navegação por Satélite) é um sistema de posicionamento baseado na triangulação de sinais de uma constelação de satélites. Dentre eles podemos destacar o US GPS e WAAS, o sistema russo GLONASS, o sistema japonês MSAS, o sistema chinês Beidou, e os sistemas europeus EGNOS e GALILEO [54].

A-GNSS

Semelhante ao A-GPS, mas agora aplicado ao sistema de posicionamento GNSS. O A-GNSS promete ser o mais robusto e eficaz sistema de localização do mundo, pois possui uma ótima performance tanto em área urbanas e rurais, quanto em áreas internas e externas. Peng [54] mostrou na Figura 2.5 como fica a qualidade do serviço em função da região em que se encontra o usuário.

[2]Em GSM, RTD é o tamanho do sinal que o celular capta da estação base. Em 3G, é o tempo de um pacote ir do cliente ao servidor e voltar ao cliente, também conhecido como *Round-Trip Time*(RTT)

[3]Consiste em uma torre que contém um equipamento de rádio.

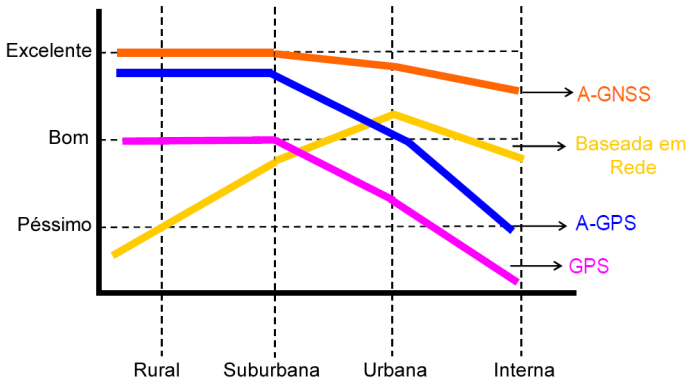


Figura 2.5: Comparativo entre os sistemas de localização [54].

Na Figura 2.5 fica evidente a grande vantagem do A-GNSS em relação aos outros sistemas de localização. Tal vantagem é devido a junção das melhores características dos sistemas já existentes com características próprias, como uma maior quantidade de bandas de frequência, três no total, contra apenas uma banda do GPS.

2.5 Considerações Finais

Os paradigmas *Publish-Subscribe* e LBS aliados aos avanços tecnológicos constituem um arcabouço tecnológico que provê um ambiente favorável para o desenvolvimento de ferramentas que façam parte do cotidiano das pessoas. Para tirar proveito deste ambiente, os softwares ubíquos devem levar em consideração variáveis importantes, como a mobilidade e a heterogeneidade. Neste projeto estas duas variáveis serão consideradas, aplicando suporte à adaptação de conteúdo sensível a contexto para dispositivos móveis e a adoção de mecanismos de localização baseados em Rede, GPS e A-GPS.

Capítulo 3

Modelos Baseados em Componentes de Software

3.1 Considerações Iniciais

A reutilização de software utilizando componentes surgiu no final dos anos 60 como uma alternativa para superar a crise do software, e tem como objetivo principal o desenvolvimento com qualidade e economicamente viável utilizando artefatos já especificados, implementados e testados [72]. A comunidade de desenvolvimento de software está gradativamente aderindo ao reuso, pois um novo software pode ser obtido praticamente a partir do código existente oriundo de outros softwares [31]. Em softwares embarcados, os componentes de software propiciam um menor custo no consumo de memória, fato importante em se tratando da escassez de recursos destes dispositivos. Neste capítulo serão apresentados os principais conceitos sobre componente de software e reusabilidade, além de uma breve descrição dos principais modelos de componentes de software encontrados na literatura.

3.2 Definições

Segundo Szyperski [73], um componente de software é uma unidade de composição com interfaces especificadas por meio de contratos e dependências de contexto explícitas. Um componente de software pode ser distribuído independentemente e está sujeito a composição com outras partes, residindo no nível de pacotes, componentes ou classes, e não no nível de objetos ou objetos distribuídos. A utilização de componentes de software no desenvolvimento de sistemas baseados em Internet vem crescendo, comparado a outros paradigmas, propiciando uma maior reusabilidade [65, 31, 67].

Reusabilidade é um atributo que se refere ao potencial de reutilização que se espera de um componente de software. A reutilização de software não só melhora a produtividade, mas também tem um impacto positivo sobre a qualidade e durabilidade dos produtos

gerados [31].

Além da melhor reutilização de código e implementação, há um ganho de tempo e diminuição do custo de manutenção; melhor encapsulamento; redução de riscos do processo; desenvolvimento mais acelerado decorrente do primeiro fato e pode auxiliar na estimativa de prazo e de custos.

Um componente pode ser considerado como uma parte substituível independente que fornece uma função clara e distinta, pode ser um pacote de software desenvolvido independentemente e entregue como uma unidade, e que oferece interfaces pelas quais é conectado, sem alterações, a outros componentes para compor um sistema maior.

A comunicação entre componentes é realizada por meio de estruturas denominadas de *interfaces*, definindo o conjunto de operações e os tipos de dados relacionados. Uma boa interface deve ser bem documentada, verificada e mantida. Mudanças internas dos componentes não devem resultar em impactos para os clientes que utilizam determinada interface.

As interfaces podem receber o nome de *receptáculos* quando estas são destinadas a receber um determinado serviço provido por outro componente, mas ambos são classificados como *pontos de interação*, pois é através destes pontos de interação que ocorre a comunicação entre componentes. Por fim, define-se *binding* como a associação entre uma interface e um receptáculo [74].

Os **Modelos de Componente de Software** definem o que são componentes, como podem ser construídos, representados, compostos ou montados, implantados e o porquê de todas as operações implementadas em sua interface. O clico de vida de um componente compreende 3 fases: *projeto, implantação e execução*. Na fase de *projeto* ocorre o processo de construção, logo após ele é catalogado e finalmente armazenado no repositório para que posteriormente possa ser recuperado. Na fase de *implantação* os componentes são recuperados do repositório e compilados ficando prontos para execução, e finalmente, na fase de *execução* ele é colocado em funcionamento.

Os modelos podem ser divididos em quatro categorias [38], este agrupamento é realizado tomando como base as ações e artefatos gerados nas fases de projeto, composição e implantação. Logo, os componentes da mesma categoria possuem o mesmo ciclo de vida:

- **Categoria 1**: Na fase de projeto, novos componentes podem ser depositados no repositório, mas não podem ser retirados dele. Não é possível fazer composição de componentes na fase de projeto. Já na fase de implantação é possível recuperar e realizar composição entre componentes. Ex: JavaBeans [71];
- **Categoria 2**: Na fase de projeto novos componentes podem ser adicionados ao repositório, não podem ser recuperados. É possível realizar composições de componentes na fase de projeto, no entanto eles não podem ser adicionados ao repositório. Na fase de implantação não há novas composições: serão as mesmas da fase de projeto. Ex: Enterprise JavaBeans [70, 44], CORBA Componente Model [49], COM [41], .NET [42] e *Web Services* [4];

- **Categoria 3**: Na fase de projeto novos componentes podem ser adicionados ao repositório de componentes e também podem ser recuperados. A composição é possível e pode ser adicionada no repositório. Na fase de implantação não é possível realizar uma nova composição, serão as mesmas da fase de projeto. Ex: Koala [75], KobrA [8] e SOFA [13];
- **Categoria 4**: Na fase de projeto não há nenhum repositório, logo todos os componentes são construídos a partir do zero, sendo possível realizar a composição nesta fase. Na fase de implantação nenhuma nova composição é possível, serão as mesmas da fase de projeto. Ex: Linguagens de descrição de arquitetura, UML 2.0 [50], PECOS [60] e Fractal [12].

3.3 Modelos de Componentes

A seguir são apresentados 12 modelos de componentes utilizados na literatura, estes foram selecionados por serem os mais citados na revisão bibliográfica realizada. Portanto, outros modelos de componente de softwares podem não constar nesta listagem. Após uma suscinta descrição das características de cada um, é apresentada ao final de cada seção o motivo da sua adoção ou não no desenvolvimento de Orion e Eros.

3.3.1 JavaBeans

O JavaBeans é um modelo elegante para construção de aplicações baseadas em componentes. Em geral, é utilizado para construção de aplicações locais ou centralizadas, ao contrário do EJB, descrito na Seção 3.3.2, que é destinado à aplicações distribuídas. A comunicação entre os componentes é baseada em eventos, com apenas alguns cliques de mouse é possível criar ricas interfaces [71]. Os JavaBeans (muitas vezes chamados simplesmente de beans) permitem que os desenvolvedores colham os benefícios do desenvolvimento rápido de aplicações Java montando componentes predefinidos de software para criar aplicativos e *applets* poderosos. Os "montadores"de componentes não precisam conhecer os detalhes de implementação de um componente. Em vez disso, o "montador"de componentes precisa conhecer os serviços fornecidos por um componente para poder fazê-lo interagir com outros [22].

A principal desvantagem decorrente do uso de JavaBeans é o grande número de arquivos, incluindo código fonte escrito em Java e descritores XML, que precisam ser mantidos para definir os beans de sessão e de entidade, os tipos de beans mais usados no desenvolvimento de aplicações.

3.3.2 EJB: Enterprise JavaBeans

Enterprise JavaBeans (EJB) é uma arquitetura de componentes destinada ao projeto e implantação de sistemas baseados em componentes. Aplicações desenvolvidas utilizando

EJB são escaláveis, transacionais e multiusuário e podem ser implantadas em qualquer servidor que suporta o padrão EJB [70]. Em outras palavras, EJB é um padrão de componente *server-side* para monitores de transação [44]. Os monitores de transação são responsáveis pela geração de transações para as interações do cliente com o componente, detecção de transações requisitadas e decisão de executar o método chamado dentro da transação do cliente, dentro de uma nova transação ou permitir que o componente gerencie os limites da transação.

Na Figura 3.1 o servidor EJB representa o ambiente compatível com a especificação da arquitetura, ele fornece um ou mais *containers* para os componentes implantados, além de ser responsável pelo gerenciamento de alocação de recursos.

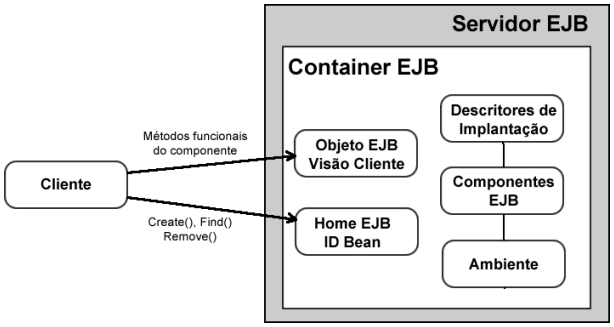


Figura 3.1: Arquitetura EJB [44].

O *Container* EJB fornece o contexto de execução e transacional aos componentes, também registra o componente no serviço de nomes, cria e destrói instâncias. Uma de suas principais funcionalidades é fornecer uma interface remota para o componente, além de gerenciar transações, estado e persistência.

Este modelo é recomendado para o desenvolvimento e implantação de servidores em ambientes distribuídos, atualmente estes servidores chamam-se *Servidores de Aplicação* [44]. De forma análoga ao JavaBeans, a quantidade de arquivos fontes é um problema que não pode ser contornado.

3.3.3 CCM: CORBA Component Model

CORBA (*Common Object Request Broker Architecture*) é um padrão desenvolvido para trabalhar como um *middleware*[1] na troca de dados entre sistemas distribuídos heterogêneos. Dessa forma, a localização de objetos não é restrita, bem como linguagem de programação, protocolos de comunicação ou hardware [49].

Uma característica importante da arquitetura CORBA é a utilização dos ORBs (*Objects Request Broker*), que trabalham como mediadores entre o cliente e os objetos dis-

[1]Define-se como middleware qualquer programa de computador que faz a mediação entre dois outros softwares. É utilizado para mover informações entre programas ocultando ao programador diferenças de protocolos de comunicação, plataformas e dependências do sistema operacional.

tribuídos. Os ORBs recebem requisições dos clientes e repassam para o componente em questão, e assim que recebe uma resposta a comunica ao cliente.

Na Figura 3.2 o conceito representado por *Objetos de Aplicação* são os componentes específicos que devem ser desenvolvidos para a aplicação. Os *Objetos de Serviços Comuns* são responsáveis por fornecer uma coleção de serviços para a aplicação (Interfaces e Objetos). Com as funções básicas utilizadas em qualquer aplicação distribuída, como serviços, notificações de eventos, controle de concorrência, etc. E por fim, as *facilidades comuns* são pacotes de serviços de objetos que facilitam o desenvolvimento de aplicações e que podem ser configurados para as necessidades de ambientes específicos.

A compreensão dos conceitos de ORB e dos serviços de objetos com sua posterior utilização bem sucedida constitui um processo demorado e custoso de aprendizado, normalmente baseado em sucessivas tentativas e erros. Portanto, CORBA possui uma curva de aprendizado muito alta e exige um elevado conhecimento de Programação Orientada a Objetos.

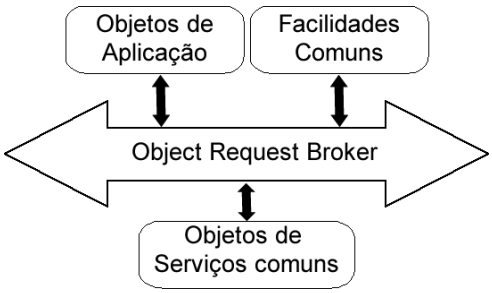


Figura 3.2: Arquitetura geral CORBA [49].

3.3.4 COM/COM+: Common Object Model

A tecnologia Microsoft COM (*Component Object Model*) está habilitada nos sistemas operacionais da família Windows. COM pode ser criado com uma variedade de linguagens de programação. A família de tecnologias COM incluem: COM+, *Distributed COM* (DCOM) e *ActiveX* [41].

Um exemplo de utilização é a funcionalidade presente no *Microsoft Word* conseguir fazer referências a dados presentes em planilhas no *Microsoft Excel*. Portanto, é possível criar aplicação que competem entre si em algumas tarefas ou controlar uma aplicação de outra.

COM+ é o nome dado ao COM baseado em serviços, pode ser considerado uma extensão do COM. Define-se o COM+ como uma infra-estrutura, portanto, não se constrói aplicações com COM+ e sim com o modelo COM. Por ser um tecnologia da Microsoft, possui a grande desvantagem de funcionar apenas na plataforma Windows.

3.3.5 .NET

O .NET é uma plataforma que tem como objetivo disponibilizar software como um serviço, especialmente na Internet, fornecer computação distribuída e componentização. A arquitetura do .NET é composta por 3(três) grandes componentes: *Common Language Runtime* (CLR) responsável pelo ambiente de execução e comum para todas as linguagens; *Class libraries* (.NET Framework) destinado basicamente à criação de aplicativos para Windows; e por fim, *ASP.NET* responsável pela criação de páginas interativas e *Web services* com SOAP[2] [42].

O .NET possui um muito menor de soluções públicas e gratuitas, mudanças de versão na plataforma tem causado dores de cabeça por falta de compatibilidade e possui pouco código aberto.

3.3.6 *Web Services*

Web Services são softwares identificados por uma URI[3], cujas interfaces e bindings são capazes de serem definidos, descritos e descobertos utilizando artefatos XML. Um Web Service suporta interações diretas com outros softwares utilizando trocas de mensagens baseadas em XML via protocolos de Internet [77, 4].

A gama de possibilidades que os *Web Services* proporcionam na integração de sistemas heterogêneos é grande, pois quaisquer sistemas, independente de ambiente de execução e linguagem de programação, podem trocar informações através de mensagens XML. Não há restrição quanto à forma que a comunicação é realizada entre os componentes do sistema distribuído que se deseja construir. Devido a facilidade de intercâmbio de dados entre softwares heterogêneos e recomendado para sistemas distribuídos, este será um dos modelos de componentes utilizados neste trabalho.

3.3.7 Koala

Koala (*C[K]omponent Organizer and Linking Assistent*) foi desenvolvido com objetivo de ser aplicado em sistemas embarcados, pois existe uma grande diversidade de softwares em dispositivos eletrônicos. A criação de um software embarcado às vezes pode atrasar o lançamento de um produto eletrônico [75]. Koala veio para sanar esse problema da indústria, pois componentes desenvolvidos em versões anteriores são reutilizados nas novas versões dos produtos embarcados, sendo necessário apenas implementar as novas funcionalidades. Este modelo possui os mesmos conceitos anteriormente vistos: *interfaces, receptáculos* e *bindings*.

Este modelo de componente por ser destinado a sistemas embarcados não se enquadra como um modelo flexível e customizável. Logo, também não é flexível o suficiente para a

[2]Segundo o W3C, SOAP é simples protocolo de troca de informações entre aplicações via XML.

[3]De acordo com a RFC 3986, URI (Uniform Resource Identifier) é um sequência compacta de caracteres para determinar ou identificar um recurso na Internet.

implementação de Orion ou Eros.

3.3.8 KobrA

KobrA (*Komponentenbasierte Anwendungsentwicklung*) é uma variação da *Unified Modeling Language* (UML). É um modelo dirigido à representação de componentes. Na especificação do componente deve ser descrito todas as propriedades que são visíveis para outros componentes. Este modelo também define que um componente pode se comportar como um sistema por si só, e que o desenvolvimento e montagem são a mesma coisa [8].

Não é adequado para este trabalho por possuir uma caracterísitca monolítica, ou seja, não é possível reutilizar partes de componente. Além disso, não cobre todo o ciclo de vida de desenvolvimento, possui diagramas até a fase implantação.

3.3.9 SOFA

O SOFA é um típico sistema baseado em componente acadêmico. Ele utiliza um modelo de componente hierárquico, onde cada componente pode ser do tipo primitivo ou composto. Um componente composto é constituído de outros, enquanto um primitivo não contém subcomponentes [13]. Em sua versão 2.0, não só herdou a característica de ferramenta de modelagem, mas também oferece uma estrutura completa de apoio a todas as fases do ciclo de vida de um componente. Está disponível para os sistemas operacionais Windows e Linux.

SOFA restringe seu uso somente na linguagem C++, o que vai contra os objetivos deste projeto que é de integrar softwares heterogêneos a um sistema de notificação encarregado de disseminar informação, independentemente da linguagem de programação.

3.3.10 UML 2.0

A UML (*Unified Management Language*) [51, 50] é uma linguagem gráfica para visualização, especificação, construção e documentação de artefatos de sistemas. Algumas características importantes devem ser ressaltadas:

- É possível modelar diversos tipos de sistema, não restringindo apenas aos baseado em software;
- Não tem relação alguma com qualquer linguagem de programação;
- É independente de ferramenta de modelagem, motivado pelo fato de ser independente de linguagem de programação;
- Ótima documentação; e
- Não possui uma estrutura rígida, logo pode ser adequada para cada situação.

Uma das características importantes adicionadas na versão 2.0 foi o surgimento dos modelos orientados à arquitetura (MDA). Com o MDA surge um novo futuro na criação de aplicações, onde será possível pegar um modelo UML e convertê-lo em aplicações funcionais, com pouco ou nenhum trabalho de programação.

A familiaridade com o modelo UML 2.0 e sua grande flexibilidade quando ao desenvolvimento em qualquer linguagem de programação, motiva sua utilização uma vez que pode ser utilizado em conjunto com o modelo baseado em *Web services*.

3.3.11 PECOS

O modelo de componentes PECOS foi originalmente desenvolvido para sistemas embarcados. Este modelo possui duas partes principais: a estrutura estática que descreve as entidades incluídas no modelo, suas características e propriedades; e o modelo de execução que define as características da execução do componente [60, 79].Os componentes no PECOS podem ser construídos hierarquicamente, assim como no modelo SOFA. Um componente que contém subcomponentes ou componentes filhos é chamado de componente composto ou componente pai. Estes subcomponentes não são visíveis fora do componente composto [60].

De maneira análoga ao Koala, PECOS é destinado a sistemas embarcados, o que não é o caso do problema abordado neste trabalho. Tal característica torna evidente a dificuldade de ambos os modelos de se adequarem a novos domínios e ambientes de execução diferentes.

3.3.12 FRACTAL

FRACTAL é um modelo de componente que tem como objetivo geral implementar, implantar e gerenciar complexos sistemas de software, incluindo sistemas operacionais e *middleware* [12]. Um componente FRACTAL é uma entidade de execução encapsulada, e que possui uma identidade distinta. Em baixo nível, cada componente é uma caixa preta. Assim como em outros modelos, existem dois tipos de interfaces: servidoras que aguardam pela invocação de operações e as interfaces clientes que requisitam operações. Este modelo é flexível o suficiente para que os programadores possam fazer adaptações que melhor se adéquem as suas necessidades.

Apesar de ser multilinguagem (JAVA, C, .NET, SmallTalk, Python) este modelo não é independente de linguagem como a UML e os *Web Services*. No entanto, possui uma característica interessante de ser reconfigurável em tempo de execução.

3.4 Considerações Finais

Neste capítulo foram apresentadas as principais características do desenvolvimento baseado em componentes enfatizando a reusabilidade de software como principal benefício.

Dentro os modelos de componentes de software estudados a UML 2.0 se mostrou mais flexível no processo de desenvolvimento, não impondo restrições e fornecendo subsídios para ser adaptado conforme as características do projeto. Desta forma, será o modelo utilizado neste projeto, juntamente com *Web Service* utilizado como mecanismo de interconexão entre sistemas heterogêneos.

Outra justificativa para a adoção da UML 2.0 é que ela também serve como meio de comunicação entre aqueles que serão os potenciais utilizadores do sistema, os que avaliam para conferir se o sistema supre as necessidades e os que desenvolverão as funcionalidades pretendidas.

Capítulo 4

Plataforma e-SAPI *bovis*

4.1 Considerações Iniciais

A crescente preocupação da sociedade sobre a origem e a qualidade sanitária da carne bovina e a pressão dos países importadores de carne, impulsionaram a necessidade de modelos e técnicas para melhoria da cadeia produtiva da carne bovina.

Neste capítulo é apresentada a plataforma Web e-SAPI *bovis* que permite a integração das informações de diferentes parceiros da cadeia de carne bovina (produtor, frigorífico, agência de vigilância animal, entre outros). O e-SAPI foi criado a partir das diretrizes do Sistema Agropecuário de Produção Integrada (SAPI) implantado pelo Ministério da Agricultura, Pecuária e Abastecimento (MAPA). O SAPI é caracterizado por uma exploração agropecuária sustentável, em conformidade com os protocolos formais de Boas Práticas Agropecuárias (BPA) [24], a fim de assegurar alta qualidade e inocuidade, tanto para os agroalimentos, quanto para os produtos, subprodutos e resíduos agroindustriais [45].

4.2 Rastreabilidade Bovina e Pecuária de Precisão Computacional

Rastreabilidade é um sistema de controle de animais que permite a identificação individual do animal ou por lote desde o nascimento até o abate, registrando todas as ocorrências relevantes ao longo de sua vida. Existe para garantir ao consumidor um produto seguro e saudável, por meio do controle de todas as fases de produção, industrialização, transporte, distribuição e comercialização, possibilitando uma perfeita correlação entre o produto final e a matéria-prima que lhe deu origem [61, 14, 68].

Em 2008 o Brasil liderou o ranking dos maiores exportadores de carne bovina no mundo, somando o volume de 2,2 milhões de toneladas equivalente carcaça e receita cambial de US$ 5,3 bilhões. Estes valores representaram uma participação de 28% do comércio

internacional, exportando para mais de 170 países, segundo a Associação Brasileira das Indústrias Exportadoras de Carnes (ABIEC) [1].

Segundo Edson et al.[16], Pecuária de Precisão (PP) pode ser definida como uma prática de manejo de rebanhos de bovinos na qual utiliza-se tecnologia de informação e comunicação para garantir a utilização das boas práticas de produção da carne. A partir de dados específicos do rebanho, de áreas de pastagens geo-referenciadas, de automação nas várias fases da industria da carne a pecuária de precisão tem por objetivo otimizar os custos de produção na obtenção de uma carne de qualidade. Além disso, a Pecuária de Precisão estuda os impactos ambientais e sociais na produção da carne e o aumento da produtividade.

A utilização da tecnologia da informação e comunicação se dá em vários níveis e áreas. O conjunto de e a complexidade dos problemas na pecuária de precisão apresentam novos desafios e oportunidades para a área de computação. Isso nos leva a introduzir a área de Pecuária de Precisão Computacional (Computação Aplicada a Pecuária de Precisão).

A Pecuária de Precisão Computacional é uma área multi-disciplinar que fundamenta-se em dispositivos de hardware e utilização de software como instrumentos fundamentais para o aumento da competitividade da industria da carne.

Neste contexto, o Brasil para continuar a ser o maior exportador de carne bovina do mundo, deve investir no desenvolvimento de soluções tecnológicas para garantir a qualidade da cadeia produtiva da carne.

4.3 Arquitetura do e-SAPI *bovis*

Visando atender às diretrizes do SAPI e implantar o programa na pecuária de corte no Estado de Mato Grosso do Sul, a Embrapa Gado de Corte firmou parceria com a Faculdade de Computação da Universidade Federal de Mato Grosso do Sul (FACOM) para o desenvolvimento de uma plataforma tecnológica para disponibilização de ferramentas Web para prover suporte à gestão da cadeia produtiva.

Em 2007 deu-se início ao desenvolvimento do e-SAPI *bovis*, um conjunto de ferramentas de alta usabilidade disponível na Web com o objetivo de auxiliar o monitoramento e o acompanhamento de informações referentes à produção bovina no Brasil. Dentre as ferramentas disponíveis nesta plataforma foi desenvolvido o módulo do e-GTA, (Guia Eletrônica de Trânsito Animal) que garante a rastreabilidade dos animais e o histórico completo do trânsito de cada indivíduo registrado no sistema por meio de coordenadas geográficas das propriedades pelas quais o animal passou [14]. Na Figura 4.1, é ilustrada uma das telas da plataforma Web onde é possível gerenciar os bovinos cadastrados nas propriedades dos pecuaristas.

Cada linha representa um animal da fazenda, onde são possíveis realizar ações como editar, visualizar, excluir, rastrear, vacinar e mostrar evolução de ganho de peso. De todas as ações mostradas na figura a mais importante é o rastreamento, onde é possível baixar um arquivo no formato KML(*Keyhole Markup Language*). Trata-se de um arquivo XML

com informações geográficas de onde o animal passou até o instante atual. Este arquivo pode ser lido por softwares como Google *Earth* e mostrar em um mapa os locais onde esteve.

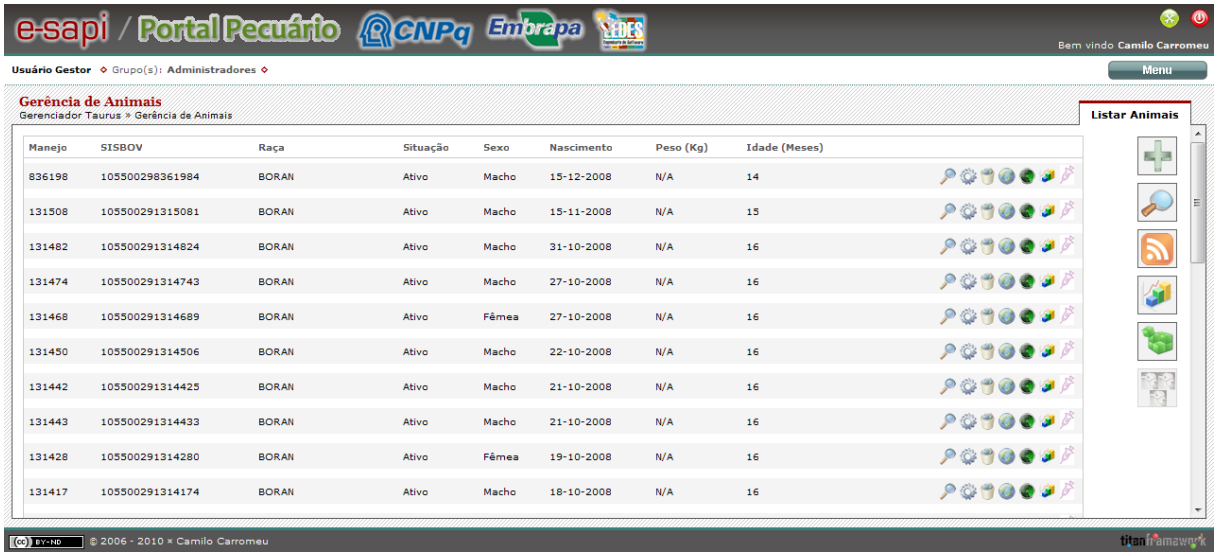


Figura 4.1: e-SAPI *bovis*, módulo de gerência de animais

O e-SAPI *bovis* é uma aplicação na Web voltada a oferecer suporte à implantação do SAPI na cadeia produtiva de carne bovina. Esta plataforma foi especificada e implementada tendo como foco principal o rastreamento da produção. Assim, o objetivo é que os produtores e proprietários rurais venham a aderir ao uso para alavancar sua produção fazendo uso de serviços até então inexistentes. Na Figura 4.2 são apresentados os dois portais informativos disponibilizados para os atores da cadeia produtiva da carne bovina[1].

Para utilizar o e-SAPI *bovis* o pecuarista deve se cadastrar na plataforma. O usuário fica vinculado às suas propriedades rurais anteriormente cadastradas. Cada uma tem, necessariamente, seu registro estadual ou CNPJ, que cumpre o papel de uma identificação única nacionalmente. Além disso, cada propriedade é cadastrada com suas coordenadas geográficas, dado importante para o geoprocessamento do trânsito de animais.

A plataforma está organizada em três grandes áreas. A primeira é o Portal e-SAPI *bovis* que contém diversas informações que auxiliam o produtor e o consumidor final; a segunda é o sistema de gestão do portal e-SAPI; e a terceira é a área de rastreabilidade que trabalha recolhendo informações de softwares de fazenda que saibam exportar seus dados via XML e alimentando a base de dados, a fim de centralizar todos os dados no portal facilitando a rastreabilidade da carne de bovinos e bubalinos. A Figura 4.3 ilustra a arquitetura do portal e-SAPI *bovis*.

Um item importante nesta arquitetura é o software cliente JAVA, chamado de **e-SAPI *client***, responsável por coletar dados armazenados em banco de dados externos ao e-SAPI e enviá-los à plataforma, por ter sido construído em JAVA é multiplataforma.

[1]Os principais atores da cadeia produtiva da carne são os pecuaristas, frigoríficos, defesa sanitária, governo e consumidores.

Cada pecuarista pode possuir seu próprio software de gerenciamento na fazenda, pode haver uma grande heterogeneidade entres estes bancos de dados e por consequência não poderiam ser migrados sem a intervenção deste software cliente. O papel desse software é padronizar os dados em um formato XML definido pela plataforma, possibilitando a migração dos dados de qualquer software de fazenda para o e-SAPI *bovis*.

Figura 4.2: Portal e-SAPI e Portal BPA.

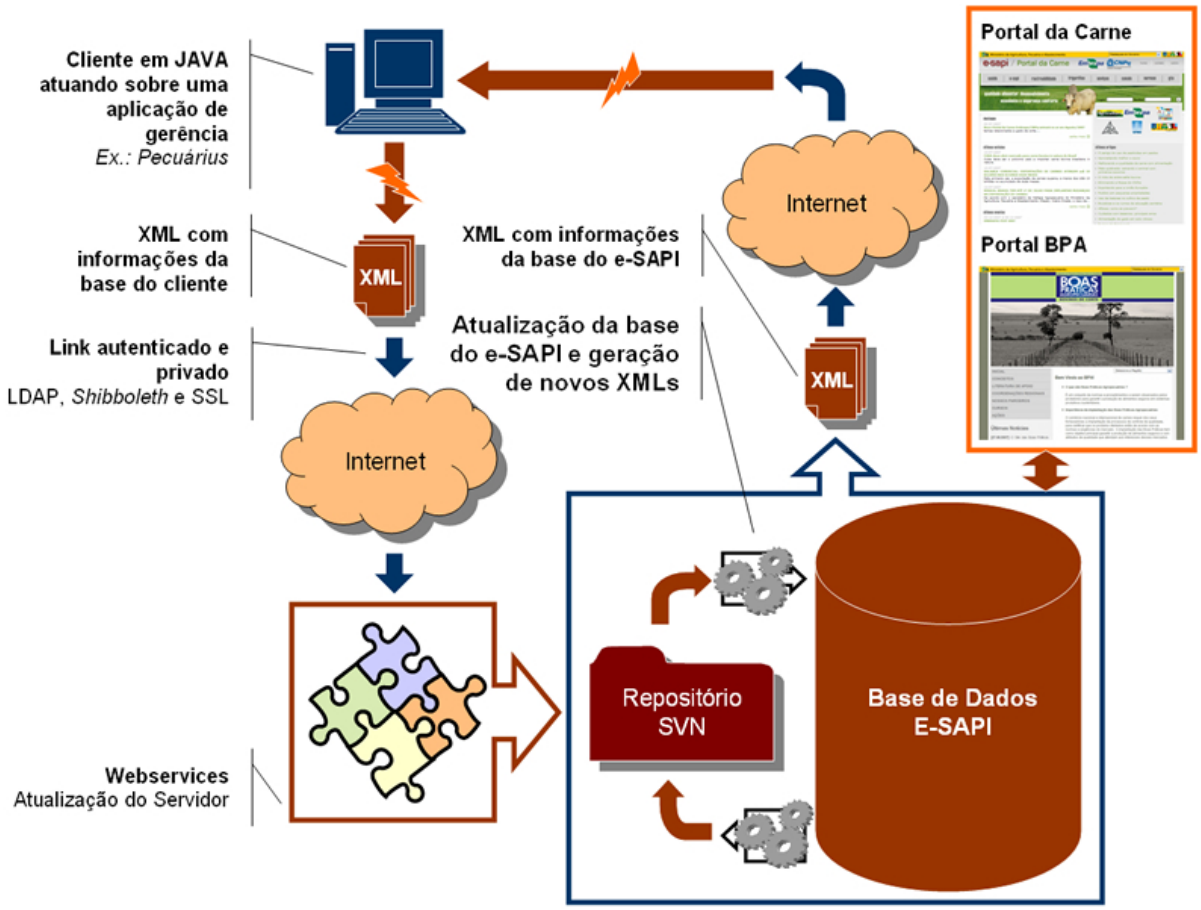


Figura 4.3: Arquitetura e-SAPI *bovis*.

Os dados enviados pelo e-SAPI *client* são capturados por um *Web Service* que faz o sincronismo entre os dados contidos na base centralizada da plataforma e os dados oriundos de softwares de fazenda. Para fazer o sincronismo dos dados é utilizado um **Repositório *Subversion* ou Repositório SVN**[2], sendo que somente as diferenças entre às bases de dados são aplicadas ao **Banco de Dados e-SAPI**. Tal banco de dados também fornece conteúdo para dois portais Web: Portal da Carne e Portal das Boas Práticas Agropecuárias (BPA). Estes dois portais são ferramentas ágeis e de fácil utilização, que viabilizam o controle pecuário, emprego das BPAs e constituem ferramentas auxiliares para a manutenção, o controle e o monitoramento das informações de um programa de rastreamento e certificação de carne.

4.4 Considerações Finais

O e-SAPI *bovis* é uma solução tecnológica abrangente para operacionalização do Sistema de Acompanhamento da Produção Integrada para a cadeia produtiva de carne bovina. Com o módulo e-GTA esta solução é mais completa permitindo exercer de forma plena a

[2]Sistema de controle de versões, foi desenvolvido para ser o substituto do antigo CVS

rastreabilidade individual ou em lotes de animais. A junção entre LBS, sistema pub-sub e a plataforma e-SAPI *bovis* trará grandes benefícios para os diferentes atores da cadeia produtiva, propiciando distribuir e divulgar informações utilizando todo o potencial das redes de comunicação e sistemas móveis.

Capítulo 5

Orion - Proposta de Componente Publish-Subscribe e LBS

5.1 Considerações Iniciais

Neste capítulo será apresentada a arquitetura e o modelo para o desenvolvimento do componente *Publish-Subscribe* integrado com LBS denominado **Orion**, utilizando uma abordagem baseada em componentes. Além disso, Orion deverá ser capaz de interagir com qualquer software de um provedor de serviço, adotando o padrão XML como forma de troca de dados entre as aplicações externas, e também deverá ser de fácil extensão e adaptação para novos domínios.

Orion terá o auxílio do software cliente chamado *mobile* Eros destinado para execução em dispositivos móveis. Terá como principais funcionalidades determinar a localização de usuários por GPS e/ou por triangulação de antenas de telefonia móvel, além da possibilidade de interagir com um subconjunto de funcionalidades de Orion capturando informações de contexto do dispositivo. O conteúdo enviado para os dispositivos móveis sofrerá customização baseado em informações de contexto enviadas por Eros.

5.2 Arquitetura e Modelo Conceitual

A Figura 5.1 ilustra a infraestrutura do componente Orion aplicado ao e-SAPI *bovis*, o porção que está circundado pela pontilhada delimita a real contribuição deste trabalho para o e-SAPI *bovis*.A arquitetura consiste de poucos elementos, como as estações base de telefonia móvel que já estarão presentes por pertencer à infraestrutura de telefonia móvel. Para determinar a localização dos usuários será levado em consideração que alguns dispositivos móveis não possuem GPS, portanto, sua localização será definida com a triangulação de antenas de telefonia. Porém, a prioridade é para os métodos de localização mais precisos, logo o aplicativo Eros tentará utilizar o GPS de imediato, caso este exista.

O servidor LBS terá a responsabilidade de gerenciar a posição geográfica corrente

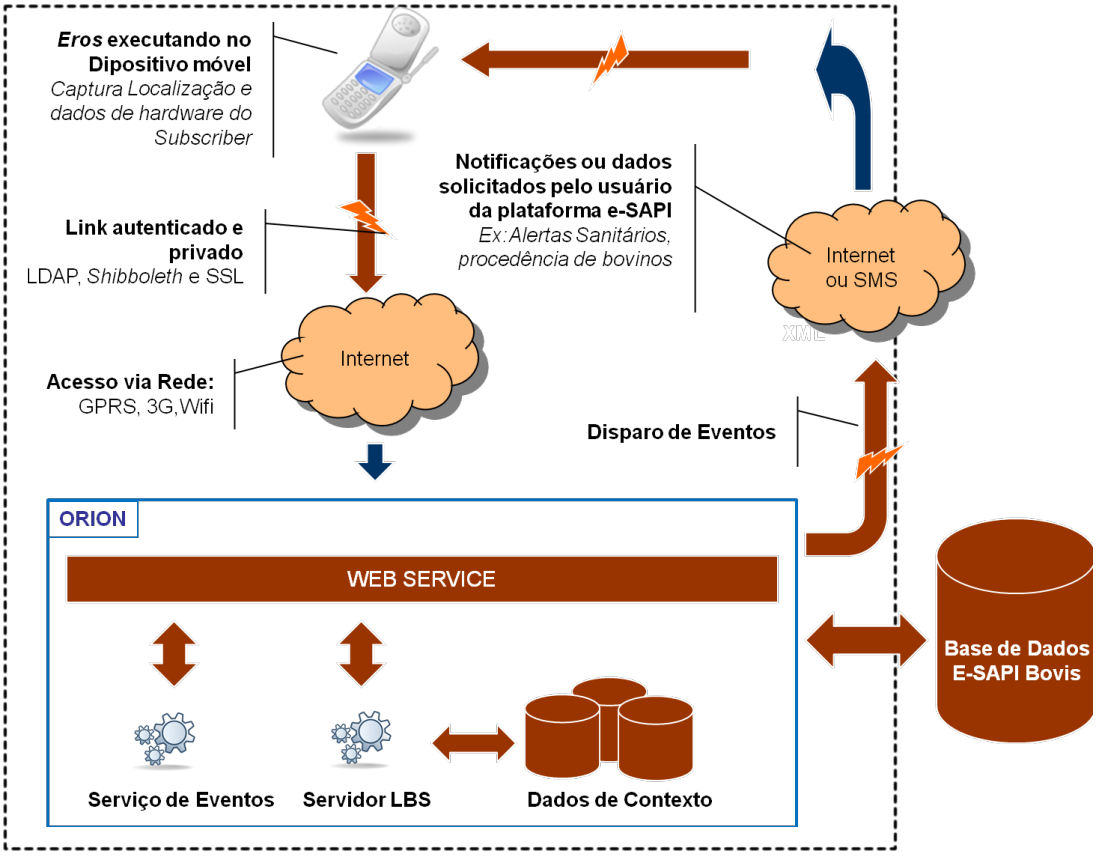


Figura 5.1: Arquitetura do Sistema Orion.

dos usuários móveis. A posição destes usuários é informada pelo software Eros, que em sua primeira versão será apenas para celulares e *smartphones* que possuem o sistema operacional Android. Os dados geográficos coletados são enviados para o servidor LBS e armazenado no banco de dados de **Dados de Contexto**.

O serviço de eventos é responsável por notificar os usuários quando um novo evento ocorrer, ou seja, quando um *publisher* publicar algo, o servidor de eventos é responsável por determinar que *subscribers* devem receber a notificação. Note que um *subscriber* pode receber uma notificação por dois motivos:

- Um *publisher* publicou algo em algum tópico de interesse, ou;
- Sua posição provoca o disparo de um evento, pois é de seu interesse receber notificações quando determinada posição geográfica for alcançada.

Os avisos poderão ser entregues via aplicativo Eros[1], caso o usuário esteja conectado ao serviço de notificação. As notificações via software Eros serão mais ricas, sendo possível a adoção de recursos multimídia na mensagem entregue ao usuário, sobretudo imagens. No entanto, o componente Orion poderá ser estendido e novos protocolos de entrega de mensagens poderão ser utilizados.

[1]Nesta versão protótipo não foi implementado o envio de SMS pelo servidor Orion

Na Figura 5.2 são representados os principais conceitos envolvidos no funcionamento do componente Orion. Apesar de apresentar o conceito de **Tópico**, Orion não pode ser considerado como *Topic-based*. Segundo Carzaniga [15], a característica de o tópico possuir variáveis vinculadas tornam o sistema do tipo *Content-based*. Por exemplo, no caso de avisos sanitários da ocorrência de Febre Aftosa em uma data região, a variável relacionada ao tópico é a coordenada geográfica de onde ocorreu o foco, ou apenas em qual Estado brasileiro ocorreu o problema. Logo, os *subscribers* podem configurar que gostariam receber avisos de foco de Febre Aftosa quando este ocorrer em estados brasileiros de seu interesse. Estes *subscribers* podem ser tanto pecuaristas, frigoríficos ou consumidores.

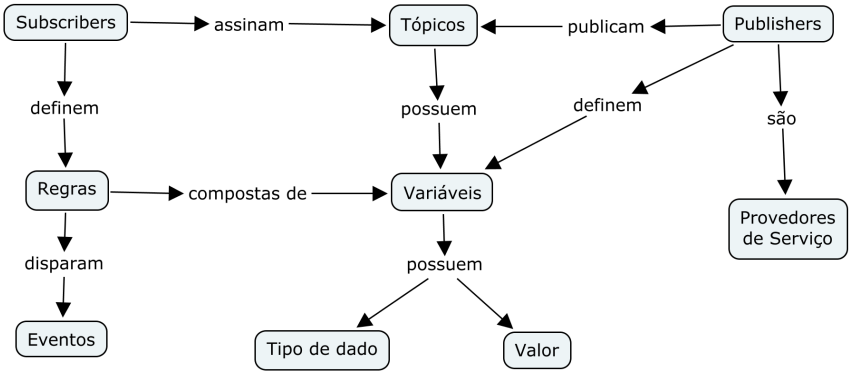


Figura 5.2: Mapa Conceitual do Sistema Orion.

Em resumo, quem define quais variáveis um tópico possuirá e quais são os possíveis valores para cada uma delas será o *publisher* e cabe ao *subscriber* definir suas próprias regras de recebimento, caso nenhuma regra seja especificada ele receberá todas as mensagens do tópico em que se inscreveu.

5.3 Modelo de Referência

Para ser possível especificar e implementar o componente Orion com qualidade e de fácil extensão, adotou-se o processo de desenvolvimento RUP (*Rational Unified Process*)[64]. Tal escolha não foi Ad hoc, pois o RUP é centrado na arquitetura baseada em componentes, além de ser projetado e documentado utilizando a notação UML [64]. Por consequência, o modelo de componentes a ser utilizado no projeto será a UML 2.0 [50]. Após estas duas escolhas foi necessário definir o modelo de referência[2] da arquitetura. Seguindo a mesma abordagem das grandes ferramentas, o BizTalk por exemplo, adotou SOA [39] como modelo de referência, vislumbrando a adoção de *Web Services* como forma de troca de dados entre as aplicações.

Na Figura 5.3 é ilustrado o modelo de referência SOA, no qual este projeto será estruturado. O objetivo é fazer com que os conceitos apresentados sejam respeitados e moldados para a realidade do projeto do componente Orion.

[2]Um modelo de referência é framework abstrato para entender os relacionamentos entre as entidades e algum ambiente. Ele permite o desenvolvimento de uma referência específica ou arquitetura concreta utilizando padrões consistentes ou especificações de apoio neste ambiente.

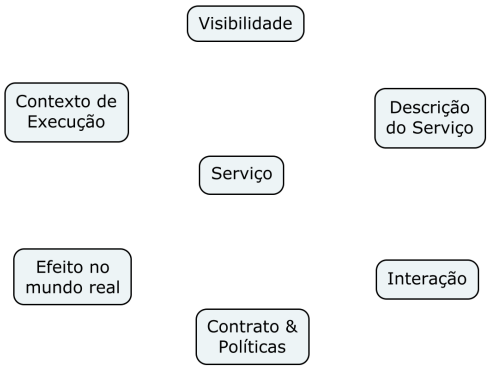


Figura 5.3: Modelo de Referência SOA [39].

O objetivo do componente Orion é prover um **Serviço** em que os (*publishers*) possam construir tópicos, anexando a estes tópicos variáveis que ajudem os *subscribers* a criar regras de recebimento de conteúdo. Por exemplo, um agricultor precisa ser avisado quando a previsão de temperatura na região dele será abaixo dos 10℃.

Quanto à **Visibilidade** do componente, ela será acessível de qualquer lugar no mundo, bastando apenas que os *publishers* estejam conectados à Internet. A **Interação** entre Orion e os provedores de serviço ocorrerá através de *Web Services* no padrão *Representational State Transfer* (REST) [29]. De maneira resumida, REST representa um modelo de como a Web deve funcionar de maneira a garantir os requisitos inicialmente estabelecidos, em particular as propriedades de escalabilidade e heterogeneidade.

No modelo REST, a camada de interação é padronizada. Assim, apenas a representação dos dados é especifica em cada implementação. Segundo Nunes e David [47], este modelo implica a utilização de um protocolo comum entre todos os intervenientes, incluindo os agentes intermediários (*caches, proxies, firewalls*). Com esta abordagem, é possível eliminar uma camada de complexidade extra no desenvolvimento das soluções. As **Interações** ocorrem na interface disponível, neste caso o *Hypertext Transfer Protocol* (HTTP), evitando a especificação de soluções próprias. Desta forma, reduzem-se as barreiras entre serviços diferentes aumentando o grau de interoperabilidade do componente Orion.

As **Descrições dos Serviços** podem ser obtidas através de uma *Web Application Description Language* (WADL) [33]. Em poucas palavras, trata-se de um padrão para descrever serviços providos por aplicativos baseados em HTTP, neste caso softwares REST. O WADL é mais simples que o *Web Service Description Language* (WSDL), por possuir um nível maior de abstração com relação a descrição dos serviços.

Diversas **Políticas** serão definidas por Orion no que se refere, por exemplo, ao tempo de permanência de uma mensagem na fila de mensagens. Toda **Interação** entre clientes e o servidor pub-sub onde haverá: inserção, atualização ou remoção, deverá ser autenticada. Simples buscas por tópicos não necessitarão ser autenticadas. Em resumo, as **Políticas e Contratos** de como os aplicativos devem interagir estão descritos no WADL, onde tanto aplicações podem ser criadas para composição automática de serviços, quanto devem ser facilmente compreensíveis por pessoas.

A correta interação entre as partes (*publishers*, *subscribers* e LBS) é denominada **Contexto de Execução**. Este contexto não é limitado a um lado da interação, ao invés disso refere-se à totalidade de interações (incluindo o provedor de serviço, *subscriber* e a infraestrutura comum necessária para mediar a interação). Este fato aumenta as condições e restrições necessárias para serem coordenadas e podem requerer uma troca de adicional de informação para completar o contexto de execução [39].

Os **Efeitos no Mundo Real** dos serviços que podem ser providos utilizando o componente Orion na rastreabilidade bovina são inúmeros, vão desde lembretes importantes de vacinação à alertas da Defesa Sanitária para zonas com epidemia de Febre Aftosa.

5.4 Mecanismo de Geoposicionamento

Os aplicativos baseados em localização estão se tornando comuns, mas devido a falta de precisão, o movimento do usuário, a variedade de métodos para obter a localização, e o desejo de conservar a bateria, obter a localização do usuário é um desafio. Para superar os obstáculos de obtenção de uma boa localização do usuário, preservando a energia da bateria, é necessário definir um modelo consistente que especifica como o aplicativo obtém a localização. Este modelo inclui quando iniciar e quando parar de ouvir as atualizações do posicionamento e quando usar os dados de localização em cache.

Eros utiliza uma sequência de passos pré-determinado para obter a melhor localização ao menor custo possível dos recursos do dispositivo. A Figura 5.4 ilustra como é realizado o processo de obtenção do posicionamento do usuário. Com esta abordagem o software *mobile* Eros utiliza o melhor mecanismo de posicionamento, no caso GPS, para determinar a posição do usuário. Se o GPS não estiver disponível, ele procura determinar a posicionamento através de *Wi-fi*; e por fim, tenta localizar o usuário por triangulação de antenas.

No entanto, ele não usa o GPS a todo momento, pois isso pode onerar a bateria, fazendo com que ela seja consumida rapidamente. Este problema foi solucionado realizando uma cache dos valores de posicionamento obtidos anteriormente. Assim, dado um intervalo de tempo delta, o software não utiliza o GPS durante esse período de tempo.

Como Eros consegue captar informações de contexto do usuário relacionado ao seu dispositivo, é possível desabilitar o a utilização o GPS em tempo de execução quando a carga da bateria chegar a 15%.

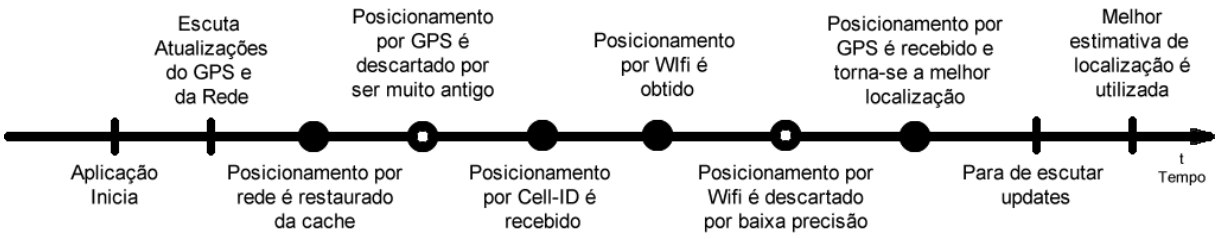


Figura 5.4: Processo de Obtenção da Localização.

A utilização de caches reduz a necessidade de sempre determinar a posição do usuário, levando em conta os mecanismos de localização por rede, Cell-ID ou GPS, se disponível.

5.5 Tecnologias Utilizadas

Após uma pesquisa para determinar quais seriam as melhores tecnologias e ferramentas de software livre necessárias para o desenvolvimento do componente Orion, as seguintes tecnologias foram elencadas:

Java

Java é uma linguagem de programação que teve seu primeiro release em maio de 1995, possui como principal característica ser multiplataforma e orientada a Objeto. A Plataforma de desenvolvimento, que possui o mesmo nome da linguagem, e a linguagem de programação foram desenvolvidos pela *Sun Microsystems* que recentemente se uniu a *Oracle Corporation*.

A característica multiplataforma foi adquirida pelo fato do código compilado pelo compilador Java gerar uma forma de código intermediário chamado de *bytecode*, que não é código nativo ao contrário do que é gerado por outras linguagens. Tal *bytecode* é interpretado pela Máquina Virtual Java (JVM) que roda sobre o sistema operacional. Possui outras características importantes como coletor de lixo, um vasto conjunto de bibliotecas, simplicidade na especificação e facilidade na criação de programas distribuídos.

Android

O Android é uma plataforma de software e sistema operacional para dispositivos móveis baseado no linux, desenvolvido pela Google e mais tarde pela *Open Handset Aliance*. Ela permite que os desenvolvedores escrevam códigos baseados na linguagem Java que utilizam as bibliotecas Java do Google, mas não suporta programas escritos em código nativo. A aliança já possui grandes nomes como LG, HTC, Motorola e Samsung, que já adotaram o Android como sistema operacional e o mercado consumidor tem crescido a cada dia com a venda de *smartphones* com este sistema operacional [3].

O android foi escolhido como tecnologia para desenvolver o aplicativo Eros devido sua grande facilidade em se trabalhar com mapas e a grande simplicidade para se obter dados de hardware do dispositivo móvel. Além disso, possui uma API simples e eficaz para obtenção da geolocalização do usuário.

PostgreSQL

O PostgreSQL é um poderoso sistema gerenciador de banco de dados objeto-relacional de código aberto. Tem mais de 15 anos de desenvolvimento ativo e uma arquitetura que comprovadamente ganhou forte reputação de confiabilidade, integridade de dados e conformidade a padrões. Roda em todos os grandes sistemas operacionais, incluindo GNU/Linux, Unix, e MS Windows. Tem suporte completo a chaves estrangeiras, junções (JOINs), visões, gatilhos e procedimentos armazenados (em múltiplas linguagens). Inclui a maior parte dos tipos de dados do ISO SQL:1999, incluindo INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, e TIMESTAMP. Suporta também o armazenamento de objetos binários, incluindo figuras, sons ou vídeos. Possui interfaces nativas de programação para C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, entre outros, e uma excepcional documentação [56].

XML

O XML (*Extensible Markup Language*) surgiu da necessidade de uma linguagem padronizada e fácil de estender. Derivada da linguagem SGML (*Standard Generalized Markup Language*) apresenta características como a independência dos dados, a separação do conteúdo e sua apresentação. Uma vez que o XML descreve dados ele pode ser processado por um aplicativo. Segundo Deitel [21], isso torna o XML uma estrutura de referência que pode ser utilizada para o intercâmbio de dados. Sua flexibilidade e seu poder o tornam perfeito para *middleware*, que deve ser interoperável ao máximo para ser eficaz.

De acordo com Deitel [21], no futuro, à medida que a Web continua a se expandir, parece provável que o XML se torne a linguagem universal para representação de dados. Dessa forma, todos os aplicativos seriam capazes de se comunicar desde que fossem capazes de entender a marcação ou vocabulário XML.

5.6 Trabalhos Relacionados

Em 2000, Carzaniga [15] elaborou um estudo focado no roteamento de eventos e filtros aplicados em arquivos XML fazendo busca utilizando XPath[3]. Logo, a busca era feita dentro de um arquivo, sem banco de dados, sem fila de entrega, via socket JAVA e os *publishers* precisavam saber dos endereços IP dos servidores que desejam publicar. O projeto foi finalizado em 2000 não estando mais em desenvolvimento.

3(três) anos depois, Paul Grace [32] desenvolveu o ReMMoC, framework que tinha como objetivo principal solucionar os problema da heterogenidade entre os dispositivos móveis. ReMMoC é um middleware baseado em Web-Services reflexiva que permite que clientes móveis a serem desenvolvidos de forma independente de mecanismos de interação.

[3]XPath é uma linguagem que descreve uma forma de localizar e processar itens em um documento XML usando uma sintaxe de endereçamento baseado no caminho através de uma estrutura lógica do documento ou hierarquia [76].

Este framework reconfigura-se dinamicamente para se adaptar ao ambiente de serviço atual.

Em 2002, Pietzuch e Bacon [55] construíram um sistema pub-sub baseado em um middleware formado por uma rede de *overlay* chamado Hermes, que interliga os nós responsáveis por executar os mecanismos para a comunicação entre os clientes. A rede de *overlay* permite que o Hermes se adapte a mudanças na topologia do middleware, além de tornar eficiente o roteamento de notificações. No entanto, esse roteamento era feito pelo tipo de dado do evento, que era provido por uma biblioteca de tipos que restringia sua heterogeneidade.

Araújo [7] em 2009, apresentou o Software Continuum que utilizava um método de filtragem de notificações semelhante ao de Carzaniga [15] utilizando XPath. Apesar de ser focado na *Computação Ubíqua*, não faz referência à utilização de LBS e nem ao suporte à adaptação de conteúdo sensível a contexto para dispositivos móveis [59];

Desta forma, existem poucos trabalhos científicos na literatura nacional nesta temática, o que mostra a relevância de estudo com o enfoque no Agronegócio, em especial Pecuária de Precisão Computacional. Após o estudo dos trabalhos relacionados, a característica de filtragem de tópicos demonstrada por Carzaniga e Araújo foi adotada. A idéia da utilização de Web Services de Grace também foi importante para propiciar uma melhor interoperabilidade entre os dispositivos.

5.7 Considerações Finais

Neste capitulo foi possível observar como foi projetado o funcionamento tanto de Eros quanto de Orion. Eros, procura não onerar o dispositivo móvel com processamento desnecessário para não ocasionar o esgotamento de recursos, como por exemplo o descarregamento da bateria. A capacidade de obter dados sobre os recursos do dispositivo móvel propicia um mecanismo eficiente de reconfiguração em tempo de execução, para não agravar a escassez de recursos.

Para validar a arquitetura proposta, protótipos do software *mobile* Eros e Web Orion foram desenvolvidos aplicados à Plataforma e-SAPI *bovis* conforme discutido neste capítulo. Os resultados obtidos com esta experiência serão descritos no capítulo seguinte.

Capítulo 6

Estudo de Caso: e-SAPI Bovis

6.1 Considerações Iniciais

Neste Capítulo é apresentado um estudo de caso no domínio da Pecuária de Precisão para a aplicação e validação do processo e as ferramentas implementadas. Optou-se por utilizar a Plataforma e-SAPI *bovis* a fim destacar as dificuldades e as principais necessidades no domínio. Assim, são apresentadas a descrição do problema da difusão de notificações entre os atores da cadeia produtiva da carne para rápida tomada de decisão.

6.2 Descrição do Problema

A difusão de informação dentro da cadeia produtiva da carne é ferramental necessário para a rápida tomada de decisão e visa minimizar os prejuízos causado por epidemias que prejudicam fincanceiramente Estados inteiros, ou até mesmo o Brasil sofrer sanções internacionais.

Em 2005, produtores de melancia, melão e abóbora do Mato Grosso do Sul forma afetados pelos efeitos das restrições no mercado interno por causa da febre aftosa no Estado. Por motivos que as prefeituras de Japorã, Mundo Novo e Eldorado, além de 52 produtores rurais não conseguiram apurar, o Paraná vetou a entrada desses produtos no território paranaense, causando prejuízos.

Diante deste constantes embargos à carne brasileira, é relevante que o e-SAPI seja capaz de que, ao descobrir um foco de doença, propague este como um alerta para as autoridades responsáveis, como as agências reguladoras, institutos de defesa e o MAPA (Ministério da Agricultura, Pecuária e Abastecimento).

Ainda no domínio da pecuária de precisão, um veterinário pode ser avisado quando um lote de animais não foi vacinado na data correta. Informar o consumidor que ele está comprando carne oriundas de áreas com suspeita de febre aftosa na tela do seu celular. Ou até mesmo informar o proprietário da fazenda sobre uma queda incomum de peso de

seus animais.

Com estas informações sendo informadas em tempo real para o pecuarista ou interessados dentro da cadeia da carne, uma grande salto no mecanismo de manejo do gado será dado no Brasil. As autoridades terão mais tempo hábil para tomarem medidas necessárias e evitar epidemias incontroláveis, acarretando o embargo da carne brasileira.

Outro cenário de aplicação real de Orion e Eros integrados com a plataforma e-SAPI *bovis*, é de se tornarem um mecanismo onipresente de motonitoramento bovino. Softwares de manejo podem enviar dados ao Orion sobre alguma anomalia ocorrida durante o manejo diário. Por exemplo, animais que estão com vacinas com vencimento próximo; utilizando chips de temperatura implantados nos animais alertar que existem bovinos febris no rebanho; alertar o pecuarista quando ocorrer alguma alteração no ganho de peso dos animais entre outros serviços.

Portanto, Orion e Eros, constituem um ferramental importante para vigilância sanitária e para o agronegócio, maximizando a obtenção de informações a acerca do rebanho e na mesma proporção aumentando a produtividade.

6.3 Software *Mobile* - Eros

Eros é um software *mobile* desenvolvido para captar os notificações geradas por Orion, conforme ilustrado anteriormente na Figura 5.1, além de prover funcionalidades adicionais para o domínio da Pecuária de Precisão. Em seu primeiro protótipo, apresenta funcionalidades básicas, porém, importantes para a cadeia produtiva da carne. Na Figura 6.1 é mostrado a tela inicial do software Eros com suas principais funcionalidades. Nesta versão protótipo, este software será para *smartphones*, sendo a implementação para dispositivos com pouca memória um trabalho futuro.

Figura 6.1: Tela Inicial Eros.

No menu notificações é possível ler os eventos gerados por Orion. Os itens e-GTAs e rastreabilidade propiciam uma ferramenta de rastreio de animais na tela do celular, com scanner de código QR Code[1] e exibição de mapas que serão discutidos nas seções subsequentes.

Para acessar aos dados do e-SAPI *bovis* via Eros, é necessário realizar um login para que seja possível autenticar o usuário e saber quais são suas propriedades e animais. Este formulário pode ser visto na Figura 6.2.

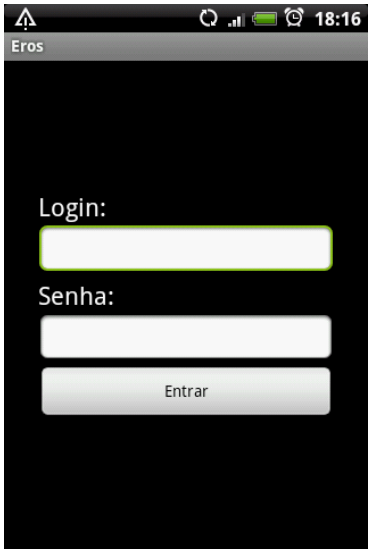


Figura 6.2: Tela de login e-SAPI.

Após autenticado, é possível saber o que o usuário pode ou não acessar dentro do software. Todo esse controle de permissões está implementado na plataforma e-SAPI. Na prática, a URL de login para o e-SAPI *bovis* deverá ser criptografada. No entanto, a senha do usuário já sai criptografada em SHA1[2] do software Eros, reduzindo assim o perigo de ser capturada no tráfego da rede ainda sem criptografia.

A próxima tela é ilustrado como o usuário consegue ver as notficações geradas por Orion. Note que na 6.3 existe uma marcação chamada **provedor** em cada notificação recebida. Esta marcação indica qual provedor de serviço publicou o evento. lembre-se que Orion é de propósito geral e pode publicar eventos não somente do e-SAPI *bovis*, mas também de outros provedores de serviço. Um mesmo usuário pode estar interessado em notificações de Febre Aftosa e eventos climáticos, este último não provido pelo e-SAPI.

Eros necessita de Internet para que possa baixar todas as notificações geradas para o usuário em questão. Para facilitar a detecção de conexão, um módulo do software mobile é responsável por gerenciar em que tipo de rede o dispositivo está conectado. Uma vez que o usuário esteja conectado em uma rede *Wifi* ou 3G, é possível baixar muito mais notificações de uma única vez do que se ele estivesse conectado em uma rede de baixa velocidade.

[1]QR Code é um código de barras em 2D que pode ser facilmente escaneado usando qualquer *smartphone*, capaz de armazenar 4296 caracteres alfanuméricos

[2]SHA1 é um função de Hash com criptografia de 160 bits.

Figura 6.3: Tela de notificações.

Anteriormente, foi citado que a plataforma e-SAPI é capaz de gerar eletronicamente a Guia de Trânsito Animal (GTA). Portanto, nada mais cômodo do que existir a possibilidade de consultar, via celular, e-GTAs mostrando em um mapa o local onde um animal específico passou. A Figura 6.4 mostra com uma linha com os locais onde um animal esteve, cada extremo de segmento de reta corresponde a um local diferente.

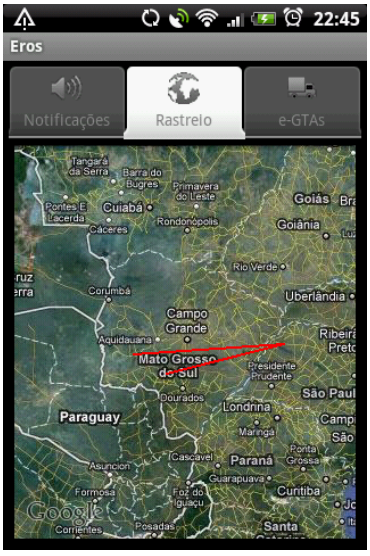


Figura 6.4: Mapa com rastreamento.

Os locais onde o animal esteve não são necessariamente propriedades rurais, pois podem ser locais de eventos, feiras, frigoríficos, entre outros. Ou seja, qualquer lugar onde o animal teve que realizar algum deslocamento. É possível buscar e-GTAs emitidas através do software Eros, conforme indicado na Figura 6.5.

A e-GTA, conforme pode ser observado no Apêndice I, possui um QR Code no canto

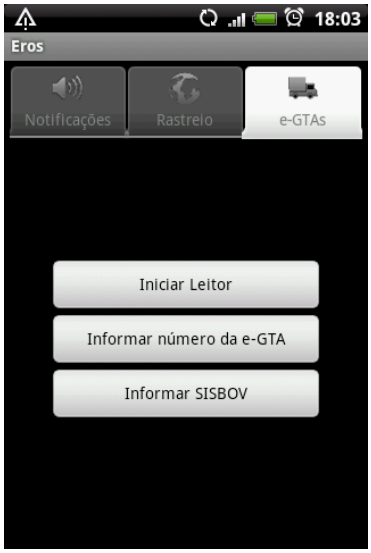


Figura 6.5: Tela de busca de e-GTA.

superior direito que informa o número da e-GTA. Eros é dotado de um leitor de QR Code conforme demonstrado pela Figura 6.6.

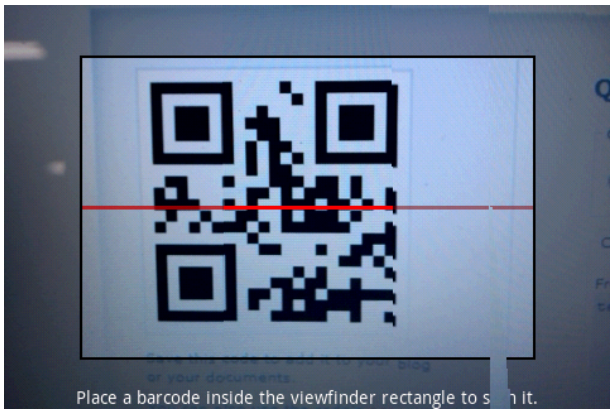


Figura 6.6: Leitor QR Code.

A Utilização do QR Code não é obrigatório, uma vez que existe uma opção de inserção manual tanto do SISBOV para identificação individual, quanto para a e-GTA para a identificação em lote.

Além das funcionalidades visíveis ao usuário, Eros consegue fazer um *dump* do estado atual da alocação dos recursos do dispositivo, fornecendo uma ferramental muito importante na obtenção de dados de contexto.

6.4 Software Web - Orion

Orion foi desenvolvido em JAVA e fornece uma interface para cadastro de tópicos, variáveis, regras e é onde está localizado o *Web Service* que fornece dados ao Eros ou onde ele recebe

dados externos de outras aplicações que desejam criar notificações.

Para o sistema de notificações utilizando o e-SAPI funcione, é necessário a criação de tópicos que mais tarde serão assinados pelos *subscribers*. Um tópico básico que o e-SAPI pode publicar é a ocorrência de focos de Febre Aftosa no território nacional. Este tópico tem como variável o Estado Brasileiro que ocorreu o foco da doença, mas nada impede que esta granularidade seja a nível de cidade. Na Figura 6.7 é possível visualizar a tela de gerência de tópicos.

Além das ferramentas básicas de inserção, remoção e edição de tópico com o título, um *payload* ou informação adicional. É possível vincular varíaveis ao probema, neste caso, o estado brasilerio que está com o foco de Febre Afotosa. A ações de criação de tópico só são visíveis aos usuários *publishers*, além disso são eles que devem atribuir valores as varíaveis de cada tópico, ocasionando um evento que será difundido aos *subscribers*.

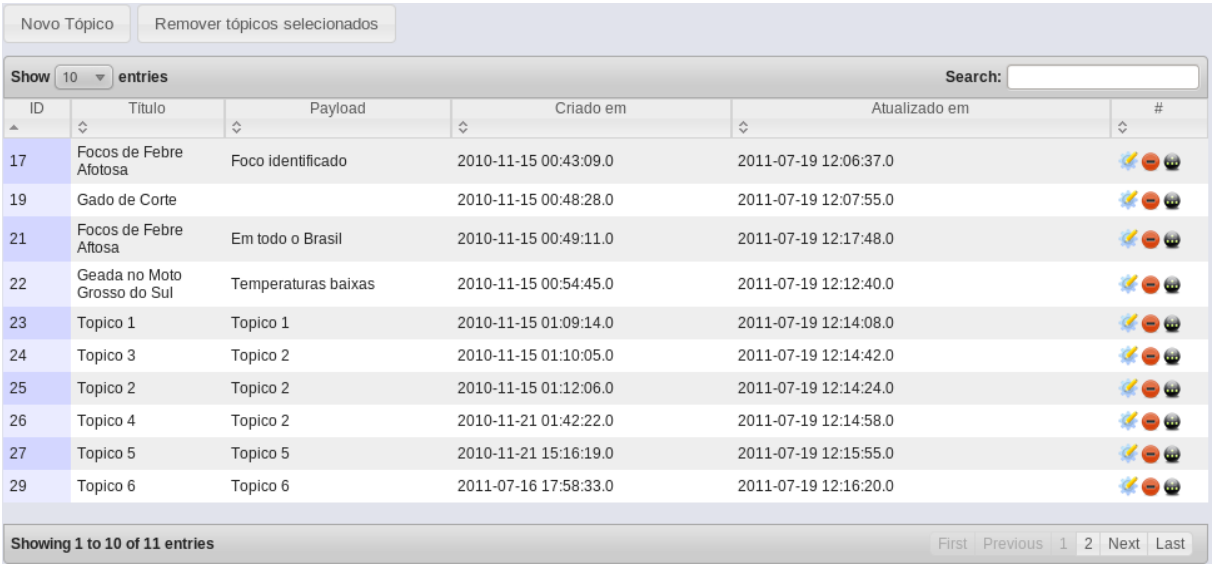


Figura 6.7: Orion - Gerência de Tópicos.

O recebimento deste aviso por parte do pecuarista pode evitar que o transporte de gado seja realizado neste(s) estado(s). Além de acionar todos os órgãos fiscalizadores para tomar as devidas ações preventivas e evitar a propagação da doença.

Após a criação do tópico é possível vincular varíaveis, que possuem um tipo definido de dado além de um comentário explicativo de sua utilização, conforme Figura 6.8. Se um foco de doença é confirmado em Mato Grosso do Sul, por exemplo, o e-SAPI atribui o valor MS para a variável **UF** (Unidade Federativa).

Para efeito de prototipação, apenas os tipos Inteiro, String, Real e Bit foram implementados, no entanto, outros tipos podem ser implementados. Cada tipo de dado define, além de valores válidos que podem ser utilizados nas regras, as possíveis operações que podem ser efetuadas, como por exemplo $<, >, =, \leq, \geq$, etc. Todos esses tipos de dados são componentes, que podem ser removidos ou edicionados de Orion a qualquer momento sem afetar o núcleo principal.

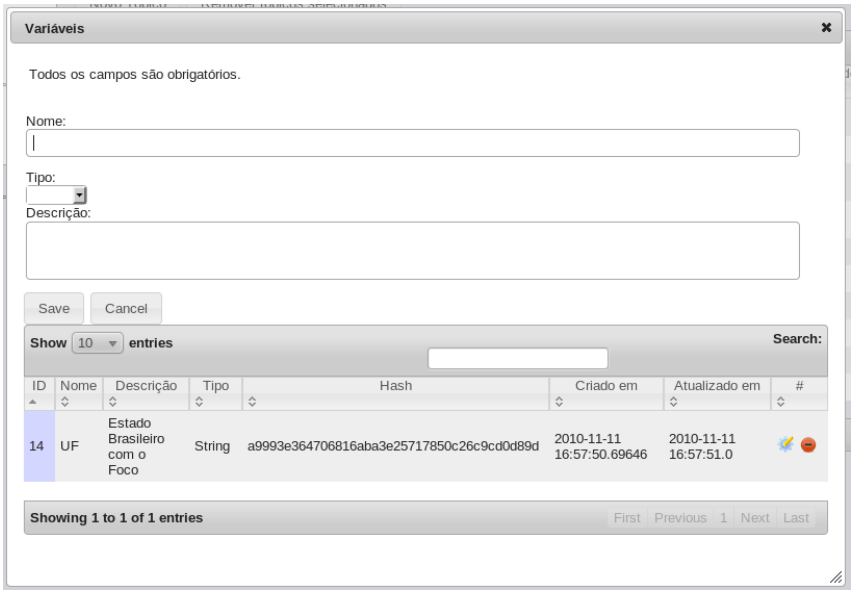


Figura 6.8: Orion - Gerência de Varíaveis.

Após a criação dos tópicos e variáveis pelo *publisher*, chega a vez do *subscriber* fazer a sua parte, assinando os tópicos publicados. A criação de regras deste protótipo do software Orion, Figura 6.9 é baseado na construção de expressões lógicas unidas por *OR* e *AND*. Neste primeiro momento somente os operadores $<, >, =, \leq, \geq$ estão disponíveis, demais operações podem ser criadas conforme novos tipos de dados forem desenvolvidos.

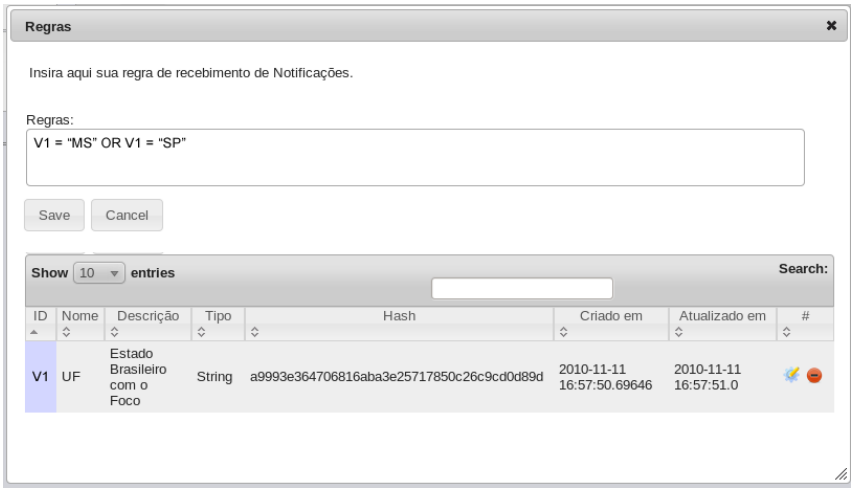


Figura 6.9: Orion - Criação de Regras.

Uma vez que as variáveis estão definidas, cabe agora ao provedor de serviço (e-SAPI) inserir valores para sejam geradas notificações para todos os *subscribers* (consumidores, vigilância sanitária, pecuaristas, compradores internacionais, entre outros).

Para a publicação de valores é necessário usuar o *Web Service* RESTFull de Orion. O mecanismo de publicação é simples e objetivo, basta enviar via POST os dados da variável que se dejesa atualizar e aguardar uma resposta de sucesso via protocolo HTTP.

O código 200 indica confirmação de atualização, 401 para não autorizado, 404 para não autorizado e 500 erro de atualização.

```
POST http://URL_ORION/api/variable/

<?xml version="1.0"?>
<variable id="10" type="STRING" date="2011-10-21T19:07:30Z">
    <value>
        MS
    </value>
</variable>
```

De forma análoga, e possível requisitar via *Web Service* todas as variáveis e valores destas variáveis que pertencem a um tópico. Como mostrado a seguir:

```
GET http://URL_ORION/api/variable/topic/201234

<?xml version="1.0"?>
<variable-list>
   <variable id="10" type="STRING" createAt="2011-10-21T19:07:30Z">
    <value createAt="2011-10-21T19:07:30Z">MS</value>
    <value createAt="2011-04-11T19:07:30Z">SP</value>
    <value createAt="2011-01-13T19:07:30Z">PR</value>
 </variable>
</variable-list>
```

O banco de dados para que toda a lógica de Orion funcione gira em torno de basicamente 4(Quatro) tabelas principais, que são ilustradas na Figura 6.10. São as entidades necessárias para armazenar tópicos, variáveis, valores que cada variável vai no decorrer do tempo e por fim, as regras definidas pelos usuários.

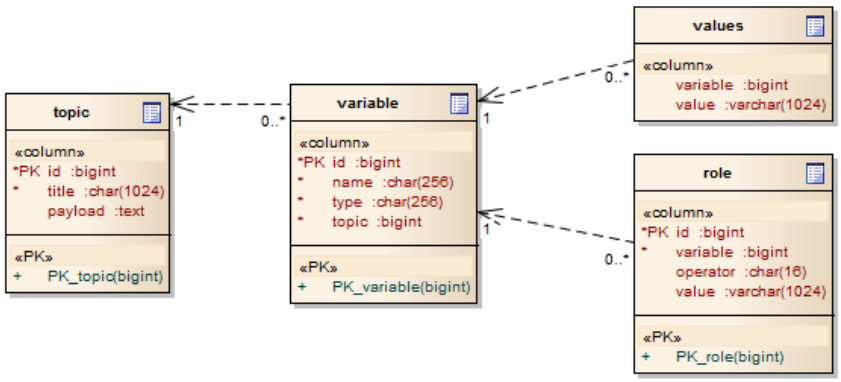


Figura 6.10: Banco de dados em UML.

No exemplo de interação entre leitura de e-GTAs com as notificações geradas por Orion para avisos sanitários, é necessário que na tabela **variable** esteja o campo NIRF,

número de inscrição do imóvel rural na Secretaria da Receita Federal, e na tabela **values** o NIRF das fazendas suspeitas de foco.

Desta forma, se um animal abatido hoje passou nesta fazenda algum tempo atrás na época do foco, o consumidor pode ser alertado do problema no ato da compra, simplesmente lendo o código QR Code impresso na carne embalada no supermercado. A Figura 6.11 ilustra esta interação entre buscar a e-GTA depois de ler o código QR Code e depois verificar por possíveis notificações de sanidade animal.

Esta busca não necessariamente precisa ser pela número da e-GTA, mas também pelo número do SISBOV (Serviço de Rastreabilidade da Cadeia Produtiva de Bovinos e Bubalinos). Individual para cada animal, a única mudança no diagrama da Figura 6.11 é a primeira chamada de Eros, que ao invés de buscar a e-GTA, busca informações de onde o animal esteve.

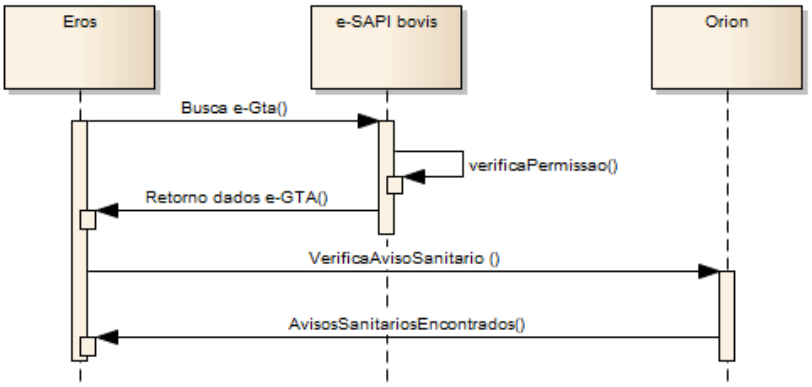


Figura 6.11: Eros interagindo com e-SAPI e Orion.

6.5 Teste e Validação

Com estas funcionalidades de Orion e Eros protipadas, várias perguntas já puderam ser respondidas, entre elas o tamanho do arquivo executável final do Eros que foi de apenas 408KB. Seu consumo de memória em tempo de excução ficou em torno de 17MB.

Em áreas sem Internet o software Eros foi capaz de detectar a reentrada e conexão em uma rede sem fio, identificando também se tal rede é de alta velocidade. Fato que fornece um mecanismo de aguardo de conexão para solicitações pendentes tanto ao e-SAPI *bovis* quanto ao Orion. No caso de uso para pecuária, a falta de conexão com redes de alta velocidade áreas rurais será um problema certo.

Orion possui um banco com poucas tabelas, excluindo tabelas de metadados como cidade e estado, e tabelas de permissões de usuários. Desenvolvido em JAVA, pode ser implantado em servidores de aplicação robustos para suportar altas cargas de acesso.

Para testar o software Eros, foi utilizado um *Smartphone* com as seguintes características:

Tabela 6.1: Dados dispositivo de teste para Eros.

Dispositivo	HTC Hero
Sistema Opercional	Android 2.1 Update 1
Resolução de Tela	320x480 ou 3.2”
Memória ROM	512 MB
Memória RAM	288 MB
SD Card	2 GB
CPU	Qualcomm MSM7200A, 528 MHz

Para testes de geoposicionamento, tanto redes Wifi, Cell-ID e GPS foram utilizados. Para tráfego de dados as redes 3G e *Wifi* foram utilizadas, ambas com desempenho rápido no tráfego de dados entre Eros e Orion.

Como mencionado anteriormente, esta implementação possui um ponto negativo que somente foi testado em celulares *smartphones* com a plataforma Android. No entanto, trabalhos futuros podem estender as mesmas funcionaliades atuais para outras plataformas. Outra possibilidade e implementar usando a tecnologia Adobe Air, que possibilida a mesma implementação funcionar em diversas plataformas.

O principal ponto positivo é que a aplicação se comportou conforme esperado quando testada por um longo período de tempo, sem consumir memória em excesso ou causar um alto processamento. Além disso, Eros consegue gerenciar o uso de funcionalidades de acordo com o contexto do usuário, conforme planejado.

Se houvesse uma mudança de domínio de aplicação, Orion seria reaproveitado 100% e Eros, precisaria apenas que a parte específica do domínio em questão fosse implementado novamente, menos de 20% do código.

6.6 Considerações Finais

A validação dos softwares Eros e Orion mostrou a gama de possibilidades de serviços para a cadeia produtiva da Carne. A notificação do consumidor final de que ele está comprando carne que possivelmente esteve em uma área infectada, é um grande avanço para a escolha de carne de boa qualidade.

A vigilância sanitária consegue realizar rastreamentos de carregamentos inteiros, apenas lendo o QR Code impresso na e-GTA com o uso de celulares. Futuramente, podem até saber o histórico de vacinação dos animais de um carregamento, aumentando a fiscalização contra carnes sem procedência e de má qualidade.

Capítulo 7

Conclusão

7.1 Contribuições

Este projeto teve início com uma revisão bibliográfica sobre sistema Publish-Subscribe e Serviços Baseados em Localização, e notou-se que existem diversos desafios a serem solucionados, mostrando este tema ser um campo de pesquisa promissor.

A necessidade de um mecanismo simples, porém preciso de notificação de usuários, levou a uma busca por softwares já existentes e identificar suas vantagens e desvantagens. Após uma intensa análise, concluiu-se que a criação de um sistema pub-sub baseado em componentes seria viável e escalável à medida que novas demandas surgissem.

A cadeia produtiva da carne é uma área da economia que necessita de mecanismos mais ágeis, automatizado e móvel para obtenção de informação acerca de eventos que envolvem a temática de Pecuária de Precisão. Desta forma, a construção de um sistema Pub-Sub aplicado ao problema da rastreabilidade, só trará beneficio à sociedade.

Após um periodo de planejamento e implementação dos softwares *mobile* Eros e Orion, iniciou-se um período de customização para o problema da rastreabilidade bovina, mostrando seu verdadeiro potencial e as reais possibilidades ao se integrar a Plataforma e-SAPI *bovis*, se tornando a mais nova tecnologia a serviço da Qualidade da Carne.

Assim, as principais contribuições deste trabalho são:

1. Revisão bibliográfica dos principais modelos de componente de software: JavaBeans, EJB, Corba, COM/COM+, .NET, Web Services, Koala, Kobra, SOFA, UML, PECOS e FRACTAL;
2. Comparativo entre mecanismos de georeferenciamento;
3. Evolução do módulo e-GTA do e-SAPI *bovis* para impressão de QR Codes. Até então, apenas existia o código de barras, que impossibilitava a quantidade de informação fornecida;

4. Especificação e implementação do Software *mobile* Eros, responsável pela coleta de informações de contexto do usuário e recebimento de notificações. Além de um leitor integrado de QR Code, tornando-se uma ferramenta mobile de consulta de e-GTAs e animais, mostrando na tela do celular o mapa por onde bovino esteve durante sua vida;

5. Especificação e implementação do Software Orion, capaz de fornecer uma interface de criação de tópicos, onde em cada tópico é possível vincular variáveis que qualificam o conteúdo dos eventos gerados por este software. Tais variáveis, podem ser atualizadas via *Web Service* de forma automatizada ou manualmente;

6. Especificação do padrão XML para envio de valores às variáveis de tópico, que podem ser recuperadas em um padrão XML também especificado neste projeto;

7. Estudo de caso com a Plataforma e-SAPI *bovis* mostrando as várias possibilidades de serviços, que visam automatizar o tráfego de informações na cadeia produtiva da carne.

8. Fortalecimento da linha de pesquisa em reutilização de software, e em georeferenciamento na FACOM/UFMS.

Este trabalho, além de publicações em congressos inerentes à área de pesquisa, também gerou outros frutos que estão listados abaixo:

Artigos

Silva, M. A. I. d. & et. al. Serviços Baseados em Localização para Sistema de Rastreabilidade na Plataforma e-SAPI bovis, VII Congresso Brasileiro de Agroinformática, Viçosa, Minas Gerais, 2009.

Silva, M. A. I. d. & et. al. Serviços Baseados em Localização e Publish-Subscribe para Pecuária de Precisão, XXXVI Conferência Latino Americana de Informática, Assunção - Paraguai 2010.

Livros

Capítulo **Orion e Eros: Mobilidade no Manejo e Segurança Sanitária** do Livro **Pecuária de Precisão** que será publicado pela Embrapa Gado de Corte, o título do livro ainda é provisório.

Projetos de Conclusão de Curso

Dois projetos de conclusão de cursos foram propostos a partir de pesquisas e problemáticas abordadas neste projeto:

- SISMA - Sistema Baseado em Localizção para Monitoramento de Areas em Smartphones, sob Orientação da Prof. Dra. Hana Karina Salles Rubinsztejn.
- DroidSisma - Sistema Baseado em Localização para Monitoramento de Areas em Smartphones com Android OS, sob Orientação da Prof. Dra. Hana Karina Salles Rubinsztejn. Este ainda está em andamento.

Projeto de Pesquisa

O projeto de pesquisa entitulado **Plataforma para a Rastreabilidade Bovina** foi aprovado pela Fundação de Apoio ao Desenvolvimento do Ensino, Ciência e Tecnologia do Estado de Mato Grosso do Sul, para dar continuidade ao projeto e transformá-lo em produto comercial com a parceria das empresas Olimpo Tecnologia e Embrapa Gado de Corte.

7.2 Limitações da Proposta

Orion ainda não está integrado com um servidor de envio de SMS, portanto a comunicação atual com o software Eros é baseado na Internet. Como definido na proposta inicial, Orion é baseado em componentes e poderá ter novos módulos que implementam novos protocolos de entrega de notificações, tornando-o ainda mais robusto.

Além disso, mesmo recebendo dados de contexto dos usuários, Orion ainda não realiza customização de conteúdo complexas, tais como envio e redução da resolução de imagens. Por outro lado, é capaz de definir a quantidade de dados que é possível enviar para um dispositivo, levando-se em conta a capacidade do hardware e o link de Internet ao qual este dispositivo está conectado.

7.3 Trabalhos Futuros

Alguns trabalhos futuros são necessários para que Orion tenha uma melhor performance no momento da avaliação de regras além de outros serviços, tais como:

- Desenvolver um mecanismo de balanço de carga para o banco de dados de Orion, utilizando, por exemplo, clusterização com PostgreSQL ou um banco de dados mais robusto como Oracle.
- Implementar as principais ferramentas de criação de tópicos, variáveis e regras de Orion no software mobile Eros, tornando a publicação de eventos possível de ser feita de qualquer lugar do mundo, sem o uso do computador.
- Implementar Eros para teste em dispositivos com pouca memória RAM;

- Implementar um estudo de caso com redes WSN para serviços climáticos, uma estação base recebe os dados do sensores em campo e os envia para orion alimentando varíaveis. Isto propiciaria, por exemplo, saber se na lavoura de uma fazenda começou a chover ou se a barragem de uma represa está chegando ao um nível crítico e os moradores devem ser alertados.

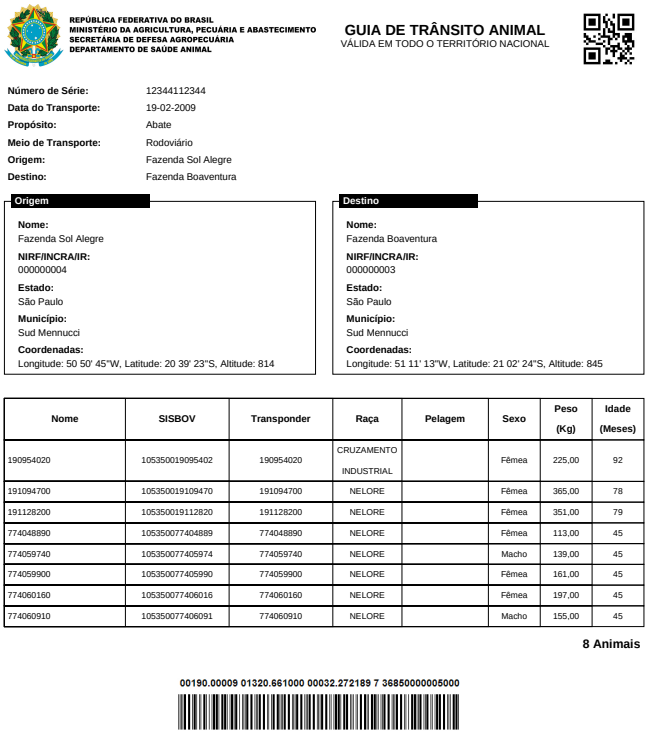
- Otimização do mecanismo de envio de notificações, agrupando os usuários em grupos de interesse antes do envio das mensagens, envitando processamento excessivo do servidor;

- Desenvolver um portal na Web para disponibilização de serviços de notificações dos mais diversos domínios, onde cada usuário poderá se cadastrar e assinar tópicos de seu interesse;

- Realizar estudo de caso na área de saúde pública, provendo avisos sobre surtos epidemiológicos na população como dengue e outras doenças infectocontagiosas.

Apêndice I: e-GTA - Guia de Trânsito Animal Eletrônica

A Figura abaixo mostra um exemplo de e-GTA eletrônica gerada pelo e-SAPI *bovis* já com um QR code impresso.



REPÚBLICA FEDERATIVA DO BRASIL
MINISTÉRIO DA AGRICULTURA, PECUÁRIA E ABASTECIMENTO
SECRETÁRIA DE DEFESA AGROPECUÁRIA
DEPARTAMENTO DE SAÚDE ANIMAL

GUIA DE TRÂNSITO ANIMAL
VÁLIDA EM TODO O TERRITÓRIO NACIONAL

Número de Série: 12344112344
Data do Transporte: 19-02-2009
Propósito: Abate
Meio de Transporte: Rodoviário
Origem: Fazenda Sol Alegre
Destino: Fazenda Boaventura

Origem

Nome:
Fazenda Sol Alegre
NIRF/INCRA/IR:
000000004
Estado:
São Paulo
Município:
Sud Mennucci
Coordenadas:
Longitude: 50 50' 45"W, Latitude: 20 39' 23"S, Altitude: 814

Destino

Nome:
Fazenda Boaventura
NIRF/INCRA/IR:
000000003
Estado:
São Paulo
Município:
Sud Mennucci
Coordenadas:
Longitude: 51 11' 13"W, Latitude: 21 02' 24"S, Altitude: 845

Nome	SISBOV	Transponder	Raça	Pelagem	Sexo	Peso (Kg)	Idade (Meses)
190954020	105350019095402	190954020	CRUZAMENTO INDUSTRIAL		Fêmea	225,00	92
191094700	105350019109470	191094700	NELORE		Fêmea	365,00	78
191128200	105350019112820	191128200	NELORE		Fêmea	351,00	79
774048890	105350077404889	774048890	NELORE		Fêmea	113,00	45
774059740	105350077405974	774059740	NELORE		Macho	139,00	45
774059900	105350077405990	774059900	NELORE		Fêmea	161,00	45
774060160	105350077406016	774060160	NELORE		Fêmea	197,00	45
774060910	105350077406091	774060910	NELORE		Macho	155,00	45

8 Animais

00190.00009 01320.661000 00032.272189 7 36850000005000

Guia de Trânsito Animal - 1 de 1

Referências Bibliográficas

[1] Abiec. Brasil lidera ranking de exportação de carne bovina, 2009. Disponível em http://www.abiec.com.br/ [08/01/2009].

[2] Adusei, I., Kyamakya, K., and Erbas, F. Location-based services: advances and challenges. *Electrical and Computer Engineering, 2004. Canadian Conference on 1* (2004), 1 – 7.

[3] Aliance, O. H. Android, 2010. Disponível em http://www.openhandsetalliance.com/ [29/01/2010].

[4] Alonso, G., Casati, F., Kuno, H., and Machiraju, V. *Web Services: Concepts, Architectures and Applications*. Springer-Verlag, 2004.

[5] Alvaro, A., Almeida, E. S. d., and Meira, S. R. d. L. A software component quality framework. *SIGSOFT Softw. Eng. Notes 35*, 1 (2010), 1–18.

[6] Apache. Activemq, 2009. Disponível em http://activemq.apache.org/ [12/01/2009].

[7] Araujo, R. B. d. Computação ubíqua: Princípios, tecnologias e desafios. *XXI Simpósio Brasileiro de Redes de Computadores* (2003).

[8] Atkinson, C., Paech, B., Reinhold, J., and Sander, T. Developing and applying component-based model-driven architectures in kobra. In *Enterprise Distributed Object Computing Conference, 2001. EDOC '01. Proceedings. Fifth IEEE International* (2001), pp. 212–223.

[9] Becker, C., and Durr, F. On location models for ubiquitous computing. vol. 9, Springer-Verlag, pp. 20–31.

[10] Bell, G., and Dourish, P. Yesterday's tomorrows: notes on ubiquitous computing's dominant vision. vol. 11, Springer-Verlag, pp. 133–143.

[11] Bellavista, P., Kupper, A., and Helal, S. Location-based services: Back to the future. *Pervasive Computing, IEEE 7*, 2 (Junho 2008), 85–89.

[12] Bruneton, E., Coupaye, T., and Leclercq, M. An open component model and its support in java. In *In Proceedings of 7th CBSE* (2004), Springer-Verlag, pp. 7–22.

[13] Bures, T., Hnetynka, P., and Plasil, F. Sofa 2.0: Balancing advanced features in a hierarchical component model. In *Software Engineering Research, Management and Applications, 2006. Fourth International Conference on* (Aug. 2006), pp. 40–48.

[14] Carromeu, C., and et al. e-gta: Sistema de guia eletrônica de trânsito de animais. *VII Congresso Brasileiro de Agroinformática* (2009).

[15] Carzaniga, A., Rosenblum, D. S., and Wolf, A. L. Achieving scalability and expressiveness in an internet-scale event notification service. In *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing* (New York, NY, USA, 2000), ACM, pp. 219–227.

[16] Cáceres, E. N., Pistori, H., Turine, M. A. S., Pires, P. P., Soares, C. O., and Carromeu, C. Computational precision livestock. *II Workshop of the Brazilian Institute for Web Science Research* (Agosto 2011).

[17] Cetic.br. Pesquisa sobre o uso das tecnologias da informação e da comunicação no brasil - 2008, 2008. Disponível em http://www.cetic.br/publicacoes/index.htm [16/02/2010].

[18] Clarke, M., Blair, G. S., Coulson, G., and Parlavantzas, N. An efficient component model for the construction of adaptive middleware. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg* (London, UK, 2001), Middleware '01, Springer-Verlag, pp. 160–178.

[19] Costa, C. A. d., Yamin, A. C., and Geyer, C. F. R. Toward a general software infrastructure for ubiquitous computing. *IEEE Pervasive Computing 7* (2008), 64–73.

[20] Coulson, G., Blair, G., Grace, P., Taiani, F., Joolia, A., Lee, K., Ueyama, J., and Sivaharan, T. A generic component model for building systems software. *ACM Trans. Comput. Syst. 26*, 1 (2008), 1–42.

[21] Deitel, H. M. *XML Como Programar*. Bookman, 2003.

[22] Deitel, H. M., and Deitel, P. J. *Java: Como Programar*, 6a edição ed. 2005.

[23] Dekleva, S., Shim, J., Varshney, U., and Knoerzer, G. Evolution and emerging issues in mobile wireless networks. *Commun. ACM 50*, 6 (2007), 38–43.

[24] Embrapa. Bpa - boas práticas agropecuárias. Disponível em http://bpa.cnpgc.embrapa.br [21/01/2010].

[25] Eugster, P. Type-based publish/subscribe: Concepts and experiences. *ACM Trans. Program. Lang. Syst. Vol. 29* (January 2007).

[26] Eugster, P. T., Felber, P. A., Guerraoui, R., and Kermarrec, A.-M. The many faces of publish/subscribe. *ACM Comput. Surv. 35*, 2 (2003), 114–131.

[27] Fcgov. Fort collins flood warning system, 2010. Disponível em http://www.fcgov.com/stormwater/fld-warning.php [15/02/2010].

[28] Fernandes, P. Publish-subscribe middleware. *Departamento de Ciência de Computadores - Faculdade de Ciências da Universidade do Porto* (2009), 1–5.

[29] Fielding, R. T. *Architectural styles and the design of networked-based software architectures*. PhD thesis, Dept. of Information and Computer Science, University of California, 2000.

[30] Gill, N. S. Importance of software component characterization for better software reusability. *SIGSOFT Softw. Eng. Notes 31*, 1 (2006), 1–3.

[31] Gill, N. S. Importance of software component characterization for better software reusability. *SIGSOFT Softw. Eng. Notes 31*, 1 (2006), 1–3.

[32] Grace, P., Blair, G. S., and Samuel, S. A reflective framework for discovery and interaction in heterogeneous mobile environments. *SIGMOBILE Mob. Comput. Commun. Rev. 9* (January 2005), 2–14.

[33] Hadley, M. J. *Web Application Description Language (WADL)*. Sun Microsystems Inc., Novembro 2006. Disponível em https://wadl.dev.java.net/wadl20061109.pdf [18/12/2009].

[34] Han, C.-C., Kumar, R., Shea, R., and Srivastava, M. Sensor network software update management: a survey. *Int. J. Netw. Manag. 15* (July 2005), 283–294.

[35] Huang, Y., and Garcia-Molina, H. Publish/subscribe in a mobile environment. *Wirel. Netw. 10*, 6 (2004), 643–652.

[36] Kos, T., Grgic, M., and Kitarovic, J. Location technologies for mobile networks. In *Systems, Signals and Image Processing, 2007 and 6th EURASIP Conference focused on Speech and Image Processing, Multimedia Communications and Services. 14th International Workshop on* (Maribor, Junho 2007), Zagreb Univ., Zagreb, pp. 319–322.

[37] Koutsiouris, V.and Polychronopoulos, C., and Vrechopoulos, A. Developing 3g location based services: The case of an innovative entertainment guide application. In *Management of Mobile Business, 2007. ICMB 2007. International Conference on the* (Toronto, Ont, 2007).

[38] Lau, K.-K. Software component models. In *ICSE '06: Proceedings of the 28th international conference on Software engineering* (New York, NY, USA, 2006), ACM, pp. 1081–1082.

[39] Mackenzie, C. M., and et. al. *Reference Model for Service Oriented Architecture 1.0*. OASIS, 2006.

[40] Microsoft. Biztalk server, 2009. Disponível em http://msdn.microsoft.com/pt-br/biztalk/default.aspx [12/01/2009].

[41] Microsoft. Com: Component object model, 2009. Disponível em http://www.microsoft.com/com/default.mspx [26/12/2009].

[42] Microsoft. .net documentation, 2009. Disponível em http://www.microsoft.com/net/ [26/12/2009].

[43] Milo, T., Zur, T., and Verbin, E. Boosting topic-based publish-subscribe systems with dynamic clustering. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 2007), ACM, pp. 749–760.

[44] Monson-Haefel, R. *Enterprise JavaBeans*, 3rd edition ed. O'REILLY, Setembro 2001.

[45] Mota, E. G., FERREIRA, J. J., SERGUEI, B., and LOPES, R. S. Sapi - sistema agropecuário de produção integrada. In *II Simpósio sobre Desafios e Novas Tecnologias na Bovinocultura de Corte - SIMBOI'05.* (Brasília-DF, 2005).

[46] Noaa. National oceanic and atmospheric administration, 2010. Disponível em http://www.noaa.gov/ [15/02/2010].

[47] Nunes, S., and David, G. Uma arquitetura web para serviços web. *XATA 2005 - XML: Aplicações e Tecnologias Associadas* (2005), 1–11.

[48] Oliveira, J. P. M. d. Computação ubíqua, o possível, o útil e o razoável, 2009. Disponível em http://palazzo.pro.br/cronicas/029.htm [18/02/2010].

[49] omg. Corba component model, versão 3.0, 2002. Disponível em http://www.omg.org/technology/documents/formal/components.html [26/12/2009].

[50] omg. *UML 2.0 Superstructure Specification.* OMG, 2003. Disponível em http://www.omg.org/cgi-bin/doc?ptc/2003-08-02.

[51] omg. Unified modeling language, 2009. Disponível em http://www.uml.org [29/12/2009].

[52] omgoficial. Omg - object management group, 2011. Disponível em http://www.omg.org/ [18/07/2010].

[53] Oracle. *Oracle Tuxedo.* Oracle Corporation, 2009. Disponível em http://www.oracle.com/techmology/products/tuxedo/pdf/tuxedo-datasheet.pdf.

[54] Peng, J.-j. A survey of location based service for galileo system. *Computer Science and Computational Technology, 2008. ISCSCT '08. International Symposium* (Dezembro 2008), 737–741.

[55] Pietzuch, P., and Bacon, J. A distributed event-based middleware architecture. *22nd International Conference on Distributed Computing Systems Workshops* (2002), 611 – 618.

[56] PostgreSQL. Sobre, 2010. Disponível em http://www.postgresql.org.br [28/01/2010].

[57] Research, T. I. Location based services market, 2008-2013, 2008. Disponível em http://www.insight-corp.com/reports/lbs08.asp [20/01/2010].

[58] Roxin, A. M., Dumez, C., Wack, M., and Gaber, J. Middleware models for location-based services: a survey. In *AUPC '08: Proceedings of the 2nd international workshop on Agent-oriented software engineering challenges for ubiquitous and pervasive computing* (New York, NY, USA, 2008), ACM, pp. 35–40.

[59] Rubinsztejn, H. K. S. *Suporte à adaptação de conteúdo sensível a contexto para dispositivos móveis em sistemas publish-subscribe.* PhD thesis, Pontifícia Universidade Católica do Rio de Janeiro, 2008.

[60] Sabil, S., and Jawawi, D. Integration of pecos into marmot for embedded real time software component-based development. In *Software Engineering Advances, 2009. ICSEA '09. Fourth International Conference on* (Sept. 2009), pp. 265–270.

[61] Sandim, H. d. C., and et al. e-sapi bovis: Plataforma web para rastreabilidade bovina. *VII Congresso Brasileiro de Agroinformática* (2009).

[62] Saraiva, A. M., Corrêa, P. L. P., Sobrinho, O. G., and Cunha, G. J. d. Resultados do seminário de rastreabilidade da informação em cadeias produtivas do agronegócio. *Seminário de Rastreabilidade da Informação em Cadeias Produtivas do Agronegócio* (2005).

[63] Schiller, J., and Voisard, A. *Location-Based Services.* Elsevier, 2004.

[64] Scott, K. *O processo unificado explicado.* Bookman, 2003.

[65] Shao, W., Huang, G., and Zhao, H. Modeling of component based systems. In *ICSE '06: Proceedings of the 28th international conference on Software engineering* (New York, NY, USA, 2006), ACM, pp. 1077–1078.

[66] Sharma, A., Grover, P. S., and Kumar, R. Reusability assessment for software components. *SIGSOFT Softw. Eng. Notes 34*, 2 (2009), 1–6.

[67] Sharma, A., Grover, P. S., and Kumar, R. Reusability assessment for software components. *SIGSOFT Softw. Eng. Notes 34*, 2 (2009), 1–6.

[68] Silva, M. A. I. d., and et. al. Serviços baseados em localização para sistema de rastreabilidade na plataforma e-sapi bovis. *VII Congresso Brasileiro de Agroinformática* (2009).

[69] Smanchat, S., Ling, S., and Indrawan, M. A survey on context-aware workflow adaptations. In *Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia* (New York, NY, USA, 2008), MoMM '08, ACM, pp. 414–417.

[70] Sun. Enterprise java beans especification, version 3.0, 2009.

[71] Sun. Javabeans specification, 2009. Disponível em http://java.sun.com/products/javabeans/docs/spec.html [26/12/2009].

[72] Szyperski, C. *Component Software - Beyond Object-Oriented Programming Second Edition*. Addison-Wesley / ACM Pres, 2002.

[73] Szyperski, C. Component technology: what, where, and how? In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering* (Washington, DC, USA, 2003), IEEE Computer Society, pp. 684–693.

[74] Ueyama, J., and et al. Component-based system software: A generic approach. *SBES 2007 - XXI Simpósio Brasileiro de Engenharia de Software* (2007).

[75] van Ommering, R., van der Linden, F., Kramer, J., and Magee, J. The koala component model for consumer eletronics software. *Computer 33* (2000), 78–85.

[76] w3c. Xpath, jan 2007. Disponível em http://www.w3.org/TR/xpath20/ [21/02/2010].

[77] w3c. Standard web services, 2009. Disponível em http://www.w3.org/standards/webofservices/ [29/12/2009].

[78] Weiser, M. The computer for the 21st century. *IEEE* (1995), 933–940.

[79] Winter, M., and et al. Components for embedded software - the pecos approach.