Faculdade de Engenharia da Universidade do Porto

FEUP

Computação Ubíqua para Aplicações em Saúde

Filipe Coelho dos Santos

VERSÃO FINAL

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Telecomunicações

Orientador: Prof. Dr. Rosaldo J. F. Rossetti

Julho de 2009

Resumo

Nestes últimos anos, os Cuidados de Saúde têm-se tornado, na sua grande maioria, dependentes dos Sistemas de Informação. A complexidade destes sistemas acarreta grandes desafios, pois altera substantivamente as infra-estruturas e a funcionalidade dos serviços existentes, sem causar debilidade nas obrigações dos profissionais de Saúde assim como no bem-estar dos pacientes.

No contacto directo com este tipo de ambientes, nomeadamente grandes Hospitais públicos portugueses, vários problemas têm sido identificados.

É nossa convicção que as tecnologias envolvidas na Computação Ubíqua suportam um grande potencial na hora de abordar alguns destes problemas.

Para isso, foi desenvolvido um protótipo de sistema middleware, baseado neste novo paradigma, com o objectivo de fornecer aos Sistemas de Informação, à disposição dos utilizadores de Hospitais e de Clínicas de Saúde, uma forma rápida e contextualizada de fornecimento de informação acerca dos Serviços, tendo a sua plataforma como base a mobilidade dos seus utilizadores, ao longo das complexas instalações e no imediato reconhecimento por proximidade do utilizador.

O sistema integra a tecnologia de Identificação por Rádio Frequência (RFID), pois a localização é um atributo particularmente precioso na computação sensível ao contexto, a qual é essencial para o mecanismo de descoberta de serviços neste tipo de ambiente. Neste sistema é abordada a Arquitectura Orientada a Serviços (SOA), que oferece uma visão realista de ambientes pervasivos, onde serviços heterogéneos são oferecidos por vários fabricantes. Também é usada uma abordagem do paradigma de Sistemas Multi-Agente (MAS), onde o sistema proposto foi projectado com o auxílio da metodologia Gaia, e é constituído por uma sociedade de agentes cooperativos, onde os agentes partilham um objectivo comum, e cada um assume um papel contendo conhecimentos, habilidades, e capacidades específicas, no sentido de realizar a sua parte rumo ao propósito da organização.

Abstract

In the past few years, Healthcare has become greatly dependent on the Information Systems. With time, the complexity of these systems brings about great challenges, by increasing transformations in existing infrastructures and services, without causing any debility on Health professionals obligations and patient´s welfare.

By direct contact with this kind of environment, especially in big Portuguese public Hospitals, several problems were identified.

It is our conviction that technologies involved in ubiquitous computing have a huge potential when some of these problems are approached.

To that end, a middleware system prototype was developed, based on this new paradigm, so it could provide Information Systems, employed by users of Hospitals and Health Clinics, with a rapid and contextualized form of information supply about services made available to users; the platform would have not only users mobility as its basis, but installations complexity and users recognition as well.

The system integrates Radio Frequency Identification (RFID) technology because location is a specially precious attribute in the context-aware computing, since it is essential for the service discovering mechanism in this kind of environment. In this system Service Oriented Architecture (SOA) is approached, which provides a realistic vision of pervasive environments, where heterogeneous services are offered by several manufacturers.

The Multi-Agent Systems (MAS) paradigm is used, underlying the proposed system projected with help of the Gaia methodology; it is formed by a cooperative agent society, where agents share a common goal, assuming each one a role with knowledge, skills and specific abilities, in a way that they can perform their part towards the organization purpose.

Agradecimentos

Este espaço é dedicado a todas as pessoas que contribuíram, de alguma forma, para a realização deste trabalho.

Um especial agradecimento à minha família pelo seu inestimável e incondicional apoio oferecido ao longo deste período.
Agradeço ao Dr. Rosaldo J. F. Rossetti pela colaboração, e conselhos dados durante a realização deste trabalho.
Agradeço também ao Dr. José Cunha, médico pneumologista no Hospital Sousa Martins, pela sua disponibilidade e perceptibilidade.
Por último, agradeço a todos aqueles que não foram mencionados, mas que de alguma forma também contribuíram para a elaboração deste trabalho

Filipe Santos

Índice

Lista de figuras

Lista de tabelas

Abreviaturas e Símbolos

AmI	Ambiente Inteligente
AOP	Programação Orientada a Aspectos
API	Interface de Programação de Aplicativos
BPM	Gestão do Processo de Negócio
BSD	Berkeley Software Distribution
CDMA	Code Division Multiple Access
CRT	Tubos de Raios Catódicos
EAI	Integração de Aplicações Corporativas
Gaia	Metodologia orientada a agentes Gaia
GSM	Global System for Mobile Communication
IP	Internet Protocol
IR	Infravermelhos
JADE	Java Agent DEvolopment Framework
JRE	Java Run Enviroment
JSP	Java Server Pages
LBS	Serviços Baseados na Localização
LEAP	Lightweight Extensible Agent Platform
LGPL	Lesser General Public License
LQI	Link Quality Indicator
MAS	Sistema Multi-Agente
OO	Programação Orientada a Objectos
OSGi	Open Service Gateway initiative
OWL	Ontology Web Language
PDA	Assistente Pessoal Digital
RBAC	Role-Based Access Control
RF	Radiofrequência
RFID	Identificação por Rádio Frequência
RSS	Potência de sinal recebido

RSSi	Indicador de Potência de sinal recebido
SOA	Arquitectura Orientada a Serviços
SOC	Computação Orientada a Serviços
TCP	Transmission Control Protocol
TDoA	Time Difference Of Arrival
TI	Tecnologias de Informação
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
W3C	World Wide Web Consortium
WiFi	Wireless Fidelity
XML	eXtensible Markup Language

Capítulo 1

Introdução

1.1 –Tema

Os Cuidados de Saúde estão a passar por uma transformação fundamental, em que a inovação incentiva o provedor de saúde a desenvolver capacidades para alcançar a agilidade, interoperabilidade e eficiência, necessárias para aumentar a qualidade e a produtividade, reduzindo custos. A inovação permite também a estas ampliar e transformar incrementalmente as infra-estruturas e serviços existentes ao longo do tempo, mas não sem causar debilidade nas obrigações dos profissionais de saúde assim como no bem-estar dos pacientes.

É nossa convicção que as tecnologias envolvidas na computação ubíqua suportam um grande potencial na hora de abordar alguns destes desafios.

O desenvolvimento de aplicações úteis no campo ubíquo dos Cuidados de Saúde requer tratar níveis substanciais de complexidade referentes à gestão do ambiente ubíquo, onde coexistem utilizadores e aplicações, uma vez que implica um ambiente computacional suavemente contínuo, tecnologia avançada de redes e interfaces específicas. Tem de ser sensível às características da presença humana e à sua personalidade, tratando das suas necessidades, capaz de ser activado com simples interacção, sintonizado com todos os sentidos humanos, adaptando-se aos utilizadores e ao contexto, actuando autonomamente. Qualidade de informação e de conteúdo deve estar acessível a qualquer utilizador, em qualquer lado, a qualquer hora e em qualquer dispositivo, mas não intrusivamente, e dum modo invisível, excepto quando preciso [Lin03].

A tecnologia de agentes tende a ser uma das principais tecnologias usadas na computação ubíqua para reconhecimento e análise de informação de contexto de ambientes distribuídos. Estes agentes de software providenciam entrega de informação mais rápida e precisa ao pessoal de saúde, permitindo uma tomada de decisão mais precisa e reduzindo os custos das transacções. Agentes podem usar e beneficiar das mais recentes tecnologias e padrões abertos, tal como SOA, trazendo assim, a promessa de facilidade de integração com componentes e sistemas legados.

1.2 – Objectivos

É objectivo desta dissertação a especificação e desenvolvimento duma arquitectura baseada em computação ubíqua para Serviços de Informação num ambiente de Clínicas de Saúde em diversas perspectivas, incluindo pacientes, médicos e serviços administrativos. A arquitectura a ser definida deverá ser baseada em serviços, e utilizar conceitos como sistemas multi-agente para integração e inter-operação de sistemas autónomos.

1.3 - Organização da Dissertação

Este relatório está estruturado da seguinte forma:

- Capítulo 2 – faz uma breve apresentação dos paradigmas de computação abordados neste projecto, nomeadamente Computação Ubíqua, Orientada a Serviços e Orientada a Agentes. É também realizado um levantamento de sistemas que empregam um ou vários destes paradigmas na sua concepção, destacando-lhes a arquitectura.

- Capítulo 3 - expõe a proposta dum Sistema de Informação para a Saúde baseado nos paradigmas revistos no capítulo anterior, na tentativa de ajudar a resolver alguns dos problemas detectados em Hospitais públicos portugueses.

- Capítulo 4 – apresenta detalhes da implementação do protótipo de acordo com o especificado no capítulo anterior. Expõe ainda uma das simulações realizadas, no intuito de comprovar a potencialidade da plataforma.

- Capítulo 5 - faz uma síntese do trabalho desenvolvido sumariando as conclusões. São também apresentadas perspectivas de desenvolvimento futuro.

Capítulo 2

Estado da Arte

2.1 - Computação Ubíqua

Revemos, neste capítulo, parte da história e as características que diferenciam o projecto de sistemas de computação ubíqua de outros sistemas de computação convencionais a que estamos habituados até hoje. Referimos, de seguida, alguns dos assuntos e dos desafios criados como resultado destas características.

2.1.1 – História

Inspirado por cientistas sociais, filósofos e antropólogos, em 1991, Mark Weiser e restantes colaboradores dos laboratórios Xerox PARK [Xer05] imaginaram o que a computação viria a tornar-se no futuro, quanto à invisibilidade da computação [Wei93] e à centralização desta nas tarefas dos utilizadores, onde o computador actua em plano de fundo e adapta-se às necessidades dos utilizadores, em oposição à situação actual, onde este deve aprender a lidar com o computador e gerir as suas aplicações. Neste novo cenário, a tecnologia é envolvida na concepção da vida do dia-a-dia, de tal modo que não se consiga distinguir desta. Weiser denominou este novo paradigma por Computação Ubíqua [Wei91].

Ainda que muitas vezes tratadas sem distinção, a computação ubíqua e a computação pervasiva referem-se a segmentos distintos de pesquisa e desenvolvimento tecnológicos [Kal02]. A computação ubíqua integra avanços na computação móvel e computação pervasiva. Enquanto a móvel centra-se no aumento de mobilidade de serviços entre ambientes, já a pervasiva concentra a sua atenção na capacidade de os dispositivos computacionais serem embutidos no ambiente, representado por um conjunto de entidades, tais como dispositivos, utilizadores e redes [Rom02], de modo a actuar de forma “discreta”, oferecendo omnipresença aos utilizadores e integrando a computação no ambiente físico em que está imersa.

2.1.2 - Características da Computação Ubíqua

Há uma série de características inerentes e únicas que distinguem a computação ubíqua do actual paradigma do desktop, tornando a convencional interacção homem-computador, que se tem vindo a construir até hoje, inadequada. Temos de nos adaptar ou então encontrar novos métodos, teorias e *frameworks* que guiem ordenadamente o desenvolvimento da computação ubíqua [Ban02].

2.1.2.1 - A Computação num Ambiente Social

A introdução da tecnologia de computação ubíqua terá um efeito social mais significativo comparado com a tecnologia convencional, alterando não só as nossas relações com os dispositivos mas também as relações individuais. Embeber-nos de materiais computacionais a par dum acrescento de outros materiais mais comuns, que não possuem capacidade computacional mas que se inserem no ambiente, resultará em novas maneiras de agir e interagir entre as pessoas naquele ambiente. Tal terá um efeito no seu comportamento social, resultando provavelmente em novas estruturas sociais; isto implica que não estamos apenas interessados nas relações interpessoais com aqueles que trabalham ao nosso lado (estando ausente a tecnologia) ou nas relações apenas entre pessoas e dispositivos (estando aqui presente a tecnologia), mas igualmente no sistema de relações entre homem-homem, que emergem na presença da tecnologia [Gru94], [Gru02], [Jes02].

2.1.2.2 - Objectivos e Situações Dinâmicas

O objectivo da computação ubíqua é estar em todo o lado a toda a hora. Utilizadores do serviço, cujos objectivos e situações se podem alterar dinamicamente, exigem que a computação ubíqua tenha a capacidade para responder dum modo adaptado.

Enquanto o utilizador poderá reconfigurar activamente o sistema de forma a adaptá-lo às novas condições, já na maioria das vezes o sistema poderá ter que inferir por si mesmo as mudanças respondendo de acordo com elas.

Assim se cria no sistema a necessidade de não só manter os objectivos do utilizador, e as tarefas e os recursos do sistema ao longo do tempo, mas também as condições ambientais. Simultaneamente, o sistema deverá ser capaz de compreender o raciocínio atrás dessas inferências referente às circunstâncias, permitindo-lhe ir aprendendo das inferências correctas e incorrectas.

Para além de ser difícil recolher toda a informação necessária para garantir um comportamento adaptável, é um desafio fazê-lo imperceptivelmente, para além de criar um enorme fardo nos recursos do sistema [Ban02].

2.1.2.3 - Heterogeneidade de dispositivos e restrições de recursos

A computação ubíqua terá de suportar múltiplos dispositivos de variadas capacidades de hardware, de modo que esteja em todo o lado. Estas capacidades resultam em transacções que influenciam inevitavelmente o desenvolvimento das aplicações e as suas capacidades. Com vários dispositivos ao alcance de cada utilizador, serão necessários mecanismos que ajudem a determinar que tarefas são as apropriadas a cada dispositivo. Passando os utilizadores dum dispositivo a outro, as aplicações, capazes de se moverem dum dispositivo para outro, ter-se-ão que reajustar à especificidade de cada dispositivo, optimizando a sua efectividade segundo as capacidades [Isl03].

À medida que os dispositivos se tornam mais móveis, mais constrangimentos surgem. Aqueles têm que ser cada vez menores, de modo a permitirem aos utilizadores que se desloquem convenientemente; este constrangimento do tamanho limita outros recursos, tais como tamanho do ecrã, poder de processamento, duração da bateria, etc., que por sua vez influenciam outros factores, como a conectividade e o desenvolvimento dos serviços e das aplicações [Ban02].

2.1.2.4 - Heterogeneidade de redes

De forma a obter mais cobertura, a computação ubíqua terá de ser compatível com o maior número de redes, para interconectar os vários dispositivos dispersos. Redes potencias incluem o GSM (*Global System for Mobile Communication*) ou o CDMA (*code division multiple access*) – desenvolvidos para os serviços de telemóvel –, as tecnologias de rede *wireless* 802.11x (cada vez mais popular), RFID, Bluetooth e IrDA, etc. Estas várias redes possuem características intrínsecas que influenciarão o modo como as usamos, tal como o diferente alcance de conectividade: Bluetooth e IrDA suportam conexões de curto alcance, enquanto a tecnologia *wireless* do 802.11x e do RFID já suporta distâncias um pouco maiores; as redes de telemóveis, por outro lado, conseguem chegar aos 35km de distância entre o aparelho e a estação.

Para além do alcance de conectividade, suportam ainda diferentes velocidades de transmissão, o que implica que uma conexão, usando uma rede particular, oferece diferentes recursos aos aparelhos e aplicações à disposição do utilizador. A mudança entre redes irá requerer uma gerência da sessão, criando desafios entre redes, quando ao serem atravessadas áreas sem sinal e outras cobertas se alterna entre online e off-line. O ideal seria que estas mudanças fossem transparentes para o utilizador, permitindo-lhe desconectar-se da rede sempre que o desejasse [Isl03].

2.1.3 - Questões e Desafios

Uma vez referidas as características, surge agora um conjunto interessante de questões e desafios.

2.1.3.1 - Sensibilidade ao contexto

Para os sistemas de computação ubíqua se tornarem invisíveis, enquanto se esforçam por serem minimamente intrusivos, têm de ser sensíveis ao contexto. Têm de saber do estado e vizinhança do utilizador, e de ter a habilidade de modificar o seu comportamento baseados nesta informação.

A sensibilidade do contexto é especialmente importante, porque a liberdade que resulta da tecnologia da computação ubíqua oferece o potencial de manejar os aparelhos durante uma deslocação, criando situações de uso e ambientes que mudam. Dada a sua cobertura, esta, por sua vez, cria desafios às infra-estruturas circundantes, feitas para suportar várias aplicações; poderá ter capacidades variáveis para realizar diferentes tarefas em tempos variados.

O contexto do utilizador engloba tanto factores humanos como do ambiente físico. Os factores humanos poderão consistir no estado psicológico (p. ex., temperatura do corpo, tensão), estado emocional (furioso, distraído, calmo), história pessoal, padrões comportamentais e circunstâncias sociais. O ambiente físico poderá consistir na localização, condições ambientais (ruído, claridade, etc.), e infra-estruturas envolventes.

Abowd e Mynatt [Abo00] referem os cinco "W's" do contexto, fornecendo um bom ponto de começo de como diferentes componentes podem estar juntos, de forma a fornecer um bom contexto e melhores tecnologias de computação ubíqua:

Quem (***W**ho*): a capacidade dos aparelhos para identificar não só o dono, mas outras pessoas e aparelhos vizinhos dentro do ambiente;

O quê (***W**hat*): a capacidade para interpretar a actividade e o comportamento do utilizador, usando esta informação para inferir o que o utilizador quer, e provendo a informação e a ajuda necessárias;

Onde (***W**here*): a capacidade para interpretar a localização do utilizador, usando-a para ajustar a funcionalidade. Este é um dos aspectos mais explorados do contexto;

Quando (***W**hen*): a capacidade para compreender a passagem do tempo, usando-a para compreender as actividades em redor e para fazer inferências;

Porquê (***W**hy*): a capacidade para compreender as razões por trás de algumas acções do utilizador. Tal pode implicar perceber o estado afectivo do utilizador (temperatura corporal, ritmo cardíaco, etc.);

Todavia, certas dificuldades essenciais abundam nesta área. Logo no cimo da lista está o problema de como reunir toda a informação requerida para perceber a situação, permitindo aos dispositivos operarem conscientes do contexto. Apesar de alguma desta informação poder fazer parte do espaço computacional, outros tipos de informação, que podem ser dinâmicos, têm de ser retirados em tempo real do ambiente do utilizador [Sat01].

Podemos encontrar exemplos de tal informação na localização, orientação, identidades de pessoas próximas, outros dispositivos com os quais o utilizador tem a capacidade de comunicar, estado psicológico, etc.

2.1.3.2 - Infra-estrutura

Com a multidão de dispositivos interconectados e invisíveis necessários para tornar realidade a computação ubíqua [Wei91] [Wei93], as infra-estruturas exigidas constituirão um maior desafio que as actuais. Dado que todos estes dispositivos precisam de comunicar entre si, as redes ou a infra-estrutura da comunicação dominarão as nossas necessidades de infra-estruturas. Isto irá criar um conjunto de questões e desafios, alguns dos quais estão já a ser atendidos, apesar de haver ainda muitas áreas potenciais de pesquisa:

Dispositivos: enquanto nos afastamos da tecnologia que conduz os dispositivos de utilização geral para outros ubíquos de tarefas específicas, estes dispositivos irão precisar de modos de se identificarem a si mesmos e de descreverem a outros o seu comportamento na rede. Para além de permitir as suas suaves inter-operações em benefício do utilizador, isto irá também facilitar ao utilizador o acrescento de novos dispositivos num sistema que já existe [Rus98].

Infra-estrutura do software: dependendo do contexto do utilizador, "a infra-estrutura do software, para aplicação ubíqua a executar, tem de ser capaz de encontrar, adaptar e entregar os serviços adequados ao ambiente informático do utilizador" [Ban02]. Para além do contexto do utilizador, outros factores, tais como as suas preferências ou as de outros utilizadores dentro do ambiente, terão também de ser levados em conta.

Espaços de informação: precisaremos de meios para criar espaços de informação. Tal ajudar-nos-á a impor ordem no caos nascido do fácil acesso que temos a vastas quantidades de informação, disponível a um simples clique, permitindo-nos criar associações entre alguns espaços que se sobrepõem [Rus98].

Redes: dado o grande número de tipos de rede disponíveis e as suas várias capacidades, as conexões serão intermitentes enquanto as aplicações mudem duma rede para outra, de forma a realizar uma tarefa específica ou prover um serviço. Haverá a necessidade de fornecer um interface standard entre os dispositivos e os vários tipos de rede, tornando independente a infra-estrutura do meio de comunicação [Kag02].

2.1.3.3 - A interacção do utilizador e a experiência

A interface do utilizador é um portal para a experiência global do utilizador. Apesar da natureza dispersa do interface permitir aos utilizadores interagir com o sistema através dum leque de diferentes dispositivos, cria, no entanto, um desafio para os designers: assegurar que a experiência da interacção para desempenhar uma tarefa particular não varie significativamente ao longo do alcance de potenciais dispositivos; doutro modo, os utilizadores, frustrados, confundir-se-iam. Se vários utilizadores vizinhos interagirem simultaneamente no mesmo interface, precisaremos de mecanismos que capturem e modelem tais questões, de forma a permitir uma resolução amigável. Noutras circunstâncias, a interacção com tais interfaces inteligentes estaria implícita –permitindo ao computador interpretar o comportamento do utilizador e as suas acções baseado no contexto.

Preocupar-nos-emos agora mais com as pessoas interagindo em ambientes mais complexos do que com interfaces ou sistemas de computador isolados. Isto significa uma radical transição do conhecimento duma interacção tradicional entre homem-computador, utilizada presentemente, para uma interacção implícita entre homem-computador [Sch00]. Estas são algumas das questões que necessitarão de ser atendidas, para além da concordância nalguns princípios, dado que os dispositivos diferentes que se reúnem para criarem uma experiência coerente provirão de diferentes fabricantes, criando um grande desafio de entendimento [Rus98].

2.1.4 - Tecnologias de Localização

Um problema de importância crucial em computação ubíqua é a determinação de localização de pessoas e objectos. Muitas decisões proactivas são essencialmente ou em parte dependentes da determinação de localização [Ben05], [Ben07].

2.1.4.1 – Sinais

As características relevantes da propagação de sinal inclui o alcance, velocidade de propagação, largura de banda disponível, características de difracção e refracção, restrições regulamentares, interferência, restrições de (potência/energia) e custo.

2.1.4.1.1 - Infravermelho

Devido à sua ubiquidade, os transreceptores de infravermelhos (IR) são baratos, compactos e de pouca potência/alimentação. A propagação de infravermelhos é rápida, mas a largura de banda efectiva é limitada pela interferência das luzes de ambiente e de outros dispositivos de infravermelhos no ambiente. Os sinais IR são reflectidos pela maior parte das superfícies mas difractam em alguns. O alcance típico é de 5 metros.

2.1.4.1.2 - Rádio Frequência

Os sinais de radiofrequência (RF) oferecem vários benefícios em relação aos de IR. Os sinais de RF difractam e passam através de materiais comuns de construção. Os sinais de RF são comparáveis aos de IR na velocidade de propagação, largura de banda e custo. Como o espectro de RF é fortemente regulado, os sistemas típicos operam aos 900MHz ou 2.45GHz e cumprem os regulamentos estabelecidos pela Anacom [Ana09], não requerendo licença no interior de edifícios, um intervalo comum de transmissão é de 10m – 30m.

2.1.4.1.3 - Campo Electromagnético Contínuo

Os campos electromagnéticos contínuos têm sido usados em muitos sistemas de posicionamento de alta precisão. Enquanto a velocidade de propagação do sinal é elevada, o seu alcance encontra-se limitado a 1m - 3m. Estes sinais são extremamente sensíveis à interferência do ambiente, por parte de um grande leque de fontes incluindo o campo magnético terrestre, CRTs (tubos de raios catódicos) e inclusive metais que se encontrem na

área de actuação. Portanto os sistemas baseados nestes sinais necessitam de calibração num ambiente controlado. Também são extremamente caros.

2.1.4.1.4 - Ultra-sons

Os sinais de ultra-sons estão-se a tornar cada vez mais comuns em sistemas de posicionamento. A lentidão da velocidade de propagação do som (~343m/s) permite medidas precisas com baixas taxas de relógio, fazendo com que os sistemas baseados em ultra-sons sejam relativamente simples e baratos. Os factores ambientais têm efeitos substanciais mas não proibitivos na propagação dos sinais. A humidade pode influenciar a velocidade dos ultra-sons em decrementos até 0,3%, a temperatura tem um efeito ainda mais acentuado, pois com um aumento da temperatura de 0 a 30ºC, a velocidade do som à alterado em 3%. Finalmente, os ultra-sons reflectem na maior parte das superfícies interiores.

2.1.4.1.5 - Outras tecnologias

2.1.4.1.5.1 - Óptica

Os sistemas ópticos, vão desde sistemas de medição através de laser até sistemas de vídeo montados em sistemas móveis e em paredes. Questões de segurança impedem a utilização de lasers em ambiente computacionais pervasivos. Sistemas de vídeo omnipresentes também levantam muitas questões, incluindo usos não planeados. Devido aos ruídos ambientais dentro do espectro da luz visível é exigido processos de análise com um elevado poder de processamento, elevando por isso os custos.

2.1.4.1.5.2 - Inerciais

Os sistemas inerciais usam giroscópios ortogonais e/ou acelerómetros montados em elementos móveis para medir movimento a partir de uma posição conhecida. Conceptualmente, a aceleração é integrada de modo a encontrar a velocidade e integrada novamente para encontrar a posição. Sem constantes calibrações, os erros de posicionamento não têm limites.

2.1.4.2 - Levantamento de Sistemas de Localização

Durante os últimos anos houve muita pesquisa e desenvolvimento em serviços baseados na localização (LBS). Alguns sistemas que possibilitam esta implementação encontram-se aqui apresentados.

2.1.4.3 - Active Badges

O sistema de Active Badges de Harter et al. tem a distinção de ser o primeiro sistemas de localização interior destinada à computação pervasiva [Wan92], [Har93]. Active Badges é um sistema de proximidade construído sobre uma conexão de transmissão por infravermelhos.

Figura 2.1 - Active Badges a)

Figura 2.2 - Active Badges b)

2.1.4.4 - Sistema de Localização Ad-Hoc

O sistema de localização *ad-hoc* (AHLoS) é caracterizado por uma rede grande e dinâmica de sensores pequenos, de baixo consumo e cooperantes [Sav01].

2.1.4.5 - RADAR

No design do sistema RADAR, Bahl et al. procuram aumentar o valor da comodidade de redes sem fio IEEE 802.11 [Bah00]. A base do método de localização do RADAR é realizada em duas fases. Primeiro, em uma fase *off-line*, o sistema é calibrado e um modelo contendo um número finito de localizações distribuídas sobre a área alvo é construído a partir das potências de sinais recebidos. Em segundo lugar, durante a operação on-line a correr na área alvo, as unidades móveis reportam sinais recebidos de cada estação base e então o sistema determina a melhor correspondência entre as observações on-line e de qualquer ponto do modelo *off-line*. Assim o ponto de localização que recebe a melhor correspondência é relatado como a estimativa de localização.

2.1.4.6 - Cricket

Criket calcula distâncias usando TDoA (*Time Difference Of Arrival*) de sinais RF e de ultra-sons sincronizados. Cada *beacon* emite um pulso RF único identificando o espaço onde se encontra. As unidades móveis computam a distância de propagação por cada *beacon* que detecta. Para cada *beacon*, a unidade móvel classifica as distâncias em incrementos de 10 polegadas e conta o número de sinais que detecta em cada incremento. A distância até ao *beacon* é seleccionada através do modo de distribuição de amostras sobre uma janela deslizante, correspondendo aos incrementos de distâncias. Cricket reporta então a localização da unidade móvel, como o espaço com a menor distância determinada e anunciada pelo *beacon* [Nis00].

2.1.4.7 - Active Bats

Este sistema de localização é um seguimento do sistema Active Badges [War97], [Har99]. É construído sobre o modelo Active Badges como um sistema central de seguimento,

desenvolvido através de dados obtidos pela sinalização de unidades móveis tal como Crickets. Este sistema determina a localização usando TDoA de sinais sincronizados de RF e de ultra-sons. Active Bats usa posicionamento hiperbólico para calcular as coordenadas dos elementos móveis em relação às conhecidas posições das unidades montadas em tectos.

2.1.4.8 - Outros

A *footprint* produzida por redes GSM/GPRS pode também ser a base para localização em ambientes interiores, similarmente com o que é executado em cenários WiFi, não obstante, é muito mais trabalhoso. Em geral, os sinais GSM em espaços interiores são instáveis devido a complexos mecanismos de reflexões, às variações resultantes no tempo e outras flutuações das condições de propagação [Ben05].

2.2 - Computação Orientada a Serviços

Computação Orientada a Serviços é um paradigma composto de um conjunto específico de princípios de design. A aplicação destes princípios no design na solução resulta na definição da solução lógica orientada a serviços. A mais fundamental unidade da orientação a serviços para a solução lógica é o serviço [Erl08].

Os serviços existem como programas de software independentes com distintas características que suportam a realização dos objectivos estratégicos associados com a computação orientada a serviços. A cada serviço é atribuído o seu próprio contexto, combinando-o com um conjunto de capacidades relacionadas com esse contexto. Essas capacidades formadas para invocação pelo consumidor externo ao programa são geralmente expressas através de um contrato público de serviço (tal como uma tradicional API).

2.2.1 - Elementos da Computação Orientada a Serviços

Compreender os elementos individuais da Computação Orientada a Serviços é tão importante como compreender como eles se relacionam uns com os outros, porque essas relações estabelecem alguns das mais fundamentais dinâmicas na computação orientada a serviços.

Portanto vamos ver esses elementos com ênfase na forma como cada um se vincula com os outros [Erl08]:

- Arquitectura Orientada a Serviços (***Service-Oriented Architecture***) representa uma forma distinta de concepção da arquitectura de uma tecnologia dando suporte à solução lógica orientada a serviços (***Service-Oriented Solution Logic***) que se compõe em serviços (***Services***) e composição de serviços (***Service Compositions***) moldados em conformidade com a orientação a serviços (***Service-Orientation Design Paradigm***).

- Orientação a serviços é um paradigma composto por princípios de design orientado a serviços. Quando aplicado a unidades de solução lógicas, estes princípios formam serviços com distintas características de suporte aos objectivos gerais e visão de computação orientada a serviços.

- Computação orientada a serviços representa uma nova geração de plataforma de computação que engloba o paradigma de orientação a serviços e arquitectura orientada a serviços com o derradeiro objectivo de criar e colectar um ou mais inventários de serviços (***Service Inventories***).

Essas relações são ilustradas na figura 2.3 adaptado de [Erl08].

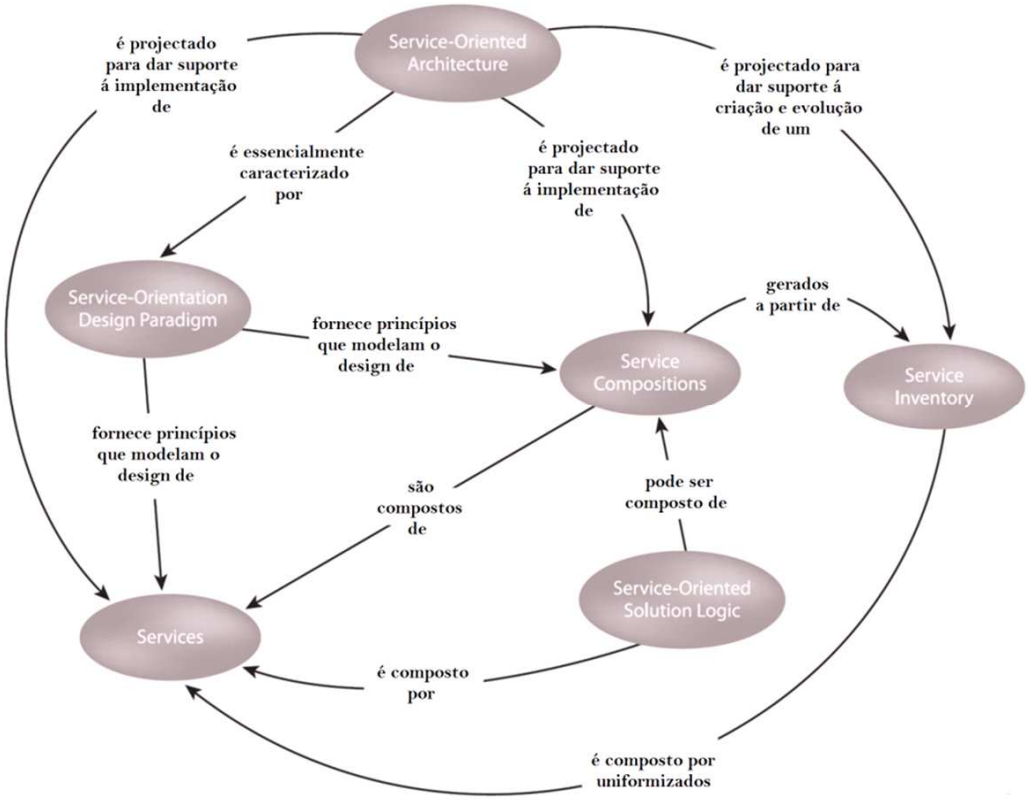

Figura 2.3 - Uma visão conceitual da forma como os elementos da computação orientada a serviços podem inter-relacionar

2.2.2 - Origens e Influências da Orientação a Serviços

Diz-se frequentemente que a melhor maneira de entender alguma coisa é obter conhecimento da sua história.

Orientação a serviços é, de forma alguma, um paradigma de design surgido do nada. É pelo contrário uma representação da evolução da tecnologia de informação (TI) e portanto tem muitas raízes nos mais antigos paradigmas e tecnologias (figura. 2.4 retirada de [Erl08]). Ao mesmo tempo, continua num estado de evolução própria, e portanto continua sobre influência das tendências e movimentos em curso.

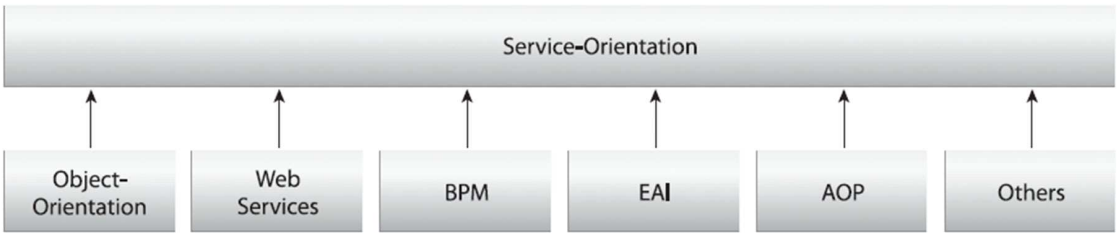

Figura 2.4 - As principais influências da orientação a serviços e também destaque às suas múltiplas origens.

Orientação a objectos (***Object-Orientation***) – os princípios e padrões detrás da análise e design orientado a objectos representa uma das mais importantes fontes de inspiração para este paradigma.

Gestão do processo de negócio (***BPM***) – a camada do processo de negócio representa uma parte fulcral de qualquer arquitectura orientada a serviços.

Serviços Web (***Web Services***) – como resultado, o Framework dos serviços Web, influenciou e promoveu diversos princípios orientados a serviços, incluindo Abstracção de Serviço, Fraco Acoplamento de Serviços, e Composição de Serviços.

Integração de Aplicações Corporativas (***EAI***) EAI introduziu middleware permitindo a abstracção de aplicações proprietárias através do uso de adaptadores, correctores, e motores de orquestração. No entanto, também ficou notório por ser extremamente complexo e dispendioso, bem como requerendo compromissos de longa duração para o vendedor da plataforma middleware e *roadmap*.

Programação orientada a aspectos (***AOP***) – Apesar de não ser um dos primordiais factores de influencia da orientação a serviços, AOP demonstra um objectivo comum enfatizando a importância de investir em unidades de solução lógica que são agnósticas para processos de negócio e aplicações e, portanto, altamente reutilizáveis. Além disso, promove o desenvolvimento baseado no papel, permitindo a colaboração entre programadores com diferentes áreas de especialização.

2.2.3 - SOA

2.2.3.1 - Definindo Melhor SOA

Note que o termo "SOA" não implica necessariamente um determinado âmbito arquitectónico. Uma SOA pode referir-se à arquitectura de uma aplicação ou à abordagem usada para padronizar a arquitectura técnica ao longo da empresa. Devido à natureza combinável de SOA (o que significa que cada arquitectura no nível da aplicação pode ser composta de diferentes extensões e tecnologias), é absolutamente possível que uma organização tenha mais de uma SOA.

Existem várias definições para o que constitui uma SOA. Umas tomam uma perspectiva técnica, outras uma perspectiva empresarial e alguns definem SOA a partir de uma perspectiva arquitectónica.

Por exemplo, o W3C (*World Wide Web Consortium*) tem uma perspectiva técnica e define SOA como "um conjunto de componentes que podem ser invocados, e cujas descrições de interface podem ser publicadas e descobertas" [W3c04].

Uma definição a partir de uma perspectiva arquitectónica é fornecida em [Mic06] onde SOA é definida como "uma arquitectura de um sistema ou aplicação construída utilizando um conjunto de serviços". Uma aplicação SOA define funcionalidades como um conjunto de serviços compartilhados e reutilizáveis.

Algumas das outras definições de SOA incluem:

Arquitectura Orientada a Serviços é uma abordagem de organizar a tecnologia de informação (TI) em que a informação, e os recursos infra-estruturais são acedidos por mensagens encaminhadas entre interfaces de rede [Mic06].

“Uma arquitectura orientada a serviços é um *framework* de aplicações que agarra os processos de negócio das aplicações e dividi-as em funções e processos de negócio individuais, chamados serviços. Uma SOA permite construir, implementar e integrar estes independentes serviços nas aplicações e plataformas em que se encontram a correr [Ibm05].

2.2.3.2 - Vantagens de SOA

È muito importante estabelecer fundamentos do porquê vendedores e comunidades de utilizadores finais dentro da industria de TI passarem pelo problema de adoptar a arquitectura orientada a serviços abarcando todas as alterações que esta comporta.

A lista de vantagens apresentada é generalizada e certamente não esta completa. Apenas é um indicador do potencial arquitectural que SOA tem para oferecer:

- **Ubiquidade**. Os serviços são acessíveis a partir de qualquer lugar e a qualquer momento.

- **Reutilização e Flexibilidade**. Tem sido um dos maiores objectivos da engenharia de software à décadas, com variáveis graus de sucesso. Em SOA, a habilidade de compor novos serviços fora do contexto dos existentes providencia possibilidade de os reutilizar e traz vantagens a uma organização que tem de ser ágil na resposta às necessidades de negócio. Nesse sentido, existe uma redução de tempo e custos e aumento de qualidade no desenvolvimento de aplicações distribuídas através da reutilização de serviços [Elf04]. [End04].
- **Manutenção**. A comunicação entre consumidores e provedores de serviços é baseado em interfaces. Isto ajuda imenso na manutenção porque são escondidos os detalhes de implementação.
- **Integração**: A maioria das organizações tem uma infra-estrutura de aplicações altamente fragmentada, na qual foram criadas um vasto número de aplicações em múltiplas plataformas de programação e infra-estruturas de comunicação de vários vendedores. Neste contexto traz muitas vantagens SOA ser um modelo de projecto baseado em interfaces, permitindo assim a integração das aplicações através de contratos entre consumidores e provedores de serviços.

2.3 - Computação Orientada a Agentes

2.3.1 - Agente e Sistema Multi-Agente

A definição de o que é um agente está longe de ser consensual, podendo encontrar-se na literatura da especialidade um grande número de definições distintas entre si. Uma das possíveis razões para tal situação prende-se com o facto do termo "agente" ser um termo usado em vastas áreas científicas. Contudo, um número crescente de investigadores encontra a seguinte definição útil [Mae96]:

"Agentes são sistemas computacionais que habitam em ambientes complexos e que, sentindo e agindo autonomamente sobre estes ambientes, realizam um conjunto de acções no sentido de atingirem os objectivos para os quais foram projectados".

Quando a visão do mundo é adoptada na perspectiva de agentes, rapidamente se torna visível que os maiores problemas requerem múltiplos agentes, para representarem não só a natureza descentralizada do problema (localizações múltiplas), como os interesses competitivos. Além do mais, os agentes necessitam de interagir uns com os outros, tanto para alcançarem os seus objectivos individuais como para gerirem as dependências de se encontrarem num ambiente comum. Um sistema multi-agente pode ser definido como uma rede de entidades desacopladas, que trabalham em conjunto para tomar decisões ou resolver

problemas que estão para além das suas capacidades individuais e/ou conhecimentos de cada entidade [Dur89]. Estas entidades são agentes autónomos e podem ser heterogéneos na sua natureza. As características dos sistemas multi-agentes são [Jen98]:

- Cada agente tem na sua posse informações ou capacidades de decisão ou de resolver problemas incompletos; cada agente tem, portanto, uma visão limitada;
- Não existe um sistema de controlo global;
- A informação encontra-se distribuída.

Duma perspectiva multi-agente, agentes em MAS são autónomos e podem envolver-se em interacções flexíveis e de alto nível. Considerando a autonomia da natureza dos agentes, autonomia significa meios que os agentes têm de possuir o seu próprio thread de controlo (e.g. são activos).

Algumas propriedades que os agentes exibem são:

- **Autonomia**. Uma vez desenvolvido o agente, capacitado com informação necessária sobre os seus objectivos, tarefas e limitações, deveria operar independentemente do seu utilizador, isto é, autonomamente em plano de fundo [Cas95] [Mae97]. Para isso, é necessário que o agente conheça o que deve fazer segundo cada caso e manter um estado interno.

- **Reactividade**. O agente pode aperceber-se do seu ambiente, interpretar as mudanças detectadas e responder adequadamente em tempo limitado.

- **Pró-actividade**. Os agentes são entidades pró-activas, isto é, tomam a iniciativa de actuar em pró dos seus objectivos e planos [Nwa96] [Woo95].

- **Habilidade Social**. Para poder actuar no seu ambiente, o agente tipicamente deve interagir com o mundo exterior. Esta interacção pode dar-se em diferentes níveis; mas, em geral, o agente necessita de comunicar-se com o ambiente, com outros agentes e com os seus utilizadores [Gen94].

A perspectiva baseada em agentes oferece um reportório de ferramentas, tecnologias e metáforas que têm um potencial de melhorar consideravelmente a maneira em que as pessoas conceituam e implementam variadíssimos tipos de software. Contudo, o desenvolvimento de complexos sistemas multi-agente requer não só novos modelos e tecnologias, mas também novas metodologias como suporte para os *designers* na abordagem da análise e *design* desses sistemas.

Enquanto a computação orientada a agentes emerge como uma poderosa tecnologia de desenvolvimento de complexos sistemas de software, apenas poucas metodologias completas e bem fundamentadas para análise e design de MAS têm sido propostas até à actual data. Assim, a prática corrente de engenharia de software orientada a agentes é discutida na secção 2.3.2.

2.3.2 – Engenharia de Software Orientado a Agentes

Existe já um grande número de publicações que abordam estes temas, tais como: o que são agentes e sistemas multi-agente, como desenvolvê-los e como implementá-los [Wss01].

Ao longo do tempo foram igualmente surgindo diversas metodologias de desenvolvimento orientadas a agentes, que em maior ou menor medida ajudam na construção destes sistemas. Uma das principais diferenças entre estas situa-se nos distintos pontos de vista utilizados no *design* do sistema.

Algumas delas, como Tropos [Gio04], [Bre04], baseiam-se na utilização de UML para modelar os sistemas baseados em agentes. Esta metodologia define quatro fases para realizar o design do sistema: análise de requisitos preliminares, análise de requisitos finais, design arquitectónico e design detalhado. Na terceira fase, a de design arquitectónico, propõem a utilização de estilos arquitectónicos para definir a organização e, consequentemente, o design da arquitectura.

Uma das primeiras metodologias orientadas a agentes que se desenvolveram foi AAII/BDI (Australian AI Institute/Beliefs, Desires & Intentions) [Kin96], que considera dois pontos de vista, um externo e outro interno. Com o ponto de vista externo identificam-se os agentes existentes, o modelo de agentes e as relações entre eles, e o modelo de interacção. O ponto de vista interno está baseado no modelo BDI para desenvolvimento de agentes [Had96].

Existem mais metodologias que consideram diferentes perspectivas para desenvolverem o sistema, como por exemplo a engenharia de "vocais" [Dem01], que considera cinco pontos de vista: Agente, Envolvência, Interacções, Organização e Utilizador (esta última foi acrescentada posteriormente). Para desenvolver cada um destes aspectos podem-se usar diferentes técnicas, e o objectivo é desenvolver bibliotecas que permitam resolver cada um destes aspectos. Mais: dependendo do sistema que se pretende desenvolver, propõe-se considerar as "vocais" em diferente ordem, dando mais relevo às "vocais" correspondentes aos aspectos primordiais do sistema. Se o mais importante é a interacção entre agentes, começa-se pela modelação da organização. Se, pelo contrário, se começar pela modelação dos agentes, a organização emergirá como resultado da interacção entre agentes individuais.

Message é uma metodologia que propõe a integração de características que aparecem noutras metodologias e resultados de investigação [Cai01], e considera cinco pontos de vista similares aos da engenharia das "vocais". A novidade radica no uso de meta modelos para descrever os aspectos relacionados com os diferentes pontos de vista.

Uma das mais citadas e usadas metodologias é Gaia [Woo00]. É uma das poucas tentativas de adaptar as necessidades da análise e design de sistemas multi-agente, e que lida tanto com o micro (intra-agente) nível como com o macro (inter-agente) nível da análise e *design*. Pretende suportar o desenvolvimento de entidades que resolvem problemas distribuídos, em que os componentes do sistema são conhecidos no período de design (e.g. sistema fechado); é esperado que os agentes cooperem no sentido de realizar a sua parte rumo ao propósito da organização. Mais detalhes da metodologia são dados aquando da apresentação do sistema proposto (secção 3.4).

Na mesma linha de Gaia desenvolveu-se MaSE [Del01], que se diferencia de Gaia na sua abordagem desde o ciclo de desenvolvimento até à construção do sistema. Contudo, MaSE está mais próxima da orientação a objectos, e considera os agentes como uma especialização dos objectos. Um dos aspectos que MaSE toma em conta são as conversas entre agentes, que se definem como protocolos de coordenação, para o qual é necessário ter definido anteriormente os papéis que se podem adaptar os agentes. Ao que acontece com as outras metodologias, MaSE deu lugar à elaboração duma ferramenta, agentTool [age09], que possibilitou a expansão rápida desta metodologia.

A metodologia ZEUS propõe o desenvolvimento do sistema em quatro fases [Col98]: a análise do domínio, o design dos agentes (que se dividem em três subfases), a realização dos agentes e o suporte em tempo de execução. ZEUS conta com bastante difusão por proporcionar, em parte, uma ferramenta de suporte às fases finais de desenvolvimento do sistema. Até surgir MaSE, ZEUS foi considerada a ferramenta de referência.

2.4 - Levantamento de Sistemas

Muitos são os sistemas que exploram o Paradigma de Computação Ubíqua. Neste capítulo são apresentados alguns desses sistemas. As suas escolhas seguiram um conjunto de critérios considerados relevantes para os objectivos da dissertação. Os principais foram não só o facto de seus autores apresentarem uma arquitectura como suporte ao esclarecimento dos respectivos sistemas, como também se ter levado em conta a quantidade de detalhes desta exposição. Além disso, estes sistemas formam um bom conjunto representativo do trabalho que tem sido desenvolvido nesta área de investigação.

Foram agrupados em temas onde cada um se destaca; existe, no entanto, uma transversalidade de âmbito em muitos deles. Assim, foram marcados como Projectos na área da Computação Ubíqua (secção 2.4.1), da Sensibilidade ao Contexto (secção 2.4.2), da Computação Orientada a Serviços (secção 2.4.3), da Computação Orientada a Agentes (secção 2.4.4) e na área da Saúde (secção 2.4.4).

2.4.1 - Sistemas na área da Computação Ubíqua

- O projecto *AURA* [Aur09], desenvolvido na *Carnegie Mellon University*, proporciona uma infra-estrutura para as aplicações se moverem facilmente dum dispositivo para outro, suportando a heterogeneidade e a variabilidade dinâmica de recursos nos ambientes inteligentes. É dividida em cinco camadas: a Camada do utilizador (camada superior) é voltada para a inclusão das aplicações. A Camada logo em baixo é composta por um componente que executa as tarefas remotas e por outro componente que dá suporte a essa

execução. A Camada de serviços (outra camada abaixo) contém um sistema de gestão de arquivos distribuídos, que suporta arquivos remotos (*Coda*), e outro que fornece serviços de sistema operacional para adaptação e monitorização de recursos (*Odisseia*). Noutra Camada ainda, encontra-se um kernel do sistema operacional, para suportar as tarefas dum sistema operacional. E a última apresenta uma rede inteligente com serviços de localização, pró-actividade, contexto e gestão de energia no protocolo.

- *EasyLiving* [Bar00] é um projecto da *Microsoft Research*, que define um conjunto de tecnologias destinadas à integração dinâmica de dispositivos, para o enriquecimento da experiência do utilizador na utilização do ambiente onde se encontra.

- Em [Tat04] são apresentados três middleware para computação ubíqua, mais precisamente para *mixed-reality*, interacção de dispositivos e computação em casa. Usa *Framework* CORBA.

2.4.2 - Sistemas na área da Sensibilidade ao Contexto

- Resultado de trabalhos desenvolvidos no *Georgia Institute of Technology*, o Context Tollkit [Gat09] tem como principais funções recolher, analisar, interpretar, transmitir e entregar informação de contexto de dados fornecidos por sensores. È composto por uma camada chamada *widget*, que recolhe dados, e por uma camada de aplicação constituída por: *Interpreters*, responsáveis por elevar o nível de abstracção duma informação contextual; *Agregators*, que colectam informações de diversas fontes de informações contextual logicamente relacionadas e as disponibilizam num repositório único; *Services*, que são componentes capazes de executar acções de acordo com o interesse e a necessidade das aplicações; e *Discoverers*, responsáveis pela manutenção dum registo das informações de todos os componentes instanciados e suas respectivas competências.

- Desenvolvida na University of Maryland, *CoBrA* [Cob09] é uma arquitectura com distribuição híbrida, composta por um componente central denominado *Context Broker*, cujas funções passam por compartilhar com os outros agentes da arquitectura uma ontologia comum para a representação das informações contextuais, realizar inferências a partir dos dados sensoriais e tratar de questões relativas à privacidade dos utilizadores e segurança do sistema. Além do seu componente central, a arquitectura *CoBrA* também incorpora outros agentes configuráveis para notificar às aplicações mudanças de contexto.

- A arquitectura *GAIA* [Rom02] foi desenvolvida para gerir um espaço activo, coordenado por uma infra-estrutura baseada em contexto, que realça a mobilidade dos utilizadores para a interacção e configuração do ambiente físico e digital. É composta por três camadas: *Aplicações*, *Framework* de aplicações e o *Kernel* de *GAIA*.

- Em [Sud07] é apresentado um *middleware framework*, que usa comunicação assíncrona, *ad-hoc*, espaço de *tuplas* e *Semantic Web*. Divide-se em quatro serviços essenciais: *Space Manager*, que organiza as unidades da memória virtual (espaço)

distribuído ao longo de várias máquinas, onde existe gestão de acesso a cada espaço; o mecanismo de *Fila de Espera*, que capacita a informação contextual do espaço para ser guardada e recuperada semanticamente – esta é aplicada em cada um dos respectivos espaços. O *Repositório de Ontologias*, o qual armazena informação contextual partilhada por todos os dispositivos e agentes de *software*. O *Reasoner*, usado para resolver dados contextuais e inferir novos conhecimentos.

- Em [Per06] propõe-se um *middleware*, *Inframware*, de suporte ao desenvolvimento e execução de aplicações móveis sensíveis ao contexto, baseado em serviços Web. A plataforma *Inframware* é constituída pelo componente *Gerente de Subscrição*, que fornece uma interface que permite aos clientes do *Inframware* removerem, adicionarem ou actualizarem pedidos de subscrições. O módulo de *Controlo de Acesso e Privacidade* actua como um filtro sobre o fluxo de dados entre a plataforma e as aplicações, com base num conjunto de restrições, envolvendo preferências e políticas de privacidade dos utilizadores e das aplicações. O *Interpretador de Contexto*, que é responsável pela manipulação, derivação, refinamento e inferência de contextos mais elaborados a partir de informações contextuais primitivas. O *Acesso e Integração de Dados* provê funcionalidades capazes de tratar e manipular informações oriundas de diversas fontes de contexto, oferecendo também uma interface homogénea e transparente de acesso aos dados. O *Gerente de Serviços*, que é responsável pelas actividades de publicação, descoberta, selecção e composição de serviços. E finalmente o *Coordenador*, cujas principais funções estão relacionadas com a monitorização e o controlo do estado geral da plataforma.

- Em [Soo05] foi proposto um novo modelo de componente, baseado em adaptação de contexto para *middleware* sensível ao contexto. Divide-se em: *Configuration Manager*, que recebe informação de configuração dos sensores e das configurações dos utilizadores; *Adaptation Manager*, que decide o comportamento do processo usando informação de contexto; *Execution Manager*, que mantém a informação dos comportamentos necessários para executar serviços; e *Communication Manager*, o responsável pela troca de mensagens XML com o objectivo de guardar ou entregar metadados.

- *SOCAM*, *Service-Oriented Context-Aware Middleware* [GuT05], desenvolvida na *Computing School of Singapure National University*, utiliza um modelo formal de contexto com o objectivo de prover prototipagem rápida de serviços e aplicações sensíveis ao contexto em ambientes de computação pervasiva. As informações contextuais são descritas numa linguagem de representação de conhecimento (*OWL - Ontology Web Language*), permitindo, com isso, a patilha de conhecimento contextual entre diferentes entidades e a realização de inferências sobre essas informações.

2.4.3 - Sistemas na área da Computação Orientada a Serviços

- Em [Sal08] é apresentada uma arquitectura de *middleware*, orientada a serviços, desenvolvida na *University of the Basques Coutry*, uma extensão da *Jini Network Technology*, explorando a tecnologia Java. É constituída por: *Control Resources*, que inclui todos os dispositivos de rede do ambiente controláveis; *Context Resources*, formada pelos sensores e outras fontes de informação ambiental da rede; *Interaction Resources*, que permite a interacção de pessoas através de *feedback* ou comandos que permitam a conclusão dos seus objectivos; e pelo *Ontology Manager*, com função de gestão da base de dados de ontologias, importando dados ontológicos dos drivers, e criando uma primeira instância do *Ontology Reasoner*, de modo a permitir um comportamento inteligente e interacção proactiva dos aplicativos com os utilizadores.

- *WASP - Web Architectures for Service Platforms*, desenvolvida na *University of Twente*, Holanda, [Doc03] –a arquitectura da plataforma tem como característica particular o uso da tecnologia de distribuição de *Web Services*, e é executada sobre uma rede móvel *3G*.

2.4.4 - Sistemas na área da Computação Orientada a Agentes

- Ichiro [Ich03] propõe um *framework* para aplicações sensíveis ao contexto, implementada por agentes móveis. Cada agente móvel pode ser executado em qualquer computador próximo do utilizador, apenas tendo de permanecer neste o tempo suficiente para realizar o serviço. O *Framework* divide-se em duas partes: *Mobile Agents* e *Location Information Servers* (*LISs*). Cada *LIS* gere múltiplos sensores que detectam as presenças de tags e mantém informação actualizada acerca destas.

- Em [Kha04] propõe-se uma arquitectura baseada em agentes. Agente executado em *PDA* do utilizador. Faz uso de vários registos: *User profile*, onde são armazenadas as propriedades e preferências pessoais do utilizador, podendo também usar políticas de adaptação e incluir informação de autorização e autenticação de utilizador; *Content Profile*, que contém meta-informação, ou seja, os dados têm tipo e formato com características e parâmetros que podem ser usados para os descrever; *Context Profile*, que possui qualquer informação que pode ser usada para caracterizar a situação ou contexto; *Device Profile*, com capacidades e características dos dispositivos, que podem incluir características do *hardware*, tipo de dispositivo, o tamanho do *display*, as capacidades de entrada/saída, o sistema operativo e os códigos de áudio e vídeo; *Network Profile*, que apresenta as características da rede.
Os agentes do sistema referem-se ao *Communication Agent* (*CA*), responsável pelo intercâmbio de comunicação entre agentes. Este contém: o *Content Profile* próprio assim como de intermediários ao longo do caminho; o *Service Discovery Agent* (*SDA*), que busca serviços na área pessoal da rede sem fios; o *User Context Agent* (*UCA*), responsável por providenciar o perfil de contexto do utilizador ao *QoS Selection and Negotiation Agent*

(*QSNA*); e o *QSNA*, que selecciona os melhores serviços que satisfazem os requerimentos dentro das preferências do utilizador.

- Em [Wan08] é apresentada a arquitectura dum sistema multi-agente, providenciando a integração rápida de dispositivos físicos de rede. Construída sobre o *Java Agent DEvolopment Framework* (*JADE*), divide-se em 3 grupos de agentes: *Interface Agents*, responsáveis por transmitirem eventos de actualizações do sistema e receber acções de controlo dos utilizadores; *Learning Agents*, que envolvem diferentes tipos de algoritmos de aprendizagem, tais como *fuzzy inference*, *neural network*, *decision tree*; e, por fim, *Control/Utility agents*, que controlam pontos no protocolo *UPnP*, passam mensagens de controlo aos correspondentes dispositivos software *UPnP* e estão à escuta dos eventos daqueles que estão registados.

- Chun-Dong et al. [Don07] apresentam um sistema sensível ao contexto, usando múltiplos agentes para processar a informação, com o *Context Collecting Agent* (*CCA*), o *Context Reasoning Agent* (*CRA*) e o *Information Processing Agent* (*IPA*). O *CCA* recolhe informações de eventos relacionados com o utilizador de vários dispositivos e apresenta-os ao *CRA*, o qual tem armazenado os perfis e gera os correspondentes comandos que são enviados para o *IPA*, que traduz estes comandos para acções, mensagens ou algo mais complexo.

2.4.5 - Sistemas na área da Saúde

- Em [Son07] é apresentado um *middleware* sensível ao contexto para aplicações várias e monitorização de pacientes. Dividindo-se em *Context Information Event Module*, realiza uma sincronização com os dispositivos de informação baseada em eventos, de modo a eliminar envio de informação desnecessária. O *Pattern Generation Module*, que classifica informação de contexto em não regular e regular usada para gerar um novo padrão. O *Tag Generation Module* classifica informação com e sem padrão. O *Quality of Service of Contex Information* realiza *feedback* entre dispositivos móveis, de modo que as informações *tag* e padrões possam ser alterados.

- Em [Cac06] é apresentado um mecanismo (algoritmo) de descoberta semântica de serviços, referindo a sua relevância no contexto organizacional onde os serviços *e-health* são usados, nomeadamente em emergências médicas e onde agentes interagem socialmente e comunicativamente.

- Em [Cab06] é apresentado *UbiMedic*, um *middleware* baseado em agentes; também é proposto um *framework* para implementação e desenvolvimento de serviços, como monitorização e comunicação. È usada a distribuição *LEAP* da plataforma *JADE*.
O framework é formado por dois níveis: o mais baixo contém vários módulos - *Autenticação*, que verifica credenciais das entidades que tentam aceder ao sistema, aceita pedidos, se quem os pede tem as credenciais correctas, as quais são mantidas numa base de dados (DB), associada a cada entidade; - *Autorizador*, onde existem permissões e regras guardadas

numa DB de acordo com o modelo *Role-based access control* (*RBAC*); As *Políticas*, operações pré-definidas que uma entidade pode realizar em resposta a um disparo de algum evento no ambiente; o *Descobridor*, que monitoriza em tempo real a disponibilidade de recursos e as entidades presentes no sistema; o *Monitor de Ambiente*, que lida com os aspectos físicos de contexto, localização dos dispositivos, capacidades computacionais, acesso a rede e acontecimentos de eventos externos; o *Gestor de Eventos*, que recolhe notificações de alterações de contexto recebido do nível da aplicação e de outros módulos do *middleware* e despacha-os para os objectos interessados.
Já os serviços de alto nível contêm o *Gestor de Contexto*, que gere a informação de contexto de cada entidade e define uma estrutura para manter a informação actualizada. Divide-se em dois módulos: o *Session Factory*, que gera os objectos necessários para as entidades acederem ao sistema (regras, permissões, perfis, políticas relacionadas com a entidade; este atribui às entidades um agente particular, *ContexAgent*, o qual representa a entidade dentro do sistema, mantendo-se actualizado através do *Gestor de Eventos*). E *Enviroment Factory*, o segundo módulo, que recolhe toda informação de contexto físico, gerando um objecto de contexto relacionado com a localização particular onde a entidade se executa. O Request Manager é outro serviço de alto nível, que recebe e tenta satisfazer os pedidos das entidades para o acesso aos recursos e funcionalidades. Cada pedido é analisado e executado mediante o estado do contexto físico (largura de banda e memória disponível) e informação de sessão (permissões e perfis).

- Em [Lup07] é proposta a *Self-Managed Cell* (*SMC*), um padrão arquitectural para aplicações ubíquas na Saúde, na qual os sensores corporais são usados para monitorizar pacientes nas suas próprias casas. È constituído por: *Descobridor de Serviços*, responsável por manter a filiação na *SMC* e informar outros serviços quando um dispositivo se junta ou deixa esta (este *Descobridor* tem também controlo de admissão baseado no perfil do dispositivo e qualquer informação de autenticação disponível); O *Barramento de Eventos*, que dispara as políticas de adaptação do comportamento do *SMC*; e o *Serviço de Gestão*, que mantém objectos de adaptação para cada um dos componentes que podem ser alvo de acções de gestão.

- Em [Pav04] mostra-se uma estrutura para o acesso a informação médica utilizando um *PDA* dentro e fora dum ambiente hospitalar, usando para tal uma tecnologia sem fios, redes móveis 802.11 e GSM/CDMA.

- Em [Kim07] é sugerida a possibilidade da interconexão entre *JADE* e o *Open Service Gateway initiative* (*OSGi*) para um sistema móvel de Cuidados de Saúde.

- Em [Han06] propõe-se um sistema construído em cima da plataforma *LEAP* (Lightweight Extensible Agent Platform), que garante a ligação duma *BAN* (*body area network*) e o subsistema do Hospital através da participação de agentes de software.

- Em [Fon05] apresenta-se um projecto académico, ainda no começo, dum sistema multi-agente para informação médica, que possibilita a pesquisa de informação médica

distribuída, não tendo por base uma partilha de bases de dados, mas sim uma negociação da informação, caso a caso, entre os diversos agentes.

Capítulo 3

Proposta

Este capítulo expõe a proposta de um sistema de informação para a saúde baseado nos paradigmas revistos nos capítulos anteriores, isto é, Computação Ubíqua, Computação Orientada a Serviços e Computação Orientada a Agentes

Na secção 3.1 é abordada uma possível união entre os vários paradigmas. Na secção 3.2 são apresentados os problemas detectados nos sistemas de informação na área da Saúde. Na secção 3.3 é dada a conhecer a proposta em si, apresentando os seus objectivos e requisitos, tendo por vista ajudar a solucionar alguns dos problemas apontados na secção anterior. Na secção 3.4 são realizadas as fases de *Análise* e *Design* do sistema segundo a metodologia Gaia. Finalmente, na secção 3.5 são apresentados casos de uso da manutenção do Sistema.

3.1 - União entre Paradigmas

Actualmente, as atenções estão voltadas para as vantagens estratégicas de utilizar infra-estruturas computacionais distribuídas, projectadas para serem dinâmicas, flexíveis, adaptáveis e inter-operáveis. O surgimento de arquitecturas orientadas a serviços colocam novas demandas em arquitecturas de software [Buh03] [Gri03] porque necessitam dar suporte a uma multiplicidade de sistemas externos, serviços e dispositivos que interagem e colaboram. Estas características têm impulsionado a demanda de aplicações baseadas em agentes.

Sistemas Multi-Agentes (MAS) possuem propriedades como autonomia e pró-actividade, que os tornam atractivos para aplicações distribuídas. Independentemente da metodologia, modelo e arquitectura adoptada, as abordagens orientadas a agentes providenciam tais conceitos de alto nível, tal como a noção de actividade, objectivo, tarefa, interacção por mensagens, que podem ser facilmente reconhecidos a nível do negócio.

Agentes podem usar e beneficiar das mais recentes tecnologias e padrões abertos, tais como SOA. De acordo com Meltzer [Mel98], Glushko [Glu99], Griss [Gri03], XML, usada em Web Services, tende a tornar-se na linguagem padrão para a interacção orientada a agentes, para codificar trocas de mensagens, documentos, facturas, ordens, descrição de serviços e

outras informações. Em parte justifica-se porque para além da sua simplicidade, clareza e concentrar-se na interoperabilidade, actualmente o tráfego global da rede é em grande percentagem composta por XML, tal como se verifica na figura 3.1 retirado de [Gee05], e é esperado o seu aumento nos anos porvir devido ao aumento da popularidade das tecnologias que assentam em XML.

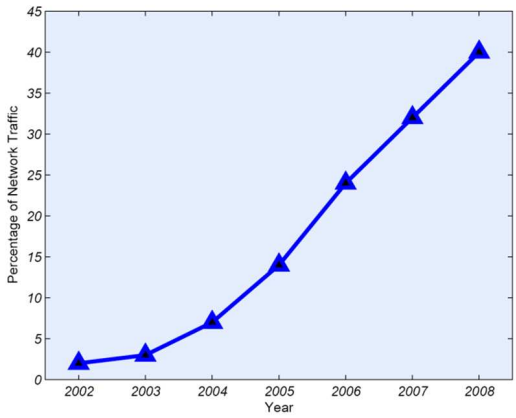

Figura 3.1 - Estimativa da percentagem de XML no tráfego global da rede

Considerando estes aspectos, acredita-se que o desenvolvimento duma plataforma MAS como uma camada de middleware inteligente, a partir de uma arquitectura orientada a serviços, para ambientes inteligentes (*AmI*), é uma mudança relevante. Combinando estas abordagens, espera-se desenvolver um sistema, no qual se têm em conta as actividades dos utilizadores e o seu contexto, para se adaptar e reconfigurar num ambiente tão dinâmico como o que se vive em Hospitais e Clínicas de Saúde.

3.2 - O Problema

Para a concretização deste trabalho fomos conhecer as reais condições de vivência e de trabalho em Hospitais, bem como os problemas com que se debatem, quer os técnicos de Saúde e funcionários de administração, quer os utentes em geral, nomeadamente os pacientes.

Assim, observámos os actos da prestação de cuidados médicos em alguns hospitais públicos portugueses. Realizámos entrevistas a médicos, enfermeiros, técnicos, administrativos e auxiliares, bem como a utentes desses serviços, no sentido de perceber quais os factores que levam a atrasos, erros e mal-estar no uso dos sistemas de informação implantados nos ambientes hospitalares.

Deste modo analisamos alguns problemas:

- A actividade clínica desenvolvida na prática diária pelos Serviços de Saúde está fortemente dependente de complexos sistemas informáticos.

- O número de interfaces que medem o acesso aos Sistemas de Informação é reduzido em número, favorecendo a formação de filas de espera por parte dos profissionais que necessitam do acesso para continuar a exercer o seu cargo.

- Essas interfaces, além de insuficientes, estão frequentemente posicionadas de forma pouco correcta, e obrigam os utilizadores a terem de afastarem-se dos locais onde contactam o doente para outro local onde se encontra a interface de acesso, criando, quanto ao tempo, uma grave descontinuidade no trabalho que estavam a realizar.

- E por estas interfaces serem fixas, leva os profissionais de Saúde a inevitáveis abandonos momentâneos dos seus doentes.

- A identificação de pacientes, habitualmente, é uma tarefa árdua, pois as pulseiras, escritas ou com códigos de barras, para serem lidas, torna-se complicado, pela difícil acessibilidade e/ou pela sua deterioração.

- O material existente nas instalações, consumíveis e equipamentos clínicos, de constante utilização, é de gestão difícil dada a clássica monitorização em uso.

- A causa apontada como a mais relevante é a capacidade de interagir com o sistema, obstáculo esse principalmente devido à sua complexidade, em volume e características.

- A acessibilidade aos Serviços de Informação, tão úteis e hoje já imprescindíveis em variadas situações, é quase sempre penosa.

- A adaptação a estes sistemas e a sua utilização metódica, por parte dos profissionais de saúde, seus utilizadores, é frequentemente uma tarefa morosa; e, um grande motivo para a sua desaprovação é o facto de se proceder repetitivamente a alterações programáticas e à revelia dos seus utilizadores.

Após esta análise, concluímos:

- É vasto e bastante complexo o ambiente hospitalar.

- A prestação de cuidados de saúde processa-se habitualmente ao longo de um labirinto de instalações, mas necessariamente de forma linear e muito precisa. Observe-se a figura 3.2, onde é apresentada uma planta da zona de um dos pisos de um Hospital português, identificando algumas das suas divisões e funções.

- Os equipamentos e os consumíveis clínicos, que são muitos e variados, deverão estar sempre disponíveis e operacionais.

- Os recursos humanos, profissionais de saúde dotados de formação especificamente diferenciada, deverão actuar sempre de forma concertada, rápida e complementar.

- Os utentes dos serviços que são o objectivo final de todo este puzzle interventivo, enfermos na saúde e/ou na dor, anseiam por cuidados céleres e acertados.

- E todos, actuando ao momento certo, no lugar certo da cadeia de serviços e de forma concertada, conseguirão o melhor resultado.

- Para esta sintonia, os Sistemas de Informação poder-se-ão constituir cada vez mais uma ferramenta essencial e indispensável.

- No entretanto, hoje em dia, ainda os sistemas distraem o utilizador, logo reduzindo a sua eficiência.

- A generalidade das Arquitecturas e Sistemas de Informação em Saúde, em implementação, carecem ainda de mecanismos, rápidos e eficazes, de entrega dos serviços necessários, e que sejam relacionados específica e cabalmente no contexto em que se inserem. A especificidade da prática médica e nomeadamente as características humanas dos seus profissionais são pouco atendidas.

- Estas Tecnologias de Informação (TI) pouco relevam a mobilidade dos seus utilizadores.

- A forma de identificação de pacientes e de gestão de material, actualmente em difusão, não é suficientemente ágil para o dinâmico ambiente Hospitalar.

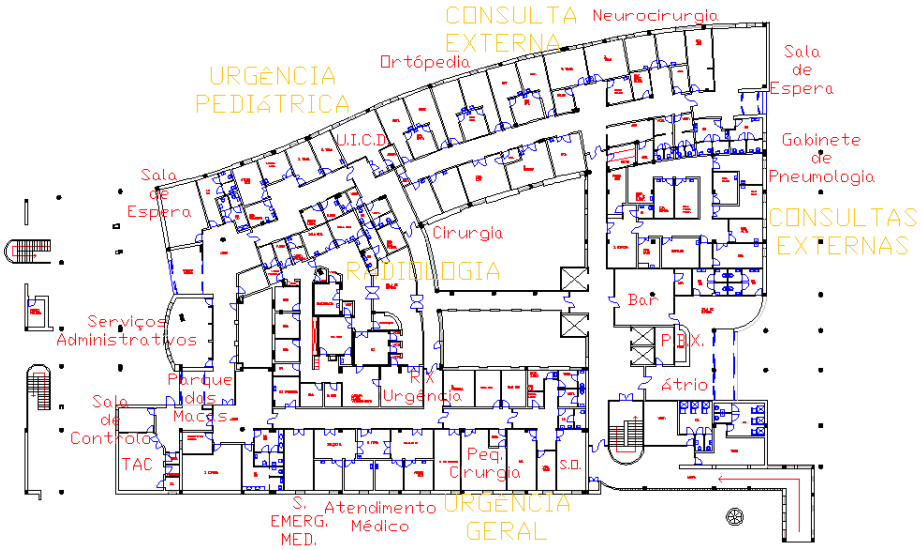

Figura 3.2 - Planta parcial de um hospital português e seus serviços

3.3 – A Proposta

3.3.1 - Visão Geral

A Plataforma apresentada tem como objectivo ajudar a colmatar várias lacunas referidas acima, nomeadamente tentar fornecer aos Sistemas de Informação, a utilizar pelos profissionais de Saúde, uma forma rápida e suficientemente contextualizada de fornecimento de informação acerca dos Serviços disponíveis, e tendo a plataforma como base, a mobilidade dos seus utilizadores, no labirinto das instalações hospitalares, a complexidade da actuação clínica e o reconhecimento rápido do utilizador.

Esta solução foi pensada para ser utilizada em aplicativos e dispositivos ubíquos, mas tal não impede o seu uso noutras plataformas mais tradicionais; tal dependerá da forma como os aplicativos/serviços foram projectados.

3.3.2 – Objectivos

- ✓ A proposta centra-se em desenvolver um middleware capaz de proporcionar a aplicativos um mecanismo automático e dinâmico de entrega de Serviços, baseado em informações de contexto relevantes - a relação do indivíduo com o local onde se encontra, principalmente acelerando e facilitando os processos de prestação de Cuidados de Saúde. Assim, sempre que um determinado utilizador mude de local, ser-lhe-ão descobertos e apresentados os Serviços que mais se ajustem nesse contexto.

- ✓ Pretende-se que o sistema favoreça a portabilidade dos Sistemas de Informação para plataformas ubíquas, mas que, ao mesmo tempo, permita o seu uso nas implementações já existentes.

- ✓ O sistema fará uso de tecnologia de identificação por rádio frequência, sem fornecer uma localização precisa, colherá os benefícios que tal tecnologia poderá revelar num ambiente tão complexo como o hospitalar.

3.3.3 – Requisitos

3.3.3.1 - Características dos utilizadores

Os utilizadores serão todos os indivíduos que frequentem o ambiente onde se instala o sistema, mais precisamente os profissionais de Saúde, os pacientes, os funcionários de administração e os visitantes. O uso da plataforma dependerá da forma como esta é utilizada pelos aplicativos, antevendo-se muito poucos ou mesmo nenhuns conhecimentos informáticos por partes dos seus utilizadores.

3.3.3.2 – Limitações

Não se pretende implementar nenhum nível de segurança.

O automatismo total da descoberta de Serviços apenas é possível se a área onde se encontra o utilizador for coberta pelo Sistema de Detecção.

Os leitores de Rfid devem apresentar a função de medição de potência do sinal recebido.

3.3.3.3 - Requisitos não funcionais

A Tecnologia RFID:

- a tag deve conseguir guardar informação de identificação;
- o leitor deve ter uma área de cobertura> 14m2;
- o sistema RFID deve usar um protocolo de comunicação simples;
- o sistema RFID deve conseguir fornecer a potência de leitura do sinal recebido;

O Sistema deve ser capaz de:

- requer informações das leituras ao leitor RFID;
- extrair os dados necessários dos campos da trama enviada pelo Leitor;
- formatar os dados, integrando-os no sistema;
- relacionar os dados recebidos com o local preciso onde foi detectada a tag;
- inferir o deslocamento da tag, através dum registo temporário;
- associar a leitura de uma nova tag do indivíduo a um evento;
- relacionar os dados do evento a um utilizador;
- obter informação (dados de contexto) acerca dos locais, indivíduos e serviços, através de registos de carácter permanente;
- permitir a alteração das informações dos registos de carácter permanente;
- cruzar os dados de contexto com os registos dos serviços;
- seleccionar e estruturar as informações dos serviços;

3.4 - Descoberta Dinâmica de Serviços na Perspectiva de Sistema Multi-Agente

Descreve-se a seguir o projecto numa perspectiva de MAS, recorrendo à metodologia Gaia descrita em [Woo00]. É também apresentada uma breve descrição das várias etapas desta metodologia, ao longo de duas grandes fases: *Análise* e *Design*.

Quando do desenvolvimento da metodologia Gaia (*Gaia*) os seus autores [Woo00] acreditavam que o uso de agentes representava um avanço na abstracção de funcionamento de sistemas, assim os desenvolvedores de software faziam e fazem o seu uso para mais naturalmente entender, modelar e desenvolver uma importante classe de sistemas distribuídos complexos.

É intenção de GAIA ser apropriado ao desenvolvimento de sistemas de larga escala de aplicações do mundo real, com as seguintes características:

- O objectivo dos agentes deve ser o de maximizar o beneficio global do sistema ainda que isto implique minimizar o próprio beneficio do agente. Gaia não esta pensado para sistemas em que existam conflitos reais entre os agentes.
- As relações entre agentes são estáticas e não se alteram em tempo de execução.
- As habilidades e os serviços dos agentes também são estáticas e não mudam em tempo de execução.
- Gaia não esta pensada para sistemas de mais de cem tipos de agentes distintos.

Gaia tenciona permitir o analista a ir sistematicamente desde uma declaração de requisitos para o desenho suficientemente detalhado que pode ser implementado directamente.

Note-se que a fase de captura de requisitos é independente do paradigma usado na *análise* e *design*.

Providencia um conjunto de conceitos específicos a agentes em que a engenharia de software pode usar para compreender e modelar um sistema complexo. Em particular, Gaia encoraja um projectista a pensar na construção de um sistema baseado em agentes como um processo de *design* organizacional.

3.4.1 - Fase de Análise

O objectivo da fase de *análise* é entender o sistema e a sua estrutura (sem referenciar detalhes de implementação). Neste caso, este entendimento é capturado na organização do sistema.

A entidade mais abstracta neste conceito é o sistema. Este é usado no seu sentido mais *standard*; contudo, está também relacionado com o significado de sociedade ou organização, quando se fala de sistema baseado em agentes. Deste modo, pensamos no sistema multi-agente como uma sociedade ou organização artificial.

O processo de análise de Gaia é composto pelo modelo de papéis, onde se identificam os papéis principais do sistema, e o modelo de interacção, onde se definem as relações entre os vários papéis numa organização multi-agente.

3.4.1.1 - Identificação dos papéis do sistema

Identificam-se seis papéis essenciais para o bom funcionamento do sistema. O *IntegratorLocative* tem como principal função fazer uso de dispositivos de localização por rádio frequência de modo a localizar as entidades físicas no espaço real. O *Displacer* tem memória dos locais por onde as entidades reais se acha, inferindo, através disso, o acto de deslocação. O *Informer*, com a sua base de conhecimento, é capaz de prenunciar acerca das entidades físicas e do local onde estas se encontram. O *Discoverer*, tendo acesso às informações dos Serviços passíveis de ser prestados, consegue seleccionar os mais apropriados

ao contexto actual. O *Consignor* acumula os Serviços seleccionados numa lista ordenada pela sua relevância e transmite-a à entidade apropriada. O *UserWizard* tende a representar o utilizador no sistema.

Tabela 3.1 - Identificação dos papéis principais do sistema

Papéis	Descrição informal
IntegratorLocative	Questiona os leitores de RFID e integra os dados das tramas provenientes dos leitores, de forma a possibilitar o seu tratamento, e apurando o local preciso onde se encontram as entidades físicas reais.
Displacer	Averigua se as entidades físicas reais se deslocam.
Informer	Fornece informações de contexto relevantes, nomeadamente acerca do tipo de papel do indivíduo e da função do local, no contexto da Clínica.
Discoverer	Estabelece relações apropriadas entre os serviços da Clínica e o contexto actual do indivíduo, e selecciona os serviços que mais se adaptam a este.
Consignor	Obtém os serviços seleccionados e envia-os à entidade competente.
UserWizard	Tende a representar os Utilizadores no sistema.

3.4.1.2 - Modelo de interacção

Inevitavelmente, existem dependências e relações entre os vários papéis na organização multi-agente. Este modelo é definido como um conjunto de definições de protocolos, para cada interacção de papel. Pode ser visto como um padrão institucionalizado de interacção, ou seja, um padrão de interacção que é formalmente definido, suficientemente abstracto e afastado de qualquer sequência particular de etapas de execução.

Nesta perspectiva, a atenção é focada essencialmente na natureza e no propósito da interacção, em vez de ordenar uma sequência precisa de troca de mensagens.

A definição de protocolo consiste nos seguintes atributos:

Propósito:	Breve descrição da natureza da interacção;
Originador:	O(s) papel(eis) responsável(eis) por iniciar a interacção;
Receptor:	O papel que interactua com o originador;
Entradas:	Informação usada pelo papel do originador enquanto promulgação do protocolo;
Outputs:	Informação fornecida pelo/para o receptor do protocolo durante a interacção;
Processamento:	breve descrição textual de qualquer processamento fornecido pelo originador do protocolo durante a interacção.

De seguida, descrevem-se brevemente os protocolos principais na definição do protótipo.

No protocolo *InicarDeslocamentos*, o *IntegratorLocative* (Originador) interage com papel *Displacer* (Receptor), tendo como finalidade informá-lo da detecção de uma *tag* no espaço controlado; note-se que o *IntegratorLocative* inicia esta interacção após ter processado as informações contidas na trama referente à t*ag* detectada; seguidamente o *IntegratorLocative* (***IL***) entrega ao *Displacer* (***DP***) os detalhes da Tag

Tabela 3.2 - Protocolo IniciarDeslocamentos

IL	***DP***	*processamentoTag*
Passa a informação obtida para ser processada pelo *Displacer*		*detalhesTag*

No protocolo *IniciarExtracçãoConhecimento*, o *Displacer* (Originador) interage com o papel de *Informer* (***IF***) (Receptor), solicitando a realização da sua função; isto acontece sempre que o *Displacer* entende que determinada entidade se deslocou, passando ao *Informer* detalhes do evento em questão.

Tabela 3.3 - Protocolo IniciarExtracçãoConhecimento

DP	***IF***	*eventoDeslocamentoTag*
Passa informação do evento referente ao deslocamento da entidade física		*detalhesEvento*

No protocolo *IniciarSelecção*, o *Informer* (Originador) interage com o *Discoverer* (***DC***) (Receptor), entregando-lhe informações de contexto do evento ocorrido; esta interacção sucede quando o Informer tem conhecimento do *local* e *indivíduo*, identificados nos detalhes do evento.

Tabela 3.4 - Protocolo IniciarSelecção

IF	***DC***	*informaçãoDisponível*
Passa informações de contexto do evento ocorrido para o ***DC*** iniciar a sua função		*contextoInf*

Após o *Discoverer* (Originador) seleccionar um dado serviço, passa as informações deste para o *Consignor* (***CS***) (Receptor), para armazená-los e lhes dar uma ordem de relevância – Protocolo *IniciarDistribuição*.

Tabela 3.5 - Protocolo IniciarDistribuição

DC	***CS***	*serviçoSeleccionado*
Passa as informações relativas aos serviços que se adequam ao contexto de modo que o *CS* crie uma lista ordenada do conjunto de serviços		*descriçãoServiço*

No protocolo *IniciarVisualização*, o *Consignor* interage com o papel de *UserWizard* (***UW***), disponibilizando-lhe a lista dos serviços sempre que finaliza a acumulação dos serviços seleccionados.

Tabela 3.6 - Protocolo IniciarVisualização

CS	***UW***	*serviçosSeleccionados*
Envia a lista ao **UW** de modo a habilitá-lo a apresentar ao utilizador a maneira que melhor lhe convém		*listaServiços*

3.4.1.3 - Elaboração do modelo de papéis

No conceito de hierarquia temos os ***papéis***, definidos por quatro atributos: *responsabilidades*, *permissões*, *actividades*, e *protocolos*.

Responsabilidades determinam a funcionalidade; é talvez o atributo principal associado ao papel. Estão divididas em duas categorias: *propriedades de execução* e *propriedades de segurança*. *Propriedades de execução* descrevem os estados das coisas que os agentes devem

trazer, dando certas condições ao ambiente. Em contraste, as *propriedades de segurança* são invariáveis; o estado das coisas aceitável é mantido ao longo de todas as etapas de execução.

As ***permissões/direitos*** associados a cada papel são a informação ou conhecimento que o agente tem, isto é: o agente para desempenhar o papel necessita de aceder, modificar ou gerar certas informações.

As ***actividades*** de um papel são computações associadas com o papel que podem ser realizadas pelo agente sem qualquer interacção com outros agentes. São, portanto, acções privadas.

Finalmente, um papel pode ser identificado por certos ***protocolos***, os quais definem a maneira de interacção com outros papéis.

A forma de representar as responsabilidades dá-se mediante a combinação de protocolos e actividades (estas últimas, sublinhadas) usando os seguintes operadores:

- *x.y* *y* vem em continuação de *x*
- *x|y* ocorre *x* ou *y*
- *x** *x* ocorre 0 ou mais vezes
- *x+* *x* ocorre 1 ou mais vezes
- x^w *x* ocorre infinitamente
- *[x]* *x* é opcional
- *x||y* *x* e *y* são concorrentes

Tabela 3.7 – Modelo do Papel *IntegratorLocative* (***IL***)

O papel tem como função questionar os leitores de RFID e integrar os dados das tramas provenientes dos leitores de forma a possibilitar o seu tratamento, apurando o local preciso onde as entidades físicas reais se encontram.
Protocolos e *Actividades*:
IniciarDeslocamentos; *QuestionarLeitor*, *ReceberTrama*, *ProcessarTrama*, *ApurarLocal*
Permissões:
Leitura: *tramaLeitor* // lê tramas dos leitores, do buffer da //porta série
Definição Local // potência+porta = local
Gerar: *detalhesCompletosTag* //estrutura de dados com as informações // relevantes da tag recebida
TramaTipoC // trama enviada para leitores ordenando // resposta com informações de leitura tags
→

Responsabilidades:
Execução:
IntegratorLocative = (*QuestionarLeitor*. *ReceberTrama*. *ProcessarTrama*.
.*ApurarLocal*. *IniciarDeslocamentos*)^{w}

Segurança: *trama* ≠ *null*
tramaRecibida = *válida*
(*portaTramaRecebida*) & (*potênciaRecebida* > definida) → *Local_ID*

Tabela 3.8 – Modelo do Papel *Displacer* (***DP***)

O papel tem como função averiguar se as entidades físicas reais se deslocam.

Protocolos e *Actividades*:

IniciarDeslocamentos, *IniciarExtracçãoConhecimento*; *InferirDeslocação*

Permissões:
Leitura: *detalhesCompletosTag*

Gera: *detalhesEvento* // evento de deslocação da entidade

Responsabilidades:
Execução:
Displacer = (*IniciarDeslocamentos*.*InferirDeslocação*.
.*IniciarExtracçãoConhecimento*)^{w}
Segurança:
detalhesCompletosTag = *válidos*
eventoDeslocação ≠ *null*

Tabela 3.9 – Modelo do Papel Informer (***IF***)

Fornece informações relevantes acerca do tipo de papel do indivíduo e da função do local, no contexto da Clínica.

Protocolos e *Actividades*:

IniciarExtracçãoConhecimento, *IniciarSelecção*; *PesquisarBaseConhecimento*, *ExtrairConhecimento*, *FormatarInformações*

Permissões:

Leitura:	*detalhesEvento*	
	informaçãoDisponível	// a base de conhecimento (*BC*)
		// acerca dos divíduos e locais
Gerar:	*contextoInf*	// as informações relevantes
		// retiradas da *BC*

Responsabilidades:

Execução:

Informer = (*IniciarExtracçãoConhecimento.Conhecimento.Selecção*)^{w}

Conhecimento = (*PesquisarBaseConhecimento*.*ExtrairConhecimento*)

Selecção = (*FormatarInformações*.*IniciarSelecção*)

Segurança:

detalhesEvento = válidos

informaçãoDisponível = acessível

contextoInf ≠ null //tem de existir informação de //conhecimento associado ao contexto

Tabela 3.10 – Modelo do Papel *Discoverer* (*DC*)

Estabelece relações apropriadas entre os Serviços da Clínica e o contexto actual do indivíduo, seleccionando os Serviços que mais se adaptam a este.

Protocolos e *Actividades*:

IniciarSelecção, *IniciarDistribuição*; *RelacionarInfContextoServiços*, *SeleccionarServiços*, *ExtrairDescriçõesServiços*

Permissões:

Leitura: *contextoInf*
registoServiços

Gera: *serviçoSeleccionado*
descriçãoServiço

Responsabilidades:

Execução:

Discoverer = (*IniciarSelecção.EscolherServiço.NotificarSelecção*)^{w}
EscolherServiço = (*RelacionarInfContextoServiços.SeleccionarServiços*)$^{+}$
NotificarSelecção = (*ExtrairDescriçõesServiços.IniciarDistribuição*)

Segurança:

contextoInf ≠ *null*
registoServiços = *acessíveis*
nº de serviçoSeleccionado ≥ *0*

Tabela 3.11 – Modelo do Papel *Consignor* (*CS*)

Obtém os serviços seleccionados, agrupando-os em níveis de especificidade, e envia-os à entidade competente.
Protocolos e *Actividades*: *IniciarDistribuição*, *IniciarVisualização*; *CriarListaServiços*
Permissões: Leitura: *serviçoSeleccionado* Gera: *listaServiços* // agrupamentos de serviços pela sua relevância
Responsabilidades: Execução: *Consignor* = (*IniciarDistribuição*.*CriarListaServiços*.*IniciarVisualização*)^{w} Segurança: *serviçoSeleccionado* ≠ *null* // existe pelo menos um serviço

Tabela 3.12 – Modelo do Papel *UserWizard* (***UW***)

Armazena as listas de serviços associadas a cada utilizador e associa-se à aplicação de alto nível que tem como função listar os serviços seleccionados e o executar dos mesmos.
Protocolos e *Actividades*: *IniciarVisualização*; *ExibirListaServiços*, *ExecutarServiços*
Permissões: Leitura: *listaServiços*
Responsabilidades: Execução: *UserWizard* = (*IniciarVisualização*.*ExibirListaServiços*.*ExecutarServiços*)^{w} Segurança: *listaServiços* = *válida* // devera conter os campos necessários, e.g. url

3.4.2 - Fase de Design

A intenção do processo clássico é a de transformar os modelos abstractos, criados durante a fase de análise, em modelos mais concretos, para que possam ser facilmente implementados. Contudo, este não é o caso do projecto orientado a agentes. A finalidade de Gaia é baixar o nível de abstracção dos modelos de análise, para que as técnicas tradicionais de projecto (incluindo técnicas orientada a objectos) possam ser aplicadas no sentido de implementar os agentes: Gaia preocupa-se com a forma como a sociedade de agentes coopera, para realizar os objectivos do sistema, e com os requisitos necessários de cada agente individual, de modo a realizá-los.

São três os modelos identificados: *Modelo de Agentes*, que identifica os tipos de agente que compõem o sistema e os agentes instanciados a partir destes tipos; o *Modelo de Serviços*, que identifica os serviços principais requeridos para realizar o papel dos tipos de agentes; e, por fim, o *Modelo de Familiaridade*, que documenta a comunicação entre os diferentes agentes.

3.4.2.1 - Modelo de agente

Um tipo de agente é melhor definido a partir de um conjunto de papéis. Uma forma de criar eficiência e optimização no projecto é agregar papéis num só tipo de agente.

O modelo de agente é definido usando uma simples árvore de tipos em que cada folha é um papel, correspondendo os nós aos tipos de agente. Levam-se em conta as seguintes anotações:

Tabela 3.13 - Qualificação das instâncias no modelo de agente

Qualificadores	**Significado**
N	existem n instâncias em tempo de execução
n..m	existem entre n e m instâncias em tempo de execução
*	existem zero ou mais instâncias em tempo de execução
+	existem uma ou mais instâncias em tempo de execução

Não existe inerência nesta metodologia, pois os agentes são sistemas de alta granularidade, tipicamente contendo um número comparativamente pequeno de papéis e tipos –um para um, na maior parte das vezes. (É óbvio que, quando se chega à implementação, a inerência pode ser de grande eficiência, maneira clássica de programação orientada a objectos (**OO**)).

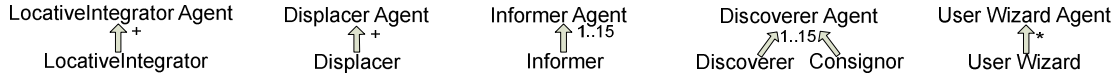

Figura 3.3 - Modelo de agente

Na figura podemos observar um esquema dos tipos de agentes que formam o protótipo e as instâncias que fazem parte de cada agente. Como se vê, o protótipo estaria formado por cinco agentes.

3.4.2.2 - Modelo de Serviços

Em termos de OO, um serviço corresponde a um método; contudo, tal não quer dizer que um serviço esteja disponível para outro agente da mesma maneira que os métodos de objectos estão disponíveis para invocação por parte de outros objectos.

Cada actividade identificada na etapa de análise corresponde a um serviço, mas nem todos os serviços correspondem a uma actividade.

Para cada serviço executado pelo agente é necessário documentar as suas propriedades, especialmente as suas entradas, saídas, pré-condições, e pós-condições. Pré e Pós condições representam moderações dos serviços. Os serviços que um agente executa são derivados da lista de protocolos, actividades, responsabilidades, e propriedades de segurança de um papel.

O modelo de serviços que identifica os serviços principais, requeridos para realizar os papéis dos agentes, é:

Tabela 3.14 - Modelo de serviços do papel de *IntegratorLocative*

Serviços	Entradas	Saídas	PréCondições	PósCondições
AdquirirTrama	*tramaTipoC*	*TramaRecebida*	*trama ≠ null*	*tramaRecibida = válida*
Integrar Dados	*trama*	*detalhesTag*	*tramaRecibida = válida*	*detalhesTag ≠ null*
Inferir Local	*potênciaTag portaTrama*	*IdLocal*	*DefiniçãoLocal*	*Verdadeiro*
Comunicação de detecção	*detalhesTag IdLocal*	*detalhesCompletosTag*	*Verdadeiro*	*Verdadeiro*

A junção das actividades *QuestionarLeitor* (Deve exigir aos leitores o envio das informações lidas das tags detectadas) e *ReceberTrama* (Deve conseguir receber as tramas de forma correcta) é equivalente ao serviço "***Adquirir Trama***", em que a entrada é o pedido de aquisição e a saída a trama propriamente dita. Tem como pré-condição a existência duma trama, e pós-condição a TramaRecebida ser válida.

A actividade *ProcessarTrama* (Deve conseguir descodificar os campos das tramas, ou seja, saber o que cada um significa) refere-se ao serviço "***Integrar Dados***", que tem por finalidade integrar os dados das tramas provenientes dos leitores, de forma a possibilitar o seu tratamento; a sua entrada são as tramas e a sua saída são os campos, de que a trama é composta, respectivamente identificados. Tem como pré-condição a validade da trama, e pós-condição os campos da trama identificados.

A actividade *InferirLocal* (Deve interpretar os campos de modo a relacioná-los com o local onde a tag se encontra) é equivalente ao serviço "***Inferir Local***", que tem como função apurar o local preciso onde as entidades físicas reais se encontram; tem como entrada a potência recebida da tag e a porta de comunicação donde foi lida a trama, e como saída o local onde se encontra a tag. Tem como pré-condição a definição dos locais, e não tem pós-condição.

O protocolo *IniciarDeslocamentos* (Deve comunicar ao agente Displacer a detecção da tag) corresponde ao serviço "***Comunicação de detecção***"; tem como entrada os dados lidos da trama e o IdLocal, e como saído o envio dos dados completos da detecção. Não tem pré nem pós condições.

Tabela 3.15 - Modelo de serviços do papel de *Displacer*

Serviços	Entradas	Saídas	PréCondições	PósCondições
Inferência de Deslocação	*detalhesCompletosTag*	*eventoDeslocação actBaseConhecimento*	*detalhesCompletosTag = válidos*	*localActualEntidade*
Iniciar da Extracção de Conhecimento	*eventoDeslocação*	*detalhesEvento*	*eventoDeslocação ≠ null*	*verdadeiro*

A junção da actividade *InferirDeslocação* (Depreende que uma entidade se terá deslocado) com o protocolo *IniciarDeslocamentos* (Recebe os dados completos da detecção) é equivalente ao serviço "***Inferência de Deslocação***"; tem como entradas a recepção dos dados completos da detecção, e como saída o evento de deslocação (caso aconteça) e a actualização da base de conhecimento. A pré-condição é que os dados recebidos sejam válidos, a pós-condição a noção da localização actual da entidade.

O serviço associado ao protocolo *IniciarExtracçãoConhecimento* (Deve comunicar ao papel Informer o evento de deslocamento) é "***Iniciar da Extracção de Conhecimento***"; tem como entrada os dados do evento de deslocamento, e como saída o envio dos dados relevantes do evento. A pré-condição é que tenha acontecido uma deslocação; não tem pós-condição.

Tabela 3.16 - Modelo de serviços do papel de Informer

Serviços	Entradas	Saídas	PréCondições	PósCondições
Extracção de Conhecimento	*detalhesEvento*	*informaçãoDisponível*	*informaçãoDisponível = acessível*	*Verdadeiro*
Disponibilizar Informações de contexto	*informaçãoDisponível*	*contextoInf*	*informaçãoDisponível ≠ null*	*contextoInf ≠ null*

O serviço associado ao protocolo *IniciarExtracçãoConhecimento* e às actividades *PesquisarBaseConhecimento*, e *ExtrairConhecimento* denomina-se "***Extracção de Conhecimento***"; tem como entradas os dados do evento de deslocação, e como saída informações do papel da entidade e informações acerca do local para onde de deslocou. A pré-condição é a da base de conhecimento estar disponível; não tem pós-condição.

O serviço associado à actividade *FormatarInformações* e o protocolo *IniciarSelecção* é "***Disponibilizar Informações de contexto***"; as entradas são as informações disponíveis, e as saídas o envio da informação de contexto. A pré-condição é a existência de informações acerca dos intervenientes no evento, a pós-condição a das informações de contexto estarem completas.

Tabela 3.17 - Modelo de serviços do papel de *Discoverer*

Serviços	Entradas	Saídas	PréCondições	PósCondições
Seleccionar serviços	*contextoInf*	*serviçoSeleccionado*	*contextoInf ≠ null registosServiços = acessíveis*	*serviçoSeleccionado ≠ null*
Extracção dos dados dos serviços	*serviçoSeleccionado*	*descriçãoServiço*	*serviçoSeleccionado ≠ null*	*verdadeiro*

Do protocolo *IniciarSelecção* e das actividades *RelacionarInfContextoServiços* e *SeleccionarServiços*, pode-se derivar o serviço "***Seleccionar serviços***"; tem como entradas os dados de contexto, e como saídas a selecção dos serviços que melhor se adaptam ao contexto; as pré-condições são o acesso aos registos dos serviços e a existência de informação de contexto, a pós-condição é a noção dos serviços que se apropriam ao contexto.

Da união da actividade *ExtrairDescriçõesServiços* e do protocolo *IniciarDistribuição* tem-se o serviço "***Extracção dos dados do serviço***"; tem como entrada a selecção dos serviços e como saída as suas descrições; tem como pré-condição a existência de serviços seleccionados.

Tabela 3.18 - Modelo de serviços do papel de *Consignor*

Serviços	Entradas	Saídas	PréCondições	PósCondições
Distribuir Lista Serviços	*ServiçoSelecionado descriçãoServiço*	*listaServiços*	*Nº serviçosSeleccionados ≥ 1*	*verdadeiro*

Dos protocolos *IniciarDistribuição* e *IniciarVisualização*, juntamente com a actividade *CriarListaServiços*, obtém-se o serviço "***Distribuir Lista Serviços***"; tem como entradas os serviços seleccionados juntamente com as suas descrições, e como saídas o envio de uma lista destes serviços para entidade respectiva. As pré-condições são a existência de pelo menos um serviço.

Tabela 3.19 - Modelo de serviços do papel do *UserWizard*

Serviços	Entradas	Saídas	PréCondições	PósCondições
Assistência ao Utilizador	*listaServiços*	*ApresentaçãoServiços* *ExecuçãoServiço*	*listaServiços = válida* *Nº descriçãoServiço ≥ 1*	*verdadeiro*

Do protocolo *IniciarVisualização* e das actividades *ExibirListaServiços*, e *ExecutarServiços*, pode-se derivar o serviço "***Assistência ao Utilizador***"; tem como entradas a lista de serviços seleccionados para o utilizador, e como saída a sua apresentação e execução. Tem como pré-condição a existência e validade de pelo menos um serviço com a respectiva descrição.

3.4.2.3 - Modelo de familiaridade

Apenas indica os caminhos de ligação de comunicação existentes entre tipos de agentes. Em particular, o propósito é identificar os potenciais engarrafamentos de comunicação, os quais causam problemas na hora de execução.

É simplesmente um gráfico, com os nós a corresponderem a tipos de agentes e os arcos a caminhos de comunicação.

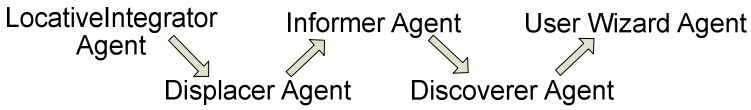

Figura 3.4 - Modelo de familiaridade

3.5 - Casos de Uso na Gestão do Sistema

Através de uma interface gráfica deve ser possível configurar o agente LocativeIntegrator. As configurações englobam as comunicações assíncronas com o agente Displacer, através do porto definido. Deve-se passar ao agente conhecimentos acerca da identificação da área coberta pelos leitores RFID; deste modo, aquando da adição duma nova comunicação por porta série associada a um leitor, o administrador deve conseguir configurá-la facilmente e associar-lhe uma identificação de local (figura 3.5).

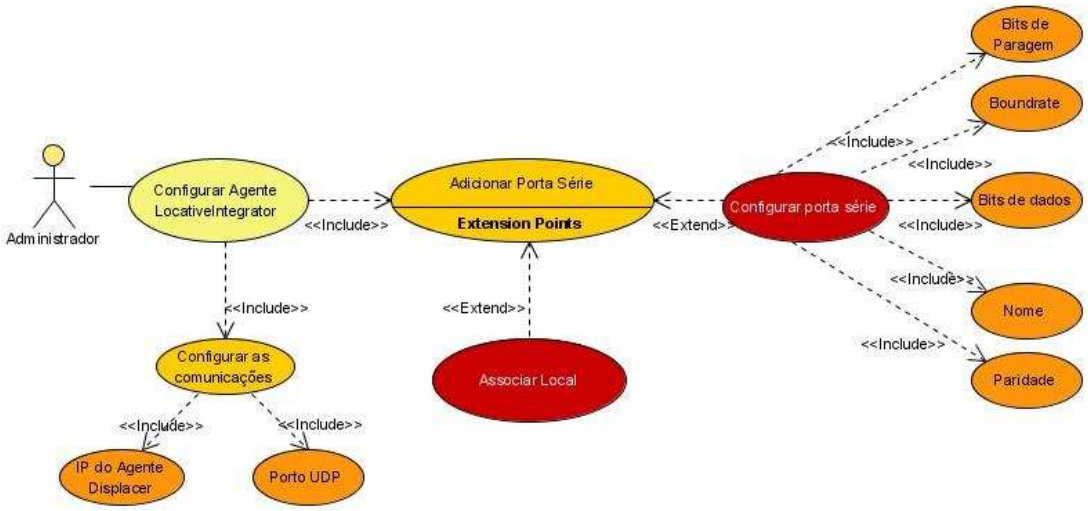

Figura 3.5 - Caso de uso do agente *LocativeIntegrator*

Uma vez que o agente Displacer deve ser capaz de inferir o acto de deslocamento de uma tag, o administrador deve conseguir configurar o tempo em que uma tag permanece sem ser detectada em determinado local, antes de inferir que esta já não se encontra lá. As configurações das comunicações também ficam a cargo do administrador. Este terá como responsabilidade configurá-las antes de executar o agente, isto é, deve informar o Agente da localização do Informer e seleccionar ambos os portos de comunicação: o de interacção, com o agente LocativeIntegrator, e o de interacção, com o agente Informer (figura 3.6).

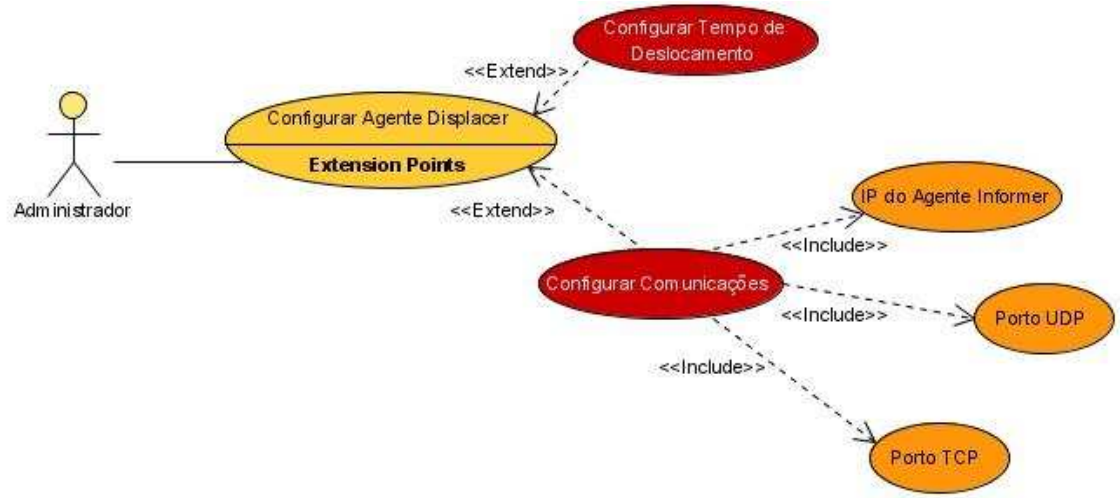

Figura 3.6 - Caso de uso do agente *Displacer*

No caso do agente *Informer*, o Administrador deve, para além de lhe configurar as comunicações com os agentes *Discoverer* e *Displacer*, ambas síncronas, configurar também o acesso à Base de Conhecimento (locais e indivíduos) (figura 3.7).

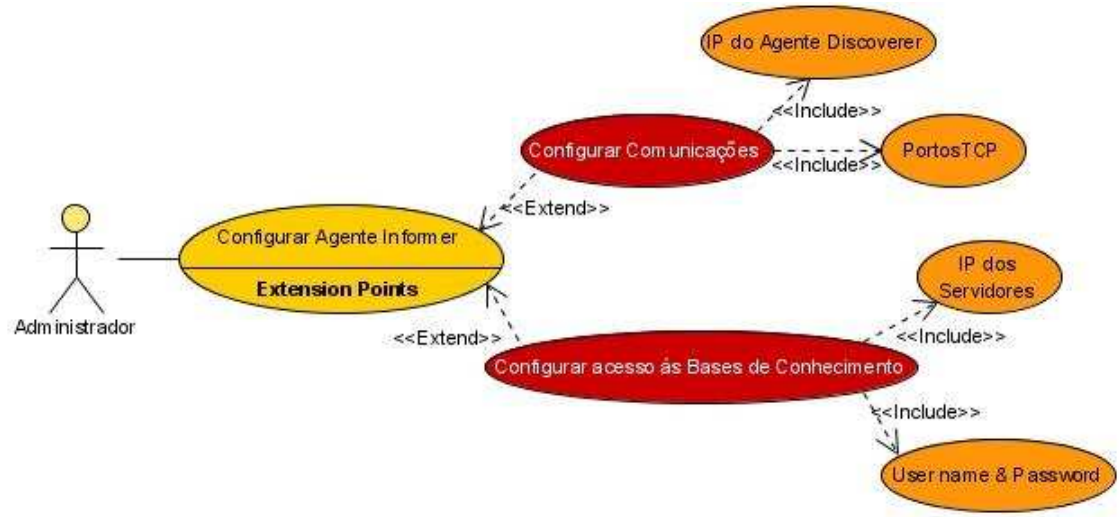

Figura 3.7 - Caso de uso do agente *Informer*

O caso de uso do Agente *Discoverer* é muito semelhante à do Agente *Informer*, apenas se alterando os agentes com quem deve interagir e os dados que permitem o acesso ao Repositório das descrições dos Serviços (figura 3.8).

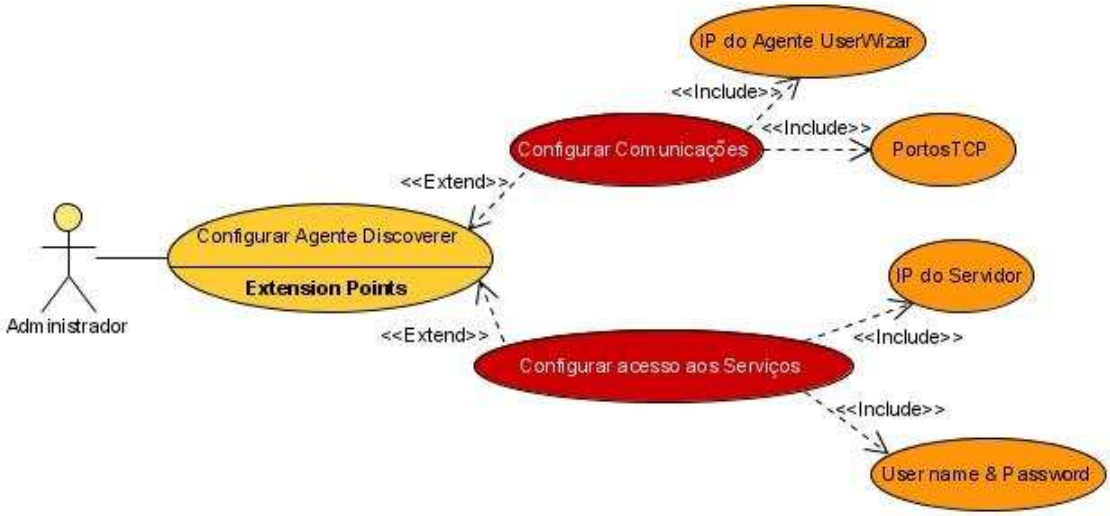

Figura 3.8 - Caso de uso do agente *Discoverer*

O agente *UserWizard* deve ser capaz de armazenar os serviços seleccionados, para serem visualizados pelos utilizadores; assim, o administrador passa-lhe a informação (caminho) da localização na máquina onde este corre, que deve ser usada para esse fim. O porto usado para interacção com o agente *Discoverer* também deve ser definido (figura 3.9).

Figura 3.9 - Caso de uso do agente *UserWizard*

As funções básicas que o administrador pode executar quanto às várias bases de dados estão representadas neste diagrama de casos de uso, do qual, por ser bastante explícito, se segue apenas uma brevíssima descrição: logo que as bases de dados sejam criadas, estas estarão acessíveis a partir do aplicativo após configurações essenciais, isto é, o endereço IP

do servidor e a autenticação correcta. De seguida, as operações de gestão das bases de dados estão acessíveis e prontas a ser usadas (figura 3.10).

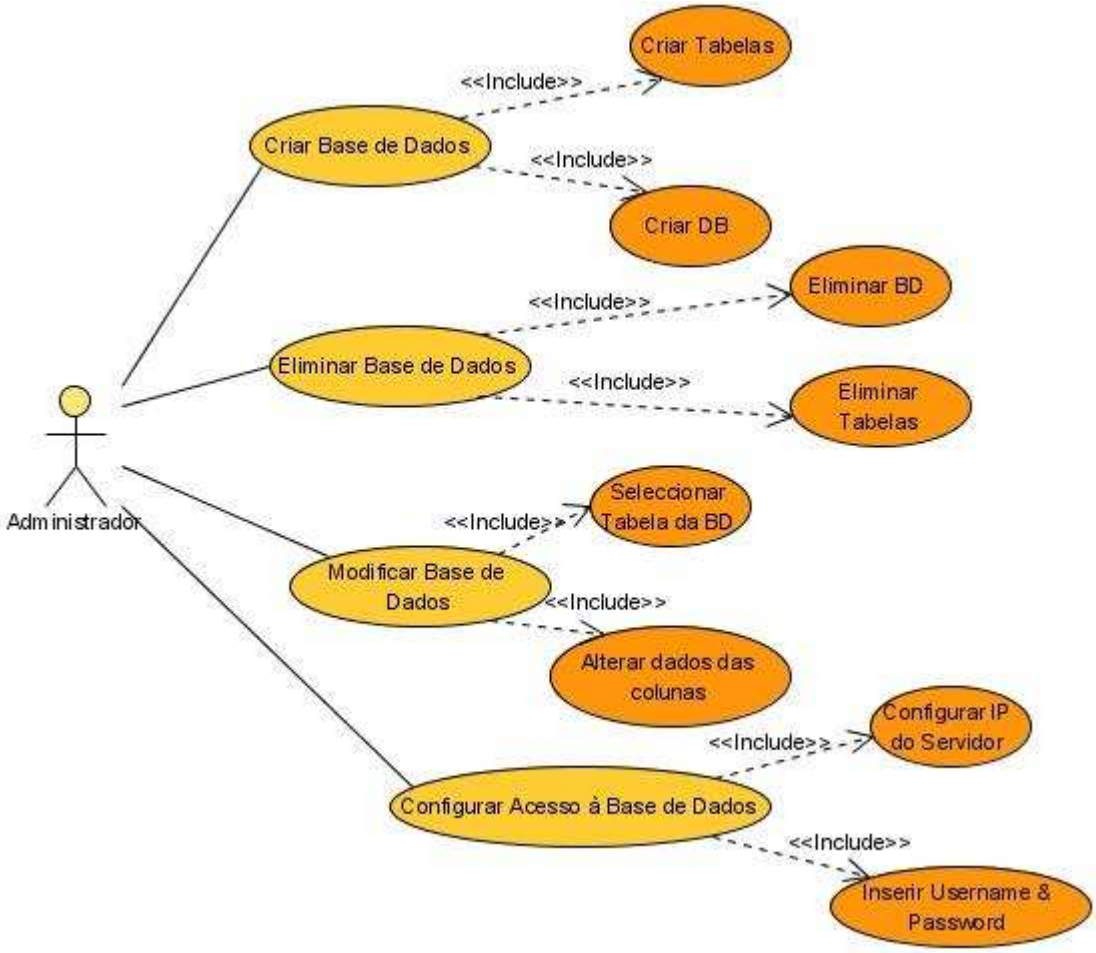

Figura 3.10 - Casos de uso do *Gestor de Base de Dados*

Capítulo 4

Implementação

Neste capítulo são descritos os detalhes da implementação do protótipo Sistema Multi-Agente especificado no capítulo anterior. Em primeiro lugar é mostrado o Sistema como um todo, secção 4.1. De seguida são apresentados os detalhes de cada um dos Agentes, secção 4.2. As interfaces gráficas do *Gestor de Bases de Dados* e seu uso são exibidos na secção 4.3. Por fim, na secção 4.4, expõe-se uma das simulações efectuadas para provar as potencialidades do sistema, tendo sido criado com esse propósito um aplicativo *JSP* acessível por um navegador Web a correr num *PDA*.

4.1 - O Sistema Multi-Agente

Nesta secção são descritos aspectos da fase de implementação e instalação, designadamente a estrutura e dependências de código fonte e de módulos executáveis, assim como a sua respectiva instalação nas diferentes plataformas computacionais subjacentes.

4.1.1 - Arquitectura Lógica

No desenvolvimento do sistema Multi-Agente, recorreu-se à linguagem de programação orientada a objectos, Java. Foram usados alguns pacotes de bibliotecas externas à distribuição do API do Java, relativamente às comunicações série, [Rxt09] - distribuído sobre licença GNU LGPL (Lesser General Public License), tal como na serialização dos objectos *XML*, [Xst09] - distribuído sobre licença *BSD* (*Berkeley Software Distribution*) usando como parser o pacote xpp3 [Xst09], para as comunicações entre agentes.

Para melhor se entenderem os pacotes de bibliotecas usadas são apresentadas na forma gráfica, figura 4.1, estas relações.

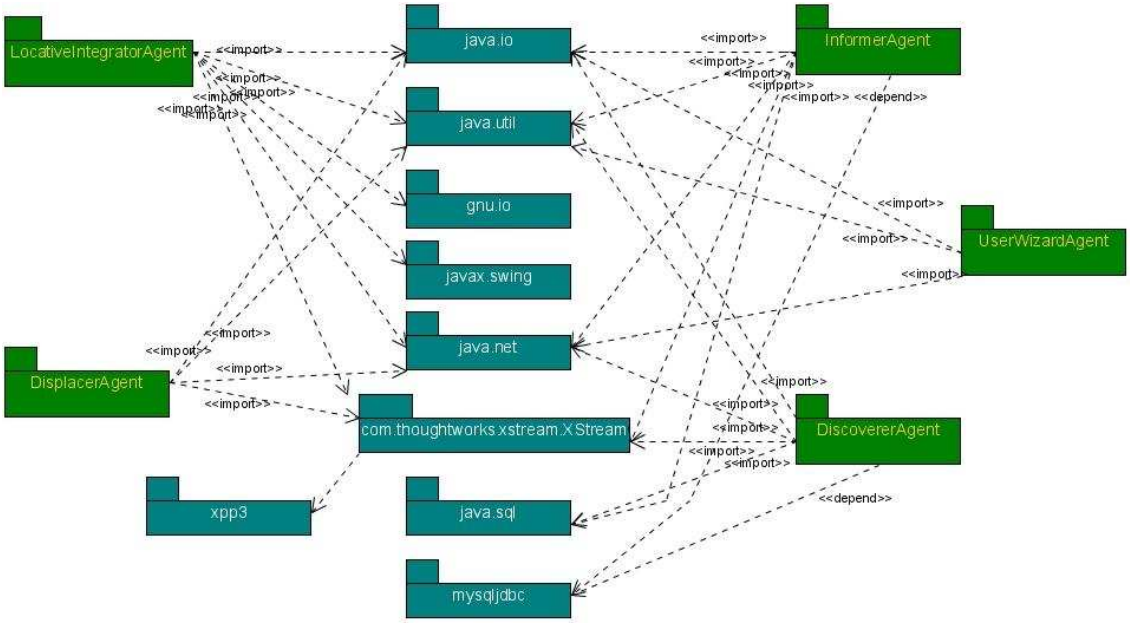

Figura 4.1 - Diagrama das dependência dos agentes com os pacotes de bibliotecas usados nas suas implementações

Maior detalhe da arquitectura lógica dos agentes, ver secção 4.2.

4.1.2 - Arquitectura Física

O diagrama de distribuição de componentes apresentado na figura 2.2, reproduz a configuração dos nós de processamento e os seus componentes em tempo real. Pretende mostrar como foi modelada a arquitectura do sistema de software na perspectiva dos seus componentes digitais, explicitando principalmente as suas múltiplas dependências, e ainda identificar quais os componentes que são instalados em cada nó computacional.

Pode-se observar que cada um dos agentes encontra-se num nó individual e que todos eles necessitam do *Java Run Enviroment* (*JRE*) de modo a serem executados.

O facto dos agentes se encontrarem em nós separados entre si não inviabiliza a solução de que todos ou parte deles habitem a mesma máquina computacional.

O agente *IntegratorLocative* encontra-se ligado ao leitor RFID e necessita do seu ficheiro de configuração '*config.ini*'.

O agente *Informer* e *Discoverer* necessitam do driver JDBC para se ligar à base de dados Hospital (tabelas: Indivíduos e Locais) e base de dados Serviços (tabela: Serviços) respectivamente.

O agente *UserWizard* produz os ficheiros '*IDuser.sev*' que serão armazenados numa directoria própria. Estes ficheiros são os usados pelos aplicativos que pretendam dinamizar a entrega dos seus serviços.

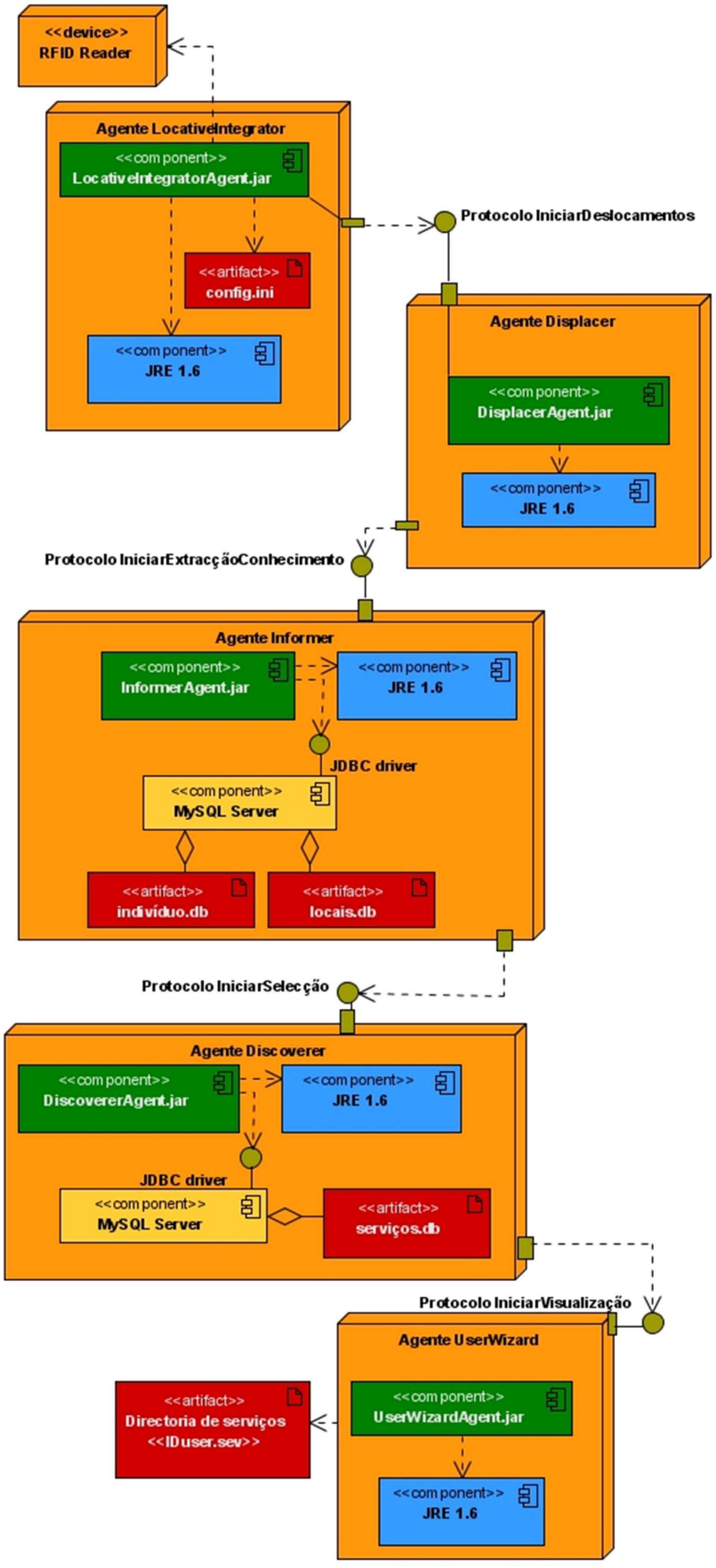

Figura 4.2 - Diagrama Distribuição de Componentes do Sistema Multi-Agente

Os protocolos estabelecidos entre os agentes são definidos pelas mensagens trocadas entre eles por interfaces bem definidas em *XML*. De seguida, são apresentadas possíveis mensagens de comunicação entre agentes, baseadas nos respectivos protocolos.

As mensagens são transmitidas sincronamente, *TCP* sobre *IP*, garantindo a entrega dos pacotes sem erros. A excepção são as mensagens baseadas no protocolo *IniciarDeslocamentos*, transmitidas assincronamente, UDP sobre *IP*.

O Protocolo IniciarDeslocamentos

```
<MESSAGE>
  <FROM>IntegratorLocativeAgent</FROM>
  <TO>DisplacerAgent</TO>
  <TAG>0001000107362014</TAG>
  <LOCAL>005</LOCAL>
</MESSAGE>
```

Protocolo IniciarExtracçãoConhecimento

```
<MESSAGE>
  <FROM> DisplacerAgent </FROM>
  <TO>InformerAgent</TO>
  <TAG>0001000107362014</TAG>
  <LOCAL>005</LOCAL>
</MESSAGE>
```

Protocolo IniciarSelecção

```
<MESSAGE>
 <FROM>InformerAgent</FROM>
 <TO>DiscovererAgent</TO>
 <ID_individuo>007</ID_individuo>
  <tag_individuo>1000107362019</tag_individuo>
  <nome_individuo>Salgado</nome_individuo>
  <cargo_intitucional_individuo>Profissional Saúde</cargo_intitucional_individuo>
  <cargo_principal_individuo>Técnico de Imagem</cargo_principal_individuo>
  <cargo_especifico_individuo>Especialista Radiologista</cargo_especifico_individuo>
  <outros_individuo>Utilizador</outros_individuo>
 <ID_local>005</ID_local>
  <descricao_local>Sala de Espera</descricao_local>
  <tipo_local>Sala Espera</tipo_local>
  <departamento_local>Radioterapia e Imagem</departamento_local>
  <clinica_local>Clínica</clinica_local>
</MESSAGE>
```

Protocolo IniciarVisualição

```
<MESSAGE>
 <FROM>DiscovererAgent</FROM>
 <TO>UserWizardAgent</TO>
 <ID>007</ID>
 <Service>
  <NAME>Serviços de Paciente</NAME>
  <DESCRIPTION>Informações dos pacientes sob responsabilidade</DESCRIPTION>
  <URL>http://symptomnav.adam.com/content.aspx?productId=44</URL>
  <PRIORITY>8</PRIORITY>
 </Service>
 <Service>
  <NAME>Serviços de Controlo Infecções</NAME>
  <DESCRIPTION>Orientação para reduzir ao máximo os índices de infecção</DESCRIPTION>
  <URL>http://symptomnav.adam.com/content.aspx?productId=44</URL>
  <PRIORITY>16</PRIORITY>
 </Service>
</MESSAGE>
```

Caso não for possível a comunicação imediata entre os Agentes, estes apenas tentarão interagir um certo número de vezes, após o qual descartam a mensagem que pretendiam transmitir.

4.2. - Os Agentes

Devido ao uso da linguagem Java, foi preciso estabelecer uma ligação entre detalhes do projecto obtidos de Gaia e a linguagem de programação usada. De modo a existir uma maior apreensão, as descrições dos agentes são acompanhadas pelo mapeamento entre protocolos e actividades obtidos por Gaia e as classes que os implementam.

4.2.1 - Agente IntegratorLocative

4.2.1.1 - Descrição

O agente *IntegratorLocative* é o Agente que está em contacto permanente com o mundo real. A percepção desse ambiente é realizada através de sensores RFID.

Tem por objectivos os serviços de integração dos dados provenientes dos sensores RFID - de forma a possibilitar o seu tratamento-, e de apurar o local preciso onde se encontram as entidades físicas reais.

Ao fazer parte duma organização de agentes, o resultado dos seus serviços devem ser encaminhados para o agente Displacer, baseando-se no protocolo *InicarDeslocamentos*, com o fim de informá-lo da detecção de uma tag num local específico.

4.2.1.2 - Configuração

As configurações necessárias do Agente *IntegratorLocative* são realizadas pelo Administrador antes da sua execução. Esta configuração é realizada através da interface gráfica apresentada na figura 4.3.

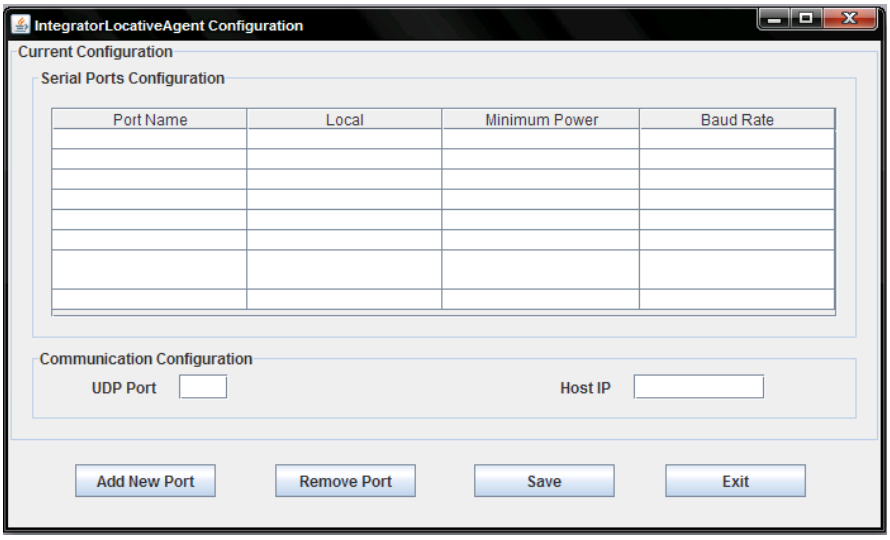

Figura 4.3 - Interface de configuração do agente *IntegratorLocative*

As configurações englobam a identificação e definição de locais. Estas configurações estão directamente relacionadas com as portas série da máquina onde é executado o Agente, e associadas a leitores RFID e potências mínimas captadas por estes aquando da detecção de uma tag. A selecção das tags a serem tratadas são apenas as com um valor *RSSi* (*Indicador de potência de sinal recebido*) superior ao definido. Esta *RSS* (*Potência de sinal recebido*) mínima está relacionada com as atenuações que o espaço livre submete o sinal RF da tag até ao leitor, permitindo delinear zonas no espaço coberto pelos leitores.

Foi acrescentada uma limitação à definição de local – um leitor apenas pode definir um local. Esta restrição deveu-se à constatação, durante a fase de testes com o leitor escolhido para a implementação, secção 4.2.1.4, de que existe muita dificuldade em delimitar um intervalo de *RSS* para a definição de um local. Essas dificuldades são provocadas por atenuações elevadas por parte de corpos humanos em constante movimento neste ambiente, isto é, a frequência de operação do sistema RFID situa-se na banda dos 2.45GHz, frequência de ressonância da água; ora, sendo os humanos constituídos por mais de 60% de H2O, os sinais RF emitidos a esta frequência têm muita dificuldade em penetrar nestes corpos, tornando o uso do *RSSi* numa forma inviável de localização precisa (erro > 1m). Mesmo assim, o uso do parâmetro *RSSi* permite traçar um limite razoavelmente preciso do espaço no âmbito do trabalho. Mas, a fim disso, tem de existir um planeamento local a local, para a colocação dos leitores RFID, e definir o *RSSi* mínimo detectado a ser considerado.

Ainda sobre a configuração da porta série, é possível seleccionar as características de transmissão a partir de valores pré-definidos.

As configurações das comunicações passam por informar o *IntegratorLocative* da localização do Agente *Displacer* e qual o porto usado. Esta comunicação será efectuada assincronamente, através duma ligação *UDP*. A escolha de comunicação assíncrona baseou-se na programação do intervalo de tempo de emissão do sinal das tags ser inferior ao 0.5 seg, e o intervalo de tempo entre questões do Agente aos leitores ser de 100 mseg; deste modo, existirão muitas leituras redundantes, e o envio dessa redundância de informações ao Agente *Displacer* é suficiente para garantir uma boa recepção de, pelo menos, uma leitura correcta, pois erros são susceptíveis de ocorrer na comunicação, podendo estes comprometer a validade da informação das leituras das tags tratadas.

A comunicação com o agente *Displacer* só é efectuada após a confirmação da existência da máquina onde este se encontra a executar.

A validade das tramas enviada pelos leitores e recebidas pelo Agente é realizada através do código de detecção de erro associado ao protocolo do sistema RFID.

4.2.1.3 - Arquitectura Lógica

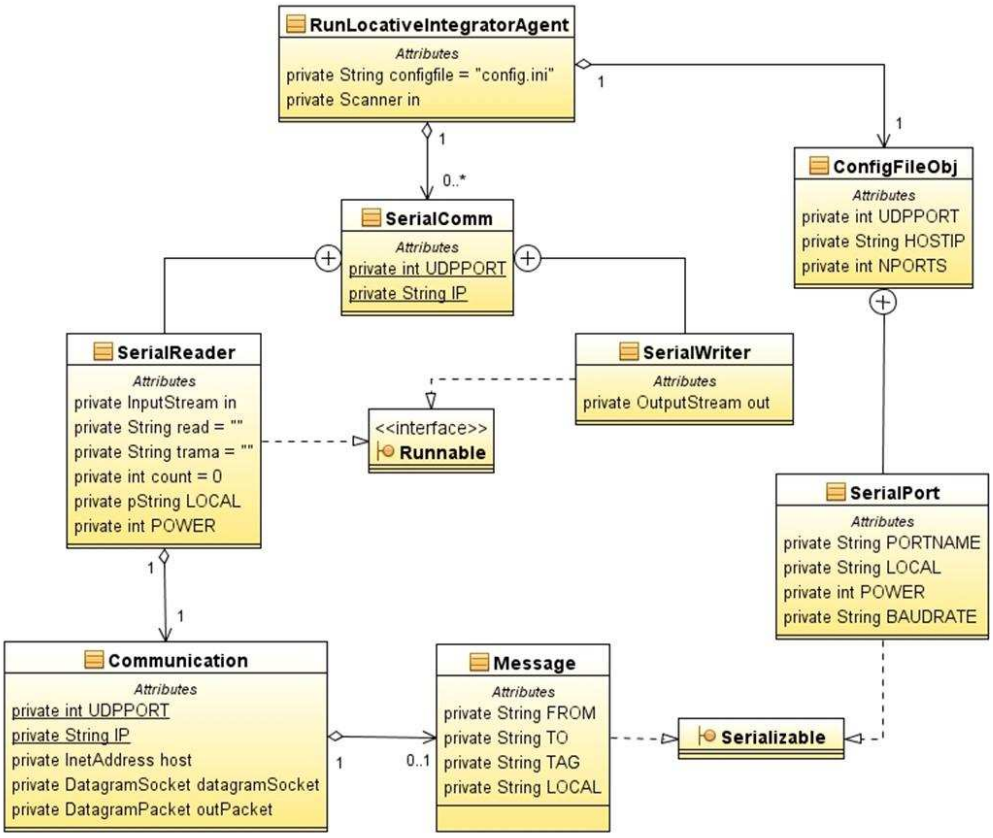

Figura 4.4 - Diagrama de classes do Agente *IntegratorLocative*

A classe ***RunLocativeIntegratorAgent*** é a que executa o agente. Primeiro verifica a existência e integridade do ficheiro de configuração, e, caso existam falhas neste, exige ao administrador uma configuração válida.

Um objecto ***ConfigFileObj*** irá conter toda a configuração do agente.

A classe ***SerialPort*** está associada a cada porta configurada, de modo a suportar os dados de configuração.

A classe ***SerialComm*** tem por objectivo a configuração das portas séries e lida com as transmissões do sistema RFID. Esta classe é composta por duas classes internas: a classe ***SerialComm.SerialReader*** lê das portas séries as tramas recebidas dos leitores RFID, integrando e tratando os dados –caso os dados forem válidos inicia o processo de comunicação com o agente *Displacer*. Já a outra classe, a **SerialComm.SerialWriter**, tem como função o envio de comandos aos leitores Rfid, neste caso o comando 'C' correspondendo ao pedido de informação da detecção de tags.

A classe ***Communication*** é responsável por transmitir ao agente ***Displacer*** os dados fornecidos pelos leitores.

A classe ***Message*** corresponde às informações de uma trama válida e associada a um local.

Tabela 4.1 - Mapeamento Lógico entre os Protocolos e Actividades do Agente IntegratorLocative estabelecidos por Gaia e as classes do Protótipo

	Denominação Gaia	**Classes *Lógicas do* Protótipo**
Actividades	*QuestionarLeitor*	*SerialComm.SerialWriter*
	ReceberTrama	*SerialComm.SerialReader*
	ProcessarTrama	*SerialComm.SerialReader*
	ApurarLocal	*SerialComm.SerialReader*
Protocolos	*IniciarDeslocamentos*	*Communication*

Para mais detalhe acerca da arquitectura lógica do agente, nomeadamente os métodos implementados, consultar o anexo A.

4.2.1.4 -Modelo da tecnologia RFID

4.2.1.4.1 - Leitor RFID

Foi usado o modelo *SYRD245-2R* da *Kimaldi*, figura 4.5, tecnologia activa RFID - as suas características mais relevantes são apresentadas na tabela 4.2.

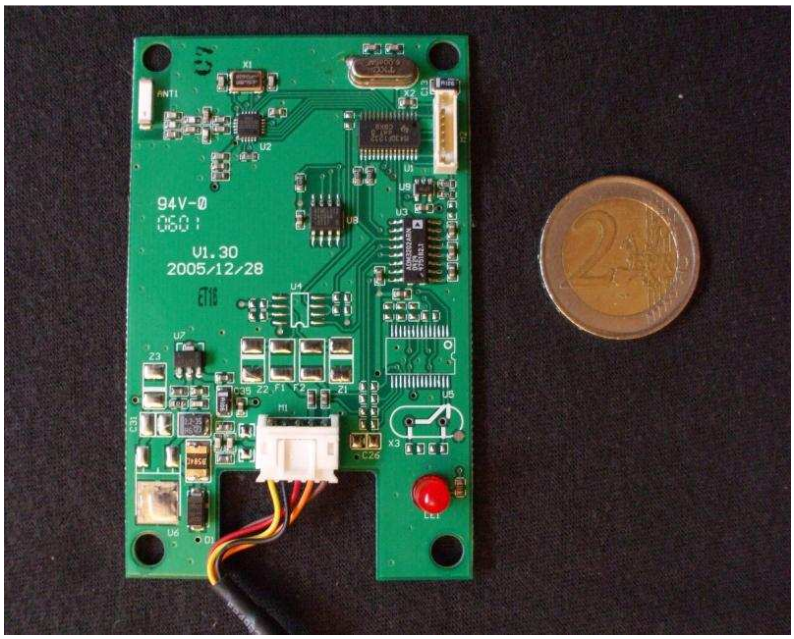

Figura 4.5 - Leitor RDID *SYRD245-2R* da *Kimaldi*

Tabela 4.2 - Carateristicas relevantes do modelo *SYRD245-2R* da *Kimaldi*

Frequência de operação	2.45 GHz (Suporte de leitura e escrita)
RSSI	0-255
LQI (Link Quality Indicator)	0-255
Canal	255
Paramêtros	A identificação do leitor é configurável (ID (1-9999)) e permite gravação de dados de utilizador (256 Bytes)
Interface	Usa porta RS232 para recepção e emissão
Antena	Embutida –alcance de leitura de 10 metros
Escrita de Tags	Permite configuração do intervalo de tempo entre activações das tags, e gravação de dados
Leitura múltipla de tags	Evita colisões de leitura e oferece um bom nível de desempenho, mesmo com interferências várias

4.2.1.4.2 - Tags RFID

Foram usadas as tags activas *SYTAG245-2S* da *Kimaldi*, figura 4.6. Permitem uma capacidade de armazenamento de dados entre 32Kb e 128Kb; a identificação é de 64 bits e o alcance típico de transmissão é de 15 m.

Figura 4.6 - Tags activas *SYTAG245-2S* da *Kimaldi*

Figura 4.7 - Chaveiro usado para transporte e protecção das tags *SYTAG245-2S*

4.2.1.4.2 - Protocolo RFID

As tramas do protocolo RFID usadas pelo agente são:

Comando: IntegratorLocative -> SYRD245-2R

STX + **INS** + { **DATA** } + **BCC** + **END**
STX : [*01*]
INS : "*C*"
DATA : nenhum (formato *Hex*)
BCC : *STX* xor *INS* (formato Hex)
END : [*0D*]

Comando: SYRD245-2R -> IntegratorLocative

STX + **INS** + { **DATA** } + **BCC** + **END**

STX : [*02*]
INS : "*C*"
DATA : "*aaaaaaaaaaaaaa*" Tag SN (8 Bytes), "*bb*" *RSSI*, "*cc*" *LQI*, …
BCC : *STX xor INS xor* {*DATA1 xor ……. xor DATAn* (formato *Hex*)
END : [*0D*]

Todos os dados dos campos são em formato ASCII, excepto STX e END
Símbolos:

[]: significa valor hexadecimal
"": significa tipo *string* de dados

4.2.2 - Agente Displacer

4.2.2.1- Descrição

O Agente *Displacer* tem como objectivo averiguar a deslocação das entidades físicas. Através da memória dos locais onde as entidades se acham é capaz de inferir o acto de deslocamento.

4.2.2.2 - Configuração

No início da execução do Agente, o Administrador deve passar-lhe alguns parâmetros essenciais ao seu bom funcionamento; se o não fizer, os valores destes são os por omissão. Estes parâmetros são passados por linha de comando do seguinte modo:

Java -jar DisplacerAgent.jar [input_UDP_PORT output_TCP_PORT destiny_Host idle_Time]

input_UDP_PORT	Porto usado na comunicação com Agente *IntegratorLocative*
output_TCP_PORT	Porto usado na comunicação com Agente *Informer*
destiny_Host	Localização do Agente *Informer*
idle_Time	Intervalo de tempo em estado não detectável que uma tag pode permanecer até o Agente *Displacer* a eliminar da memória

Na altura de decidir o valor do intervalo de tempo, é aconselhado ter em conta a dimensão da sala e fazer uma previsão da frequência de deslocamento das entidades.

O mecanismo de memorização, que tem por base a inferência de deslocamento duma entidade, é apresentado no diagrama de actividade do Agente *Displacer*, figura 4.8. Este

mecanismo ocorre quando o Agente recebe um pedido de serviço de *Inferência de Deslocação* baseado no protocolo *IniciarDeslocamento*.

O facto de as *tags* serem activas e estarem permanentemente transmitindo faz com que se deva apenas actualizar na lista o tempo da sua detecção. A entrada duma nova *tag* na lista é vista pelo Agente como um deslocamento, realizando o envio dos dados relevantes do evento ao Agente *Informer*. Por esse motivo, é necessário eliminar os dados da lista referente à detecção de uma tag que tenha ultrapassado o limite de tempo sem actividade. Desta acção depreende-se que uma entidade abandonou o local onde se encontrava.

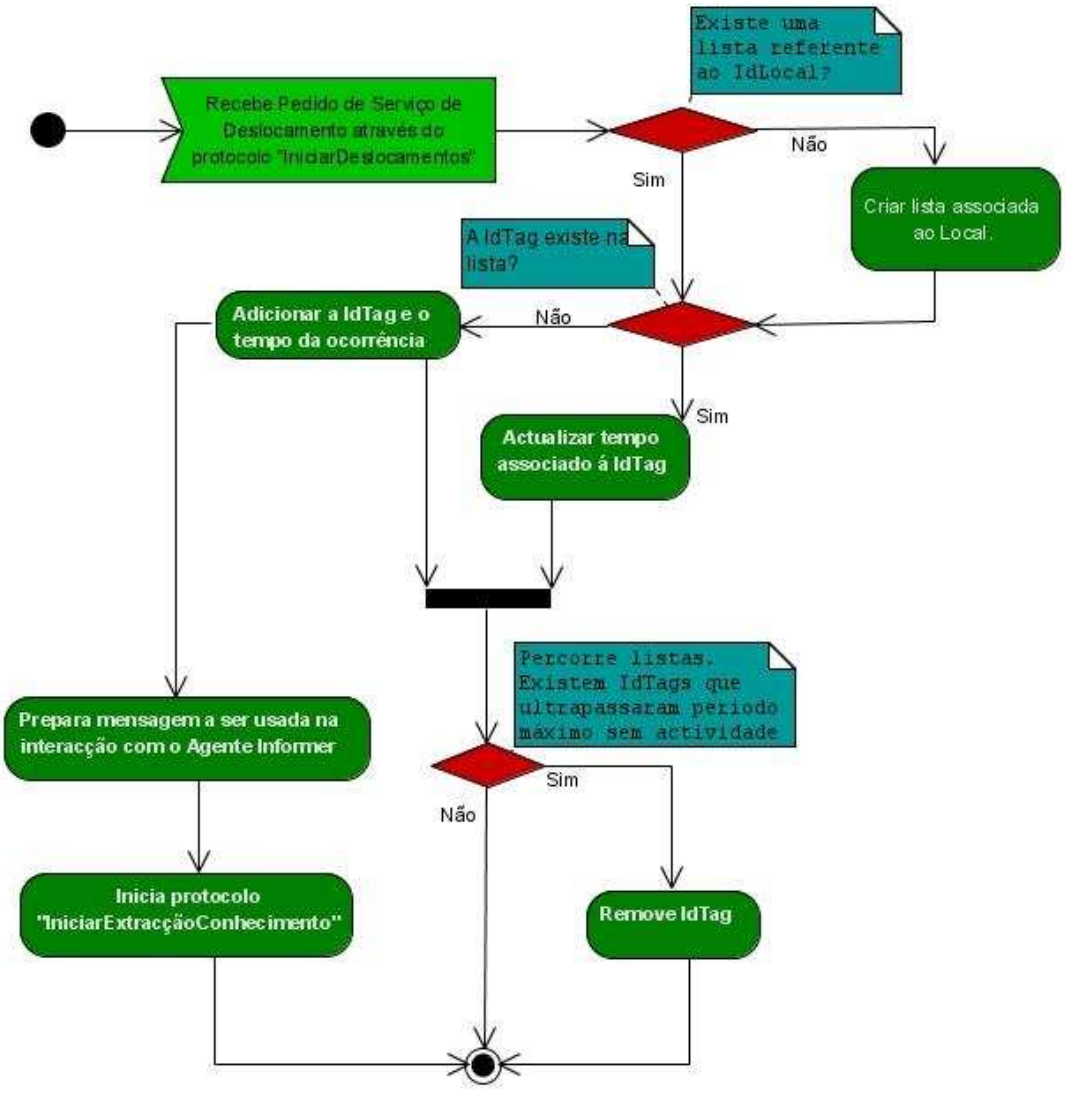

Figura 4.8 - Diagrama de actividade do Agente *Displacer*

4.2.2.3 - Arquitectura Lógica

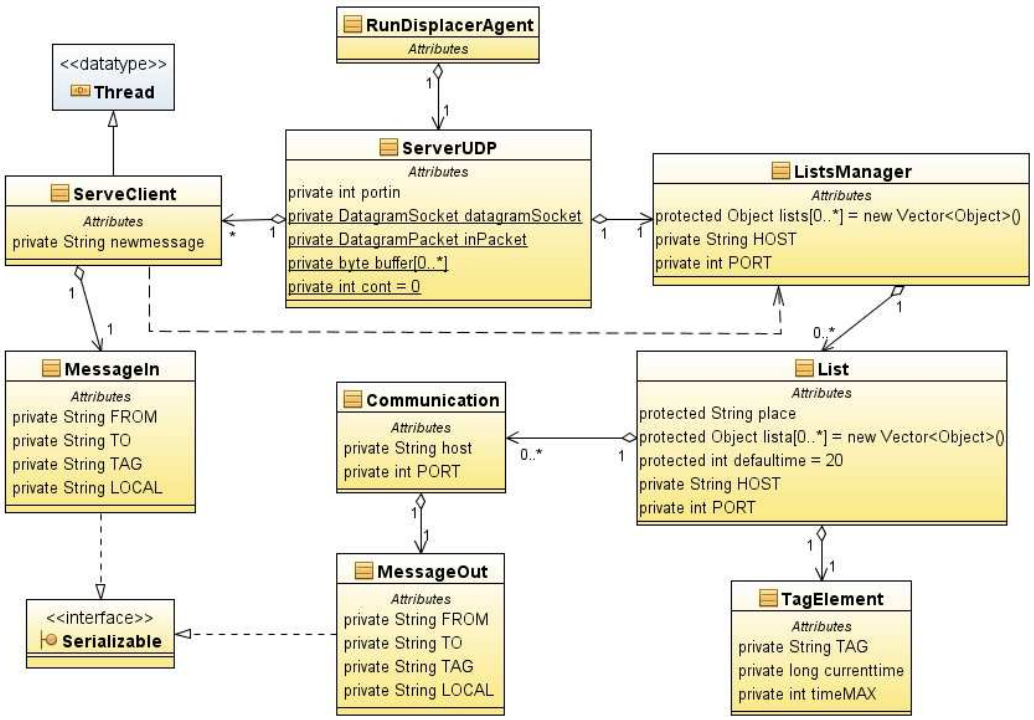

Figura 4.9 - Diagrama de classes do Agente *Displacer*

A classe ***RunDisplacerAgent*** é a classe que executa o agente.

A classe ***ServerUDP*** controla as comunicações de entrada do agente.

A classe ***ServeClient*** tem por objectivo o tratamento de cada pedido recebido do agente *IntegratorLocative*.

A classe ***MessageIn*** é muito semelhante à classe ***Message*** do Agente *IntegratorLocative*; corresponde às informações duma trama válida e associada a um local

A classe ***ListsManager*** controla as listas das tags detectadas e os dados enviados ao agente Informer.

A classe ***List*** está associada a um local; é nela onde as informações das tags são guardadas.

Um objecto ***TagElement*** representa as informações duma tag com uma variável tempo associada.

A classe ***Communication*** é responsável por transmitir as informações ao agente *Informer*.

A classe ***MessageOut*** corresponde a uma mensagem contendo informação da detecção da tag a ser enviada ao agente Informer.

Tabela 4.3 - Mapeamento Lógico entre os Protocolos e Actividades do Agente *IntegratorLocative* estabelecidos por Gaia e as classes do Protótipo

Tipo	***Denominação GAIA***	**Classes *Lógicas do* Protótipo**
Actividades	*InferirDeslocação*	*List*
Protocolos	*IniciarDeslocamentos*	*ServerUDP*
	IniciarExtracçãoConhecimento	*Communication*

Para mais detalhe acerca da arquitectura lógica do Agente, nomeadamente os métodos implementados, consultar o anexo A.

4.2.3 - Agente Informer

4.2.3.1 – Descrição

O Agente tem por objectivo fornecer informações de contexto relevantes, nomeadamente acerca do tipo de papel do indivíduo e da função do local, no contexto da Clínica ou Hospital.

4.2.3.2 - Configuração

Durante a inicialização da execução do Agente *Informer* o Administrador deve passar-lhe alguns parâmetros essenciais ao bom funcionamento, caso não o faça os valores destes são os por omissão. Estes parâmetros são passados por linha de comando do seguinte modo:

Java -jar InformerAgent.jar [input_TCP_PORT output_TCP_PORT destiny_Host database_IP username password]

input_TCP_PORT	Porto usado na comunicação com Agente *IntegratorLocative*
output_TCP_PORT	Porto usado na comunicação com Agente *Discoverer*
destiny_host	Localização do Agente *Discoverer*
database_IP	Localização da Base de Conhecimento
username	Nome de utilizador para acesso à Base de Conhecimento
password	Senha de utilizador para acesso à Base de Conhecimento

4.2.3.3 - Arquitectura Lógica

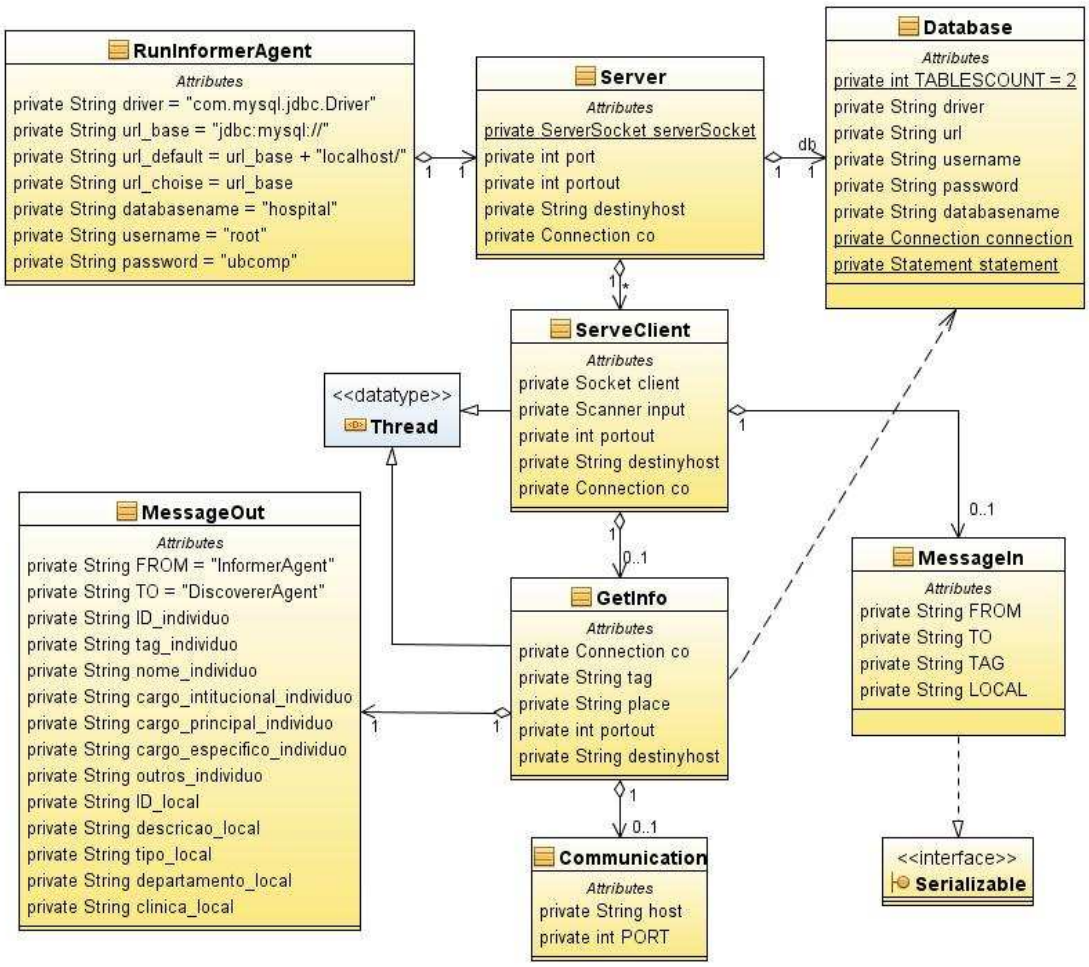

Figura 4.10 - Diagrama de classes do Agente *Informer*

A classe ***RunInformerAgent*** é a classe que executa o agente.

A classe ***Server*** controla todas as comunicações de entrada do agente.

A classe ***Database*** é responsável por conectar e gerir as bases de dados.

Um objecto ***MessageIn*** corresponde a uma informação enviada pelo agente *Displacer*.

A classe ***ServeClient*** é responsável pelo processamento das mensagens enviadas para o agente *Discoverer*.

A classe ***GetInfo*** é responsável por adquirir as informações das bases de dados.

Um objecto ***MessageOut*** corresponde a uma informação adquirida pela classe ***GetInfo***.

A classe ***Communication*** é responsável por transmitir as informações ao agente *Discoverer*.

Tabela 4.4 - Mapeamento Lógico entre os Protocolos e Actividades do Agente *Informer* estabelecidos por Gaia e as classes do Protótipo

Tipo	***Denominação GAIA***	**Classes *Lógicas do* Protótipo**
Actividades	*PesquisarBaseConhecimento*	*GetInfo*
	ExtrairConhecimento	
	FormatarInformações	*MessageOut*
Protocolos	*IniciarExtracçãoConhecimento*	*Server*
	IniciarSelecção	*Communication*

Para mais detalhe acerca da arquitectura lógica do Agente, nomeadamente os métodos implementados, consultar o anexo A.

4.2.3.4 - Arquitectura da Base de Conhecimento

Para base de Conhecimento foi usado um servidor de Base de Dados *MySQL*; devido à sua robustez e performance é dos servidores de base de dados mais utilizados.

O conhecimento encontra-se estruturado de forma organizacional e funcional, ou seja, é tido em conta o funcionamento hospitalar descrito na forma do comum organigrama usado pela maioria das administrações hospitalares. Assim, desenvolveu-se uma hierarquia de conceitos que modelam/definem o domínio Hospitalar. Deste modo, cada nível do conhecimento representa um grau de detalhe, a fim de descrever o contexto necessário ao propósito do sistema.

Para se entender melhor, apresenta-se como as tabelas da Base de Dados Hospital, figura 4.11 – 4.12, modelam o conhecimento do Agente Informer, dando-se alguns exemplos de como as usar.

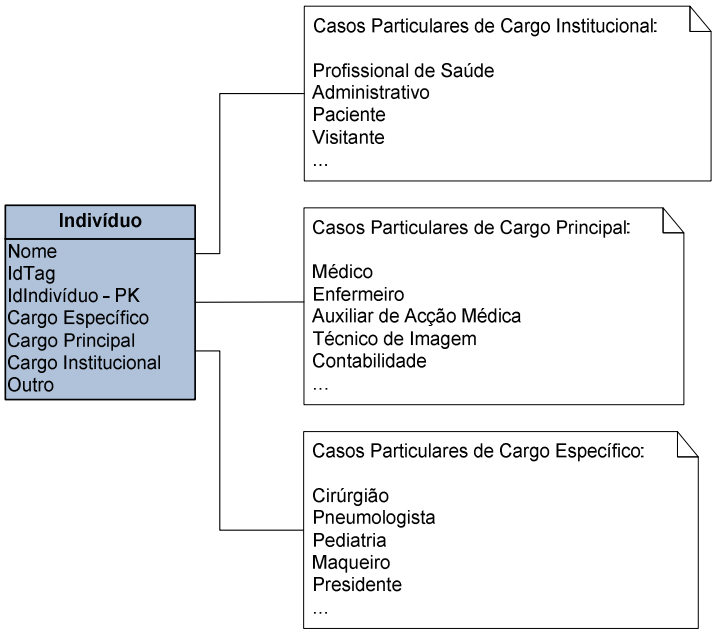

Figura 4.11 - Tabela Indivíduo na Base de Dados Hospital

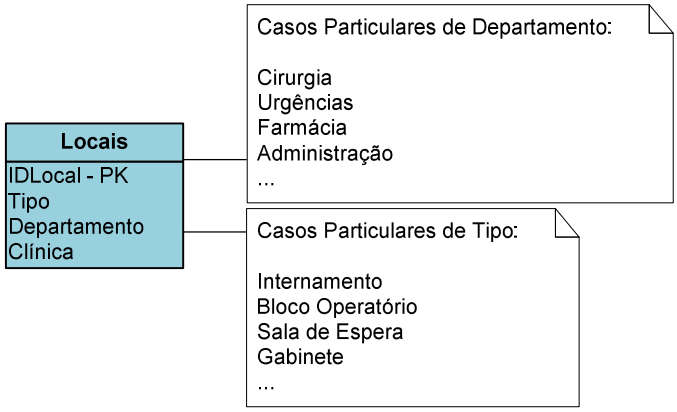

Figura 4.12 - Tabela Locais na Base de Dados Hospital

4.2.4 - Agente Discoverer

4.2.4.1 - Descrição

O objectivo do Agente *Discoverer* é estabelecer relações apropriadas entre os Serviços de Saúde e o contexto actual do indivíduo, seleccionando os que mais se adaptam a este. Agrupa-os em níveis de especificidade e envia-os à entidade competente.

4.2.4.2 - Configuração

Na inicialização da execução do Agente *Discoverer*, o Administrador deve passar-lhe alguns parâmetros essenciais ao seu bom funcionamento, caso não o fizer os valores destes são os por omissão. Estes parâmetros são passados por linha de comando do seguinte modo:

Java -jar DiscovererAgent.jar [input_TCP_PORT output_TCP_PORT destiny_Host database_IP username password]

input_TCP_PORT	Porto usado na comunicação com Agente *Informer*
output_TCP_PORT	Porto usado na comunicação com Agente *UserWizard*
destiny_Host	Localização do Agente *UserWizard*
database_IP	Localização do Repositório de Serviços
username	Nome de utilizador para acesso ao Repositório de Serviços
password	Senha de utilizador para acesso ao Repositório de Serviços

4.2.4.3 - Arquitectura Lógica

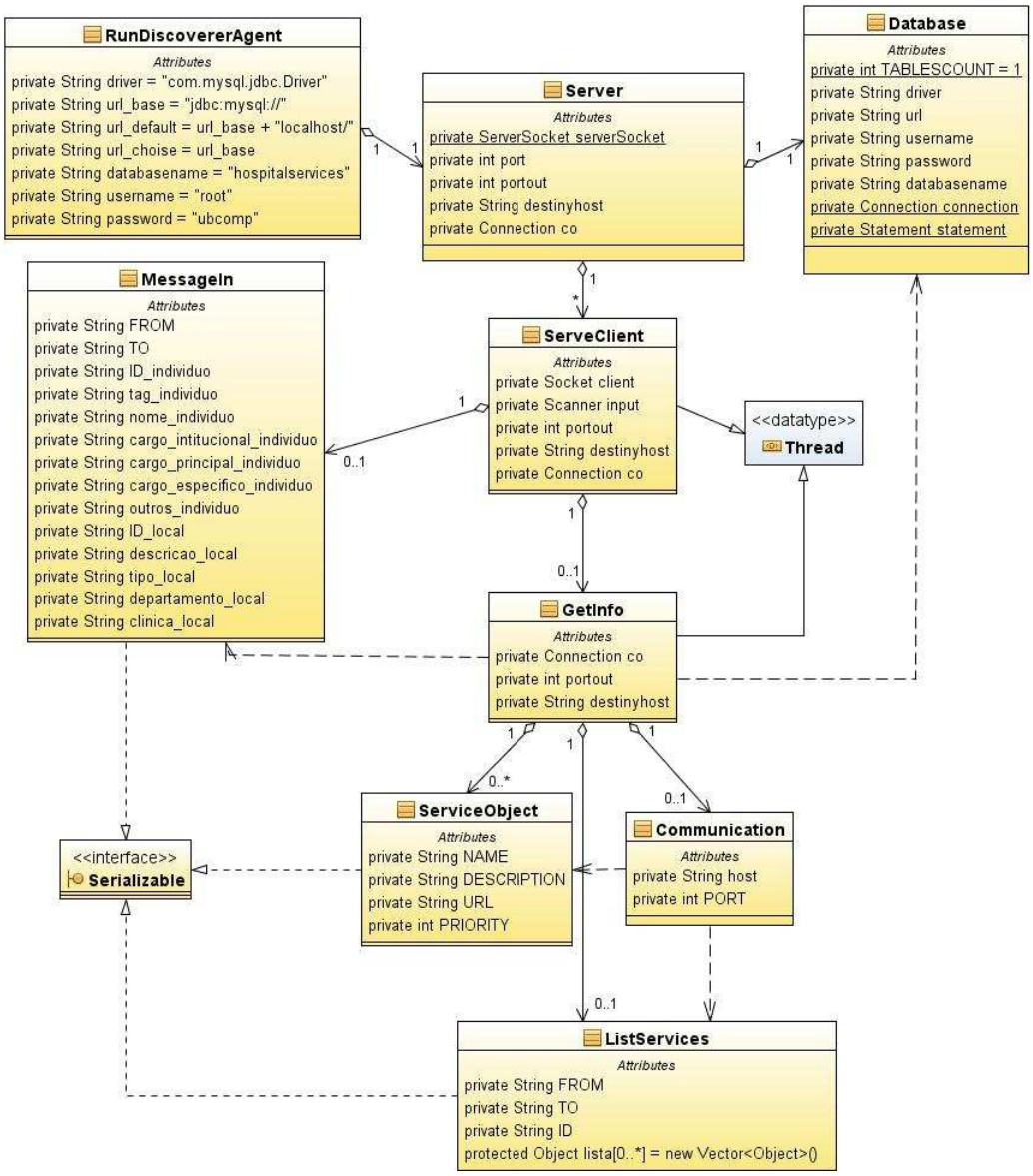

Figura 4.13 - Diagrama de classes do Agente *Discoverer*

A classe ***RunInformerAgent*** é a classe que executa o agente.

A classe ***Server*** controla todas as comunicações de entrada do agente.

A classe ***Database*** é responsável por conectar e gerir a comunicação com a Base de Dados.

A classe ***ServeClient*** é responsável pelo processamento das mensagens enviadas para o Agente *UserWizard*.

Um objecto ***MessageIn*** corresponde a uma informação de contexto adquirida através do agente Informer.

A classe ***GetInfo*** é responsável por adquirir as descrições dos serviços da Base de Dados.

Um objecto ***ListServices*** contém os serviços seleccionados.

Um objecto ***ServiceObject*** corresponde a cada serviço seleccionado.

A classe ***Communication*** é responsável por transmitir as informações ao agente *UserWizard*.

Tabela 4.5 - Mapeamento Lógico entre os Protocolos e Actividades do Agente *Discoverer* estabelecidos por Gaia e as classes do Protótipo

Tipo	***Denominação GAIA***	**Classes *Lógicas do* Protótipo**
Actividades	*RelacionarInfContextoServiços*	*GetInfo*
	SeleccionarServiços	
	ExtrairDescriçõesServiços	
	CriarListaServiços	*ListServices*
Protocolos	*IniciarSelecção*	*Server*
	IniciarDistribuição	*Communication*

Para mais detalhe acerca da arquitectura lógica do Agente, nomeadamente os métodos implementados, consultar o anexo A.

4.2.4.4 - Arquitectura do Repositório de Serviços

Para Repositório das descrições de Serviços foi usado um servidor de Base de Dados MySQL.

Nessas descrições são usados termos idênticos aos das informações de contexto. Por isso, estas descrições estão fortemente ligadas à forma como foi modelado o domínio Hospital.

A figura 4.14 mostra a estrutura da Base de Dados que suporta as descrições dos serviços.

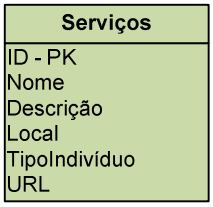

Figura 4.14 - Tabela Serviços na Base de Dados Serviços

Cada descrição de Serviço é armazenado na Base de Dados com uma identificação única, um nome, uma descrição das funcionalidades do serviço, um local onde este é adequado, um tipo de Indivíduo para quem mais se ajusta e por fim um URL (*Uniform Resource Locator*), ou seja, o endereço onde o serviço pode ser encontrado ou o *scrip* de execução deste.

O Agente *Discoverer* aceita como entrada o conjunto das informações de contexto, executa o processo de *matching* provendo como resultado uma lista ordenada das n melhores ofertas em relação a cada demanda. Esta ordenação é relacionada com o nível de especificidade da união do Local com o Indivíduo. A relevância dos serviços neste processo é vista pelo Agente como apresentado na tabela 4.6.

Tabela 4.6 – Grau de relevância (certeza) dos serviços seleccionados

Grau de relevância	*Papel*	*Local*
1	*IdIndividuo*	*IdLocal*
2	*IdIndividuo*	*Tipo de Local*
3	*Cargo Especifico*	*IdLocal*
4	*Cargo Especifico*	*Tipo de Local*
5	*Cargo Principal*	*IdLocal*
6	*Cargo Principal*	*Tipo de Local*
7	*Cargo Institucional*	*IdLocal*
8	*Cargo Institucional*	*Tipo de Local*
9	*IdIndividuo*	*Departamento*
10	*Cargo Especifico*	*Departamento*
11	*Cargo Principal*	*Departamento*
12	*Cargo Institucional*	*Departamento*
13	*IdIndividuo*	*Clínica*
14	*Cargo Especifico*	*Clínica*
15	*Cargo Principal*	*Clínica*
16	*Cargo Institucional*	*Clínica*
17	*Outros*	*IdLocal*
18	*Outros*	*Tipo de Local*
19	*Outros*	*Departamento*
20	*Outros*	*Clínica*

Pode-se observar que desta forma existem 20 níveis de especificidade. Cada nível está associado a uma prioridade (certeza do sistema). A partir desta é possível organizar por pertinência os serviços passíveis de ser apresentados aos utilizadores.

4.2.5 - Agente UserWizard

4.2.5.1 - Descrição

O objectivo do Agente *UserWizard* é representar os Utilizadores no sistema. Para isso, armazena as listas de serviços relacionadas com cada utilizador e associa-se à Aplicação de alto nível que tem como função listar os serviços seleccionados e o executar dos mesmos.

4.2.5.2 - Configurações

Na inicialização da execução do Agente *UserWizard*, o Administrador deve passar-lhe alguns parâmetros essenciais ao seu bom funcionamento, caso não o fizer os valores destes são os por omissão. Estes parâmetros são passados por linha de comando do seguinte modo:

Java -jar UserWizardAgent.jar [input_TCP_PORT service_path]

input_TCP_PORT	Porto usado na comunicação com Agente Discoverer
service_path	Directoria onde são guardados os ficheiros com lista de serviços de cada utilizador

4.2.5.3 - Arquitectura Lógica

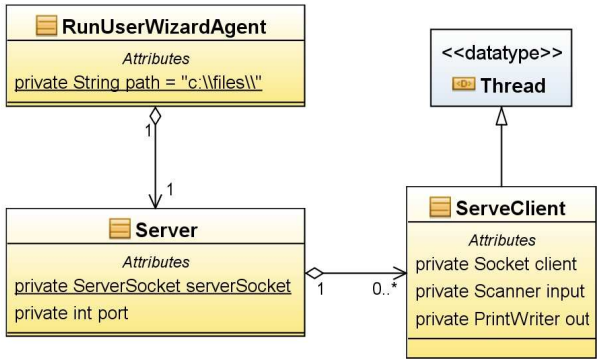

Figura 4.15 - Diagrama de classes do Agente *UserWizard*

A classe **RunUserWizardAgent** é a classe que executa o agente.

A classe ***Server*** controla todas as comunicações de entrada do Agente.

A classe ***ServeClient*** é responsável por enviar a lista de cada utilizador para um ficheiro particular numa directoria usada pelos aplicativos.

Tabela 4.7 - Mapeamento Lógico entre os Protocolos e Actividades do Agente *UserWizard* estabelecidos por Gaia e as classes do Protótipo

Tipo	**Denominação GAIA**	**Classes *Lógicas do* Protótipo**
Actividades	*ExibirListaServiços*	*Estas duas actividades ficaram associadas ao aplicativo de alto nível, que faz uso dos ficheiros gerados pelo agente UserWizard*
	ExecutarServiços	
Protocolos	*IniciarVisualização*	*Server*

Para mais detalhe acerca da arquitectura lógica do Agente, nomeadamente os métodos implementados, consultar o anexo A.

4.3 – Gestor de Bases de Dados

O módulo *Gestor de Bases de Dados* foi desenvolvido para facilitar a gestão da Base de Conhecimento do Agente *Informer*, e do Repositório das descrições dos serviços.

4.3.1 - Interface gráfica

Ao iniciar o aplicativo de gestão de bases de dados, existe a oportunidade de carregar as definições pré definidas, figura 4.16.

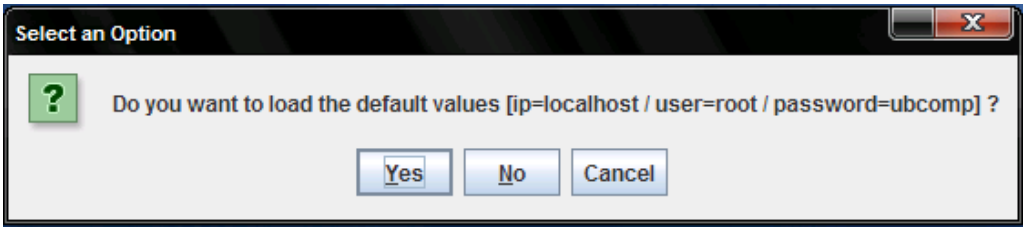

Figura 4.16 - Opção de carregamento rápido das definições do *Gestor de Bases de Dados*

Caso as bases de dados existam e os dados de acesso aos servidores forem correctos, a interface principal permite o acesso a todas as suas opções, figura 4.17.

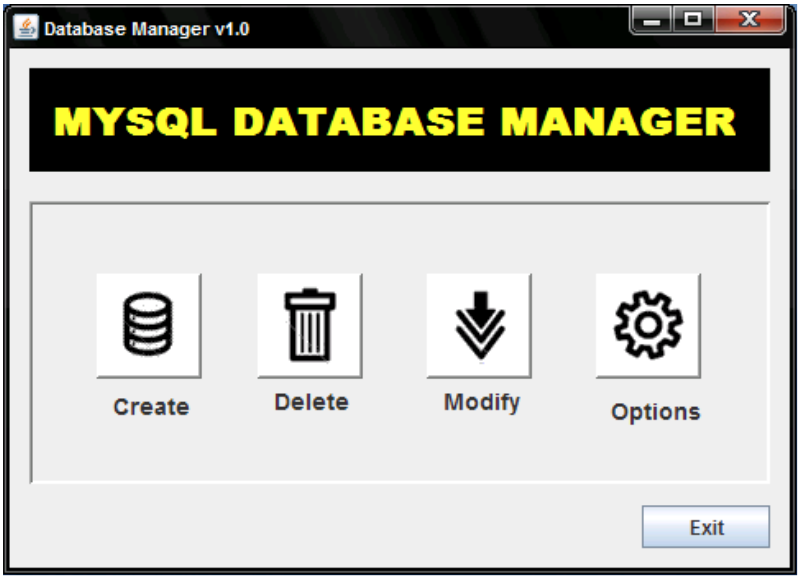

Figura 4.17 - Menu Principal do Gestor de Bases de Dados

Se os dados de acesso aos Servidores estiverem incorrectos, ou for seleccionada a opção de introdução manual destes dados, deve-se aceder à opção '*Options*' do menu principal do aplicativo figura 4.17, e na interface aberta, figura 4.18, devem-se introduzir os dados de acesso às bases de dados.

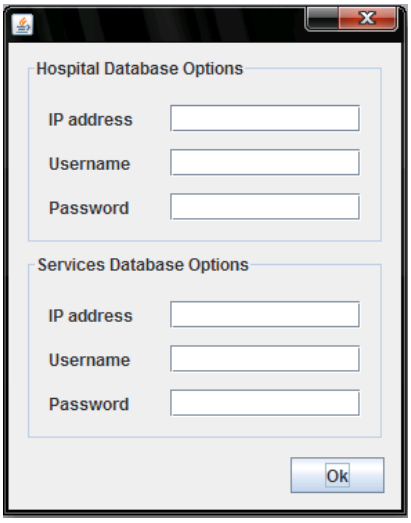

Figura 4.18 - Interface de configuração dos acessos aos servidores de Bases de Dados

Uma vez validados os acessos aos servidores de Bases de Dados, é possível aceder às restantes opções.

Se as bases de dados ainda não tiverem sido criadas, deve-se optar pela opção '*Create*' da interface principal, figura 4.17. A partir da interface aberta, figura 4.19, é possível criar as bases de dados e/ou as tabelas respectivas.

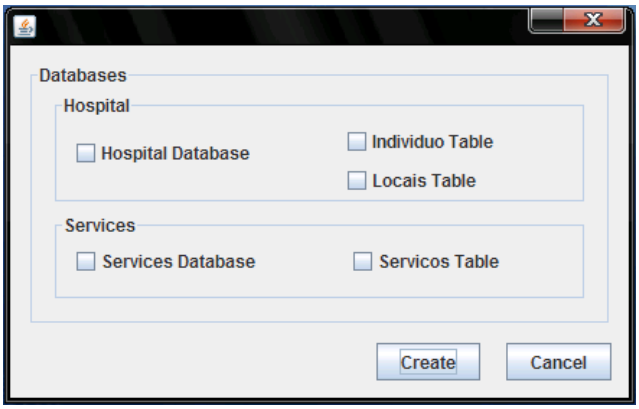

Figura 4.19 - Criar bases de dados e respectivas tabelas

Caso a opção for a de eliminar as bases de dados e/ou as suas tabelas, recorre-se a uma interface muito idêntica à da interface da figura 4.19.

A opção '*Modify*' do menu principal, figura 4.17, permite aceder à interface de gestão de conteúdo das bases de dados figura 4.20. As tabelas Indivíduo, figura 4.20, Locais, figura 4.21, e Serviços, figura 4.22, encontram-se em abas de modo a facilitar a navegação. Dentro das possibilidades de gestão encontra-se adicionar uma nova entrada figura 4.23.

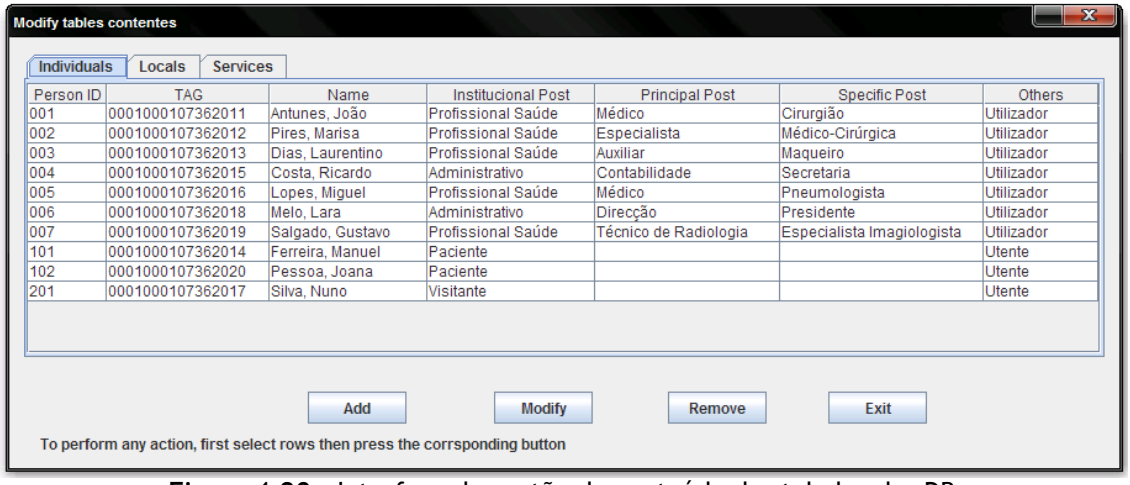

Figura 4.20 - Interface de gestão do conteúdo das tabelas das DB

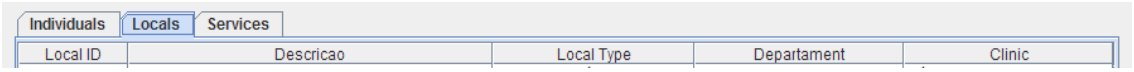

Figura 4.21 - Aba de acesso aos dados da tabela Locals da base de dados Hospital

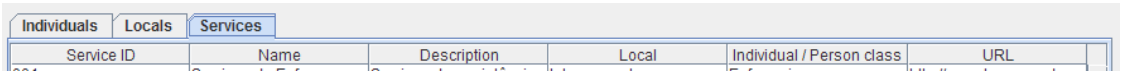

Figura 4.22 - Aba de acesso aos dados da tabela Serviços da base de dados Serviços

Caso se pretenda eliminar ou modificar, figura 4.20, deve-se seleccionar primeiro a linha pretendida e só depois a opção desejada.

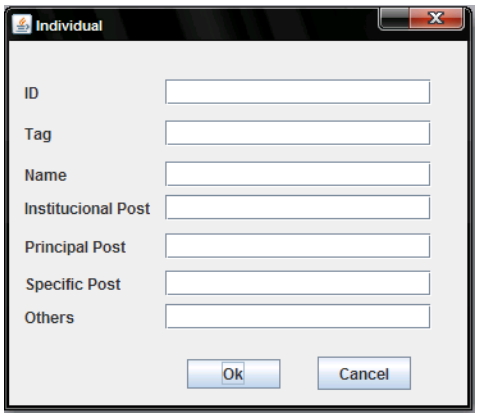

Figura 4.23 - Exemplo da Interface "adicionar" e "modificar" registo

4.4 - Simulação de um Caso

Foram realizadas algumas simulações com a finalidade de comprovar a funcionalidade da plataforma. Uma das simulações é apresentada de seguida.

Foi criada uma topologia de rede dum possível cenário de implantação da plataforma, como apresentado na figura 4.24. Pretende-se simular a deslocação de um indivíduo de um local específico para outro nas instalações dum Hospital.

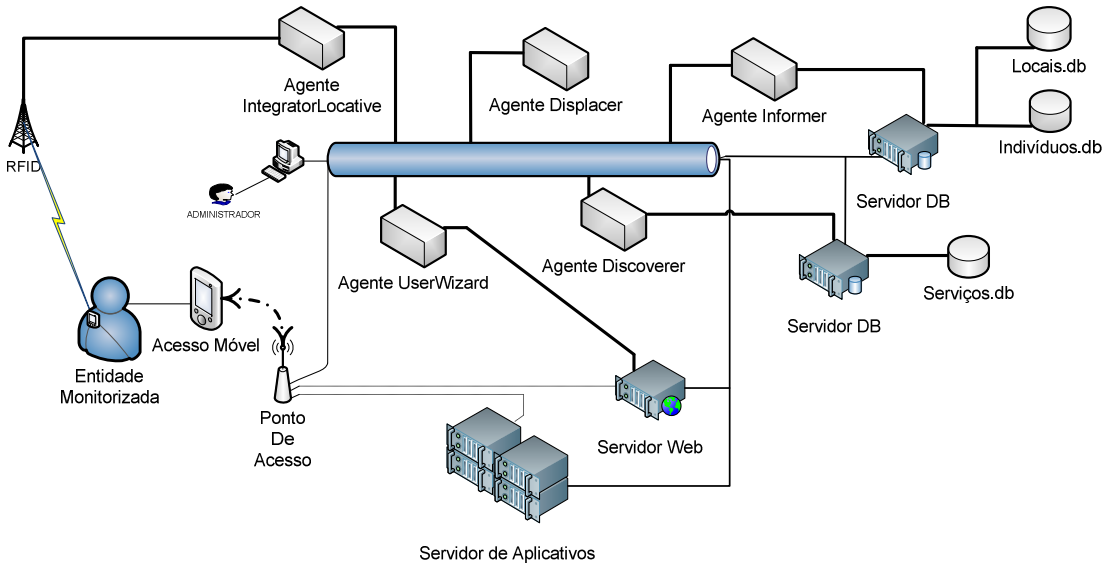

Figura 4.24 - Diagrama com a topologia da rede

Nesta topologia, os componentes do sistema encontram-se repartidos pela rede, sendo constituída por:

- Os agentes a executar em máquinas diferentes;
- As bases de dados Hospital e Serviços, colocadas nas máquinas onde correm os agentes *Informer* e *Discoverer*, respectivamente;
- Um aplicativo JSP criado, a ser executado num servidor Web, instalado na mesma máquina do Agente *UserWizard*;
- Um suposto servidor de *Aplicativos*, onde se encontram os serviços oferecidos aos utilizadores;
- Dois leitores *RFID SYRD245-2R*, dispersos por dois locais de teste (um dos leitores foi simulado a partir do envio de tramas previamente compostas);
- A tag dum indivíduo;
- Um *PDA* com acesso *WiFi*.

Para servidor Web foi elegido o *Apache Tomcat* [Tom09], por ser gratuitamente distribuído sob uma licença *Apache Software*, e por ser altamente adequado para o suporte à tecnologia *JSP* (*Java Server Pages*).

O aplicativo *JSP* desenvolvido tem por objectivo o uso dos ficheiros de serviços de cada indivíduo (*IDuser.sev*) resultantes da cooperação entre os agentes do MAS. A figura 4.25 apresenta o diagrama de distribuição de componentes de parte do sistema usado nesta simulação.

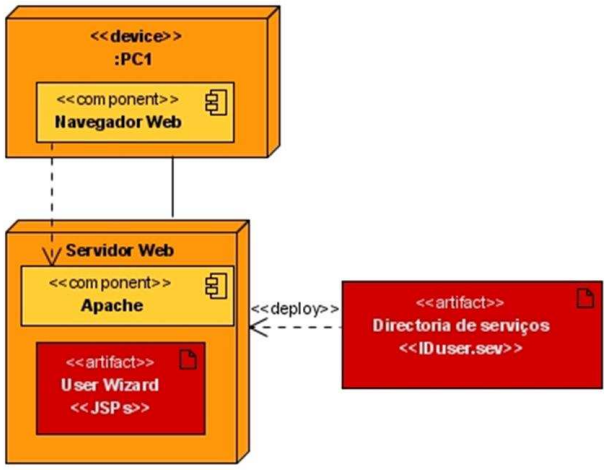

Figura 4.25 - Diagrama de distribuição de componentes de parte do sistema usado nesta simulação

As Bases de Dados foram preenchidas com dados que se julga caracterizarem indivíduos, locais e serviços intervenientes num ambiente hospitalar. As figuras 4.26 – 4.28 expõem a sua realização.

Individuals | Locals | Services

Person ID	TAG	Name	Institucional Post	Principal Post	Specific Post	Others
001	0001000107362011	Antunes, João	Profissional Saúde	Médico	Cirurgião	Utilizador
002	0001000107362012	Pires, Marisa	Profissional Saúde	Especialista	Médico-Cirúrgica	Utilizador
003	0001000107362013	Dias, Laurentino	Profissional Saúde	Auxiliar	Maqueiro	Utilizador
004	0001000107362015	Costa, Ricardo	Administrativo	Contabilidade	Secretaria	Utilizador
005	0001000107362016	Lopes, Miguel	Profissional Saúde	Médico	Pneumologista	Utilizador
006	0001000107362018	Melo, Lara	Administrativo	Direcção	Presidente	Utilizador
007	0001000107362019	Salgado, Gustavo	Profissional Saúde	Técnico de Radiologia	Especialista Imagiologista	Utilizador
101	0001000107362014	Ferreira, Manuel	Paciente			Utente
102	0001000107362020	Pessoa, Joana	Paciente			Utente
201	0001000107362017	Silva, Nuno	Visitante			Utente

Figura 4.26 - Identificação dos indivíduos intervenientes na simulação

Individuals | Locals | Services

Local ID	Descricao	Local Type	Departament	Clinic
001		Bloco Operatório	Cirurgia	Clínica
002	Sala de Observação de pacientes	Internamento	Cirurgia	Clínica
003		Gabinete	Cirurgia	Clínica
004		Sala Espera	Cirurgia	Clínica
005		Sala Espera	Radioterapia e Imagem	Clínica
006		Corredor	Urgências	Clínica
007		Direcção	Administração	Clínica
008		Logística	Farmácia	Clínica
009		Laboratório	Investigação	Clínica
010		Atendimento	Consulta Externa	Clínica
011	Sala de pequena, média cirurgia	Sala Cirurgia	Cirurgia	Clínica

Figura 4.27 - Identificação dos locais usados na simulação

Individuals | Locals | Services

Service ID	Name	Description	Local	Individual / Person class	URL
001	Serviços de Enfermagem	Serviços de assistência a p...	Internamento	Especialista	http://symptom...
002	Serviços Farmacêuticos	Pesquisa de informações a...	Internamento	Médico	http://symptom...
003	Serviços de Tesouraria	Serviços de pagamentos e r...	Administração	Contabilidade	http://symptom...
004	Serviço de Localização	Visualização da localização ...	Clínica	Profissional Saúde	http://localizaca...
005	Serviços de Paciente	Informações dos pacientes ...	Internamento	Profissional Saúde	http://symptom...
006	Serviço de Vagas	Mostra capacidade de intern...	Internamento	Médico	http://VagasSer...
007	Serviço de Radiologia	Exames radiológicos, gerai...	Radioterapia e Imagem	Técnico de Radiologia	http://symptom...
008	Serviço de Ajuda Cirúrgica	Apoio á tomada de decisão.	Sala Cirurgia	Cirurgião	http://symptom...
009	Serviços de Monitorização Cirúrgica	Ligação aos sistemas de m...	Sala Cirurgia	Cirurgião	http://symptom...
010	Serviços Equipas Cirúrgicas	Formação de equipas para ...	Cirurgia	Cirurgião	http://symptom...
011	Serviços de Atendimento	Informações gerais sobre o...	Clínica	Visitante	http://symptom...
012	Serviços de Controlo Infecções	Propiciar orientação e para ...	Clínica	Profissional Saúde	http://symptom...
019	Servicos Culturais	Informacões acerca da ofert...	Clínica	Utente	http://symptom...

Figura 4.28 - Identificação dos serviços usados na simulação

Como foi referido, esta simulação pretende seleccionar os serviços que mais se adaptem ao indivíduo, neste caso o Cirurgião *Dr. Antunes* com identificação *001*. Os dois locais escolhidos foram uma *sala de Internamento* e uma *sala de Cirurgia*, com identificação *002* e *011*, respectivamente. Uma representação gráfica das instalações e do percurso simulado pode ser observado na figura 2.29.

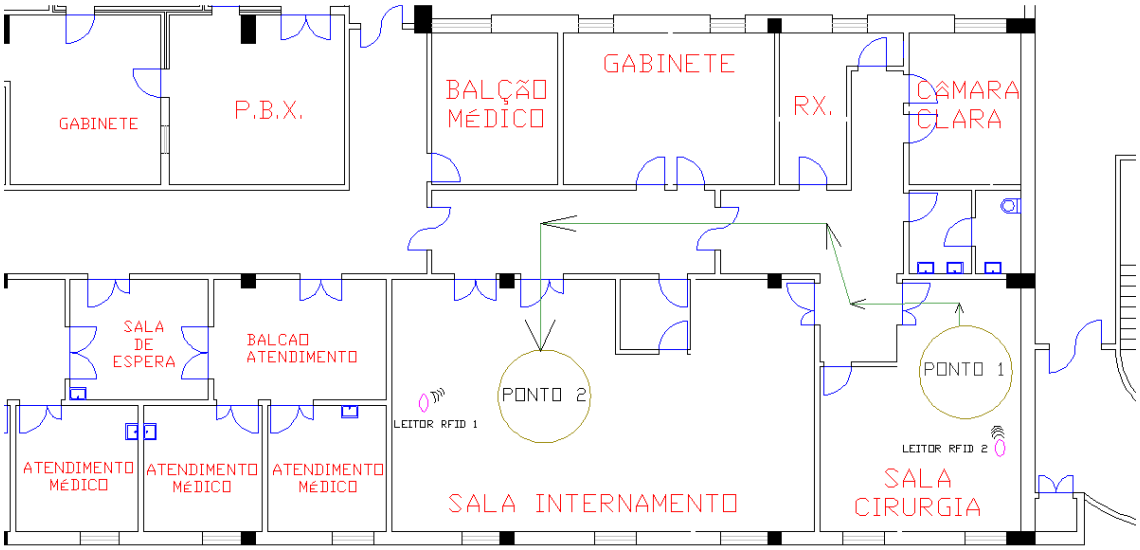

Figura 4.29 - Percurso simulado nas instalações dum hospital

O cenário contempla a realização de uma pequena cirurgia a um doente no Serviço de Urgência Hospitalar.

O médico pneumologista *Dr. Lopes* é chamado pelo médico de triagem ao Serviço de Urgência para observar um doente, que entrara com quadro clínico de insuficiência respiratória aguda, de instalação súbita. Efectuada a avaliação clínica e o estudo radiográfico, diagnostica-lhe um pneumotórax espontâneo sob tensão. É uma situação de emergência, cujo tratamento poderá ter de incluir a drenagem cirúrgica da cavidade pleural.

O *Dr. Lopes* contacta o colega cirurgião *Dr. Antunes*, que entretanto chega ao Gabinete Médico, onde já aquele se encontrava com o doente. De novo avaliado, e após conferenciarem, decidem efectuar a aplicação de um dreno torácico com drenagem sub-aquática.

O doente é transportado para a Sala de Emergência e Pequena Cirurgia. E lá, o Dr. Antunes requereu ao sistema os serviços para ele seleccionado, bastando identificar-se a partir do seu PDA (figura 4.30). Logo os serviços foram apresentados (figura 4.31), e ele seleccionou o primeiro a surgir-lhe no ecrã - "*Serviços de Ajuda Cirúrgica*". O *Dr. Antunes* visualizou os elementos da sua equipa de cirurgia, e decidiu solicitar a colaboração da enfermeira especialista médico-cirúrgica ("Instrumentista"), *Marisa Pires*, seleccionando o terceiro serviço ao seu dispor: "*Serviços de Equipas Cirúrgicas*", e procurou localizá-la através do quarto serviço sugerido, "*Serviço de Localização*".

A equipa médica procedeu à aplicação do dreno torácico com drenagem sub-aquática activa, e o doente entretanto melhorou o seu quadro clínico. O *Dr. Antunes* desloca-se à *Sala de Internamento* (*S.O.*). O sistema detecta a sua deslocação e sugere-lhe uma nova lista de serviços (figura 4.32). Nesta o Cirurgião confirma a disponibilidade de vaga na sala, através do segundo serviço exibido no seu *PDA*, "*Serviço de Vagas*". O doente é de seguida transferido para esta Sala de Internamento. E após o internamento do paciente, o *Dr. Antunes* pesquisa os fármacos ao seu dispor, através do primeiro serviço exibido pelo sistema, "*Serviços Farmacêuticos*", e acede ao terceiro serviço, "*Serviços de Paciente*", de modo a receitar-lhe o tratamento farmacológico que considera necessário durante a convalescença do paciente.

Figura 4.30 – Identificação no Sistema

Figura 4.31 – Serviços apresentados na Sala Cirúrgica

Figura 4.32 - Serviços apresentados na Sala de Internamento

Com base no preenchimento rápido das descrições dos serviços no Repositório particular, constata-se que os resultados apontam para uma boa precisão nas opções tomadas pelo Sistema no momento de oferecer os melhores serviços aos indivíduos.

Uma melhor gama de opções pode ainda ser conseguida, através dum maior planeamento no preenchimento das descrições dos serviços, e também na especificidade e variedade dos serviços oferecidos pelos aplicativos de saúde.

Capítulo 5

Conclusões

Este último capítulo apresenta uma síntese do trabalho reportado neste documento, sumariando as conclusões. São também apresentadas perspectivas de desenvolvimento futuro.

5.1 - Síntese do trabalho desenvolvido

- A primeira parte do trabalho centrou-se na realização dum estudo inicial dos paradigmas requeridos ao sistema, de forma a perceber as suas características, metodologias de desenvolvimento e potencialidades.
- De seguida, foram definidos os requisitos exigíveis a uma plataforma deste tipo, a partir de alguns dos problemas logísticos e funcionais encontrados nas infra-estruturas já implementadas até à data.
- Após a delineação dos requisitos, foi desenvolvida a análise e arquitectura do sistema multi-agente, através da metodologia Gaia.
- Por último, foi efectuada a implementação do protótipo, comprovando-se que o seu uso aporta maior dinamismo e celeridade na entrega de serviços em ambientes complexos de Saúde.

Em termos gerais, o objectivo principal deste trabalho foi concluído, e a arquitectura especificada foi implementada num protótipo funcional, com a sua API devidamente comentada com Javadoc, e fazem-se alguns comentários no código onde nos pareceu mais relevante.

Apesar de esta ser uma primeira abordagem a uma plataforma de middleware, capaz de proporcionar a aplicativos um mecanismo automático e dinâmico de entrega de Serviços, já se percebe o seu grande potencial e vantagens. Destas, é de realçar a atenuação das limitações dos aplicativos (a sua complexidade), sentidas pelos utilizadores, e traz em certa medida a noção de ubiquidade a esses mesmos aplicativos.

Com o uso de SOA foi possível alcançar o desacoplamento entre os Agentes do Sistema Multi-Agente.

Quanto aos agentes, foram implementados para serem do tipo reactivos, ou seja, reagirem a alterações no ambiente ou a mensagens de outros agentes.

O conjunto destes agentes forma um sistema pró-activo, pela capacidade de antecipar as necessidades do utilizador, isto é, age autonomamente, tomando a decisão de preparar e seleccionar uma lista de serviços que melhor se adeqúem ao utilizador, disponibilizando tal conteúdo antecipadamente ou tendo-o prontamente disponível, no caso de o utilizador o requerer.

Um cuidado a ter com a pro-actividade duma aplicação, como faz referência [Sat01], deve-se ao facto de o utilizador poder ser perturbado com aplicações que possuam um elevado nível de pró-actividade, desencadeando acções às quais o utilizador não esteja receptivo.

Maior precisão de decisão da plataforma em seleccionar os serviços que melhor se adaptem ao actual contexto do indivíduo pode ser alcançada, através da redundância de descrições de serviços com diferentes graus de especificidade. Outra forma, não muito viável mas passível de conseguir maior precisão, passa pelo simples aumento da estrutura de informação, que tem por alicerce o Conhecimento do Agente *Informer* (secção 4.2.3.4), e pelo aumento dos detalhes nas descrições dos serviços armazenados no respectivo Repositório, acedido pelo Agente *Discoverer* (secção 4.2.4.4).

Uma das conclusões, retiradas do uso da tecnologia RFID para serviços baseados na localização em recintos interiores, é que a medição da potência dos sinais RF continua a desafiar os *designers* a desenvolverem modelos práticos. O efeito de trajectórias múltiplas e as atenuações do sinal ainda se mantêm não totalmente controláveis.

O derradeiro objectivo da computação ubíqua é ser capaz de servir os utilizadores em qualquer lugar, a qualquer hora. No entanto, tendo em conta a natureza dinâmica das necessidades do utilizador e as situações decorrentes da vida do dia-a-dia, é um trabalho árduo, mas desafiador e ao mesmo tempo entusiasmante.

5.2 - Desenvolvimento futuro

O objectivo do trabalho apresentado foi essencialmente atingido. Constata-se, todavia, que existe ainda trabalho a desenvolver, para que a plataforma se aproxime do conceito ideal de sistema multi-agente baseado em SOA para serviços de informação, em ambientes ubíquos de Clínicas de Saúde.

Considera-se que o sistema poderá beneficiar da introdução duma linguagem semanticamente mais rica, para descrever as instâncias do domínio sobre o qual o matching é realizado. Resultaria mais favorável fazer uso de meta-informação, descrevendo a semântica dos objectos (locais, pessoas e outros), as suas propriedades e as relações existentes entre eles. Depois, seria necessária a aplicação de algoritmos de matching que tenham em consideração principalmente a relação de hierarquia entre conceitos no domínio de conhecimento. Os resultados de matching devem, pois, ser classificados de acordo com alguma estruturação de categorias, bem como ordenados para cada uma das categorias propostas. Assim, neste sistema, um aprofundamento da engenharia do conhecimento poderá resultar numa maior abrangência da sua utilização.

Um possível acrescento ao sistema, que ficou implícito no anteriormente referido, seria inserir a relação

Indivíduo (papel)	<-> relação	Indivíduo (papel)	para além do existente ->	Indivíduo (papel)	<-> relação	Local (função)

Uma outra possibilidade seria gerir as selecções efectuadas pelo utilizador - salvando onde, quem e quais os serviços escolhidos pelo utilizador, tendo em conta o número de vezes que esses serviços foram seleccionados, e em que variedades de ambientes foram preferenciais, trabalhando esta informação num contexto probabilístico, em constante actualização; aumentar-se-ia deste modo o grau de certeza do sistema.

Referências

[Abo00] Abowd, G. D., E. D. Mynatt "Charting past, present, and future research in ubiquitous computing." ACM Transactions on Computer-Human Interaction, Vol. 7, No. 1, Março 2000.

[age09] AgentTool. Disponível em http://eprints.agentlink.org/view/type/software.html], Acesso em Junho 2009.

[Ana09] Anacom. Disponível em [http://www.anacom.pt/streaming/qnaf2007.pdf?categoryId=248605&contentId =505624&field=ATTACHED_FILE] , Acesso em Junho 2009.

[Aur09] Aura. Disponível em [www.cs.cmu.edu/~aura]. Acesso em Janeiro 2009

[Bah00] Paramvir Bahl e Venkata N. Padmanabhan. “Enhancements to the RADAR user location and tracking system”, Microsoft Research, Redmond, WA, 2000.

[Ban02] Guruduth Banavar, Abraham Bernstein. “Software infrastructure and design challenges for ubiquitous computing applications”. In Commun. ACM 45(12):92-96, 2002.

[Bar00] Barry Brumitt, Brian Meyers, John Krumm, Amanda Kern, and Steven Shafer, “Easyliving: Technologies for intelligent environments” Technical report, Microsoft, 2000.

[Ben05] Bento, C. and Peixoto, J. and Veloso, M., "A Case-Based Approach for Indoor Location", in Proc. of the Sixth International Conference on Case-Based Reasoning, Lecture Notes in Artificial Intelligence series, Chicago, Illinois (USA), Agosto 2005

[Ben07] Bento, C. and Costa, M. and Batista, B. and Veloso, M., "A Study on the Suitability of GSM Signatures for Indoor Location", in Proc. of the the European Conference on Ambient Intelligence, European Conference on Ambient Intelligence, Darmstadt, Germany., Novembro 2007

[Bre04] Bresciani, P. et al. “TROPOS: An Agent-Oriented Software Development Methodology”. Journal of Autonomous Agents and Multi-Agent Systems, 2004

[Buh03] BUHLER, P.; VIDAL, J. M.; VERHAGEN, H., “Adaptive Workflow = Web Services + Agents”, Proceedings of the International Conference on Web Services, 2003.

[Cab06] G. Cabri, F. De Mola, N. Muratori, R. Quitadamo, F. Zambonelli, "UbiMedic: a Ubiquitous Computing Solution for Medical Emergencies", 2nd International Workshop on Multi-Agent Systems for Medicine, Computational Biology, and Bioinformatics 2006.

[Cac06] Caceres, C.; Fernandez, A.; Ossowski, S.; Vasirani, M., “Agent-Based Semantic Service Discovery for Healthcare: An Organizational Approach”, Intelligent Systems, 2006.

[Cai01] Caire, G. et al. “Agent Oriented Analysis Using Message/UML”. Em: Agent-Oriented Software Engineering, 2001.

[Cas95] Castelfranchi, C., “Guarantees for Autonomy in Cognitive Agent Architectures. Em: Intelligent Agents: Theories, Architectures and Languages”, Springer-Verlag, 1995.

[Cob09] Cobra, [http://cobra.umbc.edu/]. Acesso em Jnho 2009.

[Col98] Collis, J. C. et al., “The ZEUS Agent Building Toolkit”. BT Technology Journal, 1998.

[Del01] DeLoach, S. A. et al., “Multiagent Systems Engineering”. International Journal of Software Engineering and Knowledge Engineering, 11(3): pags. 231-258., 2001.

[Dem01] Demazeau, Y. et al., “Systems Development as Societies of Agents”, 2001.

[Doc03] Rios, D. and Dockhorn Costa, P. and Guizzardi, G. and Ferreira Pires, L. and Goncalves Filho, J., “Using Ontologies for Modeling Context-Aware Services Platforms”, Em OOPSLA 2003 Workshop on Ontologies to Complement Software Architectures, 2003.

[Don07] Chun-Dong Wang; Xiu-Feng Wang, “Multi-agent Based Architecture of Context-aware Systems”, Multimedia and Ubiquitous Engineering, 2007.

[Dur89] Durfee,H., et al.,”Negotiating tasks decomposition and allocation using partial global planning”, Distributed Artificial Intelligence (Vol.2), 229 - 243, 1989.

[Elf04] Elfatatry, A. and Layzell, P., “Negotiating in Service-Oriented Environments”, Communications of the ACM, 2004.

[End04] Endrei, M. et al., “Patterns: Service-Oriented Architecture and Web Services”, IBM International Technical Support Organization. Redbooks, Abril, 2004.

[Erl08] Thomas Erl, “SOA Principles of Service Design”, Pearson Education, Inc. 2008

[Fon05] José Fonseca, André Mora, Ana Celeste Marques, “MAMIS – A Multi-Agent Medical Information System – Um sistema multi-agente de informação médica”, Jornadas de Engenharia Electrónica e Telecomunicações e de Computadores Nov 2005.

[Gat09] Gatech, [http://www.cc.gatech.edu/fce/contexttoolkit/], Acesso em Janeiro 2009

[Gee05] D. Geer, “Will Binary XML Speed Network Traffic?" IEEE Computer, vol. 38, no. 4, pp. 16-18, 2005.

[Gen94] Genesereth, M. R. and Ketchpel, S. P., “Software Agents”. Em: Communications of the ACM,1994.

[Gio04] Giorgini, P. et al., “The Tropos Methodology: an overview”, 2004

[Glu99] Glushko, R. J.; Tenenbaum, J. M.; Meltzer, B., “An XML Framework for Agent-based E-commerce”, Communications of the ACM, 1999.

[Gri03] Griss, M. L.; Kessler, R. R. “Achieving the Promise of Reuse with Agent Component”, Proceedings of the Software Engineering for Large-Scale Multi-Agent Systems, 2003.

[Gru94] Grudin, J., "Groupware and Social Dynamics: Eight Challenges for Developers." Communications of the ACM 37(1): 92–105., 1994.

[Gru02] Grudin, J., "Group dynamics and ubiquitous computing: From “Here and Now” to “Everywhere and Forever”." Communications of the ACM 45(12): 74-78., 2002.

[GuT05] Gu, T., Pung, H. K., and Zhang, D. Q., “A service-oriented middleware for building context-aware services”, In Journal of Network and Computer Applications, Jan 2005.

[Had96] Haddadi, A. y Sundermeyer, K., “Foundations of Distributed Artificial Intelligence”, cap. Belief-Desire-Intention Agent Architectures, pags. 169-186. Wiley-Interscience, 1996.

[Han06] Byung-Mo Han Seung-Jae Song Kyu Min Lee Kyung-Soo Jang Dong-Ryeol Shin, “Multi-agent system based efficient healthcare service”, in Advanced Communication Technology, 2006.

[Har93] Andy Harter, Frazer Bennett. “Low bandwidth infra-red networks and protocols for mobile communicating devices”. ORL 93.5, Olivetti Reaserch, Cambridge, UK, 1993.

[Har99] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. “The anatomy of a context-aware application”. In Proceedings of the 5th Annual ACM International Conference on Mobile Computing and Networking (MobiCom 1999), pages 59-68, Agosto 1999.

[Ibm05] “IBM SOA Foundation: providing what you need to get started with SOA”. Service oriented architecture solutions. White Paper, 2005.

[Ich03] Ichiro Satoh, “SpatialAgents: integrating user mobility and program mobility in ubiquitous computing environments”, Wireless Communications and Mobile Computing, 2003.

[Isl03] Islam, N. and M. Fayad “Toward Ubiquitous Acceptance of Ubiquitous Computing”. Communications of the ACM. 47: 89-92. (2003).

[Jen98] Jennings, N. R., Sycara, K. and Wooldridge, M. “A Roadmap of Agent Research and Development”. Int. Journal of Autonomous Agents and Multi-Agent Systems, 1 (1). pp. 7-38., 1998.

[Jes02] Jessup, M. L. and D. Robey “The relevance of social issues in ubiquitous computing environments”. Communications of the ACM. 45: 88-91, 2002.

[Kag02] L. Kagal, V. Korolev, S. Avancha, A. Joshi, T. Finin, and Y. Yesha, “Centaurus: an infrastructure for service management in ubiquitous computing environments”, Wireless Networks vol. 8, Issue 6, November 2002.

[Kal02] Kalle Lyytinen and Youngjin Yoo, “Issues and challenges in ubiquitous computing”, Communications of the ACM, Dezembro 2002.

[Kha04] Khalil El-Khatib, Zhen E. Zhang, N. Hadibi, Gregor von Bochmann, “Personal and service mobility in ubiquitous computing environments”, Wireless Communications and Mobile Computing, 2004.

[Kim07] Nam-Ho Kim; Yi-Seok Jeong; Sang-Hwan Ryu; Dong-Ryeol Shin, “Mobile Healthcare System based on Collaboration between JADE and OSGi for Scalability”, Multimedia and Ubiquitous Engineering, 2007.

[Kin96] Kinny, D. et al., “A Methodology and Modelling Technique for Systems of BDI Agents”. En Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, tomo 1038 de LNCS, pags. 56-71, 1996.

[Lin03] Lindwer et al, “Ambient Intelligence Vision and Achievements: Linking Abstract Ideas to Real-World Concepts”, Proceedings of The Design, Automation and Test in Europe Conference, 2003

[Lup07] Emil Lupu, Naranker Dulay, Alberto Schaeffer Filho, Sye Keoh, Morris Sloman, Kevin Twidle, “AMUSE: Autonomic Management of Ubiquitous e-Health Systems”, Journal Article Concurrency and Computation: Practice and Experience Wiley DOI, 2007.

[Mae96] Maes, Pattie, Intelligent “Software: Programs That Can Act Independently Will Ease the Burdens that Computers Put on People”, IEEE Expert Systems, Dezembro de 1996.

[Mae97] Maes, P., “Humanizing the Global Computer”, In: IEEE Internet Computing, 1997.

[Mel98] Meltzer, B.; Glushko, R., “XML and electronic commerce: enabling the network economy”, SIGMOD Rec. 27, Dez. 1998.

[Mic06] “Microsoft Corporation Homepage,” http://www.microsoft.com/ Acesso em Junho 2009.

[Nis00] Nissanka B. Priyantha, Anit Chakraborty, Hari Balakrishnan, “The Cricket location-support system”, Proceedings of the 6th annual international conference on Mobile computing and networking, p.32-43, Agosto 06-11, 2000.

[Nwa96] Nwana, H. S. and Wooldridge, M., “Software Agent Technologies” Em: British Telecom Technology Journal, 1996.

[Pav04] Pavlovski, C.; HeeSam Kim; Wood, D., “Medical Information Systems: The Digital Hospital”, Ubiquitous mobility in clinical healthcare, IDEAS '04-DH. Proceedings, Set. 2004.

[Per06] Pereira Filho, J. G. ; Pessoa, R. M. ; Calvi, C. Z. ; Oliveira, N. Q. ; Barbosa, A. C. P. ; Farias, C. R. G., “Infraware: um Middleware de Suporte a Aplicações Móveis Sensíveis ao Contexto”, Em: SBRC - Simpósio Brasileiro de Redes de Computadores, 2006.

[Rom02] M. Román, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt, “A middleware infrastructure for active spaces”, IEEE Pervasive Computing, vol. 1, no. 4, pp. 74– 83, Out-Dez 2002.

[Rus98] Russell, D. M., M. Weiser, “The future of integrated design of ubiquitous computing in combined real & virtual worlds”. CHI 98 conference summary on Human factors in computing systems, Los Angeles, CA, United States, 1998.

[Rxt09] http://rxtx.qbang.org/wiki/index.php/Main_Page, acedido em Junho de 2009

[Sal08] Salvador, Z.; Larrea, M.; Lafuente, “Smart Environment Application Architecture”, A. Pervasive Computing Technologies for Healthcare, 2008.

[Sat01] M. Satyanarayanan, “Pervasive computing: Vision and challenges”, IEEE Personal Communications, 2001.

[Sav01] Andreas Savvides, Chih-Chieh Han, and Mani B. Srivastava. “Dynamic fine grained localization in ad-hoc wireless sensor networks”. In Proceedings of the 7th Annual ACM International Conference on Mobile Computing and Networking (MobiCom 2001), pages 166-179, Julho 2001.

[Sch00] Schmidt, A., "Implicit Human Computer Interaction Through Context." Personal Technologies 4(2&3): 191-199, 2000.

[Son07] Jae-gu Song; Kirn, S.; Gil-cheol Park, “Towards a Mobile and Context-Aware Technology Based Healthcare System”, Frontiers in the Convergence of Bioscience and Information Technologies, 2007.

[Soo05] Soo-Joong Ghim, Yong-Ik Yoon, Ilkyeun Ra, “A Component-Based Adaptive Model for Context-Awareness in Ubiquitous Computing”, IFIP International Symposium on Network-Centric Ubiquitous Systems, 2005.

[Sud07] Sudha, R.; Rajagopalan, M.R.; Selvanayaki, M.; Selvi, S.T., “Ubiquitous Semantic Space: A context-aware and coordination middleware for Ubiquitous Computing”, Communication Systems Software and Middleware, 2007.

[Tat04] Tatsuo Nakajima, Kaori Fujinami, Eiji Tokunaga, Hiroo Ishikawa, "Middleware design issues for ubiquitous computing", Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia, 2004.

[Tom09] Apache Tomcat [http://tomcat.apache.org/], Acesso em Junho 2009

[W3c04] "Web Services Architecture," http://www.w3.org/TR/ws-arch/ Acesso em Junho 2009

[Wan08] Wang, K.I.-K.; Abdulla, W.H.; Salcic, Z.; DeSouza, N.; Ramkumar, S., "Multiagent control system with mobile ubiquitous platform for ambient intelligence Intelligent Environments", 4th International Conference, 2008.

[Wan92] Roy Want, Andy Hopper, Veronica Falcao, and Jon Gibbons. "The active badge location system". ACM (TOIS), 10(1):91-102, Janeiro 1992.

[War97] Andy Ward, Alan Jones, and Andy Harper. "A new location technique for the active Office". IEEE Personal Communications, 4(5):42-47, October 1997.

[Wei91] Mark Weiser, "The computer for the 21st century", Scientific American, 1991.

[Wei93] Mark Weiser, "Hot topics: Ubiquitous computing", IEEE Computer, Out 1993.

[Wss01] G.Weiss., "Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence" The MIT Press, 2001

[Woo95] Wooldridge, M. and Jennings, N. R., "Intelligent Agents: Theory and Practice", 1995. Disponível em http://www.csc.liv.ac.uk/~mjw/pubs/ , Acesso em Junho 2009

[Woo00] M. Wooldridge, N. R. Jennings, and D. Kinny. The Gaia Methodology for Agent-Oriented Analysis and Design. In Journal of Autonomous Agents and Multi-Agent Systems. 3(3):285-312. 2000.

[Xer05] Xerox. Palo alto research center. http://www.parc.com/, Acesso em Junho 2009

[Xst09] http://xstream.codehaus.org/ Acesso em Junho 2009

Anexo A

Aqui são apresentados os diagramas de classe completos dos vários Agentes.

A.1 – Agente IntegratorLocative

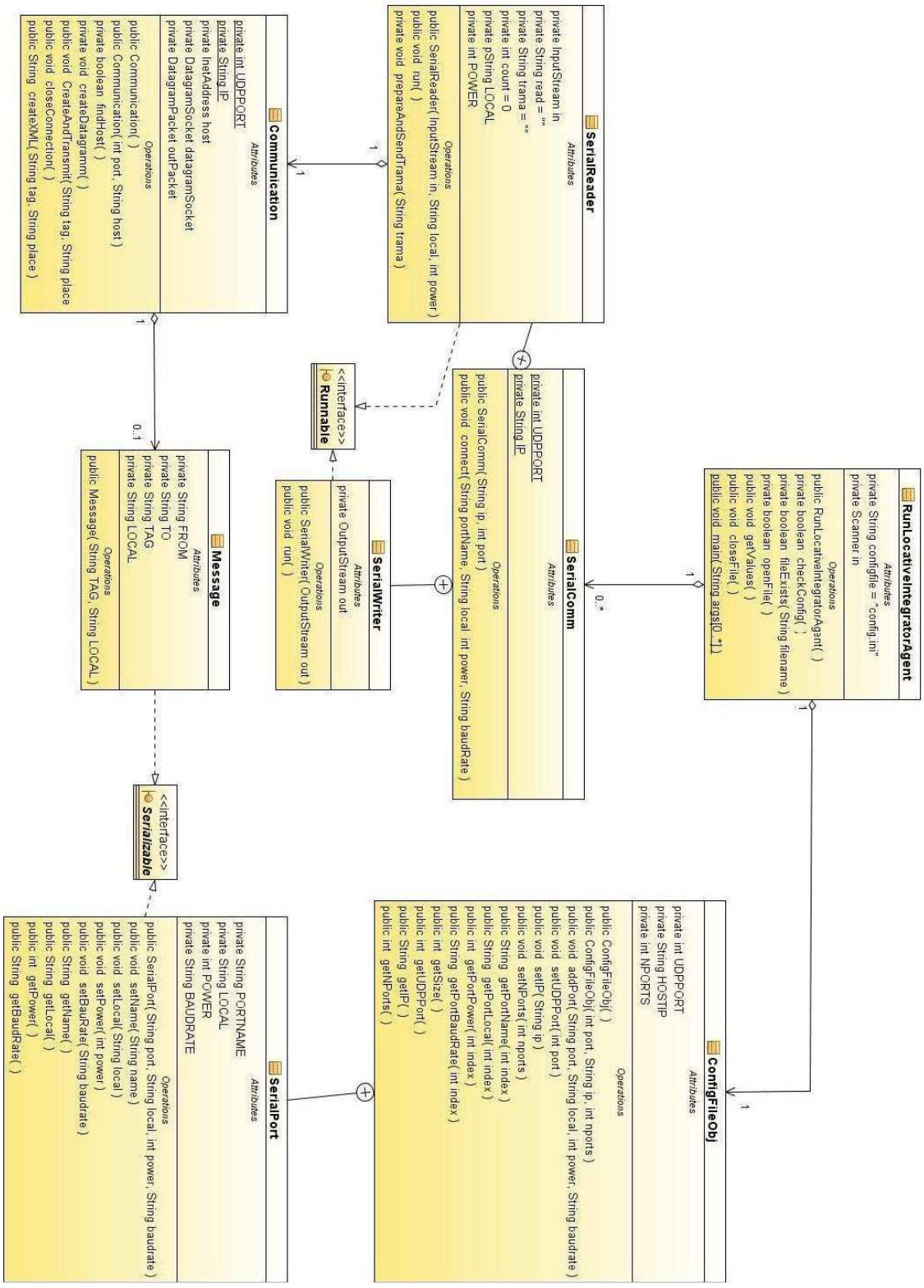

Figura A. 1 – Diagrama de Classes do Agente IntegratorLocative

A.1 – Agente Displacer

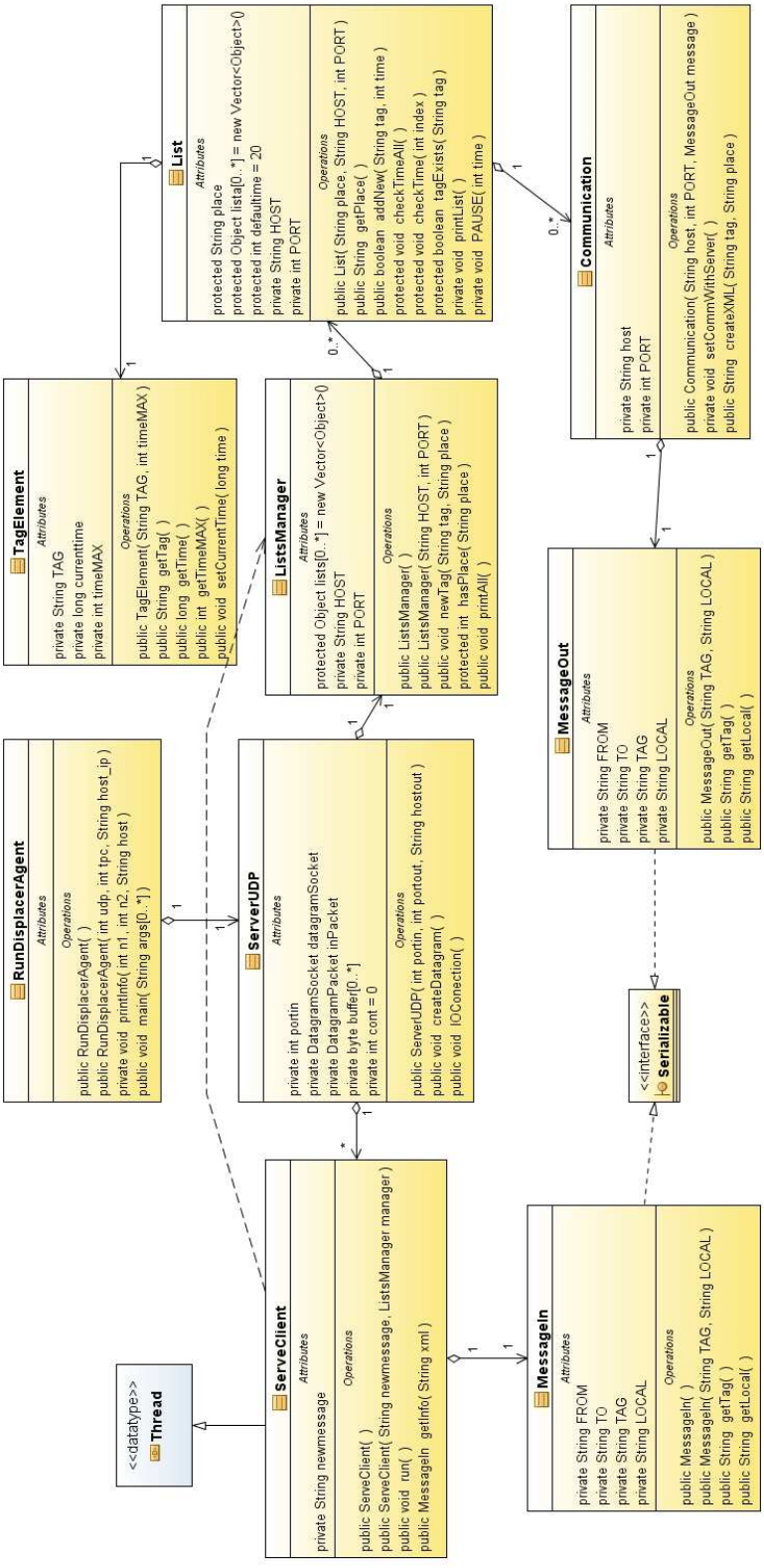

Figura A. 2 – Diagrama de Classes do Agente Displacer

A.1 – Agente Informer

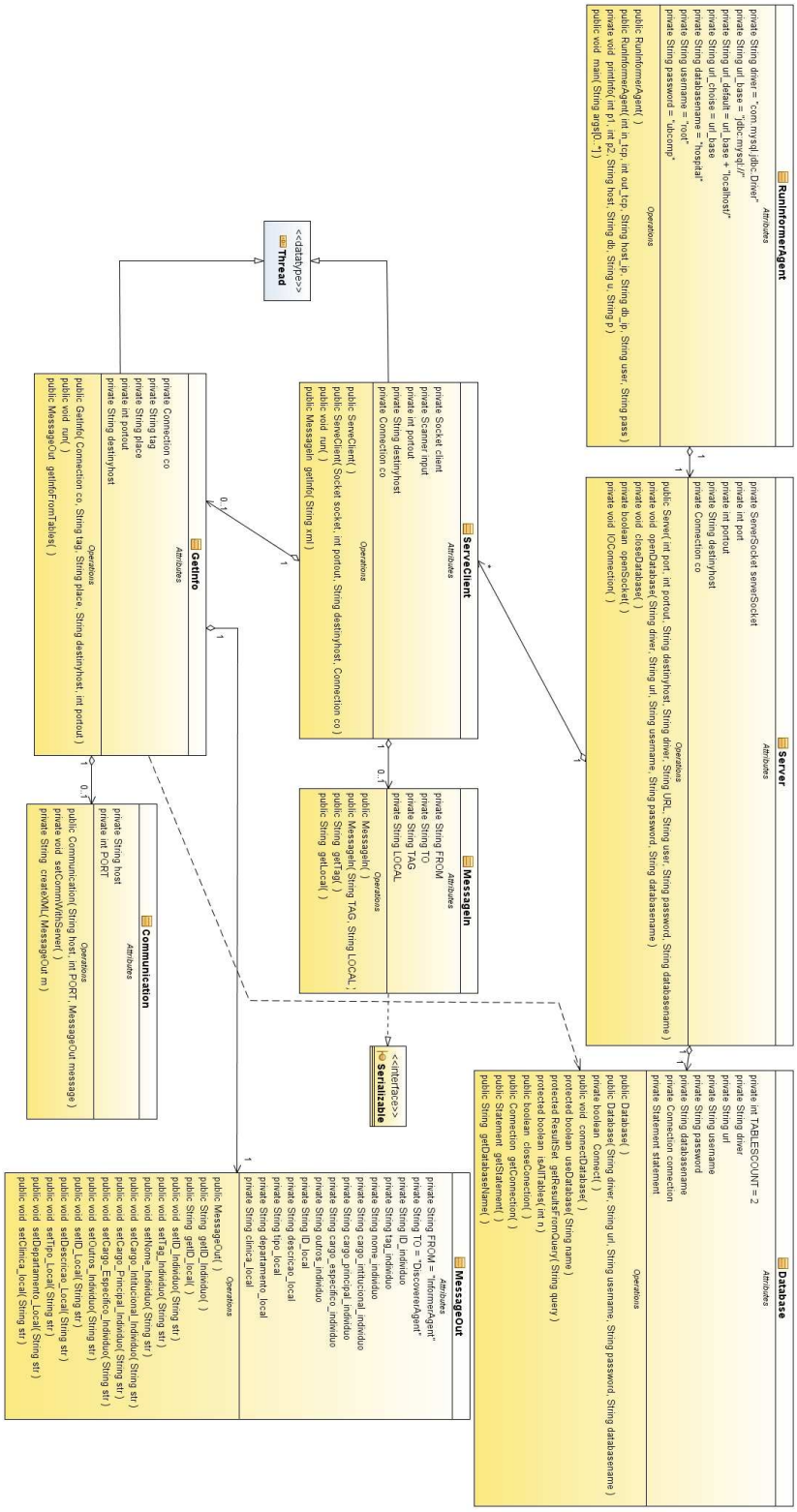

Figura A. 3 – Diagrama de Classes do Agente Informer

A.1 – Agente Discoverer

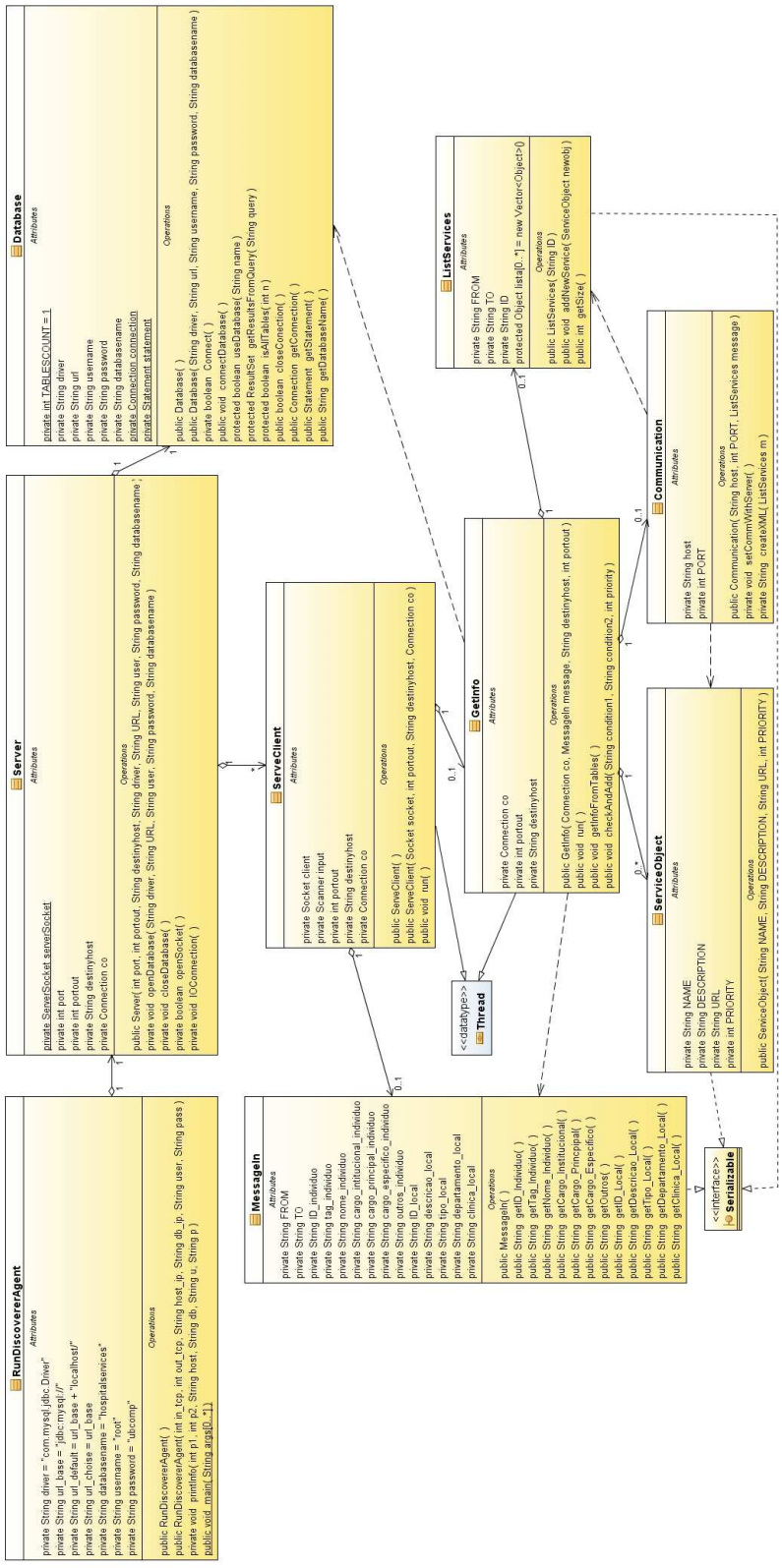

Figura A. 4 – Diagrama de Classes do Agente Discoverer

A.1 – Agente UserWizard

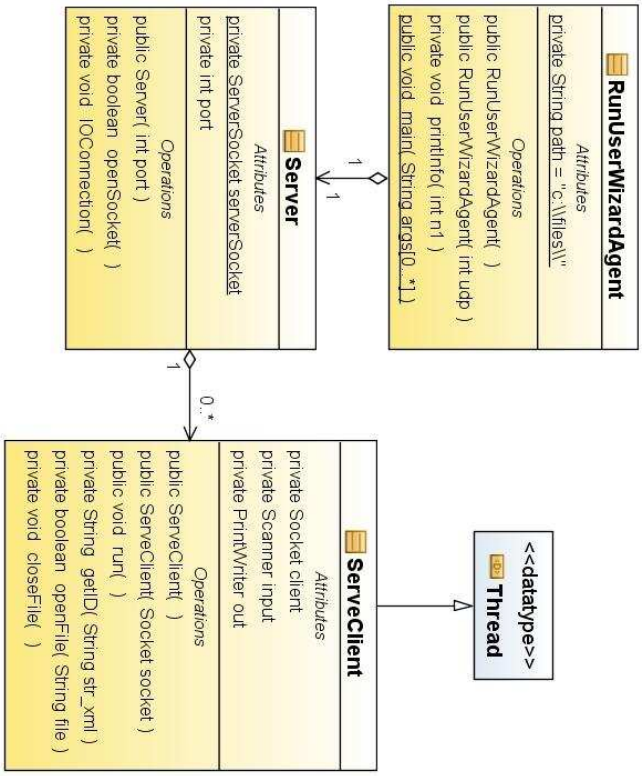

Figura A. 5 – Diagrama de Classes do Agente UserWizard