Dissertação de Mestrado em Engenharia Informática

Computação Móvel

Proposta de solução para a Interoperabilidade de Sistemas na plataforma Elder Care

Samuel Marcelino Brás

Leiria, 2011

Dissertação de Mestrado em Engenharia Informática

Computação Móvel

Proposta de solução para a Interoperabilidade de Sistemas na plataforma Elder Care

Samuel Marcelino Brás

Dissertação de Mestrado realizada sob a orientação do Professor Doutor Rui Rijo,

Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria

Leiria, 2011

À Minha Família

Agradecimentos

Ao Professor Doutor Rui Rijo pela sua orientação, disponibilidade, suporte, apoio e compreensão, sem o qual seria impossível concluir esta dissertação com sucesso.

Agradeço também ao Professor Doutor António Pereira, pelo apoio, empenho e critica construtiva demonstrada no decorrer deste projecto.

Agradeço também ao grupo de trabalho de Centro de Investigação em Informática e Comunicações (CIIC) do Instituto Politécnico de Leiria (IPL) pelo apoio, disponibilidade sempre demonstrados. Em especial ao Eng. David Bastos pelo auxílio prestado no desenvolvimento do projecto.

Agradeço ainda a todos os docentes da Escola Superior de Tecnologia e Gestão (ESTG) que de alguma forma contribuíram para que este projecto chegasse a bom porto.

A todos aqueles que de alguma forma contribuíram para a realização deste projecto.

À Sofia que esteve sempre presente nos bons e maus momentos e que esses se repitam por muitos anos.

Por fim e não menos à minha família, Elio, Gorete e Carina, a todo o apoio, compreensão e paciência que recebi durante a realização desta dissertação.

A todos, os meus sinceros agradecimentos!

Nota Prévia

A presente dissertação foi realizada no seio do Centro de Investigação em Informática e Comunicações (CIIC) do Instituto Politécnico de Leiria (IPL) integrando o esforço de investigação do projecto Elde Care.

Do trabalho efectuado nesta dissertação resultou uma publicação:

- Samuel Brás, Rui Rijo, David Bastos, and António Pereira. 2011. *Information Management, proposal for an integration platform using metadata. In CENTERIS 2011, Part III, CCIS 221 proceedings.* Vol. 221. Vilamoura, Portugal: Springer.

Resumo

A população mundial regista um acentuado e acelerado envelhecimento ao mesmo tempo que os recursos humanos, materiais e financeiros necessários para prestar cuidados adequados aos idosos são cada vez menores. A população idosa requer, entre outros, cuidados continuados, e os seus familiares têm dificuldade na resposta a essas necessidades. Com o intuito de ajudar a ultrapassar essas dificuldades, decorre um enorme esforço de investigação orientado ao desenvolvimento de vários sistemas que têm como objectivo o bem-estar biopsicossocial dos idosos. No entanto, para que estes sistemas possam trabalhar como um todo coerente e eficaz é necessária uma plataforma integradora. Não foi identificada uma plataforma que conseguisse realizar a interligação de sistemas heterogéneos e que possibilitasse a interligação de todos estes sistemas.

Neste documento é apresentado o estudo inicial realizado, a arquitectura resultante e o inicio do desenvolvimento do protótipo, de uma plataforma completa de integração de sistemas, para aplicações que abrangem diversas áreas do bem-estar biopsicossocial. Esta plataforma possibilita também o armazenamento de dados, a recepção de eventos e o tratamento de alertas.

No que toca a integração de sistemas, um dos grandes desafios é encontrar uma solução que consiga partilhar o conhecimento que cada sistema representa. A abordagem utilizada como tentativa de resolução deste desafio, consiste numa ontologia por permitir definir um domínio conceptual passível de ser partilhado e aumentado.

Para que o conhecimento seja disponibilizado, é necessário que cada sistema inseria o seu conhecimento na ontologia, seguindo um conjunto de regras fornecidas. A representação

do conhecimento é efectuado pelo seu conteúdo semântico, possibilitando a agentes inteligentes efectuar as pesquisar e cruzar informação.

Cada sistema armazena dados nesta plataforma, através da criação de uma estrutura de dados personalizada. Esta estrutura é criada com recurso a metadados, sendo única para cada sistema. Os dados recolhidos por cada sistema vão compor um conjunto de informação relevante para cada idoso.

O problema da monitorização tem sempre associado o problema da notificação. Os sistemas de monitorização podem utilizar um serviço de lançamento de eventos, que desencadeia uma acção de validação, que pode transformar o evento num alerta. Quando um alerta é recebido, são efectuadas notificações com base no tipo de alerta que foi recebido.

Com a implementação da arquitetura apresentada é possível disponibilizar um sistema de integração de plataformas, que armazena dados utilizando metadados e que consiga partilhar o conhecimento pelo seu conteudo semântico, utilizando ontologias.

Palavras-chave: gerontecnologia, interoperabilidade de sistemas, ontologia, idoso, metadados, monitorização, notificação

Abstract

The world population is showing a marked and accelerated aging at the same time that the human, material and financial resources needed to maintain adequate care for the elderly are beginning to become too scarce. The elderly population requires, among other things, continued care, and their relatives have difficulty addressing those needs. With the intent of helping to overcome those difficulties, there is a concerted investigation effort for the development of several systems, which have as its purpose the elderly biopsychosocial well-being. However so that these systems become able to work as single coherent and efficient whole it is necessary an integrating platform. A platform that would be able to interconnect heterogeneous systems and allowed the interconnection of all these systems was not identified.

In this document is presented the initial study performed, the resulting architecture and the initial development of a prototype, of a complete integration platform for systems integration, for applications that covering multiple areas of the biopsychosocial well-being. This platform also allows data storage, reception of events and the treatment of alerts.

As for systems integration, one of the biggest challenges is to find a solution that can share the knowledge that each system represents. The approach used as an attempt to solve this challenge, consists of an ontology, for it allows defining a conceptual domain able to be shared and enriched.

To make this knowledge available it is necessary that every system insert their knowledge in the ontology following the supplied set of rules. The representation of knowledge is done by its semantic content, allowing intelligent agents to search and cross-reference information.

Each system stores data in this platform, through the creation of a custom data structure. This structure is created through the use of metadata, being unique for each system. The data gathered by each system will be used to assemble a relevant information set for each elderly.

The monitorization problem is always attached with the notification problem. Monitoring systems may use an event launcher service, which triggers a validation action, which can transform an event into an alert. When an alert is received, notifications are sent depending on the type of alert received.

With the implementation of the architecture presented it is possible to make available a platform integration system, which stores data through the use of metadata and which can share knowledge by its semantic content, by using ontology's.

Key-words: Gerentotechnology, Systems Interoperability, Ontology, Elder, Metadata, Monitorization, Notification

Índice de Figuras

Índice de Quadros

Índice

Lista de Acrónimos

ANSI	***American National Standards Institute***
API	*Application Programming Interface*
AVD	Actividades da Vida Diária
BAN	*Body Area Network*
CAALYX	*Complete Ambient Assisted Living eXperiment*
CORBA	*Common Object Request Broker Architecture*
DAML	*DARPA agent markup language*
DCOM	*Distributed Component Object Model*
DR	Diagrama Relacional
DICOM	*Digital Imaging and Communications in Medicine*
ECC	*Error Correction Codes*
EDI	*Electronic Data Interchange*
ESB	Enterprise Service Bus
F-logic	*Frame Logic*
GA	Gestor de Alertas
GC	Gestor de Contactos

GI	Gestor de Informação
GPS	*Global Positioning System*
HL7	*Health Level Seven*
HTTP	*Hypertext Transfer Protocol*
ISBN	*International Standard Book Number*
LAN	*Local Area Network*
MAN	*Metropolitan Area Network*
OCML	*Open Configuration and Management Layer*
OIL	*Ontology Inference layer*
OKBC	*Open Knowledge Base Connectivity*
OMG	*Object Management Group*
OSI	*Open Systems Interconnection*
P2P	*Peer-to-Peer*
POO	Programação Orientada a Objectos
RC	Representação de conhecimento
RDF	*Resource Description Framework*
RFID	*Radio-Frequency Identification*
RIM	*Reference Information Model*
RMI	*Remote Method Invocation*
RPC	*Remote Procedure Call*
SHOE	*Simple HTML Ontology Extensions*

SOA	*Service-Oriented Architecture*
SOAP	*Simple Object Access Protocol*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
UML	*Unified Modeling Language*
W3C	*World Wide Web Consortium*
WAN	*Wide Area Network*
WCF	*Windows Communication Foundation*
WSDL	*Web Services Description Language*
WWW	*World Wide Web*
XML	*Extensible Markup Language*
XMLITS	XML *Implementation Technology Specification*
XOL	*Ontology Exchange Language*

1 Introdução

No ano corrente de 2011, estamos perante uma população cada vez mais envelhecida (United Nations, Department of Economics and Social Affairs et al. 2007) (Pedro Pita Barros e Jorge de Almeida Simões 2007). A idade da população activa tem aumentado, tornando a população idosa em membros activos da sociedade (Anthony Fleury, Michel Vacher et al. 2010). Contudo os idosos são mais propícios a doenças e a quedas, tornando-os mais vulneráveis fisicamente, requerendo uma maior monitorização (Liege Mata Álvares, Rosângela da Costa Lima et al. 2010). Outro aspecto da nossa sociedade, é a falta de tempo por parte dos membros activos da família para dar o apoio necessário aos seus elementos. Estes dois factores traduzem-se no facto de cada vez menos, as famílias terem a possibilidade de dar o acompanhamento necessário aos seus familiares (Martin Turcotte 2005), acompanhamento este que se traduz em auxílio nas actividades da vida diária (AVD), na vigilância dos problemas físicos que os idosos apresentam e na convivência necessária ao ser humano enquanto ser social (Upkar Varshney 2007).

Estes factores justificam a necessidade de sistemas que melhorem a qualidade de vida dos idosos. Um destes sistemas é o Elde Care (Marcelino 2008). O Elde Care abrange vários aspectos do bem-estar físico do idoso alargando o raio de acção à componente social e emocional. Na componente social e emocional propõe-se desenvolver um "espaço virtual" de convívio, aprendizagem e obtenção de serviços. Pretende ultrapassar os obstáculos que o teclado, o rato e as mensagens do sistema impõem com o desenvolvimento de novas abordagens de interacção humano - máquina. Sugere a existência de um centro de contactos devidamente adequado à realidade dos idosos. Deste modo propõe uma interacção humano - máquina simples e inteligente de acordo com as restrições inerentes à população idosa. Visa criar uma base de dados que agregue toda informação recolhida e que possa comunicar com

outras fontes de dados. Utiliza ainda mecanismos de reconhecimento de padrões, no sentido de identificar alterações às rotinas dos idosos que possam indiciar situações de perigo.

No que diz respeito à componente de monitorização propõe combinar tecnologias como redes de sensores sem fios e Identificação por Rádio Frequência (RFID). A monitorização permite detectar quedas do idoso, monitorizar a componente física envolvente como gás, incêndio e inundações. Pretende ainda introduzir a componente de qualidade de serviço e de segurança nestas redes de sensores sem fios.

As funcionalidades descritas são desenvolvidas em módulos separados, levando assim a uma arquitectura modular. Desta forma consegue-se dividir um problema complexo em vários mais simples. O projecto Elde Care é composto inicialmente pelos módulos: Monitorização Local, Sala de Convívio Virtual e Centro de Controlo (CC). Com a utilização da arquitectura modular, futuramente poderão ser adicionados novos módulos que acrescentem mais-valias ao projecto.

A Monitorização Local consiste em três sub-módulos: a Monitorização Corporal (*Body Monitor*), o Localizador Pessoal (*Personnel Locator*) e a Interface de Comandos Facilitada (*Easy Interface Commander*). A monitorização corporal recolhe os dados dos sensores, utilizados para monitorizar o idoso. O Localizador Pessoal, verifica se o idoso está perdido, sugerindo-lhe nesse caso, um caminho para uma localização próxima conhecida. A Interface de comandos Facilitada, consiste numa interface visual, que possibilite ao idoso interagir com o sistema Elde Care.

A sala de Convívio Virtual consiste em dois sub-módulos: o Centro de Comunicação (*Communication Center*) e o Controlo Vocal (*Voice Commander*). O Centro de Comunicação permite aos idosos comunicar por vídeo-conferência, chamada telefónica ou mensagens instantâneas. O Controlo Vocal permite ao idoso interagir com a aplicação através de voz.

O Centro de Controlo consiste em três sub-módulos: o Gestor de Informação (*Information Management*), o Gestor de Alertas (*Alert Management*) e o Gestor de Contactos (*Contact Management*). O Gestor de Informação (GI) gere toda a informação proveniente dos diversos módulos a ele acoplado. O Gestor de Alertas (GA) gere todos os alertas e eventos dos diversos módulos. O Gestor de Contactos (GC) gere a lista de contactos do idoso. A organização dos módulos e a forma como eles vão comunicar são ilustradas na Figura 1.

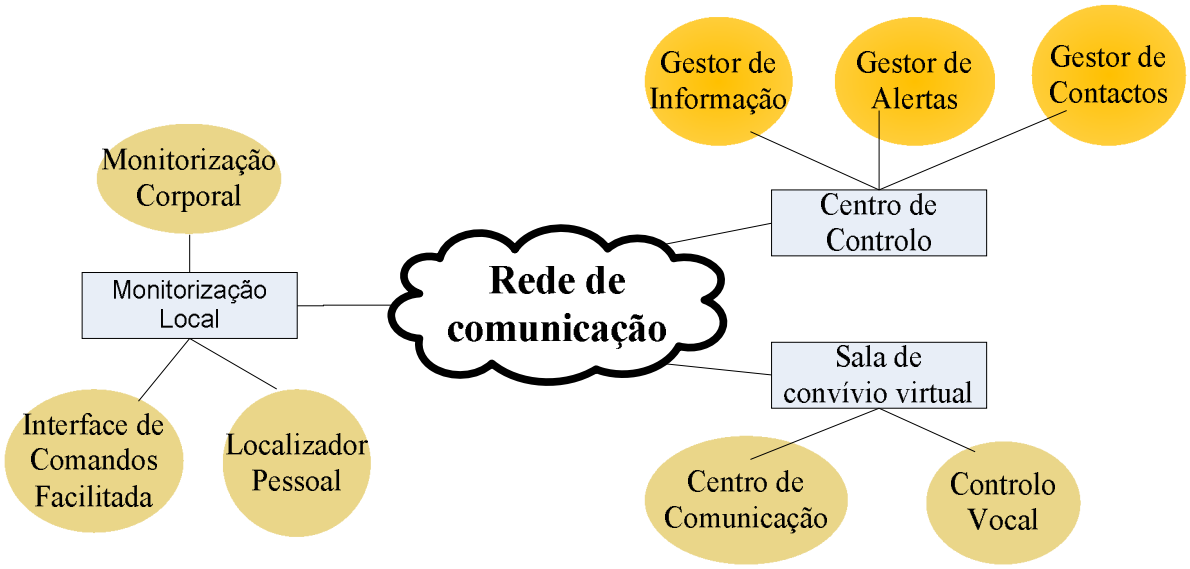


Figura 1 - Interacção entre os módulos do Elde Care. Adaptado de Information Management, proposal for an integration platform using metadata (Samuel Brás, Rui Rijo et al. 2011).

Este projecto focou-se no módulo centro de controlo, tendo sido o seu maior foco no sub-módulo GI. As suas funcionalidades vão ser descritas com maior detalhe na secção seguinte. Nesta secção foi identificado um obstáculo que afecta a população idosa, apresentando um projecto que pretende contribuir para encontrar alguns caminhos de melhoria desses problemas. Foi ainda feita uma breve descrição da composição e das funcionalidades de cada módulo pertencentes ao Elde Care. Para finalizar esta secção, foi identificado que o foco deste projecto é no sub-módulo Gestão de Informação.

1.1 Âmbito do projecto

O âmbito deste projecto engloba os três sub-módulos que constituem o centro de controlo: o Gestor de Informação (GI), o Gestor de Alertas (GA) e o Gestor de Contactos (GC).

O GI tem como objectivo interligar os diversos sub-módulos. No âmbito deste projecto, o GI irá permitir interligar a monitorização corporal e o Localizador pessoal. O GI deverá ainda permitir a comunicação de novos sub-módulos ainda não existentes nem especificados,

respeitando um conjunto de regras simples. Poderá permitir também a cada sub-módulo criar a sua estrutura de dados de forma a armazenar os dados pretendidos.

O GA tem como objectivo receber eventos despoletados pelos sub-módulos externos que, após classificação, são colocados numa lista de alertas. Um alerta é a acção desencadeada após um evento ser classificado. Os alertas podem ser classificados em quatro níveis de prioridade, baixo, médio, alto e urgente. Os níveis baixos são situações fora do normal, que directamente não colocam em risco a saúde do idoso. O nível normal são situações fora do normal, que não exigem cuidados directos de profissionais de saúde. O nível alto são situações fora do normal, que não necessitem de cuidado imediato de profissionais de saúde. Deve ser comunicado a entidades profissionais para prevenção. Os níveis urgentes são situações que necessitam de cuidados imediatos por parte de profissionais.

O GC gere e disponibiliza uma lista de contactos, personalizável por idoso.

O principal objectivo deste projecto consiste na interligação de diversos sub-módulos externos através do GI entre os quais, o Localizador Pessoal e a monitorização corporal que vai gerar alertas sempre que seja adequado. O Localizador Pessoal vai registar a localização do idoso recorrendo as coordenadas do *Global Positioning System* (GPS) detectando se este se encontra em áreas consideradas seguras. A monitorização corporal detecta a posição em que se encontra o idoso (vertical ou horizontal), possíveis quedas, o ritmo cardíaco e a sua temperatura corporal superficial. Toda essa informação vai estar centralizada no GI que a vai disponibilizar para posterior análise por um sub-módulo classificador. Este sub-módulo irá determinar a existência de padrões ou de anomalias, e se necessário gerará um alerta. Os sub-módulos externos têm de proceder a um registo no GI, informando a estrutura de dados que pretendem e os dados que vão armazenar. O GI tem a função de informar qual a estrutura de dados que disponibiliza assim como a existência de um novo tipo de dados que esteja disponível, quando solicitado.

Aquando da análise dos dados, poderá ser necessário enviar alertas para diversas entidades de saúde ou pessoas, sendo que todos os contactos estão disponíveis através do módulo GC.

Nesta secção foi apresentado o âmbito deste projecto, identificando as principais funcionalidades dos três sub-módulos que compõem o Centro de Controlo (CC) o Gestor de Informação (GI), o Gestor de Alertas (GA) e o Gestor de Contactos (GC). Foi ainda

apresentado o principal objectivo deste projecto, que é de interligar diversos sub-módulos externos que têm como objectivo melhorar a qualidade de vida dos idosos. Para conseguir este objectivo existem questões que são levantadas e que são apresentadas na secção seguinte.

1.2 Questão de investigação

"A expansão do envelhecer não é um problema. É sim uma das maiores conquistas da humanidade. O que é necessário é traçarem-se politicas ajustadas para envelhecer são, autónomo, activo e plenamente integrado. A não se fazerem reformas radicais, teremos em mãos uma bomba relógio a explodir a qualquer altura." (Koffi Anan, 2002)

A questão de investigação subjacente ao presente trabalho de investigação é: Como permitir a comunicação entre os diversos módulos, de um sistema de apoio a idosos (Elde Care), permitindo a escalabilidade, a interoperabilidade e o conhecimento do conteúdo semântico da informação a trocar?

As aplicações de monitorização baseadas em sensores têm vindo a aumentar. Estas aplicações, conseguem fazer uma monitorização mais prolongada, pois acompanham o idoso ao longo do dia. Este tipo de monitorização é essencial para vigiar os idosos. Os dados recolhidos pela aplicação, na monitorização do idoso, não são cruzados com os dados recolhidos com outras aplicações de recolha de dados. Existe assim a necessidade de criar um *middleware,* que consiga agregar os dados recolhidos pelas diversas aplicações de sensores. Os dados recolhidos são tratados por uma aplicação, que vai ter disponível não só os dados recolhidos pela aplicação de sensores, mas também pelas outras aplicações ligadas ao idoso que está a ser monitorizado. É assim possível melhorar a monitorização efectuada aos idosos, pois a monitorização é realizada como um todo e não por aplicações separadas.

Para o desenvolvimento do *middleware* vai ser desenvolvido um protótipo constituído por três sub-módulos: Gestor de Informação, Gestor de Alertas e Gestor de Contactos.

O Gestor de Informação vai permitir autenticar as ligações provenientes do exterior. Permitir criar uma estrutura de dados, para armazenamento de informação proveniente de módulos

externos. Adicionar novos módulos sem serem necessárias novas configurações. Adição de novos sensores em tempo real. Permitir actualizar a estrutura de dados que foram previamente criadas. Receber dados dos módulos externos. Estruturar os dados recebidos para estruturas internas organizadas. Disponibilizar estes dados organizados para posterior análise.

O Gestor de Alertas vai permitir receber alertas. Reencaminhar os alertas, consoante o seu nível em tempo real. Configurar alertas automáticos criados pelo idoso.

O Gestor de Contactos vai permitir a criação de um perfil de utilizador. Armazenar uma lista personalizada de contactos por idoso. Disponibilização da mesma ao cliente e ao módulo de comunicação. Possibilitar pesquisa por parte do idoso de outros utilizadores do projecto.

Pretende-se que estes três sub-módulos interligados possam constituir a base de uma aplicação escalável, com o propósito de melhorar a qualidade de vida dos idosos.

Neste capítulo foi introduzido o âmbito subjacente ao presente trabalho. Foi apresentado o problema da interligação dos diversos módulos. Foram ainda definidas quais as características que os sub-módulos pertencentes ao centro de controlo devem de ter para solucionar o problema da interligação e do armazenamento de dados. O capítulo que se segue apresenta as principais tecnologias e conceitos existentes.

2 Revisão da literatura

Após ter contextualizado o tema e apresentada a questão de investigação é essencial conhecer os conceitos e trabalhos relevantes na área. De forma a perceber qual a evolução que decorreu ao longo do tempo na melhoria da qualidade de vida dos idosos, é apresentado o conceito de gerontecnologia e alguns sistemas que forma criados ao longo do tempo, com o intuito de melhorar o bem estar dos idosos. É apresentado o conceito de interoperabilidade de sistemas, e quais foram as condições que levaram o seu desenvolvimento. São ainda mostradas as tecnologias utilizadas, de forma a resolver problemas de interoperabilidade. Como estamos perante um sistema que pretende melhorar a qualidade de vida dos idosos, foi estudado o protocolo de comunicação existente na área da saúde, o *Health Level Seven* (HL7). É dado a conhecer como surgiu, quais são os seus objectivos e como evoluiu de forma a poder responder as necessidades actuais. Para iniciar o estudo desta arquitectura é essencial conhecer os princípios da arquitectura orientada a serviços (SOA), pois permitem seguir uma metodologia que atinge a interoperabilidade. Para concluir, é apresentado o conceito de ontologia, algumas linguagens de ontologias, alguns editores de ontologias e o que são sistemas inteligentes.

2.1 Gerontecnologia

Gerontologia é o estudo do processo de envelhecimento. O termo, que foi introduzido por Élie Metchnikoff em 1903, refere-se ao estudo do processo de envelhecimento fisiológico e psicológico não só do homem, mas de todos os seres vivos (Clarisse Odebrecht M.Sc., Luciana de Oliveira Gonçalves et al. 2010). A gerontecnologia reúne duas áreas do saber, a

gerontologia e a tecnologia, resultando num conceito que designa o estudo do processo de envelhecimento com recurso as novas tecnologias.

Com o avanço do estudo da gerontecnologia conseguiu-se desenvolver ao longo do tempo diversos sistemas que melhoram a qualidade de vida dos idosos, tentando minimizar a sua dependência de terceiros.

Existem três ideias genéricas que envolvem a gerontecnologia. A primeira é a de conseguir incluir os idosos no avanço da tecnologia, não podendo estes sentir-se excluídos pela mesma. A segunda é que a idade ou o sexo não podem ser impedimento para a realização das tarefas, sendo que para executar uma tarefa difícil sem ajuda, a mesma é facilitada ultrapassada recorrendo à tecnologia correcta. Por último, a terceira é que o idoso tem de ter controlo na tecnologia que o rodeia, sendo o idoso a tomar as decisões (Thomas L. Harrington e Marcia K. Harrington 2000).

Ao longo dos anos têm sido realizados vários avanços para melhorar a qualidade de vida dos idosos. A Tabela 1 ilustra a evolução dos sistemas por ordem cronológica.

Ano	Tipo de monitorização
1988	Monitorização física, horas de sono, hábitos de higiene, peso e utilização do computador. Os dados eram armazenados, sem gerar alertas nem era efectuada uma análise dos dados em tempo real (T. Tamura, T. Togawa et al. 1988).
1992	Surgem os primeiros sistemas a alertar as urgências, aquando de uma súbita alteração na condição do paciente, sendo o sistema a efectuar a chamada. O sistema recolhe informações biológicas, físicas e outras, tal como as queixas do idoso (H. Inada 1992).
1995	Recolha de dados para monitorizar o comportamento e funcionamento da saúde do idoso, detectando modificações na condição de saúde. A análise de dados é feita *off-line*, sendo gerados relatórios para os participantes que tiveram alterações

	significativas (B.G. Cellera 1995).
1995	Incorporado um sistema de aprendizagem que controla um ambiente e consegue enviar alertas. É utilizada uma rede neuronal para aprender os hábitos dos elementos de um grupo de pessoas (temperatura e localização) (Chan, Hariton et al. 1995). A rede é treinada durante um período de tempo, sendo usada para controlar a temperatura da sala com os dados adquiridos. O autor estendeu o trabalho para reconhecer mudanças de comportamento e lançar alertas (F. Steenkeste 2001).
1998	Apresentada uma sugestão de como melhorar a qualidade de vida dos idosos conjugando ergonomia e gerontecnologia. São apresentadas soluções que melhoram a movimentação e segurança na cozinha e na entrada da casa. Estes conselhos pretendem minimizar o risco de acidentes e melhorar o conforto dentro de casa, sem esquecer as necessidades físicas e psicológicas do idoso. Assim são necessárias pequenas modificações/adaptações para melhorar a segurança e mobilidade dentro das casas (Maria Rita Pinto; Stefania De Medici; Clarke Van Sant; Alfredo Bianchi; Andre Zlotnicki e Claudio Napoli 1998).
2000	Utilização de sensores não intrusivos para detectar actividades da vida diária. Este sistema não responde aos dados recolhidos, sendo estes armazenados e analisados posteriormente (Anthony P. Glascock e David M. Kutzik 2000). No entanto a patente cobre um sistema que responde aos dados recolhidos (David M. Kutzik; Anthony P. Glascock; Douglas L. Chute; Thomas T. Hewett e Barbara G. Hornum 1994).
2004	Utilização de diversos agentes distribuídos para a monitorização do idoso. Este sistema permite adicionar novos agentes, hierarquizar a rede e prever acontecimentos (Karen Zita Haigh e Janet Myers 2004).
2008	Seis países europeus (Portugal, Espanha, Itália, Alemanha, Reino Unido e Irlanda) contribuíram para o desenvolvimento do projecto *Complete Ambient Assisted Living eXperiment* (CAALYX) (Athanasios Anastasiou, Paul Quarrie et al. 2008). Este projecto consiste em monitorizar os sinais vitais dos idosos, detecção de quedas e localização do idoso, estando este em casa ou fora da

	mesma. Os sensores utilizados são sensores médicos já existentes. Os alarmes que o sistema pode enviar são realimentados com o histórico já existente, detectando assim alterações na saúde do idoso, minimizando falsos alarmes. Enquanto o idoso está em casa é utilizado um sistema integrado (computador/ televisão) com acesso à Internet. Para completar o sistema foram utilizados telefones (N95) para fazer a detecção da localização do idoso fora de casa (Athanasios Anastasiou, Paul Quarrie et al. 2008).

Tabela 1– Evolução dos sistemas de monitorização

2.2 Interoperabilidade de sistemas

A interoperabilidade de sistemas procura criar uma *interface* de comunicação comum onde os subsistemas suportados por arquitecturas distribuídas consigam comunicar. Assim é possível criar um sistema complexo com vários sistemas simples, onde cada um deles desempenha uma função no sistema geral conseguindo "falar" uns com os outros.

A tendência dos anos de 70 e 80 era da criação de aplicações centralizadas, tendo o conceito de computador central, onde a escalabilidade não era prevista, tornado os sistemas ingovernáveis. Isto levou a grandes sistemas criados por vários subsistemas suportado por arquitecturas distribuídas (Luís Arriaga 2009). Desta forma o conceito de "ilhas isoladas de sistemas de informação" evoluiu, passando-se a ter uma cultura de sistemas de informação cooperativos. Para se conseguir tal objectivo é necessário as aplicações utilizarem uma mesma linguagem, ou então seguirem um conjunto de regras que permitam a comunicação, habilitando assim a aplicação a fazer-se entender pelos outros sistemas.

Uma solução que garante a interoperabilidade de sistemas é seguir padrões de indústria (*standards*). De uma forma simples, um padrão é uma solução comprovada para resolver um determinado problema comum a vários sistemas. No entanto, a noção de padrão já é utilizada de forma natural, pois utilizam-se soluções comprovadas para resolver problemas diários (Thomas Erl 2009). Um padrão é então definido como um documento elaborado em consenso e aprovado por um organismo reconhecido. Este documento tem regras, directrizes ou

características para atingir resultados de grau óptimo num determinado contexto (definição ISO / IEC Guia 2:2004). A utilização de padrões é útil pois simplifica o processo de desenvolvimento de *software*, dado que são utilizadas soluções padronizadas que resolvem problemas encontrados durante este processo.

Existem dois tipos de padrões, os padrões fechados ou proprietários e os padrões abertos. Os padrões fechados são normalizados, licenciados não sendo o seu acesso gratuito. As regras deste tipo de padrão têm de ser públicas (caso contrario não é considerado um padrão) (FFII 2009).

Para se conseguir interoperabilidade mais acessível, é necessário recorrer a padrões abertos. Para ser considerado um padrão aberto, tem de respeitar cinco características segundo a definição da União Europeia, (Communities 2004). A primeira é dos custos para a utilização do padrão serem baixos, não sendo um obstáculo ao seu acesso. A segunda é que o padrão tenha sido publicado. A terceira é que o padrão tenha sido adoptado com base num processo aberto de tomada de decisão (consenso, decisão aleatória ou outro processo de decisão). O quarto é que os direitos intelectuais do padrão são atribuídos a uma organização sem fins lucrativos, que se rege por políticas de acesso gratuito. O quinto é que não pode existir restrições para reutilizar o padrão.

No entanto, para existir interoperabilidade, é necessário existir comunicação entre máquinas, esta comunicação é possível através do uso de redes de dados. As redes de dados são grupos de computadores interligados por canais de comunicação que permitem a comunicação e a partilha de recursos entre utilizadores (IEEE *Standards Information Network* 2000). No entanto, apesar desta definição em comum existem vários tipos de redes diferentes, tais como:

- *Local Area Network* (LAN) – um tipo de rede bastante comum que consiste em pelo menos dois equipamentos ligados numa área geográfica limitada, como uma sala ou um edifício (IEEE 2002);
- *Metropolitan Area Network* (MAN) – é uma rede optimizada para uma área geográfica maior que a de uma LAN, que se pode estender de vários edifícios a cidades inteiras (IEEE 2002);

- *Wide Area Network* (WAN) – é uma rede que abrange uma grande área geográfica, que ultrapasse as fronteiras de uma cidade, de uma região ou mesmo de um país (Skandier 2009).

Para que redes completamente diferentes possam interagir umas com as outras foi projectado o modelo *Open Systems Interconnection* (OSI). Este modelo é baseado em 7 camadas de abstracção, que são: Camada de Aplicação; Camada de Apresentação; Camada de Sessão; Camada de Transporte; Camada de Rede; Camada de Ligação de Dados; Camada Física.

Cada camada interage apenas com a camada imediatamente abaixo, ao mesmo tempo que providencia serviços à camada imediatamente acima. As funcionalidades de cada camada são implementadas por uma ou mais entidades externas ao modelo, pois este apenas define os *interfaces* entre os diferentes computadores através dos seus protocolos. A implementação específica de como interagir com os vários componentes do próprio computador é definida por ele mesmo, permitindo assim a várias entidades em computadores diferentes interagirem através da mesma camada (ITU-T 1994).

O modelo OSI teve grandes vantagens na padronização e base para futuros desenvolvimentos, no entanto este modelo é apenas conceptual, não sendo aplicável na realidade. Como tal a maioria dos protocolos usados hoje em dia são baseados em *Transmission Control Protocol/Internet* Protocol (TCP/IP).

O modelo TCP/IP, tal como o modelo OSI, foi desenhado de forma a possibilitar que computadores possam comunicar através de uma rede, contudo os seus protocolos são diferentes. Este é composto por quatro camadas de abstracção (rfc1122 1989): Camada de Aplicação; Camada de Transporte; Camada de Internet; Camada de Rede.

O protocolo TCP/IP permite comunicação entre computadores, no entanto esta comunicação era impossível caso não tenha um destino final. Os *sockets* de rede são esse destino final, onde a informação contida na comunicação é recebida. Uma *socket* de rede é o ponto final de qualquer comunicação entre dispositivos numa rede baseada em IP, é também uma *Application Programming Interface* (API) do protocolo TCP/IP que serve para entregar os pacotes de rede ao processo ou thread apropriado.

Sabendo como a informação flui entre computadores, e como é que esta se encontra organizada de forma a estar acessível, torna-se também importante poder executar funções ou executar ordens remotamente. Desta forma consegue-se ter aplicações distribuídas, sendo que cada computador tem funções e procedimentos próprios, podendo estes ser executados sem se ter acesso físico. Estas aplicações têm a vantagem de distribuir as tarefas a executar por diversos computadores, aumentando assim os recursos disponíveis para as aplicações. Uma vantagem destas aplicações é o particionamento de tarefas por computador, ou seja poder delegar as tarefas aos computadores que se destinam a executar as mesmas. As aplicações distribuídas não se encontram limitadas a uma sala com diversos computadores ligados entre si, mas sim à rede de dados a que se encontram ligados.

Uma das tecnologias que permite executar funções remotamente é o *Remote Procedure Call* (RPC) que permite um programa de computador, executar código de outro dispositivo numa rede sem ser necessário programar a forma como essa interacção é feita. Os RPCs permitem assim grande flexibilidade, mas também têm vários problemas, como a possibilidade de a RPC falhar devido a problemas com a rede, ou o facto de existirem várias implementações diferentes, o que resulta em protocolos incompatíveis.

Quando o *software* é baseado em Programação Orientada a Objectos (POO) o princípio é o mesmo, mas invoca objectos em vez de procedimentos. Existem implementações diferentes, que invocam componentes remotamente, tais como:

- Java RMI – é uma API da linguagem de programação Java que implementa o RMI;
- *Distributed Component Object Model* (DCOM) – é uma tecnologia proprietária da Microsoft para a comunicação entre componentes de *software* através de uma rede, mas foi suplantada pela *Framework*.NET;
- JINI – é uma arquitectura de rede para a construção de sistemas distribuídos usando serviços cooperativos. Originalmente desenvolvido pela *Sun Microsystems*, em 2005 foi lançado sob a licença *open source* da Apache sob o nome de *Apache River* (LaMonica 2005);
- *Common Object Request Broker Architecture* (CORBA) – é um padrão definido pelo *Object Management Group* (OMG) que permite a componentes de *software* escritos em múltiplas línguas de programação diferente trabalhar em conjunto;

- *Windows Communication Foundation* (WCF) – é uma API pertencente à *Framework* .NET que permite construir aplicações orientadas a serviços. (Microsoft 2010).

Estas implementações estão ligadas a uma tecnologia de programação, sendo que a interoperabilidade entre máquinas com sistemas diferentes fica comprometida. Para colmatar esta falha e garantir a interoperabilidade entre sistemas, foi desenvolvida a tecnologia de *Web services* que tem muitas definições divergentes, contudo a definição dada pelo *World Wide Web Consortium* (W3C) é:

"Um sistema de *software* desenhado para suportar interacção entre sistemas na rede. Possui um *interface* descrito num formato processável por uma máquina. Para os sistemas interagirem com o *Web service* da maneira prescrita pela sua descrição são usadas mensagens *Simple Object Access Protocol* (SOAP), tipicamente enviadas usando *Hipertext Transfer Protocol* (HTTP) com serialização *Extensible Markup Language* (XML) em conjunção com outros padrões *Web (World Wide Web Consortium (W3C) 2004)*. Foi escolhida esta definição, pois o W3C é a organização de padrões internacionais para a *World Wide Web* (WWW)."

Em termos simples, um *Web service* pode ser criado em qualquer linguagem de programação. Após ser disponibilizado, poderá ser utilizado por qualquer sistema que o invoque independentemente da localização ou tecnologia deste, desde que respeite as regras do *Web Service*. Os *Web services* são dependentes de várias tecnologias de forma a poderem fazer o seu trabalho, algumas das mais importantes são:

- XML – um padrão de regras que permite codificar informação numa forma que pode ser lida por humanos e máquinas;
- WSDL – *Web Services Description Language* é uma linguagem baseada em XML que permite criar modelos para descrever um *Web service*;
- SOAP – um protocolo baseado em XML, que tem como objectivo, a troca de informação estruturada entre *Web services*.

Para além destas, existem outras extensões baseadas em XML que podem ser facilmente integradas de forma a adicionar capacidades ao *Web service*. O *WS-Security* por exemplo especifica como garantir a segurança em mensagens e o *WS-ReliableMessaging* especifica como garantir a entrega de mensagens. Estes são apenas alguns exemplos, existindo muitas mais especificações para múltiplos propósitos (Metro 2010).

Contudo, para usar todas estas tecnologias em uníssono de forma correcta é necessário planeamento e estruturação das tecnologias. Algumas das arquitecturas existentes são o *Representational State Tranfer* (REST) e o *Service-Oriented Architecture* (SOA).

O REST é uma arquitectura de *software* para sistemas distribuídos. "O REST define um conjunto de princípios arquitecturais através dos quais se pode desenhar *Web Services* que se focam nos recursos do sistema. Os estados desses recursos são endereçados e transferidos através do protocolo HTTP por uma grande gama de clientes escritos em diferentes linguagens" (Alex Rodriguez 2008).

O SOA é um conjunto flexível de princípios de desenho, usado durante o desenvolvimento de um sistema, que tem como objectivo criar um conjunto de serviços o mais independente possível (*loosely-coupled)*, que possam ser utilizados em múltiplos sistemas e que sejam de fácil descoberta pelos utilizadores, ou seja, que os utilizadores possam facilmente descobrir que serviços lhe estão disponíveis bem como os dados que podem fornecer.

No entanto são necessárias arquitecturas que definam que dados circulam na rede. É assim necessário definir quem recebe os dados, quem os envia, onde estes se encontram e como os procurar.

Estas tecnologias servem de apoio na troca de informação entre sistemas e organização dos seus dados. No entanto é necessário saber o que são os dados e como os diferenciar. Os dados são o conjunto de informação que representam os atributos qualitativos ou quantitativos de uma variável ou um conjunto de variáveis, ao passo que os metadados são dados sobre os dados (NISO 2004). Tendo como exemplo um livro, os dados seriam o seu texto. Dependo do livro, esse texto pode ser uma história, poemas, uma dissertação científica ou cultural ou uma infinidade de outros temas possíveis, as possibilidades são imensas, como é o caso com a maioria dos dados, ao passo que os metadados de um livro seriam o seu título, autor, número de páginas, número de palavras, o tema do livro, editora ou *International Standard Book Number* (ISBN) apenas para mencionar alguns.

Pode-se assim ver que os metadados formam uma estrutura formal de como representar um certo tipo de dados que pouco ou nada varia, ao passo que os dados referentes a essa estrutura tem uma grande variação do seu conteúdo ao mesmo tempo que estão confinados às regras impostas pelos metadados.

Os metadados são importantes para a interoperabilidade entre sistemas por definirem a estrutura de dados através de sistemas diferentes, seja garantido que todos os sistemas têm a mesma definição para os dados ou providenciando uma definição para os dados própria a cada sistema, que podem ser utilizadas para criar uma forma de tradução entre eles.

Uma forma simples de entender este conceito é pensar nos metadados de um sistema como uma língua, por exemplo o Português e os metadados de um outro sistema como uma outra língua, por exemplo o Inglês. Ambas as línguas têm palavras que definem os mesmos conceitos, contudo um falante de português, não vai compreender essa palavra em inglês e vice-versa, como tal vai precisar que a mesma seja traduzida para a sua própria língua.

No contexto do ElderCare os dados provêm de várias fontes, desde a miríade de dados provenientes dos vários sensores que cada cliente possui, aos dados biográficos e de contacto fornecido pelos clientes e respectivos contactos, passando pelos dados analisados pelo classificador.

Nesta secção foi apresentado o conceito de interoperabilidade de sistemas e as condições que levaram ao seu desenvolvimento. Foi apresentado ainda de que formas são utilizadas algumas das tecnologias existentes para solucionar problemas de interoperabilidade entre sistemas. Ainda foram apresentadas algumas arquitecturas de desenho existentes que ajudam a resolver problemas de interoperabilidade. Terminou-se com o conceito de dados e metadados. Na secção seguinte é apresentada uma solução de interoperabilidade na área da saúde, o *Healt Level Seven* (HL7). Esta solução foi estudada, pois foi uma das hipóteses de trabalho considerada.

2.3 Interoperabilidade de sistemas em sistemas para a saúde Health Level Seven

A norma *Health Level Seven* (HL7) foi criada em 1987, por uma associação sem fins lucrativos, sendo esta uma norma *American National Standards Institute* (ANSI) (Huff 1998). Tem como objectivo integrar, trocar, partilhar e receber dados clínicos dos pacientes, tendo sido criada a pensar na globalidade dos dados de saúde e não em especificidades dos serviços.

Antes do HL7 as *interfaces* entre sistemas eram elaboradas tanto no receptor como no emissor, sendo que estas eram pensadas para resolver problemas de comunicação entre sistemas específicos, não prevendo futuras comunicações com outros agentes, logo envolviam custos elevados com a agravante que necessitavam de várias *interfaces* para comunicar com diversos sistemas. Outra lacuna era a incompatibilidade nas normas de recolha de dados dos utentes nas diversas instituições de saúde, normas como o *Acute Coronary Syndrome Clinical Data Standards* para recolha de dados para doentes cardíacos, ou o *Digital Imaging and Communications in Medicine* (DICOM) (ASCN 2004), que permite a troca de imagens entre sistemas médicos, não permitindo a troca de informação dos pacientes (Foundation 2004). A Figura 2 mostra o aumento do número de interfaces clínicas ao longo do tempo, sendo o seu custo de desenvolvimento menor, com a adopção da norma HL7. Com o decorrer dos anos e com a adopção do HL7 o número de interfaces aumentou, pois existe uma interface genérica que possibilita comunicação entre sistemas clínicos. Este aumento de interfaces provém de um sistema padrão que permite às instituições trocar informação sem usar as suas normas internas.

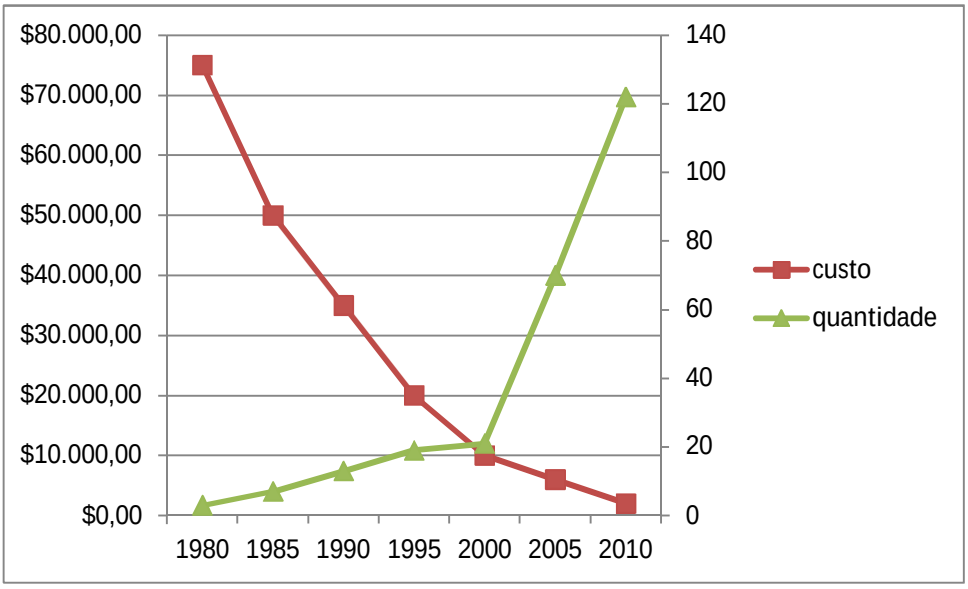


Figura 2 - Custo e quantidade de interfaces entre sistemas nos hospitais ao longo do tempo. Adaptado de Corepoint Health (Health 2009)

O HL7 foi pensado para integrar aplicações de saúde, pois estas eram elaboradas sem ter em conta a troca de informação com outros sistemas, tornando-as em ilhas isoladas de informação. Com esta norma consegue-se trocar dados que estão em diversos sistemas, possibilitando ao utente deslocar-se por diversas instituições de saúde, e levar as suas informações clínicas. O sistema que vai receber os dados clínicos sabe interpretar a informação do utente. Isto acontece mesmo que não se conheça a forma como os dados estão estruturados internamente.

Esta norma foi criada com o intuito de agregar várias instituições de saúde de vários países, de forma a existir um padrão de comunicação entre sistemas de saúde, resolvendo os problemas de comunicação (Shaver 2007).

Existem três tipos de utilizadores da norma HL7: Especialistas em *interfaces* clínicas; Governo e outras entidades políticas; Informáticos da saúde.

Os especialistas em *interfaces* clínicas têm a tarefa de transferir os dados clínicos, criando ferramentas para esse efeito e/ou criar aplicações clínicas que troquem dados com outros sistemas. Estes especialistas são responsáveis pela movimentação de dados entre aplicações clínicas ou entre as diversas instituições de saúde.

O governo e outras entidades políticas estão preocupados com o futuro da partilha de informação entre sistemas ou entidades. Procuram manter os dados clínicos organizados, possibilitando levá-los a novas áreas, como hospitais, centros de saúde, clínicas privadas, que não são abrangidas pelas *interfaces* actuais. Têm ainda a capacidade de criar padrões para a troca de informação, podendo torná-los obrigatórios.

Os informáticos da saúde, procuram criar ou adoptar uma ontologia clínica, isto é, uma estrutura hierárquica de conhecimento em saúde (um modelo de dados), terminologia (vocabulário), e fluxo de trabalho (como as coisas são feitas) (Shaver 2007).

Existe em 2011 duas versões do HL7. A versão 2.x utiliza separadores para formar a mensagem enquanto a versão 3.x utiliza a *Extensible Markup Language* (XML). Cada versão tem especificações diferentes, isto advém do grupo que a criou bem como da época em que foi criada. A versão 2.x inicialmente não foi pensada com uma escalabilidade tão grande como lhe é exigida actualmente.

```
MSH|^~\&|DDTEK LAB|ELAB-1|DDTEK
OE|BLDG14|200502150930||ORU^R01^ORU_R01|CTRL-9876|P|2.4 CR

PID|||010-11-
1111||Estherhaus^Eva^E^^^^L|Smith|19720520|F|||256 Sherwood
Forest Dr.^^Baton Rouge^LA^70809||(225)334-5232|(225)752-
1213||||AC010111111||76-B4335^LA^20070520 CR

OBR|1|948642^DDTEK OE|917363^DDTEK LAB|1554-
5^GLUCOSE|||200502150730|| ||||||||020-22-2222^Levin-
Epstein^Anna^^^^MD^^Micro-Managed Health
Associates|||||||||F|||||||030-33-
3333&Honeywell&Carson&&&&MDCR

OBX|1|SN|1554-5^GLUCOSE^^^POST 12H
CFST:MCNC:PT:SER/PLAS:QN||^175|mg/dl|70_105|H|||F CR
```

Figura 3 - Exemplo de uma mensagem hl7 v2.x a pedir um teste de glicose. Adaptada de DataDirect (Technologies 2011)

A versão 2.x é baseada em *Electronic Data Interchange* (EDI), que já se encontra amplamente implementada. Formada por linhas, que são separadas por segmentos tendo elementos opcionais como é mostrado na Figura 3, estas características tornam as mensagens flexíveis. Esta flexibilidade é penosa para os programadores/analistas, pois o processo de integração é moroso e complexo. Existem várias interpretações para as mesmas especificações, o que gera discórdia entre os intervenientes. As novas versões são lançadas com novas opções garantindo a retro-compatibilidade, isto é, garantem o normal funcionamento das aplicações mesmo com versões anteriores(CISCO 2007). A versão 2.6, a mais actual à data deste documento, conta com 170 segmentos diferentes. Devido à época em que foi criado não suporta as novas tecnologias que surgiram tais como XML, Internet e segurança.

```
<?xml version="1.0" encoding="UTF-8"?>
<Message xmlns="urn:hl7-org:v3" xmlns:dt="urn:hl7-org:v3"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <id root="2.24.750.2.937172.4433" extension="CTRL-9876"/>
                <creation_time value="2005-02-15T09:30:00-05:00"/>
                <version_id>3.0</version_id>
                <interaction_id root="2.24.750.2.937172" extension="POLB_IN004410"/>
                <processing_cd code="P"/>
                <accept_ack_cd code="ER"/>
                ...
                <interactionTarget xsi:type="POLB_MT004101">
                <ObservationEvent>
        <id root="2.24.750.2.937172.4433" extension="917363"
        assigningAuthorityName="DDTEK LAB"/>
        <cd code="1554-5" codeSystemName="LN"
        displayName="GLUCOSE POST 12H CFST:MCNC:PT:SER/PLAS:QN"/>
        <status_cd code="completed"/>
        <effective_time>
                <center value="2005-02-15T07:30:00-05:00"/>
        </effective_time>
        <activity_time>
                <center value=""/>
        </activity_time>
        <priority_cd code="R"/>
        <value xsi:type="dt:PQ" value="175" unit="mg/dl"/>
        ...
</Message>
```

Figura 4 - Excerto de uma mensagem hl7 v3.x a pedir um teste de glicose. Adaptada de DataDirect (Technologies 2011)

A versão 3.x é incompatível com a versão 2.x, pois o modelo de dados é diferente. A versão 3.x segue uma metodologia orientada a objectos. A versão 3.x engloba especificações de tipos de dados abstractos, *Reference Model Information* (RIM), XML *Implementation Technology Specification (XMLITS)* e vocabulário de domínio (Sunderraman 2004) (Johnson 1999), como se pode ver no excerto que se encontra na Figura 4.

O *Reference Model Information* (RIM) consiste num conjunto de classes representadas em *Unified Modeling Language* (UML). O UML é uma linguagem visual para especificar, construir e documentar os componentes do sistema. É uma linguagem de modelação genérica que pode ser utilizada com a maioria dos componentes (objectos, métodos) podendo ser

aplicada em todos os domínios de aplicações (saúde, finanças, telecomunicações) e plataformas de implementação (J2EE, .Net) (*Object Management Group* 2010). Cada classe tem os seus atributos estando estes associados a uma especificação de tipos de dados. As classes que formam a base do modelo RIM são:

- *Act* representa as acções que são executadas, tendo estas de ser documentadas enquanto o paciente é atendido;
- *Participation* onde é expresso o contexto de uma acção (*act*), ou seja, quem executou, a quem foi realizada, onde foi realizada;
- *Entity* onde são representados todos os seres e coisas físicas que são importantes para o tratamento;
- *Role* onde são representados os papéis que as entidades assumiram durante o tratamento;
- *ActRelationship* representa o relacionamento que existe entre as acções, a relação entre uma ordem e a sua execução;
- *RoleLink* representa o relacionamento entre papéis individuais (HL7 2004).

A XMLITS é utilizada no desenvolvimento de XMLSchemas para as especificações de HL7 V3.x, contudo este processo ainda necessita de refinamento (Sunderraman 2004).

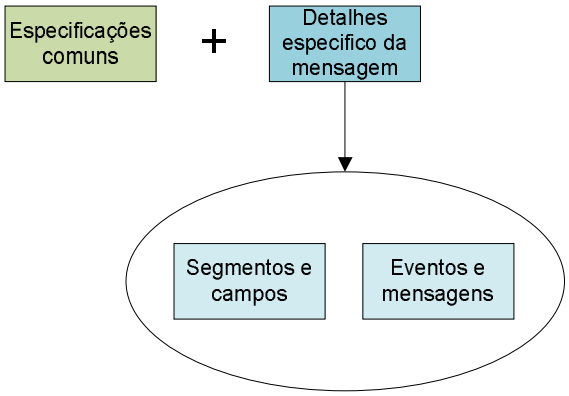


Figura 5 - estrutura do HL7 v2.x. Adaptado de HL7 V3 and the Flow of Health Information (Paterson 2010).

Devido às diferenças estruturais da versão 2.x para a 3.x, que podem ser vistas na Figura 5 e Figura 6 respectivamente, as mensagens da 3.x são mais precisas e de fácil entendimento, tendo ainda espaços abertos para futuros desenvolvimentos.

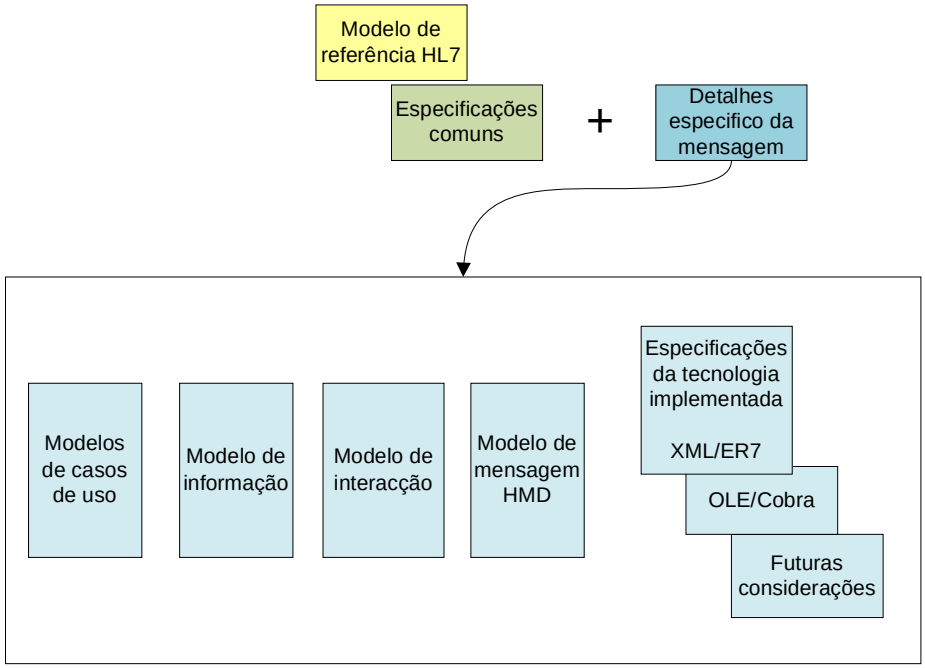


Figura 6 - estrutura do HL7 v3.x. Adaptado de HL7 V3 and the Flow of Health Information (Paterson 2010).

Nesta secção foi apresentado a norma de interoperabilidade de sistemas para a área da saúde desde a sua criação, passando pela sua evolução até à sua versão mais recente. A utilização desta norma permite a criação de uma ampla gama de mensagens, sendo estas perceptíveis em ambientes diferentes dos da sua origem. Esta propriedade permite que as informações clínicas de um paciente possam ser transferidas entre instituições médicas. Ainda são utilizadas ambas as versões, sendo a 2.x a mais utilizada pois existe há mais tempo.

2.4 Arquitectura orientada a serviços

A arquitectura orientada a serviços (SOA) estabelece um modelo arquitectural cujo intuito é aumentar a eficiência, agilidade e produtividade de um empreendimento, através do posicionamento de serviços como as principais peças através das quais será representada a lógica de solução que suportará a realização dos objectivos estratégicos do empreendimento.

Enquanto forma de arquitectura tecnológica, uma implementação SOA pode consistir de inúmeras combinações de tecnologias, produtos, APIs, e várias outras partes, que suportam a criação, manutenção e evolução de soluções orientadas ao serviço. Assim, uma arquitectura tecnológica construída usando o modelo SOA estabelece um ambiente apropriado à lógica da solução para acomodar outras soluções que sigam os princípios de desenho orientados a serviços (Erl 2005) .

A SOA segue uma metodologia que permite atingir interoperabilidade entre aplicações e a reutilização de funcionalidades independentemente da arquitectura subjacente a cada aplicação. Fornece também uma forma de desenho que guia uma organização durante todos os aspectos da criação e uso de serviços, incluindo concepção, modelação, desenho, desenvolvimento, distribuição, gestão e definição da versão.

A SOA tem como característica uma forte concentração na arquitectura incluindo, estrutura organizacional, processos, modelação e ferramentas. Permite um nível ideal de abstracção para alinhar as necessidades do projecto e as capacidades técnicas. Os serviços são criados de forma a serem reutilizáveis e abrangentes (*coars-grained*). Disponibiliza ainda uma infra-estrutura de distribuição em que novas aplicações podem ser rápida e facilmente construídas, pois consegue interligar serviços já existentes, de forma a resolver novos problemas. Disponibiliza os serviços existentes numa biblioteca de serviços reutilizável para negócios e funcionalidades informáticas.

O paradigma de desenho orientado a serviços é composto por oito princípios que fornecem regras e directrizes para ajudar a determinar como é que a lógica da solução deve ser decomposta e moldada em unidades distribuíveis. A aplicação repetida destes princípios aumenta a quantidade de características de desenho comuns a essas unidades, tornando a lógica da solução mais uniforme. Segue-se uma descrição desses princípios (Erl 2007):

- Descoberta de Serviços. A correcta integração de um serviço num sistema depende em grande parte da capacidade do mesmo de ser detectado e entendido pelos restantes nós do sistema, tanto pré-existentes como futuros. Para isso é necessário um especial cuidado com os mecanismos de comunicação aquando do desenvolvimento do mesmo.

- Reutilização de Serviços. Numa orientação com base em serviços a reutilização é colocada em ênfase, tanto que se tornou a parte central de uma análise de serviços típica, e dos processos de desenho. Forma igualmente a base chave para modelos de serviços. O advento de tecnologias de serviços maduras e não proprietárias forneceu a oportunidade de maximizar o potencial de reutilização de lógica multipropósito a um nível sem precedentes. São levantadas várias considerações de desenho para garantir que capacidades de serviços individuais são adequadamente definidas em relação a contextos de serviço agnósticos, e para garantir que possam facilitar os requisitos de reutilização necessários.

- Agradabilidade de Serviços. À medida que as soluções orientadas a serviços se tornam cada vez mais sofisticadas, também a complexidade das configurações de composição de serviços cresce. A capacidade de compor serviços de maneira eficaz é um dos pontos críticos para os objectivos da computação orientada a serviços. Assim, é esperado de um serviço que este seja capaz de participar efectivamente numa composição, independentemente do tamanho da mesma, e mesmo se o serviço não for imediatamente necessário como parte de uma composição.

- Autonomia de Serviços. Para os serviços terem capacidades constantes e confiáveis, é necessário que as suas lógicas de solução subjacentes possuam um certo nível de controlo sobre o seu ambiente e recursos. Seguindo essas necessidades os serviços tornam-se autónomos. Tal autonomia cultiva características de design que aumentam a fiabilidade e previsibilidade comportamental de um serviço, o que suporta a realização de outros princípios de design em ambientes de produção.

- Contractos de Serviço Normalizados. Um serviço expressa as suas finalidades e capacidades através de um contracto de serviço. O princípio de Contractos de Serviço Normalizados dita que serviços com o mesmo inventário de serviços devem observar

as mesmas normas de design de contractos. É talvez a peça mais fundamental da orientação a serviços na medida em que essencialmente requer que considerações específicas sejam tidas em conta ao desenhar a interface técnica pública de um serviço, e ao avaliar a natureza e quantidade de conteúdo que será publicado como parte do contracto oficial de um serviço. É dado especial ênfase a aspectos específicos de design de contractos, incluíndo a forma como serviços expressam funcionalidade, como modelos tipos de dados são definidos, e como são avaliadas e anexadas políticas. Existe um foco constante em assegurar que os contractos de serviço sejam apropriadamente granulares, optimizados e normalizados para garantir que os pontos de ligação estabelecidos pelos serviços sejam consistentes, confiáveis e governáveis.

- Baixo Grau de Acoplamento de Serviços. O conceito de acoplamento refere-se à ligação ou relação entre duas coisas. Uma medida dessa acoplagem é comparável a um nível de dependência. O princípio de Baixo Grau de Acoplamento de Serviços advoca a criação de um tipo específico de relação dentro e fora das fronteiras de serviço, com uma ênfase em diminuir o grau de dependência entre o contracto de serviço, a sua implementação, e os consumidores do seu serviço. Este princípio afirma que os contractos de serviço devem impor baixos requisitos de acoplamento por parte dos consumidores e são eles próprios desligados do seu ambiente circundante. Desta forma visa promover o design e evolução independentes da lógica e implementação de um serviço, mantendo a interoperabilidade base com os consumidores que tenham vindo a depender das capacidades do serviço. Existem vários tipos de acoplamento envolvidos no design de um serviço, cada qual pode impactar o conteúdo e granularidade do seu contracto. Atingir o nível de acoplamento apropriado requer que considerações práticas sejam equilibradas contra várias preferências de design de serviço.

- Serviços Sem Estado. A gestão de informação de estado excessiva pode comprometer a disponibilidade de um serviço, e minar o seu potencial de escalabilidade. Serviços são por isso idealmente desenhados para manterem o seu estado apenas quando necessário. O princípio de Serviços Sem Estado diz que os serviços devem, quando necessário, minimizar o consumo de recursos através da deferência da gestão de informação de estado. Aplicar o princípio de Serviços Sem Estado requer que sejam

avaliadas medidas alcançáveis para atingir um serviço sem estado, baseadas na adequação da arquitectura de tecnologia circundante para fornecer opções de delegação e deferência de gestão de estado.

- Abstracção de Serviços. O princípio de Abstracção de Serviços dita que os contractos de serviço apenas devem manter informação essencial, e que informação acerca dos serviços deve ser limitada à especificada nos contractos de serviço. A um nível mais fundamental, este princípio salienta a necessidade de esconder os detalhes subjacentes de um serviço tanto quanto possível, para preservar o baixo nível de acoplamento de serviços previamente descrito. O nível de abstracção empregue num serviço afectará a granularidade do seu contracto de serviço e os seus custos de governação.

2.5 Ontologias

O conceito de ontologia já provém da filosofia, e descreve a existência dos acontecimentos (Viinikkala 2004). Uma ontologia é uma especificação detalhada de uma conceptualização partilhada (Gruber 1993). Uma ontologia não se encontra limitada a uma hierarquia de conceitos, abrangendo também um conjunto de relações entre conceitos, restrições, axiomas, instâncias e vocabulários. Assim sendo, uma ontologia descreve um domínio, com regras e conceitos bem definidos, permitindo partilhar o conhecimento do domínio, que a ontologia representa. A utilização de ontologias para definir um determinado domínio tem várias vantagens, pois permite partilhar o conhecimento que o domínio representa, utilização de uma ontologia genérica para definir um domínio específico e representação semântica dos dados do domínio. Uma ontologia construída com qualidade tem de definir o domínio de conhecimento com clareza e objectividade, dar a conhecer e entender o conhecimento que o domínio representa e não permitir ter interpretações divergentes do domínio (Raquel Elias Carneiro e Parcilene Fernandes de Brito 2005).

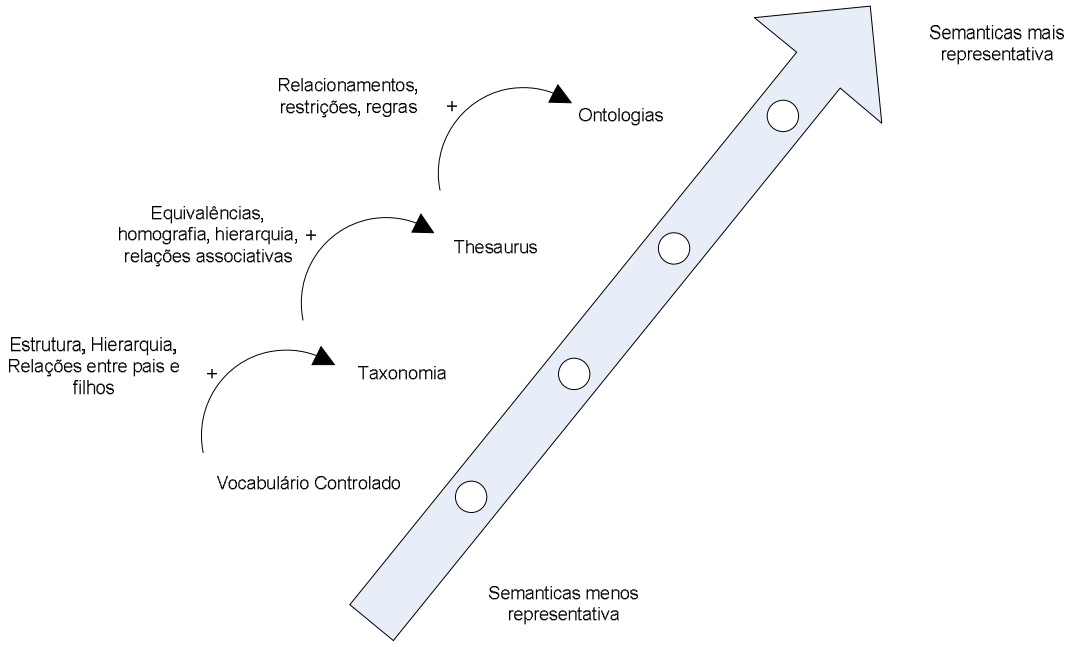


Figura 7 - Nível de representação semântica. Adaptado de Semantic Web Services: Theory, Tools, and Applications (Jorge Cardoso 2007).

Existem noções similares às ontologias, no entanto, estas não são tão abrangentes e expressivas, como a noção de ontologia. Algumas dessas noções similares são os vocabulários controlados, as taxonomias e os *thesaurus* ou bases de conhecimento. Os vocabulários controlados são uma lista de termos simples enumerados. Cada termo tem de ter um significado único e bem definido. Os vocabulários controlados estão limitados à sua lista, sendo esta lista estática. As taxonomias organizam os termos existentes em hierarquias. Nas taxonomias as relações existentes entre termos são explícitas. Desta forma é possível perceber a semântica do termo, pela análise da relação que existe entre a palavra e as palavras que se encontram ao seu redor na hierarquia (Cardoso 2007). Os *thesaurus* são uma rede de vocabulários controlados, com relações entre termos. A diferença entre os *thesaurus* e as taxonomias é que os *thesaurus,* permitem uma organização hierárquica mais abrangente e permitem ainda adicionar novas definições e relações entre termos. As ontologias são similares às taxonomias, com a diferença de usar relações semânticas mais complexas, definir regras nas relações entre os termos e na definição dos termos. As ontologias estão assim no topo da cadeia da representação semântica do conhecimento (Mário Joaquim Firmino Leite

Faria, Prof. Dr. Luís Paulo Reis et al. 2009). A Figura 7 ilustra os diferentes níveis de representação semântica e as diferenças existentes entre os vários níveis.

2.5.1 Linguagens de ontologia

O inicio das linguagens para as ontologias teve origem nos anos 90. As linguagens surgiram da evolução das linguagens de representação de conhecimento (RC). As primeiras linguagens que definiam as ontologias eram baseadas no *Knowledge Interchange Format* (KIF), como por exemplo a *Otolingua*, *Open Configuration and Management Layer* (OCML), *Frame Logic* (F-logic) e também em linguagens baseadas em descrições lógicas, como por exemplo o *LOOM*. Em 1997 foi criada uma *interface* designada de *Open Knowledge Base Connectivity* (OKBC) que consegue aceder a ontologias, implementadas em diferentes linguagens.

Com a expansão da Internet, surgiram novas linguagens para as ontologias, que ganharam ênfase, pois possibilitam a exploração de características da *Web*. A sintaxe das novas linguagens é baseada em *markup languages* como o HTML e o XML. As linguagens que mais foram utilizadas no desenvolvimento de ontologias foram o *Simple HTML Ontology Extensions* (SHOE), *Ontology Exchange Language* (XOL), *Resource Description Framework* (RDF), RDF schema, Ontology Inference layer (OIL), DARPA agent markup language (DAML), DAML+OIL e o *Web Ontology Language (*OWL). Apesar de existirem todas estas linguagens, as linguagens que até a data deste documento têm melhor suporte são o RDF, RDF schema e o OWL. O RDF foi desenvolvido pelo W3C como uma linguagem semântica para descrever recursos Web. O RDF Schema também foi desenvolvido pelo W3C como uma extensão do RDF, contudo baseada em frames primitivas. Esta linguagem fornece os elementos básicos para a descrição de uma ontologia (Jorge Cardoso 2007).

O OWL foi recomendado pelo W3C em 2004. O OWL é mais expressivo que o RDF, sendo possível descrever conceitos e expressões de forma mais detalhada. O OWL está dividido em 3 camadas, o *OWL-Lite*, o OWL-DL e o *OWL-Full*, aumentando o nível de detalhe entre as camadas. O *OWL-Lite* Representa uma hierarquia de classificação e ainda consegue representar restrições básicas. O OWL-DL permite uma expressividade máxima, sem que seja

deixado de lado a computação lógica. O *OWL-Full* Permite uma máxima expressividade com liberdade sintáctica do RDF. Esta linguagem não garante que exista computabilidade devido à sua enorme complexidade. Cada uma destas sub-linguagens é compreendida pela sua predecessora, tanto na sua expressividade como nas relações (World Wide Web Consortium (W3C) 2004).

2.5.2 Editores para ontologias

Os editores das ontologias, tal como um editor de desenvolvimento de *software*, ajudam os utilizadores no desenvolvimento das ontologias. O editor deve apoiar o utilizador na definição dos conceitos da ontologia. Um Editor possibilita a criação de classes, indivíduos, atributos, axiomas, regras e hierarquias de conceitos. Um editor facilita também na criação, visualização, navegação e manutenção da ontologia. Para realizar estas tarefas existem várias ferramentas, sendo consideradas as mais relevantes as seguintes: Otolingua, WebODE, SWOOP, Altova SemanticWorks e Protégé.

O editor Otolingua[1] (Adam Farquhar; Richard Fikes; James Rice 1997) foi criado pelo laboratório de sistemas de conhecimento da universidade de Standford. Este editor é baseado em interfaces Web e consiste num servidor que possibilita a vários utilizadores desenvolverem a mesma ontologia. Apesar de ter como linguagem base a otolingua este consegue importar e exportar para várias linguagens: OKBC, DAML+OIL, KIF entre outros. Este foi o primeiro editor de ontologias e não tem nenhum motor de inferência integrado (CISCO 2007; Jorge Cardoso 2007).

O editor WebODE[2] foi desenvolvido pelo grupo de engenharia de ontologias do departamento de inteligência artificial da universidade técnica de Madrid. Esta é uma ferramenta Web e

[1] http://www.ksl.stanford.edu/software/ontolingua/

[2] http://webode.dia.fi.upm.es/WebODEWeb/index.html

permite o desenvolvimento de ontologias e fornece serviços relacionados com a engenharia das ontologias. Para aceder às ontologias é necessária a utilização de uma API Java para serviço local ou com a aplicação instalada no servidor. Esta ferramenta gera ontologias em XML, RDF, OIL, OWL entre outras (Corcho, Fernández-lópez et al. 2002).

O Editor SWOOP[3] é uma ferramenta Web para criar e efectuar manutenção de ontologias OWL. É uma ferramenta Open Source e desenvolvida em Java pela empresa MIND Lab da universidade de Maryland. Esta ferramenta possibilita a validação OWL, partição de ontologias e diversos *plug-in* para várias sintaxes de representação. Tem uma ferramenta integrada que permite a pesquisa de conceitos semelhantes na ontologia, possibilita ainda anotações colaborativas (Kalyanpur, Parsia et al. 2006).

O Altova SemanticWorks[4] é uma ferramenta comercial que permite criar e editar ontologias RDF/OWL. A sua interface gráfica permita ser configurada de diversas formas. O Altova SemanticWorks gera ontologias em N-Triples, XML, OWL, RDF e RDF(S) (2011).

O Protégé[5] é uma ferramenta para desenvolver ontologias baseado em Java e uma *framework* de bases de conhecimento mantido pela universidade de Standford. A ferramenta foi desenvolvida pelo departamento de informática médica da universidade de Standford. Inicialmente, o Protégé estava limitado à aquisição de conhecimento de sistemas especialistas para oncologia. Após ter disponibilizado o seu código, foi possível integrar diversos componentes, que melhoram as funcionalidades que ele oferece. Nesta ferramenta já existem motores de inferência integrados. Existem três formas de contribuir para o desenvolvimento do Protégé que são *Backends*, *Slot widgets* e *separador Plug*-ins. Os *Backends* permitem armazenar e importar em diversos formatos. Os *Slot widgets* são usados mostrar e editar valores ou combinações de um domínio ou tarefa especifico. Os *separador* Plug-ins são

[3] http://code.google.com/p/swoop/

[4] http://www.altova.com/products/semanticworks/semantic_web_rdf_owl_editor.html

[5] http://protege.stanford.edu/

aplicativos que podem ser integrados e que permitem melhorar ou acrescentar funcionalidades a esta ferramenta (Tiago Semprebom; Marcus Yuzuru Camada; Igor Mendonça 2007).

2.5.3 Sistemas inteligentes nas ontologias

As ontologias são um elemento chave para a construção de sistemas abertos e dinâmicos. As ontologias são muito úteis no caso de existirem sistemas inteligentes que consigam inferir acerca da ontologia (Harry Chen 2005).

Existem dois tipos de sistemas inteligentes, os sistemas convencionais que resolvem um determinado problema específico com algoritmos específicos e os sistemas especialistas que resolvem problemas complexos, que não podem ser representados num algoritmo convencional (Darcilene Estrela, Helder Aragão et al. 2006). Os motores de inferência são classificados como sistemas especialistas pois conseguem raciocinar acerca de um problema tendo por base um conjunto de regras (ontologia), podendo ainda ter o apoio de conhecimento anterior.

Estes sistemas inteligentes têm de conseguir aplicar conhecimento e raciocínio para conseguir tirar conclusões acerca do contexto local e partilhar essa informação. Os sistemas inteligentes podem ser considerados de dois tipos, mundo aberto e de mundo fechado. De uma forma simples, os sistemas de mundo aberto assumem que tudo é verdadeiro desde que não seja provado que é falso. Os sistemas de mundo fechado consideram falso, o que não seja explicitamente provado como verdadeiro (Yuan Ren 2010).

Existem dois métodos para conseguir obter elações acerca dos factos, o *forward chaining* e o *backward chaining*. O *forward chaining* inicia nos factos até chegar a uma ou mais conclusões. O *backward chaining* inicia na conclusão a que se pretende, tentando encontrar factos que suportem o problema. O *forward chaining* é um método mais apropriado para se encontrar a solução de um problema, sem que se tenha a noção de qual irá ser o resultado final. O *backward chaining* é mais apropriado para resolver problemas que já se conheça a solução mas não se tenha os factos que a suportem (Hinkelman 2010).

Existe ainda um método que combina o *forward chaining* e o *backward chaining* que é o *mixed-mode chaining.* Este método combina as soluções anteriores de três formas. A primeira é de dar prioridade ao *backward chaining,* só utliliza o *forward chaining* quando não é possível avançar mais com o método *backward chaining.* A segunda é de dar prioridade ao *forward chaining* que funciona de forma inversa do primeiro método. O terceiro modo é de alterar o modo consoante seja conveniente para a resolução do problema *(Cardoso 2000).*

Neste capítulo foram dados a conhecer os conceitos e trabalhos relevantes na área. Foi dado a conhecer qual a evolução que decorreu ao longo do tempo na melhoria da qualidade de vida dos idosos, e apresentado o conceito de gerontecnologia, mostrando, alguns dos sistemas que forma criados ao longo do tempo, com o intuito de melhorar a qualidade de vida dos idosos. Foi apresentado o conceito de interoperabilidade de sistemas, e quais as condições que levaram ao seu desenvolvimento, mostrando como é que algumas tecnologias foram utilizadas para resolver problemas de interoperabilidade. Um caso particular de interoperabilidade estudado foi o protocolo de comunicação existente na área da saúde, o *Health Level Seven* (HL7). Foi mostrado porque surgiu, quais os seus objectivos e qual a sua evolução ao longo do tempo, de forma a adaptar-se a novas necessidades. A arquitectura orientada a serviços (SOA), aumenta a produtividade dos serviços prestados e permite seguir uma metodologia que atinge a interoperabilidade. Utilizando o paradigma orientado a serviços é composto por oito regras que ajudam a tornar uniforme a lógica da solução. Para concluir este capítulo foi introduzido o conceito de ontologia, qual a sua finalidade, algumas linguagens para a sua construção, alguns editores e apresentado o que são os motores de inferência.

Tal como em todos os sistemas, há um conhecimento que tem de ser representado. Para se representar esse conhecimento, vai ser utilizada uma tecnologia de representação semântica complexa. Assim a representação do conhecimento vai ser efectuada utilizando as ontologias, pois além de permitirem uma grande expressividade, ainda é a utilização de motores de inferência que conseguem tirar conclusões acerca do conhecimento existente.

Com a conclusão da apresentação dos conceitos importantes para o desenvolvimento da aplicação que consiga interligar diversos sistemas, é apresentado no capítulo seguinte a metodologia de investigação seguida para o desenvolvimento deste projecto.

3 Metodologia

Tendo sido feita uma contextualização dos principais conceitos relevantes para o presente trabalho, são apresentadas neste capítulo as metodologias utilizadas. Para o desenvolvimento do centro de controlo foram seguidos os seguintes passos: formulação da questão de investigação, levantamento de requisitos, desenho detalhado da solução, desenvolvimento da solução, testes da aplicação e por fim os resultados e discussão de resultados. Para o desenvolvimento da ontologia, abordou-se uma metodologia específica, designada de *101 Ontology Design Methodology(Noy e Mcguinness 2000).*

3.1 Metodologia utilizada para o desenvolvimento do centro de controlo.

Antes de começar o desenvolvimento, foi necessário formular o problema que levou ao desenvolvimento desta investigação. A formulação do problema deve ser feita como uma questão, de forma clara e precisa. O problema proposto deve de ser susceptível de solução e delimitado a uma dimensão mensurável. Utilizando estes conceitos para formular o problema consegue-se uma questão que seja perceptível que tenha uma solução com as tecnologias conhecidas e previamente estudadas. A questão tem de respeitar os meios disponíveis aquando da investigação. Após ter a formulação do problema, foi possível iniciar o processo de levantamento de requisitos.

O levantamento de requisitos é uma etapa essencial no desenvolvimento de um projecto. Esta etapa pode decidir o futuro do projecto. Os requisitos dum sistema têm de estar dimensionados, tanto as necessidades do projecto como ao tempo que lhe é dedicado.

Segundo um relatório da *Standish Group* 44% dos projectos estudados estavam mal dimensionados. Como consequência estes projectos atrasaram, ou ficaram mais caros que o esperado ou viram os seus requisitos reduzidos (Group 2009). Para conseguir efectuar um levantamento de requisitos de forma eficiente, cumprindo todas às necessidades do projecto, recorreu-se a técnicas de levantamento de requisitos. As técnicas utilizadas foram o questionário, a revisão de documentação e o *brainstorming*.

O questionário (que pode ser consultado no anexo I) permitiu organizar uma serie de questões, que foram direccionadas a grupos de trabalho com necessidades distintas. Esta técnica tem a vantagem de abranger um grande número de pessoas e de trabalhar por amostragem. Destas questões resultaram os requisitos que cada grupo de trabalho necessitava. A técnica de revisão de documentação permitiu obter requisitos necessários de documentação, manuais de projectos e relatórios. A documentação pesquisada foi orientada a projectos idênticos, onde existem requisitos comuns. A técnica de *brainstorming* foi utilizada para obter requisitos criativos. Para finalizar o levantamento de requisitos, foram apresentados a todos os grupos de trabalho, com o intuito de os dar a conhecer e informar as funcionalidades que iriam ser desenvolvidos.

Após se ter elaborado uma lista de todos os requisitos necessários para o desenvolvimento da aplicação, foi possível iniciar o processo de desenho detalhado da solução. Nesta fase foi descrito a constituição do centro de controlo, bem como a sua arquitectura. Desta forma foi possível construir um desenho da arquitectura física e lógica do centro de controlo. Também foi nesta fase que foram detalhados os sub-módulos gestor de informação e gestor de alertas, descrevendo as suas arquitecturas. Foram construídos os diagramas de casos de uso, os diagramas de classes, os diagramas de actividades, os diagramas de relacionais, e o protocolo de comunicação. Todo este detalhe permitiu, descrever como é que as funcionalidades da aplicação foram desenvolvidas.

Para iniciar o processo de desenvolvimento da aplicação, foram escolhidas as tecnologias que permitem concretizar o desenvolvimento da aplicação. Nesta etapa, foram realizados alguns testes. Por conseguinte, a fase de desenvolvimento incluiu também a fase de testes. Pois ambas decorreram em simultâneo. Para realizar os testes foi necessário desenvolver uma aplicação que permitisse simular a chamada dos serviços desenvolvidos.

Esta metodologia não poderia estar concluída sem que este protótipo não fosse testado num ambiente real. Os resultados obtidos e a discussão desses resultados foram a etapa final da metodologia.

3.2 Metodologia para o desenvolvimento de ontologias

Para o desenvolvimento da ontologia foi seguida uma metodologia que permitiu, não só desenvolver a ontologia, como também possibilitou a aprendizagem de utilização da ferramenta Protégé. Mas antes de apresentar a metodologia utilizada, são apresentados os princípios para o desenvolvimento de ontologias proposto por Gruber (Gruber 1993). Os critérios que Gruber definiu, são aceites pela comunidade das ontologias. Os conceitos propostos foram:

- Claridade - As definições devem ser formais, completas, objectivas e independentes do contexto social ou computacional. Assim são restringidas as várias interpretações possíveis para um conceito, melhorando a eficiência de comunicação entre agentes;
- Coerência - Apenas devem ser permitidas inferências consistentes com as definições existentes;
- Extensibilidade - A ontologia deve de ser projectada de forma a antecipar as tarefas futuras. Deve de ser possível expandir a ontologia, sem alterar as definições existentes;
- Codificação minimizada - A conceptualização deve de ser especificada ao nível do conhecimento. As escolhas de representação não se devem depender da notação a utilizar, ou dos problemas a nível tecnológico;
- Compromisso ontológico mínimo - A ontologia desenvolvida não deve de definir um domínio de forma tão específica que não consiga ser aplicada a domínios idênticos.

A metodologia utilizada para construir a ontologia foi a "*101 Ontology Design Methodology" (Noy e Mcguinness 2000).* Esta metodologia fornece um tutorial que foi desenvolvido por *Noy e McGuiunness* em 2000. O tutorial é um guia prático para o desenvolvimento de uma ontologia no editor Protégé. Esta abordagem é útil para o inicio da construção das ontologias,

sendo a explicação específica para o desenvolvimento apenas numa ferramenta. A vantagem desta metodologia é a de direccionar os seus leitores para um caminho rápido de desenvolvimento pois é explicado não só o processo de captura de conceitos, propriedades e restrições, como também a utilização de uma ferramenta específica.

O método proposto por *Noy e McGuiunness* segue um método de engenharia de conhecimento, através do processo iterativo e de refinamento que permite a construção das ontologias. As autoras definiram três regras que consideram fundamentais, para auxiliar no processo de decisão para a concepção. A primeira é "Não existe nenhuma maneira de modelar um domínio. Há sempre alternativas viáveis"; A segunda é "O desenvolvimento de uma ontologia é um processo iterativo"; e por último a terceira é "Os conceitos numa ontologia devem ser aproximados a objectos (físicos ou lógicos) e a relação no domínio tem de ter significado. Os conceitos têm tendência a ser nomes (Objectos) ou verbos (relações) em frases que descrevam o domínio."

O tutorial desenvolvido pelas autoras *Noy e McGuiunness* guiam o utilizador passo a passo no processo de desenvolvimento da ontologia. A implementação da ontologia é feita através de classes, restrições e propriedades. Este tutorial identifica sete actividades distintas no processo de concepção da ontologia que são: primeiro, identificação do domínio e âmbito da ontologia; segundo, reutilização de ontologias existentes; terceiro, enumeração dos termos importantes da ontologia; quarto, definição de classes e hierarquias; sexto, definição das propriedades das classes; e por ultimo o sétimo, definição de restrições.

O método proposto por *Noy e McGuiunness* é simples e prático, permitindo a aprendizagem rápida dos conceitos de construção de uma ontologia utilizando a ferramenta Protégé.

Este tutorial é prático e essencial para iniciar o desenvolvimento de ontologias, definindo o domínio no âmbito dos vinhos. Apesar dos seus conceitos estarem correctos e actuais, a versão da ferramenta Protégé utilizada nesta metodologia é antiga. Para melhorar a metodologia utilizada, sem a alterar, foi também utilizado o tutorial "*A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools*" (Horridg 2011). Este tutorial segue a mesma metodologia, estando a versão deste tutorial actualizada.

Neste capítulo foram apresentadas as metodologias utilizadas para o desenvolvimento de todo o projecto. Para o desenvolvimento do centro de controlo, apresentou-se uma

contextualização do metodo, desde como surgiu, como evoluiu e como é utilizado para solucionar problemas. Optou-se pela utilização do método científico, pois é o que permite formular o problema, e solucionar o mesmo. Para o desenvolvimento das ontologias foi utilizada a metodologia "*101 Ontology Design Methodology*", pois esta permite uma rápida aprendizagem dos conceitos, para o desenvolvimento da ontologia, bem como explica a forma de funcionamento de um programa para o seu desenvolvimento. Com a aplicação desta metodologia foi possível iniciar o desenvolvimento da aplicação, começando-se por apresentar quais as tecnologias escolhidas e como é constituído o centro de controlo.

4 Trabalho realizado

Para o desenvolvimento do centro de controlo foi definida uma metodologia que foi apresentada no capítulo anterior. Neste capítulo apresentam-se os objectivos do centro de controlo (CC) e qual a sua arquitectura desenvolvida. Sendo um servidor que presta serviços, é proposto no desenho da sua arquitectura física, a replicação de servidores, para garantir disponibilidade de serviço. Com o conhecimento dos objectivos do CC, é possível introduzir os requisitos necessários para o seu desenvolvimento. Para o desenvolvimento do CC, são apresentadas as principais opções tecnológicas subjacentes ao seu desenvolvimento.

Conhecendo os objectivos, a constituição, os requisitos e as metodologias, é possível descrever qual a proposta da arquitectura de alto nível do Centro de controlo. A arquitectura apresentada é baseada em serviços, utilizando os padrões de desenho SOA. Para o desenvolvimento da arquitectura ainda são apresentados os diagramas de casos de uso, os diagramas de actividades, os diagramas relacionais e o protocolo de comunicação a ser utilizado. Com o avançar da proposta de arquitectura, foi necessário representar conhecimento. Para se expressar conhecimento, foi utilizada uma ontologia, pois é possível efectuar relações entre as classes, definidas por regras, consegue representar a semântica do conhecimento e ainda é possível inferir acerca do domínio da ontologia. Para finalizar este capítulo, é apresentado o resultado obtido com o inicio da implementação desta arquitectura.

4.1 Objectivo do centro de controlo

O CC tem como objectivo interligar diversos módulos de forma a melhorar a qualidade de vida dos idosos. Inicialmente, o CC vai ser construído por três sub-módulos, o gestor de

informação (GI), o gestor de alertas (GA) e o gestor de contactos (GC). Futuramente, é possível acrescentar novos módulos, garantido a escalabilidade do CC. Para garantir a escalabilidade do projecto, cada sub-módulo é considerado um módulo independente que fornece serviços. Isto possibilita integrar novos módulos que sejam necessários para o CC que não tenham sido considerados neste projecto, garantindo assim a sua escalabilidade.

Como o módulo CC tem como intuito interligar diversos sistemas, que tem a finalidade, a melhoria do estado biopsicossocial do idoso, é necessário garantir confiabilidade e disponibilidade. Deste modo o tópico de tolerância a falhas assume particular relevo. Existem duas técnicas de tolerância a falhas, que são de duas classes disjuntas, o mascaramento e a detecção, localização e reconfiguração. O mascaramento impossibilita que as falhas se manifestem pois estão mascaradas na origem. Esta técnica utiliza muita redundância. Esta técnica é muito utilizada em sistemas de tempo real críticos pois não é dispendido tempo a detectar, localizar e corrigir.

O mascaramento de falhas permite que a aplicação consiga responder de forma correcta mesmo na presença de falhas. A falha não se chega a manifestar como erro e o sistema não chega a falhar ou a responder de forma incorrecta. Assim é dispensável detectar, corrigir e restaurar o sistema. No entanto podem ocorrer falhas permanentes. Estas falhas têm de ser corrigidas, pois afectam o normal funcionamento do sistema. Os mecanismos usuais de mascaramento de falhas são a redundância (replicação de componentes), codificação com *Error Correction Codes* (ECC) e blocos de recuperação.

Propõe-se a utilização de técnicas de redundância de *hardware para o CC.* A redundância de *hardware* divide-se em dois, a redundância estática e activa.

A redundância de *hardware* estática, é utilizada para mascarar falhas. As várias máquinas executam a mesma tarefa e o resultado é determinado por votação. Para utilizar este conceito é utilizado o *N-modular redundancy* (NMR). O NMR coloca N resultados de um processamento e define o resultado por votação da maioria ou por valor médio. (Lyons e Vanderkulk 1962). Como o processo de voto, está em serie com os módulos de *hardware*, este é o ponto crítico do sistema. Para minimizar as possíveis falhas, o voto é efectuado por *software*, possibilitando assim melhorar o sistema de voto e ainda detectar a ocorrência de possíveis erros. A redundância activa, detecta, localiza e recupera o sistema. Esta redundância

tem como finalidade resolver as falhas e não mascarar falhas. A estratégia utilizada é a de redundância *hot Standby*. A redundância *hot standby* coloca um controlador primário, que executa todas as tarefas, enquanto um controlador secundário está apenas sincronizado com o primeiro. O controlador secundário permanece sempre pronto a assumir o processamento no caso de ocorrer alguma falha. A troca de controladores é feita de forma automática, sem interromper o normal funcionamento do sistema. É então necessário resolver o problema do primeiro controlador para voltar a colocar o sistema a funcionar com os dois controladores.

Para manter a alta disponibilidade dos serviços do CC, propõe-se a utilização de servidores replicados e em paralelo. Para distribuir o processamento entre os blocos de servidores vão ser utilizados balanceadores de carga. Assim consegue-se manter os servidores balanceados entre eles. Os vários blocos de servidores vão estar localizados em posições geográficas distintas, evitando assim uma quebra de serviço no caso da ocorrência de uma catástrofe localizada.

Apesar de distribuir a carga pelos vários servidores, pode ser ainda possível que os servidores não consigam responder em tempo útil aos pedidos efectuados. Para garantir resposta às mensagens urgentes, existe um bloco de servidores, que se destina apenas a responder aos pedidos urgentes. Este bloco de servidores corre um sistema de tempo real. Este sistema garante o envio de uma resposta ao cliente num determinado tempo útil. A resposta enviada pode não ser o resultado do processamento do pedido, mas sim um aviso que o seu pedido não foi executado.

Para armazenar dados, são utilizados servidores de base de dados. Estes servidores encontram-se replicados em diferentes localizações geográficas. O acesso a base de dados é efectuado por um balanceador de carga que distribui a carga entre os servidores da base de dados. Para aumentar a disponibilidade do centro de controlo a informação disponível nas base de dados encontra-se replicada. Consegue-se assim garantir a falha de um servidor de base de dados, sem existir quebra de serviços.

Cada bloco de servidores, vai fornecer os mesmos serviços, prestados pelo módulo CC. O módulo CC vai disponibilizar os serviços dos sub-módulos gestor de informação, gestor de alertas e o gestor de contactos. Com estas especificações foi possível desenhar a arquitectura física do CC que se encontra na Figura 8.

O sub-módulo GI vai interligar as informações dos módulos que a ele se queiram ligar. Distingue-se do que já existe pois apesar de necessitar de funcionalidades de comunicar com diversos sistemas encontra-se uma camada acima, não tendo como função a gestão de ligações, nem a de fornecer diversas formas de ligação, mas sim a de fornecer um serviço de gestão de informação (criar estruturas, inserir dados, organizar, fornecer informação). Futuramente poderá ser implementado um *Enterprise Service Bus* (ESB) de forma a conseguir fornecer outras formas de comunicação que não só os *web-services*.

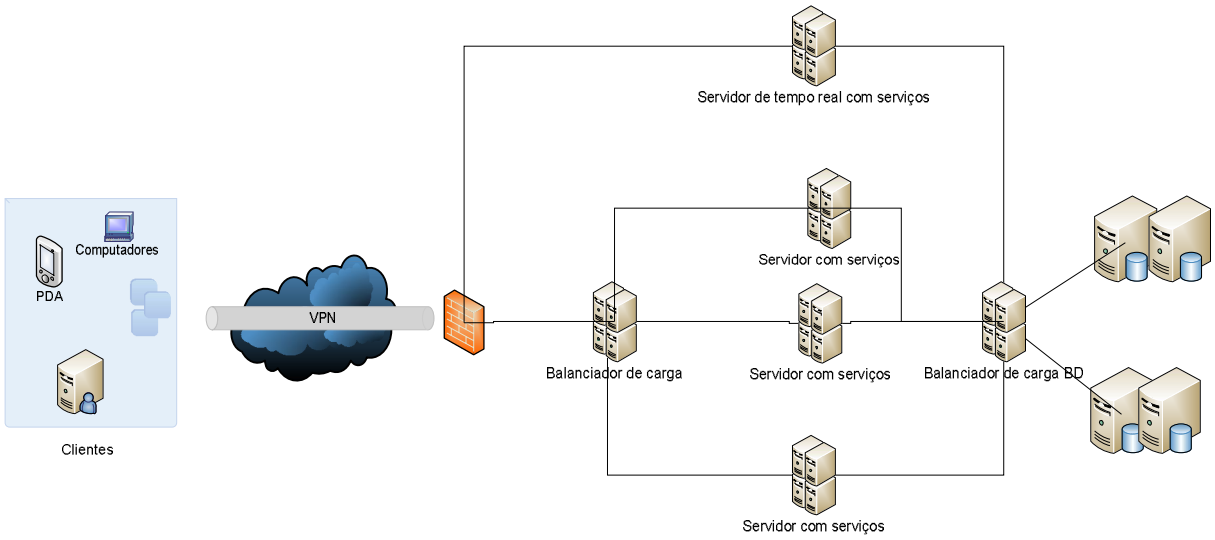


Figura 8 - Arquitectura física do Centro de Controlo

O sub-módulo GI vai possibilitar a nós sensoriais guardar a informação dos sensores que a ele estão acoplados, tendo uma estrutura pré feita para esse efeito. Para esta estrutura é necessário indicar por cada sensor quais os campos que pretende guardar e qual o tipo de dados que guarda. Internamente é criada uma tabela que possibilita o armazenamento dos dados enviados pelo sensor ficando registado quais os dados que o sensor enviou, isto possibilitam a vários sensores de um nó sensorial guardar informação numa mesma tabela.

O sub-módulo GA vai receber alertas e eventos. Tanto os alertas, como os eventos são armazenados na base de dados para futuras análises e criar um histórico por idoso. Com a existência de um histórico por idoso, é possível correlacionar os eventos lançados, com as condições de saúde do idoso. Esta funcionalidade permite classificar de forma mais rigorosa

os alertas lançados. Os eventos são armazenados e são disponibilizados para um módulo exterior que procede à sua classificação. Os eventos podem gerar alertas ou falsos positivos. Os falsos positivos são eventos que são recebidos, e que após classificação não geram nenhum alerta. Os eventos que geram alertas, são classificados e é dada uma resposta, consoante a prioridade que foi atribuída ao alerta.

O sub-módulo GC vai conter uma lista de contactos por idoso. Cada lista de contactos tem definido vários grupos atribuídos pelos idosos, para facilitar a pesquisa de contactos. Esta lista vai estar disponível para os módulos exteriores que forneçam serviços de comunicação.

4.2 Levantamento de Requisitos

Nesta secção são apresentados os requisitos necessários para o desenvolvimento do centro de controlo e dos seus três sub-módulos. Ainda são apresentados os requisitos que resultaram dos questionários efectuados aos restantes grupos de trabalho. Os requisitos do módulo centro de controlo estão classificados em três níveis, indispensável, necessário e opcional.

Código	Descrição	Prioridade
Requisitos para o Gestor de Informação		
RIM 001	Os sensores têm de ter permissões.	2
RIM 002	Definir tipo de permissões.	3
RIM 003	A informação não é apagada da base de dados.	2
RIM 004	Todos os dados têm o registo do sensor/utilizador que efectuou a alteração.	2
RIM 005	Todas as alterações efectuadas têm um registo temporal (data/hora).	3
RIM 006	Existem permissões por defeito para os sensores e outras para os utilizadores.	3
RIM 007	Existe uma tabela por cada grupo de sensores.	2
RIM 008	Grupos de sensores são sensores que registam a mesma informação.	

RIM 009	O sistema tem que ter uma implementação redundante.	2
RIM 010	O web-service envia a informação para 2 ou mais base de dados, para garantir que não existe falhas.	3
RIM 011	A comunicação dos pedidos é feita para um sistema que gere qual a base de dados tem menor carga. A resposta é enviada pelo sistema que gere as respostas dessa base de dados. Caso exista um erro, o sistema pode pedir informação à outra base de dados.	3
RIM 012	Existe uma tabela com a identificação de todos os sensores adicionados.	2
RIM 013	O sistema tem que criar uma tabela com as funções dos grupos de sensores.	2
RIM 014	O sistema tem que ser capaz de catalogar (consultar, inserir, actualizar e eliminar) qualquer tipo de equipamento fornecido pelo módulo dos sensores.	1
RIM 015	O sistema tem que ser capaz de comunicar com qualquer um dos outros módulos.	1
RIM 016	O sistema tem que ser capaz de interpretar os dados recebidos de um módulo e garantir que esses dados são capazes de ser interpretados por outros módulos.	1
RIM 017	O sistema tem que ser capaz de registar novos sensores.	1
RIM 018	O sistema tem que ser capaz de autenticar sensores.	1
RIM 019	O sistema tem que ser capaz de autenticar módulos.	1
RIM 020	O sistema tem que ser capaz de autenticar utilizadores	1
RIM 021	O sistema tem que ser capaz de bloquear um sensor (marcar como desactivado/ corrompido).	1
RIM 022	O sistema tem que ser capaz de bloquear módulos.	1
RIM 023	O sistema tem que ser capaz de bloquear utilizadores.	1
RIM 024	O sistema tem de possibilitar a consulta da informação por um módulo autenticado.	1
RIM 025	O sistema tem que ser capaz de permitir a edição das permissões dos sensores/utilizadores.	1
RIM 026	A comunicação é efectuada por web-service.	
RIM 027	O sistema permitirá configurar (consultar, inserir, actualizar e eliminar) os horários padrão do idoso, a nível de permanência e ausência do lar.	1
RIM 028	O sistema armazenará a informação clínica do idoso, bem como os seus hábitos diários.	1

RIM 029	O sistema permitirá consultas aos dados por parte do SNS, sempre que autorizado pelo idoso.	2
RIM 030	O sistema tem que poder guardar um histórico dos alertas recebido pelo gestor de alertas.	2
Requisitos para o Gestor de Contactos		
RCC 001	O sistema tem que poder criar, inserir, consultar, eliminar e editar uma lista de contactos.	1
RCC 002	O sistema tem que poder mandar SMS para membros de uma lista de contactos.	1
RCC 003	Uma lista de contactos tem que ter os atributos: nome, tipo, contactos.	
RCC 004	Um contacto tem que ter os atributos: nome, relação, telefone.	
RCC 005	O sistema tem que ser capaz de criar, inserir, consultar, eliminar, editar e pesquisar os perfis de utilizadores.	1
RCC 006	Um cliente tem que ter a possibilidade de escolher a visibilidade do seu perfil.	2
RCC 007	A visibilidade pode ter os seguintes valores: público, privado ou uma lista personalizada pelo cliente.	
RCC 008	A visibilidade dos serviços terá o valor de público por definição.	
RCC 009	Um perfil tem de ter os atributos: nome, data de nascimento, morada, password, nível acesso, visibilidade.	
RCC 010	O sistema deverá permitir efectuar a solicitação de serviços. Nomeadamente, compras online, reparações domésticas (electricista, canalizador, abastecimento gás, Pedido de transporte aos bombeiros), da marcação de consultas no centro de saúde a que o idoso pertence.	1
RCC 011	O interface do sistema tem que ser adequado a idosos que podem ter vários problemas auditivos, de visão, ou de movimento, bem como analfabetismo e/ou iliteracia digital.	1
RCC 012	O sistema deverá permitir a realização de chamadas VoIP com qualquer pessoa da lista de contactos do cliente.	1
RCC 013	O sistema deverá ter uma opção para alertar a assistência técnica.	1
RCC 014	O sistema deverá ter uma opção de fácil e rápido acesso para chamar assistência médica, com os dados necessários para a chamada previamente gravados de forma a apenas ser necessário carregar no botão e o resto da chamada ser feita automaticamente.	1
RCC 015	O sistema deverá possibilitar ao cliente consultar um médico em horários pré-determinados de forma a poder esclarecer dúvidas de foro médico.	2
RCC 016	Marcar visita de enfermaria.	1
Requisitos para o Gestor de Alertas		

RAL 001	Existem vários níveis de alerta que vão desde situação normal (nível 0), até emergência médica (nível máximo), com vários níveis intermédios.	1
RAL 002	Um dos níveis de alerta intermédios é para o alerta para situações que não são de urgência.	1
RAL 003	O alerta tem que alertar sempre alguém em caso de emergência médica.	1
RAL 004	Os vários níveis de alerta têm uma lista de contactos cada um, sendo que o nível máximo terá sempre o contacto do número de emergência nacional (?).	1
RAL 005	Existem vários alertas, um para cada atributo vital a monitorizar.	1
RAL 006	Existem também alertas para situações que não são de urgência, como por exemplo lembretes para tomar remédios.	1
RAL 007	O sistema tem que permitir criar, inserir, consultar, eliminar e editar um alerta.	1
RAL 008	O sistema tem que permitir guardar os alertas recebidos na base de dados.	1
RAL 009	O sistema tem que permitir que a família do cliente também receba os alertas.	1
RAL 010	O sistema tem que garantir a qualidade dos serviços em tempo real.	1
Requisitos para o módulo BodyMonitor		
RBM 001	Nó sensorial pode ter mais do que um sensor que regista atributos diferentes.	
RBM 002	Um nó sensorial é identificado univocamente por um id numérico de 16 bits.	
RBM 003	Um nó sensorial recolhe os dados dos seus vários sensores, efectua uma pré-avaliação dos dados e decide se é motivo de alerta, e se essa decisão for afirmativa, envia um alerta.	
RBM 004	Um nó sensorial pode recolher dados para efeitos estatísticos e envia-os consoante a necessidade.	
RBM 005	Um nó sensorial deve armazenar o ID do nó, data do alerta, tipo de alerta e os dados recolhidos pelos sensores.	
RBM 006	A periodicidade do envio dos dados deve ser a mínima possível devido ao consumo de recursos elevado em cada transmissão.	
RBM 007	Existirá um concentrador de informação na forma de um gateway (casa) ou de um telemóvel.	
RBM 008	Existem dois graus de alerta: Check, que são valores que necessitam de uma melhor análise antes de serem definidos como alerta e Critical que são valores já analisados e definidos como alerta.	
RBM 009	A arquitectura tem que ser modular de forma a permitir a integração de qualquer tipo de sensor e independente de qualquer tecnologia usada para a implementação.	
RBM 010	Tipos de sensores passíveis de serem implementados são: acelerómetro, termómetro, ECG, medidor de nível de oxigénio e medidor de nível de	

	açúcar no sangue.	
Requisitos para o módulo Segurança		
RMS 001	A ligação entre pontos é feita com recurso a SOAP 1.2.	
RMS 002	A segurança da ligação é feita com recurso a SSL.	
RMS 003	Os certificados terão para começar uma chave de 128 bits com o período de renovação a ser definido no futuro.	
RMS 004	Uma comunicação stateless é a mais adequada se o overhead adicionado não for significativo.	
RMS 005	A segurança da ligação é independente das tecnologias a utilizar pelos outros módulos.	
RMS 006	A comunicação será feita através de XML.	
Requisitos para o módulo Análise Inteligente de Dados		
RAD 001	Cada cliente necessita armazenar: nome, tipo de mobilidade que lhe está associada - conduz, usa transportes públicos.	
RAD 002	A agenda pessoal armazena: data, hora, actividade prevista, local.	
RAD 003	Tem que guardar um histórico de actividades (todos os dados da agenda para efeitos de análise histórica).	
RAD 004	Cada contacto necessita armazenar: nome, email, telemóvel, endereço postal.	
RAD 005	Um alerta tem uma lista de contactos associada.	
RAD 006	Alertas variam entre – aviso, grave, muito grave.	
RAD 007	É necessário armazenar o histórico de ocorrências (alertas efectivos, falsos alertas, ...).	
RAD 008	É necessário armazenar informação de georeferenciação adicional (além da disponível no Google Maps).	
RAD 009	A informação que é necessário consultar, é a mesma que é necessário armazenar.	
RAD 010	Os dados são armazenados localmente e há sincronizações periódicas com o servidor (GI).	
RAD 011	Aplicação para dispositivos móveis desenvolvida para Android.	

Tabela 2 - Lista de requisitos do Centro de Controlo

Valor	Descrição
1	Indispensável
2	Necessário
3	Opcional

Tabela 3 – Lista de prioridades

4.3 Tecnologias escolhidas

Após ter sido apresentada a lista de requisitos necessária para o desenvolvimento do centro de controlo, vão ser apresentadas as várias possibilidades de implementação deste *middleware*, bem como das tecnologias utilizadas no seu desenvolvimento.

Para a criação das mensagens na comunicação entre módulos foram ponderados alguns métodos. Os métodos ponderados foram: mensagens HL7 e mensagens XML construídas para a comunicação entre módulos. As mensagens HL7 são uma opção viável, pois já existe uma norma para as mensagens, é possível agregar a informação trocada dentro das mensagens, ainda tem a vantagem de existir uma grande comunidade que já conhece as mensagens. Estas mensagens não vão ser utilizadas pois iria trazer um acréscimo de complexidade ao desenvolvimento do módulo CC, sem conseguir preencher todos os requisitos necessários entre os vários sistemas. Estas mensagens são um *standard* na comunicação entre sistemas na área da saúde, mas quando usadas fora desse contexto, por exemplo nos dados de comunicação dos sensores com o CC, tornam-se insuficientes para a comunicação. Por fim as mensagens XML são a opção mais viável pois têm validação de mensagens, segue as normas de um padrão aberto, é perceptível por humanos e por máquinas e ainda tem uma comunidade a trabalhar na evolução do XML.

Estando escolhida ao tipo de mensagem a trocar nas comunicações, tem de se ponderar qual a tecnologia que se utiliza para essa troca de mensagens. Foi ponderado o CGI, recepção directamente por socket e Web-services. O CGI apesar de ser eficiente, foi abandonado por ser uma tecnologia antiga, utilizada para devolver páginas Web. Receber as mensagens directamente por socket implicava a criação de métodos que verificam o que chegou ao

socket, se essa informação era para utilização e verificar a consistência da mensagem. Estes métodos já se encontram implementados por outras tecnologias que garantem que as mensagens são bem recebidas. A solução foram os Web-Services, pois são independentes da tecnologia utilizada, utilizam o WSDL como especificação e têm a vantagem de utilizar de forma nativa o XML. Aquando da existência de sub-carga do sistema, as comunicações urgentes vão ser asseguradas com recurso aos *sockets*.

Como linguagem de programação foi escolhido o C# pois é uma linguagem que permite desenvolver aplicações bastante rapidamente, tendo a vantagem de ter um bom editor o *Visual Studio*. Como base de dados e como se vai iniciar em C# utilizamos o MSSQL pela facilidade de interligação com a linguagem de programação.

Foram apresentados os editores para o desenvolvimento da ontologia. Os editores apresentados foram o Otolingua, o WebODE, o SWOOP, o Altova SemanticWorks e o Protégé. De todas as ferramentas analisadas, não se conseguiu encontrar nenhuma que seja completa.

A ferramenta escolhida foi o Protégé, pois é das ferramentas que é mais utilizada no desenvolvimento de ontologias. Além de ser Open Source, consegue competir ou até ser melhor que algumas ferramentas proprietárias. É um editor intuitivo e existem vários plug-ins, que facilitam a criação e edição das ontologias. Como linguagem de construção da ontologia foi escolhido o OWL-DL pois permite uma expressividade máxima e ainda possibilitar efectuar inferências. As inferências vão ser efectuadas usando o motor de inferência Pellet. Este motor de inferência já se encontra adicionado no Protégé, não sendo necessário utilizar outros programas para efectuar questões há ontologia.

4.4 Proposta de arquitectura

Nesta secção vai ser apresentada a arquitectura de alto nível do Centro Controlo, quais são os benefícios na utilização dos padrões SOA e a sua aplicabilidade no gestor de informação (GI). Nesta secção também vai ser descrita a arquitectura do Gestor de informação e do Gestor de

alertas. Ainda é ilustrado o desenho da arquitectura lógica do GI resultante da aplicação dos padrões de desenho SOA.

4.4.1 Arquitectura do centro de controlo de alto nível

O Centro de Controlo (CC) é o módulo central do Elde Care e que inicialmente contém três sub-módulos, o GI, o GA e o GC. De forma a tornar o CC independente por cliente, é criado um processo por cliente, sendo que cada processo tem várias *threads* que partilham a mesma memória. Desta forma garante-se uma memória partilhada por processo, conseguindo uma independência entre a memória utilizada para cada cliente. Para aumentar a robustez do CC, todos os processos criados vão provir de um gestor, cuja função é criar, monitorizar, e armazenar os estados de cada processo. Assim no caso de um processo terminar de forma inesperada ou bloquear, é possível elimina-lo e reconstrui-lo, para que seja possível continuar a processar o pedido do cliente, de forma transparente. Com esta solução ainda é possível, criar uma rotina que verifique quais são os processos que estão a perturbar o normal funcionamento do sistema, forçando o seu término. Esta solução é transparente para o cliente, pois é criado um novo processo, com o mesmo estado do processo antigo.

Os clientes podem ser de três tipos, os clientes que consultam informação e os clientes que geram informação e os clientes que consultam e geram informação. Na Figura 9 é ilustrado como é constituído o processo, como estão ligados os três sub-módulos e como é que os clientes podem interagir com o CC.

Os geradores de informação podem criar as estruturas de dados para suportar informação. Estes também podem alterar a estrutura de dados criada por eles.

Existe o grupo de consultores de informação, estes vão consultar a informação depositada na base de dados, os geradores e os consultores podem ser a mesma entidade.

O GI é o sub-módulo do CC, responsável pela informação armazenada. O GI também tem como função interligar diversos sistemas externos. Para tal fornece serviços que possibilitam guardar informação, aceder à informação e converter a informação para que esta seja perceptível para vários sistemas. Um exemplo onde a informação vai ser convertida para o

protocolo HL7 e vice-versa. O módulo GI pode lançar eventos pré-programados pelo idoso tal como: aniversários, toma de medicação e outros que o idoso programe.

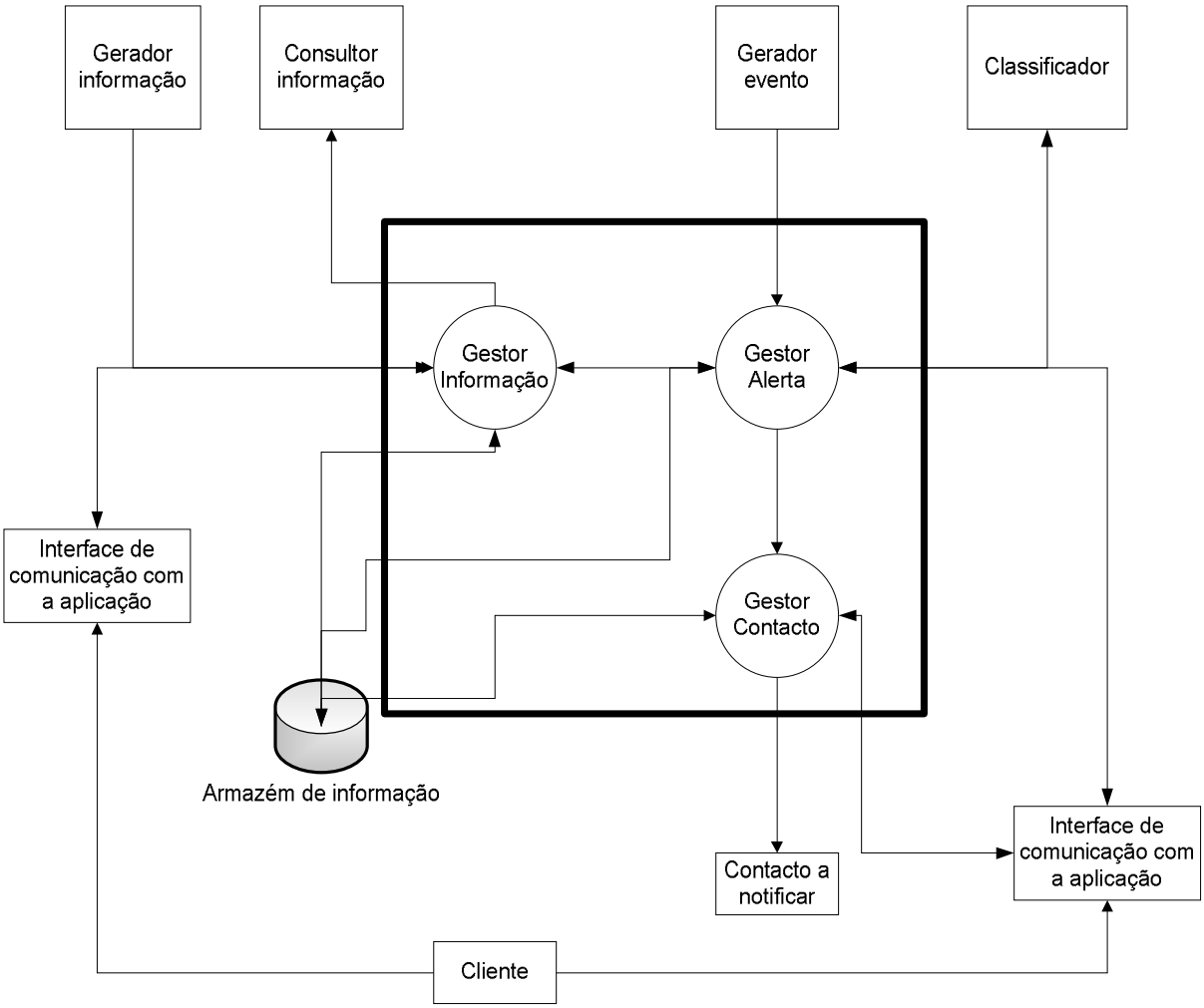


Figura 9 – Arquitectura de alto nível do Centro de Controlo

Para a interacção do cliente com a aplicação, vai existir uma interface de comunicação. Assim o cliente tem acesso à aplicação, sem ser de forma directa, sendo este acesso transparente para o cliente. Para efectuar as comunicações foi definido um fluxograma de comunicação que se encontra na Figura 10. Sempre que um sistema pretende comunicar com o CC, tem de verificar se o servidor se encontra activo, com o comando ping e o endereço do servidor. No caso de não obter resposta, deve de tentar novamente ou verificar outro endereço. No caso de

obter resposta, pode solicitar um serviço. Cada ligação apenas possibilita a utilização de um serviço. Um serviço pode utilizar uma composição de serviços.

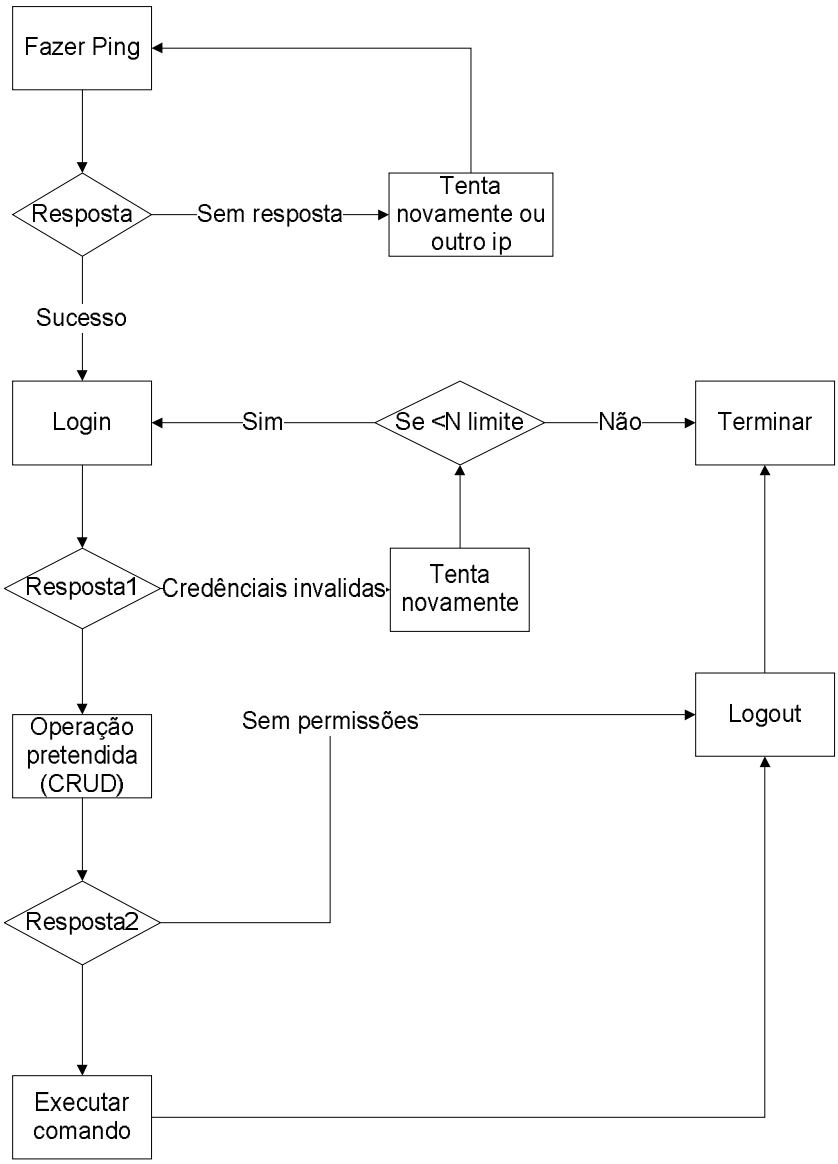


Figura 10 - Fluxograma comunicacional do Centro de Controlo

No caso da comunicação com o GC são efectuadas consultas de contactos ou pedidos de comunicação com alguém da lista de contactos.

A criação ou edição da lista de contactos do idoso tem de ser efectuada com recurso à *interface* de comunicação.

Todos os casos dos eventos e alertas são tratados pelo GA, podendo estes vir de entidades externas ou do módulo GI.

4.4.2 Benefícios da arquitectura orientada a serviços

Para chegar a atingir os benefícios que o SOA oferece é necessário que os serviços tenham determinado características, estas estão divididas em dois grupos, as primárias e as secundárias.

As características primárias que um serviço tem de ter são de ser serviços *Loosely Coupled*, ter os contractos de serviço bem definidos, tem que ser significativo para o utilizador do serviço e tem de ser baseado em padrões.

Um serviço deve ser o mais independente possível (*Loosely Coupled*), ou seja não deve depender de uma tecnologia específica, deve ser transparente em relação aos utilizadores e deve ser reutilizável em outros processos para além daquele para que foi desenhado. Deve de ter os contractos de serviço bem definidos pois um contracto de serviço é uma *interface* para o serviço que define as suas capacidades e como o invocar, separando assim o utilizador da implementação técnica do serviço. Tem que ser significativo para o utilizador do serviço, pois os serviços e os seus contractos têm que identificar de forma clara quais as suas finalidades, possibilitando alterar a implementação do serviço mantendo a sua lógica de funcionamento. Tem de ser baseado em padrões de forma a maximizar o número de pessoa que podem utilizar o serviço.

As características secundárias que um serviço tem de ter são: definir um *Service Level Agreements* (SLA) previsível, têm de ser dinâmicos, tem de ser de fácil descoberta e utilizar metadados, os desenhos dos contractos dos serviços, têm de ter em atenção os serviços relacionados, devem de ter uma implementação independente, devem de ter em consideração a necessidade de transacções de compensação, deve de ser desenhados para vários estilos de

invocação, os serviços não devem ter estado (*stateless*) e o desenho dos serviços têm de estar optimizados para melhorar a performance.

O serviço deve de definir um *Service Level Agreements* (SLA) previsíveis pois o SLA define as métricas de um serviço (tempo de resposta, disponibilidade, etc.). O serviço tem de ser dinâmico, de fácil descoberta e deve usar metadados, pois os serviços devem ser publicados de forma a poderem ser usados sem intervenção do fornecedor, e os seus contractos de serviço devem usar metadados para definir as capacidades do serviço. O desenho dos contractos de serviço tem de ter em atenção os serviços relacionados, facilitando assim a que os serviços tenham um modelo de dados consistente. A implementação deve de ser independente, pois os serviços devem ter o mínimo de dependências possível entre eles, especialmente em relação a dependências internas, o que irá possibilitar no futuro reaproveitar e combinar os serviços de formas diferentes. Considerar a necessidade de transacções de compensação o que possibilita a utilização de mecanismos de compensação no caso de ocorrer erros durante a execução do serviço. Desenhar para múltiplos estilos de invocação possibilitando invocar o serviço de várias formas diferentes (pedido/resposta, publicar/subscrever, assíncrona, etc.) que devem ser resolvidas. Os serviços não devem ter estado ou seja devem de ser (*stateless*) um serviço deve ser independente de quem o invoca e não deve manter informação específica do cliente entre invocações. Os desenhos dos serviços devem pensar na performance para que quem o invoca tenha uma resposta rápida independentemente da distância de quem o invoca. (Erl 2009)

Os serviços que possuam estas características têm os seguintes benefícios: desenvolvimento eficiente, maior reutilização, manutenção simplificada, adopção incremental, evolução graciosa.

4.4.3 Porquê a aplicação da arquitectura orientada a serviços no módulo gestor de informação?

Todas as arquitecturas orientadas a serviços apresentam problemas. Alguns destes problemas já foram identificados e apresentadas soluções práticas para os solucionar. A identificação e resolução destes problemas encontram-se identificadas nos padrões SOA. Como a

arquitectura do Gestor de Informação (GI) é orientada a serviços, é essencial identificar quais os problemas que podem surgir e aplicar as referidas soluções.

Para aplicar o paradigma de orientação a serviços à arquitectura do GI, tomou-se uma abordagem *top-down*, em que se completa uma análise de inventário. A análise de inventário produziu um esquema de serviços preliminar de alto nível, que poderá ou não ser refinado antes de passar a uma fase de implementação.

Da análise à proposta da arquitectura do GI, usando técnicas de extracção de informação e de *brainstorming*, definiram-se os seguintes inventários de serviços:

- Serviços de recepção de dados de monitorização, serviços de registo de dados de monitorização e de saúde, serviços de consulta de dados do idoso, serviços de integração de sistemas externos, serviços de armazenamento de dados, serviços de recolha de dados e serviços de validação de mensagens;
- Os serviços de recepção de dados de monitorização disponibilizam a funcionalidade de receber e processar dados dos módulos ligados ao GI com finalidade de monitorizar o idoso. A recepção dos dados é efectuada periodicamente ou em tempo real.
- Os serviços de registo de dados de monitorização e de saúde são responsáveis pelas funcionalidades de registar informações de saúde, agendar exames, emitir prescrições, entre outros;
- Os serviços de consulta de dados do idoso, permitem a especialistas de saúde e familiares aceder à informação do idoso;
- Os serviços de integrações de sistemas externos permitem às aplicações que pretendam, utilizar serviços disponibilizados por plataforma de outros sistemas. Estes serviços possibilitam a comunicação entre sistemas incompatíveis;
- Os serviços de armazenamento de dados possibilitam a criação de estruturas de dados e posteriormente o armazenamento de dados nessas estruturas;
- Os serviços de recolha de dados possibilitam a recolha de dados por utilizadores ou aplicações com permissões para recolher dados. É possível recolher os dados armazenados de outras aplicações, conhecer as estruturas de dados existentes;
- Os serviços de validação de mensagens possibilitam validar as mensagens que vão ser utilizadas para efectuar a comunicação. É assim possível verificar se a mensagem que se pretende trocar com outro sistema se encontra bem formatada;

Conhecendo o conceito da arquitectura SOA, os seus benefícios e a sua aplicabilidade no módulo GI, é possível então consultar o catálogo de padrões SOA existentes e aplicar os padrões aos serviços já disponibilizados. O resultado da análise dos padrões SOA aos serviços encontra-se no anexo II. Neste Anexo encontram-se ainda todos os princípios SOA que foram aplicados ao GI.

4.4.4 Arquitectura do gestor de informação

A arquitectura do GI foi desenhada para ser escalável, isto possibilita que qualquer tipo de sistema, que cubra uma área do bem-estar biopsicossocial, se consiga interligar ao GI. A arquitectura proposta consiste num conjunto de serviços que possibilitam: criar uma estrutura de dados, armazenar dados, interligar os vários sistemas externos ligados ao GI e compreender e utilizar a informação armazenada. A arquitectura proposta não tem em consideração os aspectos da segurança, uma vez que existe um grupo de trabalho, cuja tarefa é tratar dos aspectos da segurança de todo o projecto Elde Care.

Para armazenar a informação proveniente dos diferentes sistemas externos, é necessário criar/preparar uma estrutura que contenha os dados. Porém cada sistema externo tem necessidades diferentes. Para solucionar este problema, cada sistema externo cria uma estrutura, que albergue os dados necessários, utilizando metadados. Com esta solução, é permitido, a cada sistema, definir a estrutura de dados necessária, para armazenar os seus dados. A estrutura criada apenas pode conter tipos de dados inteiros, *string* e/ou *double*. A utilização dos metadados, permite a cada sistema, definir e criar a estrutura pretendida no GI. Para poder criar as estruturas para os vários sistemas, os metadados têm de conter informações sobre as tabelas a criar, os seus nomes, os seus campos, a definição dos campos e as conexões entre as tabelas. Esta é a informação necessária, para preparar uma estrutura de dados e permitir ao GI a recepção de dados. Todas as informações que um sistema externo envia para o GI, são armazenadas na estrutura criada para o sistema externo. A toda a informação inserida é adicionada a data e o utilizador, que fez o registo dos dados.

O sistema externo, pode precisar de efectuar alterações, à estrutura criada anteriormente. Dependendo das alterações necessárias efectuar, existe uma solução projectada. No caso de

ser necessário adicionar mais campos ou tabelas na estrutura, os metadados são actualizados e procede-se as alterações da estrutura, do sistema externo que se encontra no GI. As alterações efectuadas na estrutura e nos metadados são gravadas com o utilizador e a data da alteração. Se for necessário fazer alterações que exigem a eliminação de dados, existem duas possibilidades projectadas: a primeira, é de marcar os dados e a estrutura como apagados, procedendo então a criação da nova estrutura. A segunda é a eliminação da estrutura existente e de todos os dados nela contidos, e criar uma nova estrutura. A primeira opção é recomendada porque, embora os dados e a estrutura fiquem marcadas como apagadas, é possível restaurar os dados. Esta opção permite manter os dados, apesar de inacessíveis para os utilizadores. É assim criado um histórico dos dados que o sistema externo tem armazenado. A segunda elimina os dados e a estrutura de dados definitivamente. Após apagar ou marcar como apagado, a nova estrutura de dados pode ser criada.

Como o GI é uma plataforma integradora de sistemas externos, é necessário conhecer e dar a conhecer, quais são as funcionalidades, fornecidas pelos sistemas externos, que estão ligados ao GI. A informação das várias funcionalidades que cada sistema disponibiliza, permite ao GI, construir e disponibilizar uma lista, onde estão listados todos os serviços fornecidos. Esta lista de serviços é essencial para possibilitar a interligação de vários sistemas, pois, possibilita aos sistemas externos, conhecer e usufruir dos serviços disponibilizados. Os serviços fornecidos pelos sistemas externos, com o decorrer do tempo podem necessitar de actualizações. É então necessário notificar todos os sistemas, que estão a utilizar esse serviço, que decorreu uma actualização. A fim de resolver este problema foram equacionadas três possíveis soluções: a primeira é, "obrigar" os sistemas externos a fornecer um serviço fixo que disponibilize uma lista com todos os serviços prestados e que possibilite o acesso ao GI; a segunda é, fazer parte do processo de registo dos sistemas externos, o preenchimento de uma lista, que contenha todas as funcionalidades disponíveis e de como lhes é possível aceder. Esta lista tem de se encontrar actualizada. É necessário notificar o GI quando ocorre uma actualização; a terceira é, criar uma ontologia que permita compartilhar e questionar qualquer conhecimento no domínio GI.

A solução de garantir que o sistema externo, tenha de fornecer um serviço fixo onde são listados os seus serviços, não foi adoptada pois impossibilita actualizações. Desta forma os serviços não se podem adaptar a novas realidades que possam surgir. Outro ponto que excluiu

esta opção foi o de exigir ao sistema externo, a criação de um novo serviço fixo, unicamente para listar as funcionalidades disponibilizadas. A solução de fazer parte do processo de registo, ao preenchimento de uma lista de funcionalidades disponíveis, não foi adoptada, porque, esta lista pode não estar sempre actualizada, visto que esta lista não pertence ao domínio do GI. Esta opção tem ainda a desvantagem da construção de serviços e estruturas de dados, no GI somente para listar os serviços que os sistemas externos disponibilizam.

Ambos os métodos necessitam de notificar o GI da ocorrência de alterações, pois é mais eficiente, o sistema que efectua alterações, notificar o GI, que o GI despender recursos, para verificar a ocorrência de alterações. É também mais viável os sistemas externos, notificarem o GI, pois assim, as alterações na lista de serviços, são efectuadas aquando da notificação da alteração de serviços. No caso de ser o GI a verificar as listas dos sistemas externos, poderia existir um compasso de tempo, onde a lista de serviços poderia estar desactualizada. Esta situação poderia ocorrer quando um sistema externo modificasse a sua lista de serviços, imediatamente após a verificação da lista por parte do GI. A lista apenas iria ficar actualizada quando o GI verificasse novamente a ocorrência de anualizações. Como exemplo o GI verificava a lista das funcionalidades dos sistemas externos diariamente às 23h00m. Um sistema externo efectua as suas alterações e consegue publicar as mesmas as 24h00m. Como consequência a lista de serviços iria encontrar-se incorrecta até as 23h00m do dia seguinte.

A solução de criar uma ontologia foi adoptada, pois, fornece aos vários sistemas externos a informação necessária para interagir com os outros sistemas e que tipos de serviços estão disponíveis. Com uma ontologia é possível definir um conjunto de objectos que pertencem ao domínio do GI e quais as relações existentes entre esses objectos. Com este conjunto de objectos e relacionamentos entre os objectos é possível fazer inferências, sobre os objectos que pertencem ao domínio do GI. De forma simplificada, é fornecido aos sistemas externos uma ferramenta que possibilite questionar a ontologia. As questões podem responder a como interagir com os serviços prestados pelo GI, que serviços estão disponíveis e qual é o significado do conteúdo existente dentro do GI.

Com a utilização da ontologia, é possível extrair o conhecimento que se encontra guardado no GI, como por exemplo dados de monitorização recolhidos pelos sensores (temperatura corporal, ritmo cardíaco, tensão arterial), quais são os serviços disponibilizados pelo GI, quais são as doenças de cada idoso, entre outras). A ontologia possibilita aos sistemas externos, não

só, utilizar o GI como repositório de dados e como ferramenta de interligação entre sistemas, como é possível conhecer as estruturas de dados que cada sistema utiliza e qual o conteúdo semântico que se encontra nessas estruturas.

A ontologia possibilita a separação da lógica de negócio, da lógica dos dados. Assim é possível ter acesso à lista de serviços disponibilizada, às estruturas de armazenamento de dados e a conteúdo semântico que se encontra nas estruturas de dados, separada dos dados que se encontram armazenados no GI. Isto possibilita disponibilizar todo o conteúdo semântico do GI, sem que seja necessário aceder-lhe, mas sim acedendo à ontologia. Na Figura 11 é ilustrado um exemplo da interacção de um sistema externo, a questionar e recolher informação da ontologia sobre como utilizar os serviços prestados pelo GI.

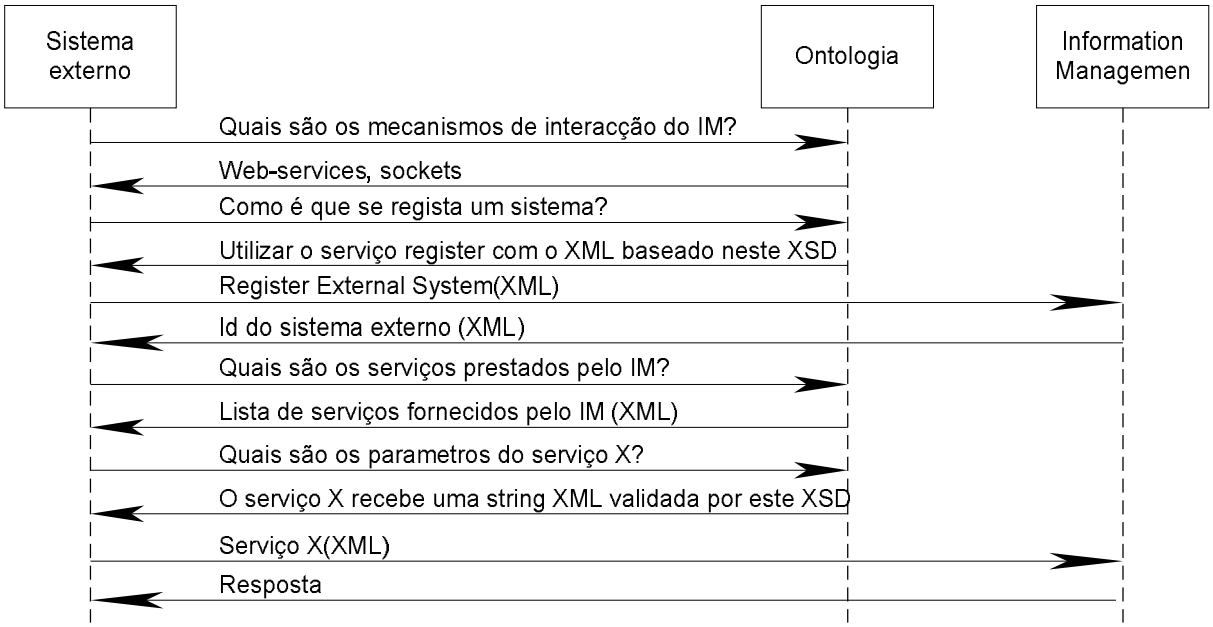


Figura 11 - Exemplo de um sistema externo a questionar a ontologia para usar os serviços disponibilizados pelo GI

Foi explicado, como é que esta arquitectura foi projectada para recolher, armazenar e partilhar conhecimento. É essencial saber quais são os mecanismos existentes que permitem a extracção de informação. Para se poder extrair informação, é necessário saber, como é que

esta se encontra organizada e que informação é que é pretendida. De forma a ser possível extrair informação, é necessário pedir, a estrutura de dados, que um sistema externo criou no GI. Na informação que é devolvida é possível obter informações sobre, quantas tabelas foram criadas, quais são os nomes das tabelas, os campos das tabelas, e o tipo de dados dos campos (números, texto). Em suma, é devolvida a estrutura de dados, que foi criada pelo sistema externo e o conhecimento que ele contém.

O GI como plataforma integradora de diversos sistemas, tem de possibilitar a sistemas de baixo nível e com pouco poder de processamento, guardar informação e integrar com outros sistemas. Estes sistemas de baixo nível, como por exemplo o detector de quedas (Felisberto, Felgueiras et al. 2011), que estão mais vocacionados para a recolha de dados dos idosos. A sua importância neste sistema é a mesma de um sistema evoluído que consegue interpretar informação de diversos sistemas. Estes sistemas de baixo nível e com pouco poder de processamento são os sistemas de sensores. Como no projecto do Elde Care já existem módulos de sensores, foi criada uma estrutura pré-definida, para armazenar os dados destes sistemas. Os sistemas de sensores são vistos como um conjunto de nós sensoriais, onde cada nó sensorial é um elemento único do sistema de sensores, e que é ligado ao idoso. O nó sensorial é um agregador de sensores, que consegue gravar os dados, acoplar vários sensores e enviar os dados de diversos sensores, tudo em tempo real.

Para registar um nó sensorial é necessário fornecer algumas informações. Como estes sistemas não têm grande poder de processamento, as informações requeridas para efectuar o registo exigem pouco conhecimento. O registo apenas pode ser feito por um utilizador que já se encontre registado no sistema. A informação necessária para o registo do nó sensorial é uma *string* com o formato XML que contem as credenciais do utilizador que efectua o registo, o modelo, o número de serie, a marca do sensor e o idoso a que se encontra ligado. Como resposta é devolvido ao nó sensorial um nome de utilizador e uma palavra-chave, ambos inteiros de 16 bits.

O registo de um sensor ocorre quando este é acoplado ao nó sensorial, este tem de se registar para poder guardar a informação que vai recolher. Para se registar, o sensor tem de indicar, o nome da tabela onde vai registar os dados, e quais os campos que a tabela vai conter. Ao nome da tabela é concatenado o identificador único do nó sensorial unido pelo carácter “_”. Esta tabela vai armazenar os dados recolhidos pelo sensor. O sensor é identificado no sistema,

pelo nome da tabela criada, a que está acoplado. Com essa informação, é possível criar a estrutura para um sensor. Quando o sensor é criado no sistema é gerado um identificador único interno do sistema. Também é possível que vários sensores de um nó sensorial, armazenarem dados na mesma tabela. Isto pode acontecer caso estejam acoplados ao mesmo nó sensorial dois sensores que registam o mesmo tipo de dados, evitando a criação de uma nova tabela, Como por exemplo estarem acoplados dois sensores de temperatura ao mesmo nó sensorial. Para identificar quais os dados armazenados é que pertencem a determinado sensor, é adicionado à chave de identificação do sensor em cada linha adicionada na tabela. Quando um sensor está registado e armazena dados numa tabela já existente, não é criada uma nova tabela, apenas é adicionado um novo sensor no sistema.

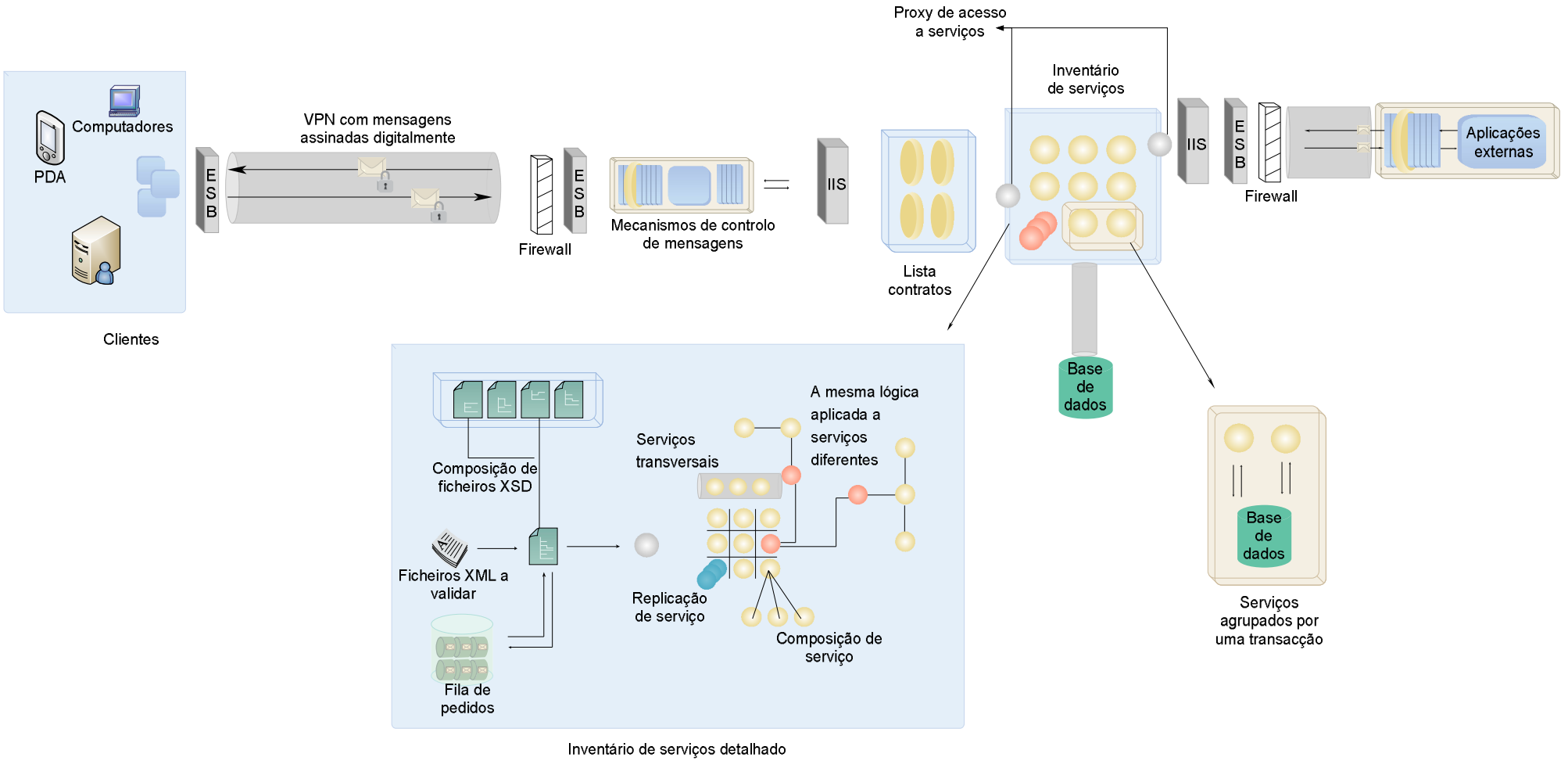


Figura 12 - Arquitectura do Gestor de Informação

O sensor quando está acoplado a um nó sensorial pode ter três estados diferentes. Os estados possíveis que um sensor pode ter são: activo, inactivo e desacoplado. O estado activo indica que o sensor se encontra a recolher dados. O estado inactivo indica que o sensor não se encontra a recolher dados e o estado desacoplado indica que o sensor já não está ligado ao nó sensorial. O desenho completo da arquitectura do GI encontra-se na Figura 12, onde é possível ver como foram conjugados todos os padrões SOA utilizados.

4.4.4.1 Diagramas de casos de uso do gestor de informação

Nesta secção são apresentados os casos de uso que representam os serviços disponibilizados pelo gestor de informação, possibilitando a interligação de sistemas. Na Figura 13 estão representadas as funcionalidades disponibilizadas pelo gestor de informação aos sistemas externos ligados. Os dados guardados podem posteriormente ser analisados por um sistema de análise de dados, que verifica possíveis anomalias no estado de saúde dos idosos.

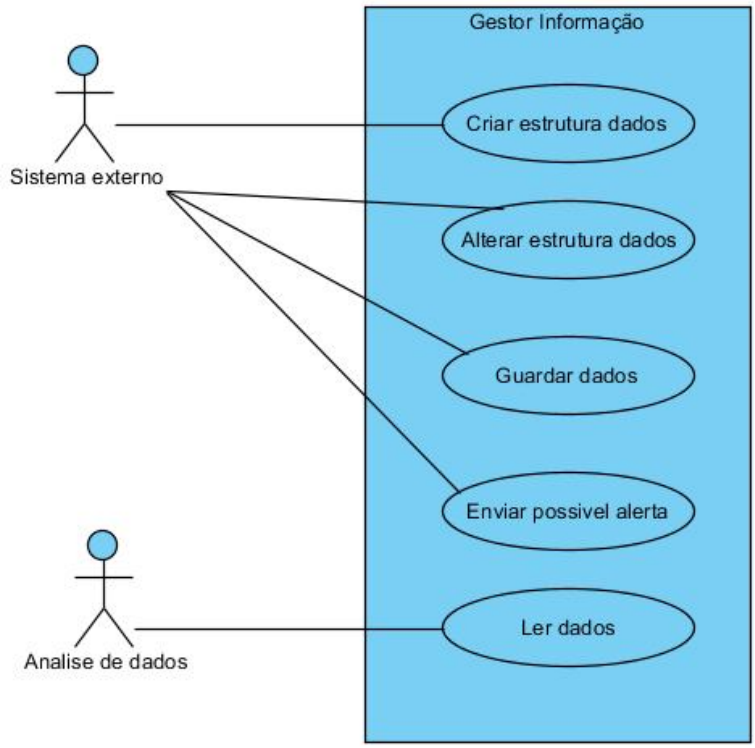


Figura 13 - Funcionalidades disponibilizadas pelo gestor de informação aos sistemas externos e à análise de dados.

Na Figura 14 estão representadas as funcionalidades disponibilizadas pelo gestor de informação aos módulos do Elde Care. O gestor de alertas tem de se autenticar perante o gestor de informação para poder armazenar dados. Após ter feito a autenticação pode guardar os dados dos eventos ou alertas com a funcionalidade de receber dados. Quando um alerta é despoletado o gestor de alertas tem consultar os dados dos contactos a notificar utilizando as funcionalidades que disponibilizam informação dos contactos.

O classificador antes de consultar ou receber dados tem de proceder à sua autenticação. Quando a sua autenticação tiver validado, pode utilizar as funcionalidades de receber e disponibilizar dados do gestor de informação, podendo assim fazer a classificação dos dados inseridos pelos sistemas de monitorização.

A interface de comunicação com a aplicação tem de se autenticar no gestor de informação, para poder posteriormente autenticar o utilizador. Após ter autenticado o utilizador, é possível receber e disponibilizar dados que estão guardados no gestor de informação que pertencem ao utilizador autenticado.

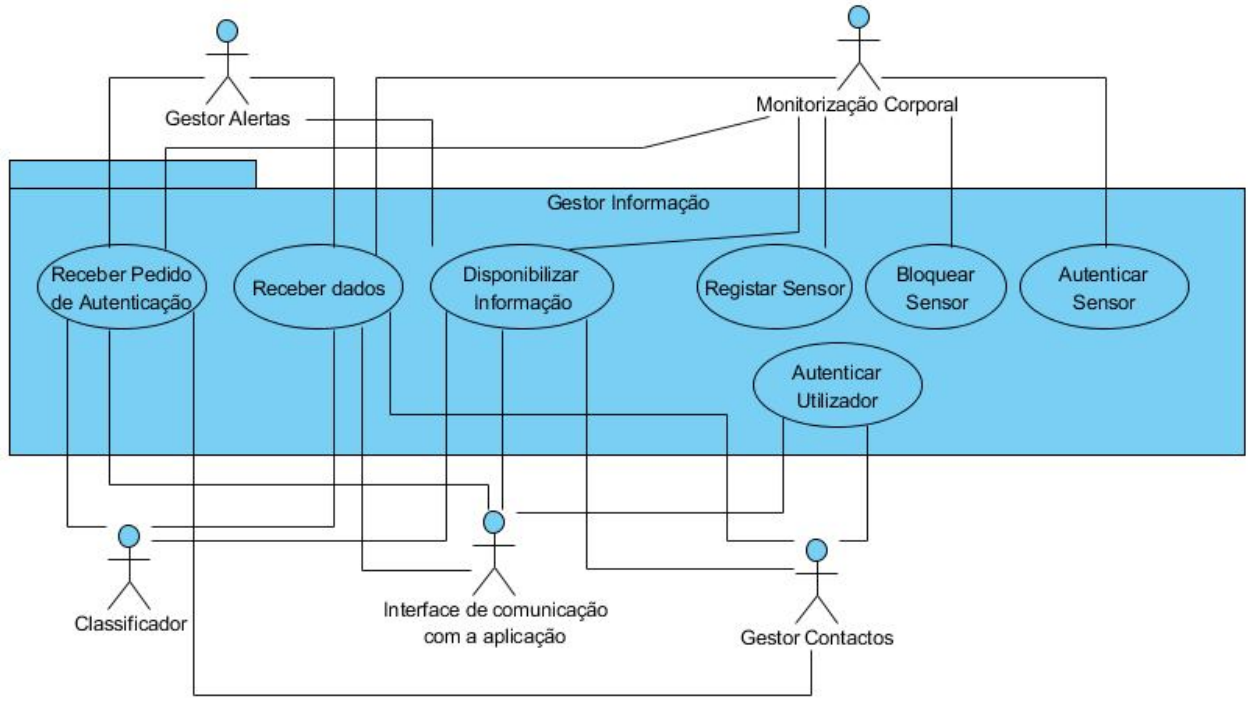


Figura 14 - Funcionalidades disponibilizadas pelo gestor de informação.

A monitorização corporal após autenticar os seus sensores pode inserir e consultar os dados que estão no gestor de informação e a que tem acesso, registar novos sensores e bloquear sensores. A informação que é colocada por este sistema pode ser verificada por um classificador, para melhorar a monitorização dos idosos.

O gestor de contactos após proceder à sua autenticação pode inserir e consultar os dados que estão no gestor de informação e a que tem acesso. O gestor de contactos pode autenticar um utilizador, podendo assim editar a lista de contactos desse mesmo utilizador.

De forma a ser possível detalhar o caso de uso da Figura 14, estes encontram-se separados nas figuras abaixo. Na Figura 15 estão as funcionalidades que o classificador tem acesso no gestor de informação. Após proceder à sua autenticação, pode receber e consultar os dados que estão no GI e que ele tem acesso. O classificador tem acesso, além da informação que pretende armazenar no GI, aos dados dos sistemas de monitorização dos idosos, para conseguir fazer uma avaliação desses mesmos dados. É assim possível melhorar a validação dos dados que os sistemas de monitorização inseriram no GI.

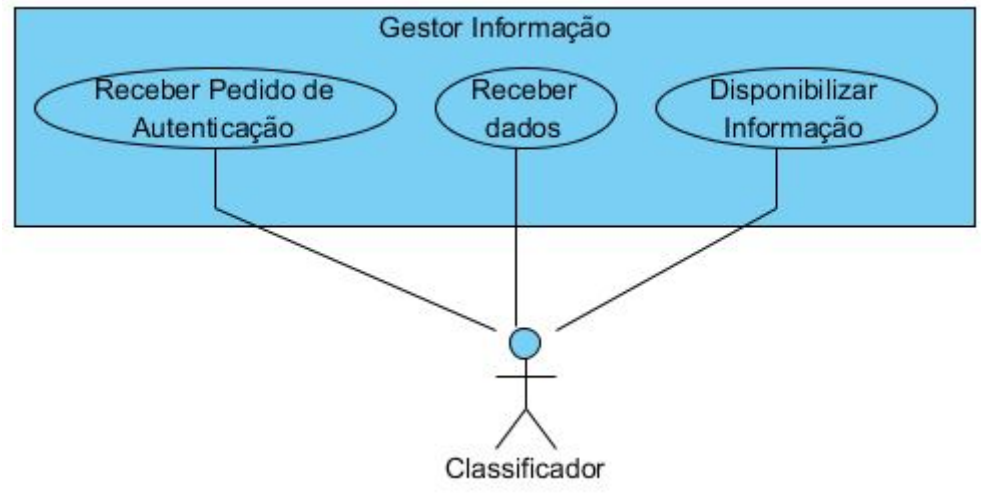


Figura 15 - Casos de uso entre o gestor de informação e um classificador.

Na Figura 16 são apresentados os serviços que o gestor de alertas tem acesso. Após proceder à sua autenticação, pode receber os dados que estão armazenados no GI e que ele tem acesso. O gestor de alertas após enviar um alerta, tem de guardar esse alerta no GI.

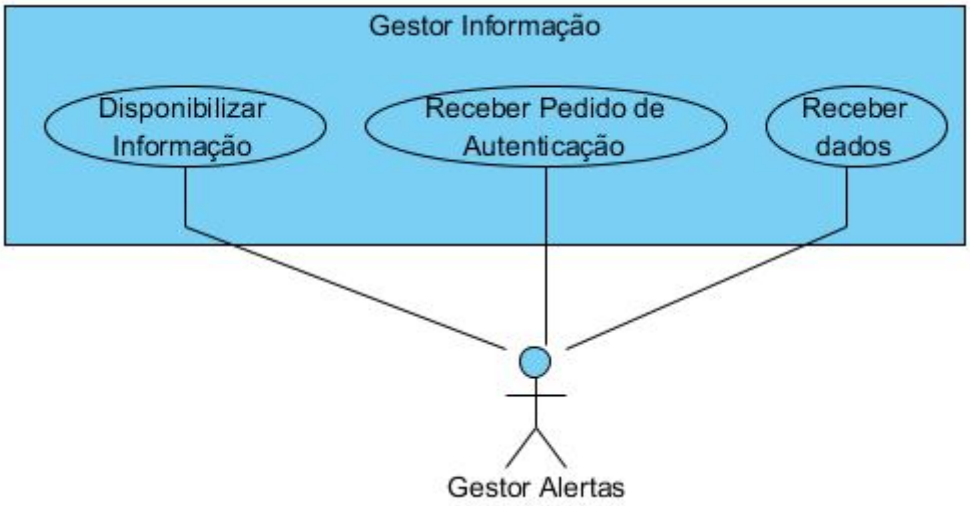


Figura 16 - Casos de uso entre o gestor de informação e o gestor de alertas.

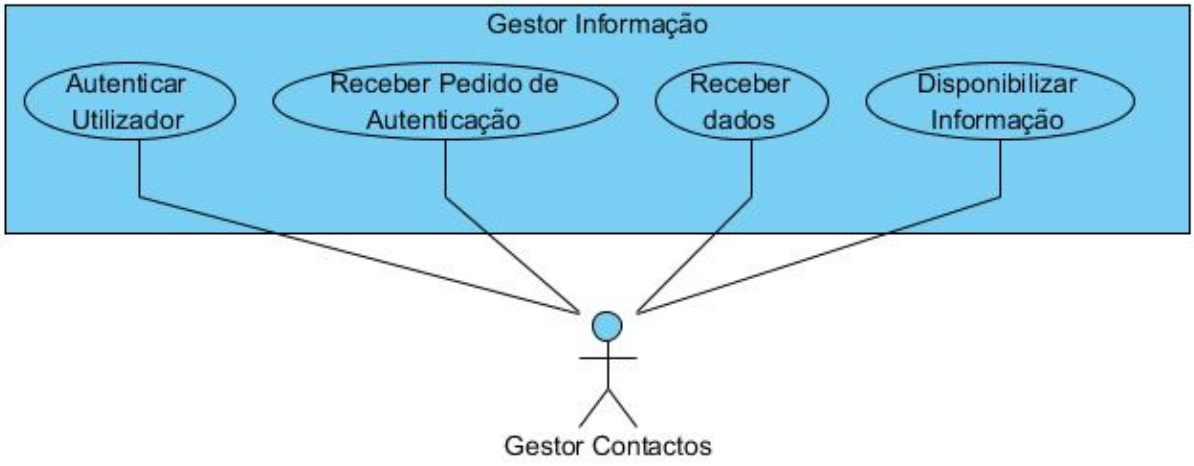


Figura 17 - Casos de uso entre o gestor de informação e o gestor de contactos.

Na Figura 17 estão apresentados as funcionalidades disponibilizadas pelo GI ao GC. O gestor de contactos, após se autenticar, pode receber e disponibilizar a informação a que tem acesso

no GI. O gestor de contactos tem acesso a todos os contactos que se encontram no GI. Quando o GC autentica um utilizador, este pode editar a lista de contactos desse mesmo utilizador. É possível consultar e editar a lista de contactos do utilizador que se autenticou.

Na Figura 18 estão representados todas as funcionalidades disponibilizadas pelo GI à monitorização corporal ou a outro modulo externo de sensores. Após proceder a autenticação, é possível registar e consultar a informação previamente inserida. Estes sistemas de monitorização têm a possibilidade de guardar dados, que não tenha proveniência dos dados de monitorização. Após autenticar os sensores, é possível guardar os dados de monitorização que os sensores registaram. Os sistemas de monitorização podem registar novos sensores, aumentando os dados de monitorização, sendo criadas novas tabelas para conter esses dados. Ainda tem a possibilidade de bloquear sensores que tenha registado anteriormente e que já não estejam a registar dados.

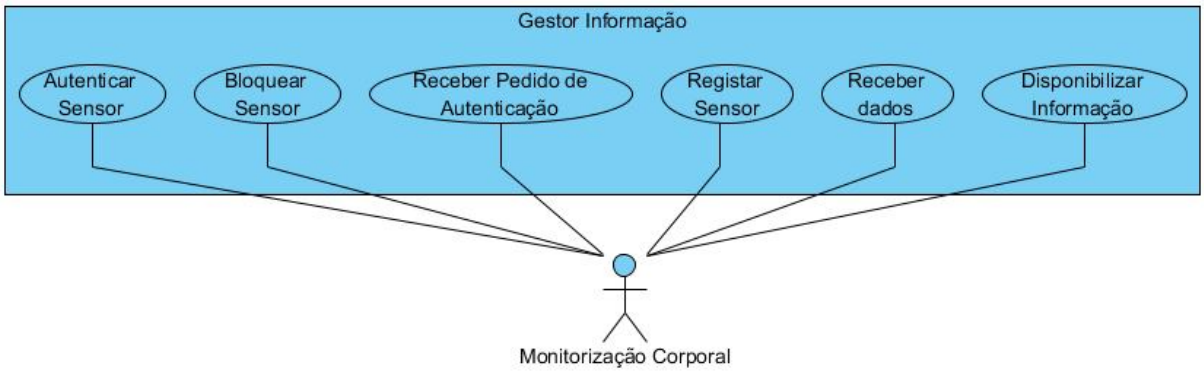


Figura 18 - Casos de uso entre o gestor de informação e a monitorização corporal ou outro módulo externo de sensores.

Na Figura 19 estão representados todas as funcionalidades disponibilizadas pelo GI à interface de comunicação com a aplicação. Após proceder à sua autenticação, é-lhe possível autenticar utilizadores. A interface permite aos utilizadores interagirem com o GI de forma a poderem guardar dados e receber os dados que forma previamente guardados. Desta forma é possível aos utilizadores interagirem com as funcionalidades do GI sem terem acesso directo.

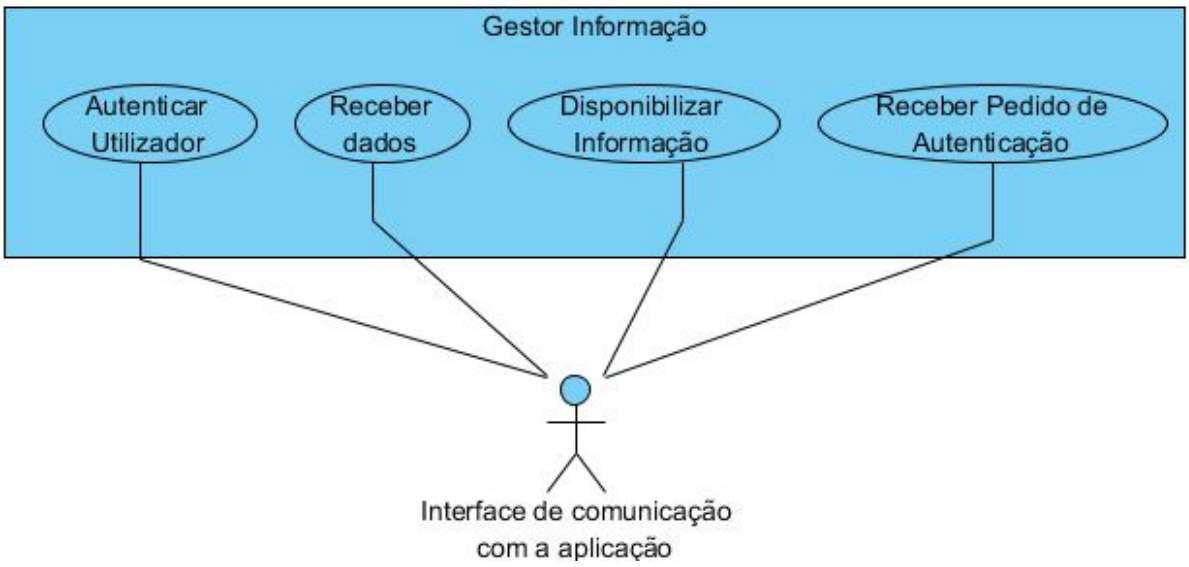


Figura 19 - Casos de uso entre o gestor de informação e a interface comunicação com a aplicação.

4.4.4.2 Diagrama de actividade do gestor de informação

Nesta secção é apresentado o diagrama de actividades do gestor de informação. O diagrama de actividades é apresentado na Figura 20. Neste diagrama estão apresentadas as possíveis actividades que é possível fazer na evocação dos serviços do gestor de informação. Como os serviços do gestor de informação não guardam o estado dos clientes (serviços *stratless*), é necessário que, a cada evocação dum serviço, se faça autenticação no gestor de informação. A autenticação vai juntamente com a mensagem XML que efectua a evocação do serviço. Após

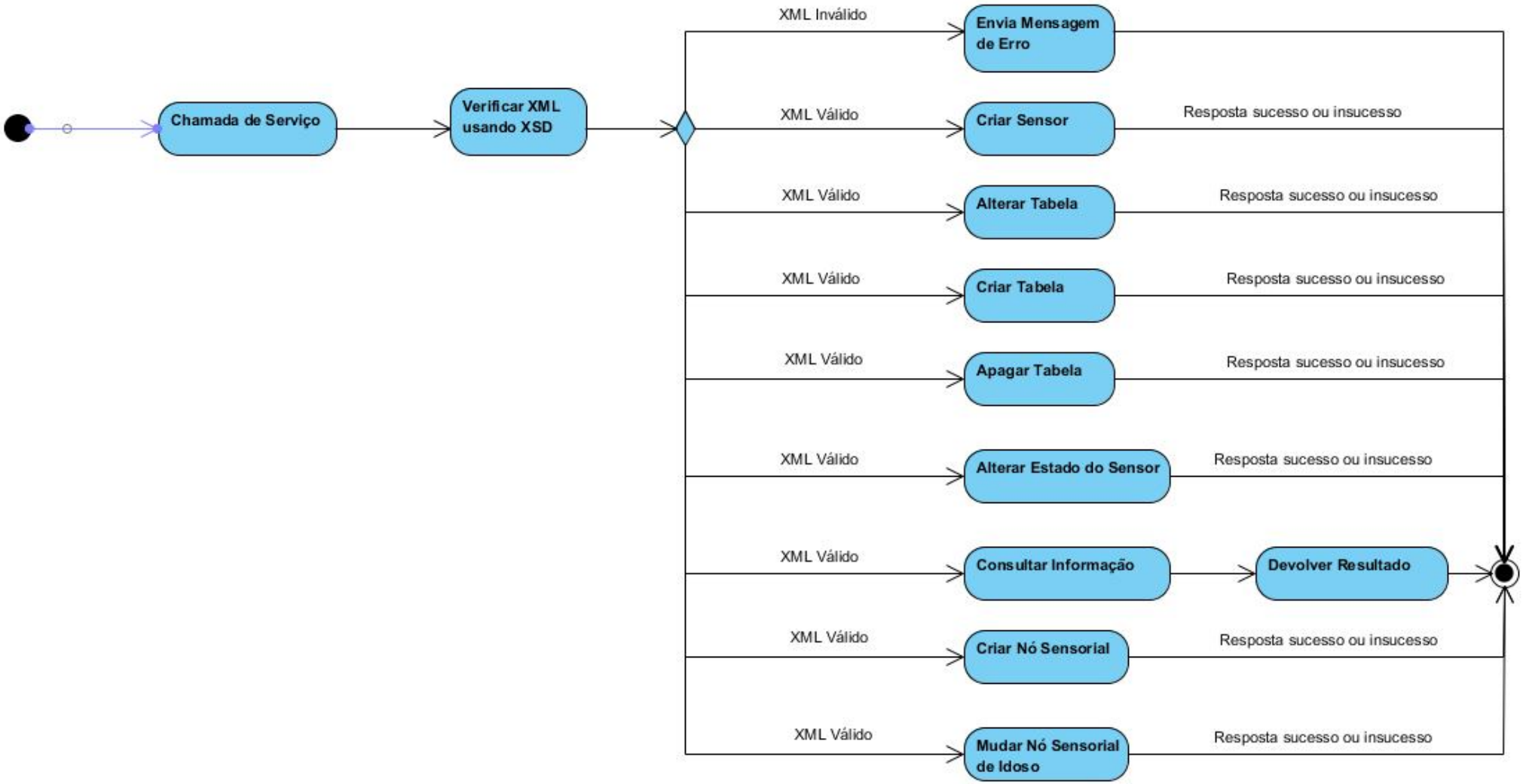


Figura 20 - Diagrama de actividades dos serviços do gestor de informação

ser evocado um serviço é verificado se a estrutura do XML é valida. Caso a estrutura não seja válida para determinado serviço, é devolvida uma mensagem de erro a informar que o XML é inválido. A actividade termina. Caso a mensagem XML seja valida, é evocado o serviço e efectuada a operação pretendida. Caso a operação termine com sucesso é devolvida uma mensagem de sucesso ao utilizador, caso contrário, é devolvida a informação ao utilizador que não foi possível efectuar o pedido. Os serviços que estão disponíveis são: criar sensor, alterar o estado do sensor, criar tabela, alterar tabela, apagar tabela, consultar informação, criar nó sensorial, mudar nó sensorial de idoso.

4.4.4.3 Diagrama relacional do gestor de informação

Nesta secção são apresentados os dois diagramas relacionais (DR) utilizados pelo gestor de informação para armazenar os dados dos sistemas externos ligados a ele. Na Figura 21 está apresentado o DR que possibilita o armazenamento dos nos sensoriais, dos seus sensores e dos metadados que eles vão gerar. Na tabela noSensorial vão estar armazenados todos os nós sensoriais que fazem parte do sistema e na tabela noSensorialLigação vai estar a indicação do nó sensorial que está ligado ao idoso. Um nó sensorial tem de estar sempre ligado a um idoso, pois é deste que os dados de monitorização vão ser registados. Este DR divide-se em dois, uma parte diz respeito aos sensores e de monitorização dos sensores, a outra parte diz respeito aos dados que um nó sensorial pretende guardar.

A primeira parte contém as tabelas sensor, sensor activo e sensor campo. Na tabela sensor encontra-se a informação dos sensores que estão ligados a um determinado nó sensorial. O estado dos sensores é guardado na tabela SensorActivo. A cada alteração do estado de um sensor é adicionado um novo registo nesta tabela. Desta forma, é possível construir um histórico do estado dos sensores. Para se consultar o estado dum sensor, tem de se consultar o último registo da tabela que diz respeito ao sensor. Os dados que o sensor vai gerar têm de ser armazenados com a utilização dos metadados. Assim cada sensor indica a estrutura da tabela onde pretende armazenar os dados, e a tabela é criada para esse sensor. Para a gestão dos metadados das tabelas dos sensores foi criada a tabela sensorCampo, que armazena os metadados das tabelas a criar para cada sensor de cada nó sensorial.

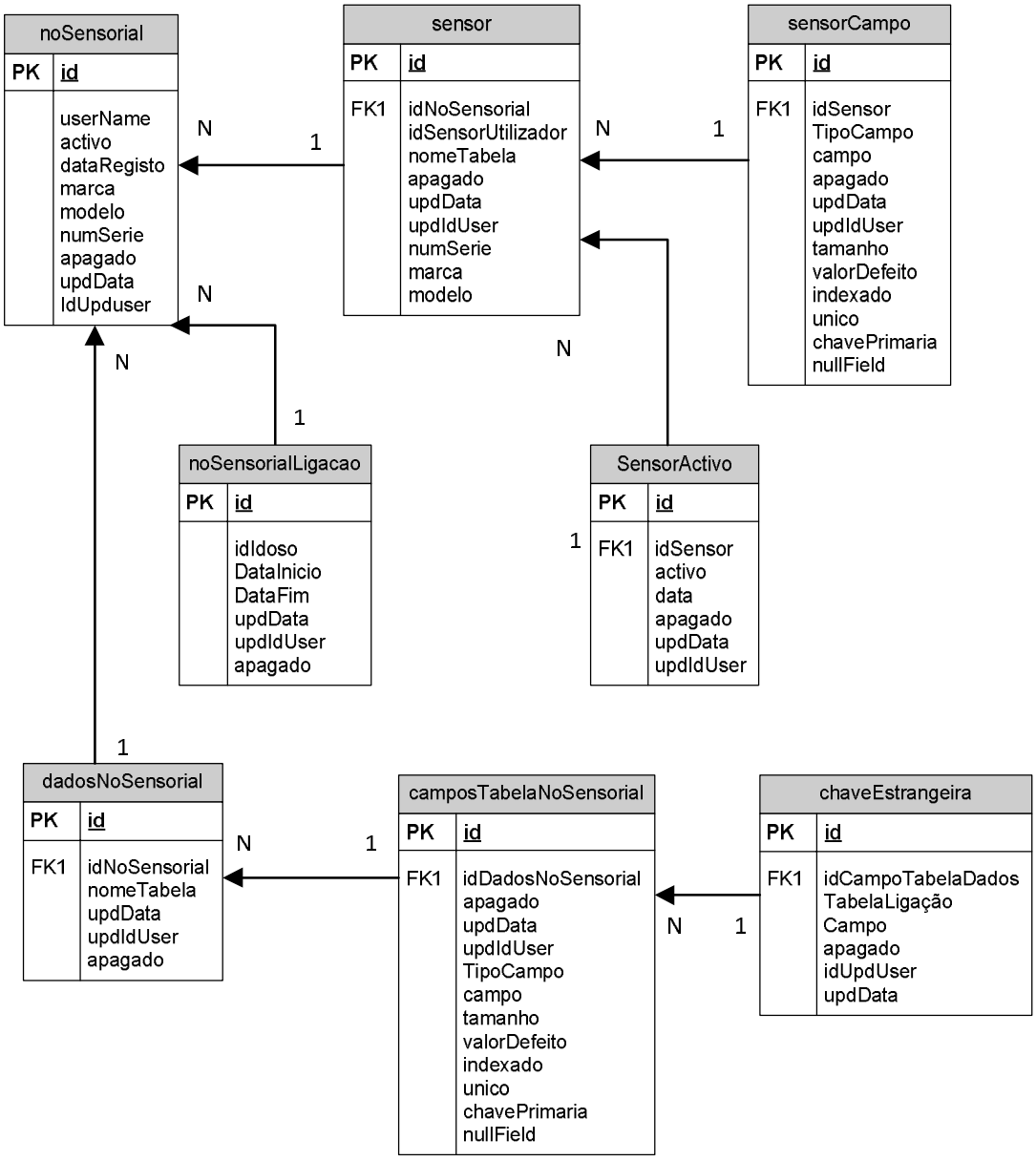


Figura 21 - Diagrama relacional para a gestão dos metadados dos nós sensoriais e dos sensores.

A segunda parte contém as tabelas dadosNoSensorial, camposTabelaNoSensorial, chaveEstrangeira. Esta estrutura permite ao nó sensorial criar uma estrutura de tabelas e ainda tema a possibilidade de as ligar entre si. Esta estrutura gere assim os metadados da estrutura que um nó sensorial necessite criar para armazenar qualquer tipo de dados. A tabela

dadosNoSensorial é armazenado o nome da tabela e a que nó sensorial pertence. A tabela camposTabelaNoSensorial armazena a informação da estrutura dos dados da tabela a criar. A tabela chaveEstrangeira permite relacionar as tabelas criadas anteriormente.

Na Figura 22 encontra-se o DR que possibilita a um sistema externo ou utilizador criar uma estrutura de tabelas para armazenar os dados que tenham necessidade. Na tabela dadosUtilizador encontra-se a informação do nome da tabela a criar e do utilizador a que pertence. A tabela camposTabelaUtilizador contém os dados da estrutura da tabela a criar. Na tabela chaveEstrangeiraUtilizador encontra-se a informação da ligação das tabelas criadas anteriormente. Esta estrutura apenas armazena metadados que posteriormente são interpretadas, para a criação das tabelas que possibilitam o armazenamento de dados.

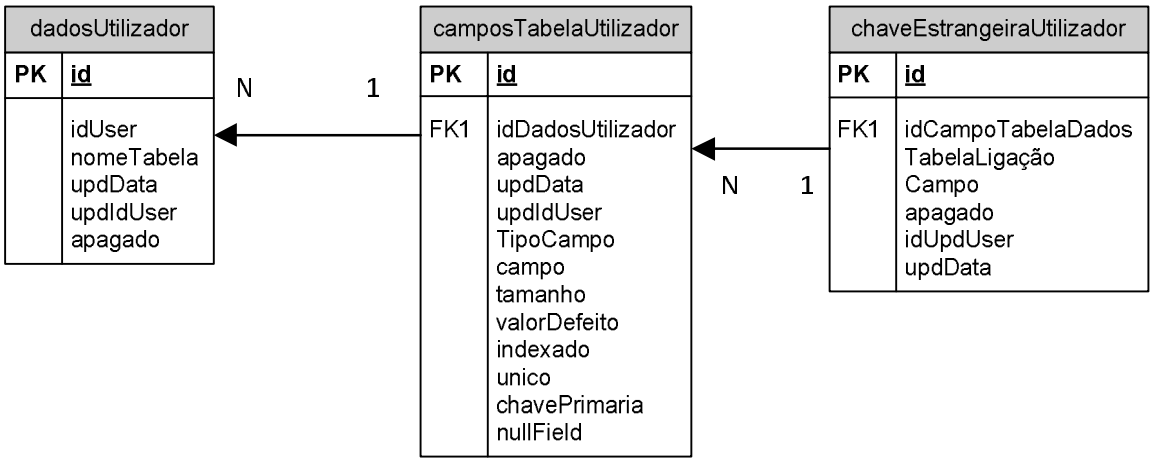


Figura 22 - Diagrama relacional para a gestão dos metadados dos utilizadores e sistemas externos.

Com a utilização destas estruturas de dados, os sistemas externos e utilizadores que estiverem ligados a este sistema tema acesso a uma base de dados remota onde lhes é possibilitado criar tabelas, apagar tabelas, editar tabelas, armazenar dados e consultar os dados armazenados de forma remota, sem terem a preocupação da manutenção da base de dados.

4.4.4.4 Protocolo de comunicação do gestor de informação

Nesta secção são apresentadas algumas das as mensagens XML que compõem o protocolo de comunicação que permite interagir com o GI. Todo o protocolo de comunicação encontra-se disponível no anexo III.

Esta mensagem permite cria a estrutura completa de um nó sensorial, incluindo as tabelas necessárias para armazenar os dados dos sensores.

```
<pedido>
	<user>bbb</user>
	<password>4565</password>
	<operacao>C</operacao>
	<noSensorial>
		<sensor>
			<nomeTabela>tabelaTeste323</nomeTabela>

			<modelo>modeloSensor</modelo>
			<numSerie>serieSensor</numSerie>
			<marca>marcaSensor</marca>
			<dados>
				<campo>aceleracao</campo>
				<tipoCampo>numero</tipoCampo>
				<definicao>
					<tamanho>50</tamanho>
					<valorDefeito>0</valorDefeito>
					<indexado>True</indexado>
					<unico>True</unico>
					<null>True</null>
					<chavePrimaria>False</chavePrimaria>
				</definicao>
			</dados>
			<dados>
				<campo>velocidade</campo>
				<tipoCampo>numero</tipoCampo>
				<definicao>
					<tamanho>50</tamanho>
					<valorDefeito>0</valorDefeito>
					<indexado>True</indexado>
					<unico>True</unico>
					<null>True</null>
					<chavePrimaria>False</chavePrimaria>

				</definicao>
			</dados>
```

```
            </sensor>
            <sensor>
                <nomeTabela>tabelaTeste346</nomeTabela>

                <modelo>modeloSensor3</modelo>
                <numSerie>serieSensor3</numSerie>
                <marca>marcaSensor3</marca>
                <dados>
                    <campo>aceleracao3</campo>
                    <tipoCampo>numero3</tipoCampo>
                    <definicao>
                        <tamanho>50</tamanho>
                        <valorDefeito>0</valorDefeito>
                        <indexado>True</indexado>
                        <unico>True</unico>
                        <null>True</null>
                        <chavePrimaria>False</chavePrimaria>

                    </definicao>
                </dados>
                <dados>
                    <campo>velocidade3</campo>
                    <tipoCampo>numero3</tipoCampo>
                    <definicao>
                        <tamanho>50</tamanho>
                        <valorDefeito>0</valorDefeito>
                        <indexado>True</indexado>
                        <unico>True</unico>
                        <null>True</null>
                        <chavePrimaria>False</chavePrimaria>
                    </definicao>
                </dados>
            </sensor>
            <modelo>modelo</modelo>
            <numSerie>serieNoSensorial</numSerie>
            <marca>marca</marca>
            <userIdoso>JoséCarpinteiro09</userIdoso>
        </noSensorial>
</pedido>
```

Este XML permite a criação de um nó sensorial e dos sensores a ele ligado. São ainda criadas as tabelas que permitem armazenar os dados dos sensores que foram adicionados.

4.4.5 Diagrama de classes

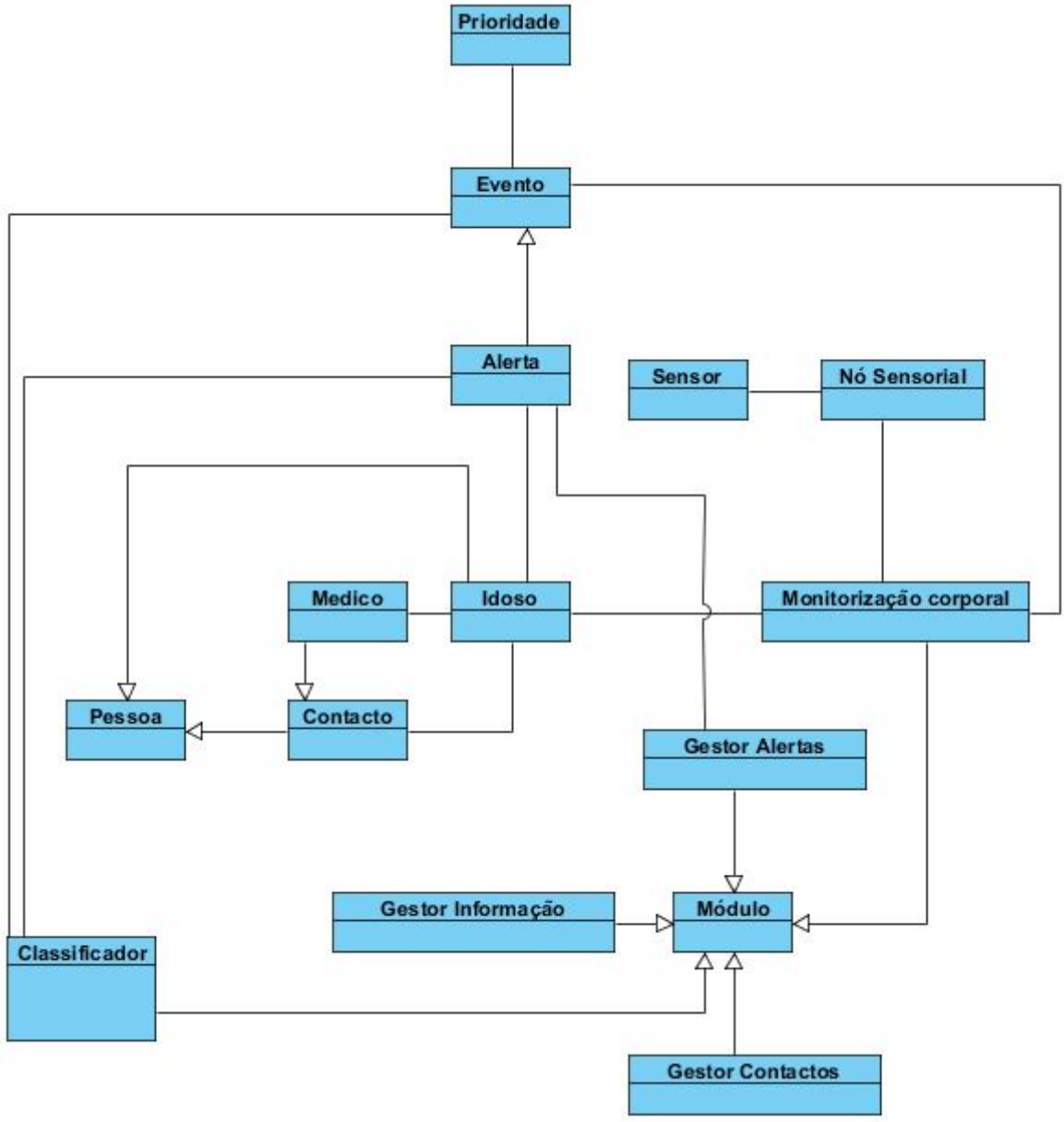


Figura 23 - Diagrama de classes antes da implementação da ontologia.

Nesta secção é apresentado o diagrama de classes que foi construído para iniciar o desenvolvimento do centro de controlo. Na Figura 23 encontra-se o diagrama de classes onde é possível ver as classes necessárias para o inicio da construção do CC. A classe Módulo representa os sistemas ligados ao Elde Care. A classe Pessoa representa todos os indivíduos

que estão registados no gestor de informação. Existem três tipos de pessoas: os idosos, os médicos e os contactos. Os idosos são os que vão usufruir das funcionalidades do sistema. Os contactos são as pessoas que compõem a lista de contactos do idoso. Os médicos são contactos que pertencem à lista de contactos e que são contactados aquando de um alerta de prioridade elevada, ou então para esclarecimento de dúvidas do idoso.

A classe monitorização corporal, contém todos os sistemas de monitorização do idoso, tais como os nós sensoriais e os sensores, representados pelas respectivas classes. Os eventos enviados pelos sistemas de monitorização estão representados na classe evento. Cada evento tem uma prioridade atribuída. Após o evento ser classificado origina um alerta, que pode ter a sua prioridade alterada. Este alerta é tratado pelo gestor de alertas, que verifica a lista de contactos do idoso para tratar do respectivo alerta.

4.4.6 Ontologia

A utilização duma ontologia, permite expressar conhecimento de forma clara e sem ambiguidades (Guarino 1998). Assim para representar o conhecimento do domínio do Elde Care vai ser utilizada uma ontologia. Sendo o gestor de informação responsável pela informação do domínio do Elde Care, a ontologia tem de expressar não só o conhecimento que se encontra no gestor de informação, como de todo o domínio do Elde Care. Como a arquitectura do GI é escalável, outros sistemas podem adicionar conhecimento ao âmbito do projecto Elde Care, tendo a ontologia de conseguir representar o novo conhecimento adicionado.

A utilização das ontologias consegue assim representar todo o conhecimento contido no Elde Care. Esta representação é feita com recurso a indivíduos, classes, atributos e relacionamentos. Os indivíduos representam os objectos mais simples da ontologia. Estes indivíduos representam objectos concretos e existentes, como as pessoas, as patologias, os sensores, os alertas, os sistemas ligados ao Elde Care entre outros. Estes indivíduos têm sempre de pertencer a uma classe, permitindo assim a sua distinção. As classes são agrupadores de conceitos idênticos e podem definir objectos complexos. Para as definir podem ser usados indivíduos, classes ou uma combinação de ambos. Os objectos têm de ter

atributos que os identifiquem e que os caracterizem. Com a conjugação de classes e de indivíduos, é possível representar os objectos que representam a realidade na ontologia. Uma ontologia não se limita a descrever objectos e a adicionar-lhe atributos. Ainda descreve como os objectos se podem relacionar entre si e como é que essa relação é efectuada. A possibilidade de criar relações nas ontologias é distinta de outras noções similares às ontologias como por exemplo os *thesaurus* ou das taxonomias, pois permite criar relações dependentes de regras.

Aquando da construção da ontologia, é construído um diagrama de classes. Neste diagrama estão representadas todas as classes que compõem o Elde Care e as suas relações hierárquicas. Neste diagrama de classes estão representados todos os objectos que estão presentes no Elde Care, onde estão representadas as relações e a estrutura das classes. As classes que compõem a ontologia encontram-se na Figura 24, onde todas as classes que têm um nível hierárquico inferior visível, são as que compõem o gestor de informação. O desenvolvimento de uma ontologia fornece não só um diagrama de classes, como ainda consegue definir como é que as classes vão interagir entre si. O diagrama de classes completo da ontologia encontra-se no final do anexo IV.

A interacção que é possível representar não é apenas entre as classes, mas sim entre os objectos que estão definidos na ontologia. As interacções representadas não se limitam à hierarquia, onde são definidos os membros de cada objecto. Estas interacções são definidas como interacções de inclusão. Com este tipo de interacções é possível definir a que classe pertence um determinado objecto, definir a relação hierárquica entre objectos e combinar diversos objectos para construir objectos compostos.

As ontologias não seriam necessárias para possibilitar todas estas representações, pois estas podem ser representadas nos diagramas de classes. As ontologias conseguem representar ainda, relações mais específicas, que apenas fazem sentido em domínios específicos. Estas relações conseguem refinar ainda mais o nível semântico da representação de determinado domínio. Com a representação destas relações, que envolvem regras específicas do domínio, é possível responder a questões particulares do domínio. Na ontologia do Elde Care é possível representar quais são os idosos que têm uma determinada patologia. Esta representação é feita com o relacionamento dos indivíduos que estão agrupados na classe patologia, com os indivíduos agrupados na classe idoso. Apesar de estas classes não estarem hierarquicamente

ligadas, e aparentemente não terem qualquer tipo de relacionamento, a ontologia consegue responder a questão: quais são os idosos que têm uma determinada patologia.

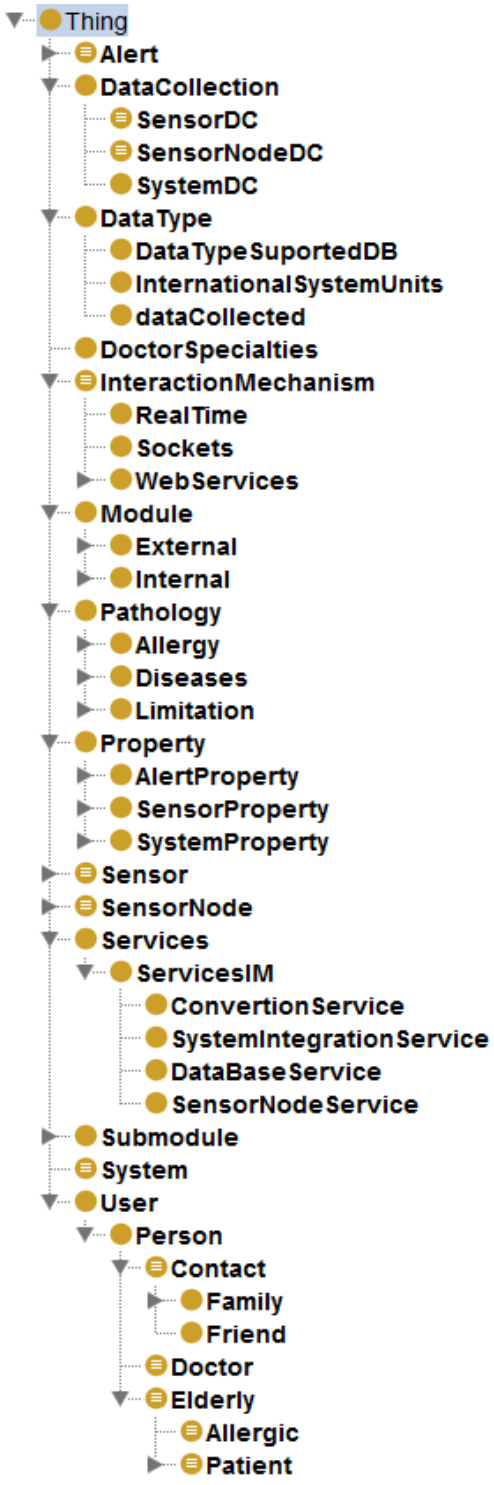


Figura 24 - Parte do diagrama de classes da ontologia

A Figura 25 apresenta como é possível questionar a ontologia de forma a saber quais são os indivíduos que têm tem uma doença específica.

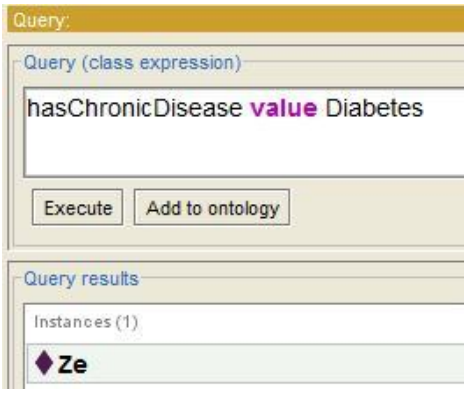


Figura 25 - Resposta da ontologia à questão dos idosos que têm diabetes

Só é possível responder a questões específicas do domínio com a adição de regras. As regras adicionadas têm de representar a realidade do domínio. Com estas regras é possível definir premissas conhecidas ou assumidas como sendo verdadeiras. Partindo de premissas que são verdadeiras, é possível utilizar sistemas inteligentes que consigam tirar conclusões lógicas e verdadeiras. Estes sistemas inteligentes são os motores de inferência. Estes conseguem derivar conclusões lógicas através de premissas. As regras que definem a ontologia são premissas, pois são elas que definem como é que os objectos existentes na ontologia vão interagir. As regras podem também indicar quem pode interagir e até que nível ou a partir de que nível. Os motores de inferência são úteis para possibilitar pesquisas aos objectos que estão inseridos na ontologia, com recurso as regras adicionadas e aos parâmetros de cada objecto. As pesquisas que são efectuadas pelos motores de inferência não necessitam de uma análise externa. Os resultados devolvidos estão sempre relacionados com a pesquisa efectuada, ao contrário dos motores de busca que encontram termos relacionados. Na Figura 26 é ilustrada a resposta do motor de inferência à questão de quais os indivíduos presentes na ontologia têm doenças crónicas. Para que o motor de inferência conseguir responder a esta questão, tiveram de ser adicionadas regras.

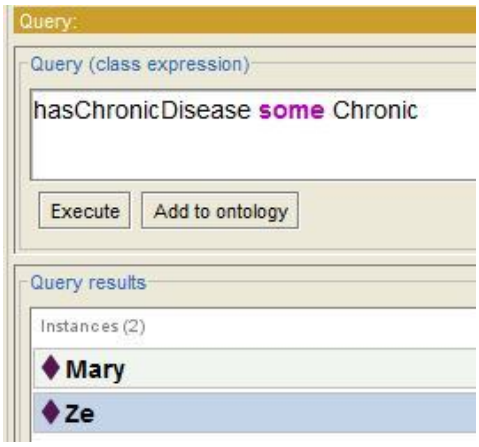


Figura 26 - Inferência acerca de pessoas com doenças crónicas

As regras adicionadas são consideradas verdadeiras pelo motor de inferência. As regras utilizadas para que o motor de inferência derivasse esta conclusão são: pessoas que têm doenças crónicas, a Mary tem Alzheimer, o Ze tem Diabetes, Alzheimer e Diabetes são doenças crónicas. Seguindo estas premissas, o motor de inferência concluiu que a Mary e o Ze são pessoas com pelo menos uma doença crónica. Segundo este modelo de inferências é possível realizar uma lista de questões simples que se pretende ver resolvidas pelas inferências. Desta forma, a informação quando é consultada, encontra-se organizada pois o motor de inferência relacionou a informação útil com a questão efectuada.

A ontologia desenvolvida, é escalável, tal como a arquitectura do centro de controlo. Isto permite a adição de mais conhecimento e a actualização do conhecimento já existente. As novas aplicações externas ao adicionar-se ao centro de controlo estão a aumentar o conhecimento que este contém. A ontologia tem de conseguir representar este novo conhecimento. Para que este novo conhecimento, seja representado na estrutura da ontologia, a sua estrutura tem de estar preparada para poder adicionar conhecimento novo. Para aplicações que já têm o seu conhecimento na ontologia, é possibilitado a edição do conhecimento já alojado na ontologia.

Para que a ontologia possa crescer, é essencial que todo este conhecimento não seja misturado. É necessário garantir que cada sistema apenas possa editar o que criou. Isto é conseguido, pois existe na ontologia uma classe que representa todos os sistemas externos adicionados. Quando um sistema externo é criado, é criada uma nova classe que vai

representar esse sistema. Esta classe apenas vai pertencer ao sistema externo que a criou, sendo ele o único que a pode editar. A ontologia já contém algumas regras de ligação de classes, no entanto cada sistema pode criar as suas regras. Nas ontologias as regras conseguem detalhar melhor o conhecimento existente. As regras criadas por cada sistema, apenas podem ser editadas por eles no entanto, é possível que sejam utilizadas por todos os sistemas. Também existem classes na ontologia que são possíveis de ser utilizadas pelos sistemas existentes na ontologia. Essas classes não pertencem a nenhum sistema externo, mas sim ao Centro Controlo.

Para possibilitar a interligação de sistemas, existem classes que definem como é que os sistemas vão trocar dados. Para trocar dados com o Centro de controlo, é sempre utilizada a norma sistema internacional de unidades (SI). Cada sistema que esteja ligada ao centro de controlo, tem de fornecer uma fórmula de conversão dos seus dados para a norma SI. Assim é possibilitado aos sistemas externos armazenar os dados na norma SI directamente, ou é possível guardar os dados, utilizando uma norma própria, sendo então essencial fornecer uma fórmula de conversão, para a norma SI. No anexo IV encontra-se o manual de utilização da ontologia. Neste manual está explicado a forma correcta de interagir com a ontologia e qual a finalidade de cada classe existente. Também se encontra explicado como é que se adicionam regras que podem ser utilizadas para definir de forma mais clara um sistema.

A ontologia desenvolvida não tem como finalidade armazenar os dados. Na ontologia é pretendido que se encontre os modelos de alto nível, os modelos físicos, os serviços disponibilizados e quais são os procedimentos necessários para aceder aos dados e serviços.

As ontologias têm uma utilidade na representação semântica do conhecimento. O conteúdo semântico consegue descrever de forma clara o significado de um objecto, sem ter em atenção ao seu nome. É assim possível utilizar um agente inteligente, para pesquisar num determinado conjunto de dados, por uma informação específica. O resultado devolvido pelo agente inteligente contém toda a informação relacionada com a pesquisa, pois o conteúdo semântico possibilita as máquinas perceber a semântica dos termos. Desta forma, é possível utilizar agentes inteligentes para pesquisar e relacionar a informação dos diversos sistemas.

A utilização de conteúdos semânticos, possibilita delegar para os agentes inteligentes a tarefa de agregar informação proveniente de diversos sistemas. O conteúdo semântico dos termos

possibilita aos agentes inteligentes efectuar pesquisas no conhecimento da Ontologia. É então possível delegar as tarefas de pesquisa, cruzamento de informação e tomadas de decisão aos agentes inteligentes, sem a necessidade da intervenção humana.

4.4.7 Arquitectura do gestor alerta

O gestor alerta (GA) tem como função receber eventos, enviados pelos vários sistemas externos que efectuam a monitorização aos idosos que participam no projecto Elde Care. Os eventos recebidos, são guardados na base de dados central, de forma a poderem ser analisados posteriormente.

Os eventos são então classificados por um classificador externo e podem originar um alerta. Os eventos que não geram um alerta são considerados os falsos positivos. Os alertas criados podem ser classificados em cinco níveis: baixo, médio, alto, urgente e crítico.

Após a recepção de um alerta pelo gestor de alertas, este verifica na base de dados central quem deve de notificar e por que métodos. Os métodos disponíveis para efectuar contactos são: e-mail, chat, mensagens de SMS e chamada telefónica. Para que seja possível efectuar estes contactos, é necessária a criação de uma lista de notificações por idoso, onde se encontram definidos os contactos das pessoas a notificar, consoante o nível de prioridade do alerta. É assim possível percorrer uma lista de contactos diferente por cada nível de prioridade.

O gestor de alertas está continuamente à escuta de novos eventos ou alertas para serem tratados.

4.4.7.1 Diagramas de casos de uso do gestor de alertas

Nesta secção são apresentados os casos de uso que representam os serviços disponibilizados pelo gestor de alertas, possibilitando a recepção de eventos, envio desses eventos para o classificados e envio de alertas. Na Figura 27 está representado o caso de uso que possibilita a recepção dum evento pelo gestor de alertas e envio desse evento para um classificador, de forma a possibilitar a classificação desse evento num alerta de uma determinada prioridade. O classificador é um sistema externo ao módulo CC.

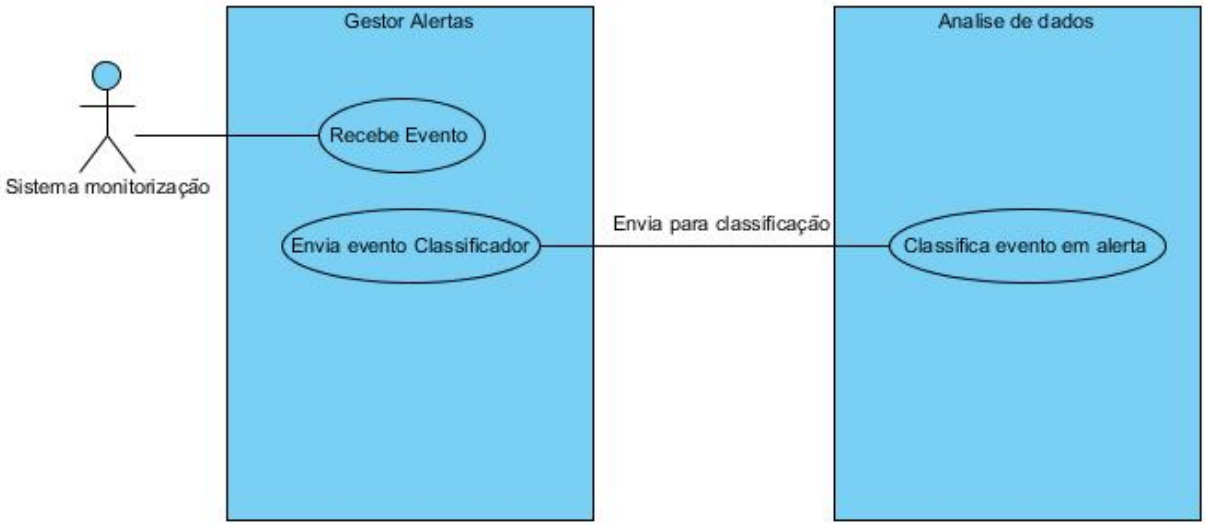


Figura 27 - Recepção de um evento e envio para o classificador

Na Figura 28 está representado o caso de uso que possibilita o envio de um alerta. Para um alerta ser enviado é necessario consultar a lista de contactos do idoso, para obter a informação de quem contactar, perante um alerta de determinada situação. Assim aquando da recepção de um alerta, o GA tem de consultar a lista de contactos do idoso que se encontra armazenada no gestor de contactos para poder notificar esses mesmos contactos, da ocorrencia de uma situação de alerta de determinada prioridade.

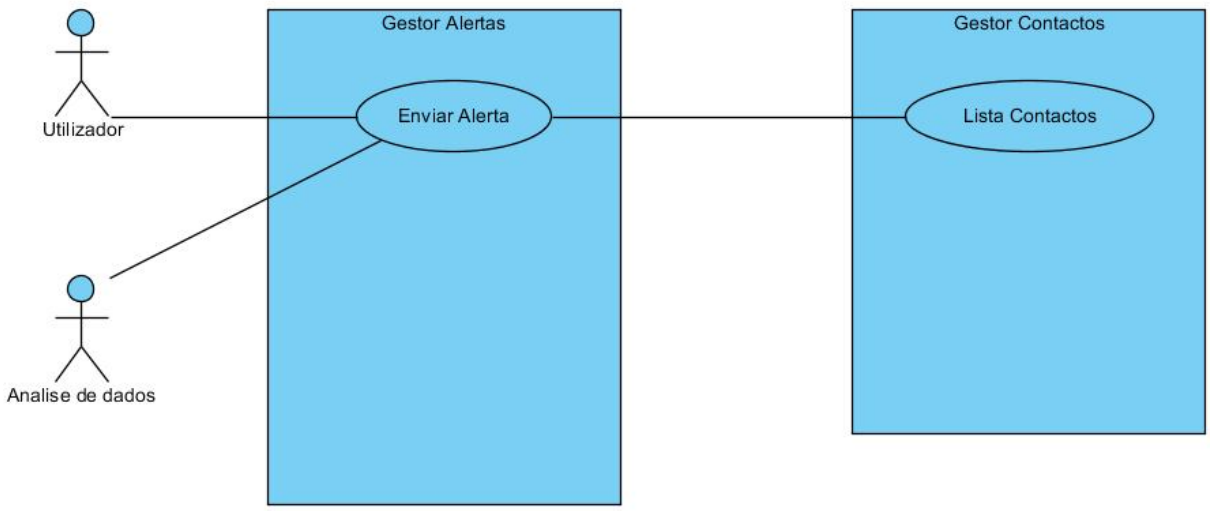


Figura 28 - Envio de um alerta.

4.4.7.2 Diagramas de actividade do gestor de alertas

Nesta secção são apresentados os diagramas de actividades do gestor de alertas. Existem dois diagramas de actividades: o diagrama de tratamento dos eventos e o diagrama dos serviços. Na Figura 29 é apresentado o diagrama de actividades do tratamento de um evento. São apresentados todas as etapas da classificação dum evento até terminar o alerta. Aquando da recepção do evento, este é guardado na base de dados e enviado para o classificador. Após ser classificado já não se trata de um evento mas sim de um alerta pois já tem uma prioridade definida. A primeira etapa do alerta é tentar contactar o idoso, para saber mais detalhes acerca do seu estado de saúde. Consoante a prioridade do alerta é concedido um tempo de espera de resposta por parte do idoso. Quanto maior a prioridade do alerta menor o tempo de espera. A resposta do idoso é adicionada à notificação a efectuar. As notificações são efectuadas seguindo a lista de notificações do idoso para determinada prioridade de alerta. Existe um limite de tentativas efectuadas para notificar as pessoas da lista de contactos, repetindo este processo, tantas vezes quantas estiverem configuradas.

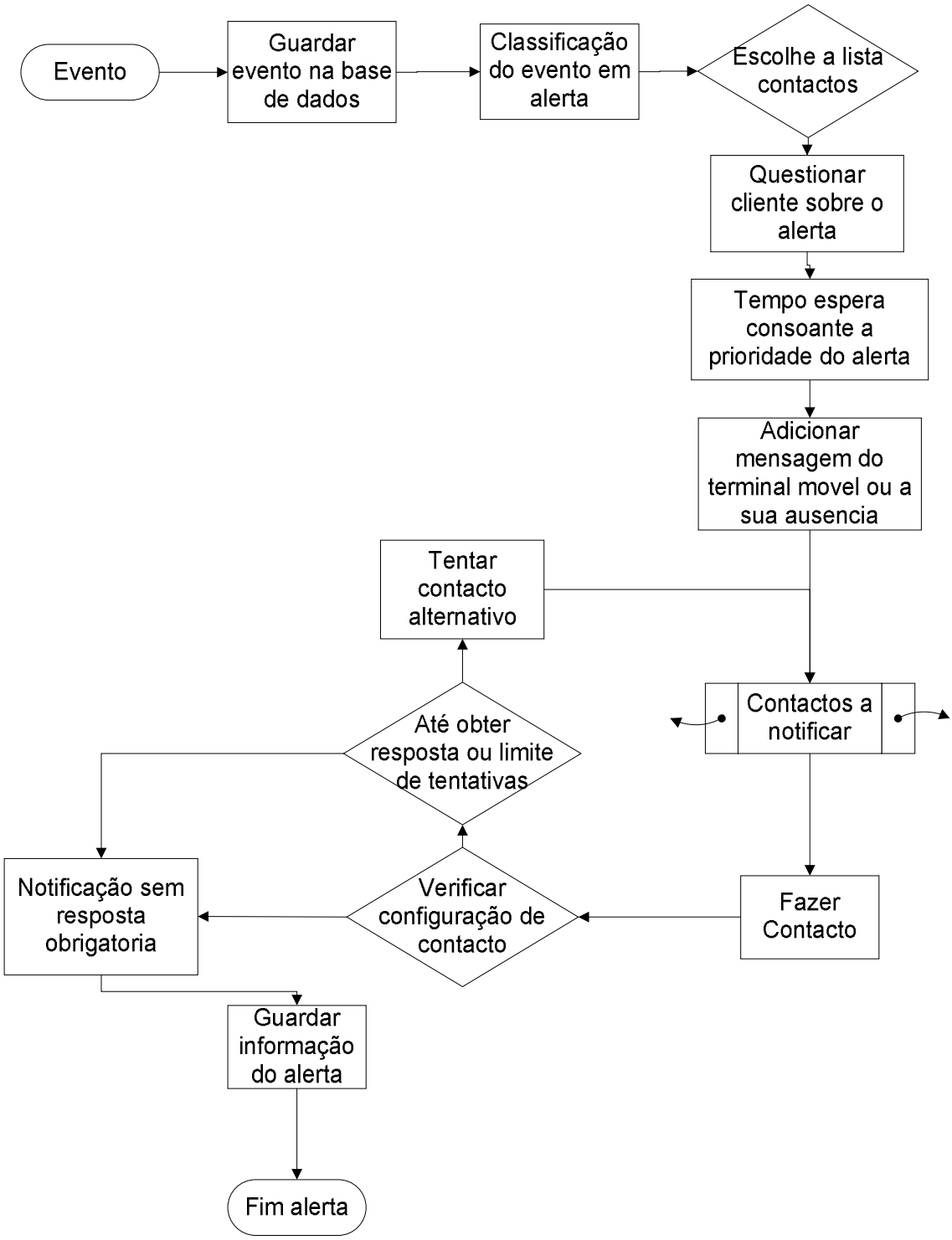


Figura 29 - Tratamento de um evento pelo gestor de alertas.

Após se ter efectuado as tentativas de notificação da lista de contactos, é enviado uma notificação que não garanta resposta nem tenha garantia de resposta como por exemplo o

envio de um e-mail. Esta notificação é enviada para todas as pessoas que estão na lista de notificações. Então este alerta é guardado na base de dados com toda a informado de notificação e é dado por terminado o alerta.

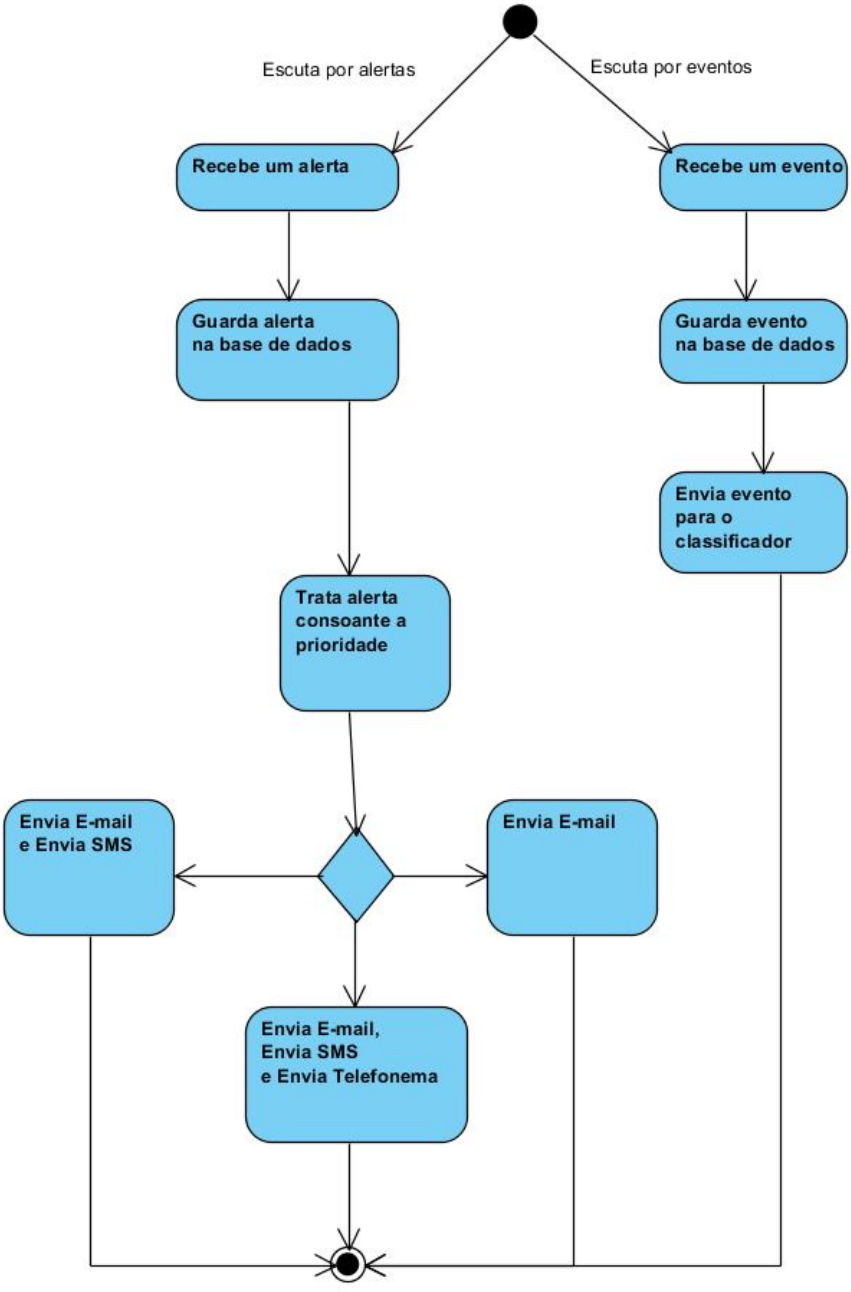


Figura 30 - Diagrama de actividades dos serviços do gestor de alertas

Na Figura 30 é apresentado o diagrama de actividades dos serviços do gestor de alertas. O GA fornece dois serviços: a recepção de eventos e a recepção de alertas. Aquando da recepção dum evento este é guardado na base de dados e enviado para o classificador. Aquando da recepção dum alerta este é guardado na base de dados e é tratado o alerta consoante a prioridade que lhe foi atribuído. As notificações a efectuar podem ser por e-mal, email e SMS ou e-mail, SMS e uma chamada telefónica.

4.4.7.3 Diagrama relacional do gestor de alertas

Nesta secção é apresentado o diagrama relacional (DR) utilizado pelo gestor de alertas para armazenar os eventos e os alertas. Na Figura 31 está apresentado o DR que possibilita o armazenamento dos eventos e alertas dos idosos. Na tabela Idoso vão estar armazenados a lista dos idosos que fazem parte do sistema. A tabela contacto vai conter todos os contactos a notificar para cada tipo de alerta. Na tabela formas de contacto estão armazenadas as formas de contacto disponíveis. Um contacto da lista de contactos pode ser notificado de diversas formas e uma forma de contacto pode ser aplicada a vários contactos é necessário a criação da tabela intermédia contacto/forma de contacto para conseguir registar as possíveis formas de contacto. Na tabela evento, vão estar armazenados todos os eventos ocorridos para cada idoso. Cada evento gerado diz respeito a um tipo de situação (queda, Idoso perdido, ente outros). O tipo de situação de cada alerta encontra-se armazenado na tabela TipoEvento. Após ser classificado o evento, este origina um alerta, com uma determinada prioridade. Tanto o evento como o alerta têm uma prioridade. A prioridade do evento é indicativa, sendo esta atribuída pelo sistema que lança o evento. Aquando da classificação, a prioridade atribuída ao evento é analisada, podendo esta ser ajustada, para representar uma melhor situação do alerta.

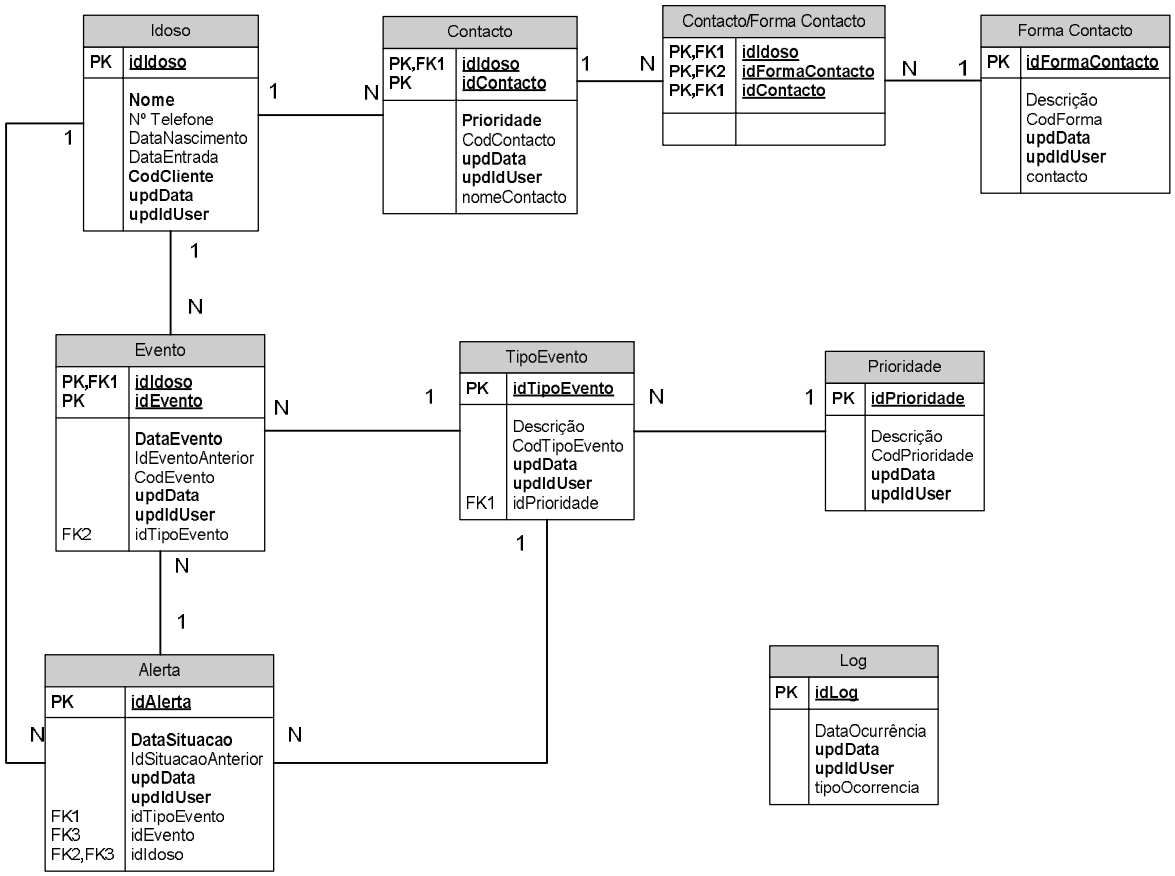


Figura 31 - Diagrama relacional para a gestão de alertas e eventos.

4.4.7.4 Protocolo de comunicação do gestor de alertas

Nesta secção são apresentadas algumas das as mensagens XML que compõem o protocolo de comunicação que permite interagir com o GA. Todo o protocolo de comunicação encontra-se disponível no anexo III.

Envio de um alerta por um módulo externo

```
<Alerta>
        <user> </user>
        <password> </password>
        <operacao> </operacao>
        <dadosAlerta>
                <evento>  </evento>
                <prioridade>4</prioridade>
```

```
            <usernameIdoso> </usernameIdoso>
        </dadosAlerta>
</Alerta>
```

Esta mensagem permite o envio de um alerta por parte de módulo externo.

Envio do Evento para o classificador

```
<login>
        <loginUtilizador><loginUtilizador>
        <Login>aaa<\Login >
        <password>X%tg0rg<\password>
<\login>
<evento>
        <idEvento><\idEvento>
        <tipo><\tipo>
        <prioridadeEmissor><\ prioridadeEmissor>
<\evento>
```

4.4.8 Definição dos alertas e lista de contacto

Os alertas por definição encontram-se configurados para um nível de prioridade. Os alertas podem ser classificados em cinco níveis, o nível baixo, o médio o alto o urgente e crítico. Os quatro primeiros níveis podem ser atribuídos aos alertas. O nível crítico suspende todos os alertas, passando o sistema a responder apenas a este tipo de alertas. Aquando da ocorrência de vários alertas do mesmo tipo espaçados por pouco tempo, o nível de prioridade desse alerta em particular aumenta um nível. Na Tabela 4 estão listados alguns alertas, e qual a prioridade que lhes vai ser atribuída. As prioridades podem ser configuradas para outro nível que não o indicado nesta tabela. Para se perceber melhor como são atribuídas as prioridades aos alertas, está descrito na Tabela 5 o que é considerado um alerta de cada nivel.Tabela 4 - Prioridade atribuidas aos alertas

Prioridade	Baixa	Média	Alta	Urgente	Definido pelo utilizador (baixa,média,alta)
Situação					
Queda Idoso					
Imobilidade prolongada					
Idoso perdido					
Idoso em zona perigosa					
Arritmia					
Quebra de tensão					
Hipertensão					
Temperatura elevada					
Temperatura reduzida					
Pedido de ajuda pelo idoso					
Perca de sinal					
Bateria fraca					
Assistolia					
Toma de medicação					
Lembrete Toma de medicação					
Eventos pré programados					
Lembretes dos eventos pré programados					
Alerta indefinido					

Tabela 4 - Prioridade atribuidas aos alertas

Prioridade:	**Classificação das mensagens quando estas ocorrem em simultâneo.**
Falso Positivo	
Baixa	Situações fora do normal, que directamente não colocam em risco a saúde do idoso
Media	Situações fora do normal, que não exigem cuidados directos de profissionais
Alta	Situações fora do normal, que não necessitem de cuidado imediato de profissionais. Deve ser comunicado a entidades profissionais para prevenção
Urgente	Situação que necessita de cuidados imediatos por parte de profissionais

Tabela 5 - Prioridades existentes para classificar alertas

Lista de Contactos

Tipo Situação	Baixa	Média	Alta	Urgente	Interrompe cadeia
Contacto					
Pessoa 1	1	1	2	3	1
Pessoa 2	2	2	3	4	1
Pessoa 3	3	3	4	5	1
Médico	-1	-1	1	2	0
112	-1	-1	-1	1	0

Tabela 6 - Lista de contactos e acções a tomar consoante o nível do alerta.

A lista de contactos é configurada por idoso. O seu processo de funcionamento é idêntico para todos, consoante o nível de prioridade, são contactadas determinadas pessoas. Consoante a prioridade do alerta, são contactados diversos contactos. Aquando da detecção de um alerta, a lista é percorrida desde a pessoa 1 até à pessoa n, até serem todos notificados da ocorrência do alerta e da sua prioridade. Existe a possibilidade de não notificar todos os contactos da lista, sendo a cadeia de contactos interrompida. Para tal, tem de se definir que para um determinado idoso e para que prioridades é que a cadeia é interrompida. A cadeia de contactos apenas é interrompida após existir um contacto que confirme que foi notificado, só assim é que se pode interromper a cadeia de notificações. Na Tabela 6, está uma lista de contactos para o idoso. É possível verificar que o medico só é contactado pelo sistema no caso da ocorrência de um alerta de alta prioridade e o 112 só é chamado no caso de um alerta urgente. Tanto o medico como o 112 ou outro serviço de emergência apenas é contactado no caso de alertas que coloquem em perigo o estado de saúde do idoso.

O médico e o 112 são ligações que são efectuadas em paralelo com a lista de contactos, pois mesmo que exista uma confirmação destas entidades, a lista de contactos do idoso é percorrida.

As notificações são efectuadas por um meio de comunicação full-duplex. A comunicação *full duplex* possibilita o envio e a recepção de dados na comunicação. Um exemplo de comunicação *full duplex* é uma chamada telefónica. Com a notificação a utilizar comunicação *full duplex* garante-se que existe uma resposta, caso contrario a comunicação nem é iniciada. Mesmo utilizando esta via de comunicação, não existe a garantia que a pessoa a notificar,

receba a notificação. O número de tentativas efectuadas para notificar um contacto é pré definido consoante o nível de prioridade do alerta, mas pode ser configurado de forma diferente. Quanto mais alto o nível do alerta, maior é o número de tentativas a efectuar. Como a utilização de comunicação exige resposta imediata que pode não surgir, a notificação é efectuada por meio de comunicação *simplex*. A comunicação *simplex* é unidireccional, apenas possibilita o envio de dados. Um exemplo de comunicação *simplex* é o envio de um e-mail.

Após a cadeia de contactos ter terminado ou tenha sido interrompida, todos os contactos a notificar recebem uma notificação por meio de comunicação *simplex*. Após esta notificação o sistema guarda quem foi notificado e o número de tentativas que teve de efectuar para conseguir notificar a lista de contactos.

4.5 Protótipo

Após se ter definido a arquitectura e definido a arquitectura do gestor de informação, foi possível iniciar o desenvolvimento do seu protótipo. O desenvolvimento deste sub-módulo iniciou pelos serviços disponibilizados para os nós sensoriais. O desenvolvimento iniciou por esta etapa, pois encontra-se também em desenvolvimento o modulo monitorização corporal que é baseado em sensores e nós sensoriais. Com o protótipo já desenvolvido é possível registar nós sensoriais e sensores. Aquando do registo dos sensores são criadas as tabelas que permitem guardar os dados desses mesmos sensores, sendo ainda possível introduzir e aceder aos dados dessas tabelas. Tal como foi definido na arquitectura, as tabelas criadas para armazenar os dados dos sensores, são criadas utilizando metadados.

A estrutura que foi criada para os nós sensoriais também já se encontra desenvolvida no protótipo. Assim os nós sensoriais conseguem criar tabelas para armazenar dados, que não sejam dos sensores. A estrutura que os nós sensoriais podem criar é mais complexa pois possibilita a criação de tabelas com recurso aos metadados, com a possibilidade de ligar as tabelas entre si. Tal como nos sensores estas tabelas são criadas utilizando os metadados.

Todas as tabelas que são criadas utilizando os metadados podem ser alteradas ou apagadas. Quando se procede a uma acção deste tipo é necessário manter os metadados coerente. Para

que a informação fique coerente é necessário recorrer a transacções, pois assim a alteração dos dados é executada como uma operação única.

Para testar o protótipo do gestor de informação foi desenvolvido um cliente cuja função é evocar os serviços disponibilizados. Esta ferramenta foi extremamente útil, pois permitiu testar os serviços desenvolvidos.

Após ter desenvolvido o protótipo do gestor de informação e uma aplicação que permitisse efectuar os testes ao servidor, foi dado inicio ao desenvolvimento do protótipo do gestor de alertas. O gestor de alertas desenvolvido recebe um alerta já classificado, e procede as notificações que estão definidas na base de dados. Quando é recebido um alerta, já é possível efectuar notificações por: e-mail, SMS e chamada telefónica pela internet. Com o desenvolvimento destes protótipos foi possível integrar um módulo de sensores que detecta quedas dos idosos. A integração deste módulo foi efectuada com sucesso, sendo possível registar os dados recolhidos pelos sensores no gestor de informação e ainda enviar alertas de quedas.

Neste capítulo foi apresentado o objectivo do centro de controlo, qual a sua constituição e qual o a sua arquitectura física. Para que seja possível detalhar a arquitectura foram apresentados os requisitos necessários para o seu desenvolvimento. Para o seu desenvolvimento, formam também mostradas quais são as tecnologias que vão ser utilizadas. A proposta de arquitectura do centro de controlo que foi estudada, permitiu especificar a constituição de cada processo que vai servir cada cliente. Como a arquitectura é baseada em serviços foram estudados os benefícios da arquitectura SOA e explicada a sua aplicabilidade ao gestor de informação.

O desenvolvimento do CC não seria possível sem antes ter detalhado a arquitectura do GI. Na arquitectura do GI foi apresentada como é que esta foi desenhada para ser escalável possibilitando a integração entre diferentes sistemas. Para conseguir organizar o conhecimento que vai estar representado no GI, são utilizadas ontologias, pois estas permitem expressar de forma coerente sem que seja possível existir diferentes interpretações. Isto apenas é possível pois as ontologias conseguem expressar conhecimento pelo seu conteúdo semântico.

Com a arquitectura do GI concluída, apresentou-se a arquitectura do GA, apresentando as suas principais funções que são as de receber eventos e receber e tratar alertas. É ainda apresentada a definição de alerta e a constituição de uma lista de contactos de notificações. Para concluir este capítulo foi apresentado o resultado obtido com o inicio do desenvolvimento do protótipo resultante desta arquitectura.

5 Considerações finais

A presente dissertação tem como principal objectivo o fornecimento duma plataforma que permita a interoperabilidade de sistemas, que têm como objectivo comum a melhoria do bem-estar dos idosos. Para permitir a interoperabilidade entre sistemas é ainda utilizada uma ontologia que permite a partilha de conhecimento pelo seu conteúdo semântico.

Neste capítulo revêem-se os conceitos apresentados na dissertação, o problema e o projecto de investigação desenvolvido. São apresentados os principais resultados obtidos e é apresentado o trabalho futuro no desenvolvimento desta arquitectura.

5.1 Síntese da tese

A população encontra-se cada vez mais envelhecida, estando os recursos humanos, materiais e financeiros a escassear, impossibilitando a prestação do devido apoio aos idosos. Para minimizar os recursos despendidos, vários sistemas estão em desenvolvimento, sendo um deles o projecto Elder Care. No entanto, e apesar do grande esforço realizado, ainda não existe uma plataforma que consiga integrar os vários sistemas em desenvolvimento.

Para tentar minimizar as falhas existentes na integração, encontra-se em estudo um sistema que possibilite a comunicação entre diferentes sistemas. Esse sistema é o gestor de informação. O GI ainda vai permitir a criação duma estrutura de dados personalizável por sistema externo, para o armazenamento de dados. Desta forma a informação gerada fica centralizada e disponível para ser consultada pelos outros sistemas. No entanto e de forma a conseguir atingir estes objectivos foi levantada uma questão de investigação: Como permitir a

comunicação entre os diversos módulos, de um sistema de apoio a idosos (Elder Care), permitindo a escalabilidade, a interoperabilidade e o conhecimento semântico da informação a trocar?

Para solucionar esta questão, foi efectuada uma revisão da literatura, que permitiu perceber o trabalho que tem vindo a ser desenvolvido e que permitiu resolver problemas de interoperabilidade. Foi então possível verificar que, conjugando tecnologias existentes, era possível definir uma possível solução para o problema proposto.

A solução proposta passa pelo desenvolvimento duma ontologia, que possibilite a adição e partilha do conhecimento semântico que cada sistema representa e disponibilizar a lista de serviços resultante da integração de sistemas. Para possibilitar a comunicação são utilizados *Web-services* pois são independentes da tecnologia, facilitando assim a comunicação entre sistemas diferentes.

5.2 Trabalho realizado

Com a disponibilização dum protótipo desta plataforma, que tem como objectivo a interligação de diversos sistemas que visam a melhoria da qualidade de vida dos idosos, é possível fornecer diversos serviços que consigam recolher armazenar e disponibilizar dados. Desta forma é assim possível que cada sistema ligado a esta plataforma se consiga especializar nas tarefas a que se propôs, delegando para o gestor de informação a tarefa de gerir os dados.

A utilização de padrões SOA traz benefícios no desenvolvimento do projecto. Estes padrões identificam e têm uma solução adequada para resolver problemas que surgem durante o desenvolvimento do projecto. Apesar de ser necessário um esforço suplementar para iniciar o desenvolvimento, este esforço é compensado no futuro pelos benefícios da utilização dos padrões. É de referenciar a alta qualidade de reutilização, a escalabilidade, a gestão do projecto a longo tempo e a redução de custos no desenvolvimento e manutenção a longo prazo. No caso da arquitectura do gestor de informação, os padrões simplificam a estrutura, ajudam na implementação e manutenção a longo prazo.

Este projecto pretende dar a sua contribuição para o problema da interoperabilidade de sistemas permitindo a diferentes projectos com propósitos similares, a troca de informação entre si. Para possibilitar a troca de dados foi desenvolvida uma ontologia que permite representar o conhecimento semântico que cada sistema representa.

Apesar da complexidade envolvida na criação duma ontologia, a sua utilização consegue modelar o conhecimento dum projecto de forma detalhada. Como as ontologias se encontram no nível superior da representação semântica, é fácil de concluir que a sua utilização, consegue expor de forma clara e precisa o conhecimento de um sistema, sem que seja necessário recorrer a outro tipo de representação. O desenvolvimento das ontologias fornece uma hierarquia de classes mais evoluída que um diagrama de classes pode oferecer. No diagrama de classes apenas se tem acesso à hierarquia das classes. Na ontologia ainda se consegue definir as regras que permitem as relações entre as classes, que pertencem a outra hierarquia. As regras que definem as interacções entre as classes são as mesmas que se têm de efectuar aquando do desenvolvimento da aplicação.

A possibilidade de poder questionar a ontologia, possibilita aos utilizadores do projecto Elder Care, obter respostas às questões relacionadas com o conhecimento que se encontra na ontologia. Como o conhecimento se encontra organizado, e é definido com regras específicas do domínio do Elder Care, o resultado das inferências é concreto e relacionado com a pesquisa. As inferências são possíveis pois os motores de inferência conseguem tirar elações lógicas das regras inseridas. As inferências apenas são possíveis pois a ontologia rege-se pelos conteúdos semânticos e consegue aplicar as regras de relacionamento entre classes. Esta definição dos termos, é perceptível por máquinas, podendo assim estas efectuar o trabalho de relacionamento entre dados. Os resultados devolvidos pelas inferências dispensam validação, pois essa tarefa é efectuada pelo motor de inferência, com recurso a semântica dos termos e as regras da ontologia, que definem o domínio do projecto.

Com a arquitectura proposta nesta dissertação, o modelo de operação de motores de base de dados é modificado, pois estas estão optimizadas para trabalhar com registos em tabelas, mas nesta arquitectura, a necessidade é de ter uma tabela por sensor, o que leva à criação de múltiplas tabelas com registos idênticos. Com esta organização dos dados, a informação armazenada pelo sensor é acedida sem a realização de consultas complexas à base de dados.

5.3 Trabalho futuro

O próximo passo no desenvolvimento do gestor de informação é de desenvolver os serviços necessários para a criação de estruturas dinâmicas, sem ter por base os nós sensoriais, possibilitando assim a criação de estruturas de dados, tanto por sistemas como por utilizadores.

A interacção com a ontologia tem de ser melhorada, não permitindo uma manipulação directa, mas sim um acesso à ontologia através de serviços do GI. Uma das funcionalidades que é essencial desenvolver é a da introdução dos sistemas externos na ontologia, que tem de ser efectuado pelo mesmo serviço de registo no GI.

O próximo passo no desenvolvimento da ontologia é de inserir o conhecimento dos restantes módulos que pertencem ao Elder Care. Como o gestor de informação é uma plataforma que pretende interligar sistemas e disponibilizar diversos serviços, pretende-se disponibilizar esta plataforma na Web como plataforma integradora de sistemas de alto nível e de baixo nível.

5.4 Conclusão

Perante um problema que se tem vindo a manifestar na área dos sistemas – a falta de interoperabilidade – realizou-se um estudo na área dos sistemas da saúde. Esta decisão prendeu-se pelo crescente aumento de sistemas que pretendem melhorar o bem-estar dos idosos e pela vantagem que estes sistemas trazem, na mais eficiente gestão de recursos.

A arquitectura aqui apresentada ainda se encontra em desenvolvimento, e tendo a consciência da necessidade da sua finalização, espera-se ter contribuído de forma útil para a abordagem do tema da interoperabilidade entre sistemas.

Os projectos que envolvem diversos sistemas, e que necessitam comunicar, podem ser abordados de diferentes formas, não existindo uma solução padrão que consiga resolver esta

problemática. Sabendo que cada sistema tem necessidades específicas, pretende-se com este projecto tentar encontrar uma solução que resolva os vários problemas inerentes na interligação de sistemas heterogéneos.

Apesar de todos os desafios encontrados ao longo do desenvolvimento deste projecto, este foi realizado com elevada motivação, tendo a esperança de que este projecto consiga definir um caminho na interoperabilidade de sistemas.

Bibliografia

Adam Farquhar; Richard Fikes; James Rice (1997). "The Ontolingua Server: A Tool for Collaborative Ontology
Construction." International Journal of Human Computer Studies.

Alex Rodriguez, I. (2008). "RESTful Web services: The basics."

Altova. (2011). "SemanticWorks - Semantic Web Tool." Retrieved 17-08-2011, from http://www.altova.com/documents/SemanticWorksdatasheet.pdf.

Anthony Fleury, Michel Vacher, et al. (2010). "SVM-Based Multimodal Classification of Activities of Daily Living in Health Smart Homes: Sensors, Algorithms, and First Experimental Results." IEEE transactions on information technology in biomedicine VOL. 14, NO. 2: 274-283.

Anthony P. Glascock and David M. Kutzik (2000). "Behavioral Telemedicine: A New Approach to the Continuous Nonintrusive Monitoring of Activities of Daily Living." Telemedicine Journal Volume: 6(Issue): 33-44.

ASCN (2004). "ASCN NEWS." Journal of Nuclear Cardiology: 101-102.

Athanasios Anastasiou, Paul Quarrie, et al. (2008). "Personal Location Aware Health Care In Europe--The Challenges From
Prototype To Product: The CAALYX Experience." eHealth International Journal.

B.G. Cellera, W. E., E.D. Ilsara, L. Betbeder-Matibetb, M.F. Harrisb, R. Clarkc, T. Hesketha, N.H. Lovelld (1995). "Remote monitoring of health status of the elderly at home. A multidisciplinary project on aging at the University of New South Wales." Internacional Journal of Bio-Medical Computing Volume 40(Issue 2): Pages 147-155.

Cardoso, A. (2000). "Inferência."

Cardoso, J. (2007). Semantic Web services, processes and applications; de Jorge Cardoso & Amit P. Sheth / Semantic Web and semantic Web services; de Liyang Yu.

Chan, M., C. Hariton, et al. (1995). Smart house automation system for the elderly and the disabled. IEEE International Conference on Systems, Man and Cybernetics. Vancouver, BC. Volume: 2: 1586 - 1589.

CISCO (2007). Cisco Unified Communications Manager JTAPI Developers Guide. I. Cisco Systems. San Jose.

Clarisse Odebrecht M.Sc., E. Luciana de Oliveira Gonçalves, et al. (2010). "Da Gerontecnologia a Gerontecnologia." Retrieved 10/2010, 2010, from http://portaldoenvelhecimento.org.br/noticias/artigos/da-gerontologia-a-gerontecnologia.html

Communities, E. (2004). "EUROPEAN INTEROPERABILITY FRAMEWORK
FOR PAN-EUROPEAN EGOVERNMENTSERVICES." European Interoperability Framework.

Corcho, Ó., M. Fernández-lópez, et al. (2002). "WebODE : An Integrated Workbench for Ontology Representation , Reasoning , and Exchange." Workbench: 138-153.
Darcilene Estrela, Helder Aragão, et al. (2006). "Uma implementação de um motor de inferência embarcado em robôs lego."
David M. Kutzik; Anthony P. Glascock; Douglas L. Chute; Thomas T. Hewett and Barbara G. Hornum (1994). System for generating periodic reports, generating trend analysis, and intervention in accordance with trend analysis from a detection subsystem for monitoring daily living activity. United States, Gerotech, Inc.
Erl, T. (2005). SOA: Service-Oriented Architecture: Concepts, Technology, and
Design.
Erl, T. (2007). SOA: Principles of Service Design.
Erl, T. (2009). "soaprinciples."
Erl, T. (2011). "Goals and Benefits of Service-Oriented Computing."
F. Steenkeste, a., , H. Bocqueta, M. Chana and E. Campob (2001). "La mise en place d'une technologie pour observer le comportement nocturne des personnes âgées en institution." Innovation and thecnology in Biology and Medicine Revue of Biomedical technology 22: 25-30.
Felisberto, F., M. Felgueiras, et al. (2011). Improving the Elder Care's Wireless Sensor Network Fall Detection System Using Logistic Regression. CENTERIS 2011, Part III, CCIS 221 proceedings. Vilamoura, Portugal, Springer. 221.
FFII, F. f. a. F. I. I. (2009). "FFII Workgroup on Open Standards." from http://action.ffii.org/openstandards.
Foundation, C. H. (2004). "Clinical Data Standards Explained."
Group, S. (2009). " CHAOS report."
Gruber, T. R. (1993). "A Translation Approach to Portable Ontology Specifications." Knowledge systems Laboratory
Gruber, T. R. (1993). "A Translation Approach to Portable Ontology Specifications by A Translation Approach to Portable Ontology Specifications." Knowledge Creation Diffusion Utilization.
Guarino, N. (1998). "Formal Ontology and Information Systems." 3-15.
H. Inada, H. H., Y. Sekita, K. Ishikawa, K. Yoshida (1992). "A Study on a Home Care Support Information System " in Proceedings of the Seventh World Congress on Medical Informatics: Pages: 349-353.
Harry Chen , T. F., Anupam Joshi (2005). "An Ontology for Context-Aware Pervasive Computing Environments."
Health, C. (2009). "The HL7 Evolution
Comparing HL7 Version 2 to Version 3,
Including a History of Version 2."
Hinkelman, P. D. K. (2010). "Forward Chaining Rules." University of Applied Sciences Northwestern Switzerland School of Business.
HL7. (2004). "HL7 Reference Information Model." from http://www.hl7.org/library/data-model/RIM/C30204/rim.htm.
Horridg, M., Ed. (2011). A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools.
Huff, S. M. (1998). "Clinical Data Exchange Standards and Vocabularies for Messages." AMIA, Inc: 62-67.
IEEE, C. S. (2002). IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture: 1.

IEEE Standards Information Network (2000). "The Authorative Dictionary of IEEE Standards Terms, Seventh Edition."
ITU-T (1994). "Information Technology - Open Systems Interconnection - Basic Reference Model : The Basic Model " INTERNATIONAL TELECOMMUNICATION UNION.
Johnson, M. E. F. R. E. (1999). Domain-Specific Application Frameworks: Frameworks Experience by Industry, New York: John Wiley & Sons.
Jorge Cardoso (2007). Semantic Web Services: Theory, Tools, and Applications, IGI Global.
Kalyanpur, a., B. Parsia, et al. (2006). "Swoop: A Web Ontology Editing Browser." Web Semantics: Science, Services and Agents on the World Wide Web 4: 144-153.
Karen Zita Haigh, L. M. K. and V. G. Janet Myers, Christopher W. Geib, John Phelps, Tom Wagner (2004). The Independent LifeStyle AssistantTM (I.L.S.A.): AI Lessons Learned. To appear in IAAI 04. San Jose CA.
LaMonica, M. (2005). "Sun Microsystems on Wednesday released its latest Jini development toolkit under the Apache open-source license." CNET News
Liege Mata Álvares, Rosângela da Costa Lima, et al. (2010). "Ocorrência de quedas em idosos residentes em instituições de longa permanência em pelotas, Rio Grande do Sul, Brasil."
Luís Arriaga. (2009). "Opinião: Interoperabilidade de Sistemas de Informação." http://tek.sapo.pt, from http://tek.sapo.pt/opiniao/opiniao_interoperabilidade_de_sistemas_de_inf_989788.html#page=2.
Lyons, R. E. and W. Vanderkulk (1962). "The Use of Triple-Modular Redundancy to Improve Computer Reliability." IBM Journal of Research and Development 6(2): 200-209.
Marcelino, I. (2008). "Estruturação de um sistema de monitorização remota e de prevenção de infoexclusão de idosos no seu domicílio."
Maria Rita Pinto; Stefania De Medici; Clarke Van Sant; Alfredo Bianchi; Andre Zlotnicki and Claudio Napoli (1998). Ergonomics, gerontechnology, and design for the home-environment. Applied Ergonomics 31: 317-322.
Mário Joaquim Firmino Leite Faria, Prof. Dr. Luís Paulo Reis, et al. (2009). Definição de uma Ontologia Aplicada ao Futebol. Mestrado Integrado em Engenharia Electrotécnica e de Computadores. Porto, Faculdade de Engenharia da Universidade do Porto
Definição.
Martin Turcotte (2005). "Time spent with family during a typical workday, 1986 to 2005."
Metro. (2010). "Metro Specifications." from https://metro.dev.java.net/guide/Metro_Specifications.html.
Microsoft. (2010, 2010-06-09). "Windows Communication Foundation." from http://msdn.microsoft.com/en-us/library/dd456779.aspx.
NISO, N. I. S. O. (2004). Understanding Metadata, NISO Press.
Noy, N. F. and D. L. Mcguinness (2000). "Ontology Development 101 : A Guide to Creating Your First Ontology." Development: 1-25.
Object Management Group, I. (2010). "OMG Unified Modeling LanguageTM (OMG UML), Infrastructure."
Paterson, G. (2010). "HL7 V3 and the Flow of Health Information." pan-Canadian Health Informatics Collaboratory project.
Pedro Pita Barros and Jorge de Almeida Simões (2007). "Portugal Health system in Transition." European Observatory on Health Systems and Policies Vol.9, No.5.
Raquel Elias Carneiro and Parcilene Fernandes de Brito (2005). "Definição de uma Ontologia em OWL para Representação de

Conteúdos Educacionais." VII ENCONTRO DE ESTUDANTES DE INFORMÁTICA DO ESTADO DO TOCANTINS.
rfc1122 (1989). rfc1122. R. Braden, Internet Engineering Task Force.
Samuel Brás, Rui Rijo, et al. (2011). "Information Management, proposal for an integration platform using metadata." In CENTERIS 2011 221.
Shaver, D. (2007). "HL7 101: A Beginner's Guide." The Record Vol. 19 No. 1 P. 22.
Skandier, D. G. T. (2009). Network+ Study Guide, Fourth Edition.
Sunderraman, J. S. H. E. D. R. (2004). Health Level-7 Compliant Clinical Patient Records System, ACM Symposium on Applied Computing.
T. Tamura, T. Togawa, et al. (1988). "A bed temperature monitoring system for assessing body movement during sleep." Clinical Physics and Physiological Measurement Volume 9, Number 2.
Technologies, D. (2011). "Converting from HL7 2.x to HL7 3.x." Retrieved 2011, from http://web.datadirect.com/resources/dis/hl72x-to-hl73x/index.html.
Thomas Erl, Ed. (2009). SOA Design Patterns. United States of America, SOA Systems Inc.
Thomas L. Harrington and Marcia K. Harrington (2000). Gerontechnology why and how, holanda, Shaker.
Tiago Semprebom; Marcus Yuzuru Camada; Igor Mendonça (2007). "Ontologias e protégé."
United Nations, Department of Economics and Social Affairs, et al. (2007). "World Population Prospects - The 2006 Revision, Highlights."
Upkar Varshney (2007). "Pervasive Healthcare and Wireless Health Monitoring." Springer Science + Business Media.
Viinikkala, M. (2004). "Ontology in Information Systems."
World Wide Web Consortium (W3C). (2004). "OWL Web Ontology Language." from http://www.w3.org/TR/owl-features/.
World Wide Web Consortium (W3C). (2004). "Web Services Glossary." from http://www.w3.org/TR/ws-gloss/.
Yuan Ren, J. Z. P., Yuting Zhao (2010). "Closed World Reasoning for OWL2 with Negation As Failure." Department of Computing Science, University of Aberdeen, Aberdeen, UK.

Anexo I – Questionarios direccionados aos grupos de trabalho

Questionário direccionado ao módulo análise localizador pessoal

Questionário

Qual a informação que cada utente tem de ter armazenada?

Idoso (nome, tipo de mobilidade que lhe está associada - conduz, usa transporte públicos);

Agenda pessoal (data, hora, actividade prevista, local);

Histórico de actividades (manter todos os dados da agenda para efeitos de análise histórica);

Contactos (nome, email, telemovel, endereço postal) de responsáveis pelo idoso para situações de risco e envio de alertas;

Alertas (tipo de alertas – aviso, grave, muito grave), lista de contactos por cada tipo de alerta,

Histórico de ocorrências (alertas efectivos, falsos alertas, ...);

Informação de georeferênciação adicional (além da disponível no google maps).

Qual a informação que é necessário consultar?

A que se indica para armazenamento;

A informação a consultar tem que parâmetros? Utente, tipo sensor, grupo de sensores, delimitação por zonas especificas do utente, entre datas ou outras.

A análise dos dados é para se efectuar remotamente?

Os dados disponíveis na aplicação móvel indicados para rmazenamento na primeira secção terão que ser sincronizados periodicamente com o sistema central;

Como são enviadas informações urgentes? (definir prioridade no xml)

Existem 2 níveis de prioridade, normal e urgente. É necessário um terceiro?

Qual tecnologia vai ser utilizada no desenvolvimento? E para comunicação?

Aplicação para dispositivos móveis desenvolvida para Android.

Questionário direccionado ao módulo segurança

Questionário

Como é efectuada a ligação entre pontos? SOAP 1.2

Qual a segurança aplicada na ligação? SSL

Utilizar certificados digitais para cifrar a informação é viável? Sim dado que a base-station possui recursos suficientes para efectuar criptografia assimétrica.

Para fazer autenticação, utilizam-se certificados digitais de 512bits renovando o mesmo de 3 em 3 meses. Julgamos que a periodicidade de revogação não é importante nesta fase do projecto, quanto ao tamanho das chaves vai depender do equipamento utilizado como base-station. Para fins de teste o mais sensato é começarmos com 128bits (pensando num telemóvel como base-station).

Para enviar informação é viável enviar um ficheiro cifrado que contenha a autenticação e os dados? Evita assim os protocolos de ligação. Uma comunicação stateless adiciona overhead ao tráfego, no entanto se for statefull têm de ser definidos os limites do que se considera uma sessão. Se o overhead adicionado não for significativo, julgamos que a primeira opção pode ser a mais adequada.

Qual tecnologia vai ser utilizada no desenvolvimento? A tecnologia utilizada neste módulo vai de encontro à tecnologia usada nos outros módulos, o nosso módulo é responsável pelas questões de segurança independentemente das tecnologias de suporte. E para comunicação? XML

Questionário direccionado ao módulo monitorização local

Questionário

→ Levando em conta que Sensor é um ***Mote*** com os diversos sensores de aquisição. Na nossa opinião estão a confundir sensor com nó sensorial, um nó sensorial pode ter vários sensores e uma rede sensorial presente no sistema vários nós sensoriais.

Como identificar univocamente um sensor?

→ (Falando do Mote) Colocando um ID numérico (16bit).

Quais as funções de cada sensor (Mote)?

→ Cada Mote deve recolher dados do acelerómetro e restantes sensores, efectuar uma pré-avaliação dos mesmos e decidir se é motivo de alerta ou não, se sim o alerta será enviado. Caso seja necessário serão colectados dados para efeitos de estatística/relatório e enviados consoante a necessidade do sistema de controlo.

Informação necessária de guardar por tipo de sensor?

→ ID do Mote, Data do alerta, tipo de alerta, dados colectados pelos sensores.

Qual a periodicidade do envio de dados?

→ Depende da utilização do nó sensorial. Normalmente, o mínimo possível, o uso de rádio é muito penoso.

Existe algum concentrador para enviar informação?

→ Gateway será Infra-estrutura da casa, ou Telemóvel.

A informação é enviada em formato xml. Qual a estrutura do mesmo?

→ O serviço é que deverá definir a estrutura, nós apenas enviaremos os dados sobre a estrutura definida pelo mesmo.

Existe outra informação que seja necessário guardar? Info da rede

→ De momento não podemos definir mais dados do que os indicados na questão 3. Qualquer outra informação deverá ser definida em conjunto.

Qual a informação que cada utente tem de ter armazenada?

→ Armazenada onde? Informação de que tipo? Nome? BI? Esta pergunta não é definida pelo nosso módulo.

Como são enviadas informações urgentes? (definir prioridade no xml)

→ Neste momento para o módulo ***fall detection*** existem 2 graus de alerta (*Critical* que serão valores já analisados e definidos como alerta e *Check* que são valores que necessitam de extra analise pois são valores que não estão definidos como normais, mas não são sem sombra de duvidas valores Critical). O Sistema deverá estar preparado para receber novos tipos de alerta.

Existem 2 níveis de prioridade, normal e urgente. É necessário um terceiro?

→ Respondido na questão anterior. Caso seja necessário mais informação entrar em contacto connosco para ver se percebemos melhor a pergunta.

Para enviar informação é viável enviar um ficheiro cifrado que contenha a autenticação e os dados? Evita assim os protocolos de ligação.

→ Não faz parte do nosso módulo. Falar com Luís e Edgar.

Que tipos de sensores são passíveis de serem implementados? (acelerómetro, medidores de tensão, …)

→ Pelo menos acelerómetro e Termómetro. No futuro está planeado a inclusão de sensores ECG. Ainda em estudo está a inclusão de sensores de medição do nível de oxigénio e açúcar no sangue. Têm que definir uma arquitectura modular onde se possa usar qualquer sensor.

Qual tecnologia vai ser utilizada no desenvolvimento? E para comunicação?

→ Deve ser elaborada uma Arquitectura independente das tecnologias...

Estrutura do xml:

```
<BodyNet>
	<Identificação>
			<Id> </Id> //é o id do sistema e não de um mote especifico
			<Nome> </Nome> // Informação redundante ? Não deveria isto estar do vosso lado ? ID → Nome
			<date></data>
	</Identificação>
	<Tipo_alerta></Tipo_alerta> // queda, ritmo cariaco etc etc
	<Prioridade> </Prioridade>

	<Dados tipo=...,amostragem=...  > 1:n //cada um dos tipos de sensor
			<data></data> 1:n //dados desse sensor
	 </Dados>
</BodyNet>
```

Anexo II – Princípios SOA utilizados para a arquitectura do gestor de informação

---

Índice das figuras ilustradas no anexo II

Efeitos do Serviço de Orientação (*Service-Orientation*)

O grande efeito que o *Service-Orientation (SO)* oferece nas aplicações é o de reutilização dos serviços que as compõem. O *Service-Orientation (*SO) estabelece uma listagem de serviços, sendo que a sua grande maioria são serviços reutilizáveis e agnósticos, assim os serviços ficam posicionados de forma lógica para que seja possível o seu acesso e reutilização. A grande vantagem da utilização do *Service-Orientation* nas aplicações é a garantia da reutilização de serviços.

Com a organização que é feita aos serviços é possível agrupa-los de forma específica, algo que não era possível antes, pois os serviços não eram construídos a pensar na reutilização. Com a ideologia do SO, a lógica é de centralizar sempre que possível, são automatizados os processos existentes e são acrescentados novos processos através da composição dos serviços disponibilizados. O que isto origina é uma mudança dos requisitos dos negócios, onde cada vez menos se pensa em novas aplicações ou extensões para aplicações, mas sim em novas composições de serviços.

Quando a composição de serviços começa a ficar mais comum, a utilização do SO começa a desvanecer. Assim para resolver este problemas, as aplicações não são mais umas caixas fechadas que resolvem um conjunto de problemas específicos. As aplicações são assim mais uma composição de serviços, que podem fazer parte de outra composição. O conceito de aplicação fechada deixa de fazer sentido, passando a existir um conjunto de serviços combinados que fazem parte de uma lógica.

Uma aplicação num ambiente deste género perde a sua identidade, pois já não é concebida para resolver apenas um problema específico, mas sim deixar um conjunto de serviços que além de resolver um problema podem fazer parte de outra composição. Pode ainda existir a ideia de que o verdadeiro SO não existe e que é de facto apenas mais uma composição de serviços. No entanto existem serviços que não são realmente de processos de negócios agnósticos.

Com o conceito do SO, a lógica central de uma aplicação não é exclusiva a apenas uma aplicação, mas sim de diversas aplicações que contribuem para a resolução de diversas lógicas.

A ideia de ter os serviços inventariados por tipo de serviço, que têm em comum um conjunto de regras padronizadas e pelas suas unidades de reutilização, desafia o conceito tradicional de "integração". O conceito tradicional de integração implicava a conexão de duas ou mais aplicações que podiam ou não ser compatíveis. No entanto, a crescente necessidade de ligar diversos serviços, obrigou a um melhoramento das ligações que eram estabelecidas e na confiança das ligações, o que tornou a integração de sistemas numa parte importante da indústria de serviços.

Os serviços desenhados para serem "intrinsecamente interoperáveis" eram construídos com o cuidado de que teriam de interagir com um vasto conjunto de serviços exigentes, muitos dos quais desconhecidos. No caso da solução lógica da empresa ser constituída por um inventário de serviços intrinsecamente interoperáveis, possibilita a liberdade de fazer diversas combinações de serviços já existentes de forma a conseguir automatizar os futuros desenvolvimentos.

Como resultado, o conceito de integração começa a desaparecer. A troca de dados entre unidades de serviço diferentes, começa a ser natural no desenho da aplicação e uma característica secundária. No entanto, isto só deve de acontecer quando a grande parte da lógica da solução da empresa é representada por um inventário de qualidade de serviços. Mas para se conseguir chegar a esse ponto de integração é necessário conseguir integrar os sistemas já existentes e mais antigos e ainda conseguir integrar os novos serviços aos serviços antigos.

Objectivos e benefícios do *Service-Oriented Computing*

É muito importante perceber, o porquê de, tanto fornecedores de serviço como consumidores de serviços das comunidades de indústria de tecnologias de informação (TI), estarem a fazer um esforço acrescido para adoptar a plataforma do *service-oriented computing*, mesmo sabendo todos os aspectos de mudança que são exigidas.

Os objectivos que o *service-oriented computing* pretende atingir é muito ambicioso e atractivo, para qualquer organização que pretenda melhorar o desempenho. Para dar forma a

visão do *service-oriented computing* foram idealizados um conjunto de objectivos e benefícios, que as empresas devem atingir para conseguir adoptar os princípios do *service-oriented computing.*

Este conjunto foca sete objectivos e benefícios que são o aumento da interoperabilidade intrínseca, o aumento da federação, o aumento da diversificação das vendas, o aumento de negócios e das tecnologias de alinhamento, o aumento do retorno do investimento (ROI), uma maior agilidade organizacional e uma redução de encargos de TI.

É essencial perceber este conjunto de objectivos e benefícios antes sequer de pensar em aplicar a computação orientada a serviços (SOA).

Este conjunto de objectivos divide-se em objectivos e benefícios estratégicos e estão inter-ligados tal como é mostrado na figura1.

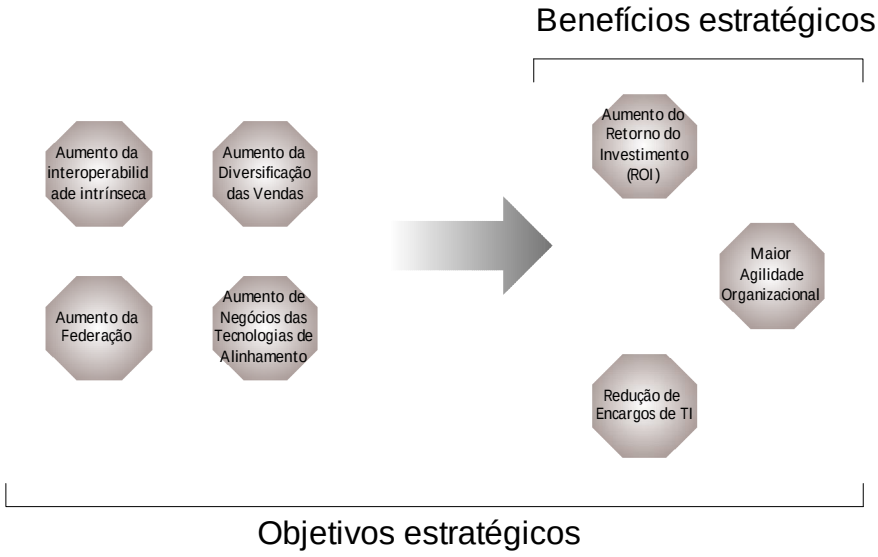


Figura 1 - Os sete objectivos identificados estão inter-relacionados e podem ser classificados em dois grupos: os objectivos estratégicos e os benefícios estratégicos daí resultantes (Erl 2011).

Descrição da arquitectura (desenho e texto)

Centralização de serviços. (*Logic Centralization*): A vantagem da utilização deste padrão é de conseguir centralizar os serviços fornecidos. Logo consegue-se ter a garantia que apenas existe um serviço que executa determinada lógica. Esta lógica pode ser utilizada por diversos serviços. A vantagem é que quando é necessário proceder a actualizações na lógica do sistema, apenas é necessário actualizar a lógica numa determinada função para que todo o sistema fique actualizado.

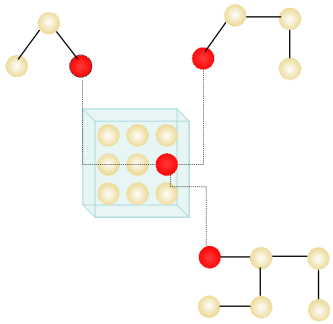


Figura 32 - Utilização dos serviços centralizados em diversos serviços.

Serviços normalizados (*Service normalization*): A utilização deste padrão garante que apenas existe um serviço que consegue executar determinada tarefa. Assim é garantido que os serviços se encontram bem delimitados.

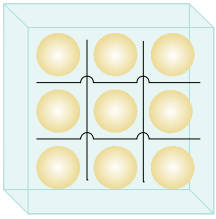


Figura 33 – Delimitação das tarefas de cada serviço.

Esquema canónico (*Canonical schema*): Para garantir que um serviço que tenha de ser usado em diferentes *Web-services* tenha a recepção da mesma informação no XML é usado apenas um XSD que valida esse tipo de informação, mesmo que essa informação se encontre replicada em diferentes ficheiros XML. Com isto é garantido que sempre que é necessário utilizar um serviço, independentemente do seu destino, a informação que se encontra no ficheiro relativa a esse serviço é validada pelo mesmo ficheiro XSD.

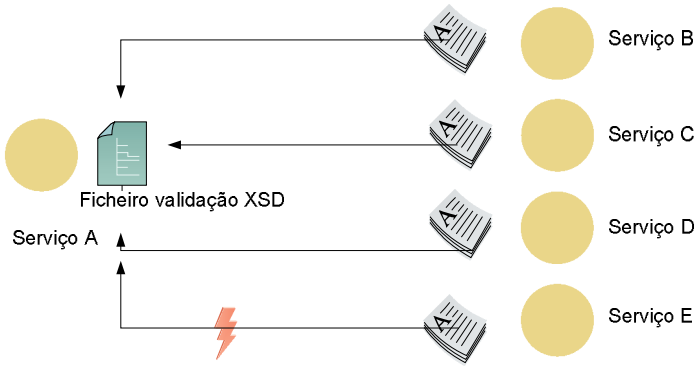


Figura 34 – Validação das mensagens para utilização de um serviço.

Abstracção de utilidade (*Utility abstraction*): Quando lógica não centrada a negócios é juntada com lógica de negócio específica, resulta em implementações redundantes de funções de utilidades em comum através de diferentes serviços. Neste projecto isto acontece com o serviço de *logging*.

A solução é criar uma camada de serviço dedicada a processar as utilidades, fornecendo reutilização de serviços de utilidades a serem usados em outros serviços do inventário. Assim, o modelo de serviço de utilidade é incorporado na análise e nos processos de desenho para apoiar a abstracção da lógica de utilidade, e mais passos são efectuados para definir contextos de serviços balanceados.

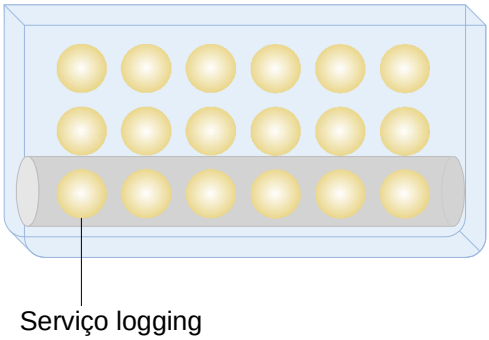


Figura 35 - Serviços transversais numa camada distinta.

Por outro lado, o facto de se estar a agrupar os serviços utilitários em camadas próprias acaba por influenciar o tamanho, complexidade e exigências da performance das composições, já que a lógica utilitária é distribuída através de múltiplos serviços.

Schema centralization: Quando se pretende utilizar um serviço é necessário utilizar mensagens XML, estas podem ser validadas por um ficheiro XSD. Para melhorar o processo de validação dos ficheiros XML são criados vários ficheiros XSD. Cada ficheiro XSD vai validar uma parte do ficheiro XML recebido. Esta funcionalidade optimiza o serviço pois o mesmo ficheiro XSD é utilizado para validar diversos ficheiros XML. Assim caso ocorra alguma actualização de um determinado serviço, apenas é necessário alterar um ficheiro XSD para essas alterações terem reproduções em vários locais de validação.

Exemplo: A autenticação é efectuada em todos os serviços. Caso o processo de validação seja alterado, apenas é necessário alterar o XSD que trata da validação para se alterar a validação em todos os serviços do projecto.

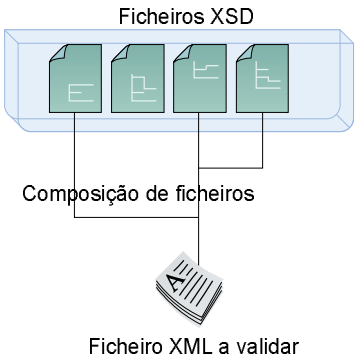


Figura 36 - Composição de ficheiros XSD para validar um ficheiro XML.

Inventory endpoint: Todos os serviços disponibilizados podem ser úteis para resolver lógicas, para as quais não foram inicialmente previstas. De forma a garantir uma maior segurança é necessário ter um serviço que execute a ponte de ligação entre o serviço e cliente. Desta forma consegue-se aumentar a segurança, proteger de ataques, não dar a conhecer a estrutura dos serviços e melhorar o acesso ao serviço, isto com políticas de acesso reforçadas. De forma a garantir a aplicação deste padrão vai ser implementado um proxy, sendo que os clientes podem usufruir dos serviços, com as mesmas funcionalidades sem aceder de forma directa ao serviço.

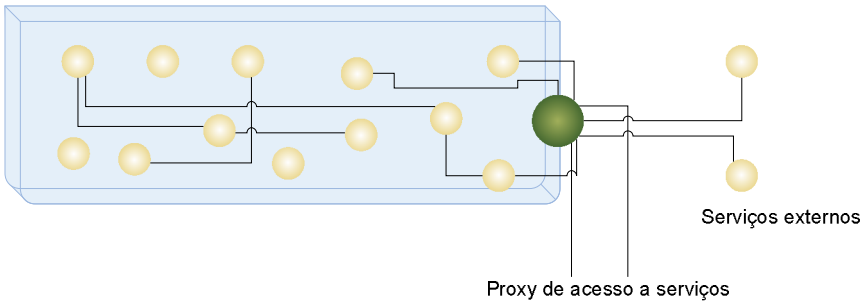


Figura 37 - Acesso a serviços usando um intermediário.

Metadata centralization: De forma a minimizar o esforço e gasto de tempo a desenvolver novos serviços, é preferível procurar na lista de serviços existente se não existe algum serviço disponível que execute o que se pretende, nem que seja de forma parcial. Assim sempre que é

necessário desenvolver um novo serviço, existe um serviço cuja única função é disponibilizar uma lista actual dos serviços já fornecidos. Desta forma, evita-se o desenvolvimento de serviços já existentes, é possível aproveitar serviços já existentes para minimizar o tempo de desenvolvimento de novos serviços e ainda é possível eliminar a redundância de serviços.

Exemplo: A informação dos serviços disponibilizada é efectuada através de WSDL.

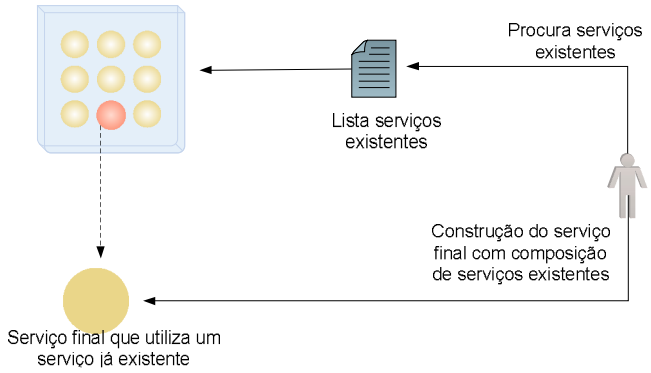


Figura 38 - Listagem dos serviços para posteriores desenvolvimentos.

Canonical versioning: Quando um cliente utiliza um serviço, é necessário saber qual a versão que o cliente suporta. Como forma de evitar incompatibilidades de versões, aquando do envio do ficheiro XML, a versão do mesmo encontra-se contida nos metadados do ficheiro. Isto possibilita ao servidor responder com a versão mais actual do serviço e que respeite as normas da versão do ficheiro.

Service encapsulation: Existem funções, que são criadas com o objectivo de resolver um determinado problema num serviço, no entanto esta função não se encontra disponibilizada para uso geral. De forma a tornar esta função num serviço e de poder ser utilizada por outras entidades, deve-se encapsular essa função num serviço fornecido. Isto possibilita o fornecimento de serviços que servem propósitos para os quais não foram inicialmente criados. Uma forma de encapsular os serviços é de publicar os mesmos com um WSDL.

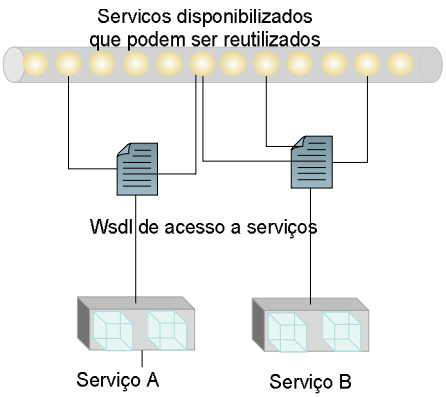


Figura 39 - Encapsulamento de serviços por um wsdl.

Implementação redundante (*Redundant implementation*): Existem serviços que são disponibilizados e que vão ser amplamente utilizados por diversos serviços. Estes serviços são um ponto de falha do sistema, pois caso ocorra um erro neste serviço o sistema fica inoperável. Para resolver este problema, podem-se criar serviços redundantes que garantam resposta mesmo no caso de existir uma falha num dos serviços. Ainda é possível transferir os pedidos que estão bloqueados e a espera de resposta para um serviço que tenha menos carga ou que não tenha reportado um erro (fail-over). É necessário existir um esforço de sincronismo entre estes serviços.

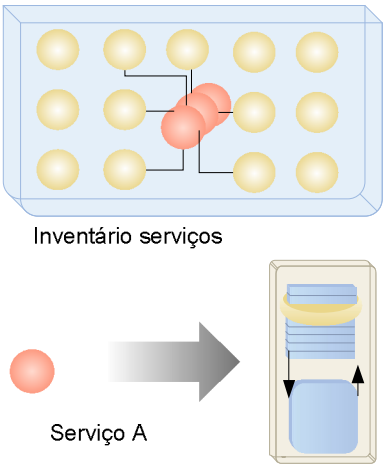


Figura 40 - Replicação de um serviço reutilizável.

Partial validation: As mensagens utilizadas para os serviços são validadas antes de existir alguma execução. Se esta validação for efectuada aquando da entrada do ficheiro, todo o ficheiro vai ser validado. Isto pode tornar o sistema lento e com um grande tempo de resposta caso tenha um grande número de ficheiros a validar. Para resolver este problema, a validação é feita de forma parcial, validando apenas, a parte do ficheiro que vai ser lida, para a execução do serviço. Desta forma é possibilitado fazer uma validação por secção minimizando o processamento aquando da entrada de múltiplos ficheiros XML que necessitem de grande processamento.

Triagem de mensagens (*Message screening*): De forma a proteger o sistema contra ataques ou mensagens XML mal formadas, que podem originar um comportamento indesejável. A forma de solucionar isto é de ter mecanismos de controlo de mensagens que garantem que quando as mensagens vão ser processadas, já não têm conteúdo prejudicial. Esta protecção vai tornar o sistema mais pesado, pois existe a verificação das várias mensagens trocadas, com rotinas especializadas para detecção de ataques. Assim são minimizados os conteúdos nocivos que entram no sistema. A validação das mensagens traz uma carga adicional ao sistema, pois estas têm de ser validadas com rotinas especiais. Pode ser também impossível verificar todos os conteúdos prejudiciais.

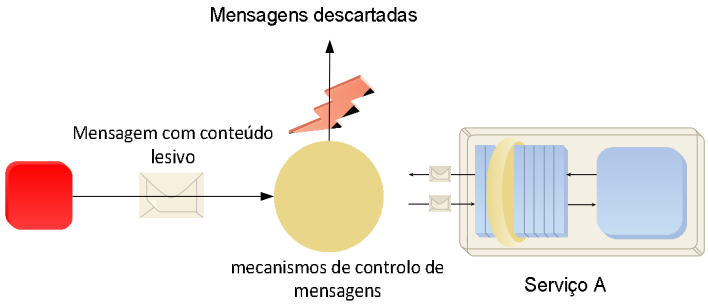


Figura 41 - Verificação da mensagem antes de evocar o serviço evitando ataques.

Subsistema confiável (*Trusted subsystem*): Existem entidades no sistema que têm acesso a serviços que não estão disponibilizados para todos os clientes, estas entidades são designadas de entidades confiáveis do sistema. Estes serviços têm de ser protegidos, pois caso sejam comprometidos podem colocar em risco a integridade do sistema. Para proteger estes

serviços, a autenticação é efectuada com as próprias cadenciais do serviço em nome do cliente. Caso estes serviços sejam atacados podem permitir o acesso a recursos internos.

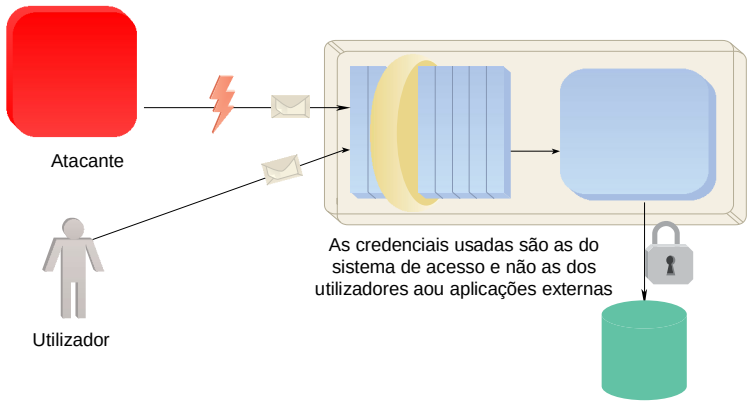


Figura 42 - Utilização de credenciais internas em nome do cliente impossibilitando o acesso directo à base de dados.

Service perimeter guard: Para garantir acesso as funcionalidades da rede interna do sistema, sem expor os seus recursos, necessitam de um sistema intermédio que garante um ponto seguro entre o sistema e os utilizadores externos. O sistema intermédio vai interpretar a mensagem e no caso de ser detectado algum tipo de ataque a mensagem é rejeitada. Apesar da aplicação deste padrão trazer um acréscimo de complexidade e processamento, melhora a qualidade do serviço prestado.

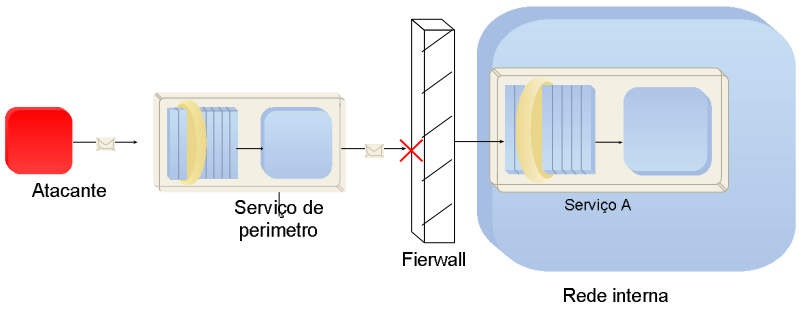


Figura 43 - Colocação de um perímetro de segurança para prevenir ataques do exterior.

Decoupled contract: De forma a prestar um serviço é necessário ter um contracto que possa indicar o tipo de serviço que vai ser prestado. Para ser considerado um serviço tem de ser um contracto físico separado da implementação do serviço. Estando o contracto desacoplado do serviço é possível separar a lógica do negócio da lógica do serviço. Assim o serviço apenas tem de respeitar o que se encontra no contracto, podendo assim evoluir para outras finalidades.

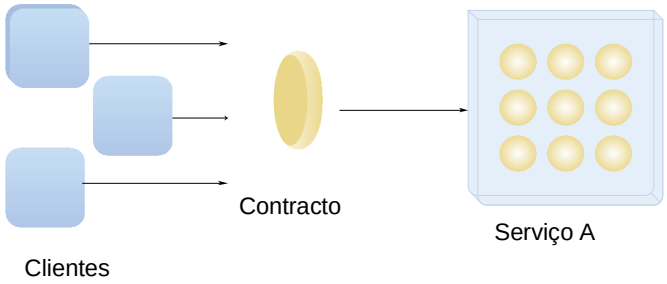


Figura 44 - Contractos separados dos serviços. O serviço atende o cliente como está no contracto sem impedir que este evolua.

Contract centralization: Cada cliente tem um contracto com uma determinada versão de um serviço. Este padrão tem como objectivo dar a conhecer as versões dos contractos existentes e ainda indicar a versão mais adequada para o serviço pretendido pelo cliente. Isto vai impedir que um cliente que tenha acesso aos serviços, por pontos distintos não tenha respostas diferentes. Assim cada cliente acede sempre a verão que se encontra no contracto que realizou. Com a aplicação deste padrão é ainda possível mostrar a evolução dos serviços, analisando as suas versões.

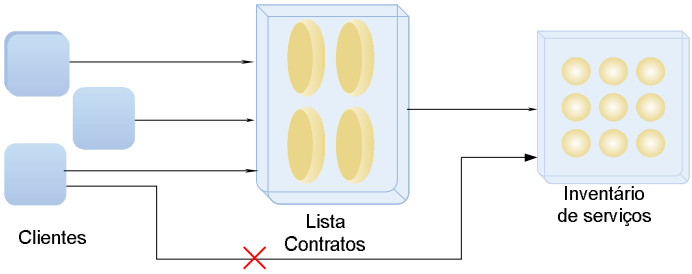


Figura 45 -Acesso aos serviços que estão no contracto.

Validation abstraction: De forma a permitir os serviços a albergar novas funcionalidades, algumas alterações têm de ser efectuadas. Além de se acrescentar funcionalidades ao serviço é necessário também alterar as validações necessárias para a utilização do mesmo. Uma forma de facilitar as validações é abstrair as mesmas da lógica do serviço. Assim sempre que um ficheiro XML entra, e é efectuado um pedido, é validado por um ficheiro XSD. A alteração da validação é independente do serviço, não sendo necessário modificar o serviço, mas sim alterar o ficheiro XSD de validação. Assim o serviço é independente da validação, tornando-o mais adaptável a novas funcionalidades.

Compatible change: Quando é necessário alterar algum contracto de serviço tem de se ter em atenção que existem clientes que podem ficar sem acesso ao mesmo. De forma a evitar tais situações, as alterações aos contractos de serviços devem de ser retro compatíveis com versões anteriores já existentes, o que evita impactos negativos nos clientes. As alterações aos contratados de serviços podem ser efectuadas por extensões de serviços, flexibilização dos serviços ou então aquando da celebração do contracto, não fazer um contracto mas sim diversos contractos, dividindo os serviços contratados por diversos contractos.

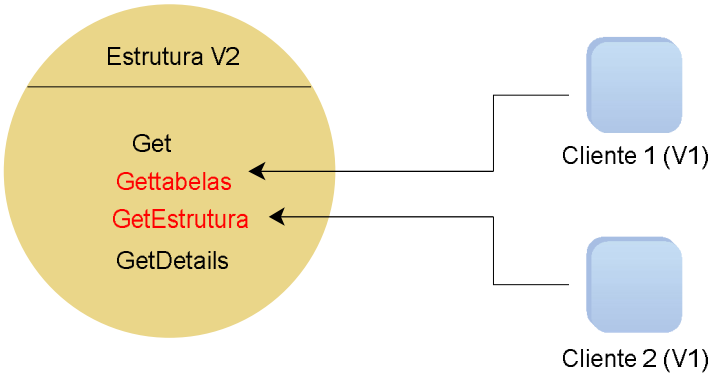


Figura 46 - Adição de novas funcionalidades ao contracto mantendo as funções da versão anterior.

Version identification: Quando é necessário alterar a versão do contracto de serviço, é essencial dar essa informação aos clientes, minimizando assim o impacto negativo que eles vão sentir. Com a disponibilização de contractos de serviços Web os números das versões são incorporados directamente na mensagem. As versões podem ser incorporadas no *namespace* ou nas anotações. Desta forma, o cliente tem sempre a informação da versão do contracto de serviço que está a usufruir.

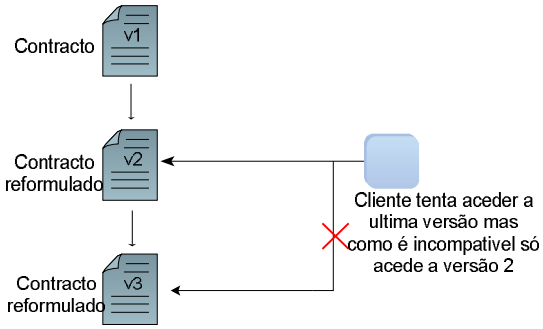


Figura 47 - A versão do contracto está no contracto e o cliente sabe a qual versão pode aceder.

Service refactoring: Existem serviços que com o passar do tempo vão ficar desactualizados ou desadequados, embora esses serviços continuem a ser utilizados pelos clientes. Para manter o serviço actualizado sem alterar o contracto se serviço, altera-se a lógica do serviço e/ou a sua

implementação e mantém-se o contracto de serviço. Para que os clientes que estão a usufruir dos serviços não necessitam de efectuar alterações nas suas aplicações. De uma forma simples mantém-se a forma de interagir com o serviço de forma a manter o contracto de serviço, mas altera-se a lógica ou a tecnologia do serviço.

.

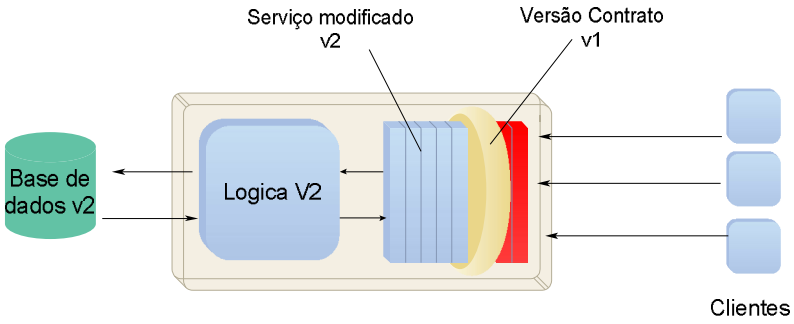


Figura 48 - Acesso a um *Web service* que tem a lógica modificada mas mantém o serviço contratado.

Service decomposition: Existem serviços que efectuam muitas tarefas, estando estes serviços por vezes implementados como um todo. Ou seja um serviço que tem diversos sub-serviços integrados na mesma lógica, o que impede a reutilização destes sub-serviços. De forma a resolver este problema, sempre que existe um serviço que tenha muita lógica para executar, essa lógica é separada. Assim são disponibilizados vários serviços mais simples o que facilita na gestão do projecto, pois os vários serviços podem ser reutilizáveis para outros propósitos, que não só o inicialmente previsto. Os serviços com lógicas extensas passam a ser executados por um conjunto de várias lógicas mais pequenas. Quando forem necessários futuros desenvolvimentos, não irá ser necessário replicar serviços, mas sim utilizar ou caso seja necessário actualizar os serviços já existentes.

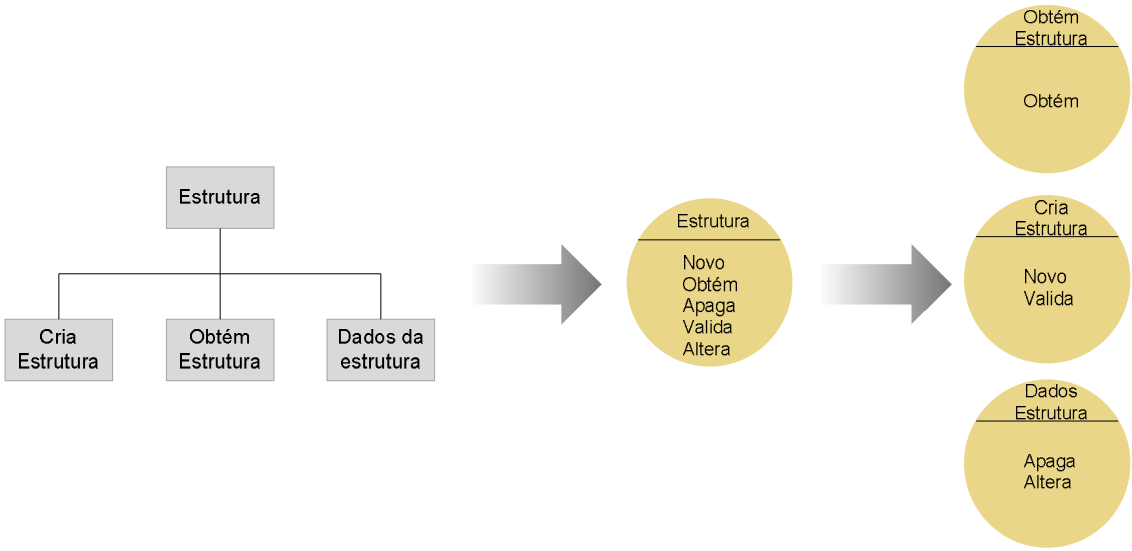


Figura 49 - Decomposição de um serviço para possibilitar uma melhor reutilização.

Proxy capability: Com o evoluir do tempo podem existir serviços cuja lógica seja demasiado extensa, nestes casos a solução é decompor a lógica em diversos serviços. Podem surgir problemas com clientes que já tenham esses serviços contratados. De forma a não cortar o acesso aos clientes o serviço é mantido com uma interface lógica, cuja função é manter a interface com o cliente estável, mas a sua lógica de execução é alterada. A interface com o cliente é passada para um proxy que consegue fazer a ligação entre o cliente e a lógica pretendida. Esta alteração é transparente para o cliente, sendo que este não percebe a alteração que ocorreu.

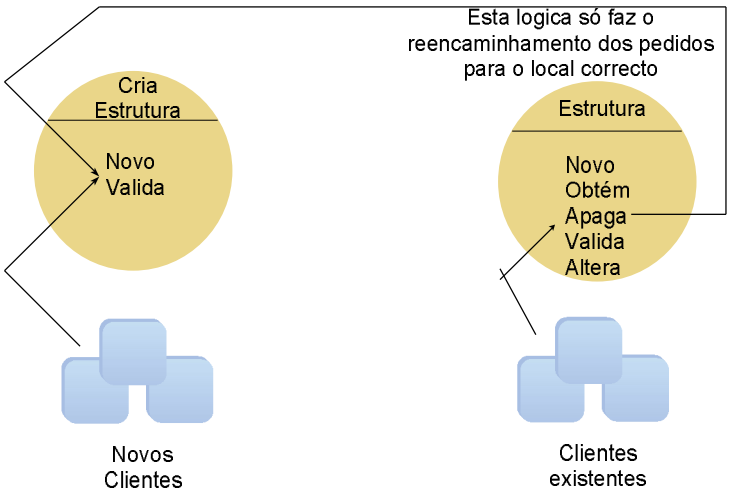


Figura 50 - Alteração da lógica para um proxy que indica onde está o serviço pretendido.

Distributed capability: Como esta plataforma vai ter algumas exigências a nível de processamento, não é recomendável ter todos os serviços alocados a apenas um servidor. De forma a melhorar o desempenho do sistema, e a garantir resposta a múltiplos pedidos, os serviços vão ser distribuídos por diversos servidores, e ter um servidor que distribui os pedidos pelos diversos servidores. Assim, o cliente não tem a percepção da distribuição existente e melhora-se o desempenho do sistema.

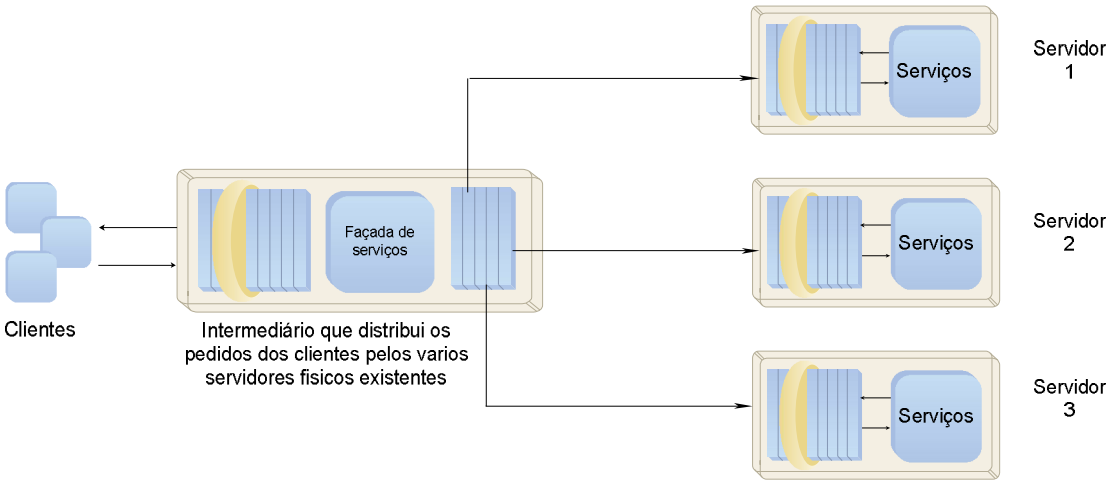


Figura 51 - Distribuição dos pedidos pelos vários servidores físicos.

Capability composition: É a capacidade que os serviços têm de agrupar capacidades de forma a resolver um problema para o qual não foram inicialmente concebidos. Consegue-se assim resolver diversos problemas com a capacidade que os serviços têm de combinar e de projectar a sua lógica a outros serviços.

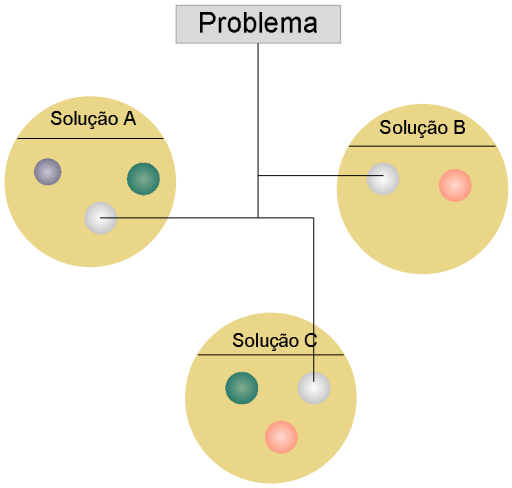


Figura 52 - Combinação da lógica nas diversas soluções para resolver o problema.

Capability recomposition: A capacidade que os serviços têm de agrupar capacidades, garante uma vantagem na resolução de novos desafios. No entanto, pode existir um desperdício de recursos caso a capacidade de se agrupar apenas possa ser feita uma vez. É possível assim, manter as capacidades dos serviços a resolver os problemas para os quais forma inicialmente concebidos e ainda têm a capacidade de se agregar a novos serviços, conseguindo assim agregar-se a várias soluções. Esta funcionalidade permite uma optimização na reutilização das capacidades dos serviços.

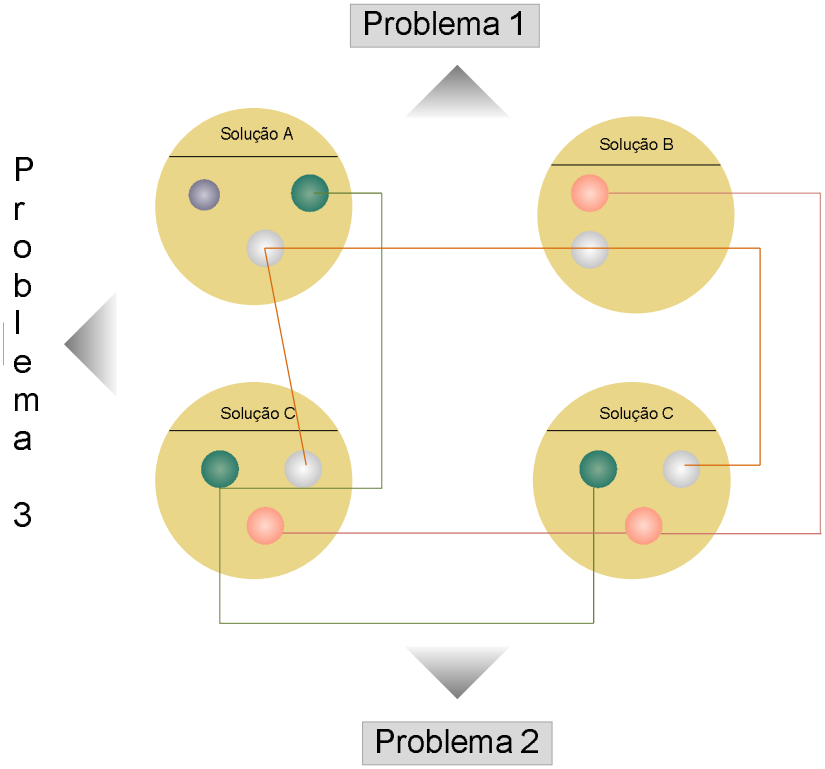


Figura 53 - Recombinação de diversas lógicas para resolver diversos problemas.

Service messaging: Para interagir com os serviços fornecidos não é necessário utilizar protocolos persistentes, pois as comunicações com os serviços são feitas por mensagens XML. Isto permite que o cliente tenha um nível de acoplamento muito baixo. O protocolo de comunicação existente é por mensagens XML, que podem englobar a execução de serviços compostos, permitindo assim ao cliente, efectuar uma ligação e usufruir do serviço pretendido sem ter uma ligação persistente.

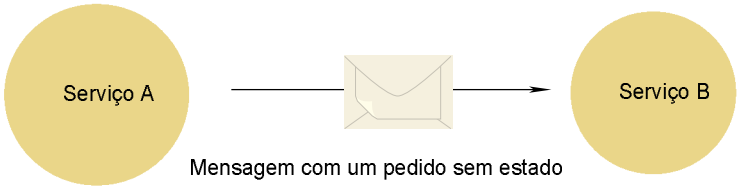


Figura 54 - Interacção entre serviços sem estado.

Messaging metadata: Como a troca de mensagens não tem uma ligação persistente, os serviços têm de ser projectados para conseguir ler os metadados provenientes nas mensagens. Para tal as mensagens que são trocadas, são em XML, onde toda a sua estrutura é armazenada em metadados. Assim pode ser feita uma validação da estrutura das mensagens para um determinado serviço.

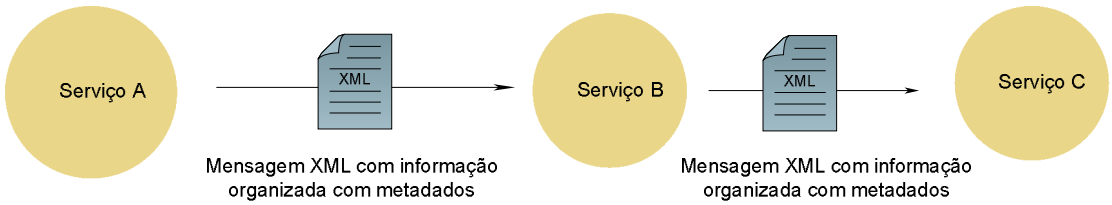
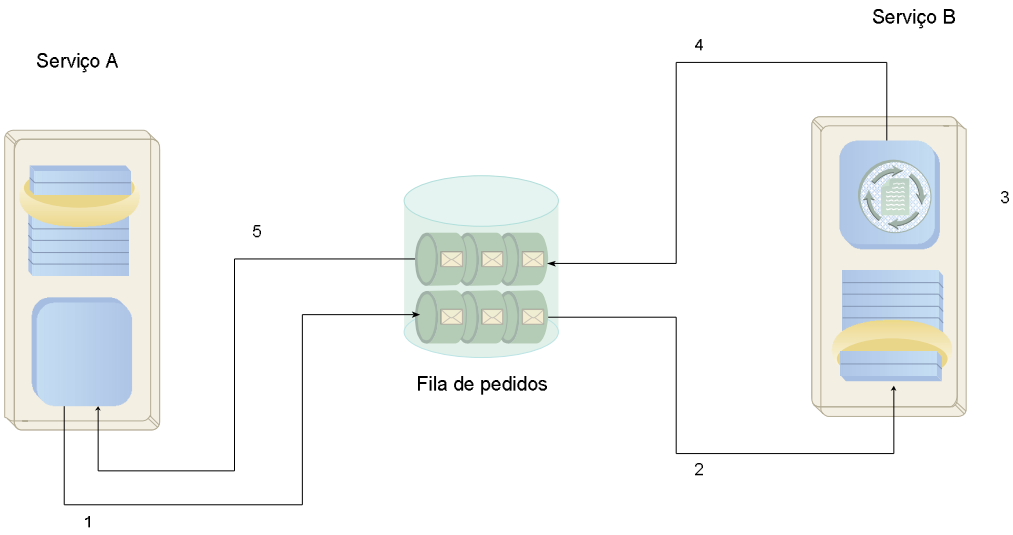


Figura 55 - Comunicação com mensagens que contêm dados e metadados com a organização da informação.

Asynchronous queuing: No conjunto de serviços prestados podem existir serviços cuja execução seja muito demorada. Enquanto a execução do serviço não tiver sido completada, o cliente fica bloqueado à espera de uma resposta proveniente do servidor. Para evitar estas situações em serviços cujo tempo de execução seja longo, o cliente envia o seu pedido para um *buffer* intermediário, após ter sido enviado o pedido, o cliente fica desconectado do serviço, não ficando assim bloqueado. Quando o serviço despachar o pedido envia a informação para o buffer, este trata de enviar a informação ao cliente que não ficou bloqueado à espera de resposta.



1 - Envio da mensagem para o gestor de mensagens sem ficar bloqueago à espera de resposta.
2 - Reencaminhamento da mensagem para o seu destino.
3 - Processamento da mensagem que pode demorar um grande período de tempo.
4 - Retorno da mensagem processada pelo serviço B para o gestor de mensagens.
5 - Recepção da mensagem após o seu processamento sem ter ficado bloqueado à espera de resposta.

Figura 56 - Envio de mensagens de forma assíncrona para não ficar com serviços bloqueados.

Reliable messaging: Os serviços disponibilizados estão acessíveis através de diversas formas de comunicação. A entrega de mensagens nem sempre pode ser garantida pelo sistema que presta serviços. Para garantir uma resposta ao cliente é necessário colocar um intermediário confiável, que consiga garantir a ambas as partes, cliente e serviço que a mensagem é entregue e processada. Assim quando se usam serviços de comunicação que não são confiáveis directamente para o sistema, é possível garantir que a comunicação é efectuada mesmo em ambientes que não são confiáveis.

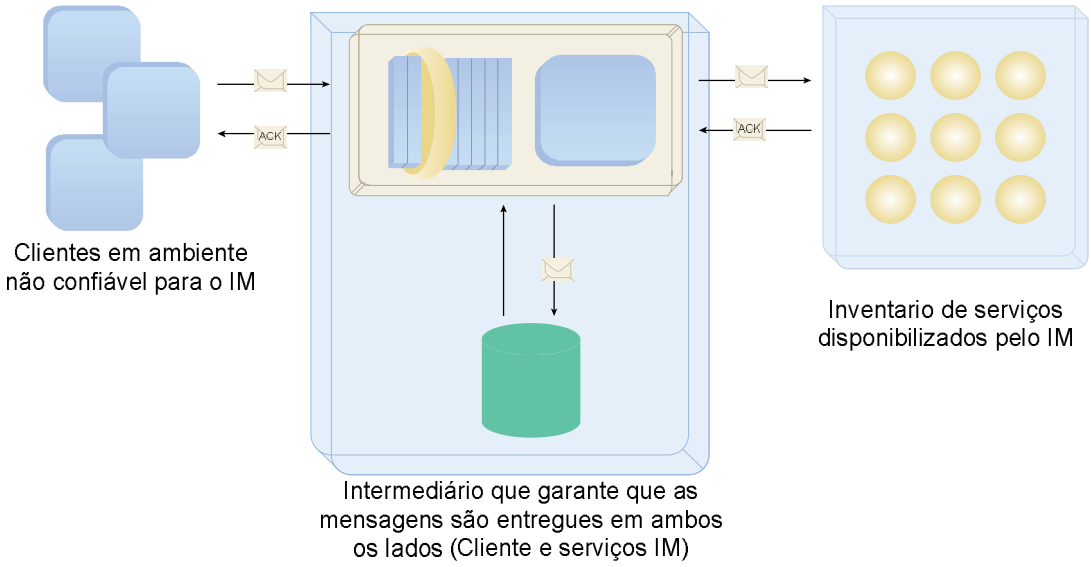


Figura 57 - Utilização de um intermediário para estabelecer uma ligação aos serviços disponibilizados pelo IM.

Event-driven messaging: Para serviços de longa duração e talvez decompostos em vários sub-serviços, o tempo de execução pode ser longo. Quando o cliente evoca um serviço com um grande período de execução, o cliente é notificado que o serviço pode ser demorado. Para que o cliente tenha a percepção da execução do serviço e qual o seu estado com o decorrer do tempo, o serviço envia notificações a indicar o seu estado. Assim é possível notificar os clientes dos vários passos importantes que decorreram com a execução do pedido pretendido. O cliente fica então com respostas parciais do seu pedido ao longo do tempo.

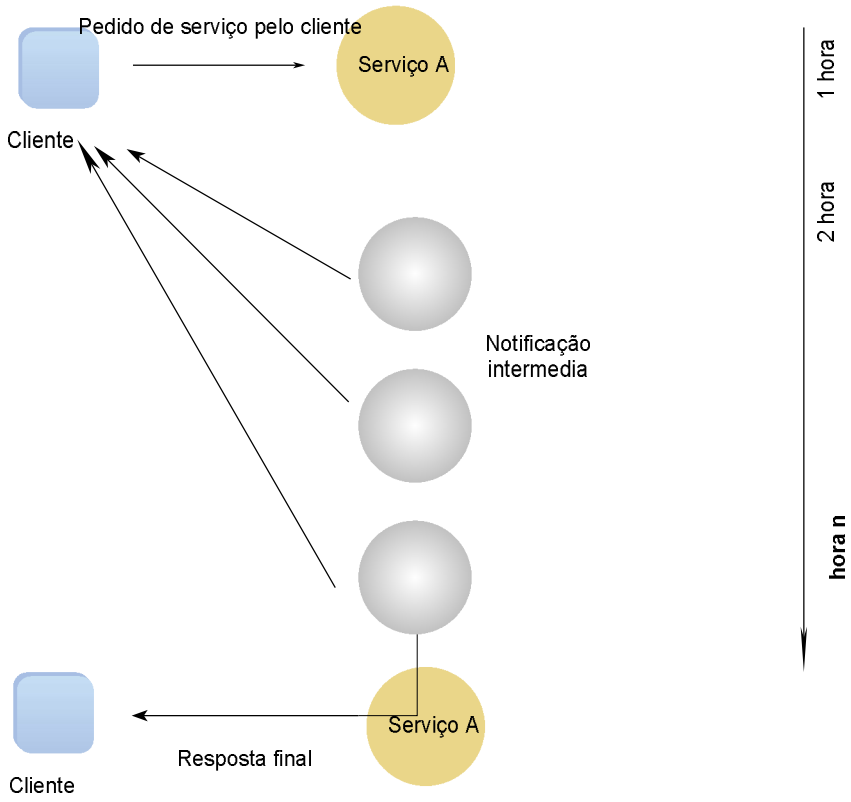


Figura 58 - Envio de respostas parciais para o cliente a notificar o andamento do serviço ao longo do tempo.

Agnostic sub-controller: Os serviços são construídos em duas camadas, estando toda a lógica do serviço na camada pai. Assim, os serviços não são passíveis de serem parcialmente reutilizáveis. O que se pretende é que os serviços sejam divididos em diversas camadas abstractas, dividindo assim o processamento a efectuar pelas várias camadas. Estas novas camadas representam novos serviços que podem ser reutilizados para solucionar outras lógicas. É assim possível utilizar lógicas parciais de um determinado serviço para outro propósito diferente sem que se tenha de reestruturar qualquer serviço.

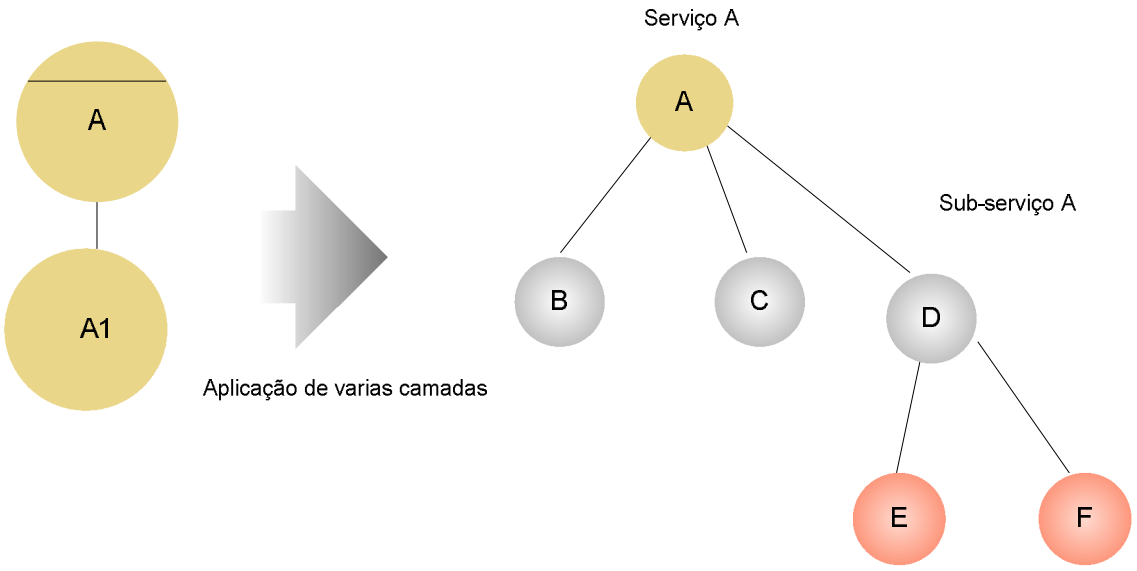


Figura 59 - Aplicação de várias camadas a um serviço possibilitando a reutilização de sub-camadas.

Composition autonomy: Os serviços fornecidos vão estar delegados em diversos servidores, sendo que existem serviços que podem necessitar de recursos de diversos servidores. Estes serviços que estão dependentes dos vários servidores não são autónomos e podem não conseguir responder em tempo útil. Para tornar estes serviços autónomos é necessário formar grupos independentes destes serviços replicados em pelo menos dois servidores. Assim sendo, os grupos de serviços são independentes e mais eficientes pois têm os recursos disponíveis no mesmo local, não necessitando de recursos externos. A aplicação deste padrão implica um cuidado extra aquando da actualização do serviço pois este encontra-se replicado e tem de se efectuar a actualização da função em diversos locais.

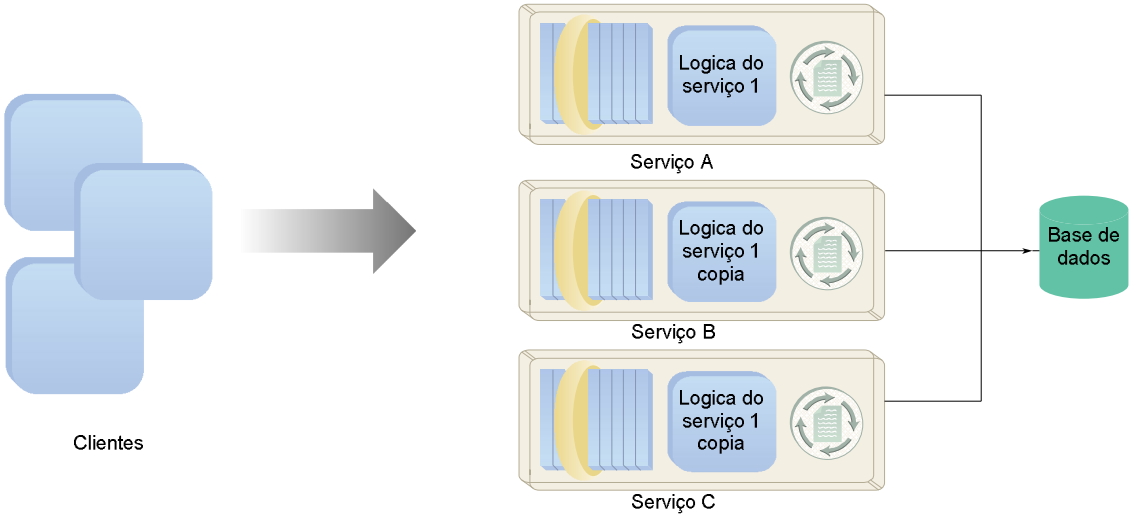


Figura 60 - Serviços que estão replicados garantindo assim uma independência de recursos.

Atomic service transaction: Quando um cliente necessita de executar diversos serviços de forma sequencial pode necessitar de definir pontos de transacção, de forma a garantir que os dados que ele alterou só surtem efeito após ter terminado uma determinada sequência de serviço. Caso a sequência que o cliente defina falhe por algum motivo é necessário garantir que as alterações que foram efectuadas sejam descartadas e que o estado original do sistema, onde foi marcado o ponto de transacção seja reposto. Como na sequência de operações podem ser evocados recursos de diversos sistemas, todos têm de receber uma notificação a confirmar que podem guardar ou que devem descartar. Estas transacções não se limitam a alterações na base de dados, mas sim a procedimentos efectuados. Exemplo: enviar um novo e-mail a pedir desculpa pelo envio anterior que não deve de ser tomado em atenção.

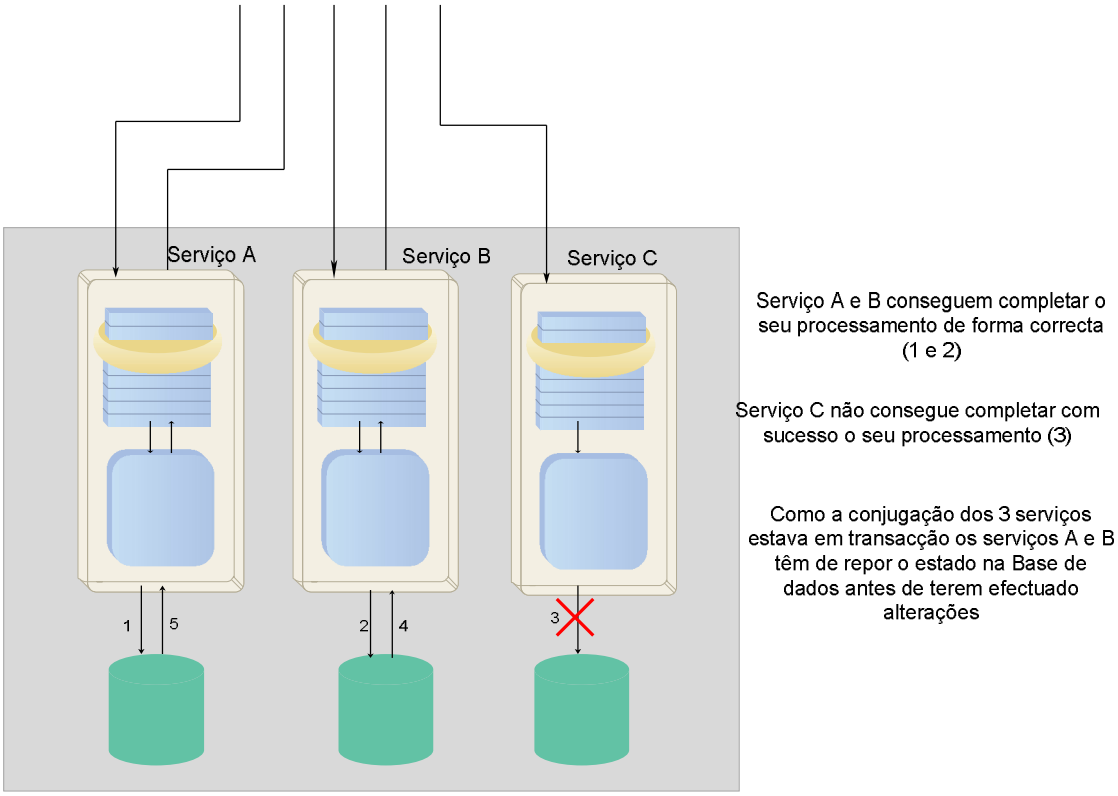


Figura 61 - Definição por parte do cliente dos serviços que pretende que estejam em transacção.

Compensating service transaction: Enquanto um serviço está a ser executado podem ocorrer falhas inesperadas. Essas falhas têm de ser tratadas com rotinas de compensação. Quando uma falha ocorre, é necessário saber se os recursos que foram alocados foram libertados. Se os recursos ficarem alocados é necessário correr uma rotina que os liberte, caso contrário estes ficam bloqueados indefinidamente. No caso de erro podem ser definidas um número de tentativas para voltar a executar o pedido por parte do cliente mas num outro servidor. É necessário definir um conjunto de acções a tomar de forma a não afectar o desempenho dos servidores e a libertar todos os recursos bloqueados. No caso de uma transacção tem de se conseguir fazer *rollback*.

Data origin authentication: De forma a garantir que uma mensagem vem de um determinado cliente e que não foi alterada em trânsito, é necessário ter mecanismos de verificação de mensagens. Uma forma de o fazer é de assinar digitalmente a mensagem, garantindo assim que esta não foi alterada. Caso seja alterada, a mensagem é rejeitada.

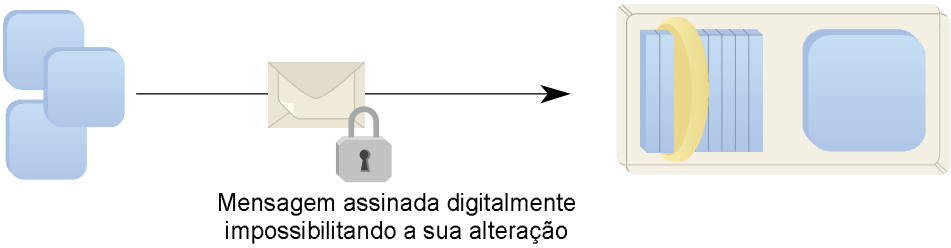


Figura 62 - Recepção de mensagens assinadas digitalmente garantindo que não foi alterada em trânsito.

Direct authentication: Como os serviços fornecidos podem por em causa dados sensíveis e que não devem de ser divulgados para todos os clientes é necessário ter um intermediário que consiga garantir que o cliente é confiável ou não. Quando o cliente envia as suas credenciais o serviço de autenticação consegue identificar se estas são validas ou se não são validas, garantindo assim o acesso restrito a informação.

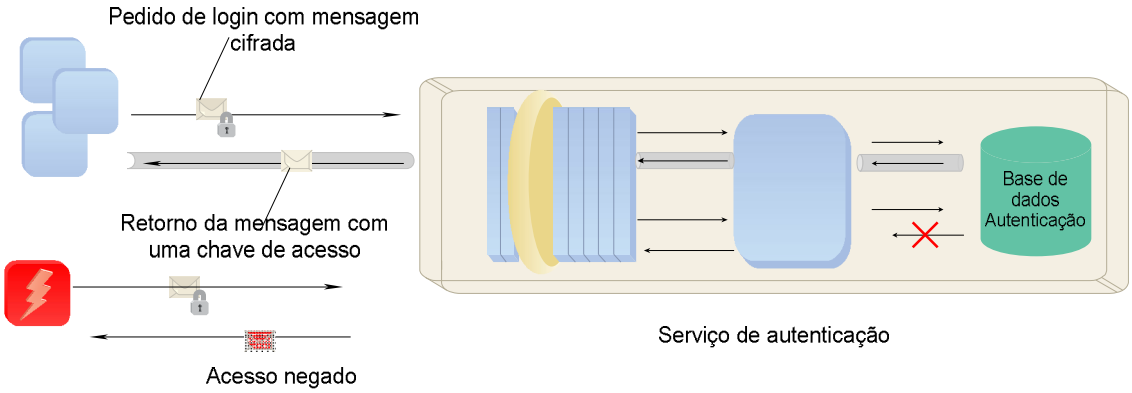


Figura 63 - Autenticação de clientes com mensagens assinadas.

Brokered authentication: Para ter acesso à plataforma, pode ser necessário ter uma entidade externa, pois o cliente pode não confiar no serviço e o serviço pode não confiar no cliente. Assim a entidade externa fornece ao cliente uma chave que permite acesso aos serviços disponibilizados e possibilita a comunicação entre o cliente e os serviços.

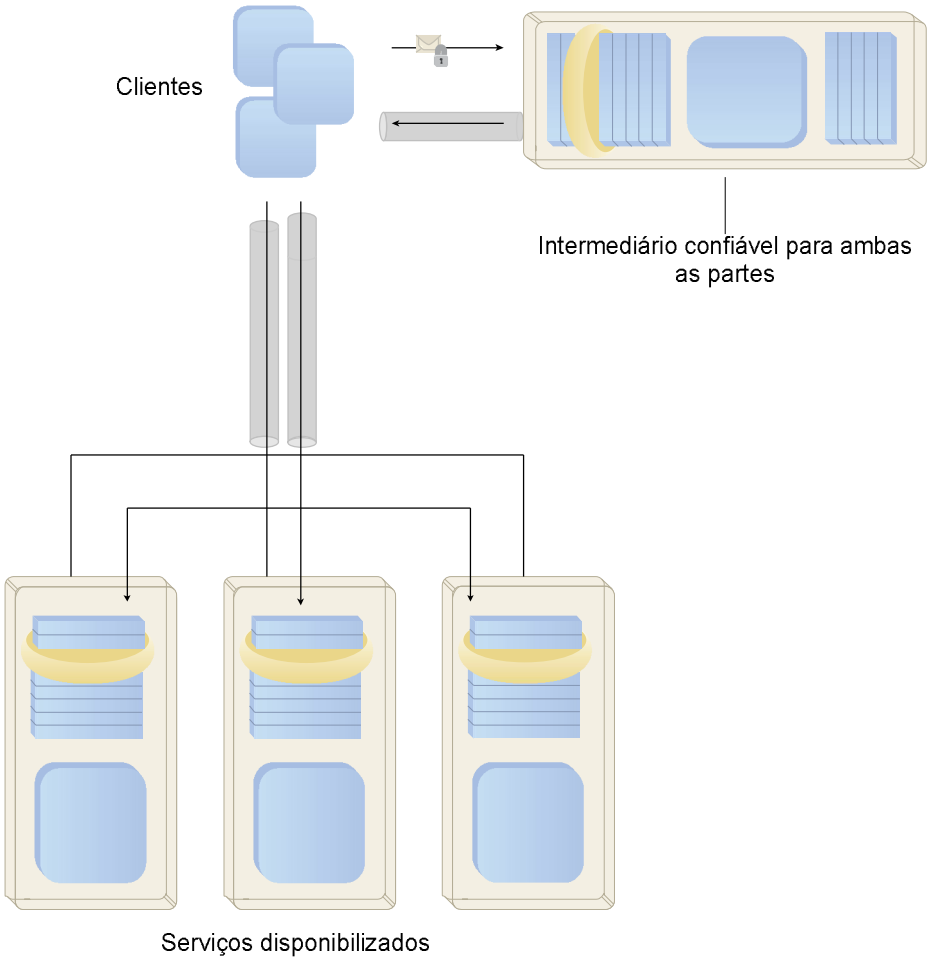


Figura 64 – Utilização de uma entidade externa para autenticar o cliente perante o serviço.

Data format transformation: Podem existir serviços que não consigam comunicar, pois o formato dos dados a trocar são incompatíveis. No entanto, é possível com uma transformação de formato de dados estabelecer comunicação. Para resolver este problema é adicionado um intermediário que vai proceder a transformação dos dados de forma dinâmica. Um exemplo onde este cenário é possível, é quando duas aplicações pretendem trocar dados de um paciente, no entanto uma delas necessita dos dados organizados numa ficha de paciente e a outra apenas tem dados de cliente, familiares, patologias ou seja tem os dados, mas não organizados. A função do intermediário é de agregar os dados da segunda aplicação e construir uma ficha paciente de forma a possibilitar a comunicação.

Data model transformation: Podem existir serviços que tenham a mesma informação, mas representam essa informação com esquemas diferente o que impossibilita a comunicação

destes serviços. Para solucionar este problema é utilizado uma tecnologia de transformação de esquemas possibilitando a comunicação. Um exemplo onde pode ocorrer esta incoerência de esquemas que representam o mesmo tipo de informação, é por exemplo um serviço clínico requisitar dados de um paciente no formato HL7 v2.x, mas o serviço que tem essa informação apenas comunica num protocolo próprio em XML. A transformação vai modificar o XML e mapeá-lo em HL7.

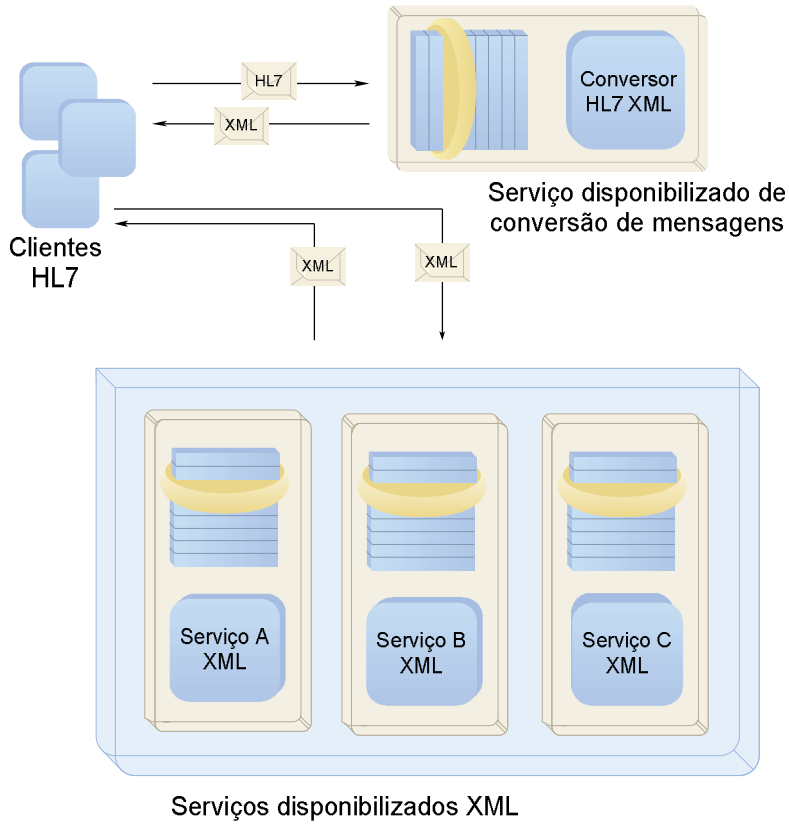


Figura 65 - Conversão de mensagens de forma a possibilitar a comunicação entre sistemas.

Protocol bridging: Serviços que usem protocolos diferentes ou de diferentes versões, encontram-se impedidos de comunicar e de trocar informação. Para possibilitar a comunicação é aplicada lógica de comunicação que tem por objectivo converter as mensagens dos protocolos dinamicamente, tornando a comunicação entre serviços possível. Um exemplo são dois serviços que comuniquem com versões diferentes do SOAP, v1.1 e v1.2. O que o

acontece é que ao invés de comunicarem directamente os serviços comunicam com um corrector de protocolos que traduz a mensagem para o formato correcto.

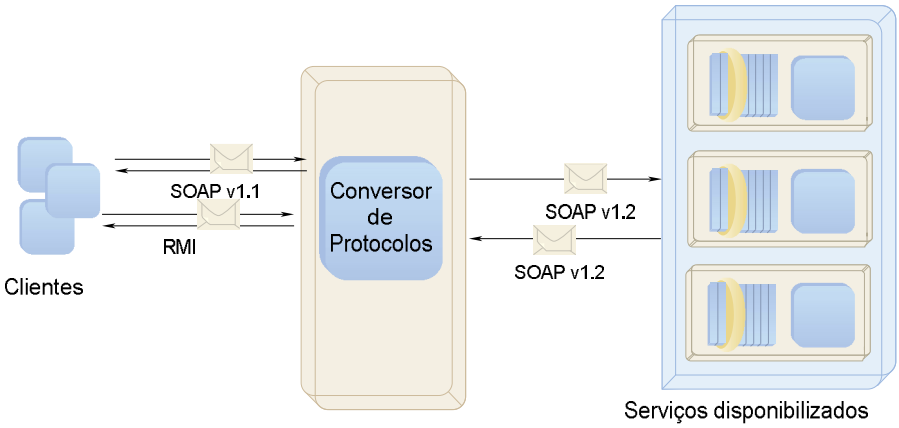


Figura 66- Conversão de protocolos durante a troca de mensagens entre serviços.

Enterprise service bus: A comunicação entre diferentes projectos é essencial. É necessário ter mecanismos que garantam a comunicação entre serviços de diferentes sistemas. Com a aplicação de um ESB é possível garantir a integração de diferentes sistemas. Com a aplicação deste padrão é adicionada uma camada intermédia que trata de problemas comuns relacionados com a confiabilidade, a escalabilidade e a disparidade das comunicações. O ESB tem ainda a funcionalidade de proxy de comunicação com o exterior. Assim o encaminhamento de mensagens desde o cliente até ao serviço tem a devida segurança aplicada.

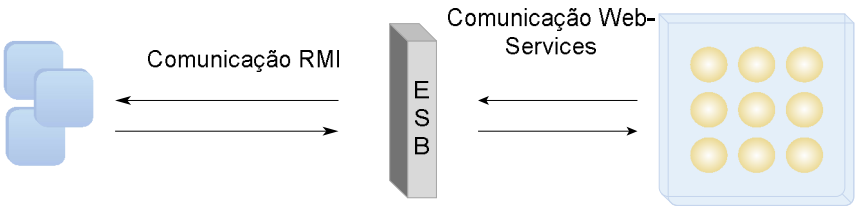


Figura 67 - Utilização de um ESB para efectuar a comunicação.

Service broker: Os serviços que são disponibilizados têm de ter uma transposição para o exterior de forma a poderem ser utilizados. Com a utilização do *Internet Information Services* (IIS), a transposição dos serviços é efectuada utilizando este padrão.

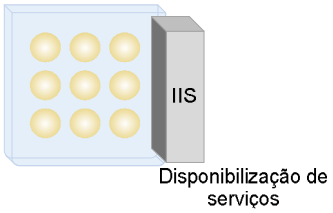


Figura 68 - Transposição dos serviços internos para o exterior.

Canonical schema Bus: Este padrão agrega as funcionalidades da comunicação entre diferentes sistemas com a necessidade de os sistemas terem as mensagens normalizadas. Assim alem de se garantir a comunicação entre sistemas diferentes, consegue-se ainda garantir que as mensagens quando são lidas pelo destinatário estão num formato correcto.

Para finalizar este anexo é apresentada uma tabela onde estão representados todos os padrões utilizados e que objectivos e benefícios do *service-oriented computing* é que eles contribuem.

	Aumento da interoperabilidade intrínseca	Aumento da Federação	Aumento da Diversificação das Vendas	Aumento de Negócios e das Tecnologias de Alinhamento	Aumento do Retorno do Investimento (ROI)	Maior Agilidade Organizacional	Redução de Encargos de TI
Centralização de serviços. (*Logic Centralization*)		1			1		1
Serviços Normalizados (*Service Normalization*)		1			1		1
Esquema canónico (*Canonical Schema*)	1	1		1		1	1
Abstracção de utilidade (*Utility Abstraction*)			1		1		1

Schema Centralization	1			1			1
Inventory Endpoint	1						1
Metadata Centralization			1		1		1
Service Encapsulation				1			1
Implementação redundante (*Redundant Implementation*)					1		1
Partial Validation	1						1
Triagem de mensagens (*Message Screening*)						1	1
Subsistema Confiável (*Trusted Subsystem*)	1						1
Service Perimeter Guard	1						1
Decoupled Contract	1	1					1
Contract Centralization	1	1					1
Validation Abstraction		1					1
Compatible Change						1	1
Version Identification	1						1
Service Refactoring			1			1	1

Service Decomposition	1	1					1
Proxy Capability	1						1
Distributed Capability	1						1
Capability Composition	1						1
Capability Recomposition	1			1	1	1	1
Service Messaging	1		1				1
Messaging Metadata	1						1
Asynchronous Queuing			1				1
Reliable Messaging			1				1
Event-Driven Messaging			1				1
Agnostic Sub-Controller					1		1
Composition Autonomy	1						1
Atomic Service Transaction			1				1
Compensating Service Transaction			1				1
Data Origin Authentication			1				1
Direct Authentication			1				1
rokered Authentication			1				1

Data Format Transformation			1	1			1
Data Model Transformation			1				1
Protocol Bridging			1				1
Enterprise Service Bus							
Service Broker	1		1				1
Canonical Schema Bus							

Anexos III – Protocolo de comunicação

Protocolo de comunicação do gestor de Informação

O protocolo de comunicação utilizado para comunicar com o GI é composto por mensagens XML. A finalidade e estrutura de cada mensagem são explicadas nesta secção.

Altera estado de sensores

```
<pedido>
        <user>aaa</user>
        <password>4565</password>
        <operacao>C</operacao>
        <noSensorial>
                <sensor>
                        <nomeTabela>tabelaTeste_aaa</nomeTabela>
                        <estado>2</estado>
                        <dataAltEstado>
                                <minutos>34</minutos>
                                <hora>14</hora>
                                <dia>24</dia>
                                <mes>2</mes>
                                <ano>2011</ano>
                        </dataAltEstado>
                </sensor>
                <sensor>
                        <nomeTabela>tabelaTeste3_aaa</nomeTabela>
                        <estado>2</estado>
                        <dataAltEstado>
                                <minutos>34</minutos>
                                <hora>14</hora>
                                <dia>24</dia>
                                <mes>2</mes>
                                <ano>2011</ano>
```

```
                </dataAltEstado>
            </sensor>
        </noSensorial>
</pedido>
```

Este XML serve para alterar o estado de um ou mais sensores de um nó sensorial. O estado de um sensor permite saber o estado em que se encontra o mesmo.

Recebe os dados do utilizador, o nome da tabela do sensor, o novo estado do sensor e a data da alteração.

Alterar tabela

```
<pedido>
        <user>aaa</user>
        <password>4565</password>
        <operacao>U</operacao>
        <tabela>
            <nomeTabela>tabelaTeste</nomeTabela>
            <dados>
                <campo>inercia</campo>
                <tipoCampo>int</tipoCampo>
                <definicao>
                    <tamanho>3</tamanho>
                    <valorDefeito>5</valorDefeito>
                    <indexado>false</indexado>
                    <unico>false</unico>
                    <null>false</null>
                    <chavePrimaria>false</chavePrimaria>
                </definicao>
            </dados>
            <dados>
                <campo>nome</campo>
                <tipoCampo>int</tipoCampo>
                <definicao>
                    <tamanho>50</tamanho>
                    <valorDefeito>vazio</valorDefeito>
                    <indexado>false</indexado>
                    <unico>false</unico>
                    <null>false</null>
                    <chavePrimaria>false</chavePrimaria>
                </definicao>
            </dados>
        </tabela>
</pedido>
```

Este XML permite a alteração da estrutura de uma ou mais tabelas da base de dados.

Recebe os dados do utilizador, o nome da tabela a alterar e os dados para alterar.

Apagar tabela

```
<pedido>
	<user>aaa</user>
	<password>4565\</password>
	<operacao>D</operacao>
	<tabelaApagar>
		<nomeTabelaApagar>TabelaTeste3</nomeTabelaApagar>
	</tabelaApagar>
	<tabelaApagar>
		<nomeTabelaApagar>TabelaTeste</nomeTabelaApagar>
	</tabelaApagar>
</pedido>
```

Este XML permite a um utilizador apagar uma das tabelas presentes na base de dados por si criadas.

Recebe os dados do utilizador, e o nome das tabelas a apagar.

Cria nó sensorial

```
<Pedido>
	<user>aaa</user>
	<password>4565</password>
	<operacao>C</operacao>
	<noSensorial>
		<modelo>modelo</modelo>
		<numSerie>serieNoSensorial</numSerie>
		<marca>marca</marca>
		<userIdoso>JoséCarpinteiro09</userIdoso>
	</noSensorial>
</Pedido>
```

Este XML permite criar um novo nó sensorial.

Recebe os dados do utilizador, e os dados do nó sensorial.

Esta mensagem permite cria a estrutura completa de um nó sensorial, incluindo as tabelas necessárias para armazenar os dados dos sensores.

```
<pedido>
	<user>bbb</user>
	<password>4565</password>
```

```
<operacao>C</operacao>
<noSensorial>
        <sensor>
                <nomeTabela>tabelaTeste323</nomeTabela>

                <modelo>modeloSensor</modelo>
                <numSerie>serieSensor</numSerie>
                <marca>marcaSensor</marca>
                <dados>
                        <campo>aceleracao</campo>
                        <tipoCampo>numero</tipoCampo>
                        <definicao>
                                <tamanho>50</tamanho>
                                <valorDefeito>0</valorDefeito>
                                <indexado>True</indexado>
                                <unico>True</unico>
                                <null>True</null>
                                <chavePrimaria>False</chavePrimaria>
                        </definicao>
                </dados>
                <dados>
                        <campo>velocidade</campo>
                        <tipoCampo>numero</tipoCampo>
                        <definicao>
                                <tamanho>50</tamanho>
                                <valorDefeito>0</valorDefeito>
                                <indexado>True</indexado>
                                <unico>True</unico>
                                <null>True</null>
                                <chavePrimaria>False</chavePrimaria>

                        </definicao>
                </dados>
        </sensor>
        <sensor>
                <nomeTabela>tabelaTeste346</nomeTabela>

                <modelo>modeloSensor3</modelo>
                <numSerie>serieSensor3</numSerie>
                <marca>marcaSensor3</marca>
                <dados>
                        <campo>aceleracao3</campo>
                        <tipoCampo>numero3</tipoCampo>
                        <definicao>
                                <tamanho>50</tamanho>
                                <valorDefeito>0</valorDefeito>
                                <indexado>True</indexado>
                                <unico>True</unico>
                                <null>True</null>
```

```
                    <chavePrimaria>False</chavePrimaria>

                </definicao>
            </dados>
            <dados>
                <campo>velocidade3</campo>
                <tipoCampo>numero3</tipoCampo>
                <definicao>
                    <tamanho>50</tamanho>
                    <valorDefeito>0</valorDefeito>
                    <indexado>True</indexado>
                    <unico>True</unico>
                    <null>True</null>
                    <chavePrimaria>False</chavePrimaria>
                </definicao>
            </dados>
        </sensor>
        <modelo>modelo</modelo>
        <numSerie>serieNoSensorial</numSerie>
        <marca>marca</marca>
        <userIdoso>JoséCarpinteiro09</userIdoso>
    </noSensorial>
</pedido>
```

Este XML permite a criação de um nó sensorial e dos sensores a ele ligados e as tabelas para armazenar os dados dos sensores.

Recebe os dados do utilizador, os dados do nó sensorial e os dados para formar a estrutura das tabelas dos sensores.

Criar tabela de sensores

```
<Pedido>
    <user>bbb</user>
    <password>4565</password>
    <operacao>C</operacao>
    <noSensorial>
        <sensor>
            <nomeTabela>tabelaTeste</nomeTabela>

            <modelo>modeloSensor</modelo>
            <numSerie>serieSensor</numSerie>
            <marca>marcaSensor</marca>
            <dados>
                <campo>aceleracao</campo>
                <tipoCampo>numero</tipoCampo>
                <definicao>
                    <tamanho>50</tamanho>
```

```
                        <valorDefeito>0</valorDefeito>
                        <indexado>True</indexado>
                        <unico>True</unico>
                        <null>True</null>
                        <chavePrimaria>False</chavePrimaria>
                </definicao>
        </dados>
        <dados>
                <campo>velocidade</campo>
                <tipoCampo>numero</tipoCampo>
                <definicao>
                        <tamanho>50</tamanho>
                        <valorDefeito>0</valorDefeito>
                        <indexado>True</indexado>
                        <unico>True</unico>
                        <null>True</null>
                        <chavePrimaria>False</chavePrimaria>

                </definicao>
        </dados>
</sensor>
<sensor>
        <nomeTabela>tabelaTeste3</nomeTabela>

        <modelo>modeloSensor3</modelo>
        <numSerie>serieSensor3</numSerie>
        <marca>marcaSensor3</marca>
        <dados>
                <campo>aceleracao3</campo>
                <tipoCampo>numero3</tipoCampo>
                <definicao>
                        <tamanho>50</tamanho>
                        <valorDefeito>0</valorDefeito>
                        <indexado>True</indexado>
                        <unico>True</unico>
                        <null>True</null>
                        <chavePrimaria>False</chavePrimaria>

                </definicao>
        </dados>
        <dados>
                <campo>velocidade3</campo>
                <tipoCampo>numero3</tipoCampo>
                <definicao>
                        <tamanho>50</tamanho>
                        <valorDefeito>0</valorDefeito>
                        <indexado>True</indexado>
                        <unico>True</unico>
                        <null>True</null>
                        <chavePrimaria>False</chavePrimaria>
```

```
                    </definicao>
                </dados>
            </sensor>
        </noSensorial>
</Pedido>
```

Este XML permite criar a estrutura da tabela para armazenar um tipo de sensor.

Recebe os dados do utilizador, e os dados para formar a estrutura das tabelas dos sensores.

Resposta de erro

```
<resposta>
        <codigoErro></codigoErro>
        <descricaoErro></descricaoErro>
</resposta>
```

Este XML é devolvido quando o utilizador efectua uma operação ilegal ou quando ocorre um erro na aplicação.

Devolve o código do erro e uma descrição do erro.

Cria tabela dinâmica

```
<pedido>
        <user>aaa</user>
        <password>4565</password>
        <operacao>C</operacao>
        <tabela>
            <nomeTabela>TabelaTesteDin4</nomeTabela>
            <dados>
                <campo>giroscopio</campo>
                <tipoCampo>string</tipoCampo>
                <definicao>
                    <tamanho>50</tamanho>
                    <valorDefeito>0</valorDefeito>
                    <indexado>true</indexado>
                    <unico>true</unico>
                    <null>false</null>
                    <chavePrimaria>true</chavePrimaria>
                </definicao>
                <chaveEstrangeira>
                    <tabelaEstrangeira>TabelaTesteDin1</tabelaEstrangeira>
                    <campoEstrangeira>id</campoEstrangeira>
                </chaveEstrangeira>

            </dados>
```

```
        </tabela>
</pedido>
```

Este XML permite a criação de tabelas dinâmicas na base de dados.

Recebe os dados do utilizador e os dados para criar a estrutura da tabela.

Inserir dados na tabela

```
<pedido>
        <user>aaa</user>
        <password>4565</password>
        <operacao>C</operacao>
        <dadosSensor>
                <nomeTabela>tabelaTeste</nomeTabela>
                <linha>
                        <dados>
                                <campo>aceleracao</campo>
                                <valor>3</valor>
                        </dados>
                        <dados>
                                <campo>velocidade</campo>
                                <valor>4</valor>
                        </dados>
                        <dados>
                                <campo>inercia</campo>
                                <valor>7</valor>
                        </dados>
                </linha>
                <linha>
                        <dados>
                                <campo>aceleracao</campo>
                                <valor>6</valor>
                        </dados>
                        <dados>
                                <campo>velocidade</campo>
                                <valor>8</valor>
                        </dados>
                        <dados>
                                <campo>inercia</campo>
                                <valor>7</valor>
                        </dados>
                </linha>
                <linha>
                        <dados>
                                <campo>aceleracao</campo>
                                <valor>12</valor>
                        </dados>
```

```
                    <dados>
                        <campo>velocidade</campo>
                        <valor>16</valor>
                    </dados>
                    <dados>
                        <campo>inercia</campo>
                        <valor>7</valor>
                    </dados>
                </linha>
            </dadosSensor>
                <dadosSensor>
                <nomeTabela>tabelaTeste3</nomeTabela>
                <linha>
                    <dados>
                        <campo>aceleracao3</campo>
                        <valor>3</valor>
                    </dados>
                    <dados>
                        <campo>velocidade3</campo>
                        <valor>4</valor>
                    </dados>
                </linha>
                <linha>
                    <dados>
                        <campo>aceleracao3</campo>
                        <valor>6</valor>
                    </dados>
                    <dados>
                        <campo>velocidade3</campo>
                        <valor>8</valor>
                    </dados>
                </linha>
                <linha>
                    <dados>
                        <campo>aceleracao3</campo>
                        <valor>12</valor>
                    </dados>
                    <dados>
                        <campo>velocidade3</campo>
                        <valor>16</valor>
                    </dados>
                </linha>
            </dadosSensor>
</pedido>
```

Este XML permite a inserção de dados numa tabela previamente criada.

Recebe os dados do utilizador, o nome da tabela a inserir os dados e os dados a inserir na tabela.

Ligar um nó sensorial a um idoso

```
<pedido>
	<user></user>
	<password></password>
	<operacao></operacao>
	<noSensorial>
		<userIdoso></userIdoso>
		<dataInicio>
			<minutos></minutos>
			<hora></hora>
			<dia></dia>
			<mes></mes>
			<ano></ano>
		</dataInicio>
	</noSensorial>
</pedido>
```

Este XML permite associar um nó sensorial a um idoso.

Recebe os dados do utilizador, o *username* do idoso e a data em que a associação foi criada.

Pedir estrutura do nó sensorial

```
<pedido>
	<user>aaa</user>
	<password>4565</password>
	<operacao>C</operacao>
	<estrutura>
		<noSensorial>aaa</noSensorial>
	</estrutura>
</pedido>
```

Este XML permite a um utilizador pedir a estrutura de um nó sensorial dos quais tem permissão.

Recebe os dados do utilizador e o nó sensorial que se deseja consultar.

Pedir informação

```
<pedido>
	<user>aaa</user>
	<password>4565</password>
	<operacao>C</operacao>
	<dadosPedidos>
		<nomeTabela>tabelaTeste</nomeTabela>
		<campos>
```

```
                <coluna>aceleracao</coluna>
                <coluna>velocidade</coluna>
            </campos>
            <condicoes>
                <condicao>
                    <operador>and</operador>
                    <parametro>
                        <coluna>aceleracao</coluna>
                        <operacao>=</operacao>
                        <valor>3</valor>
                    </parametro>
                </condicao>
            </condicoes>
        </dadosPedidos>
</pedido>
```

Este XML permite a um utilizador fazer consultas na base de dados.

Recebe os dados do utilizador, os campos a consultar, o nome da tabela a consultar e condições extra para a consulta.

Pedir XSD

```
<pedido>
        <user></user>
        <password></password>
        <operacao></operacao>
        <nomeServico></nomeServico>
</pedido>
```

Este XML permite a um utilizador pedir um dos XSDs usados pelos serviços para validar os XML.

Recebe os dados do utilizador e o nome do serviço que se deseja verificar.

Protocolo de comunicação do gestor de alertas

Nesta secção são apresentadas as mensagens XML que compõem o protocolo de comunicação que permitem enviar eventos e alertas.

Envio de um evento para o GA

```
<login>
        <Login>aaa<\Login >
        <password>X%tg0rg<\password>
<\login>
<evento>
        <tipo><\ tipo> // O tipo de evento é o que aconteceu ( queda, quebra tenção)
        <prioridadeEmissor><\ prioridadeEmissor>
<\evento>
```

Resposta

Sem permissões

```
<Resposta>
        <nomeServidor>CIICC <\ nomeServidor >
        <nonce><\nonce>
        <Efectuado>N<\ Efectuado >
        <Permissoes> CRUD<\ Permissoes >
<\Resposta>
```

Acusar recepção mensagem

```
<Resposta>
        <nomeServidor>CIICC <\ nomeServidor >
        <nonce><\nonce>
        <Efectuado>S<\ Efectuado >
<\Resposta>
```

O servidor faz login no módulo de classificação com o seu utilizador e não com as credenciais do cliente.

O login Utilizador tem de ir para o classificador para este saber qual a proveniência do evento.

Antes de enviar para o classificador o evento é guardado na base de dados.

Envio do evento para o classificador

```
<login>
        <loginUtilizador><loginUtilizador>
        <Login>aaa<\Login >
        <password>X%tg0rg<\password>
<\login>
<evento>
        <idEvento><\idEvento>
        <tipo><\tipo>
```

```
        <prioridadeEmissor><\ prioridadeEmissor>
<\evento>
```

Acusar recepção mensagem pelo classificador

```
<Resposta>
        <nomeClassificador>CIICC <\nomeClassificador>
        <nonce><\nonce>
        <Efectuado>S<\ Efectuado >
<\Resposta>
```

Resposta do classificador ao evento anterior

O classificador autentica-se com o seu login

```
<login>
        <Login>aaa<\Login >
        <password>X%tg0rg<\password>
<\login>
<evento>
        <idEvento><\idEvento>
        <tipo><\tipo>
        <classificação><\classificação>
<\evento>
```

De tempo a tempo é enviado um sinal do centro de controlo aos módulos que se encontram conectados, unicamente para saber se estão ligados. Caso não se consiga contactar com eles é gerado um alerta automático com prioridade baixa.

Vai existir uma aplicação no terminal móvel do idoso a solicitar uma acção de confirmação. Esta confirmação indica que o idoso não está inconsciente ou com algum problema.

Essa resposta vai ser incluída no contacto da lista de notificações caso o cliente consiga responder em tempo útil.

O dispositivo móvel recebe uma mensagem que lhe dá a indicação de inicio de alarme, onde recebe o Id do evento que o fez disparar e o tempo de notificação.

Caso existam vários eventos simultâneos, a notificação vai conter a informação dos vários eventos, tendo prioridade os eventos com prioridade mais elevada.

Caso o cliente não responda ao dispositivo num determinado tempo, o evento disparado pelo dispositivo móvel é ignorado.

```
<login>
        <Login>aaa<\Login >
        <password>X%tg0rg<\password>
<\login>
<avisoTerminalMovel>
        <idEvento><\idEvento>
        <tipo><\tipo>
        < tempoAviso ><\tempoAviso>
        < mensagem ><\mensagem>
<\ avisoTerminalMovel >
```

Acusar recepção mensagem pelo dispositivo móvel

```
<respostaTerminalMovel>
        <nomeDispositivo>CIICC <\ nomeDispositivo >
        <nonce><\nonce>
        <Efectuado>S<\ Efectuado >
<\respostaTerminalMovel>
```

Envio de um alerta por um módulo externo

```
<Alerta>
        <user> </user>
        <password> </password>
        <operacao> </operacao>
        <dadosAlerta>
                <evento>  </evento>
                <prioridade>4</prioridade>
                <usernameIdoso> </usernameIdoso>
        </dadosAlerta>
</Alerta>
```

Esta mensagem permite o envio de um alerta por parte de módulo externo.

Anexos IV – Manual de utilização da ontologia

A ontologia do Elde Care, está organizada de forma a ser possível consultar e adicionar informação, sendo esta ontologia evolutiva.

Um projecto é adicionado na ontologia como um Sistema, Nó Sensorial ou um Sensor. Os projectos estão ligados a um módulo ou a um sub-módulo que pode agregar um conjunto de projectos. Existem diferentes tipos de projectos, os de recolha de dados, os de consulta de dados e os que apenas fornecem serviços.

Os projectos de recolha de dados devem indicar a informação que recolhem e que pretendem partilhar, contribuindo para uma melhor avaliação do idoso. A informação que é recolhida fica então disponível para consulta, estando o conhecimento e organização disponibilizadas na ontologia.

Os projectos de consulta de dados devem indicar que tipo de dados necessitam e o que vão identificar com esses dados.

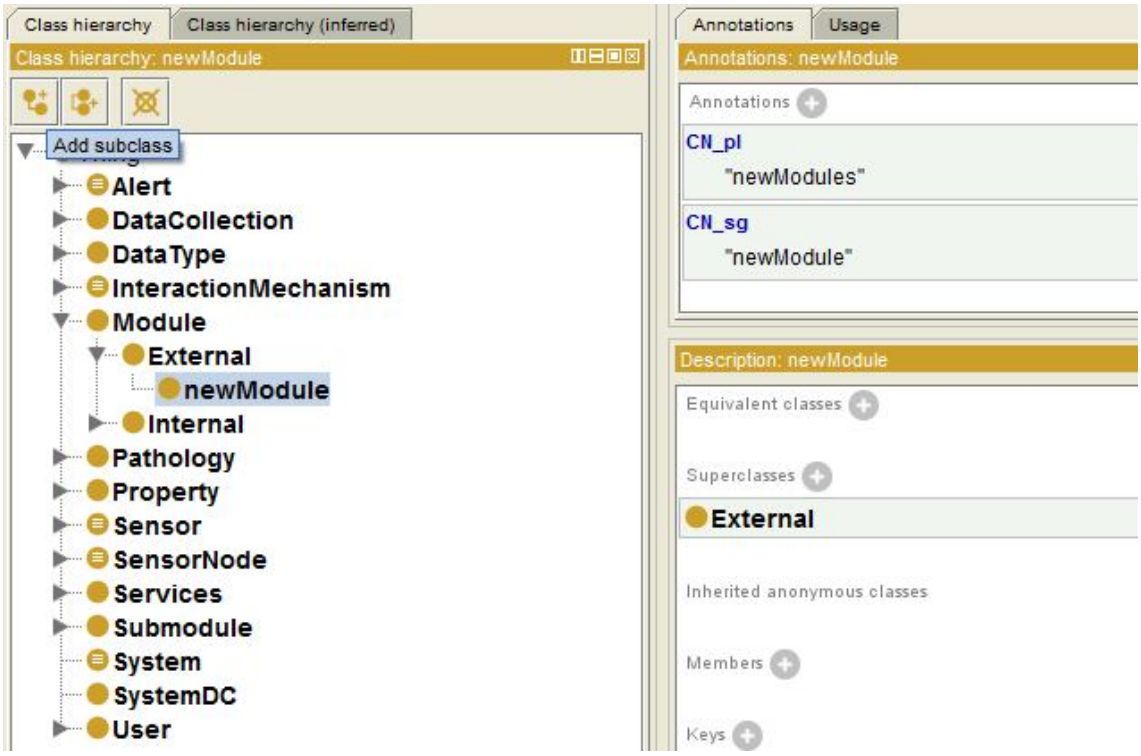
Os projectos que fornecem serviços devem indicar quais os seus mecanismos de interacção, quais os serviços que fornecem, indicar o que cada serviço realiza e como os evocar.

Como adicionar Module e subModule

O Elde Care tem uma arquitectura modular. Esta arquitectura permite a ligação de novos módulos. De forma a facilitar a organização da ontologia, os *Modules* podem conter vários

SubModules. Na classe dos *Modules* existem duas subclasses, a *Internal* e a *External*. Na classe Internal vão estar contidos todos os módulos que fazem parte do projecto Elde Care. A classe External vai conter todos os outros módulos que pretendam estar ligados ao Projecto Elde Care. Na classe SubModule devem de estar todos os sub-módulos que fazem parte de um módulo previamente criado.

Para criar um novo Module tem de se identificar se é um módulo interno ou externo. Para adicionar um módulo Externo tem de se entrar na classe Module -> External e adicionar uma nova subclasse com o nome do módulo a adicionar.

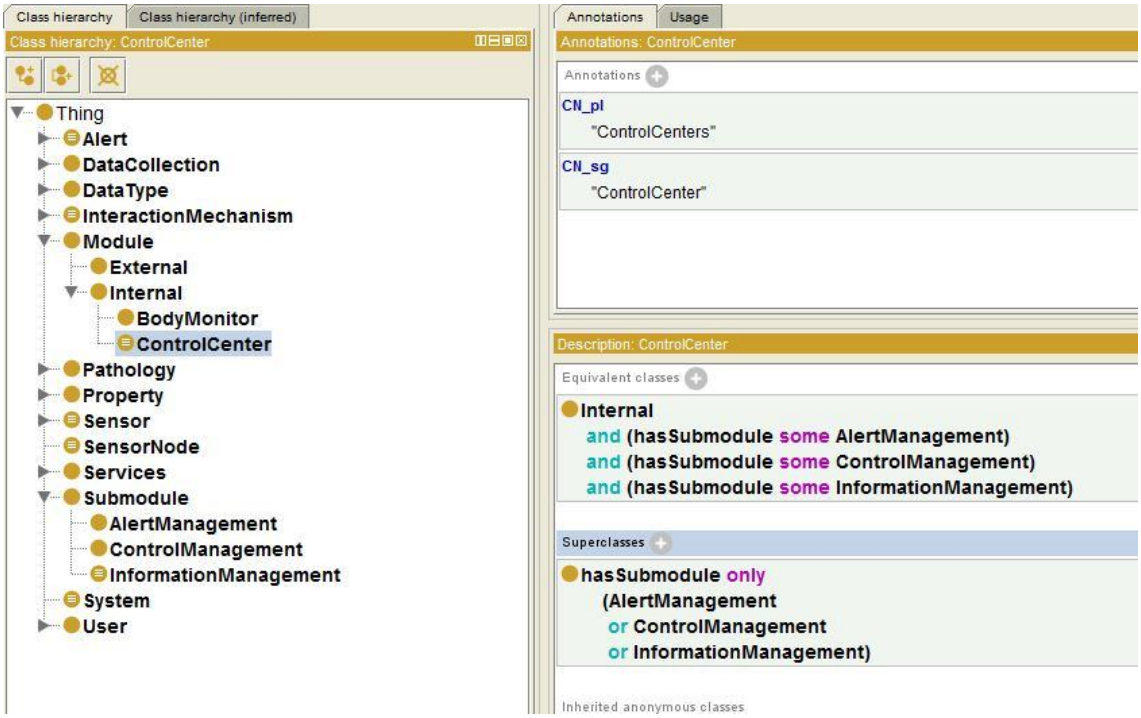


Para adicionar sub-módulos tem de se seleccionar a classe SubModule e adicionar uma nova subclasse com o nome do sub-módulo correspondente.

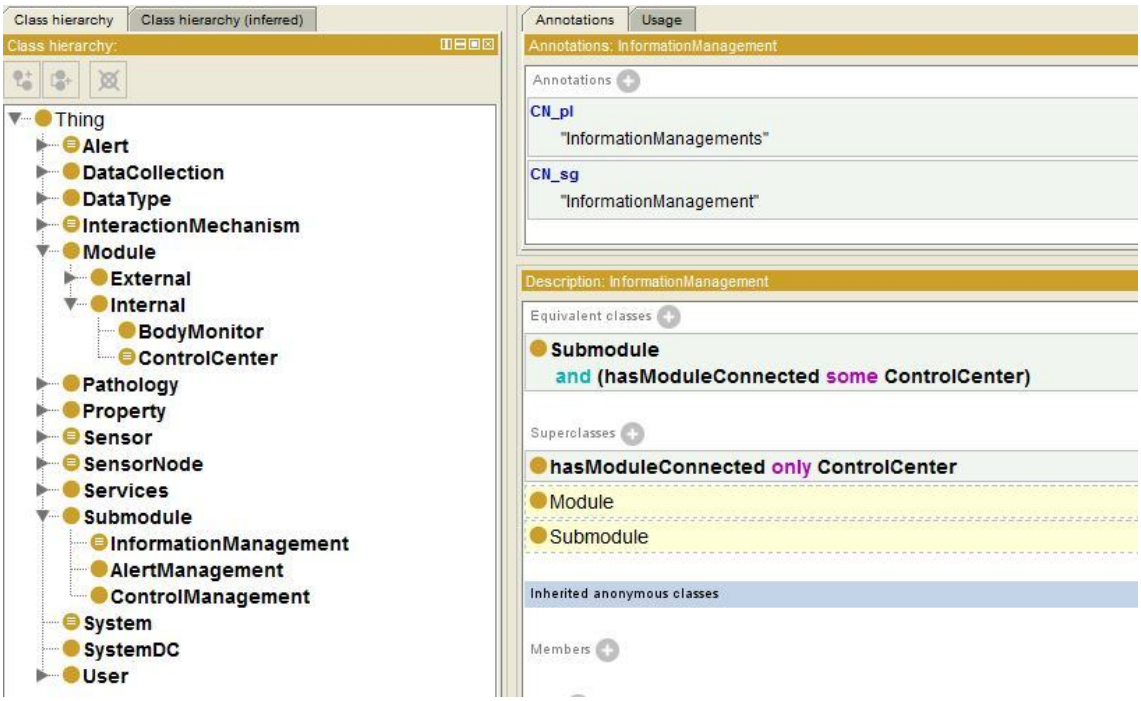
Após ter sido criado todos os módulos e sub-módulos na ontologia, é necessário ligar os sub-módulos aos módulos correspondentes. Para fazer essa ligação existem as propriedades hasSubModule e hasModuleConnected. A propriedade hasSubModule indica que sub-

módulos é que estão ligados a um módulo. A propriedade hasModuleConnected indica que sub-módulos é que estão ligados a um módulo.

Para fazer a ligação no módulo tem de se selecionar o módulo na ontologia e adicionar a propriedade *hasSubModule some* e o nome do respectivo *sub-módulo.* Adicionar todos os sub-módulos que pertencem ao módulo criado. Após ter adicionado todas as propriedades, seleccionar as propriedades e converter para uma defined Class(Ctrl+D), em seguida selecionar as propriedades criadas no separador Equivalent classes e criar um closure axiom (seleccionar, botão direito do rato e Create closure axiom).



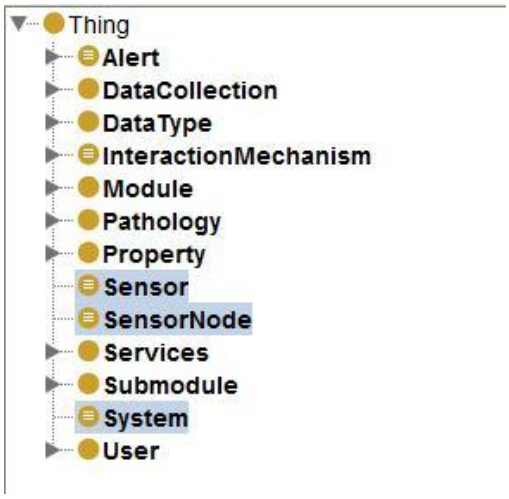
Para que os sub-módulos estejam ligados ao módulo tem de se adicionar aos submoódulos criados a propriedade hasModuleConnected para indicar a que modulo é que o sub-módulo correspondente pertence. Para fazer esta ligação, tem de se adicionar a propriedade na superclasse *hasModuleConnected some* e escolher qual o módulo a que pertence.



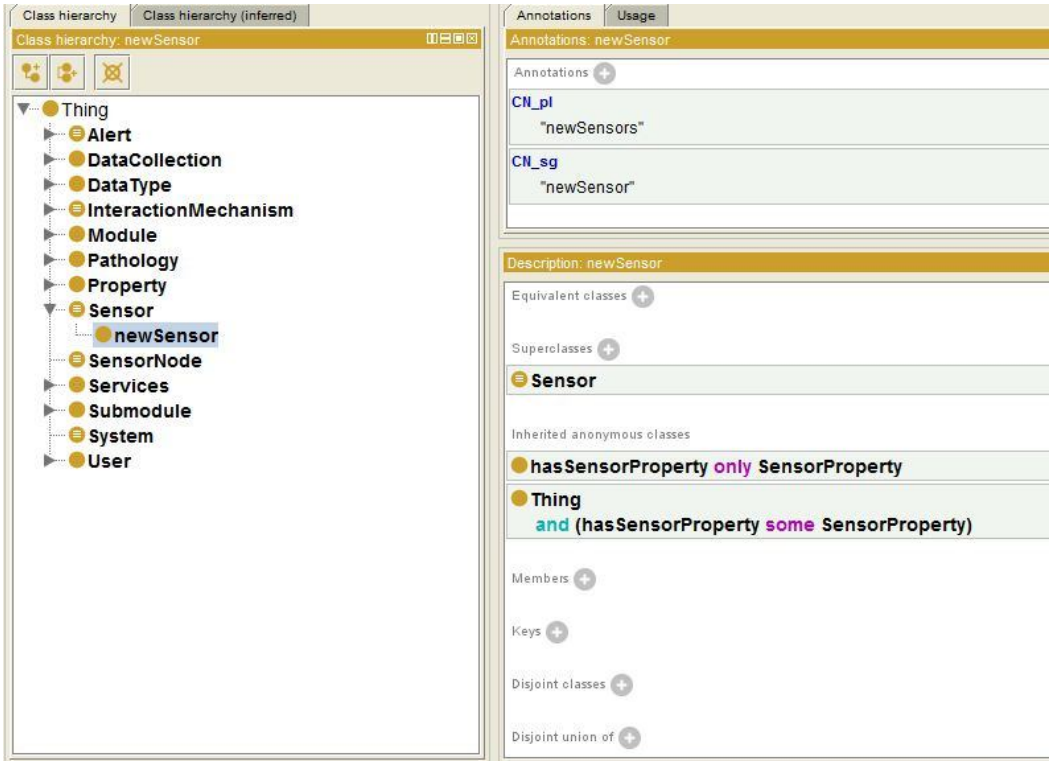
Desta forma a ligação dos *modules* com os *subModules* fica efectuada, ficando registada a informação dos sub-módulos, que pertencem ao módulo.

Como adicionar um novo projecto

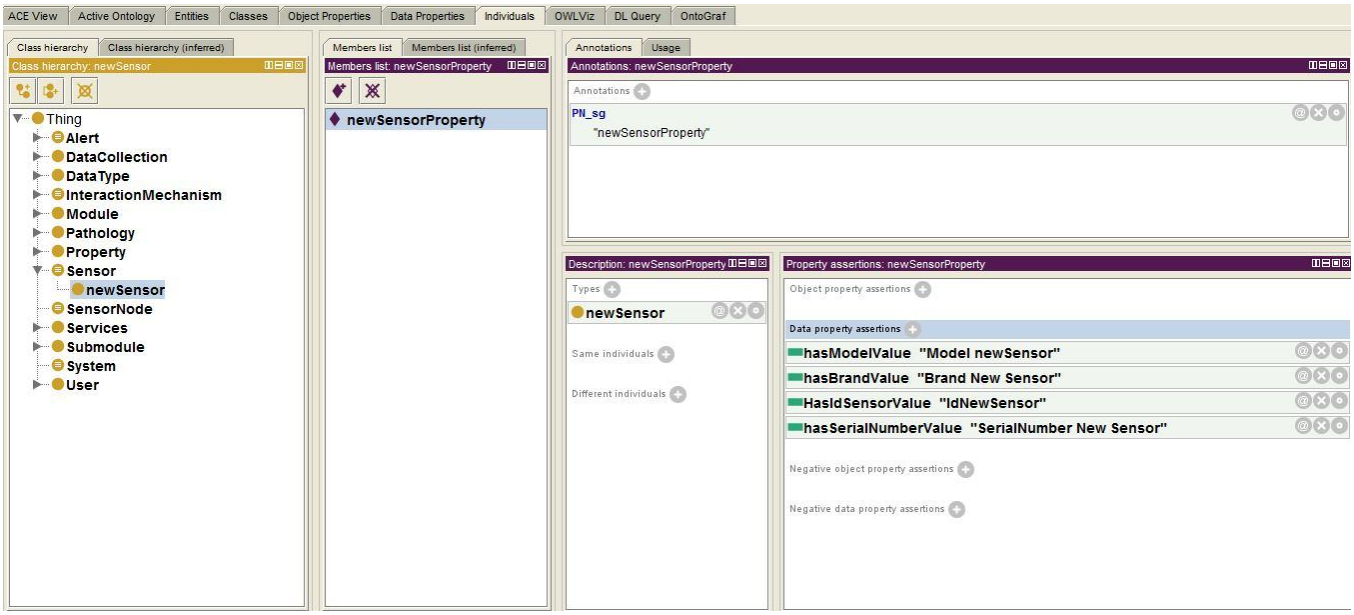
Na ontologia adicionar uma nova instância do tipo Sensor ou SensoryNode ou System.



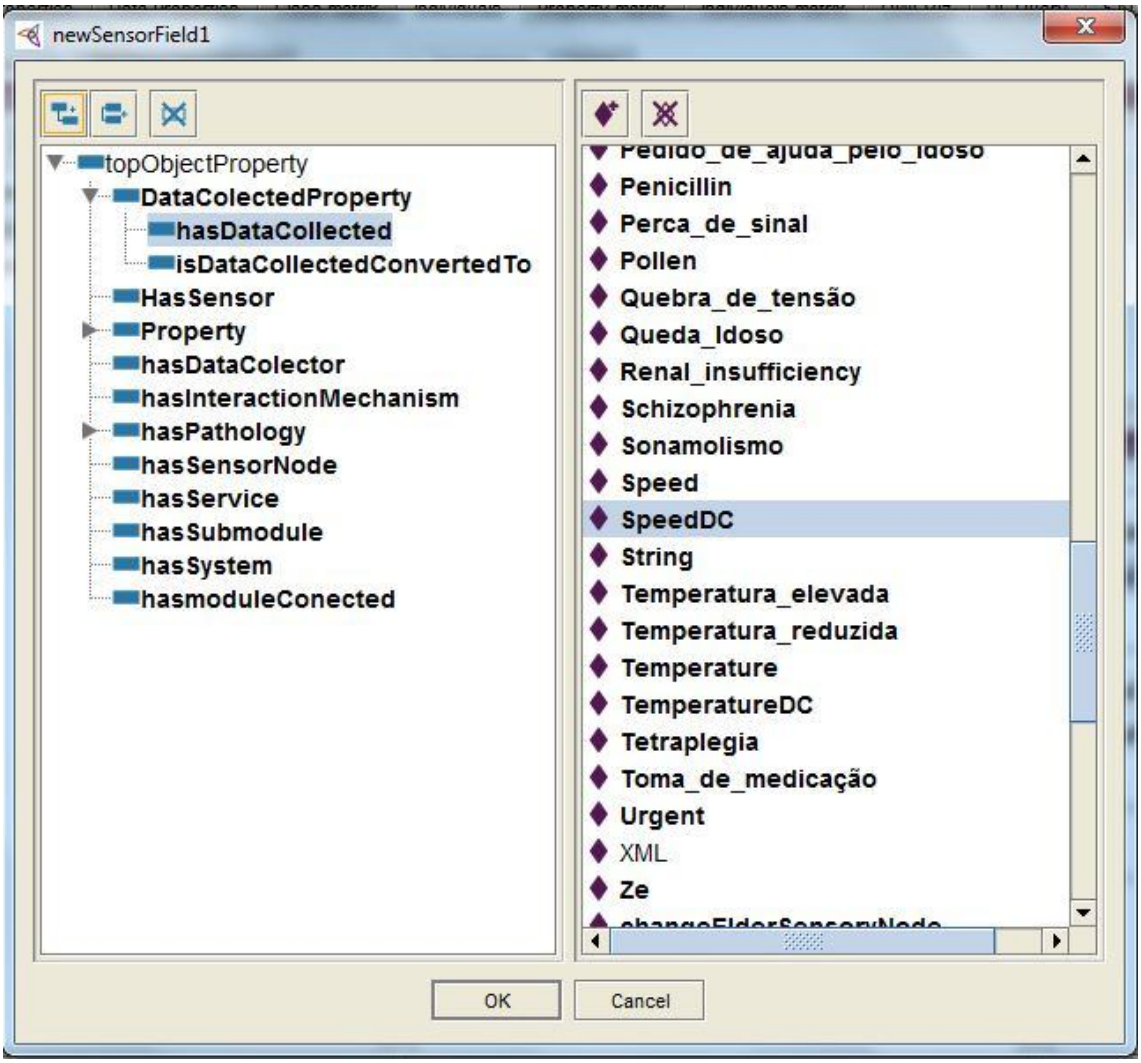
Há nova instância criada tem de se criar novos indivíduos que vão conter as suas propriedades e os dados que vai armazenar.



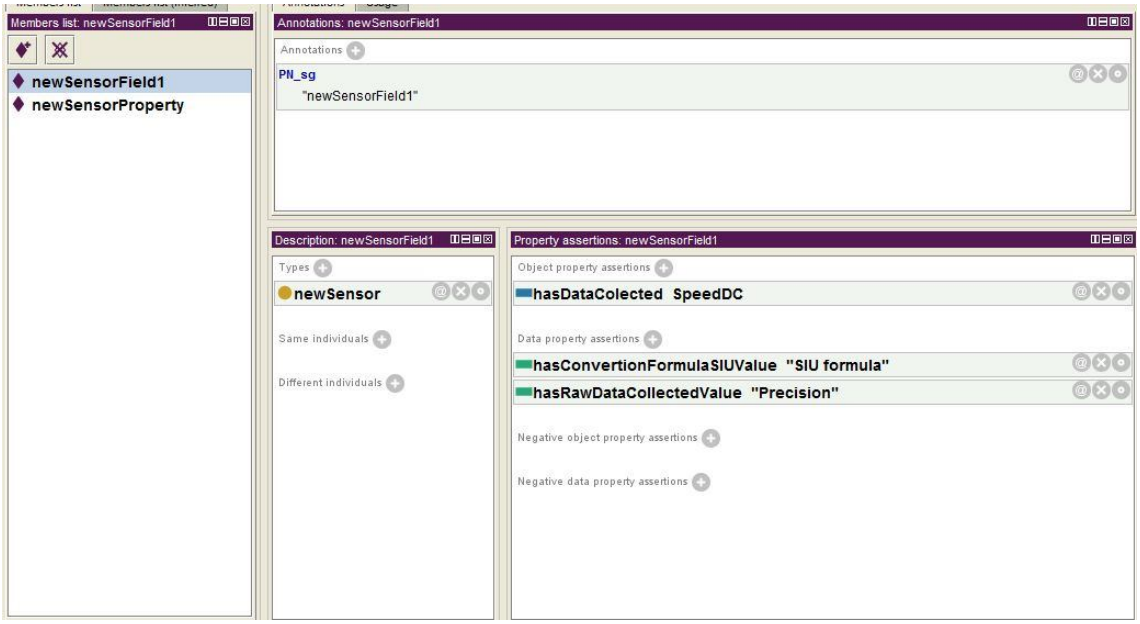
No caso de ser um Sensor ou um *SensoryNode* as propriedades são as mesmas sendo designadas por *SensorProperty*. O primeiro indivíduo a criar deve conter as propriedades do projecto (*Object property assertions*).



Os restantes indivíduos a ser criados vão indicar que tipo de dados é que o sensor recolhe. Assim para cada tipo de dados recolhido é necessário um novo indivíduo que indica que tipo de dados é recolhido, qual a fórmula de conversão para o SIU e qual a precisão dos dados recolhidos. Neste indivíduo tem de se indicar qual o tipo de dados recolhido e qual o tipo de dados que vai originar após a conversão, utilizando a formula de conversão para o SIU. Para adicionar esta informação tem de se adicionar à propriedade *Object property assertions* do tipo: *DataCollectedProperty -> hasDataCollected* e escolher o tipo de dados que o sensor vai recolher. Esta propriedade indica o tipo de dados que o sensor recolhe. A propriedade *DataCollectedProperty ->isDataCollectedConvertedTo* indica que tipo de dado é que a conversão vai originar no SIU.



As Data property assertions vão identificar indicar qual a precisão dos dados recolhidos e qual a fórmula que converte os dados para o sistema internacional de unidades(SIU). Para adicionar as Data property assertions ao sensor tem de se adicionar no campo hasPropertyValues -> hasSensorPropertyValue.

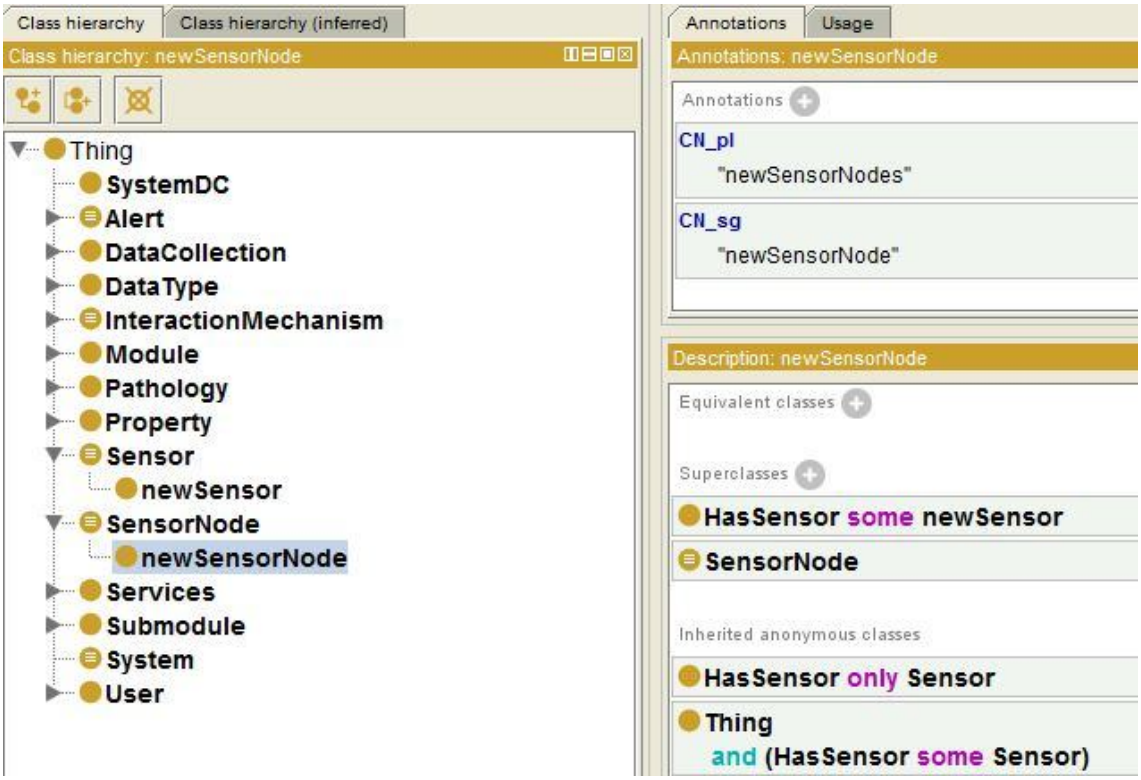


Os campos das propriedades devem de ser todos preenchidos, caso existam, pois são eles que possibilitam a identificação do Sensor ou *SensoryNode*. No caso da propriedade *HasListSensorDataCollected* tem duas sub-propriedades, uma que indicam a fórmula de conversão para o sistema internacional de unidades *hasConvertionFormulaSIUValue* e a outra que indica a precisão com que esse valor é medido *hasRawDataCollectedValue*. Estas duas propriedades devem de ser preenchidas em conjunto. No caso de o sensor recolher vários tipos de dados, deve-se adicionar um indivíduo por cada tipo de dados.

Para facilitar a identificação dos indivíduos ao projecto, é recomendável que o nome dos indivíduos inicie pelo nome do projecto concatenado com a característica do indivíduo. Por exemplo caso o indivíduo defina as propriedades de um sensor deve de ser *NewSensorProperty*.

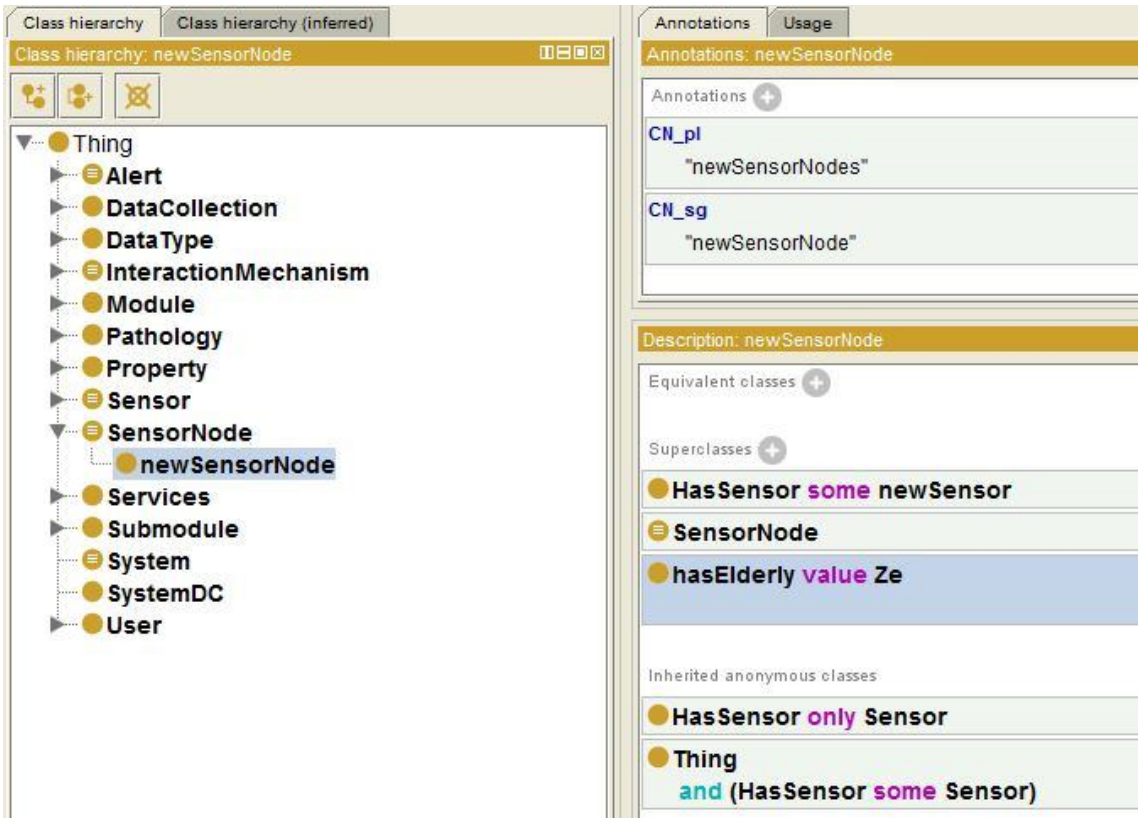
Propriedades do *SensorNode*

Um *SensoNode* tem a possibilidade de agregar diversos *Sensors*. De forma a indicar quais os sensores que se encontram ligados ao *SensorNode,* foi criada a propriedade *hasSensor* que indica quais são os *Sensors* que estão agregados ao *SensorNode*. Desta forma, é possível dar a conhecer quais são os sensores que estão ligados a um nó sensorial.



Esta propriedade também se pode aplicar aos *Ssystems*, indicando que Sensor é que fazem parte do *system*.

Um *SensorNode* pode estar ligado apenas um *elderly* de cada vez. Para incluir um idoso a um SensorNode tem de se adicionar a propriedade *hasElderly value* e o idoso correspondente.



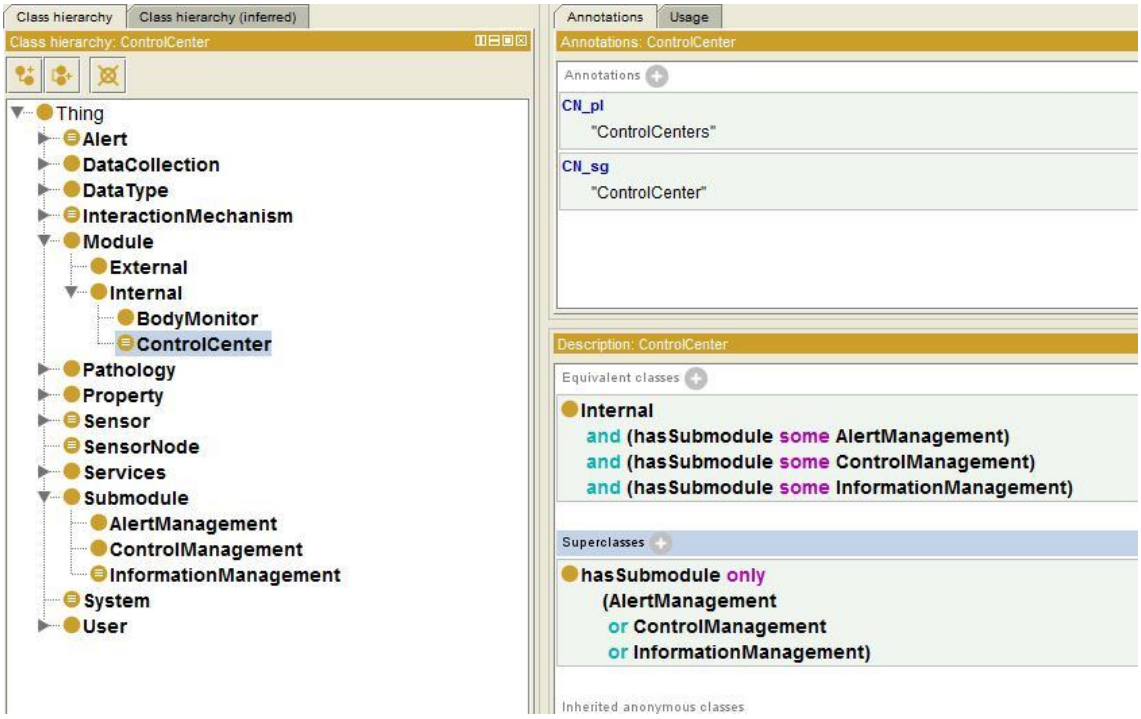
Esta propriedade também pode ser usada pelo Sensor ou pelo System. Desta forma também é possível ligar um idoso a um sensor ou a um sistema.

Ligação entre projectos, módulos e sub-módulos

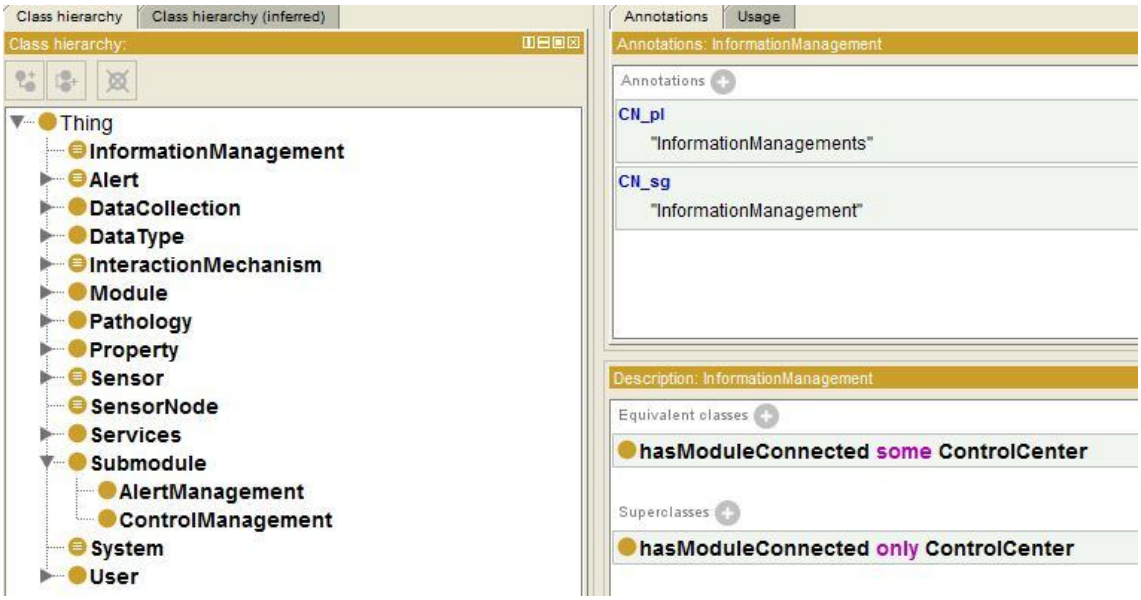
Os projectos que são adicionados ao Elde Care podem pertencer a um *module* ou a um *submodule*. Para indicar a ligação que existe entre os projectos, os *module* e *submodule* foram criadas as propriedades *hasSubModule* e *hasModuleConnected.* A propriedade *hasSubModule* indica quais são os sub-módulos que estão ligados ao *module* ou a um projecto (*Sensor*, *SensorNode*,*System*). A propriedade *hasModuleConnected* indica quais são os *module* que estão ligados ao *submodule* ou a um projecto.

De forma a indicar na ontologia as ligações existentes entre os *modules* e os *submodules* ou projectos tem de se adicionar a propriedade *hasSubModule some* e o nome do *submodule* ou

projecto. Esta propriedade é adicionada na *superclasses* do module que foi previamente criado. Após se adicionar a propriedade tem de se seleccionar a propriedade e converter para uma defined Class(Ctrl+D), em seguida seleccionar a propriedade criada no separador *Equivalent* classes e criar um *closure axiom* (seleccionar, botão direito do rato e *Create closure axiom*)



Para indicar na ontologia as ligações existentes entre os *modules* e os submodules ou projectos tem de se adicionar a propriedade hasModuleConnected some e o nome do module ou projecto. Esta propriedade é adicionada na *superclasses* do submodule ou projecto que foi previamente criado. Após se adicionar a propriedade tem de se seleccionar a propriedade e converter para uma defined Class(Ctrl+D), em seguida selecionar a propriedade criada no separador Equivalent classes e criar um closure axiom (seleccionar, botão direito do rato e Create closure axiom)



Tipos de dados existentes

Na ontologia estão definidos os tipos de dados que o projecto Elde Care conhece. Os tipos de dados estão separados em três grupos distintos: os tipos de dados suportados pela base de dados, os tipos de dados pertencentes ao *International System Units* e os tipos de dados que são recolhidos pelos projectos externos.

Para organizar estes tipos de dados, foi criado na ontologia a classe *DataType* que vai albergar todos os tipos de dados existentes no mundo Elde Care.

Os tipos de dados suportados pela Base de dados estão identificados na classe DataTypeSuportedDB. Nesta classe existem os tipos de dados que a base de dados do Elder Care trabalha. Ou seja sempre que são executados pedidos à base de dados, ela recebe e devolve apenas nos tipos de dados que são indicados nesta classe. Os tipos nesta classe são indicados pelo nome dos indivíduos que nela estão contidos. Por exemplo: *string* que indica que é um campo do tipo texto.

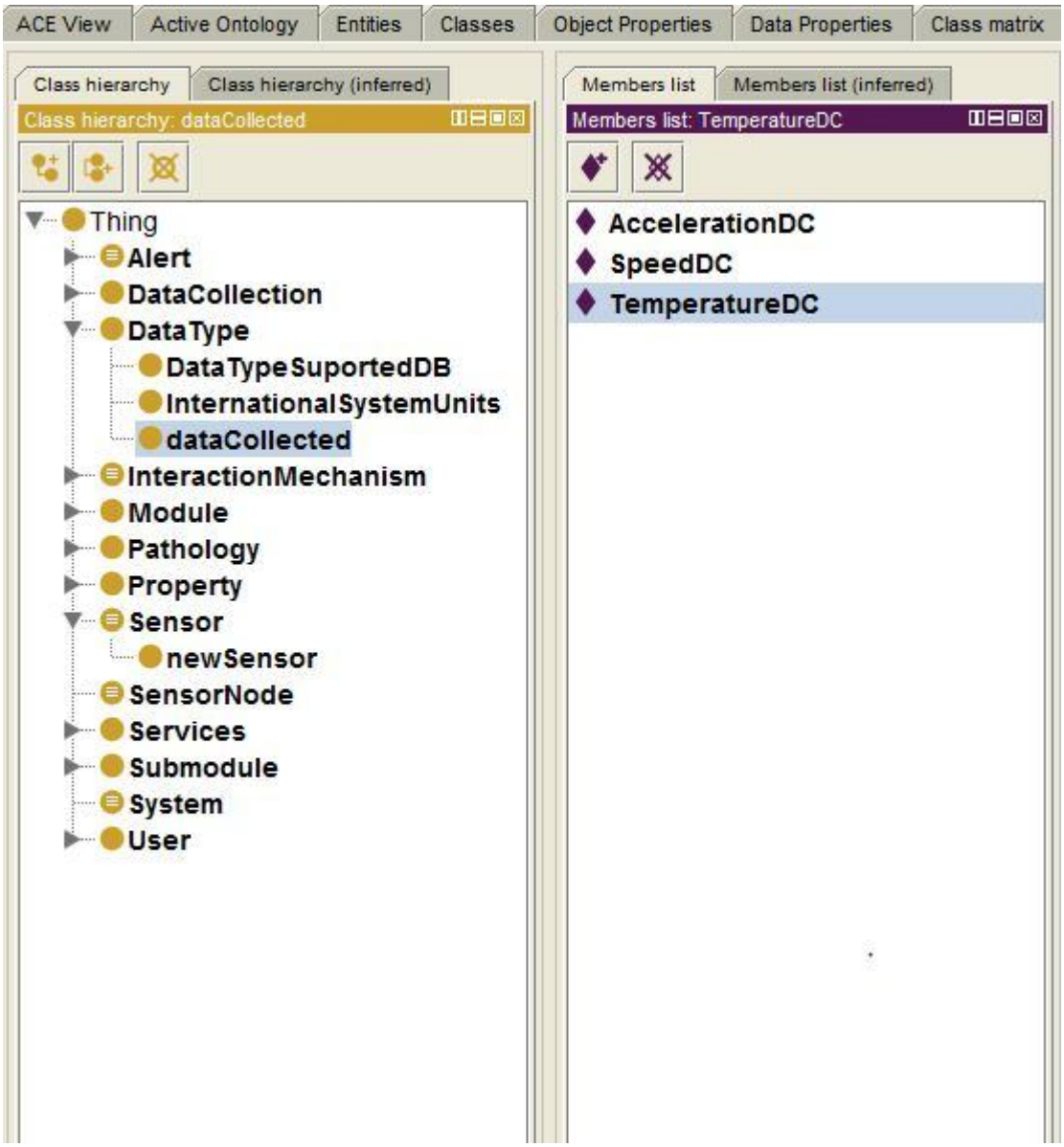
Os tipos de dados pertencentes ao SIU, indicam os tipos de dados que são devolvidos pela fórmula de conversão dos projectos. Desta forma é possível uniformizar os dados por tipos. Cada indivíduo desta classe indica o tipo de dados a que se refere, tendo como propriedade o

tipo de dados que devolve e a unidade que esse valor representa. Por exemplo o indivíduo *acceleration* tem como propriedades de unidade: *double* e como propriedade de unidades: *meter per second.*

Os tipos de dados recolhidos pelos projectos externos, indicam quais os tipos de dados recolhidos pelos projectos. Nesta classe os indivíduos não têm propriedades por estas encontram-se definidas nas definições dos projectos que as utilizam. apenas se encontra a indicação dos tipos de dados que são recolhidos, sem ter unidades específicas. Para os identificar e saber que são os dados recolhidos sem conversão, estes têm a terminação em DC indicando *DataCollected.*

Tipos de dados possíveis de adicionar

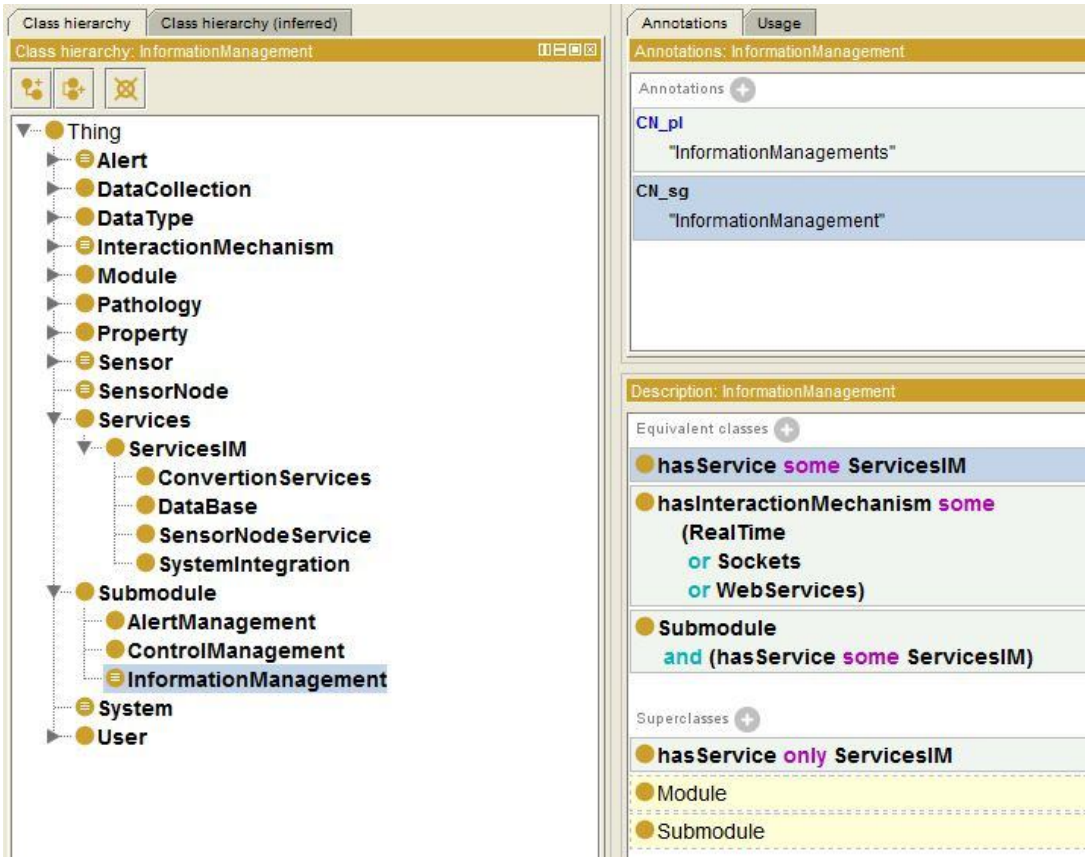
Devido ao crescente número e tipos de sensores é impossível conseguir inserir todos os tipos de dados de recolha que existem. Assim cada projecto pode adicionar novos tipos de dados dataCollected. Para adicionar um novo tipo de dados de recolha basta adicionar um novo indivíduo na classe DataType -> DataCollected e identificar esse indivíduo com a terminação em DC



As duas outras classes existentes na classe DataType só podem ser alteradas pelos administradores do sistema, ou seja, os projectos que se ligam ao Elde Care só podem adicionar dados do tipo DataCollected. Isto só acontece pois os tipos de dados suportados pela base de dados são previamente definidos para manter a compatibilidade entre os vários sistemas do Elde Care . O SIU é uma tabela que já se encontra definida.

Serviços na Ontologia

O projecto Elde Care não pretende unicamente recolher dados de projectos externos e internos. Pretende, também, divulgar os serviços que estão disponibilizados. Para que um projecto partilhe os seus serviços, tem de adicionar uma nova subclasse à classe *Services* e adicionar um indivíduo por cada serviço disponibilizado. No projecto que disponibiliza os serviços tem de se adicionar a propriedade *hasService* some ServiceIM (subclasse criada previamente) na superclasse. No Protégé tem de se seleccionar a propriedade e converter para uma *defined Class*(Ctrl+D), em seguida seleccionar a propriedade criada no separador *Equivalent* classes e criar um *closure axiom* (seleccionar, botão direito do rato e *Create closure axiom*).



Para dar a conhecer o que o serviço faz, deve-se adicionar uma breve descrição do serviço em comentário.

Como os serviços disponibilizados têm de receber um ficheiro XML cada indivíduo relativo a um serviço tem de ter a propriedade *HasXsdFileService* a indicar qual o ficheiro XSD utilizado para validar o ficheiro XML para comunicar. Desta forma é possível utilizar o serviço que disponibiliza os ficheiros XSD para pedir esse mesmo ficheiro para poder comunicar com o serviço pretendido.

Para quem pretender conhecer os serviços disponibilizados na ontologia, pode questionar a ontologia acerca dos serviços existentes. Para tal, basta ir ao separador DL Query e escrever *Services.* Na lista apresentada, pode-se escolher as subclasses que são todos os projectos que disponibilizam serviços. Caso se escolha também os indivíduos, aparece listado todos os serviços disponibilizados na ontologia por todos os projectos. Após se saber o nome do serviço e em que projecto é que se encontra disponibilizado, tem de se ir ao separador *individuals* e ver qual o ficheiro XSD que ele pretende bem como qual as suas funcionalidades, caso estejam descritas nos comentários.

Utilizadores do sistema

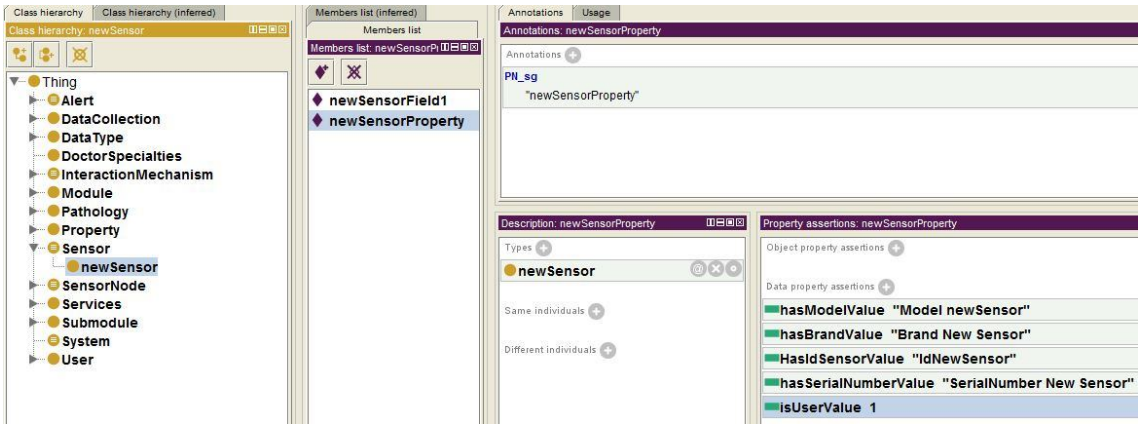
Os utilizadores do sistema são todos os elementos que se encontram registados no Elde Care, e que contribuem de forma directa ou indirectamente ao projecto. O simples facto de só consultar dados, não é um elemento suficiente para ser considerado um utilizador do sistema. Neste caso, não existe nenhuma actividade que traga mais-valias para o projecto.

Existem dois tipos de utilizadores, os projectos que se ligam ao Projecto Elde Care e que contribuem para o projecto Elde Care e as pessoas que e encontram registadas no sistema. Nos projectos que se ligam ao projecto Elde Care estão incluídos os Sensors, SensorNodes, Systems, Modules e Sub-Modules. Estes projectos têm como finalidade, contribuir de alguma forma para a melhoria da qualidade de vida dos idosos. A contribuição pode ser efectuada directa ou indirectamente, ou seja, pode contribuir ajudando directamente, fornecendo serviços, recolhendo dados, analisando dados ou simplesmente fornecendo ferramentas que ajudem na interligação de dados entre sistemas. As pessoas que estão registadas no sistema podem ser classificadas como idoso, médico ou Contacto. Sendo que uma pessoa pode estar englobada nos três grupos em simultâneo. Os idosos são as pessoas que vão ser monitorizadas por algum sistema do Elde Care. Os médicos são as pessoas que têm o direito de exercer medicina e que estão registados no sistema. Os contactos são todas as pessoas que estão registadas no sistema e que estão na lista de contacto de um idoso.

Como adicionar um Projecto como utilizador do sistema

Para que um projecto seja considerado um utilizador do sistema, tem de ser adicionado na ontologia seguindo os passos do capítulo como adicionar um novo projecto. Desta forma o projecto fica adicionado na ontologia e é-lhe atribuído uma chave única. Para que esse projecto seja considerado utilizador do sistema, tem de preencher a *data property assertion isUserValue*, definir como *boolean* e preencher com o valor “1”. Esta propriedade indica que

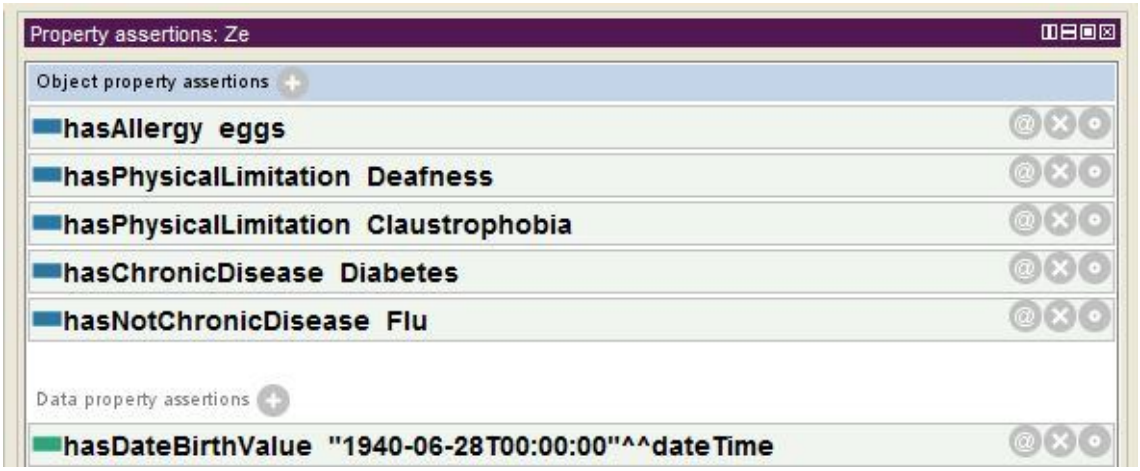
o projecto é um utilizador do sistema. Para deixar de ser um utilizador do sistema, a propriedade tem de ser preenchida com o valor “0”.



Como adicionar uma pessoa no sistema

Uma pessoa é um indivíduo que é adicionada na classe User -> Person e que pode interagir com o sistema. O nome do indivíduo criado tem de ser único, ou seja não pode estar repetido na ontologia. Todas as pessoas adicionadas têm de ter uma data de nascimento. Para tal tem de se preencher a data *property assertion hasDateBirthValue* do tipo *DateTime* onde o formato da data é yyyy-mm-ddThh:mm:ss.

Nesta ontologia é essencial saber quais as patologias que cada idoso tem. As patologias estão classificadas na classe *Pathology*. Nesta classe encontram-se as patologias que já estão registadas no sistema. As patologias estão classificadas em três grupos (Allergy, Diseases, Limitation) facilitando a sua classificação. Para incluir as patologias que cada idoso tem, existem as object property-> hasPathology que contem as propriedades hasAllergy, hasDisease e hasLimitation de forma a poder indicar quais são as patologias que os idosos registados no sistema possuam. Para adicionar estas propriedades, que indicam quais as patologias que o idoso tem, deve-se adicionar a *object property assertion* com as respectivas patologias.

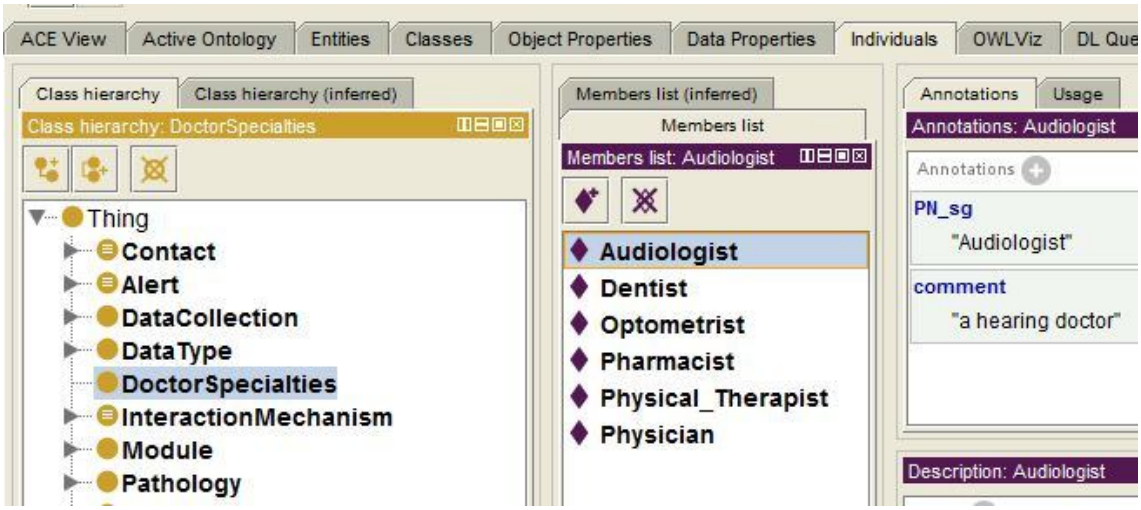


Nesta ontologia também é essencial saber quais são os médicos que existem no sistema, e quais são as suas especialidades. Desta forma é possível disponibilizar aos idosos quais são os médicos de uma determinada especialidade e que se encontram registadas no sistema. Para que uma pessoa seja classificada como medico, tem de indicar qual a especialidade que exerce. Para adicionar esta informação tem de se adicionar a propriedade isDoctor e qual a especialidade na object property assertion do doctor. As especialidades estão na classe DoctorSpecialties. Desta forma é possível saber quais são os médicos que estão registados na ontologia e qual a sua especialidade.

Adicionar novas especialidades médicas

Sempre que é adicionado um novo médico, este tem de ter a indicação da sua especialidade. Com esta informação é possível conhecer todos os médicos que se encontram na ontologia e que possam prestar serviços de uma determinada especialidade. Assim é possível informar os utilizadores, dos médicos de uma determinada especialidade.

Na ontologia existe uma classe onde se encontram as especialidades médicas existentes na ontologia, o que não implica que estejam todas as especialidades existentes. Assim é possível adicionar novas especialidades médicas ainda não existentes. Desta forma é possível adicionar novos médicos e novas especialidades médicas, conseguindo assim melhorar os serviços que podem ser prestados pelo projecto Elde Care. Para adicionar uma nova especialidade que ainda não se encontre na ontologia do Elde Care tem de se adicionar um novo indivíduo na classe *DoctorSpecialties*. Para adicionar o indivíduo tem de se ir ao separador *individuals* e adicionar um novo indivíduo na classe *DoctorSpecialties*, sendo o nome do indivíduo a especialidade do médico. De forma a melhorar a informação que se encontra na ontologia, deve-se adicionar um comentário que indica qual o foco da especialidade.



Após se ter adicionado uma nova especialidade, pode-se adicionar esta especialidade, aos médicos já existentes ou aos novos que irão ser adicionados.

Adicionar novas Patologias

As patologias de cada idoso, devem de fazer parte da sua informação pessoal, que se encontra registada na ontologia do Elde Care. Desta forma, é possível conhecer qual o melhor acompanhamento a dar a cada idoso. Também é possível que qualquer especialista que acompanhe o idoso, tenha uma visão mais completa e detalhada do quadro clínico e terapêutico do idoso.

As patologias que existem na ontologia do Elde Care encontram-se na classe *Pathology*. Esta classe encontra-se dividida em três sub-classes: *Allergy*, *Diseases* e *Limitation*. Na sub-classe Allergy constam as alergias. Esta sub-classe encontra-se dividida em outras três sub-Classes, possibilitando uma melhor organização das alergias. As três sub-classes que estão na sub-classe Allergy são: food, drug,other.

Na sub-classe Diseases constam as doenças. Esta sub-classe encontra-se dividida em duas sub-classes: a *Chronic* e a *NotChronic*, possibilitando a separação das doenças crónicas das doenças que não são crónicas.

Na sub-classe Limitation constam as limitações. Esta sub-classe encontra-se dividida em duas sub-classes: a Phisical e a Psychological, possibilitando a separação das limitações físicas das limitações psicológicas.

Pode ser necessário adicionar uma nova patologia que ainda não se encontre na ontologia. Para adicionar uma nova patologia tem de se seleccionar a classe Pathology e adicionar um novo indivíduo na sub-Classe correspondente. O indivíduo criado tem o nome da nova patologia criada. Esta tem de se encontrar na sub-classe correcta, de forma a manter a ontologia coerente. Após se ter criado a patologia e esta estar criada na sub-classe correcta, é possível adicionar esta nova patologia aos idosos.

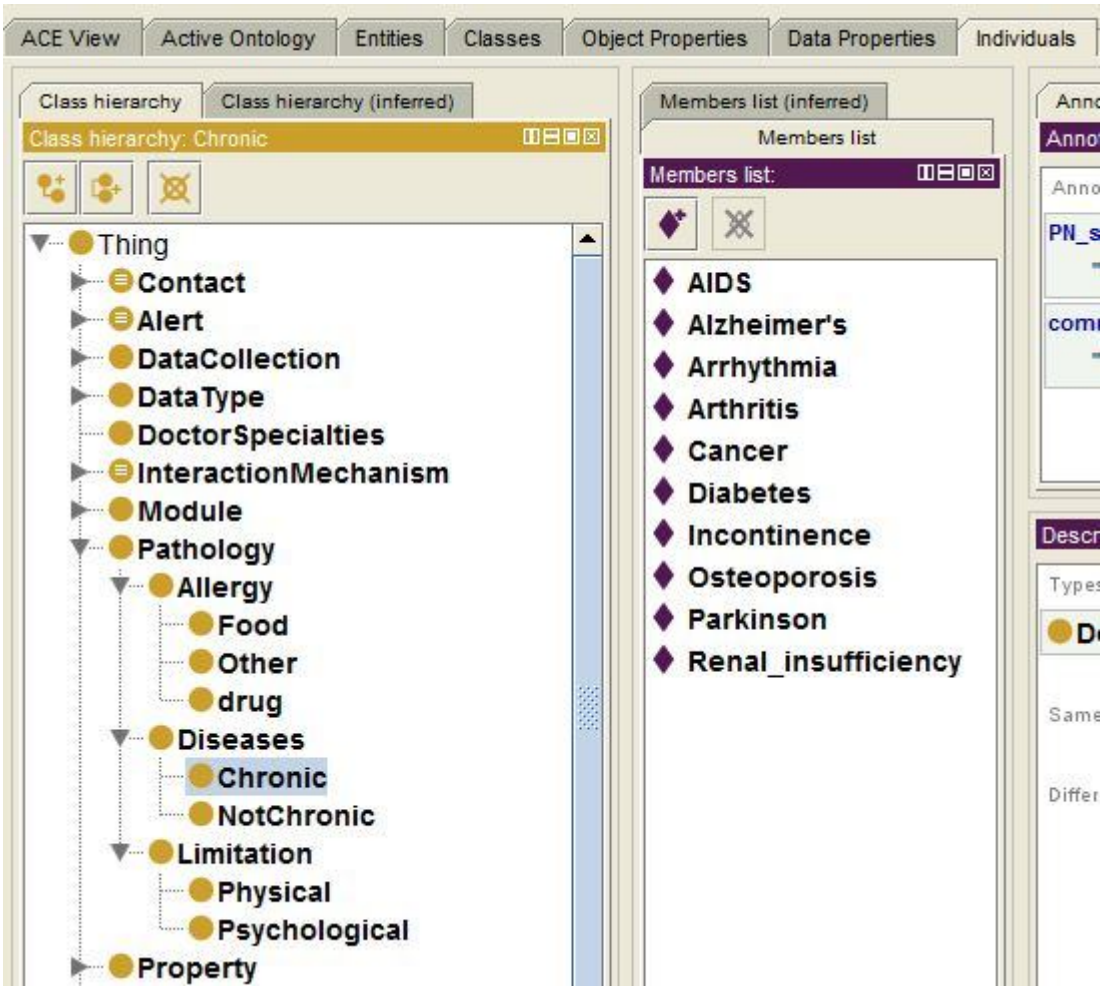


Diagrama de classes completo da ontologia do Elde Care