Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

Algoritmos de Aprendizagem de Contexto em Computação Ubíqua: Avaliação para o caso de estudo em PDAs

João José Lopes Peixoto

Coimbra, 2006

Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Informática
Pólo II da Universidade, Pinhal de Marrocos
3030 - 290 Coimbra, Portugal
Tel.: 239 790 000 - Fax: 239 701 266

Algoritmos de Aprendizagem de Contexto em Computação Ubíqua: Avaliação para o caso de estudo em PDAs

João José Lopes Peixoto

Coimbra, 2006

Dissertação submetida para satisfação parcial dos requisitos do programa de mestrado em Informática e Sistemas, Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Dissertação realizada sob orientação do
Prof. Doutor Francisco Colunas Pereira da Câmara Pereira
Professor Auxiliar
do Departamento de Engenharia Informática
da Faculdade de Ciências e Tecnologia
da Universidade de Coimbra

Aos meus pais e irmã...

Conteúdo

Conteúdo **i**

1 Introdução **7**
1.1 Motivação e enquadramento 7
1.2 Contribuições esperadas 7

2 Computação Ubíqua **9**
2.1 O Conceito 9
2.2 Proactividade 9
2.3 Contexto 10
2.4 Sensores 12

3 Estado da Arte **15**
3.1 Modelos de Contexto e Aplicações 15
3.1.1 *Context Toolkit* 15
3.1.2 *AURA* 16
3.1.3 Modelo de Contexto de Li 17
3.1.4 *Context Modelling Language* 19
3.1.5 *Place Lab* 21
3.1.6 *ContextPhone* 22
3.2 Modelos e Algoritmos de Aprendizagem 24
3.2.1 Modelos de Classificação 25
3.2.2 Modelos de Associação 37
3.2.3 Modelos de *Clustering* 39
3.3 Representação do Conhecimento 41

4 Proactividade a partir de histórico aplicacional **47**
4.1 O *PDA* como dispositivo ubíquo 47

4.2 A Ideia . . . 48
4.3 Contexto . . . 51
4.4 Arquitectura da Aplicação de Amostragem . . . 52
4.4.1 Processos . . . 53
4.4.2 Global Positioning System . . . 55
4.4.3 Global System for Mobile Communications . . . 70
4.4.4 WiFi . . . 78

5 Experimentação **83**
5.1 Introdução . . . 83
5.2 Aquisição de dados . . . 84
5.3 Análise prévia dos dados . . . 85
5.4 Algoritmos de Classificação . . . 86
5.4.1 Teste à categoria dos atributos . . . 87
5.4.2 Classificação com apenas dois atributos . . . 88
5.4.3 Classificação com os dados agregados . . . 90
5.5 Algoritmos de Associação . . . 92
5.5.1 Estudo com vários utilizadores . . . 93
5.5.2 Análise aos resultados . . . 102
5.6 Algoritmos de *Clustering* . . . 103
5.6.1 Estudo com vários utilizadores . . . 104
5.6.2 Análise aos resultados . . . 112

6 Conclusões **115**
6.1 Contribuições . . . 115
6.1.1 Proactividade . . . 115
6.1.2 Contexto . . . 118
6.2 O Futuro . . . 119
6.2.1 Computação Ubíqua . . . 119
6.2.2 Tecnológico . . . 120

Bibliografia **121**

A Arquitectura da Aplicação de Amostragem **127**
A.1 Classe GPS_COM . . . 127
A.2 Classe GPS . . . 128
A.3 Classe DadosGPS . . . 129
A.4 Classe GSMInfo . . . 131
A.5 Classe DadosWifi . . . 133

B Histórico Aplicacional com GPS e Aplicações **137**

C Histórico Aplicacional com todos os atributos **143**

D Informação Agregada dos vários Utilizadores **145**

E Resumo das regras para os vários Utilizadores **151**

F Resumo de clusters para os vários Utilizadores **157**

Agradecimentos

Com a realização deste trabalho, é mais uma meta que se atinge. E o termo meta é mesmo o mais adequado. Comparativamente, um trabalho de dissertação de mestrado é como uma maratona. Um percurso longo e que tem que ser percorrido a sós. Temos que ser nós a dar "corda aos sapatos"se queremos realmente chegar ao fim da prova.

Mas como acontece com qualquer corredor, as vitórias não são apenas suas, são também de toda a sua equipa. E é essa equipa: treinadores, corredores, preparadores físicos, roupeiros, psicólogos, etc... que devem agora ser recordados e que tanto contribuíram para a corrida.

Aos meus treinadores, Prof. Dr. Carlos Bento e Prof. Dr. Francisco Câmara Pereira, que tanto apoio, orientação e motivação prestaram de forma incansável durante o decorrer deste trabalho.

Aos meus colegas corredores do Grupo de Computação Ubíqua do Laboratório de Inteligência Artificial com quem, durante os tradicionais almoços semanais, partilhávamos soluções e dificuldades.

Aos meus colegas corredores do Departamento de Coimbra do Centro de Computação Gráfica, que tiveram a amabilidade de me ouvir falar sobre o mestrado.

Aos roupeiros, ou melhor amigos, que forneciam um novo par de calçado, incentivando e preocupando-se sempre em saber como estava a correr o trabalho.

Às barras energéticas, amigos cibernéticos, que davam uma energia extra para continuar nas noites passadas em volta do trabalho.

Às bebidas energéticas, Profª. Dra. Bernardete e Prof. Dr. Paulo Gomes, que com as ajudas pontuais forneceram lucidez e esclarecimento em algumas situações.

Por último, aos meus pais e irmã, que embora ainda não saibam o que significa Computação Ubíqua, não deixaram por isso de prestar imenso apoio.

Resumo

A Proactividade é um dos temas centrais em Computação Ubíqua. Para que um sistema seja proactivo, é necessário que seja capaz de prever e antecipar as necessidades do utilizador, de forma a minimizar a necessidade de atenção.

A tarefa de antecipação das necessidades do utilizador pressupõe que o sistema seja capaz de determinar com exactidão o contexto do utilizador e em função deste actuar de forma rápida e precisa.

Vários trabalhos foram já propostos com o objectivo de definir tanto o modelo de contexto como a informação que nele deve existir. Mas o mesmo esforço não foi ainda realizado relativamente ao raciocínio sobre o contexto.

Desta forma, o presente trabalho de dissertação de mestrado tem como objectivo estudar quais os algoritmos de aprendizagem mais adequados no âmbito do problema de proactividade em *Personal Digital Assistants (PDA*s). Para este efeito foi desenvolvida uma Aplicação de Amostragem que faz aquisição de informação de contexto, com base na localização do utilizador e na presença da tecnologia *WiFi*, com o objectivo de associar esta informação à actividade aplicacional realizada em *PDA*s.

O resultado da experimentação permite afirmar que os algoritmos de *clustering*, em particular o *SimpleKMeans*, são os mais adequados para a aprendizagem de contexto em problemas de proactividade nestes dispositivos.

Abstract

Proactivity is one of the core subjects in Ubiquitous Computing. In order for a system to be proactive, it is necessary predict and anticipate the user's needs, in order to minimize distractions on a user's attention.

The task of anticipating the user's needs assumes that the system is capable to accurately determine the user's context and act accordingly in a precise and fast way.

Several works have already been proposed with the objective of defining the context model as well the information that the model should contain. But the same effort has not yet been applied to context reasoning.

Considering this, the objective of the present work is to study which are the learning algorithms more suitable for the proactivity problem on Personal Digital Assistants (PDAs). For this purpose a Sampling Application that acquires context information, based on the user's location and on the presence of WiFi technology, was developed with the objective of associating that information to the applicational activity done on PDAs.

The results of the experience allow us to say that the clustering algorithms, especially SimpleKMeans, are the most adequate for context learning in proactivity problems on PDAs.

Capítulo 1

Introdução

1.1 Motivação e enquadramento

Pelo facto da área de Computação Ubíqua ser ainda muito recente, tem existido uma aposta forte em determinados temas, em detrimento de outros. Tendo como referência as várias camadas que devem compor um sistema proactivo, verifica-se que os temas Sensores e Contexto são os que mais têm sofrido evolução. Este facto tem a sua razão de ser, uma vez que para se poder estudar quais os mecanismos de raciocínio mais adequados para os diferentes problemas, é necessário inicialmente realizar todo um trabalho de aquisição de informação de contexto e garantir que a mesma tem o mínimo de qualidade, pois só assim podemos esperar bons resultados da parte da camada Raciocínio. Mas a área de raciocínio encontra-se ainda pouco explorada e por esta razão torna-se desejável e fundamental estudar e avaliar quais os mecanismos de raciocínio e respectivos algoritmos de aprendizagem que mais se adequam a este tipo de aplicações. Desta forma, garantimos que a informação de contexto é devidamente aproveitada, rentabilizando todo o esforço dispendido na aquisição de informação contextual. Não basta ter informação contextual com qualidade e em quantidade, se depois os mecanismos de raciocínio falham.

1.2 Contribuições esperadas

Pelos vários trabalhados já realizados nesta área, verifica-se que as escolhas sobre os modelos de aprendizagem de contexto recaem apenas em dois: Regras de Classificação extraídas directa ou indirectamente da *Ontology Web Language* (*OWL*) ou então Redes Bayesianas por serem consideradas por alguns autores como o mais indicado para lidar com a incerteza do contexto. No entanto, os modelos de aprendizagem não se resumem apenas a estes e este

facto dá origem às seguintes questões: Será que não existem outros modelos de aprendizagem igualmente interessantes para sistemas *context-awareness*? Serão estes modelos os mais indicados para dispositivos como os *PDA*s? Existirão outros modelos também suficientemente flexíveis para responderem com sucesso a situações de incerteza? Os problemas associados à proactividade podem ser solucionados com algoritmos de Classificação, de Associação ou de *Clustering*?

Nesta dissertação, procuramos responder a estas perguntas, ou contribuir para novas perspectivas sobre as mesmas. Desta forma, é inicialmente apresentado no capítulo 2 a definição e composição do conceito de proactividade.

Para que a proactividade seja uma realidade é necessário realizar a aquisição de informação de contexto. No capítulo 3, são apresentadas algumas das representações de contexto já existentes e suas aplicações. No entanto, o contexto por si só não implica que o sistema seja proactivo, é necessário que existam mecanismos de aprendizagem que a partir deste permitam prever as necessidades do utilizador. Desta forma, neste capítulo são também apresentados os vários algoritmos de aprendizagem que se encontram implementados na ferramenta *WEKA*, algoritmos estes que foram utilizados no estudo desenvolvido neste trabalho.

No capítulo 4, é apresentada a Aplicação de Amostragem que foi desenvolvida com o objectivo de obter informação de contexto de diferentes utilizadores. Esta informação é composta por atributos que determinam a localização do utilizador e a presença da tecnologia *WiFi*, assim como a informação relativa à actividade aplicacional realizada no *PDA*. A escolha por este tipo de dispositivo ubíquo está relacionada com o facto de que, para se adquirir o contexto, é necessário que as fontes de informação estejam próximas do utilizador e como estes dispositivos acompanham o utilizador para onde quer que este se desloque, são por isso os mais adequados.

A partir dos diferentes contextos obtidos, foi realizada no capítulo 5 a experimentação que tem como objectivo estudar quais os algoritmos de aprendizagem de contexto mais indicados para problemas de proactividade, em particular para dispositivos como os *PDA*s. Esta experimentação tem por base a verificação da hipótese de existência de uma correlação entre a localização do utilizador, a existência de recursos de comunicação (*WiFi*) e a actividade computacional no *PDA*.

Os resultados obtidos na experimentação permitiram verificar que os algoritmos de *clustering*, em particular o *SimpleKMeans*, são os mais adequados para a aprendizagem de contexto em problemas de proactividade, especificamente para este tipo de dispositivos.

Capítulo 2

Computação Ubíqua

2.1 O Conceito

Em 1991, Mark Weiser, então investigador da Xerox's Palo Alto Research Center (Parc), apresentou a sua visão sobre o que seria *ubiquitous computing*. Na sua essência está a criação de ambientes saturados de computação e comunicação integrada de forma agradável com os utilizadores. Estes ambientes estão assim povoados por dispositivos ubíquos, que mais não são do que máquinas computacionais completamente "invisíveis"para o utilizador, de tal forma que não necessitam de grande atenção por parte deste, permitindo que se concentre noutras actividades [Weiser, 2002].

Posteriormente, em meados dos anos 90, surgiu o termo *pervasive computing*, que representa basicamente a visão de Weiser [Satyanarayanan, 2002]. Outros termos foram surgindo, como *proactive computing*, *autonomic computing*, *wearable computing*, *situated computing*, que devido à proliferação de termos e à falta de delimitação das áreas de intervenção de cada um, tornam difíceis as suas definições. Para de alguma forma clarificar esta problemática, Satyanarayanan [Satyanarayanan, 2002], editor chefe da publicação *IEEE Pervasive Computing*, considera *ubiquitous computing* e *pervasive computing* como sinónimos para a mesma área de investigação.

2.2 Proactividade

O grande objectivo da computação ubíqua não é criar novas ferramentas ou aplicações, mas sim tornar a execução das actividades mais rápida e fácil [Weiser, 2002], torná-las **proactivas**. Isso implica antecipar e reagir de acordo com as condições do mundo físico e não reagir apenas a ele. A grande diferença é que o utilizador deixa de ser parte do sistema (pois é reduzida

a interacção entre o sistema e o utilizador) e passa a ser o supervisor do sistema, actuando apenas quando for estritamente necessário [Want et al., 2002].

Mas para que o sistema possa antecipar as necessidades e/ou intenções do utilizador, é necessário obter uma ideia clara da actividade actual do utilizador. Esta é a grande dificuldade num sistema proactivo. Assumindo que o utilizador não diz explicitamente ao sistema qual a sua actividade actual, esta terá de ser deduzida a partir de diversos factores [Davies and Gellersen, 2002], criando assim um contexto associado. Uma forma de inferir a informação acerca de pessoas e do ambiente envolvente é através de sensores que medem valores aproximados de grandezas do mundo físico [Abowd et al., 2002, Estrin et al., 2002]. Esta informação obtida a partir dos sensores, em conjunto com a informação resultante da interpretação da actividade humana, permite definir um contexto das interacções entre o utilizador e o ambiente que o rodeia [Abowd et al., 2002].

Através do contexto, é possível extrair as actividades do utilizador. Mas, para que o sistema seja proactivo, não basta saber as actividades, é também necessário antecipar as intenções do utilizador, actuando de acordo com o contexto. A actuação permite converter a informação do contexto em acção [Estrin et al., 2002], através do recurso a actuadores. Um problema associado aos actuadores é a incerteza [Estrin et al., 2002]. Estes só funcionam bem quando devidamente informados acerca do contexto [Abowd et al., 2002]. Para minimizar esta incerteza, são necessários mecanismos de raciocínio. São estes mecanismos que a partir da informação contextual permitem a adequada resposta dos actuadores ao sistema. Por sua vez, a validade da informação de contexto está directamente relacionada com a informação passada pelos sensores.

Em resumo, um sistema proactivo integra em geral quatro entidades relacionadas entre si: Sensores, Contexto, Raciocínio e Actuadores (Figura 2.1).

2.3 Contexto

A nossa interacção com o ambiente físico fornece uma quantidade de informação bastante interessante, das mais variadas formas, sem a necessidade de intervenção do utilizador. Por exemplo, caminhar num espaço é suficiente para anunciar a nossa presença e identificá-la de acordo com uma localização [Abowd et al., 2002].

Definir com exactidão o que é o contexto não é uma tarefa fácil. São várias as definições de contexto e cada uma delas verdadeira de acordo com o propósito da sua aplicação. Possivelmente a que mais consenso recolhe na comunidade científica ainda é a definição de Abowd [Abowd et al., 1999], que considera o contexto como toda a informação que pode permitir caracterizar uma situação de uma determinada entidade, onde a entidade pode ser uma pessoa, um lugar ou um objecto que se considere relevante para a interacção entre o utilizador e o ambiente. Só que de tão generalista que ela é, não diz em concreto o que é o contexto.

Figura 2.1: Representação de proactividade

"There is more information available at our fingertips during a walk in the woods than in any computer system" - Mark Weiser

A primeira definição de contexto surgiu em 1993, apresentada por Schilit [Schilit et al., 1993], onde a informação contextual era utilizada com o objectivo de criar um serviço de ambiente dinâmico, que permitia que as aplicações se adaptassem à mobilidade do utilizador. O contexto aqui considerado como ambiente é definido pelo local (salas), utilizadores e dispositivos (impressoras e monitores). Sempre que um utilizador abandona a sala onde se encontra e entra numa nova sala, as suas aplicações devem então reconfigurar-se de acordo com o ambiente físico, de forma a saber quais os dispositivos disponíveis nessa sala sem que as mesmas necessitem de ser reinicializadas. Schilit [Schilit et al., 1994] considera que os aspectos importantes no contexto são: localização do utilizador, com quem o utilizador está e que recursos estão próximos. Brown [Brown et al., 1997] define o contexto como: a localização do utilizador, identificação das pessoas dentro do ambiente do utilizador, hora do dia, estação do ano, temperatura. Ryan [Ryan et al., 1998] define o contexto como: a localização do utilizador, o ambiente, a identidade e o tempo. No projecto *CyberDesk*, Dey [Dey et al., 1998] defende que o contexto deve incluir informação sobre: em que é que o utilizador está a trabalhar, hora do dia, localização física do utilizador, estado emocional, ambiente social e objectos na sala.

Localizar as entidades identificáveis é uma característica comum em aplicações que utilizam o contexto [Abowd et al., 2002]. Mas o contexto não é apenas localização (onde) e identidade (quem). Uma completa definição de contexto envolve também quando e o quê [Abowd et al., 1999] (Figura 2.2).

Figura 2.2: Definição de contexto

Perceber e interpretar a actividade humana é um problema difícil, mas a interpretação da mesma fornece informação muito útil. Uma estratégia é incorporar informação sobre a actividade do utilizador. Qual a aplicação que está a utilizar? A que informação acedeu?

As aplicações que recorrem à informação do contexto pretendem ter informação sobre: quem, onde, quando e o quê [Abowd et al., 1999]. Estas características do contexto são instanciadas nas seguintes dimensões: Identidade, Localização, Tempo e Actividade (Figura 2.3). Estas quatro dimensões de contexto são consideradas fundamentais para a caracterização de uma situação em particular [Abowd et al., 1999].

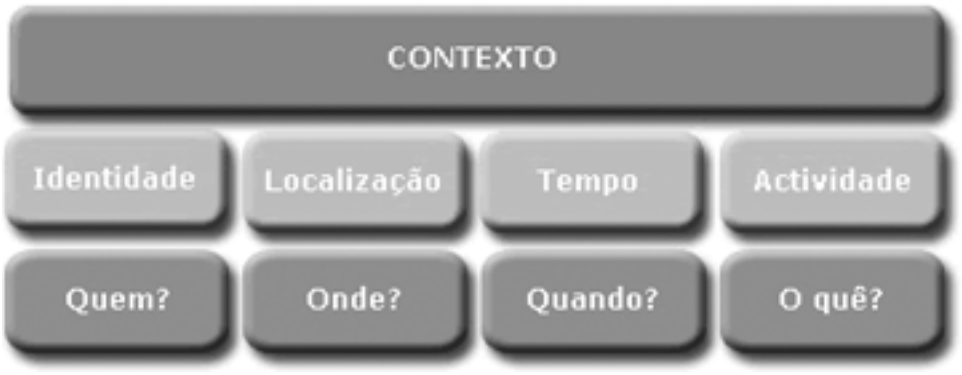

Figura 2.3: Caracterização do contexto

2.4 Sensores

O mundo físico apresenta um incrível conjunto de entradas de informação, incluindo acústica, imagem, movimento, vibração, calor, luz, pressão, entre outras. Tradicionalmente, quando

falamos em sensores, referimo-nos à utilização de uma colecção de instrumentos exteriores ao próprio dispositivo para obter informação específica [Estrin et al., 2002] ou à utilização de funcionalidades do próprio dispositivo [Pascoe, 1998]. Uma aplicação baseada no contexto deverá ter a capacidade de adquirir e analisar informação a partir de várias fontes [Yau et al., 2002]. Os sensores fazem parte dessas fontes, permitindo a interacção física entre o mundo e os dispositivos ubíquos [Estrin et al., 2002].

Dentro da computação ubíqua a área dos sensores é das que mais evoluiu ao longo dos anos. Actualmente existem trabalhos que utilizam os mais variados sensores. Desde sensores de localização através de acelerómetros [Farringdon et al., 1999], pedómetros [Lee and Mase, 2001], baseados em infravermelhos, como é o caso do Active Badge [Want and Hopper, 1992], por sinais de rádio frequência, como é o caso do Radar [Bahl and Padmanabhan, 2000] e ultra-sons [Priyantha et al., 2000], até à utilização de marcas capturadas por câmara digital [Aoki et al., 1999], existe uma grande variedade de sensores, tudo para se conseguir obter o máximo de informação do ambiente de forma a ter um contexto cada vez mais rico.

No entanto, não devemos restringir os sensores apenas a características físicas do ambiente. Se o objectivo dos sensores é fornecer informação sobre uma característica específica do contexto, também outros recursos dos dispositivos ubíquos devem ser considerados como tal. Quando falamos em recursos, estamos a falar de recursos de comunicação, como o Bluetooth, WiFi, GSM/GPRS, futuramente o UMTS e recursos de informação, como por exemplo a agenda, os processos, os ficheiros disponíveis, entre outros. Temos então sensores associados a hardware e sensores associados a software [Henricksen et al., 2002].

Capítulo 3

Estado da Arte

3.1 Modelos de Contexto e Aplicações

3.1.1 Context Toolkit

Desde 1993 que têm surgido várias definições do que é o contexto, mas só em 2000 apareceu o primeiro modelo de contexto, apresentado por Dey através do *Context Toolkit* [Dey and Abowd, 2000].

O objectivo desta ferramenta era fornecer alguma abstracção de modo a apoiar aplicações que até esse momento não utilizavam a informação contextual e permitir desse modo ter um modelo de contexto que pudesse ser utilizado por várias aplicações, sem a necessidade de criar um novo modelo sempre que se altera ou cria uma aplicação.

Esta ferramenta foi desenvolvida em Java e é constituída por três níveis de abstracção (Figura 3.1):

- ***Widgets***, que tem como função encapsular a informação de cada um dos elementos que compõem o contexto, como a localização e a actividade. Esta informação é obtida a partir dos sensores. Desta forma, as aplicações que utilizam a informação contextual não necessitam de lidar directamente sobre os sensores, apenas têm de saber qual o *widget* que tem a informação que necessitam e aceder à mesma. Se, por exemplo, tivermos um sensor de *GPS*, quem trata da comunicação e tratamento da informação (interpretação da informação em formato *NMEA*, por exemplo) é um *widget* específico que disponibilizará às aplicações valores como latitude e longitude.

- ***Aggregators***, permitem agrupar informação contextual obtida a partir dos *widgets* disponibilizando-a a entidades do contexto como os utilizadores e locais. Por exemplo, ao aceder ao *aggregator* associado aos utilizadores, ele pode devolver informação sobre a

actividade e localização deste. Esta informação é por sua vez obtida através do acesso a dois *widgets*, um que devolve a actividade do utilizador e outro que devolve a sua localização.

- ***Interpreters***, têm por missão analisar informação de contexto de baixo nível e inferir informação de contexto de alto nível. Por exemplo, a partir da informação relativa à identidade, localização e nível de som, podem inferir que está a decorrer uma reunião.

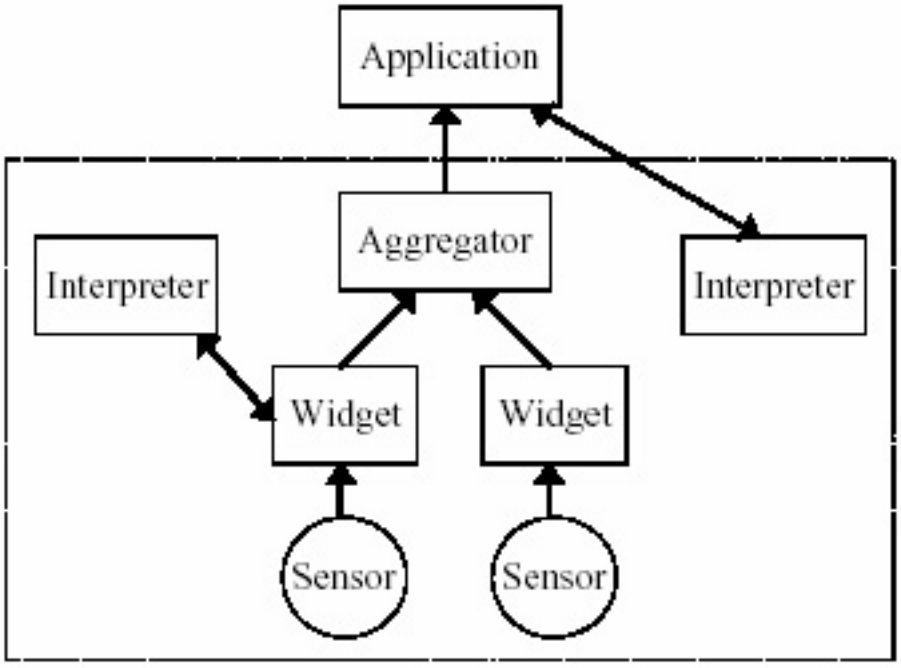

Figura 3.1: *Context Toolkit*[1]

Embora na figura anterior não seja explícito, pelas definições de cada um dos componentes do modelo podemos constatar que estão presentes três dimensões de contexto: Identidade, Localização e Actividade.

De forma a validar este modelo, foram desenvolvidos vários protótipos de aplicações que utilizam informação contextual, uma delas é o "*In/Out Board*"que apresenta quais os elementos do grupo de investigação se encontram dentro do edifício.

3.1.2 AURA

Não podemos falar de contexto em computação ubíqua sem referirmos o projecto *AURA* [Garlan et al., 2002]. Constatando-se que o recurso mais precioso num sistema computacional

[1]Imagem extraída de [Dey and Abowd, 2000]

já não é o processador, a memória, o disco ou a rede, mas sim a atenção humana, o projecto *AURA* ajuda a minimizar as distracções do utilizador, criando um ambiente que se adapta ao contexto do utilizador e às suas necessidades.

O nome do projecto está directamente relacionado com a sua essência, ter um ambiente onde as tarefas que o utilizador está a executar o acompanhem para onde quer que ele vá. Assim as duas características importantes do *AURA* são: o suporte da mobilidade do utilizador; protecção do utilizador de variações na disponibilidade dos recursos. Quando um utilizador se desloca de um ambiente para outro, o *AURA* tenta reconfigurar o novo ambiente para que o utilizador possa continuar a trabalhar nas tarefas iniciadas anteriormente. Como os recursos no ambiente alteram, o *AURA* tenta adaptar as tarefas a decorrer, reconfigurando ou substituindo serviços de forma a suportarem as alterações.

De forma a responder às características do *AURA*, foi criado um modelo de contexto que permitisse estruturar a informação relativa aos utilizadores, dispositivos, espaços físicos e rede [Judd and Steenkiste, 2002]. Este modelo é composto por entidades e por serviços que permitem a criação de relacionamentos entre entidades (Figura 3.2). Facilmente se constata que as dimensões Identidade e Localização fazem parte deste modelo de contexto.

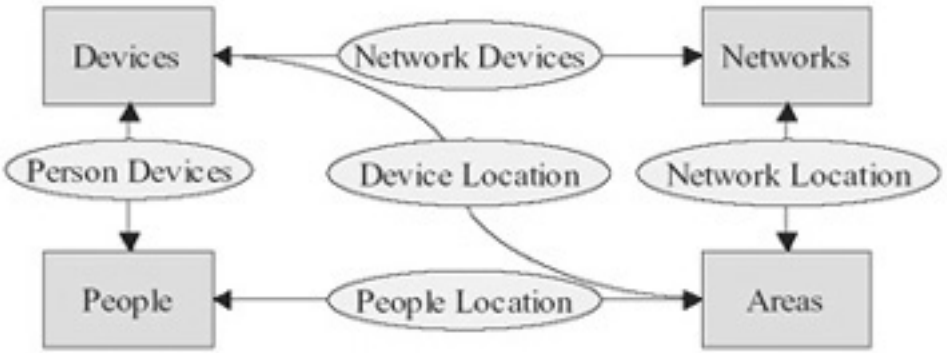

Figura 3.2: *AURA*[2]

3.1.3 Modelo de Contexto de Li

Pelos dois modelos apresentados anteriormente, conclui-se que a presença das dimensões Identidade e Localização são praticamente obrigatórias. No entanto, os mesmos apenas apresentam informação de alto nível acerca do modelo, não sendo visível de que forma é que a informação contextual é adquirida. Isto é, sabemos que o modelo tem informação sobre, por exemplo, a Localização, mas não nos é dito de que forma é adquirida essa informação. Outro

[2]Imagem extraída de [Judd and Steenkiste, 2002]

facto que não podemos esquecer é que o contexto não é só identidade de objectos e pessoas e respectivas localizações.

O modelo seguinte aborda estas duas questões, adicionando ainda o conceito de fusão de informação.

Seguindo os mesmos princípios de que uma aplicação que utilize informação contextual está interessada em saber a identidade de pessoas, suas actividades e localização, o modelo de Li [Li, 2003] apresenta ainda potenciais sensores para a aquisição da informação de contexto e uma solução para fazer a abstracção desta informação recorrendo à fusão da informação.

Se, por exemplo, a Localização pode ser obtida através de um conjunto variado de sensores (Figura 3.3), é necessário um nível intermédio de abstracção que permita combinar toda a informação para que as aplicações que acedem à informação de contexto recebam não a informação vinda directamente dos sensores, mas sim uma informação de alto nível (Figura 3.4).

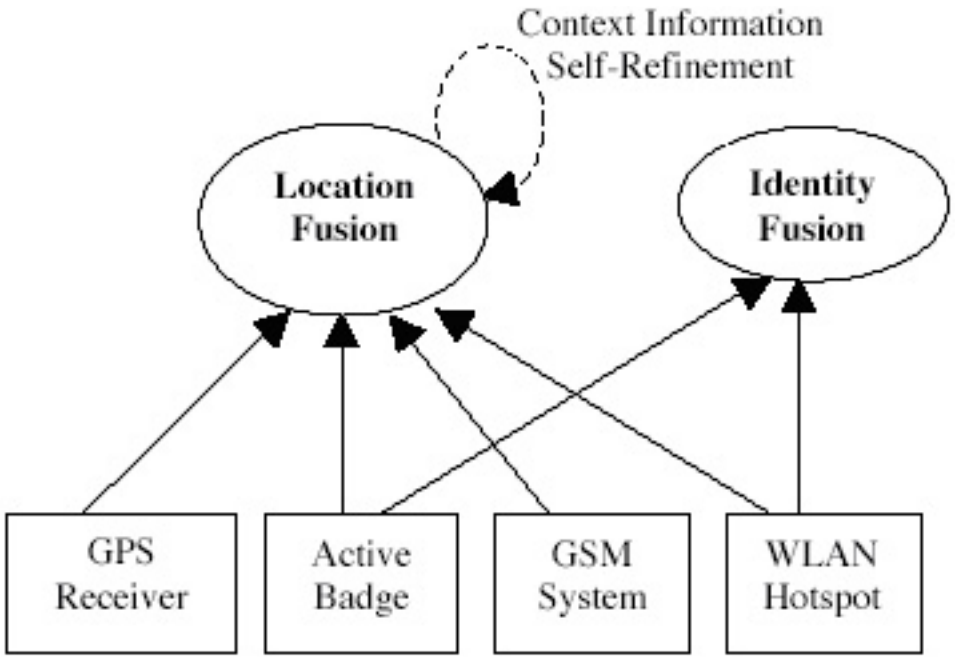

Figura 3.3: Sensores de Localização[3]

Neste modelo de contexto, além das duas dimensões de contexto comuns nos dois modelos anteriores (Identidade e Localização), são incluídas ainda mais duas dimensões (Tempo e Actividade), abrangendo assim o que se considera como as dimensões fundamentais num

[3]Imagem extraída de [Li, 2003]

modelo de contexto, com a mais valia de que parte da informação é já de alto nível, sendo inferida a partir dos sensores.

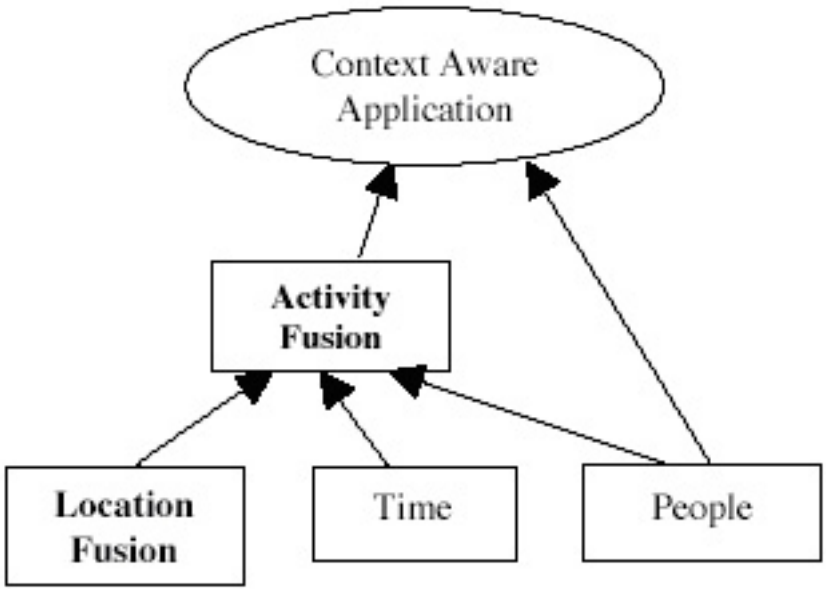

Figura 3.4: Fusão de informação de contexto[4]

De forma a validar este modelo, foi criado um protótipo de uma aplicação tipo *ICQ*, onde a informação de cada elemento do projecto e o estado actual do projecto incluindo documentos e reuniões agendadas, era difundida para todos os elementos do projecto. A informação relativa a cada elemento incluía a sua localização e os seus serviços de comunicação disponíveis. Com base na informação de contexto, foi criado um serviço de comunicação que determinava qual a acção a dar às chamadas recebidas, isto é, se eram bloqueadas, redireccionadas para o *voice mail*, ou se podiam ser atendidas pelo utilizador, de acordo com os serviços disponíveis inferidos pelo contexto e preferência do utilizador.

3.1.4 Context Modelling Language

Henricksen [Henricksen et al., 2002] apresentou em 2002 um primeiro modelo de contexto cujo objectivo visava a utilização de informação contextual em aplicações de comunicação. Estas aplicações necessitam de informação acerca dos utilizadores, dos dispositivos e canais de comunicação.

Com base nesse modelo, McFadden e os seus colegas apresentaram posteriormente uma ferramenta que tem como finalidade especificar formalmente os requisitos do contexto em

[4]Imagem extraída de [Li, 2003]

aplicações que necessitem de informação contextual, o *Context Modelling Language* (CML) [McFadden et al., 2004].

Esta ferramenta, baseada no conceito de Entidade - Relacionamento, permite definir quais as entidades de informação contextual necessárias e os tipos de informação ou factos importantes no relacionamento entre entidades. A grande mais valia desta representação é permitir observar, além das entidades envolvidas no contexto, os relacionamentos e dependências entre entidades (Figura 3.5).

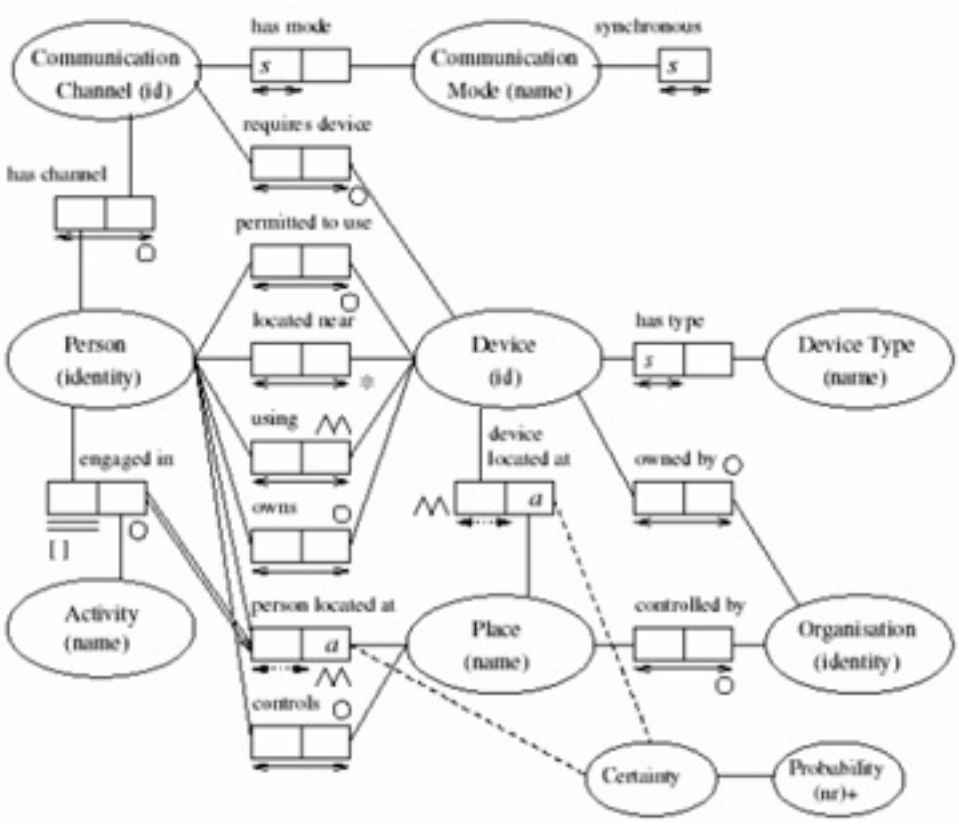

Figura 3.5: Modelo de contexto[5]

O modelo anterior (Figura 3.5) gerado utilizando o *CML* tem como objectivo a criação de uma aplicação de comunicação que auxilie os utilizadores na selecção do canal de comunicação mais apropriado para interacção com outros utilizadores. Embora neste caso o modelo de contexto seja muito específico, não sendo um modelo que se possa aplicar a todas as situações, este tipo de representação permite compreender que informação é relevante para o contexto e as associações entre as várias entidades do modelo.

[5]Imagem extraída de [McFadden et al., 2004]

3.1.5 Place Lab

Desenvolvido pela *Intel Research* desde 2003, o projecto *Place Lab* [Hightower et al., 2006], tem como objectivo adicionar a capacidade de localização aos dispositivos ubíquos, tanto para exterior como para o interior de edifícios.

O *Place Lab* é executado em dispositivos do dia-a-dia, como o computador portátil, o *PDA* ou os telemóveis, com diferentes arquitecturas e sistemas operativos.

Através do *Place Lab*, o dispositivo prevê a sua localização com base na análise de sinais de rádio por ele recebidos, quer vindo de pontos de acesso *Wi-Fi*, quer através do sinal emitido pelas células *GSM*, mais próximos.

Embora o *GPS* seja a tecnologia de localização com mais sucesso, uma vez que permite ter uma cobertura mundial via satélite, tem a desvantagem de no interior dos edifícios a sua recepção de sinal ser muito fraca, dificultando assim a obtenção de coordenadas geográficas. Por esta razão no projecto *Place Lab*, foi delegada para segundo plano, face ao *Wi-Fi* e o *GSM*. A escolha destas duas tecnologias como forma de determinar a localização está relacionada com o facto de que cada vez mais se estão a tornar ubíquas nas áreas populacionais.

A arquitectura do *Place Lab* [LaMarca et al., 2005] consiste em três elementos: os emissores de radio existentes, um sistema de informação que armazena a localização destes emissores e os utilizadores do sistema de previsão da localização actual (Figura 3.6).

Segundo Hightower [Hightower et al., 2006], a previsão da localização do utilizador é feita localmente, por questões de privacidade. Relativamente à precisão deste sistema, esta varia de acordo com o número de sinais recebidos pelo dispositivo do utilizador. No melhor dos casos, a precisão da previsão da localização pode ficar entre os 15 a 20 metros, desde que sejam recebidos pelo menos três sinais de radio numa janela temporal de 10 segundos. Em zonas onde a densidade de pontos de acesso *Wi-Fi* é menor, esta precisão pode baixar até aos 30 metros. Embora com precisão inferior ao *GPS*, a combinação dos sinais de *Wi-Fi* e *GSM*, permite ter uma melhor precisão do que utilizando unicamente os sinais de *GSM* de um único operador, cuja precisão pode variar entre os 94 e os 196 metros.

O grande problema deste sistema é que, ao contrário do que acontece com o *GPS* - onde as coordenadas referem um determinado local único no globo terrestre - no caso dos sinais de *Wi-Fi* e *GSM* essa geo-referenciação não existe, daí que seja necessário fazer um trabalho prévio de mapeamento destes sinais, associando-os a coordenadas, o que não é de todo, muito atractivo. Por exemplo, para mapear apenas a área metropolitana da cidade de Seattle, foram percorridos mais de 4350 kilómetros. Embora os utilizadores do *Place Lab* possam aceder à base de dados da *Wigle.net* [6], o sistema de informação do *Place Lab* terá que estar sempre

[6]Disponível em http://www.wigle.net, contém mais de 7 milhões de pontos de acesso *Wi-Fi* geo-referenciados

em constante actualização, uma vez que rapidamente são adicionados ou removidos pontos de acesso, assim como células *GSM*.

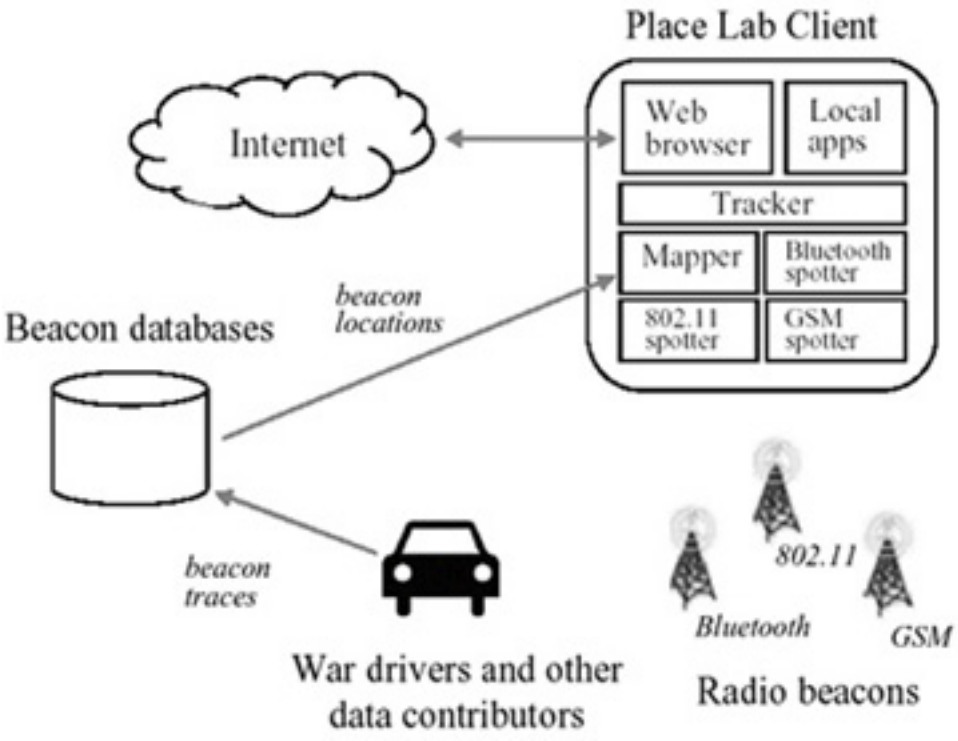

Figura 3.6: Arquitectura do *Place Lab*[7]

3.1.6 ContextPhone

O *ContextPhone* [Raento et al., 2005] é uma plataforma para *Smartphones* com sistema operativo *Symbian OS* que, aproveitando o facto destes dispositivos acompanharem o utilizador, faz a aquisição de informação contextual disponibilizando-a a outras aplicações.

Nesta plataforma (Figura 3.7), o contexto é definido por quatro tipos de informação: localização, interacção do utilizador, comunicação e ambiente físico.

Ao nível da localização, a informação é obtida a partir da identificação da célula *GSM* activa e recorrendo a um receptor de *GPS* externo ao telemóvel. Embora a localização baseada na identificação da célula *GSM* possa ter uma precisão na ordem dos quilómetros, o objectivo da sua utilização prende-se com o facto de existirem situações em que não existe cobertura *GPS*.

A interacção do utilizador é obtida a partir das aplicações activas do telemóvel, estado do telemóvel (se está activo ou em modo de espera), do perfil do alarme do telemóvel (geral,

[7]Imagem extraída de [LaMarca et al., 2005]

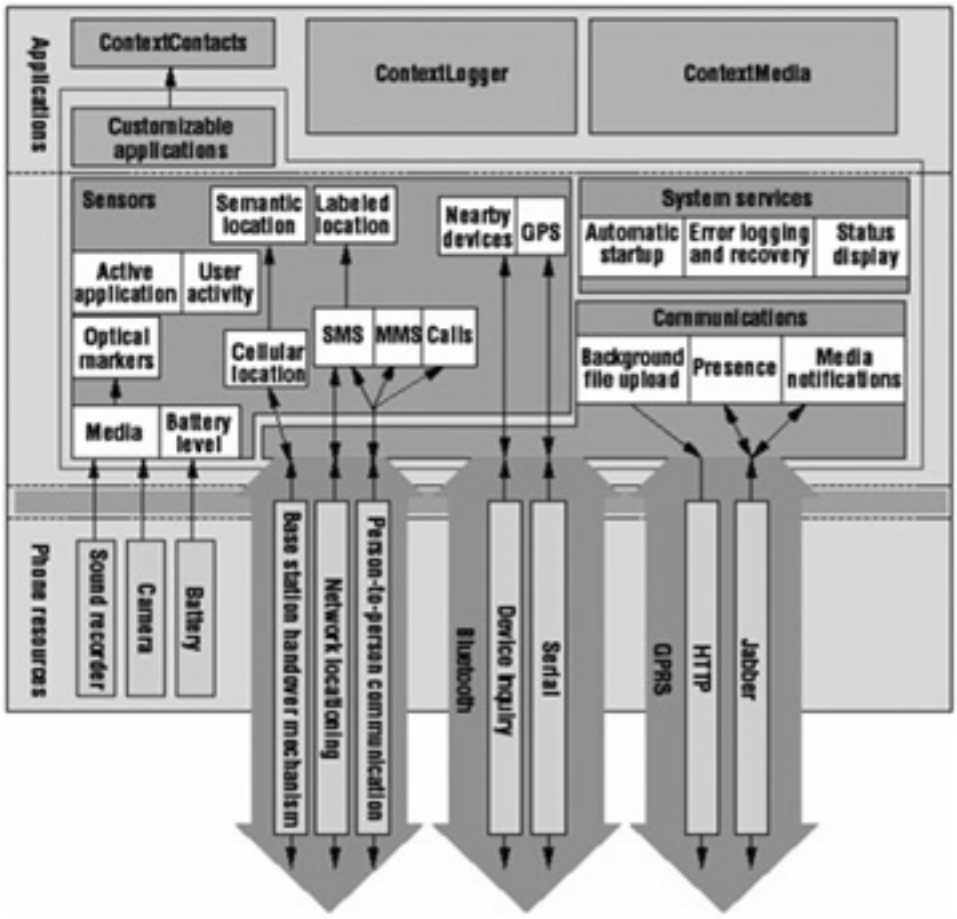

Figura 3.7: Arquitectura do *ContextPhone*[8]

silencio, reunião), do estado da bateria e da informação multimédia adquirida (por exemplo, recorrendo à câmara fotográfica que já equipa alguns telemóveis). A informação relativa à comunicação é obtida a partir das chamadas realizadas e recebidas, das mensagens SMS enviadas e recebidas, e do conteúdo das próprias SMS.

A informação relativa ao ambiente físico é obtida através da descoberta de dispositivos *Bluetooth*, rede *Bluetooth* disponível e reconhecimento de marcas utilizando a câmara fotográfica.

Embora com uma terminologia diferente ao nível da definição de contexto, o objectivo é o mesmo: identificar as Actividades, a Localização e as Identidades (pessoas e dispositivos).

Uma aplicação que foi desenvolvida nesta plataforma recorrendo à informação de contexto é o *ContextContacts*, que é uma extensão à lista de contactos que é comum nos telemóveis, adicionando a seguinte informação para cada um dos contactos: localização anterior ou actual,

[8]Imagem extraída de [Raento et al., 2005]

o estado de utilização do telemóvel e o perfil do alarme do telemóvel. Esta informação é partilhada entre os vários contactos para que cada utilizador tenha conhecimento do estado actual dos seus contactos, de forma a facilitar a decisão sobre se é o momento oportuno para entrar em contacto com um membro do grupo.

3.2 Modelos e Algoritmos de Aprendizagem

Ao contrário da área de Computação Ubíqua, a área de Inteligência Artificial conta já com longos anos de evolução. Devido a esta maturidade, temos hoje à disposição um leque alargado de algoritmos de aprendizagem, com as mais diversas características e para as mais variadas aplicações.

Embora o objectivo desta dissertação seja avaliar os vários algoritmos de aprendizagem em sistemas *context-awaness*, estudar todos os algoritmos e suas implementações seria uma tarefa com elevado custo temporal, o que não se adequa com o disponível para o desenvolvimento desta dissertação.

Por esta razão foi necessário fazer opções relativamente a quais as ferramentas de análise inteligente de dados e algoritmos de aprendizagem deveriam então ser utilizados. Como as aplicações *context-awareness* são tipicamente compostas por um conjunto de características do momento actual do contexto e por um conjunto de acções que são executadas em função desse contexto, é útil encontrar similaridades e aprender padrões recorrentes, quer seja para prever as correspondentes classes existentes através de modelos de classificação, para descobrir associações entre os vários atributos, ou então para descobrir grupos de instâncias com características semelhantes. Assim, a opção pela ferramenta *Waikato Environment for Knowledge Analysis (WEKA)* [9], parece-nos a mais adequada, uma vez que possui um conjunto alargado de algoritmos de classificação, associação e *clustering*, sendo possível a sua utilização para a avaliação e consequentes conclusões referentes aos que mais se adequam a este tipo de problemática.

Nesta secção, vão então ser apresentados os vários modelos de algoritmos disponíveis no *WEKA*, tendo como referência o livro *Data Mining - Practical Machine Learning Tools and Techniques* [Witten and Frank, 2005], que documenta os mesmos dentro desta plataforma. Os vários conceitos aqui apresentados estão assim de acordo com o referido documento.

[9] Ferramenta desenvolvida pelo Computer Science Department da Universidade de Waikato, Nova Zelândia e disponível em http://www.cs.waikato.ac.nz/ml/weka

3.2.1 Modelos de Classificação

Tabelas de Decisão

A forma mais simples e também mais rudimentar de representar a informação é através das tabelas de decisão, onde a representação dos atributos de saída é feito da mesma forma que os atributos de entrada. No exemplo seguinte (Tabela 3.1) é apresentada uma tabela de decisão cujo objectivo é determinar em que condições meteorológicas se deve ou não jogar um determinado jogo. Neste caso em particular, temos quatro atributos de entrada: *outlook, temperature, humidity* e *windy*, e um atributo de saída que corresponde a determinar se devemos ou não jogar.

O tamanho das tabelas de decisão está directamente associado ao número de atributos de entrada que se julgam ser necessários para determinar o atributo de saída sem afectar o resultado da decisão final. Assim, a criação de uma tabela de decisão pressupõe a decisão sobre quais os atributos que se julgam relevantes para o problema em questão.

Outlook	**Temperature**	**Humidity**	**Windy**	**Play**
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Tabela 3.1: Exemplo de uma tabela de decisão[10]

A partir da tabela de decisão, é criado um conjunto de regras cujo objectivo é serem interpretadas em sequência e segundo a ordem pela qual são apresentadas, sendo que as regras são exclusivas, o que significa que no caso de uma das primeiras regras ser verdadeira, as seguintes já não são verificadas. O exemplo seguinte mostra exactamente um conjunto de regras obtidas a partir da tabela de decisão anterior.

[10]Tabela extraída de [Witten and Frank, 2005]

```
If outlook = sunny and humidity = high then play = no
If outlook = rainy and windy = true then play = no
If outlook = overcast then play = yes
If humidity = normal then play = yes
If none of the above then play = yes
```

De forma a garantir bons resultados por parte deste modelo de aprendizagem, é necessário garantir que a execução das regras segue o anterior pressuposto. Se a ordem das regras anteriores for alterada ou se estas forem vistas de forma individual, é muito provável que a classificação obtida seja incorrecta. Olhando apenas para a regra `If humidity = normal then play = yes`, podemos verificar na tabela anterior que existe pelo menos um exemplo errado, daí que este pressuposto não deva ser negligenciado.

Árvores de Decisão

Este tipo de modelo de aprendizagem segue uma representação em árvore, composta por um nó inicial, a *raiz*, a partir da qual vão sendo associados outros nós descendentes ou *filhos*, de acordo com os atributos que vão sendo testados, até encontrarmos um nó sem descendente, nó *folha*, que representa o resultado do processo de classificação. Pelas características de representação destes modelos, diz-se que seguem uma aproximação do tipo *divide-and-conquer*, uma vez que à medida que se vai descendo na árvore em direcção às folhas, o número de possibilidades de resolução do problema vai diminuindo até se chegar à única solução possível, que torna a classificação correcta.

Os nós de uma árvore de decisão servem de teste a um atributo em particular. Normalmente, o teste aos nós compara um valor de um atributo com uma constante. No entanto, algumas árvores de decisão comparam dois atributos com outro, ou mesmo funções de um ou mais atributos. Os nós folha atribuem uma classificação, aplicada a todos os nós que se encontram ligados a este. Para classificar uma instância desconhecida, é percorrida a árvore de forma descendente de acordo com os valores dos atributos testados nos sucessivos nós e, quando se atinge o nó folha, a instância é classificada de acordo com a classe associada a esse nó.

No caso do atributo a testar num nó ser nominal, o número de filhos desse nó é normalmente o número de valores possíveis para esse atributo (Figura 3.8).

Se o atributo for numérico, o teste ao nó normalmente determina se o valor é maior ou menor que um valor predeterminado, existindo assim apenas duas hipóteses de escolha (Figura 3.9). Alternativamente, pode existir uma divisão em três, *maior que*, *igual a* e *menor que*.

[11] Imagem extraída de [Witten and Frank, 2005]
[12] Imagem extraída de [Witten and Frank, 2005]

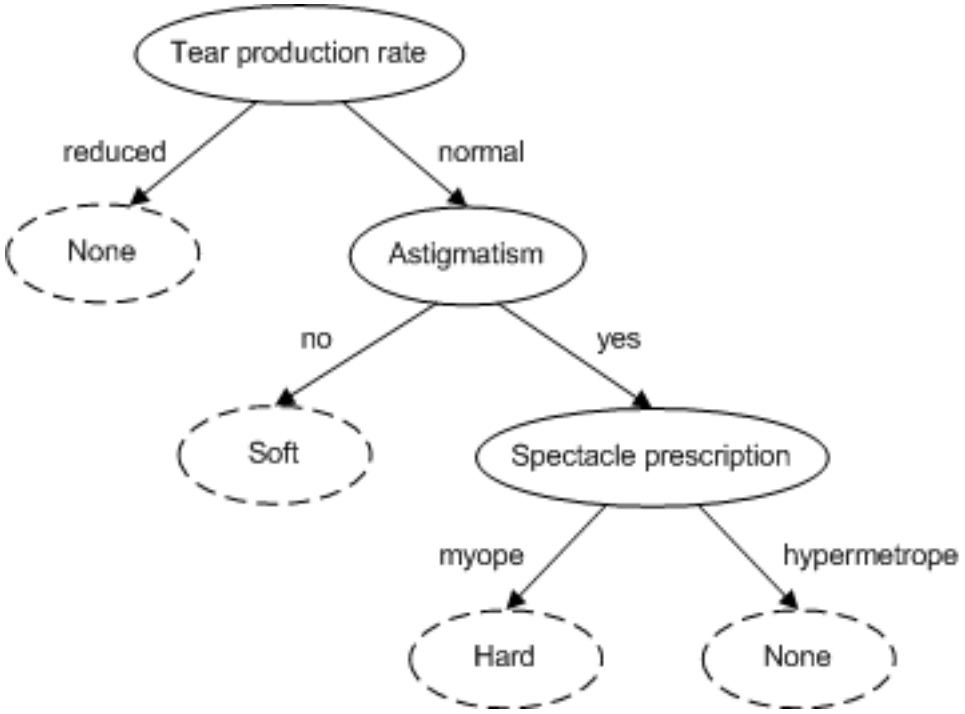

Figura 3.8: Árvore de decisão com atributos nominais[11]

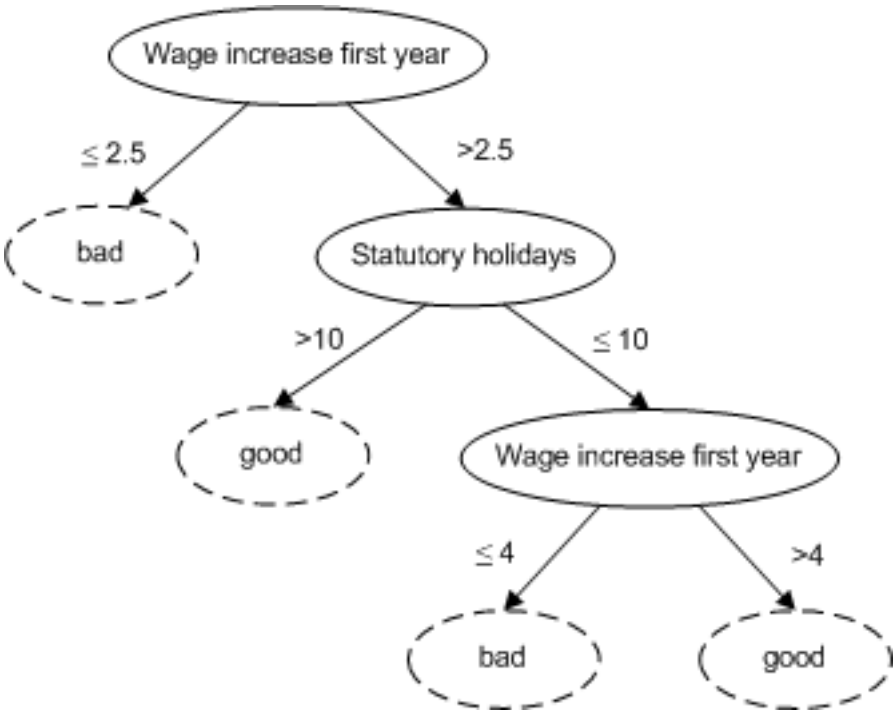

Figura 3.9: Árvore de decisão com atributos numéricos[12]

Um problema que se coloca neste tipo de modelo é quando temos atributos com valores nulos. Se, no exemplo a testar, aparecer um atributo sem um valor associado, o teste não pode ser realizado. Nas situações em que um atributo nulo tem significado, uma forma de solucionar o problema é considerar o nulo como uma hipótese para esse atributo. Se por outro lado, o facto de ser nulo se deve a uma anomalia e não é permitido que este o seja, então a solução mais simples é associar a este o valor mais predominante no conjunto de treino para esse atributo.

Outro problema com o qual temos que lidar nas árvores de decisão, é o *overfitting*. Estes algoritmos tendem a descrever muito bem as instâncias utilizadas no treino, adaptando-se ao máximo a estas, no entanto não são capazes de generalizar, o que pode levar a erros de classificação em instâncias com ruído. Pelo facto de estarem tão adaptadas às instâncias de treino, qualquer situação ligeiramente diferente pode originar um erro na classificação.

Na tabela 3.2 encontram-se os vários algoritmos de classificação baseados em árvores de decisão disponíveis no *WEKA*.

Nome	**Função**
ADTree	Árvores de decisão alternadas
DecisionStump	Árvore de decisão de apenas um nível
Id3	Utilização do algoritmo básico de *divide-and-conquer*
J48	Utilização do algoritmo *C4.5*
LMT	Utilização do modelo logístico
M5P	Utilização do modelo *M5*
NBTree	Utilização do classificador *Naive Bayes*
RamdomForest	Construção aleatória de árvores de decisão
RandomTree	Construção de uma árvore de decisão, dado um número aleatório de atributos para cada nó
REPTree	Árvore de decisão com *reduced-error pruning*
UserClassifier	Criação de uma árvore de decisão pelo utilizador

Tabela 3.2: Algoritmos de Classificação através de Árvores de Decisão

Regras de Classificação

Uma alternativa muito popular às árvores de decisão são as regras de classificação. As regras são compostas pelas pré-condições, ou *antecedentes*, e pelas conclusões, ou *consequentes*. Um antecedente é um conjunto de testes idênticos aos nós das árvores de decisão e os consequentes dizem a que classe a instância pertence quando aplicada essa regra. Normalmente, os ante-

cedentes estão associados por um `E` lógico, o que implica que a regra apenas será verdadeira se e só se todos os testes aos antecedentes forem verdadeiros.

```
If petal length < 2.45 then Iris setosa
If sepal width < 2.45 and petal length < 4.55 then Iris versicolor
```

É relativamente simples obter um conjunto de regras a partir de uma árvore de decisão, fazendo corresponder a cada folha da árvore uma regra. Os antecedentes de uma regra incluem todos os nós de teste que formam o caminho entre a raiz da árvore e a folha, enquanto que o consequente da regra é a classe associada à própria folha. Como a cada folha está associada uma regra, para cada classe existe apenas uma regra, não existindo assim qualquer tipo de ambiguidade. Por esta razão, a ordem pelas quais são executadas as regras é irrelevante. No entanto, obter as regras desta forma torna estas mais complexas do que as obtidas pelo próprio modelo de árvore de decisão, uma vez que podem conter testes redundantes pelo simples facto de não ser feita qualquer optimização, ao contrário das regras derivadas da árvore de decisão.

Umas das razões pelas quais as regras são populares tem a ver com o facto de que cada regra dá a entender que representa um conjunto independente de conhecimento. Daí afirmar-se que utiliza a técnica *divide-and-conquer*, sendo possível adicionar novas regras ao conjunto actual sem perturbar as que já existem, enquanto que numa árvore de decisão implicaria uma nova reestruturação desta. No entanto, se esta independência for ilusória significa que ao adicionarmos novas regras podemos estar a comprometer os resultados de classificação, uma vez que diferentes regras podem originar diferentes classificações para a mesma instância.

Como acontece com as árvores de decisão, também nos algoritmos de regras de classificação, existe a possibilidade de *overfitting*.

Os algoritmos de regras de classificação disponíveis no *WEKA* encontram-se descritos na tabela 3.3.

Aprendizagem baseada em Instância

A forma mais simples de aprendizagem é com o recurso à memorização. Uma vez memorizado o conjunto de instâncias de treino, sempre que seja necessário classificar uma nova instância, é feita uma pesquisa nas instâncias de treino de forma a encontrar a que mais se aproxima desta.

De notar que esta forma de representação do conhecimento extraído a partir do conjunto de instâncias é completamente diferente dos modelos anteriores. Neste caso, é apenas necessário guardar as próprias instâncias e operar com estas relacionado-as com as novas instâncias que pretendemos classificar. Ao contrário dos modelos anteriores, em que é necessário inferir um conjunto de regras ou uma árvore de decisão a partir das instâncias de

Nome	Função
ConjunctiveRule	Criação de simples regras conjuntivas
DecisionTable	Criação de uma tabela de decisão
JRip	Criação de regras rápidas utilizando o algoritmo *RIPPER*
M5Rules	Criação de regras a partir de árvores obtidas a partir de modelos *M5*
Nnge	Criação de regras utilizando o algoritmo *Nearest-neighbor*
OneR	Utilização do classificador *1R*
Part	Criação de regras a partir de árvores de decisão obtidas com o algoritmo *J4.8*
Prism	Utilização de uma algoritmo de cobertura simples
Ridor	Criação de regras a partir do método *Ripple-down*
ZeroR	Predição da classe predominante (se for nominal) ou do valor médio (se for numérico)

Tabela 3.3: Algoritmos de Classificação através de Regras

treino e guardar os modelos gerados, neste tipo de modelo as próprias instâncias representam o conhecimento.

Nos modelos de aprendizagem baseados em instância, o verdadeiro trabalho é feito apenas quando se pretende classificar uma nova instância, sendo esta a principal diferença entre este e os outros métodos de aprendizagem. A aprendizagem é feita nesta fase e não previamente.

Cada nova instância a classificar é comparada com as existentes em memória através de uma métrica de distância e é associada a esta nova instância a classe atribuída à instância mais próxima desta que se encontra em memória. Este método de classificação é chamado de *nearest-neighbor*, podendo por vezes ser utilizado *k-nearest-neighbor*, cujo objectivo é determinar as k instâncias mais próximas da nova instância. Em situações onde há grande probabilidade de existirem instâncias com ruído, originando assim um erro de classificação, a opção pelo *k-nearest-neighbor* pode ser um bom compromisso. Como a determinação da classe da nova instância tem por base as k instâncias mais próximas desta, a classe que lhe é atribuída é aquela que obteve maior número de ocorrências.

Determinar a distância entre atributos numéricos não representa qualquer dificuldade, sendo neste caso utilizada por norma a distância Euclidiana (Equação 3.1) para determinar as distâncias entre duas instâncias. A questão pode surgir quando temos atributos nominais. Neste caso a solução mais imediata passa por atribuir a distância de zero no caso dos atributos serem idênticos e a distância de um para os restantes casos.

$$\sqrt{(a_1^{(1)} - a_1^{(2)})^2 + (a_2^{(1)} - a_2^{(2)})^2 + ... + (a_k^{(1)} - a_k^{(2)})^2} \tag{3.1}$$

Uma das desvantagens apontadas a este tipo de modelo de classificação é o facto de considerar à partida que todos os atributos têm idêntica importância, o que nem sempre é verdade em casos reais. Como forma de contornar esta questão, é normal que a métrica de distância reflicta o grau de importância dos atributos, utilizando-se para tal pesos associados a cada atributo (Equação 3.2). Isto implica o conhecimento de quais os atributos mais significativos.

$$\sqrt{w_1^2(a_1^{(1)} - a_1^{(2)})^2 + w_2^2(a_2^{(1)} - a_2^{(2)})^2 + ... + w_k^2(a_k^{(1)} - a_k^{(2)})^2} \tag{3.2}$$

Um cuidado a ter neste tipo de modelo de aprendizagem é a quantidade de instâncias de treino. Determinar a distância entre grandes volumes de instâncias pode ser um processo lento e implicar um consumo exagerado de espaço em memória, uma vez que cada nova instância tem que ser testada com todo o conjunto de instâncias de treino. Existem no entanto técnicas mais elaboradas para evitar o teste de todas as instâncias.

Para os modelos de aprendizagem baseados em instâncias, o *WEKA* disponibiliza os algoritmos descritos na tabela 3.4.

Nome	**Função**
IB1	Utilização do algoritmo *nearest-neighbor*
IBk	Utilização do algoritmo *k-nearest-neighbor*
KStar	Utilização do algoritmo *nearest-neighbor* com a função de distância generalizada
LWL	Utilização de pesos nos atributos

Tabela 3.4: Algoritmos de Classificação baseados em Instâncias

Redes Bayesianas

Os modelos de redes bayesianas fazem as suas predições com base numa estimativa da probabilidade. Para cada valor de classe, estimam a probabilidade de determinada instância pertencer a essa classe. O que é estimado é a distribuição de uma probabilidade condicional dos valores dos atributos de uma classe, dados os valores dos outros atributos (baseando-se na Regra de Bayes).

$$P(A|B) = \frac{P(A \bigcap B)}{P(B)} \tag{3.3}$$

Mas ter apenas os valores das distribuições de probabilidade pode não ser a melhor forma de entender o modelo de aprendizagem, daí que os mesmos sejam representados de forma gráfica. Por esta razão se dá o nome de redes bayesianas. A partir das distribuições de probabilidade é criada uma rede de nós, um para cada atributo, ligados de forma direccional, traduzindo-se assim numa rede sem ciclos. A figura 3.10 mostra um desses exemplos.

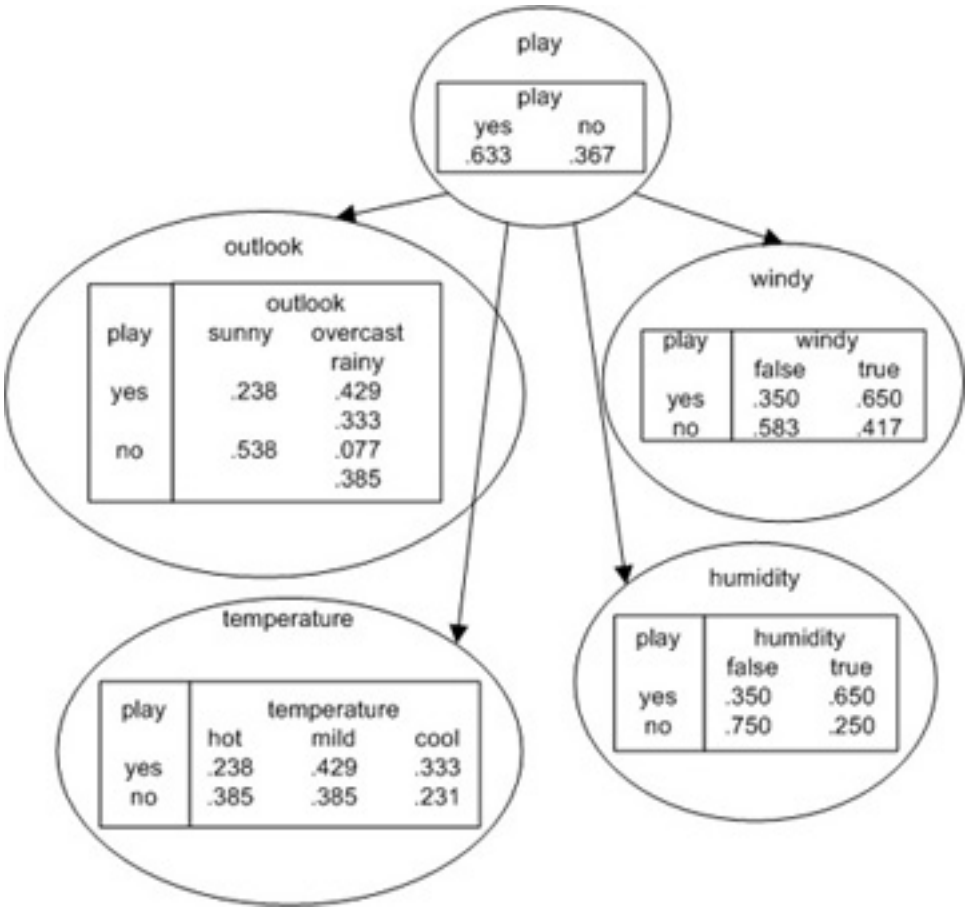

Figura 3.10: Exemplo de uma Rede Bayesiana[13]

Como podemos verificar, foi criado na rede bayesiana um nó para cada um dos quatro atributos de entrada *outlook*, *temperature*, *humidity*, *windy* e para o atributo de saída *play*. Dentro de cada nó existe uma tabela que define a distribuição de probabilidade que é utilizada para prever as probabilidades de classificação de determinada instância. Estas tabelas estão divididas em duas partes. A da esquerda contém os valores para o atributo *play*, enquanto que a da direita contém a respectiva correspondência de probabilidades para cada valor dos atributos.

Com base nestas tabelas é determinada a probabilidade para cada classe de uma instância, bastando para isso multiplicar todas as probabilidades que vão sendo obtidas ao longo da rede de acordo com os valores de cada atributo. Por exemplo, considerando uma instância com os seguintes atributos, onde o objectivo é determinar a probabilidade da classificação ser *play=yes*:

[13]Imagem extraída de [Witten and Frank, 2005]

```
outlook=rainy, temperature=cool, humidity=true, windy=true
```

Com base na figura 3.10, obtemos as probabilidades 0.333 do atributo *outlook*, 0.333 do *temperature*, 0.650 do *humidity* e 0.650 do *windy*, multiplicando estas probabilidades obtemos o valor de 0.04685. Fazendo o mesmo calculo para a classificação *play=no*, obtemos o valor de 0.00927. Como o que se pretende é a probabilidade condicional, é necessário normalizar estas, dividindo os valores obtidos pela soma das duas. Assim a probabilidade de *play=yes* é de 0.8348 e a de *play=no* é de 0.1652, que somadas têm obrigatoriamente que dar 1, uma vez que só existem estas duas classes possíveis.

Uma característica interessante deste modelo de aprendizagem é a criação de novos atributos em adição aos que já existem, conhecidos por atributos escondidos (pois não é possível observar os seus valores), que representam características ocultas do problema, sendo uma boa forma de melhorar a classificação. No entanto, implica que o processo de aprendizagem e de classificação fiquem mais complexos e por consequência mais lentos.

Dentro da plataforma *WEKA* temos à disposição vários algoritmos de aprendizagem com base no modelo de redes bayesianas, os quais se encontram na tabela seguinte (Tabela 3.5):

Name	**Function**
AODE	Estimador médio de uma dependência
BayesNet	Redes bayesianas
ComplementNaiveBayes	Criação de uma *Naive Bayes* complementar
NaiveBayes	Criação de uma rede probabilística *Naive Bayes*
NaiveBayesMultinomial	Rede *Naive Bayes* multinominal
NaiveBayesSimple	Rede *Naive Bayes* simples
NaiveBayesUpdateable	Rede *Naive Bayes* incremental
LBR	Regras obtidas pelo algoritmo *Lazy Bayesian*

Tabela 3.5: Algoritmos de Classificação Bayesianos

Funções

O *WEKA* disponibiliza ainda um conjunto de algoritmos de aprendizagem agrupados numa categoria, apelidada de Funções. Estes algoritmos têm em comum o facto dos seus modelos poderem ser descritos de forma natural através de equações matemáticas, ao contrário dos modelos baseados em árvores de decisão e regras, que por norma não têm uma formulação matemática.

Dos algoritmos disponíveis nesta categoria, quatro deles implementam a regressão linear (Equação 3.4): *SimpleLinearRegression*, *LinearRegression*, *LeastMedSq* e *PaceRegres-*

sion. Este último cria o modelo de regressão linear utilizando uma nova técnica de regressão, *Pace*, especialmente útil quando temos muitos atributos, uma vez que determina quais os menos significativos que podem ser retirados do modelo.

Uma vez que estamos a lidar com equações matemáticas, os algoritmos baseados em regressão linear não suportam valores nulos e só aceitam atributos numéricos.

O princípio da regressão linear passa por expressar a classe como uma combinação linear de atributos, tendo estes pesos associados, calculados a partir das instâncias de treino:

$$x = w_0 + w_1 a_1 + w_2 a_2 + ... + w_k a_k \tag{3.4}$$

onde x é a classe; a são os valores dos atributos; e w os respectivos pesos.

A regressão linear é uma excelente abordagem para classes e atributos numéricos. No entanto, por ser um modelo linear, tem a desvantagem de, no caso dos atributos não serem linearmente dependentes, o modelo não se ajustar muito bem às instâncias, uma vez que aproxima o resultado do treino à menor diferença média quadrática.

Além da regressão linear, existem nesta categoria três algoritmos que implementam o modelo através da regressão logística. São eles o *Logistic*, o *SimpleLogistic* e o *RBFNetwork*, embora este último utilize a regressão linear em vez da regressão logística, nas situações em que os atributos são numéricos. A regressão logística é um modelo estatístico de regressão utilizado quando os atributos são binários. A função logística é dada pela seguinte equação,

$$Pr[Y_i = 1|X] = \frac{\exp^{(\alpha+\beta_1 x_{1,i}+...+\beta_k x_{k,i})}}{1+\exp^{(\alpha+\beta_1 x_{1,i}+...+\beta_k x_{k,i})}} \tag{3.5}$$

ou de forma equivalente, pela equação,

$$Pr[Y_i = 1|X] = \frac{1}{1+\exp^{(-\alpha-\beta_1 x_{1,i}-...-\beta_k x_{k,i})}} \tag{3.6}$$

No caso da regressão linear, para determinar se a instância pertence ou não à classe, atribui-se respectivamente ao resultado o valor de 1 (Equação 3.7) ou 0 e verifica-se qual a probabilidade mais elevada. O problema surge quando o valor das probabilidades sai fora da gama 0 e 1. Assim, para evitar estas situações, a regressão logística permite ter resultados entre menos infinito e mais infinito, uma vez que transforma a variável original,

$$Pr[1|a_1, a_2, ..., a_k] \tag{3.7}$$

numa função linear,

$$log\frac{(Pr[1|a_1, a_2, ..., a_k])}{(1 - Pr[1|a_1, a_2, ..., a_k)} \tag{3.8}$$

Como resultado, o cálculo da probabilidade para determinar se a instância de teste pertence a essa classe é dado pela seguinte equação (Equação 3.9):

$$Pr[1|a_1, a_2, ..., a_k] = \frac{1}{1 + \exp^{(-w_0 - w_1 a_1 - ... - w_k a_k)}} \tag{3.9}$$

onde a são os valores dos atributos; e w os respectivos pesos.

Uma abordagem diferente de aprendizagem é determinar o hiperplano que separa as instâncias em diferentes classes. No caso de termos apenas duas classes, se as instâncias puderem ser perfeitamente separadas em dois grupos através do hiperplano, diz-se que são linearmente separáveis, sendo que o hiperplano é definido pela equação:

$$w_0 a_0 + w_1 a_1 + ... + w_k a_k = 0 \tag{3.10}$$

onde a são os valores dos atributos e w os respectivos pesos que definem o hiperplano.

Ao hiperplano resultante da separação das classes, dá-se o nome de **perceptrão**. A figura seguinte demonstra um perceptrão sobre a forma de rede neuronal (Figura 3.11):

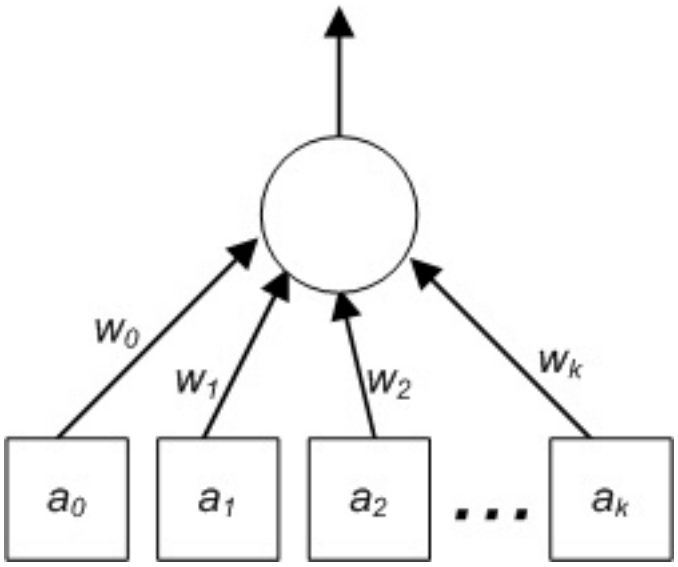

Figura 3.11: Representação do perceptrão sobre a forma de rede neuronal[14]

Dentro da categoria Funções do *WEKA*, estão disponíveis três algoritmos que fazem uso do perceptrão, são eles o *VotedPerceptron*, o *Winnow* e o *MultilayerPerceptron*.

Por último, temos o *SMO* e o *SMOreg*, que implementam o algoritmo de optimização sequencial (*SMO*), para o treino de um classificador cujo modelo é baseado em Máquinas de Vectores de Suporte (*SVM*).

[14]Imagem extraída de [Witten and Frank, 2005]

O classificador *SVM* também recorre ao hiperplano e é com base neste que são definidos os vectores de suporte, que mais não são, os vectores mais próximos do hiperplano.

Supondo que temos apenas duas classes linearmente separáveis, o *SVM* é o hiperplano que separa as mesmas, de forma a classificar correctamente todas as instâncias e posicionado-se espacialmente à máxima distância entre as duas classes. Esta distância máxima é denominada de margem máxima de separação. Vejamos a seguinte figura (Figura 3.12) que ilustra onde o hiperplano deve ser posicionado, por forma a maximizar esta margem. As duas classes estão representadas por círculos com e sem preenchimento, respectivamente.

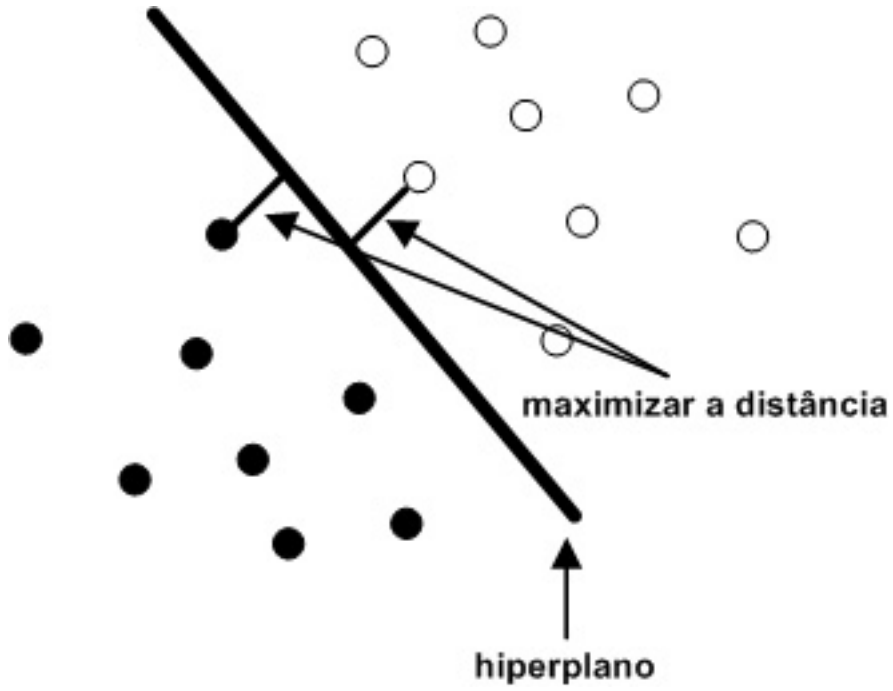

Figura 3.12: Hiperplano e margem máxima de separação

As instâncias que se encontram mais próximas da margem máxima de separação, são os vectores de suporte. Para cada classe há pelo menos um vector de suporte (Figura 3.13).

O mais importante é o facto de que o conjunto de vectores de suporte define a margem máxima de separação para o problema em questão, uma vez que é relativamente simples construir esta, a partir dos vectores de suporte. As restantes instâncias de treino que não fazem parte do conjunto de vectores de suporte, como são irrelevantes para o modelo, podem inclusivamente ser eliminadas sem alterar o hiperplano.

Os atributos nulos, nestes algoritmos, são substituídos e os atributos nominais transformados em valores binários.

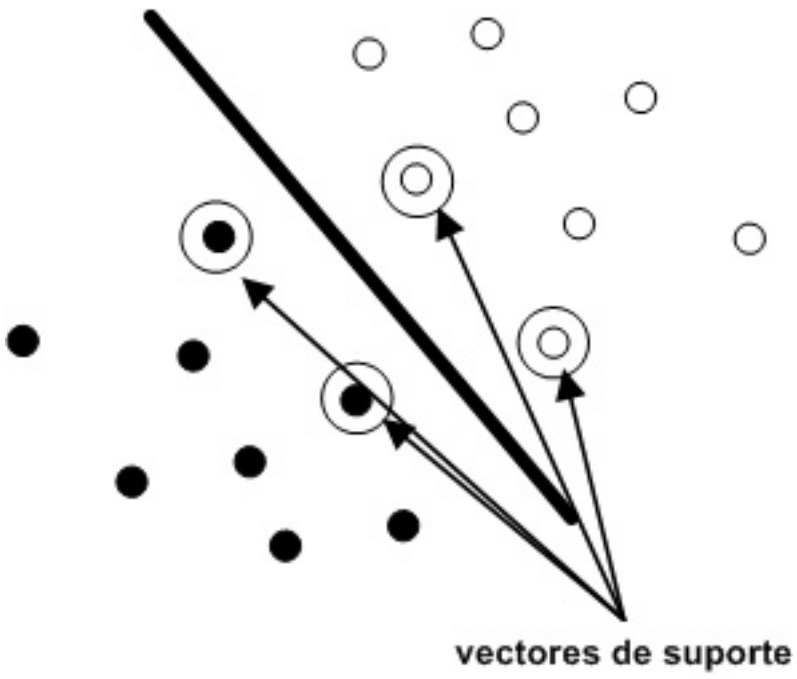

Figura 3.13: Vectores de suporte

Finalmente, a tabela 3.6 apresenta os vários algoritmos disponíveis no WEKA, agrupados na categoria de Funções.

3.2.2 Modelos de Associação

Os modelos de aprendizagem baseados em regras de associação não são muito diferentes das regras de classificação. A diferença está no facto de estes poderem prever qualquer atributo, e não apenas a classe, e de terem a liberdade para prever também combinações de atributos. Enquanto que as regras de classificação podem ser usadas como um todo, as regras de associação não têm esse objectivo, até porque podem prever diferentes regularidades existentes no conjunto de dados.

Como a partir de um conjunto de dados podem ser obtidas várias regras de associação diferentes, é importante restringir estas a um número que abranja uma quantidade alargada de instâncias e que garanta uma elevada certeza sobre as instâncias às quais são aplicadas.

A avaliação da qualidade das regras de associação que são obtidas é feita através de dois indicadores, o *coverage* e o *confidence*. O *coverage* de uma regra de associação indica o número de instâncias que esta regra permite prever correctamente. O *confidence* é o quociente entre o número de instâncias cuja previsão foi correcta e o número total das instâncias abrangidas pela regra.

Nome	Função
SimpleLinearRegression	Regressão linear baseada num único atributo
LinearRegression	Regressão linear standard
LeastMedSq	Regressão utilizando o valor da mediana em vez da média
PaceRegression	Regressão linear utilizando a técnica *Pace*
Logistic	Regressão logística
SimpleLogistic	Regressão logística com selecção de atributos
RBFNetwork	Rede que implementa a função básica radial
VotedPerceptron	Utilização do perceptrão
Winnow	Utilização do perceptrão para múltiplas actualizações
MultilayerPerceptron	Rede neuronal com retropropagação
SMO	Algoritmo de optimização sequencial para suporte de vectores de classificação
SMOreg	Algoritmo de optimização sequencial para suporte de vectores de regressão

Tabela 3.6: Algoritmos de Classificação através de Funções

Se tivermos a seguinte regra de associação:

```
If temperature = cool then humidity = normal
```

pela tabela 3.1 verificamos que existem quatro instâncias onde a regra se verifica, logo o *coverage* desta regra é de quatro. Por outro lado, como para todas as instâncias a regra é verdadeira, o *confidence* da regra é de 100%. É normal especificar um valor mínimo tanto para o *coverage* como para o *confidence* de forma a obtermos apenas as regras mais importantes, reduzindo desta forma a quantidade de regras de associação geradas.

A representação das regras de associação tem por base um conjunto de itens. Cada item é composto por um par atributo-valor. O exemplo seguinte apresenta uma situação com três itens:

```
humidity = normal, windy = false, play = yes
```

A regra de associação obtida pelos três itens anteriores pode ser descrita da seguinte forma:

```
humidity = normal AND windy = false ➠ play = yes
```

onde a parte esquerda da regra (`humidity = normal AND windy = false`) é denominada de antecedente e a parte direita (`play = yes`) é denominada de consequente.

Na ferramenta *WEKA* estão disponíveis três algoritmos de aprendizagem baseados em regras de associação. O *Apriori* começa com um *coverage* de 100% das instâncias e diminui em intervalos de 5% até que existam no mínimo 10 regras com um valor de *confidence* de 90% ou até que o *coverage* atinja o valor mínimo de 10%.

O *PredictiveApriori* combina o *confidence* e o *coverage* numa única métrica e descobre as melhores n regras de associação. Internamente o algoritmo incrementa sucessivamente o valor do *coverage*, tentando desta forma aumentar o valor da certeza da previsão.

O *Tertius* descobre as regras de acordo com uma métrica de confirmação, procurando regras com múltiplas condições no consequente, como o *Apriori*, mas diferente no facto destas condições serem OU lógico e não E lógico (Tabela 3.7). Pode ser configurado para encontrar regras que prevêem uma simples condição ou um predeterminado atributo.

Algoritmo	**Exemplo de regra**	***Coverage***
Apriori	play = no ➠ humidity = high	4
Tertius	play = no ➠ humidity = high OR humidity = normal	5

Tabela 3.7: Diferença entre o *Apriori* e o *Tertius*

3.2.3 Modelos de Clustering

As técnicas de *clustering* são aplicadas quando não existe nenhuma classe a ser inferida, mas onde as instâncias são divididas em grupos naturais. O *clustering* pressupõe que existe uma relação forte entre as várias instâncias que compõem o domínio e que as mesmas podem ser por isso divididas em grupos.

O *clustering* requere naturalmente técnicas diferentes comparado com as técnicas de aprendizagem baseadas na classificação e associação. A escolha destas técnicas deve ser ditada pela natureza do mecanismo que está por detrás de cada fenómeno de *clustering*. No entanto, como estes mecanismos são raramente conhecidos, pois a verdadeira existência dos *clusters* é afinal algo que pretendemos descobrir, a escolha é normalmente ditada pela ferramenta de *clustering* que temos disponível.

Os grupos que são identificados podem ser exclusivos e como tal uma instância pertence apenas a um grupo. Um desses exemplos é a técnica clássica de *clustering*, chamada de *k-means*, que particiona as instâncias em *clusters* disjuntos. Muito semelhante ao método *nearest-neighbor* utilizado na aprendizagem baseada em instâncias, é uma simples e poderosa técnica que é utilizada há muitas décadas.

Este método de *clustering* é simples e eficaz. Tenta formar grupos de instâncias de forma iterativa de modo a que cada grupo seja o mais homogéneo possível e haja a menor semelhança entre grupos distintos.

O seu funcionamento começa inicialmente por especificar quantos *clusters* pretendemos obter através do parâmetro *k*. Os *k* pontos são escolhidos de forma aleatória e serão os centros do *cluster*. Todas as instâncias são associadas ao centro do *cluster* mais próximo, de acordo com o quadrado da distância Euclidiana. De seguida, o centróide (*mean*) das instâncias de cada *cluster* é calculado. Estes centróides são utilizados como os novos valores do centro de cada *cluster*. Este processo é repetido continuamente até se obter os mesmos pontos associados a cada *cluster* nas várias iterações e cada um dos centros do *cluster* esteja estabilizado.

Na ferramenta *WEKA*, estão disponíveis dois algoritmos que utilizam precisamente esta técnica, o *SimpleKMeans* e o *FarthestFirst*. A grande diferença entre estes dois algoritmos está na forma como são calculados os centróides.

No caso do *SimpleKMeans*, os centróides são determinados de forma a minimizar a distância dos vários elementos que compõem o *cluster* ao centro do *cluster* [Dasgupta and Long, 2005] (Equação 3.11).

$$min|x_j - u_i|^2 \tag{3.11}$$

Onde x_j é um elemento do *cluster* e u_i o seu centróide.

Na figura 3.14 temos um exemplo dos centróides que foram determinados desta forma.

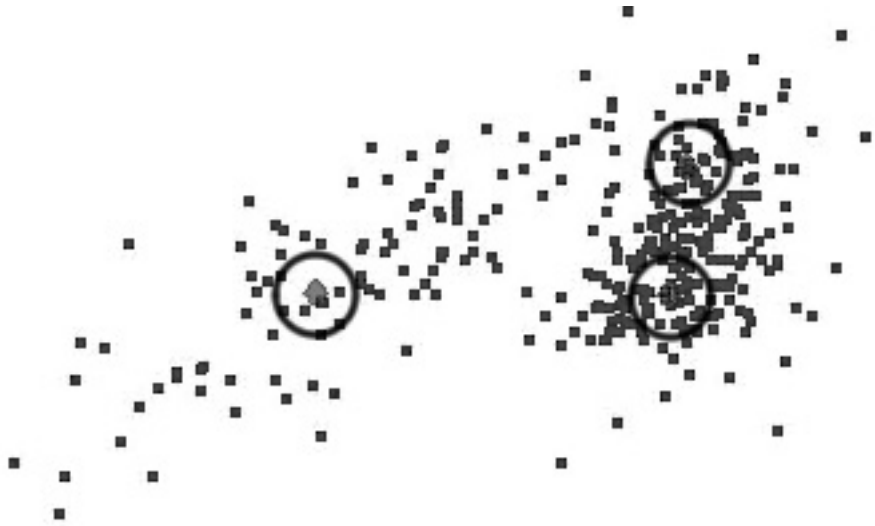

Figura 3.14: Centróides do *SimpleKMeans*

O *FarthestFirst* por sua vez, determina o centróide como o elemento do *cluster* com o valor máximo das distâncias mínimas aos centros actuais [Dasgupta and Long, 2005] (Equação 3.12). A figura 3.15 apresenta os centróides obtidos por este algoritmo.

$$max_x min_c d(x, c) \tag{3.12}$$

Onde x é um elemento do *cluster* e c o seu centróide.

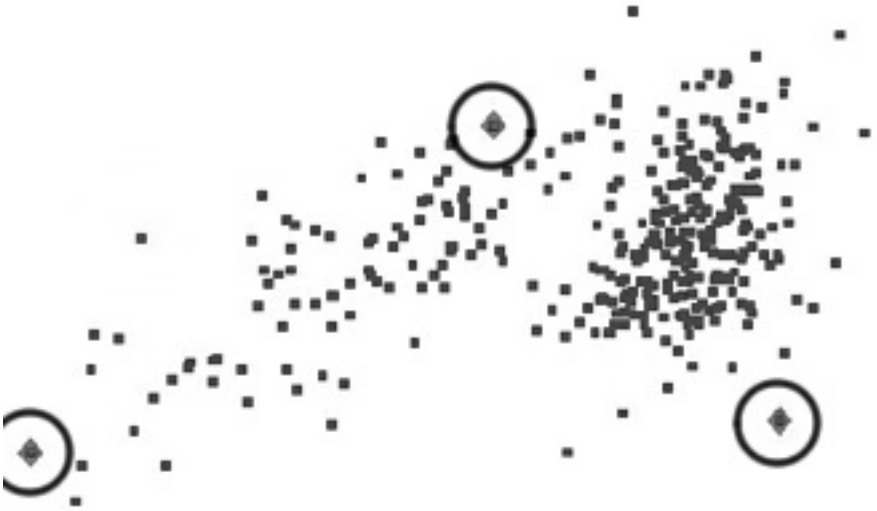

Figura 3.15: Centróides do *FarthestFirst*

3.3 Representação do Conhecimento

Tanto ao nível dos Sensores, como ao nível do Contexto, já se atingiu dentro da área um ponto de alguma estabilidade. Recentemente, as preocupações têm se centrado também nos problemas associados ao Raciocínio. De um modo geral, os problemas são apenas dois: a representação do conhecimento e os mecanismos de aprendizagem.

Relativamente ao primeiro, existem essencialmente quatro abordagens diferentes. Através de predicados [Ranganathan et al., 2004], *UserML*[15] [Heckmann and Krüger, 2003], *strings* guardadas em base de dados [Jiang and Steenkiste, 2002] ou ontologias utilizando *OWL*[16] [Strang et al., 2003, Wang et al., 2004], só a forma os torna diferentes, porque o principio é idêntico: encontrar uma representação que possa ser adoptada por vários sistemas de *context-awareness*.

No trabalho realizado por Heckmann [Heckmann and Krüger, 2003], é apresentado o *UserML*, como forma de representação do modelo de contexto do utilizador, que tem como objectivo criar uma estrutura uniforme de representação do conhecimento, de modo a que a informação sobre o seu contexto possa ser partilhada para vários sistemas, garantindo desta

[15] *User Modeling Markup Language*
[16] Ontology Web Language

forma a compatibilidade entre a informação e as aplicações. Esta representação tem por base uma estrutura em formato *XML*. Vejamos o exemplo apresentado nesse trabalho da representação do contexto utilizando o *UserML* (Figura 3.16), cujo objectivo é representar um cenário onde o sistema de navegação de um aeroporto detecta que um passageiro está sobre pressão temporal, uma vez que tem um bilhete para um voo que parte em menos de 10 minutos e ainda não chegou ao local de embarque.

```
<InferenceExplanations>
    <inference id="002">
        <inferred>
            <userplan>goto.airport-location-X34</userplan>
            <userproperty>timpressure.high</userproperty>
        </inferred>
        <inferred-from>
            <evidence>user.has-flight-ticket</evidence>
            <evidence>user.airport-location-X20</evidence>
            <evidence>boarding-time.flight</evidence>
        </inferred-from>
        <inferred-by>
            <device>334</device>
        </inferred-by>
    </inference>
</InferenceExplanations>
```

Figura 3.16: Exemplo obtido a partir do *UserML*[17]

Ranganathan [Ranganathan et al., 2004] defende outro tipo de representação, baseada em predicados. A opção por este tipo de representação prende-se com o facto de poderem ser facilmente convertidos em regras ou em qualquer outro mecanismo de aprendizagem. A representação dos predicados é feita no formato `(ContextType(<Subject>, <Object>)` ou então `(ContextType(<Subject>, <Verb>, <Object>)`. Como exemplo desses predicados temos,

```
location(jeff, in, room 3105)
activity(room 3102, meeting)
light(room 3220, dim)
office(chetan, room 3216)
```

[17]Imagem extraída de [Heckmann and Krüger, 2003]

Por sua vez, Jiang [Jiang and Steenkiste, 2002] num trabalho realizado sobre o *Aura Location Identifier* (ALI) representa o conhecimento recorrendo à construção de *strings* para descrever o contexto:

```
ali://cmu/wean-hall/floor3/3100-corridor/3115
```

o identificador `ali` indica que o utilizador está na sala `3115`, localizada no corredor `3110` do `floor3` do edifício `wean-hall` da `cmu`. Esta representação é posteriormente guardada em base de dados, sobre a qual são executadas *queries*.

Por último, vários autores têm proposto como alternativa a representação do conhecimento recorrendo a ontologias utilizando para tal a linguagem *OWL*, permitindo assim que o conhecimento seja mais explicito, mas que continue a ser computacionalmente fácil de ser interpretado. Como o seu princípio é idêntico para os vários trabalhos, vão ser apenas apresentados alguns desses autores. Strang [Strang et al., 2003], utiliza no seu trabalho sobre *Aspect-Scale-Context* este tipo de representação, assim como Wang [Wang et al., 2004] no projecto *CONON*, cujo exemplo seguinte apresenta um extracto da ontologia criada para uma casa inteligente.

```
<owl:Class rdf:ID="ContextEntity"/>
<owl:Class rdf:ID="Location">
    <rdfs:subClassOf rdf:resource="#ContextEntity"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="longtitude">
    <rdf:type rdf:resource="FunctionalProperty">
    <rdfs:domain rdf:resource="Location">
    <rdfs:range rdf:resource="xsd:double">
</owl:ObjectProperty>
<owl:Class rdf:ID="IndoorSpace">
    <rdfs:subClassOf rdf:resource="#Location"/>
    <owl:disjointWith rdf:resource="#OutdoorSpace"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="locatedIn">
    <rdf:type="owl :TransitiveProperty"/>
    <rdfs:domain rdf:resource="#Entity"/>
    <rdfs:range rdf:resource="#Location"/>
    <owl:inverseOf rdf:resource="#contains "/>
</owl:ObjectProperty>
```

Para se representar por exemplo a localização de uma pessoa, é criado um predicado com três atributos: sujeito, verbo e objecto.

```
(Wang,locatedIn, Bedroom)
```

Podemos assim afirmar que "`Wand is located in the bedroom`". Mas a representação do conhecimento por si só não significa que o sistema seja inteligente. Comparativamente, podemos até afirmar que as representações apresentadas anteriormente têm tanto de aprendizagem como uma Tabela de Decisão. Vejamos o exemplo apresentado por Heckmann [Heckmann and Krüger, 2003] (Figura 3.16) e respectiva conversão para uma tabela de decisão (Tabela 3.8).

Bilhete	**Localização**	**Voo**	**Inferencia**
True	Location-X20	True	Location-X34

Tabela 3.8: Conversão para uma Tabela de Decisão

É então necessário dotar as representações de mecanismos de raciocínio para que exista efectivamente aprendizagem. Das representações apresentadas anteriormente, nem todas fazem referência aos mecanismos de aprendizagem, no entanto existem pelo menos dois trabalhos onde é demonstrado como é realizado esse mecanismo. No projecto *CONON*, Wang [Wang et al., 2004] propõe um mecanismo baseado em Regras de Classificação. No exemplo apresentado, o objectivo é determinar o que o utilizador está a fazer a partir de um conjunto de informação de contexto (Tabela 3.9).

Situação	**Regras**
Sleeping	`(?u locatedIn Bedroom)` ^`(Bedroom lightLevel LOW)` ^ `(Bedroom drapeStatus CLOSED)` →`(?u situation SLEEPING)`
Showering	`(?u locatedIn Bathroom)` ^`(WaterHeater locatedIn Bathroom)` ^ `(Bathroom doorStatus CLOSED)` ^`(WaterHeater status ON )` → `(?u situation SHOWERING)`
Cooking	`(?u locatedIn Kitchen)` ^(ElectricOven locatedIn Kitchen)^ `(ElectricOven status ON )` →`(?u situation COOKING)`

Tabela 3.9: Regras de Classificação definidas pelo utilizador[18]

A grande questão deste mecanismo é que as regras anteriormente descritas não foram obtidas directamente da *OWL*, mas sim deduzidas desta por intermédio do utilizador. A *OWL* permite obter regras de baixo nível onde a informação está implícita, sendo por isso necessário tornar esta explicita. Esta abordagem permite criar mecanismos de raciocínio mais flexíveis e mais próximos das questões que queremos que o sistema seja capaz de responder, como por exemplo, saber a actividade do utilizador [Wang et al., 2004]. No entanto, implica uma intervenção humana, não sendo por isso um mecanismo autónomo como se desejaria.

Como refere Ranganathan [Ranganathan et al., 2004], os sistemas *context-awareness* nem sempre conseguem identificar com precisão o estado actual do contexto, sendo por isso ne-

[18]Tabela extraída de [Wang et al., 2004]

cessário algum suporte à incerteza. No seu trabalho, além da representação do conhecimento, apresenta também mecanismos de raciocínio que permitem que serviços e aplicações possam lidar com a incerteza do contexto. Neste caso em particular estes mecanismo foram incorporados no projecto *Gaia*. Neste projecto, foi desenvolvido um modelo de incerteza baseado nos predicados e aos quais se associaram valores de confiança. Este modelo tem como objectivo criar a base para o raciocínio acerca da incerteza de contexto, utilizando para isso mecanismos de aprendizagem baseados em modelos probabilísticos. Como dentro dos vários predicados existem alguns, como é o caso da localização e da actividade, que podem conter incerteza, a cada predicado é associado um valor de confiança entre 0 e 1, valor este que mede a probabilidade de o mesmo ser verdadeiro. Por exemplo,

```
prob(location(carol, in, room 3233)) = 0.5
```

significa que a probabilidade da `Carol` estar na sala `3233` é de `0,5`.

Dentro dos vários modelos probabilísticos de aprendizagem, a escolha preferencial recai sobre as redes bayesianas, pelo facto de terem resultados mais interessantes ao lidar com a incerteza, principalmente quando existe uma relação causal entre vários eventos.

A grande dificuldade das redes bayesianas está no desempenho da sua criação. À medida que esta aumenta em tamanho, o tempo necessário para a sua geração pode ser exponencial, podendo assim originar problemas de desempenho [Ranganathan et al., 2004].

Capítulo 4

Proactividade a partir de histórico aplicacional

Como apresentado na secção 2.2, o grande desafio em Computação Ubíqua passa pela criação de aplicações verdadeiramente proactivas. Como ficou demonstrado na figura 2.1, para se conseguir esta proactividade, é necessário ter em conta quatro entidades: Sensores, Contexto, Raciocínio e Actuadores.

Atendendo a que o objectivo desta dissertação está centrado na entidade Raciocínio, através da avaliação de um conjunto de algoritmos de aprendizagem, era necessário ter um sistema que fizesse aquisição do contexto do utilizador, por forma a obtermos informação de contexto necessária para a referida avaliação.

Uma das opções seria adoptar um dos sistemas já existentes, como por exemplo o *ContextPhone* [Raento et al., 2005] (secção 3.1.6), utilizando-o para fazer a aquisição de informação de contexto. No entanto, não pretendíamos que o trabalho de dissertação se resumisse apenas à questão da avaliação dos modelos de aprendizagem, pretendíamos que este trabalho fosse mais abrangente, de forma a permitir a obtenção de *know-how* também nas questões relacionadas com o contexto e seus modelos, assim como na utilização de sensores, como elementos geradores de informação.

Pelo facto de querermos criar um sistema *context-awareness* de raiz, originou-nos logo um desafio: Qual o dispositivo ubíquo a utilizar?

4.1 O PDA como dispositivo ubíquo

Uma vez que, para se adquirir o contexto é necessário que as fontes de informação estejam próximas do utilizador, tanto do ponto de vista físico como do ponto de vista virtual, e que

reflictam as alterações constantes do ambiente, dispositivos como *PDA* ou *Smartphone* são cada vez mais sérios candidatos. Estes acompanham o utilizador para onde quer que este se desloque, aliado ao facto de que estes dispositivos integram cada vez mais funcionalidades que podem ser utilizadas como sensores (câmara, microfone, *Wi-Fi*, *Bluetooth*, *GSM*, *GPRS/UMTS*, *GPS*, funcionalidades de agenda pessoal, funcionalidades de telefone, etc.). Por esta razão, a escolha do PDA com funcionalidades de telefone, como dispositivo ubíquo, pareceu-nos muito aliciante para os nossos objectivos.

Definido o dispositivo a utilizar, era agora necessário idealizar o sistema *context-awareness* que iríamos desenvolver.

4.2 A Ideia

Ao contrário do que acontece nas aplicações desenvolvidas para os computadores pessoais, no caso dos *PDA*s com sistema operativo *Windows Mobile*, quando se fecha uma aplicação, o seu processo não termina como seria de esperar, mas passa para *background*, continuando assim em execução.

Minimizar em vez de terminar o processo pode à partida parecer uma anomalia do próprio sistema operativo, no entanto resulta de uma opção que foi tomada pela própria *Microsoft*. Segundo esta [Wilson, 2003], manter os processos a correr em *background* em vez de os terminar quando estes já não estão a ser utilizados permite que o utilizador possa rapidamente reactivá-los, no ponto onde o utilizador os tinha deixado. Esta opção permite assim que os utilizadores possam rapidamente mover-se entre aplicações. Mas se por um lado esta opção reduz o tempo de mudança entre aplicações, por outro lado implica a redução dos recursos disponíveis do *PDA* (por exemplo: memória e processador) pelo facto de ficarem várias aplicações em execução, mesmo que em *background*.

Por forma a demonstrar a que ponto esta opção afecta a utilização da memória do *PDA*, vejamos o seguinte exemplo, utilizando para tal o *System Information* do *Pocket Controller*[1]. A figura 4.1 apresenta o estado de memória do *PDA*, antes de ser executada qualquer aplicação.

Como podemos verificar pela figura 4.1, neste momento temos disponíveis 9,96 MB de memória, estando 69 por cento da memória do *PDA* em utilização.

Vamos supor que o utilizador deste *PDA* pretende agora consultar uma página na Internet, escrever um pequeno texto e preparar um gráfico para a reunião do dia seguinte. Após executar estas três tarefas, que correspondem à utilização do *Internet Explorer*, *Pocket Word* e *Pocket Excel*, respectivamente, o estado da memória do *PDA* passa a ser outro (Figura 4.2).

[1] *Pocket Controller Professional*, ferramenta de sincronização do *PDA* com o computador pessoal, desenvolvido pela *SOTI*, disponível em http://www.soti.net

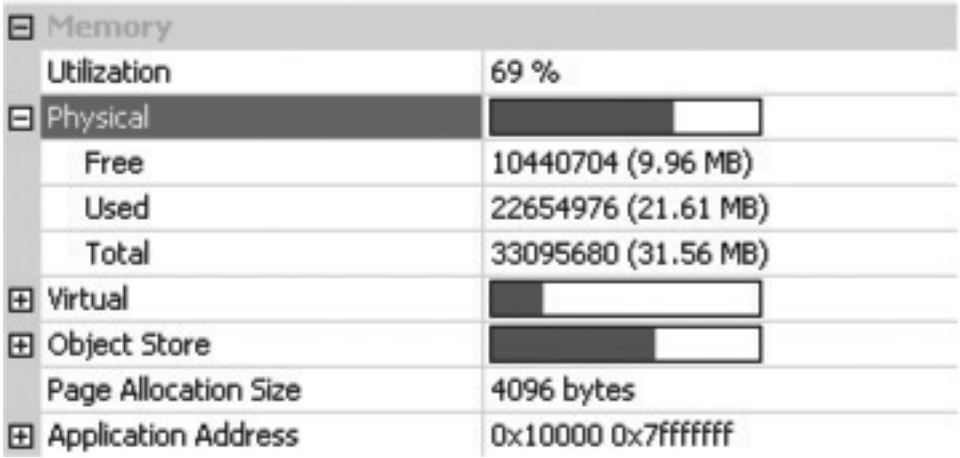

Figura 4.1: Estado inicial da memória

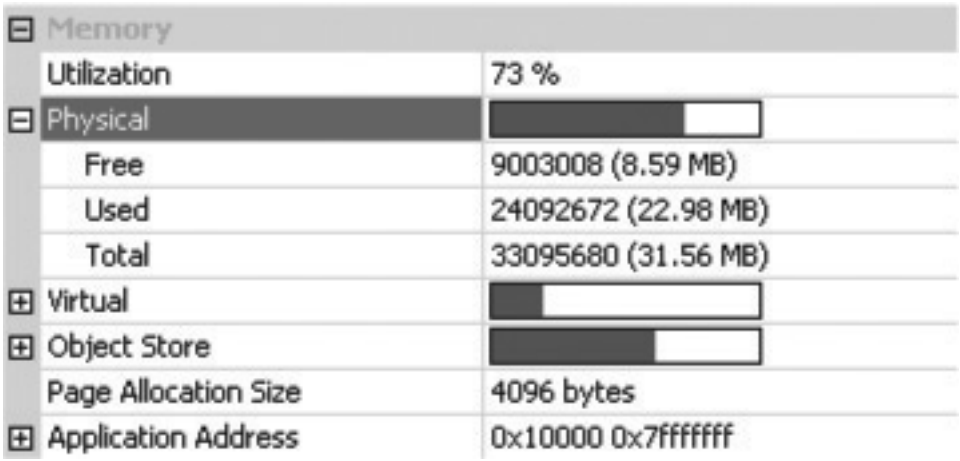

Figura 4.2: Estado da memória após utilização

Estes três passos resultaram num decréscimo da memória disponível em 10 por cento, o que corresponde a ficar disponível apenas 8,59 MB de memória. Em resumo, estas três aplicações, que a partir de determinada altura deixaram de ser necessárias, estão a consumir cerca de 1 MB da memória do dispositivo. Mas não é só ao nível da memória que está a ocorrer um desperdício, também ao nível do processador estão a ser utilizados recursos desnecessários, uma vez que, mesmo em *background*, os processos consomem processador.

Embora nos últimos anos os *PDA*s tenham tido uma evolução tecnológica bastante significativa, os seus recursos continuam limitados. Comparativamente com os computadores pessoais, estes dispositivos estão ainda muito distantes ao nível da capacidade de armazenamento, memória e processador, e como estamos a falar de um dispositivo móvel, a sua bateria ainda não tem grande autonomia, daí que a sua utilização deva ser optimizada.

O número de serviços de comunicação nestes dispositivos também tem tido grandes avanços e tecnologias como *Bluetooth* e *Wifi* são muito comuns. No entanto, quando activos, estes serviços consomem uma grande quantidade de energia, reduzindo a autonomia da bateria drasticamente. Conscientes deste problema, alguns dos fabricantes destes dispositivos adicionaram a estes serviços a opção de os desligar, após algum tempo de inactividade, por forma a minimizar o consumo desnecessário de bateria. Na figura 4.3 temos um desses exemplos para um dos modelos da *Qtek*, o G100.

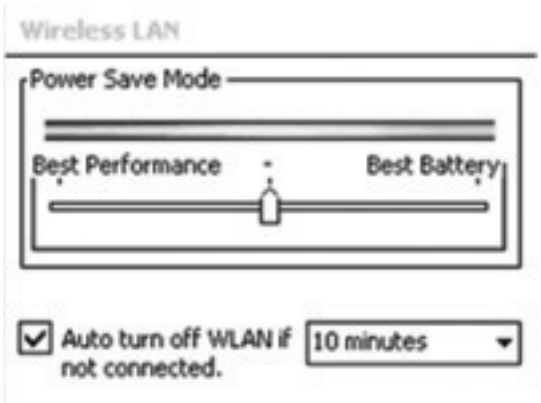

Figura 4.3: *Wireless settings* do *Qtek* G100

A questão dos recursos, tanto ao nível de memória e processador, como ao nível da autonomia da bateria, parece ser um problema pelas razões apontadas anteriormente.

Um sistema proactivo que seja capaz de inicializar e terminar processos e/ou serviços de comunicação, de acordo com as necessidades tanto do utilizador como das próprias aplicações, parece-nos de grande potencial na área de Computação Ubíqua.

No entanto, este sistema é um grande desafio, uma vez que implica a percepção (com elevada certeza) das intenções do utilizador, aliada à necessidade de um modelo de contexto que contenha o máximo de informação possível do ambiente em que se encontra o utilizador.

Como determinar algo tão simples como as intenções do utilizador é uma tarefa extremamente complexa, uma vez que saber o porquê dos comportamentos do utilizador envolve conhecer tanto as suas características físicas, como as características psicológicas, resolvemos simplificar o sistema. Esta opção em nada afecta o nosso trabalho, pois o grande objectivo desta dissertação não era o sistema em si, mas sim a avaliação dos algoritmos de aprendizagem, como forma de determinar quais os mais adequados para um sistema deste tipo.

Desta forma, decidimos implementar uma aplicação que, continuamente e em intervalos de tempo de 15 segundos, vai adquirindo informação contextual, com o objectivo de prever as 5 aplicações mais prováveis de ser executadas brevemente pelo utilizador.

Estas aplicações seriam posteriormente apresentadas em forma de lista, como um item extra do *Today Screen*[2] do *PDA*.

Esta solução, além de permitir avaliar a aquisição de informação de contexto, permite também a recolha de dados necessários para a avaliação dos algoritmos de aprendizagem. Outra mais valia é permitir melhorar a usabilidade destes dispositivos, uma vez que facilita o acesso às 5 aplicações mais prováveis.

Embora esta aplicação não tenha ainda a parte proactiva do sistema idealizado inicialmente, é um excelente ponto de partida para o mesmo, uma vez que podemos verificar se somos capazes de prever os próximos passos da interacção do utilizador com o *PDA* e assim poder garantir que a informação passada aos Actuadores é a correcta.

4.3 Contexto

Neste momento, já vimos qual o dispositivo ubíquo a utilizar e qual a aplicação idealizada para a recolha de informação de contexto necessária à avaliação dos algoritmos de aprendizagem. Falta então concretizar essa ideia. E para a concretizar é necessário decidir como vai ser definido o contexto a recolher, e por sua vez quais os sensores a utilizar para tal.

Segundo Abowd [Abowd et al., 1999], a definição completa do contexto passa por ter as seguintes dimensões: **Identidade**, **Localização**, **Tempo** e **Actividade** (Figura 2.3).

O modelo de contexto de McFadden [McFadden et al., 2004] instância muito bem estas quatro dimensões (Figura 3.5), senão vejamos:

```
Identidade → Person, Device, Organization
Localização → Place, located near Device
Actividade → Activity, using Device
```

Embora não ilustrada, a dimensão Tempo encontra-se intrinsecamente presente, uma vez que os acontecimentos têm uma característica temporal.

Para nós, nesta fase, das quatro dimensões vamos excluir a Identidade, uma vez que a aplicação recolhe informação do contexto referente apenas a esse utilizador e porque determinar quais os dispositivos existentes na zona de acção do utilizador é uma tarefa com alguma complexidade, uma vez que é necessária a implementação de um mecanismo de descoberta dos mesmos. Assim, de momento apenas nos interessam as dimensões Localização, Tempo e Actividade. A actividade apenas se refere à actividade computacional do *PDA* e não a toda a actividade do utilizador.

[2] O *Today Screen* é o écran que é apresentado por *default* no *PDA*, contendo informação variada e vários atalhos para as aplicações. É idêntico ao *Desktop* dos computadores pessoais, visível quando não está a ser executada nenhuma aplicação

A redução a apenas duas dimensões de contexto tem ainda outras motivações: primeiro, a área da Localização é a que está a ser mais explorada e onde se vê maior aplicabilidade, e segundo, acreditamos que existe de alguma forma uma relação entre as aplicações que são executadas no *PDA* e o local onde o utilizador se encontra. Assim, com estas dimensões podemos validar ou não esta nossa hipótese.

Não é preciso fazer um grande esforço para encontrar exemplos de situações onde o local influencia a utilização que damos ao *PDA*. Vejamos um exemplo muito simples de um utilizador hipotético e respectiva utilização do seu dispositivo, em vários locais ao longo de um dia. Imaginemos que este utilizador está durante a manhã no local de trabalho, sai e vai almoçar ao restaurante, regressa ao trabalho e no final do dia regressa a casa.

Local	**Interacção com o *PDA***
Trabalho	Sincroniza a agenda
	Tira apontamentos da reunião
	Faz uma apresentação
Restaurante	Faz uns telefonemas
	Responde às sms que recebeu
	Consulta o correio electrónico
Casa	Vê páginas da Internet
	Consulta correio electrónico
	Consulta a agenda para o dia seguinte
	Joga

Tabela 4.1: Exemplo da utilização do *PDA* segundo a localização

Através deste exemplo muito simples, podemos verificar que existem tarefas que são exclusivas de determinadas situações, enquanto que outras se replicam em mais do que uma situação, implicando de algum forma que, a utilização que é dada ao dispositivo é diferente ao longo do dia.

No entanto, como o ser humano adquire hábitos, é de considerar que o comportamento apresentado anteriormente se repita nos próximos dias. É com base neste princípio que acreditamos que, se formos capazes de absorver o máximo possível do contexto do utilizador para um dia, poderemos prever o seu comportamento nos dias seguintes.

4.4 Arquitectura da Aplicação de Amostragem

Tendo por base então a definição das dimensões de contexto a ser utilizadas, o nosso sistema, a que foi dado o nome de **Aplicação de Amostragem**, pode ser definido da seguinte forma (Figura 4.4).

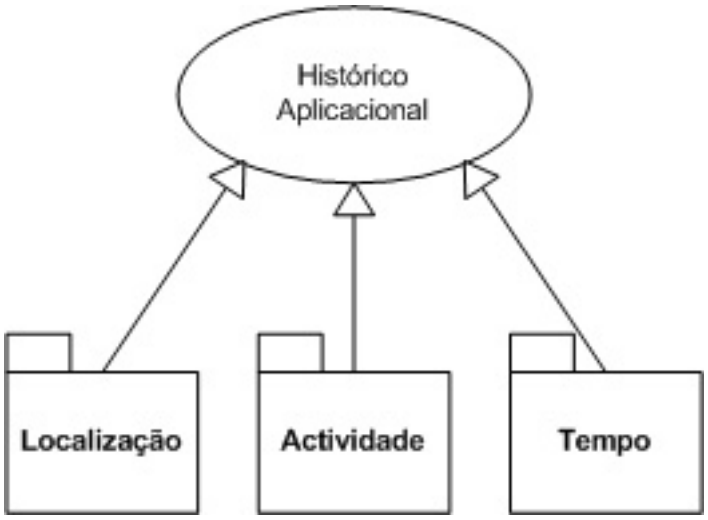

Figura 4.4: Arquitectura da Aplicação de Amostragem

Para a implementação do nosso sistema, optámos pela plataforma *Microsoft .Net Compact Framework 1.0*[3], por duas razões. Primeiro, por ser vocacionada para este tipo de dispositivos, esperando assim ter um conjunto mais alargado de funcionalidades disponíveis e segundo porque permitia-nos ganhar *know-how* numa plataforma que para nós era ainda novidade.

Dentro desta optou-se pela utilização da linguagem de programação *VB.NET*, pois era a que dava mais garantias de uma curva de aprendizagem mais rápida.

4.4.1 Processos

O primeiro módulo de *software* a ser desenvolvido, denominado de `Processo Activo` prende-se com a dimensão de contexto Actividade. Como o que pretendemos saber no nosso caso é unicamente a actividade computacional executada no *PDA*, para obtermos esta informação é necessário conhecer os vários processos em execução neste, dando particular importância ao processo que está actualmente activo.

Para determinar os processos em execução é necessário recorrer a uma estrutura de dados existente na *Compact Framework*, denominada de `PROCESSENTRY32`[4] (Figura 4.5). Esta estrutura serve como suporte à informação para cada um dos processos. Apenas nos interessavam dois atributos: o identificador do processo e o seu nome, respectivamente `szExeFile` e `th32ProcessID`.

[3]Plataforma de desenvolvimento incorporada na *Microsoft .Net Framework* (CF1.0), cujo objectivo é a criação de soluções direccionadas exclusivamente para *PDA*s e *Smartphones*. Esta versão permite o desenvolvimento de soluções para o sistema operativo *Windows Mobile 2003* e posteriores

[4]Informação disponível no MSDN em http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetcomp/html/ProcessManager.asp

```
typedef struct tagPROCESSENTRY32
{
   DWORD dwSize;
   DWORD cntUsage;
   DWORD th32ProcessID;
   DWORD th32DefaultHeapID;
   DWORD th32ModuleID;
   DWORD cntThreads;
   DWORD th32ParentProcessID;
   LONG  pcPriClassBase;
   DWORD dwFlags;
   TCHAR szExeFile[MAX_PATH];
   DWORD th32MemoryBase;
   DWORD th32AccessKey;
} PROCESSENTRY32;
```

Figura 4.5: Estrutura de dados PROCESSENTRY32

```
NK.EXE, filesys.exe, gwes.exe, device.exe, connmgr.exe,
shell32.exe, ConManClient.exe, services.exe, srvtrust.exe,
CommLoader.exe, mbutton.exe, cprog.exe, STK.exe, poutlook.exe,
mstli.exe, rapisrv.exe, fexplore.exe, repllog.exe,
rnaapp.exe, Localiza.exe, tmail.exe, pword.exe
```

Figura 4.6: Exemplo da lista de processos em execução

Como forma de obter todos os processos em execução, é necessário executar a função `CreateToolhelp32Snapshot`, que devolve um conjunto de estruturas `PROCESSENTRY32` de acordo com o número de processos em execução. Desta forma, conseguimos assim obter em cada instante os nomes dos processos e respectivos identificadores que estão actualmente em execução. O exemplo da figura 4.6 mostra a lista de processos que obtivemos, contendo apenas o nome do processo.

Esta lista de processos apenas nos indica quais os que estão actualmente em execução, não fazendo distinção entre qual destes está activo e quais os que estão em *background*. Para isso é necessário recorrer à função `GetForegroundWindow`, que devolve o identificador do processo cuja janela está actualmente visível no dispositivo. Os identificadores dos processos obtidos anteriormente através do `PROCESSENTRY32` são agora necessários para determinar o nome do processo que está actualmente activo.

A tabela 4.2 demonstra um exemplo do que consideramos de informação útil de contexto, referente à actividade computacional no *PDA*, sendo que cada linha corresponde à aplicação activa em determinado instante.

Nome	**Identificador**
poutlook.exe	-746319354
poutlook.exe	-746319354
shell32.exe	-742495894
tmail.exe	850304754
fexplore.exe	856654430

Tabela 4.2: Caracterização dos processos

Embora tenhamos disponível, além da informação anterior, a lista de processos em execução, a mesma não vai ser utilizada para caracterização do contexto, isto porque sabíamos à partida que alguns dos processos em execução nada têm a ver com o momento actual, sendo casos de processos que ficaram em execução de forma desnecessária.

Vejamos por exemplo a informação apresentada na figura 4.6, onde encontramos como último processo da lista o `pword`, que se refere à aplicação *Pocket Word*. Este processo, ao longo do período de teste em que a aplicação ficou em aquisição de informação, nunca foi utilizado, logo não faz parte do contexto aplicacional. Se fôssemos a considerar todos os processos em execução como informação caracterizadora do contexto, neste caso estávamos a ser desviados da realidade, uma vez que este nunca foi um processo activo.

Mesmo assim, embora nesta fase esta informação não tenha utilidade, posteriormente quando quisermos integrar alguma proactividade no nosso sistema, será bastante útil, uma vez que vai permitir a finalização dos processos que estão fora dos 5 potenciais a ser activados pelo utilizador.

A figura 4.7 apresenta o resultado da implementação desta classe, onde a propriedade `dados` permite obter o nome do processo actualmente activo e respectivo identificador.

A arquitectura do nosso sistema começa assim a ganhar a forma. A figura 4.8 apresenta a actualização da representação do sistema.

4.4.2 Global Positioning System

Desenvolvido a partir de 1978, pelo Departamento de Defesa dos Estados Unidos da América, inicialmente para fins apenas militares, o *Global Positioning System* (*GPS*)[5] é um sistema de navegação por satélite formado por uma constelação de mais de 24 satélites que acompanham a órbita da Terra. Estes dispositivos estão colocados de forma a que em qualquer ponto da

[5]Retirado da Wikipedia, em http://en.wikipedia.org/wiki/Global_Positioning_System

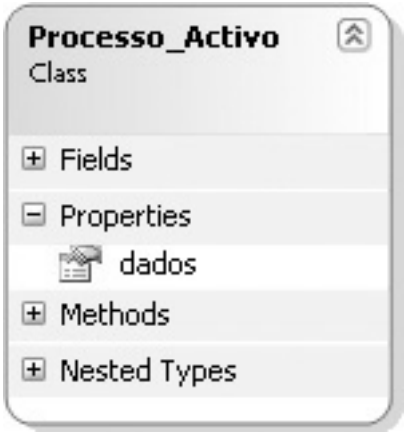

Figura 4.7: Classe Processo Activo

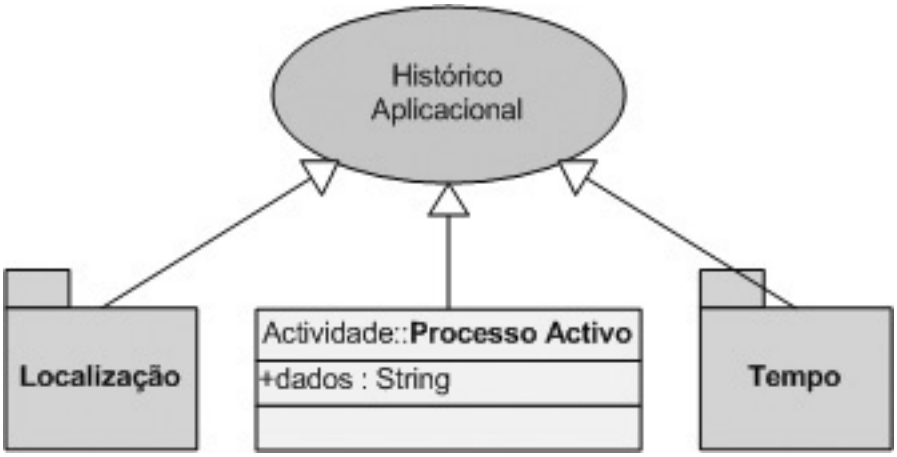

Figura 4.8: Arquitectura da Aplicação de Amostragem

Terra existam no mínimo quatro satélites visíveis no horizonte, garantindo assim a cobertura de todo o planeta.

Estes satélites emitem sinais rádio com informação horária precisa, oriunda dos seus relógios atómicos, permitindo assim que o receptor *GPS* possa determinar geograficamente a sua posição em qualquer lugar da Terra. Além de permitir determinar a posição através de coordenadas geográficas (Latitude e Longitude), permite também obter a altitude e a velocidade do receptor, entre outras.

Os receptores calculam a sua própria posição através dos sinais enviados por pelo menos três satélites, recorrendo a um processo chamado de tri-lateração[6], que utiliza a localização

[6] *Trilateration* - determinar a posição relativa de objectos através da geometria de triângulos, medindo a distância entre a posição desses objectos. Não confundir com triangulação, que utiliza a medida de ângulos em vez da distância. Retirado da Wikipedia, em http://en.wikipedia.org/wiki/Trilateration

obtida a partir de cada um dos sinais de satélite para calcular a sua posição relativa. A figura 4.9 apresenta um exemplo deste processo, utilizando a informação obtida a partir do sinal de três satélites.

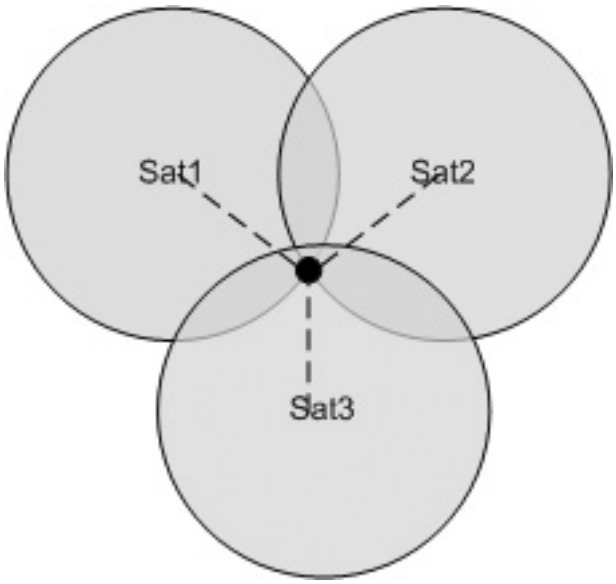

Figura 4.9: Exemplo de tri-lateração com três satélites

Para determinar individualmente a localização sugerida por cada um dos satélites, o receptor calcula a diferença entre o instante em que o sinal foi enviado pelo satélite e o instante em que foi recebido pelo receptor. Esta diferença de tempo, multiplicada pela velocidade da luz, determina a distância do receptor ao satélite. Como periodicamente o receptor recebe do satélite informação sobre o seu posicionamento, comparando a distância calculada anteriormente com esta informação, o receptor pode descobrir a sua própria localização.

Como o cálculo da diferença entre o envio e recepção do sinal é realizado pelo receptor, o grau de precisão da localização obtida depende muito da precisão do relógio do receptor e da constante sincronização deste com os relógios atómicos dos satélites. Uma variação de apenas 0,1 microsegundo corresponde, ao nível da distância, a um erro de cerca 30 metros, podendo levar a que as posições obtidas pelos sinais individuais dos satélites não se sobreponham, não sendo possível assim determinar a posição correcta do receptor. Uma forma de reduzir o erro no cálculo da localização é utilizar o máximo de satélites possíveis, para que o receptor possa matematicamente corrigir este erro de relógio. Nos sistemas comerciais, normalmente utilizam-se entre 6 a 10 sinais de satélite, permitindo que o erro de cálculo se aproxime de zero.

A posição calculada pelos receptores de *GPS* dependem assim da precisão de três elementos: relógio do receptor; informação sobre o posicionamento dos satélites; e cálculo da

diferença de tempo entre o envio e recepção dos sinais. No melhor dos casos, estes três elementos permitem ter um erro máximo de 3 metros, que é basicamente o erro provocado pelo processamento do sinal enviado pelos satélites. Na prática, o erro dos receptores de *GPS* é no máximo cerca de 15 metros, uma vez que ao erro do próprio receptor é necessário adicionar ainda erros provocados pelos efeitos atmosféricos que afectam a propagação do sinal dos satélites.

Determinada a localização e restante informação pelo receptor de *GPS*, é necessário que a mesma seja partilhada com dispositivos que façam uso desta, como por exemplos os *PDA*s. De forma a uniformizar esta informação, garantindo assim a compatibilidade entre os receptores de *GPS* e dispositivos que necessitem de informação sobre a sua localização, a maioria dos receptores de *GPS* devolvem a informação segundo a norma *NMEA*[7].

Esta norma define principalmente como os dados devem ser transmitidos, em particular o formato das mensagens enviadas pelos receptores. Estas mensagens são todas em formato ASCII e deverão seguir os seguintes critérios[8]:

- Cada mensagem inicia-se com o caracter "$"
- Os cinco caracteres seguintes identificam o tipo de mensagem
- A informação é delimitada pelo caracter ","
- O caracter "*"delimita o fim da informação
- Os dois dígitos seguintes servem para detecção de erros na mensagem

O exemplo seguinte ilustra uma das mensagens *NMEA* enviadas pelos receptores de *GPS*:

```
$GPAAM,A,A,0.10,N,WPTNME*43
```

onde `GPAAM` é o tipo da mensagem, `A,A,0.10,N,WPTNME` os dados da mensagem e `43` a verificação de erro na mensagem.

A norma *NMEA* define um conjunto alargado de tipos de mensagens [Inc., 2005a], cada uma delas com informação especifica sobre *GPS*, no entanto esta norma apenas especifica os formatos, não obrigando a que todos os tipos de mensagens estejam presentes nos vários tipos de receptores de *GPS*. Assim, é necessário definir quais os tipos de mensagens que devem ser por nós processadas.

[7]NMEA - National Maritime Electronic Association. A norma *NMEA*, divulgada inicialmente em 1983, tinha como objectivo inicial ser um standard entre os dispositivos marítimos. Esta norma define os requisitos dos sinais electrónicos, os protocolos de transmissão de dados, formatos e intervalos de tempo dos dados enviados via porta série a uma taxa de 4800 bits/segundo. Disponível em http://www.auditmypc.com/acronym/NMEA.asp

[8]Retirado da Wikipedia, disponível em: http://en.wikipedia.org/wiki/NMEA

Como pretendemos que o nosso sistema abranja o maior número de receptores de *GPS* e que, por outro lado, seja possível processar o maior número de novas mensagens no menor intervalo de tempo, considerámos apenas a utilização de dois tipos de mensagens *NMEA*: a *GGA*[9] e *RMC*[10]. Estes dois tipos de mensagens são comuns em todos os receptores de *GPS* e são as que têm maior ocorrência para o mesmo intervalo de tempo, pois são enviadas pelos receptores em intervalos de tempo de apenas 1 segundo, um valor muito interessante, comparativamente com as mensagens do tipo *GSV*[11] que são enviadas a cada 5 segundos [Inc., 2005b].

Como, para o nosso sistema, a informação importante a reter do *GPS* refere-se apenas à localização do utilizador, os dois tipos de mensagens *NMEA* escolhidos são suficientes. A tabela 4.3 apresenta a informação disponível por cada uma dessas mensagens.

Tipo de Mensagem	**Descrição**
GGA	Tempo, posição and tipo de fixação de sinal
RMC	Tempo, data, posição, trajecto e velocidade

Tabela 4.3: Mensagens *NMEA* utilizadas

Analisando com atenção a tabela anterior, verificamos que ambos os tipos de mensagens devolvem informação sobre a posição (Latitude e Longitude), o que à partida pode tornar desnecessário o processamento das duas mensagens. No entanto, pelo facto dos receptores de *GPS* necessitarem de cumprir os intervalos de tempo com que as mensagens são enviadas, surgem mensagens com informação incompleta. Daí que esta redundância seja necessária para garantir que no caso de uma mensagem estar incompleta se consiga a partir da outra obter a informação desejada.

Estes dois tipos de mensagens devolvem aos dispositivos que estão a utilizar os receptores de *GPS* as seguintes estruturas de dados (para o caso das mensagens *GGA*):

```
$GPGGA,161229.487,3723.2475,N,12158.3416,W,1,07,1.0,9.0,M
```

onde `161229.487` é a hora universal, `3723.2475` a latitude (no formato graus, minutos e segundos), `N` a orientação (Norte ou Sul), `12158.3416` a longitude, `W` a orientação (Este ou Oeste), `1` a informação quanto ao estado da posição (se fixou ou não), `07` o número de satélites usados, `1.0` a precisão, `9.0` a altitude e `M` as unidades da altitude(neste caso, metros). No caso das mensagens *RMC*:

```
$GPRMC,161229.487,A,3723.2475,N,12158.3416,W,0.13,309.62,120598
```

[9] Global Positioning System Fixed Data
[10] Recommended Minimum Specific GNSS Data
[11] GNSS Satellites in View

onde `161229.487` é a hora universal, `A` o estado da posição (se é válida ou não), `3723.2475` a latitude, `N` a orientação, `12158.3416` a longitude, `W` a orientação, `0.13` a velocidade (em nós), `309.62` a orientação terrestre em graus e `120598` a data (no formato dia, mês, ano).

O grande desafio do *GPS* começa na fase de implementação do módulo de recepção e interpretação das mensagens *NMEA* enviadas pelos receptores de *GPS*. A utilização de receptores de *GPS*, quer estes sejam externos ou internos ao próprio *PDA*, necessita sempre de um canal de comunicação entre ambos. Tecnologicamente, a *Compact Framework 1.0* tem um sério problema a lidar com este tipo de comunicações, uma vez que não se encontra presente nesta um objecto ou classe que implemente os mecanismos de comunicação via porta série. Esta comunicação tem que ser realizada artificialmente simulando um acesso a um ficheiro. Na prática significa que, em vez do tradicional nome do ficheiro, é atribuído o nome da *COM* associada ao receptor de *GPS* a que pretendemos aceder:

```
m_HComm = CreateFile("COM5:", &HC0000000, 0, 0, 3, 0, 0)
```

neste caso em particular estamos a aceder ao receptor de *GPS* através da `COM5:`. A partir deste momento, consegue-se ler e escrever na *COM*, como se de um ficheiro de dados se tratasse.

Uma vez que as mensagens *NMEA* que estamos a ter em conta são enviadas pelo receptor de *GPS* em intervalos de tempo de 1 segundo, é necessário que exista um processo continuamente em execução que, de 0,5 em 0,5 segundos, verifique se existe na porta *COM* informação para ser processada. A opção por um intervalo de tempo com metade do valor do intervalo de tempo a que as mensagens são enviadas prende-se com o facto de querermos garantir que pelo menos em cada duas execuções do processo existe informação sobre a localização do receptor de *GPS*.

As duas classes necessárias para a extracção da informação de *GPS* encontram-se descritas no Apêndice A (secções A.1 e A.2) e permitem obter a seguinte informação (Tabela 4.4):

Propriedades	**Valores**
SatDate	12-08-00 03:17:36
Status	Awaiting Fix
Lat	40433101
Lng	073175308
Sats	6
Velocidade	0,000

Tabela 4.4: Exemplo dos dados obtidos das classes de GPS

De salientar que a propriedade *Status* sofre uma transformação de forma a ser mais legível o estado actual do receptor de *GPS*. Esta propriedade pode por isso tomar um dos

seguintes valores: **Awaiting Fix**, **Sats Ok** ou **GPS Disable**. Estes valores correspondem respectivamente a: tentar fixar posição, se tem uma posição fixa ou se não existe comunicação entre o *PDA* e o receptor de *GPS*.

No discurso que utilizamos no nosso dia-a-dia, em regra geral não nos referimos aos locais através das suas coordenadas geográficas, mas sim através de nomes que lhes estão associados. Frases como:

> Pai, mãe, vou com os colegas ao Centro Comercial

são bastante comuns, ao contrário de:

> Pai, mãe, vou com os colegas às coordenadas (30,402325;8,34568)

Como se verifica, as coordenadas por si só, não são informação. É então necessário transformar estas em conceitos com os quais estamos familiarizados, como é o caso do nome dos locais.

No projecto *Context Toolkit* [Dey and Abowd, 2000], Dey apresenta um conceito muito interessante, a ideia de abstracção da informação de contexto e parte dela é obtida recorrendo aos *Widgets* (secção 3.1), que encapsulam os dados dos sensores, convertendo-os em informação útil para o contexto. No nosso sistema, a classe `GPS` funciona como um desses *Widgets*, no entanto a latitude e longitude por si só não têm grande valor, é necessário que esta informação seja associada a um local. É necessário atribuir um nome ao par de coordenadas geográficas.

Uma forma de associar o nome do local ao par de coordenadas geográficas de *GPS* seria através da introdução desse mesmo nome por parte do utilizador. Por exemplo, o sistema sempre que encontrasse um novo local pediria ao utilizador que introduzisse o nome do local, ao qual estas coordenadas corresponderiam.

Esta abordagem tem no entanto um grande inconveniente. Se o objectivo da Computação Ubíqua é auxiliar o utilizador, evitando ao máximo a utilização da atenção do utilizador na interacção com os dispositivos ubíquos (relembrando o princípio do projecto *Aura* - secção 3.1.2 - que considera a atenção humana como o recurso mais precioso), obrigar o utilizador a associar um nome a cada novo local seria tudo menos pró-activo. Assim, descartámos esta abordagem e procurámos encontrar outra que não necessitasse da atenção do utilizador. Como para o nosso sistema o nome que é atribuído ao local é o menos importante, optámos pela criação dinâmica de locais abstractos. Vejamos, por exemplo, a situação apresentada na tabela 4.1, agora com os locais abstractos (Tabela 4.5).

Analisando a tabela anterior, verifica-se que não há qualquer perda de informação, uma vez que o que pretendemos saber é que, em determinado local, indiferentemente do nome que lhe é associado, existe um conjunto único de interacções com o *PDA*.

Local	**Interacção com o *PDA***
Place1	Sincroniza a agenda
	Tira apontamentos da reunião
	Faz uma apresentação
Place2	Faz uns telefonemas
	Responde às sms que recebeu
	Consulta o correio electrónico
Place3	Vê páginas da Internet
	Consulta correio electrónico
	Consulta a agenda para o dia seguinte
	Joga

Tabela 4.5: Abstracção do local

Assim, na implementação do nosso sistema, foi criada uma nova classe denominada de `DadosGPS` que faz a abstracção da classe `GPS` permitindo a criação e leitura dos locais onde o utilizador interage com o *PDA*, assim como a detecção do nome abstracto do local onde este se encontra (Apêndice A, secção A.3).

Como nem sempre é possível ter coordenadas válidas, quer seja pelo facto de o receptor *GPS* não conseguir fixar posição, quer seja pelo facto de o mesmo receptor se encontrar inactivo, nestas situações é impossível dizer em que local nos encontramos. Como forma de contemplar estas situações, considerámos a utilização da etiqueta `No Info`, a ser apresentada em vez do nome do local. O exemplo seguinte mostra uma dessas situações:

Current_Place	**Date**	**Time**	**Day**	**GPS**
No info	08-06-06	0:49:33	Thursday	GPS Disable
LatActual	**LongActual**	**VelocActual**	**Satellites**	**LatVal**
0	0	0	0	0
LongVal	**VelocVal**	**SatVal**		
0	0	0		

Tabela 4.6: Situação em que não é possível determinar o local

Para nos deslocarmos entre locais é necessário movimento. Temos então que considerar a situação em que não estamos num local específico, mas sim em movimento. Determinar o movimento é relativamente simples, basta aceder à propriedade `Velocidade` desta classe. Assim, considerámos que para valores de velocidades superiores a 10 nós (equivale a cerca de 18,5 kilómetros/hora), não considerávamos a posição obtida como um local, mas sim como `Moving` (Tabela 4.7).

Current_Place	**Date**	**Time**	**Day**	**GPS**
Moving	08-06-06	15:55:24	Thursday	SATS OK
LatActual	**LongActual**	**VelocActual**	**Satellites**	**LatVal**
40115454	8246317	97	4	40115454
LongVal	**VelocVal**	**SatVal**		
8246317	97	4		

Tabela 4.7: Utilizador em movimento

Como podemos ver pelas tabelas anteriores, este método devolve ao sistema um conjunto alargado de informação composta por: local, informação momentânea do *GPS* e informação válida do *GPS*. É ainda adicionada a informação da data, hora e dia da semana, como forma de marcar o instante em que a mesma informação foi adquirida. s Se pensarmos que na maior parte do dia encontramo-nos dentro de edifícios e como dentro destes o *GPS* tem dificuldades em adquirir uma posição, com o recurso às últimas coordenadas válidas conseguimos manter a informação sobre o último local. Considera-se assim que o utilizador mantém a mesma localização, desde que o receptor de *GPS* esteja em funcionamento. Recorrendo à propriedade `Status` da classe `DadosGPS`, é rapidamente possível saber o estado actual do receptor, uma vez que este só se encontra em funcionamento quando esta propriedade toma um dos seguintes valores: `Awaiting Fix` ou `SATS OK`.

Como o erro no cálculo da posição por parte dos receptores de *GPS* pode atingir os 15 metros, não podemos trabalhar apenas com o valor exacto das coordenadas para determinar o local onde o utilizador se encontra, uma vez que iria originar a criação exagerada de novos locais, que na prática se referiam apenas a um. Assim, para determinarmos se estamos num local previamente guardado, consideramos duas margens, uma superior e outra inferior, com igual distância aos valores actuais das coordenadas adquiridas no momento. Se os valores actuais das coordenadas ficarem dentro da área criada pelas duas margens, então significa que estamos num local conhecido (Figura 4.10).

A noção de superior e inferior, apresentada na figura, refere-se em particular à posição geográfica do nosso país, uma vez que as coordenadas de *GPS* têm como referência o meridiano de Greenwich[12] e a linha do Equador[13] (Figura 4.11). Assim, à medida que nos deslocamos para Oeste a longitude aumenta e no caso de nos deslocarmos para Norte regista-se um aumento ao nível da latitude.

A questão que se coloca é de quanto deverá ser esta margem e de que forma a variação das

[12]Linha que divide a Terra em Este e Oeste. Em coordenadas *GPS* ao meridiano de Greenwich corresponde a Longitude de 0 graus. Informação disponível em http://en.wikipedia.org/wiki/Longitude

[13]Linha que divide a Terra em dois Hemisférios, Norte e Sul. Em coordenadas *GPS* à linha do Equador corresponde a Latitude de 0 graus. Informação disponível em http://en.wikipedia.org/wiki/Latitude

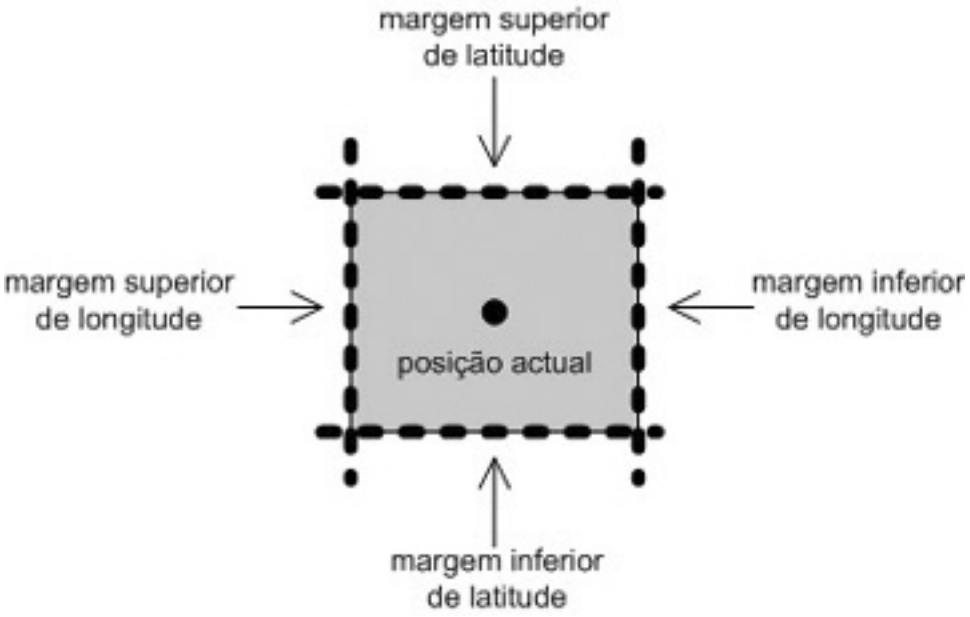

Figura 4.10: Utilização de margens para determinar o local

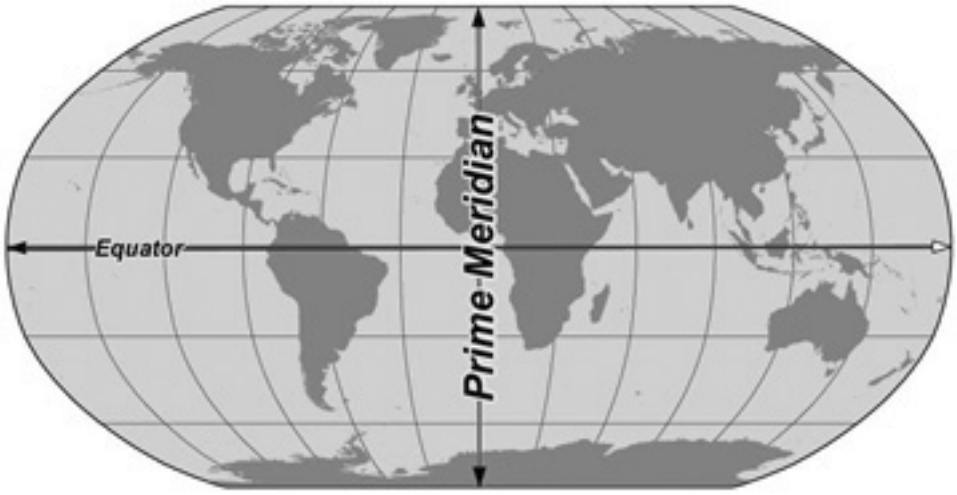

Figura 4.11: Representação geográfica da Terra

coordenadas da latitude e longitude representa ou não uma variação significativa em metros.

As coordenadas da latitude e longitude provenientes dos receptores *GPS* vêm segundo a seguinte representação: **grau**, **minuto** e **segundo**. A unidade segundo tem ainda duas casas decimais, que correspondem a centésimas de segundo. A tabela 4.8 apresenta de que forma é feita essa representação a partir dos dados obtidos do receptor de GPS, através de dois exemplos, um para a latitude e outro para a longitude.

Informação do GPS	Grau	Minuto	Segundo	Centésimas
40115454	40	11	54	54
8246317	8	24	63	17

Tabela 4.8: Representação das coordenadas

É então necessário determinar de quanto deve ser a variação ao nível das coordenadas, de forma a que a área que estamos a considerar como o local tenha algum sentido. De certa forma, existe uma relação directa entre a variação destas coordenadas e a sua correspondência em metros. Por razões associadas à própria geometria do globo terrestre, esta variação é diferente para a latitude e longitude. A tabela (Tabela 4.9) apresenta a correspondência entre as variações das várias unidades que compõem as coordenadas e a variação equivalente em metros[14].

Variação	Latitude	Longitude
1 grau	111,044 kilómetros	67,59 kilómetros
1 minuto	1,850 kilómetros	1,126 kilómetros
1 segundo	30,845 metros	18,78 metros

Tabela 4.9: Variação de coordenadas e correspondente variação em metros

Como podemos verificar pela tabela anterior a variação de 1 segundo nas coordenadas de latitude e longitude corresponde a uma variação de aproximadamente 30 e 19 metros, respectivamente. Como pretendíamos que a zona de acção de um local (`Place`) fosse definida por uma área de 100 metros, garantindo assim uma margem suficiente para a sua determinação, para cada local previamente guardado (Tabela 4.10) são determinadas as correspondentes latitudes e longitudes máximas e minímas, necessárias para representação dessa área (Tabela 4.11).

Local	Latitude	Longitude
Place1	40090000	8206593

Tabela 4.10: Exemplo da caracterização de um local

A figura 4.12 apresenta graficamente a área associada ao local.

Desta forma, sempre que é adquirido um novo par de coordenadas pelo sistema, é verificado se o mesmo está dentro destes limites, para que se possa assim determinar se estamos perante um local previamente guardado ou se é um novo local.

[14] *Understanding GPS Coordinates* - Informação disponível em http://www.co.lincoln.wa.us/GIS/GPSCoordinates.pdf

Local	**Latitude** Mínima	Máxima	**Longitude** Mínima	Máxima
Place1	40089834	40090166	8206330	8206856

Tabela 4.11: Limites que formam a área do local

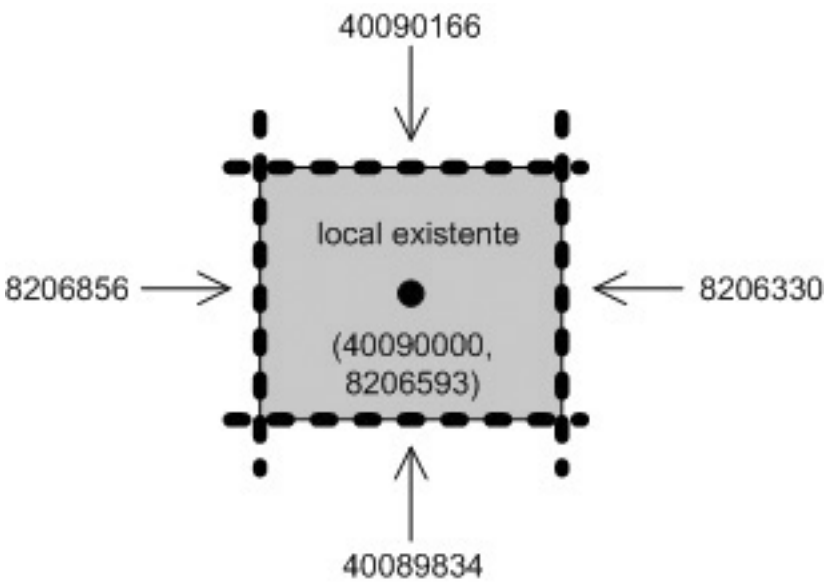

Figura 4.12: Representação gráfica da área do local

Terminada a implementação desta classe, faltava agora apenas a criação da classe principal de todo o sistema que tem como objectivo a execução de um processo que, a cada 15 segundos, obtém das classes `DadosGPS` e `Processo Activo` toda a informação considerada útil para definir o contexto do utilizador ao nível do local onde este se encontra, e da sua actividade computacional no *PDA*. Esta informação é guardada em ficheiro, para que posteriormente possamos utilizar a mesma para a avaliação dos algoritmos de aprendizagem. A definição completa da arquitectura do sistema encontra-se visível na figura 4.13.

Concluído assim o primeiro protótipo do nosso sistema é agora o momento de fazer os primeiros testes para validação do sistema, de forma a garantir a sua fiabilidade e validade, e garantir que a informação adquirida é correcta. Um exemplo da aquisição de informação de contexto que foi feita encontra-se disponível no apêndice B.

A informação descrita anteriormente resulta apenas de um único utilizador, no entanto foram realizados testes com diferentes utilizadores, que permitiram tirar algumas conclusões relativamente ao comportamento do nosso sistema.

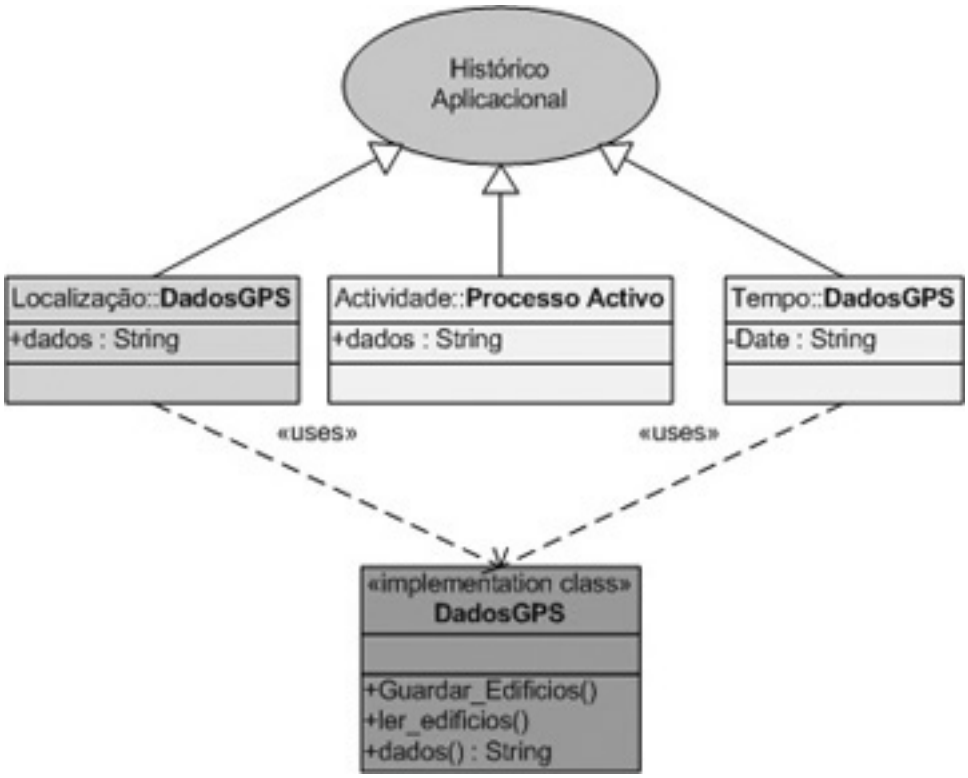

Figura 4.13: Arquitectura da Aplicação de Amostragem

Destas, as de maior importância foram as seguintes:

1 - Falha na aquisição de informação

O facto de guardarmos o instante em que a informação foi adquirida permitiu-nos constatar que existiam intervalos de tempo de tamanho variado, em que o sistema não realizava a aquisição da informação de contexto. Verificámos que esses intervalos correspondiam ao tempo em que o *PDA* se encontrava desligado.

Esta situação originou um requisito importante, era necessário manter o *PDA* sempre ligado para que fosse possível obter realmente registos de contexto em instantes de 15 segundos. Embora não seja uma opção agradável de ser tomada, por questões tecnológicas é a única forma de garantir a aquisição dos dados.

2 - Comunicação Bluetooth

A recolha de informação *GPS* para os vários utilizadores foi feita utilizando um receptor de *GPS* externo ao próprio *PDA* sendo necessário estabelecer uma comunicação entre os dois.

A grande maioria destes receptores utiliza para a comunicação a tecnologia *Bluetooht* que apesar de todas as suas vantagens, tem também os seus inconvenientes.

Durante os vários testes realizados nesta fase, observámos dois desses inconvenientes. O primeiro deles tem a ver com o facto de inexplicavelmente verificarmos que após algum tempo de utilização desta tecnologia, a comunicação entre os dispositivos deixava de ser possível. O problema neste caso pode muito bem estar do lado do *PDA*, uma vez que, os serviços de comunicação consomem demasiada bateria. Deparámo-nos com situações em que a bateria de um *PDA* que normalmente tinha uma autonomia superior a 24 horas, com os serviços de comunicação *Bluetooth* activos, era drasticamente reduzida para valores pouco superiores a 3 horas. Por esta razão, suspeitamos que seja o próprio *PDA* a terminar estes serviços, nos casos em que a autonomia da bateria atinja valores muito baixos, como forma de preservar a informação existente nestes. Não nos podemos esquecer que a informação guardada no *PDA* fica armazenada em memória volátil e quando a bateria se esgota, a mesma é perdida.

Uma vez perdida a comunicação entre o *PDA* e o receptor *GPS Bluetooth*, a única forma de reestabelecer a recepção de coordenadas, seria reiniciando a comunicação entre ambos.

Esta perda de comunicação tem duas desvantagens. A primeira é o facto do utilizador não poder, nem lhe podia ser exigido, que estivesse continuamente a verificar se a comunicação entre os dois dispositivos se mantém, de forma a detectar quando é que a mesma terminou. Como tal, durante os períodos em que não existe comunicação, não conseguimos determinar a localização do utilizador, uma vez que a propriedade `Status` da classe `DadosGPS` do nosso sistema toma o valor `GPS Disable` sempre que não existe comunicação com o receptor *GPS*, sendo por isso atribuído como local a informação `No info`. A segunda desvantagem, resulta da anterior. Tendo em conta que a maioria do tempo do nosso dia-a-dia é passado no interior de edifícios e pelo facto de ter sido perdida a comunicação, não existem por isso coordenadas válidas que permitam determinar o local actual, só voltando a ser possível obter o local quando o utilizador se encontrar no exterior, que pode significar um longo período sem qualquer informação sobre a sua localização. Temos sempre que nos recordar que dentro de edifícios muito dificilmente se consegue ter uma posição válida de *GPS*.

O segundo inconveniente da tecnologia *Bluetooth* está associado à forma como o nosso sistema lida com as comunicações via porta série. Como na *Compact Framework 1.0* o único meio disponível para realizar esta comunicação é recorrendo artificialmente ao acesso a ficheiros, também aqui se verifica o fenómeno idêntico ao descrito na primeira conclusão. Para que continuamente seja mantida a comunicação entre o *PDA* e o receptor de *GPS* é necessário que os mesmos estejam sempre ligados. Caso não se verifique, origina-se o problema descrito anteriormente de perda da informação sobre a posição actual do utilizador.

3 - Shell32

Nos primeiros testes realizados, verificámos algo interessante. O facto de existirem muitas ocorrências para um determinado processo, denominado `shell32`, que se repetiam em todos os utilizadores. No exemplo apresentado no apêndice B, referente à aquisição de dados para um único utilizador, podemos constatar vários registos onde este processo é considerado como um processo activo.

Como os utilizadores que realizaram estes testes têm por hábito minimizar as aplicações sempre que já não necessitam das mesmas, fomos levados a suspeitar que este processo estaria associado ao *TodayScreen*.

A presença deste processo ao longo das várias recolhas de informação que fizemos permite verificar as situações em que não existe actividade computacional no *PDA*, pois só assim se justifica a presença deste processo no atributo associado aos processos activos. Por esta razão, e pelo facto de existirem muitas ocorrências deste processo ao longo dos dias em que foi feita a recolha da informação de contexto, há uma indicação que a utilização aplicacional do *PDA* é ainda muito reduzida.

4 - Optimização da classe DadosGPS

A aquisição de dados de contexto por utilizadores do nosso grupo de investigação permitiu confrontar os resultados obtidos pela nossa aplicação de contexto, com o que de facto se passou ao longo dos dias de aquisição. Através do cruzamento desta informação com os dados recolhidos, podíamos assim verificar se as aplicações que eram executadas apenas em locais específicos correspondiam de facto aos resultados obtidos da aplicação e se estávamos a conseguir obter os vários locais por onde os utilizadores se movimentavam.

Analisando desta forma o comportamento da aplicação, verificámos que não estávamos a conseguir obter todos os locais como pretendido e que a aplicação considerava, como distintos, locais que na prática eram idênticos.

O exemplo seguinte demonstra uma situação em que, durante um espaço temporal de duas horas, foi utilizada algumas vezes a mesma aplicação, sendo que o utilizador apenas utiliza esta quando se encontra em casa. Na prática, o utilizador manteve-se sempre no mesmo local durante este período de tempo, no entanto a aplicação considerou seis locais diferentes (Tabela 4.12).

Daqui concluímos que a nossa aplicação necessitava de melhoramentos ao nível da detecção dos locais, o que implicava optimizações na classe `DadosGPS`. A primeira delas, foi aumentar a área associada ao local. Como a detecção do local tem por base as coordenadas válidas momentos antes de o utilizador entrar no edifício, é perfeitamente normal que devido à obstrução provocava pelos edifícios se torne mais difícil obter essas coordenadas. Como para nós não é importante que o local corresponda exactamente à porta do edifício, o aumento

	Local	**Data**	**Hora**	**Dia**	**GPS**	**LatActual**
	LongActual	**VelocActual**	**Satelites**	**LatVal**	**LongVal**	**VelocVal**
	SatVal	**Processo**				
1	Place70	09-06-06	0:35:43	Friday	SATS OK	40117665
	8248701	46	4	40117665	8248701	46
	4	PocketSudoku				
2	Place61	09-06-06	0:36:58	Friday	SATS OK	40117666
	8248723	10	5	40117666	8248723	10
	5	PocketSudoku				
3	Place73	09-06-06	2:4:3	Friday	SATS OK	40117569
	8248563	44	5	40117569	8248563	44
	5	PocketSudoku				
4	Place94	09-06-06	2:10:35	Friday	SATS OK	40117837
	8249639	28	5	40117837	8249639	28
	5	PocketSudoku				
5	Place103	09-06-06	2:17:37	Friday	SATS OK	40117847
	8249718	32	5	40117847	8249718	32
	5	PocketSudoku				
6	Place106	09-06-06	2:26:39	Friday	SATS OK	40117748
	8249169	46	4	40117748	8249169	46
	4	PocketSudoku				

Tabela 4.12: Exemplo de vários Places para o mesmo local

da área associada ao local não afecta significativamente o nosso trabalho. O que é para nós importante é determinar o local, se este fica mais longe ou mais perto da porta do edifício, é insignificante. Por esta razão o valor que era anteriormente de 50 metros (Figura 4.12), passou a partir deste momento a ser de 100 metros.

4.4.3 Global System for Mobile Communications

Como verificámos pelos primeiros testes realizados com o nosso sistema, basear a localização apenas em informação *GPS* levanta alguns inconvenientes. Quer seja pelo facto de perdermos a comunicação entre o *PDA* e o receptor de *GPS*, ou pela dificuldade em obter coordenadas de latitude e longitude válidas, existem longos períodos em que não é possível determinar o local onde o utilizador se encontra. Como para nós é importante determinar a localização do utilizador para podermos assim associar um conjunto de aplicações a cada local, é fundamental encontrar outras formas de localização que possa complementar a informação obtida via *GPS*.

Com a crescente convergência entre a tradicional agenda electrónica e o telemóvel, é hoje cada vez mais comum o aparecimento de modelos de *PDA* equipados com tecnologia móvel de comunicações, em particular o *GSM*[15].

[15]GSM - Global System for Mobile Communications

Embora esta tecnologia tenha como principal objectivo a comunicação de voz e dados, devido à sua grande difusão e elevada cobertura territorial, é por isso uma tecnologia muito interessante como forma de localização do utilizador. Observando projectos idênticos na área, como é o caso do *ContextPhone* [Raento et al., 2005] (secção 3.1.6), constata-se que a tecnologia *GSM* está já a ser utilizada com sucesso neste projecto. Por esta razão considerámos a utilização da tecnologia *GSM* como um segundo elemento de localização para o nosso modelo de contexto.

As redes *GSM*, conhecidas como redes de segunda geração (2G) e segunda geração e meia (2,5G), operam em três gamas de frequências de radio, 900 MHz (GSM900), 1800 MHz (GSM1800) e 1900 MHz (GSM1900). Esta tecnologia é representada sobre a forma de célula, não sendo por isso estranho a denominação de comunicação celular. Cada célula é denominada de estação base[16] e ao conjunto de sete estações base dá-se o nome de *cluster*. A configuração do *cluster* é repetida as vezes que forem necessárias até haver cobertura da zona que se pretende servir (Figura 4.14) [Noll, 1998].

Todas as comunicações de uma unidade móvel, vulgarmente denominada de telemóvel, passam pela estação base.

A grande vantagem da tecnologia *GSM* é permitir a mobilidade do utilizador. Esta mobilidade implica a movimentação das comunicações do utilizador ao longo das células. Por esta razão, as unidades móveis monitorizam a potência de sinal de todos os canais de rádio da zona envolvente. Este processo é denominado de *MAHO*[18].

Em cada mensagem *MAHO* gerada pela unidade móvel, é enviada uma lista de no máximo 24 estações base próximas da posição onde esta se encontra [Noll, 1998, Harte et al., 1998]. Esta informação poderia servir para determinar geograficamente a localização das unidades móveis, no entanto, utilizar as mensagens *MAHO* com esse objectivo levanta enormes dificuldades, quer seja pelo facto de ser difícil aceder a esta informação, ou pela dificuldade em determinar a localização devido às atenuações sofridas pelos sinais de rádio [Bento et al., 2005]. Por esta razão optámos por, em vez de recolher toda a informação de estações base e respectivos sinais de rádio, adquirir apenas informação sobre a estação base à qual a unidade móvel está actualmente associada. Este tipo de informação pode ser adquirida sem qualquer tipo de interacção do utilizador.

A comunicação entre as unidades móveis e as estações base é feita com o recurso ao envio e recepção de comandos, denominados de comandos AT[19] [Motorola, 2000].

A estrutura das mensagens de comandos enviadas pelas unidades móveis é composta por um prefixo, um corpo da mensagem e um terminador. O prefixo é formado pelos caracteres

[16] *Base Station*
[17] Imagem extraída de [Noll, 1998]
[18] Mobile Assisted Hand-Off
[19] AT - ATtention

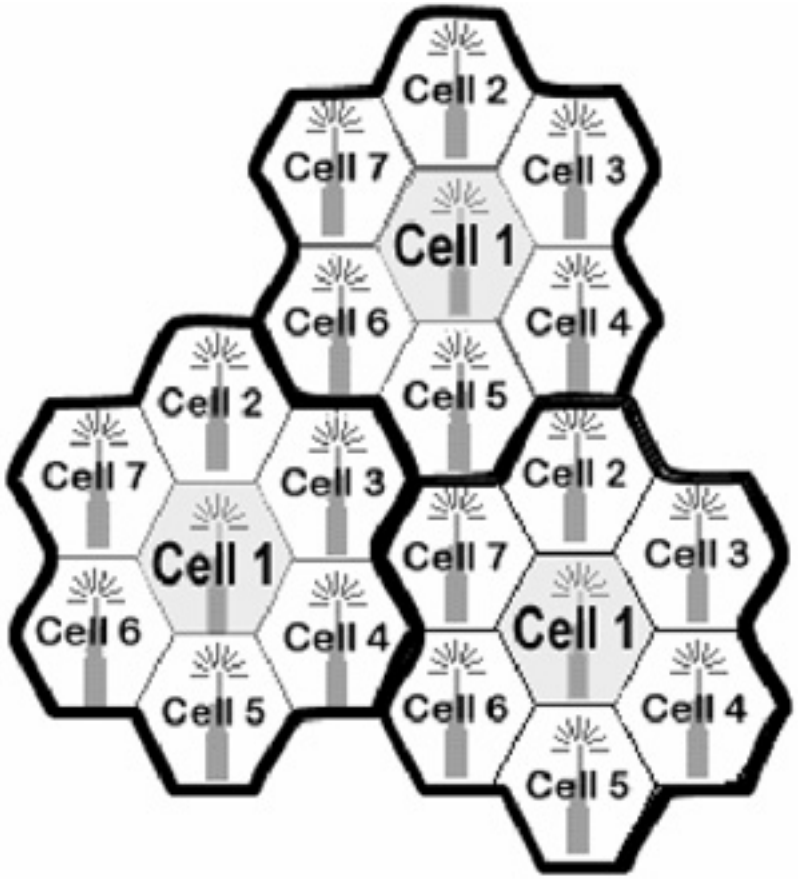

Figura 4.14: Configuração de *clusters* de *base stations*[17]

`AT` ou `at`. O corpo da mensagem é uma *string* formada por palavras-chave que indicam quais os comandos a serem executados. O terminador é composto pelo caracter `<CR>`. O exemplo seguinte demonstra uma dessas mensagens recorrendo a vários comandos AT:

```
ATCMD1 CMD2=12;+CMD1;+CMD2=,,15;+CMD2?;+CMD2=?<CR>
```

Como se verifica, a mensagem inicia-se com o prefixo, seguido de um conjunto de comandos separados pelo caracter `;` e por último, o terminador.

Posteriormente a resposta enviada pela estação base à unidade móvel tem a seguinte estrutura:

```
<TEXT><CR><LF>
```

Dentro da norma *GSM*, existe um conjunto básico de comandos que devem ser implementados nas unidades móveis. Percorrendo a lista de comandos disponíveis ao publico em geral [Motorola, 2000], constatámos que existia um que permitia determinar a identificação tanto

da célula actualmente associada à unidade móvel, como da área geográfica onde a estação base se encontra. Esse comando, denominado de `+CREG`, tem como objectivo registar uma unidade móvel na rede e/ou verificar o estado desse registo.

Para se realizar o registo na rede móvel, o comando tem a seguinte estrutura:

```
+CREG=[<n>]
```

onde `n` pode tomar os seguintes valores:

Valores	**Resultado**
0	desactiva o registo na rede
1	activa o registo na rede
2	activa o registo na rede e a informação de localização

Tabela 4.13: Valores possíveis para o registo na rede móvel

Ao enviar-se para a estação base o comando na forma `AT+creg=2<CR>`, estamos assim a registar a nossa unidade móvel e pedindo que seja activada a localização da estação base onde o registo é efectuado. Posteriormente é necessário testar o comando, enviando-o novamente à estação base, para que esta responda fornecendo a informação solicitada. Este teste é feito sobre a forma `AT+creg?<CR>`. Como estamos a questionar a estação base sobre o estado do registo na rede e respectiva localização desta, a estação base devolve à unidade móvel o mesmo comando com a seguinte estrutura:

```
+CREG: <n>,<stat>[,<lac>,<ci>]
```

onde `stat` define o estado do registo na rede, `lac` indica o código associado à área geográfica onde a estação base se encontra e `ci` indica o código identificador da célula.

Na prática, a informação da localização da estação base é feita a dois níveis. O primeiro nível é dado pelo `lac`, que identifica uma área mais alargada e o segundo dado pelo `ci` que reduz a localização à zona de acção da célula por ele identificada (Figura 4.15).

Desta forma, consegue-se assim obter não a localização da unidade móvel, mas a localização da estação base a que a mesma se encontra associada, embora normalmente esta estação base seja a que se encontra geograficamente mais próxima da unidade móvel. Por esta razão, a localização que conseguimos ter é no mínimo grosseira, uma vez que apenas conseguimos saber que estamos dentro da área de alcance de determinada estação base.

O alcance de uma estação base é muito variável dependendo das circunstâncias e pode atingir um valor máximo de 35 kilómetros, que é o alcance máximo que esta tecnologia permite devido a limitações técnicas. Na prática, as estações base estão agrupadas em áreas

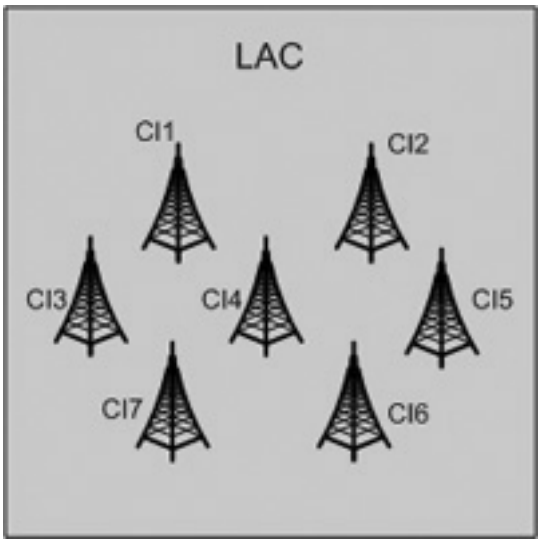

Figura 4.15: Representação do LAC e respectivas estações base associadas

de elevada densidade populacional, onde existe assim um número maior de potenciais utilizadores. Como o tráfego em cada estação base está limitado à sua capacidade (um número finito de comunicações em simultâneo), esta limitação afecta assim a distribuição espacial das estações base. Em áreas suburbanas é comum ter estações base com alcance de 2 a 3 kilómetros, enquanto que em áreas urbanas de maior densidade, o alcance é de apenas entre 500 a 1000 metros[20].

Como podemos verificar, a utilização da informação de localização da estação base associada à unidade móvel tem uma desvantagem muito grande, a sua precisão, que está muito longe da precisão que se consegue obter através do *GPS*. No entanto, como a tecnologia *GSM* tem uma cobertura territorial única e é por isso sempre possível determinar a localização da estação base, considerámos muito interessante a sua utilização como um elemento de localização no nosso modelo de contexto. Para o nosso trabalho, mais importante que a precisão, é ter disponível em qualquer instante pelo menos um elemento de localização.

Para que este elemento possa então ser um atributo da localização do utilizador, foi necessária a sua implementação através de uma classe denominada de `GSMInfo`. Esta classe tem como objectivo, sempre que seja solicitada, devolver tanto o código da área como o código de identificação da estação base.

Tal como acontecia com a classe `GPS_COM`, também para a implementação desta classe foi necessário criar artificialmente um canal de comunicação entre o *PDA* e o *modem* interno *GSM*, como forma de tornar possível o envio dos comandos AT para a estação base e posterior recepção dos resultados desse comando. A implementação desta classe encontra-se descrita

[20]Informação disponível na Wikipédia em http://en.wikipedia.org/wiki/Cell_site

na secção A.4 do Apêndice A.

Actualizando a arquitectura da nossa aplicação com esta nova classe, a mesma fica desta forma representada na figura 4.16.

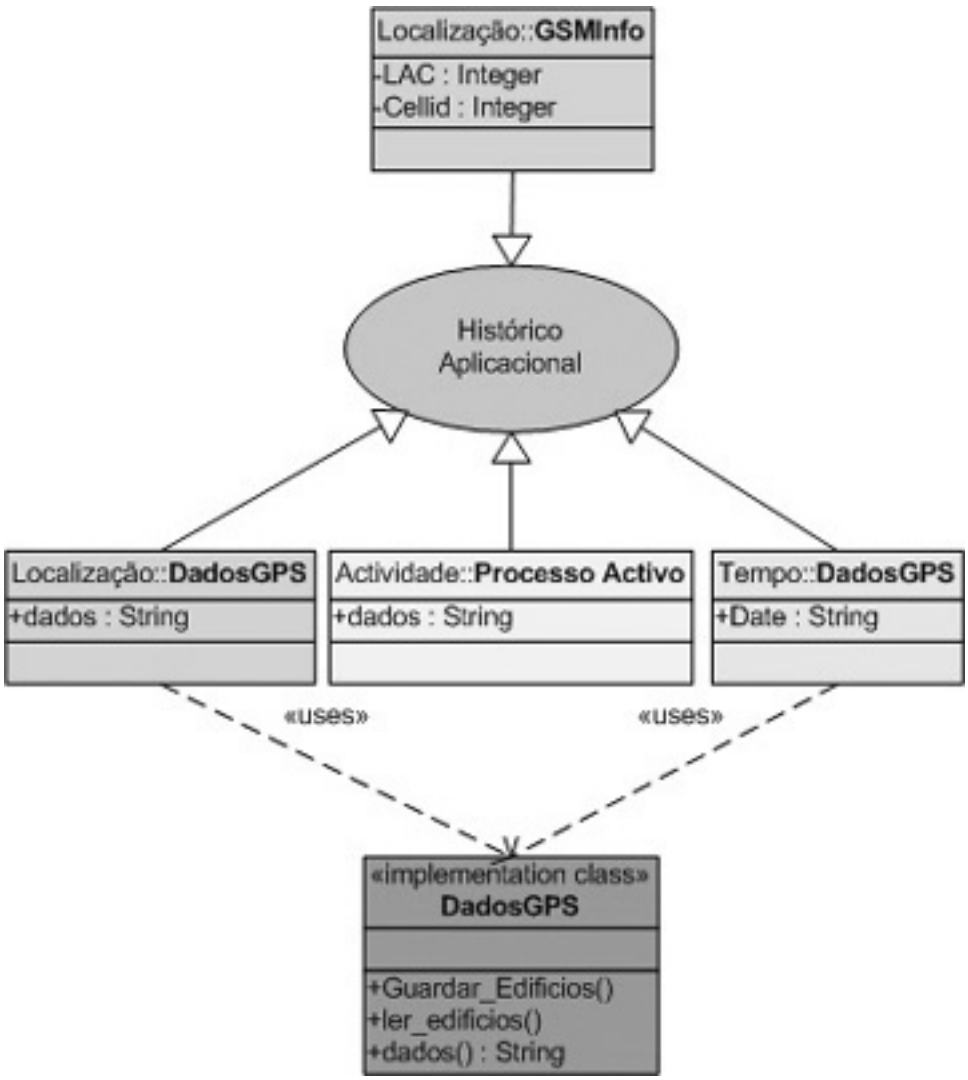

Figura 4.16: Arquitectura da Aplicação de Amostragem

Realizados novos testes agora com este segundo protótipo, verificámos situações muito interessantes relativamente à informação obtida via *GSM*. Vejamos o seguinte conjunto de dados (Tabela 4.14) obtidos durante apenas uma semana, para um único utilizador, que durante esse espaço de tempo se manteve na cidade de Coimbra. A tabela 4.14 apenas apresenta um pequeno extracto desses dados, mas que contém todas as estações base que foram identificadas e que são apenas de um único operador móvel. Não podemos esquecer que estas não são todas as estações base (`Cellid`) existentes na cidade, são apenas as que

foram associadas à unidade móvel de acordo com as movimentações do utilizador em algumas zonas da cidade.

	LAC	**Cellid**	**Local**	**LatVal**	**LongVal**	**SatVal**	**Processo**
1	15500	5591	No info	0	0	0	shell32
2	15500	5591	Place4	40115473	8246264	4	shell32
3	15500	5591	Place7	40115468	8246235	3	shell32
4	15500	5591	Place10	40115324	8246146	4	shell32
5	15500	5591	Place13	40115492	8246706	3	shell32
6	15500	5591	Place181	40112110	8249358	6	shell32
7	15500	5592	Place4	40115463	8246414	5	shell32
8	15500	5592	Place61	40117693	8248932	4	shell32
9	15500	5592	Place64	40117831	8248942	3	shell32
10	15500	5592	Place70	40117617	8248703	7	shell32
11	15500	5592	Place73	40117587	8248629	5	shell32
12	15500	5592	Place106	40117748	8249169	4	PocketSudoku
13	15500	6091	Moving	40111799	8240000	8	shell32
14	15500	6092	Moving	40111792	8249744	4	shell32
15	15500	6093	Place181	40112148	8249324	7	shell32
16	15500	11492	Moving	40114781	8243333	5	shell32

Tabela 4.14: Estações base identificadas em Coimbra para um único utilizador

O primeiro facto que se constata ao analisarmos a tabela 4.14 é a permanente existência de informação de localização obtida via *GSM* ao contrário do que acontece com a informação obtida via *GPS*.

A segunda constatação é a existência de pelo menos dois identificadores distintos de estações base associados a apenas um local obtido através da informação de *GPS*. Para o `Place4` por exemplo, temos dois identificadores de estações base, o `5591` e o `5592`. Esta situação acontece sempre que nos encontramos em locais onde a cobertura da rede móvel é feita por mais do que uma estação base e como vimos no início desta secção, é uma situação frequente como forma de garantir uma total cobertura da área, daí que a tecnologia *GSM* utilize a representação em célula para evitar que existam interferências entre canais de rádio com as mesmas frequências. Como tal, ter dois identificadores de estações base distintos para o mesmo local pode originar ambiguidade na informação e temos uma situação idêntica à que nos ocorreu no primeiro protótipo apenas com *GPS*. Mas no caso do *GPS*, conseguimos realizar optimizações de forma a minimizar esse problema, enquanto que no caso do *GSM* nada podemos fazer, uma vez que não temos qualquer controlo sobre a associação da unidade móvel à estação base.

Como forma de melhor compreendermos a informação de localização obtida, nada melhor do que representar graficamente a mesma. Para isso, foram utilizadas as coordenadas geográficas do Instituto Superior de Engenharia de Coimbra (ISEC) como ponto de referência. Com o recurso à nossa aplicação, obtivemos as seguintes coordenadas: `40115802` para a

latitude e `8246495` para a longitude.

Os valores de latitude e longitude dos vários locais adquiridos não permitem ter uma noção concreta das distâncias a que estes se encontram uns dos outros e dificulta a percepção das mesmas. Por esta razão, para o gráfico foi considerado como referência a unidade métrica com a qual estamos mais familiarizados, o metro. Assim e de acordo com a tabela 4.9, sabemos que a variação de um segundo de latitude corresponde a cerca de 30 metros, o que implica que se tivermos uma variação de 20 segundos, equivale a 600 metros de distância e por sua vez, uma variação de um segundo de longitude corresponde a cerca 19 metros, logo se variarmos 10 segundos resulta em 190 metros.

O resultado obtido da representação gráfica dos locais visitados por este utilizador encontra-se na figura 4.17.

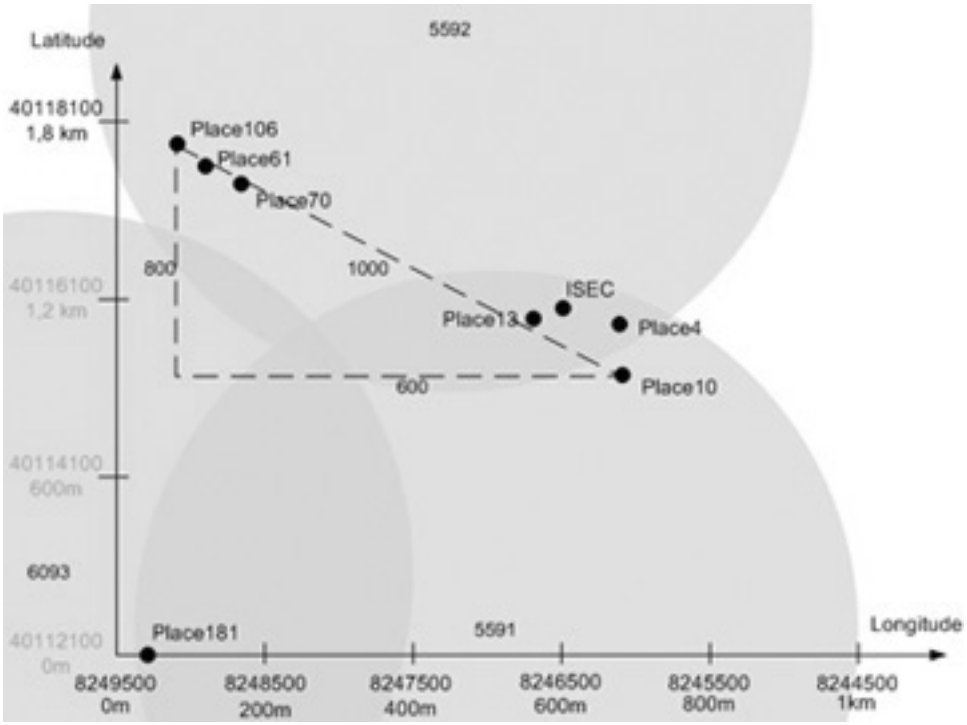

Figura 4.17: Representação gráfica da localização

No gráfico, além da representação dos locais segundo as coordenadas de *GPS*, está também presente a representação das áreas de cobertura *GSM* de acordo com a informação obtida pela nossa aplicação. Como podemos verificar, existem pelo menos três estações base que fazem a cobertura *GSM* para a zona da cidade de Coimbra que foi graficamente representada.

Esta representação gráfica permitiu-nos chegar a uma terceira conclusão. Verifica-se que, para uma área geográfica de apenas dois kilómetros por um kilómetro, existem pelo menos três identificadores distintos de estações base. Isto significa que, para locais idênticos dos

que foram obtidos em Coimbra, se consegue ter uma precisão bem inferior a 35 kilómetros. Como foi visto anteriormente, no pior dos casos uma precisão idêntica às áreas suburbanas, de 2 a 3 kilómetros. Se tivermos em conta o gráfico, é possível baixar ainda este valor para apenas 1 kilómetro nesta situação em particular. Conseguimos fazer a distinção entre dois locais (`Place106` e o `Place10`) utilizando apenas a informação de *GSM*, posicionados a uma distância de 1 kilómetro.

Embora possa parecer que a informação da localização obtida pela tecnologia *GSM* não venha a acrescentar nada ao nosso modelo, uma vez que tem uma precisão muito inferior à do *GPS*, que já estava anteriormente a ser utilizada, a figura seguinte demonstra exactamente o contrário. Com a adição desta informação, conseguimos obter uma localização hierarquizada (Figura 4.18).

Esta hierarquia permite que em situações de impossibilidade de obtenção de informação de localização através do *GPS*, seja mesmo assim possível determinar, com a tecnologia *GSM*, uma localização, embora mais alargada.

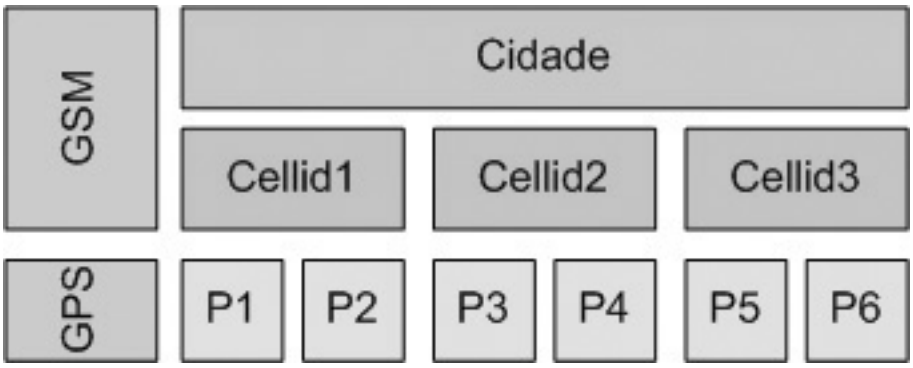

Figura 4.18: Hierarquia de localizações

4.4.4 WiFi

Com este segundo protótipo da nossa aplicação, já conseguimos ter dois elementos para caracterizar o contexto ao nível da localização e um elemento para descrever a actividade computacional no *PDA*. No entanto, sentíamos que podíamos ir mais longe ao nível da localização.

A informação de *GPS* já permitia determinar a posição do utilizador no exterior e mesmo nos casos em que esta não estava disponível, conseguíamos, recorrendo à informação obtida pela tecnologia *GSM*, saber com muito menor precisão onde este se encontrava, tanto no exterior como no interior de edifícios. Mas a localização no interior era ainda uma lacuna. Se, por exemplo, não conseguíssemos obter uma posição válida à entrada do edifício, tornava-se muito difícil apenas com *GSM* determinar se o utilizador estava ou não no seu interior

e como a localização utilizando *GSM* devolve áreas alargadas, o melhor que conseguíamos determinar era uma zona da cidade onde o utilizador se encontra.

Tal como acontece com a tecnologia *GSM*, que cada vez mais se encontra presente nos *PDA*s, também a tecnologia *Wi-Fi*[21] tem vindo a estar gradualmente disponível num maior número de modelos deste tipo de dispositivos.

Embora também esta tecnologia não tenha como objectivo principal a localização de utilizadores, em projectos dentro da área de Computação Ubíqua tem se verificado uma crescente utilização para este fim. Veja-se o caso de projectos como por exemplo o *Place Lab* [Hightower et al., 2006] (secção 3.1.5).

O termo *Wi-Fi* é uma marca registada pela *Wi-Fi Alliance*[22] e descreve, com base nas especificações **IEEE 802.11**, as tecnologias para redes locais sem fios, conhecidas por *WLAN* ou *Wireless LAN*.

O objectivo do desenvolvimento do *Wi-Fi* era permitir que dispositivos de computação móvel, como portáteis e *PDA*s, pudessem utilizar os recursos de uma rede local, sem perderem a sua mobilidade, através do envio e recepção de informação por ondas de rádio. Um dispositivo dotado desta tecnologia pode, por exemplo, aceder à internet desde que se encontre na proximidade de um ponto de acesso[23]. Uma região coberta por um ou mais pontos de acesso é denominada de *hotspot*. O alcance dos *hotspots* pode ir desde a área de apenas uma sala até à cobertura total de uma cidade.

A configuração típica de uma *WLAN* contém um ou mais pontos de acesso (APs) e um ou mais utilizadores móveis. Os pontos de acesso difundem para estes, a cada 100 milisegundos, informação sobre a identificação da rede. Esta identificação é denominada de *Service Set Identifier* (*SSID*). É com base no *SSID* que os utilizadores decidem se devem ligar ao ponto de acesso. Se na área envolvente ao utilizador existir mais do que um ponto de acesso com o mesmo *SSID*, os adaptadores de rede dos dispositivos móveis podem escolher a que ponto de acesso fazem a ligação com base na potência de sinal recebida de cada um destes[24].

É com base no identificador *SSID* que pretendemos obter a localização do utilizador. O objectivo é que cada identificador de rede possa caracterizar um local único. Só que este identificador de rede não é único, antes pelo contrário, verifica-se muitas vezes que algumas das *WLAN*s mantêm o *SSID* atribuído pelo fabricante. Desta forma, é necessário resolver esta ambiguidade e foi por isso que considerámos que devemos recolher o máximo de informação sobre a *WLAN* à qual o dispositivo móvel está ligado. Acreditamos que quanto mais personalizada for essa informação, mais facilmente conseguimos identificar essa rede como sendo única.

[21] Wireless Fidelity - comunicação sem fios utilizando frequências de rádio
[22] http://www.wi-fi.org/
[23] AP-Access Point
[24] Informação disponível na Wikipédia em http://en.wikipedia.org/wiki/Wi-Fi

	Nome	IP	Subnet	DHCP	SSID
1	No WiFi	0	0	0	0
2	TIACXWLN1	169.254.63.42	255.255.0.0	10.0.0.138	guest-e-U
3	TIACXWLN1	169.254.63.42	255.255.0.0	10.0.0.138	
4	TIACXWLN1	169.254.63.42	255.255.0.0	10.0.0.138	jnfire
5	TIACXWLN1	10.0.0.2	255.255.255.0	10.0.0.138	SpeedTouch0FD108
6	TIACXWLN1	192.168.1.7	255.255.255.0	192.168.1.1	default
7	TIACXWLN1	10.21.203.223	255.255.0.0	10.21.0.1	guest-e-U
8	TIACXWLN1	10.21.203.223	255.255.0.0	10.21.0.1	
9	TIACXWLN1	169.254.63.42	255.255.0.0	10.21.0.1	
10	TIACXWLN1	169.254.63.42	255.255.0.0	10.21.0.1	jnfire
11	TIACXWLN1	169.254.63.42	255.255.0.0	10.21.0.1	CFAC
12	TIACXWLN1	169.254.63.42	255.255.0.0	192.168.1.1	
13	TIACXWLN1	169.254.63.42	255.255.0.0	192.168.1.1	CFAC
14	TIACXWLN1	169.254.63.42	255.255.0.0	192.168.1.1	jnfire

Tabela 4.15: Resultados resumidos e intermédios da tecnologia *Wi-Fi*

O aumento de modelos de *PDA*s com um maior número de serviços de comunicação, como é o caso do *Wi-Fi*, está a originar um aumento da quantidade de aplicações que utilizam este tipo de serviços. O facto de existir um conjunto de aplicações cujo funcionamento depende da utilização de serviços de comunicação, levou-nos a considerar uma hipótese. O comportamento aplicacional do utilizador pode variar de acordo com a utilização ou não dos serviços de comunicação, em particular, o *Wi-Fi*. Com o objectivo de verificar esta hipótese, a aquisição da informação relativa ao *Wi-Fi* ganhou ainda mais importância. Se já é interessante verificar se conseguimos ou não determinar uma localização com esta tecnologia, mais interessante ainda é verificar se existem aplicações que apenas são executadas quando os serviços de comunicação associados a esta tecnologia estão activos.

A implementação desta classe encontra-se disponível no apêndice A, secção A.5.

Pelo facto da lista de adaptadores de rede apenas incluir os que estão no momento activos, permite-nos determinar uma característica muito interessante, a de saber em que instantes os serviços de comunicação associados à tecnologia *Wi-Fi* se encontram em utilização, e desta forma tentar verificar se o comportamento aplicacional do utilizador é diferente.

Os testes iniciais de aquisição de informação da tecnologia *Wi-Fi* por parte da nossa aplicação, permitiram obter os resultados apresentados na tabela 4.15.

Como já tínhamos referido anteriormente, muito dificilmente se conseguiria através do *SSID* associar uma rede *WLAN* a um único local. Estes resultados são disso uma prova. Mas a conclusão pode ainda ir mais longe, é que mesmo com este conjunto de atributos, não nos é possível identificar como única uma rede *WLAN*.

Os resultados anteriores já permitem dar por concluída a primeira parte, a detecção da presença do *Wi-Fi*, mas falta ainda implementar com sucesso um mecanismo de localização recorrendo a esta tecnologia.

Como os pontos de acesso possuem também um endereço *MAC* que os torna únicos, julgamos ser uma boa abordagem para conseguirmos assim personalizar cada uma das redes às quais o utilizador se vai ligando. O mecanismo de localização passa assim por associar um local a um ou mais endereços *MAC* dos pontos de acesso (Apêndice A, secção A.5).

Capítulo 5

Experimentação

5.1 Introdução

O trabalho da experimentação apresentada nas próximas secções deste capítulo tem por base estudar quais os algoritmos de aprendizagem que melhores resultados obtêm na determinação da correlação entre a localização do utilizador, a existência de recursos de comunicação e a actividade computacional no *PDA*. Por esta razão, os testes vão incidir sobre os algoritmos de classificação, associação e *clustering*.

Para a realização desta experimentação é necessário definir a política de divisão de dados para treino e para teste. A forma tradicional de o fazer é dividir o conjunto total em 2/3 para treino e 1/3 para teste. Esta estratégia, conhecida por *leave-one-out* tem no entanto as suas desvantagens. A principal é o facto dos dados presentes no conjunto de treino e teste poderem não ser representativos das várias classes. No pior do casos, podemos ter uma classe representada no conjunto de teste e não no conjunto de treino. Como não é possível dizer quais as instâncias que são ou não representativas, o melhor mesmo é garantir que as várias classes existentes no conjunto total de dados se encontram representadas na medida certa tanto para o conjunto de treino como de teste. A forma de o fazer é recorrendo à técnica *cross-validation*. Esta técnica permite dividir os dados em partições iguais, *folds*, sendo cada uma delas usada para teste e as restantes para treino. Por exemplo, se definirmos que queremos três partições, os dados são divididos em duas partições para treino e uma para teste, e o teste é repetido três vezes garantindo assim que cada partição de teste fez também parte do conjunto de treino. O valor mais comum para as partições é 10, não existindo uma teoria que o comprove, no entanto, testes exaustivos com numerosos conjuntos de dados e diferentes algoritmos de aprendizagem mostraram que este valor é o mais adequado [Witten and Frank, 2005]. No final dos testes, os valores medidos para os vários algoritmos

de aprendizagem são a média de 10 execuções. Esta técnica é a mais indicada para determinar a taxa de erro dos vários algoritmos de aprendizagem [Witten and Frank, 2005].

Como o *WEKA* apenas realiza automaticamente a *cross-validation* para os algoritmos de classificação, para os restantes algoritmos de associação e de *clustering* vamos utilizar a técnica *leave-one-out*.

Embora inicialmente o objectivo fosse disponibilizar a Aplicação de Amostragem a um número considerável de utilizadores, verificámos que devido a várias contrariedades, o número de utilizadores não é o que mais desejaríamos. O número de utilizadores de *PDA*s que dispõem de receptor de *GPS* é ainda reduzido e as dificuldades tecnológicas originadas pela utilização de *Bluetooth* levaram à desmotivação de muitos dos potenciais utilizadores. O facto da Aplicação de Amostragem necessitar de constante monitorização da recepção de sinal *GPS* implica um esforço considerável por parte do utilizador, o que não é de todo agradável. Como a aquisição da informação de *GSM* estava dirigida para uma marca de *PDA*s em particular, a aplicação não pode ser executada estavelmente noutros modelos. Mas mesmo que não existissem estas contrariedades, devido à heterogeneidade de utilizadores (tecnologias disponíveis e tipo de utilização) era necessário ter disponível um grande número de utilizadores, de forma a garantirmos um conjunto considerável de utilizadores para cada um dos diferentes grupos de utilizadores, o que apenas com voluntários seria muito difícil.

Embora seja um facto que o número de utilizadores é reduzido, no entanto, a amostra que obtivemos permite ter uma diversidade de situações, permitindo assim estudar o comportamento dos algoritmos com diferentes atributos e diferentes actividades computacionais.

5.2 Aquisição de dados

Como as tecnologias existentes nos diferentes modelos de *PDA*s variam de modelo para modelo e atendendo às diferentes formas de utilização que são dadas a estes dispositivos, o nosso objectivo fundamental para a aquisição de informação de contexto passa por ter exemplos diferenciados de ambas as situações. A tabela 5.1 apresenta a caracterização dos utilizadores que executaram a nossa aplicação de acordo com as diferentes tecnologias disponíveis e o número de aplicações distintas que foram utilizadas durante o teste.

No Apêndice C está disponível um exemplo resumido dos dados adquiridos, neste caso em particular para o Utilizador7, com as diferentes tecnologias disponíveis para a caracterização da localização do utilizador. Embora esteja a informação sobre a localização baseada em *GPS*, a mesma encontra-se nula, uma vez que o utilizador não dispunha de receptor de *GPS*.

Utilizador	GPS	GSM	WI-FI	APS <10	10<APS <20	APS >20
Utilizador1	X			X		
Utilizador2	X				X	
Utilizador3	X					X
Utilizador4			X		X	
Utilizador5	X			X		
Utilizador6	X	X			X	
Utilizador7		X	X	X		
Utilizador8	X	X			X	

Tabela 5.1: Caracterização dos utilizadores

5.3 Análise prévia dos dados

Como a hipótese que queremos verificar relaciona os locais com a conectividade e actividade aplicacional do *PDA*, a nossa representação do conhecimento passa por incluir os seguintes atributos, obtidos a partir dos dados adquiridos pela Aplicação de Amostragem:

Atributo	Tipo de Atributo	Formato do Atributo
LAC	Entrada	Numérico
CellID	Entrada	Numérico
Wifi	Entrada	Booleano
APs	Entrada	Nominal
Place	Entrada	Nominal
Process	Saída	Nominal

Tabela 5.2: Atributos utilizados para a representação do conhecimento

onde o `LAC` e `CellID` resultam da informação da célula *GSM*, `Wifi` se existe comunicação *WiFi*, `APs` os endereços *MAC* dos pontos de acesso, `Place` o local obtido através da informação de *GPS* e `Process` o processo actualmente em execução.

Analisando os dados recolhidos, verifica-se uma forte presença do processo `shell32` em todos os utilizadores. Este processo significa que nesse instante não existe actividade computacional no *PDA*. Pelo gráfico da figura 5.1, é possível verificar que em grande parte do tempo em que foi adquirida informação de contexto, não existiu qualquer interacção entre este e o utilizador.

Outra constatação que este processo permite elaborar é a existência de locais onde não existe qualquer actividade computacional no *PDA*. A tabela 5.3 apresenta os dados agregados por local para três dos utilizadores, contendo o número de locais distintos, com e sem o `shell32`.

Como este processo não está associado a nenhuma actividade, foi retirado dos vários conjuntos de instâncias que são utilizadas para os testes aos algoritmos de aprendizagem. Desta forma, para cada utilizador temos o número de instâncias apresentado na tabela 5.4.

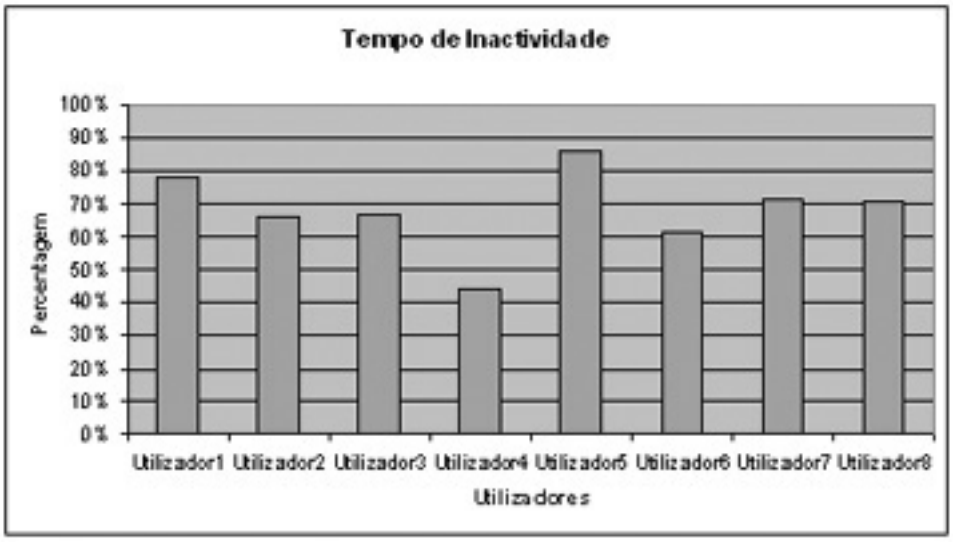

Figura 5.1: Percentagem de Inactividade

Utilizador	Locais totais	Locais apenas com actividade
Utilizador1	38	15
Utilizador2	46	31
Utilizador3	94	62

Tabela 5.3: Locais visitados e locais com actividade

Utilizador	Número de Instâncias
Utilizador1	924
Utilizador2	1336
Utilizador3	4958
Utilizador4	6184
Utilizador5	146
Utilizador6	1699
Utilizador7	653
Utilizador8	2266

Tabela 5.4: Número de instâncias utilizadas para a experimentação

5.4 Algoritmos de Classificação

A primeira experiência que vamos realizar visa estudar se o nosso problema pode ser solucionado com algoritmos de classificação. Estes algoritmos têm como objectivo, a partir dos atributos de entrada, encontrar similaridades e padrões de forma a prever a correspondente classe, associada aos atributos de saída. Caso o nosso problema seja de classificação, então pretendemos determinar quais os algoritmos mais adequados para a nossa representação de contexto.

5.4.1 Teste à categoria dos atributos

A informação de contexto pode ser representada com informação de diferentes categorias, podendo esta ser em formato numérico, booleano ou então nominal. Por esta razão, o primeiro teste realizado tem por objectivo verificar dos vários algoritmos implementados no *WEKA*, aqueles que permitem dados de diferentes formatos. Se o objectivo é estudar os algoritmos que mais se adequam ao contexto, temos que ter em consideração que a representação deste não é feita apenas num único formato.

Tendo como base os dados apresentados no Apêndice C, que apenas ilustra parte da informação que foi adquirida para o utilizador, obtemos uma situação em que estão disponíveis todas as tecnologias que foram consideradas para a nossa aplicação de amostragem, sendo por isso o caso ideal para fazer este teste.

Executando o teste com os dados na forma apresentada na tabela 5.2, os resultados obtidos no *WEKA* permitem-nos afirmar que os algoritmos apresentados na tabela 5.5 devem ser imediatamente excluídos da bateria de testes, pelo facto de não cumprirem com o requisito apresentado anteriormente.

Modelo de Classificação	**Algoritmo**	**Problema detectado**
Árvores de Decisão	ADTree	Apenas aceita classes binárias
	Id3	Não aceita atributos numéricos
	M5P	Não aceita atributos nominais
	UserClassifier	A criação do modelo é feita pelo utilizador
Regras de Classificação	M5Rules	Não aceita classes nominais
	Prism	Não aceita atributos numéricos
Bayesianos	AODE	Não aceita atributos numéricos
	ComplementNaiveBayes	Não aceita atributos nominais
	NaiveBayesMultinomial	Não aceita atributos nominais
	NaiveBayesSimple	Não aceita menos que dois valores para uma classe
	LBR	Não aceita atributos numéricos
Funções	SimpleLinearRegression	Não aceita atributos nominais
	LinearRegression	Não aceita classes nominais
	LeastMedSq	Não aceita classes nominais
	PaceRegression	Só aceita atributos binários
	VotedPerceptron	Só aceita classes binárias
	Winnow	Não aceita atributos numéricos
	SMOreg	Não aceita classes nominais

Tabela 5.5: Teste aos atributos

Desta forma, os próximos testes vão incidir sobre os algoritmos apresentados na tabela 5.6.

De salientar que todos os algoritmos baseados em Instâncias passaram o teste anterior.

Modelo de Classificação	Algoritmos
Árvores de Decisão	DecisionStump, J48, LMT, NBTree, RandomForest, RandomTree, REPTree
Regras de Classificação	ConjunctiveRule, DecisionTable, JRip, NNge, OneR, PART, Ridor, ZeroR
Instâncias	IB1, IBk, KStar, LWL
Bayesianos	BayesNet, NaiveBayes, NaiveBayesUpdateable
Funções	Logistic, SimpleLogistic, RBFNetwork, MultilayerPerceptron, SMO

Tabela 5.6: Conjunto de algoritmos a testar

5.4.2 Classificação com apenas dois atributos

Tendo por base os dados obtidos directamente da nossa Aplicação de Amostragem, verificámos que seria muito difícil determinar com sucesso uma correlação entre um local e uma aplicação. Como as aplicações não são executadas de forma exclusiva para um único local, uma vez que existem aplicações comuns a mais do que um local, assim como locais com mais do que uma aplicação, treinar um modelo de aprendizagem que consiga determinar com sucesso qual a aplicação que vai ser executada tendo por base a localização é um grande desafio. Vejamos os exemplos seguintes de instâncias adquiridas que ilustram bem este problema.

Place	Process
Place25	poutlook
Place43	poutlook
Place43	tmail
Place43	fexplore

Tabela 5.7: Exemplos de instâncias

Para este teste foram utilizados os dados dos utilizadores que tinham informação de *GPS*, uma vez que é o atributo de localização que tem maior frequência no conjunto dos vários utilizadores.

Resultados do treino

Umas das formas de avaliar a qualidade do modelo de aprendizagem é verificar a sua taxa de sucesso. Para isso, basta contabilizar o número de instâncias que foram correctamente classificadas e fazer a respectiva correspondência com as instâncias totais do treino. O gráfico da figura 5.2 apresenta a percentagem de sucesso obtida pelos diversos algoritmos estudados.

Nele podemos verificar que os algoritmos *IB1*, *ZeroR* e *NNge* são os que piores resultados obtiveram e mesmos os restantes não obtiveram resultados satisfatórios, o que só vem comprovar a hipótese que tínhamos colocado no início desta secção, que seria muito difícil treinar modelos de classificação com estes conjuntos de dados.

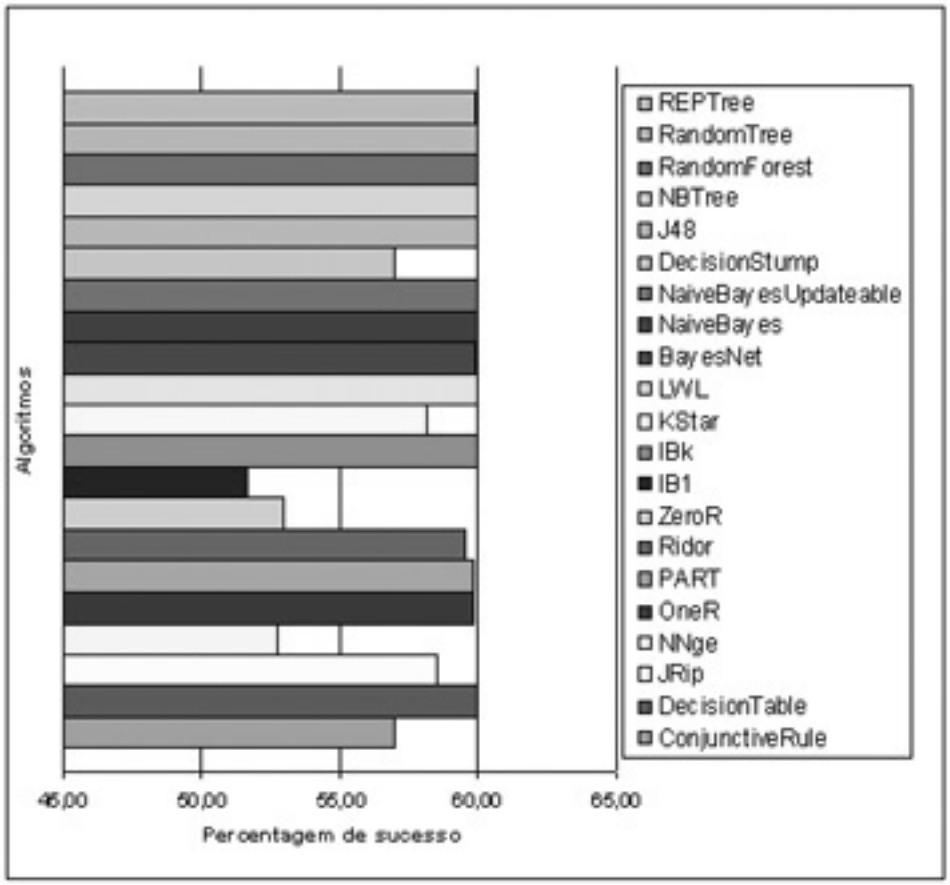

Figura 5.2: Percentagem de sucesso

De forma a compreender melhor o comportamento dos algoritmos que testámos, vejamos como exemplo os dados do Utilizador5, que tem 7 `Place` distintos para apenas 3 `Process`. Executando um teste a alguns dos algoritmos, verificamos a seguinte taxa de sucesso (Tabela 5.8).

Algoritmo	**Taxa de sucesso**
J48	67%
NBTree	67%
RandomTree	67%
DecisionTable	67%
JRip	67%
BayesNet	67%
IBk	67%

Tabela 5.8: Taxa de sucesso para os dados do Utilizador5

Todos eles obtiveram a mesma taxa de sucesso e inclusivamente a mesma matriz de

a	b	c	Classificado como
0	0	8	a = tmail
0	0	40	b = cprog
0	0	98	c = PocketSudoku

Tabela 5.9: Matriz de concordância do Utilizador5

concordância (Tabela 5.9). Esta matriz apresenta de que forma foram classificadas, na fase de teste, as várias classes obtidas através do treino do modelo de aprendizagem. Ao analisarmos os resultados obtidos pelos algoritmos *J48* e *JRIP* verificamos que apenas obtivemos uma regra e que esta contempla o processo que obteve maior número de ocorrências (Tabela 5.10).

Algoritmo	Regras
J48	: PocketSudoku.exe (146.0/48.0)
JRIP	: Process=PocketSudoku.exe (146.0/48.0)

Tabela 5.10: Regras obtidas pelos algoritmos de aprendizagem

Como o objectivo dos algoritmos de classificação é obter um modelo com a maior taxa de sucesso possível e uma vez que os dados com os quais estamos a realizar a experiência contêm para o mesmo atributo `Place` valores distintos do atributo `Process`, o comportamento dos algoritmos resume-se a determinar, para cada `Place`, o `Process` com maior frequência. Mas se chegámos a esta conclusão então é verdade que não é necessário nenhum modelo de aprendizagem, uma vez que, com um simples algoritmo, obtemos idênticos resultados (Tabela 5.11).

```
Enquanto existirem instâncias
    Para cada instância
        Se o Place já existe
            Se o Process já existe
                Incrementar o número de ocorrências
Para cada Place
    Escolher o Process com maior ocorrências
```

Tabela 5.11: Algoritmo equivalente

5.4.3 Classificação com os dados agregados

Pelos resultados anteriores demonstra-se que a representação apresentada na tabela 5.7 não permite determinar a correlação por nós pretendida. Nestes exemplos, verificámos que para o mesmo utilizador existem locais distintos associados à mesma aplicação, assim como várias aplicações associadas ao mesmo local. Este facto sustenta a hipótese de que, a existir uma correlação entre aplicações e locais, esta deverá ser na forma:

$$Loc_1, Loc_2, ..., Loc_n \rightarrow \{Apl_1, Apl_2, ..., Apl_n\} \tag{5.1}$$

onde a partir de vários atributos de localização `Loc`, se corresponde uma lista de aplicações `Apl` que são executadas associadas a esse local. Por exemplo,

$$15500, 5592, Place1 \rightarrow \{PocketSudoku, cprog, tmail\} \tag{5.2}$$

onde `15500` corresponde ao atributo `LAC`, `5592` ao atributo `CellID`, `Place1` ao atributo `Place` e `PocketSudoku, cprog, tmail` à lista de aplicações. Como o objectivo é determinar as 5 aplicações mais comuns, a lista `Apl` é então limitada a esse valor.

Assim, a próxima experiência pretende verificar se esta formatação dos dados permite verificar essa correlação. De forma a representar os dados na formatação referida, é necessário agregar os mesmos segundo a informação de localização, com o objectivo de determinar as aplicações que têm maior frequência para cada situação de localização em particular, extraindo posteriormente as 5 aplicações com maior frequência.

O processo de agregação consiste em determinar todas as instâncias distintas que existem no conjunto de dados e contabilizar o número de ocorrências para cada uma dessas instâncias. A tabela 5.12 demonstra como é feita a agregação.

LAC	**CellID**	**Place**	**Process**	**Ocorrências**
15500	5592	No info	PocketSudoku	63
15500	5592	No info	cprog	40
15500	5592	Place61	PocketSudoku	14
15500	5592	Place94	PocketSudoku	8
15500	5592	No info	tmail	8

Tabela 5.12: Exemplo dos dados agregados para o Utilizador5

Ao ser realizada a transformação para o formato apresentado na equação 5.1 permite-nos obter a informação apresentada na tabela 5.13. Os vários resultados obtidos através da agregação dos dados encontram-se disponíveis no Apêndice D.

LAC	**CellID**	**Place**	**Processes**
15500	5592	No info	PocketSudoku, cprog, tmail
15500	5592	Place61	PocketSudoku
15500	5592	Place94	PocketSudoku
15500	5592	Place73	PocketSudoku
15500	5592	Place70	PocketSudoku
15500	5592	Place103	PocketSudoku
15500	5592	Place106	PocketSudoku

Tabela 5.13: Informação agregada para o Utilizador5

Se compararmos esta informação com a matriz apresentada na tabela 5.9 facilmente se percebe que a taxa de sucesso é muito superior uma vez que estão contempladas todas as hipóteses quer ao nível dos processos, quer ao nível da localização.

A tabela 5.13 também permite retirar outra conclusão. Se através desta formatação de dados conseguimos determinar as aplicações mais prováveis de ser executadas, então não necessitamos de nenhum modelo de aprendizagem uma vez que apenas com a agregação dos dados conseguimos realizar o nosso objectivo. Os modelos de aprendizagem apenas obteriam o óbvio, por exemplo, que no `Place 106` a lista da aplicações era neste caso `PockerSudoku`. Seria um erro tentar obter um modelo de aprendizagem quando através de uma estrutura de representação conveniente, como por exemplo `XML` (Figura 5.3), o problema ficaria solucionado.

```
<Inferences>
    <inference id="001">
        <place>No info</place>
        <list_processes>
            PocketSudoku, cprog, tmail
        </list_processes>
    </inference>
    <inference id="002">
        <place>Place61</place>
        <list_processes>
            PocketSudoku
        </list_processes>
    </inference>
</Inferences>
```

Figura 5.3: Exemplo da representação utilizando o *XML*

5.5 Algoritmos de Associação

Pelas conclusões obtidas na experiência anterior, verificámos que os modelos de Classificação não eram os indicados para o nosso problema, uma vez que a solução resumia-se apenas à agregação dos dados. No entanto, a agregação implica por sua vez um trabalho de tratamento da informação realizado pelo utilizador e não pelo próprio sistema. Foi então necessário encontrar outra abordagem de forma a serem determinadas automaticamente pelo sistema as possíveis associações entre a informação de localização e os processos em execução.

Tendo como objectivo principal determinar a existência de correlações entre as localizações do utilizador e a sua actividade computacional, o que procuramos são exactamente associações entre os vários locais e as aplicações. Neste caso, o problema não é uma questão de classificação de classes, mas sim de obtenção de associações entre os vários atributos.

A próxima experiência passa assim pela utilização dos algoritmos de associação disponíveis no *WEKA* com o objectivo de determinar se estes são capazes de determinar a correlação por nós pretendida, sem que seja necessário qualquer tratamento da informação obtida pela nossa Aplicação de Amostragem.

Uma vez que os três algoritmos de associação do *WEKA* realizam apenas o processo de treino, não podemos nesta experiência utilizar a técnica de *cross-validation* como fizemos anteriormente. É necessário assim dividir os dados obtidos em conjuntos de treino e de teste. Por essa razão recorremos à técnica *leave-one-out*, dividindo o conjunto total em 2/3 para treino e 1/3 para teste. Os próximos testes apresentam as regras de associação obtidas pelos diferentes algoritmos para cada um dos utilizadores.

5.5.1 Estudo com vários utilizadores

Utilizador1

Algoritmo	**Regras**
Apriori	Place=Place1 ➠ Process=tmail
	Place=Place49 ➠ Process=SuDoKu
PredictiveApriori	Place=Place46 ➠ Process=poutlook
Tertius	Place = Place1 ➠ Process = tmail
	Place = Place49 ➠ Process = SuDoKu
	Place = Place109 ➠ Process = SuDoKu
	Place = No info ➠ Process = fexplore
	Place = Place46 ➠ Process = poutlook
	Place = Place31 ➠ Process = poutlook
	Place = Place28 ➠ Process = poutlook
	Place = Place34 ➠ Process = poutlook
	Place = Place37 ➠ Process = poutlook
	Place = Place52 ➠ Process = fexplore
	Place = Place43 ➠ Process = poutlook
	Place = Place25 ➠ Process = poutlook
	Place = Place55 ➠ Process = tmail
	Place = Place43 ➠ Process = tmail
	Place = Place25 ➠ Process = tmail
	Place = Place49 ➠ Process = fexplore
	Place = Place43 ➠ Process = fexplore

Tabela 5.14: Regras de Associação para o Utilizador1

Como se verifica pela tabela 5.14 o algoritmo de associação que melhor responde ao nosso problema é sem dúvida o *Tertius*. Resumindo as regras obtidas por este algoritmo de forma

a obtermos apenas uma ocorrência do `Place` e respectivos `Process` para cada um deles, obtemos a informação apresentada na tabela 5.15.

Place	**Processes**
No info	fexplore
Place1	tmail
Place109	SuDoKu
Place25	poutlook, tmail
Place28	poutlook
Place31	poutlook
Place34	poutlook
Place37	poutlook
Place43	poutlook, fexplore, tmail
Place46	poutlook
Place49	SuDoKu, fexplore
Place52	fexplore
Place55	tmail

Tabela 5.15: Associações resumidas para o Utilizador1

Comparando a tabela 5.15 com a informação agregada (Apêndice D, Tabela D.1) obtida para este utilizador na secção 5.4.3, verificamos que os resultados não são muito diferentes. Com base nesta informação e de forma a determinar a taxa de sucesso das regras de associação, desenvolvemos uma aplicação de teste que, a partir dos conjuntos de dados de teste e das regras de associação, verifica se para cada `Place` os `Process` executados estão presentes nas regras que foram obtidas. A tabela 5.16 apresenta os resultados obtidos com o algoritmo *Tertius*.

Parâmetro	**Valores**
Número de instâncias	308
Número de correctos	250
Taxa de sucesso	81%

Tabela 5.16: Resultados do conjunto de teste

Utilizador2

Para este utilizador os resultados obtidos pelos dois primeiros algoritmos de associação são pouco interessantes como se pode ver pela tabela 5.17.

Algoritmo	**Regras**
Apriori	Place=No info ➠ Process=poutlook
PredictiveApriori	Place=Place352 ➠ Process=TomTom Navigator

Tabela 5.17: Regras de Associação para o Utilizador2

Mas fazendo o resumo das regras obtidas pelos *Tertius*, conseguimos obter resultados mais interessantes (Apêndice E, Tabela E.1). Este resumo tem por base 47 regras de associação obtidas por este algoritmo.

Determinando a taxa de sucesso a partir das regras de associação obtidas, conseguimos ter os seguintes resultados (Tabela 5.18).

Parâmetro	**Valores**
Número de instâncias	445
Número de correctos	318
Taxa de sucesso	71%

Tabela 5.18: Resultados do conjunto de teste

A taxa de sucesso para este utilizador foi ligeiramente inferior, muito possivelmente devido ao facto de existirem menos ocorrências para cada processo nos vários locais, dificultando assim a determinação das regras de associação.

Utilizador3

Resultados obtidos pelos dois primeiros algoritmos (Tabela 5.19).

Algoritmo	**Regras**
Apriori	Nenhuma regra
PredictiveApriori	Place=Place214 ➠ Process=WordNetCE

Tabela 5.19: Regras de Associação para o Utilizador3

O resumo das regras obtidas pelo *Tertius* encontra-se disponível no Apêndice E, Tabela E.2. Este resumo tem por base as 70 regras de associação obtidas por este algoritmo. Determinando a taxa de sucesso a partir das regras de associação obtidas, verificamos os seguintes resultados (Tabela 5.20).

Parâmetro	**Valores**
Número de instâncias	1653
Número de correctos	665
Taxa de sucesso	40%

Tabela 5.20: Resultados do conjunto de teste

Os resultados obtidos para este utilizador não foram os esperados, uma vez que a taxa de sucesso é muito baixa. A explicação para este facto encontra-se na tabela 5.21.

Como se verifica, para o `Place = No info` existem 911 ocorrências desta instância nos dados de teste associadas ao `Process = PocketSudoku`, sendo o conjunto de teste composto por 1653 instâncias. O algoritmo de associação criou várias regras contendo este processo

Place	Process	Ocorrências
No info	PocketSudoku	911
No info	WordNetCE	151
No info	poutlook	31

Tabela 5.21: Ocorrências para os dados de teste

mas para outros locais e não para este em particular, diminuindo assim a taxa de sucesso das regras obtidas.

Utilizador4

Ao contrário do que acontecia com os utilizadores anteriores, cujos dados continham informação de localização e à qual estava associado um processo, este utilizador tem apenas informação sobre a comunicação *WiFi* e processos. É por esta razão um excelente exemplo para verificarmos se existe alguma correlação entre as aplicações e os serviços de comunicação.

Pelos testes aos algoritmos *Apriori* e *PredictiveApriori* obtivemos os seguintes resultados (Tabela 5.22).

Algoritmo	Regras
Apriori	Wifi=F ➠ SSID=0.0
	SSID=0.0 ➠ Wifi=F
	Process=poutlook AND Wifi=F ➠ SSID=0.0
	SSID=0.0 AND Process=poutlook ➠ Wifi=F
	SSID=default ➠ Wifi=T
	Process=cprog AND Wifi=F ➠ SSID=0.0
	SSID=0.0 AND Process=cprog ➠ Wifi=F
PredictiveApriori	SSID=0.0 AND Process=poutlook ➠ Wifi=F

Tabela 5.22: Regras de Associação para o Utilizador4

Pela tabela 5.22 já se consegue verificar determinadas características entre os processos e a tecnologia *WiFi*, mas com o algoritmo *Tertius* essas características são ainda mais evidentes. Vamos começar por ver apenas as regras obtidas por este algoritmo que associam o `SSID` da *WLAN* com o facto de a tecnologia *WiFi* estar presente (Tabela 5.23).

SSID	Wifi
0.0	F
default	T
jnfire	T
guest-e-U	T
SpeedTouch0FD108	T

Tabela 5.23: Associação entre o SSID e a tecnologia WiFi

Os resultados obtidos são exactamente o que seria de esperar. Quando existe comunicação *WiFi* é perfeitamente normal que exista o nome da rede *WLAN* associado. Mas a conclusão mais importante é verificarmos que existem aplicações que apenas estão presentes quando a comunicação *WiFi* está activa (Tabela 5.24).

Wifi	**Processos**
T	fexplore, TELNET, tmail, poutlook, iexplore, InstMsgr, WirelessMgr, WLanMgr, cprog, fexplore
F	cprog, poutlook, repllog, Camera, device

Tabela 5.24: Associação entre o WiFi e as aplicações

Aplicações como o `TELNET`, `iexplore`, `InstMsgr`, entre outras, necessitam de um serviço de comunicação para poderem desempenhar as suas funções, daí que apenas estejam presentes quando os serviços de comunicação estão activos. Desta forma, conseguimos demonstrar que existe de facto uma correlação entre os serviços de comunicação e a actividade computacional no *PDA*. Como forma de quantificar esta correlação e a partir das regras obtidas (Apêndice E, Tabela E.3) executámos com o conjunto de dados de teste a aplicação que determina a taxa de sucesso das regras de associação (Tabela 5.25).

Parâmetro	**Valores**
Número de instâncias	434
Número de correctos	433
Taxa de sucesso	99,76%

Tabela 5.25: Resultados do conjunto de teste

Estes resultados são bons indicadores da correlação entre a tecnologia de comunicação e a utilização aplicacional do *PDA*. É óbvio que a correlação neste caso era de esperar que fosse realmente bastante forte, só tendo disponível o serviço de comunicação é que faz sentido utilizar aplicações como, por exemplo, o `TELNET`.

Utilizador5

Este utilizador continha já informação referente à tecnologia *GSM*, no entanto, pelo facto de tanto o atributo `LAC` como o atributo `CellID` terem sempre o mesmo valor, estes atributos não foram considerados para a geração das regras de associação, uma vez que não eram atributos discriminatórios.

Os resultados obtidos pelos dois primeiros algoritmos (Tabela 5.26).

O resumo das regras obtidas pelo *Tertius* encontra-se disponível no Apêndice E, Tabela E.4. Este resumo tem por base as 8 regras de associação obtidas por este algoritmo. Determinando a taxa de sucesso a partir das regras de associação obtidas, verificamos os seguintes resultados (Tabela 5.27).

Algoritmo	Regras
Apriori	Process=cprog ➠ Place=No info
	Place=Place61 ➠ Process=PocketSudoku
PredictiveApriori	Process=cprog ➠ Place=No info

Tabela 5.26: Regras de Associação para o Utilizador5

Parâmetro	Valores
Número de instâncias	48
Número de correctos	27
Taxa de sucesso	56%

Tabela 5.27: Resultados do conjunto de teste

Como acontecia com o Utilizador3 também neste caso para o `Place = No info` não foi criada a regra que obtinha maior número de ocorrências, a regra que associa este local ao `Process = PocketSudoku`. Por esta razão os resultados foram muito abaixo do que se pretendia.

Utilizador6

Os dados obtidos para este utilizador têm informação sobre localização através das tecnologias *GSM* e *GPS*, e aplicações em execução. É por esta razão um bom exemplo para verificarmos se a informação de *GSM* permite obter algumas conclusões, em particular verificar se a sua recolha traz alguma melhoria ao sistema.

Realizando os testes com os dois primeiros algoritmos, obtivemos os seguintes resultados (Tabela 5.28).

Algoritmo	Regras
Apriori	CellID=33951 ➠ LAC= 18
	CellID=33951 AND Place=No info ➠ LAC=18
	Process=tmail ➠ LAC=18
	Process=tmail ➠ CellID=33951
	Process=tmail ➠ Place=No info
	CellID=33951 AND Process=tmail ➠ LAC=18
	LAC= 18 AND Process=tmail ➠ CellID=33951
	Process=tmail ➠ LAC= 18 AND CellID=33951
	Place=No info AND Process=tmail ➠ LAC=18
	LAC= 18 AND Process=tmail ➠ Place=No info
PredictiveApriori	Process=tmail ➠ LAC=18 AND CellID=33951

Tabela 5.28: Regras de Associação para o Utilizador6

Através do algoritmo *Apriori* em particular, já se consegue verificar associações entre os vários atributos. As mais óbvias relacionam o `LAC` com o `CellID`, relação esta que seria inevitável uma vez que a mesma célula *GSM* tem sempre o mesmo identificador da área

onde esta se encontra posicionada. Mas mais importante é verificar que o atributo `Process` tem relacionamentos com outros atributos, além do habitual relacionamento com o atributo `Place` que já tínhamos visto anteriormente com os outros utilizadores.

Realizando agora o teste com o algoritmo *Tertius*, obtemos um número elevado de regras de associação com diferentes conjuntos de atributos. O primeiro teste que vamos realizar é com base apenas nas regras de associação que relacionam o atributo `Place` com os processos. Pretendemos assim verificar qual a taxa de sucesso apenas com este atributo, incrementando posteriormente os atributos de localização com o objectivo de determinar se existe algum ganho ao nível da taxa de sucesso com o aumento do número de atributos.

Com apenas os dois atributos obtivemos 19 regras de associação disponíveis de forma resumida no Apêndice E, Tabela E.5. Determinando a taxa de sucesso a partir das regras de associação obtidas, verificamos os seguintes resultados (Tabela 5.29).

Parâmetro	**Valores**
Número de instâncias	549
Número de correctos	402
Taxa de sucesso	73%

Tabela 5.29: Resultados do conjunto de teste

Estes resultados são já muito interessantes, no entanto como existem situações em que a informação de *GPS* não está disponível e como tal aparece como `Place = No info`, pretendemos saber se adicionando a informação da célula *GSM*, em particular o seu identificador, conseguimos melhorar a taxa de sucesso que obtivemos até ao momento. Assim nesta fase vamos utilizar as regras de associação que relacionam os dois atributos de entrada `CellID` e `Place` com o atributo de saída `Process`.

Utilizando apenas as 30 primeiras regras de associação de um total de 100 regras, disponíveis de forma resumida no Apêndice E, Tabela E.6, que abrangem a maioria das situações existentes no conjunto de treino, obtivemos a seguinte taxa de sucesso (Tabela 5.30).

Parâmetro	**Valores**
Número de instâncias	549
Número de correctos	438
Taxa de sucesso	80%

Tabela 5.30: Resultados do conjunto de teste

Como se verifica, apenas implementando 1/3 da totalidade das regras, embora sejam as regras mais significativas, obtivemos um ligeiro aumento na taxa de sucesso. O facto mais interessante que podemos verificar na tabela E.6 é a existência de regras cuja informação *GSM* complementa a informação de *GPS*, o que de certa forma era o esperado, mas mais importante é verificar que existem regras que apenas contêm informação de *GSM*. Por esta

razão, conseguimos verificar que a informação de localização baseada em *GSM* é também importante para a correlação entre os locais e a actividade computacional, embora desta só seja verdadeiramente significativa a informação do `CellID`, uma vez que a informação obtida pelo `LAC` não é discriminatória.

Utilizador7

Para este utilizador, foram obtidos dados referentes às tecnologias *GSM* e *WiFi*, e aplicações em execução. Pelos testes realizados com os dados do Utilizador4 verificamos que existe uma forte correlação entre as aplicações e o estado da comunicação *WiFi*. De forma a dar maior suporte a essa afirmação vamos inicialmente realizar o teste apenas com esses dois atributos, `Wifi` e `Process`. Foi também recolhida informação relativa ao ponto de acesso da rede *WLAN* a que o utilizador se ligava, no entanto como o endereço do ponto de acesso era sempre o mesmo não permite obter grandes conclusões. Apenas se verifica que quando existe o endereço do ponto de acesso também está activa a comunicação *WiFi*. Por esta razão este atributo foi retirado uma vez que não acrescentava nada ao modelo.

Realizando os testes com os dois primeiros algoritmos com apenas dois atributos, obtivemos os seguintes resultados (Tabela 5.31).

Algoritmo	**Regras**
Apriori	Process=device ➠ Wifi=F
PredictiveApriori	Process=device ➠ Wifi=F

Tabela 5.31: Regras de Associação para o Utilizador7

Realizando agora o teste com o algoritmo *Tertius* obtivemos 6 regras de associação (Tabela 5.32).

Wifi	**Processes**
F	device
F	iexplore
F	WirelessMgr
T	cprog
T	TELNET
T	fexplore

Tabela 5.32: Associações com dois atributos para o Utilizador7

Na tabela anterior podemos verificar uma situação interessante, a existência do processo `iexplore` sem que exista o serviço de comunicação. Tendo em conta os dados adquiridos a regra está correcta e o mais provável é o utilizador ter estado durante esse tempo a consultar informação local, sem ter a necessidade de serviços de comunicação.

Determinando a taxa de sucesso a partir das regras de associação anteriores, verificamos os seguintes resultados (Tabela 5.33).

Parâmetro	Valores
Número de instâncias	217
Número de correctos	179
Taxa de sucesso	82%

Tabela 5.33: Correlação entre o Wifi e os Processos

Mais uma vez se constata a forte correlação entre a tecnologia de comunicação e os processos em execução.

Além do teste anterior é também importante saber se apenas com a informação celular para a localização, conseguimos estabelecer associações com os diferentes processos. O próximo teste utiliza assim como atributo de entrada, o `CellID` e como atributo de saída, o `Process`.

Realizando o teste apenas com os dois atributos referidos anteriormente, o algoritmo *Tertius* obteve 23 regras de associação, disponíveis de forma resumida no Apêndice E, Tabela E.7.

Determinando a taxa de sucesso a partir das regras de associação anteriores, verificamos os seguintes resultados (Tabela 5.34).

Parâmetro	Valores
Número de instâncias	217
Número de correctos	212
Taxa de sucesso	98%

Tabela 5.34: Correlação entre o CellID e os Processos

Os resultados anteriores foram para nós uma grande surpresa. Nunca esperámos que apenas com a informação de *GSM* como atributo de localização fosse possível obter resultados tão elevados.

Utilizador8

Os resultados obtidos anteriormente para o Utilizador7, pelo facto de serem tão bons com uma tecnologia cuja finalidade não é de todo a localização, incentivou-nos a fazer idêntico teste para o Utilizador8. Este utilizador tem informação tanto de *GPS* como de *GSM*.

Realizando o teste com o algoritmo *Tertius* com apenas dois atributos obtivemos 14 regras de associação que estão disponíveis de forma resumida no Apêndice E, tabela E.8).

Determinando a taxa de sucesso a partir destas regras de associação, verificamos os seguintes resultados (Tabela 5.35).

Mais uma vez se verifica uma forte correlação entre a localização, utilizando a informação de célula *GSM*, e as aplicações em execução.

Parâmetro	Valores
Número de instâncias	757
Número de correctos	686
Taxa de sucesso	91%

Tabela 5.35: Correlação entre o CellID e os Processos

Outro aspecto que queríamos verificar era se com informação apenas de *GPS* conseguíamos obter valores tão interessantes como os que foram obtidos anteriormente. Recorrendo novamente ao algoritmo *Tertius* realizamos o teste apenas com o atributo de entrada `Place` e o atributo de saída `Process`.

Realizando o teste obtivemos igualmente 14 regras de associação que estão disponíveis de forma resumida no Apêndice E, tabela E.9).

Determinando a taxa de sucesso a partir destas regras de associação, verificamos os seguintes resultados (Tabela 5.36).

Parâmetro	Valores
Número de instâncias	757
Número de correctos	615
Taxa de sucesso	81%

Tabela 5.36: Correlação entre o Place e os Processos

Os resultados foram ligeiramente inferiores, no entanto continuam a ser interessantes.

5.5.2 Análise aos resultados

Os resultados obtidos através dos algoritmos de Classificação tendo por base os dados adquiridos directamente da nossa Aplicação de Amostragem mostraram que esta não era a melhor abordagem. As regras obtidas têm por base o processo com maior ocorrências de acordo com cada um dos locais. Apenas conseguíamos relacionar um local com uma aplicação, ficando assim longe do nosso objectivo que era determinar as 5 aplicações mais comuns para cada um dos locais. Realizando a agregação dos dados verificámos que dessa forma conseguíamos atingir esse objectivo. No entanto, a agregação teria que ser feita pelo utilizador e não pelos algoritmos de aprendizagem, verificando-se inclusivamente que depois de feita a agregação não era necessário qualquer tipo de algoritmo de aprendizagem porque as regras que obteriam eram facilmente detectáveis pela simples leitura dos dados agregados. Por esta razão, experimentámos a abordagem utilizando os algoritmos de associação.

Através dos algoritmos de associação, conseguimos eliminar o processo intermédio de agregação dos dados, uma vez que a mesma é feita através das regras de associação obtidas por estes algoritmos. Mas a experimentação demonstrou que os resultados não eram tão bons como seria de esperar, inclusivamente que só o algoritmo *Tertius* era o mais indicado. Se, para

os utilizadores que dispunham de informação de localização baseada em *GPS*, conseguimos bons resultados para o Utilizador1, Utilizador2 e Utilizador8, o mesmo não se verificou para o Utilizador3 e Utilizador 5. Para estes utilizadores, como o algoritmo não conseguiu determinar a regra associada à instância que tinha maior frequência no conjunto de treino, a taxa de sucesso foi fortemente penalizada.

Embora os resultados da verificação da correlação entre localização e aplicações não tenham sido os melhores, no caso da correlação entre a tecnologia de comunicação e as aplicações, isso não aconteceu. Tanto para o Utilizador4 como para o Utilizador 7 verificámos que a correlação não só existe como é bastante forte.

Os testes com os dados do Utilizador6 permitiram observar que ao adicionarmos a informação da célula *GSM* como atributo de localização, consegue-se aumentar ligeiramente a taxa de sucesso comparando-a com a situação em que temos apenas como atributo de localização a informação de *GPS*.

Mais interessante ainda foi verificar, com os dados do Utilizador7 e Utilizador8, que bastava a informação da célula *GSM* para se obter uma excelente correlação entre a localização e os processos em execução. A esta constatação não está isento o facto de, ao contrário do que acontece com o *GPS*, que em determinadas alturas não é possível determinar a localização, esta tecnologia está sempre presente, sendo possível a todo o instante saber a informação da célula actual. Outro ponto a favor do *GSM* é o facto da informação da célula permitir a localização numa área, sendo por isso mais tolerante que o *GPS* pois neste último, a localização é feita de forma mais restrita, onde, à mínima variação de coordenadas se pode originar a associação a outro local.

5.6 Algoritmos de Clustering

Os resultados obtidos pelo algoritmo de associação *Tertius*, em particular para o Utilizador3 e Utilizador5, ficaram longe do resultado esperado em virtude de não ter sido capaz de determinar as regras que abrangiam a maioria dos dados de treino. Por esta razão era necessário encontrar formas alternativas de determinar as relações entre a localização e a actividade aplicacional no *PDA*. Até ao momento, tendo já resultados dos algoritmos de Classificação e de Associação, faltava fazer a experimentação para os algoritmos de *Clustering*.

Atendendo a que o objectivo a atingir é determinar uma correlação entre a localização e as aplicações, e como para diferentes locais temos associados um conjunto de aplicações, determinar grupos compostos por local e aplicações dentro do conjunto de dados é a situação indicada para a utilização de algoritmos de *Clustering*.

Assim, os primeiros testes que realizámos são precisamente sobre os dois utilizadores com piores resultados nos algoritmos de associação, com o objectivo de verificarmos se com os algoritmos de *clustering* a taxa de sucesso aumenta para valores satisfatórios.

Dos vários algoritmos de *clustering* existentes no *WEKA* apenas realizámos testes com o *FarthestFirst* e *SimpleKMeans*. No caso do *Cobweb*, embora se permita a criação de *clusters* hierárquicos, a informação obtida nos vários nós e folhas da árvore não é de fácil leitura, sendo por isso difícil dizer quais os processos e locais que estão presentes em cada um desses elementos. O *EM* embora atribua a cada instância uma probabilidade de a mesma pertencer a cada um dos *clusters* obtidos, também não apresenta de forma directa a composição de cada *cluster*. O *MakeDensityBasedClusterer* não é propriamente um algoritmo de *clustering* uma vez que utiliza um dos algoritmos existentes para determinar a densidade e distribuição normal para cada um dos *clusters*.

Os dados utilizados para a experimentação destes dois algoritmos são os mesmos que foram utilizados para os algoritmos de associação, os mesmos conjuntos de treino e de teste. Garantimos assim que a *performance* dos modelos obtidos apenas depende dos algoritmos e não dos dados que foram utilizados, permitindo realizar comparações da taxa de sucesso entre algoritmos de associação e algoritmos de *clustering*.

Uma questão típica que se coloca na criação de modelos de aprendizagem baseados em algoritmos de *clustering* é o número de *clusters* que deve ser definido. Como não existe nenhum valor predefinido, uma vez que o número de *clusters* depende em muito do tipo de problema que está a ser estudado, a abordagem que seguimos foi de inicialmente atribuirmos um limite de 10 *clusters* e progressivamente este valor ser aumentado em intervalos de 10 até que seja atingido um valor de taxa de sucesso satisfatório.

5.6.1 Estudo com vários utilizadores

Utilizador3

Este utilizador obteve para os algoritmos de associação uma taxa de sucesso de apenas 40%. Como é um valor demasiadamente baixo, o teste seguinte pretende verificar se com os algoritmos de *clustering* é possível aumentar a taxa de sucesso.

Realizando o treino com os dois algoritmos seleccionados obtivemos os seguintes resultados (Tabela 5.37).

Algoritmo	**Clusters**	**Sucesso(Treino)**
FarthestFirst	10	99%
SimpleKMeans	10	100%

Tabela 5.37: Clusters para o Utilizador3

Verifica-se que apenas com o limite de 10 *clusters* a taxa de sucesso do conjunto de treino é muito satisfatória, não sendo por isso necessário aumentar o número de *clusters*.

Determinando a taxa de sucesso para o conjunto de dados de teste a partir dos *clusters* obtidos pelos dois algoritmos, verificamos os seguintes resultados (Tabela 5.38).

Algoritmo	Instâncias	Correctos	Taxa de sucesso
FarthestFirst	1653	990	60%
SimpleKMeans	1653	1465	89%

Tabela 5.38: Resultados do conjunto de teste

Como se verifica pela tabela anterior o algoritmo *SimpleKMeans* obteve óptimos resultados para este utilizador. Tendo em conta que este utilizador tem 47 locais e 21 aplicações distintos, com apenas 10 *clusters* os resultados são muito interessantes. O resumo dos *clusters* obtidos pelo *SimpleKMeans* no formato idêntico ao utilizado nos algoritmos de associações, encontra-se disponível no Apêndice F, Tabela F.1.

Utilizador5

A experiência anterior demonstra que os algoritmos de *clustering* têm melhor *performance* que os algoritmos de associação em particular para esse utilizador. Era por isso importante verificar se para o Utilizador5 o mesmo facto se verifica.

Realizando o treino com os dois algoritmos de *clustering* obtivemos os seguintes resultados (Tabela 5.39).

Algoritmo	Clusters	Sucesso(Treino)
FarthestFirst	10	100%
SimpleKMeans	10	100%

Tabela 5.39: Clusters para o Utilizador5

Não foi necessário aumentar o número de *clusters* uma vez que para ambos os algoritmos se conseguiu uma percentagem de certeza de 100%.

Determinando a taxa de sucesso para o conjunto de dados de teste a partir dos *clusters* obtidos pelos dois algoritmos, verificamos os seguintes resultados (Tabela 5.40).

Algoritmo	Instâncias	Correctos	Taxa de sucesso
FarthestFirst	48	48	100%
SimpleKMeans	48	48	100%

Tabela 5.40: Resultados do conjunto de teste

Para este utilizador verificou-se que ambos os algoritmos obtiveram excelentes resultados e inclusivamente obtiveram os mesmos *clusters*. O resumo dos *clusters* obtidos encontra-se disponível no Apêndice F, Tabela F.2.

Utilizador1

Para os dois utilizadores que obtiveram piores resultados com os algoritmos de associações, já verificámos que os algoritmos de *clustering* têm resultados bastante melhores, sendo por isso a escolha mais adequada encontrada até ao momento. Agora é importante verificar se para os restantes utilizadores a opção pelos algoritmos de *clustering* se mantém como a mais correcta.

Realizando o treino para o Utilizador1 com os dois algoritmos de *clustering* obtivemos os seguintes resultados (Tabela 5.41).

Algoritmo	**Clusters**	**Sucesso(Treino)**
FarthestFirst	10	88%
FarthestFirst	20	97%
FarthestFirst	30	97%
SimpleKMeans	10	98%
SimpleKMeans	20	97%
SimpleKMeans	30	97%

Tabela 5.41: Clusters para o Utilizador1

Uma vez que com 30 *clusters* os resultados do treino era iguais aos obtidos com apenas 20 *clusters*, a taxa de sucesso foi determinada apenas com os 20 *clusters*.

Determinando a taxa de sucesso para o conjunto de dados de teste a partir dos *clusters* obtidos pelos dois algoritmos, verificamos os seguintes resultados (Tabela 5.42).

Algoritmo	**Instâncias**	**Correctos**	**Taxa de sucesso**
FarthestFirst	308	267	87%
SimpleKMeans	308	305	99%

Tabela 5.42: Resultados do conjunto de teste

Como aconteceu com o Utilizador3, mais uma vez o algoritmo *SimpleKMeans* obteve melhores resultados que o *FarthestFirst* e comparando estes resultados com os que foram obtidos pelo algoritmo de associações verifica-se que a taxa de sucesso aumento significativamente. O resumo dos *clusters* do algoritmo *SimpleKMeans* encontra-se disponível no Apêndice F, Tabela F.3.

Utilizador2

Este utilizador é o último exemplo onde a informação que dispomos apenas se refere à localização obtida por *GPS* e o processo actualmente em execução.

Realizando o treino com os dois algoritmos de *clustering* obtivemos os seguintes resultados (Tabela 5.43).

Algoritmo	Clusters	Sucesso(Treino)
FarthestFirst	10	100%
SimpleKMeans	10	100%

Tabela 5.43: Clusters para o Utilizador2

Mais uma vez se verifica que com apenas 10 *clusters* consegue-se treinar o modelo de aprendizagem com a totalidade de sucesso.

Determinando a taxa de sucesso para o conjunto de dados de teste a partir dos *clusters* obtidos pelos dois algoritmos, verificamos os seguintes resultados (Tabela 5.44).

Algoritmo	Instâncias	Correctos	Taxa de sucesso
FarthestFirst	445	135	30%
SimpleKMeans	445	328	74%

Tabela 5.44: Resultados do conjunto de teste

Os resultados obtidos pelo algoritmo *SimpleKMeans* foram ligeiramente superiores ao que tinham sido obtidos pelo algoritmo de associação. O resultado obtido pelo *FarthestFirst* foi inesperado e começou a dar a entender que o algoritmo de *clustering* mais adequado para a nossa situação é mesmo o *SimpleKMeans*. O resumo dos clusters do algoritmo *SimpleKMeans* encontra-se disponível no Apêndice F, Tabela F.4.

Utilizador4

Os testes realizados até ao momento com os vários utilizadores permitem verificar que existe uma correlação entre o atributo `Place` e o atributo `Process` e que a mesma pode ser obtida através dos algoritmos de clustering com elevado sucesso.

O próximo teste com os dados do Utilizador4 tem como objectivo verificar se estes algoritmos são capazes de determinar uma correlação entre a comunicação utilizando a tecnologia *WiFI* e os diferentes processos.

Realizando o treino para o Utilizador4 com os dois algoritmos de *clustering* obtivemos os seguintes resultados (Tabela 5.45).

Algoritmo	Clusters	Sucesso(Treino)
FarthestFirst	10	99%
SimpleKMeans	10	100%

Tabela 5.45: Clusters para o Utilizador4

Os resultados do treino demonstram que é possível determinar com sucesso clusters entre o atributo `Wifi` e o atributo `Process`.

Determinando a taxa de sucesso para o conjunto de dados de teste a partir dos *clusters* obtidos pelos dois algoritmos, verificamos os seguintes resultados (Tabela 5.46).

Algoritmo	**Instâncias**	**Correctos**	**Taxa de sucesso**
FarthestFirst	434	356	82%
SimpleKMeans	434	431	99%

Tabela 5.46: Resultados do conjunto de teste

Os resultados obtidos são idênticos aos que se registaram com o algoritmo de associação. Verifica-se mais uma vez uma correlação entre a tecnologia de comunicação e o tipo de actividade aplicacional no *PDA*. O resumo dos *clusters* do algoritmo *SimpleKMeans* encontra-se disponível no Apêndice F, Tabela F.5.

Utilizador6

Na experiência realizada para este utilizador com o algoritmo de associação, obteve-se uma taxa de sucesso de 80% utilizando como atributos de entrada o `CellID` e o `Place` e como atributo de saída o `Process`.

Como até ao momento para os algoritmos de *clustering* ainda não foi utilizado o atributo `CellID`, o próximo teste tem como objectivo verificar se este atributo é útil para o modelo de aprendizagem ou se por outro lado pode ser desprezado.

Realizando o treino para o Utilizador6 com os dois algoritmos de *clustering* obtivemos os seguintes resultados (Tabela 5.47).

Algoritmo	**Clusters**	**Sucesso(Treino)**
FarthestFirst	10	99%
SimpleKMeans	10	99%
FarthestFirst	20	98%
SimpleKMeans	20	100%

Tabela 5.47: Clusters para o Utilizador6

Como os resultados do treino são diferenciados para os dois algoritmos, no caso do *FarthestFirst* vão ser utilizados apenas 10 *clusters*, uma vez que a taxa de sucesso no treino foi superior, enquanto que para o algoritmo *SimpleKMeans* vão ser utilizados 20 *clusters*.

Determinando a taxa de sucesso a partir dos *clusters* obtidos pelos dois algoritmos, verificamos os seguintes resultados (Tabela 5.48).

Algoritmo	**Instâncias**	**Correctos**	**Taxa de sucesso**
FarthestFirst	549	190	35%
SimpleKMeans	549	449	82%

Tabela 5.48: Resultados do conjunto de teste

Mais uma vez se verifica que o algoritmo *FarthestFirst* tem piores resultados e que os resultados obtidos pelo *SimpleKMeans* são idênticos aos que se registaram com o algoritmo de associação. O resumo dos *clusters* do algoritmo *SimpleKMeans* encontra-se disponível no Apêndice F, Tabela F.6. Nesta tabela é possível verificar que, para a mesma situação do atributo `Place = No info`, existem diferentes células *GSM* associadas, o que significa que este atributo é discriminatório e como tal deverá ser mantido pelo sistema.

Utilizador7

No teste com o algoritmo de associação, verificámos que, para este utilizador, existia uma forte correlação entre o atributo `CellID` e o atributo `Process` permitindo assim associar uma localização apenas com informação da célula *GSM* às aplicações em execução. O próximo teste pretende verificar se essa correlação também se verifica utilizando algoritmos de *clustering*.

Realizando o treino com os dois algoritmos de *clustering* e apenas com os dois atributos referidos anteriormente, obtivemos os seguintes resultados (Tabela 5.49).

Algoritmo	**Clusters**	**Sucesso(Treino)**
FarthestFirst	10	100%
SimpleKMeans	10	100%

Tabela 5.49: Clusters para o Utilizador7

Determinando a taxa de sucesso a partir dos *clusters* obtidos pelos dois algoritmos, verificamos os seguintes resultados (Tabela 5.50).

Algoritmo	**Instâncias**	**Correctos**	**Taxa de sucesso**
FarthestFirst	217	94	43%
SimpleKMeans	217	185	85%

Tabela 5.50: Resultados do conjunto de teste

Pela primeira vez se verifica que os resultados do algoritmo de associação são superiores ao algoritmo *SimpleKMeans*, embora a taxa de sucesso deste último continue a ser um bom resultado. De salientar o facto de tanto o algoritmo de associação como o *SimpleKMeans* demonstrarem que apenas com informação celular é possível ter uma correlação entre a localização e os processos.

O resumo dos *clusters* do algoritmo *SimpleKMeans* encontra-se disponível no Apêndice F, Tabela F.7.

Uma vez que este utilizador também tem informação sobre a tecnologia *WiFi*, o próximo teste tem como objectivo verificar se existe uma correlação entre a localização, a presença desta tecnologia e actividade computacional.

Realizando o treino com os três atributos, obtivemos os seguintes resultados (Tabela 5.51).

Algoritmo	**Clusters**	**Sucesso(Treino)**
FarthestFirst	10	100%
SimpleKMeans	10	100%

Tabela 5.51: Clusters para o Utilizador7

Determinando a taxa de sucesso a partir dos *clusters* obtidos pelos dois algoritmos, verificamos os seguintes resultados (Tabela 5.52).

Algoritmo	**Instâncias**	**Correctos**	**Taxa de sucesso**
FarthestFirst	217	25	12%
SimpleKMeans	217	168	77%

Tabela 5.52: Resultados da correlação

Analisando o resumo dos *clusters* do algoritmo *SimpleKMeans* (Tabela 5.53), verifica-se que existem aplicações em execução que necessitam de serviços de comunicação, por exemplo `iexplore` e `TELNET`, e que em algumas situações o serviço de comunicação está inactivo. O mesmo facto já se tinha verificado também com o algoritmo de associação. No entanto, é possível verificar que, quando o serviço de comunicação está activo, a aplicação mais comum é o `iexplore`, o que sugere que esta aplicação tem uma relação directa com o serviço de comunicação. Por outro lado também é verdade que esta aplicação não é frequente em todos os locais obtidos pelo `CellID`, sugerindo então que o local onde o utilizador se encontra condiciona a execução desta aplicação.

Estas duas constatações permitem assim confirmar a nossa hipótese, pelo menos para o perfil que este utilizador representa. Existe uma correlação entre a localização do utilizador, a presença de serviços de comunicação e a actividade computacional.

CellID	**Wifi**	**Processes**
6591	F	iexplore
7081	T	iexplore
7081	F	TELNET
10181	F	iexplore
17813	F	device
27982	F	device
32583	F	iexplore
33952	F	device
41071	F	iexplore
41071	T	iexplore

Tabela 5.53: Resumo dos clusters para o Utilizador7

Utilizador8

Nas experiências realizadas com os dados deste utilizador para o algoritmo de associação *Tertius*, verificámos correlações fortes entre a localização e a actividade computacional, quer utilizando o `CellID`, quer utilizando o `Place` como atributo de localização.

Com os algoritmos de *clustering* pretendemos fazer um teste ligeiramente diferente, idêntico ao que foi realizado para o Utilizador6: utilizar em simultâneo os dois atributos de localização (`CellID` e `Place`) de forma a mais uma vez verificar se a informação da célula *GSM* é discriminatória nos vários *clusters* obtidos, justificando assim a sua utilização.

Realizando o treino para o Utilizador8 com os dois algoritmos de *clustering* obtivemos os seguintes resultados (Tabela 5.54).

Algoritmo	**Clusters**	**Sucesso(Treino)**
FarthestFirst	10	100%
SimpleKMeans	10	100%

Tabela 5.54: Clusters para o Utilizador8

Determinando a taxa de sucesso a partir dos *clusters* obtidos pelos dois algoritmos, verificamos os seguintes resultados (Tabela 5.55).

Algoritmo	**Instâncias**	**Correctos**	**Taxa de sucesso**
FarthestFirst	757	239	32%
SimpleKMeans	757	422	56%

Tabela 5.55: Resultados do conjunto de teste

Os resultados obtidos não foram nada satisfatórios e como tal realizamos novamente o teste aumentando o número de *clusters*, uma vez que verificámos que existiam muitos `Place` que não eram abrangidos pelos *clusters*. Assim, com o limite máximo de 20 *clusters* obtivemos os seguintes resultados com o conjunto de teste (Tabela 5.56).

Algoritmo	**Instâncias**	**Correctos**	**Taxa de sucesso**
FarthestFirst	757	270	36%
SimpleKMeans	757	606	80%

Tabela 5.56: Resultados do conjunto de teste

Como se verifica, o facto de se obterem bons resultados durante o treino do modelo de aprendizagem não significa que, para o conjunto de teste, os resultados sejam idênticos. O aumento do número de *clusters* resultou numa melhoria bastante significativa dos resultados com o conjunto de teste.

O resumo dos *clusters* do algoritmo *SimpleKMeans* encontra-se disponível no Apêndice F, Tabela F.8.

Mais uma vez se verifica que a utilização da informação da célula *GSM* permite a distinção entre locais que apenas com a informação de *GPS* seria impossível. Vejamos a tabela 5.57 que demonstra esse facto. Para as situações apresentadas, se apenas tivéssemos informação de *GPS*, o algoritmo considerava como pertencentes ao mesmo *cluster*. Como tal o atributo `CellID` é discriminatório e deverá por isso ser mantido no sistema.

CellID	Place
5591	No info
5592	No info
6191	No info
21993	No info

Tabela 5.57: Locais distintos com ausência de informação *GPS*

5.6.2 Análise aos resultados

As experiências com os algoritmos de associação permitiram verificar que os resultados destes eram superiores aos obtidos pelos algoritmos de classificação e que a única forma de os algoritmos de classificação poderem vir a ser utilizados seria no caso de se realizar a agregação dos dados.

Da mesma forma que os com os algoritmos de associação, também os algoritmos de *clustering* não necessitam de agregação dos dados, essa agregação é de certa forma realizada pela criação dos diferentes *clusters*, não sendo por isso necessária a intervenção do utilizador.

Ao contrário do que ocorreu com os algoritmos de associação, os algoritmos de *clustering* obtiveram bons resultados para todos os utilizadores (Tabela 5.58).

Utilizador	Taxa de Sucesso
Utilizador1	99%
Utilizador2	74%
Utilizador3	89%
Utilizador4	99%
Utilizador5	100%
Utilizador6	82%
Utilizador7	77%
Utilizador8	80%

Tabela 5.58: Taxa de sucesso do conjunto de testes para os diferentes utilizadores

Definindo a localização apenas pelo atributo `Place`, os resultados para os utilizadores Utilizador1, Utilizador2, Utilizador3 e Utilizador5 utilizando o algoritmo de *clustering SimpleKMeans* mostraram que existe uma correlação entre a localização e a actividade, e que a mesma pode ser determinada com sucesso utilizando este algoritmo. Mas esta correlação também existe quando substituímos o atributo de localização pelo `CellID`, como se verificou com o Utilizador7.

Os testes com os utilizadores Utilizador6 e Utilizador8 também mostraram que a utilização dos dois atributos `Place` e `CellID` como elementos de localização, além de permitirem resultados interessantes ao nível da correlação entre localização e aplicações, permitem ter uma diferenciação entre locais mesmo em situações onde a informação de *GPS* não existe.

A correlação entre os recursos de comunicação e a actividade também ficou demonstrada através da experiência com o Utilizador4.

Por último, o teste com o Utilizador7 demonstrou que, para este utilizador, a correlação entre a localização, recursos de comunicação e aplicações existe e pode ser determinada com sucesso através de algoritmos de *clustering*.

Temos assim excelentes indicadores relativamente à hipótese que pretendíamos demonstrar.

Os resultados das várias experiências com os algoritmos de aprendizagem demonstraram que os algoritmos de *clustering* são os mais adequados para o tipo de problema que estamos a estudar, em particular o algoritmo *SimpleKMeans*. É por esta razão o algoritmo de eleição para problemas idênticos ao nosso.

O número reduzido de dados de diferentes utilizadores que serviram de amostra para a nossa experimentação não nos permite afirmar que o algoritmo *SimpleKMeans* terá sempre excelentes resultados, no entanto, devido à diversidade de tecnologias e tipos de utilização dos vários utilizadores, verificamos que, independentemente destas, o algoritmo obteve sempre resultados superiores relativamente aos restantes algoritmos.

Outro facto relacionado com as amostras é a localização utilizando a tecnologia *WiFi*. Como nos nossos exemplos apenas temos situações em que a comunicação é feita em exclusivo com um ponto de acesso, não nos foi possível verificar que esta tecnologia pode ou não ser utilizada como um atributo de localização. No entanto, para esta tecnologia, verificámos que a presença do seu serviço de comunicação é um elemento interessante para caracterizar o contexto, uma vez que o comportamento aplicacional varia de acordo com o estado desta.

Além deste atributo, verificámos que, para caracterizar a localização do utilizador, os atributos `Place` e `CellID` são de facto muito interessantes. O primeiro associado à informação de *GPS* e o segundo associado à tecnologia *GSM*. Ao longo da experimentação, verificámos que, se pretendermos uma localização mais rigorosa, embora também mais sensível, o *GPS* é uma boa opção. Se por outro lado, pretendermos uma localização mais abrangente, mas sempre disponível, a opção pelo *GSM* prevalece. No caso de termos disponíveis ambas as tecnologias, então a sua utilização simultânea deverá ser considerada, pois verificámos que os resultados melhoram significativamente.

Como o contexto é mais do que apenas localização, verificámos que a tecnologia *WiFi* pode ser utilizada para outra dimensão, a Identidade. Esta dimensão é responsável por identificar tanto as pessoas que se encontram junto do utilizador, como os recursos disponíveis. Como

conseguimos obter quer a presença da rede, quer a sua própria identificação, é de facto um elemento que pode ser interessante para caracterizar o ambiente envolvente do utilizador.

Capítulo 6

Conclusões

6.1 Contribuições

No presente trabalho apresentámos a experimentação realizada sobre os algoritmos de aprendizagem, com o objectivo de estudar quais os mais adequados a situações de proactividade, em particular, para dispositivos como os *PDA*s.

Analisemos então as principais contribuições obtidas, fruto da experimentação e do desenvolvimento da Aplicação de Amostragem, utilizada para a aquisição de informação de contexto.

6.1.1 Proactividade

A Proactividade, apresentada no capítulo 2, consiste em determinar as próximas acções do utilizador com o objectivo de minimizar o consumo da sua atenção. De uma forma simplista, podemos dizer que a proactividade é como uma situação de "causa - efeito", ou um agente reactivo, onde a causa é caracterizada pelo contexto do utilizador e o efeito são as acções executadas por este. A Aplicação de Amostragem desenvolvida para este trabalho, apresentada no capítulo 4, é um exemplo de um possível sistema proactivo. O contexto é neste caso definido pela localização do utilizador e pela disponibilidade de uma tecnologia de comunicação, e as acções são caracterizadas pela actividade computacional no *PDA*.

O contexto em computação ubíqua tem por isso um objectivo muito específico, responder a uma simples questão: `Porquê?`. Mas embora a questão seja simples, a sua resposta é pelo contrário bastante complexa. Vejamos como exemplo a simples acção de atender uma chamada num telemóvel ou *PDA*. Esta acção é caracterizada apenas por duas opções: Sim ou Não. Se a resposta ao `Porquê?` for dada pelo contexto apenas definido pela localização, temos a situação apresentada na tabela 6.1.

Localização	**Atender**
Casa	Sim
Trabalho	Não
Restaurante	Não
Cinema	Não
Centro Comercial	Sim

Tabela 6.1: Exemplo da simples acção de atender uma chamada

O exemplo anterior pode facilmente ser representado por um simples modelo de aprendizagem baseado em classificação. Sendo a `Localização` o atributo de entrada e `Atender` o atributo de saída, facilmente se percebe que o modelo de classificação não teria qualquer dificuldade em obter uma elevada taxa de sucesso.

O exemplo anterior é, no entanto, pouco realista. É um facto que de certa forma a localização afecta a decisão que se toma ao atender uma chamada, mas existem outros factores também importantes nessa decisão. Podemos ter situações em que necessitamos de atender chamadas provenientes de determinados indivíduos e então as situações possíveis ganham nova forma (Tabela 6.2).

Localização	**Chamador**	**Atender**
Casa	-	Sim
Trabalho	Chefe	Sim
Trabalho	Esposa	Sim
Trabalho	Outros	Não
Restaurante	-	Não
Cinema	-	Não
Centro Comercial	-	Sim

Tabela 6.2: Evolução do contexto

Podemos ir adicionando ao contexto elementos que podem facilmente ser obtidos por sensores, contexto este a que demos o nome de **contexto sensorial**. No entanto a mente humana é complexa e todas as decisões que tomamos têm por base outros aspectos que dificilmente se conseguem obter por sensores. Por exemplo, o estado psicológico do indivíduo, se está calmo, se está sob stress, se está bem humorado ou se pelo contrário está triste, são atributos que pesam muito nas acções do dia-a-dia e que de momento não existe forma de as obter através de sensores. A este tipo de contexto demos o nome de **contexto não sensorial**. Pelo facto de não existirem sensores, o contexto fica assim mais pobre e leva a que tenhamos para as mesmas situações, respostas diferentes. Vejamos o exemplo apresentado na tabela

6.3 que ilustra esta situação, onde para o atributo `Chamador` com o valor `Amigo X` temos respostas diferentes.

Localização	**Chamador**	**Atender**
Trabalho	Chefe	Sim
Trabalho	Esposa	Sim
Trabalho	Amigo X	Não
Trabalho	Amigo X	Sim

Tabela 6.3: Evolução do contexto

Neste exemplo é possível verificar que para a mesma situação de contexto, a resposta é diferente, uma vez que não foi possível representar uma característica do contexto não sensorial, por exemplo, o facto do utilizador estar muito ocupado com o trabalho e como tal não poder dispensar a sua atenção para atender a chamada.

O facto de existirem características de contexto que dificilmente possam ser representadas, implica que por mais sensores que tenhamos disponíveis, o contexto ficará sempre incompleto. É com esta realidade que os sistemas proactivos têm que lidar.

Recorrendo à metáfora do relacionamento de Entidades utilizado nos sistemas de informação, podemos afirmar que os modelos de classificação têm bons resultados se a relação for do tipo "n para 1", isto é, se os vários conjuntos de instâncias semelhantes estiverem associados a uma única classe, com é o caso do exemplo da tabela 6.2. No entanto, como o mais comum é ter situações de contexto incompleto, o problema de aprendizagem é mais próximo do exemplo da tabela 6.3, onde para a mesma situação de contexto temos diferentes respostas e respostas idênticas para diferentes situações de contexto. O caso mais geral será então ter relacionamentos do tipo "m para n", como aconteceu neste trabalho com o contexto dos vários utilizadores. Pela experimentação realizada neste trabalho, verifica-se que, para este tipo de problemas os algoritmos de aprendizagem baseados em classificação não são os mais indicados.

Os algoritmos de associação, pelo facto de criarem uma quantidade alargada de regras de associação e por isso ser necessário conhecer bem o problema de forma a determinar quais as regras que são efectivamente significativas, não são por isso os mais indicados para problemas de proactividade. Como a escolha terá que ser feita pelo utilizador, quebrando assim a autonomia dos algoritmos, implica o consumo da atenção do utilizador. Tendo em conta os resultados obtidos pela experimentação, verificámos também que os seus resultados não são os melhores, sendo por isso mais um ponto negativo deste tipo de algoritmos de aprendizagem.

A experimentação demonstrou que os algoritmos mais indicados para este tipo de problemática são sem dúvida os de *clustering*, em particular o *SimpleKMeans*. Os resultados

obtidos foram bastante interessantes, sendo apenas possível apontar um ponto negativo, a indicação do número de *clusters* que pretendemos obter. Mas este ponto pode ser facilmente corrigido, através de um simples processo iterativo de aumento do número de *clusters* durante a fase de treino do modelo de aprendizagem. Se o algoritmo iniciar com um valor predefinido de 2 *clusters* podemos em cada iteração verificar os resultados obtidos pelo treino e iterativamente aumentar o número de *clusters* até se atingir um valor de taxa de sucesso no treino que seja considerado como o valor óptimo.

No sistema que desenvolvemos para o estudo dos algoritmos de aprendizagem é de esperar que ao fim de algum tempo o utilizador frequente outros locais ou utilize novas aplicações, estando por isso perante um contexto dinâmico. A estratégia que apresentámos anteriormente para o *SimpleKMeans* permite que ao fim de algum tempo possa ser gerado um novo modelo de aprendizagem com resultados diferentes tanto ao nível do número de *clusters*, como ao nível da definição dos próprios *clusters*. Este princípio permite a constante adaptação do modelo de aprendizagem à dinâmica do contexto.

6.1.2 Contexto

A utilização de tecnologias como o *GPS* e o *GSM* para a localização do utilizador, não é novidade. Existem já vários trabalhos dentro da área que recorrem com sucesso a elas. Por esta razão os resultados que obtivemos com o *GPS* foram os que se esperavam. No entanto, relativamente à tecnologia *GSM*, a nossa abordagem foi diferente da tradicional.

Em projectos, como por exemplo o *Place Lab*, a localização é feita recorrendo à aquisição da identificação de todas as células *GSM* existentes na área e respectivas potências de sinal. Esta abordagem requer que as diferentes situações adquiridas sejam posteriormente associadas a um identificador do local através por exemplo, de um algoritmo de classificação. No nosso caso, como pretendíamos evitar esta necessidade, optámos pela utilização apenas do identificador da célula *GSM* a que o dispositivo está associado. A realização da experimentação permitiu verificar que apenas com esta informação já se consegue determinar a localização do utilizador. É verdade que não temos a mesma granularidade que se consegue com a abordagem do *Place Lab* ou mesmo com o *GPS*, mas para situações em que a localização do utilizador possa ser feita com maior tolerância, esta tecnologia pode ser utilizada com sucesso, recorrendo apenas a um serviço disponível em qualquer *PDA* com telemóvel sem qualquer custo adicional, quer seja do serviço ou de equipamento, e sem a utilização de mecanismos de aprendizagem das localizações.

A localização é sem dúvida um atributo importante na caracterização do contexto. No entanto, não é o único e como tal existem outros atributos também significativos para o contexto, como é o caso da identidade. É sobre este atributo que julgamos que a nossa contribuição para o contexto é mais significativa.

A obtenção de informação relativamente à presença da tecnologia *WiFi* e da identificação da rede *WLAN* a que o utilizador se encontra ligado, é uma característica de contexto muito interessante. Como vimos na experimentação realizada neste trabalho, o comportamento do utilizador é diferente consoante a presença ou não desta tecnologia. Por este facto julgamos que a recolha de informação relativa aos serviços de comunicação disponíveis, quer seja *WiFi* ou outros, deverá ser considerada nas situações em que à partida se verifique que estes serviços possam ser discriminatórios para a caracterização do contexto.

6.2 O Futuro

6.2.1 Computação Ubíqua

A Aplicação de Amostragem que desenvolvemos recorre apenas a sensores internos do próprio dispositivo e a um receptor de *GPS* que cada vez mais se torna como um elemento interno dos *PDA*s.

O facto do contexto estar dependente das tecnologias existentes nos dispositivos, leva a que estes estejam fechados sobre si mesmo e que o contexto não seja por isso mais enriquecido.

Relembrando a definição de computação ubíqua de Mark Weiser, onde o que se espera são ambientes saturados de computação e comunicação integrada de forma agradável, verificamos que este ambiente ainda não existe. Assim, julgamos que o futuro da computação ubíqua passa por ter, não dispositivos ubíquos, mas sim redes ubíquas de dispositivos. Na prática a essência destas redes está na partilha de informação entre os diferentes dispositivos. Vejamos um exemplo prático do que acontece actualmente e do que seria se tivéssemos perante uma rede ubíqua.

Cada vez mais se constata a adesão quer aos *PDA*s, quer aos sistemas de navegação incorporados nos automóveis. Actualmente o que se verifica é que ambos os dispositivos apenas falam para eles próprios, não partilhando qualquer informação. Vamos imaginar que um determinado indivíduo tem registado na agenda do seu *PDA* uma reunião para esse dia em determinado local e já inseriu no sistema de navegação da sua viatura o percurso que vai realizar. Como os dispositivos não partilham a informação, muito possivelmente o *PDA* apenas alerta o utilizador para a reunião já em cima da hora.

Vejamos agora a situação caso existisse a partilha de informação. Quando o utilizador inseriu o percurso no sistema de navegação, este calculou de imediato a duração da viagem. Se esta informação fosse transmitida ao *PDA*, este poderia decidir avisar o utilizador não antes do evento, mas sim minutos antes do momento em que se deveria dar a partida para o local da reunião, de forma a que o utilizador possa chegar no horário previsto.

Este é apenas um exemplo do ganho que se obteria na partilha de informação entre dispositivos e concerteza que existem muitos outros exemplos. Mas para que esta ideia seja

concretizada é necessário ainda um longo percurso a percorrer, principalmente na criação de normas tanto de comunicação como de estruturas de informação, pois só assim seria possível que diferentes dispositivos ¨falassem¨a mesma linguagem.

6.2.2 Tecnológico

Durante o desenvolvimento do presente trabalho por diversas vezes verificámos que a utilização dos *PDA*s e os receptores de *GPS* originava sérios problemas. Ou porque a comunicação *Bluetooth* era interrompida, ou porque um dos dispositivos ficava sem bateria, etc..., o que dificultou por um lado a aquisição de mais informação de contexto sobre o utilizador e por outro lado a utilização da Aplicação de Amostragem por mais utilizadores.

Com o crescente aumento de modelos de *PDA*s equipados com receptores internos de *GPS* estas questões tecnológicas tendem a ser diluídas. Uma vez que o receptor faz parte do próprio dispositivo, a comunicação *Bluetooth* deixa de ser necessária, eliminando assim parte do problema. Esta convergência permite também que seja necessário transportar apenas um único dispositivo e que a preocupação sobre a autonomia seja apenas uma.

Outro problema associado ao *GPS* estava relacionado com a forma como a *Compact Framework 1.0* acedia à informação. Com a recente disponibilização por parte da *Microsoft* da *Compact Framework 2.0*, o acesso à informação de *GPS* está mais facilitado. Nesta *framework* está disponível o *GPS Intermediate Driver*[1] que serve de *interface* entre o *PDA* e o receptor de *GPS*, disponibilizando para todas as aplicações que o necessitem, a informação referente à latitude e longitude da localização actual do utilizador.

Mas dentro do *PDA* existe ainda informação que pode ser interessante para caracterizar o contexto em determinadas aplicações *context-awareness*, como por exemplo, a informação da agenda. A *Microsoft*, ciente desta necessidade, incorporou na nova *framework* o *State and Notification Broker API*[2]. Este *API* permite monitorizar determinados aspectos do estado actual do *PDA*, sendo possível desta forma obter informação que pode ser utilizada para a definição do contexto, por exemplo, qual o próximo evento na agenda, assim como, criar automatismos de resposta a alterações do estado do dispositivo.

[1] Mais informação disponível em: http://msdn2.microsoft.com/en-us/library/ms850332.aspx

[2] Mais informação disponível em: http://msdn.microsoft.com/windowsmobile/reference/programming/api/default.aspx

Bibliografia

[Abowd et al., 2002] Abowd, G., Mynatt, E., and Rodden, T. (2002). The Human Experience. *IEEE Pervasive Computing*, 1(1):48–57.

[Abowd et al., 1999] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. (1999). Towards a Better Understanding of Context and Context-Awareness. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pag. 304–307, London, UK. Springer-Verlag.

[Aoki et al., 1999] Aoki, H., Schiele, B., and Pentland, A. (1999). Realtime Personal Positioning Systems for Wearable Computers. In *Proc. 3rd Int'l Symp. Wearable Computers*, pag. 37–43, Los Alamitos, California. IEEE CS Press.

[Bahl and Padmanabhan, 2000] Bahl, P. and Padmanabhan, V. (2000). Radar: An In-Building RF-Based User Location and Tracking System. In *Proc. 19th Ann. Joint Conf. IEEE Computer and Communications Societies (Infocom 2000)*, pag. 775–784, Los Alamitos, California. IEEE CS Press.

[Bento et al., 2005] Bento, C., Peixoto, J., and M., V. (2005). A Case-Based Approach for Indoor Location. In *Sixth International Conference on Case-Based Reasoning*, pag. 78–90, Chicago, Illinois (USA). Springer.

[Brown et al., 1997] Brown, P., Bovey, J., and Chen, X. (1997). Context-Aware Applications: From the Laboratory to the Marketplace. *IEEE Personal Communications*, 4(5):58–64.

[Dasgupta and Long, 2005] Dasgupta, S. and Long, P. M. (2005). Performance guarantees for hierarchical clustering. *J. Comput. Syst. Sci.*, 70(4):555–569.

[Davies and Gellersen, 2002] Davies, N. and Gellersen, H. (2002). Beyond Prototypes: Challenges in Deploying Ubiquitous Systems. *IEEE Pervasive Computing*, 1(1):26–34.

[Dey et al., 1998] Dey, A., Abowd, G., and Wood, A. (1998). CyberDesk: A Framework for Providing Self-Integrating Context- Aware Services. In *International Conference on Intelligent User Interfaces*, pag. 47–54.

[Dey and Abowd, 2000] Dey, A. K. and Abowd, G. D. (2000). The Context Toolkit: Aiding the Development of Context-Aware Applications. In *Workshop on Software Engineering for Wearable and Pervasive Computing*.

[Estrin et al., 2002] Estrin, D., Culler, D., Pister, K., and Sukhatme, G. (2002). Connecting the Physical World with Pervasive Networks. *IEEE Pervasive Computing*, 1(1):59–69.

[Farringdon et al., 1999] Farringdon, J., Moore, A., Tilbury, N., Church, J., and Biemond, P. (1999). Wearable Sensor Badge and Sensor Jacket for Context Awareness. In *Proceedings of the 3rd IEEE International Symposium on Wearable Computers*, pag. 107–113, Washington. IEEE Computer Society.

[Garlan et al., 2002] Garlan, D., Siewiorek, D., Smailagic, A., and Steenkiste, P. (2002). Project Aura: Towards Distraction-Free Pervasive Computing. *IEEE Pervasive Computing*, 1(2):22–31.

[Harte et al., 1998] Harte, L. J., Smith, A., and Jacobs, C. (1998). *Is-136 Tdma Technology, Economics and Services*. Mobile Communications Series. Artech House Publishers, 1st edition.

[Heckmann and Krüger, 2003] Heckmann, D. and Krüger, A. (2003). A User Modeling Markup Language (UserML) for Ubiquitous Computing. In *User Modeling*, volume 2702 of *Lecture Notes in Computer Science*, pag. 393–397. Springer.

[Henricksen et al., 2002] Henricksen, K., Indulska, J., and Rakotonirainy, A. (2002). Modeling Context Information in Pervasive Computing Systems. *Lecture Notes In Computer Science, Proceedings of the First International Conference on Pervasive Computing*, 2414:167–180.

[Hightower et al., 2006] Hightower, J., LaMarca, A., and Smith, I. (2006). Practical Lessons from Place Lab. *IEEE Pervasive Computing*, 5(3):32–39.

[Inc., 2005a] Inc., S. T. (2005a). NMEA Reference Manual. Technical report, SiRF Technology, Inc., San Jose, CA 95112 U.S.A.

[Inc., 2005b] Inc., S. T. (2005b). SiRF Binary Protocol Reference Manual. Technical report, SiRF Technology, Inc., San Jose, CA 95112 U.S.A.

[Jiang and Steenkiste, 2002] Jiang, C. and Steenkiste, P. (2002). A Hybrid Location Model with a Computable Location Identifier for Ubiquitous Computing. In Borriello, G. and Holmquist, L. E., editors, *Ubicomp*, volume 2498 of *Lecture Notes in Computer Science*, pag. 246–263. Springer.

[Judd and Steenkiste, 2002] Judd, G. and Steenkiste, P. (2002). Providing Contextual Information to Ubiquitous Computing Applications. Technical Report CMU-CS-02-154, Department of Computer Science, Carnegie Mellon University.

[LaMarca et al., 2005] LaMarca, A., Chawathe, Y., Consolvo, S., Hightower, J., Smith, I., Scott, J., Sohn, T., Howard, J., Hughes, J., Potter, F., Tabert, J., Powledge, P., Borriello, G., and Schilit, B. (2005). Place Lab: Device Positioning Using Radio Beacons in the Wild. In *Proceedings of Pervasive 2005*.

[Lee and Mase, 2001] Lee, S. and Mase, K. (2001). Recognition of Walking Behaviors for Pedestrian Navigation. In *Proc. IEEE Conf. Control Applications*, pag. 1152–1155, Piscataway, N.J. IEEE Control Systems Soc.

[Li, 2003] Li, W. (2003). A Service Oriented SIP Infrastructure for Adaptive and Context-Aware Wireless Services. In *Proceedings of MUM2003*.

[McFadden et al., 2004] McFadden, T., Henricksen, K., and Indulska, J. (2004). Automating Contextaware Application Development. In *First International Workshop on Advanced Context Modelling, Reasoning And Management, UbiComp 2004*.

[Motorola, 2000] Motorola (2000). ISU AT Command Reference. Technical report, Motorola Personal Communications Sector.

[Noll, 1998] Noll, A. M. (1998). *Introduction to Telephones and Telephone Systems*. Artech House Telecommunications Library, 3rd edition.

[Pascoe, 1998] Pascoe, J. (1998). Adding Generic Contextual Capabilities to Wearable Computers. In *2nd International Symposium on Wearable Computers*, pag. 92–99.

[Priyantha et al., 2000] Priyantha, N., Chakraborty, A., and Balakrishnan, H. (2000). The Cricket Location-Support System. In *Proc. 6th ACM Int'l Conf. Mobile Computing and Networking (Mobicom)*, pag. 32–43, New York.

[Raento et al., 2005] Raento, M., Oulasvirta, A., Petit, R., and Toivonen, H. (2005). ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications. *IEEE Pervasive Computing*, 4(2):51–59.

[Ranganathan et al., 2004] Ranganathan, A., Al-Muhtadi, J., and Campbell, R. H. (2004). Reasoning about Uncertain Contexts in Pervasive Computing Environments. *IEEE Pervasive Computing*, 3(2):62–70.

[Ryan et al., 1998] Ryan, N., Pascoe, J., and Morse, D. (1998). Enhanced Reality Fieldwork: the Context-Aware Archaeological Assistant. In *Computer Applications in Archaeology*.

[Satyanarayanan, 2002] Satyanarayanan, M. (2002). A Catalyst for Mobile and Ubiquitous Computing. *IEEE Pervasive Computing*, 1(1):2–5.

[Schilit et al., 1994] Schilit, B., Adams, N., and Want, R. (1994). Context-Aware Computing Applications. In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pag. 85–90.

[Schilit et al., 1993] Schilit, B., Theimer, M., and Welch, B. (1993). Customising mobile applications. In *USENIX Symposium on Mobile and Location-Independent Computing*.

[Strang et al., 2003] Strang, T., Linnhoff-Popien, C., and Frank, K. (2003). Integration Issues of an Ontology Based Context Modelling Approach. In *Proceedings of the IADIS International Conference WWW/Internet 2003, ICWI 2003*, pag. 361–368. IADIS.

[Wang et al., 2004] Wang, X. H., Zhang, D. Q., Gu, T., and Pung, H. K. (2004). Ontology Based Context Modeling and Reasoning using OWL. In *2nd IEEE Conference on Pervasive Computing and Communications Workshops (PerCom 2004 Workshops)*, pag. 18–22. IEEE Computer Society.

[Want and Hopper, 1992] Want, R. and Hopper, A. (1992). Active Badges and Personal Interactive Computing Objects. *IEEE Trans. Consumer Electronics*, 38(1):10–20.

[Want et al., 2002] Want, R., Pering, T., Borriello, G., and Farkas, K. (2002). Disappearing Hardware. *IEEE Pervasive Computing*, 1(1):36–47.

[Weiser, 2002] Weiser, M. (2002). The Computer for the 21st Century. *IEEE Pervasive Computing*, 1(1):19–25.

[Wilson, 2003] Wilson, J. (2003). Differences in Microsoft .NET Compact Framework Development between the Pocket PC and Windows CE .NET. Technical report, Microsoft, MSDN Library, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetcomp/html/netcfPPCtoCE.asp.

[Witten and Frank, 2005] Witten, I. H. and Frank, E. (2005). *Data Mining - Practical Machine Learning Tools and Techniques (Second Edition)*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, San Francisco, CA 94111, second edition.

[Yau et al., 2002] Yau, S., Karim, F., Y., W., Wang, B., and Gupta, S. (2002). Reconfigurable Context-Sensitive Middleware for Pervasive Computing. *IEEE Pervasive Computing*, 1(3):33–40.

Apêndice A

Arquitectura da Aplicação de Amostragem

A.1 Classe GPS_COM

A classe associada à leitura de mensagens *NMEA* oriundas do *GPS*, denominada de **GPS_COM**, é descrita da seguinte forma (Figura A.1):

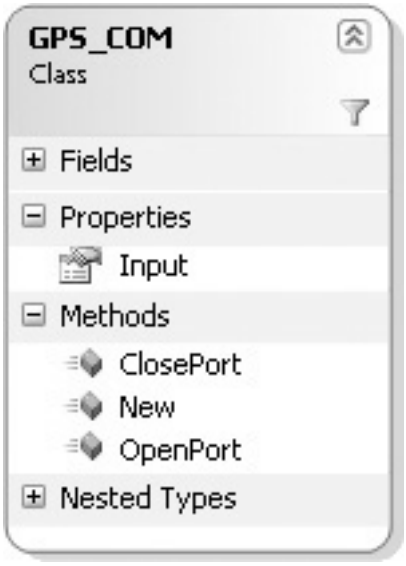

Figura A.1: Classe de comunicação com o *GPS*

o método `OpenPort` estabelece a ligação ao porto do *GPS*, a propriedade `Input` lê a cada

0,5 segundos os dados disponíveis no porto e por último, o método `ClosePort` que liberta o porto.

A.2 Classe GPS

Os resultados obtidos através da propriedade `Input` da classe `GPS_COM`, são utilizados numa outra classe denominada de **GPS**, que recebe da anterior os dados lidos em formato *string* e faz o respectivo processamento das mensagens *NMEA*. Esta última classe é responsável principalmente pela extracção da informação como: latitude, longitude, satélites, estado, velocidade e hora de satélite; que se encontrava anteriormente formatada segundo a norma *NMEA*. Na figura A.2 é possível verificar as propriedades associadas à informação extraída das mensagens *NMEA* e os três métodos necessários à extracção dessa informação.

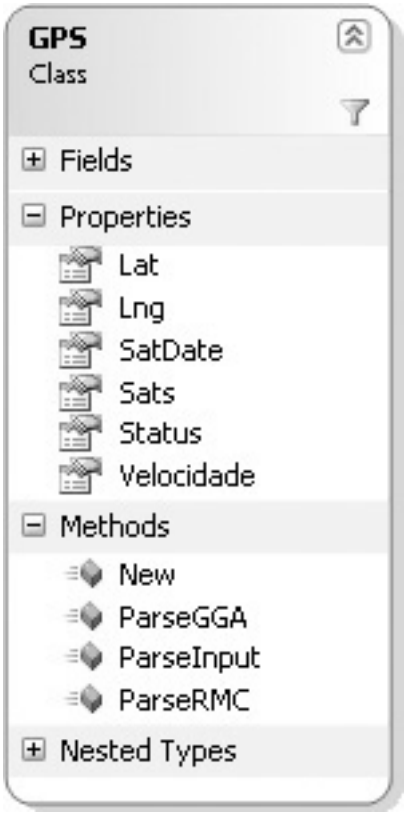

Figura A.2: Classe de extracção de dados GPS

O método `ParserInput`, tem como função apenas verificar de que tipo é a mensagem a processar, encaminhando ora para o método `ParserGGA`, ora para o método `ParserRMC`, a mensagem. Estes dois últimos métodos analisam a mensagem e associam às várias proprie-

dades desta classe os valores extraídos. Vejamos um exemplo para ilustrar o funcionamento desta classe. Supondo que as mensagens recebidas são as seguintes:

```
$GPRMC,031736,V,4043.3101,N,07317.5308,W,0.000,0.0,120800
```

```
$GPGGA,031736,4043.3101,N,07317.5308,W,1,06
```

os métodos `ParserRMC` e `ParserGGA` são executados e a partir destes são actualizados os valores das propriedades. Dentro destes métodos são ainda feitas algumas adaptações à informação. É retirada a casa decimal tanto à latitude como à longitude, de forma a simplicar os cálculos, convertida a data e hora para o formato próprio da linguagem e associado aos valores possíveis do *Status*, a informação de **Awaiting Fix**, **Sats Ok** ou **GPS Disable**, dependo do seu valor. Esta última transformação permite maior legibilidade do estado actual do receptor *GPS*, respectivamente: ou se encontra a fixar posição, ou tem posição fixa ou por último, não existe comunicação entre o PDA e o receptor de *GPS*.

A.3 Classe DadosGPS

Na implementação do nosso sistema, foi criada uma nova classe denominada de **DadosGPS**, que faz a abstracção da classe `GPS` permitindo a criação e leitura dos locais onde o utilizador interage com o *PDA*, assim como a detecção do nome abstracto do local onde este se encontra (Figura A.3).

Dentro desta classe existem dois métodos: **Guardar_Edificios** e **ler_edificios**. O primeiro método é responsável por a cada novo local que seja detectado, actualizar a informação sobre os locais, inserindo num ficheiro de dados em formato *XML* (Figura A.4) a informação referente ao local: nome, latitude e longitude. O método `ler_edificios` realiza a leitura dos vários locais que se encontram no ficheiro de dados e constrói uma estrutura de dados com os mesmos. Por último e o mais delicado, é a propriedade `dados` que começa inicialmente por verificar se o `Status` da informação *GPS* é válida, isto é, se as coordenadas de *GPS* correspondem efectivamente a uma posição fixa, assim como os valores para as propriedades `LatValida` e `LongValida`. Estas duas últimas propriedades guardam sempre as últimas coordenadas válidas da posição, uma vez que a cada instante podem existir valores nulos para a latitude e longitude. Se as coordenadas forem válidas, então o método começa por verificar se as mesmas correspondem a um local previamente inserido, se corresponderem obtemos assim o nome do local, senão é adicionado um novo local, sendo-lhe associado um nome abstracto, iniciado por `Place`, seguido de um número sequencial, para garantir a unicidade do local.

Esta classe é executa em intervalos de tempo de 1 segundo, devido ao facto de as mensagens *NMEA* que estamos a processar também serem produzidas dentro do mesmo intervalo

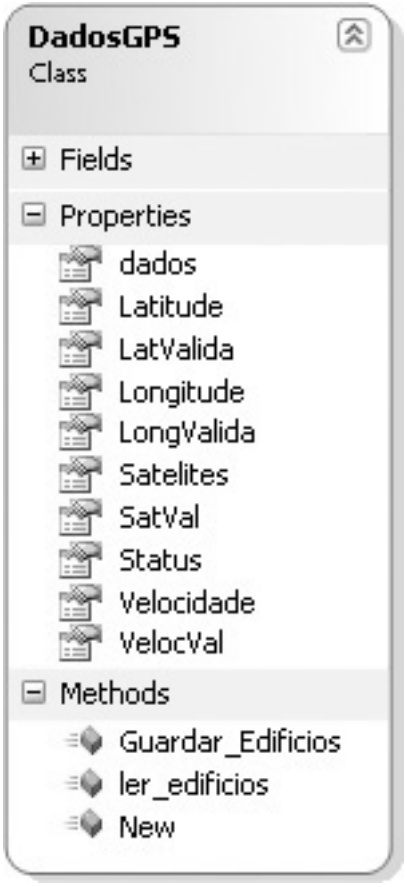

Figura A.3: Classe de abstracção dos locais

de tempo. Garantimos assim que a informação do local onde o utilizador se encontra está sempre actualizada.

Além da optimização referida na secção 4.4.2, foi realizada uma segunda optimização que tem a ver com a qualidade dos dados de *GPS*. Dos vários atributos de *GPS* que estávamos a recolher existe um que não estava a ser utilizado para determinar o local e que pode ser interessante. Até ao momento para determinar um local apenas utilizávamos a latitude e longitude. Mas sabemos que o receptor de *GPS* pode receber sinal de vários satélites e que quanto maior o número de satélites disponíveis, melhor será a precisão dos valores de latitude e longitude que são calculados. Relembremos o facto de que quanto maior o número de satélites, mais facilmente o erro de cálculo de *GPS* se aproxima de zero.

Este facto levou-nos a considerar também a propriedade `Satelites` presente na classe `DadosGPS`, como elemento importante para determinar quando as coordenadas devem ou não ser consideradas válidas. Sabemos que são necessários pelo menos 3 satélites para fazer a tri-lateração e assim obtermos uma posição de *GPS* válida. Mas como com apenas três

```
<?xml version="1.0" ?>
<buildings>
    <building>
        <nome>Place1</nome>
        <lat>40130666</lat>
        <long>8262793</long>
    </building>
</buildings>
```

Figura A.4: Estrutura de dados para representar os locais

satélites o erro de cálculo pode ainda ser significativo, considerámos que as mesmas apenas seriam válidas quando o número de satélites fosse de pelo menos quatro. Assim, para que as coordenadas actuais sejam consideradas de válidas é necessário cumprir três requisitos:

```
If Latitude > 0 and Longitude > 0 and Satelites > 3
```

A.4 Classe GSMInfo

Como a aplicação que estávamos a desenvolver era para ser executada principalmente em *PDA*s da marca *Qtek*, era necessário saber qual a porto série associado ao *modem GSM* que era por estes utilizado para o envio e recepção dos comandos AT. Utilizando para isso a aplicação *Regedit*, semelhante à que existe nos computadores pessoais, é possível verificar no registo do *PDA* que para esta marca em particular, existe um *driver* denominado de `Serial_Cmd` que serve de interface entre o *modem GSM* e a estação base (Figura A.5).

Pela figura anterior é possível verificar que associado a este *driver* está um porto série `Prefix = COM` com o atributo `Index = 2`. Na prática significa que utiliza para a comunicação o porto `COM2:` e que será a partir deste que a nossa classe deverá aceder à informação da localização da estação base. Implementando da forma idêntica ao *GPS*, é necessário inicialmente estabelecer a ligação a este porto:

```
hCom= CreateFile(L"COM2:",GENERIC_READ|GENERIC_WRITE,0,0,3,0,0)
```

A partir deste momento tanto o envio como a recepção de comandos AT funcionam de forma idêntica à escrita e leitura de ficheiros, respectivamente. Assim, seguindo os passos necessários apresentados anteriormente para obter a informação da estação base, primeiro temos que realizar o registo na rede:

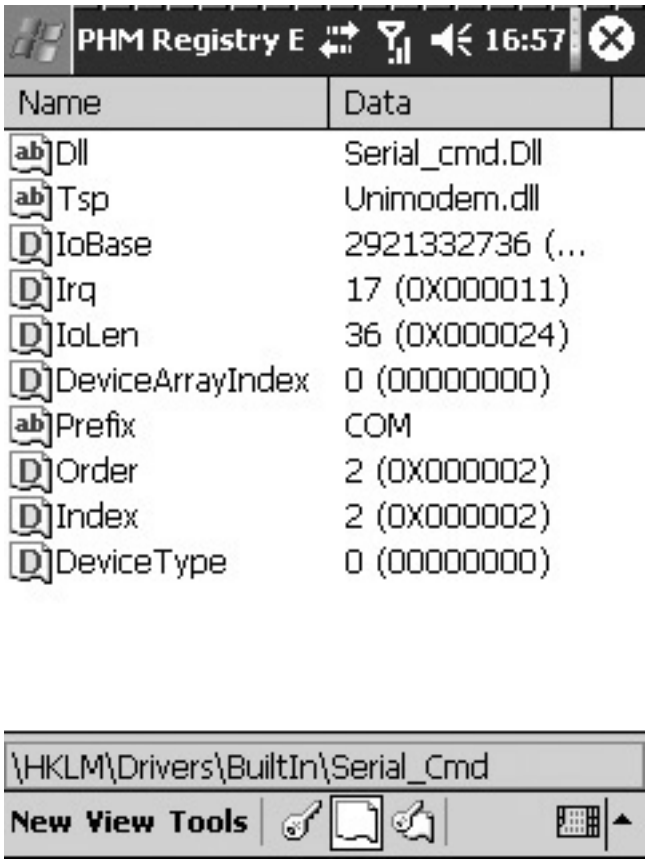

Figura A.5: Caracterização do *driver* Serial_Cmd

```
WriteFile(hCom, "AT+creg=2\r", 10, &nWritten, NULL)
```

Concluído o registo, é novamente enviado à estação base um comando AT, pedindo informação sobre a localização desta:

```
WriteFile(hCom,"AT+creg?\r", 9, &nWritten, NULL)
```

Como a estação base responde para o mesmo porto, é necessário agora ler os resultados enviados por esta:

```
ReadFile(hCom, buf+bufpos, 256 - bufpos, &nRead, NULL)
```

A partir deste momento é apenas necessário verificar se a mensagem enviada pela estação base contém o comando `CREG` e a partir desta extrair os identificadores da área e da estação base, e associar os mesmos a duas propriedades existentes na classe `LAC` e `Cellid`, respectivamente.

A classe aqui descrita, pode ser representada da seguinte forma (Figura A.6), onde o método `OpenPort` estabelece a ligação ao porto, o método `WriteAT` realiza os dois envios de

comandos AT, o método `ReadAT` aguarda a resposta da estação base e extrai a informação da mensagem e por último, o método `ClosePort` que liberta o porto.

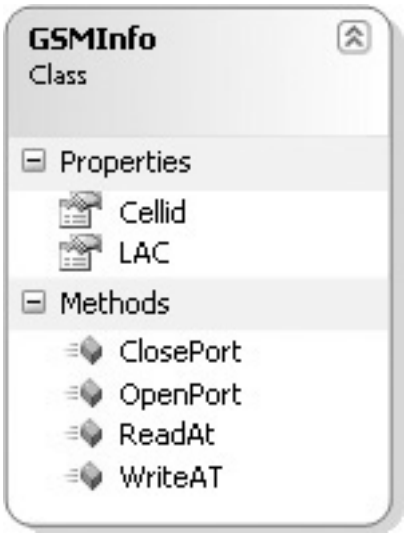

Figura A.6: Classe de localização GSM

Esta classe é utilizada pela classe principal da aplicação, sendo por isso executada em intervalos de 15 segundos.

A.5 Classe DadosWifi

A implementação tanto de um mecanismo de detecção da presença de serviços de comunicação utilizando a tecnologia *Wi-Fi*, como de um mecanismo de localização baseado na informação desta, originou um desafio. A *Compact Framework 1.0* não tem disponível qualquer classe para esta tecnologia e como tal foi necessário encontrar alternativas. A solução que encontrámos foi, adicionar outra *framework* que estende a mesma, tentando assim combater algumas lacunas que esta versão continha.

Esta *framework* denominada de *OpenNETCF Smart Device Framework*[1] tem nas versões mais recentes, disponível dentro do *namespace OpenNETCF.Net* um conjunto de classes que permite obter informação sobre a tecnologia *Wi-Fi*.

A primeira classe a considerar é a `Networking` (Figura A.7), que permite obter os vários adaptadores de rede existentes nos dispositivos.

Esta informação obtêm-se recorrendo à função `GetAdapters` da própria classe. Como resultado esta função devolve um conjunto de adaptadores de rede disponíveis. O exemplo

[1] *Framework* livre, disponível em http://www.opennetcf.org

Figura A.7: Classe *Networking*

seguinte demonstra como é obtida essa informação:

```
Private MyNetworking As OpenNETCF.Net.Networking
Private MyAdapters As OpenNETCF.Net.AdapterCollection
MyAdapters = MyNetworking.GetAdapters
```

O conjunto de adaptadores de rede obtidos é guardado numa estrutura de dados do tipo `AdapterCollection` (Figura A.8) e é com base nela que detectamos qual dos vários adaptadores identificados é o que corresponde à tecnologia *Wi-Fi*. De salientar que a lista de adaptadores obtida, apenas contém os que estão actualmente em funcionamento.

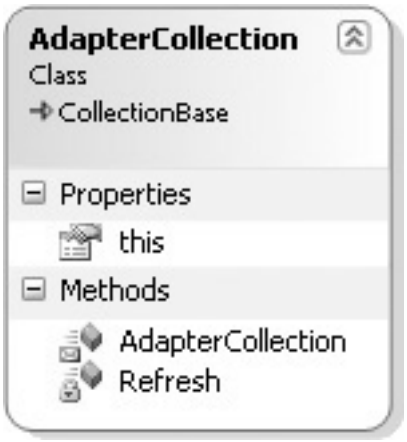

Figura A.8: Classe *AdapterCollection*

Para se obter o adaptador associado ao *Wi-Fi* basta procurar no conjunto de adaptadores, aquele cuja propriedade `IsWireless` é verdadeira. Esta propriedade faz parte da classe `Adapter` (Figura A.9).

O mecanismo de detecção da presença do serviço, resume-se apenas à obtenção da informação da propriedade `Name` da classe `Adapter` (Figura A.9). Quando existir informação

nesta propriedade é porque o serviço está em utilização. No nosso caso, sempre que a propriedade não tem nenhum valor atribuído, é lhe associado o valor `No WiFi` por defeito. O exemplo seguinte (Tabela A.1) demonstra as duas situações.

Propriedade *Name*	**Descrição**
TIACXWLN1	Presença de *Wi-Fi*
No WiFi	Ausência total ou momentânea de *Wi-Fi*

Tabela A.1: Exemplo dos valores possíveis para a propriedade *Name*

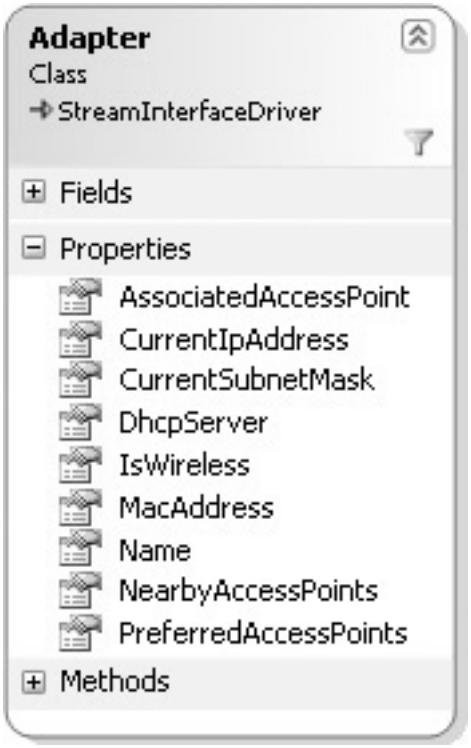

Figura A.9: Classe *Adapter*

Embora esta informação seja suficiente para determinar a presença ou não dos serviços de comunicação *Wi-Fi*, na nossa aplicação considerámos ainda a aquisição de mais características associadas ao adaptador de rede. Das várias propriedades associadas à classe `Adapter` recolhemos informação também das seguintes propriedades (Tabela A.2).

As propriedades `PreferredAccessPoints` e `NearbyAccessPoints` permitem obter uma lista de pontos de acesso, ou relacionados com a rede à qual estamos actualmente ligados ou às outras redes que se encontram disponíveis na vizinhança, respectivamente.

Propriedade	**Descrição**
CurrentIpAddress	Endereço IP da rede
CurrentSubnetMask	Máscara da *Subnet* da rede
MacAddress	Endereço *MAC* do adaptador de rede
DhcpServer	Endereço do servidor *DHCP*
AssociatedAccessPoint	*SSID* da rede
PreferredAccessPoints	*AP*s actualmente ligados
NearbyAccessPoints	*AP*s disponíveis na vizinhança

Tabela A.2: Restantes propriedade adquiridas

Apêndice B

Histórico Aplicacional com GPS e Aplicações

Neste apêndice é apresentado um extracto dos dados recolhidos recorrendo à nossa aplicação, relativos à aquisição do contexto, contendo informação sobre o local (apenas com a informação de *GPS*) e a aplicação que se encontrava activa nesse instante. Estes dados são o reflexo do primeiro protótipo funcional do nosso sistema, tendo servido apenas como teste à fiabilidade deste e validação dos dados adquiridos.

	Local	**Data**	**Hora**	**Dia**	**GPS**	**LatActual**
	LongActual	**VelocActual**	**Satelites**	**LatVal**	**LongVal**	**VelocVal**
	SatVal	**Processo**				
1	No info	1/29/2006	14:48:19	Sunday		0
	0	0	0	0	0	0
	0	BtConnManagerCE				
2	No info	1/29/2006	14:48:34	Sunday		0
	0	0	0	0	0	0
	0	BtConnManagerCE				
3	No info	1/29/2006	14:49:19	Sunday		0
	0	0	0	0	0	0
	0	BtConnManagerCE				
4	No info	1/29/2006	14:49:34	Sunday		0
	0	0	0	0	0	0
	0	shell32				
5	No info	1/29/2006	14:49:49	Sunday		0
	0	0	0	0	0	0
	0	shell32				
6	No info	1/29/2006	14:50:04	Sunday		0
	0	0	0	0	0	0
	0	shell32				

	Local	**Data**	**Hora**	**Dia**	**GPS**	**LatActual**
	LongActual	**VelocActual**	**Satelites**	**LatVal**	**LongVal**	**VelocVal**
	SatVal	**Processo**				
7	No info	1/29/2006	15:27:51	Sunday		0
	0	0	0	0	0	0
	0	cprog				
8	No info	1/29/2006	15:28:07	Sunday		0
	0	0	0	0	0	0
	0	cprog				
9	No info	1/29/2006	16:49:27	Sunday		0
	0	0	0	0	0	0
	0	fexplore				
10	No info	1/29/2006	16:49:42	Sunday		0
	0	0	0	0	0	0
	0	fexplore				
11	No info	1/29/2006	16:49:57	Sunday		0
	0	0	0	0	0	0
	0	fexplore				
12	No info	1/29/2006	18:24:51	Sunday		0
	0	0	0	0	0	0
	0	WordNetCE				
13	No info	1/29/2006	18:25:06	Sunday		0
	0	0	0	0	0	0
	0	WordNetCE				
14	No info	1/29/2006	18:25:21	Sunday		0
	0	0	0	0	0	0
	0	WordNetCE				
15	No info	1/29/2006	20:37:46	Sunday		0
	0	0	0	0	0	0
	0	TELNET				
16	No info	1/29/2006	20:38:02	Sunday		0
	0	0	0	0	0	0
	0	TELNET				
17	Place1	1/29/2006	12:04:29	Sunday	AWAITING FIX	40090000
	8206593	0	0	40090000	8206593	0
	0	shell32				
18	Place1	1/29/2006	12:09:44	Sunday	AWAITING FIX	40090000
	8206593	0	0	40090000	8206593	0
	0	shell32				
19	Place1	1/29/2006	12:10:44	Sunday	AWAITING FIX	40091800
	8206593	0	0	40091800	8206593	0
	0	shell32				
20	Place1	1/29/2006	12:22:05	Sunday	SATS OK	40091678
	8206573	0	3	40091678	8206573	0
	3	shell32				
21	Place1	1/29/2006	12:22:19	Sunday	SATS OK	40091678
	8206573	0	3	40091678	8206573	0
	3	BtConnManagerCE				
22	Place1	1/29/2006	12:22:34	Sunday	SATS OK	40091678
	8206573	0	3	40091678	8206573	0
	3	BtConnManagerCE				
23	Place1	1/29/2006	12:22:50	Sunday	SATS OK	40091678
	8206573	0	3	40091678	8206573	0
	3	shell32				

	Local	**Data**	**Hora**	**Dia**	**GPS**	**LatActual**
	LongActual	**VelocActual**	**Satelites**	**LatVal**	**LongVal**	**VelocVal**
	SatVal	**Processo**				
24	Place1	1/29/2006	12:23:05	Sunday	SATS OK	40091678
	8206573	0	3	40091678	8206573	0
	3	fexplore				
25	Place1	1/29/2006	12:23:20	Sunday	SATS OK	40091678
	8206573	0	3	40091678	8206573	0
	3	fexplore				
26	Place1	1/30/2006	17:32:48	Monday	AWAITING FIX	0
	0	0	0	40091873	8206657	0
	0	shell32				
27	Place1	1/30/2006	17:33:03	Monday	AWAITING FIX	0
	0	0	0	40091873	8206657	0
	0	shell32				
28	Place2	1/28/2006	14:35:50	Saturday	SATS OK	40091912
	8206531	0	3	40091912	8206531	0
	3	shell32				
29	Place2	1/28/2006	14:36:05	Saturday	AWAITING FIX	40091922
	8206530	0	0	40091922	8206530	0
	0	shell32				
30	Place2	1/28/2006	14:36:20	Saturday	AWAITING FIX	40091922
	8206530	0	0	40091922	8206530	0
	0	shell32				
31	Place2	1/28/2006	14:36:35	Saturday	AWAITING FIX	40091922
	8206530	0	0	40091922	8206530	0
	0	shell32				
32	Place2	1/28/2006	14:36:50	Saturday	AWAITING FIX	40091922
	8206530	0	0	40091922	8206530	0
	0	shell32				
33	Place3	02-01-2006	10:03:09	Wednesday	SATS OK	40101228
	8230508	0	3	40101228	8230508	0
	3	Camera				
34	Place3	02-01-2006	10:04:55	Wednesday	SATS OK	40102362
	8235505	0	4	40102362	8235505	0
	4	Camera				
35	Place3	02-01-2006	10:05:55	Wednesday	SATS OK	40106260
	8234717	0	3	40106260	8234717	0
	3	Camera				
36	Place4	02-01-2006	10:10:11	Wednesday	SATS OK	40111657
	8249914	0	0	40111657	8249914	0
	0	Camera				
37	Place4	02-01-2006	10:09:56	Wednesday	SATS OK	40111733
	8248660	0	4	40111733	8248660	0
	4	Camera				
38	Place4	02-01-2006	10:08:41	Wednesday	SATS OK	40111777
	8242613	0	3	40111777	8242613	0
	3	Camera				
39	Place5	1/28/2006	16:23:25	Saturday	SATS OK	40124986
	8250597	0	2	40124986	8250597	0
	2	shell32				
40	Place5	1/28/2006	16:24:55	Saturday	SATS OK	40128386
	8248326	0	2	40128386	8248326	0
	2	shell32				

	Local	**Data**	**Hora**	**Dia**	**GPS**	**LatActual**
	LongActual	**VelocActual**	**Satelites**	**LatVal**	**LongVal**	**VelocVal**
	SatVal	**Processo**				
41	Place5	1/28/2006	16:25:56	Saturday	SATS OK	40130901
	8248705	0	4	40130901	8248705	0
	4	shell32				
42	Place6	1/28/2006	16:28:26	Saturday	SATS OK	40131383
	8262109	0	2	40131383	8262109	0
	2	shell32				
43	Place6	1/28/2006	16:27:41	Saturday	SATS OK	40132934
	8257915	0	4	40132934	8257915	0
	4	shell32				
44	Place6	1/28/2006	16:29:41	Saturday	SATS OK	40137882
	8260974	0	5	40137882	8260974	0
	5	shell32				
45	Place7	1/28/2006	16:31:26	Saturday	SATS OK	40154760
	8266008	0	4	40154760	8266008	0
	4	shell32				
46	Place7	1/28/2006	16:31:56	Saturday	SATS OK	40159587
	8265551	0	5	40159587	8265551	0
	5	shell32				
47	Place8	1/28/2006	16:37:57	Saturday	SATS OK	40181318
	8211461	0	2	40181318	8211461	0
	2	shell32				
48	Place8	1/28/2006	16:37:27	Saturday	SATS OK	40182211
	8218752	0	6	40182211	8218752	0
	6	shell32				
49	Place9	1/28/2006	16:36:11	Saturday	SATS OK	40182678
	8233335	0	4	40182678	8233335	0
	4	shell32				
50	Place9	1/28/2006	16:36:27	Saturday	SATS OK	40184369
	8230985	0	4	40184369	8230985	0
	4	shell32				
51	Place10	1/28/2006	16:40:42	Saturday	SATS OK	40184855
	8194682	0	3	40184855	8194682	0
	3	shell32				
52	Place10	1/28/2006	16:41:42	Saturday	SATS OK	40188108
	8199539	0	3	40188108	8199539	0
	3	shell32				
53	Place10	1/28/2006	16:42:57	Saturday	SATS OK	40192822
	8204076	0	2	40192822	8204076	0
	2	shell32				
54	Place11	1/28/2006	16:45:43	Saturday	SATS OK	40202788
	8209544	0	2	40202788	8209544	0
	2	shell32				
55	Place11	1/28/2006	16:45:57	Saturday	SATS OK	40203698
	8208445	0	2	40203698	8208445	0
	2	shell32				
56	Place11	1/28/2006	16:47:28	Saturday	SATS OK	40208034
	8214434	0	3	40208034	8214434	0
	3	shell32				
57	Place11	1/28/2006	16:48:28	Saturday	SATS OK	40211376
	8216572	0	3	40211376	8216572	0
	3	shell32				

	Local	**Data**	**Hora**	**Dia**	**GPS**	**LatActual**
	LongActual	**VelocActual**	**Satelites**	**LatVal**	**LongVal**	**VelocVal**
	SatVal	**Processo**				
58	Place11	1/28/2006	16:48:43	Saturday	SATS OK	40212330
	8217614	0	2	40212330	8217614	0
	2	shell32				
59	Place12	1/28/2006	16:48:58	Saturday	SATS OK	40213558
	8217729	0	1	40213558	8217729	0
	1	shell32				
60	Place12	1/28/2006	16:50:13	Saturday	SATS OK	40217332
	8221545	0	1	40217332	8221545	0
	1	shell32				
61	Place12	1/28/2006	16:51:13	Saturday	SATS OK	40221350
	8224310	0	0	40221350	8224310	0
	0	shell32				
62	Place12	1/28/2006	16:52:13	Saturday	SATS OK	40225461
	8224290	0	2	40225461	8224290	0
	2	shell32				
63	Place12	1/28/2006	16:52:28	Saturday	SATS OK	40226712
	8223723	0	3	40226712	8223723	0
	3	shell32				
64	Place12	1/28/2006	16:52:58	Saturday	SATS OK	40226855
	8225871	0	0	40226855	8225871	0
	0	shell32				
65	Moving	1/28/2006	16:41:12	Saturday	SATS OK	40186630
	8196429	57	1	40186630	8196429	57
	1	shell32				
66	Moving	1/28/2006	16:41:57	Saturday	SATS OK	40189056
	8200036	57	2	40189056	8200036	57
	2	shell32				
67	Moving	1/28/2006	16:46:58	Saturday	SATS OK	40206716
	8211310	57	1	40206716	8211310	57
	1	shell32				
68	Moving	1/28/2006	16:49:58	Saturday	SATS OK	40217428
	8219767	57	3	40217428	8219767	57
	3	shell32				
69	Moving	1/28/2006	16:44:12	Saturday	SATS OK	40195930
	8205891	59	3	40195930	8205891	59
	3	shell32				
70	Moving	1/28/2006	16:49:28	Saturday	SATS OK	40215707
	8217615	59	4	40215707	8217615	59
	4	shell32				
71	Moving	1/28/2006	16:51:43	Saturday	SATS OK	40223303
	8225039	59	2	40223303	8225039	59
	2	shell32				
72	Moving	1/28/2006	16:51:59	Saturday	SATS OK	40224570
	8224630	59	2	40224570	8224630	59
	2	shell32				

Tabela B.1: Exemplo dos dados adquiridos

Apêndice C

Histórico Aplicacional com todos os atributos

Neste apêndice é apresentado um extracto dos dados recolhidos através da Aplicação de Amostragem com toda a informação de contexto disponível.

	LAC	CellID	Device	SSID	AP	Place	Process
1	18	41071	No WiFi	0	0	No info	fexplore
2	18	10181	No WiFi	0	0	No info	shell32
3	12	6591	No WiFi	0	0	No info	shell32
4	18	10181	TIACXWLN1	default	00-04-E2-FF-6C-F4	No info	fexplore
5	12	32583	TIACXWLN1	default	00-04-E2-FF-6C-F4	No info	shell32
6	18	10181	No WiFi	0	0	No info	iexplore
7	18	10181	TIACXWLN1	default	00-04-E2-FF-6C-F4	No info	iexplore
8	18	41071	TIACXWLN1	default	00-04-E2-FF-6C-F4	No info	TELNET
9	18	41071	TIACXWLN1	default	00-04-E2-FF-6C-F4	No info	cprog
10	12	7081	TIACXWLN1	default	00-04-E2-FF-6C-F4	No info	iexplore
11	12	32583	TIACXWLN1	default	00-04-E2-FF-6C-F4	No info	iexplore
12	12	7081	TIACXWLN1	default	00-04-E2-FF-6C-F4	No info	iexplore
13	18	41071	TIACXWLN1	default	00-04-E2-FF-6C-F4	No info	iexplore
14	18	41071	No WiFi	0	0	No info	iexplore
15	12	7081	TIACXWLN1	default	00-04-E2-FF-6C-F4	No info	iexplore
16	18	41071	No WiFi	0	0	No info	iexplore
17	18	10181	No WiFi	0	0	No info	iexplore
18	12	6591	No WiFi	0	0	No info	cprog

	LAC	CellID	Device	SSID	AP	Place	Process
19	12	6591	No WiFi	0	0	No info	shell32
20	18	41071	TIACXWLN1	default	00-04-E2-FF-6C-F4	No info	shell32
21	18	41071	No WiFi	0	0	No info	shell32
22	12	32583	No WiFi	0	0	No info	shell32
23	12	7081	No WiFi	0	0	No info	shell32
24	12	43812	No WiFi	0	0	No info	shell32
25	12	43812	No WiFi	0	0	No info	device
26	18	9862	No WiFi	0	0	No info	device
27	18	17811	No WiFi	0	0	No info	device
28	18	28002	No WiFi	0	0	No info	device
29	18	33951	TIACXWLN1	0	00-04-E2-FF-6C-F4	No info	shell32
30	18	33951	No WiFi	0	0	No info	gwes

Tabela C.1: Exemplo dos dados adquiridos

Apêndice D

Informação Agregada dos vários Utilizadores

Place	Processes
No info	fexplore, tmail
Place1	tmail, fexplore.exe
Place109	SuDoKu, tmail,fexplore
Place25	tmail, poutlook, fexplore
Place28	poutlook
Place31	poutlook
Place34	poutlook
Place37	poutlook
Place40	poutlook
Place43	tmail, poutlook, fexplore
Place46	poutlook
Place49	SuDoKu, fexplore, tmail
Place52	fexplore
Place55	tmail

Tabela D.1: Informação agregada para o Utilizador1

Place	Processes
No info	cprog, poutlook, TomTom Navigator, tmail, fexplore
Place1132	cprog, tmail, fexplore, TomTom Navigator
Place1144	TomTom Navigator
Place1297	eUtility
Place13	TomTom Navigator
Place1309	cprog, TomTom Navigator
Place1378	TomTom Navigator, Ctlpnl
Place1423	cprog
	continuação...

	continuação...
Place1582	poutlook
Place1585	poutlook
Place1591	poutlook
Place1702	TomTom Navigator
Place1723	TomTom Navigator
Place1768	TomTom Navigator
Place1783	TomTom Navigator, tmail
Place1786	TomTom Navigator
Place1822	TomTom Navigator
Place1834	TomTom Navigator
Place184	cprog
Place1858	TomTom Navigator
Place196	TomTom Navigator
Place241	TomTom Navigator
Place244	TomTom Navigator, fexplore
Place253	TomTom Navigator, cprog, poutlook, fexplore
Place307	cprog
Place34	TomTom Navigator
Place352	TomTom Navigator
Place355	TomTom Navigator
Place37	fexplore, AcroRd32
Place667	TomTom Navigator
Place70	cprog

Tabela D.2: Informação agregada para o Utilizador2

Place	**Processes**
No info	PocketSudoku, WordNetCE, poutlook, fexplore, Notes
Place10	fexplore
Place13	PocketSudoku, poutlook, fexplore, tmail, TomTom Navigator
Place130	poutlook
Place133	poutlook
Place136	poutlook
Place139	poutlook
Place142	poutlook
Place145	poutlook
Place148	poutlook
Place151	poutlook
Place154	poutlook
Place157	poutlook
Place16	PocketSudoku, fexplore, GPSTuner, TomTom Navigator, GPSTest
Place160	poutlook
Place163	poutlook
Place166	poutlook
Place169	poutlook
Place172	poutlook
Place175	poutlook
Place178	Notes, poutlook
Place19	PocketSudoku, Destinator, fexplore
Place214	WordNetCE
	continuação...

	continuação...
Place22	PocketSudoku, fexplore
Place25	PocketSudoku, fexplore
Place445	PocketSudoku
Place451	PocketSudoku
Place541	PocketSudoku
Place622	PocketSudoku
Place658	WordNetCE
Place661	WordNetCE
Place682	poutlook
Place685	poutlook
Place688	poutlook
Place691	poutlook
Place694	poutlook
Place697	poutlook
Place700	poutlook
Place703	poutlook
Place706	poutlook
Place709	poutlook
Place712	poutlook
Place715	poutlook
Place718	poutlook
Place721	poutlook
Place724	PocketSudoku
Place727	PocketSudoku
Place730	PocketSudoku
Place733	PocketSudoku
Place817	PocketSudoku
Place826	PocketSudoku
Place829	PocketSudoku
Place832	PocketSudoku
Place841	PocketSudoku
Place844	PocketSudoku
Place847	PocketSudoku
Place850	PocketSudoku
Place853	PocketSudoku
Place856	PocketSudoku
Place859	PocketSudoku
Place862	PocketSudoku
Place937	WordNetCE

Tabela D.3: Informação agregada para o Utilizador3

SSID	**Wifi**	**Processes**
	T	poutlook,cprog, fexplore, WLanMgr, tmail
0	F	poutlook, cprog, repllog, tmail, Camera
default	T	TELNET, iexplore, InstMsgr, poutlook, tmail
guest-e-U	T	cprog
jnfire	T	poutlook, fexplore, tmail, cprog, TELNET
SpeedTouch0FD108	T	TELNET

Tabela D.4: Informação agregada para o Utilizador4

LAC	CellID	Place	Processes
15500	5592	No info	PocketSudoku, cprog, tmail
15500	5592	Place61	PocketSudoku
15500	5592	Place94	PocketSudoku
15500	5592	Place73	PocketSudoku
15500	5592	Place70	PocketSudoku
15500	5592	Place103	PocketSudoku
15500	5592	Place106	PocketSudoku

Tabela D.5: Informação agregada para o Utilizador5

LAC	CellID	Place	Processes
12	6591	No info	solitare, TELNET, iexplore.exe, WLanMgr
18	6641	No info	PocketSudoku, cprog
18	6773	Place493	PocketSudoku
12	7081	No info	TELNET, cprog, solitare, Camera, iexplore
18	8481	Place493	PocketSudoku
18	8481	No info	BtConnManagerCE, Camera
18	9863	No info	poutlook, PocketSudoku, calc, cprog, WLanMgr
18	10181	No info	cprog, TELNET, iexplore, Camera, solitare
18	16521	No info	PocketSudoku, cprog, Camera
12	17741	No info	solitare, TELNET, TomTom Navigator, iexplore, cprog
18	17833	No info	cprog
12	32583	No info	TELNET
18	33951	No info	tmail, eJaKi, poutlook, Camera, cprog
18	33951	Place187	poutlook
18	33951	No info	pword, WLanMgr
18	33952	No info	Camera
18	34133	No info	Camera
18	41071	No info	solitare, iexplore, TELNET, cprog, TomTom Navigator
0	41071	No info	iexplore
18	53651	No info	cprog.exe

Tabela D.6: Informação agregada para o Utilizador6

LAC	CellID	Wifi	AP	Processes
18	6581	F	0	device
12	6591	F	0	iexplore, cprog, TELNET
18	6771	F	0	device
18	6772	F	0	device
18	6802	F	0	device
12	7081	F	0	iexplore, TELNET, cprog
12	7081	T	00-04-E2-FF-6C-F4	iexplore
18	9862	F	0	device
18	10181	F	0	iexplore
18	10181	T	00-04-E2-FF-6C-F4	iexplore, fexplore, cprog
18	15252	F	0	device
18	16391	F	0	device
18	16521	F	0	device
18	16523	F	0	device
12	17741	F	0	TELNET
18	17811	F	0	device
18	17812	F	0	device
18	17813	F	0	device
18	27982	F	0	device
18	28002	F	0	device
18	29741	F	0	device
12	32583	F	0	iexplore, TELNET
12	32583	T	00-04-E2-FF-6C-F4	iexplore
18	33951	F	0	device
18	33952	F	0	device
18	41071	F	0	iexplore, fexplore, WirelessMgr
18	41071	T	00-04-E2-FF-6C-F4	iexplore, cprog, TELNET
12	43812	F	0	device

Tabela D.7: Informação agregada para o Utilizador7

LAC	CellID	Place	Processes
15500	4591	No info	cprog, poutlook
15500	4593	No info	cprog
15500	4594	No info	cprog, poutlook
15500	5591	No info	PocketSudoku, cprog, tmail, pvbload, fexplore
15500	5591	Place328	cprog, tmail
15500	5591	Place319	PocketSudoku, tmail, fexplore
15500	5591	Place322	cprog, poutlook, fexplore
15500	5591	Place310	PocketSudoku, tmail, fexplore
15500	5591	Place376	cprog
15500	5591	Place340	tmail
15500	5591	Place334	fexplore
15500	5592	No info	PocketSudoku, cprog, fexplore, tmail, pvbload
15500	5592	Place319	PimHelper
15500	5593	No info	pvbload
15500	5594	Place322	cprog
15000	6191	No info	solitare
15000	6191	No info	PocketSudoku
15000	6191	No info	tmail
15000	6191	No info	poutlook
15000	6191	No info	TomTom Navigator
15000	6191	No info	fexplore
15000	6191	No info	pword
15000	6191	No info	cprog
15000	6191	No info	clock
15000	6191	Place322	PocketSudoku
15000	6191	Place370	PocketSudoku, poutlook, PimHelper
15000	6191	Place289	PocketSudoku
15000	6191	Place295	PocketSudoku
15000	6191	Place292	PocketSudoku
15000	6191	Place343	Backlight
15500	13791	No info	pword, pxl
15500	13793	No info	tmail
15000	21193	No info	PocketSudoku
15000	21292	Place271	PocketSudoku
15000	21993	No info	solitare, clock, PocketSudoku, poutlook, ctlpnl
15000	21993	Place271	PocketSudoku
15500	40191	No info	PocketSudoku, tmail

Tabela D.8: Informação agregada para o Utilizador8

Apêndice E

Resumo das regras para os vários Utilizadores

Neste apêndice estão disponíveis as várias regras de associação utilizando o algoritmo *Tertius* para os diferentes utilizadores. Devido à grande quantidade de regras obtidas, as mesmas encontram-se assim resumidas segundo a informação de localização e de comunicação *WiFi*.

Place	**Processes**
No info	poutlook, cprog, tmail, BloodPressureTracker, clock
Place1132	cprog, tmail, fexplore
Place1144	TomTom Navigator
Place253	TomTom Navigator, fexplore
Place352	TomTom Navigator
Place667	TomTom Navigator
Place34	TomTom Navigator
Place1585	poutlook
Place1423	cprog
Place1378	TomTom Navigator, ctlpnl
Place1309	cprog, TomTom Navigator
Place1723	TomTom Navigator
Place244	TomTom Navigator, fexplore
Place307	cprog
Place1768	TomTom Navigator
Place1822	TomTom Navigator
Place1834	TomTom Navigator
Place1297	eUtility
Place37	fexplore
Place1591	poutlook
Place1582	poutlook
	continuação...

	continuação...
Place70	cprog
Place184	cprog
Place1702	TomTom Navigator
Place13	TomTom Navigator
Place1786	TomTom Navigator
Place1783	TomTom Navigator
Place241	TomTom Navigator
Place1858	TomTom Navigator

Tabela E.1: Associações resumidas para o Utilizador2

Place	**Processes**
No info	WordNetCE, fexplore, iexplore, TomTom Navigator, Wceload
Place10	fexplore
Place13	PocketSudoku
Place16	PocketSudoku, GPSTuner, GPSTest
Place19	PocketSudoku, Destinator
Place22	PocketSudoku, fexplore
Place25	fexplore, PocketSudoku
Place130	poutlook
Place133	poutlook
Place139	poutlook
Place142	poutlook
Place148	poutlook
Place151	poutlook
Place157	poutlook
Place160	poutlook
Place166	poutlook
Place169	poutlook
Place175	poutlook
Place178	Notes, poutlook
Place214	WordNetCE
Place445	PocketSudoku
Place451	PocketSudoku
Place658	WordNetCE
Place661	WordNetCE
Place682	poutlook
Place685	poutlook
Place688	poutlook
Place694	poutlook
Place697	poutlook
Place703	poutlook
Place706	poutlook
Place709	poutlook
Place715	poutlook
Place718	poutlook
Place727	PocketSudoku
Place733	PocketSudoku
Place817	PocketSudoku
Place826	PocketSudoku
Place829	PocketSudoku
	continuação...

	continuação...
Place832	PocketSudoku
Place841	PocketSudoku
Place844	PocketSudoku
Place847	PocketSudoku
Place850	PocketSudoku
Place853	PocketSudoku
Place856	PocketSudoku
Place859	PocketSudoku
Place862	PocketSudoku
Place937	WordNetCE

Tabela E.2: Associações resumidas para o Utilizador3

Wifi	**Process**
T	fexplore
T	TELNET
T	tmail
T	poutlook
T	iexplore
T	InstMsgr
T	WirelessMgr
T	WLanMgr
T	cprog
F	cprog
F	poutlook
F	repllog
F	Camera
F	device

Tabela E.3: Associações resumidas para o Utilizador4

Place	**Processes**
No info	cprog
Place61	PocketSudoku
No info	tmail
Place94	PocketSudoku
Place103	PocketSudoku
Place73	PocketSudoku
Place70	PocketSudoku
Place106	PocketSudoku

Tabela E.4: Associações resumidas para o Utilizador5

Place	**Processes**
Place493	PocketSudoku
No info	tmail, eJaKi, solitare, cprog, TELNET
Place187	poutlook

Tabela E.5: Associações resumidas para o Utilizador6

CellID	Place	Processes
-	Place493	PocketSudoku
-	No info	tmail, eJaKi, solitare, cprog, TELNET
6591	-	solitare
6641	-	PocketSudoku
6773	-	PocketSudoku
7081	-	TELNET, cprog
8481	-	PocketSudoku
9863	-	poutlook
10181	-	cprog, TELNET
16521	-	PocketSudoku, cprog
17741	-	solitare
32583	-	TELNET
33951	-	tmail, eJaKi, poutlook
41071	-	solitare, iexplore
33951	No info	tmail, eJaKi
33952	-	Camera
53651	-	cprog

Tabela E.6: Associações com três atributos para o Utilizador6

CellID	Processes
6581	device
6591	iexplore
6771	device
6772	device
6802	device
7081	iexplore, TELNET
9862	device
10181	iexplore, fexplore
15252	device
16391	device
16521	device
16523	device
17741	TELNET
17811	device
17812	device
17813	device
27982	device
29741	device
32583	iexplore
33951	device
33952	device
41071	iexplore, cprog, WirelessMgr, fexplore
43812	device

Tabela E.7: Associações com dois atributos para o Utilizador7

CellID	Processes
153	clock
4591	cprog, poutlook
4593	cprog
4594	cprog
5591	cprog, tmail, PocketSudoku, pvbload
5592	PocketSudoku, cprog, fexplore, pvbload
5593	pvbload
6191	solitare, PocketSudoku, TomTom Navigator, poutlook, pword
13791	pxl, pword
13793	tmail
21193	PocketSudoku
21292	PocketSudoku
21993	solitare, clock, poutlook, repllog, ctlpnl
40191	PocketSudoku, tmail

Tabela E.8: Associações com dois atributos para o Utilizador8

Place	Processes
No info	solitare, clock, pvbload, PocketSudoku, fexplore
Place271	PocketSudoku
Place289	PocketSudoku
Place292	PocketSudoku
Place295	PocketSudoku
Place310	PocketSudoku
Place319	tmail, PocketSudoku
Place322	cprog, poutlook
Place328	cprog, tmail
Place334	fexplore
Place340	tmail
Place343	Backlight
Place370	PocketSudoku, poutlook, PimHelper
Place376	cprog

Tabela E.9: Associações com dois atributos para o Utilizador8

Apêndice F

Resumo de clusters para os vários Utilizadores

Place	Processes
No info	PocketSudoku, WordNetCE, fexplore, mstli, poutlook
Place13	PocketSudoku
Place16	PocketSudoku
Place178	poutlook
Place658	WordNetCE
Place214	WordNetCE

Tabela F.1: Resumo dos *clusters* para o Utilizador3

Place	Processes
Place94	PocketSudoku
No info	tmail, cprog, PocketSudoku
Place70	PocketSudoku
Place61	PocketSudoku
Place73	PocketSudoku
Place103	PocketSudoku
Place106	PocketSudoku

Tabela F.2: Resumo dos *clusters* para o Utilizador5

Place	Processes
No info	fexplore, tmail
Place1	tmail
Place25	tmail, poutlook
Place28	poutlook
Place31	poutlook
Place34	poutlook
Place37	poutlook
Place43	tmail, poutlook
Place46	poutlook
Place49	SuDoKu, tmail, fexplore
Place52	fexplore
Place55	tmail
Place109	tmail, SuDoKu, fexplore

Tabela F.3: Resumo dos *clusters* para o Utilizador1

Place	Processes
No info	fexplore, cprog, TomTom Navigator, poutlook, BloodPressureTracker
Place34	TomTom Navigator
Place667	TomTom Navigator
Place1591	poutlook
Place1582	poutlook

Tabela F.4: Resumo dos *clusters* para o Utilizador2

Wifi	Processes
F	poutlook, cprog, repllog
T	poutlook, iexplore, cprog, fexplore, TELNET, InstMsgr,tmail

Tabela F.5: Resumo dos *clusters* para o Utilizador4

CellID	Place	Processes
6591	No info	solitare
6641	No info	PocketSudoku
7081	No info	TELNET, cprog
8481	Place493	PocketSudoku
10181	No info	TELNET, cprog
16521	No info	PocketSudoku, cprog
17741	No info	solitare
33951	No info	tmail, eJaKi, Camera, poutlook
41071	No info	iexplore, solitare, fexplore, cprog

Tabela F.6: Resumo dos *clusters* para o Utilizador6

CellID	Processes
6591	iexplore
6802	device
7081	TELNET
7081	iexplore
10181	iexplore
17813	device
27982	device
32583	iexplore
33952	device
41071	iexplore

Tabela F.7: Resumo dos *clusters* para o Utilizador7

CellID	Place	Processes
5591	No info	PocketSudoku, cprog, tmail
5591	Place319	PocketSudoku, tmail
5591	Place322	cprog
5591	Place328	cprog
5592	No info	cprog, PocketSudoku
6191	No info	solitare, PocketSudoku, tmail, cprog,poutlook
6191	Place289	PocketSudoku
6191	Place322	PocketSudoku
6191	Place370	PocketSudoku
21993	No info	solitare, clock
21993	Place271	PocketSudoku

Tabela F.8: Resumo dos *clusters* para o Utilizador8