BUILDING AN MULTI-AGENT WHISKY RECOMMENDER SYSTEM

by

Torje Mjønes Coldevin

A thesis submitted in partial fulfilment of the requirements for the degree of

Master of Information Technology

University of Oslo

February 2005

UNIVERSITY OF OSLO

ABSTRACT

BUILDING AN MULTI-AGENT WHISKY RECOMMENDER SYSTEM

by Torje Mjønes Coldevin

MAS (Multi-Agent Systems), classifiers and other AI (Artificial Intelligence) techniques are increasingly becoming more common. The capability to handle complex and advanced problems by MAS was explored in this thesis. An MAS duty-free shopping recommender system was built for this purpose. The MAS system was part of a larger system built by the AmbieSense project. In addition, the AmbieSense project had built a prototype that was tested at Oslo Airport (OSL) Gardermoen. As a test case, the duty-free shopping system was set to classify and recommend whiskies.

The system incorporated several AI techniques such as agents, ontology, knowledge base and classifiers. The MAS was built using the JADE-LEAP framework, and Protégé was used for constructing the knowledge base. Various tests were performed for testing the system. Firstly, the agent communication was monitored using a Sniffer Agent. Secondly, the system's ability to run on mobile devices was tested on a PDA and various mobile phones. Thirdly, the MAS abilities compared to a 'normal' computer program were tested by replacing agents at run-time, using several JADE platforms, and by the experience gathered during development and the use of the developed system. Lastly, the recommendation was cross-validated against Dr. Wishart's whisky classification.

Weka was employed as a tool for testing different features and classifiers. Different classification algorithms are explained such as NNR (Nearest-Neighbour Rule), Bayesian, CBR (Case-Based Reasoning), cluster analysis and self-organizing feature maps. Two classification algorithms were tested; NNR and Bayesian. Features were tested using feature evaluation algorithms; information gain and ReliefF. The accuracy of the classification was tested using 10 fold cross-validation.

The testing showed that it is possible to make an MAS handling complex and advanced problems. It has also been shown that an MAS have some benefits in the areas of reliability, extensibility, computational efficiency and maintainability when compared to a

'normal' program. The error rate produced by the classifier was 56%; a figure which is too high for a recommendation system. Improvements could probably be achieved by finding better features or by selecting a different classifier. The system developed does not necessarily have to be used for duty-free shopping but could also be used for any shopping items.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to thank my supervisor; Lecturer Gisle Hannemyr at the University of Oslo for giving me the opportunity to write a thesis in the field of Artificial Intelligence and for his help and tutoring throughout the course of my work. Also, I would like to thank CognIT a.s for assisting me in finding a suitable assignment for my thesis, reading, commenting and the support of my work. Especially, I would like to thank Dr. Robert Engels, Cand. Phil. Till C. Lech and MSc Leendert W. M. Wienhofen. Further, I would like to thank my friend; Ole Johan Kristiansen for his help in proof reading the thesis. Last but not least, I would thank my girlfriend; Grace E.P Yeo for her support during the course of the thesis and for her contribution to poof reading.

Torje M. Coldevin
Oslo, Norway
January 2005

GLOSSARY

ACC. Agent Communication Channel
ACL. Agent Communication Language
AI. Artificial Intelligence
AID. Agent IDentifier
AMS. Agent Management System
API. Application Program Interface
CBR. Case Based Reasoning
CDC. Connected Device Configuration
CLDC. Connected Limited Device Configuration
CORBA. Common Object Request Broker Architecture
DAML. DARPA Agent Markup Language
DF. Directory Facilitator
ER. Entity Relationship
FIPA. Foundations for Intelligent Physical Agents
GUI. Graphical User Interface
HTTP. Hypertext Transport Protocol
IIOP. Internet Inter-ORB Protocol
IP. Internet Protocol
J2ME. Java 2 Mobile Edition
J2SE. Java 2 Standard Edition
JADE. Java Agent DEvelopment Framework
JVM. Java Virtual Machine
KB. Knowledge Base
KIF. Knowledge Interchange Format
KQML. Knowledge Query and Manipulation Language
KVM. Kilo Virtual Machine
LEAP. Lightweight Efficient Application Protocol
MAS. Multi-Agent System
MIDP. Mobile Information Device Profile
MTP. Message Transport Protocol
NNR. Nearest-Neighbour Rule
OSL. Oslo Airport Gardermoen
OWL. Web Ontology Language
PDA. Personal Digital Assistant

RDF. Resource Description Framework
RDFS. RDF Schema
RMI. Remote Message Invocation
SL. Subset Language
SSL. Secure Socket Layer
UML. Unified Modeling Language
URI. Uniform Resource Identifier
URL. Uniform Resource Locator
XML. eXtensible Markup Language

Chapter I

INTRODUCTION

My thesis is part of a larger research project called AmbieSense. The project is funded by EU and host many different companies, organisations and research institutions such as Siemens, CognIT, Oslo Airport, Sevilla Global, SINTEF, NTNU, Robert Gordon University, Lonely Planet, Reuters and Yellow Map.

AmbieSense is a project which combines different technologies (hardware and software) such as high-tech antennas, mobile devices, context sensitive software and artificial intelligence.

A new antenna called Context Tag has been developed by Siemens and SINTEF and is based on Bluetooth technology, which is a specification for short-range wireless communication. Due to Bluetooth's short-ranged nature, it can transmit information only to nearby wireless devices. A Context Tag is basically a Bluetooth antenna that transmits information about its location to nearby wireless devices. In other words, when a user with a wireless device encounters a Context Tag, information about the Context Tag's location is automatically sent to the user's wireless device.

An example where this technology could be useful would be in a museum, where each room contains a Context Tag. When a user enters a room, information relating to the displayed items in the room is displayed on the user's mobile device. Another example could be an airport where information about departure time and check-in are provided on the travellers' PDA or mobile phone.

The software needed to provide this information is developed by CognIT, Robert Gordon University, SINTEF and NTNU. It is based on an MAS *(Multi-Agent System)*, where several agents can cooperate and solve problems. The system is going to be 'context aware', which means that the system can act based on context information which is perceived, stored and analysed. An example of this behaviour could be a mobile phone that knows which ring signal is suited for a current situation. I.e. when you are in a meeting the phone turns off sound signals and only offers vibration, or if you are in a nightclub the phone turns the ring signal to maximum. A context aware system needs input through sensors like hearing, vision, a keyboard or other input devices. Input that the phone could use would be the calendar or direct input from the keypad. A more

advanced approach would be that the phone could record sound from the environment and then decide which ring signal is best suited.

AmbieSense uses an advanced approach for handling contexts where the system records different contexts and later match these stored contexts with the current situation. In this way, the system knows how to react to the current situation. For the phone example, this means that if the surrounding conditions are the same as before experienced, the ring signal is set back to what was done previously.

The project's goal is to make a general framework that allows for different application of the technology. As a demonstration of what this technology can be used for, AmbieSense is producing a prototype based on a traveller's scenario. There is already a developed prototype in cooperation with Yellow Map which gives a traveller map information and position gathered from GPS.

The system is now going to be extended to handle calendar information, personal preferences, information about the environment, recommendations about interesting sights and transport planning. Basically the system is going to support a traveller on its journey from home to its destination and back again. There are several information vendors which are supplying information content to the project such as Reuter, Lonely Planet and Yellow Map. There are two users which are testing this system; Oslo Airport and Sevilla Global. Oslo Airport is going to have Context Tags mounted at several locations like the check-in, at some duty-free shops, cafes and so on.

I am involved in CognIT's developer's team where software associated with the Context Tags is being developed. My task is to give content to the system where travellers and tourists can get personalised information adapted to their own interests. I have focused on duty-free shopping as a service that travellers might want and especially on whisky shopping. Whisky shopping is chosen due to its limited domain, around 80-100 products, but big enough for a recommender system to be appreciated. I am especially interested in technologies such as MAS, recommender systems and artificial intelligence in general.

I have defined some scientific problems which I am going to answer in the course of my thesis, they are:

How can we make an MAS with 'intelligence' that can handle complex and advanced problems? An example for a system like this could be a whisky recommender system, which gives offers on interesting whiskies, recommend interesting whiskies and informs about cheap offers. The system must be able to classify a whisky based on some specified

properties and recommend a suited candidate to the user. Is an MAS better suited than a 'normal' computer program?

How can we measure that an agent is good enough? By 'good enough', I am referring to the accuracy of the prediction given by the agent's classifier. The accuracy expected should probably be in the area of less than 10-20% wrongly classified whiskies. Which methods can we use? Some methods such as mathematical methods (one example could be NNR *(Nearest-Neighbour Rule)*), trial and error, the used of known test data or comments from experts.

Chapter II

MULTI-AGENT SYSTEMS

One of the main themes in my thesis is multi-agent systems. This chapter will explain the theory behind it and definitions commonly used. An MAS *(Multi-Agent System)* is as we would expect a system that consists of several agents. To fully understand what a multi-agent system is and which benefits it can give us, we have to know what an agent is and the difference between a single-agent system and an MAS.

2.1. Agent

Agents are becoming increasingly more popular in mainstream computer science and in the field of AI. They can be found in e-mail programs, running on a server providing OS-services or with the client as a GUI-agent. So what is an agent really? The word agent comes from the Latin word *agree* which means to do (Norvig & Russell (2003) p. 4). If we look up the word in a dictionary we find the following definition: **a·gent** "One that acts or has the power or authority to act" (www.dictionary.com). Act is something that every computer program does, so what then distinguishes a 'normal' computer program from an agent? Norvig and Russell (2003) defines an agent as; "anything that can be viewed as perceiving the environment through sensors and acting upon that environment through actuators" (p. 32). Following this definition a computer program must have some sensors like a camera, a keyboard or other sensors and be able to act from these inputs through some actuators like a screen. Wooldridge and Jennings (1995) gives an agent the following properties: autonomy, social ability, reactivity and pro-activeness (p.4). Autonomy and reactivity are covered by Norvig & Russell's definition. Autonomy comes from autonomous which means something that is not controlled by others or by outside forces (www.dictionary.com). An autonomous agent is therefore an agent that relies on sensory input for making decisions, instead of relying on knowledge given at design time. An agent is reactive because it perceives the environment and responds to it. In addition Wooldridge & Jennings have properties of being social and pro-active; Social reflects the fact that agents interacts with other agents through communication. Pro-active is the ability to take initiative to pursue goals.

In the field of AI, agents are often given even more abilities to make them more human-like, common abilities are: knowledge, belief, intention and obligation.

So far I have not been talking about an 'intelligent' agent. Because of the controversy and problems of defining the word intelligence, I will use the term rational agents to describe 'intelligent' agents. Norvig & Russell (2003) defines a rational agent as; "For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has" (p. 36). This basically means that a rational agent is an agent that takes the optimal expected decision in every situation, based on the information it possesses. However, this does not mean that the decision is the right decision.

2.2. Difference between Multi-Agent systems and Single-Agent Systems

An agent in an MAS must be influenced by neighbour agents through communications or by other means. In a single agent system, an agent is only concerned about its own goals, actions and knowledge. Other agents may exist, but they are not accounted for. Although it might seem that single-agent systems should be simpler than MAS, when dealing with complex tasks the opposite is often the case (Stone & Veloso (1997) p. 5). Multi-agent systems consist of several agents who model each others goals and actions (Stone & Veloso (1997) p. 6). In a single agent system, an agent must be able to solve all tasks which it is required to, whereas in an MAS, an agent can cooperate with other agents to solve more complex tasks than it could do on its own.

1. Sycara (1998) mentions some characteristic with an MAS (p. 2).

2. Each agent has incomplete information or capabilities to solve a problem.

3. There is no system global control.

4. Data is decentralized.

5. Computation is asynchronous.

An MAS can in some situations be thought of as a problem solving organisation. But in others, competition is more important than cooperation. A stockbroker MAS can be an example of this where every agent is competing for the best deals.

Whichever MAS it is, be it competitive or cooperative, agents must be able to consider other agents' actions. Stone & Veloso have made a taxonomy which divide agents by the degree of awareness they exercise, from the simplest scenario with homogenous agents to the most complex scenario with heterogeneous communicating agents (Stone & Veloso (1997) c. 4-6).

1. Homogenous Non-Communicating Multi-Agent Systems

2. Heterogeneous Non-Communicating Multi-Agent Systems

3. Heterogeneous & Homogenous Communicating Multi-Agent Systems

2.2.1. Homogenous Non-Communicating Multi-Agent Systems

Homogenous agents have the same internal structure, goals, knowledge and actions. This makes it easy for homogenous agents to predict what other agents might be doing next. The only difference between two agents is their current location (Stone & Veloso (1997)).

2.2.2. Heterogeneous Non-Communicating Multi-Agent Systems

Heterogeneous agents give more power through differences and special abilities, but it also adds more complexity. Heterogeneous agents have different internal structure, with different goals, models of the world and actions they can perform. Hence heterogeneous agents must have an ability to predict what other agents actions are going to be. To achieve this, heterogeneous agents need to be more complex than homogenous agents to be able to observe and learn how other agents behave (Stone & Veloso (1997)). The benefit is that heterogonous agents can use the differences and join them together to solve tasks which they could not have solved on their own.

2.2.3. Heterogeneous & Homogenous Communicating Multi-Agent Systems

Adding communication gives the agents possibilities to coordinate and work more efficiently than without communication. This applies to both heterogeneous and homogenous agents. An MAS that communicate can perform tasks with same complexity as single agent systems can perform, this of course depends on how well the communication works between the agents. We are going to look into agent communication more thoroughly later in this chapter.

2.3. Pros and Cons Multi-Agent System

So far we have seen different kinds of multi-agent systems without looking at which benefits an MAS design brings and if there are any major drawbacks with this design.

Carnegie Mellon University (2001) has listed some benefits that are important with an MAS:

- An MAS distributes computational resources and capabilities across a network of interconnected agents. Whereas a centralized system may be plagued by resource limitations, performance bottlenecks, or critical failures, an MAS is decentralised and thus does not suffer from the 'single point of failure' problem associated with centralised systems.

- An MAS allows for the interconnection and interoperation of multiple existing legacy systems. By building an agent wrapper around such systems, they can be incorporated into an agent society.

- An MAS models problems in terms of autonomous interacting component-agents, which is proving to be a more natural way of representing task allocation, team planning, user preferences, open environments, and so on.

- An MAS efficiently retrieves, filters, and globally coordinates information from sources that are spatially distributed.

- An MAS provides solutions in situations where expertise is spatially and temporally distributed.

- An MAS enhances overall system performance, specifically along the dimensions of computational efficiency, reliability, extensibility, robustness, maintainability, responsiveness, flexibility, and reuse.

Lesser (1995) look at some of the problems that MAS have to deal with compared to a single-agent system:

- Limited communication bandwidth and the computational costs of packaging and assimilating communicated information.

- The heterogeneity of agents, which makes it difficult to share information and the potential for competitive agents who, for their own self-interest, are not willing to share certain information.

- The dynamic character of the environment due to changing problems, agents, and resources, and the inability to predict with certainty the outcome of agents' actions.

We can see that an MAS is not suited for every situation. Since it depends on communication, it requires that there is always a minimum of bandwidth available for communication. The communication also comes with a price, with extra computational costs and added complexity. However, if these requirements are met, an MAS can be more robust, flexible and extensible than a single agent system. Most of the problems associated with MAS are overcome by new technology which provides faster and more efficient communication, better algorithms for planning and cooperating in an MAS.

2.4. Agent Communication

As mentioned earlier, agents need to communicate so that they can work and cooperate efficiently together. Agent communication is a field of research where three key elements are of importance (Flores-Mendez (1999)):

- A common agent communication language and protocol

- A common format for the content of communication

- A shared ontology

2.4.1. Agent Communication Language

There are two main approaches for design of an ACL *(Agent Communication Language)* Procedural and declarative language. Procedural is 'common' programming language like Java or C, where the programmer specifies the sequence of steps to be executed.

Declarative language describes the relationship between variables based on declarative statements such as through functions, definitions and assumptions. They are also most commonly used for agent communication. This is because a declarative language gives the opportunity to use knowledge stored in the system for tasks not planned (Flores-Mendez (1999)). There are several declarative languages, such declarative languages are; *Prolog* which is one of the first declarative languages and KQML *(Knowledge Query and Manipulation Language)* which is one of the most popular languages for agent communication.

2.4.2. Ontology

For agents to communicate it is not sufficient with a common language, there is also a need for sharing the same understanding about objects and concepts in the world in which they operate.

An ontology is defined in the dictionary as: **on·tol·o·gy** "The branch of metaphysics that deals with the nature of being" (www.dictionary.com). Like the definition of agent, AI (*Artificial Intelligence)* researchers like Gruber (1993) have their own meaning of ontology which differs from the general definition.

"An ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy, where an ontology is a systematic account of Existence. For AI systems, what 'exists' is that which can be represented. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the universe of discourse. This set of objects, and the describable relationships among them, are reflected in the representational vocabulary with which a knowledge-based program represents knowledge. Thus, in the context of AI, we can describe the ontology of a program by defining a set of representational terms. In such an ontology, definitions associate the names of entities in the universe of discourse (e.g. classes, relations, functions, or other objects) with human-readable text describing what the names mean, and formal axioms that constrain the interpretation and well-formed use of these terms. Formally, an ontology is the statement of a logical theory."

An ontology can be seen as a common model or vocabulary of a domain, described by objects, properties and relationships. An example could be two persons talking about an archive. Both parties know the English word "archive", but both parties have different interpretations of the word related to the context in which it is used. An archive worker might think of the word archive as a physical archive where paper files are stored, whereas an IT-professional would think of a zip-file. Both parties have a different mental model of the same word. Ontologies can be used as a joined understanding of an area or domain, such as medical terms, IT expressions and so on. Agents can in the same way use ontologies which are suited to its particular task.

Ontologies can be valuable for agent communication to avoid misunderstanding and to share knowledge in a formal representation. Ontologies are also playing an important role

for making internet available for machines in the future where different information is modelled by different ontologies.

For sharing ontologies, common *knowledge representation languages* are developed, some of them are *First-Order Logic*, RDF (*Resource Description Framework)* and OWL (*Web Ontology Language).* Let us have a look at the different languages, since they might be useful when creating an ontology or a knowledge base (knowledge base is described in chapter 5.2.2.2.1).

2.4.2.1. First-Order Logic

First-Order logic is one of the earliest and most basic forms of knowledge representation of languages. First-Order logic is based on mathematical logic, which has its origins in philosophy. It consists of three types of symbols; constants, predicates and functions, which represent objects, relations and functions respectively (Norvig & Russell (2003) p. 246). An example of a First-Order logic sentence would be:

Malt(Glenfiddich)

In plain English this sentence says: "Glenfiddich is a malt" The constant or object in this sentence is Glenfiddich and the predicate or relation is Malt.

Let us say we have the general rule stating:

Malt(x) => Whisky(x)

Then we can infer the following:

Malt(Glenfiddich) => Whisky(Glenfiddich)

We have inferred that if Glenfiddich is a malt then it is also a whisky. Inference is one of the major benefits of using a knowledge representation language.

The biggest drawback using First-Order logic for ontologies is the fact that it is not possible to represent exceptions and uncertainty. Even though we have a rule stating that

tomatoes are red, it is possible to have green, yellow and orange tomatoes. For handling such cases more advanced representation is needed (Norvig & Russell (2003) p. 321).

2.4.2.2. RDF/RDFS & OWL

RDF is a language set out to describe properties and metadata about web resources in a machine parsable form. A web resource could be a web page but because of the general description of RDF, a web resource could be any resource which can be described on the Web (Manola & Miller (2004) p.1).

A resource's properties are expressed as RDF statements. Each statement consists of a triplet; a subject, a predicate and an object. The subject is the thing that has properties, this could be a web page or, in my project, a whisky. The predicate describes the property of the subject, this could be the author of the Web page or the age of the whisky. Finally the object describes the value of the property which could be "John Smith" or, in the whisky example, the number 16.

Since RDF is created for handling resources on the Web, it can identify the subjects, objects and predicates using a unique identifier. This identifier is called URI *(Uniform Resource Identifier)*. The unique identifier gives benefits when sharing information with others on the Web; if you are talking about a resource you simply pass the URI of that resource, and there would be no confusion about which resource that is in question. Objects may also be represented as literals; a character string. To further clarify the RDF language let us look at an example:

If we want to represent the English sentence; "The webpage www.example.org has an author John Doe"

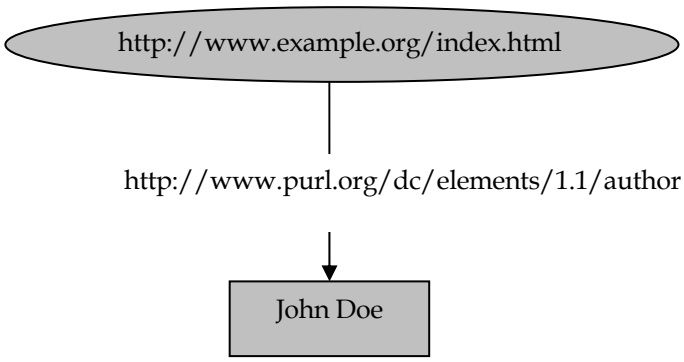


Figure 2.1: Example - RDF

From the figure 2.1 we see that the web page and the predicate are specified by a URI. In particular, the web page uses a special URI address called URL *(Uniform Resource Locator)* and the predicate refers to the Dublin Core. The Dublin Core is a collection of standard properties used for describing a web document and contains properties such as title, creator, subject, description and more (Manola & Miller (2004) ch.6).

It would be rather inconvenient to express RDF using drawings, and quite hard for machines to understand, so the RDF provides a XML-based syntax. The same sentence as described earlier in figure 2.1 is here represented as a RDF/XML.

```
<?xml version="1.0"?>

<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
xmlns:dc="http://www.purl.org/dc/elements/1.1/">

  <rdf:Description about="http://www.example.org/index.html">
    <dc:author>John Doe</dc:author>
  </rdf:Description>
</rdf:RDF>
```

Figure 2.2: Example - RDF/XML

Note that the property *author* use the abbreviated *dc*, instead of the long URI. This makes it more readable. From the XML syntax we can see that RDF is quite tedious to read and write.

RDF is often used in conjunction with RDFS *(RDF Schema)*. RDFS provides the ability of declaring classes. This means that in addition to only describing properties of a resource as RDFS does, it is also possible to describe the class which the resource belongs to. This gives the RDF/RDFS language some of the same abilities as an object-oriented programming language. RDF Schema even allows instances to be created for each class (Manola & Miller (2004) c. 5). RDFS does not allow multiple inheritances, nor does it identify two similar classes, and finally it does not allow inference such as transitivity and symmetry. OWL is a vocabulary extension of RDF/RDFS which targets these problems (Dean & Schreiber (2004)).

More information about RDF, RDFS & OWL can be found on the W3C web pages (www.w3.org/RDF/).

Chapter III

RECOMMENDER SYSTEMS

One of the tasks in the AmbieSense project is to support duty free shopping (also called tax-free shopping). Since I want to equip my agents with some kind of 'intelligence', an obvious choice would be to give some agents expert abilities.

To illustrate how this system might work, we can look at an example. For instance, if a traveller wants to buy a whisky, the system should be able to recommend a whisky that the traveller is interested in. The recommendation is based on the information stored in the user's context. The user context contains the user's preferences and other information relevant to the user. One possible scenario would be a traveller who wants to buy a Glenfiddich whisky, but the duty free shop does not have this whisky in stock. The system then recommends a Glen Spey since it knows that this whisky has a taste which is similar to a Glenfiddich, and is available at the duty free shop.

There are several ways to recommend an item. One way can be to look at predefined classifications made by experts. Whiskies are by some experts and classification systems grouped into 10 categories, where whiskies in the same category are said to be similar. A computer system could easily use a classification like this for recommending a whisky. For the system to recommend a similar whisky it only has to perform a table lookup.

This approach proves to be efficient if we only want to find a whisky similar in taste to another whisky, based on the structures already drawn by the expert. Users can prefer certain features which divide the whiskies into other structures than a 10 categories classification. If we want a system to handle every user's special preferences, we need something more sophisticated. A system like this could be based on some kind of classification technique. Computer based classification systems are often known as 'machine learning', 'artificial intelligence', 'pattern recognition' or by other names.

Since this system must be able to recommend all kinds of duty free products, a general framework would be a good help. Properties for such a framework can be a general storage of attribute data for each product, and storage of user specific data. Another property could be to support the classification process; where experimenting with

different combinations of features and different classification techniques should be easy. This is because the task of building a classification system involves some trial and error between different features and classification techniques. Some frameworks have been developed, one example of such a framework is 'The Agent Development Framework' developed by Athanasiadis et al. (2003). However, this framework is only limited to a Rule-based of classification algorithm.

To summarize, there are 3 questions which has to be answered before making a classification system:

1. What do we want to classify?
2. Which feature do we want to use?
3. How do we want to classify?

3.1. Types of recommender systems

A system concerned about guiding and recommending users are called a recommender system. Konstan et al. (2001) have as part of the GroupLens research project made a taxonomy which divides recommender systems into 6 categories, according to algorithms and approach used (p. 12-13):

1. Raw retrieval
2. Manually selected
3. Statistical summarization
4. Attribute-Based Recommendations
5. User-to-User (Person-to-Person) Correlation
6. Item-to-Item Correlation

Note that an individual system can be a combination of some of these suggested systems.

Which kind of recommender system we choose depends on what kind of recommendation we want the system to perform. What kind of recommender system we choose decides how the classification is done, and which features we need.

3.1.1. Raw retrieval

This is nothing more than a database of items which the user can query. This can be useful in scenarios where the user has a specific request; if a user for instance wants a whisky below a specific price, all whiskies below that price are returned by the system (Konstan et al. (2001) p. 12).

Systems like this are easy to implement and every system with a well designed database can easily adopt this ability. Konstan et al. (2001) call this a "null recommender" system, pointing out the fact that the system really does not recommend anything else then what the user has requested (p. 12).

3.1.2. Manually selected

This is also a basic system which gives sets of recommendations manually selected by editors, artists, critics or other experts (Konstan et al. (2001) p. 12). An example could be a list created by the famous whisky expert Michael Jackson presenting his ten favourite whiskies. It is important to note that all customers get the same recommendation, a so called 'non-personalised' application. Another important key issue is that the system does not need any computation at all; it is just presenting a list or some text manually compiled by an expert.

Systems/sites like this are quite common; Whisky Magazine (www.whiskymag.com) is an example of such a system, where whisky experts have comments about several hundred whiskies.

The downside with these kinds of systems is that they do not provide any personalised recommendation adapted to the particular user, and that they require human expertise.

3.1.3. Statistical summarization

This gives statistics in a non-personal manner such as lists of the customers rating of a particular product (voting system) or the lists of the most frequent bought products (Konstan et al. (2001) p. 12-13). Amazon.com customers ratings, is an example of a system like this. Another example is a Top 20 music board, where the 20 most sold albums are listed.

Statistical summarization is popular and is often used by e-commerce sites since it is easy to compute and gives valuable information. The drawback with this approach as with manually selected is that it does not support personalised recommendations.

3.1.4. Attribute-Based Recommendations

A system like this recommends a product based on its properties. For instance, if a customer wants a smoky whisky, all whiskies tasting smoky are recommended. This kind of recommender system categorizes items by their attributes.

This approach requires that we have information about the item in some machine parsable form, or attributes have to be assigned by hand when the product is added to the system (Shardanand (1994) p. 14-15). However, when the product is added, it can be recommended instantly. If we add a book and it is classified as a 'romance' then the recommender system will recommend this book for customers who are interested in romance. These systems can be more advanced where several attributes are considered and matched with the users' preferences.

It might be interesting to find which whiskies a particular user prefers. Then we are operating with two classes, the 'likes' and 'dislikes' class. More advanced systems can have user profiles for each user and match this with products.

The biggest problem with this approach is that the system needs knowledge about items which it is going to recommend. With current technology such knowledge is hard to gather in areas like sound, photographs, art, video, physical items and some multimedia by a machine (Shardanand (1994) p. 14-15). In those cases the system has to be combined with human knowledge.

3.1.5. User-to-User (Person-to-Person) Correlation

This system is more advanced than the preceding systems and is based on the assumption that there are general trends and patterns within the taste of a person and between groups of people (Shardanand (1994) p. 17). For example if a person listens to music like heavy metal then his or her preferred motor bike might be a Harley. By exploiting this assumption people can be grouped and their interests can be predicted by looking at other users' preferences within the group. The person who likes heavy metal and drives a Harley would most likely prefer a Jack Daniels as a whisky. Even though he or she has never tasted a Jack Daniels before, a user-to-user system would be able to recommend such a whisky. This is because most of the people in his or her group prefer this whisky.

This approach has been developed at MIT labs under the name "collaborative filtering", also called "social filtering" (Maes & Shardanand (1995) p. 1). It has taken many forms and the most known systems are Amazon, Firefly and Ringo. These systems are recommending things like books and music.

In the Ringo system, an early version of the more famous Firefly system, each user have to grade artists on a scale from 1 lowest to 7 highest, on how much they liked his/her music. All scores are then saved in a user profile. Each user profile is then matched with other users' profiles to decide similarity. To find similarity between two user profiles the 'mean square difference' is calculated. The lower the score the more similar the two profiles are. Users with a low score are grouped together in one user group. Users in one group are expected to prefer the same products (Shardanand (1994) p. 28 -30). The Ringo system used the Pearson *r* algorithm to calculate the mean square difference (Shardanand (1994) p. 45), but according to Breese (1998) several 'well-known' techniques to measure correlation can be used, ranging from cosine similarity calculation to Bayesian networks and nearest-neighbour method (p. 2-7).

A good thing about this approach is that it can easily be used on any kind of product (Maes & Shardanand (1995) p. 8). It can recommend products which a specific user has never seen before. It utilizes learning and improves its accuracy over time, this is valid for the whole group.

Problems with this approach are that a user has no direct impact on the system. If the user is not satisfied with the systems recommendation, the only thing he or she can do is to wait and hope that the system improves.

Similarity between users are computed on all users and the more users the heavier computing needs to be done so it does not scale well (Torres (2003)). This is unfortunate since a system like this requires a minimum amount of users to be effective. Maes & Shardanand (1995) describes this quantity as the "critical mass" for the system (p. 7-8). A typical system like this would have a "critical mass" of several thousand users (Maes & Shardanand (1995) p. 7). It is possible to use virtual users to help on the problem (Shardanand (1994) p. 76). There might be problems with new items added to the system which no users have previously reviewed before (Dai & Mobasher (2002) p. 1).

3.1.6. Item-to-Item Correlation

This system is quite similar to People-to-People correlation, but instead of looking at associations between people, we look at associations between products.

This system recommends items found frequently in association with other items. For instance, the system knows that customers who frequently buy the whisky Laphroaig often buy the whisky Talisker as well. This information can then be used by the system to recommend Talisker to a user which often buys Laphroaig, but have never bought Talisker before. A typical example of this approach is the shopping cart example, where the system can recommend products from the products already in the shopping cart.

Since we are looking for a recommender system that utilises some classification technique, the 3 first recommender systems as we have seen are not really candidates. The three later one however have embedded some kind of classification technique to it. Even though we are going to focus on one of the recommender system categories, the finished system might be a combination of any of the 6 categories. A recommender system incorporating classification techniques are able to give rational and personal recommendations.

Chapter IV

CLASSIFICATION

There are several techniques developed for machine intelligence which can be embedded in any of the three more advanced recommender systems. The technique varies in complexity from simple techniques like Association Rules and Decision Trees to more advanced techniques like Linear Discriminant Functions and Neural Networks. I am only going to briefly discuss some of the more popular techniques which could be interesting for my thesis (These are also techniques which I am familiar with). The techniques are developed in different areas such as pattern recognition, machine learning and artificial intelligence. What benefits can these different techniques give, and are there any drawbacks with any of these techniques?

4.1. Classification techniques

4.1.1. Bayesian probability

Bayesian probability is named after the British mathematician Thomas Bayes who lived from 1702 to 1761 (Wikipedia (2004)). He developed the famous formula known as Bayes' rule. This is a statistical approach which is used in AI under the name Bayesian decision theory. For an overview I am going to present Bayes' rule and the use of this theory in AI.

Bayes' rule:

$P(A \mid B) = P(B \mid A)\, P(A) / P(B)$

This rule basically makes it possible to calculate a conditional (a posterior) probability from two unconditional (a prior or marginal) probabilities and one conditional probability. From the rule we can see that P(A|B) is a conditional probability which means the probability of A given that we know B has occurred. P(B|A) then means the probability of B given that A has occurred. P(A) and P(B) stands for the probability of A and B to occur.

How can Bayes' rule be useful? It requires three terms just to compute one conditional probability (Norvig & Russell (2003) p. 480). To illustrate why Bayes' rule might be useful, I am going to show an example taken from Norvig & Russell (2003) (p. 480):

A doctor knows that the disease meningitis causes the patient to have a stiff neck, 50 % of the time. The doctor also knows that the chance for a person to get meningitis is 1/50,000 and the frequency of people with stiff neck is about 1 out of 20. By using Bayes' formula we can calculate the probability of a person having meningitis if he has stiff neck.

Probability of having meningitis: $P(A) = 1/50{,}000$.

Probability of suffering from stiff neck for a person: $P(B) = 1/20$.

Probability of having stiff neck given that the person have meningitis: $P(B \mid A) = 0.5$.

Probability of a person having meningitis given the person is suffering from stiff neck:

$P(A \mid B) = (0.5 \times 1/50000) / (1/20) = 0.0002$

Figure 4.1: Example - Bayes' rule

As we see from the above example (figure 4.1) we can calculate important information with this rule Probabilistic information is often available in the form P(effect|cause) and that is why this approach can be quite valuable (Norvig & Russell (2003) p. 481). Now let us see how Bayesian theory can be used for classification.

4.1.1.1. Bayesian decision theory

In Bayesian decision theory the following simple rule is used (Duda et al. (2001) p. 23):

Decide $\omega 1$ if $P(\omega 1 | \chi) > P(\omega 2 | \chi)$; otherwise decide $\omega 2$

χ = Known density/probability.

ω_1 = class 1

ω_2 = class 2

$P(\omega_1 | \chi)$ = Probability/density of class 1 given we know the density χ.

$P(\omega_2 | \chi)$ = Probability/density of class 2 given we know the density χ.

This expression is basically saying; select the class with highest a posterior probability. Thus, this is a good tool for AI systems that implements decision theory. Bayesian decision theory is a 'parametric' approach, which makes the assumption that the distribution of the density $P(\omega | \chi)$ is known. The distribution can be multivariate normal distribution (Gaussian) or any other distribution. Bayesian decision theory is a so called 'supervised learning' approach. This means that each sample in the training set must be labelled with a class. In other words, when training a Bayesian classifier we need to know which class each sample originates from.

Overall the Bayesian approach is popular amongst the AI developers, this is due to good accuracy if good evidence (a prior) information about the problem is available and the distribution is known or at least estimated. Another reason is that compared to other approaches, less training data is needed because of the a prior information supplied.

One problem with this approach is that it relies on a known density distribution. It can be difficult to say anything about the distribution, and estimating the distribution can prove to be a non-trivial task (Duda et al. (2001) p. 64). We do not always have a priori knowledge available about the problem we are solving, and so a Bayesian approach is useless. The most important problem however, is that each feature added gives an exponential growth in computation time. There are solutions to this problem which have resulted in a growing popularity of Bayesian classifiers in AI application (Norvig & Russell (2003) p. 482).

One approach to avoid the problem would be to always assume independent features; this approach is called Naïve Bayesian (idiot Bayes). By assuming independent features we achieve a very simple classifier (linear) compared to the one of the full Bayesian approach, (joint distribution/density) this without loosing too much accuracy if our assumption is not correct (Duda et al. (2001) p. 53). Bressan & Vitarà (2002) even claims that Naïve Bayesian outperforms several standard classifiers still when the independence assumption is not met (p.1). Another approach which is growing more popular is Bayesian networks. Bayesian networks give a more compact representation of a joint distribution. Instead of exponential growth of complexity as the full Bayesian approach gives, Bayesian networks give a linear growth of complexity. Bayesian networks can however, be complicated to construct (Norvig & Russell (2003) p. 97). I will come back to independent features later.

4.1.2. Nearest neighbour

This is a method which can be used for many of the same tasks as a Bayesian classifier and can also be used for supervised learning. But what differs is that the nearest neighbour classification does not need any assumption about the density distribution, therefore a non-parametric approach. Nearest neighbour classification actually bypasses probability estimation and goes directly to decision functions. This is good news because often it can be quite hard to assume any density distribution (Duda et al. (2001) p.161).

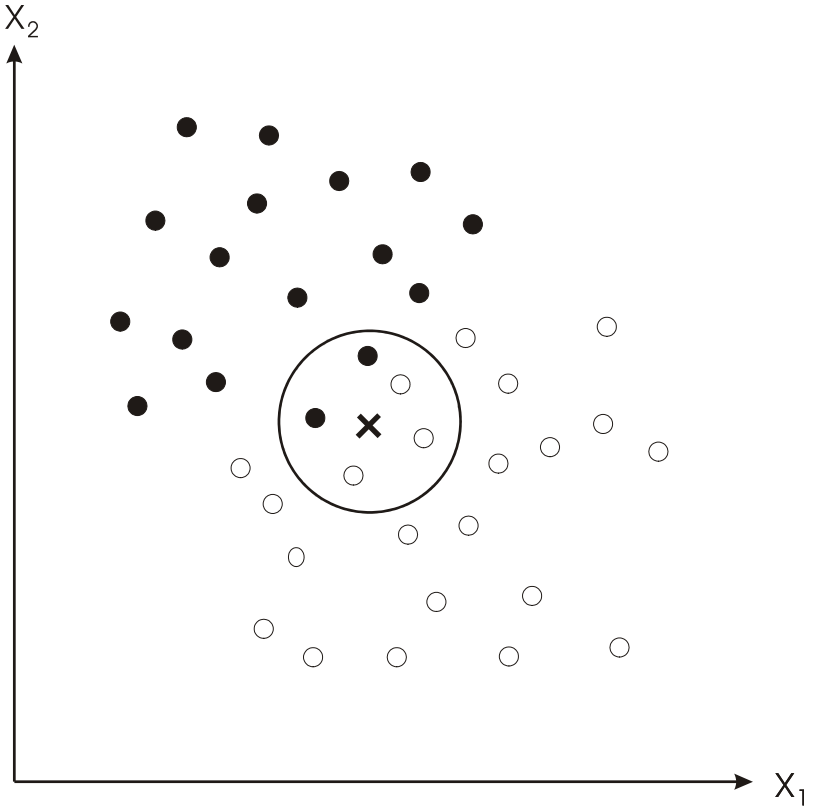


Figure 4.2: 5-nearest-neighbours rule

The figure above illustrates the principles with the k-nearest-neighbour rule. X is the unknown sample we want to classify. The black and white dots are samples respectively from the two classes ω_1 and ω_2. The classification is simply to select the class which has highest density within the sphere. The sphere indicates the 5 closest samples to X. X would in this example be classified as class ω_2, since the majority of samples in the spherical region originate from class ω_2 (white dots)

In this example (figure 4.2) the 5-nearest-neighbours rule was used. However the number of neighbours can be varied, thus the name k-nearest-neighbour rule. The more neighbours the lower error rate and higher computation costs. It is also possible to look at only one neighbour; this is called the nearest-neighbour rule. The nearest neighbour rule is often used as a first estimate of data due to an asymptotic error rate and easy implementation. It has an error rate of maximum 2 times Bayes error rate. Other techniques usually give better results (Duda et al. (2001) p.177). Instead of using a fixed

number of neighbours, a fixed window size can be used and are so called 'window methods'.

As we have seen, these methods are intuitive for a human to understand, and quite easy to implement. However, there are problems with high computational complexity both in space and time (Duda et al. (2001) p.184). Another fact is that there are other methods which have a lower error rate.

4.1.3. Cluster analysis

Sometimes we do not know anything about the classes involved or it is too costly to label the training data. We are then operating with unlabeled samples in our training set; this is often referred to as unsupervised learning (Duda et al. (2001) p.517). The task is then to find clusters in the data, where a cluster consists of samples of the same class.

There are several techniques developed for finding these clusters, some of them need to make basic assumptions about number of classes and density distribution. A more general approach is to measure similarity between samples, and then define a criterion which specifies clustering quality. Similarity measures can be common distance measures like Euclidian distance or more specialized distance measure like principal components, which is invariant to rotation of the axis (Duda et al. (2001) p.539). Criterion functions should be selected according to which clusters we expect. For instance the Sum-of-Squared-Error Criterion expects compact clouds that are well separated from one another (Duda et al. (2001) p.542). Criteria functions have proved useful to many problems, but it has a major problem; they can not detect clusters within clusters (sub-clusters)

For problems where sub-clusters are present, hierarchical clustering can be used (Duda et al. (2001) p. 552). There are many examples on problems which require sub-clusters; for instance cars are ordered in brands and models, where model is a sub-cluster of brand. Each model again might consist of subcategories based on engine size, accessories and so on.

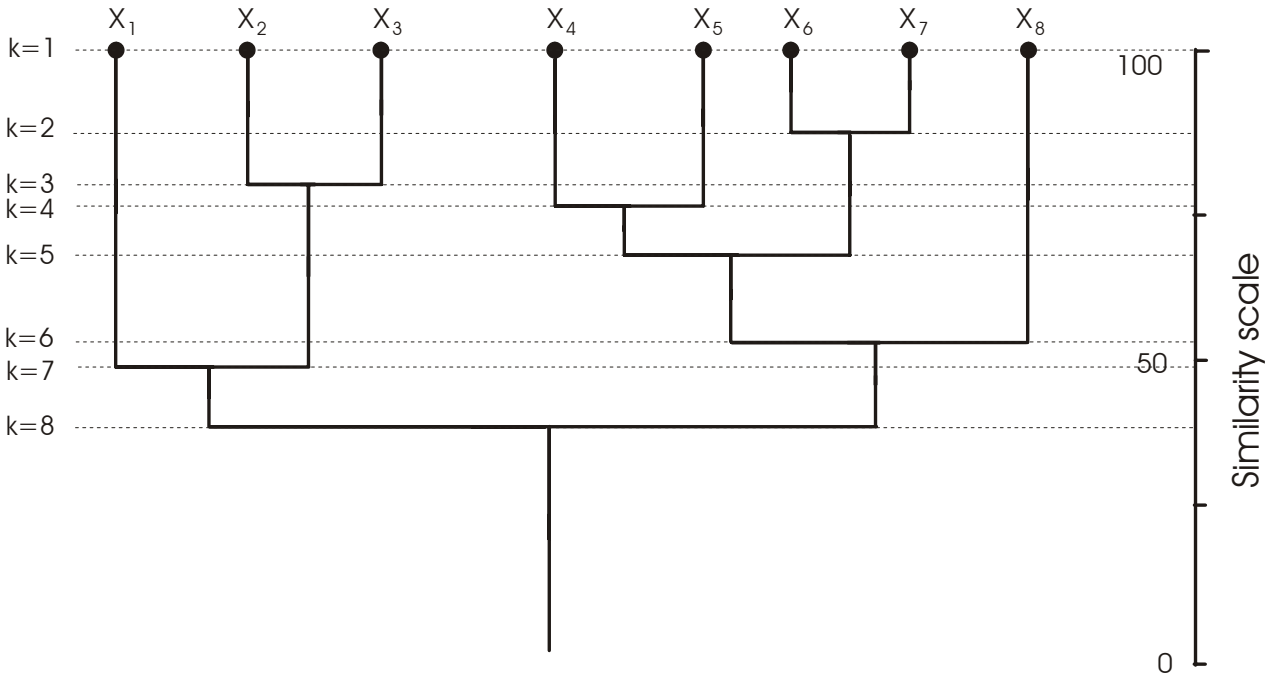


Figure 4.3: Dendrogram

Hierarchical clustering can be represented in a tree form called a dendrogram, as seen above (figure 4.3). This dendrogram shows 8 samples named X_1 to X_8. Each level k = 1 to k = 8 represents a cluster. At level k = 1 each sample is a singleton cluster and at level k = 2 X_6 and X_7 are forming a cluster. The scale to the right shows similarity between the clusters, for example the similarity of samples in the cluster formed at level k = 6 are about 50. This similarity score can be obtained with different procedures divided into Divisive (top-down) and agglomerative (bottom-up) procedures. The nearest-neighbour algorithm can be used as a bottom-up procedure for extracting similarity. Bottom-up procedures start with each sample as a singleton cluster and then iteratively merge clusters which are similar (Duda et al. (2001) p. 581). Whereas top-down procedures start with a single cluster for the samples and divides them into smaller clusters, hierarchical clustering is mostly used because of its simplicity and ability to tackle sub-clusters.

4.1.4. Self-organising maps

Sometimes the high dimensionality of the data that are being classified can cause problems, since they are hard to visualize and analyse. One solution to this problem is

called low dimensional representation. Low dimensional representation is really a method for mapping multidimensional data onto a two-dimensional space and not a method for finding clusters. It is important for such approach to preserve neighbourhoods and relative distances when projected into a lower dimensional room. There are several approaches for doing this projection as accurate as possible and one popular approach is Self organizing feature maps developed by Kohonen (1984). Self-organizing feature maps generate a topologically correct map of the data, preserving neighbourhoods. The mapping is done by a neural network, where similar data are mapped onto the same node or to a neighbouring node in the map. This arrangement of the clusters on the map reflects the attribute relationships of the clusters in the input space (Koua (2003) p. 1).

To further illustrate what self-organizing feature maps can be used for, we can look at a map made by Chen (1995) and his colleagues which describe the content of Yahoo! in 1995.

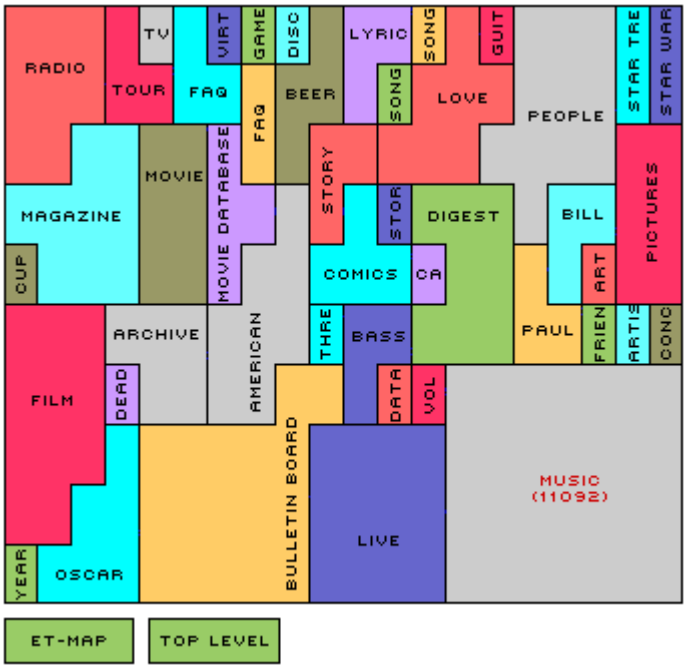


Figure 4.4: Yahoo self-organizing feature map (Chen (1995))

As we see from the figure above (figure 4.4) we get a 2-dimentional 'birds-eye' view of the Yahoo! site, which in reality contains multi-dimensional data. Self-organizing feature maps are a valuable tool for visualising multi-dimensional data into a human understandable form.

4.1.5. Case Based Reasoning

CBR *(Case-Based Reasoning)* originates from cognitive psychology theories and especially from Schank (1982) and his students work on dynamic memory at Yale University in the early 1980s.

CBR tries to mimic a human's ability to recognise similar situations. For instance if a car does not start a car mechanic would know that for this particular car type it is more likely that the battery is causing the fault than that the spark plugs are to blame. Of course the more cars and situations the car mechanic has experienced, the faster and more accurate his reasoning becomes. CBR is based on the belief that new problems are often similar to previous problems. Therefore in most cases an old solution could be applied to the new problem. (Aitken (2004))

All CBR methods have the following process in common (Harrison (1997)):

- Retrieve the most similar case (or cases) comparing the case to the library of past cases.
- Reuse the retrieved case to try to solve the current problem.
- Revise and adapt the proposed solution if necessary.
- Retain the final solution as part of a new case.

Some of the characteristics of a domain that indicate that a CBR approach might be suitable include (Harrison (1997)):

- Records of previously solved problems exist.
- Historical cases are viewed as an asset which ought to be preserved.
- Remembering previous experiences is useful.
- Specialists talk about their domain by giving examples.
- Experience is at least as valuable as textbook knowledge.

A CBR engine usually consists of a data retrieval algorithm and a matching algorithm. The matching algorithm could be as simple as a string matching algorithm. Because of the simplicity of the algorithms used and the versatility of the CBR approach it has grown in popularity over the last years (Aamodt & Plaza (1994)).

4.2. Some classification systems examples

Some attempts have been made to make classification systems for whisky. I am going to present three different systems made. For each system, I am going to look at what kind of classification technique they use and which features they need for the classification.

4.2.1. Whisky Classified

When building a classification system, it is important to find features that are independent, and it must be possible to give a value which is scalable. This is described more in depth in chapter 6. Wishart (2002) has in his classification project "Whisky Classified" divided the taste of a whisky into 11 different features as seen in figure 4.5. Additional information is available at www.whiskyclassified.com.

Feature	← 1	2	3	4	5 →
Body	Light				Heavy
Sweetness	Dry				Sweet
Smoky	Non-Peaty				Peaty
Medicinal	Salty				Non-Salty
Tobacco	Tea				Feinty
Honey	Non-Vanilla				Vanilla
Winey	Non-Woody				Woody
Nutty	Non-Oaky				Oaky
Malty	Non-Cerealy				Cerealy
Fruity	Non-Estery				Estery
Floral	Non-Herbal				Herbal

1.Not Present, **2**.Slight Hint, **3**.Medium Note, **4**.Definite Note, **5**.Pronounced feature.

Figure 4.5: Wishart features (Wishart(2002))

Each feature is graded from 1 to 5. The feature Body, for example can have values ranging from 1 to 5 where 1 means light and 5 means heavy.

The model for representing the taste was developed in cooperation with several whisky experts and has been approved by several distilleries. 86 whiskies were specified by the new taste model developed, and these data were later analysed using a cluster algorithm. From this analysis 10 clusters were found and whiskies in the same cluster were said to have similar taste. A recommendation is based on recommending whiskies in the same cluster/class as the whisky the user likes.

4.2.2. www.celticmalts.com

Kraaijeveld (2001), has also used a cluster algorithm when he was trying to figure out how ancient Celtic whiskies tasted. By looking at production methods from around 1880 he tried to classify whiskies in clusters of similarity and give each cluster a taste description. It would have been quite time-consuming to gather information about which production

methods each distillery used, but Kraaijeveld was lucky to have detailed descriptions about each distillery around 1880. Features like grains used, kiln, kiln fuel, mash tun and wash back contents, still type and contents, number of distillations and warehouse capacity were documented.

The reason for Kraaijeveld to use this technique rather than tasting notes, was that tasting notes had not yet been invented at that time. This approach also gives other benefits; classification can be made objective, with less interference by human's subjectivity. It is also possible to predict how a whisky would taste even before it is made.

4.2.3. Classification of Pure Malt Whiskies

Lapointe and Legendre (1994) used a different approach in their classification project. Instead of using a cluster algorithm the use a Naïve Bayes (TDF-IDF) classifier which is usually used for analysing documents. This approach treats every word which can be used to describe a whisky as a feature. This literally means that we can deal with several hundreds or even thousands of features.

Features could be aromatic, peaty, sweet, light, fresh, dry, fruity, grassy, salty, sherry, spicy, rich. Each whisky is given a score on how many of the features (words) that is used to describe the whisky.

Whisky	**Word**									
	Smoky	Peaty	Sweet	Light	Fresh	Fruity	Dry	Grassy	Salty	Sherry
Macallan		X				X				X
Chivas Regal	X					X		X		X

Figure 4.6: Example - Naïve Bayes (TDF-IDF)

From figure 4.6 we can see that Chivas Regal would receive a higher score than the Macallan. Each of the features is given a weight after how frequently it appears. Whiskies with equal score are said to be similar.

Lapointe and Legendre have not received the same support from the 'whisky world' as Wishart's classification.

Chapter V

AGENT PLATFORM

We have so far looked at general AI techniques and different technology developed by the AmbieSense project. How can the different technology support duty-free shopping? In this chapter the final system combining all of these technologies is presented. First let us get a clearer view on what the system are set to do. To illustrate the tasks of the system we can have a look at the Oslo Airport test mentioned earlier in chapter 1. For testing the technology, the AmbieSense project has made a test scenario at Oslo Airport (OSL) Gardermoen.

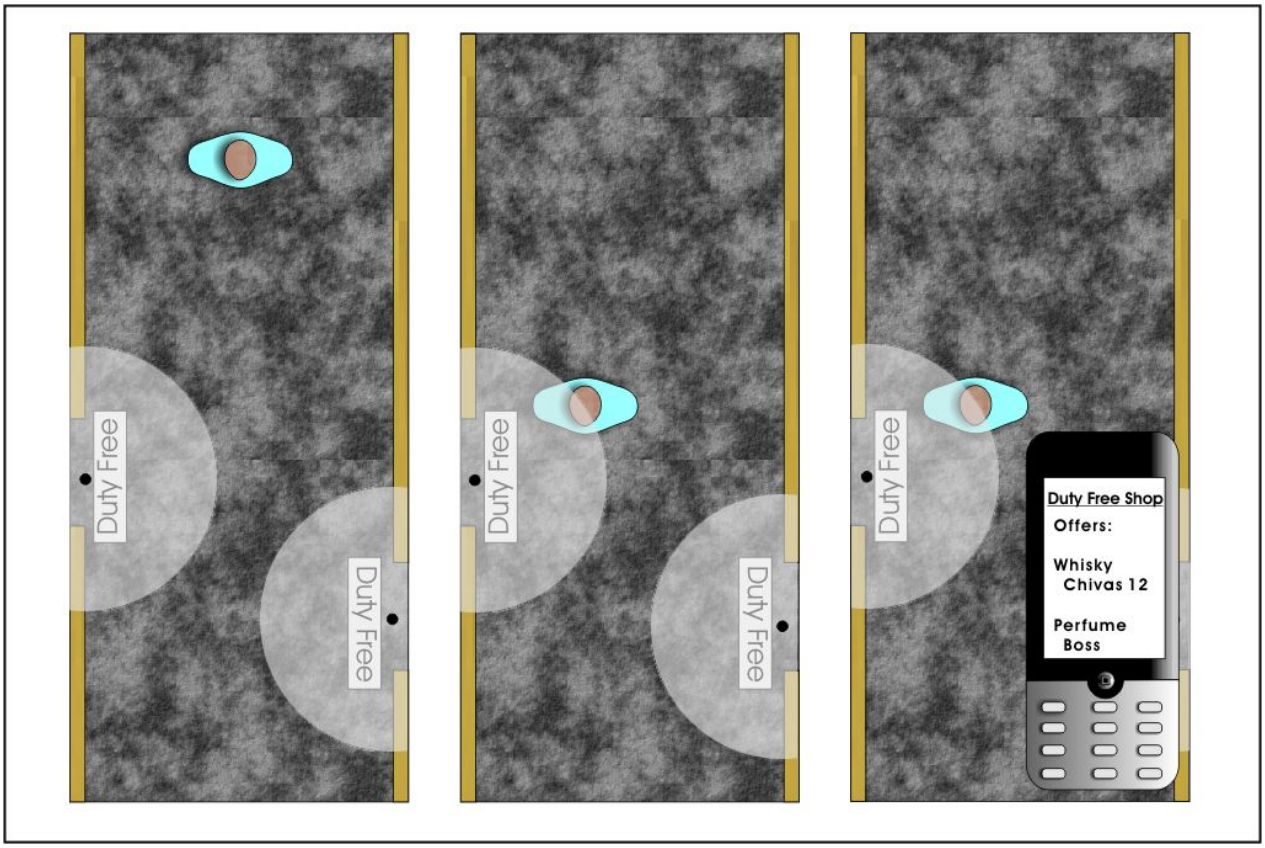


Figure 5.1: Oslo Airport (OSL) Gardermoen scenario

Figure 5.1 shows how AmbieSense technology might be applied to support duty-free shopping at OSL airport. Here we see a traveller passing by a duty-free shop fitted with a Context Tag.

Information about the current location is sent from the Context Tag and the traveller's context is updated. Information sent from the Context Tag could contain information about map coordinates, duty-free shops, check-in, travel agencies, cafés and so on. The new information is then matched with the recorded interests of the traveller. Only if the information offered match the traveller's profile and the system recognises the situation as suited for this kind of information is the information passed on to the user. This can be seen on the last picture in the sequence of figure 5.1, where two offers appear on the user's/traveller's mobile phone; one for a 12 year old Chivas whisky and another for a Boss perfume. Several factors are evaluated before the system presents these offers to the user such as interest in whisky, time before take-off, money available and what kind of travel (business or pleasure). If, however, the traveller's favourite whisky is out of stock or if he wants to try a new whisky, the system can recommend a whisky based on what he usually drinks.

5.1. AmbieSense MAS

The AmbieSense project decided to use an MAS because it provides the flexibility, scalability and extensibility which are needed by such a system. More about the motivation for using an MAS can be found in the documentation of the AmbieSense project, especially in D8 chapter 2.2 (Myrhaug et al. (2004)). In particular, JADE *(Java Agent Development Framework)* was chosen as a MAS framework.

5.1.1. JADE

JADE is one of many frameworks for building an MAS; a few others include Aglet, FIPA-OS, Odyssey, Voyager and Zeus. Jade is a MAS framework based on FIPA *(Foundation for Intelligent Physical Agents)* specifications (Bellifemine et al. (2003)). FIPA is a European organization for standardization of MAS (FIPA (2003)). Like most MAS framework JADE gives benefits like built in support for communication and predefined agent behaviours. In addition, JADE also supports mobile platforms through LEAP *(Lightweight Efficient Application Protocols)*.

5.1.1.1. The JADE platform

The JADE platform consists of two agents which are always running; the AMS *(Agent Management System)* agent and the DF *(Directory Facilitator)* agent. The AMS agent administers the lifecycle of the other agents with tasks such as registering, deregistering, relocation and stopping of agents. When agents register they are given a unique number called the AID *(Agent Identifier)* (Bellifemine et al. (2004)). The DF Agent works like the yellow pages, where an agent can request a service, let us say plumber, the DF Agent would search for all plumber agents and return their AID.

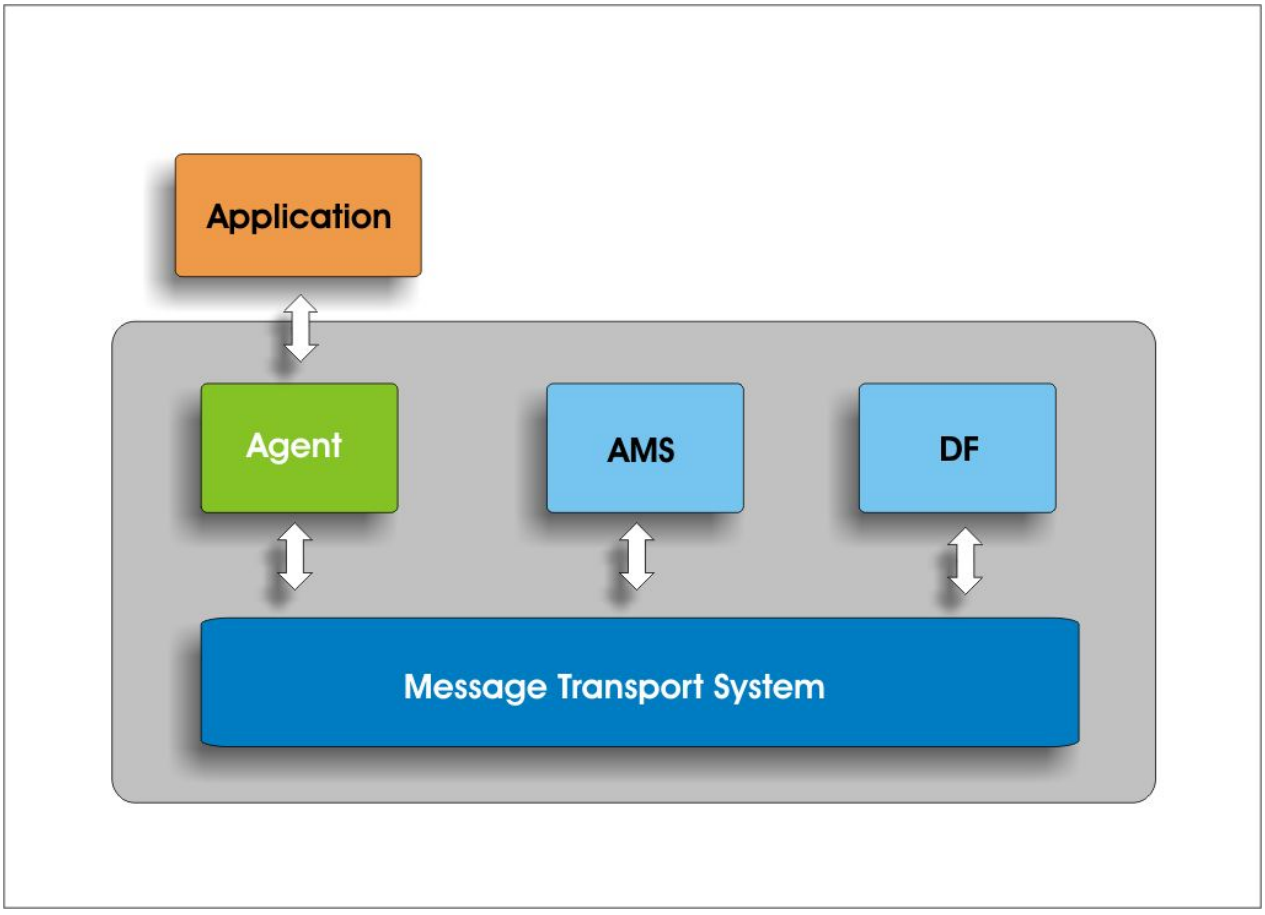


Figure 5.2: The JADE Agent Platform (Belligemine et al. Figure 1 p.7)

Message Transport System also known as ACC *(Agent Communication Channel)*, are software components controlling exchange of messages between agents, internally and from other platforms.

The agent platform can be distributed across different kinds of machines which not even need to share the same operating system. Only one Java application runs on each host machine, and each agent runs in a tread. This allows a JVM *(Java Virtual Machine)* to be shared between several agents. JADE also allows agents to move from one machine to another during run-time. When a developer develops an agent he or she does not have to consider where the agent is going to be run. The JADE platform provides containers as a habitat for agents to run in. In theory a container could support an infinite number of agents running simultaneously. However, in reality the host machine limits the number of agents able to run. Agents can also move between different containers, while running.

5.1.2. LEAP (Lightweight Efficient Application Protocols)

The JADE platform is developed for running on PCs and servers and is not suited for mobile devices for several reasons (Caire (2003) p.3):

- Big memory footprint (several Mbytes)
- Requires JDK 1.4 or later, which is not normally supported by mobile devices
- Requires a fixed network

To overcome these problems, JADE is used in conjunction with LEAP. LEAP is a set of protocols specially adapted for low bandwidth communication, ideal for wireless and mobile applications (Banan (2000)).

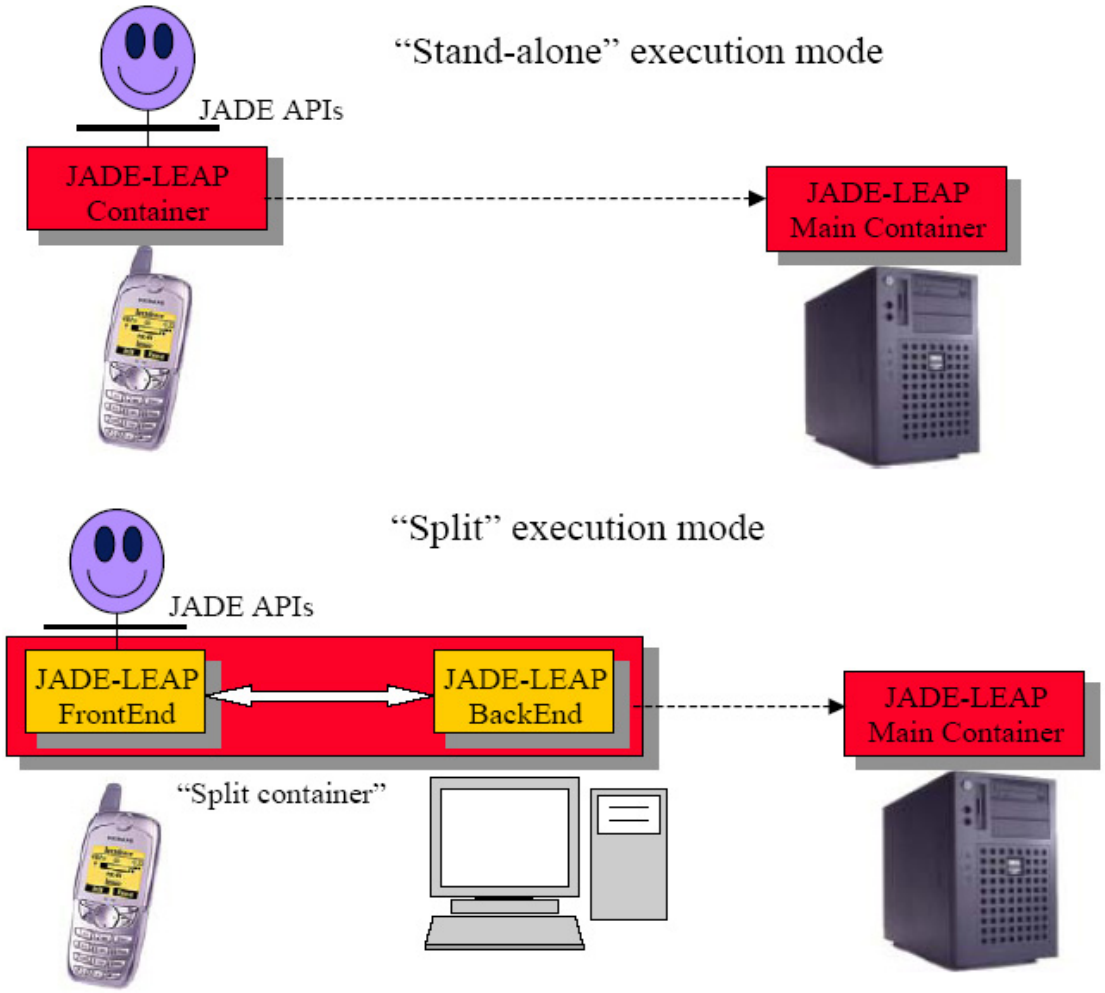


Figure 5.3 LEAP split execution (Caire (2003) p.7)

LEAP gives the possibility to run the agent on a split container, which means that an agent can be split into a front-end running on the mobile device and a backend running on a server, as seen on figure 5.3. This allows heavy application to be started from a mobile device, but the actual computation is done on a server. In the JADE framework, every agent with a GUI *(Graphical User Interface)* is programmed with the possibility of being run on a split container (Bellifemine et al. (2003) p.15). This means that for the programmer creating an agent, running on a split container requires the same amount of work as creating a non split agent.

```
public class PrefsAgent
    extends GuiAgent {

      //Agent content
}
```

Figure 5.4: Example - agent with GUI

Programming wisely, an agent with a GUI has to extend the GuiAgent class. Figure 5.4 illustrates how this is done in Java.

5.1.3. Reason for choosing JADE-LEAP

The reason for choosing JADE as the MAS framework is illustrated in figure 5.5 which is made from table 3 in appendix 1 of D8 in the AmbieSense documentation (Myrhaug et al. (2004)).

Framework	AR1	AR2	AR3	AR4	AR5	AR6
	Mobile Platform	Open platform	FIPA	Programmable	Open Source	Representation language
Tryllian	yes	yes	yes	yes	no	yes
April	no	no	yes	yes	yes	yes
Comtec	no	yes	yes	yes	yes	yes
FIPA-OS	yes	yes	yes	yes	yes	yes
Grasshopper	yes	yes	yes	yes	no	yes
JACK	no	yes	yes	yes	no	yes
JADE-LEAP	yes	yes	yes	yes	yes	yes
JAS	yes	yes	yes	no	yes	no
ZEUS	n/a	yes	yes	yes	yes	yes

Figure 5.5 MAS frameworks based on table 3 Appendix 1 D8 (Myrhaug et al. (2004))

From the figure, we can see that only FIPA-OS and JADE-LEAP fulfil the requirement AR1-AR6 set by the AmbieSense project. From these two candidates, JADE was selected due to the following reasons (Myrhaug et al. (2004)):

- There has been former positive experience with the JADE platform in the consortium.

- LEAP was developed in another IST project. Using this framework in AmbieSense contributes to a consistent way of utilising EU-funded research.

- With the release of JADE version 3 (19.03.2003), the JADE developer team manages LEAP; LEAP is now integrated in JADE as an add-on, thus ensuring closer integration of these frameworks as well as support and further development.

- JADE-LEAP has a very active developer community, thus ensuring support and further development even after the project's end.

5.1.4. The AmbieSense Architecture

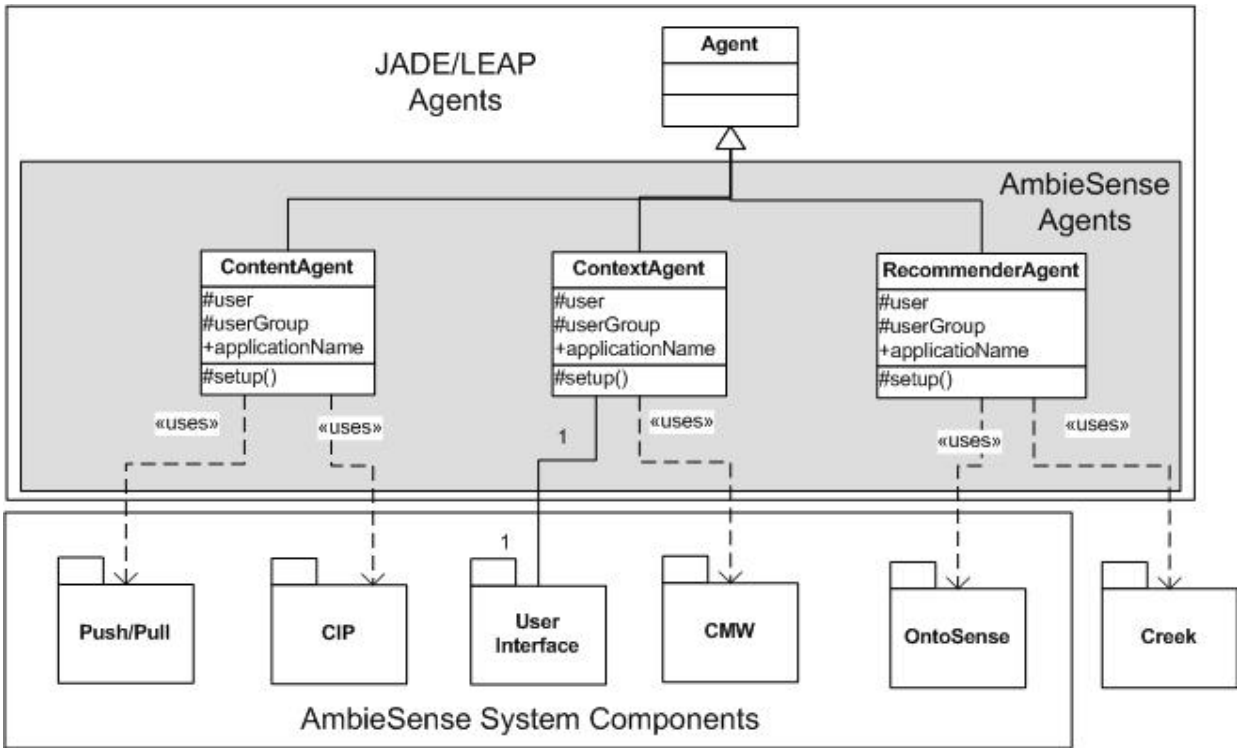


Figure 5.6: AmbieSense architecture, UML class diagram

Figure 5.6 was taken from the AmbieSense documentation D9 page 8 (Wienhofen et al. (2004)). As we can see from the figure 5.6, the AmbieSense system consists of 3 main agents; a Context Agent, a Recommender Agent and a Content Agent.

5.1.4.1. The Context Agent

The Context Agent is responsible for storing and retrieving contexts from the context middleware.

According to the AmbieSense project (Myrhaug et al. (2004)):

"A context describes aspects of a situation seen from a particular actor's point of view. An actor can in the widest sense for instance be a person, thing, matter, or organism. In this way a context is actually defined as something separate from the situation itself. A context in AmbieSense is a representation inside the computer. It represents aspects of a situation in the real world. (…) AmbieSense chose to implement this structure by developing Java-classes integrated with Java technology."

The OSL airport scenario described earlier in this chapter uses two kinds of contexts; one user context and one tag context. The user context stores information about the user, whereas the tag context is a context describing a Context Tag. When a traveller/user passes a Context Tag, the tag context is merged with the user context. Basically this means that the user's context is updated with local information received from the tag context. A user's context is grouped into five sub-categories:

- Social Context
- Task Context
- Personal Context
- Environment Context
- Spatio-Temporal Context

This context hierarchy model was first described by Myrhaug (2001).

Each category can themselves hold sub-categories or attributes.

Social Context - Contains information such as friends, neighbours, roles and relatives.

Task Context – Describes what the user is doing. This can be expressed as goals, tasks, actions, activities, or events.

Personal Context – Divided into two sub-categories; the physiological context and the mental context. The physiological context holds information such as pulse, blood pressure, weight, fingerprint and more. The mental context contains information such as mood, stress, expertise, interest etc.

Environment Context – Captures entities surrounding the user like things, services, temperature, light, humidity, noise and persons.

Spatio-Temporal Context - Holds information about time and space. Information that might be stored could be location, time, heading and other information.

A tag context is similar to the user context, but lacks the personal context category.

```
<?xml version="1.0" encoding="UTF-8"?>
<contextTemplate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <contextTemplateName>user context template</contextTemplateName>
   <contextTemplate>
      <contextTemplateName>spatio-temporal context</contextTemplateName>
      <attribute>
         <attributeName>current time</attributeName>
         <string>noon</string>
      </attribute>
      <attribute>
         <attributeName>current location</attributeName>
         <string>CHELSEA</string>
      </attribute>
   </contextTemplate>
</contextTemplate>
```

Figure 5.7: XML representation of context in AmbieSense

For further illustration of how a context is represented in the AmbieSense project figure 5.7 shows an example of a user context represented in xml. Contexts are stored as Java objects, but can be translated into xml. Here we see a user context with information about current time and current location. This information is stored as two attributes, each with a name and a value. All kinds of information about the user can be represented in this way. We can recognise the context hierarchy where the user context is the root category, and spatio-temporal is a child or sub-category.

This is just a brief description of the context hierarchy, it is described in more detail in Chapter 5 of D2 in the AmbieSense documentation (Myrhaug et al. (2004)). The context hierarchy represents the common basis of knowledge between the agents in the AmbieSense MAS, and it creates the foundations of the OntoSense ontology. The OntoSense ontology is used by the agents in the AmbieSense MAS when they are communicating, to ensure that they share the same understanding of concepts. New concepts and attributes can be added to the AmbieSense ontology as long as they reside inside one of the five main categories (Myrhaug et al. (2004)). For example, if we want to add information about the user's mood, a mood attribute could be added under the personal-mental category.

Each user context is stored in the context middleware. The context middleware acts as a database which store, search, merge and retrieve context information for each user of the system. It also keeps a history of previous contexts, which can be restored if needed. This is particularly useful for recognizing previously encountered situations.

The context middleware also holds the responsibility for the security of the contexts. One mechanism ensuring security is a single access point to the context middleware. All access to the context middleware is done through the Context Agent. The Context Agent validates users by a user name, a password and a X.509 certificate, before it can access the context middleware. The information sent between the context middleware and the Context Agent is encrypted, so is the information residing in the context middleware. SSL *(Secure Socket Layer)* is used to secure the network connections (Myrhaug et al. (2004)).

5.1.4.2. The Recommender Agent

Based on the user context, the Recommender Agent can predict which information the user wants. For this task, it is equipped with Creek; a CBR engine. The theory behind CBR has been introduced earlier in chapter 4. Other techniques could be employed which better fits the information recorded. An example where it is important to give right information would be in a situation whereby the plane leaves in 10 minutes. In such situation, the traveller should get a reminder on his/her flight instead of information about duty-free shopping. The system should be able to recognize the situation from earlier cases (contexts) where the time to departure are 10 minutes or less, and act according to these previously recorded cases (contexts). Previously recorded cases (contexts) are either learnt by earlier experience or learnt by training.

5.1.4.3. The Content Agent

The Content Agent's task is to keep track on valid content providers depending on the situation. The Content Agent receives an update message from the Context Agent each time a context change occurs. Based on the context and which information the Recommender Agent recommends, the Content Agent finds content providers suited for the requested information. For example, if duty-free shopping is requested the Content Agent search for available duty-free shops nearby. The information is retrieved from the shopping web site of OSL.

5.2. Duty-free (whisky) shopping MAS

The AmbieSense project focuses on making a general framework, whereas my thesis is giving content to the system, and demonstrating how it can be used for the specific scenario of duty-free shopping. Before the actual solution is presented, let us look at the initial requirements that were set.

5.2.1. Requirements

The system must consist of at least two agents; one GUI/User agent that communicates with the user and one expert agent which recommends products to the user. There are two actors to consider in this system; the user/traveller and the duty-free shop.

The GUI/User agent must provide the following:

- Ask user about products he or she might be interested in.
- Store a profile containing products the user is interested in.

The expert agent must provide the following:

- Recommend relevant products to the user.
- Classify new products based on some defined properties.

The user must be able to:

- Register which products he or she is interested in and specify the price range.
- Accept an offer.
- Easily shut off information about offers.
- Access the system on their mobile device without having to install a lot of programs.

5.2.2. Duty-free (whisky) shopping - MAS architecture

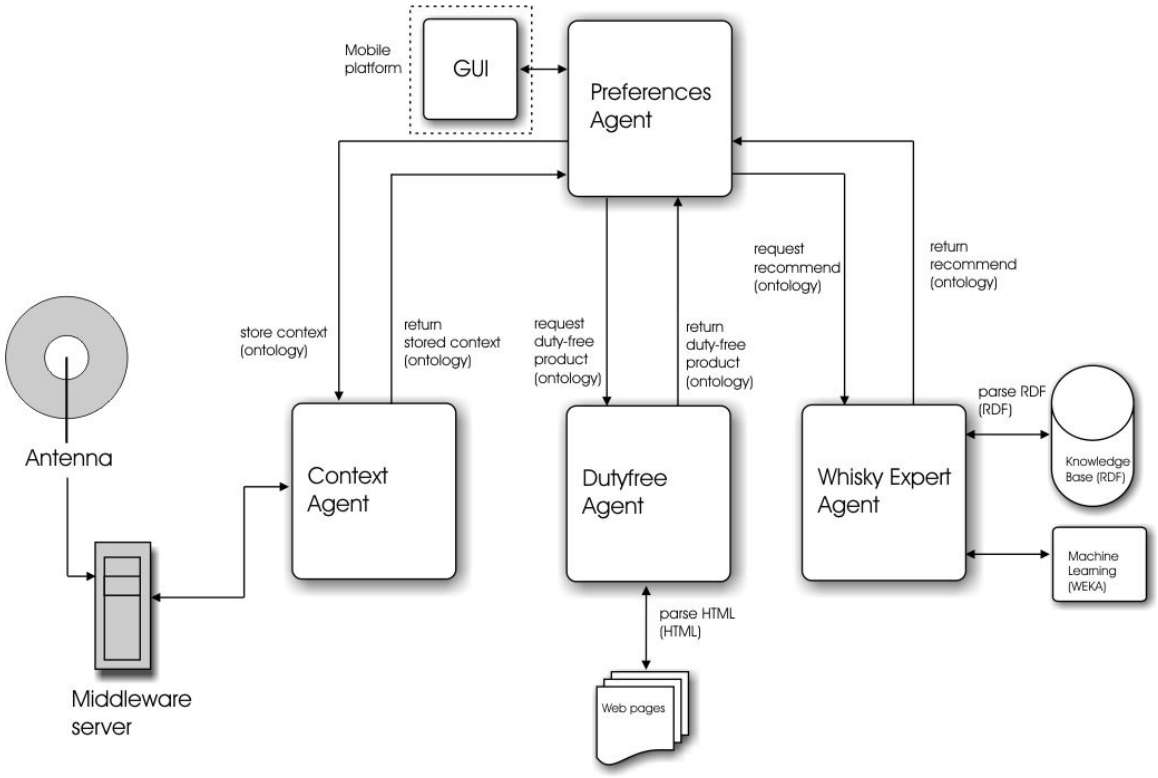


Figure 5.8: Duty-free MAS, conceptual model

This is a conceptual model of the final MAS architecture provided as an overview. A more detailed UML can be found in Appendix A. We can recognize the Context Agent from the AmbieSense framework; the other agents (Recommender Agent and Content Agent) are also present in my application as we will see when each agent is presented.

5.2.2.1. Preferences Agent

The Preferences Agent is the traveller's personal agent. This agent runs solely or partially on the traveller's mobile device. It is responsible for taking the user's preferences and informs the users about things he/she is interested in. The Preferences Agent has to cooperate with other agents to achieve this, and works as a coordinator or a switchboard

between them. The Preferences Agent is implemented as a split agent with a GUI, this allows it to run on mobile devices.

Responsibilities:

- Build GUI dynamically from duty-free ontology and information gathered from the Whisky Expert Agent.
- Receive preferences from user, and pass them to the Context Agent.

5.2.2.2. Whisky Expert Agent

In many ways this agent is similar to the Recommender Agent, but instead of recommending the next move for the system, it recommends a whisky to the user. For the duty-free shopping domain there would be one expert agent for each production such as a cognac expert, perfume expert and so on. The Whisky Expert Agent can be compared to a human whisky expert where his responsibilities are to:

- Give information about a whisky on request.
- Recommend a whisky on request, based on some criteria.
- Classify new whiskies into categories of similar whiskies.
- Be able to extend its knowledge about whiskies.

Instead of communicating with other humans, the expert agent must communicate with other agents. The whisky expert must have expert knowledge just like its human counterpart and be able to extract a recommendation from this knowledge. To be able to do this, the Whisky Expert Agent consists of a knowledge base and a classification system. The knowledge base is used to store all kinds of information about whiskies and the classification system is used for classifying new whiskies, which can later be recommended to a user (agent). Since the Whisky Expert Agent is supposed to make expert decisions it important that the accuracy of the recommendations have a certain standard. A realistic measure would be a accuracy within the range of 10% or less mistakes.

5.2.2.2.1. Knowledge Base

Noy & McGuinness (2001) defines a knowledge base as "an ontology together with a set of individual classes constitutes an knowledge base" (p.3). Also, according to Norvig & Russell (2003), a KB *(Knowledge Base)* is a collection of sentences written in a knowledge representation language and represents some assertion about the world. The knowledge base stores the knowledge the agent has. The knowledge base can be used to store knowledge acquired or to access knowledge stored. A knowledge base also has the ability to infer new knowledge from the knowledge already stored (p.195).

With these definitions in mind, you could think of a knowledge base as a kind of database. Instead of an ER-model defining the structure, an ontology defines the structure. Where a database stores and retrieves data, a knowledge base stores and retrieves knowledge. In addition a knowledge base has the ability to infer knowledge from knowledge already stored.

This way of looking on a knowledge base may work in many situations. However, the distinction between the ontology and the knowledge base is not always clear, and sometimes it can be hard to establish a clear-cut border between them (Noy & McGuinness (2001) p.3).

5.2.2.2.2. Protégé

Protégé is a tool made for constructing domain ontologies. Nevertheless, it may also serve as a standalone KB for storing and retrieving knowledge (protégé.standford.edu). When used as a KB, it is possible to use the supplied GUI for entering data and make queries. It has also the possibility for applications to access the knowledge base using the supplied Java API. A KB constructed in Protégé can easily be exported into a knowledge representation language such as RDF, OWL, DAML or into a Java ontology. It is even possible to export the KB into a normal database format such as JDBC.

For my thesis, I used Protégé as a construction tool for the KB. The KB was later exported into RDF/RDFS. RDF/RDFS was used because I wanted an open platform which could easily be accessed by any program using an RDF parser. The ontology used by the JADE agents were also extracted from the same design using a plug-in called the BeanGenerator (Acklin (2004)). The BeanGenerator creates Java code compliant with the JADE ontology specification.

5.2.2.2.3. Weka

The classification system was built using Weka. Weka is a collection of classification algorithms implemented in Java. (Frank et al. (2000)) Weka allows the classification algorithms to be applied in two ways; from a GUI where different kinds of visualisation and experimenting can be done, or directly from a java program through an API. This gives the system developer the opportunity of experimenting with different learning algorithms using the GUI until a suited one is found, and later implement it into the agent using the Java API. In addition, Weka supports different kind of pre-processing, association rules, feature validation and more. Basically, Weka supports the whole classification process as we will see in chapter 6.

5.2.2.3. Context Agent

This is the exact same agent as described earlier in the overview of the AmbieSense architecture. The context used is extended to contain whisky information and this is further explained later in this chapter.

5.2.2.4. Duty-free Agent

The Duty-free Agent could either be the Content Agent described in the AmbieSense framework or a sub-agent only handling duty-free content. In my scenario, it does not matter if it is the main Content Agent or just a content provider for duty-free shopping. Either the Content Agent knows every domain from check-in to duty-free shopping or it only knows which agent it can delegate the different tasks to. This really depends on the size of the total domain.

Responsibilities:

- Search the Internet for relevant duty-free items available on the current location.
- Retrieve prices on requested duty-free items.

5.3. Communication

JADE provides an extensive support for agent communication. Figure 5.9 shows the different layers of communication in JADE.

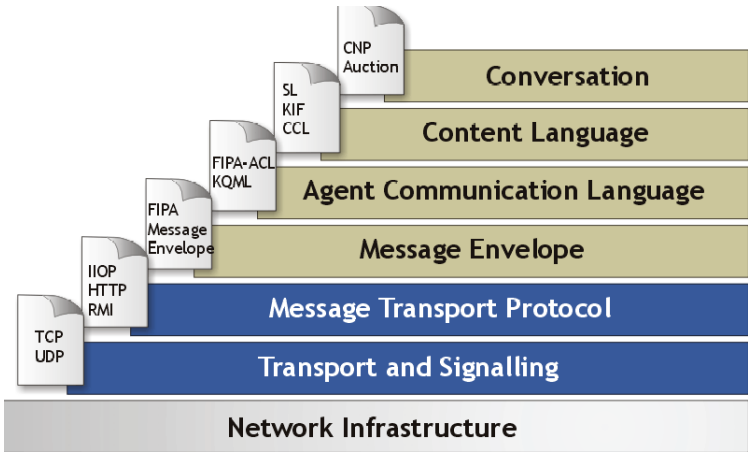


Figure 5.9: JADE/FIPA communication model (Helin (February 2003)))

The 4 layers from the top are administered by JADE (golden), and the two following lower layers are administered by LEAP (blue). I am not going to go through the whole model, only highlight layers which are of interest.

The MTP *(Message Transport Protocol)* layer in JADE are divided into; intra platform communication (Within one JADE platform), and inter platform communication (Between different JADE platforms). Jade uses the RMI *(Remote Message Invocation)* for intra platform communication and HTTP *(Hypertext Transport Protocol)* or IIOP *(Internet Inter-ORB protocol)* for communication between agent platforms. The inter platform communication enables agents to communicate across different machines and locations. IIOP was used in earlier versions of JADE, but from version 3.2 HTTP is used as the standard protocol. The reason for selecting HTTP as the standard protocol is because it provides shorter and easier addresses instead of the awkward CORBA *(Common Object Request Broker Architecture)* address used by IIOP (Grimshaw (July 2004)). HTTP is also simpler to implement and is because of that it better suited for small devices (Helin (2003) p.1). Earlier performance problems with HTTP have been solved by using bit-efficient Agent Communication Languages. Even this is not enough to beat IIOP when it comes to transferring a number of bytes (Helin (2003) p.3).

The Agent Communication Language layer clearly contains the ACL *(Agent Communication Language)*. The ACL is, as you might recall from chapter 2, one of three key elements for agent communication. The other two are a common content language and a shared ontology. JADE has support for the commonly used KQML and FIPA-ACL which is FIPA's alternative to KQML.

The Content Language layer describes which kind of content is being sent, such as formulas or plain text. Several languages are supported; for example FIPA-SL *(Subset Language)*, FIPA–RDF0 (FIPA version of RDF), FIPA-CCL (FIPA version of Content Choice Language), KIF *(Knowledge Interchange Format)* and the FIPA-KIF (FIPA version of KIF). Ontologies can easily bind into any of these languages. More about FIPA's content languages can be found at the FIPA web page (FIPA (1999)).

For the project, FIPA-ACL was chosen as the agent communication language because it is the default language in JADE and most of the JADE documentation uses this language. There has been a controversy between the two languages arguing which is the better language. FIPA-ACL comes with formal semantics which gives a standard interpretation, KQML on the other hand does not provide formalised semantics, but KQML has the benefit of being more widely used (Huns & Singh (1997)). However, for my project it is important that the agents speak the same language, and not which language they speak. As a Content Language, FIPA-SL is used because it is the standard content language in JADE, but also because it is more efficient than for example FIPA-RDF. The ontology can be described as java classes when used by FIPA-SL.

5.3.1.1. Ontology

The ontology used is designed in Protégé and then exported into java as explained earlier. Here is an UML showing the structure of the Ontology:

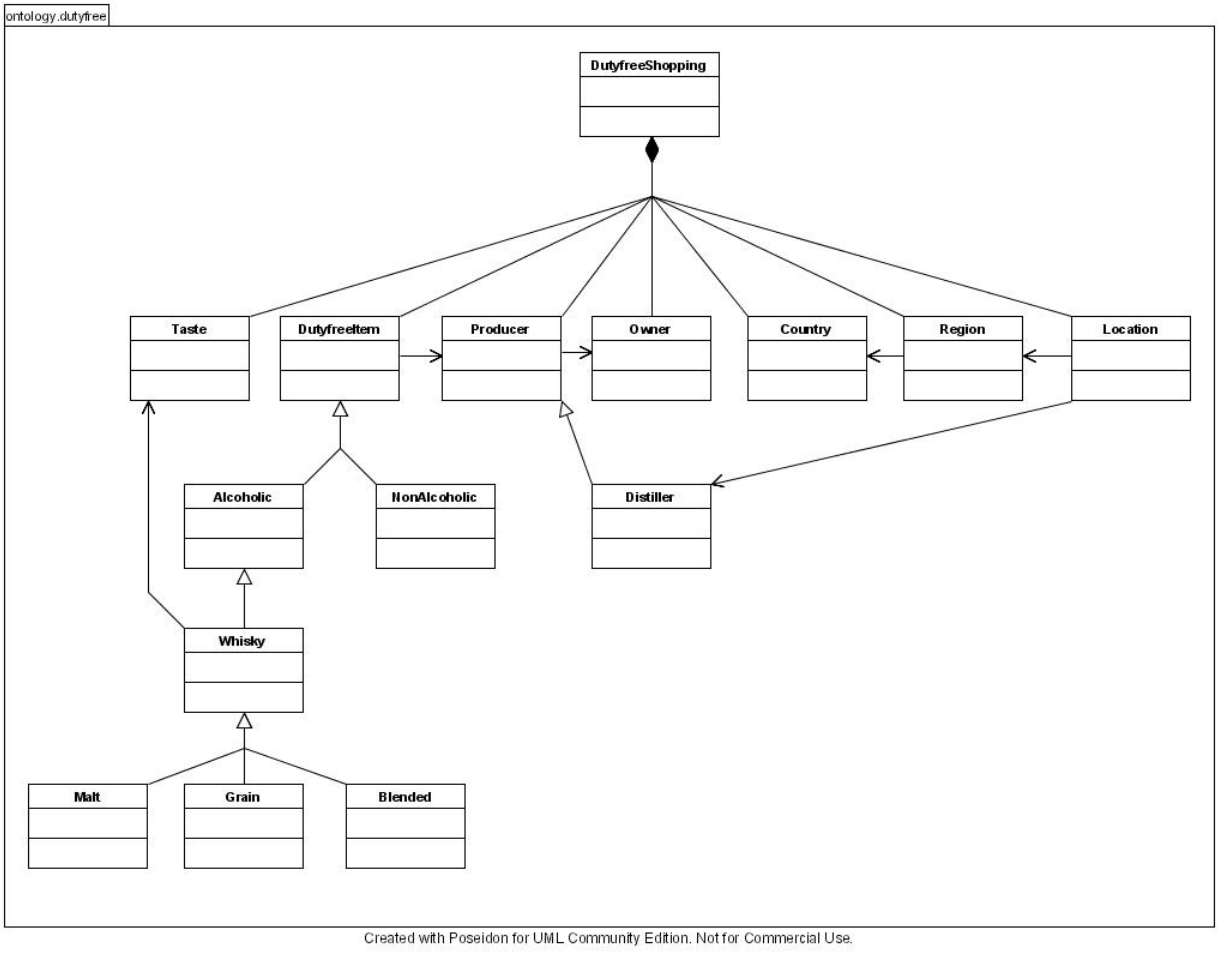


Figure 5.10: Duty-free Shopping Ontology, UML class diagram

The ontology tries to model any object in the duty-free shopping domain. It is constructed in such a way that it can easily be extended to contain other products in the future. If we want to add a new product such as the drink; cognac, it could be added to extend the class; Alcoholic. Not only are drinks able to be added, other products like perfume or chocolate could be added, thus extending the DutyfreeItem class. Nevertheless, for my scenario whiskies are the only product which is modelled. We can also see that Whisky extends Alcoholic and that it could be; Malt, Grain or Blended. A whisky also has a taste which is modelled as an own class.

Since the ontology is based on the same design as the KB, it contained the same structure and fields as the KB. There are both benefits and drawbacks with this approach. The benefits are that both the KB and ontology can be maintained in the same design and that

everything in the KB is guarantied to be understood by the ontology. The drawback is that differences which actually exist between the KB and the ontology are not captured. For example, the price property is not part of the knowledge base because the prices are gathered from the duty-free shop. Agents need to know and communicate about price, but it is not stored in the knowledge base. Even though we store price in the KB, it would not mean the same thing because prices from several different shops can not be represented.

The Ontology was added to the context middleware by extending the context hierarchy as shown on figure 5.11.

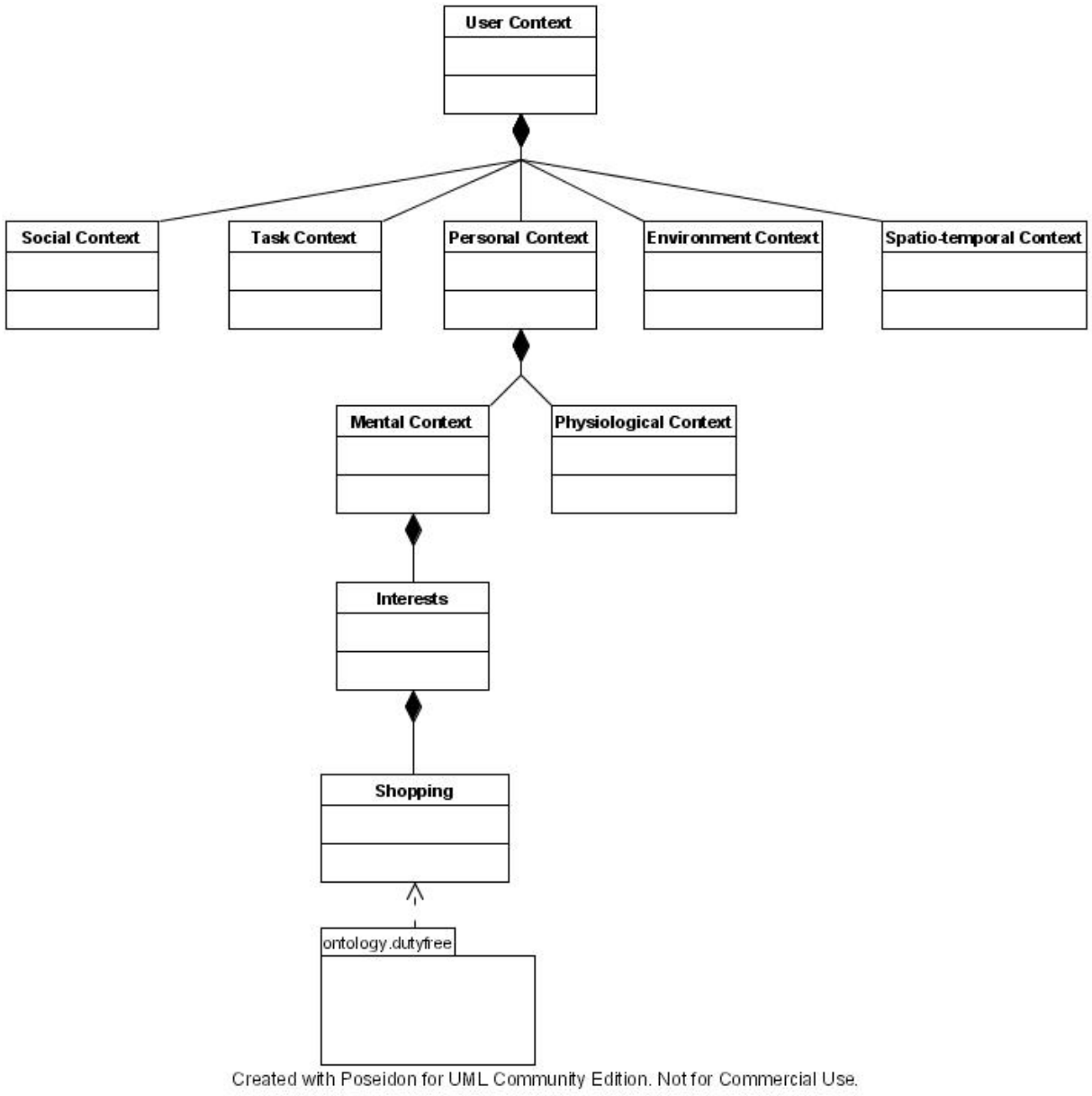


Figure 5.11: Context hierarchy, UML class diagram

From the figure, we can see that a new category Interest is added under the mental context. Interest describes the user's interests such as sports, food, music and so on. Shopping is one of such interests and we can see that Shopping are added as a sub-category of interests. Shopping actually contains the whole duty-free shopping ontology described in figure 5.11.

5.4. Recommendation

The recommendation process is not only handled by the Whisky Expert Agent, but is also a result of several agents working together. Let us look at the initial scenario where a traveller passes a duty-free shop, but this time we look at how the agents interact.

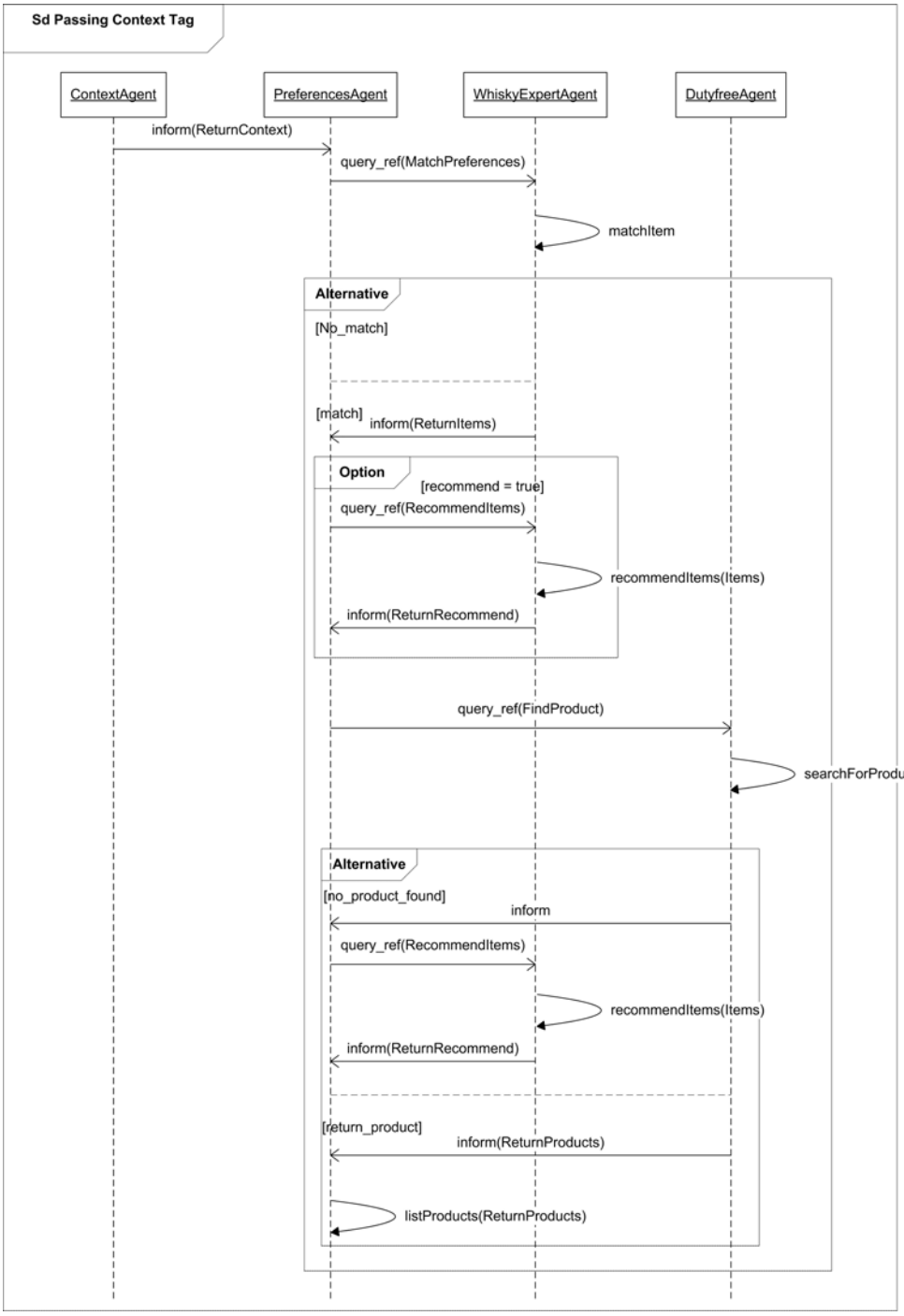


Figure 5.12: User passing Context Tag, AUML sequence diagram

Since standard UML (UML 1.0) is unable to express the complexity of agent communication and behaviours, I have used AUML *(Agent UML)* instead (www.auml.org). AUML is not yet an official standard and is currently under development.

The figure presented here is not complete, but is meant as an illustration of the communication taking place between the agents when a traveller passes a Context Tag. Only agents are shown in the diagram and components like the Context Tag and the context middleware are left out.

Figure 5.12 describes the scenario where a traveller passes a Context Tag. The scenario is initiated by the Context tag which sends information to the context middleware.

The Context Agent informs the Preference Agent that duty-free shopping is wanted by sending an updated context containing information about the location and information on the wanted shopping items.

The Preferences Agent sends the user's whisky preference to the whisky expert for matching. If the Whisky Expert Agent finds matching whiskies, they are returned to the Preferences Agent. The returned matched whiskies can be seen as the users preferred whiskies. The preferred whiskies are either sent back to the whisky expert used as the basis for a recommendation or they are sent directly to the Duty-free Agent. This depends on if the user wants a recommendation or not. If the user wants a recommendation then the whisky expert will recommend some whiskies and they are added to the already matched whiskies and are sent back to the Preference Agent. The Preference Agent then sends the matched and recommended whiskies to the Duty-free Agent which is searching for these products in the nearby shops. If the matching and recommended products are found, the prices are added and they are sent back to the Preferences Agent. The Preferences Agent lists the found products in a sorted way by relevance. As we can see, the communication is rather complex and difficult to even describe with words. Nevertheless it is important to notice the different stages which lead to the final recommendation.

5.4.1. Matching

For the system to perform any recommendation, the user has to give some input about his or her whisky preferences. The user can either directly choose one or more whiskies he or she prefers, or enter some properties that the preferred whisky should possess.

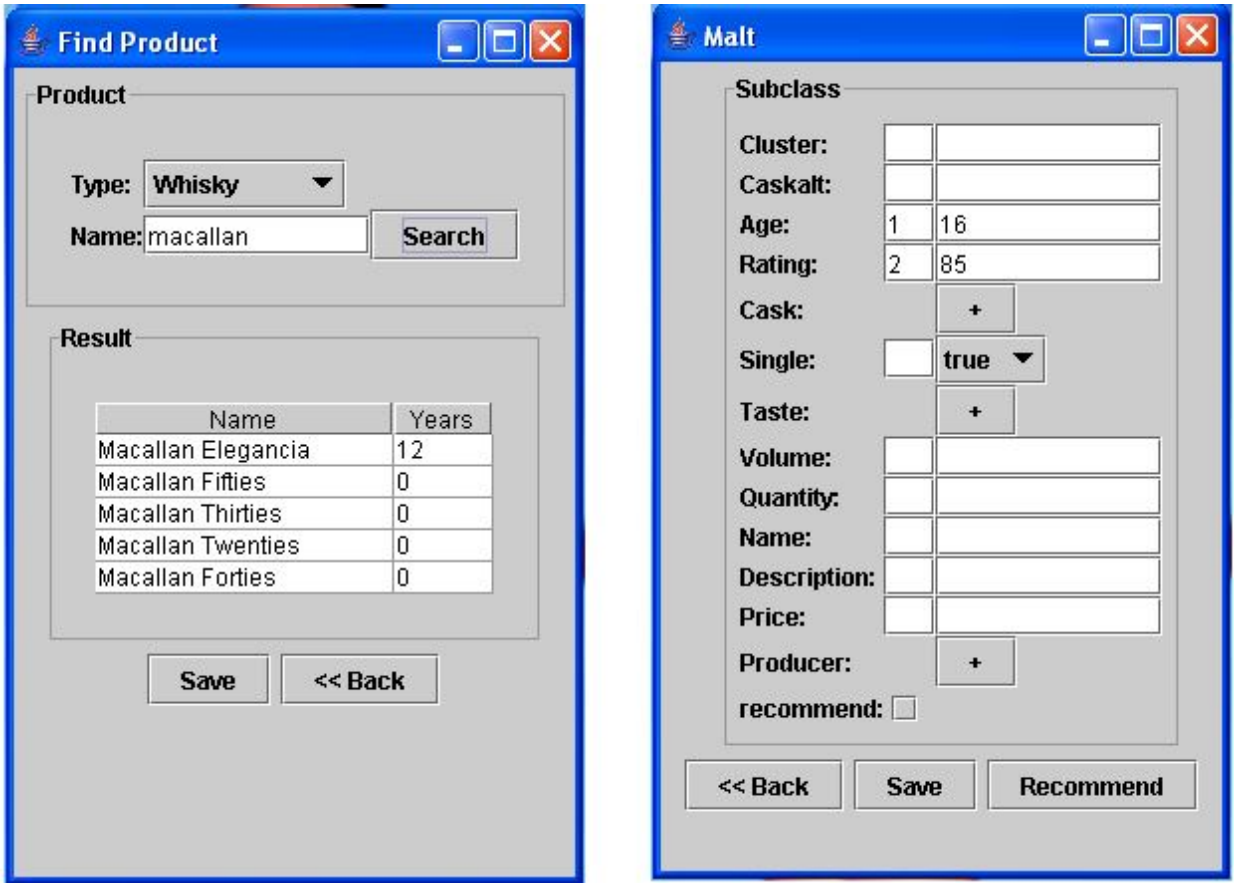


Figure 5.13: Selecting preferred product (left), Specify properties (right)

Figure 5.13 (left) shows how a user can search for a specific product by entering the name and the type of the product. From figure 5.13 (right), we can see the menu where the user can fill in values for preferred properties such as age, price and producer found in the second column. Each property can be ranked from 1 – 10 in the first column found in the menu, whereby 1 is the most preferred property and 10 is the least preferred. But he or she does not however, need to fill up all the boxes in the column. Instead, figures inserted will only reflect on the user's choice. For example, if he or she only has age, cask and taste in mind, then he / she will only need to fill up 1 – 3 in the first column.

The system tries to match these properties to whiskies already known by the system which is contained in the KB. This matching technique can be sophisticated, but in my system I used a simple algorithm for matching (string matching). From this matching, hopefully one or more whiskies are found and a list will be created.

5.4.2. Classification

The whiskies preferred by the user or matched by the system are used as the basis for the recommendation. It is important to notice that it is only when a new whisky is entered into the system that the classification algorithm is used to classify the whisky into one of ten categories. When the system recommends a whisky to the user, it only searches for other whiskies classified into the same class as the whiskies the user prefers. More about the classification algorithm can be found in chapter 6.

5.4.3. Ranking

The recommended whiskies and the preferred whiskies are ranked before they are presented to the user. The ranking is simple and it arranges the cheapest whisky first, and whiskies with the same price are then ranked by their score given by a whisky expert. This ranking could be improved by calculating a score combining the price and score given by a whisky expert, for finding the whisky with best value for money. The user could also be able to enter his or her own ranking.

```
Preferred products:


Matched products:
Macallan Thirties    219  95
Macallan Twenties    219  95
Macallan Forties    219  94
Macallan Fifties    219  93


Recommended products:
Dalmore  12  229  79
Glendronach   249  79
```

Figure 5.14: Output from recommender system

We can see that the output is arranged into three different categories; Preferred, Matched and Recommended products. The preferred products reflect products which the user has specifically requested. The matched products refer to products the system has found based on the properties the user specified. Finally, the recommended products are the products which the system recommends. We can also see that under each category the whiskies are ranked by price (first number) and by score (last number).

5.5. Testing

The testing of the system was done as four separate tests; testing of the mobile platform, testing of multi agent platform (recommendation), testing of AmbieSense framework and testing of classification. Together these tests covered most aspects of the final working system.

5.5.1. AmbieSense framework

I did not participate in the testing of the AmbieSense framework. Nevertheless I am going to give a brief description of the testing. More about the testing can be found in the AmbieSense documentation D9 chapter 2.4 and 3 (Myrhaug et al. (2004)).

The testing of the AmbieSense project consisted of 3 steps:

- Component testing
- Integration testing
- Complete system testing

5.5.1.1. Component testing

The AmbieSense framework consists of different separate components like Context Tags, context middleware and the agent system. Each of these components was tested separately to ensure that they complied with the requirements. For some of the components, test applications have been developed and used for testing.

5.5.1.2. Integration testing

Integration testing was performed to see how certain components worked together. For instance, the context middleware and the Context Agent was developed respectively by SINTEF and CognIT, and had to be tested to see that they worked together in an integration test.

5.5.1.3. Complete system testing

For a complete test of the AmbieSense system, it was tested on OSL airport, described earlier as the OSL scenario. Two test runs had been done on site in May and August 2004. The testing included both a test of the functionality and a user acceptance testing.

The OSL scenario focuses on presenting three types of content to the user/traveller; dining information, shopping information and service information such as flight information. Context Tags were placed in different locations such as: check-in area, security check, post security check, Explorer Bar, Thune, Gate 39, Gate 43 and four tags in the non-domestic departure area.

The functionality testing included tests of:

- Running of the agent platform with the different protocols; IIOP and HTTP
- Testing two PDAs (Compaq iPAC 3870) with WLAN and Bluetooth
- Running agents in three different setups; distributed, partially distributed and locally.
- Response time and uptime.

The user acceptance testing consisted of:

- Users using the system
- Interviews while and after the users were using the system (Use and interview lasted for about 25 – 40 minutes)

5.5.2. Multi Agent System (Recommendation)

The testing was also here divided into 3 different stages:

- Component testing

- Integration testing
- Complete systems testing

5.5.2.1. Component testing

The component test involved testing of each agent separately to see that they worked as planned. Initially, I started with only dummy agents (not to be mistaken with dummy agents created in the JADE administration GUI) which gradually were developed to fully functional agents. Also, components integrated into agents such as the classifier and the knowledge base were tested separately.

5.5.2.2. Integration testing

The integration testing was not only conducted between finished agents, but also with dummy agents that had a hard coded behaviour. The communication between agents was monitored by a Sniffer Agent. The Sniffer Agent creates a sequence diagram which can be displayed graphically, of the communication taking place between the respective agents. The scenario described in figure 5.12 was used as a test scenario when testing the agents. The reason for using figure 5.12 as a test scenario was that it is on of the most complicated scenarios involving all the agents, other simpler scenarios should work if this one does. However, any communication tested between two agents could be seen as an integration test.

5.5.2.3. Complete system testing

Most of the system was built as described earlier in this chapter, but some parts were partially or fully simulated. Another important restriction was that the MAS were not running on any mobile devices. However, agents were running on different JADE platforms and physical machines.

Since I did not have access to every component in the whole AmbieSense project, and I did not have a Context Tag, some parts had to be simulated. The context agent was only

storing contexts locally and not in the context middleware. The Context Tag/antenna was simulated as an agent, which gave a location every 10 seconds. The Duty-free Agent did not search the internet as intended, but instead it searched a text file which where a compilation of the Euroshop web page, see Appendix B.

The testing itself was divided into two different categories; testing of recommendation and testing of the MAS platform.

The testing of the recommendation was basically to see if the system produced a recommendation as expected. I will explain more about this in the discussion of the findings.

Testing of the MAS platform was performed to see if the system benefited from it. Tests here included running of agents on different JADE platforms and servers, adding and replacing agents at runtime.

5.5.3. Mobile platform

If the whisky recommender system is to be used by travellers it has to be available on their mobile device, be it a notebook, a PDA or a mobile phone. To ensure that JADE-LEAP could work sufficiently in such a scenario, agents were tested on some available mobile devices. Emulators from Nokia and Sony Ericsson were also used during the testing. Instead of testing the Preferences Agent, a simple Demo application consisting of two simple agents were used. Instead of using my own agent which might contain errors that could influence the final result, the Demo application was chosen as it had already been tested and run before on mobile devices. JADE-LEAP can be run on three different Java configurations:

J2SE *(Java Standard Edition)* is 'normal' Java which is run on servers and PC in a fixed network.

Personal Java is the 'old' standard for running Java on mobile device. Personal Java is currently listed as "end of life" by Sun Microsystems (2004a), which means that it is not supported by Sun anymore and is replaced by newer standards such as MIDP 2.0 and also by J2SE which some newer PDAs support.

MIDP 1.0 *(Mobile Information Device Profile)* is today's runtime environment for mobile devices such as mobile phones and PDAs. MIDP contains a reduced set of the J2SE library and only provides a minimum of functionality required by mobile application (Sun Microsystems (2002)). MIDP 1.0 is gradually replaced by MIDP 2.0 which adds additional functionality for handling graphics and communication (Sun Microsystems (2004b)). For this project the requirements of socket support is the most important new feature with MIDP 2.0 as I will explain later. Most new phones support MIDP 2.0. JADE-LEAP can run on both MIDP 1.0 and MIDP 2.0 devices.

MIDP was chosen for running JADE-LEAP on mobile devices, since Personal Java is no longer supported by Sun. The testing was performed on a PDA (HP Jornada 540) and on several mobile phones.

Before the Demo application could be run on a mobile device, it had to be compiled into a JADE-LEAP MIDP agent. For compiling agents into JADE-LEAP the following had to be done:

- JADE 3.1 was installed on the computer the agent was compiled on, thereafter LEAP was copied into an add-on folder located in the JADE root-folder
- The build tool Ant had to be installed
- The J2ME wireless toolkit had to be installed

For compilation of the Demo application the build tool Ant was used. Ant gives the possibility of compiling an application into all the different Java configurations (MIDP, Personal Java and J2SE) in one operation. However, before this was possible every compiler had to be specified in the buildLEAP.properties file. The compilation creates a .jad and a .jar file. The .jad file contains various information such as the size of the .jar file, the path to the .jar file and the functions required for the .jar file to run. The .jar file is the compiled and packed version of the application.

After the application was compiled, it was ready to be tested on a mobile device. However, instead of copying the files to each mobile device, the files were put on a web server, so they were easily accessible for downloading.

Both the stand-alone and the split mode described earlier in this chapter were tested when running the agent on the mobile devices. On devices only supporting MIDP, only split

mode is recommended due to limitations in memory and/or processing capabilities (Caire (2003) p.7).

5.5.4. Classification

The ability of the classifier was tested, a full description of how this was done are described in chapter 6.

Chapter VI

FINDINGS CLASSIFICATION

In this chapter we are going to see which considerations have to be taken when designing a classification system and how we can measure the performance of the classification. In the end of the chapter the findings for the selected classifier are presented.

6.1. Designing a classification system

The design of a classification system usually contains different stages:

- Data collection
- Feature choice
- Model choice
- Training
- Evaluation

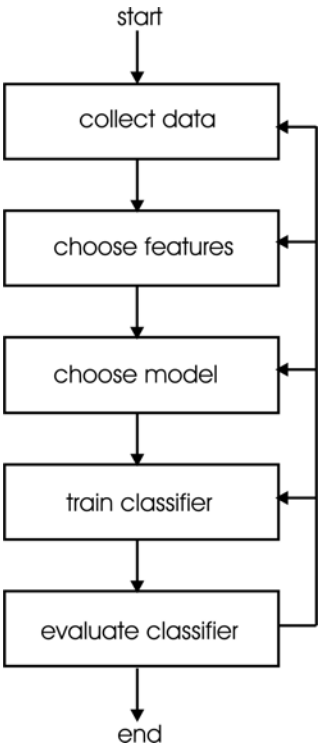


Figure 6.1: Design cycle classifier (Duda et al. (2001) p. 14 figure 1.8)

As seen from the model, the design process is an iterative process where it is possible to go back to any of the previous steps until the result is satisfactory.

6.2. Data Collection

Collecting data usually accounts for a large part of the cost in developing a pattern classification system. It is important to collect as many 'typical' examples of data which may occur in the 'real' world as possible. Usually a minimum of 30 samples per class is needed for a proper classification. As example data, I used the same 86 whiskies that Wishart used in his classification project. All these whiskies are Scottish single malt whiskies. The reason for choosing the same whiskies as he did is because I could then use his results as labels for my training data.

Since my goal is not to find new measuring techniques or secrets of whisky production, I have based my system on easily available data from books and the internet. In cases where data are conflicting, more sources were used to cross validate the data.

Media	Name	Author	Publisher	Information
Book	The Single Malt Whisky Companion : A Connoiseur's Guide	Arthur, Helen	Macmillan, 1997	General information
Book	Whisky fra hele verden	Laurin, Urban	Landbruksforlaget, 1998	General information
Book	Maltwhisky : håndbok	Jackson, Michael	Gyldendal, 2000	Rating and general information
Book	Whisky classified	Wishart, David	Pavilion Books Limited, 2002	General information and class labels.
Internet	"Ecosse: Whisky et Distilleries" www.whisky-distilleries.info			Distillery information
Internet	"Whisky magazine" www.whiskymag.com			General information
Internet	"Whisky distilleries, producers and distributors" www.awa.dk/whisky/stills/			Distillery, producers and distributors information
Internet	"Scotchwhisky.net" www.scotchwhisky.net			Distillery information
Internet	"The Edinburgh Malt Whisky Tour" www.dcs.ed.ac.uk/home/jhb/whisky/scotland.html			Location data

Figure 6.2: Sources used for collecting data

The data collected are incorporated into a knowledge base which is trying to reflect domain knowledge through relations between the data.

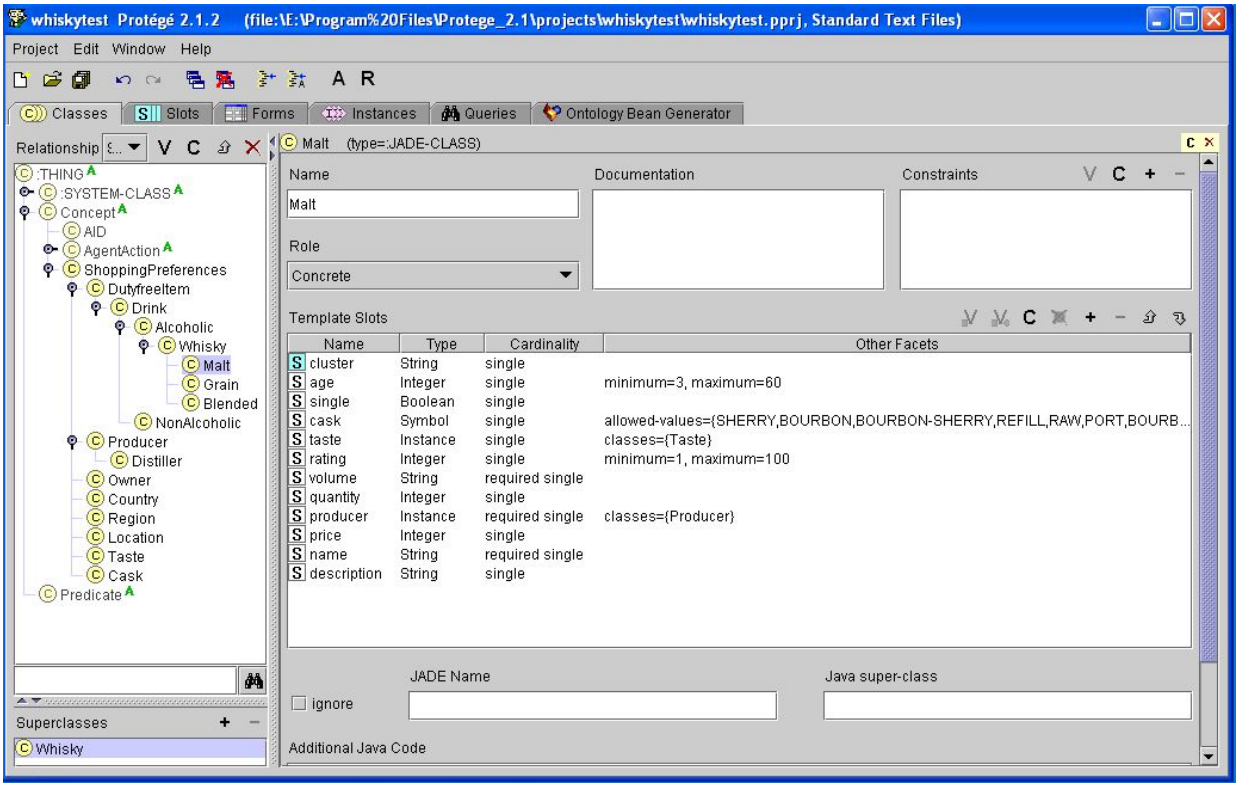


Figure 6.3: Structure of final Knowledge base

6.3. Selecting features

After collecting data for the system, it is important to find features from the data which can be used by the classifier. Features are also known as attributes. A good feature is a measurement which is similar to objects in the same class and clearly different to those that is not. In the case of separating balls from boxes, a good feature would be the shape, because all balls have a round shape (except an American football), which is different from the square shape of a box. For other scenarios, other features would be better soothed like light intensity, length or width of the item. The better the feature we choose, the easier the job of the classifier becomes (Duda et al. (2001) p. 11). This is why it is important to find good features. Before we select features, it is important to be aware of the fact that there are different kinds of features; each handled differently. We should also know which abilities a good feature should hold.

There are two main categories of features namely discrete features and continuous features.

A feature is discrete if its set of possible values is a collection of isolated points on the number line. However, a feature is continuous if the set of values form an entire interval on the number line (Devore & Peck (1990) p. 5). This means that a discrete feature is a variable which can take a limited set of values in an interval, whereas a continuous feature can take on any value in an interval. An example of a discrete feature could be a person's hair colour or a person's age. Whereby the hair colour is black, brown, blonde or other colour nuances, and the age is 16 or 65 and so on. Discrete instances often have the same value. A continuous feature could be the body temperature measured for each person. For example, if we measure two persons' body temperature with a three digit thermometer we could measure the same value 37.6, but with an increased resolution of four digits we might measure the different values of 37.62 and 37.69. We see that continuous values can be measured discretely but have an underlying nature which makes them different from a discrete value.

Why is it important to differentiate between continuous and discrete features? In statistics we operate with probabilities for discrete values, but for continuous values it is not possible to calculate because the probability is close to zero (Duda et al. p.618). Instead, we have to look at continuous values as intervals and calculate densities instead of probabilities. Different algorithms and techniques used in the classification process can either work on discrete or continuous values, and some can work on both.

When we search for features it is also desirable if the features we find are independent. Independent features are features that are not correlated with each other, and two independent features A and B can be expressed as following (Norvig & Russell (2003) p.478):

$P(A \mid B) = P(A)$ *or* $P(B \mid A) = P(B)$ *or* $P(A \wedge B) = P(A)\,P(B)$

The reason for selecting independent features is that it leads to a simple (linear) classifier (Duda et al. (2001) p. 53).

For selecting features, domain knowledge can help us to find the best suited features. However, there are some mathematical models that are developed which can be used for testing the quality of the found features. Mathematical models can also be used to reduce the number of features, which shorten the calculation time. The area of mathematically based feature selectors is currently under a lot of research, and it is beyond the scope of

this thesis. Nevertheless, I am going to give a brief introduction and then focus on the techniques used in my research instead. There are two main categories of mathematical feature selectors; filters and wrappers. Filters are evaluating features independent of the learning algorithm itself, whereas wrappers typically use the targeted learning algorithm to calculate the accuracy of features (Hall (1999) p. 2).

Even though filters and wrappers are supplied by Weka, I will apply solely filter-based techniques in my thesis as they are easier to understand and use.

Hence, I am going to use the most common filter technique called information gain, and a technique called ReliefF; also supported by Weka (Kononenko & Robnik-Sikonja (1997) p.1).

According to Norvig and Russell (2003), information gain is "the difference between the original information requirements and the old information requirement" (p.660). Information gain can be expressed by the formula (Norvig, P. & Russell, S. (2003) p.660):

$Gain(A) = I(p/p+n, n/p+n) - Remainder(A)$

By looking at the formula you might think that information gain is something which only concern mathematics. But most people have a built-in ability to perform some form of information gain evaluation when taking decisions. To illustrate this, think of the game *20 questions*. If you are not familiar with the game, I will explain how it works. The game is played by two persons. Person A is thinking of an answer and the other person B is trying to guess what the answer is. Person B initiates the game by stating the domain in form of a question. Person A then tries to guess the answer. The only feedback person B is allowed to give is 'yes' or 'no'. Now back to the example: You are given the problem: "I am thinking of a number between 1 and 1000". To answer this problem, there are several questions that could be asked such as "Is the number a prime?", "Is it 467" or "Is it between 1 and 500". However, most people would first ask "Is it between 1 and 500" because this question would classify most data into its right class.

If we calculate the information gain using the formula, we would get the same result. I am not showing the calculations here, but Greiner and Schaeffer (January 2003) provide the full calculation needed.

Unfortunately, information gain only works with discrete values and assumes independence between features. ReliefF on the other hand can estimate feature quality even though there are strong dependencies between features, and does not rely on

discretising of continuous features but can handle continuous features directly (Kononenko & Robnik-Sikonja (1997) p.1).

When we know which abilities we are looking for in a good feature, it is time to find suitable feature candidates for my system. To further select features, it is important with domain knowledge since it allows us to find good feature candidates more quickly than a solely mathematical approach. In the following section, I am also going to present feature candidates that I found and some of the underlying factors playing a role in the whisky domain.

There are several factors in the production of a whisky which influence the final taste. Some of the most important factors are cask, water, location and barley.

6.3.1. Cask

The cask (barrel) in which a whisky is stored is important for the taste and aroma it develops. A study done by the distillery Glenmorangie shows that 50-60 % of the taste is due to the cask storage (Laurin (1998) p. 27).

There are several features of a cask which may be important for the taste; wood, size of cask, the re-use of the cask and the mixture of casks.

6.3.1.1. The wood:

- American oak
- European oak –Spanish, French

The casks used for whisky production are either made from American or European oak.

American oak casks are more moderate in flavour than the more tannin-rich flavour of European oak casks. European oak can again be divided into Spanish and French oak whereby Spanish oak is the most tannin-rich between them.

6.3.1.2. The size:

- Puncheon (580 l)
- Butt (500 l)
- Hogshead (250-305 l)
- American Barrel (173-191 l)
- Quarter (127-159 l)
- Octave (45-68 l)

The size of the cask also affects the final taste of the whisky. The smallest casks produce a more oaky and woody taste, whereas the biggest casks affect the taste to a lesser degree.

6.3.1.3. The re-use:

- Ex-Bourbon (1st filling after bourbon production)
- Ex-Bourbon-refill (2nd or more fills after bourbon production)
- Ex-Sherry (1st. filling after sherry production)
- Ex-Sherry-refill (second or more fills after sherry production)
- Ex-Port
- Ex-Madeira
- Ex-Rum
- Refill (used by grain whisky or other whisky)
- Raw (never used before)
- Ex-Bourbon-Sherry (the finish (usually last two years) is done on a sherry cask)

The previous use of the cask also influences the taste of the whisky. The above is just a selection of possible refill casks. There are even more varieties such as French wine casks and others, which have different influence on the final flavour. There is also a difference between sherries; an olorosso sherry has a more distinct flavour than an amontillado or a fino. One reason for this difference might be that the olorosso cask is made from Spanish oak, whereas an amontillado or a fino cask is made of French oak.

Ex-Sherry casks produce a whisky with a taste of sherry and chocolate, whereas ex-bourbon casks produce a softer and fruitier flavour. Ex-bourbon casks also have an ability to cover the smoky flavour some whiskies have. To preserve this, sherry casks can be used.

6.3.1.4. Mixing casks:

Whiskies are usually stored in maturing casks for most of the storage time, but often they are given two years in a finishing cask. However, there are examples where half of the storage time is done in finishing casks. Ex-bourbon casks are usually used for maturing and ex-sherry casks are used as finishing; this is by some distilleries called 'wood finish'. In 1996 Glenmorangie was the first distillery to start the trend of finishing the whisky in a second cask (Wishart (2002) p.26). Many distilleries have following this trend and have done experiments with whiskies finished in ex-rum, ex-port, and ex-wine and other casks.

There are some dependencies between the cask size and previous content of the cask; butts are usually used for sherry, and hogshead and American barrel are used for bourbon.

Due to this dependency and problems finding information about barrels size, I have decided to try and compile all the cask information into a single scalable feature.

Alternative ranking:

1. Raw
2. Sherry
3. Port
4. Sherry-Refill

5. Sherry-Bourbon
6. Bourbon
7. Bourbon-refill
8. Refill

This arrangement tries to represent the attributes of a cask on a two-dimensional scale which incorporates data such as where the oak originates from, number of times the cask has been re-used and the fact that the cask often has been used for other drinks prior to the whisky production, for example sherry, bourbon, port, madeira or rum.

I have also made an alternative separation of cask information where I have divided the information into four features; wood, reuse, maturing and finish. This hopefully separates whiskies stored on ex-bourbon casks more from whiskies stored on ex-sherry casks than the first alternative.

Wood, is a number value ranging from 0 to 100% whereby the number corresponds to how much time the cask storage is done in a European or American wood cask. 30% describes the situation where a whisky has been stored 30% of the time in a European oak cask and 70% in an American oak cask.

Re-use, is a number value ranging from 0 to 4 where the number corresponds to how many times the cask has been reused. 0 means that the cask has never been used before (raw cask), four means that the cask has been used four times or more.

Maturing, describes which previous content was held by the cask before it was used for maturing whisky. Ex-bourbon casks are usually used for maturing, but some makers only use ex-sherry casks. Maturing casks previously containing whisky are called refill casks. A refill cask might have been refilled more than 3 times, and the wood character will be largely reduced. Whiskies matured on refill casks are less modified by the cask than whiskies stored on 'fresher' casks.

Finish, is the name of the drink previously stored on the cask used for finishing. Normal reuse casks are ex-sherry casks, ex-port casks, ex-rum casks, but recently ex-wine is also being used.

Thus, there are two alternatives:

Alternative 1:

Feature			
Name	Value	Range	Scaling
Cask	Discrete	1-8	1.Raw, 2.Sherry, 3.Port, 4.Sherry-Refill, 5.Sherry-Bourbon, 6.Bourbon, 7.Bourbon-Refill, 8.Refill

Figure 6.4: Cask - Alternative 1

Alternative 2:

Feature			
Name	Value	Range	Scaling
Wood	Discrete/ Continuous	0-100%	0% = stored on American cask, 100% = stored on European cask.
Re-use	Discrete	0-4	0. No refill, 1.1st refill, 2.2nd refill, 3.3rd refill, 4. four refills or more
Maturing	Discrete	1-5	1.Port, 2.Sherry, 3.Rum, 4.Whisky, 5.Bourbon
Finish	Discrete	1-5	1.Port, 2.Sherry, 3.Rum, 4.White-Wine, 5.None

Figure 6.5: Cask - Alternative 2

6.3.2. Location

Scottish whiskies are typically divided into five categories according to the region which they are produced. Whiskies produced near the sea often have a salty character, and this is said to be a combination of the storage of casks in shelters without proper insulation and the casks' ability to absorb oxygen from the air. (Laurin (1998) p. 27) The climate at the location may also alter the whisky during storage. In a dry climate, water evaporates more than alcohol and in a moist climate vice versa will happen. Different regions might also have their own specific production methods. For example, whiskies produced in the region Islay are known for a taste with a lot of peat. This is because there is a tradition for drying the barley on peat fire. These are just a few factors which are dependent on the region. Since there is a lot of implicit information in this region label, I have decided to try two scenarios; one with only region and another where I have separated the region into a

physical location measured in longitude and latitude. In addition, the physical location should be combined with information about production techniques. One obvious reason to split the region into smaller information is that new information about production can be easily added without thinking of dependencies.

Again, two alternatives:

Alternative 1:

Feature			
Name	Value	Range	Scaling
Region	Discrete	1-6	1.Speyside, 2.Highland, 3.Campbeltown 4.Lowland, 5.Islands, 6.Islay

Figure 6.6: Location - Alternative 1

Alternative 2:

Feature			
Name	Value	Range	Scaling
Latitude	Discrete/ Continuous	0-90	Decimal degrees
Longitude	Discrete/ Continuous	0-180	Decimal degrees

Figure 6.7: Location - Alternative 2

6.3.3. Water

Water and barley are the two main ingredients of a whisky and play an important role for the final product. The water quality is described by several factors ranging from measurable abilities such as pH value and minerals content, to less measurable abilities such as pureness, peatness, and sweetness. Soft water (low pH) is often preferred when making whisky as it is said to absorb more flavour from the barley than hard water (high pH value). Experiments done by some distilleries have shown that water with lighter peating produced a whisky with a less peated style (Wishart (2002) p.19).

Due to the lack of resources and the time constraints, I have chosen not to include water as a feature. With sufficient time I would have included a feature describing the peatiness of the water.

6.3.4. Barley

Good barley must of course be free of mould and insects. But there are also some types of barley which are better than others. Golden promise is known to be the best barley because of its high starch content which yields more alcohol than other barley. Another important factor is how it is dried. In older days, every distillery dried their own barley over a peat fire. Today, it is more common to buy finished coke dried barley, which results in a less peaty whisky (Wishart (2002) p.20). It was hard to find information about the barley for every whisky, again with sufficient time I would have included a feature describing how the barley was dried.

6.3.5. Washback

The whisky is fermented in large vessels called washbacks. These washbacks are usually made from wood, but some distilleries use stainless-steel. It is believed that the bacteria living in the wooden washbacks give additional taste to the whisky (Wishart (2002) p.21). Since Wishart (2002) describes the material used for the washback by each distillery, and that it might influence the taste, I have added washback as a feature candidate.

Feature			
Name	Value	Range	Scaling
Washback	Discrete	1-2	1.Wood, 2.Stainless-Steel

Figure 6.8: Washback

6.3.6. Distilling

The distilling process describes the process where the alcohols, esters, aldehydes and acids are separated from the yeast. The process is monitored and controlled by a stillman. The distillation process consists of three phases; first phase produces low-wines. Low-

wines only consist of around 21-23% alcohol. The second phase is called the middle cut and produces alcohol around 70%. The final phase consists of feints which are oily substances that can ruin the whisky. These feints are re-distilled and used in the final product. The stillman's job is to mix spirit from the different phases into a whisky which is ready for cask storage. The stillman's skill not only influence the distillation process, but also the still used. A still with a high tall neck produces a clean and fruity whisky, whereas a short and wide still produces a powerful and fat whisky. Some distilleries use triple distillation which produces a cleaner and less tasty whisky then 'normal' double distillation.

It would be difficult to assess the stillman's knowledge and skill in one or more features even though this probably holds some of the 'secrets' of the whisky's flavour. However, both shape of the still and number of distillation are candidates that are better suited as features because they are easy to express in a computer system. I would have used the shape of the still and the number of distillations if I have had this information available for every whisky. Since I do not have complete information, I have to exclude these candidates, once again with more time it should be possible to gather the missing information.

6.3.7. Age

Age is describing how long a whisky has matured oak casks. The longer a whisky is stored in a cask, the smoother and softer the taste gets. However, it also acquires a more woody taste which is not to everyone's liking. Another side effect is that the alcohol volume also decreases over time (Laurin (1998) p. 31). Even though age has an influence on the taste of the whisky, it does not provide any information dividing two whiskies from each other. A Macallan would still be a Macallan even though it is stored for 15 years instead of 12 years.

6.4. Selecting classifier (model)

The next phase is to select a classifier. We usually start with a simple model, and to improve accuracy a more advanced classifier can be used. For selecting a proper classifier there are several factors to consider. Are we going to use a classifier utilising supervised, unsupervised or reinforced learning? To further explain the different forms of learning, we can think of a scenario with a teacher and a pupil. With supervised learning, the pupil

gets all the correct answers from the teacher and learns from it. Mathematically speaking, the pupil tries to learn a function from examples of its input and output. Instead of being told the right answer, the teacher can provide feedback in the form of correct or wrong. This is called reinforced learning. The pupil is searching for a function which is not giving a false answer. A false answer leads to a rejection of the current function. If the pupil is left with no feedback from the teacher, we call it unsupervised learning. Unsupervised learning is the situation where the pupil is learning patterns in the input where no output is specified (Norvig & Russell (2003) p.650).

Since my training set is equipped with class labels, the most sensible thing is to use a classifier that supports supervised learning. There are several different supervised classifiers, and it is important to select the right one. As discussed earlier in chapter 4, there are different benefits and drawbacks with the different classifiers. Duda et al. (2001) state that there is no ultimate best classifier which works best in all situations, through the use of the *No Free Lunch Theorem* and the *Ugly Duckling Theorem* (p. 454-461). This leaves the responsibility of selecting a good classifier to the designer of the pattern recognition system. There are several factors which are important when selecting the appropriate classifier:

- Accuracy
- Speed of learning
- Speed of classification
- Space requirements
- Specialisation
- Pre-processing
- Easy to understand

6.4.1.1. Accuracy

The most important ability for a classifier is of course the ability to make accurate predictions. Unfortunately, increased accuracy often leads to high requirements of

training data, computing power, pre-processing and space. The accuracy of a classifier is usually measured in error rate.

6.4.1.2. Speed of learning

When we are talking about the speed of learning, we are not only thinking of the time the training process takes, but also the amount of training data needed for training the classifier. Learning speed is often expressed as a learning curve which can be used to compare different classifiers. Decision trees often provide a good result on small training sets; in those cases where prior information is available, a Bayesian classifier usually performs better.

6.4.1.3. Speed of classification

In laboratory experiments, the speed of the classification process is usually of lesser importance. However, when taking a decision out in the field the speed of the classifier might be crucial. Usually specialised classifiers are faster than general classifiers.

6.4.1.4. Space requirements

If space is limited, some classifiers perform better than others. There is a huge difference in memory usage between the different classifiers. A NNR classifier has to store every sample in memory, whereas a Bayesian classifier only needs to store a formula for each class.

6.4.1.5. Specialisation

There is a range of specialised classifiers for different fields and they perform better than the more general approaches. These special methods often require more pre-processing than a more general approach. An example of a specialised classifier is the LeNet neural network classifier which is specialised to recognize handwritten and machine written characters (Norvig & Russell (2003) p.753). The huge drawback with LeNet is that it is useless on other classification problems.

6.4.1.6. Pre-processing

Some classifiers need the data on a certain form or the classifier must be configured to fit the data. Before a neural network can be used, an appropriate number of layers and nodes have to be found according to the data being classified.

6.4.1.7. Easy to understand

This might sound like an odd factor, but it might be one of the most important factors. This is because it makes it easier to understand the result we get and the type of changes that are needed to produce a better result.

6.4.1.8. Thesis requirements

Hence, being accurate is one of the most important requirements for my classifier so that the recommendation is as good as possible. It is also important to use a classifier which works with small training sets. The memory and classification speed is less important both because the project is done in a lab environment and also because there is a small training set with few features. Therefore, any classifier can handle it with ease. Since it is unlikely that someone has made a special classifier for classification of whiskies, I would try to use a general classifier. Due to the fact that the classifier selected is going to be used as part of my thesis, it would be beneficial if the classifier is easy to understand. I decided to use the two most common supervised classifiers described earlier in chapter 4; Bayesian and NNR. Both of these classifiers are supported by Weka.

6.4.2. Training

For training the classifier we need a training set selected from the data collected. For unsupervised learning we can use the samples collected in the data collection phase directly. On the other hand for supervised learning, we need to supply the class labels to every sample in the test data.

6.4.3. Evaluation

When evaluating the results it is important to look back at the requirements used when selecting the classifier and see if the requirements are fulfilled.

6.4.3.1. Cross-validation

Supervised learning provides a smart way to test the classifier often referred to as cross-validation. By dividing the training data into two disjoint sets; a training set and a test set, it gives us the opportunity to test the classifier with samples we already know of the correct class.

The training data are usually divided into a training set consisting of 66% of the data and a test set which consists of the remaining 33%. However, this ratio can be adjusted to fit the size of the test data. For small amounts of test data, we usually want to use as much of the test data as possible for training. The most extreme variant of this is called 'leave one out', this approach uses one sample as a test set and the rest for training. A common mistake is to include the same sample both in the training and test set; this is known as 'testing on the training set' (Duda et al. (2001) p. 483).

Weka provides something they call 10 fold cross-validation testing. This is the cross-validation test done 10 times. For each validation the data are randomly split into a training set and test set. It is ensured that each class is equally represented in the training set. The 10 fold cross-validation calculates the means error rate for the classifier.

Cross-validation can be used empirically to test different classifiers and their performance. (Also different features) The result is an estimate of the 'real' life performance, and not the actual 'real' life performance. Duda et al. (2001) show that the result provided by the cross-validation is optimistic, but the accuracy increases as more training data are supplied (p. 484-485).

6.4.3.2. Confusion matrix

A confusion matrix is used to give a visual representation of the classification.

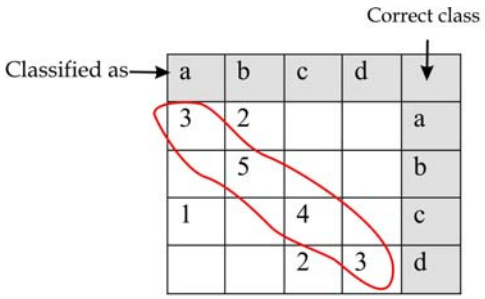


Figure 6.9: Confusion matrix

The confusion matrix tells us which class a sample is classified as, and which class is the correct class for that sample. A perfect classification is recognized by having all the values on the diagonal in the red area on the figure. From figure 6.9 we can see that for class A three samples are correctly classified, but two samples are incorrectly classified as class B.

6.4.3.3. Findings: Features ranked by information gain

Information Gain Ranking		
Ranking	Feature	Score
1	Region	0.756
2	Maturing	0.261
3	Washback	0.15
4	Finish	0.127
5	Wood	0.0
6	Re-use	0.0

Figure 6.10: Findings - Features ranked by information gain

6.4.3.4. Findings: Features ranked by ReliefF

ReliefF Ranking		
Ranking	Feature	Score
1	Region	0.109
2	Wood	0.044
3	Washback	0.026
4	Maturing	0.019
5	Finish	0.012
6	Re-use	0.003

Figure 6.11: Findings - Features ranked by ReliefF

6.4.3.5. Findings: 10 fold cross-validation

Run	Classifier	Features	Build Time (sec.)	Error rate (%)	Confusion matrix											Comment
1	NNR	Region, Cask	0	80.4878	a	b	c	d	e	f	g	h	i	j		With basic features
					4	2									a	
					3	2	2	1		2		2		1	b	
						2	3	1	1	1		1			c	
						2	1	1	3	1	1				d	
						1	1	3	2				1		e	
					1	1	1	2	2		1	2			f	
							1	1	1	1	1	1			g	
						4	1			4					h	
						1					2		3		i	
						1						2	3		j	
2	NaiveBayes	Region,	0.02	81.7073	a	b	c	d	e	f	g	h	i	j		With basic

		Cask					6									a	features
						1	8	1	1				1		1	b	
							5					1	3			c	
							3	2		2	1			1		d	
								3	2		2			1		e	
							6			1		2	1			f	
							2	1		1	2					g	
							3	1			1	1	3			h	
							2		1					3		i	
										1		1	1	2	1	j	
3	NNR	Longitude, Latitude, Cask	0	81.7073		a	b	c	d	e	f	g	h	i	j		Used Longitude, Latitude instead of Region
						3	1	1					1			a	
						3	2	2	2	1	2				1	b	
							1	2	1		2	2	1			c	
							2	2		2	2		1			d	
								1	1	2	1	2			1	e	
							1	2	2	1	1		3			f	
								2		1		1	1	1		g	
						1				1	1	3		1	2	h	
										1		1	1	3		i	
							1			1			3		1	j	
4	NaiveBayes	Longitude, Latitude, Cask	0.02	80.4878		a	b	c	d	e	f	g	h	i	j		Used Longitude, Latitude instead of Region
						3	3									a	
						4	2	3	1	1	1			1		b	
						1	2	2				1	3			c	
						1	1	3		2		1			1	d	
								5		1				1	1	e	
						2		5	1	1	1					f	
						1		4						1		g	
							1	2	1		2		2		1	h	
							1							2	3	i	
													1	2	3	j	
5	NNR	Region, Wood, Re-use, Maturing, Finish	0	80.4878		a	b	c	d	e	f	g	h	i	j		Region with cask divided into wood, Re-use, Maturing, Finish
						3	1	1	1							a	
						2	3		2		2	2	1		1	b	
								1	2		3		3			c	
							2	2	2	1	1			1		d	
									2		2	1	1	1	1	e	
						1	1	1	2	1	3		1			f	
						1			3		1		1			g	
						1	1	2			1	1	2		1	h	
								1	1	2					2	i	
							1			1			1	1	2	j	
6	NaiveBayes	Region, Wood, Re-use, Maturing, Finish	0.02	80.4878		a	b	c	d	e	f	g	h	i	j		Region with cask divided into Wood, Re-use, Maturing, Finish
						3	2						1			a	
						7		2			3				1	b	
							1	2		1	2		2		1	c	
						2	1	1		2	2			1		d	
								1	1	1	1	1	1		2	e	
							2			1	4	2	1			f	
							1	1		2	1		1			g	
							1			1	2	2	1		2	h	
						2		1						1	2	i	
													1	1	4	j	
7	NNR	Region,	0	84.1463		a	b	c	d	e	f	g	h	i	j		Used only

		Maturing, Finish			3	1				1	1				a	the three best features from Info. gain ranking. (except Washback)
					1	1	1	2	2	2	1	2	1		b	
						2	1	2	1	2		1			c	
						3			3		1	1	1		d	
							1	1	1		3	1		1	e	
					1	2			1	3	2	1			f	
						1		1	1	1	1	1			g	
						2	2			2	2	1			h	
								1	2		1			2	i	
						1							3	2	j	
8	NNR	Region, Wood, Maturing	0	80.4878	a	b	c	d	e	f	g	h	i	j		Used only the three best from ReliefF ranking. (except Washback)
					2	2	1					1			a	
					1	6	1	2		1		1		1	b	
						4		2		1		2			c	
						3	1		1	1	2		1		d	
						1		1			2	3		1	e	
						4	2			1	1	2			f	
						1			1		2	1	1		g	
					1		2		1	1	2	1		1	h	
								2	1				2	1	i	
						1			1			1	1	2	j	
9	NNR	Region, Wood, Re-use	0	75.6098	a	b	c	d	e	f	g	h	i	j		Used Wood and Re-use
					3	1	1			1					a	
					2	4	1			3		2		1	b	
					1		3	2		1	1	1			c	
						3	1		3	1			1		d	
						1		2	2	1		1		1	e	
						1	1	2	1	3		2			f	
						1		2	1		2				g	
						2	2		3					2	h	
								1			1		1	3	i	
						1			1			1	1	2	j	
10	NaiveBayes	Region, Wood, Re-use	0.02	80.4878	a	b	c	d	e	f	g	h	i	j		
					4	1						1			a	
					9		3							1	b	
						1	1		2	1		4			c	
					3	1	1		1	2				1	d	
					1		2		1	1		2		1	e	
					1	3			2	1		3			f	
					1	1	2		2						g	
					1	1	2		1			4			h	
					2	1	1						1	1	i	
					1							1		4	j	
11	NNR	Longitude, Latitude, Wood, Re-use, Washback	0	75.6098	a	b	c	d	e	f	g	h	i	j		Tried washback
					3	2	1								a	
					2	4	2	2			1	1		1	b	
						2	3			1		3			c	
						2	1		1	2	2			1	d	
							1	1	3		1		1	1	e	
							2	4	3		1				f	
						1			1	1	3				g	
						1			3		3	1		1	h	
						1			1				2	2	i	
									1			2	2	1	j	
12	NNR	Region,	0	71.9512	a	b	c	d	e	f	g	h	i	j		Washback

		Wood, Reuse, Washback			2	3	1								a	and Region.
					3	5		2				2		1	b	
					1	1	2	2			1	2			c	
					1	1	1	3	1	1			1		d	
						1		1	3		2			1	e	
						3	1		2	2	1	1			f	
						1		1			2	1	1		g	
						1	3		1		1	2		1	h	
							1	2	1				1	1	i	
						1			2			1	1	1	j	
13	NNR	Region, Wood, Reuse, Washback	0	68.2927	a	b	c	d	e	f						Reduced to 6 classes
					3	1	1		1		a					
					2	9	4	4	2	1	b					
						6	2	3	4	2	c					
					1	3	3	7	2		d					
					1	3	1	4	3	3	e					
							1		3	2	f					
14	NNR	Region, Wood, Reuse, Washback	0	56.0976	a	b	c	d								Reduced to 4 classes
					15	9	2	2	a							
					10	16	5	2	b							
					4	5	3	3	c							
						1	3	2	d							

Figure 6.12: Findings - 10 fold cross-validation

6.4.3.6. Overfitting

Even though we get a result as expected, this does not necessary means that our features and classification are correct. Our classifier might suffer from overfitting. This describes the situation where we have selected an overly complex model for our classifier which makes it able to classify the training data perfectly, but work poorly on new unseen data.

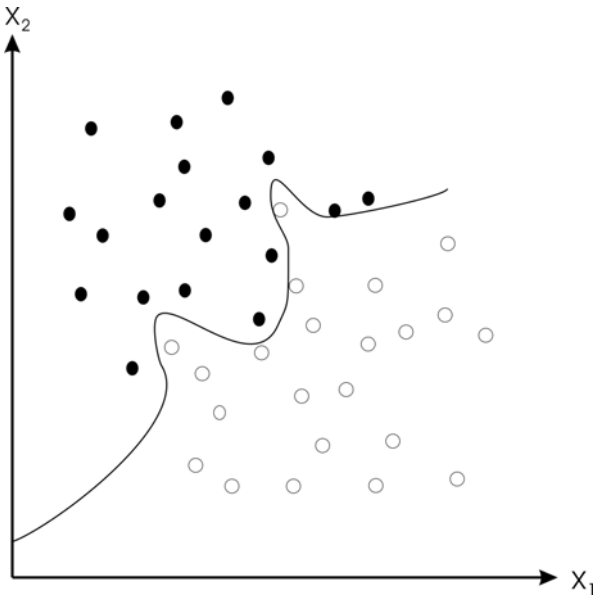


Figure 6.13: Overfitting

This figure (figure 6.13) shows an example of overfitting, we can se that the decision boundary is overly complex.

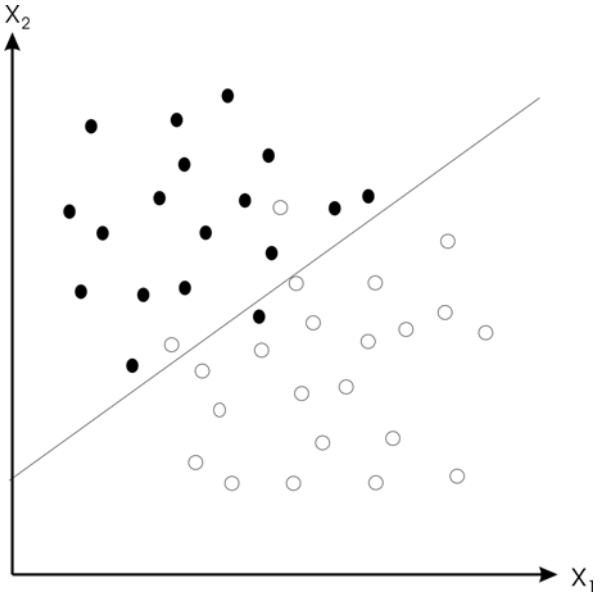


Figure 6.14: Simple-Linear

Figure 6.14 shows the same data as figure 6.13, but with a simple linear boundary. We can see that some samples are not correctly classified. However, when new unknown samples arrive, this classifier might prove to be better than the overly complex classifier in figure 6.13.

Chapter VII

FINDINGS AGENT PLATFORM

As we saw earlier in chapter 5, the testing was conducted in 4 separate tests; testing of the mobile platform, testing of the multi agent platform (recommendation), testing of the AmbieSense framework and testing of the classification. The findings from the classification system have already been shown in chapter 6. In this chapter, the findings of the remaining three tests are presented.

7.1. AmbieSense framework

As mentioned earlier in chapter 5, the testing was performed by the AmbieSense project (Wienhofen et al. (2004)), and I will only provide a brief summary of the work they have done. Each component was tested separately and together to ensure that they integrated with each other. No problems during the testing of the components were reported. The integration testing was more interesting since different components were made by different parties in the project. For instance, the developed agents developed by CognIT had to be tested with the context middleware developed by SINTEF. The integration testing only gave results as expected, and everything worked as planned. The final complete system testing on Gardermoen provided more mixed results.

7.1.1. Complete system testing

The complete testing consisted of functionality testing and user acceptance testing.

7.1.1.1. Functionality testing

The functionality testing included 4 tests as described earlier in chapter 5:

1. Before the testing at OSL Gardermoen, the system had been tested in a laboratory wired and wireless using both the IIOP and the HTTP protocol. Both protocols worked well, but the HTTP protocol was easier to use since the IP address for the

machines could be used instead of the long CORBA address used by IIOP. The network used on OSL Gardermoen was a newly installed WLAN, which proved to be rather unstable and had caused several time-outs. First the IIOP protocol was used for connecting the agent platform. However, the agent platform was not able to run stable on this setup due to several time-outs when trying to retrieve content from the Content Agent. Another problem discovered was that a content message took approximately 1 to 7 seconds to be dispatched. The HTTP protocol was used with the same result. Since none of the protocols proved to give satisfactory inter-platform communication, all agents and content was run from the handheld device. With this setup, everything worked fine and the latency was reduced to less than 1 second.

2. The testing was performed on two PDAs. The PDAs was Compaq iPAC 3870 running Linux OS with a Blackdown VM. The PDAs used a WLAN card to communicate with the JADE platform and Bluetooth to communicate with the Context Tag. There were no problems during testing.

3. The tested agents were running 3 different setups; distributed, partially distributed and locally. The distributed setup consisted of running each agent on best suited locations such as Context Agent on the mobile device, Recommender Agent and the Content Agent running on a server. The partially distributed setup involved all agents on the mobile device and the content on a server. In the last setup all agents and content were stored on the mobile device. The distributed setup did not work particularly well as already explained earlier. In the laboratory, this setup had proved to work well. The partially distributed setup did not prove to be much better since it got timed out when retrieving content.

4. Response times were measured to less than 1000 ms/request and were evaluated as good by the users. The uptime of the agent system was 100% during the testing on OSL Gardermoen.

Agent Setup	Laboratory		Field scenario OSL Gardermoen	
	IIOP	**HTTP**	**IIOP**	**HTTP**
Distributed	Worked fine	Worked fine	Network Address Translation problems. Content messages use 1 to 7 seconds for being dispatched.	Content messages use 1 to 7 seconds for being dispatched.
Partially Distributed	Worked fine	Worked fine	Time-outs when retrieving content.	Time-outs when retrieving content.
Locally	Worked fine	Worked fine	Worked fine.	Worked fine.

Figure 7.1: Findings - Functionality testing

7.1.1.2. User acceptance testing

The user acceptance testing results only contain comments from users about the agent-based content delivery, since other comments about speed, reliability of the WLAN and hardware will be covered in the Test and Evaluation Report (which is not yet finished). The AmbieSense project noted that the results from the survey were difficult to validate statistically, but they found some tendencies (Wienhofen et al. (2004)). In particular, there were three areas users had comments about; recommendation versus search, the quality of the recommendation and the profiles granularity.

1. In general the users liked the concept of getting recommendations based on a profile instead of searching for information themselves.

2. The users found the content presented by the agents useful and relevant, particularly the flight information status overview which provided information about the next flight and so on. The only exception was the shopping information which the users did not see the need for. The general attitude was that OSL Gardermoen is too small for this kind of information, but would probably be useful on larger airports such as Frankfurt. Instead of shopping information the users wanted information such as news and weather updates.

3. Another problem many users complained about was the granularity of the user preferences. Some wanted to specify more or less information about their preferences. More about the results of the user interviews can be found in the AmbieSense D9 documentation (Wienhofen et al. (2004)).

7.2. Multi Agent System (recommendation)

The testing of the MAS was done as described earlier in chapter 5 solely in laboratory conditions.

7.2.1. Component testing

Here too, each component was tested to verify that they worked as planned. There were not any real problems with the component testing; the only thing special was that a component usually was the agent itself. Compared to 'normal' components, it took some time to understand which abilities the agent platform could provide to the agents and how these abilities could be used. For instance, all the communication between agents is handled by the agent platform. Hence, the agent had to use the right function calls for utilising this ability. There is, however, a lot of good documentation and tutorials on the JADE web site (jade.tilab.com). When one agent was produced, this was used as a blueprint for the basic functionality for the rest of the agents. In particular, the way of communication and subscribing the DF Agent is handled equally in all agents. Another problem I had was due to the Java code generated by the BeanGenerator. The BeanGenerator was used to generate the code for the ontology as described earlier in chapter 5. The Java code contained references to the BeanGenerator itself something which seemed unnecessary. Fields (slots) with a set of optional values were defined as symbols for some strange reason. To fix these problems I simply removed all references to the BeanGenerator in the generated Java code, and the symbols were converted into normal Stings.

7.2.2. Integration testing

The integration testing was performed using a scenario where a user is passing a context tag. The scenario used for testing is described in the use case (figure 5.12) presented in chapter 5. For capturing the communication between the different agents, a Sniffer Agent was used as described in chapter 5.

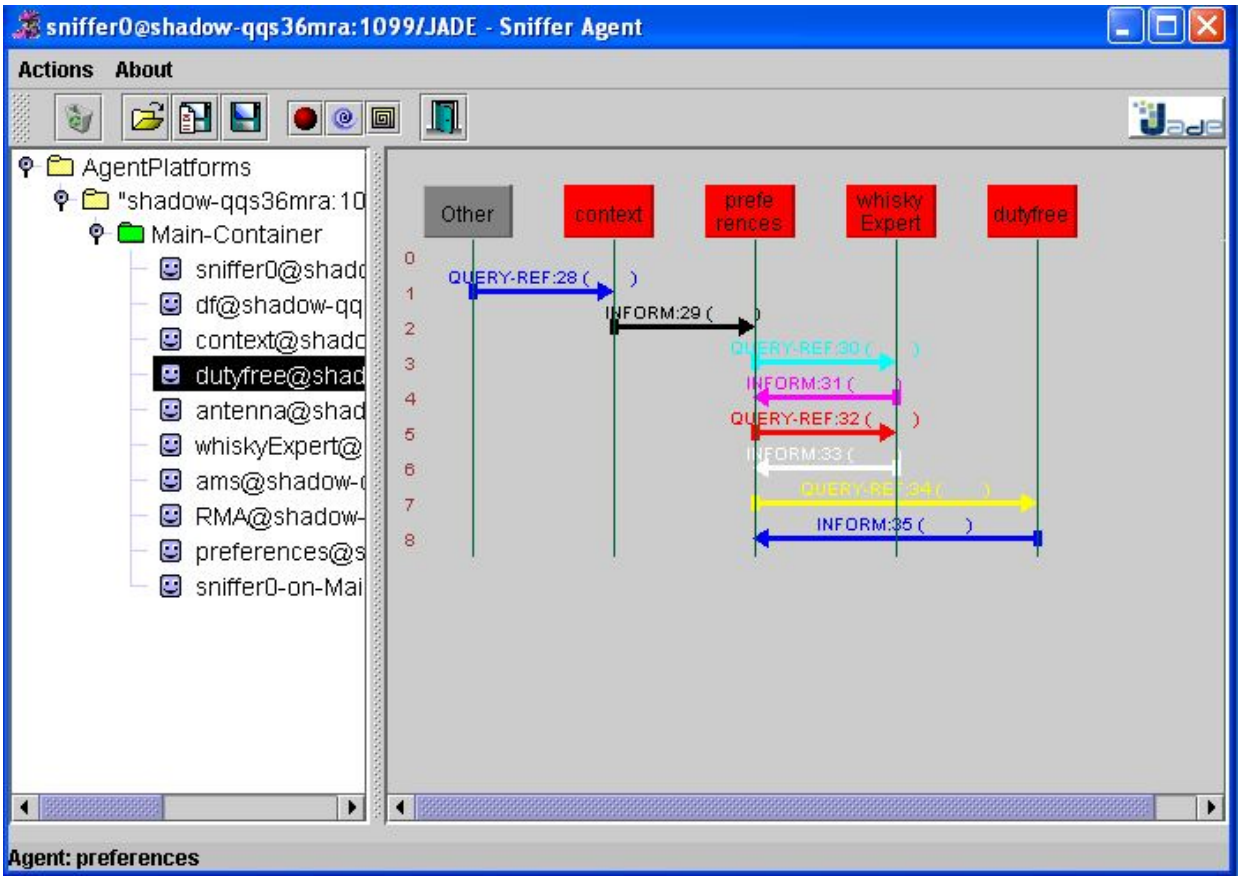


Figure 7.2: Findings - Sniffer Agent

Figure 7.2 shows the sequence diagram created by the Sniffer Agent. We can see that the sequence of events was initiated from 'other', which in this case was the antenna.

During this testing, I encountered some problems where agents did not respond in time to the requesting agent. The requesting agent did stop working because it was waiting for a reply which never came, or came too late. Most of these problems were solved by

programming waiting behaviours or by letting the agent continue without the requested information.

7.2.3. Complete system testing

As described earlier in chapter 5, the complete system testing was divided into a test of the recommendation and a test of the MAS platform.

7.2.3.1. Recommendation

For the recommendation, I did testing with two scenarios; a scenario where the user preferred the whisky Glenmorangie 10 years old and a second scenario where the user wanted any whisky with rating 85.

```
Preferred products:

Glenmorangie  10  259  80

Matched products:


Recommended products:

Glen Garioch  15  239  79

Balblair  16  289  76

Oban  14  369  79
```

Figure 7.3: Findings – Selecting preferred product

From figure 7.3 we can see that the user prefers Glemorangie 10 years old, and 3 whiskies are recommended.

```
Preferred products:


Matched products:
Glenlivet  12  209  85
Ardbeg  10  289  85


Recommended products:
Laphroaig  10  289  86
Speyburn  10  219  71
Dalwhinnie  15  269  76
Talisker  10  279  90
```

Figure 7.4: Findings – Specify preferences by properties

In the second scenario; figure 7.4, we can see that the system had found two whiskies with 85 in rating (the last number), listed under matched products. Based on both of these matched whiskies, the system recommended 4 additional whiskies.

7.2.3.2. Test of MAS platform

The testing of the MAS platform consisted of two tests:

- Running agents on different JADE platforms and machines
- Adding and replacing agents at runtime

For running agents on different JADE platforms, two computers running Windows XP were used. Both computers had JADE and Java installed. One computer was used as the server hosting every agent except the Preferences Agent, the other as the client. The client machine hosted the main platform whereas the server machine hosted the remote platform. The setup was as follows:

JADE Platform	Agents Hosted	Machine Name	IP-address	Operating System
main	Preferences	shadow-qqs36mra		Windows XP
remote	WhiskyExpert,Dutyfree, Context, Antenna(dummy)	shadow-hh17tdm5		Windows XP

Figure 7.5: JADE platform setup

Before the application could be run, the two JADE platforms had to be connected. It proved to be a difficult and time consuming process. Firstly, the remote platform had to be connected to the main platform with IP address and name of the AMS Agent. Secondly, the remote platform had to set one of the DF Agents to main DF Agent and to sub or child DF Agent.

When the two JADE platforms were connected, the application was run to see if it worked as it did on a single platform.

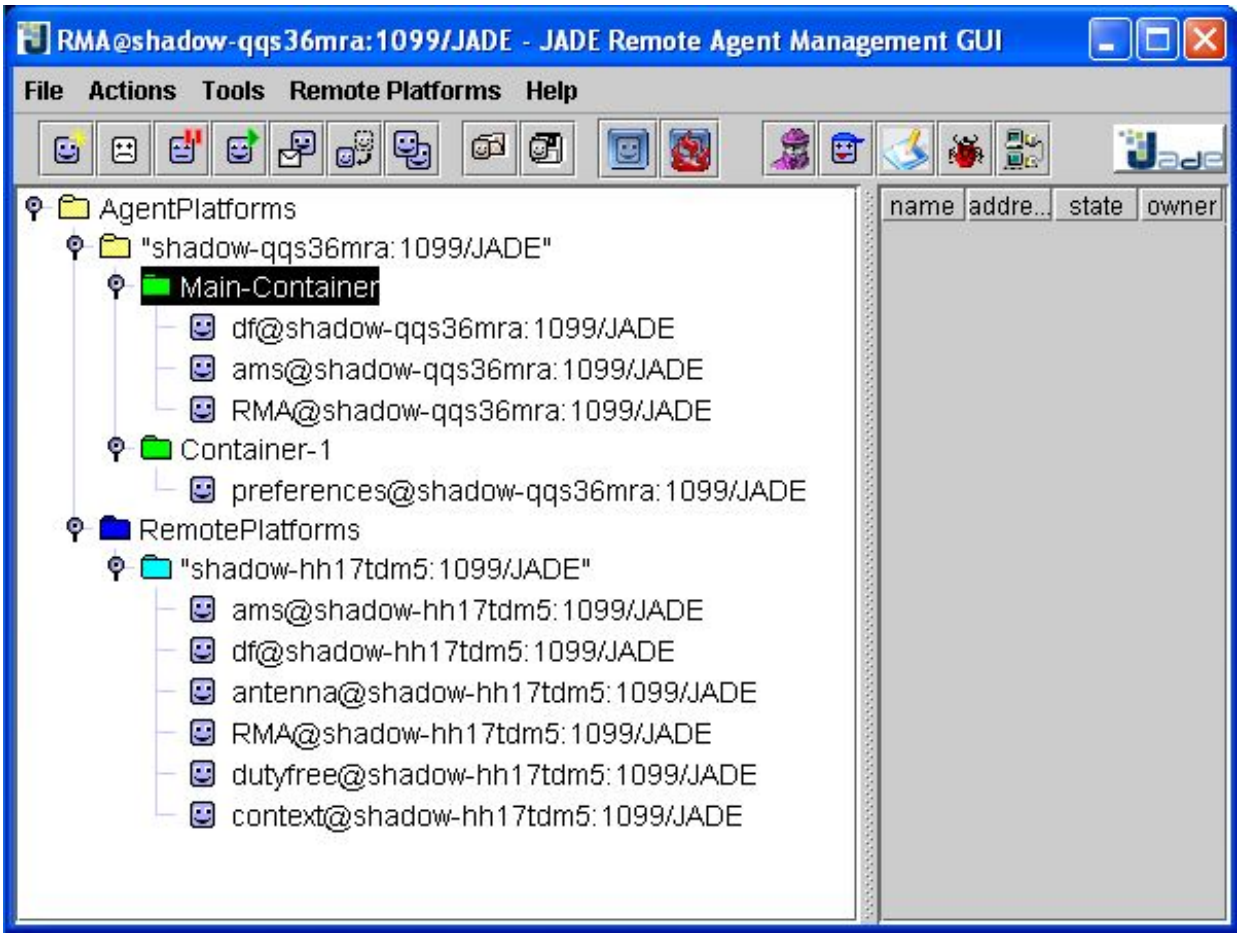


Figure 7.6: Findings - Running on remote platform

Figure 7.6 shows how the platform setup was represented in the JADE administration GUI. During the first test the agents could not find each other. After some investigation, I found that the search depth is defaulted to 1, which means that an agent only contacts the main DF Agent to search for other agents/services. If, however, the search depth is set to 2, the main DF's immediate children are also contacted. After the search depth was set to 2, the system ran as expected and no slowdown was experienced when running the application.

The testing of replacing one agent at runtime was carried out on one JADE platform. Two agents of the same kind; ContextOne and ContextTwo were started.

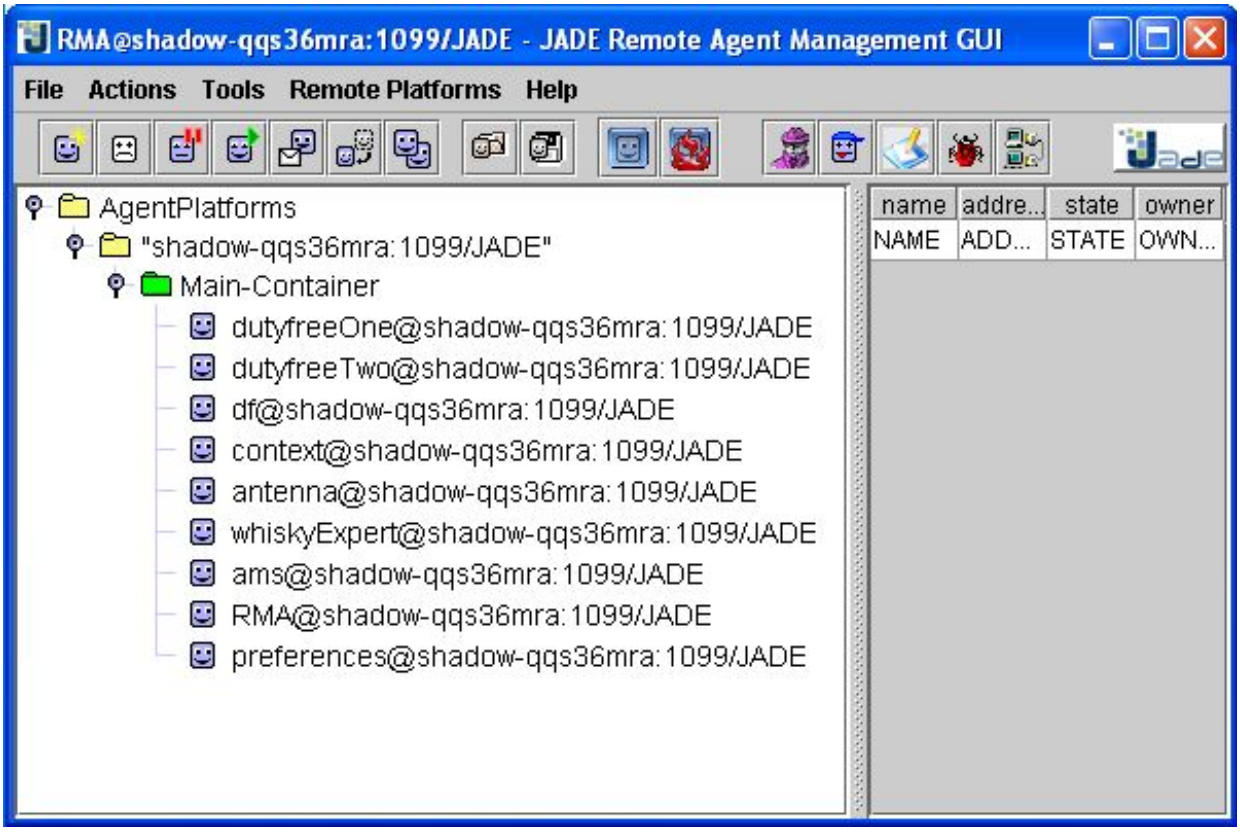


Figure 7.7: Findings - Replacing one agent at runtime

From figure 7.7 we can see both Context Agents running (ContextOne and ContextTwo). The agent in use was 'killed' with the agent 'kill' button to see if the other agent was utilised as a replacement. The particular agent used for testing was the Context Agent, but any other agent could be used. In the start, the ContextTwo Agent was not utilised because the Preferences Agent did not know that the ContextOne Agent was not available. This was solved by implementing automatic deregistering from the DF Agent when an agent dies.

7.3. Mobile platform

As described earlier in chapter 5, the testing of the mobile platform had been divided into different tasks:

- Compile application into right Java configuration
- Distribute application to mobile device or emulator
- Run application on mobile device or emulator as split or standalone if possible

7.3.1. Compiling application

When compiling the agent to MIDP several error messages occurred.

D:\JADE\ADD-ONS\LEAP\j2se\src\jade\core\management\BEAgentManagement
Service.java:680: warning: finally clause cannot complete normally

The error message shown here was just the first of many. After searching the Internet, I found that the problem was caused by different versions of the Wireless Toolkit (Tognalli (2004)). JADE-LEAP (version 3.1) was based on using Wireless Toolkit 2.0 while I had used Wireless Toolkit 2.1. The solution was to replace some of the files in the WTK2.1 installation with files from WTK 2.0. When this problem was solved, I ended up with 3 files; demoJ2SE, demoMidp.jar and demoMidp.jad.

7.3.2. Distributing application

For transferring the application, it was put on a web server. The only important thing was to set the right MIME type on the files so the web server could correctly serve the files. For the .jad file the MIME type was set to text and for the .jar file the MIME type was set to application. No problems were experienced with using the web server for distributing the JADE agents.

7.3.3. Running application

For running the application, different devices were used ranging from different emulators to one PDA and several mobile phones.

7.3.3.1. Running on HP Jornada 548 PDA

The PDA used for testing was a HP Jornada 548 from the year 2000. Hence, the PDA was relatively old and the official VM for this PDA Chai VM only supported Personal Java. There exists commercial VM that support MIDP, but since this is a research project, none of them were tried. Sun has an open source VM that supports MIDP called KVM *(Kilo Virtual Machine)*. However, it had to be ported to the specific platform (SH3/Windows CE 3.0). No VM could be found for this PDA so no testing was done on this PDA. Figure 7.8 shows a list of the different VMs tried.

Alternative VMs for HP Jornada 548

Virtual machine	Operating system	MIDP	CLDC	CDC	Free	Comment
PersonalJava Runtime Environment	WindowsCE 2.1	✗	✗	✗	✓	No versions for WindowsCE 3.0. EOF
IBM J9 VM	Palm-OS, WindowsCE and other non-PDA-OS	✓	✓	✗	✗	
Jeode EVM	WindowsCE 2.12 - 3.0 and other non-PDA-OS	✗	✗	✗	✗	PersonalJava
CrEme	WindowsCE	✗	✗	✗	✗	PersonalJava
Kada VM	PalmOS og WindowsCE	✗	✓	✓	✗	Only evaluation sample available for download
Chai VM	WindowsCe and other non-PDA-OS.	✗	✓	✗	✗	Only supports PersonalJava and not MIDP.
SuperWaba	Palm-OS, WindowsCE and Symbian	✗	✗	✗	✓	Alterative to MIDP, and not a MIDP implementation
K Virtual Machine	Palm-OS	✗	✓	✗	✓	Can be ported to other platforms

Figure 7.8: Findings - Virtual Machines

The list (figure 7.8) shows if the VM has support for MIDP, CLDC *(Connected Limited Device Configuration)*, CDC *(Connected Device Configuration)*, it also shows which OS the VM is compatible with and if the VM is free to download.

7.3.3.2. Running on mobile phones

Running on mobile phones did not prove to be any easier then running on the PDA. For testing we had some phones from Sony Ericsson and Nokia available. Here are the results:

Brand	Model	Did run
Nokia	6210 (Series40)	No
Nokia	7250 (Series 40)	No
Sony Ericsson	T610	No
Sony Ericsson	P800	No

Figure 7.9: Findings - Mobile phones running Demo application

The different mobile phones tested had different limitations making it impossible to run the Demo application on most of the phones tested.

Phones in series 40 from Nokia are limited to run applications smaller than 64 KB. The series 60, however, is not prone to this limitation. The Demo was unfortunately 170 kb after optimisation which made it unable to run on any series 40 phones.

JADE-LEAP supports MIDP 1.0 for mobile phones but also requires that the device it is run on supports sockets. The MIDP 1.0 standard does not require socket support; thus, some mobile device does not have socket support. (Java Community Process (1998)) Most of Sony Ericsson's mobile phones support MIDP 1.0, but they do not have socket support. This meant that JADE-LEAP is unable to run on most of Sony Ericsson's phones such as T68, T610 and T630. Only the so called 'smartphones' from Sony Ericsson support sockets. Today it is only P900 and Z1010, but also the new K700 and S700 that support sockets. For an overview of the findings, I have produced a list of the most common mobile phones on the Norwegian market listing their abilities to run JADE-LEAP.

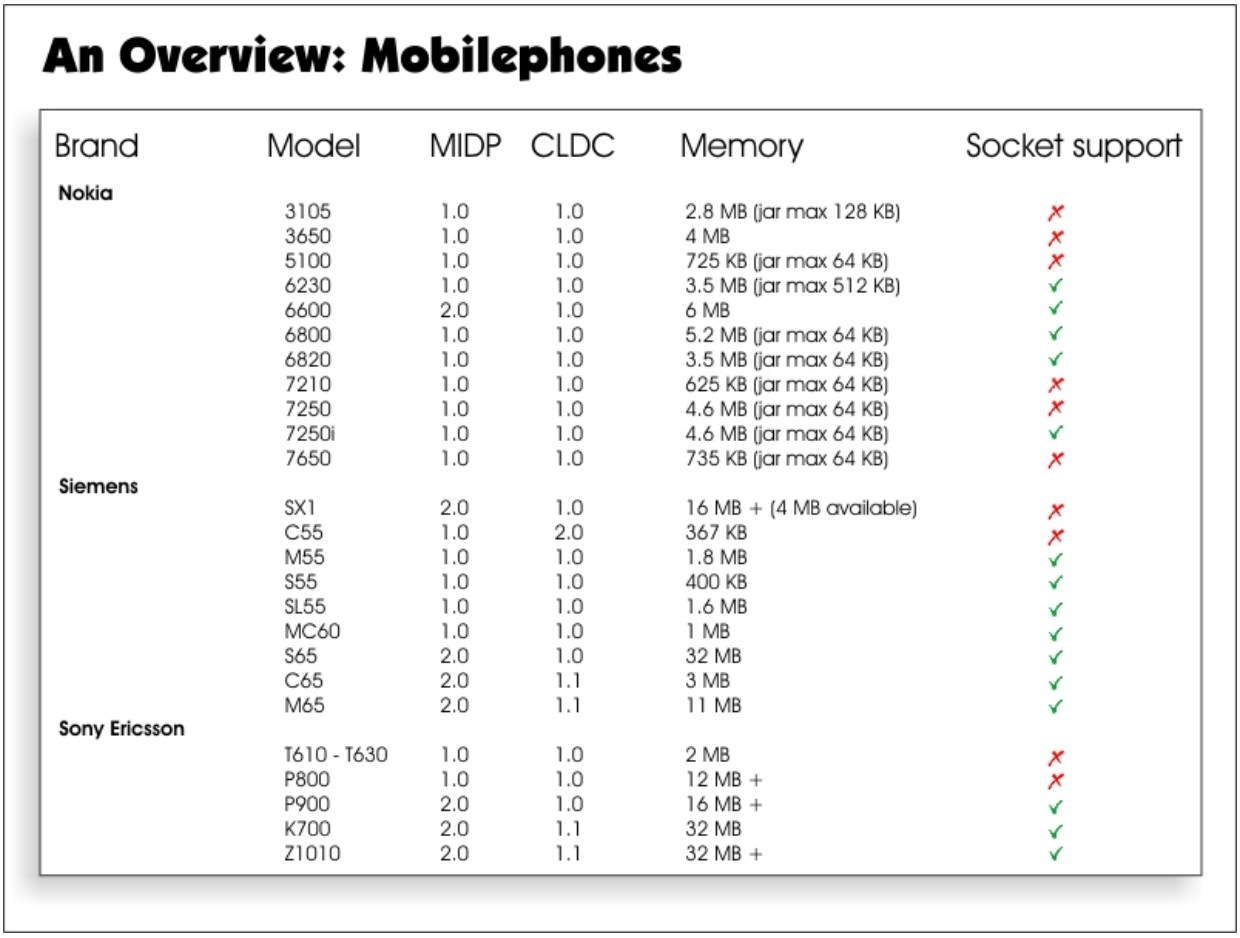


Figure 7.10: Findings - Mobile phones (general overview)

The information presented in figure 7.10 was gathered from different sources such as Jimm Mobile SourceForge (2004), Nokia Forum (forum.nokia.com) and Sony Ericsson Developer Forum (developer.sonyericsson.com).

7.3.3.3. Running on emulators

The testing on the mobile phone emulators went much better where the test application did work on all emulated phones. For emulating Sony Ericsson phones, the WTK 2.0 for Sony Ericsson phones found in Sony Ericsson's J2ME SDK 2.1 package were used. This special edition of WTK is able to emulate all of today's available Sony Ericsson phones. Phones that were tested with the emulator were Sony Ericsson T610 and P800. For Nokia

phones, Nokia's own Nokia Developer's Suite 2.0 was used for emulating the phones. Phones that were tested (emulated) with the emulator were the series 40 phone Nokia 6210.

Chapter VIII

DISCUSSION

In chapters 6 and 7 the findings of the testing were presented. What information can we get from these findings? The testing was conducted in 4 main areas as described earlier in chapter 5.5; AmbieSense framework, multi-agent system (recommendation), mobile platform and the classification.

8.1. AmbieSense framework

Even though the testing of the AmbieSense system was not done by me, I am going to comment on the findings which are important for my system. The testing of the AmbieSense system consisted of 3 tests presented earlier in chapter 5.5.1; components testing, integration testing and complete system testing. The component and integration testing did work as planned. The JADE framework can maybe take some of the credit for the success. We will look into that later when my system is discussed.

8.1.1. Complete system testing

The complete system testing was divided into a functionality testing and a user acceptance testing.

8.1.1.1. Functionality testing

The functionality testing included 4 tests.

1. The result from trying the different protocols did not give immediate answers since none of the protocols could provide a sufficient inter-platform communication. The problems were most likely due to the newly installed WLAN, and not the JADE platform or protocols.

2. The testing of the PDAs worked well, the JADE agents ran successfully on these PDAs compared to the PDA that I had used because they used a VM that supported J2SE.

3. The testing of the three different setups (distributed, partially distributed and locally) showed that running each agent locally on the mobile device was the only setup which worked satisfactory. Again, this was probably due to the WLAN and not the JADE platform as such.

4. Both the response time and up time were acceptable. However, more information on this will be available in the Test and Evaluation Report from the AmbieSense project.

From the functionality testing it became clear that the JADE platform relied heavily on the network and did not work in a distributed setup when the network bandwidth was too low or unstable. However, this is not something special for JADE alone but is true for every MAS as discussed earlier in chapter 2.3.

8.1.1.2. User acceptance testing

The AmbieSense system was tested by some users, what could they tell us about the system?

1. It is good that the users liked the concept of getting recommendations based on a profile instead of searching for items themselves. The users were positive about the idea since they did not have to physically search for the item themselves. Hence, one of the foundations of the system developed by the AmbieSense project is the concept of getting recommendations based on a profile. This is also relevant for my system since it also uses profiles to store information about users' duty-free shopping preferences.

2. When the users say they do not really see the point of the shopping recommendation, this is not good news for my whisky recommender system. Nevertheless, some users say that shopping recommendation could be useful in larger airports. This probably translates to that it would be useful in cases where many offers/services are available, such as urban areas.

3. When the users say they want either a higher or lower granularity of the context, this should not be a problem for the framework itself since it is flexible and can contain any degree of granularity. The problem lies more on how this is communicated to the user. A better designed GUI where the user could specify the granularity him- or herself could be one possible way to solve this problem.

My overall impression from the user acceptance testing is that most of the innovations of the AmbieSense project were appreciated by the test users. The biggest problem was probably that the test scenario at OSL was not big enough for the users to see the value of some of the information presented.

8.2. Multi-Agent System (recommendation)

So far we have looked at the testing done by the AmbieSense project, let us now look at the testing I did on the system I developed, and see what the findings can tell us. Let us start looking at the experience gathered when testing the MAS. The testing was divided into a component testing, an integration testing and a complete system testing, as described in chapter 5.5.2.

8.2.1. Component testing

The testing of the components did not show any problems except the extra code produced by the BeanGenerator, described in chapter 7.2.1. It is strange that the BeanGenerator creates code which relies on BeanGenerator libraries to work. The modularity is lost with this approach because other agents communicating with the ontology created by the BeanGenerator need to have BeanGenerator installed. Luckily, not many lines of code had to be removed.

Another interesting discovery when testing the components was JADE's built in support for testing components. Components can be tested using dummy agents which communicate with them. The dummy agents does not have to be programmed, but can easily be created from/with the JADE administration GUI.

8.2.2. Integration testing

For the integration testing the scenario described in the use case (figure 5.12) was used. To find out if the agents performed as planned, we can compare the use case (figure 5.12) which contains the planned scenario with the actual findings from chapter 7.2.2, figure 7.2; produced by the Sniffer Agent. When comparing them, we can see that they resemble each other. However, there are some differences between them; the use case figure 5.12 contains some alternatives where the actual performed scenario only contains one alternative. Another difference is that internal communication is not shown in the result made by the Sniffer Agent figure 7.2. By the similarity of the findings and the original design presented in the use case, we can conclude that the integration between the agents worked as planned.

Also for the integration testing dummy agents were used. But these were simple agents programmed in Java/JADE, rather than the built-in dummy agents started from the administration GUI.

The problems mentioned about asynchronicity can be compared with working with threading in Java. It is extra challenging to plan processes which are not working synchronously to each other. The Agent UML was a good tool for overcoming some of these problems where it provides possibilities to plan and model alternative actions. JADE also provides different modes for an agent to wait on other agents by using 'receiving block', which means that an agent waits until a message arrives or 'normal receive', where the agent does other activities while waiting for messages.

From the integration testing we have seen that the Sniffer Agent can be a valuable tool for evaluating the agent communication in the MAS. Dummy agents can be produced and used as placeholders for the actual agent which had not yet been developed. When designing agents, it is important to remember that they work asynchronous and are able to handle situations where messages are missing or too late.

8.2.3. Complete system testing

The complete system testing was divided into testing of the recommendation and testing of the MAS platform, described earlier in chapter 5.5.2.3.

8.2.3.1. Recommendation

The testing of the recommendation was to see if the system produced a recommendation as expected. The results can be found in chapter 7.2.1.

In the first scenario, figure 7.3; we can see that the system has Glenmorangie 10 years old as the preferred product and Glen Gardioch 15 years old, Balblair 16 years old and finally Oban 14 years old as recommended whiskies. Glenmorangie is in class c, and therefore, the system recommends other available whiskies in class c. It is easy to verify the result by cross-referencing the whiskies list (Appendix C) in class c with available whiskies listed in the Euroshop list (Appendix B).

In the second scenario, figure 7.4 is a little more advanced whereby the system has to first do a matching in the database to find the whisky the user prefers. Again, we can verify the result by looking at the list of whiskies (Appendix C), the only two whiskies with rating 85 are the Glenlivet and the Ardbeg. The whiskies recommended are either in class d or e since Glenlivet is class d and Ardbeg is class e.

As we can see, the system produces a recommendation as expected but it should be noted that the recommendation is really a basic database search rather than a sophisticated 'intelligent' prediction. As we remember from chapter 3.1.1, a system like this is classified as a raw retrieval system. However, the system is not a pure raw retrieval system. For the final ranking of the whiskies, manually selected recommendation is used where the score given by whisky experts are used for ranking the whisky. The matching process where the user can select some features he or she prefers is a simple kind of attribute-based recommendation. The system also has the ability to recommend whiskies that have not been seen before without any help from a human expert.

There are several reasons for why we ended up with a recommendation system as such, and not any of the more advanced recommender systems. By selecting one of the more advanced recommender systems a higher degree of personalisation could probably be achieved. The more advanced systems were person-to-person correlation, item-to-item correlation and attribute-based. The person-to-person and item-to-item system would be possible to implement since the context contains all the information such a system would need. The huge drawback with any of those systems is that they require a large number of users, therefore it would be difficult to gather enough test users for such an approach. A pure attribute-based approach would also be difficult because of the data required. For training the classification algorithm, at least 20 samples are needed per class. Let us say we wanted to build a system with two classes; 'interesting' and 'not interesting'. Since we

operate with two classes, the user needs to supply the system with 30 whiskies he or she likes and 30 whiskies he or she does not like. How many 'normal' people have tasted 60 different whiskies? People who have tasted 40 whiskies are probably experts themselves and would not be in need of a whisky recommender system. As we can see, a system with a high degree of personalisation would be difficult to construct based on the data available.

As we can see, the system is not really an advanced recommender system. It does use raw retrieval, manually selected recommendation and a simple version of attribute-based recommendation. But it does not contain advanced recommendation techniques like user-to-user correlation or an item-to-item correlation recommendation. Nevertheless, the system can give a recommendation which is not that basic if we take into account that the system can handle yet unseen whiskies, without any help from a human expert. The recommendations do also have a degree of personalisation. Whiskies selected are based on whiskies preferred by the user or by some feature specified by the user. The classification process itself does not contain any personalisation because the users' preferences are not part of the classification process.

8.2.3.2. MAS platform

The testing of the MAS platform was divided into two tests; running of agents on different JADE platforms and servers, as well as adding and replacing agents at runtime (see chapter 5.5.2.3). The findings were presented in chapter 7.2.3.2.

Running agents on different JADE platforms and servers worked well, but the administration when joining two platforms was both time consuming and awkward. To make the configuration easier the AMS and DF Agents should be default agents, and the only configuration would be to specify the correct IP-address for the remote platform. Another feature which could ease the administration would be a profile which stored the information, so that it easily could be retrieved later.

Adding and replacing agents at run-time worked well after each agent automatically were given the ability to deregister when 'killed'.

From the MAS platform testing we have seen that the JADE platform gives some interesting benefits like reliability, extensibility, computational efficiency and maintainability.

8.2.3.2.1. Reliability

An agent that fails can easily be replaced by another agent. Even platforms can be replaced if they fail due to a hardware or software failure.

8.2.3.2.2. Extensibility

If the problem requires it, additional agents can easily be added. Agents can also be moved to a more powerful server at runtime.

8.2.3.2.3. Computational efficiency

Another benefit when developing JADE agents is that the system developed is multi-threaded since each agent runs in a separate tread. This is going to be more interesting when computers with more than one processor arrive this year. However, most servers can benefit from this today.

8.2.3.2.4. Maintainability

The JADE framework gives extensive support for communication which saves time in development and ensures compatibility with other systems.

8.3. Mobile platform

The testing on the mobile platform was divided into three different tasks; compiling, distributing and running the application (see chapter 5.5.3). The findings was presented in chapter 7.3.

8.3.1. Compiling

Compiling the application did not prove to be difficult, but it was rather time consuming because of the configuration involved. The error encountered did also contribute to the time spent. The error is fixed in the version 3.2 of JADE.

Despite the time spent to understand and configure the compilation environment, it is impressive to see that an application programmed in Java can work on a mobile device without writing the application especially for that purpose. The programmer does not have to worry about which platform the application is going to run on until the deployment.

8.3.2. Distributing

The use of a web server for distributing the application proved to be a good idea. This made the application accessible and it required minimal from the user to install and run the application. The .jar file had to be downloaded to the user's mobile device and then double clicked to run. The user did not have to worry about the JADE platform at all, as it is embedded into the .jar file that was distributed. When running the application, it looks just like any other ordinary application.

8.3.3. PDA

Running of the Demo application was stranded because no free MIDP compatible VM was found for the particular PDA and OS. The search for a MIDP compliant VM for the PDA showed that Sun does not have the same strong position on mobile device as they have on stationary devices (see figure 7.8). On the mobile platform it is a variety of different VMs. Some support MIDP, while others have their own proprietary support of Java libraries like the SuperWaba VM. Another important issue is that most of the VM are licensed and not free for the user to use and download. This could actually mean that a user have to pay for running his or her own Java program. When this is said, most of today's PDAs have a pre-installed VM which supports either MIDP or J2SE.

8.3.4. Mobile phones

The Demo application could only run on one of the mobile phones that were tested. Problems running the application were either due to a lacking support of sockets (SonyEricsson) or a limitation of the maximum application size (Nokia series 40).

The overview presented in figure 7.10 shows that not only the phone I tested lacks socket support. Another thing to note is that all of the MIDP 2.0 phones have socket support, this is because the MIDP 2.0 standard requires socket support as mentioned earlier in chapter 5.5.3. When MIDP 2.0 becomes the dominant standard on mobile phones, then JADE could truly reach its potential on the mobile platform.

8.3.5. Emulators

As mentioned earlier in chapter 7.3.3.3, the Demo application worked on all emulators. However, this is not just positive. Even phones that were proved not to work in real life, worked on the emulator. For example, the Sony Ericsson T610 did work perfectly on the emulator, but real life testing showed that lack of socket support made it useless for running JADE. The same was also experienced using the Nokia emulator. The lesson to be learnt here is to always test applications on the actual phone the application is meant for.

Unfortunately, I was not able to run the application on any of the tried mobile devices. The problems encountered when trying to run the Demo application on a mobile device were not due to problems in the JADE framework, but rather maturity problems with the Java platform. When the MIDP 2.0 standard is used on all 'normal' mobile phones, hopefully these problems could be solved.

8.4. Classification

The classification testing consisted of an evaluation of the features and a 10 fold cross-validation, as described in chapter 6.

8.4.1. Feature evaluation

From figure 6.10 chapter 6.4.3.3 we can see a ranking of the 6 best features that was found, using the information gain algorithm. Some features are not in the list since they are directly dependent on other features. For instance, Region is in the list instead of Longitude and Latitude. Both Region and the combination of Longitude and Latitude describe the location of where a whisky has been produced. The use of all three features would provide redundant information which should be avoided because it means that those features are dependent of each other. A change in Region would necessarily lead to a change in either Longitude or Latitude. It is also important to note that Longitude and Latitude are independent but can be used together. Region is in the list because it scored higher than the combination of Longitude and Latitude. It is not easy to compare one feature against two features, so this result has to be handled with caution. In the 10 fold cross-validation testing both combinations are going to be tested despite the result from the feature evaluation algorithms.

From the information gain ranking figure 6.10, we can see that both the features Wood and Re-use scored 0. The features are in the list even tough they received a mediocre score of 0, because they scored better using the ReliefF algorithm (figure 6.11). The mediocre score when using the information gain algorithm is probably due to the fact that it lacks support for continuous features.

An interesting result was produced in run 9 (figure 6.12) where the two lowest ranked features from the information gain algorithm were used. The error rate decreased to 75.6% compared to using the highest ranked features in run 7 (84.1%) and run 8 (80.5%). The Re-use feature was actually ranked last by both feature evaluation algorithms, but in real life it proved to be a valuable feature.

From using the feature evaluation algorithms we have seen that they can provide valuable information about suitable features. However, it is important to be aware of the strengths and limitations using the different algorithms. These algorithms do not necessarily come up with all the best features, so it is still advisable to do some testing on different features. There are probably better features than the 6 listed here. But unfortunately, it was difficult to find the necessary data for other better features.

8.4.2. 10 fold cross-validation testing

The first test-runs produced some exceptional results with an error rate of only 23 %. For this testing, the samples were manually divided into a training and test set. Unfortunately, after some investigation it became apparent that the exceptional results were due to some samples that appeared in both the training and the test set. After this experience, a better approach using the 10 fold cross-validation testing provided by Weka as described in chapter 6 were applied.

There are several lessons to be learnt from the results presented in figure 6.12 in chapter 6.

- Information rich features
- Classifier
- Reduced classes
- Error rate
- Other features

8.4.2.1. Information rich features

One of the things that would be interesting to find out during the testing; if there is any difference using a feature containing much information or using several more atomic features joined together. In particular, it was tested with features describing the location with the possibility to use the feature; Region (Alternative 1) or a combination of the features Longitude and Latitude (Alternative 2). The whisky cask used for storage was either described by one feature; Cask (alternative 1) or by several features Wood, Re-use, Maturing and Finish (alternative 2).

First, let us have a look at the features describing the location. In run 1, Region was used for describing the location, and in run 3, Region was replaced by the features Longitude and Latitude. From these two runs, we can see that there is not much difference between these two combinations. Nevertheless, Region produced a better error rate of 80.5% compared to 81.7% when Longitude and Latitude were used. In run 11 and 12, this

difference is even more apparent giving the feature; Region an error rate of 72% compared to the error rate of 76.6% when the features Longitude and Latitude were used.

At least when combined with the features used in my testing, the feature Region proved to be the better feature for describing the location. This was further proven by the scores received from the feature evaluation algorithms.

Let us now look at the results from the features describing the cask. In run 1, Cask was used and in run 5, the features Wood, Re-use, Maturing and Finish were used. We can see that the error rate is equal in the two runs; 80.5%. However, as described earlier, the classification had further improved by using only some of the 4 features (Wood, Re-use); this can be seen in run 9, 11 and 12.

From the results it is not possible to draw any conclusion on whether it is better to use a broad information rich feature or several smaller combined features. Nevertheless, we can see that several small features provide greater flexibility where features containing irrelevant or redundant information can be removed.

8.4.2.2. Classifier

From the results in figure 6.12, it is easy to conclude that the Bayesian classifier did not work well with the data. This is probably because there is no obvious a priori information which could be used to increase the accuracy of the classification. From the testing, we can see that the build time was not a factor since it even with the slowest classifier was close to zero. Other classifiers could have been used to improve the accuracy of the classification. But before this is done, it is important to find more and better features.

Natural classifier candidates would have been the improved version of the NNR, the 5-NNR and maybe even a neural network. Both these algorithms can easily be tested using Weka. While working on the classification I have come up with an idea of making a specialised classifier for the duty-free scenario. The classifier would be an unsupervised classifier based on the NNR algorithm. A recommendation would simply be to recommend the nearest neighbour in vector space without any concern of the class label. The benefits with such an approach would be that whiskies would have to be labelled, and the user could prefer certain features before others. Ultimately, this could have resulted in a more personalised recommendation. On the downside, since the algorithm

sketched here is not in Weka, the algorithm would have to be programmed in Java. It could also be difficult to prove the quality of the recommendation.

8.4.2.3. Reduced classes

When we look at the results (figure 6.12), one apparent problem is that the error rate is ranging between 75 and 85%. This might seem quite high, but when we know that the error rate of selecting randomly among 10 classes is 90%, the result is not that bad after all. The main problem having to deal with 10 classes is that for each class, around 30 samples per class is required as explained earlier in chapter 6. This means that for 10 classes, 300 pre-classified whiskies are required for the system to work sufficiently. For my system, which contains only around 80 pre-classified whiskies, 3-4 classes would be more fitting. To further decrease the error rate, some classes were joined together. The classes were joined using classes described by Wishart (2002). He had done some classification where whiskies were divided into 4 and 6 classes. The results of this decrease in classes are shown in run 12 and 13; the numbers of classes in the two runs were 6 and 4 classes respectively. By decreasing the number of classes, the error rate was lowered to 68.3% when using 6 classes and 56.1% when using 4 classes. The result when using 4 classes proved to be 'good', especially when taking into consideration that the random error rate for 4 classes is 75%. What does it mean to reduce to 4 classes, is this only a good thing? When reducing the number of classes to 4, the granularity of the prediction is lowered. It is just like an expert of cars, he or she would be able to correctly identify cars into the country they were produced, rather than identifying the correct maker of the car. Since the classification is going to be used to make expert recommendations, it is important to keep both the granularity of the prediction and the quality of the prediction as high as possible. The natural compromise would be 6 classes. However, since the error rate produced by reducing the number of classes is still not what you would expect from an expert (error rate less than 10-20%), better features or a better classifier should be found before experimenting further with a reduced number of classes.

8.4.2.4. Other features

To further improve the result, could other features have been used? Let us have a look at the best run, run number 12, to see what information it might provide. From the confusion matrix we can see that the classes with highest error rate were the classes h, i

and j of, 87.5%, 83.3% and 83.3% respectively (The values are calculated by dividing the correct classified samples on the total samples in the class). Class b on the other hand had the lowest error rate of 62%. Are there any obvious features that could classify the last 3 classes better? This figure provided by Wishart (2002) might give some useful insight.

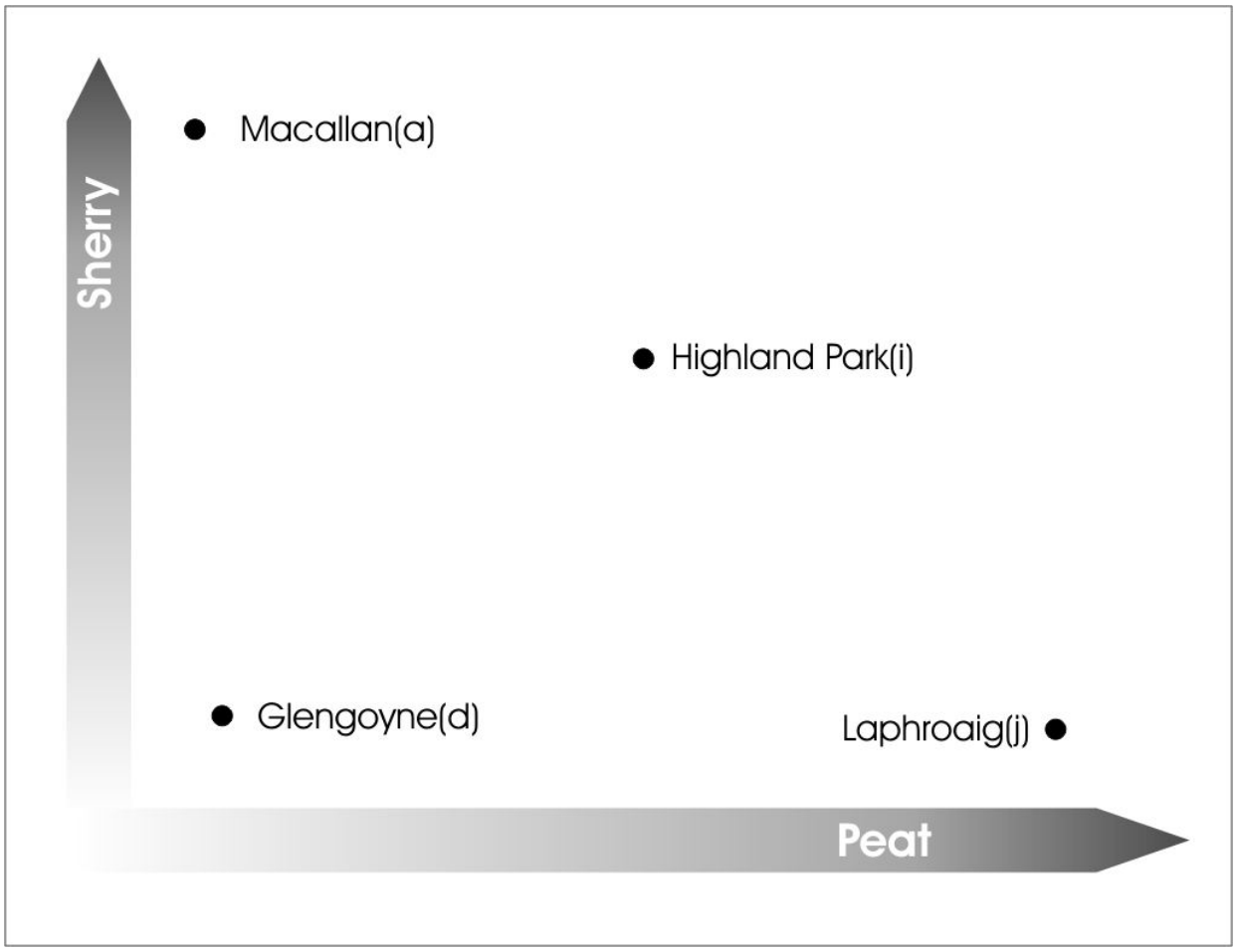


Figure 8.1: Whisky taste diagram (Wishart (2002) p. 37)

From figure 8.1, we can see that Macallan and Glengoyne have a taste with little peat, whereas Highland Park and Laphroaig have a taste where peat is highly present. When we compare this with the result from run 12 (figure 6.12), which was the test-run with lowest error rate (except run 13 and 14 where number of classes was reduced), we can see that Highland Park and Laphroaig are both members of classes which were difficult to

classify (Class i and j). This gives some indications that more features describing the peat content could produce a better classification.

In chapter 6, some features candidates containing information about peat content were mentioned; water character and the drying of the barley. Both these candidates should have been tested if sufficient information had been found. Personally, I believe that drying the barley over a peat fire has the most influence on the final taste since most of the distilleries known for their peaty whisky use this drying method.

From the testing of the classifier, we have seen that the accuracy is not as good as required for the system. Experiments with more features should be done to increase the accuracy of the classification. A better suited classifier, either a general classifier or a specialised classifier could have been tried. Finally, the accuracy can be further increased by reducing the number of classes.

We have also seen that the system is flexible when it comes to testing different features and classifiers.

8.5. Summary

The overall impression of the testing is that the system works as planned with the only exception of the mobile platform and the quality of the classification. The problems on the mobile platform would probably be solved with the new MIDP 2.0 standard, whereas the quality of the classification could probably be improved by using more and better features.

The system also serves as a framework connecting different AI techniques, such as agents, ontologies, knowledge base and classification. The framework proved to be flexible when it came to testing different features and classifiers. This is important since it is built to handle different duty-free products, which might require different features and classifiers.

The GUI was not tested because it did not have a high priority in the project. This does not mean that the GUI is not important. On the contrary, the reason why the GUI was left out is because it would require too much time and could easily represent a thesis itself.

Chapter IX

CONCLUSION

9.1. Scientific questions

I am now going through the scientific questions asked in chapter 1 to see if they can be answered by the thesis.

How can we make an MAS with 'intelligence' that can handle complex and advanced problems?

In chapter 7, I have shown that it is possible to make a rational MAS that can handle complex and advanced problems. The system developed serves as an application where users can specify duty-free shopping interests and get recommendations. Nevertheless, it also serves as a framework where duty-free products can be added and classified. To accomplish this, the system integrates different AI techniques such as agents, ontologies, knowledge base and classification.

Is an MAS better suited than a 'normal' computer program?

It have been demonstrated in chapter 7 that an MAS can give benefits compared to a normal computer program when it comes to reliability, extensibility, computational efficiency and maintainability. When the MIDP 2.0 standard is more widely used, the JADE platform can also provide benefits from its split-container ability as described in chapter 5.1.2.

How can we measure that an agent is good enough? Which methods can we use?

In chapter 6, I have shown different methods which can be used to measure the quality of a classification system. With the use of these methods, I have discovered that the classifier still needs some improvements to predict with the accuracy needed for such a system.

9.2. What have I learned?

Before writing this thesis, I had only theoretical knowledge about agents and even less knowledge about MAS. Hence, through the course of this project, I have gathered hands-on understanding on the subject of MAS. One important discovery was the benefits the agent framework gives the programmer, but also the new problems that emerged such as planning asynchronous activities.

Another thing that I have learnt was the difficulties of getting the data needed for the classification. This was a far bigger challenge than selecting the right features and classifier or programming the agents.

A side effect of this project was the increased knowledge about whiskies. Whether or not this is a positive side effect, it still remains to be seen.

9.3. Further work

The classifier has to be improved for making classifications as expected of an expert, either by finding better features or by selecting a different classifier. More duty-free products could also be added to the system. The GUI should be improved so that it is more user-friendly and could support advanced search in the KB using predicates.

The system I have created are not necessarily limited to duty-free shopping, but can be used for handling any shopping items, and could be used by any shop selling items.

On a bigger scale, the framework made by the AmbieSense project has a wider application than the airport scenario already demonstrated. For example, the system could be used in a museum, informing users about items in the room they are visiting.

BIBLIOGRAPHY

Aamodt, A. & Plaza, E. (1994) *Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches.* AI Communications. IOS Press[online], Vol. 7: 1, pp. 39-59. Available from: http://www.iiia.csic.es/People/enric/AICom.pdf [Accessed 12.06.2004]

Acklin (2004) *Acklin BV– BeanGenerator.* Available from: http://acklin.nl/page.php?id=34 [Accessed 07.08.2004]

Aitken, S. (2004) *Introduction to Case-Based Reasoning.* Artificial Intelligence Application Institute. Available from: http://www.aiai.ed.ac.uk/project/cbr/cbrintro.html [Accessed 2.06.2004]

Athanasiadis, I. Kehagias, D. Mitkas, P. A. & Symeonidis, A. L. (2003) *Application of Data Mining and Intelligent Agent Technologies to Concurrent Engineering.* Available from: http://agentacademy.iti.gr/pdf/mitaod.pdf [Accessed 05.12.2004]

Banan, M. (2000) *LEAP: One Alternative to WAP.* Available from: http://www.freeprotocols.org/LEAP/Manifesto/article/LEAP-OneAlternative/split/node3.html [Accessed 12.01.2005]

Bellifemine, F. Caire, G. Poggi, A. & Rimassa, G. (2003) *JADE – A White Paper.* [online]. Published in exp.telecomitalialab.com, Volume 3, 2003. Available from: http://exp.telecomitalialab.com/upload/articoli/V03N03Art01.pdf [Accsessed 10.01.2005]

Bellifemine, F. Caire, G. Rimassa, G. & Trucco, T. (2004) *JADE PROGRAMMER'S GUIDE JADE 3.2.* Available from: http://sharon.cselt.it/projects/jade/doc/programmersguide.pdf [Accessed 01.08.2004]

Breese, J. S. Heckerman, D. & Kadie, C. (1998) *Empirical Analysis of predictive algorithms for collaborative filtering.* Available from: http://citeseer.nj.nec.com/breese98empirical.html [Accsessed 10.03.2004]

Bressan, M. & Vitrià, J. (2002) *Independent Component Analysis and Naive Bayes Classification.* Available from: http://www.cvc.uab.es/~jordi/IASTED%202002%20-%20Malaga%20-%20NBICA.pdf [Accessed 06.06.2004]

Carnegie Mellon University (2001) *Multi-Agent Systems.* Available from: http://www-2.cs.cmu.edu/%7Esoftagents/multi.html [Accessed 15.01.2005]

Chen, H. (1995) *A map of Yahoo*. Available from: http://mappa.mundi.net/maps/maps_009/index.html [Accessed 03.04.2004]

Caire, G. (2003) *LEAP User Guide 3.1*. Available from: http://jade.tilab.com/doc/LEAPUserGuide.pdf [Accessed 10.03.2004]

Dai, H. and Mobasher, B. (2002) *Using Ontologies to Discover Domain-Level Web Usage Profiles*. Available from: http://km.aifb.uni-karlsruhe.de/semwebmine2002/papers/full/bamshad.pdf [Accessed 06.07.2004]

Dean, M. & Schreiber, G. (2004) *OWL – Web Ontology Language Reference*. W3C Recommendation. Available from: http://www.w3.org/TR/owl-ref/#Intro [Accessed 10.10.2004]

Devore, J. & Peck, R. (1990) *Introductory Statistics*. West Publishing Company. ISBN 0-314-56884-0.

Duda, R.O. Hart, P.E. & Stork, D.G. (2001) *Pattern Classification*. Second edition. John Wiley & Sons, Inc.

FIPA (1999) *Fipa Spec 18- 1999 - FIPA Content Language Library*. Available from: http://cyber.felk.cvut.cz/gerstner/dai/repository/docs/FIPA/fipa-99-18.pdf [Accessed 23.04.2004]

FIPA (2003) *About FIPA* Available from: http://www.fipa.org/about/index.html [Accessed 28.11.2004]

Flores-Mendez, R. A. (1999) *Towards a Standardization of Multi-Agent System Frameworks*. Available from: http://www.acm.org/crossroads/xrds5-4/multiagent.html [Accessed 14.03.2004]

Frank, E. Kaufmann, M. & Witten, I. H. (2000) *Data Mining: Practical machine learning tools with Java implementations*. San Francisco. Available from: http://www.cs.waikato.ac.nz/ml/weka/ [Accessed 05.10.2004]

Greiner, R. & Schaeffer, J. (January 2003) *Information Gain*. AIxploratorium: Decision Trees - page 4. Available from: http://www.cs.ualberta.ca/~aixplore/learning/DecisionTrees/InterArticle/4-DecisionTree.html [Accessed 11.04.2004]

Grimshaw, D. (July 2004) *JADE Administrative Tutorials*. Available from: http://jade.tilab.com/doc/tutorials/JADEAdmin/JadePlatformTutorial.html [Accessed 14.08.2004]

Gruber, T. R. (1993) *A Translation Approach to Portable Ontology Specifications*. Available from: http://ksl-web.stanford.edu/KSL_Abstracts/KSL-92-71.html [Accessed 12.01.2005]

Hall, M. A. (1999) *Feature selection for Discrete and Numeric Class Machine Learning*. Available from: http://citeseer.ist.psu.edu/cache/papers/cs/10392/http:zSzzSzwww.cs.waikato.ac.nzzSzmlzSzpublicationszSz1999zSz99MH-Feature-Select.pdf/hall99feature.pdf [Accessed 15.08.2004]

Harrison, I. (1997) *Case Based Reasoning*. Artificial Intelligent Application Institute. Available from: http://www.aiai.ed.ac.uk/links/cbr.html [Accessed 25.06.2004]

Helin, H. (February 2003) *Agent Communication*. Lecture notes. Available from: http://www.cs.helsinki.fi/u/hhelin/opetus/oat/ [Accessed 10.12.2004]

Helin, H. (September 2003) *Jade goes Wireless – Gearing up Agents for the Wireless Future*. [online]. Published in exp.telecomitalialab.com, Volume 3, 2003. Available from: http://exp.telecomitalialab.com/upload/articoli/V03N03Art03.pdf [Accessed 10.12.2004]

Huns, M. N. & Singh, M. P. (1997) *Conversational Agents*. [online]. Published in IEEE Internet Computing, April-March. Available from: http://www.csc.ncsu.edu/faculty/mpsingh/papers/columns/aow-1-2-97.pdf [Accessed 10.11.2004]

Java Community Process (1998) *Mobile Information Device Profile (JSR-37)*. Available from: http://www.jcp.org/aboutJava/communityprocess/final/jsr037/ [Accessed 23.04.2004]

Jimm Mobile SourceForge (2004) Available from: http://jimm.sourceforge.net/ [Accessed 22.04.2004]

Kohonen, T. (1984) *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, Heidelberg.

Kononenko, I. & Robnik-Sikonja, M. (1997) *An adaptation of Relief for attribute estimation in regression*. Available from: http://citeseer.ist.psu.edu/robnik-sikonja97adaptation.html [Accessed 19.10.2004]

Konstan, J.A. Schafer, J.B. & Riedl, J. (2001) *E-Commerce Recommendation Applications*. Available from: http://citeseer.nj.nec.com/schafer01ecommerce.html [Accessed 20.03.2004]

Konstan, J.A. Schafer, J.B. & Riedl, J. (1999) *Recommender Systems in E-Commerce*. Available from: http://www.grouplens.org/papers/pdf/ec-99.pdf [Accessed 20.03.2004]

Koua, E.L. (2003) *Using Self-Organizing Maps for Information*

Visualization and knowledge discovery. Available from: www.itc.nl/library/Papers_2003/art_proc/koua.pdf [Accessed 23.04.2004]

Kraaijeveld, A. (2001) *Celtic Spirits*. Available from: www.celticmalts.com/journal-a15.htm [Accessed 06.03.2004]

Lapointe, F. J. and Legendre, P (1994) *A Classification of Pure Malt Scotch Whiskies*. Available from: www.dcs.ed.ac.uk/home/jhb/whisky/lapointe/text.html [Accessed 12.03.2004]

Laurin, Urban (1998) *Whisky fra hele verden*. Oslo, Landbruksforlaget.

Lesser, V. (1995) *Multiagent systems: an emerging subdiscipline of AI*. Available from: http://portal.acm.org/citation.cfm?id=212121&coll=portal&dl=ACM&CFID=21100812&CFTOKEN=44506202 [Accessed 07.02.2004]

Maes, P. & Shardanand, U. (1995) *Social information filtering: Algorithms for automating "word of mouth"*. Proceedings of CHI'95 -- Human Factors in Computing Systems, 210-217. Available from: http://jolomo.net/ringo/chi-95-paper.pdf [Accessed 19.02.2004]

Manola, F. Miller, E. (2004) *RDF Primer*. Available from: http://www.w3c.org/TR/rdf-primer/ [Accessed 10.11.2004]

Miyahara, K. & Pazzani, M. (2000) *Collaborative Filtering with the Simple Bayesian Classifier*. Pacific Rim International Conference on Artificial Intelligence. p 679-689. Available from: http://citeseer.nj.nec.com/478266.html [Accessed 04.03.2004]

Murthy, K. V. S. (1997) *On Growing Better Decision Trees from Data*. Available from: http://www.tigr.org/~salzberg/murthy_thesis/survey/node16.html [Accessed 12.05.2004]

Myrhaug, H. I. (July 2001) *Towards Life-Long and Personal Context Spaces*. Workshop on User Modelling for Context-Aware Applications. Sonthofen, Germany. Available from: http://orgwis.gmd.de/~gross/um2001ws/papers/myrhaug.pdf [Accessed 03.01.2005]

Myrhaug, H. I. Whitehead, N. Goker, A. Faegri, T. E. Lech, T. C. (2004) *AmbieSense - A System and Reference Architecture for Personalised Context-Sensitive Information Services for Mobile Users*. Proceedings of the second European Symposium on Ambient Intelligence (EUSAI 2004) 2004: 327-338.

Norvig, P. & Russell, S. (2003) *Artificial Intelligence: A modern approach*. Second edition. Pearsons Education, Inc.

Noy, N. F. & McGuinness, D. L. (2001) *Ontology Development 101: A Guide to Creating Your First Ontology*. Available from: http://protege.stanford.edu/publications/ontology_development/ontology101.html [Accessed 17.04.2004]

Oard, D. W. & Marchionini, G. (1996) *A Conceptual Framework for Text Filtering*. Available from: http://www.ee.umd.edu/medlab/filter/papers/filter/filter.html [Accessed 15.10.2004]

Schank, R. (1982) *Dynamic Memory: A Theory of Reminding and Learning in People and Computers.* Cambridge University Press, 1982.

Shardanand, U (1994) *Social Information Filtering for Music Recommendation*. Master & Bach. thesis MIT. Available from: http://citeseer.nj.nec.com/shardanand94social.html [Accessed 21.02.2004]

Stone, P. & Veloso, M. (1997) *Multiagent Systems: A Survey from a Machine Learning Perspective.* Available from: http://www-2.cs.cmu.edu/afs/cs/usr/pstone/public/papers/97MAS-survey/revised-survey.html [Accessed 19.02.2004]

Sun Microsystems (2002) *The Mobile Information Device Profile.* Available from: http://java.sun.com/products/midp/midp-ds.pdf [Accessed 25.04.2004]

Sun Microsystems (2004a) *End Of Life Preannouncement.* Available from: http://java.sun.com/products/personaljava/index.jsp [Accessed: 25.04.2004]

Sun Microsystems (2004b) *What's new in MIDP 2.0.* Available from: http://java.sun.com/products/midp/whatsnew.html [Accessed 10.12.2004]

Sycara, K.P (1998) *Multiagent systems.* [online]. Published in AI Magazine. Available from: http://www.aaai.org/AITopics/html/multi.html [Accessed 13.02.2004]

Tatcmura, J. Santini, S. & Jain, R. (1999) *Social and content-based information filtering for a web graphics recommender system*. Available from: http://citeseer.nj.nec.com/189257.html [Accessed 04.04.2004]

Tognalli, E. (2004) *Re: [jade-develop] compiling JadeLeap 3.1 error.* Jade developers archive [online]. Available from: http://sharon.cselt.it/projects/jade/jade-develop-archive/0339.html [Accessed 18.03.2004]

Torres, R. Abel, M & Reategui, E. (2003) *Recommendation Frames: an Item-to-Item Approach to Recommender Systems*. Available from: http://www.inf.ufrgs.br/~rtorres/ijcai2003.pdf [Accessed 10.06.2004]

Wienhofen, L.W. M. Lech, C. T. Engels, R. H. P. Bremdal, B. A. (2004) *Intelligent, Personalised Agents For Mobile Use*. AmbieSense deliverable No.9.

Wilson, D.C. Leake, D.B. & Bramley, R. (2000) *Case-Based Recommender Components for Scientific Problem-Solving Environments.* Proceedings of the Sixteenth IMACS World Congress, 2000. 6 pages. In press. Avalable from: http://www.cs.indiana.edu/~leake/papers/p-00-01.pdf [Accessed 03.05.2004]

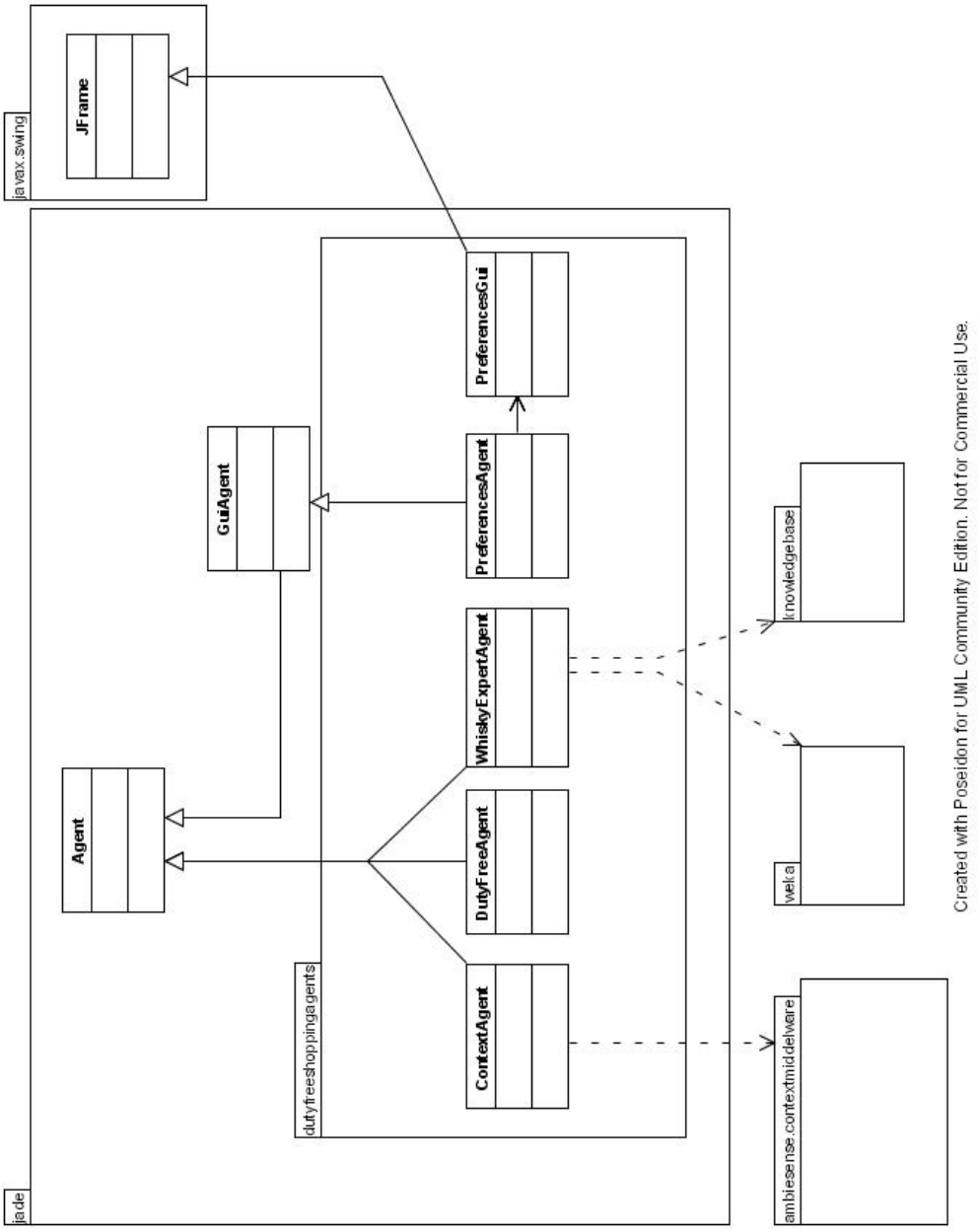
Wikipedia (2004) *Thomas Bayes*. Availabe from: http://en.wikipedia.org/wiki/Thomas_Bayes [Accessed 05.02.2004]

Wishart, D. (2002) *Whisky classified : choosing single malts by flavour*. London, Pavilion Books Limited. ISBN/ISSN: 1862055270.

Wooldridge, M. & Jennings, N. R. (1995) *Intelligent Agents: Theory and Practice*. Available from: http://citeseer.ist.psu.edu/rd/95695169%2C97055%2C1%2C0.25%2CDownload/http://citeseer.ist.psu.edu/cache/papers/cs/318/http:zSzzSzwww.shiratori.riec.tohoku.ac.jpzSz~kinozSzAg-Theor-Pract.pdf/wooldridge95intelligent.pdf [Accessed 05.05.2004]

APPENDIX

Appendix A – Duty-free Agent System (UML)



Appendix B – Euroshop list

Highland
Balblair Elements 100 179
A beautifully balanced, complex and satisfying whisky.
Dalmore Black Isle 12 YO 100 229
Fruity, spicy taste, slightly smoked.
Glen Garioch 15 YO 100 239
Smoky, sweet flavoured taste in between camphor and sandlewood
with smoky finish.
Glen Grant Highland Malt 100 199
A light, fruity, malty and somewhat oily taste.
Glenmorangie 10 YO 100 259
Slightly smoky, oily and creamy taste.
Glenmorangie Woodfinish Madeira 100 279
Spicy fresh , sweet citrus taste.
Glenmorangie Woodfinish Port 100 279
Butterscotch and dark chocolate taste, with fresh minty notes.
Glenmorangie Woodfinish Sherry 100 279
Full bodied sherry wine notes, traces of honey, nuts and hints of vanilla.
Johnnie Walker Pure Malt 100 259
Blending of superb single malts.
Slightly fruity with hints of cedar wood and honey
Oban Highland Distillers Edition 100 369
Montilla Fino cask wood gives sea-laced flavours beautifully enhanced
Old Pulteney 12 YO 100 189
A rather pungent, smoky and peaty taste.
Speyburn 10 YO 100 219
Fresh, slightly sweet and malty
Aberfeldy 12 YO 70 179
Malty sweetness, faint smoky peatiness.
Balblair 16 YO 70 289
Balanced and complex, sweet to a start, slightly dry.
BallantineÂs Pure Malt 70 209
Fresh,clean smoothness balanced by soft sweet orange,cinnamon and ginger
Islands
Highland Park 12 YO 100 239
Slightly salty, peaty and fruity taste.
Highland Park 18 YO 70 349
Heather-honey sweetness,smooth and round
Isle of Jura Legacy 10 YO 100 209
Fresh, peaty, sweet, slightly oily. Some saltiness.
Isle of Jura 16 YO 70 289
Mellow, smooth and less peaty than other island malts
Scapa 12 YO 100 219
A smooth, sweet and malty taste.
Fruity with notes of apple, vanilla and heather-honey.

Talisker Skye 10 YO 100 279
A sweetish seaweedy aroma with a pungent peaty ruggedness.
Talisker Skye Distillers Edition 100 369
Amorose cask wood amplifies the sweetness and the rugged,
spicy,peaty character.
Islay
Ardbeg 10 YO 100 289
Islay whisky, earthy,peaty,salty and robust.
Bowmore Darkest 75 359
Rich, smoky flavour, warm chocolate sweetness.
Bowmore Dusk 75 359
Superb balance between the finesse of France and the power
of Islay.Aromas of chocolate,roses,soft fruits and peat smoke
Bowmore Dawn 75 359
Sweet flavours of grapes och black plums together with the
characteristic smoky taste of Bowmore
Bowmore 12 YO 100 189
Smoky, lightish, burnt heather, tarry and chocolate taste.
Bowmore 12 YO PET 50 109
Smoky, lightish, burnt heather, tarry and chocolate taste.
Bowmore Cask Strength 100 249
Oak hidden with a burnt sugar sweetness.
Bowmore Mariner 15 YO 100 329
Sweety, fruity, medium peaty. Smooth and mellow.
Bunnahabhain 12 YO 100 209
Smooth and gentle, clean, nutty-malty sweetness.
Lagavulin Islay Distillers Edition 100 379
Pedro Ximinez cask wood gives sweetness to the intense peat flavour.
Laphroaig Cask Strength 10 YO 100 289
Rich peat smoke with some sweetness and strong hints of the sea.
Laphroaig Collection 3x33 100 319
Contains: Laphroaig 10YO, Laphroaig 10YO Cask Strength, Laphroaig 15YO
Laphroaig Islay 10 YO 100 219
A rich, sweet and gingery flavour with hint of oil, peat and tars.
Lowland
Auschentoshan 10 YO PET 50 149
A light soft taste with an orange-based fruity sweetness.
Glenkinchie Lowland 10 YO 100 249
Dry and smoky taste. Soft.
Glenkinchie Lowland Distillers Edition 100 339
Amontillado cask slightly nutty flavour augments the sweet and dry blend.
Auchentoshan 3 Wood 75 349
Fruit and syrup.Hazelnut, hints of cinammon and lemon. Sweet.
Speyside
Aberlour 15 YO 100 349
Nutty, spicy with sherry-accent.
Balvenie Doublewood 12 YO 100 209
Matured in two wood. Smooth and mellow single malt. Full bodied.
Balvenie Founders Reserve 10 YO 100 189
Smoky, mellow aroma. Honey notes.

Balvenie Mixed Assortment 60 479
Benriach 10 YO 100 189
Light balanced fruity flavours with soft sweetness.
Cardhu Malt 12 YO 100 269
Round and mellow, sweet with a delicate peatiness.
Cragganmore Speyside 12 YO 100 259
Good firm body, smokey finish. Pleasantly dry, delicate aroma.
Speyside
Cragganmore Speyside Distillers Edition 100 349
Port wine cask adds deep succulent notes to this Speyside malt.
Dalwhinnie Highland 15 YO 100 269
Light and aromatic with soft heather honey finish. Rich in body.
Dalwhinnie Highland Distillers Edition 100 359
Oloroso cask wood reflects and complements the peat and heather notes.
Glen Deveron 10 YO 100 219
A smooth, sweet flavoured taste in between camphor
and sandalwood. Salty finish.
Glen Keith 10 YO 100 189
Dry, thin taste with almond oil, some bitterness.
Glendronach 15 YO 100 249
Smooth, malty fine sherry character. Hint of smoke.
Glenfarclas 12 YO 100 199
Delightful fruit, oak and sweet taste.
Glenfiddich12 YO 100 179
A smooth, sweet, fruity and malty taste.
Glenfiddich 12 YO PET 50 99
A smooth, sweet, fruity and malty taste.
Glenfiddich Solera 15 YO 100 279
Smooth, delicate oak notes, great depth of flavour, long finish.
Glenfiddich Ancient Reserve 18 YO 70 329
Sea-air aromas, hints of cashew nuts, salty, lingering finish.
Glenfiddich Havana 21 YO 70 769
Matured in Havana Rum casks
Knockando 100 199
Perfectly balanced with a unique delicacy and fruitiness.
Longmorn 15 YO 100 199
A silky-sweet, nutty and well-balanced taste.
Strathisla 12 YO 100 199
A sweet, warm and fruity (apple) taste.
Tormore 12 YO 100 219
Well balanced,smooth single malt with honey taste
The Macallan Elegancy Vintage 100 259
100% matured in sherry oak casks. Light and sweet with
citrus, apple and toffee flavours.
The Macallan Fifties 50 219
Rich Macallan with strong sherry, resinous spice, dried fruits,
a touch of nuttiness, toffee and wood.
The Macallan Forties 50 219
Fresh apple fruit woody and slightly nutty.
The Macallan Thirties 50 219

Peat smoke, apples, citrus orange and spice.
The Macallan Twenties 50 219
Classic Macallan with sherries, spicy cloves, rich oily viscosity.
The Glenlivet 12 YO 100 209
Fruity, floral notes and creamy taste with a honeyed sweetness.
The Glenlivet 12 YO PET 50 99
Fruity, floral notes and creamy taste with a honeyed sweetness.
The Glenlivet French Oak 12YO 100 249
Rich floral and soft fruity flavours with some sweetness and spicy oakiness.
The Glenlivet 18 YO 100 379
Rather fiery, citrus - lime, sweet, dry, some cocoa in the finish.
The Glenlivet 15 YO 100 289
Soft, mellow fruity flavours, enchanced by sweet fragrance.
Whiskey De Luxe
BallantineÂs 12 YO Reserve 100 219
Lightly smoked flavour with a hint of the cask.
BallantineÂs 12 YO Reserve 50 119
Lightly smoked flavour with a hint of the cask.
Chivas Regal 12 YO 100 239
Lightly smoked, fruity and full flavour.
Chivas Regal 12 YO PET 50 129
Lightly smoked, fruity and full flavour.
Chivas Regal 18 YO 100 389
Smooth,soft,rich flavours with slight smokiness.
Chivas Royal Salute 21 YO 70 699
A well-balanced smokiness with a malty taste.
The Famous Grouse Gold Reserve 12 YO 100 219
A sweet, fruity and oily taste.
Johnnie Walker Black Label 12 YO 100 219
Full-bodied taste with vanilla flavour and somewhat smoky aftertaste.
Johnnie Walker Black Label PET 50 119
Full-bodied taste with vanilla flavour and somewhat smoky aftertaste.
Other Malt
The Famous Grouse Vintage Malt 12 YO 100 209
Created from the finest whiskies of a single yearÂs distillation
Discovery Malt Pack 3x33 100 249
Isle of Jura 10YO,Dalmore 12YO,Tamnavulin 12YO.
Scotch, Standard
Ballantine's Finest 100 169
A sweet and slightly smoky taste.
Ballantine's Finest PET 50 99
A sweet and slightly smoky taste.
Bell's 100 169
Lightly smoked flavour with hints of malt, and cask.
Grant's Finest 100 169
A full and fruity taste.
Grant's Finest PET 50 99
A full and fruity taste.
Grant's Super Strength 100 179
A full and fruity taste.

```
J&B   Rare    100   169
Slightly smoky, long, sweet taste.
Johnnie Walker Red Label     100   169
A dry, smoky and balanced taste.
Johnnie Walker Red Label PET    50   99
A dry, smoky and balanced taste.
The Famous Grouse    100   169
A sweeet, oily and full taste.
The Famous Grouse PET    50   99
A sweeet, oily and full taste.
The Famous Grouse Cask Strength    100   199
Upper Ten PET    50   89
```

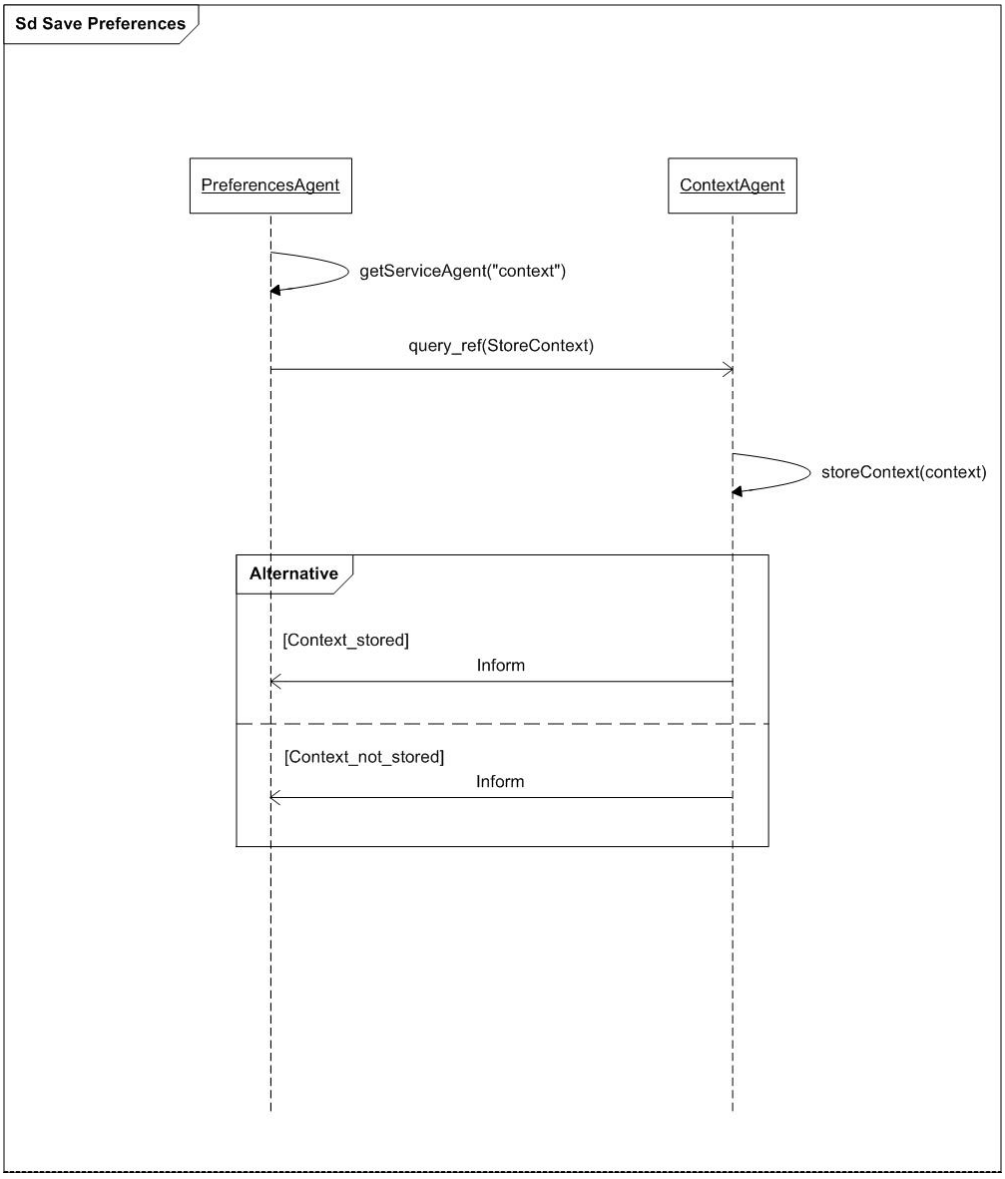
Appendix C – Whisky list

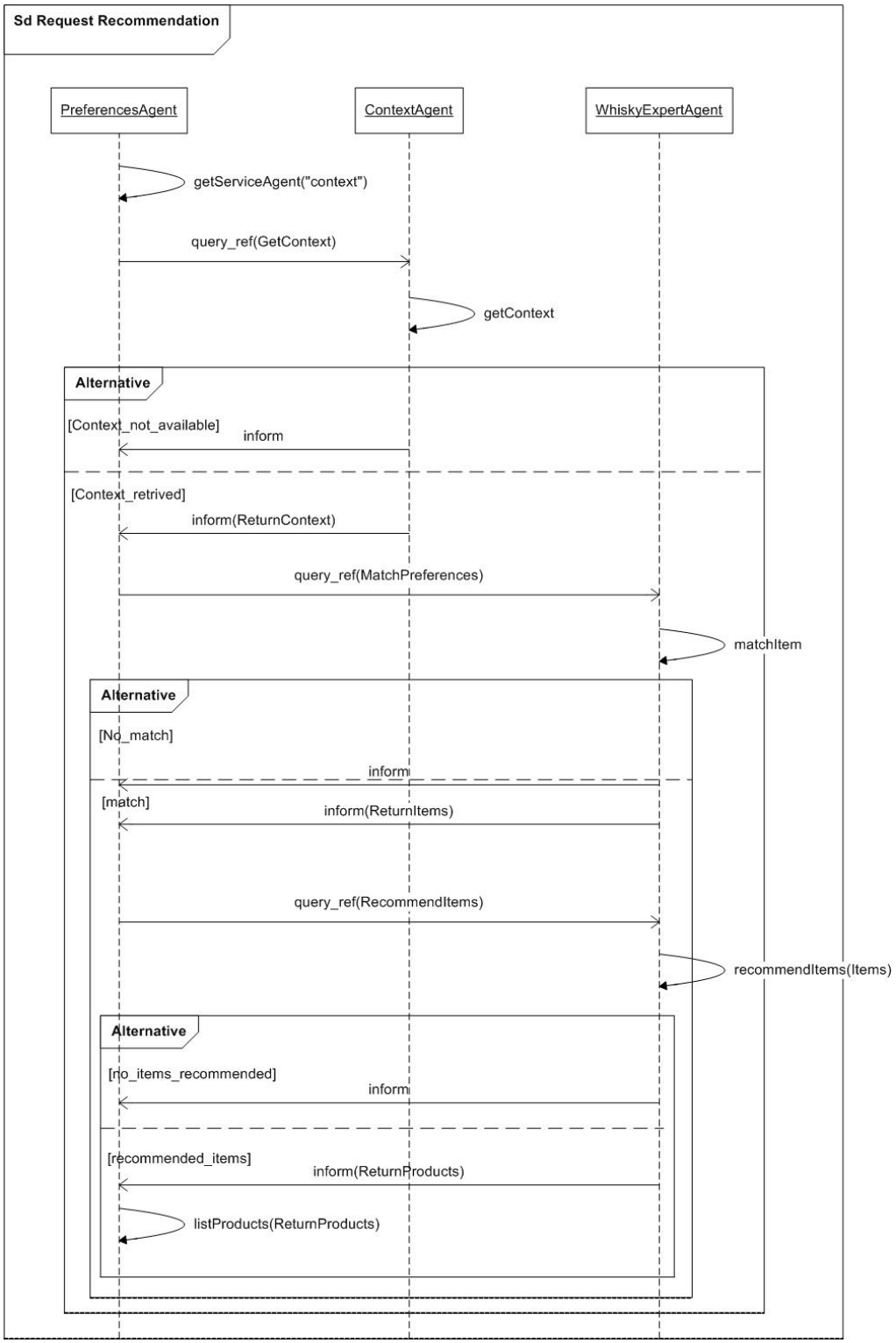
Brand	Age	Wood	Reuse	Maturing	Finish	Cask	WashBack	Class
Dailuaine	16	50	1	Bourbon	Sherry	Bourbon-Sherry	Wood	a
Dalmore	12	50	1	Bourbon	Sherry	Bourbon-Sherry	Wood	a
Glendronach	15	100	1	Sherry	None	Sherry	Wood	a
Macallan	12	100	1	Sherry	None	Sherry	Stainless-steel	a
Mortlach	16	100	1	Sherry	None	Sherry	Wood	a
Royal Lochnagar	12	10	1	Bourbon	Sherry	Bourbon-Sherry	Wood	a
Aberfeldy	12	50	1	Bourbon	Sherry	Bourbon-Sherry	Wood	b
Aberlour	10	50	1	Bourbon	Sherry	Bourbon-Sherry	Stainless steel	b
Ben nevis	10	50	1	Bourbon	Sherry	Bourbon-Sherry	Stainless steel	b
Benrinnes	15	50	1	Bourbon	Sherry	Bourbon-Sherry	Wood	b

Blair Athol	12	0	2,5	Bourbon	None	Bourbon-Refill	Stainless steel./Wood	b
Cragganmore	12	0	4	Whisky	None	Refill	Wood	b
Edradour	10	100	1	Sherry	None	Sherry	Wood	b
Glenfarclas	10	100	1,5	Sherry	None	Sherry-Refill	Wood	b
Glenturret	12	50	1	Bourbon	Sherry	Bourbon-Sherry	Wood	b
Knockando	12	10	2,5	Bourbon	Sherry	Bourbon-Sherry	Wood	b
Longmorn	15	50	1	Bourbon	Sherry	Bourbon-Sherry	Stainless-steel	b
Scapa	12	0	1	Bourbon	None	Bourbon	Wood	b
Strathisla	12	50	1	Bourbon	Sherry	Bourbon-Sherry	Wood	b
Balvenie Founder's Res.	10	30	1	Bourbon	Sherry	Bourbon-Sherry	Wood	c
Benriach	10	50	2	Bourbon	Sherry	Bourbon-Sherry	Stainless steel	c
Dalwhinnie	15	0	1	Bourbon	None	Bourbon	Wood	c
Glen Elgin	12	10	1	Bourbon	Sherry	Bourbon-Refill	Wood	c
Glen Ord	12	10	1	Bourbon	Sherry	Bourbon	Wood	c
Glendullan	12	0	4	Whisky	None	Refill	Wood	c
Glenlivet	12	5	2,5	Whisky	Sherry	Bourbon-Sherry	Wood	c
Linkwood	12	0	4	Whisky	None	Refill	Wood	c
Royal Brackla	10	0	1	Bourbon	None	Bourbon	Wood	c
An Cnoc	12	0	4	Whisky	None	Refill	Wood	d
Auchentosan	10	50	1	Bourbon	Sherry	Bourbon-Sherry	Wood	d
Aultmore	12	0	1	Bourbon	None	Bourbon	Wood	d
Cardhu	12	0	1	Bourbon	None	Bourbon	Wood	d
Glen Grant	10	50	1	Bourbon	Sherry	Bourbon-Sherry	Wood	d
Glengoyne	10	100	3	Sherry	None	Sherry-Refill	Wood	d
Mannochmore	12	10	2	Bourbon	Sherry	Bourbon-Sherry	Wood	d
Tamdhu	8	40	2	Bourbon	Sherry	Bourbon-Sherry	Wood	d
Tobermory	10	50	1	Bourbon	Sherry	Bourbon-Sherry	Wood	d
Bladnoch	10	50	1	Bourbon	Sherry	Bourbon-Sherry	Wood	e
Bunnahabhain	12	10	1	Bourbon	Sherry	Bourbon-Sherry	Wood	e
Glen Moray	12	5	1	Bourbon	white-wine	Bourbon	Stainless-steel	e
Glenallachie	12	0	2,5	Bourbon	None	Bourbon-Refill	Stainless-Steel	e
Glenkinchie	10	0	4	Whisky	None	Refill	Wood	e
Glenlossie	10	0	1	Bourbon	None	Bourbon	Stainless-steel	e
Inchgower	14	0	1	Bourbon	None	Bourbon	Wood	e
Tomintoul	10	5	1	Bourbon	Sherry	Bourbon-Refill	Stainless-steel	e
Ardbeg	10	0	1,5	Bourbon	None	Bourbon-Refill	Wood	f
Ardmore	11	0	1,5	Bourbon	None	Bourbon-Refill	Wood	f
Auchroisk(The singleton)	10	10	1	Bourbon	Sherry	Bourbon-Sherry	Stainless steel	f
Deanston	12	50	2,5	Bourbon	Sherry	Sherry-Refill	Stainless-steel	f
Glen Deveron	10	50	2,5	Bourbon	Sherry	Bourbon-Sherry	Stainless-steel	f
Glen Keith	10	10	2	Bourbon	Sherry	Bourbon-Sherry	Wood	f

Glenrothes 1989	12	50	1	Bourbon	Sherry	Bourbon-Sherry	Wood	f
Old Fettercairn	10	40	2,5	Bourbon	Sherry	Bourbon-Sherry	Wood	f
Tomatin	10	5	1	Bourbon	Sherry	Bourbon	Stainless-steel	f
Tormore	10	0	2,5	Bourbon	None	Bourbon-Refill	Stainless-steel	f
Tullibardine	10	5	1	Bourbon	Sherry	Bourbon	Wood	f
Arran	8	50	2	Bourbon	Sherry	Bourbon-Sherry	Wood	g
Dufftown	15	50	1	Bourbon	Sherry	Bourbon-Sherry	Stainless-steel	g
Glen Spey	12	0	2,5	Bourbon	None	Bourbon-Refill	Stainless-steel	g
Glenfiddich	12	0	1	Bourbon	None	Bourbon	Wood	g
Miltonduff	12	0	2,5	Bourbon	None	Bourbon-Refill	Stainless-steel	g
Speyburn	10	0	1	Bourbon	None	Bourbon	Wood?	g
Balblair	16	0	1	Bourbon	None	Bourbon	Wood	h
Craigellachie	14	5	1	Bourbon	Sherry	Bourbon	Wood	h
Glen Garioch	15	50	1	Bourbon	Sherry	Bourbon-Sherry	Stainless-Steel	h
Glenmorangie	10	0	1,5	Bourbon	None	Bourbon-Refill	Wood	h
Oban	14	0	1	Bourbon	None	Bourbon	Wood	h
Old Pulteney	12	10	1	Bourbon	Sherry	Bourbon-Sherry	Stainless-steel	h
Strathmill	12	0	2,5	Bourbon	None	Bourbon-Refill	Stainless-steel	h
Tamnavulin	12	0	2,5	Bourbon	None	Bourbon-Refill	Stainless-steel	h
Teaninich	10	50	1	Bourbon	Sherry	Bourbon-Sherry	Stainless-steel?	h
Bowmore	12	30	1	Bourbon	Sherry	Bourbon-Sherry	Wood	i
Bruichladdich	10	60	1	Bourbon	Sherry	Bourbon-Sherry	Wood	i
Glen Scotia	14	50	2	Bourbon	Sherry	Bourbon-Sherry	Stainless-steel	i
Highland Park	12	50	1	Bourbon	Sherry	Bourbon-Sherry	Wood	i
Isle of Jura	10	0	4	Whisky	None	Refill	Wood	i
Springbank	10	30	1	Bourbon	Sherry	Bourbon-Sherry	Wood	i
Caol Ila	15	50	1	Bourbon	Sherry	Bourbon-Sherry	Wood	j
Clynelish	14	0	1	Bourbon	None	Bourbon	Wood	j
Lagavulin	16	10	1	Bourbon	Sherry	Sherry-Refill	Wood	j
Laphroaig	10	0	1	Bourbon	None	Bourbon	Stainless-steel	j
Talisker	10	0	1	Bourbon	None	Bourbon	Wood	j

Appendix D – Use case




Sd Request Recommendation
PreferencesAgent
ContextAgent
WhiskyExpertAgent
getServiceAgent("context")
query_ref(GetContext)
getContext
Alternative
[Context_not_available]
inform
[Context_retrived]
inform(ReturnContext)
query_ref(MatchPreferences)
matchItem
Alternative
[No_match]
inform
[match]
inform(ReturnItems)
query_ref(RecommendItems)
recommendItems(Items)
Alternative
[no_items_recommended]
inform
[recommended_items]
inform(ReturnProducts)
listProducts(ReturnProducts)