# Automatic Construction of Virtual Technical Documentation

A Thesis submitted

to Cardiff University

for the degree of

Doctor of Philosophy

by

**Charilaos Pasantonopoulos BEng MSc**

Manufacturing Engineering Centre,

Cardiff University,

United Kingdom

2005

## *Synopsis*

The main objective of the research reported in this thesis is the generation of intelligent documentation of complex manufactured products. The construction of documentation is a major part of product support that increases the competitiveness of the product and its effective and proper use during its life-time. At the same time, it is also a complicated and time-consuming task occupying highly trained personnel. The need for systems that increase the productivity of documentation authoring teams becomes even more apparent as other design and manufacturing activities progress towards their automation and integration through the various CAD/CAM systems.

The first target of this research is to provide a system to assist the integration of the documentation authoring to the Product Data Management System, the primary source of product data. The second target is to increase the efficiency of the information systems used for this purpose and automate the process to the extent that is feasible using the available tools and ideas that are proposed in this thesis.

The first contribution presented in this thesis is an Object-Oriented framework that aims to assist software developers in the design and implementation of electronic documentation systems.

The second contribution is a novel distributed architecture for an intelligent documentation system that will allow the automation of a major part of the authoring procedure and the generation of electronic manuals based on information reused from the Product Data Management System.

The third contribution is a new technique for the generation of virtual documents. The technique is rule-based so as to support the decision-oriented nature of data selection within the authoring procedure. The technique follows knowledge-based principles allowing authors to design documents at a higher level of abstraction.

## ***Acknowledgements***

I would like to thank my supervisor Prof. D.T. Pham for giving me this opportunity to conduct research in his laboratory.

I would also like to thank the Intelligent Information Systems group which helped with their feedback and support throughout these years.

I must thank everyone at the Manufacturing Engineering Centre for the various times they helped.

Finally I want to thank my family for tolerating me all these years.

# *Declaration*

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed [illegible] ( C.Pasantonopoulos - Candidate )

Date 4/2/2005

Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by giving explicit references. A bibliography is appended.

Signed [illegible] ( C.Pasantonopoulos - Candidate )

Date 4/2/2005

Statement 3

I hereby give consent for my thesis, if accepted, to by available for photocopying and for inter-library loan and for the title and synopsis to be available to outside organisations.

Signed [illegible] (C.Pasantonopoulos - Candidate )

Date 4/2/2005

# *Table of Contents*

## *List of Figures*

## *List of Tables*

## *Abbreviations*

| | |
|---|---|
| ACC | Agent Communication Channel |
| ACL | Agent Communication Language |
| AI | Artificial Intelligence |
| AMS | Agent Management Service |
| AOP | Agent-Oriented Programming |
| AP | Application Protocol |
| API | Application Programming Interface |
| BIG | resource-Bounded Information Gathering |
| CAD | Computer Aided Design |
| CAM | Computer Aided Manufacturing |
| CE | Concurrent Engineering |
| CGI | Common Gateway Interface |
| CLIPS | C Language Integrated Production System |
| DBMS | Data Base Management System |
| DF | Directory Facilitator |
| DME | Device Modelling Environment |
| DOI | Digital Information Identifier |
| DOM | Document Object Model |
| DSSSL | Document Style Semantics and Specification Language |
| DTC | Design To Criteria Scheduler |
| DTD | Document Type Definition |
| EE | Extended Enterprise |

| | |
|---|---|
| FIPA | Foundation for Intelligent Physical Agents |
| HTML | HyperText Markup Language |
| HyTime | Hypermedia/Time-based structuring language |
| IA | Information Agent |
| IB | Information Brick |
| IE | Information Element |
| IIO | Intelligent Information Object |
| IO | Information Object |
| IPM | Intelligent Product Manual |
| JDBC | Java Data Base Connectivity |
| JESS | Java Expert System Shell |
| JSP | JavaServer Pages |
| MIT | Massachusetts Institute of Technology |
| NL | Natural Language |
| NLG | Natural Language Generation |
| OO | Object-Oriented |
| OOAD | Object-Oriented Analysis and Design |
| OOP | Object-Oriented Programming |
| OQL | Object Query Language |
| OS | Operating System |
| PDF | Portable Document Format |
| PSS | Product Support System |
| PDM | Product Data Management |
| PHP | Hypertext PreProcessor |
| RETSINA | Reusable Environment for Task-Structured Intelligent |

|  |  |
|---|---|
|  | Networked Agents |
| RIO | Reuse of Information Objects |
| SAX | Simple API for XML Parsing |
| SGML | Standard Generalised Markup Language |
| SQL | Standard Query Language |
| STEP | Standard for the Exchange of Product model data |
| VD | Virtual Document |
| VTDS | Virtual Technical Documentation System |
| VTM | Virtual Technical Manual |
| XML | eXtensible Markup Language |

# Chapter 1 – Introduction

In this chapter, a brief introduction to this thesis is given. The chapter continues with a description of the motivation and objectives for this research and concludes with an outline of the thesis consisting of a short description of the contents of each chapter and its purpose.

# 1 Introduction

Today's requirements for "complete solutions" and not mere products have leveraged the role of product manuals. These complete solutions are a combination of products and services. The role of the company in this new strategy generates the need for basic and custom services. This role places the documentation within the basic services, as part of the product support, and within the custom services, as part of the product information used internally within an enterprise for the product customisation. Within the tight customer-provider relationship and the new relationships created by the Extended Enterprise (EE) model [Haake01] between the partnerships, product information has emerged as an ingredient that requires more flexibility, transportability, and intelligence than ever before.

An Intelligent Product Manual (IPM) is a conceptual model for a product manual that goes beyond the conventional manuals to intelligent electronic Web-based forms. A development methodology for the IPM model, an adaptive IPM architecture that challenges the problems of user-adaptation, and a case-based reasoning approach for adaptive hypertext have also been developed. Furthermore, utilising the Concurrent Engineering (CE) approach for IPM authoring, an integrated authoring environment that enables the CE approach to be applied to the documentation development was constructed, and a collaborative authoring environment was implemented [Setchi99, Pham02]. Finally, the integration of expert systems in IPMs, as a troubleshooting facility, led to an agent-based expert system which automatically updates its fault knowledge, and automatically classifies new fault data provided by the maintenance engineers [Soroka00].

Modern approaches to documentation systems have successfully incorporated hypertext, multimedia, adaptive hypertext, and personalised interfaces. Automatic generation of the documentation has been extensively exploited only in the realm of source code documentation using literate programming techniques [Knuth84], mainly by the JavaDoc [Friendly95] system. However, in the generic sense of documentation, automation has not been yet utilised, except in minor research projects. Stepping into the area of automatic generation allows viewing product manuals as Virtual Documents (VD) [Gruber95]. The concept of "virtual documentation" can promote manuals to a new level of existence.

Other trends include the use of Artificial Intelligent techniques, such as Intelligent Agent systems and Information Agents that can be deployed in the area of documentation for helping in the various stages of design, construction and maintenance, or even user adaptation.

## 1.1 Motivation

As has been established, research in Intelligent Product Manuals [Pham99, Pham00] has been striving to extend the reality of product documentation to meet the current needs and technological trends in today's manufacturing.

The main inspiration for this research comes from four subjects: Object-Oriented systems, Information Objects, Virtual Documents, and the current status of software documentation.

The documentation systems for software frameworks and components, being closer to information technology developments, have strongly benefited over the last few years from literate programming techniques and Java's JavaDoc toolkit [Javadoc04].

Technical documentation of large complicated products has the disadvantage of lacking a strictly formalised framework, which programming languages possess. This makes the process of producing such documentation much more complicated and reduces the degree of automation that can be applied. Thus, a way of handling the structure and the meaning of the product data is needed. The data have to be handled explicitly, in order to impose on the arbitrary product a formality, a set of rules and strategies that will enable more efficient handling of the product information. As a result, it is possible to automate partially the construction of the documentation as in the case of software documentation.

Virtual Documents can be very useful tools for the generation of technical documentation providing integration with the most up-to-date data and user specific document generation. According to Gruber et al [Gruber97], one drawback that is encountered is the modelling bottleneck. In order for the author to design virtual documents, he/she has to prepare models, which can be a very complicated and time-consuming task.

## 1.2 Research Objectives

The main objectives of this research are the following.

1. The Identification of today's Technical Documentation requirements. (Chapter 2 )
2. The creation of an Object-Oriented Framework that will increase the productivity of the software developers involved in documentation system projects. (Chapter 3)
3. An analysis of the Information Object (IO) concept, and an evaluation of its benefits and inefficiencies within the documentation authoring process. (Chapter 4)
4. The exploration of the Information Object as an active entity and its realisation as an Autonomous Agent that provides an Active and Intelligent Information Object (IIO). (Chapter 4)
5. Examination of the advantages of the novel IIO-based architecture for the IPMs. (Chapter 4)
6. The introduction of a technique for the generation of Virtual Documents that has a focus on Technical Documentation and offers better productivity and ease of encoding. (Chapter 5)
7. Creation of an ontology for Virtual Technical Documentation Systems (VTDS). (Chapter 5)
8. Construction and presentation of case studies of automatically generated Virtual Technical Manuals (VTM) based on the findings. (Chapters 3,4,5)
9. Discussion of the possible extensions and shortcomings of this research and findings. (Chapter 6)

## 1.3 Thesis Outline

The remainder of this Thesis comprises five chapters.

Chapter 2 surveys the literature relevant to the field of research. The subjects of Virtual Documents, Information Objects, and Autonomous Agents are reviewed in order to clarify the benefits that they intrinsically provide. The different approaches are criticised and a good understanding is reached that yields an effective combination of ideas and technologies that will meet the expectations of the research objectives.

Chapter 3 presents an Object-Oriented framework for Virtual Documentation. The basic requirements and principles of a documentation system are modelled into an Object-Oriented design to provide an infrastructure for producing documentation systems. A web-based documentation application is presented that generates a dynamically updated virtual web-site.

Chapter 4 is concerned with Intelligent Information Objects. A novel architecture for IPMs is proposed based on the Intelligent Information Object (IIO) concept. This new concept encompasses aspects from autonomous agents and the concept of information objects. An IIO forms a concept-specific information agent that offers several benefits when related to technical documentation, as illustrated in the rest of the chapter.

Chapter 5 describes a rule-based approach to virtual document generation. The focus is on the description of Virtual Documents and on a criticism of the current approaches. A novel approach using knowledge-based techniques is proposed, which aims to address the problems concerning the description of Virtual Documents.

Chapter 6 concludes the thesis. The results of this research are shown in the form of distinct contributions and the final conclusions are stated. A vision for the future of the presented systems is also given in this chapter.

# Chapter 2 – Literature Review

In this chapter, the literature on the topics of Product Documentation, Information Objects, Virtual Documents, Autonomous Agents, and Rule-Based Systems is presented. At first, current work in Product Support and documentation is shown. Next, the concept of Information Objects is analysed. Then, the different systems proposed for Virtual Documents are described. Consequently, Autonomous Agents are studied with an emphasis on those that are focused on information acquisition and synthesis. Finally, Rule-Based Systems are shown as a way of realising an intelligent system that could fulfil the needs of intelligent behaviour within the documentation authoring procedures.

# 2 Literature Review

## 2.1 Product Support and Documentation

The objective of performance support systems (PSS) is to enhance the performance of users supporting them in their daily work activities and tasks for a specific product. Bezanson [Bezanson95] defines performance support as "a product or process attribute that aims to enhance user performance, through a user interface and support environment that anticipates user needs and supports them conveniently and effectively". According to Cantanto [Cantanto96], PSSs are "integrated, readily available sets of tools that help individuals do their job and increase their productivity with minimal support". Desmarais et al. [Desmarais97] and Sleight [Sleight93] identify as the primary objective of PSSs the support of the users with specific tasks related to the product. Bezanson [Bezanson95] outlines also the learning aspects of support systems by providing "just in time support" and giving the users the ability to retrieve information at the moment it is needed.

By the use of performance support systems the value of the manufactured product offered increases. The users find it easier to work with the supported product, the companies achieve more effective usage of the product and thus increased productivity, and in the same time they can reduce costs for training and hiring expert users for supporting, maintaining, and operating their infrastructure. Desmarais et al. [Desmarais97] shows claims of a number of companies that have achieved large gains by the use of PSSs. This also counterbalances the costs of investment for the development of such a system.

### 2.1.1 Traditional Product Support

Traditionally, product support includes the paper-based product manuals, lectures, courses from expert instructors, consultation from representatives of the company etc. Other aids are known such as pocket reference cards, colour codes, lists of abbreviations, and specification sheets. These can be thought to be adequate for some products but as the product complexity and the volume of information increases problems start to appear and their effectiveness is reduced. According to Bezanson [Bezanson95], "traditional user support methods are no longer effective", because "traditional training methods are not responsive to the individual needs as they emphasize training rather than learning". Furthermore, according to Pham et al. [Pham99] problems arise with the collection, integration, and retrieval of product information, the book oriented form of the documents, the reference manual (as opposed to educational) structure, and the presentation formats that may be unavailable or non-suitable under different conditions. Ventura [Venture00] focuses more on the problems associated with portability, complexity, accuracy, reliability, and maintainability of the product information presented to the user.

### 2.1.2 Advanced Product Support

Advanced product support employs personal computers, web technologies, and artificial intelligence techniques to enhance the product support and limit the role of costly human experts. According to Cantando [Cantando96], the purpose of an electronic PSS is to replace or supplement human experts, paper-based documentation, and costly training programs. Sleight [Sleight93] defines five characteristics for electronic PSSs. Namely that they are computer-based, provide access to discrete and specific task-related information during task performance, are used on the job, are controlled by the user, and reduce the need for prior training. Advanced PSSs can include help systems, front end to information databases, expert systems and knowledge bases, application and productivity software, learning experiences, assessment feedback and monitoring [Cantando96].

Performance Support Systems may be stand-alone or embedded. In the case of embedded solutions the PSS works in accordance with the main system as in the case of wizards provided along with popular software products. The embedded system complements and interacts with the main system. In the stand-alone form PSSs can be used on their own independently for learning separately from the system they are intended to support.

Intelligent Product Manuals [Pham00] have started a long line of research, advancing product specific and documentation centred performance support issues. Intelligent Product Manuals (IPM) have produced Electronic Documentation solutions resulting in Portable Data Format (PDF), HyperText Mark-up Language (HTML), and eXtensible Mark-up Language (XML) format documentation systems. Setchi has presented an Integrated Authoring Environment [Setchi99] for Intelligent Product Manuals accompanied by an authoring methodology. This allows the authors to operate on top of Pro-Intralink and allows the documentation authors to use the utilities offered by Pro-Intralink and author the documentation along with the product data and finally present them as a web site. The contributions continue with an Adaptive Product Documentation [Setchi99] system that uses user models and CBR (Case-Based Reasoning). It presents user-tailored documentation to the users that increases the productivity and the effectiveness of the manual. Finally, Soroka [Soroka00] developed an agent-based system that performs automatic updating of the fault data used by an expert system for supporting troubleshooting procedures performed by technical personnel dealing with a manufactured product.

### 2.1.3 Efforts on Automatic Documentation Generation

The main efforts in the automatic generation of documentation are focused on software engineering. They start with the Literate Programming paradigm and they are extensively utilised in the Java documentation system JavaDoc. In order to achieve automation of the process a method is needed that will integrate the design procedure and the documentation authoring and a system that will exploit this integration to extract, process and produce the documentation output. The main drawback in this approach is the transfer of the authoring as part of the designer workload and the increase of the complexity of the design as a process.

Literate Programming [Knuth84] started promoting a methodology and a system for producing software applications and their documentation with an integrated strategy where the documentation is written as part of the design process and is then extracted and formatted by a separate software system. This changes the traditional attitude towards the design of programs, transferring the focus from explaining to the computer what should be done, to explaining to other humans what the computer has to do. The designer can then be viewed as an essayist writing a report about the program in human terms and including parts of programming language code as a consequence of the concepts and ideas explained therein. This tries to impose a different approach to the design in that the primary target is the explanation and the code produced is a product that closely follows this explanation. At the end of the process the human language explanation is used to automatically produce documentation, and the code fragments are assembled to produce the actual design implementation.

The Literate Programming paradigm could be seen as a largely ambitious plan. Sun Microsystems created JavaDoc [Friendly95], the Java Documentation system that is strongly influenced by the literate programming paradigm but has more modest targets since it has to be a commercially viable solution. In the Literate programming paradigm the author/designer is focused on the high level description of the design and the computer produces the implementation of the design from the literate description. The goal of JavaDoc is to take a design expressed in the usual way and produce a quality literate description. The Java documentation tool automatically generates cross-references, indexes, and outputs Document Type Definition (DTD) compliant HTML code. In the Literate Programming paradigm the purpose is to actually produce the design, so the literate description is going into much more detail within the method implementations of the program. The JavaDoc toll targets to produce Application Programming Interface (API) documentation and thus eliminates the need for that level of complexity. The documentation in this case focuses on the choice of elements to be extracted and the commenting syntax. These two elements are further processed to produce the cross-referencing and the indexes, resulting in a fully web-enabled documentation.

Another approach to the creation of technical documentation is that of Automatic Generation, based on Natural Language Generation (NLG) by Reiter et al [Reiter95].

Reiter et al comment on the advantages of NLG such as, reduced cost to generate and maintain documentation, guaranteed consistency between documentation and design, conformance to standards, multi-linguality, user model based tailoring of documentation, and multimodality. As in the case of JavaDoc where additional work is needed from the designer to add the comments following the specified syntax, again in this case the designers are expected to add annotations in the CAD files and/or additional information through some knowledge representation system. In the NLG approach there is a fixed overhead for producing the knowledge base for the language generation itself. Additional work is needed to make the linguistic structures, vocabulary etc. This can be limited by using standards such as the European Association of Aerospace Industries (AECMA) Simplified English that is popular in the aerospace industry. Another problem that arises is the Quality Assurance of the documentation. As in the case of Literate Programming, where a more ambitious higher level documentation is targeted, the intelligence of the system compromises the predictability and thus may introduce errors in the generated document. In many cases this is not acceptable since it might induce severe hazards. Computation time can also be a problem in automatic documentation systems since the generation time and thus the response time of the system has to be in the magnitude of a few seconds in order for the system to be useful.

In conclusion, the Literate Programming approach may be too ambitious but it makes a valuable point. It emphasises the need for the designer to be conscious of his involvement with the documentation of the final product and attempts to produce a methodology and a system that will ease the contribution of the designer to the documentation authoring. The JavaDoc system takes a much more restricted approach and makes a commercial viable system. The NLG approach goes into a very detailed documentation generation but has the drawback of requiring a considerable amount of work in order to extract the necessary information and the linguistic part of the system.

## 2.2 Information Objects

The task of preparing documentation is an important part of the product design process. The documentation has to be up-to-date, accurate, and concise. The performance of the product and its marketability are linked to the quality of its documentation.

The focus is on the collection, processing, and structuring of product data in order to generate high quality product manuals. In other words, the main interest is on the procedure of authoring technical documentation. In this section a review on the Information Object (IO) concept is provided. Although IOs are extensively used, there is no single clear definition. Different authors view the term IO from different perspectives and apply it in different areas. The aim is to add together these efforts, and complete any missing parts, in order to make a complete evaluation and identify any possible shortcomings and extensions to the concept.

The Information Object has been considered as a basic concept of Virtual Documents [Paradis98, Vercoustre97, and Harris95], and other dynamic hypermedia applications [Setchi99]. It has also appeared in the literature as the cornerstone of an attempt to tackle the problem of producing technical documentation using Object-Oriented techniques and methods [Price97, Bist96, Mattheus92], and as a base concept in an effort to bridge the gap between product data and product documentation [Tucker97]. Overall, the Information Object concept can be found even to evolve as a standard for indexing of information [Ions03, Doi03].

### 2.2.1 Concept of an Information Object

The term Information Object has been used for any grouping of data, homogeneous or heterogeneous, text, graphics, or multimedia that can be included in a document. This data can either be pure or tagged by meta-data using a mark-up language that expresses presentational - HTML, SGML etc. - or knowledge - XML - semantics. Information Objects are handcrafted pieces of information, parts of existing documents, or structured data, that the author transforms in various ways and reuses in new documents. The concept of the Information Object has been studied with respect to reusability strategies [Paradis98, Vercoustre97, Harris95] and structuring techniques on information systems [Bist96, Price97]. At the same time, it provides the means for applying the Object-Oriented (OO) model to technical information systems [Bist96, Price97, Matthews92].

The IO has been very valuable, as a reuse-facilitator of information (1) and as a structuring component of a document (2). It defines the information as a unit providing encapsulation and thus making it easier to modify/alter it in order to fit it into a newly assembled document [Bist96, Vercoustre97, Paradis98] (3). The IO clarifies the characteristics that this information must have in order to be reusable. It enables this information to be rated according to attributes, such as granularity, modularity, autonomy, homogeneity etc. [Ranwez99, Vercoustre97], and properties such as size, style, role, coherence, ontology and others (4). It helps authors to divide information in chunks according to the concepts that they represent [Bist96, Ranwez99, Vercoustre97] (5), and thus build documents by assembling these objects, according to the way they are related, in various different orders and combinations [Ranwez99, Vercoustre97] (6).

In summary, the advantages (Table 2.1) include the following:

1. The IO approach enables reuse.
2. It is a basic information-structuring concept.
3. It applies the Divide and Conquer method on information providing encapsulation.
4. It defines a chunk of information as a separate entity with specific characteristics.
5. It enables a conceptual separation of the information.
6. It provides a basis on which one can see the relationships between the concepts more clearly.

Tucker and Harvey define the Information Object in the context of product documentation, as "a locution of product data that describes one idea" [Tucker97]. They base their research on the fact that in the first steps of the product life-cycle the STEP standard [Owen97, Kemmerer99] is heavily used by CAD/CAM and Product Data Management (PDM) applications. In the later stages, SGML [Sgml04] has proved to be very useful for the management and presentation of product documents. They investigate a way to introduce continuity from engineering design data (STEP)

**Table 2.1 Information Object Advantages**

| Information Object Advantages |
|---|
| Information Reuse |
| Information Structuring |
| Divide & Conquer |
| Application of characteristics |
| Conceptual Separation |
| Conceptual Relationships |

to technical document presentation data (SGML). They define a Perceptual Information Object (presentation view) and a Conceptual Information Object (conceptual view), and research the technical and application differences between STEP and SGML in order to find a way to harmonise the documentation process from the design to the presentation stage.

Paradis et al in their research on Virtual Documents, call Information Objects "the pieces of information that their document interpreter gathers from heterogeneous data sources and manipulates in order to fill-in a predefined template" [Paradis98]. This template is constructed by a user with an integrated editor, and thus generates a Virtual Document. They present a centralised system that addresses the problems of plurality of heterogeneous sources and efficient evaluation of the Virtual Documents, minimising the re-evaluation of information objects in these case where some of the information object sources have not been updated.

Ranwez and Crampes define in a similar way the Information Brick (IB) as "a fragment of a document that can be rendered in at least one medium, is characterised by a conceptual model, and is insert-able into a real document" [Ranwez99]. According to the authors, the building of a real document is then a matter of selecting, organising, and assembling the pertinent IBs. The IBs can also be segmented to sub-bricks and then assembled into composite bricks. Another important aspect is that they make a distinction between "homogeneous" and "non-homogeneous" information bricks. They also define the homogeneous IBs on the basis of the source of extraction of the information, and they note the characteristics, such as size, style, ontological domain, role, and content coherence, that are affected.

Vercoustre and Paradis present a language for Information Object reuse that allows users to write virtual documents, in which dynamic information objects can be retrieved from various sources, transformed, and included along with static information in SGML documents. The information objects can be SQL tables, Objects from an object Database, or semi-structured fragments of HTML. The language allows for flexible templates to be written, that include static data, queries, and mapping of the retrieved data to the virtual document, thus enabling the integration of the Information Objects by creating a higher level design view of the document [Vercoustre97].

Harris and Ingram describe the transition of a documentation team in IBM from conventional documentation to Information Element (IE) based Virtual Documentation. They define the Information Element as "a piece of text that comprises one or more topics to be presented as a unit" [Harris95].

Bist proposes the use of Object-Oriented (OO) modelling for technical documentation. He states that the structured design and the typical top-down documentation structure fails to accommodate the needs of reuse, flexibility, conceptual clarity, and non-book-oriented presentation. He argues that the use of the Object-Oriented modelling will solve these problems and further on will enable teamwork. Bist mentions that none of the existing tools follow this paradigm and tries to describe the features that an OO Technical Documentation system should have. This OO approach to authoring strongly depends on the use of Information Objects that are based on an OO Analysis of the domain in hand [Bist96].

Price presents an Object-Oriented methodology for the design of documentation systems based on Information Objects [Price97]. He describes an analysis procedure where concepts are discovered which qualify as information objects, and then are categorised in types according to the nature of the information they provide. After that some characteristics are identified in each object, such as function or responsibility, sub-components, membership, attributes etc. These information objects can then be reused in various ways, implementing a web-site, a CD-ROM, an on-line help system, or a printable document [Price97].

Setchi defines Information Objects as "a data structure that represents an identifiable and meaningful instance of information in a specific presentation form" [Setchi99]. She defines product and documentation elements. These elements are a higher granularity construct that is created for organisational and addressing purposes, and which in turn contains the information objects. Each information object is characterised by a set of meta-data that enable management, usage, type, and knowledge representation criteria.

Various information systems use the Information Object for indexing the information. In other words, it is employed as a unique identifier for keeping track of the data and the changes made [Ions03]. Furthermore, as the Internet evolves and electronic publishing becomes more and more popular, the Information Object is used for indexing, as a working standard through the Digital Information Identifier (DOI) [Doi03].

Finally, CISCO is active in the area of Information Objects with the RIO Project [Rio03]. The Reusable Information Objects (RIO) strategy describes a methodology for constructing training courses made out of reusable information objects that increase the productivity of the technical writer and improve the structure and overall quality of the course. The user benefits from a consistent approach throughout the course, just-in-time information delivery, customisation, and improved search [Rio03].

### 2.2.2 Areas of Application

The Information Object has been defined in four different ways for use in four different areas. In most cases the concept is used as a simple indexing mechanism. Vercoustre et al [Vercoustre97a] and Ranwez [Ranwez99] define it for use within Virtual Documents. Matthews et al [Matthews92], Price [Price97] and Bist [Bist96] use it for applying Object-Orientated methods to the documentation authoring process. Tucker [Tucker97] defines it in the realm of Technical Documentation. Harris and Ingram [Harris95] combine the two into an approach for producing Virtual Technical Documentation.

#### *Indexing in Information Systems*

In all information systems, data are stored in some medium to provide resilience. In some cases databases are used and in some other cases the data are stored in files. In all cases, the data are indexed in a way that is dependent on the storage strategy chosen. The files are described by names, the databases contain tables etc. In order for the system to be able to find and handle the information more easily, it is more comfortable to have a unique identifier that represents the information. Some information systems have already started using the term Information Object for defining such a uniquely named set of information items.

In conclusion, the key aspect of the Information Object applied in the information systems today is that of the unique identifier.

## *Generation of Virtual Documents*

Paradis et al [Paradis98a] use Information Objects to generate Virtual Documents. The Information Objects are data stored in the tables of a relational database, in the objects of an Object-Oriented database, or in HTML and SGML fragments. An editor is used to create document prescriptions, which are processed by a document interpreter that has access to the Information Objects and thus creates the final document. The Information Objects are returned by a query, and then are included and transformed to fit into the new output document.

In a more practical way, Harris and Ingram [Harris95] prepare Information Objects, Document Structures, and Content Templates and use conditional statements and an Effectivity Specification to generate Virtual Technical Documents. The document structures form a template and the content models arrange the organisation of the information elements. The complete document source is processed in a document processor and the conditionals are triggered according to the Effectivity Specifications. The output document is finally presented as a printed document, an online hypertext, or a multimedia document.

Ranwez and Crampes [Rawnez99] attempt a more theoretical exploration of the IO. They start by forming definitions for the concepts of Document, Information Brick (Object), Virtual and finally Virtual Document. Then they try to formalise the Virtual Document by focusing on the Information Brick/Object combined with a type of engine and a set of specifications.

The key idea behind the Information Object as a virtual document building unit is the arrangement of the object, according to the specifications, as a composite part.

### *Application of Object-Orientation on Documentation*

Bist [Bist96] discusses the application of the Object-Oriented model into the technical documentation domain. The analogy between the programming paradigm and the authoring practices is presented and Object-Orientation is proposed as a modernisation of technical authoring. The Object-Oriented model is considered mainly as an entity-relationship way of analysis, and information objects are constructed on top of the defined entities. Finally, tools for equipping such an authoring environment are described that allow for IO based authoring, diagrams, shared resources, different levels of information views, and multiple output formats.

Price [Price97] writes about Object Orientation from the perspective of applying the methodology to the authoring process. He uses basic steps for identifying Information Objects and tries to define the attributes that they should have. Again as in the case of Bist [Bist96] the implementation is conceived in a way that leads towards hypertext authoring.

Matthews and Grove [Mathews92] present a more elaborate investigation of the topic. They do take into account the functional part of an Object, conceptualising the Information Object as a proper state and behaviour entity, responding to messages and servicing requests through its interface. They discuss the encapsulation provided and they present the interface as possible information types and possible questions posed to the object. They also define in the document hierarchy structural and behavioural links. The structural links are normal links that the user has to trigger with his actions, and behavioural are links between the objects that they themselves handle in the background without the intervention of the user. Unfortunately Matthews and Groves do not provide further detail nor do they attempt an implementation that will provide evidence of the benefits.

The main idea behind an Object-Oriented view of documentation can be seen as the mapping of the information objects onto a conceptual structure, such as the one that a class hierarchy forms for Object-Oriented programming. The idea of adding a functional part to the information also appears but has not been successfully and sufficiently defined.

## *Technical Product Documentation*

Technical Product Documentation is strongly influenced by the product design. As the product is designed in a hierarchical way forming a system of discrete parts, the data are generated and stored in a similar way.

Tucker [Tucker97] attempts to harmonise the design data with the documentation objects. The information objects are defined as a locution of product documentation that describes one idea and are modelled by SGML, which in turn is embedded within STEP files. Finally, the information objects are collected and mapped onto some already defined publishing structure.

The main focus is both on having a continuity that will allow the information to travel from the design stages to the final documentation stage in a more effective and easy manner, and on establishing a conceptual continuity between the product structure and the documentation.

### 2.2.3 Implementations

As far as the implementation of the information objects is concerned, most of the authors propose an SGML approach [Tucker97, Bist96, Vercoustre97, and Paradis98]. In the case of Tucker [Tucker97], HyTime [Hytime04], and DSSSL [Dsssl04] is a proposed solution for presenting the information, and in the case of Vercoustre and Paradis a document interpreter does the processing of the document. Ranwez and Crampes use their Conceptual Evocative Engine and XML for implementing their information objects [Ranwez99]. Tucker proposes the embedding of the information objects within the STEP data and the creation of a STEP AP for handling the embedded information objects. Also proposed is modelling of the information objects along with the product using EXPRESS [Tucker97].

## 2.3 Virtual Documents

Virtual Documents (VDs) is an area of research that has been viewed through different prisms by various researchers. These prisms include model-based generation, data reuse, generation through natural language techniques, and more practical and simpler approaches which are applied in industrial environments.

### 2.3.1 Model-based Virtual Documents

A Virtual Document according to Gruber is a document generated on demand in response to user input [Gruber97]. Gruber et al propose a model-based approach to enable this generation. The model consists of a system that describes numerical models for various devices and thus can simulate their operation.

The DME (Device Modelling Environment) is a tool for modelling and simulating dynamic engineering systems. The user can work with the DME's libraries of components to connect them and thus create a model of a device. The DME can then compute numerical and discrete simulations to predict the behaviour of the device. Based on this simulation the system can produce explanations concerning the device and its inner workings. These explanations are natural language generated fragments that are sorted and presented to the end-user in the form of a Virtual Document [Gruber97].

One implication of Model-Based Virtual Documentation as it is identified by Gruber et al, is the integration of documentation and design practices. The usual process model has documentation as a separate part decoupled from the design that leads to out-of-date and inaccurate documentation. With a virtual approach to documentation, the design and documentation processes are integrated and the engineer can author the documentation by annotating the models already needed for analysis. Another basic point that should be mentioned is the capability of Virtual Documents to provide a shared context of collaborative work, since they can integrate shared databases and resources of a distributed team working on top of a computer network.

Finally, an inherent limitation is identified in Virtual Documents. Virtual Documentation is limited by an authoring bottleneck in a similar way that static documents are limited by the authoring bottleneck. The models need a considerable amount of work to be prepared and implemented. This makes virtual documentation expensive. The only way around this is when the domain of interest has very stable domain models that are easy to express with a formal representation [Gruber95].

### 2.3.2 Virtual Documents for Reuse of Data

Vercoustre and Paradis [Vercoustre98, Vercoustre97a] are mainly concerned with the reuse of existing resources such as documents and databases within the newly generated Virtual Documents. In this case, the Virtual Document is not based on a complex model and a modelling environment, but in a user defined prescription that provides a template with which the document processor works to assemble the pieces of information from the information sources and integrate them into the virtual document.

Vercoustre and Dell'Oro [Vercoustre96] evaluate different approaches for the generation of the VD. They use three different implementation approaches, a CGI script, the O2 Object-Oriented Database, and an OQL API. Moreover, an application is built on top of PHP [Williams04] and SQL [Earp03]. They focus more on the PHP server with which, they claim, simpler documents are easy to make and more complex ones are feasible. Through this study they come to the conclusion that four things are needed for the generation of virtual documents. These are: query, mapping, construction, and presentation. They propose the creation of languages supporting these functions and they conclude that the DSSSL (Document Style Semantics and Specification Language) is a good candidate for such an application. Finally, they discuss the possibility and the benefits of extending an editor as an integrated development environment for Virtual Documents.

Paradis [Paradis98b] presents a language for Virtual Documents with query and mapping capabilities. With this language, queries, such as the ones made with SQL or OQL, can be easily integrated within the document and the query result can be further manipulated to select a specific subset of answers. Furthermore, the language allows the definition of mapping constructs between results and tags (such as HTML), in order to transform the data into a form that can be seemingly integrated into the virtual document. As an HTML example, a list dynamically retrieved by a query can be mapped into a <LI> </LI> tag, and a table retrieved from a relational database can be mapped and presented into an HTML table [Musciano97].

Vercoustre and Paradis [Vercoustre98] describe the RIO (Reuse of Information Objects) approach and architecture, with the language shown by Paradis [Paradis98b] enhanced to support definition of links between Virtual Documents. They also propose further extensions to the language, such as control instructions for adapting the resulting document according to more specific queries.

## 2.3.3 Virtual Documents through Natural Language Generation

Dale *et al* aim towards a more puristic Virtual Document that is generated starting from more fine-grained components and Natural Language Generation (NLG) techniques [Dale96, Dale97, Dale98a, Dale98b, Dale98c, Dale98d].

Natural language generation techniques combined with raw data can offer better flexibility and functionality to virtual documents, compared to the methods using metadata annotated fragments [Dale98a]. Some of the advantages reported by Dale *et al* are: the description of the internal data included in the source (e.g. database), the contextual tailoring according to variables (e.g. user model), and multi-lingual output. The proposed architecture is a typical NLG system that wraps the text into an HTML template, while the selection of the links from the user is used as a discourse goal for the system. The only problem identified is that an existing database created for a different purpose might not be optimally structured for such a task [Dale98a].

Although Natural Language (NL) offers many attractive advantages [Reiter95] when it is used to produce extensive documents, and more specifically, technical documentation, some problems arise. The generation system needs a considerable knowledge base, which makes it more costly than producing normal documents. In addition, the increased possible outputs may not be welcome in documents that have to undergo a quality assurance process [Reiter95]. Church and Rau point out that the most successful NLG systems attribute much of their success to the limited vocabulary, grammar, and semantics they have [Church95]. These standards may be an advantage in technical documentation, where simplified English might be in use (e.g. aerospace industry), but they can be a serious disadvantage for the more general notion of Virtual Documents.

### 2.3.4 Practical Applications of Virtual Documents

Harris and Ingram [Harris97] have assessed virtual document generation in industrial environments. They present two case studies on documentation systems for IBM.

Harris et al [Harris97] show a Virtual Documentation System using the FrameMaker electronic publishing tool. The system uses variables and conditional statements to produce conditional views of the data superset. Harris clearly states the cost effectiveness of Virtual Documentation but also presents the difficulties technical writers encounter in such a paradigm shift.

Ingram [Ingram97] again from IBM, presents a methodology for producing virtual technical documentation. The methodology consists of five parts: information modelling, content modelling, design for reuse, standards and templates, and the incremental development and review techniques. Information modelling is the process of mapping product information to customer requirements using control and book definition files. Content modelling is a refinement of the information model that resembles, outlines, and helps engineers to review the content model; and information developers to guide the information gathering and writing process. Design for reuse handles the reusability factor of information models and content models, in order to render them with an abstraction that will make them reusable, thus forming the basis for future Virtual Documents. Standards and templates are the specification of style definitions, in order for the system to be able to update document formats from a centralised control file. The incremental development and review stages consist of the early creation of usable drafts that promote the quality assurance of the final Virtual Document System. The author then shows graphs with the productivity (page authoring per month) and cost (development cost per month) results. These show the page development to be slightly lower in the first year compared to the conventional authoring, and then having a dramatic increase (from 2 to 22) in the monthly generated pages. The inverse happens to the cost of the document production. The most important argument from Ingram is his view of virtual document development as software development, although it reduces the hope of virtual document development becoming a simple process that can be applied by non-programmers.

## 2.4 Intelligent Agents

The concept of agents has been quite confusing as far as its definition is concerned. The different definitions by different researchers on agent technology are numerous [Franklin96]. The fact remains that everybody accepts it as the modern approach to AI [Norvig95]. It is also viewed as the latest paradigm in software engineering [Jennings00] and has been successfully applied in many areas [Jennings98]. Although there are many diverse approaches to agency, some basic notions have started becoming a standard [Fipa04] and thus provide a more usable and productive framework for the creation and deployment of multi-agent systems.

### 2.4.1 Autonomous Agents

Autonomous agents are software applications, programs, or computational entities that act autonomously, are proactive, realise a set of goals and tasks, perceive an environment, act, reason, and have social ability. These are only a few of the properties autonomous software agents have according to the different views of researchers.

#### *Agent Definitions*

One of the most popular definitions of an agent is that it is "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors" [Norvig95]. This definition is generic, but it gives a good idea of the basic concept of agency, which communicates the fact of someone or something that acts on behalf of someone else. As Maes puts it "Autonomous Agents ... realise a set of goals or tasks for which they are designed" [Maes95]. A stricter definition would be too limiting because, as Norvig states, "the notion of an agent is meant to be a tool for analysing systems, not an absolute characterisation that divides the world into agents and non-agents" [Norvig95].

#### *Agents as a new Software Paradigm*

According to a more flexible point of view autonomous software agents are an approach, a paradigm for the analysis and design of software systems. Agent-Oriented

Programming (AOP) proposed by Shoham [Shoham93] aims to build systems as societies of agents. The main features that this approach aims to exploit are that agents are autonomous concurrent processes, are cognitive systems with beliefs, goals etc, are reasoning systems, and are communicating with other agents. AOP has three layers. The first is the logical system for the definition of the mental states and behaviour, the second is an interpreted programming language for implementing the logical specifications, and the third is a layer that produces the executable agents that form the specifications [Wooldridge97].

Wooldridge et al define an Agent-Oriented analysis and design software development methodology [Wooldridge99]. The main characteristic of the methodology is that it abstracts the system in two levels, an agent micro-level, and a social macro-level. They separate the methodology into an analysis stage and a design stage. The analysis has as a purpose to build the organisational model of the system. The organisational model is decomposed into the role model and the interaction model. Then the design process produces the agent model, the service model, and the acquaintance model. Once this methodology is followed, the system is decomposed into a level that can be handled with the traditional Object-Oriented design process.

## *Agent Applications*

Since the concept of autonomous agents is so global, its applicability is equally broad. Autonomous agents have been used to implement various types of systems, and in different domains. Jennings and Wooldridge categorise agent applications into four domains, industrial, commercial, medical and entertainment applications [Jennings98]. In industrial applications, process control, manufacturing and air traffic control, are some domains that are considered promising for multi-agents systems. In commercial applications, agents are used for information management, electronic commerce, and business process management. In the medical context, such systems are utilised in patient monitoring and health care management. Finally, entertainment agents are quite heavily used in computer games.

### 2.4.2 Information Agents

Information Agents (IA) are autonomous agents that focus on assisting users to handle large information spaces. Nowadays, Google has indexed 8,058,044,651 web pages [Google05]. Information Agents use artificial intelligence, knowledge-based systems, distributed information systems, information retrieval, and human computer interaction, to improve and partially automate the users searching capabilities.

#### ***Information Agent Definition***

An Information Agent is defined by Klusch as "an intelligent agent that has access to one or more heterogeneous and geographically distributed information sources, and which pro-actively acquires, mediates, and maintains relevant information on behalf of users or other agents, preferably just-in-time" [Klusch01]. According to this definition an Information Agent is responsible for information acquisition and management, information synthesis and presentation, and intelligent user assistance.

to all the objects that provide services, such as the PDM Monitor, the User Model, the Logic Engine etc.

As a result, with this architectural pattern, the Document Component forms a structured community of objects and acquires some basic "social" abilities thus achieving communication between the Document Components. Furthermore, the roles that different parts of the system serve (PDM, User Model etc.) and their use, is controlled more effectively by the Document Components.

The second main behaviour of the Component is that it is a placeholder for data. As such, it plays two main roles, being an extractor of data and a responder to queries. The extraction is isolated in the Extract Information Interface and the response to queries is isolated in the Detailed Query Interface. The extraction is the part that is associated with the PDM monitor and gets the data, while the detailed query is the interface that ensures that the component can answer specific queries that can be used by the other components.

With this interface two new views of the information in the PDM are created. One view responds to the question 'what should be extracted from the PDM?' and the other to the question 'what are the sub-sets of this extracted information that could be useful to the other Document Components?'. This way the information is viewed as a large selection set extracted from the original data and subdivisions of this set suitable for reuse in the communication between the components.

The Report Interface (Figure 3.2) generates reports of the data contained in the component in the form of HTML Documents. These reports are generated using the Logic Engine and the User Model for selection and adaptation of the data that will be presented to the specific user in hand and the use of the Detailed Query Interface.

A Report is a description that coordinates the interaction between the Components and the use of the Detailed Query Interface. It provides a central point of reference that shows the selected information reused through the queries between the Components.

The Rule Engine component (Figure 3.2) is an abstraction that supports simple "if then" rules, and if needed acts as a wrapper to a proper rule based engine.

The Text Processing subsystem (Figure 3.2) includes the Text Processor and the Product Dictionary. The Product Dictionary keeps a record of all the component names instantiated. Also, the names are broken down if they are multiple words and the relevance to existing terms in the dictionary is checked. In the case that two terms are found to be related, the larger one acts as the main term and the other is noted as a related sub-term. The text processor uses this dictionary to parse the text content of the components and to automatically create hyperlinks pointing to the related component. An additional link is created as a star (*). This link shows the lexically related terms as well as additional links that point to a proper dictionary database.

The User Management subsystem (Figure 3.3) contains the Login Handler and the User Model components. The Login Handler identifies the user according to his username and password. Then the User Model object, according to the username, finds and retrieves the previously set values of the User Model for the specific user.

The PDM Integration (Figure 3.3) component consists of three parts: the PDM Monitor, which is the main controller class, the update checker, which uses a timer and probes the files to check for updates, and the node objects that act as an intermediate placeholder for the data. The PDM Integration allows the system to access the PDM database in a higher level of abstraction where the data are handled in larger, meaningful parts that answer to specific questions about the product and the parts that it involves.

The Web Integration (Figure 3.3) module comprises of a set of Servlets that feed the output of the components reporting interface to the web. The Servlets also handle the requests received by parsing the parameters and feeding them to the root component. The Main Servlet holds the state management of the web application by utilising sessions and cookies (an encoded signature placed on the web browser) [Cookie04, Netscape04] that identify the user. The Side Servlet holds the lexically related links and the links to the Dictionary Servlet, while the Menu Servlet serves the structural menu created by the components of the application.

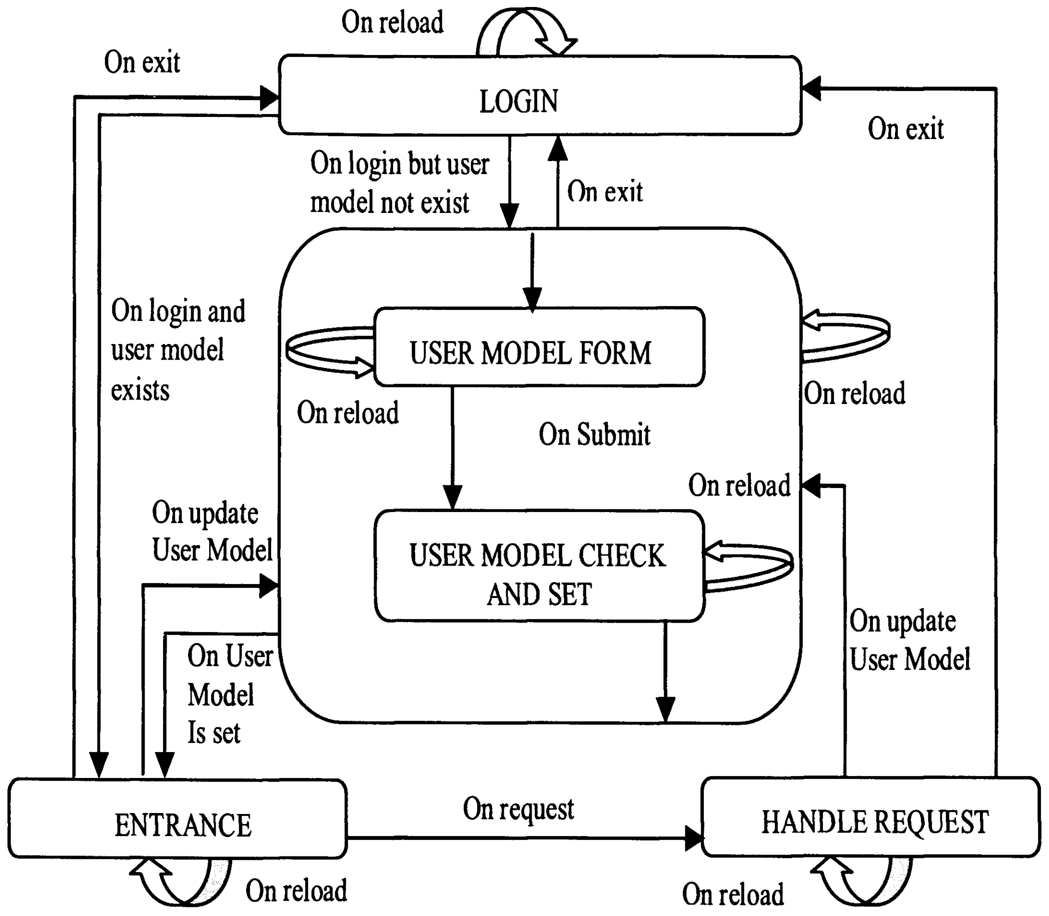


**Fig. 3.4 The Web Application State Machine (class Main)**

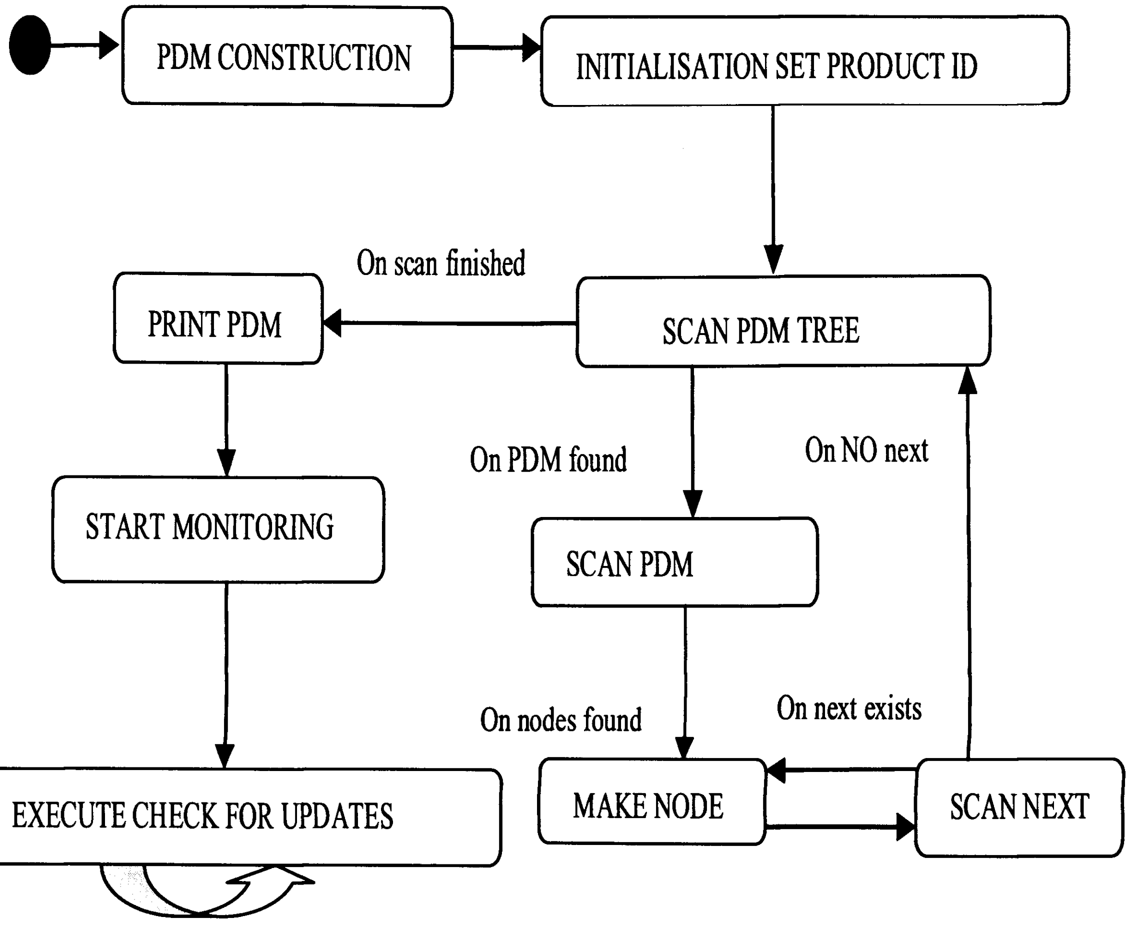


**Fig. 3.5 The PDM Monitor State Machine**

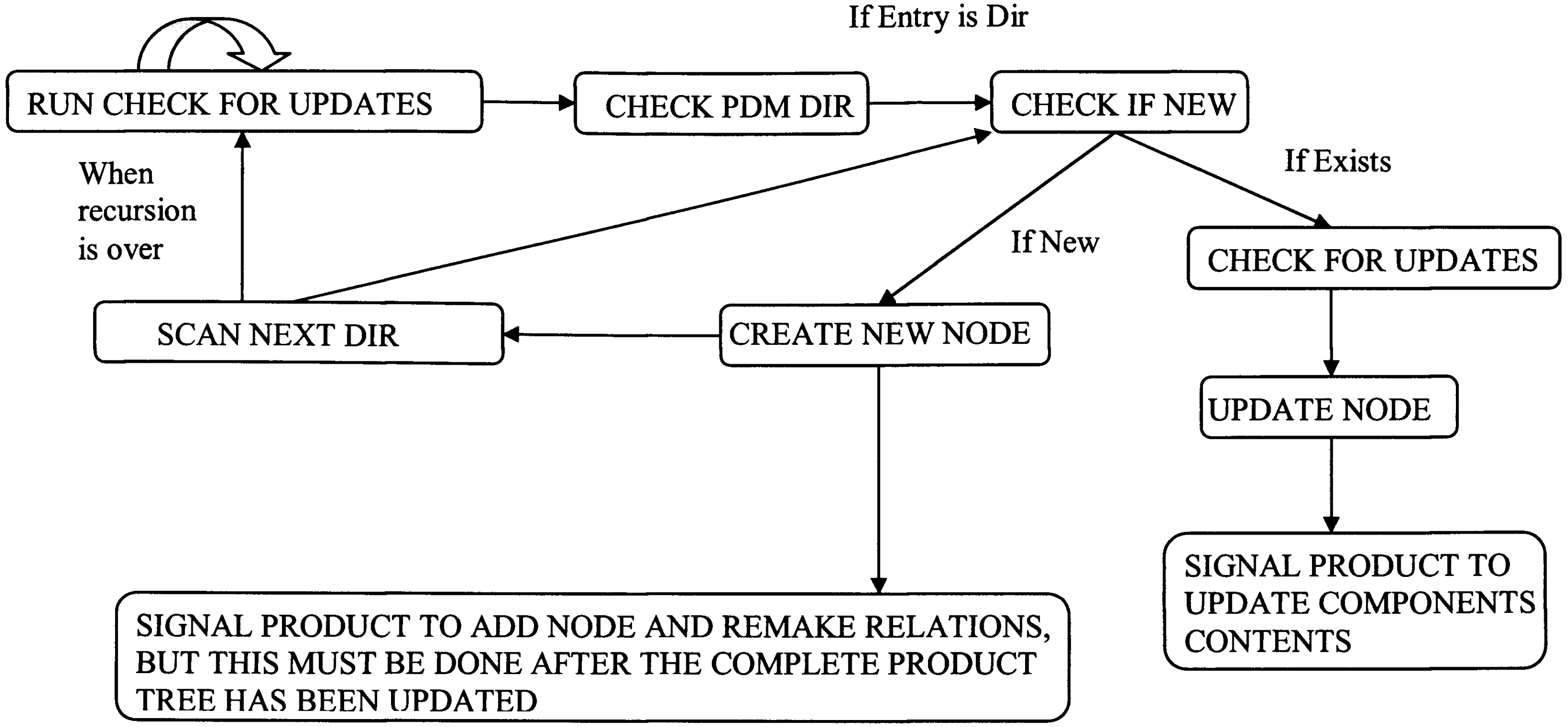

If Entry is Dir
RUN CHECK FOR UPDATES
CHECK PDM DIR
CHECK IF NEW
When recursion is over
If Exists
If New
CHECK FOR UPDATES
SCAN NEXT DIR
CREATE NEW NODE
UPDATE NODE
SIGNAL PRODUCT TO UPDATE COMPONENTS CONTENTS
SIGNAL PRODUCT TO ADD NODE AND REMAKE RELATIONS, BUT THIS MUST BE DONE AFTER THE COMPLETE PRODUCT TREE HAS BEEN UPDATED

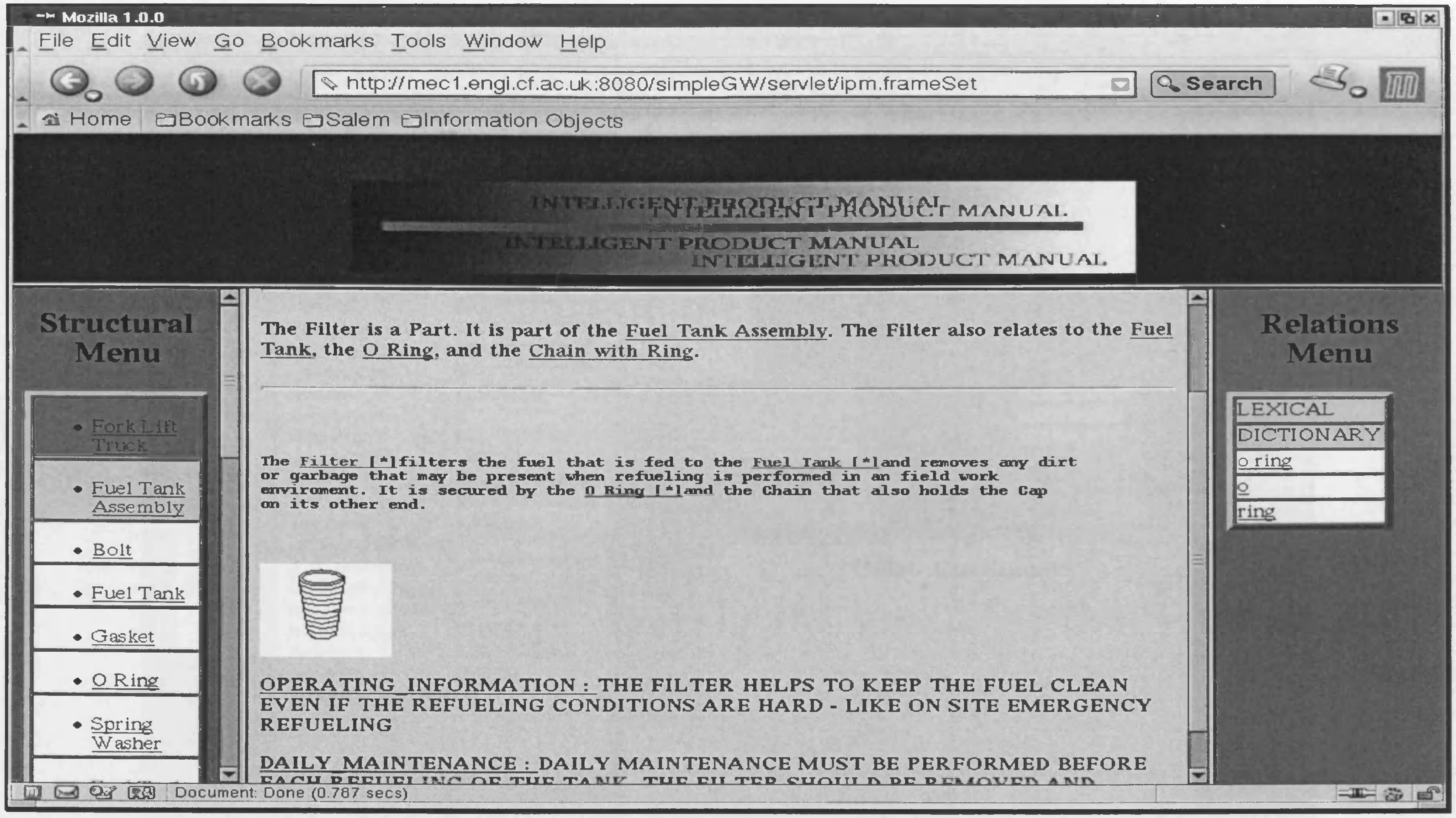
**Fig. 3.6 The State Machine for Update Checking**

## 3.6 An Application Based on the Object-Oriented Framework for Product Documentation

The framework presented has a twofold use. It can be utilised as a complete and ready made application or as an extensible software framework for the development of more evolved electronic documentation systems. In this section an application is shown - the framework is deployed as is without any need for writing additional code - which presents partial documentation for a Fork Lift Truck and two assemblies, the Braking System assembly and the Fuel Tank assembly. This documentation system can be used by users with different profiles to browse the up-to-date documentation customised according to their user models thus achieving better results in the delivery of information.

Figures 3.7 and 3.8 show the resulting output from the web application. In the left frame, the menu that was prepared by the components is presented. In the case that the links are followed, a coded request towards the information components of the web application will be generated. In the middle frame, the report developed by the currently addressed component is illustrated. The first part of the report is a dynamically generated natural language section describing the structural positioning of the component.

The next part is a textual description of the entity represented by the component that has been pre-processed by the text processor; hyperlinks have been added dynamically. Next to the link there is the star (*) link that produces the results in the right frame. Then a graphic is presented. Finally, a selection from the contents of a


Mozilla 1.0.0
File Edit View Go Bookmarks Tools Window Help
http://mec1.engi.cf.ac.uk:8080/simpleGW/servlet/ipm.frameSet
Search
Home
Bookmarks
Salem
Information Objects
INTELLIGENT PRODUCT MANUAL
INTELLIGENT PRODUCT MANUAL
INTELLIGENT PRODUCT MANUAL
INTELLIGENT PRODUCT MANUAL
Structural Menu
Fork Lift Truck
Fuel Tank Assembly
Bolt
Fuel Tank
Gasket
O Ring
Spring Washer
Relations Menu
LEXICAL
DICTIONARY
o ring
o
ring
Document: Done (0.787 secs)

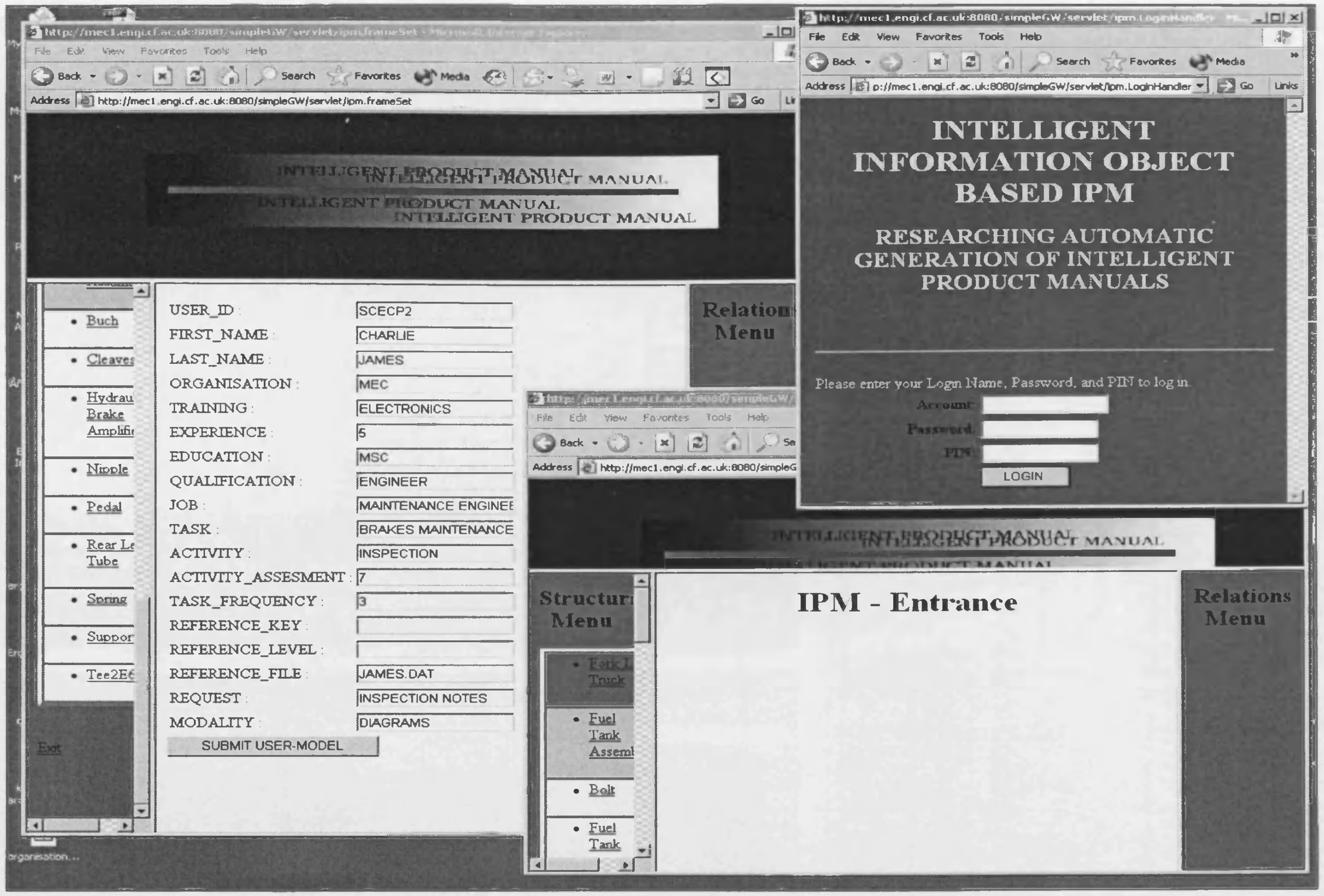
The Filter is a Part. It is part of the Fuel Tank Assembly. The Filter also relates to the Fuel Tank, the O Ring, and the Chain with Ring.

```
The Filter [*]filters the fuel that is fed to the Fuel Tank [*]and removes any dirt
or garbage that may be present when refueling is performed in an field work
enviroment. It is secured by the O Ring [*]and the Chain that also holds the Cap
on its other end.
```

OPERATING_INFORMATION : THE FILTER HELPS TO KEEP THE FUEL CLEAN EVEN IF THE REFUELING CONDITIONS ARE HARD - LIKE ON SITE EMERGENCY REFUELING

DAILY_MAINTENANCE : DAILY MAINTENANCE MUST BE PERFORMED BEFORE EACH REFUELING OF THE TANK. THE FILTER SHOULD BE REMOVED AND

**Fig 3.7 A Request served by the Application Produced with the Object-Oriented Framework**



| | |
|---|---|
| USER_ID : | SCECP2 |
| FIRST_NAME : | CHARLIE |
| LAST_NAME : | JAMES |
| ORGANISATION : | MEC |
| TRAINING : | ELECTRONICS |
| EXPERIENCE : | 5 |
| EDUCATION : | MSC |
| QUALIFICATION : | ENGINEER |
| JOB : | MAINTENANCE ENGINEE |
| TASK : | BRAKES MAINTENANCE |
| ACTIVITY : | INSPECTION |
| ACTIVITY_ASSESMENT : | 7 |
| TASK_FREQUENCY : | 3 |
| REFERENCE_KEY : | |
| REFERENCE_LEVEL : | |
| REFERENCE_FILE : | JAMES.DAT |
| REQUEST : | INSPECTION NOTES |
| MODALITY : | DIAGRAMS |

http://mec1.engi.cf.ac.uk:8080/simpleGW/servlet/ipm.frameSet

http://mec1.engi.cf.ac.uk:8080/simpleGW/servlet/ipm.LoginHandler

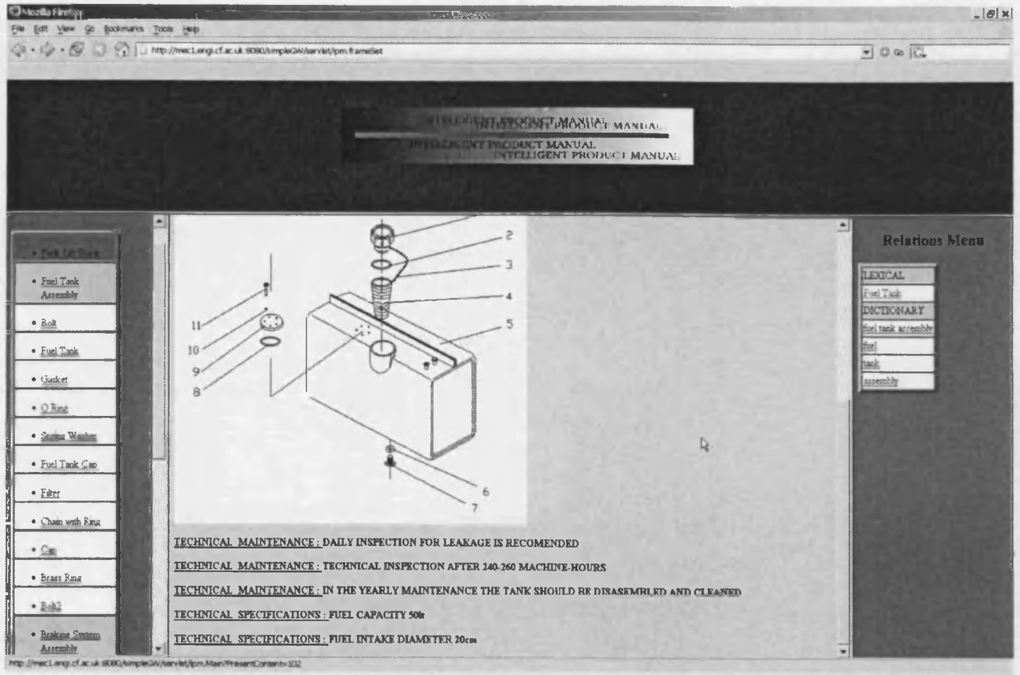
INTELLIGENT INFORMATION OBJECT BASED IPM

RESEARCHING AUTOMATIC GENERATION OF INTELLIGENT PRODUCT MANUALS

Please enter your Login Name, Password, and PIN to log in.

Account:
Password:
PIN:

**Fig 3.8 Entrance, User Model Form and Login States**



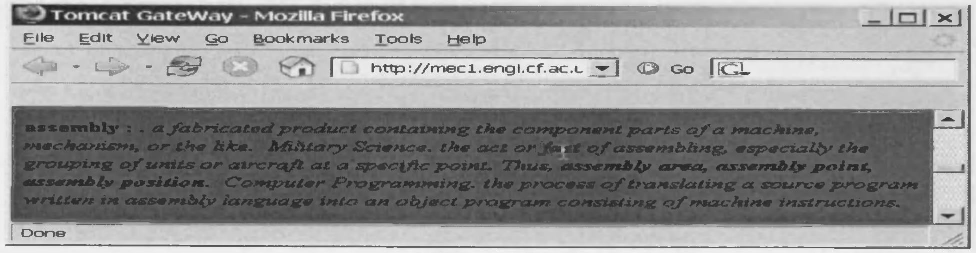
TECHNICAL MAINTENANCE : DAILY INSPECTION FOR LEAKAGE IS RECOMENDED

TECHNICAL MAINTENANCE : TECHNICAL INSPECTION AFTER 240-260 MACHINE-HOURS

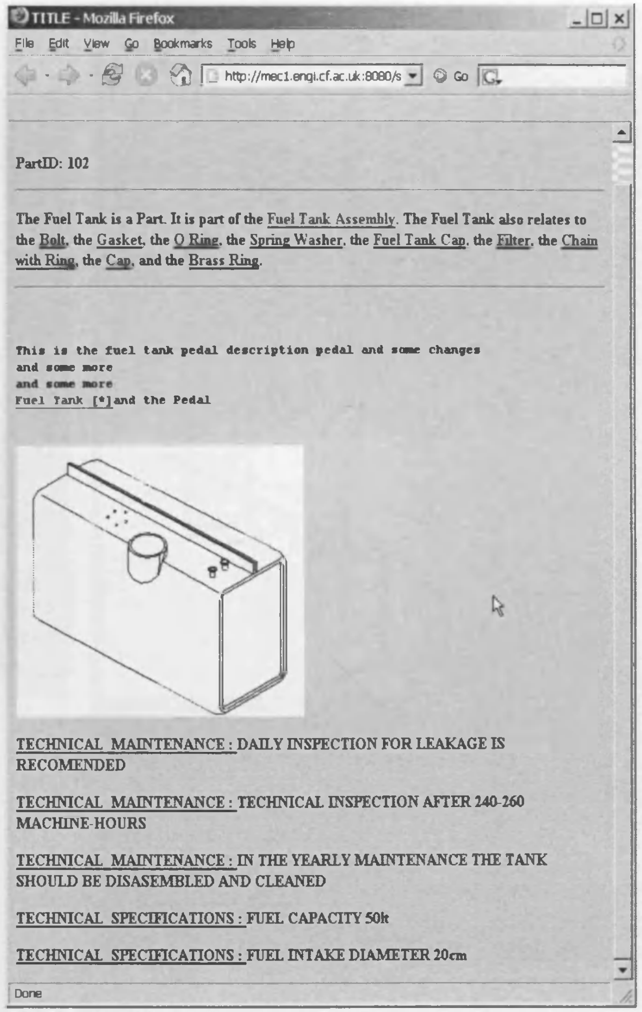
TECHNICAL MAINTENANCE : IN THE YEARLY MAINTENANCE THE TANK SHOULD BE DISASEMBLED AND CLEANED

TECHNICAL SPECIFICATIONS : FUEL CAPACITY 50lt

TECHNICAL SPECIFICATIONS : FUEL INTAKE DIAMETER 20cm


Tomcat GateWay - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://mec1.engi.cf.ac.u
Go
Done

**assembly** : . *a fabricated product containing the component parts of a machine, mechanism, or the like. Military Science. the act or fact of assembling, especially the grouping of units or aircraft at a specific point. Thus,* ***assembly area, assembly point, assembly position.*** *Computer Programming. the process of translating a source program written in assembly language into an object program consisting of machine instructions.*


TITLE - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://mec1.engi.cf.ac.uk:8080/s
Go
Done

PartID: 102

The Fuel Tank is a Part. It is part of the Fuel Tank Assembly. The Fuel Tank also relates to the Bolt, the Gasket, the O Ring, the Spring Washer, the Fuel Tank Cap, the Filter, the Chain with Ring, the Cap, and the Brass Ring.

```
This is the fuel tank pedal description pedal and some changes
and some more
and some more
Fuel Tank [*]and the Pedal
```

TECHNICAL MAINTENANCE : DAILY INSPECTION FOR LEAKAGE IS RECOMENDED

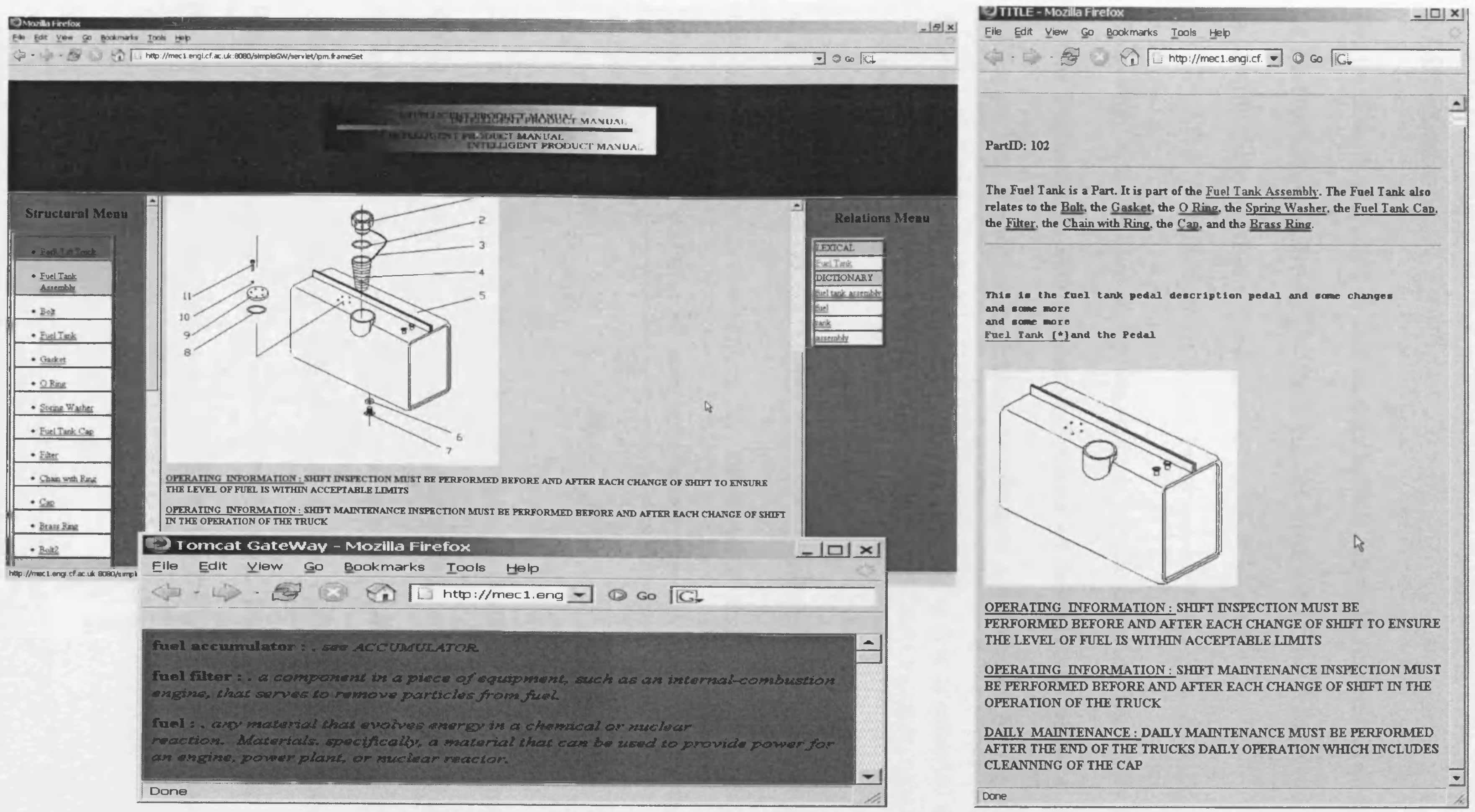
TECHNICAL MAINTENANCE : TECHNICAL INSPECTION AFTER 240-260 MACHINE-HOURS

TECHNICAL MAINTENANCE : IN THE YEARLY MAINTENANCE THE TANK SHOULD BE DISASEMBLED AND CLEANED

TECHNICAL SPECIFICATIONS : FUEL CAPACITY 50lt

TECHNICAL SPECIFICATIONS : FUEL INTAKE DIAMETER 20cm

**Fig 3.9 Content Adapted on a Technical User Profile**



OPERATING INFORMATION : SHIFT INSPECTION MUST BE PERFORMED BEFORE AND AFTER EACH CHANGE OF SHIFT TO ENSURE THE LEVEL OF FUEL IS WITHIN ACCEPTABLE LIMITS

OPERATING INFORMATION : SHIFT MAINTENANCE INSPECTION MUST BE PERFORMED BEFORE AND AFTER EACH CHANGE OF SHIFT IN THE OPERATION OF THE TRUCK

**fuel accumulator :** . *see ACCUMULATOR.*

**fuel filter :** . *a component in a piece of equipment, such as an internal-combustion engine, that serves to remove particles from fuel.*

**fuel :** . *any material that evolves energy in a chemical or nuclear reaction. Materials. specifically, a material that can be used to provide power for an engine, power plant, or nuclear reactor.*

PartID: 102

The Fuel Tank is a Part. It is part of the Fuel Tank Assembly. The Fuel Tank also relates to the Bolt, the Gasket, the O Ring, the Spring Washer, the Fuel Tank Cap, the Filter, the Chain with Ring, the Cap, and tha Brass Ring.

```
This is the fuel tank pedal description pedal and some changes
and some more
and some more
Fuel Tank [*]and the Pedal
```

OPERATING INFORMATION : SHIFT INSPECTION MUST BE PERFORMED BEFORE AND AFTER EACH CHANGE OF SHIFT TO ENSURE THE LEVEL OF FUEL IS WITHIN ACCEPTABLE LIMITS

OPERATING INFORMATION : SHIFT MAINTENANCE INSPECTION MUST BE PERFORMED BEFORE AND AFTER EACH CHANGE OF SHIFT IN THE OPERATION OF THE TRUCK

DAILY MAINTENANCE : DAILY MAINTENANCE MUST BE PERFORMED AFTER THE END OF THE TRUCKS DAILY OPERATION WHICH INCLUDES CLEANNING OF THE CAP

**Fig 3.10 Content Adapted on an Operator User Profile**

## 3.7 Discussion

The framework presented in this chapter models the main features of an application for documentation systems. The main entities involved in the process are identified and abstracted in a software system.

The system, as it is shown in the case study, is capable of generating a fully functional Virtual Document. It allows dynamic updating of the data and offers a rich set of utilities such as the dynamic linking, personalised output based on the XML data, dictionary support etc. The application can be executed as it is or extended to provide solutions customised to more specific needs of documentation systems.

As a reusable framework, the system can assist in the increase of productivity for the documentation system developers by sub-classing and extending the existing subsystems or by addition of new subsystems that can interact with the already implemented ones.

The framework proposes as main utilities for product documentation systems the User Modelling and Management for promoting security and intelligence, the Rule Engine for enhancing the intelligence and decision making of the system, the Document Component for the structural decomposition of the documents, the integration to the Product Data Management system for better updating, the implementation of a web accessible layer for improved accessibility to the end-user and the use of Text Processing for the mining of the textual resources for related references.

# Chapter 4 - An Intelligent Information Object Architecture

This chapter provides a definition for an Intelligent Information Object. The original concept of Information Objects that was presented in the literature review is analysed, its shortcomings are presented, and the ways of Information Objects taking an intelligent form are discussed. The Information Object concept is extended and promoted to that of an Autonomous Agent, thus addressing the fourth aim of this research. Then, a new architecture that is based on Intelligent Information Objects is presented, realising the fourth objective. Finally, an application of Intelligent Information Objects implementing an Intelligent Product Manual is presented. The results of the system are shown and the effect of the use of the Intelligent Information Object concept is discussed.

# 4 An Intelligent Information Object Architecture

In the realm of technical documentation, it is expected that the product is designed and the data stored and structured within a Product Data Management (PDM) system. In an ideal world, it would be preferable to have the documentation produced automatically, without any human intervention, such as suggested by Literate Programming techniques [Knuth84] and tools such as JavaDoc [Friendly95]. This ideal documentation would be presentable in many formats, printable, and electronic. It could use many modalities and have many forms, such as that of a user manual, maintenance manual, or specifications. Furthermore, the documentation would be adapted to, and be adaptable by, the user as in the case of Adaptive Intelligent Product Manuals (IPM). It should be capable of delivering just in time and should cater for personalised information.

Two main stages can be readily identified in the information involved in product documentation. First, there are raw data that are structured by the PDM system and then, in the last stage, a complete and coherent document is formed and is presented to the end-user (Figure 4.1). In between, there is a gap that technical authors have to bridge in order to generate the technical manual. The authoring procedure is a highly complex human task and a time-consuming endeavour. These factors compromise the success and limit the profits of the potential product. Thus, a method is needed for facilitating this data transformation process and moving towards its automation.

The Information Object has been an important and valuable tool for authoring as shown in many studies. It has helped developers to divide and conquer the procedure of technical documentation authoring. It has enhanced its management, and has made user adaptation feasible [Harris95, Price97].

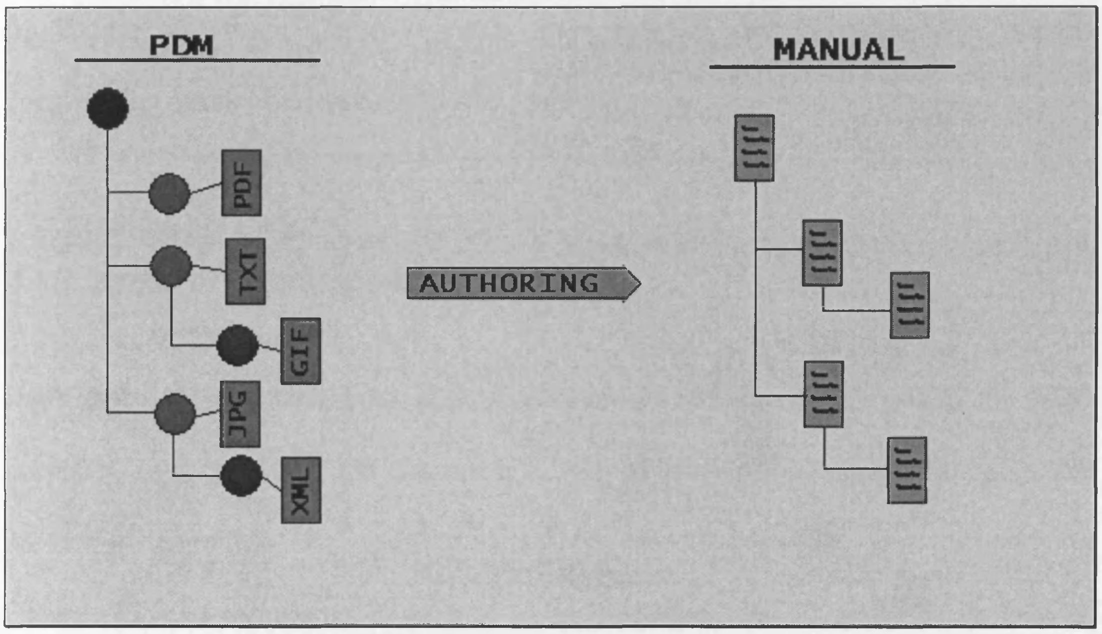


**Figure 4.1 The Process of Documentation Authoring**

document component nature, and as Information Elements to promote the unique identification of the information. The most popular name was found to be "Information Object", and therefore this was adopted in this research. Furthermore, the term Object works better for the Object-Orientation part of the literature.

An Information Object (IO) is defined by the phrases, "a locution of product documentation" [Tucker97], "a fragment of a document" [Ranwez99], "one or more topics" [Harris95] and "pieces of information gathered from data sources" [Paradis98a] (Table 4.1). In order to produce something that will encompass all the existing definitions, the highest abstraction will be chosen. The phrase "a set of data" will be used, ignoring the documentation/document specification that points to the type of use of the data, and also to the topics that deal with the problem of decomposition of information objects to smaller information objects.

According to the existing definitions an IO has a property, namely "characterising", "describing" or "presenting" an "idea", a "conceptual model", or a "unit" (Table 4.2). This will be combined with the principle of unique identification that comes from the generic notion of the information object within information systems [Ions03], and it will be represented by the phrase "it is uniquely identified by a concept".

A second property defined is that of being "insertable into a new document". This states the fact of the IO being a composite part of a document (Table 4.2). As has been mentioned, the information object can be part of a document or a stand-alone piece of information. The best choice to formulate this is: "it can be a component of a document".

A third property is that it is "manipulatable" by an actor in order to show or hide different parts of information or alter in any way part of its contents to present a specific view to a user, either modified to fit better within a document or a personalised view adjusted to specific user needs. A more appropriate choice would be a very high level view of this property, simply stating that the information object "can enable control over its contents and itself" in order to capture all the aspects of modification of the information object. This allows the option of including modification of the contents as well as structural modifications of a set of information objects and their interconnections; having in mind that the pointers to the related IOs can be part of an IO's contents.

Finally, the Object-Oriented definition is ignored because it is a major issue by itself. It must be said that in the literature various attempts have been made to approach the information object through an object orientated perspective. Some useful parts of the literature mostly apply the methodological view of object orientation, but none actually defines Object-Oriented frameworks capable of handling the authoring in a proper Object-Oriented way, that is in the way it is applied in programming through Object-Oriented languages. This matter strongly depends on the definition of object orientation that each person will choose to start with. For this reason, this will not be further analysed. A flexible phrase stating that the information object "adheres to a certain degree with the methods and practices of Object-Oriented analysis and design" will be added to the definition of information objects.

**Table 4.1 Information Object Definitions, Part 1**

| No. | Subject | | | Definition |
|---|---|---|---|---|
| 1 | [Tucker97] | An Information Object | is | a locution of product documentation |
| 2 | [Ranwez99] | An Information Brick | is | a fragment of a document |
| 3 | [Harris95] | An Information Element | comprises | one or more topics |
| 4 | [Paradis98a] | Information Objects | are | pieces of information gathered from data sources |

**Table 4.2 Information Object Definitions, Part 2**

| Property | property specifier | Property | property specifier |
|---|---|---|---|
| Describing | one idea | | |
| Characterising | a conceptual model | insertable | into a new document |
| Presenting | a unit | | |
| Manipulatable | in order to create virtual documents | | |

### 4.1.3 An Overall Definition

The overall definition that encompasses all up-to-date work on Information Objects is the following.

An Information Object defines a set of data that is uniquely identified by a concept, can be a component of a document, can enable control over its contents and itself, and adheres to a certain degree with the methods and practices of Object-Oriented analysis and design.

### 4.1.4 Advantages

According to the literature [Paradis98, Vercoustre97, Harris95, Bist96, Price97], the main advantages that have been identified in the use of the Information Object concept are that it enables the reuse of product data, facilitates information structuring, and allows better information management. The latter advantage arises from the application of divide-and-conquer strategies to the authoring process. The encapsulation of information by an IO and the definition of information entities with specific characteristics help technical writers to organise better the processing of the reused information. Finally, the conceptual separation allowed by IOs and the relationships that they enable to be defined between concepts result in generated documents with increased consistency and clarity.

These advantages allow documentation authors to structure and manage their documentation processes and techniques and thus increase their productivity and quality of documentation design.

### 4.1.5 Disadvantages

Overall, it can be stated that the main shortcoming of the information object concept is the lack of automation in its use. This can be divided into three main areas. These are, lack of a global definition that will enable the automation, lack of an autonomous nature that will realise the automation, and lack of use of intelligent techniques that will further enhance the automation of the Information-Object-based authoring process.

#### *Lack of Global Definition*

The Information Object has been given different definitions and names and has been used in different ways and for different purposes. This leads to confusion as to what the IO actually is. An overall study would improve the concept and would put it in a more rigid foundation, making it more usable and useful.

#### *Lack of Autonomous nature*

Continuing on the line of thinking that led the IO to Object Orientation, which actually means the action and processing of the data that the IO holds, the "action" can be generalised. One can think of the IO not just as an object that provides methods for manipulating the data it contains, but as an actor, an agent that autonomously acts on the data it contains to achieve a specific goal. Agent-Oriented Programming (AOP) is the future successor of Object-Oriented Programming (OOP) [Tveit01], so the concept of the Information Object is naturally driven to that of an autonomous agent by following the evolution of software engineering practices.

Also as Tveit states "Agents are similar to objects, but they also support structures for representing mental components, i.e. beliefs and commitments" [Tveit01]. Having in mind the difference in semantical complexity of data as in strings or integers used in normal programming practices compared to the data held by an information object, a higher level approach is much more suited to the information object type of data and behaviour.

### *Lack of Intelligent Techniques*

The above reflections on a global definition of the IO and on its autonomy suggest that the information object has much to benefit from artificial intelligence techniques. Furthermore, the application of text planning, text mining, and natural language generation could be applied within, and in combination with, the information object either for automating some of the characteristics that have been defined or for generating part of the descriptions, included in the information object, which are based on the semantics of its interconnections with other information objects. Other techniques might also prove to be useful and help in the development of a more evolved information object concept.

### 4.2.2 Application of AI Techniques

Various Artificial Intelligence techniques can be utilised for adding intelligence to information. In the case of IO, techniques such as text mining, knowledge-based information retrieval, and natural language generation, can prove to be very efficient for discovering relations between the IOs, presenting adapted information, and generating descriptions.

#### *Natural Language Processing and Generation*

Natural Language Generation has been shown to be very helpful, but it suffers from restrictions in the size of the document and the domain dictionary. The best results are achieved when the generated text is in a very restricted and well-defined sub-language [Church95]. Information Objects help subdivide the information in smaller and more easily controllable units where the NLG could prove to be more applicable. As an example of this, the generation of short paragraphs describing the structure between information objects can be presented, given that the meaning of the structural links follows a specific pattern.

#### *Knowledge-Based Information Retrieval*

One technique that has proved very useful for information delivery is that of Knowledge-Based Information Retrieval. It has been used effectively in many applications including the current IPMs [Setchi99] with simple Information Objects. Intelligent Information Objects can benefit from such a technique if they are thought of and implemented as elements that are tagged with an appropriate language such as XML. This would allow an information agent to apply its ontology on the contained data and to execute an inference for selecting the relevant XML fragments to be displayed to the user, or to decide upon the way that these data have to be presented.

***Text Mining***

Within the context of Information Objects, text mining could be used for the identification of references from within an Information Object, which could be linked with the conceptual model represented by another Information Object. These could then be used to represent some semantic relationship between the Information Objects realised as an HTML hyperlink.

## 4.3 An Intelligent Information Object Definition

An Information Object acts as a document structuring component. It gives authors the opportunity to define methodologies for the generation of documents and thus formalise the authoring procedure. It gives a clearer and more solid foundation on the art of authoring which was, and is still is, a very complex human activity, by dividing the information in meaningful discrete units, and forming the document by assembling them [Ranwez99, Price97, Bist96].

An information agent is an autonomous agent. According to Maes [Maes95] "Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed". More specifically, according to Klusch [Klusch01] an information agent satisfies the requirements of information acquisition and management, information synthesis and presentation, as well as intelligent user assistance.

The resulting combination is a computational system realising goals and tasks, such as the acquisition, management, synthesis, and presentation as described in the information object based authoring methods and practices. This allows the use of the agent notion to automate the construction and the deployment of the information objects. In this manner the complexity that a monolithic approach to the document construction introduces is overcome.

Based on the existing definitions for the Information Object and the definition of the Information Agent, a definition is provided for an Information Object that will be intelligent and will improve the authoring of technical documentation. The domain of application is restricted to technical documentation because that is the area in which the information object has been established. Technical documents have inherent characteristics that make information objects a more productive approach, an effect that cannot be expected from other types of documents.

The following definition is therefore proposed.

*An Intelligent Information Object is a concept-specific autonomous agent that searches an appropriately structured information source and discovers the concepts to be represented. It extracts a set of data to initialise its contents, updates itself, can render its contents customised according to external parameters, and interacts with* *other Intelligent Information Objects in order to build a complete instance of a document.*

The Intelligent Information Object and its main characteristics, as they are implied/stated in the definition, are shown in Figure 4.2. This illustrates that an Information Object will automate a major part of the technical authoring process and will generate intelligent virtual documents of customised information that will increase the productivity of information delivery to the user.

## 4.4 A Novel Architecture for Information

The use of the Intelligent Information Object produces a novel architecture for electronic technical documentation. This architecture is based on Intelligent Information Objects and uses the concept to build a highly dynamic middle layer where the information will be formed according to the conceptual model of the existing information objects. At the same time the information in this middle layer will be interlaced due to the active nature of Intelligent Information Objects.

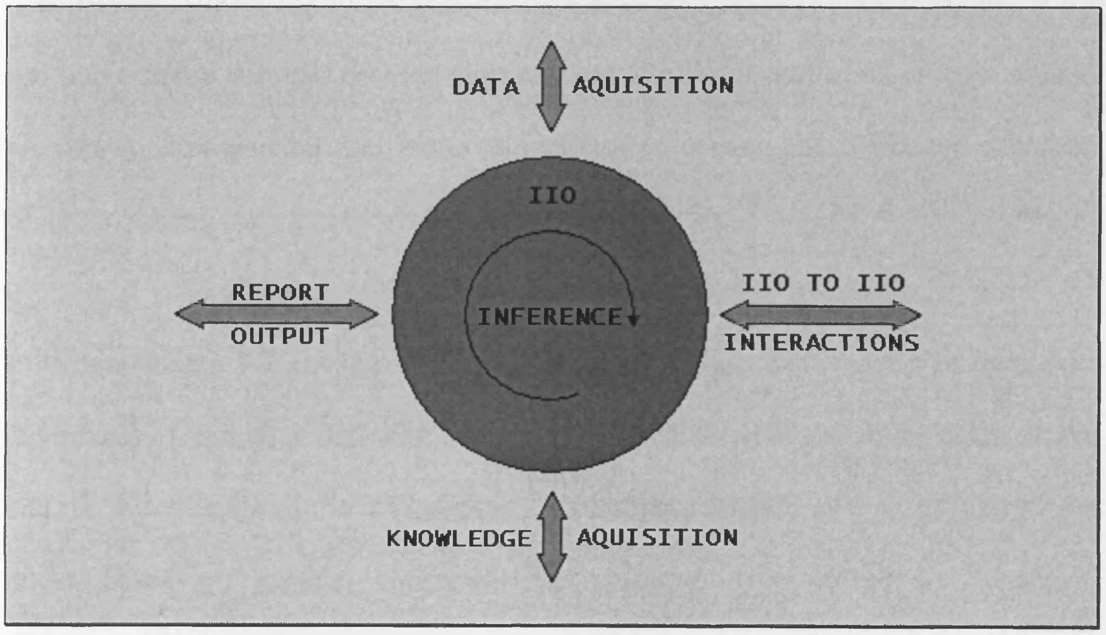


**Figure 4.2 Intelligent Information Object**

### 4.4.1 A Dynamic Middle-Layer Based on Intelligent Information Objects

The phases that can be identified in the procedures of technical authoring are those of information extraction, information processing, and generation of the output of the virtual documentation to be presented. The required information has to be extracted from the Product Data Management System. Then the technical author or documentation team has to process the data in various ways (Figure 4.3), and in the end one or more manuals that the company has to offer as part of its product support are created. The problem lies in the information processing phase where a collection of very complex tasks has to be executed.

In the Intelligent Information Object architecture, and through the use of Intelligent Information Objects, a buffering zone is introduced between the information source and the presentation of the information. This buffer is realised as a set of information nodes based on specific concepts. The nodes are represented by Intelligent Information Objects that are able to apply transformations upon the contained information and on the structure imposed on the nodes.

This creates an intermediate stage where the pertinent information has been extracted, but has not been fully formed, thus resulting in an instance of a Virtual Document (Figure 4.4). In this stage, the information exists in a completely dynamic state and part of the procedure has occurred, including the information extraction from the sources and its separation/classification in Information Objects.

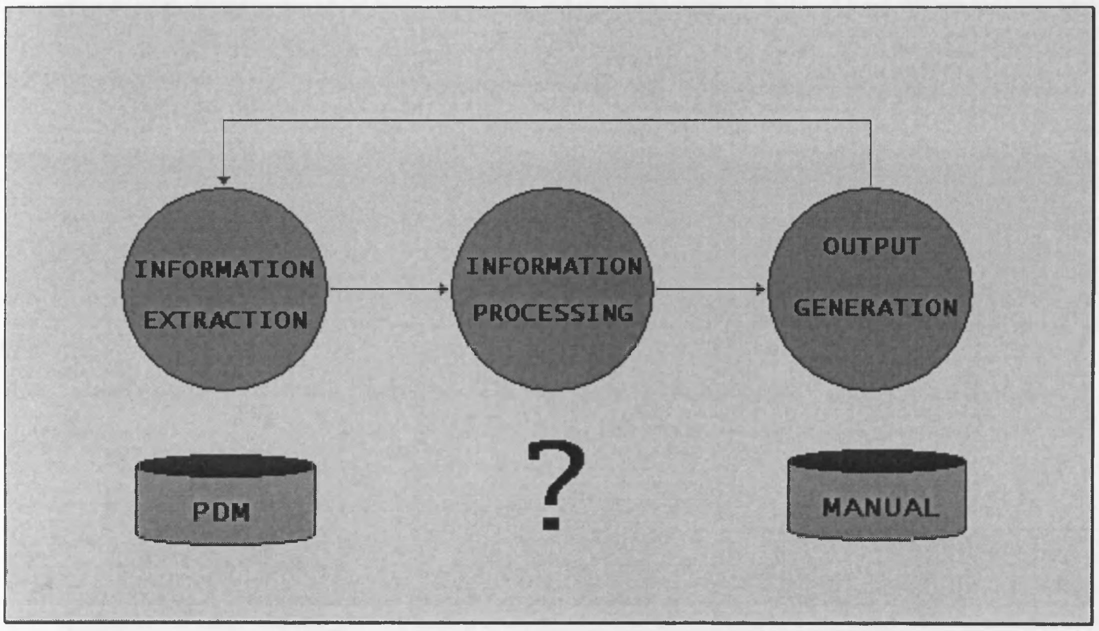


**Figure 4.3 Phases of Technical Authoring**

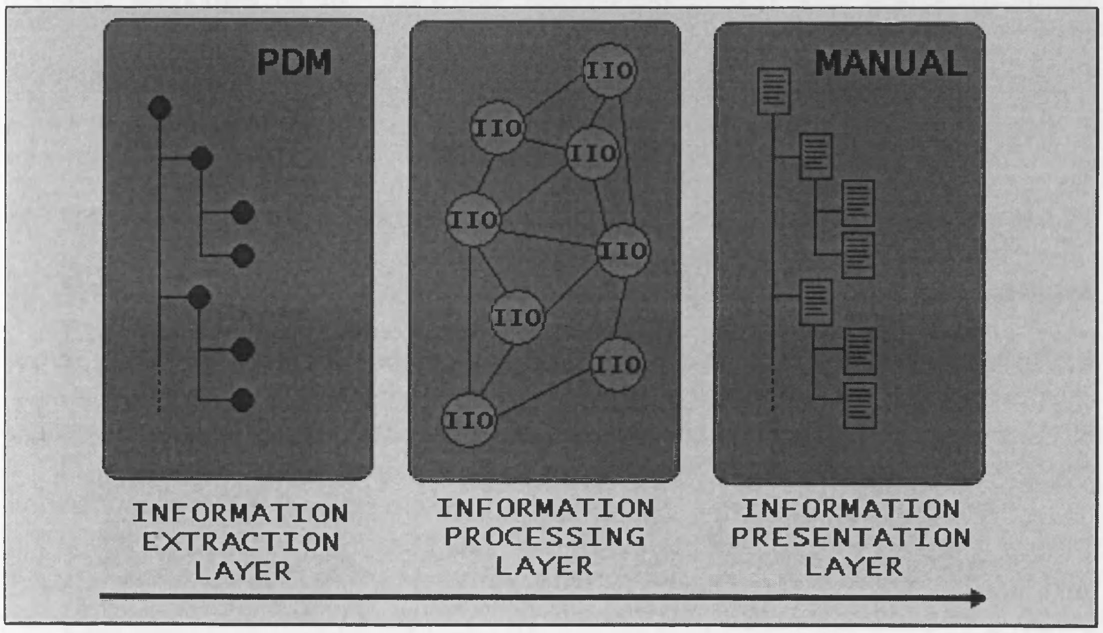

PDM
IIO
MANUAL
INFORMATION EXTRACTION LAYER
INFORMATION PROCESSING LAYER
INFORMATION PRESENTATION LAYER

**Figure 4.4 A Dynamic Information Middle Layer**

However, the second stage of processing the information during which the information becomes even more structured, in accordance with the User Model and other structural relations deriving from the analysis of textual data, has yet to be applied.

### 4.4.2 Consequences of the Intelligent Information Object Organisational Structure

The consequences of the intelligent information object architecture affect many systemic properties of the electronic documentation and also enable up-to-date technological approaches to be applicable.

The conceptual separation, encapsulation, and structuring characteristics enforced by the IO concept can be fully utilised within the IIO architecture. The localisation of the search within the strict boundaries of the IIO for references to other parts of the conceptual model can yield more focused and enriched results, as well as a wealth of associations between the IIOs.

Distributed objects can be reused at runtime and be part of many different products simultaneously, since in an autonomous agent architecture each agent can locate and associate itself with another at runtime, independently of the exact location and/or system where each of the components resides.

The distribution of information in nodes, which comprise a conceptual model that acts as a dynamic network, provides a more computationally effective way of dealing with the extraction and processing of the product data. In such a distributed architecture, different tasks can be executed in parallel and thus the procedure of the generation of the final Virtual Manual can be more effective.

## 4.5 Realisation of an Intelligent Information Object Architecture

Realising the IIO Architecture involves an innovative set of cutting-edge technologies. The IIO has been defined as an autonomous agent and thus it will be realised and deployed within an agent platform. Furthermore, in order to support the knowledge-based extensions and web deployment, an expert system shell providing an inference engine will be used, and a servlet engine will provide the gateway towards the World Wide Web.

### 4.5.1 Useful Technologies

After researching the available tools implementing the technologies that were needed by the systems specifications the following list of tools has been selected.

- The FIPA-OS Agent Platform.
- The JESS Java Expert System Shell.
- The Tomcat Servlet Container.

#### *FIPA-OS Agent Platform*

The FIPA-OS for Intelligent Autonomous Agent development was the choice of platform for the deployment of the agents. FIPA-OS is an open source project created by Emorphia [Emorphia05] in order to have a platform to be used as a benchmark for the FIPA Standards. According to the standards reference model it supports DF (Directory Facilitator) and AMS (Agent Management Service) agents, a transport layer (Message Transport System), and connections between distributed platforms through ACC (Agent Communication Channel) (Figure 4.5). The FIPA-OS Platform and FIPA ACL are equipped with performatives enabling the use of Speech Acts, Communication Protocols and Conversation Management [Poslad00a, Poslad00b]. With the use of performatives the agents can use communicative acts (such as inform, request, confirm etc.) within their messages and protocols that will specify the interactions in the specific conversational pattern. For example the FIPA "request" protocol specifies that after a request the receiver can reply with an "agree" or a "refuse" performative. After an "agree" the continuation will be an "inform" act or a "failure" of the conversation. This way the FIPA specification enables the management of the agent interactions through this formalised conversational tactics.

### *JESS Expert System Shell*

The JESS Java Expert System Shell is a rule engine written entirely in Java. It is based on the CLIPS System and uses the Rete algorithm to realise a production system. The Rete [Forgy82] algorithm has advantages over the previous algorithms for rule-based systems. Typically, the set of rule is stable, the knowledge base continuously changes by the addition of new facts. The amount of new facts added is comparatively small. A simple algorithm would take all the rules and in every cycle iterate through the rule set and try to match the left hand side with the facts in the knowledge base. This is a very inefficient tactic since the same operations with the already known results are constantly repeated. Rete instead remembers the past test results and only the new facts are tested. In addition to this JESS can be seamlessly integrated in any Java application and there is an existing wrapper in order for it to be used as an

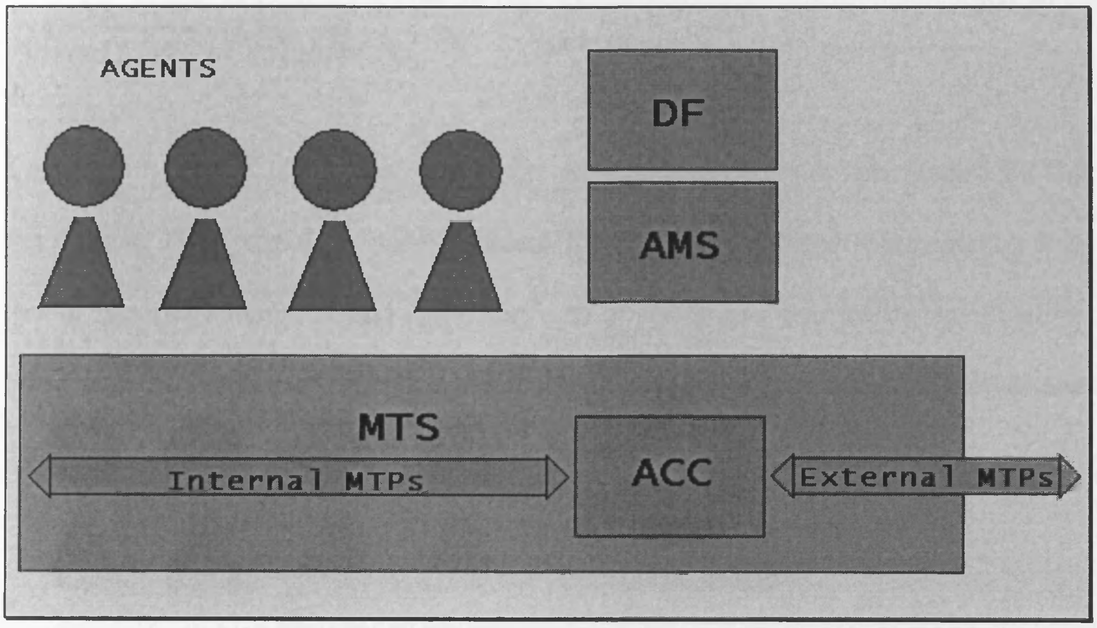


**Figure 4.5 The FIPA-OS Multi-Agent platform [Poslad00a]**

inference engine within the FIPA-OS platform. It is quite often used as an expert system for the construction of Intelligent Agents [Friedman03].

### *Tomcat Servlet Container*

Tomcat is a Servlet Container used in the official reference implementation for the Servlet and JSP technology specifications. Servlets offer a better performance than that of the traditional CGI and again they can be easily and seamlessly integrated to other Java Applications. The Servlet Container cooperates with the Web Server and handles the dynamic content requests.

The Tomcat Servlet container will be used in this architecture to provide a gateway towards the World Wide Web, thus allowing the output of the dynamic middle layer of the IIO architecture to be delivered anywhere and at any time it is needed, facilitating the use of any of the available electronic formats supported by today's web technologies [Tomcat03].

A typical scenario of Tomcat cooperating with the Apache web server is shown in Figure 4.6. The web server receives HTTP requests and when targeting a dynamic resource the request is forwarded to Tomcat to handle the loading and execution of the appropriate Servlets.

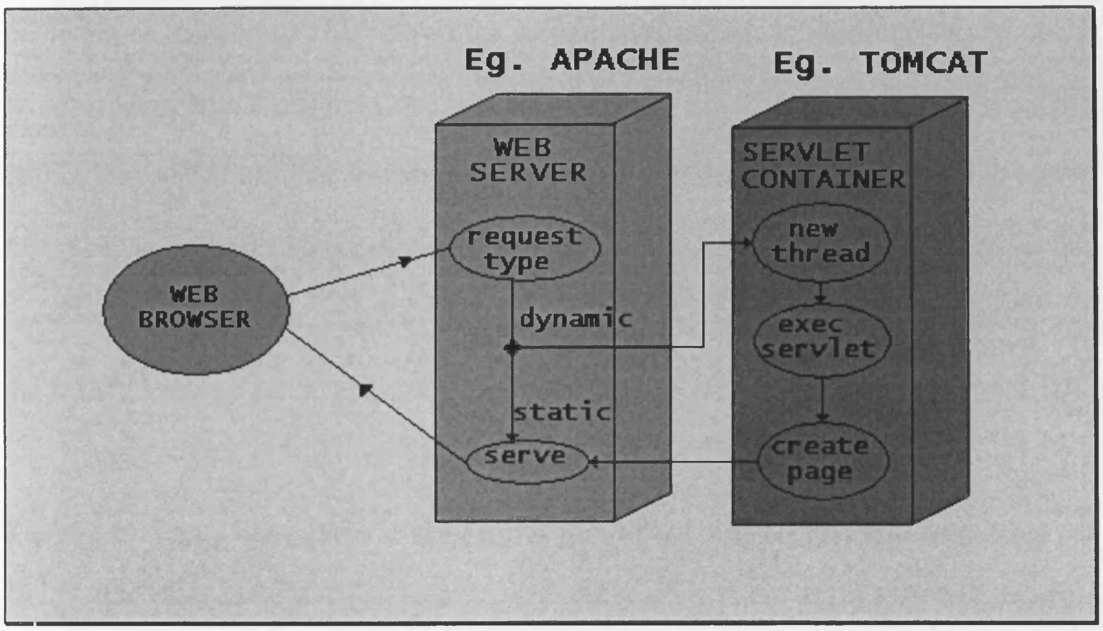


**Figure 4.6 The Web Server and Servlet Container Components**

## 4.5.2 System Architecture

The architecture that has been described defines a multi-agent system with knowledge-based agents operating autonomously. It is separated in three main layers: the web presentation layer, the agent middle-ware layer, and the supportive layer. The web presentational layer consists of the Servlet Container that handles the output servlets. These arrange the generic presentation of the web-site that the system publishes as an output. This allows for further customisation where the servlet could be substituted with a different one, so that instead of a web document could wrap the output into a PDF or other format. The second part of the web layer is the web server that receives requests from the clients' browser and passes it to the appropriate servlet to be processed. Finally, the web layer consists of the World Wide Web facility and the HTML browser of the end-user (Figure 4.7).

The agent middle-ware layer is the central part of the architecture and contains a set of resource agents that provide standard services and the Intelligent Information Object agents (Figure 4.7). The resource agents' services act as utility agents that the IIOs contact to fulfil specialised tasks and access data from outside of the system.

The supportive layer consists of more fundamental facilities which the agents can use such as databases containing user models or technical dictionaries, inference engine etc. The supportive layer could also be used as an extension and customisation layer, where various resources more specific to the instance of the system in hand can be plugged-in to enrich the provided services (Figure 4.7).

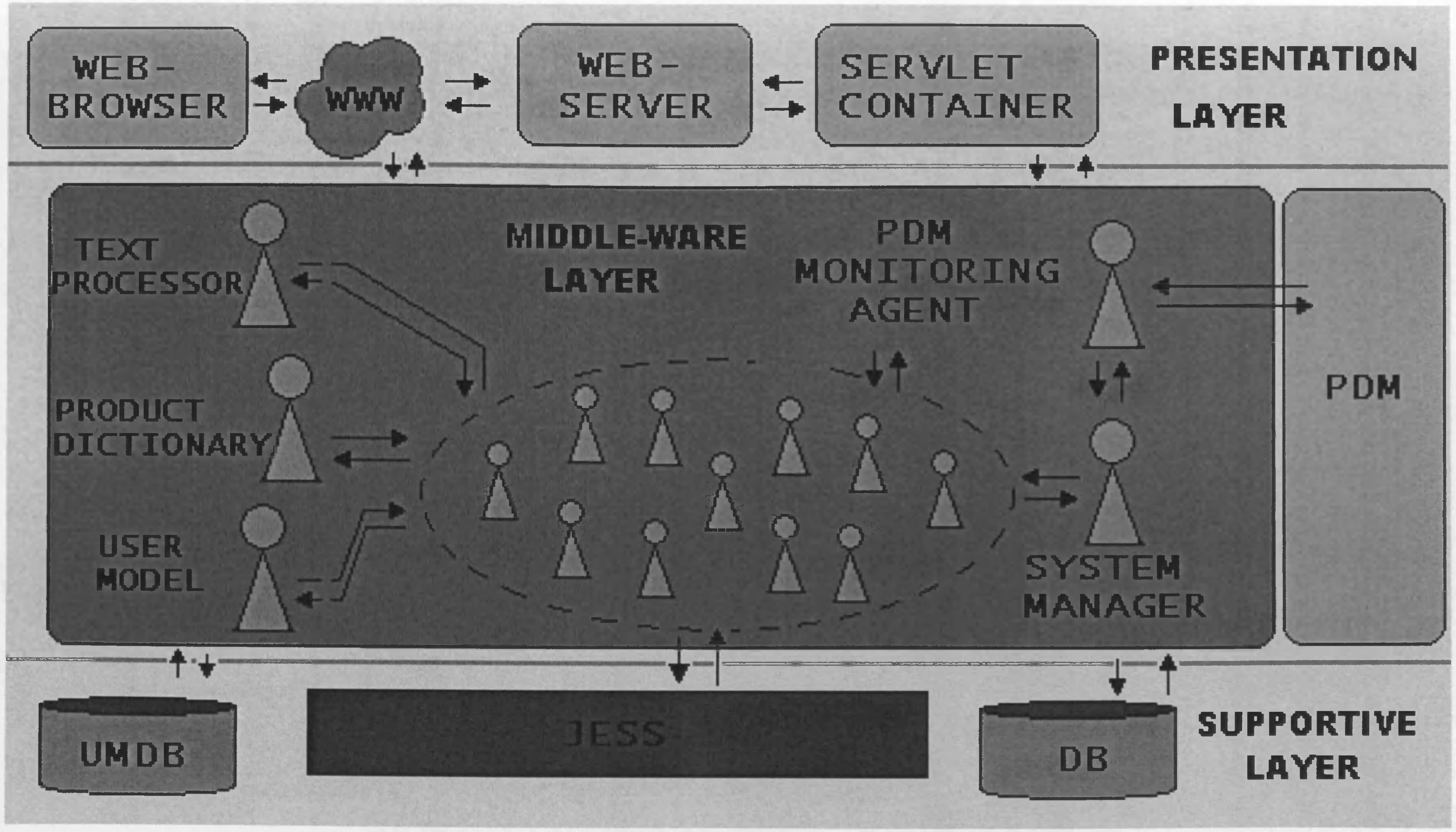


**Figure 4.7 Overall System Architecture**

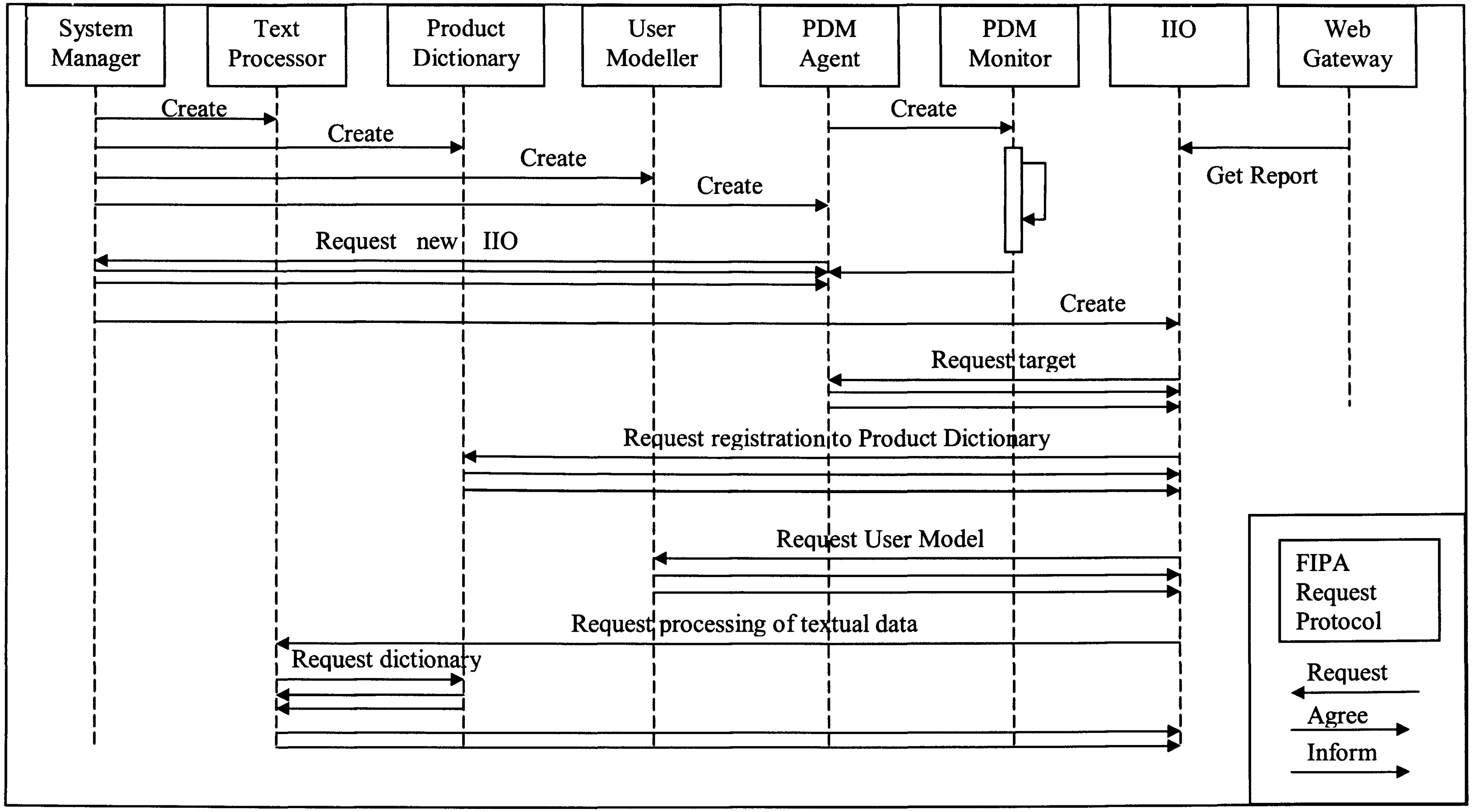


**Figure 4.8 IIO System Interaction**

The PDM Monitoring module is an agent that scans the Product Management System (PDM) and identifies new nodes or extracts data from these nodes, either for initialisation of an IIO or for updating purposes. When a new node is found in the system then the PDM Agent will request the System Manager to create a new IIO. If some data of an existing IIO have been updated, the agent will notify the IIO to update the specific data, thus minimising the updating procedure.

### *Intelligent Information Object Unit*

The Intelligent Information Object is the main unit of the presented system. The IIO utilises the services offered by the resource agents. It keeps the data and organises the way the various services will be used. The system's resource agents offer management and interface components. The IIO is the cornerstone/vital gear that utilises the various services to realise its goal of cooperatively producing the documentation.

The Product IIO is first created by the system manager agent, and then it contacts the PDM monitoring agent and extracts the relevant data. After the contents of the IIO have been initialised, the IIO can look at the subsystems it includes and delegate to the system manager the creation of the respective IIOs (Figure 4.9). These new IIOs will request their data and thus the procedure will propagate through the product tree having all the necessary population of IIOs created and initialised. Moreover, during the initialisation phase the IIO will contact the product dictionary agent to register with it.

After the data initialisation has been completed, each IIO will be receiving update messages from the monitoring agent whenever any of the data changes and so their content can always be up-to-date. In the data pre-processing phase, the IIOs will contact the text processing utility agent to send along their textual content so that possible links to the other current IIOs of the systems' population (Figure 4.9) may be discovered. The text processing utility agent talks with the product dictionary agent to obtain the list of the current members of the community. Then the IIO goes into an idle state until it is contacted either with a data update signal from the PDM monitoring agent or with a request for data. The request for data is a signal received from the servlet and directed to the product IIO, and then each IIO propagates it, searching cooperatively to find the IIO it addresses.

In the data presentation phase, the IIO has been found and it is asked for information. Then it can look-up the user, by contacting the user model agent, and request its user model. This enables it to infer, through the use of the Jess production system, knowledge content that is more relevant to the user needs. In this way, it can parse the XML content and customise it, filtering out the redundant information.

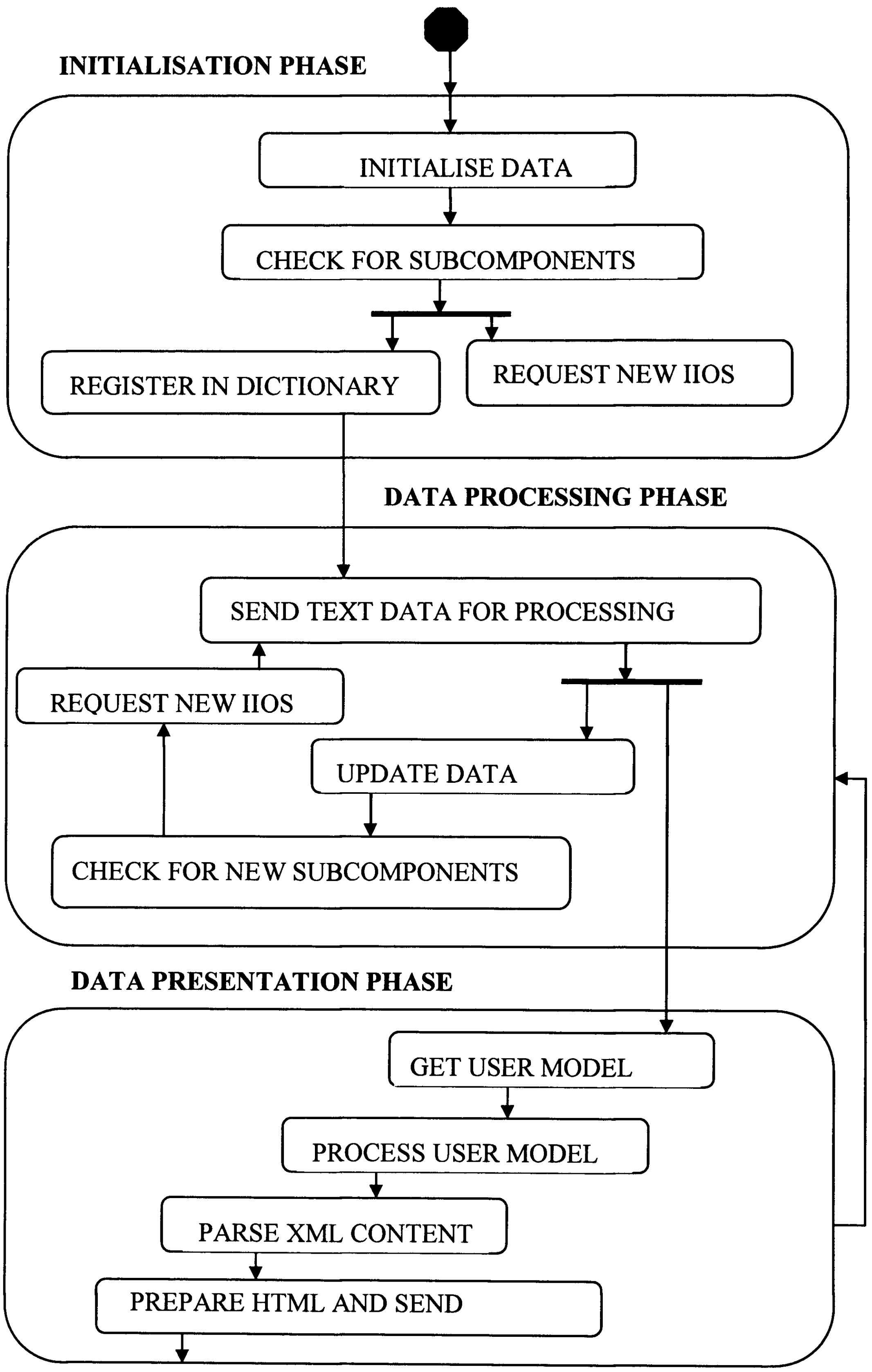


Figure 4.9 Internal IIO States

### *User Interface*

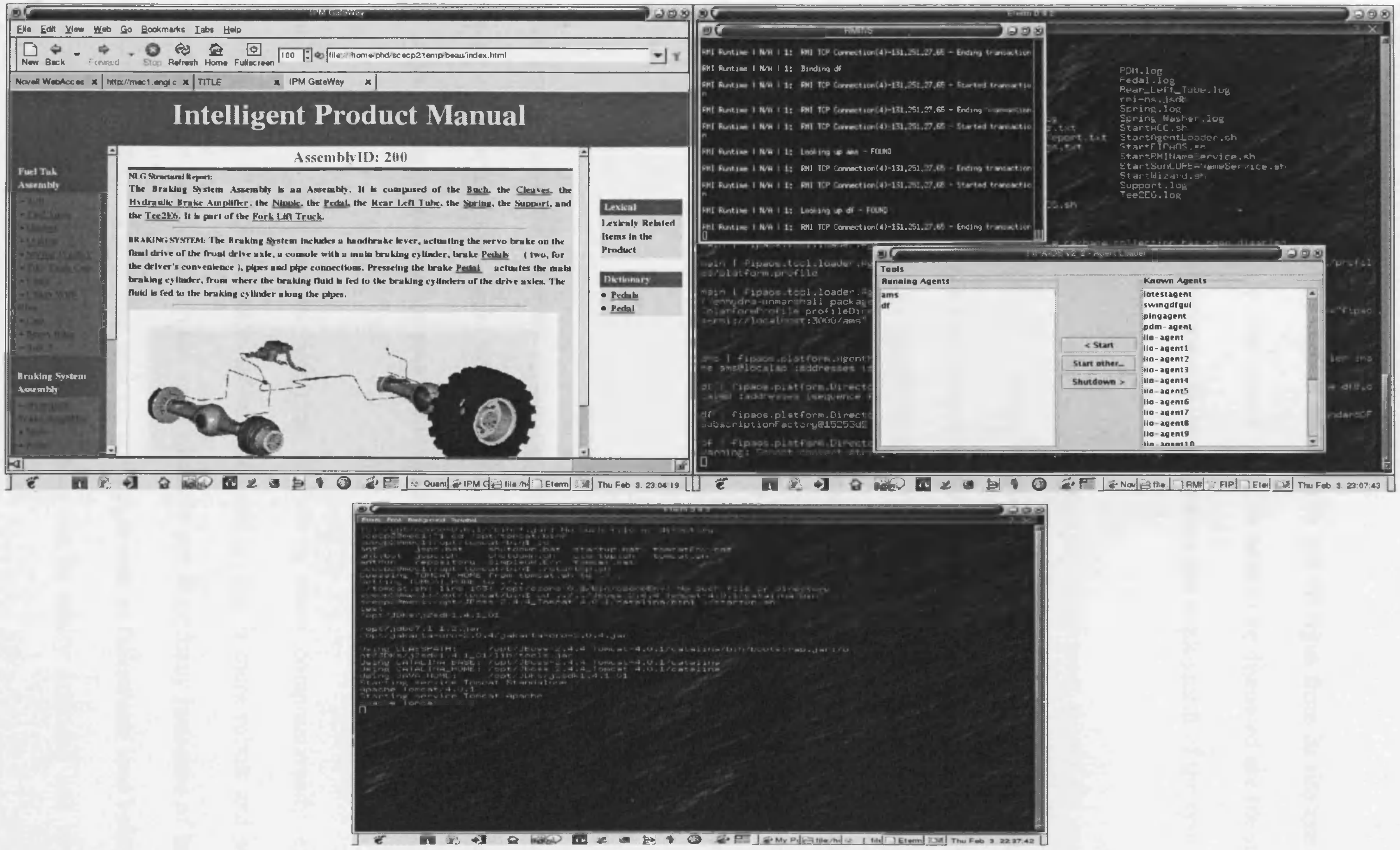
For the user interface of the system a Java Servlet approach was chosen to dynamically generate a web interface accessible from anywhere with full flexibility and customisation. The contents that the Servlet passes to the web server are requested and prepared by the community of the IIO's. One Servlet is responsible for the presentation of a side menu that presents the product structure. The IIOs cooperate to add themselves in a properly formatted HTML list that afterwards is submitted to the Servlet. The IIOs handle the main content preparing a standardised report according to their data, a textual pre-processed part, and a part that consists of natural language generated according to a template and part of the data. The IIOs also generate an adapted report that presents the XML content after it has been filtered according to the user's user-model. The user interface provides a secondary menu of lexically related links to other IIOs, and links to a dictionary of basic terms related to the IIO's name.

## 4.6 An Intelligent Information Object Case Study

For a case study, a braking system assembly has been chosen to demonstrate the IIO in action. The resulting web-site is generated on-the-fly by the relevant data and is constantly updated as the data evolve.

### 4.6.1 System Presentation

A sample of the automatically constructed web-site is shown (Figure 4.10). The first section of the IIO report is a natural language generated text describing the structural position of the part represented by this intelligent information object. The second section is a short description of the part such as the one normally existing in every PDM describing each node. Then a graphic is retrieved and presented. The content continues with a piece of text tagged by XML, specifying the knowledge type it represents. After the appropriate processing of the user model and the XML, the applicable instructions are shown to the user. Finally, on the left, the structural menu is shown, which has been created by the Intelligent Information Objects. To do this the IIOs added their names on a common repository and passed the result to the appropriate Servlet. Also, the system presents a right menu of the dictionary and lexically related items found. The star (*) next to the automatically generated links has to be selected to focus the right menu to the lexical or dictionary links that the user would like to see.



Intelligent Product Manual

AssemblyID: 200

NLG Structural Report:
The Braking System Assembly is an Assembly. It is composed of the Buch, the Cleaves, the Hydraulic Brake Amplifier, the Nipple, the Pedal, the Rear Left Tube, the Spring, the Support, and the Tee2E6. It is part of the Fork Lift Truck.

BRAKING SYSTEM: The Braking System includes a handbrake lever, actuating the servo brake on the final drive of the front drive axle, a console with a main braking cylinder, brake Pedals ( two, for the driver's convenience ), pipes and pipe connections. Presseing the brake Pedal actuates the main braking cylinder, from where the braking fluid is fed to the braking cylinders of the drive axles. The fluid is fed to the braking cylinder along the pipes.

```
RMI Runtime | N/A | 1:  RMI TCP Connection(4)-131.251.27.65 - Ending transaction
RMI Runtime | N/A | 1:  Binding df
RMI Runtime | N/A | 1:  RMI TCP Connection(4)-131.251.27.65 - Started transactio
n
RMI Runtime | N/A | 1:  RMI TCP Connection(4)-131.251.27.65 - Ending transaction
RMI Runtime | N/A | 1:  RMI TCP Connection(4)-131.251.27.65 - Started transactio
n
RMI Runtime | N/A | 1:  Looking up ams - FOUND
RMI Runtime | N/A | 1:  RMI TCP Connection(4)-131.251.27.65 - Ending transaction
RMI Runtime | N/A | 1:  RMI TCP Connection(4)-131.251.27.65 - Started transactio
n
RMI Runtime | N/A | 1:  Looking up df - FOUND
RMI Runtime | N/A | 1:  RMI TCP Connection(4)-131.251.27.65 - Ending transaction
```

```
PDM.log
Pedal.log
Rear_Left_Tube.log
rmi-ns.jsdb
Spring.log
Spring_Washer.log
StartHCC.sh
StartAgentLoader.sh
StartFIPAOS.sh
StartRMINameService.sh
StartSunLDPENameService.sh
StartWizard.sh
Support.log
Tee2E6.log
```

Running Agents:

```
ams
df
```

Known Agents:

```
iotestagent
swingdfgui
pingagent
pdm-agent
iio-agent
iio-agent1
iio-agent2
iio-agent3
iio-agent4
iio-agent5
iio-agent6
iio-agent7
iio-agent8
iio-agent9
iio-agent10
```

```
[illegible]
```

Figure 4.10 The Multi-Agent System Generated Documentation

## 4.7 Discussion

An Intelligent Information Object with an architecture that emerges from its use can present interesting results after this study. The main points that have to be discussed are the strengths and weaknesses of the approach adopted in the realisation and deployment of the system.

The effect of the IIO as an autonomous/active structuring component shows the amount of manipulations that can be achieved in automating a large part of the selection-processing-presentation cycle of documentation. The autonomy is a quality that can offer useful results allowing technical authors to deal mainly with the design of the documentation rather than select-and-paste or formatting activities. This automation on the other hand has its tradeoffs. It needs standards and agreements as to what the data selection process will be, what manipulations will be imposed on the data, and what the presentational requirements will finally be.

With these points in mind, an agent system can be addressed by other user programming, knowledge engineering, or machine learning techniques and by the development of a set of standards for the organisation of the data within the PDM.

The main strengths that can be identified include a more computationally effective organisation of the IPM architecture, a distributed system that is more robust and scalable. The system with its deployment on the agent platform can have many instances of its utility agents executed in parallel providing back-up services in case of failure and load balancing in the case of heavy traffic. Furthermore, the system can be easily extended and have new functionality added, since all the tasks executed are heavily compartmentalised within the boundaries of the responsibilities of the agents.

## 4.8 Summary

The concept of an Information Object was studied and criticised in order to identify possible shortcomings and extensions. Possible solutions to the problems identified were proposed. The effects of these solutions led to the definition of a new concept, namely an Intelligent Information Object. The effects of the Intelligent Information Object resulted in a novel architecture that gave a new perspective on Intelligent Product Manuals. A tactic for its realisation was formed and analysed. The relevant needs and technologies were specified. The system was implemented and its advantages and shortcomings were discussed. The results showed a robust, computationally effective architecture and a good combination of systemic properties that automates a large part of procedures that otherwise can be tedious and time consuming. By its nature, the system is dynamic and easily expandable to accommodate different needs and features.

# Chapter 5 - A Rule-Based Approach to Virtual Document Generation

In this chapter, a novel technique for describing virtual documents is presented. First, virtual documents are introduced and then existing techniques are discussed together with their disadvantages and shortcomings. Solutions are proposed for the problems that were identified. The consequences of these enhancements are analysed and the novelty of the proposed technique is further explained. The chapter continues with an example of an ontological description for virtual documentation and finally the proposed architecture is described. The chapter concludes with a case study.

# 5. A Rule-Based Approach to Virtual Document Generation

As mentioned previously, Virtual Documents (VDs) are documents generated electronically on demand. They can provide a report that includes the most recent data, since the data are retrieved and integrated at run-time. Also VDs can have content that is tailored according to a set of user specific parameters, which enable better adaptation and customisation of the information delivery to the end-user.

The level of abstraction in the design of the VD can vary from simple copy-and-paste of data, in a form or template such as the results of a web search-engine, to more elaborate approaches that utilise the dynamic changes to the document in more complex ways [Gruber95, Paradis98a]. A major feature of virtual documents according to Vercoustre and Paradis is that they focus on the reuse of existing data instead of duplication and include them in new documents [Vercoustre97b]. Virtual documents have been used for producing dynamic web-sites and dynamically composing explanations for the inner workings of devices on the fly based on model-based reasoning.

The literature highlights the absence of abstraction within virtual document development. The writers of Virtual Documents have to possess or acquire programming skills, in order to put together such documents in some formal computable definition or be able to use a very complex system made for this specific purpose [Gruber95].

Rule-based systems are systems that use expert knowledge codified as heuristic rules to help humans or agents to accomplish tasks that involve complex decision making [Giarratano98].

```
<HTML>

<?define $staff as sql(select * from STAFF where
portfolio="EDC")>
<?define $proj as oql(select p from Proj p where
p.portfolio="EDC")>
<?define $pub as
url(http://www.mel.dit.csiro.au/pubs/pubs.html)>

<H1>Activity report for 1997</H1>

<H2>Participants</H2>

<UL>

<LI><?map $staff.name></LI>

</UL>

<H2>Projects</H2>

<UL><?map $p in $proj>

<LI>
<H3><?$p.summary.header.title></H3>
<P><?$p.summary excluding header></P>
</LI>

</UL>

<H2>Publications</H2>

<P><?pick $onepub from $pub.H2 as $header[$i],
$pub.[$i+1].LI as $ onepub,
$staff.name as $name
where $header contains "1997", $ onepub contains $name>
</P>

</HTML>
```

**Figure 5.1: A Virtual Document Prescription**

```
<html>
<body>

<?php

$conn=odbc_connect('mydatabase','','');

if (!$conn) {
   exit("Connection Failed: " . $conn);
}

$sql="SELECT * FROM mytable";

$rs=odbc_exec($conn,$sql);

if (!$rs) {
   exit("Error in SQL");
}

echo "<table><tr>";

echo "<td>Column 1</td>";

echo "<td>Column 2</td></tr>";

while (odbc_fetch_row($rs))
{
   $data1=odbc_result($rs,"data1");
   $data2=odbc_result($rs,"data2");
   echo "<tr><td>$data1</td>";
   echo "<td>$data2</td></tr>";
}

odbc_close($conn);

echo "</table>";

?>

</body>
</html>
```

**Figure 5.2: A PHP Document Prescription**

```
#!/usr/local/bin/perl
use CGI;
use DBI;
$query = new CGI;
use CGI::Carp qw(fatalsToBrowser);

print "Content-type: text/html\n\n";
print "<html><head><title>Perl CGI Example";
print "</title></head><body>";
print "Perl CGI Example</h1><p>";

$dbh = DBI->connect("dbi:mysql:mydatabase","example","")
       or die("Couldn't connect");

$query->import_names('R');

$sth = $dbh->prepare("select * from mytable where feature = ?")
       or die("Couldn't prepare");

$sth->execute($R::myquery);

if($sth->rows == 0)
{
   print "No information for " . $R::myquery;
}
else
{
        print "<table border=2>\n";
        while( $resptr =  $sth->fetchrow_hashref() )
        {
                print "<tr>";
                print "<td>" . $resptr->{"data1"};
                print "<td>" . $resptr->{"data2"};
                print "<td>" . $resptr->{"data3"};
                print "<td>" . $resptr->{"data4"};
                print "\n";
        }

        print "</table>\n";
}
print "</body></html>\n";

$dbh->disconnect;
```

**Figure 5.3: A Perl CGI Document Prescription**

more complicated, and conditional statements can be seen in the scripts that enforce error handling in the case that the instructions given fail to execute.

The other alternative is to use a custom made system, as in the case of DME, that allows the generation of virtual documents around a very specific domain and only after a large amount of modelling and implementation work.

Existing methods of Virtual Document generation abstract the data in order to make the composition easier for the author. This abstraction level is done in most cases on top of wrappers that help with the better integration of the various data sources (SQL, OO databases, HTML, SGML etc.) and the document interpreter (Figure 5.4) handles the processing and integration of the data.

In the cases that the virtual documents are described by proper scripting or programming languages, the execution of the script can result in custom made documents according to user models, but not without significantly complicating the encoding of the document.

Moreover, the existing techniques do not focus on reuse of the code of the document prescriptions themselves. The complete focus has been devoted to the reuse of the underlying data.

The development of the documents has separated the author from the final outcome of his efforts, since by coding a document in an intermediate language the author has no direct contact with the final document viewed by the end-user.

There are five main disadvantages that can be seen in the existing VD prescription methods. First, the languages are oriented towards more complex principles than the ones needed by a pure technical or other type of writer with weak programming skills. Second, the level of abstraction of these languages drives the author into a level of detail that is not needed. Third, the document preparation is a highly decision-oriented process that involves the ordering of thematic groups in knowledge types, thus making a declarative and rule-based approach more appropriate. Fourth, the encoding used does not encourage code reusability. Finally, the Virtual Documents produced can only be seen and evaluated by the user, thus not catering for a continuous and interactive development session, as far as the writer is concerned.

### 5.1.1 Complexity of Encoding

The encoding of Virtual Documents is the major problem that has to be faced. Virtual Documents can be encoded using a variety of programming and scripting languages such as Java (Servlets), CGI (C, C++, Perl) and PhP. This is a major drawback, since it limits the authoring of such documents to highly trained programmers and alienates the traditional authors that hold all the qualifications associated with the actual art of documentation authoring.

The main attempts on the topic have been made by Vercoustre and Paradis [Vercoustre97b] who have invested significant efforts to ease the encoding of the virtual documents. They have partially left the programming approach by designing a language which is based on SGML, is simplified as much as possible, and is accompanied by an editor so as to make authoring as easy as possible. However, this procedural language is rather complicated for authors without any programming background. Furthermore, it does not support adaptation of the document according to a user model or other parameters.

### 5.1.2 Weak Level of Abstraction

The level of abstraction in the current Virtual Document techniques suffers since the authors' approach to Virtual Document design concentrates purely on the data level, having no intermediate level between raw data and the document design level.

In most cases, the data are abstracted according to the source. This means that a data repository, such as a relational or Object-Oriented database, XML database or an SGML document, is referenced and a query is made to retrieve some portion of the data included within. Therefore, the author has to be knowledgeable about the actual content, about the type of the data source, and about the way the data are stored within.

### 5.1.3 Lack of Decision Orientated Semantics

The process of documentation authoring involves activities of selection and pre-processing of existing data that have been created during the product design stages and are present within the Product Management System.

For the creation of more advanced types of documentation, the authors have to decide both among different types of manuals that focus on specific topics and about the personalisation of the manuals on standardised user models, in order to achieve just-in-time knowledge delivery and to increase the effectiveness of the manual.

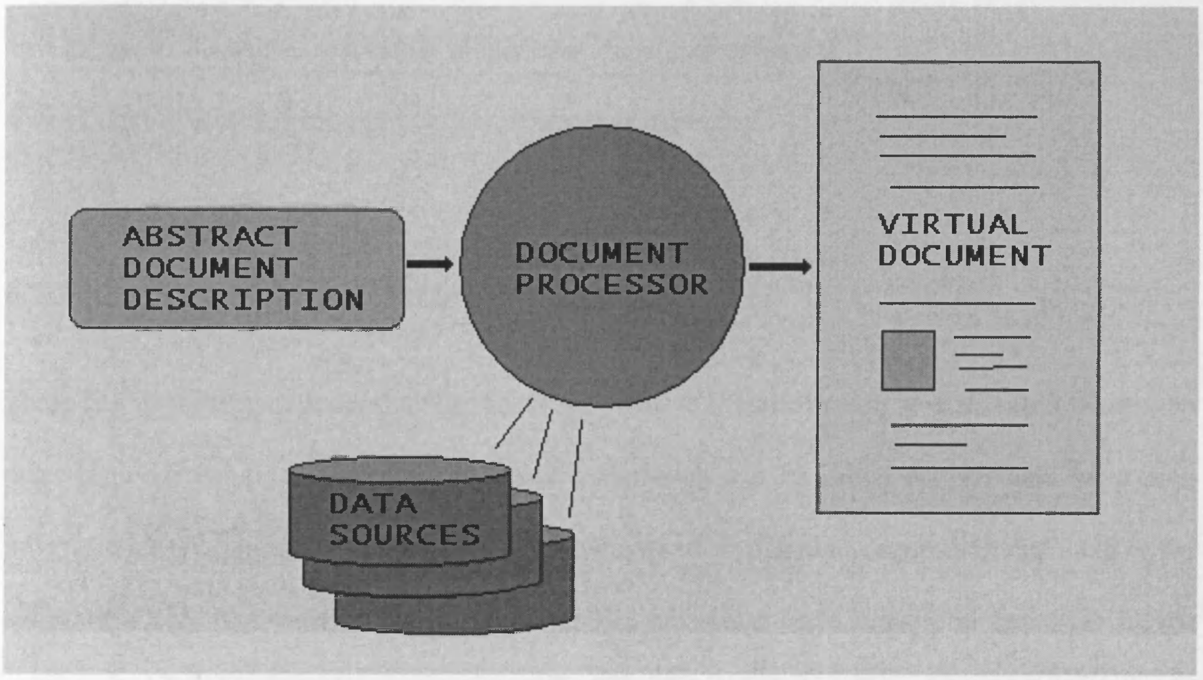


**Figure 5.4 A Typical Virtual Document Generation System**

from the final outcome and thus establish an iterative and interactive approach which would permit one to go through successive development cycles and improve the final product.

The Virtual Document is finally realised the moment the end-user requests the actual information from the server. This breaks the development cycle. The development of Virtual Documents thus cannot follow the usual iterative development which has proven to be particularly useful in all design procedures. If we compare this to the traditional software development life-cycle, it actually means that the engineer writes the software and has no way of testing to see if the application behaves as expected.

## 5.2 Enhancements to Existing Techniques

There are four main aspects that can result in positive enhancements to the methods used for describing virtual documents. A declarative approach can be more comfortable for a non-programmer. A language that will have a more decision-oriented expression can help in the adaptation of the document. Finally, knowledge reuse and code reuse can make the efforts dedicated to the process of the creation of the virtual documents much more productive.

### 5.2.1 A Declarative Approach

A solution that can ease the composition of the prescriptions of the Virtual Documents is the use of a declarative approach. Declarative approaches are more compatible with the ways a non-programmer thinks and will allow authors to express their authoring expertise more fluently.

### 5.2.2 A Knowledge-Based Approach

A knowledge-based approach can offer great benefits in the preparation of the document prescription. The main problem in the authoring phase of the Virtual Document is the fact that the designer integrates a multitude of data sources within the current prescription. Each data source may have its own type of content and it may externalise it through different interfaces (SQL, OO Languages etc.). In the case of traditional programming approaches to VD generation, the author is a programmer who has knowledge of the internal ways the data sources operate and uses an application programming interface to access the source and extract the data he/she needs to add to the document. In order to ease the development of the required prescriptions, the data source has to be concealed from the user. Therefore, the level of abstraction used in the access procedure of the data source has to be increased.

### 5.2.3 A Rule-Based Approach

A Rule-based approach is a decision-oriented approach that can offer good results. It can ease the burden of generating adapted documents, since the selection of the content according to a user model is clearly a decision making process [Turban88].

### 5.2.4 Use of Ontologies

Reuse of the code which is involved in the virtual document generation process is a vital characteristic within the proposed approach. Ontology is a candidate that can assist in knowledge reuse [Gruber93]; additionally, recycling of rules is useful.

Ontologies define a common vocabulary representing the knowledge that must be shared between different systems and/or reused at the future. An ontology defines "terms", like "person", and properties like name, address, age etc. These terms (vocabulary) can be translated into a frame-system or a relational database and thus be usable by different systems that employ the respective technology. In the definition of these terms, knowledge is abstracted in an implementation independent way, and as a result the reusability increases.

By the use of ontologies, a conceptualisation of the virtual document knowledge base is developed that can be reused and extended by knowledge engineers, in order to provide enhanced knowledge bases which support a virtual document system.

### 5.2.5 Interactive Development

In order to enable an iterative design, a number of requirements should be met. First, the dynamically generated document has to be stored so that it may be inspected by a representative of the authoring group. Second, a more automated approach would involve having embedded rules within the system which will notify the authors when a specific instance of document is generated that cannot be fully realised. Since the document is dynamically generated all the possible outcomes cannot be known and it is most probable that components of a specific combination of information objects will be missing. These rules will provide a boolean feedback (exists/not exists) to the authors and alert them into designing the specified piece of information. In addition, a more complex evaluation scheme could be used.

## 5.3 A Novel Technique for the Description of Virtual Documents

An approach that can handle the five problems which have been identified would use a set of rules encoded by the technical writers. The structure of such a system should be divided into four main subsystems, the structural, the content management, the formatting, and the evaluation (Figure 5.5).

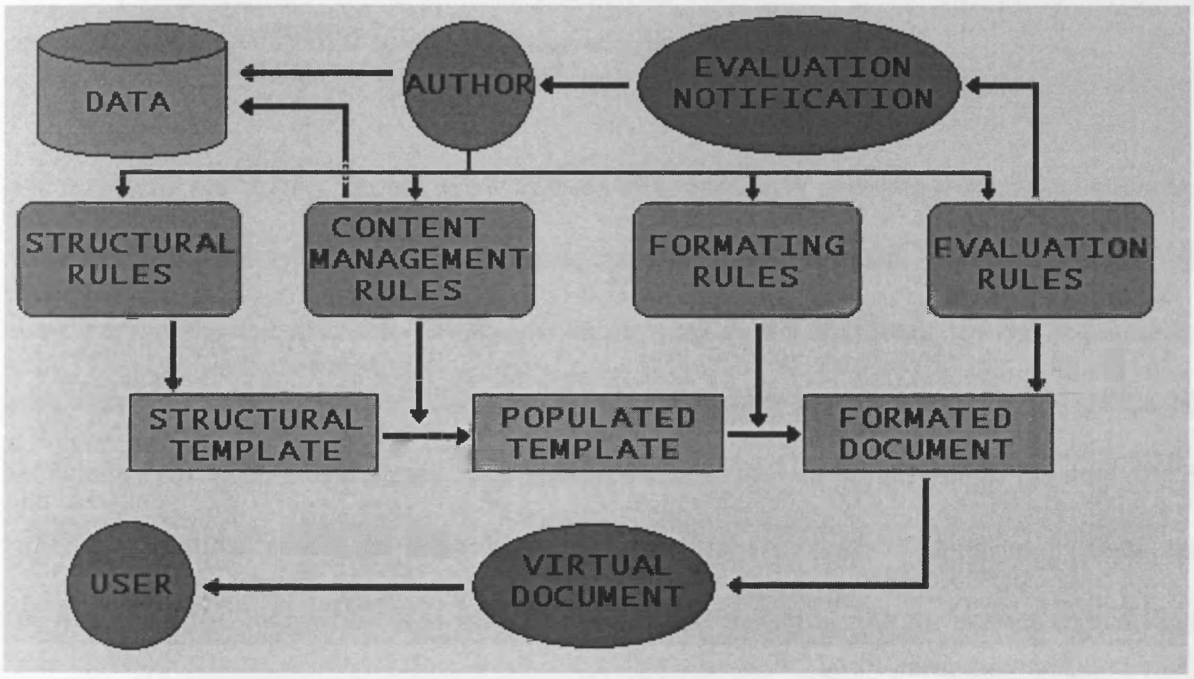


**Figure 5.5 A Rule-Based System for the Generation of Virtual Documents**

The main consideration that has to be addressed is how this system should be structured in order to facilitate the task of the developer in organising such a set of rules that would produce a Virtual Document. The most basic characteristics that are identified in a document are structure, content, and formatting. There should be separate and interdependent sets of rules concerning these three characteristics, in order to have a clearer and better system organisation that would assist in the system development. Furthermore, a set of rules operating as constraints that are met during the generation of the document should be created. These rules would evaluate the generation process and will allow the developer to receive feedback on the quality of the document that was generated.

The structural set of rules should allow authors to dynamically generate a document structure that realises the best result for the user needs and the content targeted. These rules define the titles, paragraphs and graphics that should be included in the document and the sequence in which they should be presented. The content rules define the knowledge types that should be included in each part of the structure. These types should be identified within the data of the product and reused within the document. The content should depend on the user needs and the structural positions where it is placed. The rule-set arranging the formatting should map the structural and user ontology to an appropriate format for presentation.

## 5.4 A Higher Level of Abstraction in Document Generation

The rule-based technique has several advantages when applied to the design of virtual documents. Firstly, it is a declarative approach with which the experts can encode their knowledge without getting involved with the detailed programming aspects. Secondly, since the approach is knowledge-based and focuses on knowledge types combined in documents, the approach results in a higher level of abstraction in the description of Virtual Documents.

This is more productive and closer to the way technical writers approach the task of authoring documentation.

One problem that arises from this approach is the uncertainty of whether the abstraction that is referenced exists in the system or not. This can be solved by introducing constraints, which are either satisfied or not, during the instantiation of a specific instance of a document. This data can be returned as feedback to the technical writer, in order to compensate for the lack of information or the specific knowledge structure and add it to the system.

Moreover, the productivity of this approach can be improved and the design can be facilitated through the use of an ontology and a knowledge acquisition tool or environment, such as the Protégé knowledge acquisition tool. The ontology can assist in the reuse of knowledge and the introduction of standardised framework to save work done by the document designers.

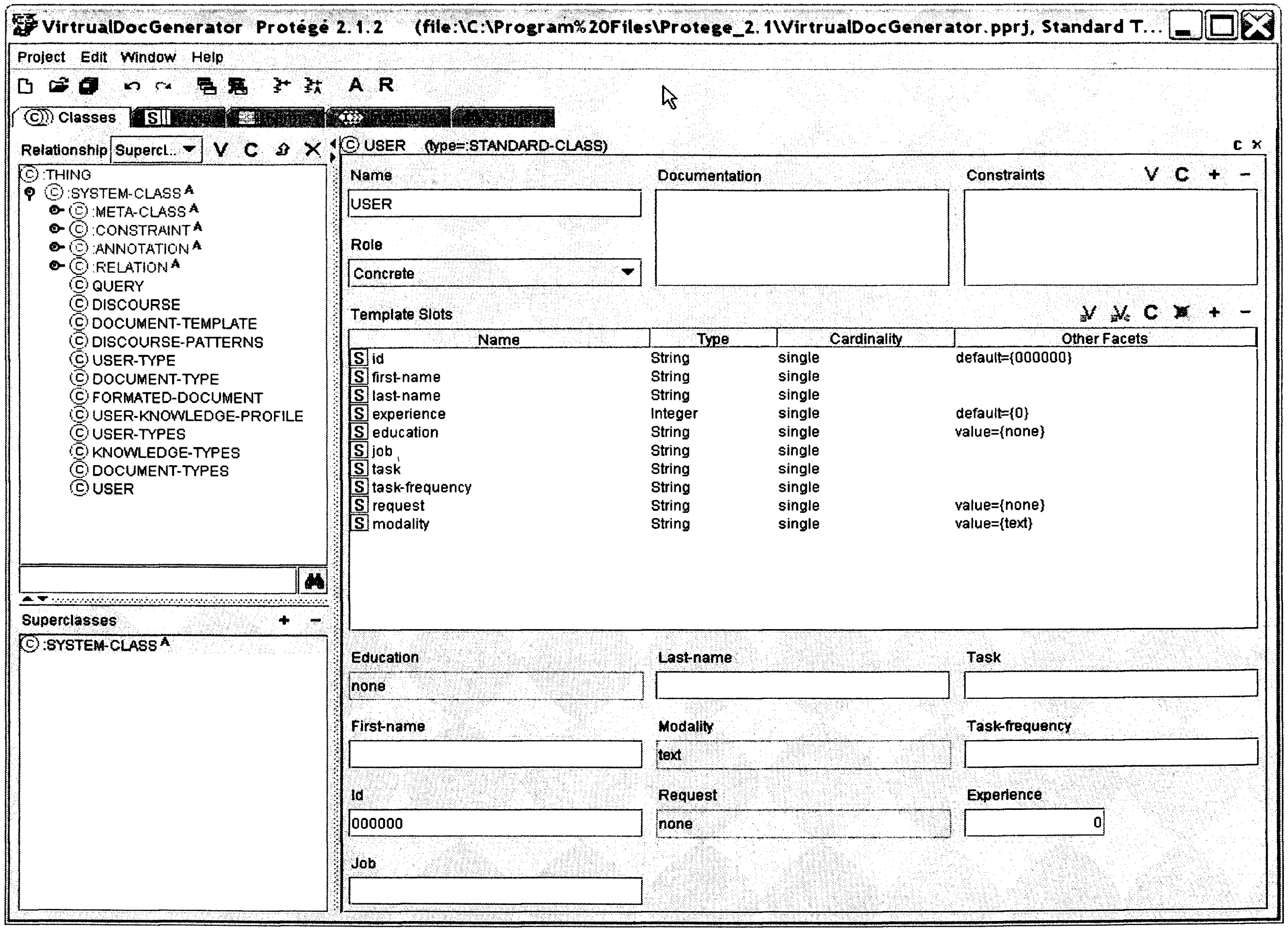
A further advantage introduced by such a system is that by nature it can be integrated with an agent system such as the one shown in Chapter 3 and thus be deployed into a more effective architecture.

## 5.5 An Ontology for Virtual Documents

Creating a knowledge-based system requires a considerable amount of work. One of the shortcomings of a rule-based system for the generation of virtual documents would be that its development is time-consuming and labour intensive. In order to increase the productivity of the knowledge-based description, the creation of an ontology describing the domain is proposed.

lists knowledge types the user is expected to be more familiar with in order to provide emphasis on the less known. The "set user type" set of rules is mapping attributes of the user model to a predefined set of user types that qualify the user to a role associated to the product. The user model includes a more specific query attribute that is handled by the "get user query" module and sets the query template with details for the current query. Upon the query and the user type specifications a discourse pattern is selected by the "select discourse pattern module". The discourse pattern is a sequence of knowledge types that are considered appropriate for presentation to the specified user type and the query submitted to the system. These are listed in the "discourse patterns" template. Based on the user type, a document structure is selected such as "Technical Specification Manual" or "Maintenance Manual" with the "select document structure" rules. The different types of possible documents are specified in a "document types" template. The "document template" is instantiated by populating the document structure implied by the document type with the respective discourse pattern. The instantiated document is then processed by the "format document" module that applies formatting rules to the document structure based on document structure elements, such as titles and paragraphs, and also the knowledge types that are defined in the user knowledge profile. Finally, the formatted document is exported to be processed by the document processor that will populate it with information objects.

The common ground between the Protégé ontology and the CLIPS system is that each of the classes declared in the Protégé translates into a CLIPS template as shown in Figure 5.6(b). The CLIPS code defines the template and the corresponding slots that represent attributes with which the user is characterised. As an example, the ontology designed translates to 300 lines of code declaring the templates and facts of the rule-based system.


VirtrualDocGenerator Protégé 2.1.2 (file:\C:\Program%20Files\Protege_2.1\VirtrualDocGenerator.pprj, Standard T...
Project Edit Window Help
A R
Classes
Relationship Supercl...
V C
Superclasses
C USER (type=STANDARD-CLASS)

:THING
- :SYSTEM-CLASS A
  - :META-CLASS A
  - :CONSTRAINT A
  - :ANNOTATION A
  - :RELATION A
  - QUERY
  - DISCOURSE
  - DOCUMENT-TEMPLATE
  - DISCOURSE-PATTERNS
  - USER-TYPE
  - DOCUMENT-TYPE
  - FORMATED-DOCUMENT
  - USER-KNOWLEDGE-PROFILE
  - USER-TYPES
  - KNOWLEDGE-TYPES
  - DOCUMENT-TYPES
  - USER

Superclasses: :SYSTEM-CLASS A

Name: USER

Role: Concrete

Documentation:

Constraints:

Template Slots

| Name | Type | Cardinality | Other Facets |
|---|---|---|---|
| S id | String | single | default={000000} |
| S first-name | String | single | |
| S last-name | String | single | |
| S experience | Integer | single | default={0} |
| S education | String | single | value={none} |
| S job | String | single | |
| S task | String | single | |
| S task-frequency | String | single | |
| S request | String | single | value={none} |
| S modality | String | single | value={text} |

| | | |
|---|---|---|
| Education: none | Last-name: | Task: |
| First-name: | Modality: text | Task-frequency: |
| Id: 000000 | Request: none | Experience: 0 |
| Job: | | |

Figure 5.6(a) The Protégé 2000 Ontology Development Environment [Noy2001a]

```
; The User Model of the current user
(deftemplate USER "The User Model"
     (slot id (type INTEGER))
     (slot first-name (type STRING))
     (slot last-name (type STRING))
     (slot experience (type INTEGER))
     (slot education (type STRING))
     (slot job (type STRING))
     (slot task (type STRING))
     (slot task-frequency (type INTEGER))
     (slot request (type STRING))
     (slot modality (type STRING))
)
```

**Figure 5.6(b) A CLIPS Template created for a Protégé class**

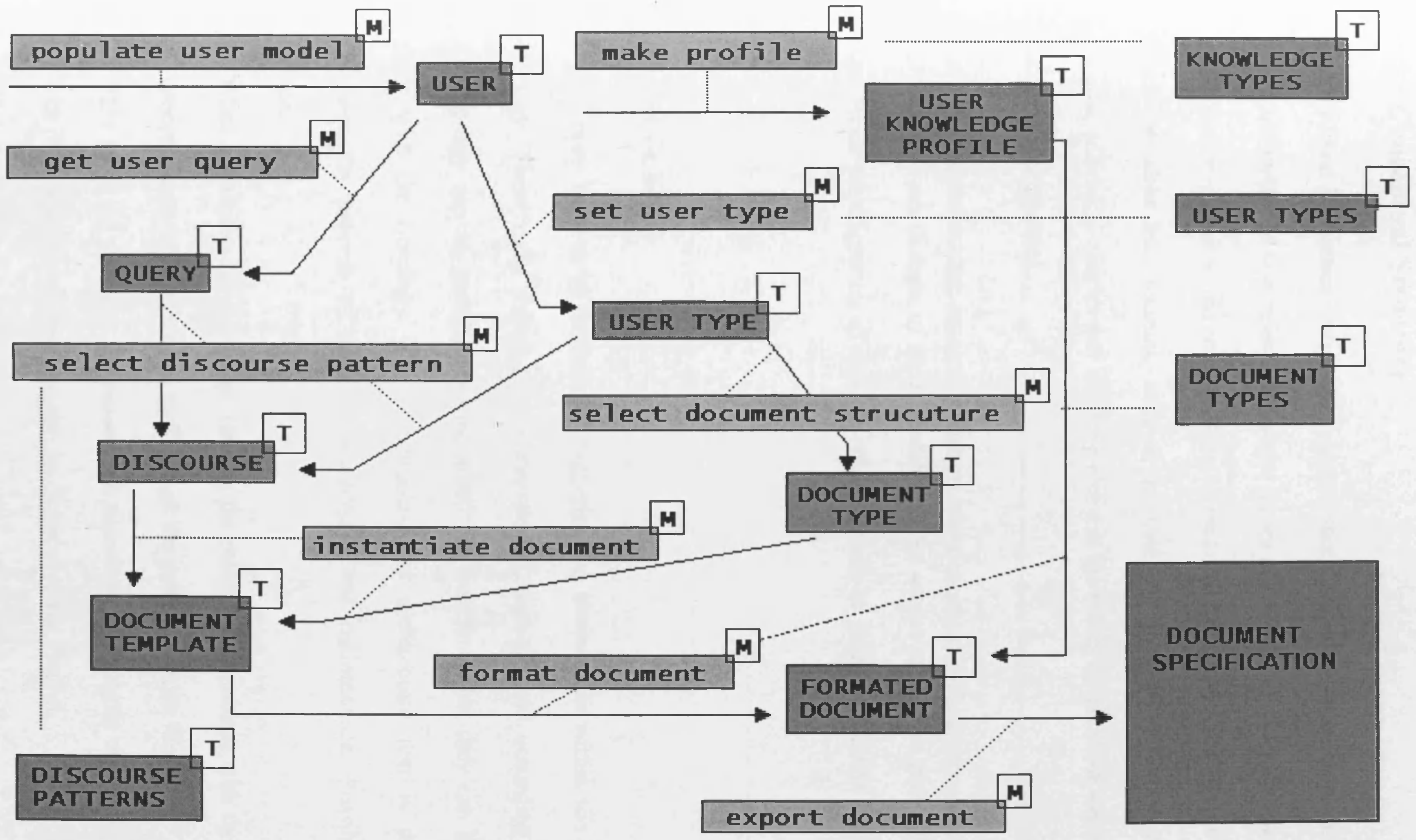


Figure 5.7 Virtual Document Ontology

### 5.5.1 Conceptual Structure

In the Virtual Document Ontology, concepts should appear that allow modelling the document itself, and that refer to structural issues as well as the presentational and the qualitative properties of the content. Also, as part of the document, extension concepts should be taken into account, such as the user and his characteristics, discourse patterns, additional queries and anything else that the author might think can influence the document qualities.

All the above are thought of as interacting entities whose properties and their current state decide the triggering of specific rules, in order to create a specific instance of a document.

### 5.5.2 Rule Sets

Sets of rules have to be defined for triggering the interactions within the designed ontology. These rules follow a condition-action strategy, and according to their purpose they can be grouped into modules. The benefit is that they can be reused along with the ontology. The component-based architecture that is developed alleviates the process of system management and maintenance. Possible cases include:

1. When concepts of the ontology change, the rules that are related with the updated concepts have to be altered as well, but all the rest can remain the same.
2. Portions of the ontology are reused and therefore the modules that are associated with the respective entities can also be included in the system.

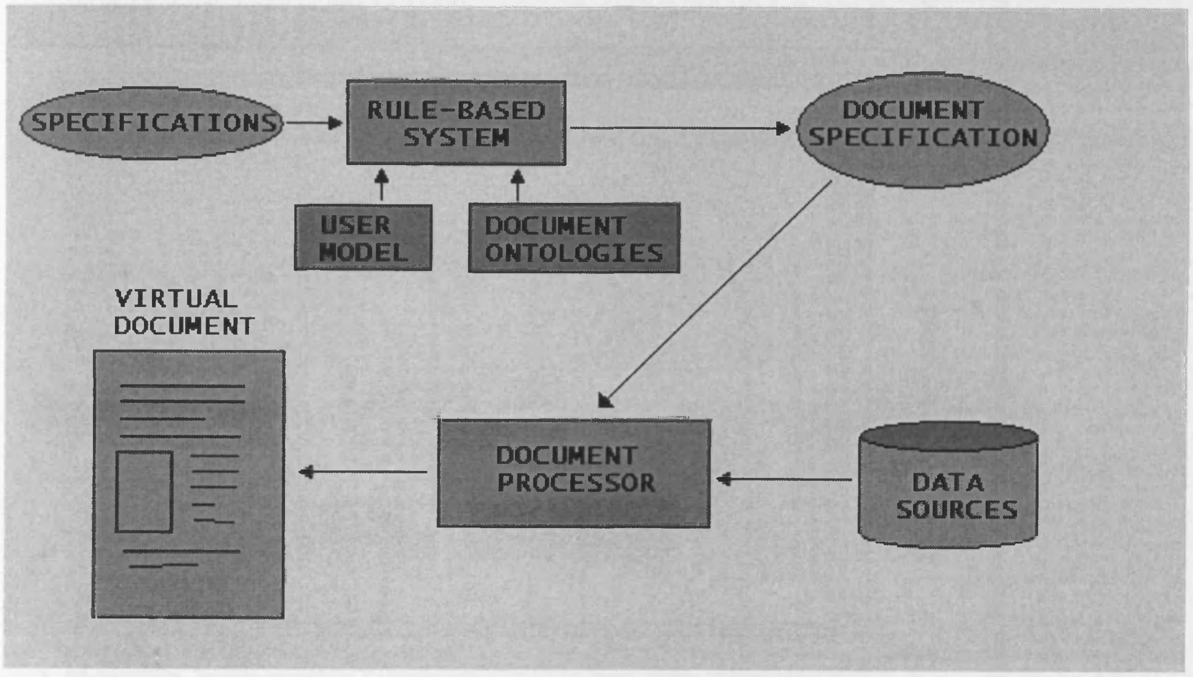
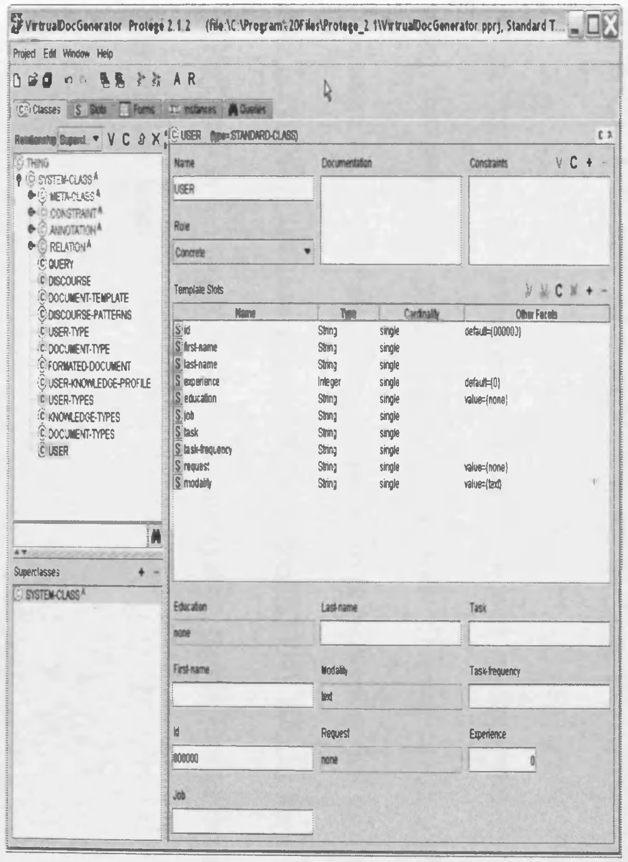


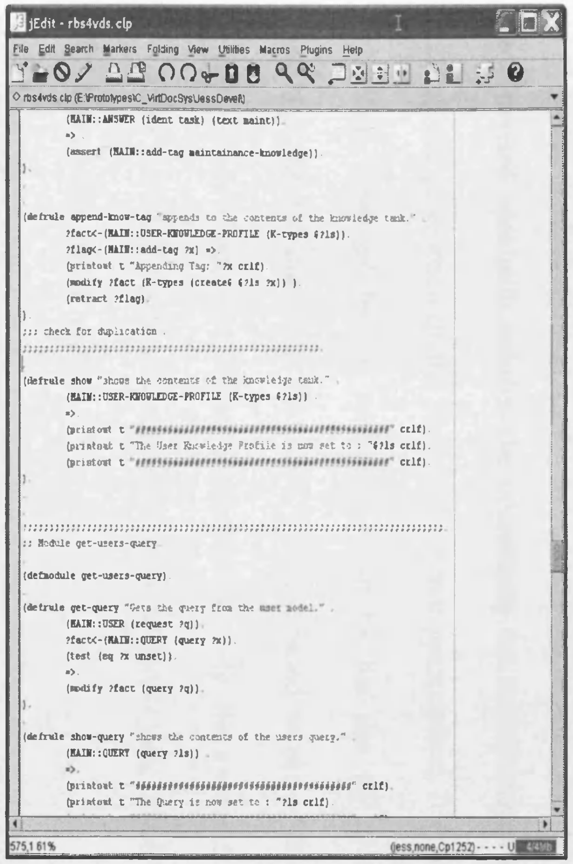
**Figure 5.8 A Rule-Based Virtual Document System Architecture**



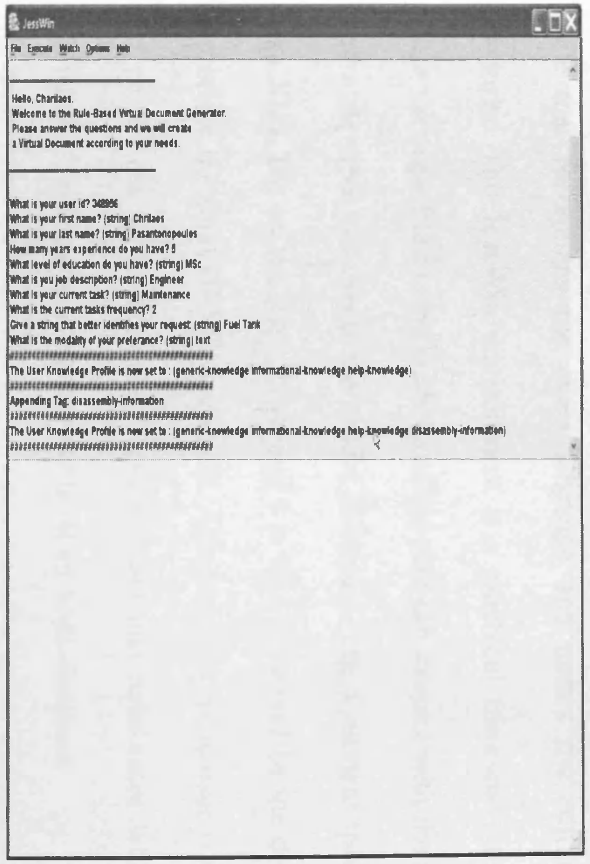
Name: USER
Role: Concrete

| Name | Type | Cardinality | Other Facets |
|---|---|---|---|
| id | String | single | default={000000} |
| first-name | String | single | |
| last-name | String | single | |
| experience | Integer | single | default={0} |
| education | String | single | value={none} |
| job | String | single | |
| task | String | single | |
| task-frequency | String | single | |
| request | String | single | value={none} |
| modality | String | single | value={text} |

| | | |
|---|---|---|
| Education: none | Last-name: | Task: |
| First-name: | Modality: text | Task-frequency: |
| Id: 000000 | Request: none | Experience: 0 |
| Job: | | |


jEdit - rbs4vds.clp
File Edit Search Markers Folding View Utilities Macros Plugins Help
rbs4vds.clp (E:\Prototypes\C_VirtDocSys\JessDevel)
575,1 61%
(jess,none,Cp1252) - - - - U

```
        (MAIN::ANSWER (ident task) (text maint))
        =>
        (assert (MAIN::add-tag maintainance-knowledge))
)


(defrule append-know-tag "appends to the contents of the knowledge task."
        ?fact<-(MAIN::USER-KNOWLEDGE-PROFILE (K-types $?ls))
        ?flag<-(MAIN::add-tag ?x) =>
        (printout t "Appending Tag: "?x crlf)
        (modify ?fact (K-types (create$ $?ls ?x)) )
        (retract ?flag)
)
;;; check for duplication
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule show "shows the contents of the knowledge task."
        (MAIN::USER-KNOWLEDGE-PROFILE (K-types $?ls))
        =>
        (printout t "##############################################" crlf)
        (printout t "The User Knowledge Profile is now set to : "$?ls crlf)
        (printout t "##############################################" crlf)
)


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Module get-users-query

(defmodule get-users-query)

(defrule get-query "Gets the query from the user model."
        (MAIN::USER (request ?q))
        ?fact<-(MAIN::QUERY (query ?x))
        (test (eq ?x unset))
        =>
        (modify ?fact (query ?q))
)

(defrule show-query "shows the contents of the users query."
        (MAIN::QUERY (query ?ls))
        =>
        (printout t "##########################################" crlf)
        (printout t "The Query is now set to : "?ls crlf)
```



```
Hello, Charilaos.
Welcome to the Rule-Based Virtual Document Generator.
Please answer the questions and we will create
a Virtual Document according to your needs.

What is your user id? 348956
What is your first name? (string) Chrilaos
What is your last name? (string) Pasantonopoulos
How many years experience do you have? 5
What level of education do you have? (string) MSc
What is you job description? (string) Engineer
What is your current task? (string) Maintenance
What is the current tasks frequency? 2
Give a string that better identifies your request: (string) Fuel Tank
What is the modality of your preferance? (string) text
##############################################
The User Knowledge Profile is now set to : (generic-knowledge informational-knowledge help-knowledge)
##############################################
Appending Tag: disassembly-information
##############################################
The User Knowledge Profile is now set to : (generic-knowledge informational-knowledge help-knowledge disassembly-information)
##############################################
```

**JessWin** **JEdit on JESS Mode** **Protégé Environment**

**Figure 5.9 A Rule-Based Virtual Document System Interface Components**

mode that recognizes the JESS language and makes the editing easier for the developer. The JessWin environment is a graphical front-end for the Jess engine, where the clips file can be loaded and the user can interact with the rule-based system to provide input and create the virtual document. The document specification is saved as an XML file on the file system and it is then processed by the document processor to produce the final HTML document. The document processor is a Java application that parses the XML document specification and substitutes with the appropriate information objects that are retrieved from an XML database.

The proposed approach reduces the complexity by introducing a rule-based system handling the generation of the virtual document prescription. It enhances the level of abstraction observed by the authoring team by the use of conceptual structures describing the document and the users. The rule based nature of the system promotes the description of adaptive virtual documents. Finally, the approach enables the reuse of code in the level of document prescriptions by employing ontologies in the design of the knowledge-based system.

## 5.7 A Case-Study of a Rule-Based Virtual Document System

As an example, the document ontology provided is employed to create a document using the technique proposed in this chapter (Figure 5.10). Following the usual procedure of an expert system, the program presents a series of questions to acquire data concerning the user (populate-user-model). This data is kept in the user model (USER) and is the means with which the system makes further decisions based on its internal knowledge.

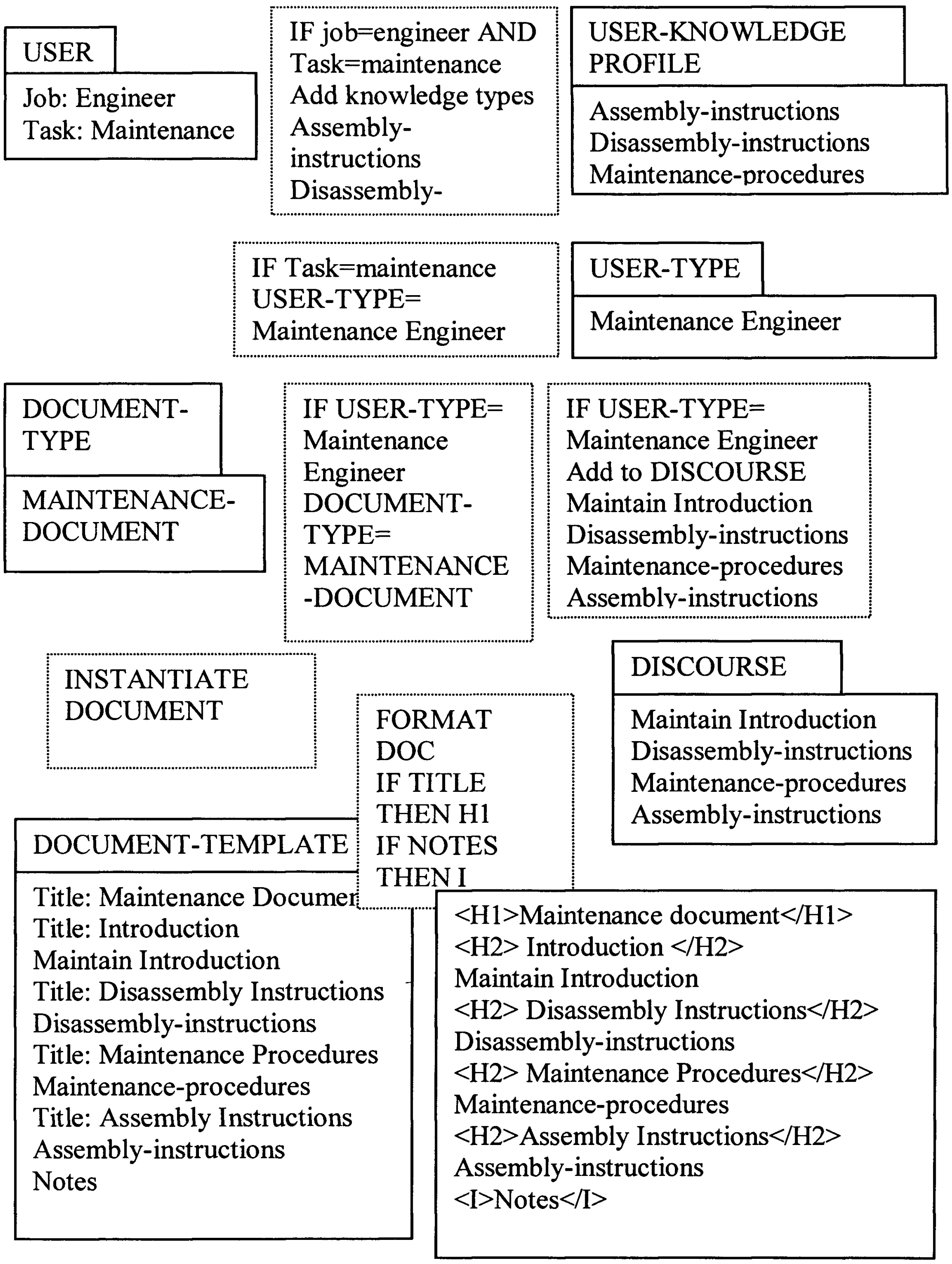

USER
Job: Engineer
Task: Maintenance
IF job=engineer AND Task=maintenance Add knowledge types Assembly-instructions Disassembly-
USER-KNOWLEDGE PROFILE
Assembly-instructions
Disassembly-instructions
Maintenance-procedures
IF Task=maintenance USER-TYPE= Maintenance Engineer
USER-TYPE
Maintenance Engineer
DOCUMENT-TYPE
MAINTENANCE-DOCUMENT
IF USER-TYPE= Maintenance Engineer DOCUMENT-TYPE= MAINTENANCE -DOCUMENT
IF USER-TYPE= Maintenance Engineer Add to DISCOURSE Maintain Introduction Disassembly-instructions Maintenance-procedures Assembly-instructions
INSTANTIATE DOCUMENT
DISCOURSE
Maintain Introduction
Disassembly-instructions
Maintenance-procedures
Assembly-instructions
FORMAT DOC IF TITLE THEN H1 IF NOTES THEN I
DOCUMENT-TEMPLATE
Title: Maintenance Documen
Title: Introduction
Maintain Introduction
Title: Disassembly Instructions
Disassembly-instructions
Title: Maintenance Procedures
Maintenance-procedures
Title: Assembly Instructions
Assembly-instructions
Notes
<H1>Maintenance document</H1>
<H2> Introduction </H2>
Maintain Introduction
<H2> Disassembly Instructions</H2>
Disassembly-instructions
<H2> Maintenance Procedures</H2>
Maintenance-procedures
<H2>Assembly Instructions</H2>
Assembly-instructions
<I>Notes</I>

**Figure 5.10 A Rule-Based Virtual Document Example**

The system holds a pool of knowledge types (KNOWLEDGE-TYPES) from which a subset is selected with the use of the "make-profile" module of rules (make-profile) and the resulting knowledge profile (USER-KNOWLEDGE-PROFILE) is being stored. The user type (USER-TYPE) is decided from an existing set of user types (USER-TYPES) and the set-user-type (set-user-types) rules.

Based upon the user type a document type (DOCUMENT-TYPE) is selected (select-document-structure) from the set of defined document types (DOCUMENT-TYPES) within the system. This describes the document and implies a document structure that is realized later on by the system.

The user type is used in conjunction with the query to select a discourse pattern (DISCOURSE) that describes the information objects and their sequence. This pattern will then be presented to the user. A set of defined discourse patterns (DISCOURSE-PATTERNS) is used to choose the appropriate pattern for the current problem.

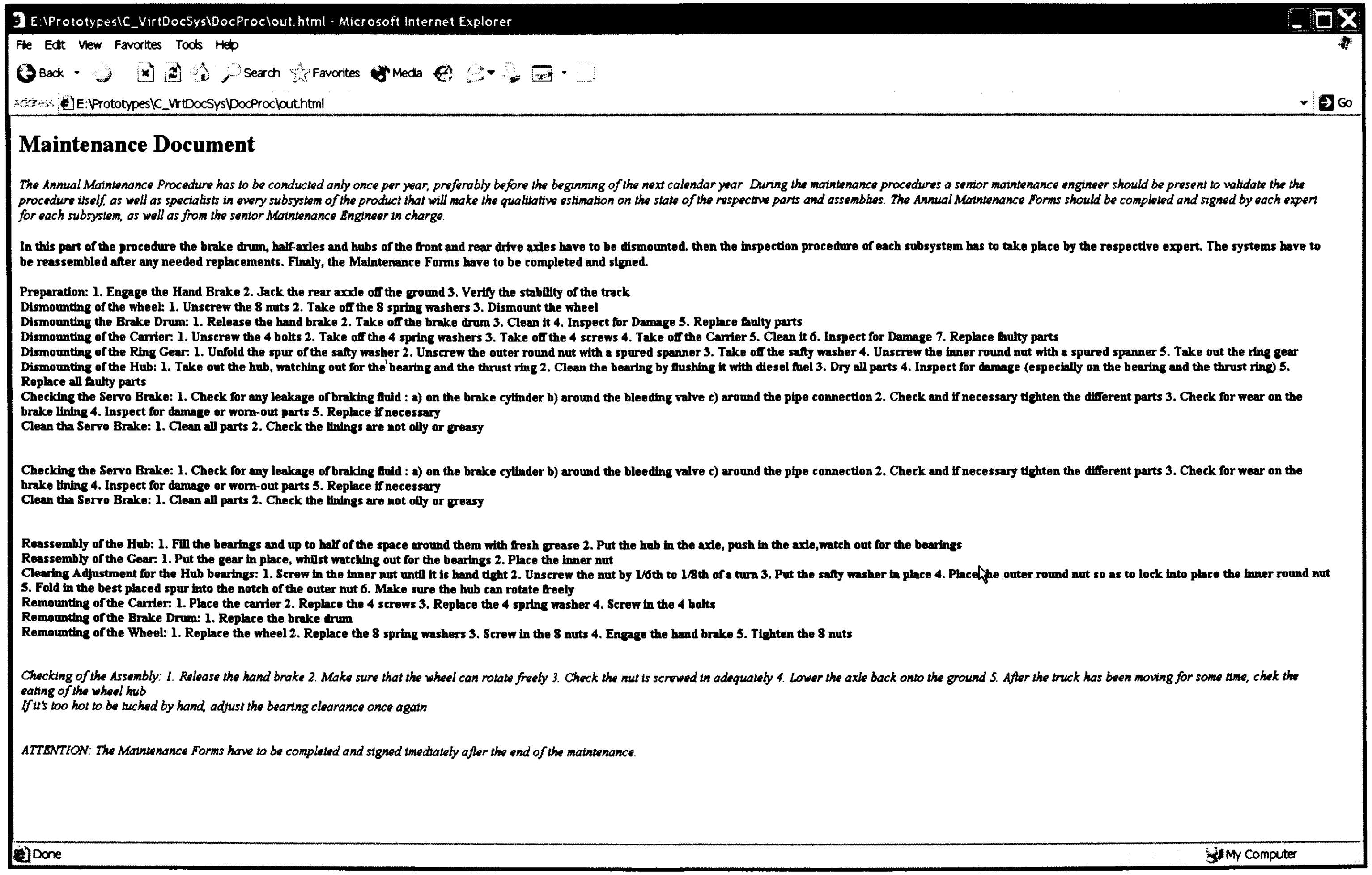
The document type and the discourse are used as the basis for the instantiation of the document by the "instantiate document" procedure. This is the only part of the system that deviates from the declarative rule-based strategy in order to tackle more easily the sequential nature of the actions that need to be taken. At this point, a procedural construct is used to make the mapping between the discourse and the document structure and produce the final document template (DOCUMENT-TEMPLATE). The document template is populated by a variable number of sections that match the respective entries in the discourse pattern.

Finally, the template is processed with a formatting module (format-document) of rules that map the different document items to a specific formatting. The final outcome of this process is the formatted document prescription (FORMATTED-DOCUMENT). This can now be exported by the system and is ready to be used as the prescription that the virtual document processor instantiates.

In Figures 5.11 and 5.12, two instantiations can be seen presenting maintenance information to a novice and a senior maintenance engineer. For the novice, a more extended document is presented that starts with introductory information, then shows the main disassembly, assembly and inspection instructions and closes with reminders and final notes. The senior maintenance engineer only gets the main instructions and graphics that will provide essential information in the fastest possible way.

## 5.8 Discussion

In this chapter, a novel technique for the generation of virtual documents was introduced. This new technique simplifies the encoding of virtual documents, increases the level of abstraction, introduces code reuse in the description of virtual documents, and applies a decision-oriented strategy in their development.


E:\Prototypes\C_VirtDocSys\DocProc\out.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address E:\Prototypes\C_VirtDocSys\DocProc\out.html
Go
Done
My Computer

## Maintenance Document

*The Annual Maintenance Procedure has to be conducted anly once per year, preferably before the beginning of the next calendar year. During the maintenance procedures a senior maintenance engineer should be present to validate the the procedure itself, as well as specialists in every subsystem of the product that will make the qualitative estimation on the state of the respective parts and assemblies. The Annual Maintenance Forms should be completed and signed by each expert for each subsystem, as well as from the senior Maintenance Engineer in charge.*

**In this part of the procedure the brake drum, half-axles and hubs of the front and rear drive axles have to be dismounted. then the inspection procedure of each subsystem has to take place by the respective expert. The systems have to be reassembled after any needed replacements. Finaly, the Maintenance Forms have to be completed and signed.**

**Preparation: 1. Engage the Hand Brake 2. Jack the rear axxle off the ground 3. Verify the stability of the track**
**Dismounting of the wheel: 1. Unscrew the 8 nuts 2. Take off the 8 spring washers 3. Dismount the wheel**
**Dismounting the Brake Drum: 1. Release the hand brake 2. Take off the brake drum 3. Clean it 4. Inspect for Damage 5. Replace faulty parts**
**Dismounting of the Carrier: 1. Unscrew the 4 bolts 2. Take off the 4 spring washers 3. Take off the 4 screws 4. Take off the Carrier 5. Clean it 6. Inspect for Damage 7. Replace faulty parts**
**Dismounting of the Ring Gear: 1. Unfold the spur of the safty washer 2. Unscrew the outer round nut with a spured spanner 3. Take off the safty washer 4. Unscrew the inner round nut with a spured spanner 5. Take out the ring gear**
**Dismounting of the Hub: 1. Take out the hub, watching out for the bearing and the thrust ring 2. Clean the bearing by flushing it with diesel fuel 3. Dry all parts 4. Inspect for damage (especially on the bearing and the thrust ring) 5. Replace all faulty parts**
**Checking the Servo Brake: 1. Check for any leakage of braking fluid : a) on the brake cylinder b) around the bleeding valve c) around the pipe connection 2. Check and if necessary tighten the different parts 3. Check for wear on the brake lining 4. Inspect for damage or worn-out parts 5. Replace if necessary**
**Clean tha Servo Brake: 1. Clean all parts 2. Check the linings are not oily or greasy**

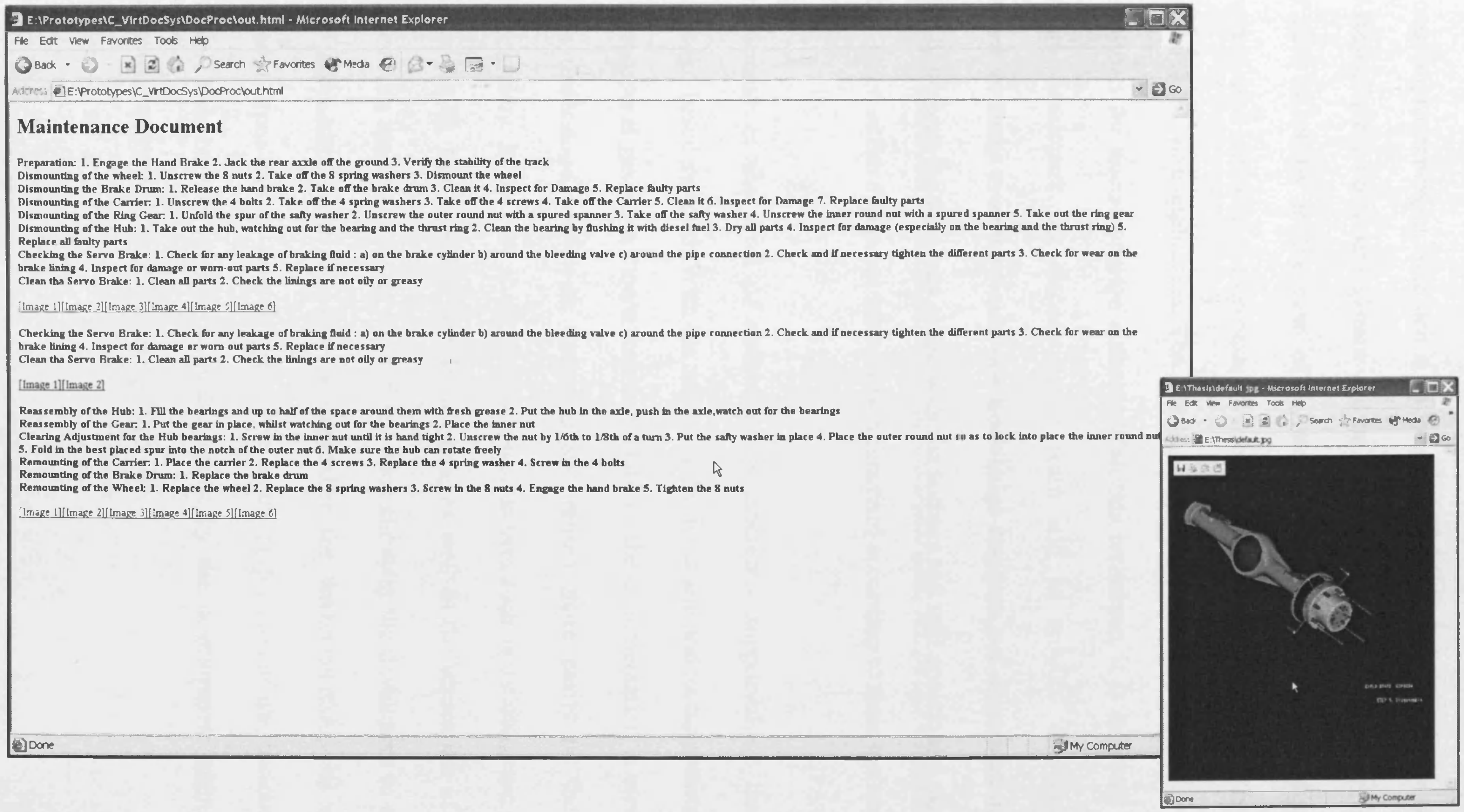
**Checking the Servo Brake: 1. Check for any leakage of braking fluid : a) on the brake cylinder b) around the bleeding valve c) around the pipe connection 2. Check and if necessary tighten the different parts 3. Check for wear on the brake lining 4. Inspect for damage or worn-out parts 5. Replace if necessary**
**Clean tha Servo Brake: 1. Clean all parts 2. Check the linings are not oily or greasy**

**Reassembly of the Hub: 1. Fill the bearings and up to half of the space around them with fresh grease 2. Put the hub in the axle, push in the axle,watch out for the bearings**
**Reassembly of the Gear: 1. Put the gear in place, whilst watching out for the bearings 2. Place the inner nut**
**Clearing Adjustment for the Hub bearings: 1. Screw in the inner nut until it is hand tight 2. Unscrew the nut by 1/6th to 1/8th of a turn 3. Put the safty washer in place 4. Place the outer round nut so as to lock into place the inner round nut 5. Fold in the best placed spur into the notch of the outer nut 6. Make sure the hub can rotate freely**
**Remounting of the Carrier: 1. Place the carrier 2. Replace the 4 screws 3. Replace the 4 spring washer 4. Screw in the 4 bolts**
**Remounting of the Brake Drum: 1. Replace the brake drum**
**Remounting of the Wheel: 1. Replace the wheel 2. Replace the 8 spring washers 3. Screw in the 8 nuts 4. Engage the hand brake 5. Tighten the 8 nuts**

*Checking of the Assembly: 1. Release the hand brake 2. Make sure that the wheel can rotate freely 3. Check the nut is screwed in adequately 4. Lower the axle back onto the ground 5. After the truck has been moving for some time, chek the eating of the wheel hub*
*If it's too hot to be tuched by hand, adjust the bearing clearance once again*

*ATTENTION: The Maintenance Forms have to be completed and signed imediately after the end of the maintenance.*

**Figure 5.11 An Instantiation of a Rule-Based Virtual Document**


E:\Prototypes\C_VirtDocSys\DocProc\out.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address E:\Prototypes\C_VirtDocSys\DocProc\out.html
Go
Done
My Computer
E:\Thesis\default.jpg - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address E:\Thesis\default.jpg
Done
My Computer

Maintenance Document

Preparation: 1. Engage the Hand Brake 2. Jack the rear axxle off the ground 3. Verify the stability of the track
Dismounting of the wheel: 1. Unscrew the 8 nuts 2. Take off the 8 spring washers 3. Dismount the wheel
Dismounting the Brake Drum: 1. Release the hand brake 2. Take off the brake drum 3. Clean it 4. Inspect for Damage 5. Replace faulty parts
Dismounting of the Carrier: 1. Unscrew the 4 bolts 2. Take off the 4 spring washers 3. Take off the 4 screws 4. Take off the Carrier 5. Clean it 6. Inspect for Damage 7. Replace faulty parts
Dismounting of the Ring Gear: 1. Unfold the spur of the safty washer 2. Unscrew the outer round nut with a spured spanner 3. Take off the safty washer 4. Unscrew the inner round nut with a spured spanner 5. Take out the ring gear
Dismounting of the Hub: 1. Take out the hub, watching out for the bearing and the thrust ring 2. Clean the bearing by flushing it with diesel fuel 3. Dry all parts 4. Inspect for damage (especially on the bearing and the thrust ring) 5. Replace all faulty parts
Checking the Servo Brake: 1. Check for any leakage of braking fluid : a) on the brake cylinder b) around the bleeding valve c) around the pipe connection 2. Check and if necessary tighten the different parts 3. Check for wear on the brake lining 4. Inspect for damage or worn-out parts 5. Replace if necessary
Clean tha Servo Brake: 1. Clean all parts 2. Check the linings are not oily or greasy

[Image 1][Image 2][Image 3][Image 4][Image 5][Image 6]

Checking the Servo Brake: 1. Check for any leakage of braking fluid : a) on the brake cylinder b) around the bleeding valve c) around the pipe connection 2. Check and if necessary tighten the different parts 3. Check for wear on the brake lining 4. Inspect for damage or worn-out parts 5. Replace if necessary
Clean tha Servo Brake: 1. Clean all parts 2. Check the linings are not oily or greasy

[Image 1][Image 2]

Reassembly of the Hub: 1. Fill the bearings and up to half of the space around them with fresh grease 2. Put the hub in the axle, push in the axle,watch out for the bearings
Reassembly of the Gear: 1. Put the gear in place, whilst watching out for the bearings 2. Place the inner nut
Clearing Adjustment for the Hub bearings: 1. Screw in the inner nut until it is hand tight 2. Unscrew the nut by 1/6th to 1/8th of a turn 3. Put the safty washer in place 4. Place the outer round nut so as to lock into place the inner round nut 5. Fold in the best placed spur into the notch of the outer nut 6. Make sure the hub can rotate freely
Remounting of the Carrier: 1. Place the carrier 2. Replace the 4 screws 3. Replace the 4 spring washer 4. Screw in the 4 bolts
Remounting of the Brake Drum: 1. Replace the brake drum
Remounting of the Wheel: 1. Replace the wheel 2. Replace the 8 spring washers 3. Screw in the 8 nuts 4. Engage the hand brake 5. Tighten the 8 nuts

[Image 1][Image 2][Image 3][Image 4][Image 5][Image 6]

Figure 5.12 A Rule-Based Virtual Document Example Output

The simplification of the encoding is significant from the point of view that the usual procedural programming approach is no longer needed. However, this does not mean that the complete document system will be easily constructed by non programmers. The gain is mostly in the ease of converting the instructions/knowledge of the documentation author to a rule-based system as opposed to developing a traditional procedural software application. The rule-base of the system should be relatively easy to construct for documentation authors. With this technique, it is expected that the software developers per documentation team will be reduced drastically. The approach is likely to require only one knowledge engineer per documentation team, who will prepare the ontology along with the authors and will encode the parts of the system (such as the document instantiation function) according to their instructions.

The increase in the level of abstraction is inherently supported by the use of knowledge-based systems. With the design of the document and its supporting entities as a conceptual model, a clearer representation of the documentation is created that can be understood, reviewed, extended and reused more easily by the human authoring team. In the case study, conceptual structures such as the discourse patterns, the user model, the user and document types, as well as the separation of rules in modules with specific purpose are defined, thus allowing the developers to create an higher abstraction layer and also conceptualise the documentation, the user, the knowledge types and the processes involved in the preparation of the documentation. However, this benefit implies an added effort by the development team and an involvement of knowledge engineering expertise.

The added knowledge-engineering work is counter-balanced by the use of the documentation ontology. The ontology allows the reuse of these models and thus increases greatly the productivity of the process. The ontology can be reviewed, enhanced and reused by graphically manipulating the conceptual structure in the Protégé environment. This representation translates to reusable CLIPS templates and fact declarations that can be directly included to the rule-based system code. In addition, they result in reusable code for the rule based system.

The decision-oriented strategy of the rule-based system can be questioned for the normal and simpler virtual documents. As long as the focus moves to adapted virtual documents, where the document is tailor-made according to a user profile, the facts are dramatically changed. In this case, the benefit of the decision-oriented strategy of rules is obvious for the selection of the information and structure, as well as for formatting and for introducing additional parameters that might influence the document generated.

Finally, it can be said that this technique might not bring virtual document generation to the end user, but it can significantly improve the state of virtual documentation from the developers' perspective. The improvement anticipated is proportional to the work done on the ontologies and the standards for documentation systems.

## 5.9 Summary

Existing Virtual Document techniques have been studied and analyzed. Some shortcomings have been identified, such as the complexity of encoding, the weak level of abstraction, lack of decision-oriented strategy, lack of code reuse and non-interactive development cycle. These problems have been further discussed. A declarative programming approach, rule-based systems, and ontologies have been proposed as solutions that could improve the state of virtual document generation. The novelty of the technique has been discussed and its advantages have been presented, showing as a result a higher level of abstraction and better expressiveness in the document prescriptions. The importance of the ontology utilization in terms of productivity increase has also been stated. Finally, an example has been presented and the results have been discussed showing an improvement from the developers' perspective and a reduction of programming expertise in the process.

# Chapter 6 – Contributions, Conclusions, and Further Work

In this chapter, the contributions presented in this thesis are summarised, the conclusions that have been reached are stated and further work is proposed in order to extend this line of research.

# 6. Contributions, Conclusions, and Further Work

## 6.1 Contributions

The research presented in this thesis aims to promote the use, quality and efficiency of electronic documentation. The main aim is to help documentation developers automate the process of creating Intelligent Product Manuals. At first, the trends towards virtual documentation and towards automatic generation of documentation are identified. A link is formed between software documentation and manufactured products. Then, the contributions that are presented are as follows:

1. Object-Oriented Framework for Virtual Documentation

The Object-Oriented Framework provides an infrastructure that enhances the productivity and the efficiency of developing electronic documentation systems. It allows developers to reuse and extend basic functionality components of a documentation system and to have a virtual documentation system ready with minimal effort. A definition is provided that encompasses all the aspects that existing research has found, and a critique on the advantages and disadvantages is detailed.

2. A Global Definition on the Information Object

The Information Object is a useful concept for the development of documentation. It enables reuse, promotes robust structuring, and allows for better conceptualisation of the data. The drawbacks of the IO are that it is defined loosely, does not have an active nature, and does not exploit intelligent techniques. The existing definitions are studied and a definition is provided that unites the existing work on the concept.

3. Intelligent Information Object Definition

The Intelligent Information Object is a concept-specific autonomous agent that searches an appropriate structured information source and discovers the concepts to be represented. It extracts a set of data to initialise its contents, updates itself, can render its contents customised according to external parameters, and interacts with other Intelligent Information Objects in order to build a complete instance of a document. The IIO is delegated to handle information concerning a meaningful entity in the domain in hand. It acts proactively to keep this information up to date and independently mines the information to keep itself closely related to the data of the other information objects.

4. Intelligent Information Object Architecture

The Intelligent Information Object acts as a building block for the IIO documentation system architecture. This architecture creates a highly dynamic middle layer where the information takes form according to the conceptual model of the existing information objects and at the same time it is interlaced by the active nature of the Intelligent Information Objects. This middle layer acts as a buffer between the data repository and presentation layers. Within this buffering zone, the information is processed and enhanced.

5. Rule-Based Approach to Virtual Documents

A Rule-Based Approach to virtual document authoring uses a declarative paradigm and is knowledge-based. It offers a better level of abstraction, ease of coding and a decision-oriented strategy that is very important for user adaptation. With the employment of ontologies a significant code reuse can also be achieved, which in turn, increases the efficiency of the authoring process.

6. Ontology for Virtual Technical Documentation

The Ontology for Virtual Documentation systems describes the documentation domain from the point of view of adaptively generated reports. It provides a basis for creating virtual document prescriptions conditionally, that is according to the sets of rules that are used. It is an extensible asset and can be used as a starting point for the developers to build on top their specific requirements.

## 6.2 Conclusions

1. Techniques for automatic construction of documentation can be realised in the case of technical documentation as in the case of software documentation.
2. In the case of technical documentation, there is no strict formalism as in the case of programming languages, but the PDM structure and addition of meta-data can overcome the difficulties of this lack of formality.
3. Object-Oriented frameworks are a viable solution for technical documentation that can increase productivity, but may be limiting in the case of larger products.
4. The Intelligent Information Object can provide good results with large products exploiting the PDM structuring, but it needs agreements, in a finer level of detail, on the ways that the data will be organised on the product nodes.
5. The Intelligent Information Object Architecture and the middle layer it creates for the pre-processing of the data is very promising and can provide added value to the information through the dynamic mining of relationships between the contents of the IIOs and the entities they represent.

6. Virtual Document prescriptions can be expressed as a rule-based system with a strong advantage for adaptive generation of documentation.
7. The abstraction level and the reusability of the document prescriptions are enhanced with the rule-based technique for virtual documents.

## 6.3 Further Work

For Intelligent Information Objects, a possible research direction could be to organise strategies that will allow a collection of Intelligent Information Objects to generate larger reports co-operatively.

Another direction relating to Intelligent Information Objects could be the addition of an IIO layer structured not upon the product structure, but on the knowledge structure that is present within the documentation data. This will allow for manuals that do not follow the hierarchical product structure but a thematic structure.

For Virtual Documents, a matter for future research could be a solution to the problem of the discontinuity of design that their dynamic generation introduces. This would involve the development of a method for the evaluation of the generated document in order to provide feedback automatically to the author for enabling the iterative nature of document design.

Finally, methodologies for the organisation of the data within the PDM system, with minimal involvement from the product designers, are always a challenge. Agent systems could be a valuable solution for the processing of the data and the partial automation of the selection and transformation of media.

# APPENDIX

## APPENDIX A – Object-Oriented Framework Sample Code

```
package ipm;
// XML HANDLING
import org.w3c.dom.*;
import javax.xml.parsers.*;
// FILE VECTOR ETC
import java.util.*;
import java.lang.Character;
import java.io.File;
import java.io.*;
/**
* The Component class is designed to be a singleton in order to avoid syncronisation * problems in Updating. The instance is provided through the static public instance() * which when called the first time initialises the root Object. The default and normal   * constructors are private to avoid further instantiation.
* members.tripod.com/rwald/java/articles/singleton_in_Java.html
*/
public class Component {
        //###############################
        //## DATA
        //###############################
        private String ID= new String();
        private String Name= new String();
        private String Type= new String();
        private String Description= new String();
        private Document XmlDescriptionFiles;  // XML files
        private String GrfxFiles= new String();            // BLOBS ?
        private Component   SuperComponent;    // Objects
        private Component[] SubComponents;     // Objects
        private Component[] Relations;         // Objects
        private StringBuffer Rules= new StringBuffer();
        private static String webappDir;
        //###############################
        //## IIO BASICS
        //###############################
        /**
         *          Registry of ptrs to all the instantiated Component objects.
         */
        static private Hashtable        _Registry=new Hashtable();
        /**
         *          Counter of Components so far instansiated.
         */
        static private int              componentNo;
        /**
         *          Singleton implementation.
         */
        static private Component        _instance=null;
        //###############################
        //## REPORT
        //###############################
        /**
         *          Storage for the generated report.
         */
```

```
static private StringBuffer     _Report= new  StringBuffer();
/**
 *          Storage for the recursively generated menu.
 */
static private          StringBuffer    MenuOutput  = new  StringBuffer();
//################################
//## DATA-KNOWLEDGE-UM-XML
//################################
/**
 *          Local reference to the PDM Agent.
 */
static private PDM              _PDM;
/**
 *          User Model Object.
 */
static private userModel        UM;
/**
 *          Object Specific Logic Engine.
 */
private     LogicEngine      LE;
/**
 *          User Specific Knowledge Tags produced by the Objects Reasoning process.
 */
private static LinkedList       userKnowledgeTags= new LinkedList();
//#############################
//## NLP
//#############################
/**
 *          Holder for the Side Lexical Menus.
 */
private      LinkedList       sideMenus = new LinkedList();
/**
 *          Product Dictionary.
 */
private static ProductDictionary dictionary=
                new ProductDictionary(webappDir);
/**
 *          The Text Processor Object.
 */
private      TextProcessor    textProcessor=
               new TextProcessor(dictionary,webappDir);
//#############################  -
//## CONSTRUCTORS
//#############################
/**
 *          The first constructor gets a pointer to the PDM agent and creates the product IIO,
 *          that is the root of the product tree, the rest of the component instantiations are
 *          performed by the second type of constructor provided.
 */
private Component(PDM PDMptr)  {  // Constructor for the root ( No parent )

          try {
                    PrintStream myErr= new PrintStream(new FileOutputStream(new
                    File(webappDir+"logs/Components.log")));
                    System.setErr(myErr);
          }catch(java.io.IOException e){System.err.println("problem :"+e.getMessage());}


          System.err.println("DEBUG SINGLETON THIS default :"+this);
```

```
		_PDM=PDMptr;

		new Component(_PDM.getProductID(),null);

				// If this is the initialisation of the last component – then start relation
mapping
		if(componentNo==_PDM.getPopulation()) {

			System.err.println("");
			System.err.println("################################## ");
			System.err.println("## PRODUCT TREE COMPLETE         ");
			System.err.println("## ALL COMPONENTS INITIALISED    ");
			System.err.println("################################## ");
			System.err.println("Total From PDM  :"+_PDM.getPopulation());
			System.err.println("Total Instances :"+componentNo);
			System.err.println("");
			System.err.println("################################## ");
			System.err.println("## STARTING RELATION MAPPING     ");
			System.err.println("################################## ");
			System.err.println("");
			// THINGS TO BE DONE ONLY ONCE
			dictionary.eliminateEntrySubsets();
			Component test=this.getProductPtr();
			test.initRelationMapping();
		}
		System.err.println("DEBUG SINGLETON THIS END");
	}
	/**
	 *	The second type constructor gets the parent componet and the ID of the component
	 *	to be created and creates the new IIO.
	 */
	private Component(String _ID, Component Parent){
		if(_instance==null)
			_instance=this;
		SuperComponent=Parent;
		ID =_ID;
		System.err.println("DEBUG SINGLETON THIS normal :"+this);
		_Registry.put(ID,this); // add an ID – ptrToInstance to the Registry
		componentNo++;
		this.init();
	}
	/**
	 *	This method adds a new component after the product structure has been initialised.
	 */
	public void addComponent(String childID, String ParentID) {
		// find parent in Registry
		Component parent=(Component)_Registry.get(ParentID);
		Component newComponent=new Component(childID,parent);
		// update parent 4
		// parent.Update(parent.getID(),4);
	}
	/**
	 *	This method implements the singleton patern.
	 */
	static public synchronized Component instance(PDM PDMptr,String _webappDir) {
		if(_instance==null) {
			webappDir=_webappDir;
			new Component(PDMptr);
			// Constructor is called here but the instance is set by the normal constructor
		}
```

```
			return _instance;
		}

		/**
		 *		This method initialises the component ( extracts data, init logic,
		 *		and adds components name to the product dictionary ).
		 *	Does not mess with relations other than part-of.
		 */
		public void init() {
			System.err.println("Initialisation of Component for ---> "+ID);
			this.extractName();
			this.extractType();
			this.extractDescription();
			this.extractXmlDescriptions();
			this.extractGrfx();Coherence
			this.extractSubComponents();
			this.extractRules();
			LE = new LogicEngine(Rules);
			dictionary.addEntry(Name,ID);
		}
		/**
		 *		This method is called when the product structure is complete ( no references to non-initialised
		 *		objects ie. PDM no. of objects = no. of Components ) to map the relations and convert our
product
		 *	tree to a graph.
		 */
		public void initRelationMapping() {
//			dictionary.eliminateEntrySubsets();
			Description=textProcessor.processText(Description,this.ID);
			System.err.println("The Descr:"+Description);
			sideMenus=textProcessor.getSideMenu();
			System.err.println("The side:"+sideMenus.toString());
			System.err.println("Mapping Relations of Component for ---> "+ID);
			this.extractRelations();
			try{
				for(int i=0;i<SubComponents.length;i++)
					System.err.println(SubComponents[i].getID());

				for(int i=0;i<SubComponents.length;i++) {
					SubComponents[i].initRelationMapping();
				}
			}catch(Exception e){System.out.println("EDW  : "+e.getMessage()); }
		}
		/**
		 *		This method is called when the product structure is updated ( called by PDM agent )
		 *		and re-extracts the data. If the name is updated, the dictionary is also updated, if
		 *		the description is updated, it is also re-processed by the text processor, is the
		 *		sub-components are updated and some of them are new , they are created. The components
		 *		rules are also reloaded.
		 */
		public void Update(String _ID_,int type) {
			if(this.ID.equals(_ID_)) {
				if(type==1) {
					this.extractName();
					dictionary.updateEntry(Name,ID);
				}else
				if(type==2) {
					this.extractType();
				}else
				if(type==3) {
```

```
                                        this.extractDescription();
                        Description=textProcessor.processText(Description,this.ID);
                        }else
                        if(type==4) {
                                        this.extractSubComponents();
                        }else
                        if(type==5) {
                                        this.extractRelations();
                        }else
                        if(type==6) {
                                        this.extractXmlDescriptions();
                        }else
                        if(type==7) {
                                        this.extractGrfx();
                        }else
                        if(type==8) {
                                        this.extractRules();
                        }
                        LE.reloadRules(Rules);
                }
                else for(int i=0;i<SubComponents.length;i++)
                                        SubComponents[i].Update(_ID_,type);
        }
        /**
         *      This method simply returns the ID of the Component.
         */
        public String getID() {         return  ID; }
        /**
         *      This method simply returns the Name of the Component.
         */
        public String getName() { return Name; }
        /**
         *      This method simply returns the Type of the Component.
         */
        public String getType() { return Type; }
        /**
         *      This method simply returns the Description of the Component.
         */
        public String getDescription() { return  Description; }
        /**
         *      This method simply returns the XML associated with the Component.
         */
        public Document getXmlDescriptions() { return XmlDescriptionFiles; }

        /**
         *      This method simply returns the Grafic file of the Component.
         */
        public String getGrfxFiles() {  return  GrfxFiles; }

        /**
         *      This method simply returns the list of Super-Components of the Component.
         */
        public String getSuperComponents() {
                String Name=new String(" ");

                if(SuperComponent!=null)
                        Name=SuperComponent.getName();

                return Name;
        }
        /**
         *      This method simply returns the list of Sub-Components of the Component.
         */
        public String[] getSubComponents() {
                String[] Names=new String[1];
                Names[0]=" ";

                if(SubComponents!=null) {
                        Names=new String[SubComponents.length];

                        for(int i=0;i<SubComponents.length;i++) {
                                Names[i]=SubComponents[i].getName();
                        }
                }

                return Names;
        }
        /**
         *      This method simply returns the list of Related Components to the Component.
         */
        public String[] getRelations() {
                String[] Names=new String[1];
                Names[0]=" ";
                if(Relations!=null) {
                        Names=new String[Relations.length];

                        for(int i=0;i<Relations.length;i++) {
                                Names[i]=Relations[i].getName();
                        }
                }
                return Names;
        }
        /**
         *      This method extracts from the PDM agent the Name of the Component.
         */
        public void extractName() { Name=_PDM.getName(ID); }
        /**
         *      This method extracts from the PDM agent the Type of the Component.
         */
        public void extractType() { Type=_PDM.getType(ID); }
        /**
         *      This method extracts from the PDM agent the Description of the Component.
         */
        public void extractDescription() { Description=_PDM.getDescription(ID); }
        /**
         *      This method extracts from the PDM agent the XML of the Component.
         */
        public void extractXmlDescriptions() {  XmlDescriptionFiles=_PDM.getXmlDescriptions(ID); }
        /**
         *      This method extracts from the PDM agent the Grafics of the Component.
         */
        public void extractGrfx() {
        GrfxFiles=webappDir.substring(webappDir.indexOf("/webapps/")+8)+_PDM.getGrfxFiles(ID).substring(_
PDM.getGrfxFiles(ID).indexOf("PDM"));
        }
        /**
         *      This method extracts from the PDM agent the list of Sub-Components of the Component.
         */
        public void extractSubComponents() {
```

```
		Vector Sub=_PDM.getSubComponents(ID);
		if(Sub!=null) {
			SubComponents=new Component[Sub.size()];
			for(int i=0;i<Sub.size();i++) {
				// look if object exists in registry
				boolean exists=_Registry.containsKey(Sub.get(i));
				if(exists) {
					SubComponents[i]=(Component)_Registry.get(Sub.get(i));
				}
				// if not create it.
				else {
					SubComponents[i]=new Component((String)Sub.get(i),this);
				}
			}
		}
}
/**
 *	This method extracts from the PDM agent the list of Related Components to the Component.
 *	Uses the pointers from the Registry in order not to replicate the objects.
 */
public void extractRelations() {
		Vector Relat=_PDM.getRelations(ID);
		if(Relat!=null) {
			Relations=new Component[Relat.size()];
			for(int i=0;i<Relat.size();i++)
				Relations[i]=(Component)_Registry.get(Relat.get(i));
		}
}
/**
 *	This method extracts from the PDM agent the Rules of the Component.
 */
public void extractRules() { Rules=_PDM.getRules(ID); }
public StringBuffer getReport() { return _Report; }
/**
 *	We can't have recursive functions returning value , so we made Report void and
 *	we added the _Report static to our class . The recursion passes through the tree
 *	and when the target is found the report is stored in _Report ( static )
 *	so then we call getReport to collect.
 */
public void Report(String _ID_) {
		if(this.ID.equals(_ID_)) {
			_Report=this.Report();
		}
		else for(int i=0;i<SubComponents.length;i++)
				SubComponents[i].Report(_ID_);
}
// the ending step of the resursion ... the actual report.
public StringBuffer Report() {
		StringBuffer  HTMLOutput = new  StringBuffer();
		HTMLOutput.append("<HTML><HEAD><TITLE>");
		HTMLOutput.append("TITLE");
		HTMLOutput.append("</TITLE></HEAD>");
		HTMLOutput.append("<BODY BGCOLOR=#CCCCCC>");
		HTMLOutput.append("<B><P><BR>");
		HTMLOutput.append("<B><P><BR>");
		HTMLOutput.append(Type);
		HTMLOutput.append("ID: ");
		HTMLOutput.append(ID);
		HTMLOutput.append("<HR><P>");
		HTMLOutput.append("<B>");
```

```
                HTMLOutput.append("The ");
                HTMLOutput.append(Name);
                HTMLOutput.append(" is a");
                if(Type.equals("Assembly"))
                        HTMLOutput.append("n");
                HTMLOutput.append(" "+Type+".");
                HTMLOutput.append(" ");
                if(Type.equals("Product")||Type.equals("Assembly")) {
                        HTMLOutput.append("It is composed of");
                        try{
                                for(int i=0; i<SubComponents.length; i++) {
                                        if((i==SubComponents.length-1)&&(SubComponents.length!=1))
                                                HTMLOutput.append(" and");
                                        HTMLOutput.append(" the ");
                                        HTMLOutput.append("<a
href=\"http://mec1.engi.cf.ac.uk:8080/simpleGW/servlet/ipm.Main?PresentContent="+
                        ipm.URLUTF8Encoder.encode((String)SubComponents[i].getID())+" \"
target=\"Main\" > "+
                                (SubComponents[i].getName())+"</a>");
                                        if((i==SubComponents.length-1)&&(SubComponents.length!=1))
                                                HTMLOutput.append(".");
                                        else HTMLOutput.append(", ");
                                }
                        }catch(Exception e) { System.err.println("TI PAPARIA "+e.getMessage()); }
                }
                HTMLOutput.append(" ");
                if(Type.equals("Part")||Type.equals("Assembly")) {
                        HTMLOutput.append("It is part of the ");
                        HTMLOutput.append("<a
href=\"http://mec1.engi.cf.ac.uk:8080/simpleGW/servlet/ipm.Main?PresentContent="+
          ipm.URLUTF8Encoder.encode((String)SuperComponent.getID())+" \" target=\"Main\" > "+
          (SuperComponent.getName())+"</a>");
                        HTMLOutput.append(".");
                }
                HTMLOutput.append(" ");
                if(Relations.length!=0)
                        HTMLOutput.append("The "+Name+" also relates to");
                try{
                        for(int i=0; i<Relations.length; i++) {
                                if((i==Relations.length-1)&&(Relations.length!=1))
                                        HTMLOutput.append(" and");
                                HTMLOutput.append(" the ");
                                HTMLOutput.append("<a
href=\"http://mec1.engi.cf.ac.uk:8080/simpleGW/servlet/ipm.Main?PresentContent="+
                                ipm.URLUTF8Encoder.encode((String)Relations[i].getID())+" \"
target=\"Main\" > "+
                                        (Relations[i].getName())+"</a>");
                                if((i==Relations.length-1)&&(Relations.length!=1))
                                        HTMLOutput.append(".");
                                else HTMLOutput.append(", ");
                        }
                }catch(Exception e) {
                        System.err.println("TI PAPARIA "+e.getMessage());
                }
                HTMLOutput.append("<HR>");
                HTMLOutput.append("<P><BR><PRE>");
                HTMLOutput.append(Description);
                HTMLOutput.append("</PRE><B><P><BR>");
                HTMLOutput.append("<IMG ALIGN=TOP SRC=\"");
                HTMLOutput.append(this.getGrfxFiles());
                HTMLOutput.append("\">");
                HTMLOutput.append(" ");
                HTMLOutput.append(this.parseXMLDocs().toString());
                HTMLOutput.append("</BODY>");
                HTMLOutput.append("</HTML>");
                return HTMLOutput;
        }
        public void dumpRegistry() {
                System.err.println(" ");
                System.err.println("########################################");
                System.err.println("## STATIC COMPONENT REGISTRY DUMP     ");
                System.err.println("########################################");
                System.err.println(" ");
                System.err.println(_Registry.toString());
                System.err.println(" ");
        }
        public void numberOfComponents() {
                System.err.println(" ");
                System.err.println("######################################## ");
                System.err.println("## NUMBER OF COMPONENTS INSTANSIATED : ");
                System.err.println("## No: "+ componentNo              );
                System.err.println("######################################## ");
                System.err.println(" ");
        }
        private Component getProductPtr() {
                Component[] temp=new Component[componentNo];
                int k=0;
                for( Enumeration e=_Registry.elements();e.hasMoreElements();) {
                        temp[k]=(Component)e.nextElement();
                        k++;
                }
                System.err.println(" ");
                System.err.println("######################################## ");
                System.err.println("## LOOKING FOR PRODUCT IN REGISTRY     ");
                System.err.println("######################################## ");
                System.err.println(" ");
                try{
                        for(int i=0;i<temp.length;i++) {
                                if(temp[i].getType().equals("Product")) {
                                        System.err.println(" ");
                                        System.err.println("########### PRODUCT FOUND
###########    ");
                                        System.err.println(" ");

                                        return temp[i];
                                }
                        }
                } catch(Exception e) { System.out.println("Exc :"+e.getMessage()); e.printStackTrace(); }
                return null;
        }
        public StringBuffer getMenu() {
                MenuOutput=new StringBuffer();
                System.err.println(" ");
                System.err.println("######################################## ");
                System.err.println("## MAKING THE PRODUCT MENU           ");
                System.err.println("######################################## ");
                System.err.println(" ");
                MenuOutput.append("<HTML><HEAD><TITLE>");
                MenuOutput.append("TITLE");
                MenuOutput.append("</TITLE></HEAD>");
                MenuOutput.append("<BODY BGCOLOR=#CCCCCC>");
                MenuOutput.append("<CENTER><H2>Structural Menu</H2></CENTER>");
                MenuOutput.append("<TABLE BORDER=\"4\">");
                this.makeMenu();
                MenuOutput.append("</TABLE><BR><BR> ");
                MenuOutput.append("<p><a href=\"http://mec1.engi.cf.ac.uk:8080"+
                                                "/simpleGW/servlet/ipm.Main?"+
                                                "PresentContent=Exit \" target=\"TopFrame\"> Exit </a>");
                MenuOutput.append("<p><a href=\"http://mec1.engi.cf.ac.uk:8080"+
                                                "/simpleGW/servlet/ipm.Main?"+
                                                "PresentContent=UpdateUM \" target=\"Main\"> Update User
Model </a>");
                MenuOutput.append("</BODY>");
                MenuOutput.append("</HTML>");
                return MenuOutput;
        }
        private void makeMenu() {
                try{
                        System.err.println("APPENDING MENU ITEM ... ID :"+ID+" NAME :"+Name);
                        MenuOutput.append("<TR><TD");
                        if(this.Type.equals("Assembly"))
                                MenuOutput.append(" BGCOLOR=\"#C0C0C0\" ");
                        if(this.Type.equals("Part"))
                                MenuOutput.append(" BGCOLOR=\"#FFFFFF\" ");
                        MenuOutput.append("><p><UL><LI>");
                        MenuOutput.append("<a href=\"http://mec1.engi.cf.ac.uk:8080"+
                                                "/simpleGW/servlet/ipm.Main?"+
                                                "PresentContent="+
                                                URLUTF8Encoder.encode(this.ID)+
                                                "\" target=\"Main\">"+
                                                this.Name+
                                                "</a></UL></LI></TR></TD>");
                        for(int i=0;i<SubComponents.length;i++)
                                SubComponents[i].makeMenu();
                }catch( Exception e ) { System.err.println("Here"); }
        }
        public StringBuffer getSideMenu(String ID,String menuNo) {
                MenuOutput=new StringBuffer();
                System.err.println(" ");
                System.err.println("########################################## ");
                System.err.println("## MAKING THE SIDE MENU            ");
                System.err.println("########################################## ");
                System.err.println(" ");
                MenuOutput.append("<HTML><HEAD><TITLE>");
                MenuOutput.append("TITLE");
                MenuOutput.append("</TITLE></HEAD>");
                MenuOutput.append("<BODY BGCOLOR=#CCCCCC>");
                MenuOutput.append("<CENTER><H2>Relations Menu</H2></CENTER>");
                MenuOutput.append("<TABLE BORDER=\"4\">");
                System.err.println("ID:"+ID+"menu no:"+menuNo+"Registry:"+_Registry.toString());
                if(ID!=null && menuNo!=null) {
                        ((Component)_Registry.get(ID)).makeSideMenu(menuNo);
                }
                MenuOutput.append("</TABLE><BR><BR> ");
                MenuOutput.append("</BODY>");
                MenuOutput.append("</HTML>");
                return MenuOutput;
        }
        private void makeSideMenu(String menuNo) {
                System.err.println("");
        System.err.println("###############################################");
        System.err.println("THE SIDE MENU FOR:"+Name);
        System.err.println(sideMenus.toString());
        System.err.println("###############################################");
        System.err.println("");
        for(int i=0; i<sideMenus.size() ; i=i+3) {
                if(((String)sideMenus.get(i)).equals(menuNo)) {
                        System.err.println("APPENDING SIDE MENU ITEM ...");
                        MenuOutput.append("<TR><TD");
                        MenuOutput.append(" BGCOLOR=\"#C0C0C0\" ");
                        MenuOutput.append("><p>");
                        MenuOutput.append("LEXICAL</TR></TD>");
                        for(int y=0;y<((LinkedList)sideMenus.get(i+2)).size();y++) {
                                MenuOutput.append("<TR><TD");
                                MenuOutput.append(" BGCOLOR=\"#FFFFFF\" ");
                                MenuOutput.append("><p>");
                                MenuOutput.append(((LinkedList)sideMenus.get(i+2)).get(y));
                                MenuOutput.append("</TR></TD>");
                        }
                        MenuOutput.append("<TR><TD");
                        MenuOutput.append(" BGCOLOR=\"#C0C0C0\" ");
                        MenuOutput.append("><p>");
                        MenuOutput.append("DICTIONARY</TR></TD>");
                        for(int x=0;x<((LinkedList)sideMenus.get(i+1)).size();x++) {
                                MenuOutput.append("<TR><TD");
                                MenuOutput.append(" BGCOLOR=\"#FFFFFF\" ");
                                MenuOutput.append("><p>");
                                MenuOutput.append(((LinkedList)sideMenus.get(i+1)).get(x));
                                MenuOutput.append("</TR></TD>");
                        }
                }
        }
}
public void setUserModel(userModel um) {
        UM=um;
        this.processUserModel();
}
public void setWebAppDir(String dir) {
        webappDir=dir;
}
private void processUserModel() {
        userKnowledgeTags=new LinkedList();
        String[] attribs=new String[UM.getAttributes().length];
        attribs=UM.getAttributes();
        for(int i=0;i<attribs.length;i++) {
                userKnowledgeTags.addAll(LE.match(UM.getAttributeValue(attribs[i])));
        }
}
private StringBuffer parseXMLDocs() {
        StringBuffer buffer=new StringBuffer();
        for(int i=0;i<userKnowledgeTags.size();i++) {
                parseXML(XmlDescriptionFiles,(String)userKnowledgeTags.get(i),buffer);
        }
        return buffer;
}
private static void parseXML( org.w3c.dom.Node node , String tag , StringBuffer buffer ) {
        if(node.getNodeName().equals(tag.trim())) {
                org.w3c.dom.Node _child = (org.w3c.dom.Node)node.getFirstChild();
                buffer.append("<P><u>"+tag.trim()+" : </u>"+_child.getNodeValue()+" ");
        }
```

```
                org.w3c.dom.Node _child = (org.w3c.dom.Node)node.getFirstChild();
                for (; _child != null; _child = (org.w3c.dom.Node)_child.getNextSibling()) {
                        parseXML( _child, tag , buffer );
                }
        }
}
```

## *APPENDIX B - FIPA-OS Agent Sample Code*

```
package fipaos.IIO_Agent_System;
//-----------------------------------------------------------
//import the superclass
import fipaos.skill.jess.JessAgent;
// Import the fipa classes
import fipaos.ont.fipa.*;
// Import the agent management classes (as we are registering with the platform)
import fipaos.ont.fipa.fipaman.*;
// We will also need to import agent classes
import fipaos.agent.*;
// We'll need tasks
import fipaos.agent.task.*;
// We'll need conversation
import fipaos.agent.conversation.*;
// Import registration exception classes
import fipaos.platform.ams.AMSRegistrationException;
import fipaos.platform.df.DFRegistrationException;
// ACL message
import fipaos.parser.acl.*;
// JESS classes needed
import jess.JessException;
import jess.Value;
// Finally import the diagnostics class for output
import fipaos.util.DIAGNOSTICS;
import java.util.*;
// XML HANDLING
import org.w3c.dom.*;
import javax.xml.parsers.*;
import java.io.*;
//-----------------------------------------------------------
public class IIOAgent extends JessAgent
{
        //////////////////////////////////////////////////////////////////////
        // Agent Data
        String _ID=new String();
        String _NAME=new String(getAID().getName().substring(0,getAID().getName().indexOf('@')));
        String _TYPE= new String();
        String _DESCRIPTION= new String();
        String _GRAPHICS= new String();
        Vector _PARENTS= new Vector();
        Vector _SUBCOMPONENTS= new Vector();
        Vector _RELATIONS= new Vector();
        Document _XML;
        StringBuffer _RULES= new StringBuffer();
        String _SIDEMENU= new String();
        StringBuffer    _Report= new  StringBuffer();
        StringBuffer    MenuOutput  = new  StringBuffer();
```

```
		LinkedList	userKnowledgeTags= new LinkedList();
		// End Agent Data
		/////////////////////////////////////////////////////////////////////////

		/////////////////////////////////////////////////////////////////////////
		// Agent Management  -- Construction, Shutdown
		IdleTask MyTask = new IdleTask();
		/**
		 * Constructor: instansiates the superclass, and sets a listener task.
		 *
		 * @param   platform_location  location of the platform profile
		 * @param   name  name of the agent
		 * @param   owner  owner of the agent
		 */
		public IIOAgent(String platform_location, String name, String owner)
		{
			super(platform_location, name, owner);
			//start listening to the message requests
			super.setListenerTask(MyTask);
			// use the push model of agent comms
			startPushing();
			// Now we can register with the local AMS (i.e. the platform) as the fist action our agent
			// makes
			try
			{
				// Attempt to register with AMS
				registerWithAMS();
				DIAGNOSTICS.println("Registered with AMS", this, DIAGNOSTICS.LEVEL_MAX );
			}
			catch (AMSRegistrationException amsre)
			{
				// An exception has occured - this indicates that the AMS registration failed for some
reason
				// Display the exception
				DIAGNOSTICS.println(amsre, this, DIAGNOSTICS.LEVEL_MAX );
				// We can easily find the exception reason from the exception....
				String reason = amsre.getExceptionReason();
				// If the exception reason is not "already-registered", we can probably ignore the
failure
				// (since this Agent was probably not de-registered the last time it ran!)
				if ( reason == null ||
!reason.equals(FIPAMANCONSTANTS.AGENT_ALREADY_REGISTERED))
				{
					// Shutdown nicely :)
					shutdown();
					return;
				}
			}
			DIAGNOSTICS.println("Ready!", this, DIAGNOSTICS.LEVEL_4);
			initialiseIntInfoObj();
			DIAGNOSTICS.println("Initialising IIO ...", this, DIAGNOSTICS.LEVEL_4);
			try {
				PrintStream myErr= new PrintStream(new FileOutputStream(new
				File(_NAME+".log")));
				System.setErr(myErr);
			}catch(java.io.FileNotFoundException e){System.err.println("problem :"+e.getMessage());}
			System.err.println("");
			System.err.println("##########################################");
			System.err.println("## AGENT CONTENTS                    ");
			System.err.println("##########################################");
		System.err.println("name        :"+_NAME);
		System.err.println("type        :"+_TYPE);
		System.err.println("id          :"+_ID);
		System.err.println("description :"+_DESCRIPTION);
		System.err.println("graphics    :"+_GRAPHICS);
		System.err.println("parents     :"+_PARENTS);
		System.err.println("subcomponent:"+_SUBCOMPONENTS);
		System.err.println("relations   :"+_RELATIONS);
		System.err.println("xml         :"+_XML);
		System.err.println("rules       :"+_RULES);
	}
	/**
	 * Shuts down the agent by first deregistering with the DF and AMS
	 * (if registered) and then invoking shutdown() on the FIPAOSAgent shell
	 */
	public synchronized void shutdown()
	{
		// Check if we've registered with the DF
		if (registeredWithDF() == true)
		{
			try
			{
				// Attempt to deregister with DF
				deregisterWithDF();
				DIAGNOSTICS.println("Deregistered with DF", this,
DIAGNOSTICS.LEVEL_MAX );
			}
			catch (DFRegistrationException dfre)
			{
				// A problem deregistering occured..... we can obtain the reason though!
				String reason = dfre.getExceptionReason();

				DIAGNOSTICS.println(dfre + ":" + reason, this,
DIAGNOSTICS.LEVEL_MAX );
			} // end catch
		}
		// Check if we're registered with the AMS
		if (registeredWithAMS() == true)
		{
			try
			{
				// Deregister with AMS
				deregisterWithAMS();
				DIAGNOSTICS.println("Deregistered with AMS", this,
DIAGNOSTICS.LEVEL_MAX );
			}
			catch (AMSRegistrationException amsre)
			{
				// A problem deregistering occured..... we can obtain the reason though!
				String reason = amsre.getExceptionReason();

				DIAGNOSTICS.println(amsre + ":" + reason, this,
DIAGNOSTICS.LEVEL_MAX );
			}
		}
		// Now call shutdown in the agent shell to release the core components
		super.shutdown();
	}
	// End Agent Management
	////////////////////////////////////////////////////////////////////////////////
```

```
        ////////////////////////////////////////////////////////////////////////////////
        // Agent Utility methods   - init, inference
        /**
         * Agent specific Initialisation as opposed to platform init that is done within the
         * constructor.
         *
         * Calls all the Requests from the Pdm Agent for the Information Object Data.
         */
        private void initialiseIntInfoObj( ) {

                sendRequest("get","id",_NAME);
                sendRequest("get","type",_NAME);
                sendRequest("get","description",_NAME);
                sendRequest("get","xml",_NAME);
                sendRequest("get","graphics",_NAME);
                sendRequest("get","parent",_NAME);
                sendRequest("get","subcomponents",_NAME);
                sendRequest("get","relations",_NAME);
                sendRequest("get","rules",_NAME);
                DIAGNOSTICS.println("Initialisation of IIO completed !", this, DIAGNOSTICS.LEVEL_4);
                sendUMRequest("get","model","scecp2");
                DIAGNOSTICS.println("IIO getting user model!", this, DIAGNOSTICS.LEVEL_4);
        }
        /**
         * Runs the JESS engine.
         *
         * @param   factorial  the number we want to calculate the factorial for.
         * @return     result, or 0 if failed
         */
        public int runEngineCycle(int factorial)
        {
                int int_result=0;
                DIAGNOSTICS.println("Task is to do factorial of " + factorial, this, DIAGNOSTICS.LEVEL_2);
                try
                {
                        // load the rulebase
                        engineExecuteCommand("batch factorial.clp");

                        // reset the Jess - ie. apply the rules just given
                        reset();

                        //run the do-factorial function and covert result into an int
                        Value result = engineExecuteCommand("do-factorial " + factorial);
                        int_result = result.intValue(getGlobalContext());
                }
                catch (JessException je)
                {
                        DIAGNOSTICS.println("JESS couldn't run the command: " + je, this,
DIAGNOSTICS.LEVEL_4);
                }
                catch (Throwable t)
                {
                        DIAGNOSTICS.println("Problem (not JESS related): " + t, this,
DIAGNOSTICS.LEVEL_4);
                }
                return int_result;
        }
        // End Agent Utility Methods
        ////////////////////////////////////////////////////////////////////////////////
        ////////////////////////////////////////////////////////////////////////////////////////////
        // Outgoing Communication Messsages
        /*
         * Forms Requests towards pdm-agents in the form of [command operand name].
         * Were command is like "get", operand is like "graphics", and name is a valid
         * agent name.
         * example: "get graphics Bolt" (get the graphics file of the Bolt agent...)
         */
        public void sendRequest( String command, String operand, String agentName )
        {
                ACL _acl= new ACL();
                _acl.setPerformative( "request" );
                try {
                        _acl.setSenderAID( getAID() );
                        _acl.setReceiverAID( new AgentID( "(agent-identifier :name pdm-agent@localap )" )
);
                }
                catch ( Exception ex ) {
                        DIAGNOSTICS.println("Agent ID was not valid", this, DIAGNOSTICS.LEVEL_MAX );
                }
                _acl.setProtocol( "fipa-request" );
                _acl.setContentObject("(" + command + " " + operand + " " + agentName + ")");
                _acl.setPerformative( FIPACONSTANTS.REQUEST );
                try
                {
                        MyTask.send( _acl );
                }
                catch ( Throwable mse )
                {
                        DIAGNOSTICS.println("Problem with the message " + _acl + "\n" + mse, this,
DIAGNOSTICS.LEVEL_MAX);
                }
                DIAGNOSTICS.println("IIO Sending PDM request!"+_acl, this, DIAGNOSTICS.LEVEL_4);
        }
        /*
         * Forms Requests towards pdm-agents in the form of [command operand name].
         * Were command is like "get", operand is like "graphics", and name is a valid
         * agent name.
         * example: "get graphics Bolt" (get the graphics file of the Bolt agent...)
         */
        public void sendDictionaryRequest( String command, String operand, String agentName )
        {
                ACL _acl= new ACL();
                _acl.setPerformative( "request" );
                try {
                        _acl.setSenderAID( getAID() );
                        _acl.setReceiverAID( new AgentID( "(agent-identifier :name product-dictionary-
agent@localap )" ) );
                }
                catch ( Exception ex ) {
                        DIAGNOSTICS.println("Agent ID was not valid", this, DIAGNOSTICS.LEVEL_MAX );
                }
                _acl.setProtocol( "fipa-request" );
                _acl.setContentObject("(" + command + " " + operand + " " + agentName + ")");
                _acl.setPerformative( FIPACONSTANTS.REQUEST );
                try
                {
                        MyTask.send( _acl );
                }
                catch ( Throwable mse )
                {
                        DIAGNOSTICS.println("Problem with the message " + _acl + "\n" + mse, this,
DIAGNOSTICS.LEVEL_MAX);
                }

                DIAGNOSTICS.println("IIO sending dictionary request!"+_acl, this, DIAGNOSTICS.LEVEL_4);
        }
        /*
         * Forms Requests towards TextProcessing-agents in the form of [command id text].
         * Were command is like "process", id is like "203", and text is a portion of text.
         *
         * example: "process 1000 #&# This is the description of the part nomber 1000 blah blah blah"
         * (process the text following the #&# sign)
         */
        public void sendTPRequest( String command, String id, String text )
        {
                ACL _acl= new ACL();
                _acl.setPerformative( "request" );
                try {
                        _acl.setSenderAID( getAID() );
                        _acl.setReceiverAID( new AgentID( "(agent-identifier :name text-processing-
agent@localap )" ) );
                }
                catch ( Exception ex ) {
                        DIAGNOSTICS.println("Agent ID was not valid", this, DIAGNOSTICS.LEVEL_MAX );
                }
                _acl.setProtocol( "fipa-request" );
                _acl.setContentObject("(" + command + " " + id + " #&#" + text + ")");
                _acl.setPerformative( FIPACONSTANTS.REQUEST );
                try
                {
                        MyTask.send( _acl );
                }
                catch ( Throwable mse )
                {
                        DIAGNOSTICS.println("Problem with the message " + _acl + "\n" + mse, this,
DIAGNOSTICS.LEVEL_MAX);
                }

                DIAGNOSTICS.println("IIO sending text processing request!"+_acl, this,
DIAGNOSTICS.LEVEL_4);
        }
        /*
         * Forms Requests towards TextProcessing-agents in the form of [command id text].
         * Were command is like "process", id is like "203", and text is a portion of text.
         *
         * example: "process 1000 #&# This is the description of the part nomber 1000 blah blah blah"
         * (process the text following the #&# sign)
         */
        public void sendUMRequest( String command, String oper, String username )
        {
                ACL _acl= new ACL();
                _acl.setPerformative( "request" );
                try {
                        _acl.setSenderAID( getAID() );
                        _acl.setReceiverAID( new AgentID( "(agent-identifier :name user-model-
agent@localap )" ) );
                }
                catch ( Exception ex ) {
```

```
				DIAGNOSTICS.println("Agent ID was not valid", this, DIAGNOSTICS.LEVEL_MAX );
			}
			_acl.setProtocol( "fipa-request" );
			_acl.setContentObject("(" + command + " " + oper + " " + username + ")");
			_acl.setPerformative( FIPACONSTANTS.REQUEST );
			try
			{
				MyTask.send( _acl );
			}
			catch ( Throwable mse )
			{
				DIAGNOSTICS.println("Problem with the message " + _acl + "\n" + mse, this,
DIAGNOSTICS.LEVEL_MAX);
			}
			DIAGNOSTICS.println("IIO sending text processing request!"+_acl, this,
DIAGNOSTICS.LEVEL_4);
		}
		// End Outgoing Communication Messages
		/////////////////////////////////////////////////////////////////////////////////////////

		/////////////////////////////////////////////////////////////////////////////////////////
		// Incomming Communications
		//-----------------------------------------------------------------------
		// INNER TASKS
		//-----------------------------------------------------------------------
		public class IdleTask extends Task
		{
			public void send(ACL _acl)
			{
				forward(_acl);
			}
			public void handleAgree(Conversation conv)
			{
				DIAGNOSTICS.println("IIO Received a
Agree!"+conv.getACL(conv.getLatestMessageIndex()), this, DIAGNOSTICS.LEVEL_4);
			}

			public void handleNotUnderstood(Conversation conv)
			{
				DIAGNOSTICS.println("IIO Received a NOT
UNDERSTOOD!"+conv.getACL(conv.getLatestMessageIndex()), this, DIAGNOSTICS.LEVEL_4);
			}

			public void handleFailure(Conversation conv)
			{
							DIAGNOSTICS.println("IIO Received a FAIL
*?>,.<xCX!"+conv.getACL(conv.getLatestMessageIndex()), this, DIAGNOSTICS.LEVEL_4);
			}
			/**
			 * This method is called (by Task) when this agent receives a request message.
			 *
			 * @param conv the conversation just starting
			 */
			public void handleRequest(Conversation conv)
			{
				DIAGNOSTICS.println("IIO Received a request!", this, DIAGNOSTICS.LEVEL_4);
				// get the last message (the request message).
				ACL msg = conv.getACL(conv.getLatestMessageIndex());
				//get the content
				String content = (String) msg.getContentObject();
```

```
                    //take the brackets from the message and trim
                    content = content.replace('(', ' ');
                    content = content.replace(')', ' ');
                    content = content.trim();
                    try
                    {
                            String command= new String();
                            String name= new String();
                            String nodeName= new String();
                            command=content.substring(0,content.indexOf(' ')).trim();
                            name=content.substring(content.indexOf(' ')).trim();
                            DIAGNOSTICS.println("Request is : "+"*"+command+"*"+name+"*", this,
DIAGNOSTICS.LEVEL_4);
                            // if it works, send agree
                            if(command.equals("get"))
                            {
                                    sendAgree(conv);
                                    // run the engine and examine the result
                                    if ( name.equals("id") ) // OK
                                    {
                                            //calculation successful
                                            sendInform(conv, _ID, name);
                                    }
                                    else if ( name.equals("type") ) // OK
                                    {
                                            sendInform(conv, _TYPE, name);
                                    }
                                    else if ( name.equals("description") ) // OK
                                    {
                                            sendInform(conv, _DESCRIPTION, name);
                                    }
                                    else if ( name.equals("xml") )
                                    {
                                            sendInform(conv, _XML, name);
                                    }
                                    else if ( name.equals("graphics") ) // OK
                                    {
                                            sendInform(conv, _GRAPHICS, name);
                                    }
                                    else if ( name.equals("parents") )
                                    {
                                            sendInform(conv, _PARENTS, name);
                                    }
                                    else if ( name.equals("subcomponents") )
                                    {
                                            sendInform(conv, _SUBCOMPONENTS, name);
                                    }
                                    else if ( name.equals("relations") )
                                    {
                                            sendInform(conv, _RELATIONS, name);
                                    }
                                    else if ( name.equals("rules") )
                                    {
                                            sendInform(conv, _RULES, name);
                                    }
                                    else
                                    {
                                            //calculation failed
                                            sendFailure(conv, "error ... No subject to get !!");
                                    }
```

```
                                        }
                                        else
                                        {
                                                sendFailure(conv, "error ... No get command found !!");
                                        }

                                }
                                catch (Exception nfe)
                                {
                                        //int conversion didn't work
                                        DIAGNOSTICS.println("Request didn't involve an acceptable ID: '" +
content + "'", this, DIAGNOSTICS.LEVEL_4);
                                        sendNotUnderstood(conv, nfe.toString());
                                }
                } // handleRequest()
                public void handleInform(Conversation conv)
                {
                                DIAGNOSTICS.println("IIO Queried!", this, DIAGNOSTICS.LEVEL_4);
                                // get the last message (the request message).
                                ACL msg = conv.getACL(conv.getLatestMessageIndex());
                                DIAGNOSTICS.println(msg, this, DIAGNOSTICS.LEVEL_4);
                                String replyTo = new String(msg.getInReplyTo());
                                if( replyTo.equals("store")) {

                                } else
                                if ( replyTo.equals("model")) {
                                        DIAGNOSTICS.println("Getting Knowledge Tags"+((String)
msg.getContentObject()).replace('(', ' ').replace(')', ' ').trim(), this, DIAGNOSTICS.LEVEL_4);
                                        userKnowledgeTags=((LinkedList) msg.getContentObject());
                                } else
                                if ( replyTo.equals("id")) {
                                        DIAGNOSTICS.println("Getting _ID"+((String)
msg.getContentObject()).replace('(', ' ').replace(')', ' ').trim(), this, DIAGNOSTICS.LEVEL_4);
                                        _ID=((String) msg.getContentObject()).replace('(', ' ').replace(')', ' ').trim();
                                        sendDictionaryRequest("store",_NAME,_ID);
                                } else
                                if ( replyTo.equals("type")) {
                                        DIAGNOSTICS.println("Getting _TYPE"+((String)
msg.getContentObject()).replace('(', ' ').replace(')', ' ').trim(), this, DIAGNOSTICS.LEVEL_4);
                                        _TYPE=((String) msg.getContentObject()).replace('(', ' ').replace(')', '
').trim();
                                } else
                                if ( replyTo.equals("description")) {
                                        DIAGNOSTICS.println("Getting _DESCRI"+((String)
msg.getContentObject()).replace('(', ' ').replace(')', ' ').trim(), this, DIAGNOSTICS.LEVEL_4);
                                        _DESCRIPTION=((String) msg.getContentObject()).replace('(', '
').replace(')', ' ').trim();
                                        sendTPRequest( "process", _ID, _DESCRIPTION );
                                        sendTPRequest( "get-side-menu", _ID, _DESCRIPTION );
                                } else
                                if ( replyTo.equals("xml") ) {
                                        DIAGNOSTICS.println("Getting
_XML"+(Document)msg.getContentObject(), this, DIAGNOSTICS.LEVEL_4);
                                        _XML=(Document)msg.getContentObject();
                                } else
                                if ( replyTo.equals("graphics")) {
                                        DIAGNOSTICS.println("Getting _GRAFIX"+((String)
msg.getContentObject()).replace('(', ' ').replace(')', ' ').trim(), this, DIAGNOSTICS.LEVEL_4);
                                        _GRAPHICS=((String) msg.getContentObject()).replace('(', ' ').replace(')', '
').trim();
```

```
                        } else
                        if ( replyTo.equals("parent")) {
                                DIAGNOSTICS.println("Getting
_PARENTS"+(Vector)msg.getContentObject(), this, DIAGNOSTICS.LEVEL_4);
                                _PARENTS=(Vector)msg.getContentObject();
                        } else
                        if ( replyTo.equals("sub-components")) {
                                DIAGNOSTICS.println("Getting _SUBS"+(Vector)msg.getContentObject(),
this, DIAGNOSTICS.LEVEL_4);
                                _SUBCOMPONENTS=(Vector)msg.getContentObject();
                        } else
                        if ( replyTo.equals("relations")) {
                                DIAGNOSTICS.println("Getting
_RELAT"+(Vector)msg.getContentObject(), this, DIAGNOSTICS.LEVEL_4);
                                _RELATIONS=(Vector)msg.getContentObject();
                        } else
                        if ( replyTo.equals("rules")) {
                                DIAGNOSTICS.println("Getting
_RUL"+(StringBuffer)msg.getContentObject(), this, DIAGNOSTICS.LEVEL_4);
                                _RULES=(StringBuffer)msg.getContentObject();
                        } else
                        if ( replyTo.equals("process")) {
                                DIAGNOSTICS.println("Getting _DESCRIPTION"+((String)
msg.getContentObject()).replace('(', ' ').replace(')', ' ').trim(), this, DIAGNOSTICS.LEVEL_4);
                                _DESCRIPTION=((String) msg.getContentObject()).replace('(', '
').replace(')', ' ').trim();
                        } else
                        if ( replyTo.equals("get-side-menu")) {
                                DIAGNOSTICS.println("Getting side menu"+((String)
msg.getContentObject()).replace('(', ' ').replace(')', ' ').trim(), this, DIAGNOSTICS.LEVEL_4);
                                _SIDEMENU=((String) msg.getContentObject()).replace('(', ' ').replace(')', '
').trim();
                        } else {
                                DIAGNOSTICS.println("Inform in reply to unknown signal: "+
msg.getContentObject().toString(), this, DIAGNOSTICS.LEVEL_3);

                        }
                } // handleInform()
                /**
                 * This method would be called when this agent would like to send a not-understood
                 * message.
                 *
                 * @param conv the conversation which this failure message would belong to
                 * @param content the content of this failure message
                 */
                private void sendNotUnderstood(Conversation conv, String content)
                {
                        ACL nu_msg = conv.getFilledInACL(); //conv.getMessage(
conv.getLatestMessageIndex() );
                        nu_msg.setReceiverAID(conv.getSenderAID(0));
                        nu_msg.setSenderAID( _owner.getAID() );
                        nu_msg.setContentObject("("+ content +")");
                        nu_msg.setPerformative( FIPACONSTANTS.NOT_UNDERSTOOD );
                        try
                        {
                                forward(nu_msg);
                        }
                        catch ( Throwable mse )
                        {
                                DIAGNOSTICS.println("Problem with the message " + nu_msg + "\n" +
mse, this, DIAGNOSTICS.LEVEL_MAX);
                        }
                        DIAGNOSTICS.println("Sent not-understood: "+ nu_msg, this,
DIAGNOSTICS.LEVEL_3);
                }
                /**
                 *
                 *
                 *
                 * @param conv the conversation which this failure message would belong to
                 * @param content the content of this failure message
                 */
                private void sendFailure(Conversation conv, String content)
                {
                        ACL failure = conv.getFilledInACL(); //conv.getMessage(
conv.getLatestMessageIndex() );
                        failure.setReceiverAID(conv.getSenderAID(0));
                        failure.setSenderAID( _owner.getAID() );
                        failure.setContentObject("("+ content +")");
                        failure.setPerformative( FIPACONSTANTS.FAILURE );
                        try
                        {
                                forward(failure);
                        }
                        catch ( Throwable mse )
                        {
                                DIAGNOSTICS.println("Problem with the message " + failure + "\n" + mse,
this, DIAGNOSTICS.LEVEL_MAX);
                        }
                        DIAGNOSTICS.println("Sent failure: "+ failure, this, DIAGNOSTICS.LEVEL_3);
                }
                /**
                 * Send an agree message.
                 *
                 * @param conv The conversation that this message belongs to
                 **/
                private void sendAgree( Conversation conv )
                {
                        ACL reply = conv.getFilledInACL(); //conv.getMessage(
conv.getLatestMessageIndex() );
                        reply.setReceiverAID( conv.getSenderAID( 0 ) );
                        reply.setSenderAID( _owner.getAID() );
                        reply.setInReplyTo( conv.getACL( 0 ).getReplyWith() );
                        reply.setContentObject( conv.getACL( 0 ).toString() );
                        reply.setPerformative( FIPACONSTANTS.AGREE );
                        try
                        {
                                forward( reply );
                        }
                        catch ( Throwable mse )
                        {
                                DIAGNOSTICS.println("Problem with the message " + reply + "\n" + mse,
this, DIAGNOSTICS.LEVEL_MAX);
                        }
                        DIAGNOSTICS.println("Sent agree: " + reply , this, DIAGNOSTICS.LEVEL_3);
                }
                private void sendInform( Conversation conv, Object content, String inReplyTo )
                {
                        ACL acl = conv.getFilledInACL();
                        DIAGNOSTICS.println( acl, DIAGNOSTICS.LEVEL_MAX );
```

```
                    acl.setReceiverAID( conv.getSenderAID( 0 ) );
                    acl.setSenderAID( _owner.getAID() );
                    acl.setInReplyTo( inReplyTo.trim() );
                    acl.setContentObject(content);
                    acl.setPerformative( FIPACONSTANTS.INFORM );
                    try
                    {
                            forward( acl );
                    }
                    catch ( Throwable mse )
                    {
                            DIAGNOSTICS.println("Problem with the message " + acl + "\n" + mse,
this, DIAGNOSTICS.LEVEL_MAX);
                    }

                    DIAGNOSTICS.println("Sent inform: " + acl , this, DIAGNOSTICS.LEVEL_3);
            }
        }// IdleTask
        // End Incomming Communications
        ////////////////////////////////////////////////////////////////////////////////////////////////////
}
```

## APPENDIX C - Rule-Based System (CLIPS) Code Fragments

### APPENDIX C.1 - Template Definitions

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Template Definitions – Data Structures
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;  User Model  ;;;;;;;;;;;;;;;;;;;;;;
; The user model of the current user
(deftemplate USER "The User Model"
        (slot     id (type INTEGER))
        (slot     first-name (type STRING))
        (slot     last-name (type STRING))
        (slot     experience (type INTEGER))
        (slot     education (type STRING))
        (slot     job (type STRING))
        (slot     task (type STRING))
        (slot     task-frequency (type INTEGER))
        (slot     request (type STRING))
        (slot     modality (type STRING))
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;  User Types  ;;;;;;;;;;;;;;;;;;;;;;
; List of all user types allowed in the system
(deftemplate USER-TYPES "Main User Profiles"
        (multislot user-types)
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;  User Type  ;;;;;;;;;;;;;;;;;;;;;;
; The user type of the current user
(deftemplate USER-TYPE "Main User Profiles"
        (slot     user-type (type STRING))
)
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;  Knowledge Types   ;;;;;;;;;;;
; List of all the knowledge types allowed in the system
(deftemplate KNOWLEDGE-TYPES "Knowledge Types Found Across the System"
        (multislot knowledge-types)
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;  User Knowledge Types   ;;;;
; List of knowledge types concerning the current user
(deftemplate USER-KNOWLEDGE-PROFILE "The Knowledge Types that correspond to the User Model"
        (multislot K-types)
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;  Query       ;;;;;;;;;;;;;;;;;;;;;;;;;;;
(deftemplate QUERY "A string that best identifies the Users target"
        (slot       query (type STRING))
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;  Discourse Pattern       ;;;;;;;;;;;;;;;;
(deftemplate DISCOURSE "A Discourse Pattern Described"
        (multislot discourse-pattern)
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;  Discourse Patterns      ;;;;;;;;;;;;;;;;
(deftemplate DISCOURSE-PATTERNS "The pool of system wide Discourse Patterns"
        (slot                   pattern-name (TYPE STRING))
        (multislot discourse-pattern)
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;  Document Structures     ;;;;;;;;;;;;;;;;
(deftemplate DOCUMENT-TYPE "A Structural Document Pattern Described"
        (slot                   document-name (type STRING))
)
(deftemplate DOCUMENT-TYPES "List of Document Patterns"
        (multislot              document-names)
)
(deftemplate TECHNICAL-MAINTAINANCE-MANUAL "A Structural Document Pattern Described"
        (slot                   title (type STRING))
        (slot                   title-format (type STRING))
        (slot                   introduction (type STRING))
        (slot                   introduction-format (type STRING))
        (multislot section-ids)
        (slot                   notes (type STRING))
        (slot                   notes-format (type STRING))
        (slot                   closure (type STRING))
        (slot                   closure-format (type STRING))

)
(deftemplate TECHNICAL-SPECIFICATION-MANUAL "A Structural Document Pattern Described"
        (slot                   title (type STRING))
        (slot                   title-format (type STRING))
        (multislot sections)
)
(deftemplate SECTION "A Section of Document Described"
        (slot                   id  (type STRING))
        (slot                   title (type STRING))
        (slot                   title-format (type STRING))
        (multislot paragraphs)
)
(deftemplate PARAGRAPH "A paragraph"
        (slot text)
        (slot style)
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;  Questionaire Data      ;;;;;;;;;;;;;;;
(deftemplate QUESTION "The Question Model"
        (slot text (type STRING))
        (slot type (type STRING))
        (slot ident (type STRING))
)
(deftemplate ANSWER "Temporary Holder for the Users Answers"
        (slot ident (type STRING))
        (slot text (type STRING))
)
(deftemplate STATE "Represents the Current State of the System"
        (slot current (type STRING))
)
(deftemplate SECTION-ID "Represents the Current State of the System"
        (multislot ids)
)
```

## APPENDIX C.2 - Rule Definitions

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Module ask
(defmodule ask)
(deffunction ask-user (?question ?type) "Ask a question, and return the answer"
  (bind ?answer "")
;;  (while ((not (is-of-type ?answer ?type))(not (eq ?answer ""))) do
      (printout t ?question " ")
      (if (eq ?type string) then
        (printout t "(string) "))
      (bind ?answer (read))
;;        )
  (return ?answer)
)
(deffunction is-of-type (?answer ?type) "Check that the answer has the right form"
  (printout t "checking " ?answer " is of " ?type)
  (if (eq ?type string) then (return (stringp ?answer))
    else (if (eq ?type number) then (return (numberp ?answer)))
    else (return (> (str-length ?answer) 0))
  )
)
 (defrule ask::ask-question-by-id
  "Given the identifier of a question, ask it and assert the answer"
  (declare (auto-focus TRUE))
  (MAIN::QUESTION (ident ?id) (text ?text) (type ?type))
  (not (MAIN::ANSWER (ident ?id)))
  ?ask <- (MAIN::ask ?id)
  ?fact<- (MAIN::USER (id ?x))
  =>
  (bind ?answer (ask-user ?text ?type))
  (assert (ANSWER (ident ?id) (text ?answer)))
  (set-value ?fact ?id ?answer)
  (retract ?ask)
  (return)
)
(deffunction set-value (?fact ?slot ?answer) "Set the value in the user model."
        ;;(printout t "checking " ?slot " " ?answer crlf)
            (if (eq ?slot id              ) then (modify ?fact (id               ?answer)))
            (if (eq ?slot first-name      ) then (modify ?fact (first-name       ?answer)))
            (if (eq ?slot last-name       ) then (modify ?fact (last-name        ?answer)))
            (if (eq ?slot organisation    ) then (modify ?fact (organisation     ?answer)))
            (if (eq ?slot training        ) then (modify ?fact (training         ?answer)))
            (if (eq ?slot experience      ) then (modify ?fact (experience       ?answer)))
            (if (eq ?slot education       ) then (modify ?fact (education        ?answer)))
            (if (eq ?slot qualification   ) then (modify ?fact (qualification    ?answer)))
            (if (eq ?slot job             ) then (modify ?fact (job              ?answer)))
            (if (eq ?slot task            ) then (modify ?fact (task             ?answer)))
            (if (eq ?slot activity        ) then (modify ?fact (activity         ?answer)))
            (if (eq ?slot activity-assesment) then (modify ?fact (activity-assesment  ?answer)))
            (if (eq ?slot task-frequency  ) then (modify ?fact (task-frequency   ?answer)))
            (if (eq ?slot reference-key   ) then (modify ?fact (reference-key    ?answer)))
            (if (eq ?slot reference-level ) then (modify ?fact (reference-level  ?answer)))
            (if (eq ?slot reference-file  ) then (modify ?fact (reference-file   ?answer)))
            (if (eq ?slot request         ) then (modify ?fact (request          ?answer)))
            (if (eq ?slot modality        ) then (modify ?fact (modality         ?answer)))
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Module startup
(defmodule startup)
(defrule print-banner
  =>
  (printout t "Type your name and press Enter> ")
  (bind ?name (read))
  (printout t                                        crlf)
  (printout t "*******************************************************" crlf)
  (printout t " Hello, " ?name ".                          " crlf)
  (printout t " Welcome to the Rule-Based Virtual Document Generator." crlf)
  (printout t " Please answer the questions and we will create      " crlf)
  (printout t " a Virtual Document according to your needs.         " crlf)
  (printout t "                                         " crlf)
  (printout t "*******************************************************" crlf)
  (printout t                                        crlf)
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Module get-user-model
(defmodule get-user-model)
(defrule request-id
  =>
  (assert (ask id))
  (assert (MAIN::STATE (current request-first-name)))
)
(defrule request-first-name
  (MAIN::STATE (current request-first-name))
  =>
  (assert (ask first-name))
  (assert (MAIN::STATE (current request-last-name)))
)
(defrule request-last-name
  (MAIN::STATE (current request-last-name))
  =>
  (assert (ask last-name))
  (assert (MAIN::STATE (current request-experience)))
)
(defrule request-experience
  (MAIN::STATE (current request-experience))
  =>
  (assert (ask experience))
  (assert (MAIN::STATE (current request-education)))
)
(defrule request-education
  (MAIN::STATE (current request-education))
  =>
  (assert (ask education))
  (assert (MAIN::STATE (current request-job)))
)
(defrule request-job
  (MAIN::STATE (current request-job))
  =>
  (assert (ask job))
  (assert (MAIN::STATE (current request-task)))
)
(defrule request-task
  (MAIN::STATE (current request-task))
  =>
  (assert (ask task))
  (assert (MAIN::STATE (current request-task-frequency)))
)
(defrule request-task-frequency
  (MAIN::STATE (current request-task-frequency))
  =>
  (assert (ask task-frequency))
  (assert (MAIN::STATE (current request-request)))
)
(defrule request-request
  (MAIN::STATE (current request-request))
  =>
  (assert (ask request))
  (assert (MAIN::STATE (current request-modality)))
)
(defrule request-modality
  (MAIN::STATE (current request-modality))
  =>
  (assert (ask modality))
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Module make-user-knowledge-profile
(defmodule make-user-knowledge-profile)
(defrule set
        => (assert (MAIN::add-tag disassembly-information))
)
;; Rules for knowledge selection ;;;;;;;;;;;;;;;;;;;;;;
;; for each slot in the user model we can have a number of acceptable
;; values
(defrule set2
        (MAIN::ANSWER (ident task) (text maint))
        =>
        (assert (MAIN::add-tag maintainance-knowledge))
)
(defrule append-know-tag "appends to the contents of the knowledge tank."
        ?fact<-(MAIN::USER-KNOWLEDGE-PROFILE (K-types $?ls))
        ?flag<-(MAIN::add-tag ?x) =>
        (printout t "Appending Tag: "?x crlf)
        (modify ?fact (K-types (create$ $?ls ?x)) )
        (retract ?flag)
)
;;; check for duplication
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(defrule show "shows the contents of the knowledge tank."
          (MAIN::USER-KNOWLEDGE-PROFILE (K-types $?ls))
          =>
          (printout t "##################################################" crlf)
          (printout t "The User Knowledge Profile is now set to : "$?ls crlf)
          (printout t "##################################################" crlf)
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Module get-users-query
(defmodule get-users-query)
(defrule get-query "Gets the query from the user model."
          (MAIN::USER (request ?q))
          ?fact<-(MAIN::QUERY (query ?x))
          (test (eq ?x unset))
          =>
          (modify ?fact (query ?q))
)
(defrule show-query "shows the contents of the users query."
          (MAIN::QUERY (query ?ls))
          =>
          (printout t "############################################" crlf)
          (printout t "The Query is now set to : "?ls crlf)
          (printout t "############################################" crlf)
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Module decide-user-type
(defmodule decide-user-type)
(defrule decide-user-type "shows the contents of the knowledge tank."
          ?fact<-(MAIN::USER-TYPE (user-type ?ut))
          (MAIN::USER (job engineer))
          (test (eq ?ut unset))
          =>
          (modify ?fact (user-type engineer))
)
(defrule show-user-type "shows the contents of the user type."
          (MAIN::USER-TYPE (user-type ?ls))
          =>
          (printout t "############################################" crlf)
          (printout t "The User Type is now set to : "?ls crlf)
          (printout t "############################################" crlf)
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Module select-document-structure
(defmodule select-document-structure)
(defrule set-document-structre "sets the document structure."
          ?fact<-(MAIN::DOCUMENT-TYPE (document-name ?x))
          (MAIN::USER-TYPE (user-type engineer))
          (MAIN::USER (task maint))
          (test (eq ?x unset))
          =>
          (modify ?fact (document-name technical-maintainance-manual))
)
(defrule show-document-type "shows the contents of the document type."
          (MAIN::DOCUMENT-TYPE (document-name ?ls))
          =>
          (printout t "############################################" crlf)
          (printout t "The Document Type is now set to : "?ls crlf)
          (printout t "############################################" crlf)
)
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Module select-discourse-pattern
(defmodule select-discourse-pattern)
(defrule select-discourse-pattern-1 "Selects an appropriate discourse pattern"
	?fact<-(MAIN::DISCOURSE (discourse-pattern $?x))
	(MAIN::USER-TYPE (user-type engineer))
	(MAIN::USER (task maint))
	(MAIN::USER-TYPE (user-type engineer))
	(MAIN::DISCOURSE-PATTERNS (pattern-name maintainance-discourse) (discourse-pattern $?pl))
	(test (member$ unset $?x) )
	=>
	(modify ?fact (discourse-pattern $?pl))
)
(defrule show-document-type "shows the contents of the discourse pattern."
	(MAIN::DISCOURSE (discourse-pattern $?ls))
	=>
	(printout t "##########################################" crlf)
	(printout t "The Discourse Pattern is now set to : "$?ls crlf)
	(printout t "##########################################" crlf)
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Module make-document
(defmodule make-document)
(defrule start-doc-build-technical-maint-man "start building the document."
	(MAIN::DISCOURSE (discourse-pattern $?x))
	(MAIN::DOCUMENT-TYPE (document-name ?y))
	?fact2<-(MAIN::SECTION-ID (ids $?z))
	?fact<-(MAIN::TECHNICAL-MAINTAINANCE-MANUAL (section-ids ?w))
	;;(test (printout t "discourse:"$?x crlf))
	;;(test (printout t "doc-name :"?y  crlf))
	;;(test (printout t "sectionid:"?z  crlf))
	;;(test (printout t "fact     :"?fact crlf))
	=>
	(build-document ?x ?y ?z ?fact ?w ?fact2)
	(printout t "##########################################" crlf)
	(printout t "The Document is now being built. "       crlf)
	(printout t "##########################################" crlf)
)
(defrule show-doc
	(MAIN::TECHNICAL-MAINTAINANCE-MANUAL
		(title ?x)
		(introduction ?y)
		(section-ids $?z)
		(notes ?a)
		(closure ?s)
	)
	=>
	(printout t "##########################################" crlf)
	(printout t "The Document has been generated. "       crlf)
	(printout t "##########################################" crlf)
	;;(printout t "title:"?x crlf)
	;;(printout t "introduction:"?y crlf)
	;;(printout t "sections:"$?z crlf)
	;;(printout t "notes:"?a crlf)
	;;(printout t "closure:"?s crlf)
)
(defrule show-sect
	(MAIN::SECTION
		(id ?x)
		(title ?y)
```

```
                    (paragraphs $?z)
        )
        =>
        ;;(printout t "section id  :"?x crlf)
        ;;(printout t "section title:"?y crlf)
        ;;(printout t "section parag:"$?z crlf)
)
;; Where ?discourse is the discourse for which we build.
;; ?doc-type is the document type we want
;; ?section-ids is the list of all the existing section ids
;; ?fact is the fact no of the document structure
;; ?ls is the list of the current sections in it
;; ?fact2 is the fact holding the section ids
(deffunction build-document (?discourse ?doc-type ?section-ids ?fact ?ls ?fact2) "Builds the document."
        ;; For each of then make a section with a unique id
        ;;(printout t "#### In build ###############" crlf)
        (bind ?number (length$ ?discourse))
        ;;(printout t "#### we have "?number" sections." crlf)
        (while (> ?number 0)
                ;;(printout t "#### doing section "?number crlf)
                (bind ?section-id (unique-id ?section-ids ?fact2))
                ;;(printout t "#### section id is :"?section-id crlf)
                (modify ?fact2 (ids (create$ ?section-ids ?section-id)))
                ;;(printout t "#### mod glob ids to :" ?section-ids ?section-id crlf)
                (bind ?section-ids (fact-slot-value ?fact2 ids))
                ;; Fill in the section with the topic
                (assert (SECTION (id ?section-id)
                                        (title (nth$ ?number ?discourse))
                                        ;;(paragraphs ?)
                        )
                )
                ;; Add the id of the section to the paragraphs of the document.
                (modify ?fact
                        (section-ids
                                (create$
                                        ?section-id
                                        (fact-slot-value ?fact section-ids)
                                )
                        )
                )
                (bind ?number (- ?number 1))
        )
        ;;(printout t (fact-slot-value ?fact title) crlf)
        ;;(printout t (fact-slot-value ?fact introduction) crlf)
        ;;(printout t (fact-slot-value ?fact section-ids) crlf)
        ;;(printout t (fact-slot-value ?fact notes) crlf)
        ;;(printout t (fact-slot-value ?fact closure) crlf)


)
(deffunction unique-id (?section-ids ?fact) "makes an id that is unique."
        ;;(printout t "####### In unique  ################" crlf)
        ;;(printout t "####### ids :" ?section-ids crlf)

        (while (< 1 2)
                (bind ?x (random))
                (if (not (member$ ?x $?section-ids)) then
                        ;;(printout t "####### the unique :" ?x crlf)
                        (return ?x)
                )
```

```
          )
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Module format-document
(defmodule format-document)
(defrule title-style
          (test (printout t "set title form" crlf))
          ?fact<-(MAIN::TECHNICAL-MAINTAINANCE-MANUAL
                    (title ?x)
                    (title-format unset)
          )
          =>
          (modify ?fact (title-format <b>))
)
(defrule intro-style
          (test (printout t "set intro form" crlf))
          ?fact<-(MAIN::TECHNICAL-MAINTAINANCE-MANUAL
                    (introduction ?y)
                    (introduction-format unset)
          )
          =>
          (modify ?fact (introduction-format <i>))
)
(defrule note-style
          (test (printout t "set note form" crlf))
          ?fact<-(MAIN::TECHNICAL-MAINTAINANCE-MANUAL
                    (notes ?a)
                    (notes-format unset)
          )
          =>
          (modify ?fact (notes-format "<i><b>"))
)
(defrule closure-style
          (test (printout t "set close form" crlf))
          ?fact<-(MAIN::TECHNICAL-MAINTAINANCE-MANUAL
                    (closure ?s)
                    (closure-format unset)
          )
          =>
          (modify ?fact (closure-format <i>))
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Module export-document
(defmodule export-document)
(assert (MAIN::STATE (current export1)))
(defrule export-doc-a
          (test (printout t "axp a fire" crlf))
          (MAIN::TECHNICAL-MAINTAINANCE-MANUAL
                    (title ?x)
                    (title-format ?f)
                    (introduction ?y)
                    (introduction-format ?g)
          )
          =>
          (printout t "##########################################" crlf)
          (printout t "Exporting Document. "      crlf)
          (printout t "##########################################" crlf)
          (printout t "" ?f ?x ?f crlf)
          (printout t "" ?g ?y ?g crlf)
          (assert (MAIN::STATE (current export2)))
```

```
)
;; ?fact points to the section
(deffunction export-sections (?fact2) "function for exporting the sections of the document"
        (printout t "" (fact-slot-value ?fact2 id) crlf)
        (printout t "" (fact-slot-value ?fact2 title) crlf)
)
(defrule export-doc-b
        (MAIN::STATE (current export2))
        (test (printout t "secs fire" crlf))
        ?fact<-(MAIN::TECHNICAL-MAINTAINANCE-MANUAL (section-ids $?z))
        ?fact2<-(MAIN::SECTION (id ?x))
        (test (not (eq (nth$ 1 $?z) 0)))
        =>
        (printout t "next" (nth$ 1 $?z) crlf)
        (printout t "ids left:" $?z crlf)
        (modify ?fact (section-ids (delete$ $?z 1 1)))
        (export-sections ?fact2)
        (assert (MAIN::STATE (current export3)))
)
(defrule export-doc-c
        (MAIN::STATE (current export3))
        (test (printout t "Notes fire" crlf))
        (MAIN::TECHNICAL-MAINTAINANCE-MANUAL
                (notes ?a)
                (notes-format ?h)
                (closure ?s)
                (closure-format ?k)
        )
        =>
        (printout t "" ?h ?a ?h crlf)
        (printout t "" ?k ?s ?k crlf)
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Main
(deffunction run-system ()
  (reset)
  (focus startup
        get-user-model
        make-user-knowledge-profile
        get-users-query
        decide-user-type
        select-document-structure
        select-discourse-pattern
        make-document
        format-document
        export-document)
  (run)
)
;;   (while TRUE
  (run-system)
;;   )
```

## APPENDIX D – PDM Systems

The Product Data Management (PDM) Systems are Database systems that enable control of the product design data and the design process itself. They provide Component and Document classification as well as a product tree abstraction and facilities for easily querying the data as part of their data management procedures. Furthermore, they offer a data vault with versioning utilities.

Process Management is the second main advantage of PDM systems. This includes Work Management, Workflow Management, and Work History Management. Work Management arranges the roles and persons that have control over specific subsets of data. Workflow Management defines a timeline that manages the transfer of control over the data from one user to another. And Work Management keeps track of all the events and actions that occur during the products life-cycle.

Benefits of PDM systems include:

- Reduced Time-to-Market
- Improved Design Productivity
- Improved Design and Manufacturing Accuracy
- Better use of Creative Team Skills
- Comfortable to Use
- Data Integrity Safeguarded
- Better Control of Projects
- Better Management of Engineering Change
- A Major Step Toward Total Quality Management

# References

**[Bales02]** Bales, D. K., (2002), **“Java Programming with Oracle JDBC”**, Thomson International, MA, USA.

**[Bezanson95]** Bezanson, W. R. (1995), **"Performance Support Online, Integrated Documentation and Training"**. Proceedings of the 13th Conference on Engineering from Chaos: Solutions for the Growing Complexity of our Jobs, Sept 30-Oct 3, 1995, Savannah, GA, USA, pp. 1-10.

**[Bist96]** Bist, G. (1996), **"Applying the Object-Oriented Model to Technical Information"**. IEEE Transactions on Professional Communication, Vol. 39, pp.49-57, 1996.

**[Cantanto96]** Cantanto, M. (1996), **“Vision 2000: Multimedia Electronic Performance Support Systems”**, Proceedings of the 14th Annual International Conference on Marshalling New Technological Forces: Building a Corporate, Academic, and User Oriented Triangle, Research Triangle – United States, Oct 19-22, pp. 111-114.

**[Cawsey98]** Cawsey, A, (1998), **“The Essence of Artificial Intelligence”**, Prentice Hall, London, UK.

**[Cgi04]** 2004, **The Common Gateway Interface** , [WWW] URL: http://hoohoo.ncsa.uiuc.edu/cgi/ [Accessed 17 February 2004].

**[Church95]** Church, K.W., Rau, L.F., (1995), **"Commercial Applications of Natural Language Processing"**, Communications of the ACM, Vol. 38, Issue 11, pp.71-79.

**[CLIPS04]** 2004, **The CLIPS Programming Language - Wikipedia**, [WWW] URL: http://en.wikipedia.org/wiki/CLIPS_programming_language [Accessed 17 February 2004].

**[Cookie04]** 2004, **Cookie Central – The Cookie Concept**, [WWW] URL: http://www.cookiecentral.com/c_concept.htm [Accessed 17 February 2004].

**[Desmarais97]** Desmarais, M. C., Leclair, R., Fiset, J., and Talbi, H., (1997), Cost-Justifying Electronic Performance Support Systems, Communications of the ACM, Vol. 40, No. 7, Jul. 1997, pp.39-48.

**[Doi03]** 2003, **DOI - Digital Object Identifier**, [WWW] URL: www.doi.org [Accessed 17 February 2003].

**[Dsssl04]** 2004, **DocBook DSSSL StyleSheets**, [WWW] URL: http://docbook.sourceforge.net/projects/dsssl [Accessed 23 December 2004].

**[Earp03]** Earp, R. and Bagui, S., (2003), **"Learning SQL: a step by step guide using Oracle"**, Addison-Wesley, MA, USA.

**[Emorphia05] 2005, Emorphia Home Page,** [WWW] URL:http://www.emorphia.com [Accessed 14 January 2004].

**[Fipa04]** 2004, **FIPA Home Page**, [WWW] URL: http://www.fipa.org [Accessed 14 January 2004].

**[Forgy82]** Forgy, C.L., (1982), **"Rete: A Fast Algorithm for the many pattern / many object pattern match problem",** Artificial Intelligence, 19(1), pp.17-37, 1982.

**[Friedman03]** Friedman-Hill, E., (2003), **"Jess in Action: Rule-Based Systems in Java"**, Manning Publications, Greenwich, UK.

**[Friendly95]** Friendly, L., (1995), **"The Design of Distributed Hyperlinked Programming Documentation"**, Proceedings of the International Workshop on Hypermedia Design '95 (IWHD '95), F. Garzotto et al. eds., Springer, Monpellier, pp. 151–183.

**[Gamma95]** Gamma, E, Helm, R., Johnoson, R., Vlissides, J., (1995), **"Design Patterns: Elements of Reusable Object-Oriented Software",** Addison-Wesley, MA, USA.

**[Giarratano94]** Giarratano, J. C, (1994), **"Expert Systems: Principles and Programming",** PWS Publishing, Boston, MA, USA.

**[Goldfarb99]** Goldfarb, C. F. and Prescod, P., (1999), **"The XML Handbook"**, Prentice Hall, London, UK.

**[Google05]** 2005, **Google Home Page**, [WWW] URL: http://www.google.co.uk [Accessed 14 January 2005].

**[Gruber95]** Gruber, T.R, Vemuri, S., Rice, J., (1995), **"Virtual Documents that explain How Things Work: Dynamically generated question-answering documents",** Knowledge Systems Laboratory, Stanford University, Technical Report, 1995.

**[Gruber97]** Gruber, T.R, Vemuri, S., Rice, J., (1997), **"Model-based virtual document generation",** International Journal of Human-Computer Studies, Vol. 46, Issue 6, pp. 687-706.

**[Gruber93]** Gruber, T.R., (1993), **"Toward Principles for the Design of Ontologies Used for Knowledge Sharing"**, Knowledge Systems Laboratory, Stanford University, Technical Report, 1993.

**[Haake01]** Haake, J.M., (2001), **"Applying Collaborative Open Hypermedia Concepts to Extended Enterprise Engineering and Operation"**, Open Hypermedia Systems and Structural Computing: 6th International Workshop, OHS-6, 2nd

International Workshop, SC-2, San Antonio, Texas, USA, May 30 - June 4, 2000, pp. 17-27.

**[Harris95]** Harris, S.L. and Ingram, J.H. (1995), **"Real Information, Virtual Documents"**, Proceedings of the 13th annual International Conference on Systems Documentation, (pp.71--76), Savannah, Georgia, United States.

**[Hytime04]** 2004, **ISO 10744: Hypermedia/Time-based Structuring Language (HyTime)**, 2nd edition, [WWW] URL: http://xml.coverpages.org/hytime.html [Accessed 31 December 2004].

**[Ions03]** IONS, (2003), **"Information Object Numbering Systems"**, [WWW] URL : http://www.jisc.ac.uk/uploaded_documents/ACF5B.doc [Accessed 12 November 2004].

**[Jackson98]** Jackson, P, (1998), **"Introduction to Expert Systems"**, Addison-Wesley Longman Limited, Essex, UK.

**[Javadoc04]** 2004, **javadoc-The java API Documentation Generator**, [WWW] URL: http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/javadoc.html [Accessed 10 January 2004].

**[Jennings98]** Nicholas R. Jennings and Michael J. Wooldridge, **"Applications Of Intelligent Agents"**, in Nicholas R. Jennings and Michael J. Wooldridge (Ed.), Agent Technology Foundations, Applications, and Markets , Springer-Verlag, NJ, USA, 1998.

**[Jennings00]** N. R. Jennings (2000) **"On Agent-Based Software Engineering"** Artificial Intelligence, 117 (2) 277-296.

**[Kemmerer99]** Kemmerer, S. J., (1999), **"STEP: The Grand Experience"**, Manufacturing Engineering Laboratory, National Institute of Standards and Technology, NIST Special Edition, Technical Report.

**[Klusch01]** Klusch, M., (2001), **"Information Agent Technology for the Internet: A Survey"**, Journal on Data and Knowledge Engineering, Special Issue on Intelligent Information Integration, D. Fensel (Ed.), Vol. 36(3), pp.337-372.

**[Knuth84]** Knuth, D., E., (1984), **"Literate Programming"**, Oxford University Press, The Computer Journal, Vol.27, Issue 2, pp. 91-111, UK.

**[Lesser03]** Lesser, V., Horling, B., Klassner, F., Raja, A., Wagner, T. and Zheng, S. X. Q., (2003), **"BIG: A Resource-Bounded Information Gathering Agent"**, UMass Computer Science Technical Report 1998-03, [WWW] URL: http://dis.cs.umass.edu/research/big/big.html [Accessed 12 November 2003].

**[Maes95]** Maes, P., (1995), **"Artificial Life Meets Entertainment: Life Like Autonomous Agents"**, Communications of the ACM, Vol. 38(11), pp.108-114, 1995.

**[Matthews92]** Sky Matthews, Carl Grove,(1992), **"Applying Object-Oriented Concepts to Documentation"**, Proceedings of the 10th annual international conference on Systems documentation, (pp.265-271), Ottawa, Ontario, Canada, 1992.

**[Musciano97]** Musciano, C. and Kennedy, B., (1997), **"HTML, The Definitive Guide"**, O'Reilly, 2nd edition, CA, USA.

**[Negnevitsky02]** Negnevitsky, M, (2002), **"Artificial Intelligence: A Guide to Intelligent Systems"**, Addison-Wesley/Pearson Education , Harlow, England.

**[Netsage02]** 2002, **NetSage: Network Monitoring System Configuration Database**, [WWW] URL: http://www.net.cmu.edu/netsage/index.html [Accessed 22 March 2002].

**[Netscape04]** 2004, **Client Side State – HTTP Cookies**, [WWW] URL: http://wp.netscape.com/newsref/std/cookie_spec.html [Accessed 22 March 2004].

**[Nodine00]** Nodine, M., Fowler, J., Ksiezyk, T., Perry, B., Taylor, M. and Unruh, A., (2000), **"Active Information Gathering in InfoSleuth"**, International Journal of Cooperative Information Systems, Vol. 9, Issues 1-2, pp. 3-28.

**[Noy2001a]** N. F. Noy, M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, and M. A. Musen. **"Creating Semantic Web Contents with Protege-2000"**. IEEE Intelligent Systems 16(2):60-71, 2001.

**[Noy2001b]** N. Noy and D. L. McGuinness. **"Ontology Development 101: A Guide to Creating Your First Ontology"**. Technical Report, Knowledge Systems Laboratory, Stanford University, 2001.

**[Oro04]** 2004, **Jakarta ORO**, [WWW] URL: http://jakarta.apache.org./oro [Accessed 14 March 2004].

**[Owen97]** Owen, J., (1997), **"STEP: An Introduction"**, Information Geometers, 2nd edition, Winchester, UK.

**[Paradis98a]** Paradis, F. and Vercoustre, A.N. and Hills, B. (1998), **"A Virtual Document Interpreter for Reuse of Information"**, Proceedings of Electronic Publishing '98, published as Lecture Notes on Computer Science 1375, (pp.487-498), Saint-Malo, France.

**[Paradis98b]** Vercoustre, A.N., Paradis, F., (1998), **"Reuse of Linked Documents through Virtual Document Prescriptions"**, Lecture Notes in Computer Science, Vol. 1375, pp. 499-512.

**[Perry04]** Perry, B. W., (2004), **"Java servlet and JSP cookbook"**, O'Reilly, CA, USA.

**[Pham99]** Pham, D. T., Dimov, S. S. and Setchi, R. M., (1999), **"Intelligent Product Manuals"**, Proceedings of the Institution of Mechanical Engineers, IMechE, Vol. 213, Part I, pp. 65-76.

**[Pham00]** Pham, D. T., Dimov, S. S. and Peat, B. J., (2000), **"Intelligent Product Manuals"**, Proceedings of the Institution of Mechanical Engineers, IMechE, Vol. 214, Part B, pp. 411-419.

**[Pham02]** Pham, D. T., Setchi, R. M. and Dimov, S. S., (2002), **"Enhanced Product Support through Intelligent Product Manuals"**, International Journal of Systems Science, Vol. 33, Issue 6, pp. 433-449.

**[Poslad00a]** Poslad S. J., Buckle S. J., Hadingham R., (2000), **''The FIPA-OS agent platform: Open Source for Open Standards,''** Proceedings of PAAM 2000, Manchester UK, April 2000, pp. 355–368.

**[Poslad00b]** Poslad S. J., Calisti, M., (2000), **''Towards improved trust and security in FIPA agent platforms,''** In Autonomous Agents 2000 Workshop on Deception, Fraud and Trust in Agent Societies, Barcelona, June 2000.

**[Price97]** Price, J. (1997), **"Introduction: Special Issue on structuring complex information for electronic publication"**, IEEE Transactions on Professional Communication, Vol. 40, pp.69-77, 1997.

**[Ranwez99]** Ranwez, S. and Crampes, M. (1999), **"Conceptual Documents and Hypertext Documents are two Different Forms of Virtual Document"**, Workshop on Virtual Documents, Hypertext Functionality and the Web, In 8th International World Wide Web Conference, Toronto, Canada, May 1999.

**[Reiter95]** Reiter, E. Mellish, C. and Levine, J. (1995), **"Automatic Generation of Technical Documentation"**, Journal of Applied Artificial Intelligence, Vol. 9, pp. 259-287, 1995.

**[Rio03]** 2003, **RIO - Reusable Information Object Strategy**, [WWW] URL: http://www.cisco.com/warp/public/779/ibs/solutions/publishing/whitepapers/ [Accessed 12 May 2003].

**[Rodriguez04]** Rodriguez, I., Nunez, M. and Rubio, F., (2004), **"Specification of Autonomous Agents in E-commerce Systems"**, Lecture Notes in Computer Science, Vol. 3236, pp. 30-44.

**[Rudolph04]** 2004, **Some Guidelines for Deciding Whether to Use A Rules Engine**,[WWW] URL: http://herzberg.ca.sandia.gov/jess/guidelines.shtml [Accessed 30 December 2004].

**[Rumbaugh91]** Rumbaugh, J, (1991), **"Object-Oriented Modelling and Design"**, Prentice Hall, NJ, USA.

**[Setchi99]** Setchi, R., (1999), **"Enhanced Product Support through Intelligent Product Manuals"**, PhD Thesis, Cardiff University, UK.

**[Sgml04]** 2004, **Standard Generalised Markup Language (SGML)**, [WWW] URL: http://xml.coverpages.org/sgml.html [Accessed 30 December 2004].

**[Sleight93]** Sleight, D. A., (1993), **"Types of Electronic Support Systems: Their Characteristics and Range of Desings"**, Educational Psycology Michigan State University, [WWW], <URL: http://www.msu.edu/~sleightd/epss_copy.html> [Accessed 30 September 2004].

**[Soroka00]** Soroka, A., (2000), **"An Agent Based System for the Acquisition and Management of Fault Knowledge for Intelligent Product Manuals"**, PhD Thesis, Cardiff University, UK.

**[Sun04]** 2004, **Java Servlet Technology**, [WWW] URL: http://java.sun.com/products/servlet/ [Accessed 30 December 2004].

**[Sycara03]** Sycara, K., Paolucci, M. Van Velsen, M. and Giampapa, J., (2003), **"The RETSINA MAS Infrastructure"**, Autonomous Agents and Multi-Agent Systems, Vol. 7, Issue 1-2, pp. 29-48.

**[Tomcat03]** The Jakarta Site, (2003), **"Apache Jakarta Tomcat"**, [WWW] URL : http://jakarta.apache.org/tomcat/ [Accessed 12 November 2003].

**[Tucker97]** Tucker, H. and Harvey, B. (1997), **"SGML Documentation Objects within the STEP Environment"**, In SGML Europe '97, Barcelona, Spain. [http://www.eccnet.com/papers/step.html].

**[Turban88]** Turban, E., (1988), **"Decision support and expert systems: managerial perspectives"**, Collier Macmillan, London, UK.

**[Tveit01]** Tveit, A., (2001), **"A survey of Agent-Oriented Software Engineering."** Proceedings of the First NTNU Computer Science Graduate Student Conference. Norwegian University of Science and Technology May 2001.

**[Ventura00]** Ventura, C. A., (2000), **"Why Switch From Paper to Electronic Manuals?"**, Proceedings of the ACM Conference on Document Processing Systems, January 2000, pp.111-116.

**[Vercoustre97a]** Vercoustre, A.M., Dell'Oro, J., and Hills, B., (1997), **"Reuse of Information through Virtual Documents"**, in Second Australian Document Computing Symposium, Melbourne Australia, pp. 55-64.

**[Vercoustre97b]** Vercoustre, A.N. and Paradis, F. (1997), **"A Descriptive Language for Information Object Reuse through Virtual Documents"**, In 4th International Conference on Object-Oriented Information Systems (OOIS'97), Brisbane, Australia, pp. 239-311, November 1997.

**[Williams04]** Williams, H. E. and Lane, D., (2004), **"Web database applications with PHP and MySQL"**, O'Reilly, 2nd edition, CA, USA.

**[Wooldridge97]** Wooldridge, M., (1997), **"Agent-Based Software Engineering"**, IEE Proceedings Software Engineering, 144(1), pages 26-37, February 1997.

**[Wooldridge99]** Wooldridge M. J., Jennings N. R. and Kinny D. **"A methodology for Agent-Oriented analysis and design"**, *Proceedings of the third International Conference on Autonomous Agents* , pages 69-76, Washington, USA, 1999.

**[Wurman98]** Wurman, P. R., Wellman, M. P. and Walsh, W. E. (1998), **"The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents"**, Proceedings of the Second International Conference on Autonomous Agents (Agents-98), May 1998, Minneapolis, USA, [WWW] URL: http://www.csc.ncsu.edu/faculty/wurman/Papers/Wurman-Agents98.pdf [Accessed 14 February 2003].

**[Xerces02]** 2002, **Xerces Java Parser**, [WWW] URL: http://xml.apache.org/xerces-j [Accessed 2 December 2002].