# A SOFTWARE PRODUCT LINE APPROACH TO ONTOLOGY-BASED RECOMMENDATIONS IN E-TOURISM SYSTEMS

**BY**

**JUSTINE OLAWANDE DARAMOLA**
**(CU05PG000158)**

**A THESIS SUBMITTED TO THE DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES TO THE SCHOOL OF POSTGRADUATE STUDIES**

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF DOCTOR OF PHILOSOPHY DEGREE OF COVENANT UNIVERSITY OTA, OGUN STATE, NIGERIA

June 2009

# CERTIFICATION

This is to certify that this thesis is an original research work undertaken by **Justine Olawande Daramola** and approved by:

1. Name: **Professor Mathew Olusegun Adigun**
   **Supervisor**

   Signature [illegible] Date: ____________

2. Name: **Dr. Charles Korede Ayo**
   **Co-Supervisor**

   Signature ____________ Date: ____________

3. Name: **Dr. Ezekiel Femi Adebiyi**
   **Head of Department**

   Signature ____________ Date: ____________

4. Name: **Professor Adetokunbo Babatunde Sofoluwe**
   **External Examiner**

   Signature ____________ Date: ____________

## DECLARATION

It is hereby declare that this research was undertaken by Justine Olawande Daramola. The thesis is based on his original study in the department of Computer and Information Sciences, College of Science and Technology, Covenant University, Ota, under the supervision of Prof. M. O. Adigun and Dr. C.K Ayo. Ideas and views of this research work are products of the original research undertaken by Justine Olawande Daramola and the views of other researchers have been duly expressed and acknowledged.

**Prof. M. O. Adigun**
**Supervisor**

Signature [illegible] Date ____________________

**Dr. C.K. Ayo**

**Co-Supervisor**

Signature ____________________ Date ____________________

# DEDICATION

This thesis is dedicated to God Almighty, the Omnipotent and Omniscient, the giver of life and the ingenious architect of my destiny for His faithfulness and immense graciousness towards me.

## ACKNOWLEDGEMENTS

I am deeply indebted to God my Father and Friend, the author of wisdom and understanding for His faithfulness and generous endowments of grace that saw me through my doctoral studies. My deep and sincere appreciation goes to the Chancellor, Dr. David Oyedepo and the members of the Board of Regents of Covenant University for the vision and mission of the University. Also, special thanks to the management staff of the University: the Vice Chancellor, the Registrar, the Deans of the Colleges, the Heads of Department for their commitment to the pursuit of excellence and sound academic scholarship.

My earnest appreciation goes to my supervisors, Professor Mathew Adigun in his capacity as my main supervisor who provided the research direction and qualitative guidance for the work. I am deeply appreciative of the fellowship he offered me to visit his research centre in South Africa, an adventure that gave focus to my research work, brought immense capacity development and opened me up to new vistas of academic opportunities and provided a gateway to international scholarship. I thank Dr. Charles Ayo, my co-supervisor, for his fatherly disposition, ever-ready willingness to assist and motivate, and more importantly his critical review of the work and useful suggestions towards its completion.

Thanks to Dr. Ezekiel Adebiyi, for his encouragement, support and motivation, and in his capacity firstly, as my teacher since undergraduate days and currently in his capacity as the Acting Head, Department of Computer and Information Sciences. I also thank the Dean of the College of Science and Technology (CST), Professor James Katende for his support and encouragement all through the course of the work. I thank the Dean of Post Graduate School, Professor C.O. Awonuga and all my teachers at the postgraduate school. I thank all my friends and colleagues in the Department of Computer and Information Sciences for their support and encouragement all through the course of this work. My special appreciation goes to Dr. Nicholas Ikhu-Omeregbe whose thesis provided useful guide for my writing. I also thank Dr. Innocent Chiluwa for helping to proof-read my thesis for grammatical and lexical accuracy.

Lastly, but deliberately I want to pay tribute to a special category of people in my life. I call them *'My Heroes, My Mentors'*. These are the people that God have used to discover, nurture and deepen my academic career. Firstly, I thank Dr. O.O. Olugbara, my teacher and a friend who supervised my undergraduate work. The wonderful seeds that he sowed at that time have indeed germinated into a vibrant growing tree. I also appreciate his contributions to my Ph.D work particularly at the onset. My thanks, also goes to Professor O.M. Bamigbola who supervised my masters degree thesis at the University of Ilorin and gave me a sound foundation in academic scholarship. He also facilitated my association with Professor Mathew Adigun. In the last four years of my career, I have enjoyed a very impactful and generous relationship with Professor Mathew Adigun, who has been for me a great mentor, a ladder and a crucial divinely-ordained connector to my next level. His deep passion and appetite for research and intellectual innovations are rare and very contagious, for this I am grateful. I am indebted to Dr. Charles Ayo for providing an atmosphere that facilitated a catalytic lift in my academic career while in office as the Acting Head of the Department of Computer and Information Sciences. The challenges and responsibilities delegated to me under his leadership greatly engendered the development of my academic and leadership potentials. Also, I must not fail to mention crucial incidences of destiny that brought me across Professor Sofoluwe and Dr. Adegboyega Ojo (Guru) both of the Department of Computer Sciences, University of Lagos in the early stages of my undergraduate life. The unique privilege of working as an undergraduate intern at the Joint Konsult Computer Services Limited, Iwaya, Lagos, offered me the opportunity of being under the tutelage of Professor Sofoluwe. It was during this period that I caught the dream of an academic career. I thank Professor Sofoluwe for being a good role model and providing an intellectually stimulating environment that encouraged a vital seed of destiny to be sown in my young life. I thank Dr. Adegboyega Ojo (Guru) who I also met at Joint Konsult Limited for helping me to appreciate the need to develop a hunger for continuous learning and a versatile disposition to the pursuit of knowledge, Dr. Ojo taught me so much, that I thought he knew everything, for this I remain grateful.

A very special class of people in my life are those that I call divine gifts. Firstly, my appreciation goes to my parents, Elder and Dcns. Daramola for investing their lives and resources to provide for me quality moral and formal education. Finally, my lovely wife Fisayo Oluyemisi and my two brilliant soldiers David and Davida, it is a privilege to have all of you as mine, thanks for your support and contributions to the success of this endeavour, God keep you all for me.

## ABSTRACT

This study tackles two concerns of developers of Tourism Information Systems (TIS). First is the need for more dependable recommendation services due to the intangible nature of the tourism product where it is impossible for customers to physically evaluate the services on offer prior to practical experience. Second is the need to manage dynamic user requirements in tourism due to the advent of new technologies such as the semantic web and mobile computing such that e-tourism systems (TIS) can evolve proactively with emerging user needs at minimal time and development cost without performance tradeoffs.

However, TIS have very predictable characteristics and are functionally identical in most cases with minimal variations which make them attractive for software product line development. The Software Product Line Engineering (SPLE) paradigm enables the strategic and systematic reuse of common core assets in the development of a family of software products that share some degree of commonality in order to realise a significant improvement in the cost and time of development. Hence, this thesis introduces a novel and systematic approach, called Product Line for Ontology-based Tourism Recommendation (PLONTOREC), a special approach focusing on the creation of variants of TIS products within a product line. PLONTOREC tackles the aforementioned problems in an engineering-like way by hybridizing concepts from ontology engineering and software product line engineering. The approach is a systematic process model consisting of product line management, ontology engineering, domain engineering, and application engineering. The unique feature of PLONTOREC is that it allows common TIS product requirements to be defined, commonalities and differences of content in TIS product variants to be planned and limited in advance using a conceptual model, and variant TIS products to be created according to a construction specification. We demonstrated the novelty in this approach using a case study of product line development of e-tourism systems for three countries in the West-African Region of Africa.

**CONTENTS**

# LIST OF FIGURES

# LIST OF TABLES

# ABBREVIATIONS

| | |
|---|---|
| AO | Accommodation Ontology |
| ARS | Accommodation Recommender System |
| CRS | Commonality and Reuse Strategy |
| DCO | Destination Context Ontology |
| DL | Description Logics |
| DRS | Destination Recommender System |
| EJB | Enterprise Java Bean |
| Java EE | Java Enterprise Edition |
| OWL | Web Ontology Language |
| PLONTOREC | Product Line for Ontology-based Tourism Recommendations |
| RDF | Resources Description Framework |
| RRS | Restaurant Recommender System |
| RS | Recommender Systems |
| SAS | Simple Application Strategy |
| SPL | Software Product Line |
| SPLE | Software Product Line Engineering |
| TIS | Tourism Information Systems |
| TRS | Tourism Recommender Systems |

# CHAPTER ONE
# INTRODUCTION

## 1.1 BACKGROUND INFORMATION

One critical challenge sequel to the advent of globalization is information explosion, particularly with respect to the gamut of information available on the web. This often leads to the phenomenon of information overload. A typical scenario is where people get too much irrelevant information alongside relevant ones as a response to queries posed on the web. Intelligent search agents are a class of software application that has been mostly engaged to solve this problem. Notable search agents on the web include google (www.google.com), mama (www.mama.com), and altavista (www.altavista.com) to mention just a few. However, in recent times another class of intelligent software applications that has gained relevance in addressing the problem of information overload when searching for relevant information on the web is recommender systems (Konstan et al., 2004).

Recommender Systems (RS) are a class of intelligent software applications that offer suggestions to information-seeking users as a response to user queries or knowledge gained during interaction with the user. They mostly leverage in-built logical reasoning capability or algorithmic computational schemes to deliver their recommendation functionality. Over the years, RS have enjoyed great application in the e-commerce domain because of their ability to provide assistance to information-seeking users.

In the tourism domain, recommendation services are particularly important because of the information-intensive nature of the tourism industry where access to relevant and useful information is advantageous both to the consumer and the marketer of tourism products (Henriksson, 2005). Tourism Recommender Systems (TRS) are a class of RS that are usually embedded in Tourism Information Systems (TIS) in order to deliver intelligent travel guide and planning recommendation functionalities. TIS are software applications that are deployable on the web and on small hand-held devices dedicated to the provision of tourism support services. TIS share many attributes in common and generally perform similar functions such as providing useful information to prospective tourists and helping in travel planning and management. They mainly differ in the nature of local information content they deliver and the scope of tourism interest that is being promoted. They can be variously engaged in the promotion of tourism at the local and enterprise, provincial, regional, national, and continental levels. This degree of observable similarity in TIS makes them good candidates for a product line development initiative, which seem not yet a prevalent practice in the e-Tourism domain. However, the fact that Tiscover AG (http://www.Tiscover.com) renders tourism support services for eight different countries around the world is a clear indication of the viability of Software Product Line Engineering (SPLE) in the tourism domain.

SPLE is a software development paradigm that enables the strategic and systematic reuse of core assets in the development of a family of software products that share some features in common. It leverages the existence of certain core reusable components in order to realise a significant improvement in the cost and time of development (Daramola et al., 2009). A Software Product Line (SPL) is a set of software intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a set of core assets in a prescribed way (Bass & Kazman, 2003).

Generally, prospective tourists want more intelligence in TIS that can make the quality of service more acceptable. One of the core desirable areas is the aspect of tourism recommendations, as it relates to travel advisory and planning services. However, literature has revealed that one of the fundamental problems of existing recommender systems' algorithms is the issue of trustworthiness of recommendations (Sarwar et al., 2001; Adomavicius & Tuzhilin, 2005; Adomavicius, 2005). Customers need recommendations they can trust, one that minimizes *false positive errors* - a scenario in which products are recommended (positive), though the customer does not like them (false). The need for trustworthy tourism recommendations is particularly compelling because of the intangible nature of the tourism product in which customers cannot physically evaluate the services on offer until practically experienced.

Also, the advent of new emerging technologies like the semantic web and mobile computing is making the modern day tourist to exhibit highly sophisticated, dynamic and increasing complex consumer characteristics that is now a concern for the providers of e-tourism services (Werthner & Klein, 1999; Steinbauer, 2005). What is yet to appear in literature is software product development approach through which e-tourism systems can be made to evolve intelligently in tandem with dynamic user requirements. These aforementioned scenarios present two critical challenges for developers of TRS which are (1) how to boost the quality of tourism recommendations and make them more dependable to foster users' confidence and (2) how to respond to the dynamic nature of user requirements whereby variants of TIS products can evolve with emerging user needs at minimal time and development cost without performance tradeoffs.

In this thesis, we address these two concerns. First, we have adopted an ontology-based approach in tackling the challenge of improving the dependability (trustworthiness) of tourism recommendations. Ontology has been used to introduce multi-dimensionality into tourism recommendations with the use of contextual information in order to eliminate the limitations of existing tourism recommendation formalisms which are strictly two dimensional.

Secondly, we have adopted SPLE-based approach, which explores the similarity in the functionalities of TIS in tackling the problem of dynamic user requirements in tourism. In this thesis, SPLE is considered viable and promising in the tourism domain because it enables the definition of system instances dictated by marketing and product plan specifications from prospective users and make dynamic software evolution a part of its core practice which tallies with the dynamism of the e-tourism domain. SPLE is expected to (1) engender the development of adaptable core reusable components that will deliver desirable tourism recommendation services; and (2) provide a way to effectively manage emerging user requirements variations with respect to specific tourism promotion scenarios.

## 1.2 STATEMENT OF THE PROBLEM

Tourism is an information-intensive industry with a long value chain of stakeholders. The tourism product has an intangible nature which makes it impossible for prospective travellers to touch the product before the trip. These two factors make the need for dependable recommendations in tourism most compelling. However, one of the fundamental concerns of existing recommender systems is the need to improve the trustworthiness of recommendations (Adomavicius & Tuzhilin, 2005; Adomavicius, 2005). Therefore, in order to obtain dependable tourism recommendations, the recommender systems that are employed in tourism must necessarily tackle the issue of trustworthiness of recommendations. The advent of new technologies such as the semantic web and mobile computing has introduced additional dynamic challenges to e-tourism requirements and increased the sophistication and complexity of consumer behavior in tourism (Werthner & Klein, 1999; Steinbauer, 2005). These dynamic challenges must be addressed by next generation TIS if they are to outperform existing TIS platforms (Staab et al., 2002; Steiner, 2002). Next generation TIS are those that are equipped with semantic web, context-aware and content-sharing capabilities that can cater for the dynamic challenges of the e-tourism domain. (Staab et al., 2002; Daramola et al., 2008).

The research presented in this thesis is intended to address the challenge of improving the dependability of tourism recommendations and providing timely response to dynamic user requirements in e-tourism. Concisely, the research questions investigated in this thesis are:

- *How do we improve the dependability of tourism recommendations in a way that foster users' confidence? And*
- *How can a TIS developer organization build such intelligent TIS in tandem with the dynamic nature of user requirements in e-tourism?*

## 1.3. AIM AND OBJECTIVES OF THE STUDY

The aim of this research work is to evolve a new product line approach that will engender improved dependability of tourism recommendations and dynamic software evolution of TIS. To achieve this aim, the following objectives were formulated:

- To create a theoretical and product-oriented framework from which credible African-based tourism promotion initiatives can evolve in tandem with state-of-the-art practices in the global e-tourism domain;
- To demonstrate the potential of the software product line engineering paradigm in responding to the dynamism of the e-tourism industry, by evolving a generic reference architecture for intelligent component services in e-tourism;
- To build an infrastructural asset-base of intelligent components for tourism recommendations. This will include a suite of tourism ontologies and tourism recommender system components;
- To investigate the possibility of improving the quality and dependability of tourism recommendations by using an ontology-based approach; and
- To validate the plausibility of the novel approach by using a case study of product line development for a west-African tourism context.

## 1.4 METHODOLOGY

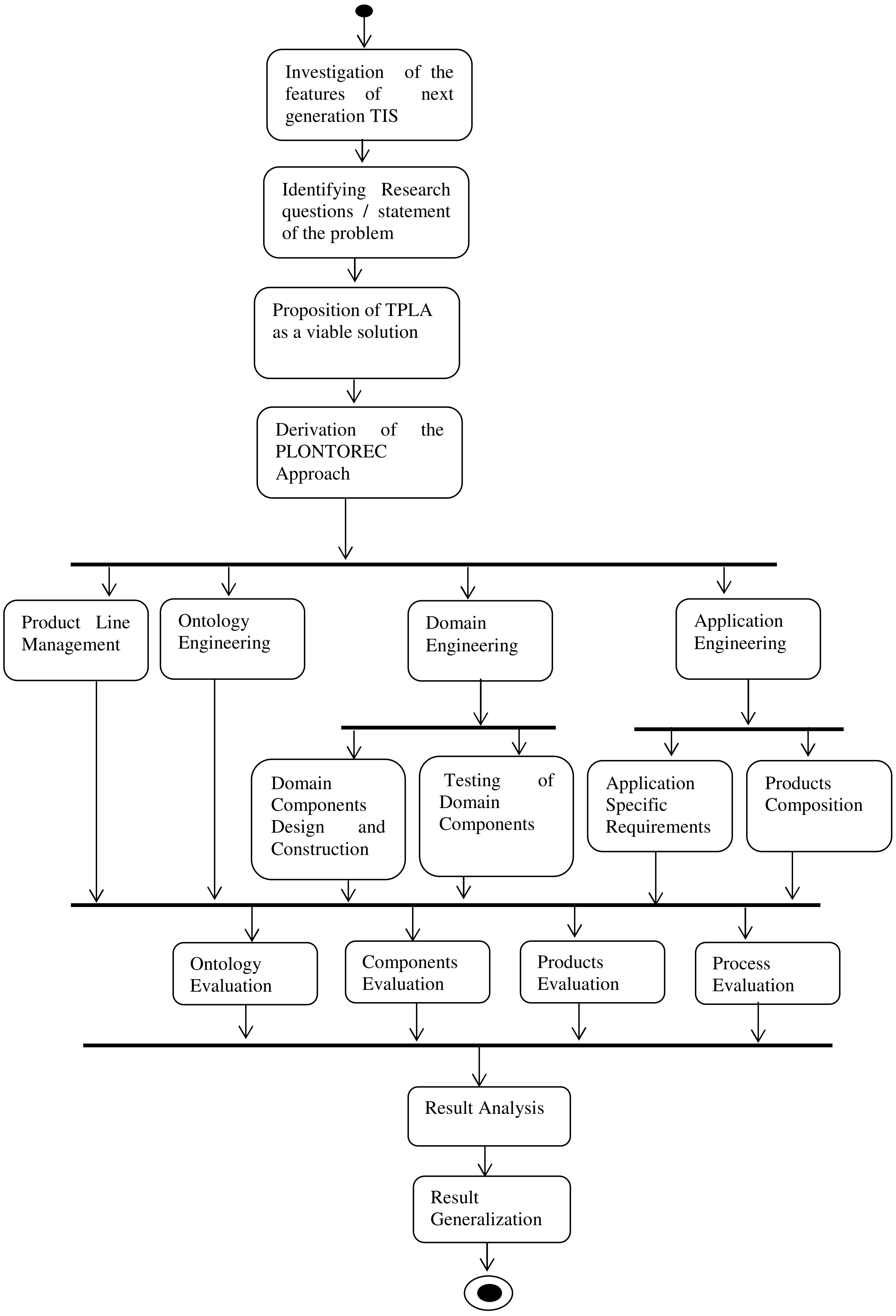
In setting out to achieve the stated objectives of this thesis, first we selected to pursue a PL approach as a candidate solution model to realizing improved intelligence in TIS and proactive management of dynamic user requirements in e-tourism. This was aimed at creating a foundational platform for developing reusable components with intrinsic intelligent attributes that can be leveraged in the development of next generation TIS. To realize this, we analysed the state-of-the-art in the tourism domain, through an extensive review of literature, a study of many existing TIS, and identification of the base requirements for TIS. This culminated in the creation of generic reference Tourism Product Line Architecture (TPLA) that is proposed as a potential platform for the evolution of intelligent component services in e-tourism, and particularly next generation TIS.

Secondly, using the proposed TPLA as a platform, this thesis introduces a novel unified solution approach called **P**roduct **L**ine for **On**tology-based **To**urism **Rec**ommendations (PLONTOREC) in order to facilitate improved dependability of tourism recommendations from TIS and proactive evolution of such TIS in response to dynamic user requirements in tourism. The PLONTOREC approach is a hybrid of software product line engineering and ontology engineering that is dedicated to the production of recommendation-intensive TIS. The PLONTOREC development lifecycle (discussed in Chapter 3), which consists of Product line management, Ontology engineering, Domain engineering and Application engineering was systematically demonstrated to realize the objectives of this thesis.

During the product line management phase, the feasibility and risk assessment of the PL-based approach was attended to. Also, issues of configuration management, organization, evaluation, and control of product line process were considered based on established product line practice models. Next, ontology engineering was undertaken. Our approach to improving the quality of tourism recommendation favoured the use of ontology, so as to engender the leveraging of deep concrete knowledge of tourism objects for the purpose of generating intelligent and dependable recommendations. The use of ontology was also intended to engender knowledge reuse and semantic interoperability within specific tourism value chains. Hence a suite of relevant ontologies for the tourism domain was developed using the methontology approach (Gomez-Perez et al., 2004). Specifically two Knowledge Representation (KR) ontologies were implemented in the Web Ontology Language (OWL) using the Protégé 3.3.1 Ontology development tool (http://www.3WC.org). These are: 1) The Destination Context Ontology (DCO): in which probable tourist locations were represented using city, town, and village abstractions. The ontology is a semantic model of the social context information of specific destination types and the concrete semantic relationships that exist among them; 2) The Accommodation Ontology (AO): which contained the semantic descriptions of services, facilities, attractions, gastro outfits and location of all category of accommodation types relevant to tourism. This includes hotel categories (1-star, 2-star, 3-star, 4-star, 5-star), guesthouses, and rented apartments.

In domain engineering, the core components of the product line were constructed. Specifically, two tourism recommender systems were constructed, which are notably a Destination Recommender Systems (DRS) component and an Accommodation Recommender Systems (ARS) component. Detailed domain requirements about these two aspects of tourism recommendations were obtained from literature and the prominent existing commercial products, in order to formulate appropriate design for the systems and establish a basis for possible extensions. During domain realization, the recommender system components were implemented as parameterized components that can be instantiated with different tourism information contents and contextual information (as captured in specific knowledge-bases) to suit different tourism promotion scenarios. The recommender components were implemented using a combination of Java Servlet technology, Enterprise JavaBeans (EJB) component technology, and Java Protégé ontology API. The DRS is based on a hybrid architecture that combines content-based and case-based reasoning techniques for its recommendations (Vozalis & Margaritis, 2003; Konstan et al., 2004). The DRS makes use of the destination context ontology and a database of national tourism assets (http://www.nigeriatourism.net) to deliver its recommendations. The ARS does ontological filtering of information contained in the accommodation ontology to generate its recommendations. Other components such as reusable web layout templates and content management components were also constructed during domain realization. Additionally, the constructed core components and particularly the tourism recommender components were tested and evaluated. Empirically evaluation was undertaken using usability metrics to certify their efficiency and fitness for product line compositions. The content evaluation of the two ontologies was also undertaken.

Also, application engineering was undertaken, during which time the tourism recommendation component infrastructures were instantiated with application-specific requirements and used in generating three variant TIS prototypes (for: Nigeria, Ghana and Ivory Coast).
Lastly, an evaluation of the PLONTOREC approach and its products was undertaken in order to access how well it fulfils its set objectives. Thereafter, the result of the evaluation experiments were analysed in order to establish a basis for the generalization of our results. A schematic model of the methodology of this thesis using a UML activity diagram is shown in figure 1.1.



Investigation of the features of next generation TIS
Identifying Research questions / statement of the problem
Proposition of TPLA as a viable solution
Derivation of the PLONTOREC Approach
Product Line Management
Ontology Engineering
Domain Engineering
Application Engineering
Domain Components Design and Construction
Testing of Domain Components
Application Specific Requirements
Products Composition
Ontology Evaluation
Components Evaluation
Products Evaluation
Process Evaluation
Result Analysis
Result Generalization

**Figure 1.1: A model conceptualization of the methodology of this thesis**

## 1.5 SIGNIFICANCE OF THE STUDY

This research work has direct bearing on the tourism industry and particularly e-tourism because of the following:

- The creation of a reusable asset base of intelligent tourism recommendation components offers a response to the quest for reasonably improved intelligence and dependability of recommendations by TIS;
- The study demonstrates the feasibility of PL approach to engineering TIS as none is known to have been reported in literature as yet. Also, PL approach provides a better way to manage the dynamic nature of user requirements in the e-tourism domain; and
- An ontology-based approach will provide a platform for data interoperability, knowledge reuse, and business model standardization within a specific tourism value chain community;
- A novel software research effort in the area of e-tourism will provide the quality boost needed especially in most parts of Africa where tourism is largely undeveloped;
- The study provides a platform for increased publicity and promotion of tourism as a veritable tool for economic development in developing countries.

## 1.6 MOTIVATION FOR THE STUDY

The tourism industry has emerged as a veritable tool for economic advancement in many parts of the world, especially in Europe, the Caribbean Islands and Far East Asia. In Western Europe, tourism is indeed a big industry contributing significantly to the annual GDP (Staab et al., 2002) and is also perceived as a viable tool for continental integration. This further justifies the huge funds being expended by the EU on tourism-based research. However, tourism is still largely undeveloped in most parts of Africa despite the enormous tourism potentials of most African countries. Drawing from the scenario presented above, the motivation for this work is two-fold. The first stems from trends in the global e-tourism domain, while the second is derived from events in the local African tourism context.

In recent years, e-tourism, which entails the promotion of tourism interest through the electronic media, particularly the web, has become a very popular and convenient medium for tourism business transactions. The requirements and expectations of users have become very dynamic and getting increasingly complex due to the advent of new emerging technologies particularly the semantic web. Prospective users of TIS wants more advanced functionalities in tourism support systems that can possibly eliminate the need for human travel agents, in order words there is a quest for smarter and more intelligent systems. One of the areas where improvement is desired is tour recommendations, and travel advisory and planning services, which is still an open issue to which this thesis is making a contribution (Steiner, 2002).

Secondly, there is a dearth of e-tourism research-focused activities geared at advancing the cause of tourism from an African-based context. This thesis seeks to create both a theoretical and product-oriented framework from which credible African-based tourism promotion initiatives can evolve in tandem with state-of-the-art practices in the global e-tourism domain. This desire informed our decision to adopt a software product line-based approach to engineering e-tourism systems in this thesis.

## 1.7 CONTRIBUTIONS TO KNOWLEDGE

The specific contributions of this work apply to the e-tourism domain at the local and global levels.

Firstly, although the demand imposed on TIS by the advent of new technologies and user dynamism has received significant attention in the areas of personalized and context-based services, mobility and embedded intelligence, and semantic interoperability, relatively minimal efforts have been made in other areas (Staab, 2002; Steinbauer, 2005). What is missing in literature is a viable product development model that can cause TIS products to evolve intelligently. The traditional software engineering approaches will not suffice because of the heavy intelligence requirements that must be fulfilled by next generation TIS. Also conventional artificial intelligence approaches cannot achieve much without being complemented by core technologies of the Internet architecture, the web, telecommunication, and software engineering.

The predictable functional characteristics of TIS and their context specific nature make them attractive for a SPLE development approach. The existence of organization such Tiscover AG (http://www.Tiscover.com) and Ectrl Solutions (http://www.etrlsolutions.com) that has successfully deployed TIS in different national contexts allude to the fact that substantial software reuse exist in the e-tourism industry, though the degree or mode of such reuse remain very obscure. To the best of our knowledge the use of SPL as a potential product development approach for solving the problem of dynamic user requirements in e-tourism is being attempted for the first time. Hence, this study presents software product line engineering as a viable solution approach to solving the problem of dynamic user requirements in e-tourism.

Secondly, thus far traditional recommendation formalisms have followed a strictly two-dimensional approach which is based on the user's preferences and the products description. This thesis introduces a third dimension that engages the use of contextual information about the social attributes of destinations as an integral factor in destination recommendation. The value of this approach is that it allows the use of domain specific knowledge relative to specific tourism contexts for destination recommendations which have the potential to improve the dependability and utility of destination recommendation. By doing this, intelligence and semantic web capabilities easily become incorporated into destination recommendation by using the concept of ontology. This is indeed an innovation in destination recommendation relative to existing approaches.

Thirdly, domain specific knowledge where available can be reused and shared and this might enhance interoperability. To the best of our knowledge this work provides within the West African tourism value chain the first suite of tourism ontologies for generating semantic web content.

Lastly and notably, this thesis presents a first attempt to create a product line of recommendation-intensive TIS products. To do this, the Product Line for Ontology-based Tourism Recommendations (PLONTOREC) approach was introduced.  PLONTOREC is a novel software product line approach that enables the creation of dependable and intelligent recommendation-intensive TIS products and also facilitates the proactive evolution of such TIS products in tandem with dynamic user requirements.

The interdependency between the core subject components and the contributions of this thesis are shown in Figure 1.2 using a dependency graph (Ziegler, 2005). The graph consists of six nodes (four subject component nodes and two contribution nodes marked with concentric circles). Edges point from nodes exerting an impact on those they influence. Hence, our approach of building a suite of tourism ontologies, affects the quality of recommendations of the tourism recommender systems. Similarly, the ontologies and tourism recommender systems become valuable reusable assets for the pursuit of a PL initiative using PLONTOREC. PLONTOREC is used to produce variant TIS products, through the customization and integration of existing reusable infrastructures in the asset base such as tourism recommender systems and ontologies. TIS developers that adopt the PLONTOREC approach leveraging the infrastructures available in the asset base can evolve TIS products that can proactively evolve with emerging user needs at minimum time and cost. Also, users can have access to more dependable recommendations from TIS because of improved intelligent behaviour due to ontological enabling of such systems with specific domain knowledge that is based on contextual information about real world scenarios and phenomena.

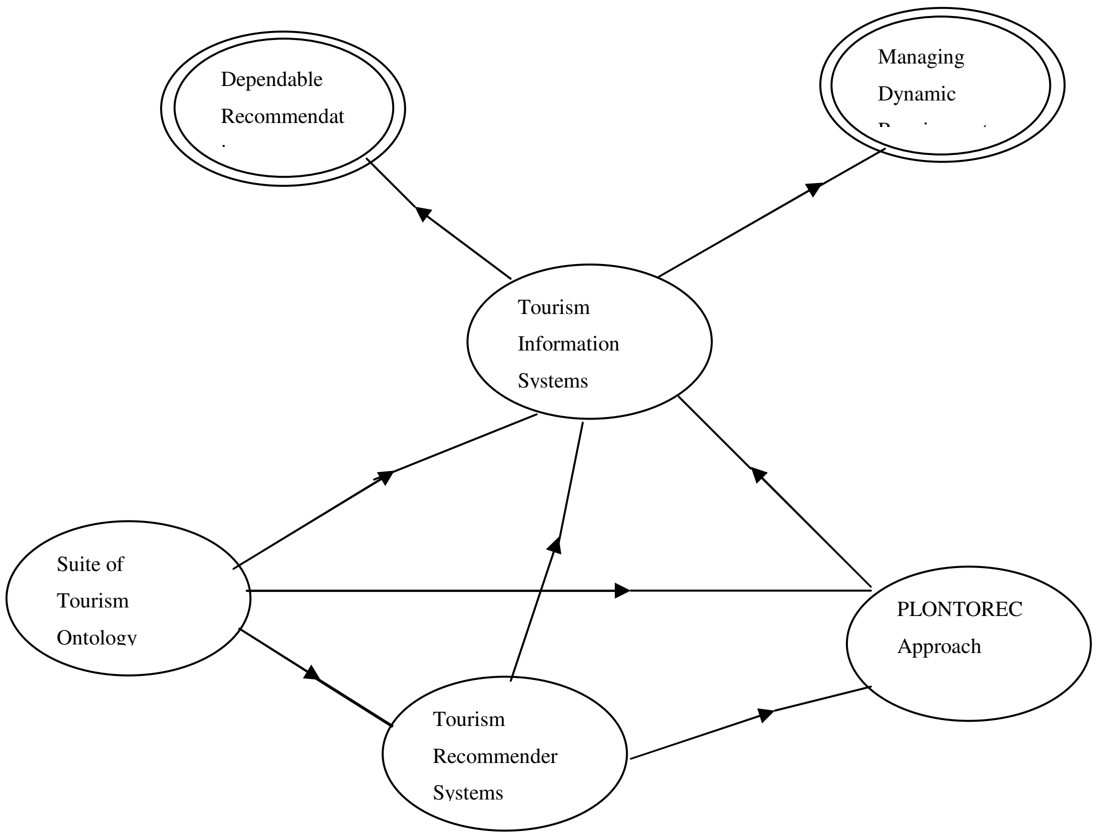


**Figure 1.2: Dependency graph modelling of the components and contributions of this thesis.**

## 1.8 DELIMITATION OF THE SCOPE OF THE STUDY

The main focus of this thesis is to demonstrate the feasibility of a PL approach to engineering TIS and the creation of a platform for intelligent and dependable tourism recommendations. However, although the theoretical concepts canvassed in this study are applicable to both the mobile and the web-based TIS, the prototype implementations in this work are limited to the web-based TIS.

## 1.9 THESIS ORGANIZATION

Chapter One of this thesis presents a general introduction, highlighting the motivation for the research, the aim and objectives of the work and its contributions.

Chapter Two undertakes a critical review of the domain of e-tourism systems and challenges of dynamic user requirements in tourism. The chapter presents a review of related work and defines the context of the research undertaken in this work by indentifying the gaps that exist in literature. The chapter concludes with the proposal of a generic product line architecture for next generation TIS in response to the dynamic challenges of e-tourism systems.

Chapter Three, introduces the Product Line for Ontology-based Tourism Recommendations (PLONTOREC) approach a unified solution platform to solving the research questions raised in this thesis.

Chapter Four presents the details of a case study of product line development that was undertaken to validate the PLONTOREC approach. Specifically, the description of the product line management, ontology engineering, domain engineering, and application engineering activities in PLONTOREC are discussed.

In Chapter Five, the details of the evaluation procedure for the PLONTOREC process and its products are discussed.

Finally, in Chapter Six, we give the summary, conclusion and a discussion of the future research outlook of this thesis.

# CHAPTER TWO

# E-TOURISM SYSTEMS AND DYNAMIC USER REQUIREMENTS

## 2.1 INTRODUCTION

The issue of intelligent-enabling of TIS is one that has continued to attract interest in e-tourism research because of the increasing potentials for new intelligent possibilities as a result of the advent of new technologies, such as the semantic web and mobile computing. This is further amplified by the growing complex nature of user requirements, which is evolving rapidly with new emerging technologies that place a heavy demand on TIS developer organizations. Today, most users will readily jettison the services of human-agents for their travel transactions in preference for a software agent-oriented approach, if the recommendations available on e-tourism platforms become more dependable.

Our approach in this thesis explores the ontological enabling of TIS as a basis for intelligent recommendations in a SPL development context. This is to engender the generation of dependable tourism recommendations and the evolution of relevant TIS products in tandem with dynamic user requirements in the e-tourism domain. In this section, we discuss relevant research issues that define the context of this thesis and the basis for its contributions.

## 2.2 WHY INTELLIGENT SYSTEMS FOR TOURISM?

Tourism is a global information-intensive industry that has a long chain of stakeholders including service providers, marketers, managers, and consumers. The information-driven nature of the tourism industry and the advent of the World Wide Web make e-Tourism the most convenient platform for the advancement of the tourism business industry. This has also brought about the advent of TIS that are offering various tourism support services on the web and mobile computing platform.

However, the active human-machine interaction afforded by the e-tourism platform compels TIS to exhibit increasingly high level of intelligence to earn the required level of acceptability in terms of performance. This has also made the need for improved intelligent behaviour in TIS more compelling. The notion of intelligent TIS is a highly attractive area of e-tourism research, as lots of potentials are not yet fully exploited (Felfernig et al., 2006). A survey of literature reveals some of the areas where a greater measure of intelligence is required or currently non-existent in e-tourism. These include semantic interoperability and medicated architectures; e-business frameworks supporting processes across virtual organizations; mobility and embedded intelligence; natural multi-lingual interfaces; personalization and context-based services; data mining and knowledge management (Staab et al., 2002). Evidence from literature reveals that the most significant research efforts have been in the areas of personalization and context-based services, mobility and embedded intelligence while relatively minimal efforts have been made in other areas.

Some of the research projects on intelligent systems for tourism include AMBIESENSE (Lech et al., 2005), which entails the development of ambient intelligence application systems for mobile users in travel and tourism. AMBIESENSE technology provides travel and tourism support services to mobile users that are ambient, ubiquitous, personalised, and sensitive to individual user's context. The system leverages a system architecture that enables ambient information services to be delivered to mobile users. CAPITALS ITTS (http://cordis.europa.eu/data/PROJ_FP5/ACTIONeqDndSESSIONeq112422005919ndDOCeq405ndTBLeqEN_PROJ.htm) is an acronym for Capitals Providing Integrated Travel and Tourism Services. It was designed as a ubiquitous and intelligent info-mobility and geo-information systems. The CAPITALS ITTS Project provided a platform for Integrated Travel and Tourism Services (ITTS) for users in five EU capitals (Brussels, Berlin, Madrid, Paris and Rome) with well-developed mobility service platforms. CRUMPET (http://www.eml-development.de/english/research/crumpet/index.php) is an acronym for Creation of User-friendly Mobile services Personalised for Tourism. The CRUMPET project was implemented to validate, and trial tourism related value-added services for nomadic users (across mobile and fixed networks) using agent technology. The implementation was based on FIPA-OS: a standards-compliant open source agent framework, extended to support nomadic applications, devices and networks. DIETORECS (Pühretmair et al., 2002) is the project code for Intelligent Recommendation for Tourist Destination Decision Making. It is an Intelligent System for Improved Tourism and Travel Services that implements a recommendation system for destination decision-making. The system is web-based and integrates data managed by two existing tourist portals. DIETORECS provides personalized recommendations based on user profile and contextual information. It is a conversational system adapting the dialogue process as it learns more about the user. CATIS (Pashtan et al., 2004) is a context-aware tourist information system on mobile devices that leverages Web Services and XML technologies for its implementation. The CATIS incorporates a number of context variables relating to mobility, such as time and location, and type of device. The profile of other tourism-centred EU projects can be found in (ftp://ftp.cordis.europa.eu/pub/ist/docs/transport environment/intelligentsystems_for_tourism_en.pdf).

In (Henriksson, 2005) the profile of some ontology-based EU projects that were aimed at enabling semantic web capabilities and semantic interoperability between e-tourism services and resources are given. This includes the following: The HARMONISE project (Dell'Erba et al., 2002), which is a prominent ontology-based solution for the interoperability problems in the European travel and tourism market. The Harmonise project is aimed at providing a knowledge sharing and ontology mediation platform for the diverse e-commerce applications within the European e-tourism market sphere. The ontology used focussed specifically on the events and accommodation sub-domains of tourism. HI-TOUCH (http://icadc.cordis.lu/fepcgi/srchidadb?CALLER=PROJ_IST&
ACTION=D&RCN=63604&DOC=20&QUERY=3) is the acronym for e-organisational metHodology and tools for Intra-European sustainable Tourism. The aim of the Hi-Touch project is to develop software tools to be used by travel agency sales assistants for providing a tourist prospect with the best-adapted offer. The developed tools leverage ontological databases and semantic descriptors, and multi-lingual thesaurus to deliver their functionalities. SATINE (http://www.srdc.metu.edu.tr/webpage/projects/satine/) is an acronym for Semantic-based Interoperability Infrastructure for Integrating Web Service Platforms to Peer-to-Peer Networks. The ongoing project will be used to create a semantic based infrastructure that will enable the

Web Services on well-established service registries like UDDI or ebXML to seamlessly interoperate with Web Services on P2P Networks. Relevant travel ontologies will be developed and the semantics of the Web Services will be based on standard specifications like the one produced by Open Travel Alliance. The semantic infrastructure will be used to develop an innovative business pilot application in the tourism industry. IM@GINE IT (http://dbs.cordis.lu/fepcgi/srchidadb?ACTION=D&
SESSION=296320041126&DOC=53&TBL=EN_PROJ&RCN=EP_RPG:508008&
CALLER=PROJ_IST) is the acronym for Intelligent Mobility AGents, Advanced Positioning and Mapping Technologies INtEgration Interoperable MulTimodal, location based services. The IM@GINE IT project aims to develop one and single access point, through which the end user can obtain location-based, intermodal transport information, mapping and routing, navigation and other related ubiquitous services in Europe, at anytime, and in a personalized way. The technology will rely on a common transport and tourism ontologies for semantic web applications to be developed.

The inference that can be drawn from the survey of intelligent systems in tourism is that the growing complexity of user requirements as a result of the advents of new technologies like the semantic web and mobile technologies has brought about new dynamic challenges in e-tourism. Therefore, providers of e-tourism support services must evolve new approaches for developing intelligent e-tourism systems that can cater for these dynamic challenges.

## 2.3 A CASE FOR RECOMMENDATION-INTENSIVE TIS

The dilemma that a typical user go through during the process of products selection from diverse alternatives in travel and tourism domain could be minimized if many more of the existing e-tourism support platforms are equipped with intelligent recommendation services. The inclusion of tourism recommender systems functionalities in Tourism Information Systems would ensure that users receive intelligent guides when making decisions on important tourism and travel concerns such as accommodation, destinations, and travel plan packages. Current statistics revealed that not many of the existing TIS platforms go beyond just providing tourism related information and booking services (Felfernig et al., 2005). Tiscover AG (http://www.Tiscover.com) and Expedia (http://wwww.expedia.com) are prominent among the relatively few and popular e-tourism platforms where such recommendation services are available. This scenario makes it very compelling for many more of the existing e-tourism supports platforms to provide recommendations on various tourism objects of interest, in order words become more recommendation-intensive. Hence, the challenge of producing recommendation-intensive Tourism Information Systems that is capable of enhancing users' decision-making process and gaining their trust is one to which developers of TIS must respond.

## 2.4 AN OVERVIEW OF RECOMMENDER SYSTEM TYPES AND TECHNIQUES

Recommender systems (RS) are a class of information filtering systems that act as a personalized decision guide for users, aiding them in decision making about matters related to personal taste. RS generally rely on in-built logical reasoning capability or algorithmic computational schemes to deliver their recommendation functionality. RS (Resnick et al., 1997) have found a great deal of significance in a variety of applications. These include music, online communities, web stores and general e-commerce. In most cases, people tend to associate recommender systems with e-commerce sites, where recommender systems are extensively used to suggest products to the customers and to provide customers with information to help them decide which products to purchase.

The two fundamental algorithmic techniques for computing recommendation are *Content-based Filtering (CBF) and Collaborative Filtering (CF).* A CBF system selects items based on the correlation between the content description and the user's preference, while a CF system chooses items by correlating the similarity in the rating of an item by several people. The *hybrid approach* is a third technique that tries to alleviate the limitations of the content-based and collaborative filtering approaches.

### 2.4.1 Content-based Filtering (CBF)

Content-based filtering (CBF) correlates the content description of items with the preferences selected by the user for generating recommendations. It allows automatic categorization and recommendation of information to a user based on the user's personal preferences (Herlocker, 2000). To achieve this, the content descriptions of candidate items are compared with the specified user preferences and the best-matching items are recommended.

The two most prominent content-based filtering techniques were derived from information retrieval and information filtering. The first is by computing nearest-neighbor vector-space similarity between the items vector and vector containing information about the user. An example is Term Frequency Indexing (Salton & Buckley, 1998), which is used in document retrieval, where vectors are used to represent the documents and user preferences. A one-dimensional vector space is used to represent each word in the database with each part of the vector containing the frequency of occurrence of the respective word in the document or the user query. The document vectors that are found to be the closest to the query vectors are considered most relevant to the user's query. The similarity is computed using the cosine similarity metric (Balabanovi´c & Shoham, 1997; Baeza-Yates & Ribeiro-Neto, 1999) based on the Term Frequency/Inverse Document Frequency (TF-IDF) weights obtained. In order words a document $D$ is represented as an $m$ dimensional vector, where each dimension corresponds to a distinct term and $m$ is the total number of terms used in the collection of documents. The document vector is written as $D = (w_1, \ldots, w_m)$, where $w_i$ is the weight of term $t_i$ indicating its importance. If document $D$ does not contain term $t_i$ then weight $w_i$ is zero. Using the TF-IDF scheme the term weights of each $t_i$ can be determined. In this case the weight of a term depends on how often a term appears in a particular document and how frequently it occurs in the entire document collection. This is computed as:

$$w_i = tf_i . \log\left(\frac{n}{df_i}\right) \qquad (2.1)$$

where $tf_i$ is the number of occurrences of term $t_i$ in document $D$, $n$ is the total number of documents in the collection and $df_i$ is the number of documents in which term $t_i$ appears at least once. The assumptions behind TF-IDF are based on two characteristics of text documents. First, the more times a term appears in a document, the more relevant it is to the topic of the document. Second, the more times a term occurs in all documents in the collection, the more poorly it discriminates between documents. Also, user profiles can be represented just like documents by one or more profile vectors. The degree of similarity between a profile vector *P*, where P = ($u_1$,…,$u_k$) and the Document *D* can be determined by using the cosine measure:

$$\cos(D,P) = \frac{D.P}{\|D\|.\|P\|} = \frac{\sum_k u_k.w_k}{\sqrt{\sum_k u_k^2.w_k^2}} \qquad (2.2)$$

Consequently, given a user whose profile indicates a preference for reading software engineering articles, a recommender system using the cosine similarity measure will assign higher similarity score *cos(D.P)* to documents that  that have high-weighted software engineering terms in $w_k$ and lower similarity score to the articles where software engineering terms  are weighted less. Examples of content-based recommender systems that are based on nearest-neighbour vector space techniques include: Fab (Balabanovi´c & Shoham, 1997; Baeza-Yates & Ribeiro-Neto, 1999) and the systems reported in (Alspector et al., 1998; Pazzani, 1999; Ferman et al., 2000; Mukherjee et al., 2001).

Another prominent content-based approach is the use of Bayesian classifiers where recommendation is seen as a classification task.  A Bayesian classifier learns content features to classify unseen items into a positive class $c_1$ (relevant to the user) or a negative class $c_2$ (irrelevant to the user) (Pazzani & Billsus, 1997). Bayesian classifiers use Bayes' theorem of conditional probability:

$$P(Q|M) = \frac{P(M|Q).P(Q)}{P(M)} \qquad (2.3)$$

They also make the naive assumption that product description features are independent, which is usually not the case. For a particular class $Q_i$, the probability of a product $p_k$ belonging to class $Q_i$, given its n feature values $M_1, \ldots, M_n$, is defined as follows:

$$P(Q_i|M_1,\ldots,M_n) = \frac{1}{\Omega}.P(Q_i).\prod_{j=1}^{n} P(M_i|Q_i) \qquad (2.4)$$

Variable Ω represents a scaling factor only dependent on $M_1, \ldots, M_n$. Probabilities P($Q_i$) and P($M_j$ |$Q_i$) can be estimated from training data. Examples of approaches based on Bayesian classifiers include: (Lang, 1995; Lam et al., 1996; Sollenborn & Funk, 2002; Ghani & Fano, 2002; Lam & Riedl, 2004).

Other content-based techniques that have their root in IR systems include Boolean search indexes, where keywords in a query are combined with Boolean operators (Cleverdon, 1967; Herlocker, 2002); and natural language query interfaces, where queries are specified in natural sentences (Lewis & Sparck-Jones, 1996). Examples of content-based filtering recommender systems include: Letizia (Lieberman, 1995), which is a user interface that assists users browsing the web. The system tracks the browsing behaviour of a user and tries to anticipate what pages a particular user may find interesting. Syskill & Webert (Pazzani et al., 1996) is a system that predicts web pages that a user will find interesting based on a user's rating of web pages over time. Higuchi (Jennings & Higuchi, 1992) recommends news items to users using a neural network model.

### 2.4.2 Collaborative Filtering

Collaborative Filtering (CF) uses the ratings of an item by several other users to generate recommendation for a new user after sufficient similarity has been established (Goldberg et al., 1992). Therefore CF uses valuation instead of analysis, by categorizing information based on the user's opinion instead of the information itself. CF algorithms typically operate on a set of users $U = \{u_1, u_2, \dots, u_n\}$, a set of products $P = \{p_1, p_2, \dots, p_m\}$, and partial rating functions $r_i: P \rightarrow [-1, +1]^{\Psi}$ for each user $u_i \in U$. Negative values $r_i(p_k)$ denote dislike, while positive values express $u_i$'s liking of product $p_k$. When $r_i(p_k) = \Psi$ it means that $u_i$ has not rated $p_k$. With this characteristic CF offers some comparative advantages over CBF. First, it is possible to generate recommendations that are independent of the content itself. Second, it is possible to filter and recommend information based on social attributes of the user, such as taste or quality, and thirdly, it is possible to receive useful but unexpected recommendations that are relevant to the user. Lastly, CF helps to create user communities, which is not possible with CBF.

However, CF systems have two drawbacks, first is the fact that recommendations are made to users based on the approximations of other humans, which means that they cannot always be accurate and objective, especially when dealing with non-commodity items, where human preference are very personal (e.g. services, tourism etc.). Another problem is the issue of sparsity, in which calculations are based on sparse and incomplete data, which means the recommendation cannot be trusted because it is based on too few data. These two reasons explain why the recommendations given by CF systems are generally correct, but sometimes very wrong. Due to this fact, CF recommendations are not usually engaged in domains where a higher risk is associated with the acceptance of a recommendation. CF and CBF are combined in many cases into an integrated hybrid filtering solution in order to override the limitations of the individual approaches. Examples of CF implementation projects reported in literature include GroupLens (Resnick et al., 1994; Konstan et al., 1997), Ringo (Shardanand & Maes, 1995), Video Recommender (Hill et al., 1995) and MovieLens (Dahlen et al., 1998). Commercial websites such as: Amazon (www.amazon.com), CDNow (www.cdnow.com), MovieFinder (www.moviefinder.com) and Launch (www.launch.com) make use of collaborative filtering approaches for recommendation.

The two main approaches to achieving collaborative filtering are the memory-based (user-based) collaborative filtering and the model-based (item-based) collaborative filtering.

**i) Memory-based CF approach**

This is also known as the nearest-neighbor approach (Goldberg et al., 1992; Lang, 1995; Pazzani & Billsus, 1997; Linden et al., 2003). It predicts a user's interest in an item based on the ratings for that item by other users who have similar profiles. It is an implementation of the "Word of Mouth" phenomenon, because a database of all known preferences of all users is kept and some computation carried out on the stored users' preferences to generate the prediction.

To implement this, the rating function obtained after a user $d_i$ has rated all items of interest is denoted by the vector $r_i$. Thereafter, the similarities $c(d_i, d_j)$ between all pairs $(d_i, d_j) \in D \times D$ are computed using either the Pearson Correlation Similarity (Konstan et al., 1997) or the Cosine/Vector Similarity metrics (Baeza-Yates & Ribeiro-Neto, 1997), which are the main proximity metrics employed in the recommender systems literature. The cosine similarity measure is widely used in information retrieval to quantify the similarity between two vectors by estimating the cosine of their angles (See equation 2.2 in section 2.3.1).

Pearson correlation is a common statistical correlation coefficients derived from a linear regression model (Konstan et al., 1997), it is similar to cosine similarity, but measures the degree to which a linear relationship exists between two variables. If symbols $\bar{u}_i$, $\bar{u}_j$ denote the averages of vectors $u_i$, $u_j$, then the Pearson correlation metric between $u_i$ and $u_j$ is given as:

$$sim(u_i, \bar{u}_j) = \frac{\sum_{k=0}^{|B|} (u_{i,k} - \bar{u}_i) \cdot (u_{j,k} - \bar{u}_j)}{\sqrt{\sum_{k=0}^{|B|} (u_{i,k}^2 - \bar{u}_i^2)^2 \cdot \sum (u_{j,k} - \bar{u}_j)^2}} \qquad (2.5)$$

After using the cosine similarity measure or Pearson correlation to compute similarities $c(d_i, d_j)$ between all user pairs $(d_i, d_j) \in D \times D$, neighborhoods $prox(d_i)$ of top-M most similar neighbors are built for every peer $d_i \in D$. After this, predictions are computed for all products $b_k$ that $d_i$'s neighbors have rated, but which are yet unknown to $d_i$, i.e, predictions $w_i(b_k)$ for $b_k \in \{ b \in B \mid \exists\, d_j \in prox(d_i) : r_j(b) \neq \Psi$ (no rating) $\}$:

$$w_i(b_k) = \bar{r}_i + \frac{\sum_{d_j \in prox(d_i)} (r_j(b_k) - r_j) \cdot c(d_i, d_j)}{\sum_{d_j \in prox(d_i)} c(d_i, d_j)} \qquad (2.6)$$

Predictions are thus based upon weighted averages of deviations from $d_i$'s neighbors' means. For top-N recommendations, a recommendation list of items $Pw_i:\{1, 2, \ldots, N\} \rightarrow B$ is computed, based upon predictions $w_i$. $P_{wi}$ is a ranked list of recommendations in descending order, giving highest predictions first. Examples of memory-based CF systems include: The Tapestry system (Goldberg et al., 1992), GroupLens (Konstan et al., 1997) and Ringo (Shardanand & Maes, 1995). Some of the shortcomings of memory-based CF are (Sarwar et al., 2001; Hofmann, 2004):

- **Sparsity:** This is a scenario where the active users have purchased or rated very limited products out of the available total. This leads to the problem of insufficient ratings for such items i.e. sparse user-item matrices, inability to locate sufficiently close neighbors and ultimately weak recommendations.
- **Scalability:** The nature of a user-based approach to CF is such that the number of users and items will grow over time, which is bound to increase the complexity of computation. Because of this, a typical memory-based CF system with millions of users and items will

suffer from serious scalability problems as the number of user and items continue to grow.

- **Learning:** The memory-based CF is not based on any explicit statistical model, and as such nothing is learned about users or items that can provide a basis for generalization for future predictions.

**ii) Model-based collaborative filtering**

The shortcomings of memory-based CF systems, especially the lack of scalability and learning have led to the emergence of the concept of model-based CF approach (Sarwar et al., 2001; Karypis, 2001; Deshpande & Karypis, 2004) Model-based CF has the advantage of improved computational complexity characteristics and the ability to separate the model building process from actual computation of recommendation. Particularly, in instances where there are far greater number of users than products i.e. $|U| >> |P|$, the model-based CF has been shown to have better computational performance compared to user-based CF (Sarwar et al., 2001). Just like the memory-based CF, recommendation computation is based on the ratings $r_i(p_k)$ that users $u_i \in U$ provide for products $p_k \in P$, but, unlike memory-based CF, similarity values s*im* are computed for products rather than users, hence sim: $P \times P \rightarrow [-1, +1]$. In this wise, two products $p_k$ and $p_n$ are considered similar, i.e., have large sim $(p_k, p_n)$, if they get identical ratings from many users or user who rate one of them tend to also rate the other. This is followed by the computation of the neighbourhood of $p_k$ using the Cosine similarity metrics or the Pearson correlation similarity metrics i.e. $\text{prox}(p_k) \subseteq P$ of top-M for each $p_k$. Predictions $w_i(p_k)$ are computed as follows:

$$w_i(p_k) = \frac{\sum_{p_n \in P_k^{'}} \left( sim(p_k, p_n) \cdot r_i(p_n) \right)}{\sum_{p_n \in P_k^{'}} \left| sim(p_k, p_n) \right|} \qquad (2.7)$$

where

$$P'_k := \{p_n \in P \mid p_n \in \text{prox}(p_k) \wedge r_i(p_n) \neq \Psi\}$$

This approach emulates the real-life behaviour of users, whereby a user $u_i$ judges the value of an unknown product $p_k$ by comparing $p_k$ to known, similar items $p_n$ and considering how much $u_i$ appreciated items $p_n$. Finally, a top-N recommendation list $L_{wi}$ is generated by arranging recommendations according to $w_i$ in descending order. Typical examples of commercial systems that are based on model-based CF are: success Amazon.com (Linden et al., 2003) and TiVO (Ali, K. & van Stam, 2004).

### 2.4.3 Hybrid Approach

The hybrid approach is a combination of CBF and CF techniques in order to eliminate certain limitations of both techniques (Adomavicius & Tuzhilin, 2005). There are four main approaches for combining the two techniques into a hybrid recommender system. These are:

- **Combining separate recommender systems**
  In this approach predictions from separate implementations of content-based and collaborative techniques are combined within a single system framework (Pazzani, 1999; Claypool et al., 1999). To give a final result, the ratings obtained from the individual recommender systems are combined into a final recommendation, or the best recommendation chosen after a quality assessment of recommendations from both systems have been carried out.
- **Adding content-based characteristics to the collaborative approach**
  In this approach some content-based characteristics are integrated into the collaborative approach. Content attributes and not the commonly rated items are used to calculate the similarity between two users. This innovation helps to overcome some of the sparsity-related problems of a purely collaborative approach, since in most cases it is not common for two users to have a significant number of commonly rated items between them (Pazzani, 1999). Another benefit is that accurate recommendation can be obtained directly when the content attributes of an item match the user's profile and not until when an item gets rated by a similar user.
- **Adding collaborative characteristics to the content-based approach**
  In this approach some collaborative characteristics are integrated into the content-based approach. One way to achieve this is to create a collaborative view of a collection of user profiles represented by term vectors (Soboroff & Nicholas, 1999). This will result in performance improvement when compared to a purely content-based approach.

- **Developing a single unifying recommendation approach**
  In this approach a general framework that incorporates both content-based and collaborative characteristics is created. For example in (Basu et al., 1998), the use of content-based and collaborative characteristics was proposed, such as the age or gender of users or the genre of movies, in a single rule-based recommendation classifier.

### 2.4.4 Knowledge-based Recommender Systems

Knowledge-based recommenders, though sometimes regarded as fundamentally content-based systems are a class of recommender systems that exploit deep knowledge about the product domain in order to determine recommendations. They make use of knowledge about users and products to generate a recommendation and reasoning about what products meet the user's requirements. A knowledge-based recommender system avoids the problem of sparsity associated with both CBF and CF systems (Pazzani, 1999). The recommendations of knowledge-based recommender systems do not depend on a base of user ratings. It does not have to gather information about a particular user because its judgements are independent of individuals' tastes. These characteristics make knowledge-based recommenders very valuable systems when used independently and also when used to complement other types of recommender systems. Examples of knowledge-based recommender systems include: The PersonalLogic recommender system that offers a dialog that effectively walks the user down a discrimination tree of product features (Bhargava et al., 1999). The restaurant recommender entree (Burke et al., 1996; Burke et al., 1997) makes its recommendations by finding restaurants in a new city similar to restaurants the user knows and likes. The system allows users to navigate by stating their preferences with respect to a given restaurant, thereby refining their search criteria. Other implementations of knowledge-based recommender systems are discussed in (Burke, 2000; Thompson et al., 2004; Herlocker et al., 2004; Felfernig & Kiener, 2005; Jiang et al., 2005). However there two major drawbacks of knowledge-based recommender systems, which are the expensive nature of knowledge engineering endeavours which makes them more costly to implement, and the static nature of their suggestions ability (Burke, 2000).

### 2.4.5 Evaluating Recommender Systems

An evaluation of recommender systems is crucial in order to access the quality of recommendations made by them. Recommender Systems evaluation methods can be broadly classified as accuracy metrics and non-accuracy metrics. The different evaluation schemes that have found relevance in these two categories are discussed as follows:

#### 2.4.5.1 Accuracy Metrics

Accuracy metrics can be classified as those designed to assess the accuracy of single product predictions and those that are meant for decision-support in order to evaluate the effectiveness of the system in helping users to distinguish between high-quality items and the rest of the product items. These metrics operate on the assumption that a binary rating scheme is preferred in rating products (Ziegler, 20005). Accuracy metrics are classified as:

**i) Predictive Accuracy Metric**

Predictive accuracy metrics measures the closeness of the predicted ratings of a product by a system to true user ratings. In order words, how much predictions $w_i(p_k)$ for products $p_k$ deviate from user $d_i$'s actual ratings $r_i(p_k)$. The most prominent and widely used is the mean absolute error (MAE) (Shardanand & Maes, 1995; Herlocker et al., 2004). MAE is an efficient metric for the statistical accuracy of predictions $w_i(p_k)$ for sets $P_i$ of products:

$$|\overline{E}| = \frac{\sum_{p_k \in P_i} |r_i(p_k) - w_i(p_k)|}{|P_i|} \qquad (2.8)$$

Another metric closely associated to MAE, is the mean squared error (MSE), which squares the error before summing. So that large errors become much more pronounced than small ones. Although MAE and MSE are very efficient for predicting recommendations, they are not suitable for evaluating the quality of top-N recommendations (Herlocker et al., 2004).

**ii) Decision-Support Metrics**

The adjusted concepts of Precision and Recall, borrowed from information retrieval, are used to assess how relevant a set of ranked recommendations is for the active user in making decision.

**Recall, Precision, FI Measures**

In RS the goal is to retrieve a fixed number of N relevant items to be suggested as part of a list. To compute recall and precision, first the data is divided into two disjoint sets, the training set and the test set. Then the filtering algorithm employed by the system is made to work only on the training set to generate a ranked list of recommended items (say the top-N set). Thereafter, the test set which represents the portion of the initial data set that was not used by the recommender system is now used with the algorithm to generate recommendations. The two recommendations are then compared to find items in the test set that are also included in the generated top-N set. The set of items that appear in both sets will become members of a special set, called the hit set. Therefore, recall and precision for top-N recommendation systems can now be defined as follows:

• Recall is the ratio of hit set size over the test set size:

$$recall = \frac{size\,of\ hit\,set}{size\,of\ test\,set} = \frac{|test \cap top-N|}{|test|} \qquad (2.9)$$

• Precision is the ratio of hit set size over the top-N set size:

$$precision = \frac{size\,of\ hit\,set}{size\,of\ top-N\,set} = \frac{|test \cap top-N|}{N} \qquad (2.10)$$

The denominator in equation 2.10 becomes N because the size of the top-N set is N. One peculiarity of the recall-precision metric is that increasing the size of number N usually results in an increase of recall, while at the same time precision is decreased. But since both measures are important in evaluating the quality of systems that generate top-N recommendations, the two can be combined into a single metric, called the F1 metric.

The standard F1 metric is a widely used metric in information retrieval and recommender systems research (Sarwar et al., 2001; Herlocker et al., 2004), which assigns equal weight to both recall and precision:

$$F1 = \frac{2 * recall * precision}{recall + precision} \qquad (2.11)$$

To use this metric, F1 is computed for each individual user and then, the average over all users are computed to represent the score of the top-N recommendation list (Sarwar et al., 2000; Sarwar et al., 2001).

**Breese Score**

The Breese score (also known as weighted recall) is an extension to the recall metric proposed by Breese et al. (1998). The concept is based on the understanding that the expected utility of a recommendation list equates to the probability of viewing a recommended product contained in that top-N list multiplied by its utility, which is either 0 or 1 for binary ratings. Breese score, further assumes that each successive item in a list is less likely to be viewed by the active user with exponential decay. To determine the expected utility of a ranked list $Pw_i$, the test set and training set is first obtained just as in the case of the ordinary recall, then expected utility of $Pw_i$ is computed as:

$$U(P_{w_i}, Q_i) = \sum_{k=1}^{|Q_i|} \frac{1}{2^{(k-1)/(\alpha-1)}} \qquad (2.12)$$

where $Q_i = \{q_1, q_2, .. q_f\}$ is the test set, while parameter α denotes the viewing half-life. Half-life is the number of the product on the list such that there is a 50% chance that the user, represented by training set $R_i$, will review that product. Finally, the weighted recall of $P^x_i$ with respect to $Q_i$ is defined as follows:

$$BScore(P_i^x, Q_i) = \frac{100 \cdot U(P_{w_i}, Q_i)}{\sum_{k=1}^{|Q_i|} \frac{1}{2^{(k-1)/(\alpha-1)}}} \qquad (2.13)$$

Breese score is identical to unweighted recall when the assumption $\alpha = \infty$ is made. The Receiver Operating Characteristic (ROC) is another form of decision-support metric ROC (Good et al., 1999; Schein et al., 2002; Melville et al., 2002). It measures the extent to which an information filtering system is able to successfully distinguish between signal and noise. The NDPM (Balabanovi´c & Shoham, 1997), which compares two different, weakly ordered rankings, is another decision-support metric that is less frequently used.

### 2.4.5.2 Non-Accuracy Metrics

Non-accuracy metrics are intended to assess other aspects of a recommender system that the accuracy metrics are unable to capture. Examples include the usability of the system and satisfaction, which are different from the correctness, or usefulness of recommendation. So far, non-accuracy metrics have been used as important supplements for accuracy metrics. The various forms of non-accuracy metrics include:

**i) Coverage**

Coverage is the most widely used non-accuracy evaluation metrics (Good et al., 1999; Herlocker et al., 1999; Middleton et al., 2004). Coverage measures the percentage of elements part of the problem domain for which predictions can be made. For instance, in the memory-based (user-based) collaborative filtering approach, the coverage for the entire set of users is computed as follows:

$$Coverage = \frac{100 \cdot \sum_{d_i \in D} \left| \{ p \in P \,|\, \exists d_j \in prox(d_j) : r_j(p) \neq \psi(no\,rating) \} \right|}{|P| \cdot |D|} \qquad (2.14)$$

**ii) Novelty and Serendipity**

The novelty and serendipity metrics measure the non-obviousness of recommendations made by a recommender system. There are instances when recommendations are accurate but useless in practice; an example is for a system to suggest bananas to customers in a grocery store. Although this is accurate, but it is still useless because people do not require a recommendation to purchase bananas, since it is a very popular product, most people will likely buy bananas without a recommendation (Terveen & Hill, 2001). However the recommendation of a product in the same store that is not ordinarily desired by a user but relevant is a good mark of novelty and serendipity attribute of the recommender system concerned (Herlocker et al., 2004).

### 2.4.6 Improving Recommender Systems

Although significant advancements have been made in the development of recommender systems, there yet exist the need to improve on the capabilities of existing recommendation techniques and technologies. Some of the desired improvements are discussed as follows:

**Non-intrusiveness**

Intrusiveness is a measure that defines the degree of user's intervention that is required for a recommender system to generate accurate recommendations. Most recommender systems are intrusive, requiring a great deal of user involvement before recommendations can be constructed. It is often time consuming when users have to explicitly indicate their preferences for specific items using a binary or numerical scale. The use of a binary scale only helps the user to indicate whether an item is liked or disliked while using a numerical scale, helps the user to express in more detail, the degree of preference for an item. For example, Syskill & Webert (Pazzani et al., 1996) is an intrusive system that uses binary rating to capture a user's impression about a website visited (either liked or disliked), while the GroupLens system (Konstan et al., 1997) is an intrusive system that allows users to rate Netnews articles on a numerical scale of one (bad) to five (good) after reading it. Nonintrusive systems use implicit approach to limit the extent of user involvement in capturing the rating of items. To achieve this, a nonintrusive system interprets user behaviour or selections gathered over time. This could be when browsing data in web applications, purchase history in web stores, or other types of information access patterns. However the drawback of nonintrusive ratings is that they are often inaccurate and cannot fully replace explicit ratings provided by the user. Therefore, there remains the need to evolve recommendation formalisms that will minimize intrusiveness while maintaining a satisfactory level of accuracy.

**Contextual Information**

Recommender systems need to give accurate recommendations in order to foster users' confidence in them, which will also increase their utility. The current generation of recommender systems operates in two-dimensional *User* x *Item* space. They focus only on user and item information to generate recommendations and do not consider the use of additional contextual information, which may be crucial in some applications (Adomavicius & Tuzhilin, 2005). This two-dimensional approach is also at variance with reality because in many cases, the items preferred by a user may change depending on the context; therefore conventional systems have inherent problems. Context is any information that can be used to characterize the situation of an entity. An entity can be a person, place or object that is considered relevant to the interaction between a user and an application, or relevant to both the user and applications themselves (Dey, 2001). In reality the utility of certain recommended item depends on time and/or location. It also depends on the person with whom the recommended item will be shared, and under what circumstances. For example, a travel recommender system should not only recommend some vacation spots based on what a user and other similar users liked in the past. It should also consider the time of the year, persons the user is traveling with, and other relevant contextual information (social, environmental, political etc.). To introduce the use of contextual information in recommender systems, the content to be recommended needs some meta-data attached to it, which should be a formal description of the different contexts in machine-readable form. Formal meta-data models such as formal logics, ontology, and knowledge bases come to mind in this respect to improve the accuracy and dependability of recommendations in recommender systems (Park et al., 2006). This is one aspect of contribution that is explored in this thesis.

**Comprehensive Understanding of Users and Items**

In most of the existing recommendation methods only limited knowledge of the user and item is exploited in generating recommendations. The systems do not take full advantage of the information in the user's transactional histories and other available data. For example, classical collaborative filtering methods do not use user and item profiles at all for recommendation purposes and rely exclusively on the ratings information to make recommendations (Adomavicius & Tuzhilin, 2005). Although some attempts have been made to incorporate user and item profiles into the implementation of some recommender systems (Pazzani, 1999; Billsus & Pazzani, 2000; Pennock & Horvitz, 2000), these profiles still tend to be quite simple and do not utilize some of the more advanced profiling techniques. In addition to using traditional profile features, such as keywords and simple user demographics (Billsus & Pazzani, 2000; Mooney & Roy, 2000), more advanced profiling techniques based on data mining rules (Fawcett & Provost, 1996; Adomavicius & Tuzhilin, 2001), sequences (Mannila et al., 1995), and signatures (Cortes et al., 2000) that describe a user's interests can be used to build user profiles. Also, in addition to using the traditional item profile features, such as keywords (Pazzani, 1999, Bhargava et al., 1999), similar advanced profiling techniques such as data mining can also be used to build comprehensive item profiles. Once user and item profiles are built, the most general ratings estimation function can be defined in terms of these profiles that will improve the accuracy of recommendations (Adomavicius & Tuzhilin, 2005).

## Evaluating Recommender Systems

The most commonly used metrics to measure the effectiveness of recommendations (Herlocker et al., 2004) are the *coverage* and *accuracy* metrics. Coverage metrics is used to determine the percentage of items for which a recommender system is capable of making predictions. For accuracy *statistical* or *decision-support* measures (Herlocker et al., 2004) are used to evaluate recommender systems. Statistical accuracy metrics uses techniques such as mean absolute error (MAE), root mean squared error and correlation between predictions and ratings to compare the estimated ratings against the actual ratings. While decision-support measures determine how well a recommender system can make predictions of items that would be highly relevant to the user. The most dominant approach to do this is the use of the *precision* and *recall* metrics. Precision is the measure of truly high ratings among those that were predicted to be high by the recommender system, while recall is the measure of correctly predicted high ratings among all the ratings known to be high.

Despite the popularity of these measures, they have certain limitations. First, is the fact that they can only provide an evaluation of the system on the items that have been rated by the users and not item that are not rated. This gives a false impression of preferences because users tend to rate the items they like, not the items that they dislike. Thus, evaluation results only show how accurate the system is on items that have been rated by users, other than the general ability of the system to properly evaluate an item.

Secondly, the accuracy and coverage metrics do not capture the "quality" and "usefulness" of recommendations. Imagine a recommender system for a supermarket. Recommending obvious items such as milk and bread that the users are likely to buy, will give high accuracy rates. However, it will not be very useful for the customer. It is therefore important to develop measures that also capture the business value of recommendations such as return on investments (ROI) and customer lifetime value (LTV) measures (Schmittlein et al., 1987; Dwyer, 1989; Rosset et al., 2002).

**Other improvements**

Other research issues within recommender systems include multicriteria rating (Statnikov & Matusov, 1995; Ehrgott, 2000), scalability (Sarwar et al., 2001; Schafer et al., 2001), explainability (Herlocker et al., 2000), trustworthiness (Dellarocas, 2003) and privacy (Herlocker et al., 2000).

## 2.5 RECOMMENDATION TECHNOLOGIES IN TOURISM

Tourism Recommender Systems (TRS) are the class of intelligent systems that render tourism-related information services in the form of guides and suggestions to users. This class of systems can be broadly classified as web-based tourism recommender systems and mobile recommender systems. An overview of the existing mobile and web-based tourism recommender technologies is given in the sequel sections.

### 2.5.1 Mobile Tourism Recommender Systems

Mobile Tourism Recommender Systems (MTRS) are intelligent systems that deliver valuable tourism contents and information to users' mobile phones or PDAs. Thus far in the MTRS arena, concentration had been on the delivery of personalized context-aware information notification services for users in ubiquitous fashion. For example Cyberguide (Abowd et al., 1997) is a mobile guide system that displays point of interests (POIs) on an interactive map. The development of an electronic tour guide for the city of Lancaster was described in (Cheverst et al., 2000). COMPASS (van Setten et al., 2004) is a system that provides context-aware route guide recommendations which was implemented in the Netherlands. MobiDenk (Krösche et al., 2004) is a multimedia-enriched location-aware information system for the conservation of historic sites. Berlintainment (Wohltorf et al., 2005) offers information guide on the location of entertainment tourism resources in the city of Berlin. The etPlanner system, which is currently being developed by Austrian Network for E-Tourism (ANET), is a MTRS that is designed to render relatively more substantial recommendations. It targets widespread use among tourists by eliminating the need for client-side installation requirements. It allows two types of communication with its users, which is one of its novelties. First, information seekers have personalized browsing access to categories like events, sights, restaurants or accommodations. In a second step, users may also receive personalized push messages that inform them about changing weather conditions if they are out hiking or make them propositions on leisure activities based on their preferences. A first version of the system has already been deployed for public use and there are plans to extend the scope of its recommendations (Felfernig et al., 2005). Other examples of MTRS initiatives include CATIS (Pashtan et al., 2004), CRUMPET (http://www.eml-development.de/english/research/crumpet/index.php), and AMBIESENSE (Lech et al., 2005), which have been described in section 2.2.

### 2.5.2 Web-based Tourism Recommender Systems

Web-based Tourism Recommender Systems (WTRS) are intelligent systems that are usually embedded in e-Tourism portals (i.e. TIS) in order to deliver travel information guide, travel advice and travel planning recommendations. A survey of most of the existing web sites revealed that very few go beyond pure booking system functionalities to providing intelligent recommendations (Felfernig et al., 2005). In the travel and tourism domain, the two most successful recommender system technologies are TripMatcher (used by www.ski-europe.com, etc.) from Triplehop, and Me-Print (used by travelocity.com), which is an expert advice platform from VacationCoach (Staab et al., 2002). The implementation of these two recommender systems emulates the interaction of a travel agents and a user in which the user inquires on a possible holiday destination. They largely use a content-based approach for generating recommendations, as they allow the capturing of user's travel preferences and constraints before constructing intelligent recommendations of a list of possible destinations. Me Print from VacationCoach exploits user profiling by asking the user to identify with one of the available specific travel activity classes (for example, as a "culture creature," "beach bum," or "trail trekker" etc.) in order to induce implicit needs that the user does not provide. The user can also input precise profile information by completing the appropriate form. In TripMatcher a more sophisticated approach is used to reduce user input. The system guesses the importance of attributes that the user does not explicitly mention. It then combines statistics on past user queries with a prediction computed as a weighted average of importance assigned by similar users (Staab et al., 2002).

Another successful travel and tourism recommendation technology though relatively new is the trip@dvice (http://www.nutking.ectrldev.com/nutking/), which has been applied in some e-tourism portals (e.g. visiteurope.com) (Venturini & Ricci, 2006). It is the product of the 5th EU Framework Programme project 'DIETORECS' (Pühretmair et al., 2002) which is now being managed by ECTRL Solutions (http://www.ectrlsolutions.com). Trip@dvice predominantly uses Case-Based Reasoning (CBR) as its recommendation technology. CBR is a problem-solving paradigm that is based on solving new problems based on past experiences, premised on the belief that similar problems have similar solutions. At the instance of a new problem, a past, already solved similar case is retrieved, and then used to solve the current one (Vozalis & Margaritis, 2003). In Trip@dvice every completed travel plan is stored in the Case Base as an instance of good example, so that during a new user's travel recommendation session, the system retrieves cases similar to the one under construction. The similarity function uses the current information and historic users' profiles and travel characteristics to generate highly personalized results. Additionally, it uses a ranking technology that sorts suitable items from a catalogue and presents candidate trips by using the user input and the satisfaction of other users on similar trips. This ranking is applicable to complete travels packages as well as for single travel products such as destination, accommodation, and services (http://www.ectrlsolutions.com).

## 2.6 LIMITATIONS OF EXISTING APPROACHES

A study of existing tourism recommendation technologies conducted in the course of this research work reveals some limitations of existing approaches. Firstly, elaborate recommendation functionalities have not been implemented on the mobile platform. This limitation derives from the limited computational processing capacity of mobile devices and the smallness of the screen size (Goren-Bar, 2004). On the web platform, Me-Print and TripMatcher technologies are content-based approaches that leverage knowledge to deliver recommendations. However, the two platforms are limited to destination recommendations. Recommendations on other forms of tourism objects such as accommodation, cruises, services etc. were not covered

(Staab et al., 2002). Another successful recommendation technology is the trip@dvice (see http://www.nutking.ectrldev.com/nutking/), which has been applied in some e-tourism portals (e.g. visiteurope.com) (Venturini & Ricci, 2006). Trip@dvice predominantly uses case-based reasoning as its recommendation technology but unlike TripMatcher and Me Print offers a range of recommendation services on several tourism objects. One characteristic common to all of these implementations is the fact that the parameters used for destination recommendation were strictly two-dimensional (i.e. the user's travel preferences and the description catalog of travel destinations). This imposes a limitation of the quality of recommendations because it fails to capture other important dimensions that are crucial to the provision of credible recommendation.

Specifically, the use of relevant contextual information that can improve the quality and dependability of recommendations was not considered (Adomavicius & Tuzhilin, 2005; Adomavicius, 2005). For example the use of the social and environmental attributes information of destinations as an additional factor for destination recommendations have the potential to improve the dependability of generated recommendations. This is because inclusion of such important contextual information about a place to visit would ensure that recommendations are based on deeper knowledge of the destination domain in a way that closely model reality. The dependability of tourism information is most important because the tourism product by its nature is intangible, one that the traveler cannot touch before the trip. This is one major reason why recommendations on destination, accommodation, and other travel services must be accurate and credible, one that fosters a user's confidence, an attribute that existing tourism recommendation formalisms do not yet possess.

Hence, recommendation formalisms that exploit deep knowledge of both the user and the tourism object, and other relevant contextual information in a way that closely model reality is required in order to improve the dependability of existing approaches. This is one of the cardinal objectives of this research work and to which this thesis makes a contribution.

## 2.7 ONTOLOGY-BASED TOURISM RECOMMENDATIONS

This thesis posits that one way to achieve dependable tourism recommendations is to engage knowledge representation formalisms that can sufficiently capture all relevant facts about tourism objects in a domain on which approaches to rendering tourism information services can be based. An ideal approach to achieve this is the use of ontologies which provide the platform on which recommendation formalisms that exploit deep knowledge of the user, tourism objects, and other relevant contextual information can be built.

The use of ontologies has the potential to solve a number of problems in tourism. First, the fact that it allows the sharing of domain knowledge using a common vocabulary across heterogeneous platforms means it can be used to solve interoperability problems (Dell'Erba et al., 2002). Secondly, ontology enables the sharing of common understanding of the structure of information among people and software agents (Noy & McGuinness, 2003); this also can help to standardize business models, business processes and knowledge architectures in tourism. Thirdly, ontology serves as a model of knowledge representation from which knowledge bases that describes specific situations can be built. These reasons motivated our notion of ontology-enabled TIS. This is premised on the belief that an ontology-based framework that enables the leveraging of factual knowledge about a specific tourism context for recommendations has potentially high tendency to enhance the quality and credibility of tourism recommendation services for such a context.

### 2.7.1 What is Ontology?

The word 'ontology' was originally taken from the field of philosophy and is concerned with the study of the nature of being. In the context of Artificial Intelligence (AI), there exist sundry definitions of ontology (Gomez-Perez et al., 2004; Noy & Hafner, 1997; Noy & McGuinness, 2003), each one trying to introduce its own emphasis. However, one of the most common definitions of ontology in literature is that: *'An ontology is a formal explicit specification of a shared conceptualisation of a domain'*. *Conceptualisation* entails the use of abstract models to

depict what is understood about entities in a domain of interest. *Explicit* means that the concepts used and the constraints on them are clearly defined while *formal* means that entities in the ontology are represented in full or semi-machine processable form. Also, the fact that it is shared means that the knowledge captured in the ontology is mutually agreeable to a group of people.
In this thesis, we define ontology as: *A formal model of a domain of interest that depicts the domain as an aggregation of its known relevant elemental concepts and the semantic relationships between them that provides a platform for knowledge sharing and reuse*. This connotes that an ontology is a deliberate semantic description of what is generally known about some real world phenomena in a domain of interest using concepts and relationship abstractions in a way that is readable by both man and machine.

### 2.7.2 The Components of an Ontology

An ontology essentially consists of a vocabulary of terms in a domain of interest and their meanings. This includes definition of concepts, the properties of the concepts, and the interrelationship between concepts. Ontologies are classified into lightweight and heavyweight categories based on the nature of their composition (Gomez-Perez et al., 2004). The main components of a lightweight ontology are the concepts of a domain, the interrelationship between concepts, and the properties of each concept. However heavyweight ontologies consist of concepts, concept properties and concepts interrelationships just like lightweight ontologies but have also included in their definition the axioms and restrictions on concepts, concepts properties and concepts relationships. Generally, the notion of concepts (sometimes called *classes*) in ontology is akin to classes in the object-oriented paradigm. The properties of a concept (sometimes called *slots*) are the features and attributes of that concept which can take specific value types (e.g. boolean, integer, string, float, date, etc.). The restrictions are formal logics constraints that are defined on the properties of a concept or on inter-concepts relationships.

The taxonomic relationships between classes in an ontology are mostly defined through inheritance using *'ISA'* relationships which specifies a subclass A as *'a kind of'* the superclass B. For example, if the class *Location* defines all kinds of places where people live, then all addresses will be an instance of class *Location*. However, *City, Town*, and *Village* are different kinds of location where people live, each of which can be represented as a subclass of the *Location*. Other kinds of relationship like ‘part-whole’ (“PartOf") or synonym ("SynOf"). Additionally, other application specific relationships that might exist can be represented in an ontology (Necib & Freytag, 2005).

In practical terms, developing an ontology is no more than 1) defining classes in the ontology; 2) arranging the classes in a taxonomic (subclass–superclass) hierarchy; 3) defining properties of classes and describing allowed values for these properties; and 4) supplying the values for the properties for the instances. Therefore it is possible to create a knowledge base by defining individual instances of these classes, filling in specific property value information and additional property restrictions.

### 2.7.3 Ontology Development Process

An ontology can be built either from scratch, through the re-engineering of other existing ontologies or by a process of ontology merging or ontology learning approach. In 1997, a proposal that emulates the IEEE standards for software development was formulated as the ontology development process, which was based on the framework of the METHONTOLOGY methodology for ontology construction (Gomez-Perez et al., 2004). The ontology development process refers to the set of activities involved in building ontologies. These activities have been categorized into three, which according to (Gomez-Perez et al., 2004) are:

i) **Ontology Management Activities**: The sub-activities that fall into this category include scheduling, control and quality assurance activities.

- Scheduling activity: This entails the identification of the tasks to be performed, the arrangement of the tasks and the time and resources that are needed for the completion of the tasks. Scheduling is particularly important for ontologies that reference ontologies stored in ontology library or for ontologies that require a high level of abstraction or generality.

- Control activity: This moderates the entire development process to ensure that scheduled tasks are executed as planned.
- Quality assurance activity: This is designed to ensure that the quality of every product of the ontology development process (ontology, software and documentation) is satisfactory.

ii) **Ontology Development Activities**: The sub-activities that fall into this category are pre-development, development and post-development activities.

- Pre-development: This encapsulates the processes of environmental study and feasibility study. The environmental study is used to identify where the ontology will be used and the types of applications that will be integrated with the ontology. During feasibility study, the possibility and the suitability of building the ontology is critically examined prior to further investment of time and resources.
- Development Phase: This consists of the specification, conceptualisation, formalization and implementation activities. During the specification activity, the reason for building the ontology, its intended uses and prospective end-users of the ontology are stated. During the conceptualisation activity, abstraction models are used to represent knowledge of the domain in a meaningful way. The formalization activity entails the transformation of the conceptual model into a formal or semi-formal model that is machine-readable. The implementation involves building the formal model in a particular ontology language.
- Post Development: This consists of the maintenance, usage and reuse phases of the ontology development. The maintenance activity involves updating and making corrections to the ontology after it has been built. The use and reuse of the ontology by other ontologies or applications is also considered as activities of post development.

iii) **Ontology Support Activities**: These are the set of activities that are carried out simultaneously with development activities of the ontology development process without which building the ontology would be impossible. Its sub-activities include knowledge acquisition, evaluation, integration, merging, alignment, documentation, and configuration management.

- Knowledge acquisition activity: This entails sourcing for domain knowledge to be stored in the ontology from domain experts or through ontology learning (Kietz et al., 2000).

- Ontology Evaluation activity: This involves a procedure for the technical assessment of various aspects of the ontology such as its components, software environment, and its documentation (Gomez-Perez et al., 1995).
- Ontology Integration activity: This involves the establishment of relevant mappings between terms of different ontologies. Integration activity is a necessity when existing ontologies are being reused in the development of a new one.
- Ontology Merging activity (Gangemi et al., 1999; Noy & Musen, 2001; Stumme & Maedche, 2001): Ontological merging entails bringing together several ontologies in the same domain to create a new one that unifies the concepts, vocabulary, restrictions of the source ontologies. This merging can be done either at run- time or at design-time.
- Ontology Alignment activity: This activity is used to establish different kinds of links (mappings) between ontologies that are to be integrated. Ontology alignment preserves the original ontologies and does not merge them.
- Documentation activity: This entails generating sufficient documentation of products of the ontology development process and the various stages involved.
- Configuration Management: This keeps an inventory of the various versions of the ontology and their documentation. It takes care of version control for change management purposes.

In figure 2.1 a schematic view of the activities of the ontology development process is presented using UML activity diagram notations. The activity graph in figure 2.1 consists of three activity swimlanes that are used to depict the three parallel development activities (i.e. management, development-oriented and support). The ontology development process starts with scheduling which is a management activity. Thereafter the control activity starts, which bears relevance right from the predevelopment phase and throughout the entire ontology development process. Transition flow goes from quality assurance activity node to core development activities node to show that it guards the core development activities after the stages of predevelopment. The support activities are also shown to directly complement the core development activities using a directed transition flow.  Also, the various component subactivities of the management, development and support activities are shown using the subactivity state notation of UML activity  diagram.

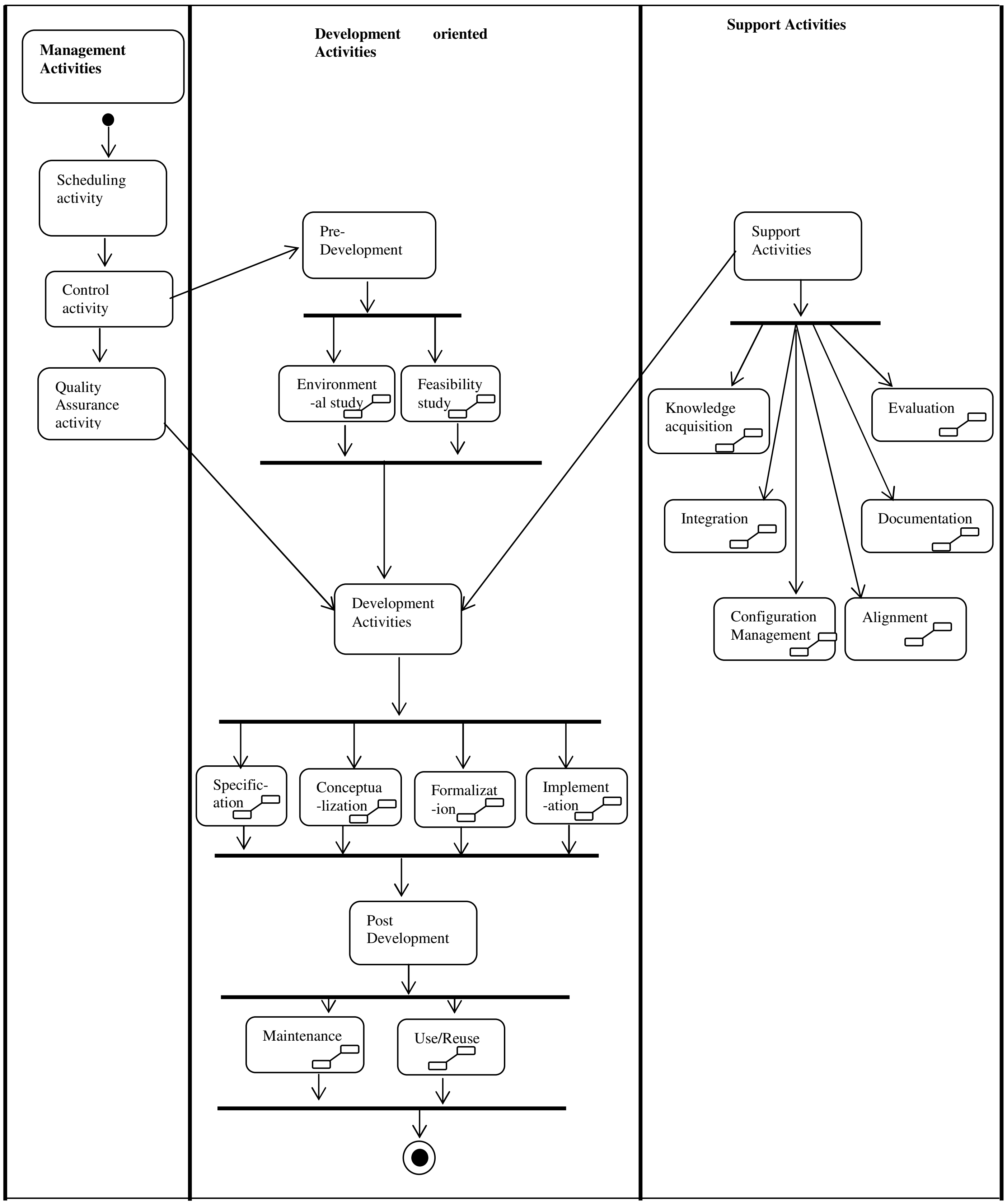


**Figure 2.1 Activities of the ontology development process: Adapted from (Gomez-Perez et al., 2004)**

### 2.7.4 Ontology development methods and methodologies

According to Noy & McGuinness (2003), there is no absolutely one correct way or methodology for developing ontologies. However there are some fundamental rules in ontology design that can help to make wise design decisions. These are given as follows:

- There is no one correct way to model a domain- there are always viable alternatives. The best solution almost always depends on the application that one has in mind and the extensions that are anticipated.
- Ontology development is necessarily an iterative process.
- Concepts in the ontology should be close to objects (physical or logical) and relationships in the domain of interest. These are most likely to be nouns (objects) or verbs (relationships) in sentences that describe the domain.

Several classical methodologies and methods for building ontologies have been reported in literature. Some of these serve to build ontologies from scratch or by reusing other ontologies. They include:

1) The Cyc method (Lenat & Guha, 1990);
2) The Uschold and King's method (Uschold & King, 1995);
3) The Gruninger and Fox's methodology (Gruninger & Fox, 1995);
4) The KACTUS approach (Bernaras et al.,1996);
5) METHONTOLOGY (Gomez-Perez, 1996);
6) The Sensus method (Swartout et al., 1999); and
7) The On-To-Knowledge methodology (Staab et al., 2001).

In this research thesis the two KR ontologies developed were built using the METHONTOLOGY methodology. This is because METHONTOLOGY is one of the most detailed approaches to ontology development, with the most accurate description of its activities and very good tool support (Gomez-Perez et al., 2004).

#### 2.7.4.1 METHONTOLOGY Methodology

The METHONTOLOGY methodology (Fernandez-Lopez et al., 1997; Gomez-Perez, 1998; Fernandez-Lopez et al., 1999) was developed within the Ontology group at Universidad Politecnica de Madrid. It facilitates the construction of ontologies at the knowledge level. Methontology was derived from the main activities of the software development process and the knowledge engineering methodologies (Gomez-Perez et al., 2004). The core characteristics of this methodology include: 1) identification of the ontology development process; 2) a life cycle that is based on evolving prototypes; and 3) techniques to carry out each activity in the management, development-oriented and support activities (Gomez-Perez et al., 2004). The methodology has adequate support tool to aid the ontology development process. ODE (Blazquez et al., 1998), WebODE (Arpirez et al., 2003), Protégé (Noy et al., 2000; Knublauch et al., 2003), OntoEdit (Sure et al., 2003), etc. are some of the available tools that give automation support to the methodology.

#### 2.7.4.2 Ontology crossed life cycles

The ontology development process (see figure 2.1) was based primarily on the framework of METHONTOLOGY and refers only to those activities performed during ontology building but fails to identify the order in which such activities should be performed. However, the ontology life cycle defines the order of activities in the ontology development process. The METHONTOLOGY approach proposes an ontology construction life cycle that is based on evolving prototypes. It allows the stepwise refinement of the components of the ontology as new versions or prototypes evolve which makes the ontology to be very dynamic and susceptible to change and growth (Gomez-Perez et al., 2004).

As a rule, METHONTOLOGY begins with the schedule activity that identifies the tasks to be performed, their arrangement, and the time and resources needed for their completion. After that, the ontology specification activity starts and simultaneously several activities such as the management activities (control and quality assurance) and support processes (knowledge acquisition, integration, evaluation, documentation, merging, alignment, and configuration management) also start. All the management and support activities are performed in parallel with the development-oriented activities (specialization, conceptualization, formalization, implementation and maintenance) during the whole life cycle of the ontology (Gomez-Perez et al., 2004).

After the first prototype has been specified, the conceptual model is built within the ontology conceptualization activity. Thereafter the formalization (though not mandatory) and implementation activities are carried out. The activities are iterative such that if there is a need for modifications they can be revisited. The figure 2.2 below shows the ontology development life cycle in METHONTOLOGY (Gomez-Perez et al., 2004). The figure shows the composition of the three main activities. The subactivities of the management and support activities are executed simultaneously with the development subactivities. Also, the figure also reveals that much of the efforts in support activities go into the knowledge acquisition and evaluation tasks.

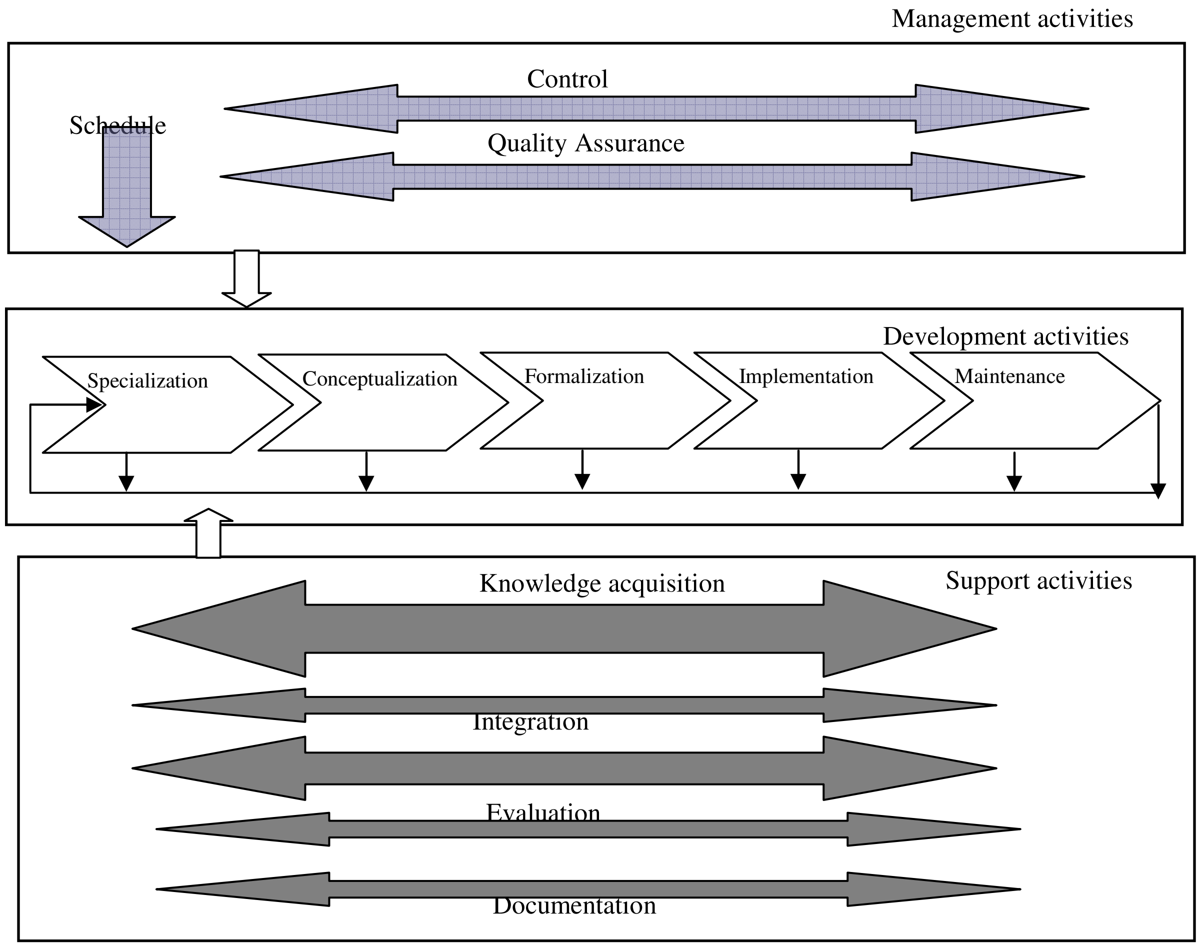


**Figure 2.2: The Methontology Development Life Cycle (Gomez-Perez et al., 2004)**

### 2.7.5 Ontology Languages

In selecting a language for developing an ontology, the preference of the developer is paramount. However factors such as the level of expressiveness of a language, its underlying knowledge representation paradigm and the reasoning mechanism attached to it must rank highest in the consideration of an ontology language.

Over the years, several AI-based languages for implementing Ontologies have been created (Gomez-Perez et al., 2001; Su & Ilebrekke, 2002; Gomez-Perez et al., 2004). Ontolingua, which is based on the knowledge representation paradigm of frames and first order logic is the most complete of the ontology languages and the one considered as a *de facto* standard by the ontology community. Other languages that have been used for implementing ontologies include: KIF, CARIN, LOOM, CycL, OCML, FLogic, OKBC, etc. These languages are underlined by diverse knowledge representation paradigms such as: frames, description logics, first order logic, and production rules.

In the recent years, the advent of the Internet has brought about the creation of new web standard formats for information exchange such as XML and RDF. As a result, new XML-based ontology specification languages such as XOL, OIL, OML, DAML+OIL, OWL, RDF Schema and XML Schema have also emerged. These new languages have two roles: The first is that they can be used to provide the semantics of information contained in electronic documents; and the second is that they can be used for the exchange of ontologies across the web. SHOE is another web ontology language that is not based on XML but rather combines frames and rules. It is an extension of HTML because its original specification was presented very early in 1996.

### 2.7.6 Ontology Tools

Broadly speaking ontology tools can be classified into three categories: web-based, computer-based and client-server tools. Generally ontology tools serve to minimize the complexity of the ontology development process by aiding different aspects of the ontology development process such as conceptualization, implementation, consistency checking and documentation (Duineveld et al., 1999; Fensel & Perez, 2002; Gómez-Pérez et al., 2004). Particularly, they enable the creation, editing, and managing of ontologies written in the various languages. Examples of Web-based tools are: Ontolingua, WebOnto, OntoSaurus, WebODE, SymOntoX, APECKS, IKARUS and CO4. Computer-based tools include Protégé-2000, ODE, KADS22, OntoEdit, OilEd, JOE, Apollo, CODE4, DOE, DUET, GKB-Editor, IODE, KAON Tool Suite, OCM, Ontology Editor and VOID. Finally examples of client-server tools include LinKFactory and OpenKnoME.

## 2.8 THE CHALLENGE OF DYNAMIC USER REQUIREMENTS IN TOURISM

The tourism product is intangible, heterogeneous (i.e. a trip may have many parts) and non-persistent (i.e. tourism services and product cannot be reserved for a particular consumer for long) (Henriksson, 2005). These core characteristics inevitably influence the nature of information exchange within the tourism value chain which includes: tourist, tour operator, travel agent, hotelier, government, destination management organizations (DMO), Airlines and so on (Henriksson, 2005). e-Tourism which is the use of ICT applications for enabling the effective flow of information and business transaction in tourism faces a big challenge because of: 1) the unique nature of the tourism product, 2) the long tourism value chain; and 3) the heterogeneous nature of ICT infrastructures.

The most important stakeholder in the tourism value chain is the consumer, who is at the end of the value chain and whom all efforts and services in tourism are constructed to benefit. According to Steinbauer (2005), the modern tourist is prone to the following characteristics: 1) become more mobile and critical; 2) become less loyal and frequently change their product preferences; 3) look for more specialized products and ask for better service; 4) want more and better information; 5) compare more products in more detail; 6) have fast changing needs and belong to different niches at the same time; and 7) tend to make more but shorter vacations. These complex and dynamic characteristics of tourism consumer behaviour portend critical challenge for providers of e-tourism services (Steinbauer, 2005; Werthner & Klein, 1999).

Notably, the characteristics listed as 2 to 6 above, are particularly relevant to the context of this thesis, because it raises the concern of how developers of tourism support systems can effectively respond to the trends in consumer behaviour. The question that comes to mind is: Is there a software development approach or methodology that could be engaged to tackle this dynamic nature of consumer behaviour? This therefore provides the basis for the second research question of this thesis (see Section 1.2), to which this thesis also provides an answer.

## 2.9 SPLE - A PANACEA FOR MANAGING DYNAMIC USER REQUIREMENTS IN TOURISM?

A candidate software development paradigm that possesses the potential to cater for the challenge of dynamic user requirements in e-tourism is Software Product Line Engineering (SPLE). A software product line approach by its characteristics enables the definition of system instances dictated by marketing and product plan specification from prospective users (Bass & Kazman, 2003). It also engenders software evolution within a family of closely related software products by ensuring that the inter-product commonalities and variabilities among products are well exploited for versioning and maintenance purposes (Gamma et al., 2005; Shaw & Garlan, 1996). This suggests that if a tourism market niche that cuts across segments of a specific tourism value chain with minimal variations and predictable change patterns is identified by a software development organization then the feasibility of a carefully planned SPL approach can be explored. This will enable a software development organization to manage the dynamism of e-tourism requirements that pertain to that domain.

Further to this argument is the fact that generally TIS share many attributes in common and mostly perform similar functions. They mainly differ in the nature of local information content they deliver and the scope of tourism interest that is being promoted whether at the national, continental, regional, state, local and enterprise levels. The similarity in TIS functionalities makes them good candidates for a product line development, which seems not yet a prevalent practice in the e-Tourism domain (Daramola et al., 2008). However, the fact that Tiscover AG (www.Tiscover.com) among others renders tourism support services for eight different countries of the world, with same set of functionalities but unique contents is a clear indication of the viability of Software Product Line Engineering (SPLE) in the tourism domain.

### 2.9.1 What is Software Product Line?

A Software product line (SPL) is a set of software intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and are developed from a common set of core assets in a prescribed way (Bass & Kazman, 2003; Ezran et al., 2002).

It entails the production of a set of software products using common core assets. This core asset may be a software component, a process model, a plan, a document or any other useful resource for building a system. For example, one of the most important core assets in a product line is the software architecture. Another important one is the product line's scope, which is a statement of what products the core assets are intended to support. The scope defines the commonality and variability (ways in which they differ from each other) that defines every product in the software product line. Software architecture provides a context in which other assets can be developed with the right flexibility to satisfy the products in the product line.

Software Product Line Engineering (SPLE), which is based on exploring inter-product commonality, is rapidly emerging as a viable and important software development paradigm. Also, it facilitates the production of tailor-made systems built specifically for the needs of particular customers or customer groups by exploiting the commonalities shared by software products to realize order-of-magnitude improvements in time-to-market, cost, productivity, quality and other business drivers. It also enables rapid market-entry and flexible response, and provides a capability for mass customization of software products.

A product in the software product line is formed by taking applicable components from the base of common assets, tailoring them as necessary, through pre-planned variation mechanisms such as parameterization or inheritance, adding any new component that may be necessary and

assembling the collection according to the rules of a common reference architecture. This connotes that building a system in a product line becomes more of assembly or generation than of creation. The predominant activity also becomes integration rather than programming.
There are other terms that have been used in literature that convey essentially the same meaning as the set of terms used in this thesis. In some cases, the term *product family* has been used to refer to a product line; *platform* is used to refer to the set of core assets, and the term c*ustomizations* is used to refer to the products of the SPL. The term *domain engineering* is used to refer to core asset development and *application engineering* is used to refer to product development. Typically, the technical practice of software product line engineering can be defined as:

***SPLE = Domain Engineering + Application Engineering***

Where domain engineering consist of the aspects of core assets development, while application engineering entails the generation of the multiple products that constitute the SPL using reusable components in the asset repository.

### 2.9.2 Merits of the SPL Approach

The adoption of SPL approach for the engineering of TIS has significant merits. However, for SPL to engender systematic and strategic reuse within an organization, it must be actively supported with adequate managerial policy to back the technical initiative. For example an organization will need to migrate from developing a single product to developing product families. Hence product family-oriented abstractions must be developed from requirements, and relevant core reusable assets built that can be subsequently leveraged in the development of variant products by the organization. This offers significant advantage in terms of cost and time of development when compared with traditional reuse approaches.

Another advantage of SPL over traditional reuse is that the cost of maintenance is reduced because the products are built on a common platform, all products using the platform can share the maintenance costs of the platform, while cost of maintaining the variability in products is relatively minimal.

### 2.9.3 Product Line Artifacts

The key artifacts of a product line are the product line requirements, the product line architecture (PLA) and the product line components. These three artifacts differ from their single product equivalent and are described as follows:

**Product Line Requirements:** These are base requirements that span several products in contrast to their single product equivalent. It defines the basic limits that must be satisfied by each component in the PLA. This means that the product line requirements must be written with variation points to be able to capture variations between individual products within the product line. Product line requirements can be classified as "Non-reusable", "directly reusable", 'variable" or "obsolete" (Mannion et al., 2000).

**Product Line Architecture:** The software architecture of a program or computing system is the structure or structures of the system, which comprises software elements, the externally visible properties of those elements, and the relationships among them. The externally visible properties refer to those assumptions other elements can make of an element (component), such as its provided services, performance, characteristics, fault handling, shared resource usage and so on (Bass & Kazman, 2003). The success of any software project depends on the architecture because it is an important fundamental artifact of a system. Also, the quality and attributes of a system such as performance, modifiability, reliability and usability are all derived directly from the architecture. In a product line, the dominant core asset is the reference architecture for the product line, which is used at every product instantiation. It defines a set of explicitly allowed variation that represents the individual products that can be built with a product line. A number of variability mechanisms exist of which the FORM and FAST methods are prominent examples (Svahnberg et al., 2001; Thiel & Hein, 2002). Also, some of the Product line architecture design methods available in literature include: COPA, FAST, FORM, KobrA and QADA (Matinlassi, 2004).

**Product Line Components**: The components in a PL can either be part of the core assets or they can be developed for product specific reasons. Even though PL development draws significantly from component-based development (CBD) (Szyperski et al., 2002), the notion of components in the two concepts differs. This is because:

- Product line components are typically not independently deployed as CBD components;

- Product line components are assembled in a prescribed way specified by their production plan and the PLA; and
- Product line components implements variability mechanism specified by the product line architecture (Thiel & Hein, 2002).

Figure 2.3 shows an overview of the activities and artifacts necessary for component design and implementation in a SPL paradigm context.

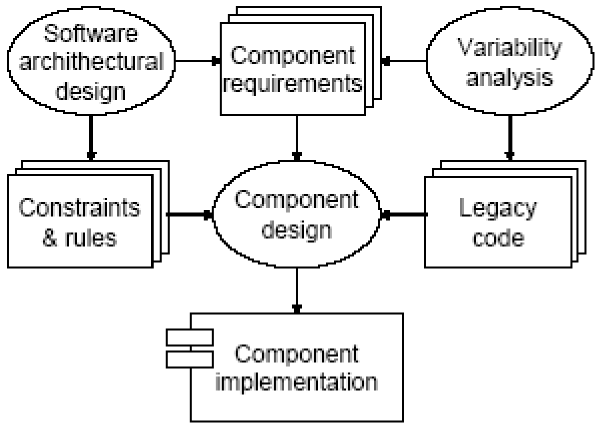


**Figure 2.3: Activities and deliverables in SPL component development (Bosch, 2000)**

### 2.9.4 State-of-the-art in Software Product Line Research

The trend of current research activities in SPL has among its main emphasis issues involving the design and evaluation of product line architectures, definition of product line scope through the specification of feature variability and dependency among products, automatic creation of generic product line architecture from requirements, model driven product line architectures (MDPLA), support tools for model driven development (MDD) for product lines and industrial case study reports of product line practices.

A number of product line architecture design methods have been discussed in literature this includes: Component-Oriented Platform Architecting (COPA) (America et al., 2000), Family-oriented Abstraction, Specification and Translation (FAST) process (Weiss et al., 1999), Feature-

Oriented Reuse Method (FORM) (Kang et al., 1998), KobrA (Atkinson et al., 2002) and Quality-driven Architecture Design and Architecture (QADA) Analysis (Niemelä, 2006).

Perry (1998) suggests useful ways of 'genericizing' architectural descriptions with an analysis of the strengths and weaknesses of each approach. Among those suggested are: 1) use of software architectural style, 2) defining a variance-free architecture, 3) use of a parametric description using varying binding times, 4) use of service-oriented description for selective provisioning and 5) use of under-constrained architecture. The generic architecture for Tourism Product Line Architecture (TPLA) proposed in this thesis is a product of an integration of all five underlining concepts, which makes it sufficiently generic for the tourism domain. A description of a reusable architecture for federated client/server systems is given in (Gomaa & Farrukh, 1999) although not specifically dedicated to a product line paradigm.

The work by Deng et al, (2007) deals with the challenges of evolution in Model-Driven Software Product-line Architectures. The Koriandol system (Balzerani et al., 2005) is product line architecture for general web applications. The special feature of Koriandol in contrast to other component-based systems is that its components have variability handling mechanism built into them. Koala is an implementation of software component model designed for creating a large variety of products (van Ommering, 2002). It is specifically dedicated to the modelling of embedded systems. The modelling and specification of a PLA for a family of meshing tools is given in (Bastarrica et al., 2006). Meshing tools are pieces of software that are used to generate and manage discretization of a domain that find application in mechanics design and medicine. The PLA was modelled with ArchStudio tool and formally specified using xADL (Dashofy et al., 2001). Among the reported case studies on architectural analysis and design of product lines include (Lutz & Gannod, 2003), which is an evaluation of an existing product line of Interferometers. The work in (Schwanke & Lutz, 2003) is an experiences report on the architectural design of a modest product family in this case some medical image processing products. A number of other case studies were reported in (Bosch & Svahnberg, 1999) and (Clement & Northrup, 2002), but none of these reports is specific to the tourism domain.

ATAM (Architecture Trade-off Analysis Method) is a popular scenario-based architecture analysis method that is discussed in (Kazman et al., 1998) and a survey of several software architecture analysis methods can be found in (Ionita et al., 2002).
The modelling of feature variability and dependency is an important aspect of product line practice. Feature-Oriented Domain Analysis (FODA) feature diagram (Kang et al., 1990) and Feature-Oriented Reuse Method (FORM) (Kang et al., 2002) are few of the popular approaches mostly used. Ye and Liu (Ye & Liu, 2005) initiated a newer feature modelling approach, which captures both the feature tree view of product line components, and a dependency view.

Also of interest are the various approaches for documenting the different views of software architecture. Kruchten presented the "4+1" views model in (Krutchen et al., 1995), while in (Hofmeister et al., 2001) the views model of Hofmeister et al. was presented. Other prominent approaches to documenting views of architecture are the Software Engineering Institute's (SEI) Views and Beyond Approach (V&B) (Clements, 2005), IBM Standard for Architecture Description (ADS) (Youngs et al., 1999) and the HP (Hewlett-Packard) Template for Documenting Views of Software and Firmware Architectures (Ogush et al., http://www.architecture.external.hp.com)

## 2.10 THE CONTEXT OF THIS RESEARCH

From the foregoing issues, a number of gaps exist in literature which defines the context of this research. The first is the need for the generation of more dependable tourism recommendations which have not been adequately addressed by existing TIS platforms. The second is the problem of managing dynamic user requirements in tourism to which literature has not been able to provide a product development-based solution approach till date.  These two gaps become the premise for the central research question being investigated in this thesis, which is:

How do we facilitate more dependable recommendations in TIS and at the same time cause such systems to evolve in tandem with the dynamic nature of user requirements in e-tourism?

For adequate explication, the central question has been split into the following two research questions:

1. *What methods are needed to enable dependable recommendations in TIS in order to foster users' confidence?*
2. *Given the frequently changing and growing nature of user requirements in tourism, what approach is required by TIS developer organizations to facilitate the proactive evolution of their products in order to cater to dynamic user requirements?*

This thesis aims at proposing a viable solution to these questions.

## 2.11 PROPOSAL OF TOURISM PRODUCT LINE ARCHITECTURE (TPLA) FOR NEXT-GENERATION TIS

The advent of new technologies such as the semantic web and mobile computing have offered new transactional possibilities that have complicated the nature of e-tourism requirements, a challenge that currently eludes the capability of existing TIS (Staab et al., 2002; Felfernig et al., 2005). Hence, next generation TIS must be equipped with semantic web, personalization, context-aware and content-sharing capabilities that can cater for the dynamic challenges of the e-tourism domain.

As an aftermath of a detailed study of the emerging user requirements and technology needs in the e-tourism domain, a generic reference architecture for Tourism Product Lines (TPLA) is proposed. The TPLA is conceptualised as a platform for evolving intelligent component services in TIS in response to the dynamic challenges of the e-tourism domain. It is a layered architecture of core reusable components that can be leveraged for the development of TIS product family. This is the one of the new perspectives offered by this thesis (see Daramola et al., 2008), which also serves as a springboard for the rest of the work.

### 2.11.1 Justification for the TPLA

A detailed study of tourism domain clearly reveals that most TIS share similar visions, objectives and similar functionalities. Also, a survey of the e-tourism domain and literature shows that the features found in TIS can be grouped into three main functional service categories namely: 1) *Information services*: these involve the provision of relevant information content to the user, content sharing and content syndication (Hammersley, 2003). The information service provider builds its content and publishes or shares it for specific purposes, 2) *Transaction Service*: here the service provider may receive inputs that are consumed in the process of constructing something of value for an actor with specified minimum level of quality and may also initiate other transactions at run-time (B2B transactions) and 3) *Third Party Service:* these are services provided by e-Business and e-Commerce entities that are external but interoperate with the TIS through web services.

SPL by the nature of its characteristics is intrinsically suited to handling some of the dynamic challenges, which the next generation TIS must address. For the realization of a product line objective in tourism a reference architecture for the product line is crucial. The success of any software project depends on the architecture because it is a primary and important fundamental artifact of a system from which its quality attributes are derived. Essential quality attributes of a system such as performance, modifiability, reusability and usability are all dependent on software architecture of a system (Bass & Kazman, 2003). In a product line, the dominant core asset is the reference architecture of the product line, which shows the configuration of core components that are used at every product instantiation.

### 2.11.2 Description of the TPLA

*(A significant part of this section, has been published in (Daramola et al., 2008): Information and Communication Technologies in Tourism 2008)*

The reference architecture for e-Tourism product line (TPLA) (see Figure 2.4) is a layered architecture consisting of five layers. Each layer represents specific infrastructural abstractions of the product line architecture. The description of the specific layers is given as follows (Daramola et al., 2008):

**1. Client Layer:** The client layer abstraction is comprised of client devices through which the services of the TIS can be requested. This includes PDA, web browser (through Laptop and PC) and i-Mode device. Components at this level consume the services of the architecture.

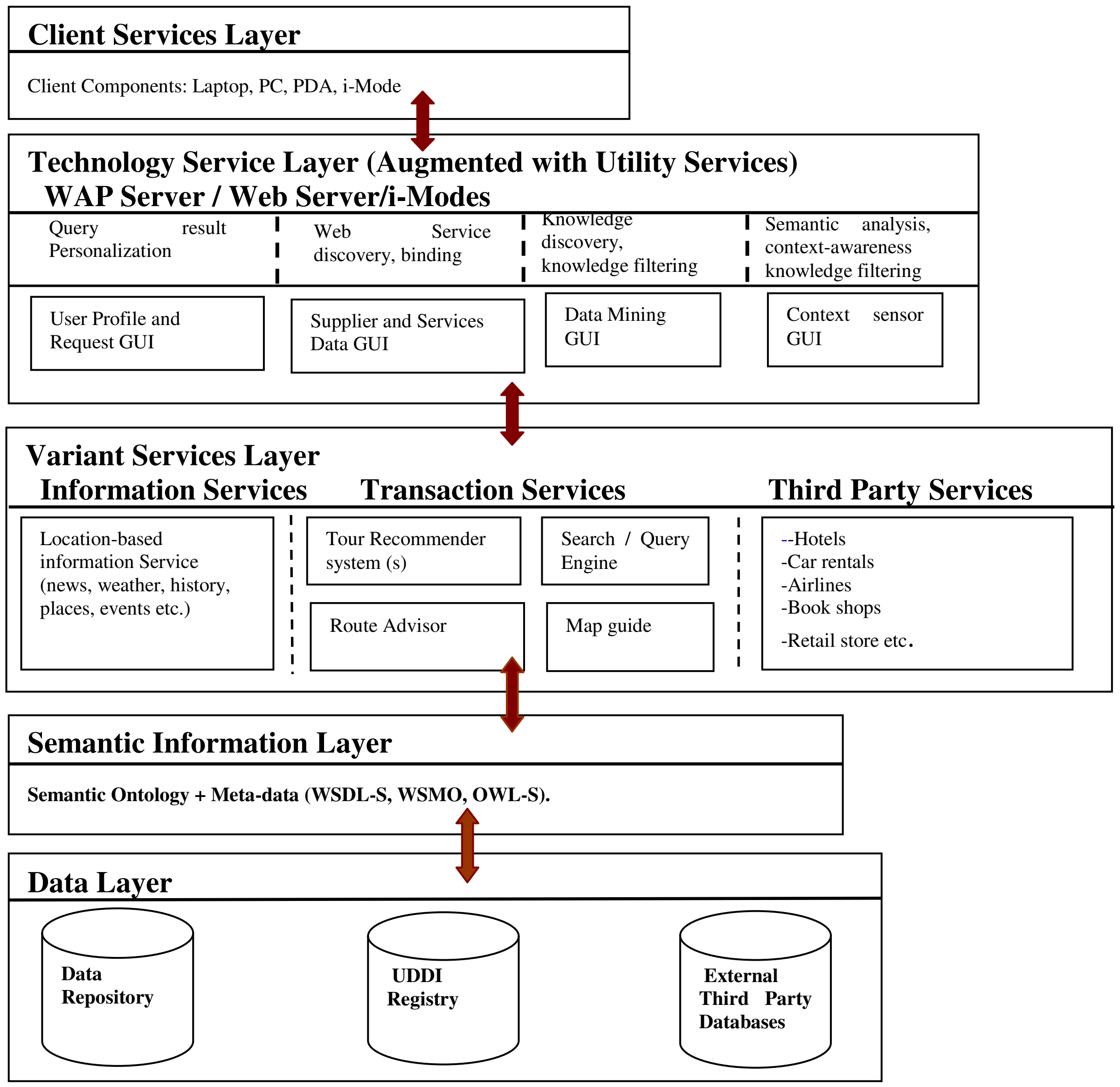

Client Services Layer
Client Components: Laptop, PC, PDA, i-Mode
Technology Service Layer (Augmented with Utility Services)
WAP Server / Web Server/i-Modes
Query result Personalization
Web Service discovery, binding
Knowledge discovery, knowledge filtering
Semantic analysis, context-awareness knowledge filtering
User Profile and Request GUI
Supplier and Services Data GUI
Data Mining GUI
Context sensor GUI
Variant Services Layer
Information Services
Transaction Services
Third Party Services
Location-based information Service (news, weather, history, places, events etc.)
Tour Recommender system (s)
Search / Query Engine
Route Advisor
Map guide
--Hotels
-Car rentals
-Airlines
-Book shops
-Retail store etc.
Semantic Information Layer
Semantic Ontology + Meta-data (WSDL-S, WSMO, OWL-S).
Data Layer
Data Repository
UDDI Registry
External Third Party Databases

**Figure 2.4 A Layered Reference Architecture for Tourism Product Line (Daramola et al., 2008)**

**2. Technology Service Layer:** The Technology Service layer defines the implementation platform for all the services in the PLA. This can be WAP for WAP-enabled mobile applications, i-mode for i-mode-enabled applications or HTTP for web clients. The Technology Service layer is augmented with the implementation of a set of four graphic user interfaces (GUI) collectively referred to as Basic Utility Services. These are:

- ***User profile and request interface***: responsible for collection of information on the preferences of users and rendering personalized services by exploiting the knowledge gained from previously stored user profile.
- ***Data mining interface***: responsible for knowledge discovery services using in-built association rule mining, collaborative filtering and classification algorithms.
- ***Supplier and services data***: responsible for content upload and data storage services.
- ***Context sensor services***: responsible for the tracking of the environment, social, task, and spatio-temporal (time, location, direction, speed, shape) contexts of the user. A comprehensive context list is built by this component and is used to aid the delivery of services.

These four distinct functionalities in the Technology Service are provided for every system in the product line. Every system in the product line (PL) must be able to hold a conversation with Technology Service before it can request any of the utility services in line with the principle of conversation before composition. This is for the determination of appropriate communication protocol depending on the nature of requesting client and the provision of data-aware and context-aware services, which only Technology Service is mandated to provide on demand.

**3. Variant Services Layer**: The Variant Services Layer consists of the class of all services that are not basic to the architecture. This set of optional services can be further sub-classified into information services, transaction services and third party services. Information services component represent a loci of computation that are responsible for the provision of location-based information contents such as: news, events, places, accommodation and weather reports. Transaction services are logic components that are responsible for the delivery of services such as travel recommendation, destination recommendation, route advisory services, query search, map guide, etc. While third party services are the external e-Commerce sites, which provide web services that can be discovered and consumed by the TIS. Examples of these include car rental services, hotel accommodation booking services, bookshops, shopping stores, etc.

**4. Semantic Information Layer:** This layer of abstraction defines the semantic awareness that is exhibited by all components in the architecture. The components of this layer are the various middleware semantic models of knowledge representation specifically designed to enable intelligent attributes like context-awareness, personalization and semantic awareness in the logic components of the TPLA. Candidate semantic models available in this layer include ontologies, knowledge bases, OWL-S, WSMO, WSDL-S, and XML. The implementation of these semantic models will facilitate improved query processing, information exchange and cross platform interoperability among various information systems yielding high quality services. Three of the specific services that will be provided at this layer of abstraction include:

i) *Context-awareness*: this refers to the ability of a system to make use of information about the device platform, the user, and the surrounding environment in the delivery of its services.

ii) *Personalization*: this is a form of context-awareness in which the system gathers user-information during interaction with the user in order to construct a response that uniquely fits the user's preferences.

iii) *Semantic Web Services*: this will enable the automatic annotation, advertisement, discovery, composition and execution of inter-organization business logic, making it possible for several organizations and individuals to communicate with each other to carry out various commercial activities and to provide value-added services (Cardoso, 2004).

**5. Data Layer**: The data layer is composed of a set of database abstractions that stores the information content delivered by the TIS. These include data repositories, the UDDI registry from which third party web services are discovered, and all other external databases to which the TIS can bind.

## 2.12 SUMMARY

The chapter presents the issues that define the research context of this thesis. It started with a discussion of the necessity for intelligent systems for tourism and the progress made so far in the course of intelligent enabling of e-tourism systems. Secondly, an argument for recommendation-intensive TIS is presented as justified by the information intensive nature of the tourism industry and the intangible nature of the tourism product which makes it impossible for customer to touch a product before a trip. This is followed by an overview of recommender system types and techniques. The key aspects discussed include content-based, collaborative filtering, knowledge-based, and hybrid recommender system. Thereafter, the chapter specifically reviews the recommendation technologies in tourism, and the limitations of existing approaches and the gaps that this thesis attempts to fill. Next is the subject of ontology-based tourism recommendations as a way of improving the dependability of tourism recommendation. After this the subject of dynamic requirements in tourism is critically examined with the identification of the Software Product Line Engineering (SPLE) paradigm as a possible panacea for managing dynamic requirements in tourism. The chapter closes by formally articulating the research context of this thesis and the proposal of a Tourism Product Line Architecture (TPLA) as a means for handling dynamic challenges in next generation TIS.

# CHAPTER THREE

# PRODUCT LINE FOR ONTOLOGY-BASED TOURISM RECOMMENDATIONS (PLONTOREC) APPROACH

## 3.1 INTRODUCTION

The **P**roduct **L**ine for **On**tology-based **To**urism **Rec**ommendations (PLONTOREC) approach is the proposed solution to the two research questions posed in this thesis. The chapter presents an overview of PLONTOREC as a novel hybrid of software product line engineering and ontology engineering dedicated to the development of recommendation-intensive TIS. It gives insight into its strategy and underlining assumptions, its process architecture, and its main sub-processes. In addition the modalities for the validation of the PLONTOREC approach are discussed. The chapter closes with a summary and discussion on expected results.

## 3.2 OVERVIEW OF THE PROPOSED SOLUTION: PLONTOREC APPROACH

The vision of Product Line for Ontology-based Tourism Recommendations (PLONTOREC) approach originated from the generic tourism product line architecture (TPLA) presented in section 2.10. It is a product realization concept for TIS that share in the attributes of the generic TPLA proposed in (Daramola et al., 2008). PLONTOREC is a novel hybridization of ontology engineering and software product line engineering concepts that is dedicated to the development of TIS. It is a specialized software development approach that thrives on carefully planned knowledge reuse and software reuse initiatives that are designed to enable dependable and intelligent tourism recommendations in TIS. It is also aimed at providing a platform for such TIS to evolve proactively in tandem with dynamic user requirements. PLONTOREC is proposed as a unified approach to tackling the two research questions that have been highlighted in this thesis. It is designed as a software development approach that facilitates the creation of recommendations-intensive TIS products in TIS development organizations that brings a boost in productivity and minimizes cost. Further details on the PLONTOREC approach are presented next.

### 3.2.1 Limitation and Assumptions

The application of PLONTOREC for the development of TIS is constrained by a set of preconditions that assures of its feasibility in a given domain. These are:

1. All developed TIS belong to the same organization or a consortium of collaborating organizations;
2. The content configurations of different variants of TIS products within the product family are known and predetermined in advance;
3. The process description for developing specific kinds of TIS product is also predetermined; and
4. Planning of the structure of components and reuse context is done proactively in advance.

In Addition, PLONTOREC is based on the following assumptions:

1. New products evolve by composition, using existing components in the common asset base;
2. New versions of TIS products are variations of existing ones, having many things in common with the old versions; and also
3. The points of variability are minimal and predictable;

PLONTOREC is designed for specialized unified knowledge and software reuse-oriented development of TIS. It does not address the general-purpose reuse context. As such, the limitation and assumptions of PLONTOREC are all directly derived from the principles that govern the practice of SPL initiatives (Shaw & Garlan, 1996; Gamma et al., 2005). The limitation is meant to provide a guide on how the technical and organizational aspects of the product line should be managed. The set of assumptions on the other hand are those that facilitate the highest payoffs in PL development and specify the scenario when PLONTOREC is optimally applicable.

### 3.2.2 The PLONTOREC Process Architecture

The PLONTOREC process architecture provides insights into the activities involved in the creation of TIS products using the PLONTOREC approach (see Figure 3.1). It is an adaptation of the software product line and ontology development process life cycles, which are the two standard system development practices encapsulated in PLONTOREC. The SPL practice is divided into three component processes: Product line management, Domain engineering and Application engineering, while the ontology development activities which is the fourth component process represents the Ontology engineering practice in PLONTOREC.

The flow of activities in PLONTOREC is not necessarily sequential and it is possible to iterate through the different processes. PLOTONREC is initiated from product line management to ontology engineering, domain engineering and terminates with application engineering. After each stage of domain engineering and application engineering, product line management process is repeated in order to re-evaluate the PLONTOREC approach with updated data from domain analysis. This is to ascertain whether the process should proceed or be halted.

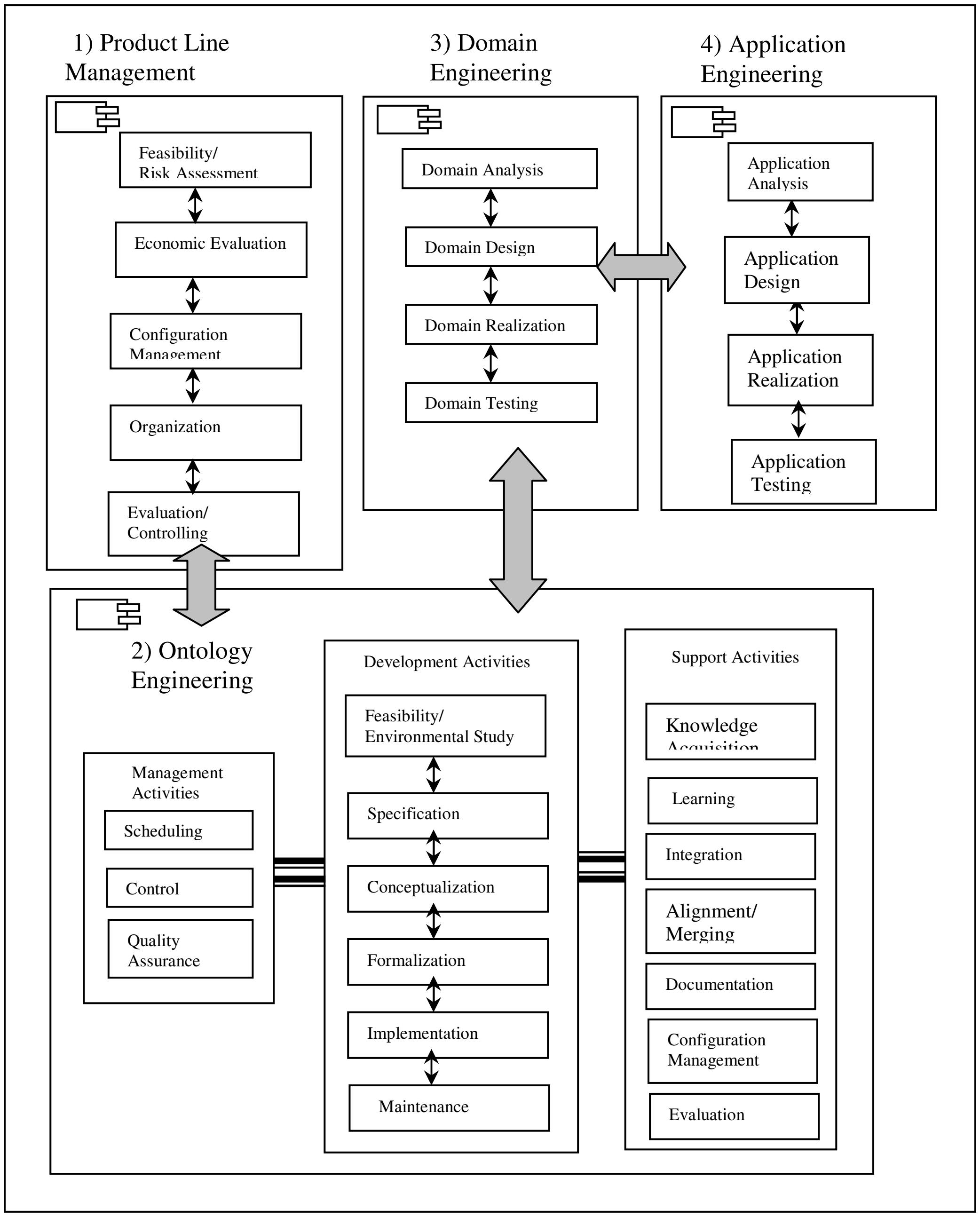


**Figure 3.1**. **The Process Architecture of PLONTOREC**

### 3.2.3 Product Line Management in PLONTOREC

The product line management sub-process in PLONTOREC defines the set of activities that provides the necessary managerial guide and organizational control that complements the technical aspects of domain engineering, application engineering and ontology engineering of the PLONTOREC approach. Product line management is carried out at specific interval periods or at the end of each sub-process. The main activities of product line management include:

**i) Feasibility and Risk Assessment**: This involves an assessment of the technical and organizational viability of the PLONTOREC approach and a determination of the risks associated with it. The tools used for these activities include results obtained from interviews, surveys, observations, market analysis, domain field studies and product reviews. Qualitative assessment methods such as questionnaire or quantitative assessment methods such as measurements and estimation are used to achieve this. A product line management overview document is generated at the end of the exercise, which contains the general information about the product line that is to be created (Thiel & Hein, 2002; Gamma et al., 2005).

**ii) Economic Evaluation**: This involves the evaluation of the economic viability of the PLOTONREC approach. In order to do this a quantitative comparison of the situation where PLONTOREC is used and when not used is required. The measurement is done on the basis of effort in person months involved in the cases with PLONTOREC and without PLONTOREC. The metric is assumed to have direct impact on the economic factors such as cost, net present value etc. The model for estimating effort in PLONTOREC is derived from the standard model already developed for SPL (Bockle et al., 2004). This is given as:

$$\mathbf{C_f = C_{org} + C_{cab} + {}^{n}\sum_{i=1}(C_{unique}(P_i) + C_{reuse}(P_i))}$$

Where $\mathbf{C_{org}}$ ,$\mathbf{C_{cab}}$ , $\mathbf{C_{unique}}$ , $\mathbf{C_{reuse}}$ are cost functions.

The effort in PLONTOREC is estimated as follows:

$$\mathbf{E_{plontorec} = E_{org}+ E_{dom}+ E_{onto}+E_{ontoupdate} + N *(E_{reusewith}+E_{uniquewith}+J*E_{updatewith})}$$

Where

$\mathbf{E_{org}}$: Effort to introduce the product line, adapt the organization, train staff etc.

$\mathbf{E_{dom}}$: Effort expended in domain engineering for the development of core assets, cost of commonality and variability analysis

$\mathbf{E_{onto}}$: Effort expended in the development of relevant ontologies

$\mathbf{E_{ontoupdate}}$: Effort expended in updating content of ontologies after initial development and its maintenance

**N**: Number of TIS products in the product line

$\mathbf{E_{reusewith}}$: Average effort in application engineering for the reuse of existing core assets e.g. choosing, configuration, searching and integration of core assets.

$\mathbf{E_{uniquewith}}$: Average effort to extend core assets base with core assets unique to a product, effort with manual adaptations of core assets after creation.

**J**: Average planned number of content update cycles for one TIS product

$\mathbf{E_{updatewith}}$: Average effort of updating the product-related core assets in the core asset base;

For a well-designed PLONTOREC approach $\mathbf{E_{reusewith}+E_{uniquewith}}$ should be relatively small compared to $\mathbf{E_{org}+ E_{dom}+E_{onto}}$ as similarly applicable to well-engineered product line initiative.

The effort without PLONTOREC can be estimated as follows:

$\mathbf{E_{without} = N * (E_{uniquewithout} + J*E_{updatewithout})}$

Where

**N**: number of individual TIS products in the product line;

$\mathbf{E_{uniquewithout}}$: Average effort to create one unique TIS;

**J**: Average planned number of content update cycles for one information product;

$\mathbf{E_{updatewithout}}$: Average effort of updating one TIS product.

In order to determine the economic justification for PLONTOREC, we adopt a similar model as used for conventional SPL (Bockle et al., 2004). Hence for a well designed PLONTOREC approach, $\mathbf{E_{reusewith}+E_{uniquewith} < E_{uniquewithout}}$, and $\mathbf{E_{updatewith} < E_{updatewithout}}$, and $\mathbf{E_{reusewith}+E_{uniquewith}}$. It must be noted that PLONTOREC must first invest $\mathbf{E_{org} + E_{dom} + E_{onto}}$, which will prove advantageous after several TIS are realized. A significant and unique advantage of PLONTOREC is the improved dependability of intelligent recommendations from TIS products, which may also provide good justification for the initial efforts expended on domain engineering and ontology engineering. In all cases PLONTOREC will be considered successful or viable if the $\mathbf{E_{plontorec} \leq E_{without}}$ .

Nevertheless, the cost differential between $E_{plontorec}$ and $E_{without}$ becomes less significant when the benefit of more dependable recommendations in TIS offered by the PLONTOREC approach is considered.

iii) **Configuration Management:** The essence of configuration management is to ensure that changes in the products of the product line are well managed. A configuration represents a fixed arrangement of a set of items at some point. A configuration item is the smallest unit of change. The description of a configuration that gives the details of each item that belong to that configuration is called a baseline (Tichy, 1992). Configuration management in PLONTOREC ensures that changes that need to be made to products by way of upgrade and versioning are carefully planned in a way that makes them technically realizable without disrupting the design of the product line. Some of the issues determined during configuration management include identifying: the core asset to be upgraded, the new core assets to be added to the core asset base, the new products that need to be added to the product line based on market demands or trends in consumer behaviour, and the versions of core asset artifacts that must be used to implement specific TIS.

iv) **Organization Aspects:** This represents the set of organizational initiatives that are implemented to guarantee the effectiveness of all the technical sub-processes of PLONTOREC and its overall success. The activities expected in the context of an organization include: assigning appropriate role responsibilities to groups and individual staff, providing necessary infrastructure, defining clear and measurable goals and objectives. Role responsibilities are defined in a general way by assigning every staff member to the specific subprocesses (i.e. product line management, ontology engineering, domain engineering, and application engineering). For some product lines, only one employee can be responsible for all subprocesses, if the effort required is relatively small.

v) **Evaluation and Controlling:** This provides an avenue for the evaluation and controlling of the entire PLONTOREC approach. Periodical evaluations are carried out at specific points, notably after ontology engineering, domain engineering and application engineering to determine whether the process should proceed. Results obtained from the feasibility and risk assessment are used, with the same question asked repeatedly concerning the status of development in the product line. If all answers to questions related to feasibility are affirmative then feasibility is certified, if any of the answers is negative then PLONTOREC should not be used. If any of the answers is uncertain then it should be resolved to yield positive or negative answer, so that the decision of whether to continue with PLONTOREC or not can be taken. In the same way risk assessment is carried out together with cost and benefit evaluation. Also, for control purposes the configuration management schedules are renewed based on emerging market and product views in order to ensure that correct steps and decisions that will facilitate the objectives of PLONTOREC are taken (Tichy, 1992).

Product line management is generally concerned with the management and organizational issues of PLONTOREC at initiation and throughout its entire lifecycle. It ensures that accurate decisions are made based on current realities for the success of the product line initiative.

### 3.2.4 Ontology Engineering in PLONTOREC

Ontology engineering is the sub-process that focuses on developing the reusable knowledge artifacts needed for the execution of the PLONTOREC approach. During this period, the suite of ontologies that are relevant to the goals of the product line development are either constructed from scratch or built by re-engineering existing ontologies. Ontology engineering starts with the scheduling activity during which time the task to be performed, the arrangement of such tasks, the time and resources needed for their completion are all identified. Thereafter the development activity starts with the specification of the ontology, at the same time the various management activities (i.e. control, quality assurance) and support activities (i.e. knowledge acquisition, ontology learning, ontology evaluation, ontology merging, documentation, and configuration management) also starts. Section 2.6.3 can be referenced for further details on the key activities of ontology engineering. Also the details of the development of two knowledge representation ontologies are discussed in sequel sections (Chapter 4) of the case study part of this thesis.

### 3.2.5 Domain Engineering in PLONTOREC

Domain engineering is the sub-process of PLONTOREC that is concerned with the construction of all reusable software assets that are used for building the variant TIS products in the product line. The artifacts created during domain engineering are the core assets that makeup the core assets repository for the product line development. During domain engineering, the reference architecture for the product line is created that consists of the core components of the product line. Also during domain engineering the components are constructed, tested and certified for deployment in application engineering. The details about key activities in domain engineering are discussed as follows:

i) **Domain Analysis:** This is the activity carried out at the beginning of domain engineering to systematically analyse the problem domain and to structure the knowledge in a way that is useful for other phases of the product line development process. The most important sub-activities of domain analysis are domain requirements engineering and domain scoping (Arango, 1994). During domain analysis in PLONTOREC, the requirements that span the entire scope of the tourism domain under consideration are captured, while domain scoping (Pohl et al., 2005) is also carried out to identify what should be in the product line and what should not. It is also at this point, that the configurations of all possible variants of products that will constitute the product line are determined. The inputs for the domain analysis include results of interviews, market surveys, existing systems and other requirements documents in the specific domain.

ii) **Domain Design:** This activity generates the reference architecture for the product line based on the requirements gathered from domain analysis, and additional abstraction models that facilitate the development of TIS products. The different aspects of domain design include: conceptual design, logical design and physical design. The dominant artifacts of conceptual design are: 1) *Conceptual product line model*: which is a model of the reference product line architecture which incorporates the basic and optional features available, and from which TIS products in the product line are instantiated; and 2) *Content component model*: which defines the details of content components such as modules, functions, logic components, subsystems, database etc. The logical design consists of: 1) *Core asset version graph model*: which defines the available versions of core assets that can be used in different TIS product configurations. It specifies the uniform way in which core assets are to be versioned in order to engender the evolution of core assets in tandem with dynamic requirements or improvements in core functionalities; and 2) *Product map template:* which enables systematic management of the use of core assets in realizing specific TIS products. Data obtained during application engineering and from the reference product line architecture are used to determine the configuration of core assets to be used in the composition of specific TIS products. It is implemented as a two-dimensional matrix of a listing of features and variant TIS products. Additionally, the physical design consists of 1) Construction specification: which defines the construction workflow model that is used to create every possible TIS product in the product line; and 2) Workflow design patterns: which are pre-constructed and tested pieces of reusable workflow patterns that can be relevant in domain engineering. In addition, design of other software artifacts relevant to the course of development are also implemented during domain design.

iii) **Domain Realization:** This activity involves the platform specific implementation of core assets that are used in application engineering. The inputs received from domain design are engaged in the actual construction of all content components (core assets) according to the specification of the content component model. Other software assets such as helper programs and general interface layouts are also realized.

iv) **Domain Testing:** This activity involves the certification of constructed core assets and ensuring quality assurance. The constructed domain core assets are tested, analysed and evaluated. Syntax checking, content validation, integration testing and validation testing are carried out under specified conditions to ascertain the quality of constructed core assets. This activity provides the certification for the engagement of created core assets in application engineering.

### 3.2.6 Application Engineering

Application engineering is the sub-process of PLONTOREC that is concerned with the creation of specific TIS products through the reuse of core assets created in domain engineering. The core activities of application engineering are application analysis, application design, application realization, and application testing. These activities are described as follows:

i) **Application Analysis**: The objective of application analysis in PLONTOREC is to capture the specific requirements of individual TIS products that will be created using the core assets created during domain engineering. The requirements are obtained from customers or group of customers who demand for products with specific content configurations. The input to application analysis includes domain requirement specification, domain design models, application specific interviews and surveys, and feedbacks from application testing.

ii) **Application Design**: This activity is used to determine the configuration of specific products in the product line. It involves the creation of application-specific design models and a product map based on the conceptual product line model and the core asset graphs developed in domain engineering. The configurations of specific products are obtained from the conceptual product line model while the product map documents the selected configurations for specific TIS products. It also determines the versions of specific core assets that are to be used for realization of specific TIS products. Additionally, the application analysis document is used as input to application design.

iii) **Application Realization**: This involves the generation of TIS products based on their specific configuration design using the core assets created in domain engineering. The predefined construction workflow model already specified in domain engineering is used to realize specific products in the product line. It primarily deals with product composition and assembly of variant TIS products leveraging the reusable core assets already developed in domain engineering.

iv) **Application Testing**: This is concerned with the quality assurance of generated TIS products. The TIS products are tested and validated using the domain requirements documents, application requirement documents, and domain test artifacts and application test artifacts. Some of the tests carried out include syntax checks, integration tests, validation tests etc. The output of application testing includes feedback to application analysis, feedback to domain testing, and the tested and validated TIS products.

## 3.3 FORMAL DEFINITION OF PLONTOREC

Based on the conception of PLONTOREC as a formal process that specializes software product line practices for the development of TIS, a formal and precise definition of the PLONTOREC approach is necessary. In order to achieve this, the definition of SPL given in (Prankatius et al.,

2007) has been adopted but adapted to fit the specific context of our approach, which hybridizes software product line and ontology engineering.

A PLONTOREC-generated product line $TISPL_{plontorec}$ (F, FTree, O, Pr, As, Cs) consist of:

- **A set F of features**, such that $f \in F$: F = (name, type, annotation); i.e. each feature has a name, a type and an annotation. Feature types are classified as common, optional or alternative i.e. type $\in$ {common, optional, alternative}.

  It is distinguished between the sets of

  - common features CF: = $\{f \in F \mid type(f) = common\}$ with $CF \neq \emptyset$
  - optional features OF: = $\{f \in F \mid type(f) = optional\}$
  - alternative features AF: = $\{f \in F \mid type(f) = alternative\}$.

  annotation is a description of feature in a natural language.

- The features are organized in *a* **feature tree FTree** with nodes Q. Each node $n_i \in Q$, except the root node $n_r \in Q$, has a type and contents, i.e. $n_i$ = (type, contents) with node type $\in$ { common, optional, alternative}. Every node is linked to features in F through its contents, and each feature occurs in exactly one node. The root node $n_r$ however has no corresponding feature in F as an exception. The connection between a node $n_i$ and features is given as follows:

- $$contents(n_i) = \begin{cases} f \in CF & \text{, if } type(n_i) = common; \\ f \in OF & \text{,if } type(n_i) = optional; \\ \left.\begin{array}{l} (Y, min, max) \in N \text{ with} \\ \quad Y \subseteq AF;\ min, max \in N \end{array}\right\} & \text{, if } type(n_i) = alternative; \end{cases}$$
- if $type(n_i)$ = alternative then $n_i$ must be a leaf in FTree.

- **A set of ontologies O** = $\{o_1 \ldots o_t\}$ and for $o \in O$: o = (oid, name, type, annotation, ONTO) with
  - oid is a unique identifier for the ontology $o_i$
  - a name, a type and an annotation (which is a description in natural language).
  - a representation ONTO = $\{G(V,E), \beta, \alpha, N, T\}$ with

- a set $\beta = \{c_1, \ldots c_n\}$ where $c_i \in \beta$ is a concept name;
- a set $\alpha = \{r_1 \ldots r_m\}$, where $r_i \in \alpha$ is the type of the binary; relation relating two concepts, such that $c_i$ and $r_i$ are non-null strings);
- a directed graph G(V, E) representing the ontology, where V is a finite set of vertices and E is a finite set of edges: Each vertex of V is labelled with a concept and each edge of E represents the inter-concept relationship between two concepts. Such that the label of a node $v \in V$ is defined by a function $N(v) = c_i$ that maps v to a string $c_i$ from $\beta$. The label of an edge $e \in E$ is given by a function $T(e) = r_i$ that maps e to a string $r_i$ from $\alpha$.

- **A set of TIS software products Pr** = $\{p_1, \ldots p_k\}$ and for $p \in$ Pr: p = (pid, $FTree_p$, $F_p$) where
  - *pid* is a unique identifier for the software products p.
  - $FTree_p$ is a feature tree for the product p, which is an instance of FTree. In order words $FTree_p$ is a subtree of FTree, with root node $n_r$ that is by default included into $FTree_p$;
  - a node $n_i$ of the type common in FTree has to be included in $FTree_p$ if its immediate predecessor was included, i.e. for an included node, all immediate successor of type common have to be included in $FTree_p$;
  - From a node $n_a$ of type alternative in FTree, not all alternative features must be chosen for $FTree_p$. The corresponding node $n_a'$ = (type', contents') in $FTree_p$ has type' = alternative and contents' $(n_a') = X \subseteq Y$ with $\min \leq |X| \geq \max$, i.e. at least and at most max alternative feature have to be chosen from the set Y
  - The set $F_p = CF_p \cup OF_p \cup AF_p$ denotes all common, optional and alternatives features in **$FTree_p$** which are finally in the contents of the included nodes, these features will realize the functionality of the product p.

- **A set of core assets As** := $\{a_1 \ldots a_j\}$ which are used to build a feature or a subset of features in FTree. Furthermore, for $a \in$ As: *a = (aid, content, annotation),* which means that a core asset can conceptually consist of
- a unique identifier aid;

- some content which can be for example a document, code, a model etc.
- some annotations related to the content, e.g. natural language descriptions, metadata, process specification for its construction etc.

- **A construction specification Cs** = (model, annotation, B) which specifies how to create every possible product in the TIS product line from core assets. In particular:
  - there is a model which describes the overall construction;
  - an annotation adds additional information e.g. as natural language descriptions;
  - B is the built-in-from-relation $B \subseteq P \times A$ indicating which TIS product are built using core assets.

The definition above characterizes a PLONTOREC product line as consisting of a set of features **F** that are organized in a feature tree **FTree**, a suite of ontologies **O** relevant to specific tourism objects of interest, a set of TIS products **Pr** that are built from reusing a set of core assets **As**, and a construction specification model **Cs** that defines how individual recommendation-enabled TIS products are built from the core assets leveraging specific ontologies. The components of the definition as further explicated as follows:

**Feature**s (F): The features in the product line are strictly classified into three crisp sets of: common features, optional features and alternative features. In the feature tree each node is connected to a feature, and a feature must occur in only one node. The set of features must not be empty, and there must be sufficient commonalities among the features of products in the domain to justify the need for a product line. The types for features and nodes are used to impose constraints on the choice of features for particular products.

**Feature Tree (FTree):** The feature tree is used as an organization structure for features and as a means to model feature dependencies and constraints. Also, the feature configuration of every concrete product exists as an instance of the feature tree, FTree. For the creation of such an instance, it is assumed that starting with the root which is always included in every product only those nodes from FTree are chosen which contain features that should be implemented in a product. For a specific product, the resulting instance $FTree_p$ is also a tree. The selection process of features in the nodes of the tree FTree is influenced by already chosen parents and by the type of a node (Bosch & Svahnberg, 1999).

**Ontologies (O)**: The suite of ontologies provides the basis for intelligent knowledge-based recommendation that is inherent in every product in the TIS product line. The features configuration of specific products determines the set of ontologies that are relevant to each of them. Such ontologies are also built and maintained (updated) as the feature configuration of associated TIS products evolve.

**Products (Pr):** These are the end products of PLONTOREC; the variability among products is determined by their distinct features configurations. This also determines the specific intelligent recommendation attribute inherited by such TIS products.

**Core Assets (Cs)**: Each core assets has a unique identifier, a content part and annotation part. The content part can be for example a code component, an architecture, a design diagram, a text case etc. The annotation part can be for example metadata, a natural language description with details on how to use the asset during the construction of a product, or even a more precise process specification for application engineering.

**Construction Specification (Cs)**: The construction specification is the product line scope that defines how each TIS in the product line will be constructed. It is a construction model for the whole product line. This also includes relevant annotation in natural language, which may contain non-model information of the production plan. The built-in-from-relation B defines how specific core assets are used to build specific products.

## 3.4 TOOL SUPPORT FOR PLONTOREC

In order for PLONTOREC to evolve into a standardized and repeatable practice that is industrially applicable, adequate tool-support for implementing the approach is essential. An expansive tool-support base for the execution of the PLONTOREC approach has been identified, which is drawn mainly from the fields of software engineering and ontology engineering. These tools have been classified into functional categories as follows:

- Requirements Engineering: DOORS, Accept 3600, Accompa, RequisitePro, SpeeDev, TigerPro, Raven, Gmarc etc. (http://easyweb.easynet.co.uk /~iany/other/vendors.htm#Doors);

- Software Architecture Specification and Modelling: xADL (Dashofy et al., 2001), ACME (Garlan et al., 1997); ArchStudio 4.0 (http://www.isr.uci.edu /projects/archstudio), Ménage (Garg et al., 2003) etc.
- Software design: UML-based tools (Microsoft Visio, Rational Rose, ArgoUML etc.), MDA tools (Eclipse Modelling Framework (EMF) (http://www.eclipse.org), Visual Paradigm, Enterprise Architect, AndroMDA (www.modelbased.net/mda_tools.html) etc.
- Software Programming: Integrated Development Environments (IDEs) e.g. NetBeans 5.x, Eclipse 3.x, Microsoft .Net etc.
- Ontology Development: Protégé 3.x (http://protege.stanford.edu/), OntoEdit (http://www.ontoprise.de/documents/tutorial_ontoedit.pdf), Ontolingua (Farquhar & Fikes, 1996), Ontology learning tools (Kietz et al., 2000).

Some of these tools were engaged in the case study section (chapter 4) of this thesis in detailing the practical application of the PLONTOREC approach.

## 3.5 APPLICATION SCENARIOS

The PLONTOREC approach is designed to find application in contexts where several similar TIS products are to be developed with minimal variations among them. The following are typical examples:

1. A TIS developer organization that has the responsibility of developing tourism promotion solutions for different countries. For example West Africa, Central Africa, Southern Africa or North Africa. A scenario akin to what obtains at TISCOVER AG (http://www.Tiscover.com), which has implemented tourism solution for 8 countries of the world. The TISCOVER tourism portal is a multi-lingual website which delivers exactly the same set of functionalities for all 8 countries but with unique local contents. This kind of scenario presents a good ground for product line development. Therefore, a PLONTOREC approach that will enable the development of recommendation intensive TIS by profitably exploiting the commonalities and variabilities that exist between the different countries can be adopted. The variabilities could be in terms of the language of

information presentation (e.g. English, French, Portuguese, Swahili, Hausa etc), information contents (local to each country), context information and web interfaces that reflects peculiar national identities. In the same vein, the commonalities will be in the predefined functionalities that are made available in all products in the product line.

2. Instances of an organization having to provide tourism support solution for a set of states or regional governments within a country. This kind of scenario also allows exploiting what these governments have in common and the variabilities that exist among them using the PLONTOREC approach.
3. Instances of having to implement tourism support services solutions for a particular category of tourism service providers. For example hotels, restaurants, café etc.
4. Implementing tourism promotion solutions for Destination Management Organizations (DMO). Examples include city DMO, Site DMO, Regional DMO etc.

## 3.6 VALIDATION APPROACH

In order to validate the plausibility of the proposed solution approach, a case study of product line development using the PLONTOREC approach will be reported in chapter 4 to show the practical real-life application scenario of the PLONTOREC approach. This is done to validate the hypothesis that: *The PLONTOREC approach provides an integrated process platform to enabling dependability of intelligent TIS recommendations and proactive management of dynamic user requirements in e-tourism within the context of TIS development organization.*

## 3.7 RELATED WORK

So far, to the best of our knowledge, there is no research effort in product line development that is specific to tourism that has been reported in literature. This is irrespective of the fact that there are ample evidences of the viability of a product line approach in the tourism domain. However, the Koriandol system (Balzerani et al., 2005) is a product line architecture for general web applications of which TIS is a subset. The special feature of Koriandol is that its components have variability handling mechanism built into them in contrast to other component-based systems. The flexibility of the Koriandol architecture gives the impression that it could be specialized to fit for a tourism product line context with some kind of effort if so desired.

Another related work is the CWAdvisor (Felfernig et al., 2006), which is an integrated environment for the development of knowledge-based recommender applications. The CWAdvisor is presented as a domain-independent, knowledge-based recommender environment, which assists users by giving intelligent recommendations to ensure that appropriate choices are made, additional selling opportunities are identified, and explanations provided for suggested solutions in customer-oriented sales transactions. The CWAdvisor environment can be configured for a specific application domain in order to obtain knowledge-based recommendations. The similarity between CWAdvisor and the PLONTOREC approach proposed in this thesis is that they both provide an integrated framework for the generation of knowledge-based recommendations leveraging deep knowledge of customer and products. However, the differences are as follows: 1) While CWAdvisor presents a software environment for users to obtain knowledge-based recommendations, PLONTOREC offers a software development process that enable the building of knowledge-based recommendation-intensive systems; 2) While CWAdvisor presents a customer-oriented software environment that can be configured for specific application per time, PLONTOREC presents a developer-oriented product line process for generating series of knowledge-based recommender products for the tourism domain; and 3) While CWAdvisor makes use of a in-built recommender knowledge-base consisting of product properties, customer properties (obtained at run time) and constraints, PLONTOREC makes use of formal knowledge representation ontologies that can be used by other semantic web applications; 4) While PLONTOREC is specialized for the tourism domain, the plausibility of the CWAdvisor in the tourism domain cannot be ascertained because the two application scenarios discussed in (Felfernig et al., 2006) belong to the commodity item category, where the tourism product does not belong because of its unique nature (Henriksson, 2005).

Another related work is the PLANT approach as reported in (Prankatius et al., 2007). PLANT is an acronym for Product Lines for Digital Information Products. It is a dedicated software product line development process for the generation of families of digital information products such as product lines for e-learning courses, product lines for e-books, product lines for e-news and product lines for audio-based products. The PLONTOREC approach proposed in this thesis bears similarity with the PLANT approach in that they are both specialized concepts of product line development dedicated to specific product domains. However, while the PLANT approach thrives solely on software reuse, PLONTOREC is a hybridization of software reuse and knowledge reuse concepts.

Hence, the PLONTOREC approach is unique and novel, offering a unified solution platform for enabling intelligent and dependable recommendations in TIS and managing of dynamic user requirements.

## 3.8 SUMMARY AND DISCUSSION

In this chapter the concept of Product Line for Ontology-based Tourism Recommendations (PLONTOREC) approach has been presented as an integrated solution model for the two research questions posed in this thesis. PLONTOREC is a specialized product line engineering approach for creating families of TIS products. In PLONTOREC, software reuse and knowledge reuse concepts are engaged to enable intelligent and dependable recommendations in TIS and also facilitate the evolution of such TIS products in an organized way in response to the dynamic nature of user requirements. In addition, the PLONTOREC approach provides a platform for the realization of a family of recommendation-intensive TIS without incurring undue cost overruns while in pursuit of good quality. The practical application of PLONTOREC will be discussed in the subsequent chapters.

# CHAPTER FOUR
# PLONTOREC IN PRACTICE

## 4.1 INTRODUCTION

This chapter presents details of a case study of a real-life product line development scenario where the PLONTOREC approach has been applied. The core motivation of this case study is to demonstrate the PLONTOREC approach in practice so as to validate the approach and provide a basis for its evaluation.

In order to achieve this, a SPL project was undertaken within the framework of the Software Engineering Research Group of Covenant University (SERCU). This was aimed at developing recommendation-intensive TIS platforms for an enhanced and more sophisticated approach to the promotion of tourism in the ECOWAS region of West Africa. Three countries adjudged to have the greatest tourism potentials within the geographical region were selected. These are Nigeria, Ghana and Cote D'ivoire (Ivory Coast). Currently there is not one e-tourism platform that offers intelligent recommendations about available tourism products that exist within the region (http://www.touringghana.com; http://www.viewghana.com; http://www.tourisme. com; http://www.nigeriatourism.net)

This chapter reports the practical application of the PLONTOREC process life cycle as undertaken in a case study aimed at validating the plausibility of the PLONTOREC approach.

## 4.2 PRODUCT LINE MANAGEMENT (PLM) IN PLONTOREC

The PLONTOREC was initiated with the PLM. The essence of the PLM activities was to provide the necessary managerial guide and organizational control that complements the technical aspects of the PLONTOREC approach. PLM was carried out at the end of each sub-process of PLONTOREC to determine whether the PL endeavour should continue.

### 4.2.1 Feasibility and Risk Assessment

The first thing that was done during PLM was to undertake feasibility assessment and risk assessment. In the feasibility assessment four specific pertinent questions to determine whether there exist sufficient grounds for a SPL pursuit were asked following the guideline provided in (Prankatius et al., 2007). The four questions were rated on an ordinal scale of 1-5, with 5 representing the highest level of consent and 1 the lowest i.e.

The rule used for decision-making are stated as follows:

1. PLONTOREC should only be used if the answers to four questions are all in the affirmative with a selection of options 4 or 5.
2. If options 1or 2 is selected for any then PLONTOREC should not be used.
3. If any case of indecision arises (i.e. selection of option 3), then PLONTOREC should not proceed until the question has been resolved to a selection of either 4, 5, 1 or 2.

An overview of the content of the PLM documentation produced after the exercise that captures the questions, and answers and justifications is given as follows:

i) Is a PL approach technically feasible in the case at hand?

*1) Totally infeasible; 2) Almost infeasible; 3) Not sure; 4) Almost totally feasible; 5) Totally feasible.*

**Answer**: *Totally feasible (5).*

**Justification***: The three countries share a lot in common in terms of concept and orientation of tourism. The central objective of tourism promotion platform in the countries considered is to create a platform for the discovery and increased awareness of their untapped tourism potential to boost trade and economic development. Therefore, reusable functional component models of TIS systems that are based on observed generic characteristics can be built. These components can then be subsequently customized and adapted by using carefully planned reuse and specialization schemes like code reuse, parameterisation, composition, and inheritance to suit the specific needs of each country.*

ii) Do commonalities exist among products that can be technically exploited?

*1) No known commonalty that can be technically exploited; 2) Very few commonalities exist that can be technically exploited; 3) Not sure; 4) Sufficient commonalities exist that can be technically exploited; 5) Several commonalities exist that can be technically exploited.*

**Answer**: *Sufficient commonalities exist that can be technically exploited (4).*

**Justification***: The TIS platforms have common characteristics and are intended to be functionally identical. In terms of offering information services, recommendation services about similar tourism objects such as accommodation, destination, restaurants etc.*

iii) Are the variable points in product already known?

*1) Variable points are not known or unpredictable; 2) Very few variable points are known; 3) Not sure; 4) Most variable points are known or predictable; 5) All variable points are known in advance.*

**Answer**: *All variable points are known in advance (5).*

**Justification***: The points of differences in the products are all known in advance. The content composition of each of the TIS must be local and peculiarly relevant to the particular country concerned, the web layouts must also be peculiar to each country. All service rendering components must be customized to fit specific national instances. Also services are designated as common or optional depending on state of infrastructure in each country. Information services are common, while some categories of recommendation services are optional. The optional services are those that may not be relevant to some specific TIS platforms based on the limited level of development in the country concerned.*

iv) Is there a commitment from the developer organisation to adopt a PL approach?

*1) No management support; 2) Very little management support; 3) Not sure; 4) Sizeable management support with approval; 5) Strong and encouraging management support and approval.*

**Answer**: *Strong and encouraging management support and approval (5).*

**Justification***: Since this was being undertaken as a research endeavour motivated by the desire to make quality contribution to knowledge, the support from all concerned stakeholders (student, supervisors, and collaborators) was total and encouraging.*

Risk assessment was undertaken in order to compare the expected investments in PLONTOREC with the possible benefits that can be gained from the pursuit of a PL initiative. The concern at this stage was to determine the level of predictability of demand for products in the PL and the rate at which products in the PL are expected to evolve. The typical questions asked are as follows:

In the tourism domain where PLONTOREC will be applied, how often are radical changes expected to occur?

**Answer**: *Changes occur but they are not radical in most cases.*

**Explanation**: *Changes occur regularly in the tourism domain as an advent of growth and development but are seldom radical in nature. Trends in the economy, politics and emerging technologies tend to affect the behaviour of tourists but not necessarily the status of tourism objects in many places. Hence tourism products are expected to evolve with time based on new requirements that emerge from consumer behaviour.*

Is the demand for tourism information predictable?

**Answer**: *Predictable.*

**Explanation**: *The core objectives of users are to obtain travel information about specific tourism objects such as events, destinations, accommodation etc. This will provide a basis for the adoption of a PL approach where reuse and variability schemes can be built on these predictable requirements.*

Are there strategic advantages that can be derived from taking to a PL approach?

**Answer**: *Strategic advantages are expected.*

**Explanation**: *A PL-based approach will provide a flexible platform for the evolution and maintenance of products through carefully planned reuse and versioning schemes* (Bosch & Svahnberg, 1999; Clement & Northrup, 2002).

### 4.2.2 Organization, Evaluation and Control

Based on collective experience in software development within our research group and expert opinion of a TIS development expert from Tiscover AG (http://www.Tiscover.com), we were able to do an approximate initial estimation of the cost of a PLONTOREC endeavour relative to single TIS product development based on the modified SPL economic evaluation model that we derived in Section 3.2.3. The conclusion at the end of the PLM phase including economic evaluation and configuration management encouraged the pursuit of the PLONTOREC approach.

## 4.3 ONTOLOGY ENGINEERING IN PLONTOREC

Since the central objective of the PLONTOREC approach is to enable dependable recommendations in TIS, an ontology-based approach that will provide a platform for the leveraging of deep knowledge about the tourism domain of interest was favoured. A key concept of the PLONTOREC approach is to build specific knowledge representation ontologies that relate to specific classes of tourism objects in a domain of interest that will provide a basis for obtaining knowledge-based recommendations about them. Typical examples of such tourism objects include accommodation, travel destinations, restaurants, transportation routes, and events.

In this regard, an ontology is conceived as a formal semantic representation of what is known about specific tourism objects in a particular tourism domain (e.g. national, regional, local etc.). The ontology defines all concepts about and around the tourism object that have touristy value, and the semantic relationships between the concepts. It also offers a platform for sharing and reuse of its stored knowledge within a specific tourism value chain. This connotes that the PLONTOREC ontologies are deliberate semantic descriptions of what is generally known about some real world phenomena in a domain of interest using concepts and relationship abstractions in a way that is readable by both man and machine. In the specific instance of our case study two tourism-related ontologies were developed. These are the Destination Context Ontology (DCO) and the Accommodation Ontology (AO).

### 4.3.1 The Destination Context Ontology (DCO)

The motivation for the DCO was the quest to engage a multi-dimensional approach to destination recommendation with the use of contextual information different from the 2-dimensional approach currently engaged in most of the existing recommendation platforms (Adomavicius & Tuzhilin, 2005; Adomavicius, 2005). Indeed, many of the existing DRS have placed more emphasis on user's travel activity preferences, the facilities and services, and the type of accommodation available at specific destinations without much consideration for the social attributes of such destinations. The social attributes of a destination such as the general scenery (atmosphere), security, population size, flow of traffic, behaviour of inhabitants, linguistic complexity and many other factors are very crucial to the outcome of peoples' touristy experience in most cases. This is particularly crucial in the context of many of the developing nation where there exist many social and environmental concerns as a result of underdevelopment. We believe that incorporating contextual information about the social attributes of prospective destinations can enhance the dependability of destination recommendations. Hence, the notion of the DCO is conceived as a model of knowledge representation ontology that captures contextual information about the social attributes of prospective destinations in a specific tourism domain.

#### 4.3.1.1 Using the Methonthology Approach for DCO Development

The Methonthology methodology (see section 2.6.4.1) for ontology development was selected for the development of the DCO. This was primarily due to the fact that Methontology is one of the most elaborate approaches to ontology development, with very good tool support (Gomez-Perez et al., 2004). The activities undertaken in developing the DCO are discussed in sequel.

**1. Management Activities in DCO Development**

The development of the DCO was started with the scheduling activity. During this period, we were able to set an agenda for the development of the ontology. It was agreed that the DCO should be a model of knowledge representation ontology that can be instantiated with information contents to realize knowledge bases that suits different scenarios. It was also decided that DCO should be implemented as Web Ontology Language (OWL) ontology, since it is intended to enable web-based recommender system applications. The Protégé 3.3.1 ontology development editor was selected as the implementation tool. The types of ontology support activities that were adjudged relevant to the DCO development process were knowledge acquisition, ontology documentation, and ontology evaluation.

In order to ensure that all the scheduled tasks of the ontology development process were achieved, only one person was response for the technical activities of ontology development while the non-technical aspect of data gathering was delegated to student assistants. This arrangement proved quite useful for effective control.

Also, in order to ensure good quality of the ontology, an ontology development tool that have in-built features for formalization, documentation and evaluation was selected for the ontology development.

## 2. Support Activities in DCO Development

The three types of ontology support activities undertaken in the course of developing the DCO are: Knowledge acquisition, ontology documentation and ontology evaluation. The knowledge acquisition entailed the collation of available facts about the contextual attributes of major tourism destinations within the West African sub-region. The specific focus of our case study was the social contextual attributes of tourism destinations in Nigeria, Ghana and Cote D'ivoire. Data were collected about five attributes of possible destinations which are *Weather Temperature (i.e. average daily temperature), Scenery (i.e. the layout and nature of environment), Volume of Traffic, Crime Rate,* and *Status (the size and population of the destination, and its rating in terms of level of development).* The sources of information included: National Websites, Tourism documents from National Tourism Agencies, and Geographical information extracted from literature (including maps).

The Protege 3.3.1 ontology development tool possesses in-built features for ontology documentation which was used for documenting the ontology. Also, Protégé has in-built features for evaluating the syntactic and semantic correctness of ontologies. This feature was used to evaluate the consistency of classes and semantic completeness of formal logics expressions in the ontology.

## 3. Development Activities in DCO Development

The development activities undertaken in respect of the DCO can be broadly classified as pre-development, development and post-development. These activities are described in the following sections:

### i) Feasibility and Environmental Study

The DCO ontology was intended as a kind of knowledge representation ontology to provide semantic-awareness capabilities for tourism and travel support applications in a particular domain. The environmental study process revealed that although there exist a number of standard tourism ontologies (http://protege.stanford.edu/; http://www.ontoprise.de/documents/ tutorial_ontoedit.pdf) whose structure can be emulated by the DCO, we did not find any that adequately fits into the context and content of the specific tourism interest we have in mind. This imposed the need to build the DCO from scratch. Also during this time, the suitability of building the ontology was critically examined. The trade-offs in terms of costs in time and resources together with inherent benefits were considered, with the consensus that the ontology development process should proceed.

### ii) Specification of the DCO

The purpose of the DCO is to provide a semantic representation of contextual knowledge about prospective tourism destinations in a form that can be used by web-based tourism support systems for the generation of knowledge-based tourism destination recommendations.

### iii) Conceptualization of the DCO

In the specific case study considered, the DCO is a semantic representation of the contextual information about five social attributes of the destination abstractions that exist within the West

African tourism context. Every prospective destination in West Africa can be broadly categorized into three based on the social, demographic and geographical characteristics of such locations. The three types of destinations are City, Town, and Village. Therefore, a conceptual taxonomy of destinations was developed consisting of three class abstractions: *City, Town* and *Village* with *'ISA'* relationships. The five social attributes of a tourist location that were of interest were: *Weather Temperature, Scenery, Volume of Traffic, Crime Rate,* and *Status*. These attributes were modelled as properties of a destination using *'FeatureOf'* association. Each of the five attributes consists of a set of five possible values from which values that define the characteristics of a typical destination is derived. These are given as follows:

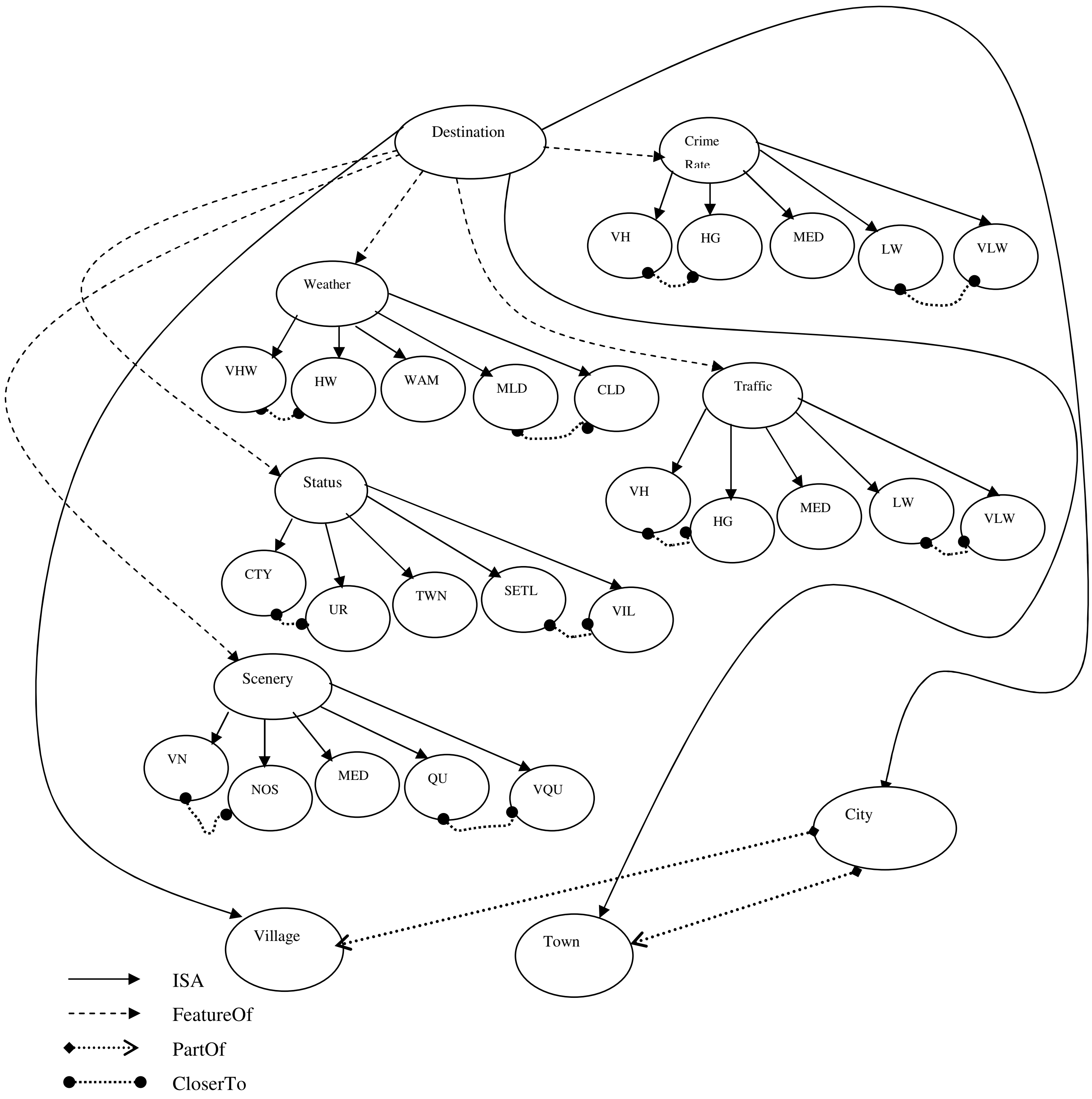
- Weather Temperature = {"Cold", "Mild", "Warm", "Hot", "Very Hot"}
- Scenery = {"Very Quiet", "Quiet", "Medium", "Noisy", "Very Noisy"}
- Volume of Traffic = {"Very Low", "Low", "Medium", "High", "Very High"}
- Crime Rate = {"Very Low", "Low", "Medium", "High", "Very High"}
- Status = {"City", "Urban", "Town", "Settlement", "Village"}

Such that, if C is a vector denoting the social attributes of a destination, then

$$\mathbf{C_{(Ibadan)} = <Mild, Medium, Medium, Low, City>}$$

Connotes that Ibadan as a destination has *Mild* weather temperature, *Medium* scenery rating, *Medium* volume of traffic, *Low* crime rate and a *City* rating in terms of metropolitan status. The semantic relationships that may exist between different instances of specific social attribute classes were modelled with the *'CloserTo'* association. For example 'Hot Weather' is specified as symmetrically closer to 'Very Hot Weather', in order to provide adequate basis for reasoning about entities represented in the ontology. The relationships between the different destination abstractions were represented using *'PartOf"* association, whereby Villages and Towns are conceived as extensions of specific City destinations. Our conceptualisation of the DCO is illustrated with the semantic graph shown in figure 4.1.

In the figure 4.1 Town, Village and City were shown to be kinds of Destination using *ISA* relationship denoted with the solid line arrow connections between the different nodes in the graph. The feature attributes of a destination such as Crime Rate have been represented using the dotted line arrow connections which represent a *FeatureOf* association. Chain-like arrow connections with two shaded circles at both ends represents *CloserTo* association, which defines the semantic closeness between two entities in the ontology. For example Very High (VH) and High (HG) are represented as symmetrically close to each other compared to Very High and Very Low. Also, dotted arrow connections with a shaded circle at one end are used to denote *PartOf* relationships that exist between the destination instances in the ontology. In this way, towns and villages are related to specific city destinations as extensions.



VHW-Very High Weather; HW – High Weather; WAM – Warm; MLD – Mild; CLD – Cold

VH – Very High; HG – High; MED – Medium; LW- Low; VL – Very Low

VN – Very Noisy; NOS – Noisy; QU – Quiet; VQU – Very Quiet

**Figure 4.1 A Semantic Graph of Concepts in the DCO**

**iv) Formalization of the DCO**

Formally, we define the DCO as a set C and a set R as follows:

C= {Destination, Town, City, Scenery, Weather Temperature, Traffic Volume, City Status, Crime Rate} and R = {“ISA”, “PartOf”, “FeatureOf”, “CloserTo”}, where each $c_i \in C$ is a concept name and $r_i \in R$ is the type of relationship relating two non-empty concepts or concept properties.

We have adopted UML (Unified Modelling Language) (Booch et al., 2000) notations to formalize our conceptualization of the DCO. UML is ideal because of the objected-oriented nature of the relationships that exist among the entities in the ontology. In figure 4.2, the UML is used as a representation language to describe the components of the DCO. Concepts in the DCO are represented as classes. ISA relationships between classes were formalized as generalizations. A generalization relationship which shows one class to be a subclass of another is modelled using the hollow arrow. PartOf and FeatureOf relationships are modelled as stereotyped UML associations. Classes are represented as rectangles, while Associations are represented as arrows. Additionally, the multiplicity of each of the Associations is shown.

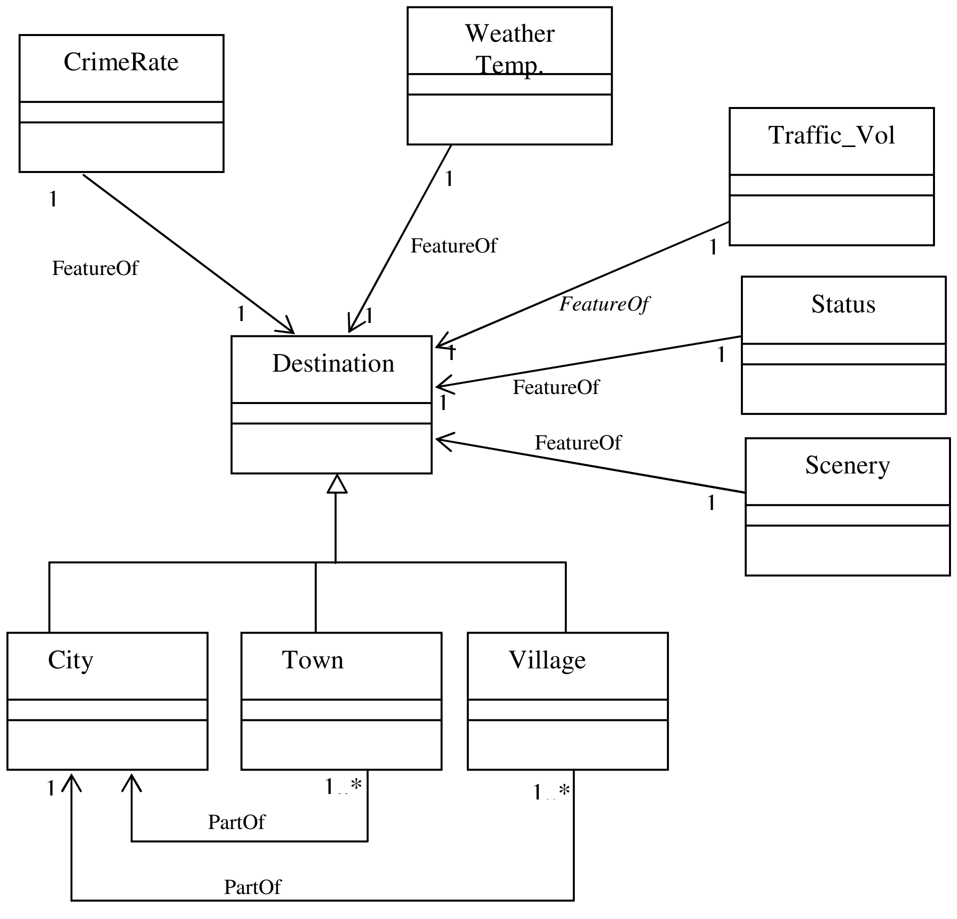


**Figure 4.2 UML Representation of the DCO**

v) **Implementing Ontology with the Web Ontology Language (OWL)**

The OWL (Ontology Web Language) most preferably referred to as Web Ontology Language (OWL) (http://www.w3.org/TR/owl-ref/) is one of the most recent and popular ontology languages. It has been adopted as a semantic web standard by the World Wide Web Consortium (W3C) (http://www.w3.org/TR/owl-ref/), for formally specifying knowledge in the web. OWL facilitates machine interpretation of Web contents in a way that is better than XML, RDF, and RDF Schema (RDF-S) by making use of additional vocabulary apart from formal semantics (http://www.w3.org/TR/owl-features/). The three types of languages are the: OWL Lite, OWL DL, and OWL Full.

**Concept axioms in OWL**

In OWL, concepts are defined using concept axioms. The *owl:Class* notation is used with a concept identifier. For example:

*<owl:Class rdf:ID="Destination"/>.*

However, this is only a simple declaration and does not give much information about the concept. Hence, concept axioms normally contain additional components to state their characteristics. Together with concept declarations, OWL contains three helpful language constructs to form concept axioms: *rdfs:subClassOf* (which indicates that a concept is described as a subset of another concept), *owl:equivalentClass* (which indicates that a concept is an equivalent of another concept), *owl:disjointWith* (which indicates that a concept has no common members with another concept).

**Role axioms in OWL**

A role axiom defines characteristics of a role, for example,

*<owl:ObjectProperty*
*rdf:ID="hasCrimeRate"/>.*

Four kinds of constructs for role axioms are supported in OWL as follows:

i) RDF Schema constructs: *rdfs:subPropertyOf*, *rdfs:domain* and *rdfs:range.*
ii) Relations to other properties: *owl:equivalentProperty* and *owl:inverseOf*
iii) Global cardinality constraints: *owl:FunctionalProperty* and *owl:InverseFunctionalProperty*
iv) Logical property characteristics: *owl:SymmetricProperty* and *owl:TransitiveProperty*

These constructs allow the definition of roles in more details. For example to capture two relation, we could have two properties *has_Status* and *is_Status_of* where *has_Status* is an inverse role of *is_Status_of.*

**Individual axioms in OWL**

Individuals in OWL are the instances of classes. Two types of individual axioms can exist in OWL ontology. These are:

i) Individual axioms about concept membership and role values

For example:

```
<City rdf:ID=”Lagos”>
has_Weather rdf:resource=”# Cold_Weather”>
< /Lagos>
```

The first line of code indicates that “*Lagos”* is an instance of concept “City” and the second line says *“Lagos”* has a role assertion *(Lagos, Cold_Weather):has_Weather*. This connote that the value of the object property *“has_Weather”* for the City instance “*Lagos”* is “*Cold_Weather”*

ii) Individual axioms about identity

For example:

```
<City rdf:ID="Lagos">
  <owl:differentFrom rdf:resource="#Ibadan"/>
</City>
```

This indicates that *"Lagos"* and *"Ibadan"* are two different instances (individuals) of the class *"City"*.

**vi) Implementation of the DCO**

The DCO was implemented as an OWL ontology using the Protégé 3.3.1 Ontology tool. The OWL ontology consists of five disjointed classes namely: *CrimeRate, Scenery, Traffic, CityStatus, Weather* and *Destination*. Three classes: Town, City, Village were defined as subclasses of the Destination class. The classes: *CrimeRate*, *Scenery, Traffic, CityStatus, and Weather* which represents the attribute features of a destination were defined as OWL Values Partition. A partition of a concept C is a set of subclasses of C that does not share common instances (disjointed classes) but cover C, that is, there are not instances of C that are not instances of one of the concepts in the partition. Hence, we have the following five values partitions defined in the ontology:

*CrimeRate = {Very_High_Crime, High_Crime, Medium_Crime, Low_Crime, Very_Low-Crime}*
*Scenery = {Very_Noisy, Noisy, Medium_Noise, Low_Noise, Very_Low_Noise}*
*Temprature = {Cold_Temp, Mild_Temp, Warm_Temp, Hot_Temp, Very_Hot_Temp}*
*Traffic = {Very_High_Traffic, High_Traffic, Medium_Traffic, Low_Traffic, Very_Low_Traffic}*
*Status = {City, Urban, Town, Settlement, Village}*

The 'FeatureOf' relationship between a Destination and each of the feature classes were modelled using corresponding OWL functional Object properties of *hasCrimeRate, hasScenery, hasTraffic, hasStatus and hasWeather* respectively. This ensures that a particular functional object property maps to only one specific subclass of the corresponding feature values partition i.e.:

$$hasCrimeRate\ (Destination) \rightarrow Low\_Crime \in CrimeRate$$

Which means that the object property hasCrimeRate must necessarily takes its value from one of values in the CrimeRate value partition. The 'CloserTo' and 'PartOf' relations between entities in the ontology were modelled as inverse and symmetric object properties. This ensures that if A is 'CloserTo' B, then B 'CloserTo' A. As such many of the subclasses in the feature value partition have relevant 'isCloserTo' property defined on them.

During application engineering specific instance of classes in the ontology (OWL individuals) were created to populate the ontology with concrete facts that pertain to specific destinations within the West African sub-region. Figures 4.3 - 4.5 are snapshots from the DCO implementation in protégé.

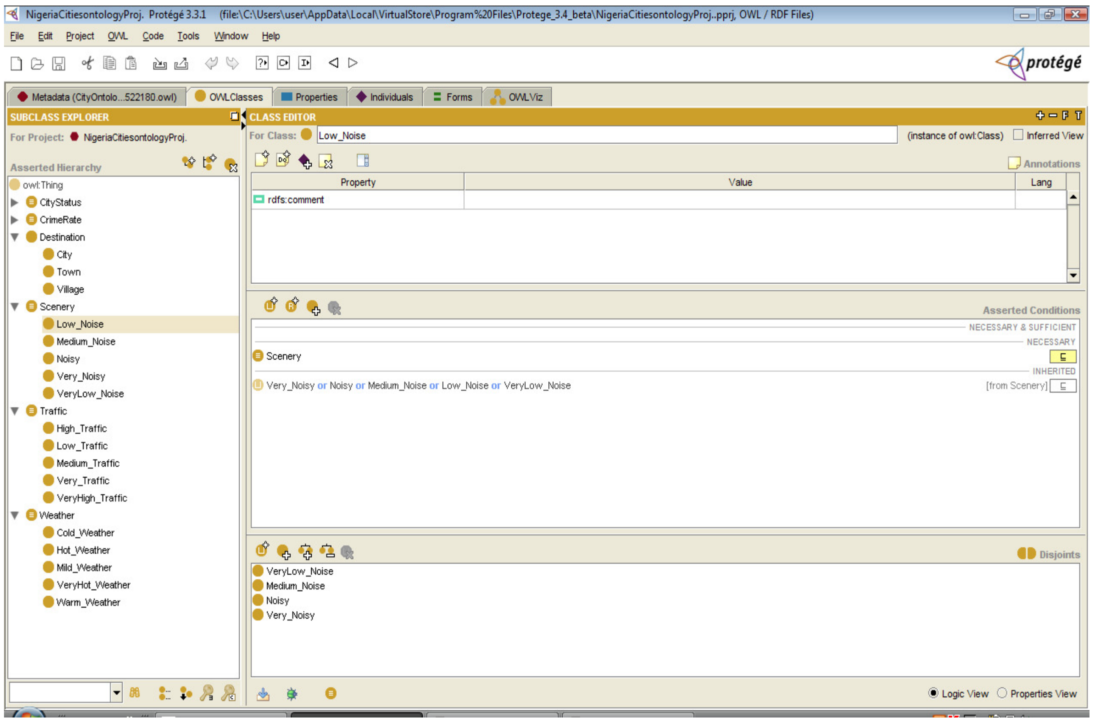


**Figure 4.3 A Snapshot Classes of the DCO in Protégé 3.3.1**

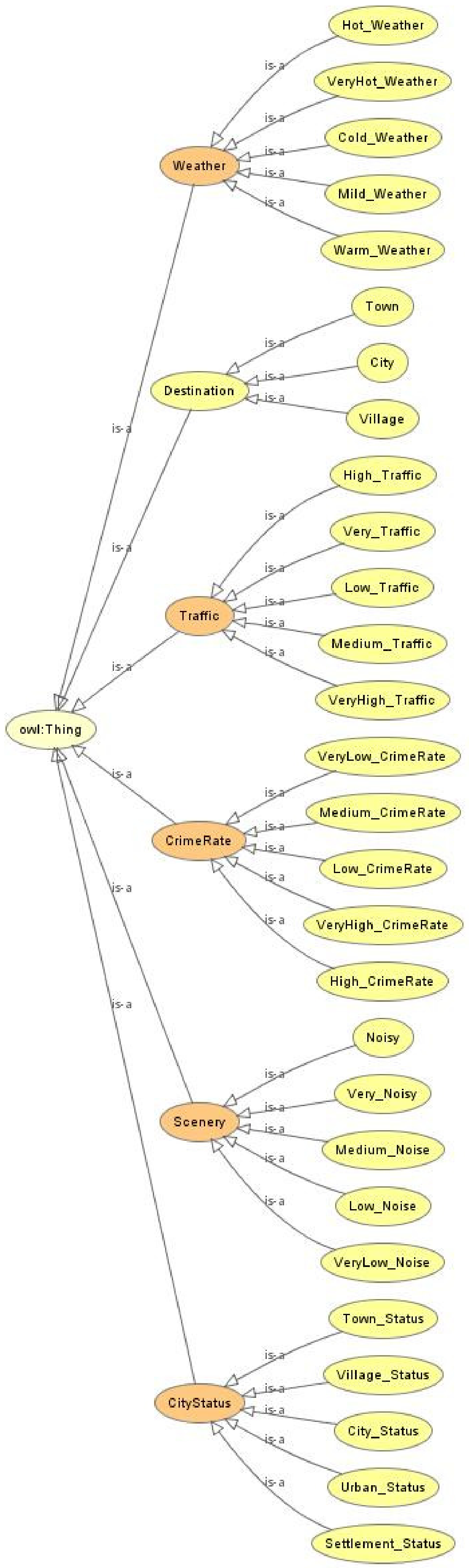


**Figure 4.4 A Graphical Model of Classes in the DCO using Protégé Visualization Tool**

## vii) OWL Representation of DCO

A fragment of the OWL representation of the ontology is shown in Figure 4.5 below. This fragment shows the description of the *City class (1), CityStatus (2),* and *CrimeRate (3)* value partitions. Also shown are the *hasCityStatus (4)* and *hasCrimeRate (5)* properties.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<rdf:RDF xmlns="http://www.owl-ontologies.com/CityOntology1203522180.owl#"
    xml:base="http://www.owl-ontologies.com/CityOntology1203522180.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <owl:Ontology rdf:about=""/>
<owl:Class rdf:ID="City">                                          (1)
      <rdfs:subClassOf rdf:resource="#Destination"/>
      <owl:disjointWith rdf:resource="#Village"/>
      <owl:disjointWith rdf:resource="#Town"/>
   </owl:Class>
    <owl:Class rdf:ID="CityStatus">                                (2)
      <owl:equivalentClass>
         <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
               <owl:Class rdf:about="#City_Status"/>
               <owl:Class rdf:about="#Settlement_Status"/>
               <owl:Class rdf:about="#Town_Status"/>
               <owl:Class rdf:about="#Urban_Status"/>
               <owl:Class rdf:about="#Village_Status"/>
            </owl:unionOf>
         </owl:Class>
      </owl:equivalentClass>
   </owl:Class>
<owl:Class rdf:ID="CrimeRate">                                     (3)
      <owl:equivalentClass>
         <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
               <owl:Class rdf:about="#High_CrimeRate"/>
               <owl:Class rdf:about="#Low_CrimeRate"/>
               <owl:Class rdf:about="#Medium_CrimeRate"/>
               <owl:Class rdf:about="#VeryHigh_CrimeRate"/>
               <owl:Class rdf:about="#VeryLow_CrimeRate"/>
            </owl:unionOf>
         </owl:Class>
      </owl:equivalentClass>
   </owl:Class>
<owl:ObjectProperty rdf:ID="hasCityStatus">                        (4)
      <rdf:type rdf:resource="&owl;FunctionalProperty"/>
      <rdfs:range rdf:resource="#CityStatus"/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID="hasCrimeRate">                      (5)
      <rdf:type rdf:resource="&owl;FunctionalProperty"/>
      <rdfs:range rdf:resource="#CrimeRate"/>
   </owl:ObjectProperty>
</rdf:RDF>
```

**Figure 4.5 OWL Representation of Classes in the DCO**

### 4.3.2 The Accommodation Ontology (AO)

The AO is a semantic representation of the attributes of the various types of tourism accommodation. It was modelled after the Harmonise ontology (Dell'Erba et al., 2002), which captures facts about accommodation types and events in the European tourism domain. The aim of the AO is to facilitate generation of knowledge-based recommendation for informed decision in seeking tourism accommodation in a particular domain. It is a knowledge base of facts about available types of tourism accommodation resources and artifacts within a particular tourism domain. The AO captures explicit details about five specific attributes of tourism accommodation types (e.g. hotel, guest house, hostel, chalet etc.) in order to enable dependable knowledge-based recommendations. These are 1) *Services*: the description of kinds of services rendered in a place of tourism accommodation; 2) *Gastro*: the profile of eateries, cuisines or restaurant near a tourism accommodation l; 3) *Attraction*: special attractions within or near a hotel; 4) *State*: province or region where an accommodation is located; and 5) *Facilities*: physical facilities available in the tourism accommodation.

#### 4.3.2.1 Using the Methonthology Approach for AO Development

Similarly, the Methonthology methodology for ontology development was used for developing the AO just like the case with the DCO. The activities undertaken in developing the AO are discussed in sequel.

**1. Management Activities in AO Development**

The management activities in AO development began with the scheduling activity, where the decision to implement the AO as a knowledge representation OWL ontology for facts about tourism accommodation was made. The Protégé 3.3.1 ontology development editor was selected as the implementation tool for the AO. The Ontology support activities that were identified as relevant to the AO development process were knowledge acquisition, ontology documentation, and ontology evaluation. During this period roles were also assigned to a number of staff assistants on the projects to alleviate the demands of knowledge acquisition and data gathering.

## 2. Support Activities in AO Development

The three types of ontology support activities undertaken in the course of developing the AO are: Knowledge acquisition, ontology documentation and ontology evaluation. The knowledge acquisition involved gathering data on available tourism accommodation resources within our domain of West African sub-region. Three groups of data collectors were simultaneously engaged to dig out facts on various types of accommodation in the three West African countries of interest (Nigeria, Ghana and Ivory Coast). The sources of information included: DMO Websites, documents from National Tourism Agencies, product brochures obtained from operators and available information on operators' websites.

The documentation and syntactic evaluation of the ontology was undertaken using the Protégé 3.3.1 ontology development tool.

## 3. Development Activities in AO Development

The details of the pre-development, development and post-development activities that are associated with the AO are described as follows.

### i) Feasibility and Environmental Study

The structure of the AO emulated the Harmonise Ontology (Dell'Erba et al., 2002). The difference is mainly in the scope of the information covered and the content. While Harmonise contains facts about events and accommodation types, the AO is limited to facts on accommodation. During this period, a decision to build the AO was taken, because it was considered a feasible and worthwhile endeavour with obvious attendant benefits when executed.

### ii) Specification of the AO

The purpose of the AO is to provide a semantic representation of knowledge about the specific attributes of available types of tourism accommodation within a domain. This is intended to enable semantic web applications that need such for the generation of knowledge-based tourism destination recommendations or semantically enabled tourism query processing.

### iii) Conceptualization of the AO

Five attributes of tourism accommodation types were considered most crucial drawing knowledge gained from literature. These are: 1) *Services*: which are the various kinds of services rendered by a place of tourism accommodation; 2) *Gastro*: which defines types of eateries, cuisines or restaurant near a tourism accommodation; 3) *Attraction*: which describes the types of special attractions within or near a hotel; 4) *State*: which describes the province, city or region where an accommodation is located; and 5) *Facilities*: which describes the types of physical facilities available for the comfort of guests in the tourism accommodation. Different types of accommodation types were also identified which includes: hotel, rented apartments, guest house, luxury hotel, mini-hotel etc.

Based on these observations, a conceptual taxonomy of accommodation was developed consisting of an Accommodation superclass and five disjointed classes (Attraction, HotelServices, Facilities, Gastro, and State) which represent the feature attributes of every instance of the Accommodation class. The Accommodation class is abstracted as an exhaustive decomposition of all available accommodation types which are its subclassses. The subclasses of Accommodation are: Hotel, Hostel, GuestHouse, WholeHouse, Chalet, and LuxuryHotel.

The subclasses of Accommodation are linked to it through *'ISA'* relationship, while each of the classes representing an accommodation attribute is linked to the Accommodation class via *'FeatureOf'* relationship.

In the figure 4.6, Hotel, Hostel, GuestHouse, WholeHouse, Chalet, and LuxuryHotel were shown to be kinds of Accommodation using *ISA* relationship denoted with the solid line arrow connections between the different nodes in the graph. The feature attributes of an

Accommodation class such as Services was represented using the dotted line arrow connections which represent a *FeatureOf* association.

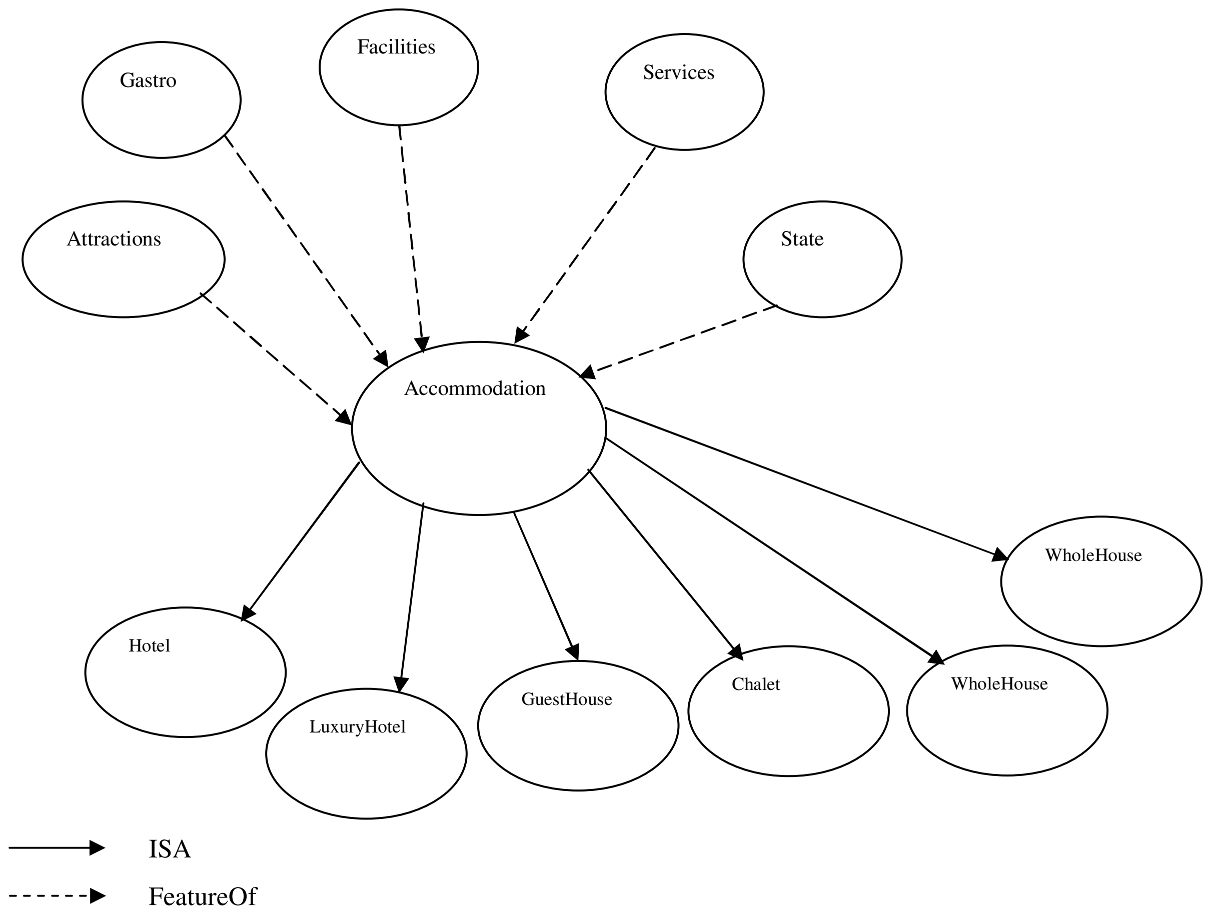


**Figure 4.6 A Semantic Graph of Concepts in the AO**

## iv) Formalization of the AO

Formally, we define the components of AO as a set C and a set R as follows:

C= {Accommodation, Hotel, Hostel, GuestHouse, WholeHouse, Chalet, LuxuryHotel, Gastro, Services, Facilities, Attractions, State} and R = {“ISA”, “FeatureOf”}, where each $c_i \in C$ is a concept name and $r_i \in R$ is the type of relationship relating two non-empty concepts or concept properties.

Just as in the DCO, we engaged the UML (Unified Modelling Language) (www.omg.org/technology/documents/formal/unifiedmodelinglanguage.htm) notations to formalize our conceptualization of the AO as shown in figure 4.7; the UML is used here as a representation language to describe the components of the AO.

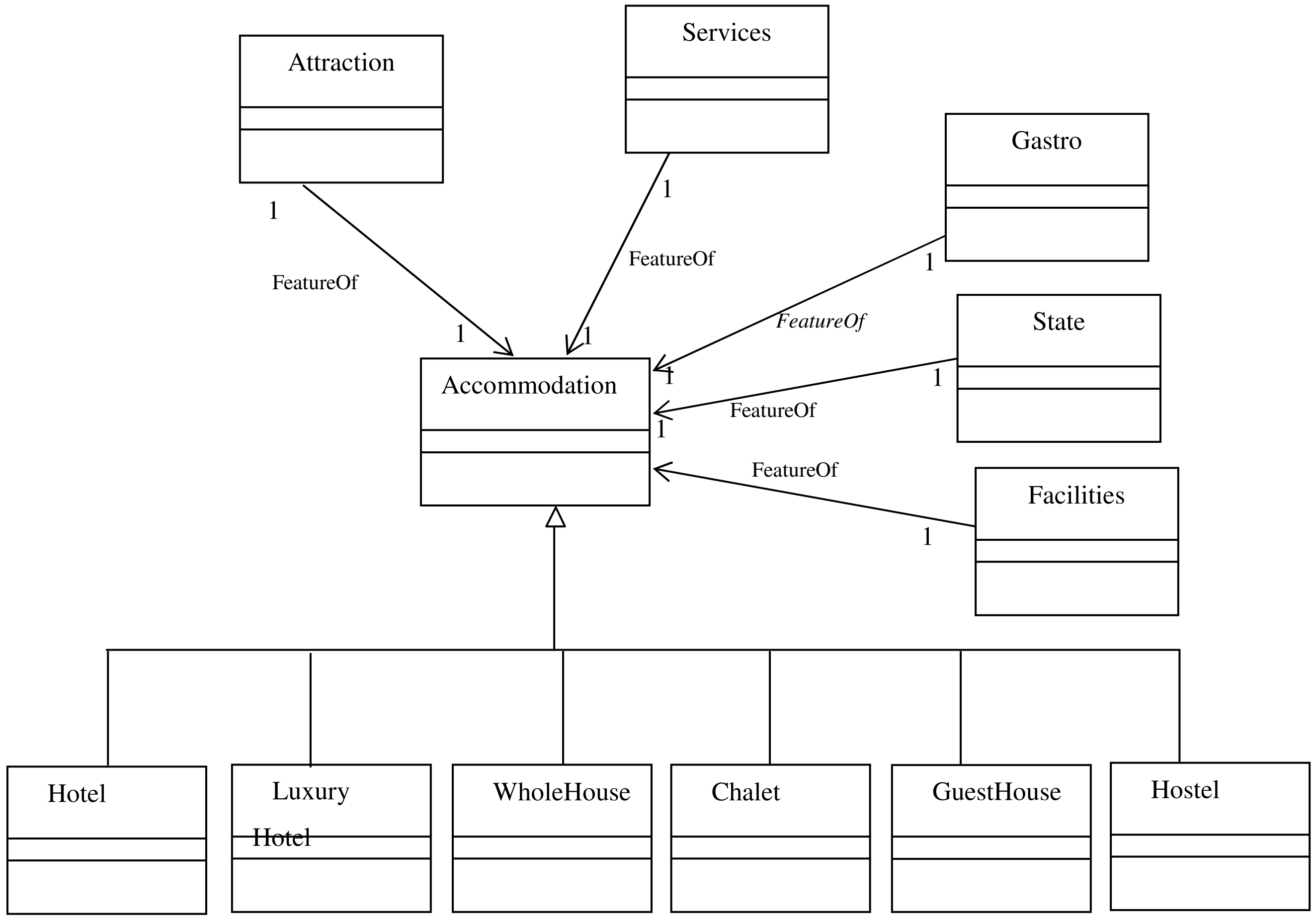


**Figure 4.7 UML Representation of the AO**

### v) Implementation of the AO

The AO was implemented as an OWL-KR ontology using the Protégé 3.3.1 Ontology Editor. The OWL ontology consists of six disjointed classes namely: *Accommodation, Attraction, Facilities, Services, Gastro* and *State*. Six classes: *LuxuryHotel, Hotel, GuestHouse, Hostel, WholeHouse* and *Chalet* were defined as subclasses of the *Accommodation* class. The classes:

*Accommodation, Attraction, Facilities, Services, Gastro* and *State* which are the product features of a tourism accommodation were related to the *Accommodation* class by using OWL object properties.

The object properties in the ontology are *hasServices, hasGastro, hasAttraction, hasState,* and *hasFacilities*. While *hasState* was defined as a functional property that maps an accommodation type to a particular state in the country, all the other object properties are non-functional properties, that have their maximum cardinality set to 20. This ensures that up to 20 different object property values can be specified for each of the attributes classes of *Attraction, Facilities, Services, Gastro* for every instance of an *Accommodation* class. During application engineering specific instance of classes in the ontology (OWL individuals) were created to populate the ontology with concrete facts that pertain to specific destinations within the West African sub-region. Figure 4.8 are snapshots from the implementation of the AO using protégé 3.3.1.

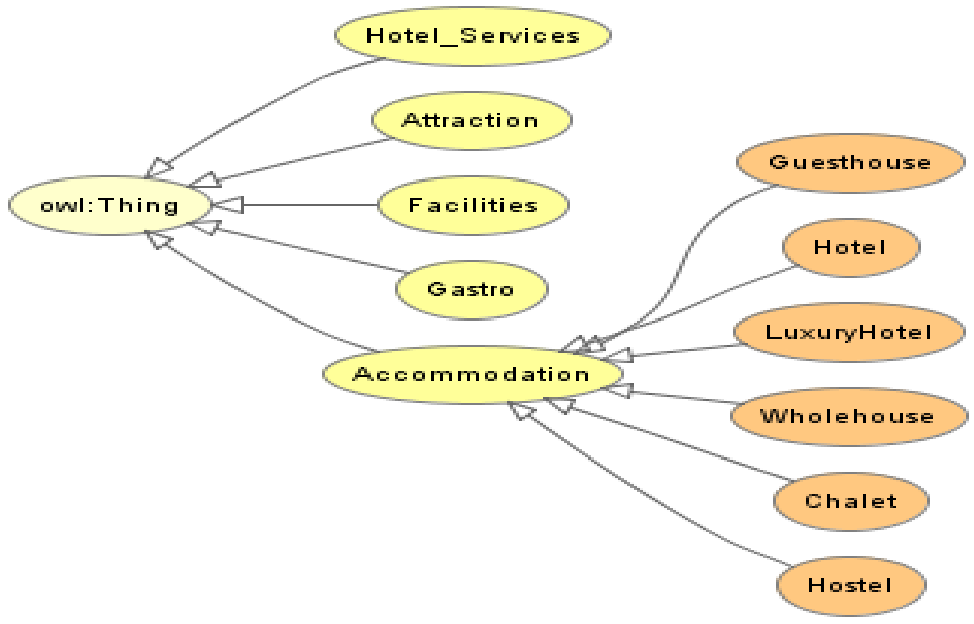


**Figure 4.8 A Snapshot AO Classes in Protégé 3.3.1 OWLViz - Tab**

**vi) OWL Representation of the AO**

A fragment of the OWL representation of the ontology is shown in Figure 4.9 below. This fragment shows the description of the Hotel *(1), Gastro (2),* and *Facilities (3)* classess. Also shown are the *hasGastro (4)* and *hasFacilities (5)* properties.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<rdf:RDF xmlns="http://www.owl-ontologies.com/Ontology1203659989.owl#"
     xml:base="http://www.owl-ontologies.com/Ontology1203659989.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <owl:Ontology rdf:about=""/>
<owl:Class rdf:ID="Hotel">                                                        (1)
      <owl:equivalentClass>
         <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
               <owl:Restriction>
                  <owl:onProperty rdf:resource="#hasAttraction"/>
                  <owl:maxCardinality rdf:datatype="&xsd;int">20</owl:maxCardinality>
               </owl:Restriction>
               <owl:Restriction>
                  <owl:onProperty rdf:resource="#hasFacility"/>
                  <owl:maxCardinality rdf:datatype="&xsd;int">20</owl:maxCardinality>
               </owl:Restriction>
               <owl:Restriction>
                  <owl:onProperty rdf:resource="#hasGastro"/>
                  <owl:maxCardinality rdf:datatype="&xsd;int">20</owl:maxCardinality>
               </owl:Restriction>
               <owl:Restriction>
                  <owl:onProperty rdf:resource="#hasServices"/>
                  <owl:maxCardinality rdf:datatype="&xsd;int">20</owl:maxCardinality>
               </owl:Restriction>
            </owl:intersectionOf>
         </owl:Class>
      </owl:equivalentClass>
      <rdfs:subClassOf rdf:resource="#Accomodation"/>
      <owl:disjointWith rdf:resource="#LuxuryHotel"/>
      <owl:disjointWith rdf:resource="#WholeHouse"/>
      <owl:disjointWith rdf:resource="#Chalet"/>
      <owl:disjointWith rdf:resource="#Hostel"/>
      <owl:disjointWith rdf:resource="#Guesthouse"/>
   </owl:Class>
<owl:Class rdf:ID="Gastro">                                                       (2)
      <owl:disjointWith rdf:resource="#State"/>
      <owl:disjointWith rdf:resource="#Attraction"/>
      <owl:disjointWith rdf:resource="#Facilities"/>
      <owl:disjointWith rdf:resource="#Hotel_Services"/>
   </owl:Class>
<owl:Class rdf:ID="Facilities">                                                   (3)
      <owl:disjointWith rdf:resource="#State"/>
      <owl:disjointWith rdf:resource="#Attraction"/>
      <owl:disjointWith rdf:resource="#Gastro"/>
      <owl:disjointWith rdf:resource="#Hotel_Services"/>
   </owl:Class>
 <owl:ObjectProperty rdf:ID="hasFacility">                                        (4)
      <rdfs:domain rdf:resource="#Accomodation"/>
      <rdfs:range rdf:resource="#Facilities"/>
   </owl:ObjectProperty>
 <owl:ObjectProperty rdf:ID="hasGastro">                                          (5)
      <rdfs:domain rdf:resource="#Accomodation"/>
      <rdfs:range rdf:resource="#Gastro"/>
   </owl:ObjectProperty>
 </rdf:RDF>
```

**Figure 4.9 OWL Representation of Classes in the AO**

### 4.3.3 Content Evaluation of PLONTOREC Ontologies

In the course of our ontology engineering endeavour, detailed content evaluation of the DCO and AO were undertaken. The goal of content evaluation of the two ontologies was to detect inconsistencies or redundancies that may exist in the ontology before they are engaged in TIS applications development. According to Corcho et al. (2004), the application of content evaluation techniques should take place during the entire ontology life-cycle, as well as during the entire ontology-building process (Corcho et al., 2004). Also such evaluation procedure should support the evaluation of concept taxonomies, properties, relations and axioms. This is because there is a compelling need for ontologies to satisfy stronger requirements such as: correctness, consistency, completeness, and conciseness) as it migrates to the commercial domain. Therefore ontology evaluation ontology tools that can prevent possible anomalies in ontologies, both in the research area and in the industrial area, are needed in order to provide reliable ontology-based systems (Hartmann et al., 2005).

However, most of the well-known ontology development tools like OILed (Bechhofer et al., 2001), OntoEdit (Sure et al., 2002), and Protègé, (http://protege.stanford.edu/) support content evaluation mainly in the form of circularities detection, but lack the capability to identify inconsistencies and redundancies in concept taxonomies. This brought about the need for us to engage a complementary ontology evaluation tool such as ODEval (Corcho et al., 2004).

#### 4.3.3.1 Validation and Evaluation with ODEval

ODEval (Corcho et al., 2004) is a tool that is mostly used to evaluate concept taxonomies of RDF(S), DAML+OIL, and OWL from a knowledge representation point of view. It is a complement for ontology parsers and ontology platforms. ODEval has capability to automatically detect possible problems in ontology concept taxonomies as it relates to inconsistency (circularity issues and partition errors), and redundancy problems. This tool is used when the development of ontologies has finished. ODEval uses a set of algorithms based on graph theory (Goodaire & Parmenter, 1998). An ontology concept taxonomy is considered as a directed graph *G (V, A)*, where *V* is a set of vertex and *A* is a set of directed arcs. For each language and each type of problem, the elements in the sets *V* and *A* are different. It is used to detect possible anomalies from a knowledge representation point of view. Hence, it is used to help ontology developers in designing ontologies and helps ontology engineers to reuse ontologies.

Some of key problems that may exist when taxonomic knowledge is modelled are given as follows:

- **Circularity problems**: This is when a class is defined as a specialization or generalization of itself. A circularity error is classified as being at distance zero if a class generalizes or specializes itself. A Circularity error of distance one occurs if there exist a *Class_A* that is defined as a subclass of *Class_B* and *Class_B* is also defined as a subclass of *Class_A*. Thus circularity errors are classified based on the number of relations. ODeval looks for cycles in the graph *G (V,A )* that represents an ontology.
- **Partition problems**: This involves detecting errors in disjoint groups: an error occurs in a disjoint decomposition or a partition, formed by the classes {$class\_A_1$, $class\_A_2$,…,$class\_A_n$}, if there are common elements in two or more branches of the partition.
- **Redundancy problems**: This occurs when for each class *class_A* in *V* and each arc $r_i$ in *A* whose origin is *class_A*, taking $r_i$ out of *A* and check if this change affects the set of elements reachable from the *class_A*. If no change, this means at least one of the $r_i$ is dispensable. In this way, at least one problem can be found.

Figure 4.10 gives a preview of some of the potential problems that might appear in taxonomies.

The ODEval tool was used on both the DCO and AO ontologies immediately after their development using the Protégé tool. Immediately after developing each of the DCO and AO ontologies, their RDF codes contained in their respective files OWL files were ported to ODEval for content evaluation. We were able to get the two ontologies successfully parsed and certified for consistency and lack of redundancy using the ODEval tool. Snapshots of results obtained from the evaluation procedure are shown in the Appendix of this thesis.

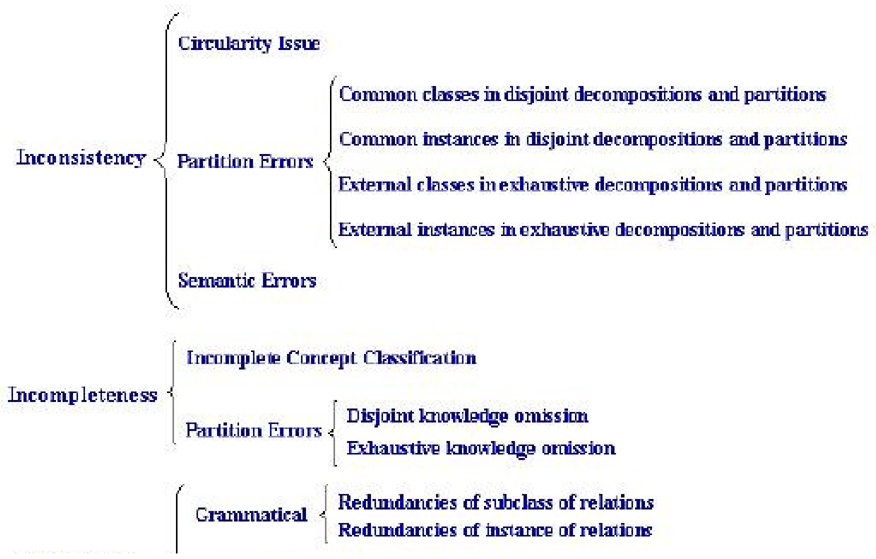

Inconsistency
Circularity Issue
Partition Errors
Common classes in disjoint decompositions and partitions
Common instances in disjoint decompositions and partitions
External classes in exhaustive decompositions and partitions
External instances in exhaustive decompositions and partitions
Semantic Errors
Incompleteness
Incomplete Concept Classification
Partition Errors
Disjoint knowledge omission
Exhaustive knowledge omission
Grammatical
Redundancies of subclass of relations
Redundancies of instance of relations

**Figure 4.10 Potential problems that might appear in taxonomies copied from (Gomez-Perez et al., 2004)**

## 4.4 DOMAIN ENGINEERING IN PLONTOREC

Domain engineering is the sub-process of software product line engineering that is concerned with the construction of all reusable software assets that are used for building the variant products in the product line. Domain engineering starts with domain analysis, whereby the domain that is relevant for the product line is surveyed and basic requirements are collected. The results are used in domain design to create abstract models that encapsulates the requirements of all products in the product line. These models belong to different layers of abstraction, ranging from architecture design to component design. Content components and other implementation-related core assets are implemented in domain realization, tested and certified in domain testing before they are later used for product composition in application engineering. The iterative and dependency characteristics of domain engineering sub-processes are shown in figure 4.11.

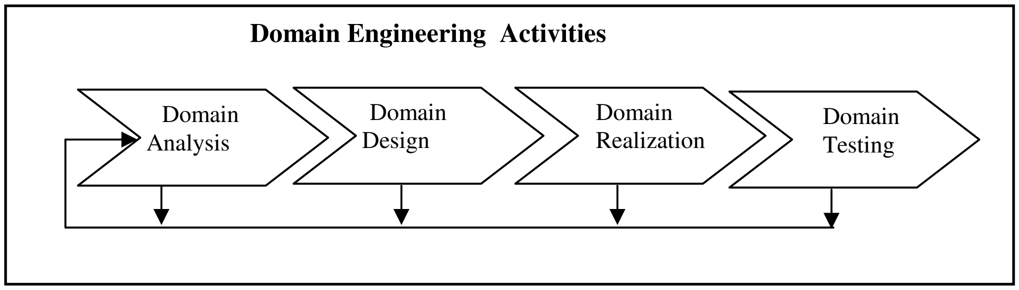


**Figure 4.11 Sub-processes of Domain Engineering**

### 4.4.1 Domain Analysis

The essence of the domain analysis process is to systematically capture the requirements for all TIS products that will be eventually built in the product line. Therefore, during this time domain requirements were gathered. The sources of information included the websites of national tourism authorities of Nigeria, Ghana and Cote D'ivoire (http://www.nigeriatourism.net; http://www.touringghana.com; http://www.tourisme.com), tourism information about these countries gathered from web sources, information documents on tourism obtained from national tourism agencies of the countries involved, and interaction with tourism experts.

The domain analysis process produced a document that contained natural language descriptions for scoping, and the necessary requirements for developing TIS for these countries. Details are presented in the following sections.

#### 4.4.1.1 Domain Requirements Engineering

Domain requirements engineering extends traditional requirements engineering by capturing the commonalities and variabilities among software products in a product line in addition to individual product requirements. The domain requirement engineering was initiated with domain scoping, which is discussed next.

#### 4.4.1.2 Domain Scoping

The essence of domain scoping is to define the limit of membership of a product line. It specifies what is in the product line and what is not. The three activities undertaken during domain scoping are:

- **Portfolio Scoping**: This helped to identify which products have sufficient commonalities to be part of the product line. In our case study, the products of the product line were TIS that are expected to particularly offer knowledge-based recommendations on specific tourism objects such as destination, accommodation, travel packages, entertainment, restaurants etc.
- **Information domain scoping**: This was used to identify the domain for the product portfolio. The core functionalities that were relevant in the domain were also identified. In our case study, the domain of consideration is tourism, with particular focus on three countries in the West African region. These are Nigeria, Ghana and Cote D'ivoire.
- **Asset scoping**: This was used to identify the reusable parts that can be used to realize different functionalities. The core assets that were identified as relevant to the three national tourism domain considered are tourism recommender systems, web layout templates, OWL ontologies, database query component, and database content builder component.

### 4.4.1.3 Extracting the Base Requirements of the Product Line

The Application-Requirements Matrix approach was used to capture the commonality and variability among TIS products in the product line (Pohl et al., 2005). An Application-Requirements Matrix is a *n x m* matrix of identified application-requirements and individual software products of a product line. The column headings of the Application-Requirements Matrix are the software products while the row headings are the application requirements. At the intersection of a row and a column, a mark is used to indicate if a particular requirement is mandatory for a particular application. The requirements that have the mandatory mark in every cell of one row are the common requirements, whereas the others are variable requirements.

Based on interaction and tourism information gathered from the different sources consulted, the desired features in the TIS products of the product line are shown in Table 4.1. The requirements that are marked '*' represents optional features.

**Table 4.1 Application-Requirements Matrix obtained from Domain Analysis**

| Requirements | **Nigeria-TIS** | **Ghana-TIS** | **Ivory Coast-TIS** |
|---|---|---|---|
| **Destination Recommendation** | X | X | X |
| **Accommodation Recommendation** | X | X | X |
| **Restaurant Recommendation** | * | * | * |
| **Travel Recommendation** | * | * | * |
| **Web Layout** | X | X | X |
| **Language Translation Feature** | | | X |
| **Database Query Feature** | X | X | X |
| **Database Content Update** | X | X | X |

The common requirements that pertain to all three TIS products in the product line as obtained from the Application-Requirement Matrix are listed as follows:

**Destination Recommendation**: to offer guide to users on choice of destination to visit based on their individual tourism activity preferences.

**Accommodation Recommendation**: to offer guide to users on available accommodation types based on their preferences in terms of desired services, attractions, location, and available facilities.

**Restaurant Recommendation**: to offer recommendations to users on available restaurants and gastronomy preferences, spending budgets, location etc. This is specified as an optional requirement.

**Travel Recommendation**: to offer appropriate travel package recommendation to users based on their personal preferences on multiple tourism objects such as accommodation, travel activities, gastronomy, flight etc. This is specified as an optional requirement.

**Web Interface Layout**: This is a web-based GUI that is needed to access the features of the TIS.

**Database Query Feature**: This feature is to enable the information search and querying of tourism information based on stored contents.

**Database Content Update**: This feature is to enable users to upload information about new or existing tourism assets that they know about if such information do not previously exist. This will help to populate the tourism asset database of a domain with usable data and current information.

**Language Translation Feature:** This feature enables the multi-lingual translation of web contents into alternative languages of English and French.

The information obtained from the Application-Requirement Matrix provided the basis for the formulation of a reference architecture for the TIS product line.

### 4.4.2 Domain Design

The domain design phase of the case study was executed using the Feature Oriented Domain Analysis (FODA) method approach (Kang et al., 1990). FODA begins with domain scoping, which is followed by domain modelling and architecture modelling. Feature modelling is a widely used technique to represent the commonality and variability of product variants on a feature level in an implementation-independent way. Feature models describe the possible configurations with all available options and constraints that are considered relevant to a product line (Kang et al., 1990; Gomaa, 2005).

### 4.4.2.1 Feature Modelling of the Product Line

Feature modelling is the main activity of the domain modelling phase, where the characteristics that are visible to the end-user are abstracted as features. Features are categorized into functional features (i.e., functions of the application seen by the user), operational features (related to the operation of an application from the user's perspective), and presentation features (related to the presentation of information to users). All features are represented in a feature model which consists of (Kang et al., 1990):

1. **A feature diagram**: which graphically depicts a hierarchy of features, and has a distinguished root. The nodes of the diagram other than the root represent features which can be mandatory, optional (drawn with a circle above the feature name), or alternative (drawn as children of the same parent feature, with an arc intersecting the connecting lines). The feature diagram presents a view of all relevant features in a domain which can be eventually included into a product variant. In a feature diagram, there is no notational distinction between functional, operational, or presentation features.
2. **Composition rules**: which additionally express dependencies between features: mutual dependency ("requires") or mutual exclusion ("mutex-with"). Composition rules are additional constraints limiting the choice of features.
3. **A record of trade-offs, rationales, justifications**: This offers guidance during the selection of features.
4. **A record of system features**: This keeps record of which features are used in which systems with which values. This bears some similarities with a product map.

Each feature in a feature model must have a distinct name that is also included in a domain terminology dictionary which is used throughout the modelling phase, and which describes the meaning of features. The validity of a feature model, in terms of how well it captures all relevant features and feature combinations, is usually verified by domain experts. The feature model of the TIS product line is shown in figure 4.12.

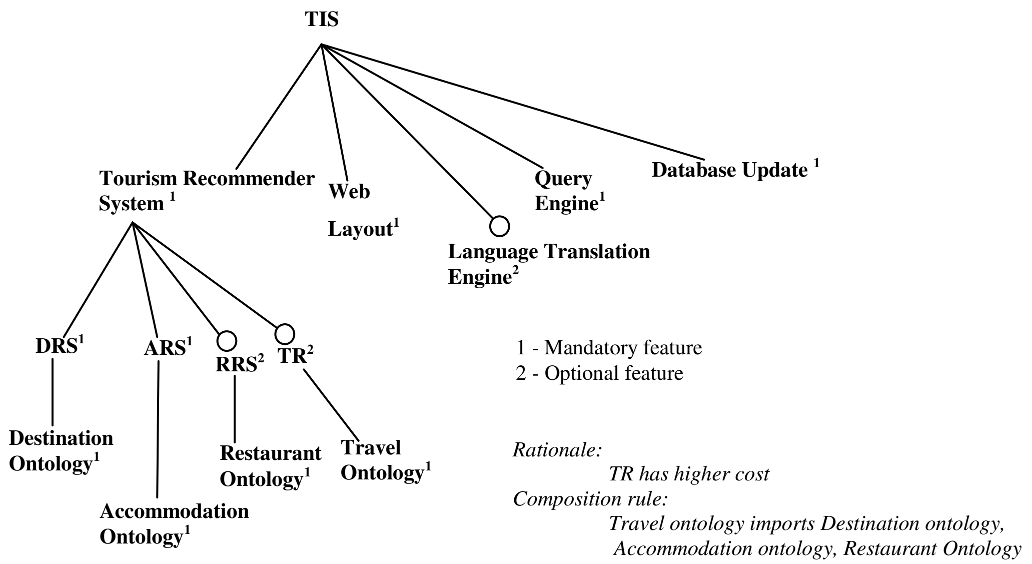


**Figure 4.12 FODA feature Model for TIS Product Line**

In Figure 4.12 Composite features (e.g., "Tourism Recommender System") consist of several other features, while atomic features are not subdivided further (e.g. "Web Layout"). A line in the diagram models the "requires" relationship between the possible features of a TIS software, and in addition every feature has an imaginary flag (not shown) to mark if it is chosen for a product or not. If a parent feature is not chosen in an instance, then all its children cannot be chosen. The root "TIS" is, by definition, chosen for every product configuration. The mandatory features "DRS", "ARS" have to be implemented in every product. The features "Language Translation Engine", "RRS" and "TR" are optional, i.e., they are included only if desired by a customer. An additional composition rule specifies that when the optional feature "TR" is chosen, then its subfeature "Travel Ontology" must import other ontologies. A rationale provides a notification guide on the relative cost of choosing the "TR" optional feature.

#### 4.4.2.2 Reference Architecture for the Product Line

Based on the outcome of domain analysis and feature modelling, a reference architecture which is called Tourism Information System Product Line Architecture (TISPLA) was formulated. The TISPLA presents a logical view of the basic building blocks of all products in the product line, as well as the commonality and variability that exist among the products of the product line. It is the foundational template from which all products in the PL evolve and embraces all possible configurations of products that are realizable in the PL.

In figure 4.13 an architecture diagram (Ogush et al., http://www.architecture.external.hp.com) is used to show the structural elements of the TISPLA. The architectural diagram gives a logical view of the components in the PL and their interconnection paths using the UML class diagram. The TISPLA is represented as a composite aggregation of all its components. Components are modelled by the UML class symbol, while interconnections between components are modelled by associations. The associations represent direct connections between components. The direction of the association shows which component initiates the communication. Components and associations are also stereotyped to show the type of component (common, optional) or a connection. In the figure specific knowledge-based recommender systems that are enabled by relevant ontologies were designated as common or optional features of the TISPLA. The web interface is also shown as an aggregation of the information query, content builder and tourism recommender system components.

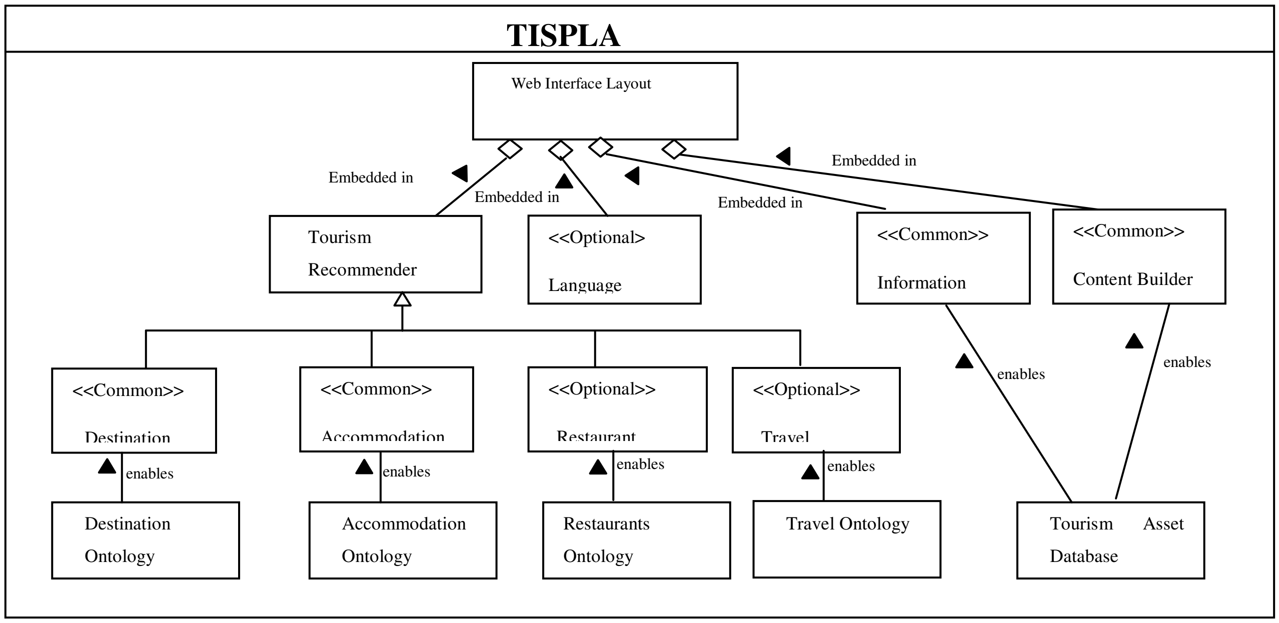


**Figure 4.13: The Architecture diagram of the TISPLA**

#### 4.4.2.3 Architecture Modelling and Specification of the TISPLA

In a product line, the dominant core asset is the reference architecture of the product line, which is used at every product instantiation. Hence, the need for the engagement of a formal mechanism to precisely define, evaluate and document the software product line architecture. The formal specification of architecture has the potential to improve both quality and productivity in the software development process because it facilitates the promotion of insight and understanding of system properties at a higher level of abstraction than at module and codes levels. It provides a basis for formal reasoning and a rigorous analysis of critical non-functional system properties like modifiability, flexibility, reliability, extensibility and reusability (Daramola et al., 2008).

The TISPLA was modelled as a layered style architecture using the Archstudio 4 (Garg et al., 2003) architecture modelling tool, while an architecture description language (ADL) xADL 2.0 was used to formally describe its components. ADLs are a class of formal specification languages that are equipped with formal constructs for describing the elements of software architecture such as components, connectors and their configurations. The xADL 2.0 that was used for the specification of the TISPLA is a highly extensible XML-based ADL embedded within the Archstudio 4 modelling framework. It is preferred to other ADLs because it makes a logical distinction between design-time (architectural prescriptions) and run-time (architectural descriptions) state of a system in contrast to the other ADLs that assume the two to be the same. Also, xADL has a rich tool support and a highly extensible nature that allow users to independently extend its XML-based schema to suit their preferred semantic contexts. Additionally, it provides support for product line modelling and model-based system instantiation (Dashofy et al., 2001).

**i) C2 style Model of the TISPLA**

The TISPLA was modelled as an aggregation of concurrent components tied together by message routing devices, which are the connectors (see Figure 4.14). Request is sent from the client layer (Web Layout Component) at the bottom and notification from the top after a response has been constructed. The tourism recommender components in the architecture leverage the semantic knowledge representation at the semantic ontology layer in order to improve the quality and dependability of recommendations to requesting clients. The rule of interaction among the components of the TISPLA follows the c2 architectural style (Whitehead Jr. et al., 1995). The choice to model the TPLA as c2 style architecture using Archstudio 4 also has the advantage of automatically generating the equivalent formal description of the architecture using the xADL 2.0 language. The c2 style imposes the principle of substrate independence on the components of the architecture in which a component in the architecture hierarchy is only aware of the component above it. This enables high substitutability that offers a boost for modifiability and extensibility, especially in a product line context as it provides a platform for the dynamic evolution of products. The c2 style also supports the use of parameterizable components thereby facilitating the reusability of the architecture. Customization of components is also possible based on the c2 style model of the TISPLA (Whitehead Jr. et al., 1995). Thus the c2 style of the TISPLA gives an insight into a measure of elasticity and extensibility of the architecture, which makes it potentially suitable as a reference architecture for the TIS product line.

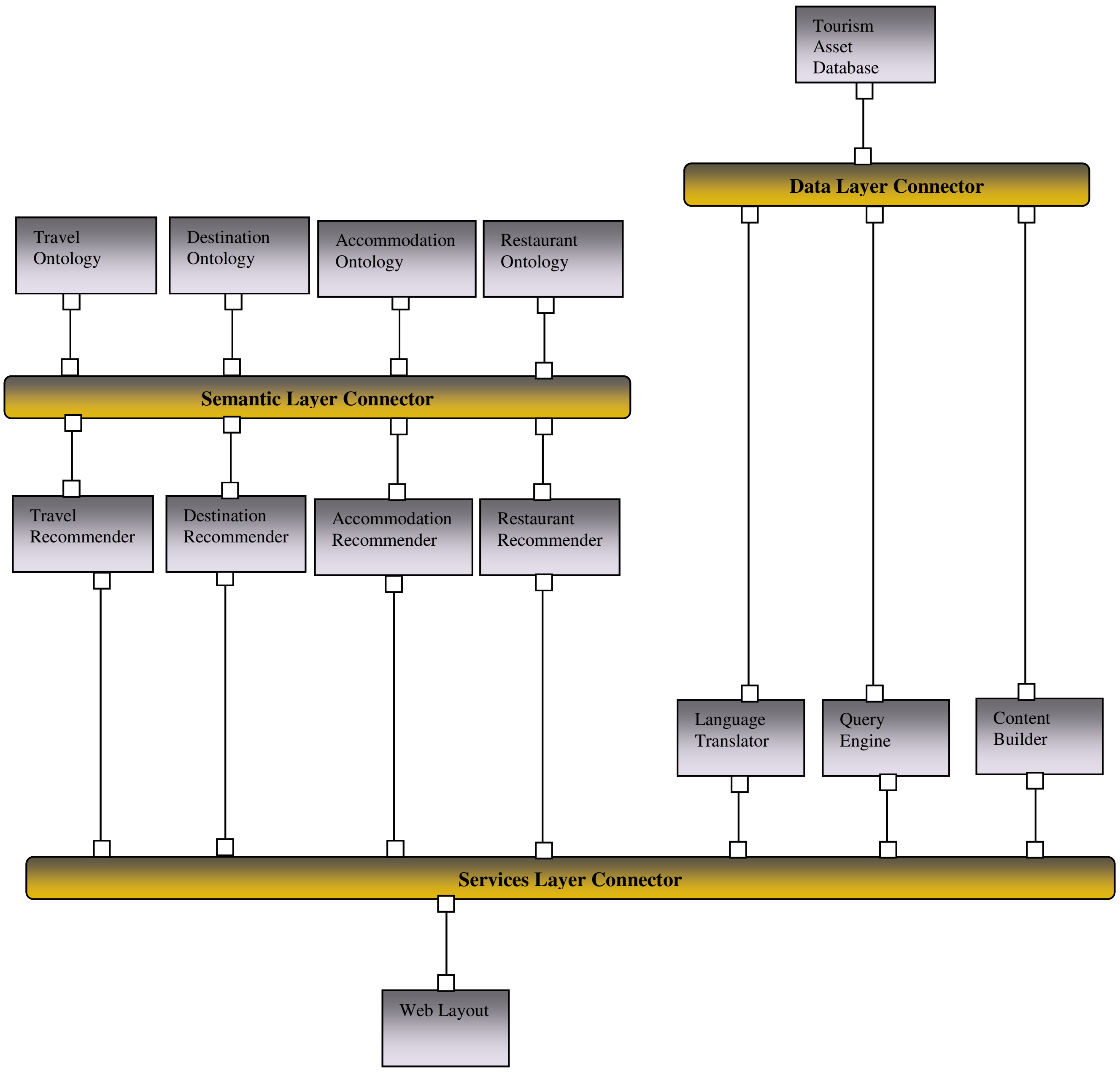


**Figure 4.14 The c2-style layered View of the TISPLA in Archstudio**

### ii) Formal Specification of Components with xADL

The xADL schemas for the component type structures in the TISPLA were generated by Archstudio. Each layer of the architecture is defined as a structure in xADL. The excerpts from the complete specification of the TISPLA are shown in figures. 4.15-4.18. In figure 4.15, the

TISPLA is shown to contain 4 structures, each structure representing a layer of the TISPLA architecture. Figure 4.16 and 4.17 show xADL specifications of the Information Query (a basic component) and Language Translation Engine (an optional component) components of the TISPLA.

The extensible nature of xADL schemas was engaged to extend the specifications of the Recommender System components in the TISPLA with the addition of the *"<UseResource>"* schema to indicate the essential resources required by the components to realize their respective functionalities. In Figure 4.18 the Destination Recommender component is specified as basic component in the product line that requires the services of semantic components of the TISPLA. The extension was made in order to promote a better understanding of the semantic properties of the component concerned in contrast to normal xADL descriptions that does not capture the semantic attributes of components (http://www.isr.uci.edu/projects/xarchuci/). In Figure 4.19, a sample specification of a semantic component type (Destination Context Ontology) is shown with the lookup implementation extension schema in xADL used to indicate the implementation source (source file) of the ontology component.

```
+<types:archStructure types:id="ClientLayer" xsi:type="types:ArchStructure">
+<types:archStructure types:id=" LogicServicesLayerStructure "types:ArchStructure">
+<types:archStructure types:id="SemanticLayerStructure" xsi:type="types:ArchStructure">
+<types:archStructure types:id="DataLayerStructure" xsi:type="types:ArchStructure">
```

**Figure 4.15 Structures in the TISPLA**

```
<types:component types:id=" InformationQueryComp " xsi:type="types:Component">
<types:description xsi:type="instance:Description">Database Query Engine</types:description>
<types:interface types:id="UserProfile_upper" xsi:type="types:Interface">
<types:description xsi:type="instance:Description">Upper Interface</types:description>
<types:direction xsi:type="instance:Direction">inout</types:direction>
</types:interface>
<types:interface types:id=" InformationQuery_bottom " xsi:type="types:Interface">
<types:description xsi:type="instance:Description">Bottom Interface</types:description>
<types:direction xsi:type="instance:Direction">inout</types:direction>
</types:interface>
<types:interface types:id=" InformationQuery _bottom " xsi:type="types:Interface">
<types:description xsi:type="instance:Description">Bottom Interface</types:description>
<types:direction xsi:type="instance:Direction">inout</types:direction>
</types:interface>
</types:component>
```

**Figure 4.16 xADL Specification of Information Query Component**

```
<types:component types:id="LangTranslationComp" xsi:type="options:OptionalComponent">
<types:description xsi:type="instance:Description">Language Translation Engine</types:description>
<types:interface types:id="UpperInterface" xsi:type="types:Interface">
<types:description xsi:type="instance:Description">Upper Interface</types:description>
<types:direction xsi:type="instance:Direction">inout</types:direction>
</types:interface>
<types:interface types:id="BottomInterface" xsi:type="types:Interface">
<types:description xsi:type="instance:Description">Bottom Interface</types:description>
<types:direction xsi:type="instance:Direction">inout</types:direction>
</types:interface>
<options:optional xsi:type="options:Optional"/>
</types:component>
```

**Figure 4.17 xADL Specification of Optional Language Translation**

```
<types:component types:id="DRS_RecommComp" xsi:type="types:Component">
<types:description xsi:type="instance:Description">Destination Recommender</types:description>
<types:interface types:id="UpperInterface" xsi:type="types:Interface">
<types:description xsi:type="instance:Description">Upper Interface</types:description>
<types:direction xsi:type="instance:Direction">inout</types:direction>
</types:interface>
<types:interface types:id="BottomInterface" xsi:type="types:Interface">
<types:description xsi:type="instance:Description">Bottom Interface</types:description>
<types:direction xsi:type="instance:Direction">inout</types:direction>
</types:interface>
<types:useResource type:id="DRS_RecommComp _neededResource1 " xsi:type="types:useResource">
<types:description xsi:type="instance:Description">uses Technology Layer Service</types:description>
<types:resourceid xsi:type="instance:resourceid">#SemanticComponentType</types:resourceid>
</types:useResource>
</types:component>
```

**Figure 4.18 xADL Specification of Destination Recommender Component**

```
<types:componentType types:id="DestinationOntologycomponentType" xsi:type="implementation:VariantComponentTypeImpl">
<types:description xsi:type="instance:Description">SemanticComponent Type</types:description>
<types:signature types:id="SemanticTypeUPsignature" xsi:type="types:Signature">
<types:description xsi:type="instance:Description">Semantic Component upper Signature</types:description>
<types:direction xsi:type="instance:Direction">inout</types:direction>
<types:serviceType xsi:type="types:SignatureServiceType">Provides</types:serviceType>
</types:signature>
<types:signature types:id="SemantictypeBTSgnature" xsi:type="types:Signature">
<types:description xsi:type="instance:Description">Semantic Component Lower Signature</types:description>
<types:direction xsi:type="instance:Direction">inout</types:direction>
<types:serviceType xsi:type="types:SignatureServiceType">Provides</types:serviceType>
</types:signature>
<implementation:implementation xsi:type="lookupimplementation:LookupImplementation">
<lookupimplementation:name
xsi:type="lookupimplementation:LookupName">http://sample.org/destinationontology</lookupimplementation:name>
</implementation:implementation>
</types:componentType>
```

**Figure 4.19 xADL Specification of Destination Context Ontology Component**

### 4.4.2.4 Description of DRS Component

The design of the DRS was based on a hybrid architecture that leverages content-based filtering and case-based reasoning (Vozalis & Margaritis, 2003) for destination recommendations. The set of travel activity preferences of a user is used as input, which is then correlated with the content description of various destinations to construct an ordered list of top nearest neighbourhood matches. Therefore, destination recommendation can be represented as an event-matching problem such that:

Given the conjunction predicate $User_j$ that denotes the activity preferences of a user and their associated priority ratings i.e.

$$User_j = a_1 r_1 \wedge a_2 r_2 \wedge a_3 r_3 \dots \wedge a_k r_k$$

where each $a_i$ is a specific travel activity feature, and $r_i$ the priority rating score of $a_i$. We define a predicate function

$$\mathbf{pred}(a_i) = \begin{cases} 1 \text{ (if } a_i \text{ has been selected)} \\ 0 \text{ (if } a_i \text{ has not been selected)} \end{cases}$$

such that $P_j$ becomes a pattern vector for the activity preferences of $User_j$:

$$P_j = \langle x_1.r_1, x_2.r_2, \dots x_k.r_k \rangle \text{ where each } x_i = \{0,1\} \text{ and integer } r_i \text{ such that } 0 \leq r_i \leq 5.$$

If $V = \{a_1, a_2, \dots a_n\}$ is the set of possible travel activities and $U = \{c_1, c_2 \dots c_m\}$ is the set of possible destinations then recommendation is given as: $F(V) \rightarrow X$ where $X \subset U$.

In our approach, we incorporated the description of the social attributes of a destination as defined in the Destination Context Ontology (DCO). Such that if the matrix $S_{mj}$ represents the description of j (where j is the maximum cardinality for social attributes) social attributes of m cities, then the augmented recommendation function becomes:

$F(V, S_{mj}) \rightarrow X^*$ where $X^* \subset U$.

Given that $X \, \Theta \, S_{mj} \rightarrow X^*$ where $\Theta$ is an ontology filtering operator, and $X^* \subset U$ is a re-ordering of X.

The hybrid DRS architecture consists of the following (see Figure 4.20):

- A Content-Based Filter (CBF)**:** This is responsible for generating the initial top-N recommendations after performing nearest-neighbour vector space matching between a given set of selected travel activities and activity features of prospective destinations. A personalized frequency-based metric $T_{ij}$ is computed for each possible destination after using a set of knowledge-based rules to associate specific tourist assets stored in a tourism asset database with particular travel activities i.e.

$$T_{ij} = \sum(k_j f_i)P_i \quad (4.1)$$

Where

$k_j$ = number of times activity $a_i$ has been selected by $user_j$ / number of times $user_j$ has traveled, hence $k_j$ is a personalization factor for $user_j$ based on the travel history.

$f_i$ = frequency count of assets for activity $a_i$ in a destination / total frequency count of assets for activity $a_i$. in the database.

$P_i$ = the priority score rating of activity $a_i$, if $a_i$ has been selected or 0 if not selected

- **A Cased-Based Reasoner**: The case-based filtering component endows the DRS with alternative personalization capability leveraging users' travel history. To achieve this, the systems stores the activity preferences profile and recommended results of all user sessions in its case base such that when a new user arrives, it does case matching using the cosine similarity metric (Vozalis & Margaritis, 2003) (see Equation 2.2) to determine the best-match from the case base. The recommendations for the best-matching case are given as the initial recommendations for the new case thus acting in this context as an exemplar case-based reasoner (Porter, 1987). This makes the system to generate its recommendations faster in that the use the content-based filtering approach is avoided.
- **Ontology Engine:** The ontology engine in the DRS architecture consumes the initial recommendations of the content-based / case-based filters and revises it after performing ontological reasoning based on facts stored in the ontological knowledge base (which is an instantiation of the DCO with the specific facts of a domain) so that a re-ordered top-N list of recommendations is produced.

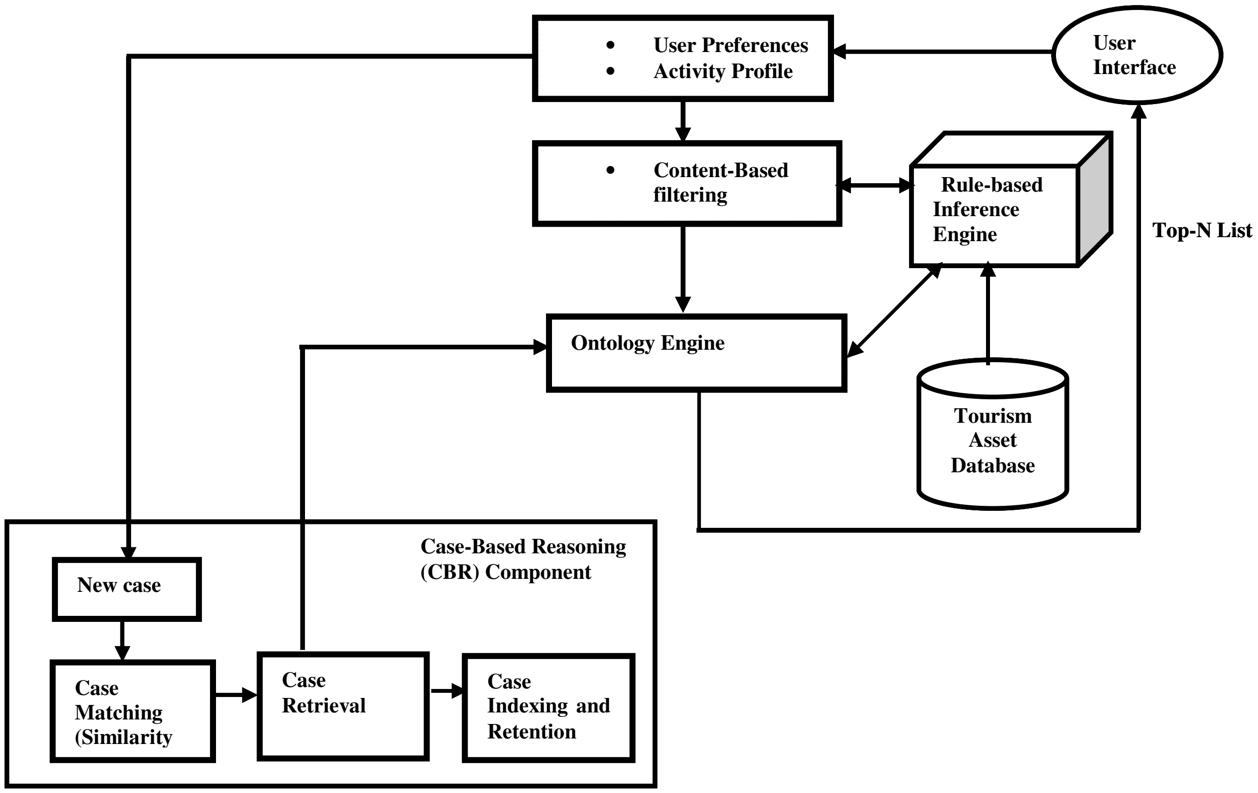


**Figure 4.20 Schematic Hybrid Architecture of DRS Component**

**DRS Algorithm**

```
Function DRS (User_j: List, N):List
    {The function matches the selected preferences of a user with the content description of possible destinations to
    return an ordered list of top-N destinations;
    User_j is a list of travel activity preferences of a user; N is number of products to recommend
    CosineMetric(): computes a similarity score for user_j using the cosine similarity metric
    Similarityscore[]:An array of similarity scores
    CaseBased-Filter(): implements a case-based reasoning algorithm; returns a list of size n
    Content-Based Filter(): implements a nearest-neighbour search algorithm; returns a list of size n
    OntoFilter(): is an ontology reasoner function; returns a List of size n
    Exist(): implements a database find function; returns a Boolean result; Flist: list of initial N-recommendations}
    SimilarityScore[j] ← CosineMetric(User_j)
    If Exist(SimilarityScore[j]) and Exist(User_j) then return CaseBased-Filter(User_j, N)
      else if FList ← ContentBased-Filter(User_j, N)
       return OntoFilter(Flist, N)
End function
```

#### 4.4.2.5 Description of the ARS Component

The ARS is a knowledge-based recommender system that leverages the knowledge captured about specific accommodation types in the AO to generate recommendations. By so doing deep knowledge filtered from the content description of key attributes of different accommodations types are used for recommendations. Formally, we could say that:

Given $V = \{a_1, a_2, \ldots a_n\}$ as the set of available accommodations and the vector U as the selected accommodation preferences of a user (in terms of type, facilities, services, attraction, gastro, and location of accommodation) i.e. $U_j = \langle x_1, x_2, x_3 \ldots x_k \rangle$

Then recommendation is given as: $F(V, U) \rightarrow N$ where N is an ordered list $\subset V$

The architecture of the ARS consists of the following (see Figure 4.21):

Inference Engine: This provides a basis for reasoning for decision making by the ARS.

Ontological Knowledge base: This is an instantiation of the Accommodation Ontology (AO) using specific instances. Facts about specific accommodation types such as hotels, guest houses, rented apartments etc. are captured in the knowledgebase.

Ontological Filtering Component: This executes an algorithm that matches the content descriptions of accommodation instances with the specified preferences of the user. It returns a Top –N list, where N is the number of product recommendations required by the user.

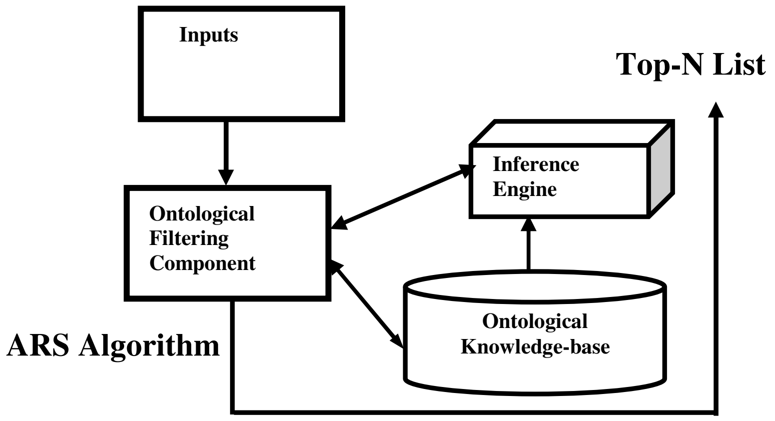


**Figure 4.21 Schematic Architecture of the ARS Component**

```
Function ARS ($U_j$ : List, N):List
    { The function matches the attributes of accommodation selected by a user with the content description of
    available accommodation types to return an ordered list of top-N destinations;
    $User_j$ is a list of selected attributes of accommodation by a user
    OntoRecommend(): is an ontology reasoner function; returns a List of size n
        If $U_j \leftarrow \{v_1, v_2, v_3 \ldots, v_n\}$
         OntoRecommend Computes $v_1 \Lambda v_2 \Lambda \ldots v_n$
    Rlist: list of initial N-recommendations }
    Rlist $\leftarrow$ OntoRecommend($U_j$, N)
      return Rlist
End function
```

### 4.4.2.6 Description of Other Components

The design of other content components of the TISPLA architecture was undertaken during domain design. These include:

i) **The Web Layout**: A web layout template was designed and stored as a cascading style sheet file using the Dream Weaver and Macromedia Flash design tools as platform on which the web interfaces of the TIS products will be based. The layout has four ports which represents the four core functionalities that will be realized in each of the TIS product. These are: destination recommendation, accommodation recommendation, Information search, and content update.

ii) **Tourism Asset Database**: A database schema design to store information on available tourism assets was also formulated. The structure of the tourism database was modelled following the structure of Canadian Tourism Board Database. Table 4.2 gives an overview of the structure of the tourism asset database.

**Table 4.2: Overview of the Structure of Tourism Asset Database**

| Field Name | Description | Type | Size |
|---|---|---|---|
| Assetcode | The code used to represent a tourist asset | String | 10 |
| Asset | Name of a tourism asset | String | 150 |
| Category | The tourism asset category | String | 50 |
| Subcategory | The tourism asset subcategory | String | 50 |
| District | The district in which a tourism asset is located | String | 150 |
| LocalGovt | Local government in which a tourism asset is located | String | 100 |
| State | The state in which an asset is located | String | 100 |
| Province-Region | The province or region in which a tourism asset is located | String | 100 |
| Authority | The tourism authority or private enterprise that manages or owns the tourism asset | String | 50 |
| City-Town | The city or town where the tourism asset is situated | String | 50 |
| Latitude | Latitude of the location of the tourism asset | Number | 12 |
| Longitude | Longitude of location of the tourism asset | Number | 12 |
| Last-update | The date when information on tourism asset was first supplied or last updated | Date | 12 |
| Route | Description of the route to the location of the tourism asset or its map information | String | 100 |
| Source | The source or provider of information on tourism asset | String | 50 |
| Picture | Snapshot of the image of the tourism asset | Image | 65535 |

### 4.4.3 Domain Realization

The domain realization phase involved the construction of the domain components using Java programming language implementation technologies. The content components that were implemented include 1) Destination Recommender System 2) Accommodation Recommender System 3) Database Query Component and 4) Database Layout Template. A fifth component, the Language Translation Engine was sourced as a standard Plug-in component from the Java open source platform. The details of the components and tools used for implementation and the implementation procedure are presented next.

### 4.4.3.1 Implementation Components and Tools

The software tools used for the implementation of content components include the following:

i) **NetBeans 5.5:** The NetBeans integrated development environment (IDE) is a free, open-source IDE for developing Java applications, including enterprise applications. NetBeans 5.5 supports the Java Enterprise Edition 5 (Java EE 5) platform.

ii) **Java EE Components:** A *Java EE component* is a self-contained functional software unit that is assembled into a Java EE application with its related classes and files that communicate with other components. The Java EE specification defines the following Java EE components:
   - Application clients and applets are components that run on the client.
   - Web components include Java Servlet, JavaServer Faces, and JavaServer Pages (JSP) technology components. They run on the server.
   - Enterprise JavaBeans (EJB) components are business components that run on the server.

iii) **Java Servlet Technology:** This enables the definition of HTTP-specific servlet classes. A servlet class extends the capabilities of servers that host applications that are accessed by way of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers.

iv) **JavaServer Pages Technology:** This allows the addition of snippets of servlet code directly into a text-based document. A JSP page is a text-based document that contains two types of text: static data (which can be expressed in any text-based format such as HTML, WML, and XML) and JSP elements, which determines how the page constructs dynamic content.

v) **Enterprise JavaBeans Technology:** An Enterprise Java Beans (EJB) component, or *enterprise bean*, is a body of code having fields and methods to implement modules of business logic. An enterprise bean is a building block that can be used alone or with other enterprise beans to execute business logic on the Java EE server. It is a server-side component written in the Java programming that encapsulates the business logic code that fulfills the purpose of the application. The enterprise beans implements the business logic in methods that when invoked enables clients to access the services provided by the components. There are two kinds of enterprise beans: session beans and message-driven beans. A *session bean* represents a transient conversation with a client. When the client finishes executing, the session bean and its data are gone. A *messagedriven bean* combines features of a session bean and a message listener, allowing a business component to receive messages asynchronously. Commonly, these are Java Message Service (JMS) messages. In Java EE 5, entity beans have been replaced by Java persistence API entities. An entity represents persistent data stored in one row of a database table. If the client terminates, or if the server shuts down, the persistence manager ensures that the entity data is saved.

vi) **Java Database Connectivity API:** This allows SQL commands to be invoked from Java programming language methods. The JDBC API is used in an enterprise bean when there is a need for a session bean to access the database. The JDBC API can also be used from a servlet or a JSP page to access the database directly without going through an enterprise bean. The JDBC API has two parts: an application-level interface used by the application components to access a database, and a service provider interface to attach a JDBC driver to the Java EE platform.

vii) **Sun Java System Application Server Platform Edition 9:** This is a fully compliant implementation of the Java EE 5 platform. It provides the necessary middleware infrastructure support for all the Java APIs. The Application Server includes a number of Java EE tools that are not part of the Java EE 5 platform but are provided as a additional support to the developer.

viii) **Macromedia DreamWeaver:** This is a rapid application development tool for web design and website development. It is the most popular visual HTML editor. It enables the creation of web page templates, cascading style sheets and offers support for multiple client-side programming languages such JSP, ASP.Net, VBscript, JavaScript etc.

ix) **MySQL Database:** This is a database management system tool for managing the data storage and retrieval. MySQL also has excellent query facilities and very suitable for the configuration of network data servers on the web.

x) **Protégé Ontology Editor:** Protégé is a flexible, configurable platform for the development of arbitrary model-driven applications and components. It has an extensible and customizable toolset for constructing ontologies and for developing applications that use these ontologies. Some of the outstanding features of Protégé include: 1) Automatic generation of graphical-user interfaces, based on user-defined models, for acquiring domain instances; 2) Extensible knowledge model and architecture; 3) Possible embedding of standalone applications in Protégé knowledge engineering environment and vice versa; and 4) enabling the scalability of ontologies to very large knowledge bases. Protégé also has an open architecture that allows programmers to integrate plug-ins, which can appear as separate tabs, specific user interface components (widgets), or perform any other task on the current model. The Protégé-OWL editor provides many editing and browsing facilities for OWL models, and therefore serves as an attractive starting point for rapid application development.

xi) **Protege-OWL API:** This is an open-source Java library for the Web Ontology Language (OWL) and RDF(S). The API provides classes and methods to load and save OWL files; to query and manipulate OWL data models; and to perform reasoning based on Description Logic engines. Furthermore, the API is optimized for the implementation of graphical user interfaces. The API is designed to be used in two contexts: 1) For the development of components that are executed inside the Protégé-OWL editor's user interface; and 2) For the development of stand-alone applications (e.g., Swing applications, Servlets, or Eclipse plug-ins)

xii) **Pellet Reasoner:** This is a Description Logic Reasoner that allows automated reasoning to be performed over an ontology. A Description Logic Reasoner performs various inferencing services, such as computing the inferred superclasses of a class, determining whether or not a class is consistent (a class is inconsistent if it cannot possibly have any instance), deciding whether or not one class is subsumed by another, etc.

xii) **Language Translation Engine**: This is an open source translation engine implemented by Google that equips content management systems on the web with language translation capabilities.

### 4.4.3.2 Implementation Details

The implementations of components were based on Java Platform Enterprise Edition (JEE) using the NetBeans 5.5.1 Java IDE. The recommender system components were implemented as stateless session beans (Enterprise Java Beans - EJB) that have their functionalities triggered using Java Servlet technology. Each of the recommender system EJBs were made to reference the relevant Protégé Ontology Java AP1 in order to enable the required ontology querying and description logics reasoning capabilities. The Pellet 1.5 Descriptive Logics (DL) reasoner (http://pellet.owldl.com) was used as the reasoning engine for the ontology-based transactions. The tourism asset database was implemented in MySQL, which exploits the JDBC technology to connect to the EJBs. The database query component was implemented as a generic parameterizable EJB that accepts user request to construct responses. The web layout template was implemented a cascading style sheet file using Macromedia Flash and Macromedia Dream Weaver tools. All components were deployed on the Sun Application Web Server 9.0 which serves as the middleware infrastructure for all server-based services. In figure 4.22 a view of the run-time deployment architecture of the domain components is presented. It is a 3-tier architecture showing the configuration of the content components as deployed on the Java EE server in the middle layer. The data layer (backend systems) consisting of ontologies and databases makes up the third layer while request for services are made through the client layer.

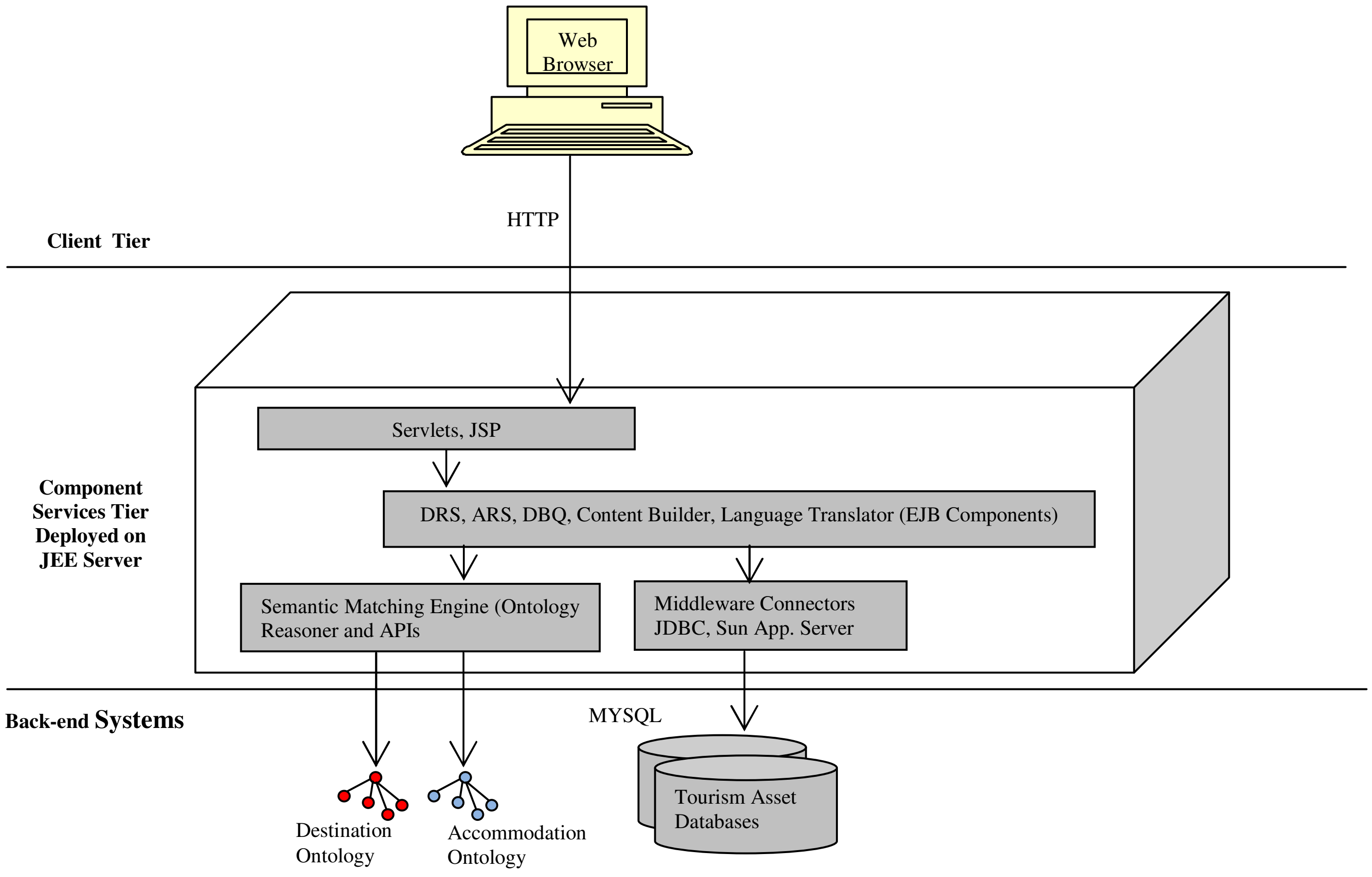

Web Browser
HTTP
Client Tier
Component Services Tier Deployed on JEE Server
Servlets, JSP
DRS, ARS, DBQ, Content Builder, Language Translator (EJB Components)
Semantic Matching Engine (Ontology Reasoner and APIs
Middleware Connectors JDBC, Sun App. Server
Back-end Systems
MYSQL
Tourism Asset Databases
Destination Ontology
Accommodation Ontology

**Figure 4.22: Deployment Architecture of System Components**

### 4.4.4 Domain Testing

Domain testing entails the testing of core asset components developed during domain realization. It differs from traditional application testing, in that the domain core assets are not yet a complete executable application which can be subjected to full-scale testing. However, it is desirable to test core assets, so as to assess their fitness for product line composition and in order to be able to correct defects noticed in them as early as possible. Among the varied domain system testing strategies that exist, the Simple Application Strategy (SAS) and Commonality and Reuse Strategy (CRS) was found most feasible in a product line context, hence as suggested in (Pohl et al., 2005) a combination of SAS and CRS was used for our domain testing.

#### 4.4.4.1 Using Simple Application Strategy (SAS)

The SAS of domain testing entails the creation of a sample prototype application with a typical product configuration using the implemented domain components as core assets. The purpose of SAS is to facilitate early validation and performance assessment of the core assets used. In essence, the SAS subsumes the unit testing and integration testing of the domain components used in a sample application. Also, the test cases generated during SAS can later be customized for reuse during application testing.

In our case study, we engaged SAS domain testing by developing a prototype web platform for one of the products (The Nigeria-TIS). The configuration of the sample application included the destination and accommodation recommendation features, database query, and database update features. This enabled us to test all the implemented core assets in order to certify them for use in application engineering. During SAS domain testing, some of the initial defects noticed in the functionalities of the domain components were corrected. Some of the issues had to do with the accuracy of recommendations particularly when the Top-n value supplied by a user is more than the number of generated recommendations from the available content catalog. An instance of this is if a user wants a Top-6 recommendation in a category where only four products exist. In this kind of situation a system will be reckoned as functioning well if it can return the relevant Top-4 rated products instead of doing something else. Also, several issues of inter-components interactions (between Servlet and EJB components, EJBs and Ontology Reasoner components, tourism database and EJB components) were resolved. Also, exception handling issues that were necessary to enhance the robustness of the domain components were attended to during SAS domain testing.

#### 4.4.4.2 Using Commonality and Reuse Strategy (CRS)

The goal of CRS domain testing is to assess the integration of the common parts of each product with the variable parts. In the CRS domain testing, test cases are defined for the common and variable parts of an application. Common parts are tested with the appropriate test cases as far as possible. Later, all predefined test cases are reused in application testing for a chosen system configuration to test the variable parts and once again the common parts to see if they work as intended.

In our case study, we constructed test cases for the common components of our product configurations based on the specification of the Application-Requirement Matrix (see Table 4.1). The configuration of the Nigeria-TIS was then used during application testing to test the integration of the common parts (components) with the variable components to ensure that they work together perfectly.

## 4.5 APPLICATION ENGINEERING IN PLONTOREC

Application engineering deals with the creation of specific products in the product line through the reuse of domain core assets created in domain engineering and exploiting the product line variability**.** The core activities of application engineering are application analysis, application design, application realization, and application testing (see Figure 4.23).

In application engineering the parameterizable core assets are configured with concrete parameters. Then, they are assembled to realize the needed features. Also some product-specific additions are made in order to cater for product specific requirements. If domain engineering was well conducted, the effort expended in application engineering should be much lower than in single system development. In the particular instance of our case study three TIS products were considered. These are the Nigeria-TIS, Ghana-TIS and Ivorian–TIS. The details of the sub-processes of application engineering that was applied in realizing these products are presented in the sequel sections.

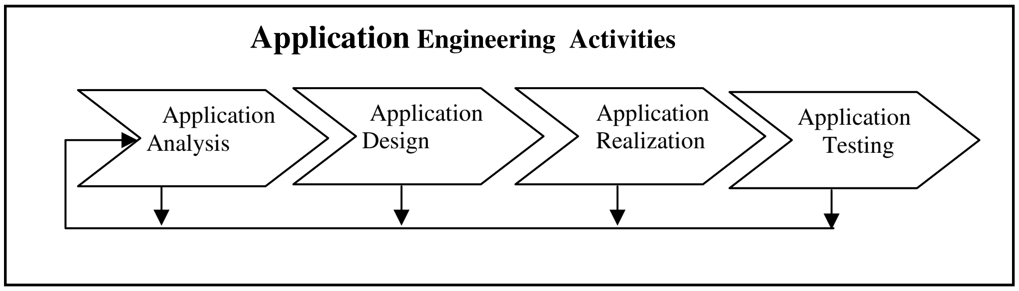


**Figure 4.23 Sub-processes of Application Engineering**

### 4.5.1 Application Analysis Process

The focus of application analysis process is to obtain requirements that are specific to concrete products in the product line. Application analysis enabled us to pay attention to requirements that were specific to the three national TIS products. Information contents local to specific countries were sourced; attention was also given to the need to develop customized layouts that reflects the unique national identities and essential attributes of the respective countries. The biases of each country with respect to hospitality were also noted during this phase. The summary of core application requirements for the three products is given as follows:

- **Product-Specific requirements for Nigerian-TIS**
  - *A web layout that distinctively represents the culture, tradition, characteristics and ambience of the Nigeria nation and people. These include national flags, national logos and images of national monuments unique to Nigeria.*
  - *Provision of destination recommendation service to prospective visitors offering guide on places to visit based on their preferred travel activity preferences.*
  - *Provision of guide on available accommodation facilities in Nigeria (hotels, guest houses, chalets etc.) based on a user's preferences in terms of desired services, attractions, facilities, gastronomy, location preferences. The recommendations are further constrained by the selected accommodation type and budget of the user.*
  - *Provision of query facilities that enable users to inquire information about existing tourism assets in Nigeria.*
  - *Creation of a platform that enables storing of information about new tourism artifacts and the updating of information on existing Nigerian tourism assets.*
- **Product-Specific Requirement for Ghana-TIS**
  - *A web layout that distinctively represents the culture, tradition, characteristics and ambience of the nation and people of Ghana. These include national identifiers and images of national monuments that are peculiar to Ghana.*
  - *Provision of destination recommendation service to prospective visitors offering guide on places to visit based on their preferred travel activity preferences.*
  - *Provision of guide on available accommodation facilities in Ghana (hotels, guest houses, chalets etc.) based on a user's preferences in terms of desired services,*

*attractions, facilities, gastronomy, location preferences. The recommendations are further constrained by the selected accommodation type and budget of the user.*

- *Provision of query facilities that enable users to inquire information about existing tourism assets in Ghana.*
- *Creation of a platform that enables storing of information about new tourism artifacts and the updating of information on existing Ghana tourism assets.*

• **Product-Specific Requirement for Ivorian -TIS**

- *A web layout that distinctively represents the culture, tradition, characteristics of the nation and people of Cote D'ivoire. These include national identifiers and images of national monuments that are peculiar to Cote D'ivoire.*
- *Provision of destination recommendation service to prospective visitors offering guide on places to visiting based on their preferred travel activity preferences.*
- *Provision of guide on available accommodation facilities in Cote D'ivoire (hotels, guest houses, chalets etc.) based on a user's preferences in terms of desired services, attractions, facilities, gastronomy, location preferences. The recommendations are further constrained by the selected accommodation type and budget of the user.*
- *Provision of query facilities that enable users to inquire information about existing tourism assets in Cote D'ivoire.*
- *Creation of a platform that enables storing of information about new tourism artifacts and the updating of information on existing tourism assets in Cote D'ivoire.*
- *A multi-language language web platform that allows presentation in the language of English and French.*

The output of the application analysis activity revealed that the application requirements were in tandem with the previously established domain requirements, which suggest that they are realizable through a product line approach.

### 4.5.2 Application Design Process

During application design, the artifacts of domain design are instantiated with the product-specific requirements obtained from application analysis to create design for individual products in the product line. Based on the results of application analysis, valid features configurations were chosen for each application as derived from the feature model developed in domain design (discussed in Sect. 4.4.2.1). Also, the reference architecture obtained from domain design is adapted with concrete data, and specific parameters to realize variant applications using the established variation points.

In our case study, the application design for each of the TIS products were derived directly from the reference architecture given in figure 4.13. First the concepts of the two ontologies (i.e. DCO, AO) were adapted to fit the specific context of individual countries. For example the concept '*State*' which connotes a regional unit of governance in the Nigerian context was customized as *'Province'* in the Ghanaian and Ivorian contexts of national governance. Thereafter the ontologies were instantiated with specific national information contents of the three countries to become national tourism knowledge bases (i.e. Nigerian Destination Ontology, Nigerian Accommodation Ontology, Ghana Destination Ontology, Ghana Accommodation, Ivorian Destination Ontology, and Ivorian Accommodation Ontology). The Tourism databases for the three countries were designed and populated with relevant local contents. The 3 databases had largely identical structure except for differences in the nomenclature of few fields (but the total number of fields was the same).

The valid feature sets derived from the domain feature model (see Figure 4.12) are shown for each of the TIS product variant. They are enumerated below (the root node is omitted and composite features contain subfeatures in parentheses):

> *Nigerian-TIS = {Tourism Recommender System (DRS (Nigeria Destination Ontology), ARS (Nigeria Accommodation Ontology)), web layout, query engine, database update (Nigeria Tourism Asset Database)}*
>
> *Ghana-TIS= {Tourism Recommender System (DRS (Ghana Destination ontology),*

*ARS (Ghana Accommodation Ontology)), web layout, query engine, database update (Ghana Tourism Asset Database)}*

*Cote D'ivoire-TIS= {Tourism Recommender System (DRS (Ivorian Destination ontology), ARS (Ivorian Accommodation Ontology)), web layout, language translation engine, query engine, database update (Ivorian tourism Asset Database)}*

The feature tree (FT) model of the three TIS product instances are shown in figures 4.24 –figures 4.26.

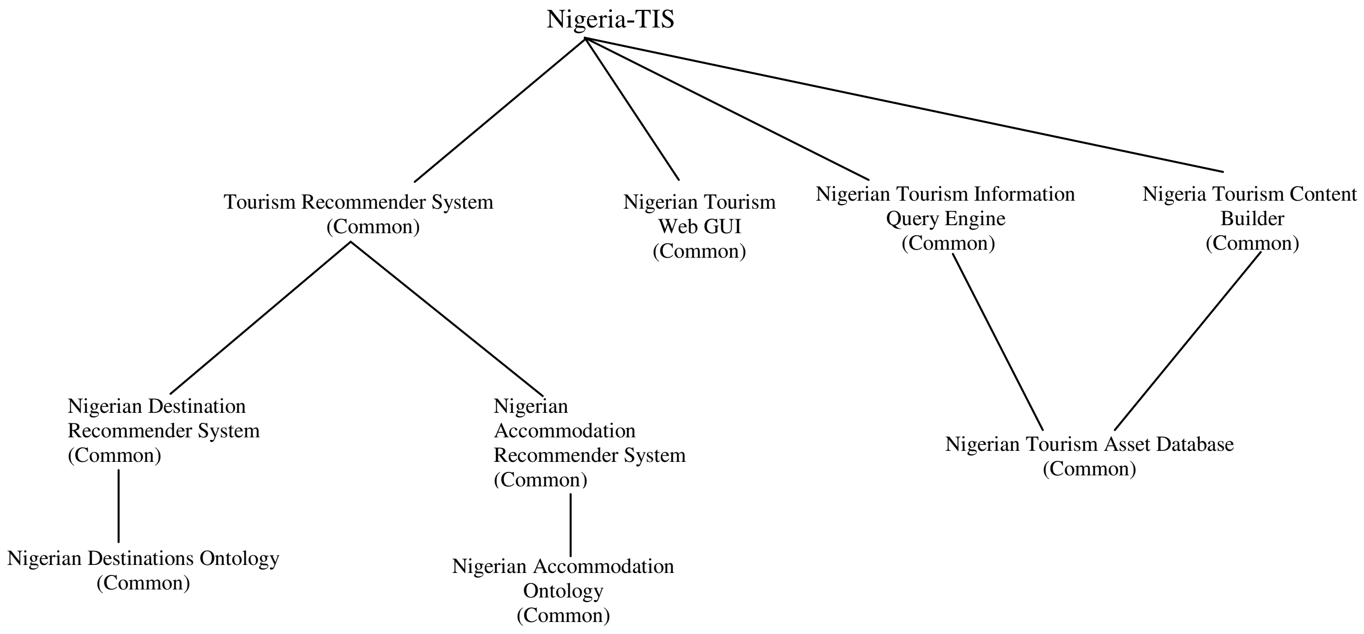

Nigeria-TIS
Tourism Recommender System (Common)
Nigerian Tourism Web GUI (Common)
Nigerian Tourism Information Query Engine (Common)
Nigeria Tourism Content Builder (Common)
Nigerian Destination Recommender System (Common)
Nigerian Accommodation Recommender System (Common)
Nigerian Tourism Asset Database (Common)
Nigerian Destinations Ontology (Common)
Nigerian Accommodation Ontology (Common)

**Figure 4.24 Feature Tree Model of Nigeria-TIS**

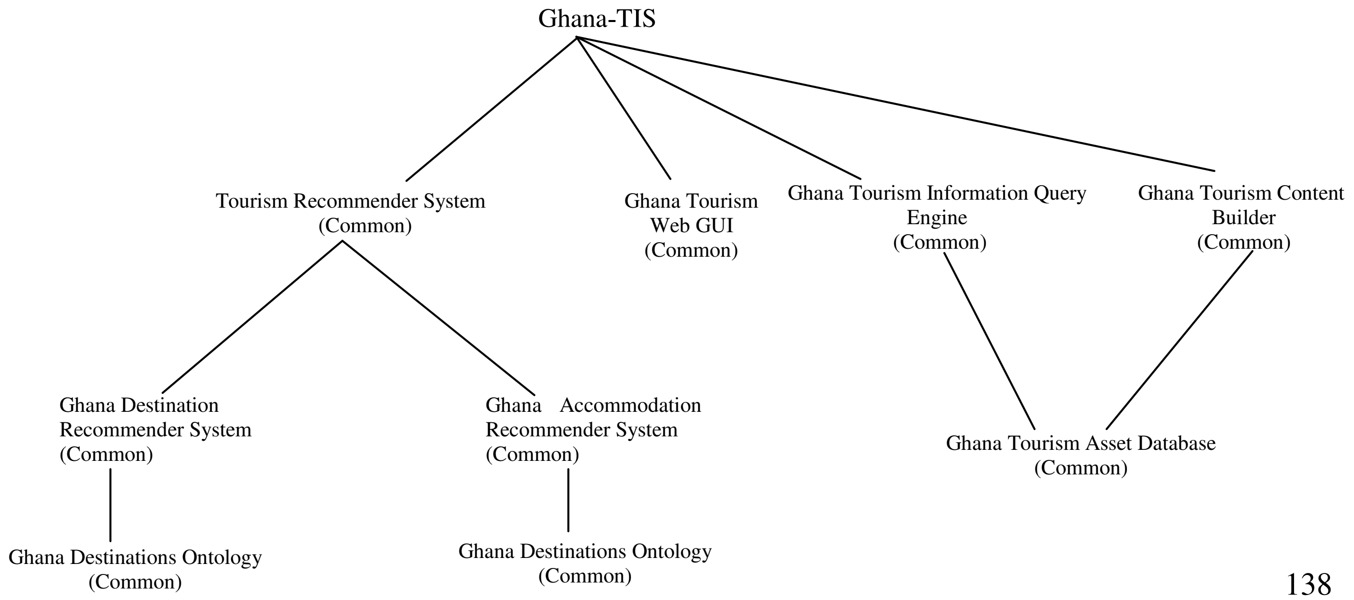

Ghana-TIS
Tourism Recommender System (Common)
Ghana Tourism Web GUI (Common)
Ghana Tourism Information Query Engine (Common)
Ghana Tourism Content Builder (Common)
Ghana Destination Recommender System (Common)
Ghana Accommodation Recommender System (Common)
Ghana Tourism Asset Database (Common)
Ghana Destinations Ontology (Common)
Ghana Destinations Ontology (Common)

**Figure 4.25 Feature Tree Model of Ghana-TIS**

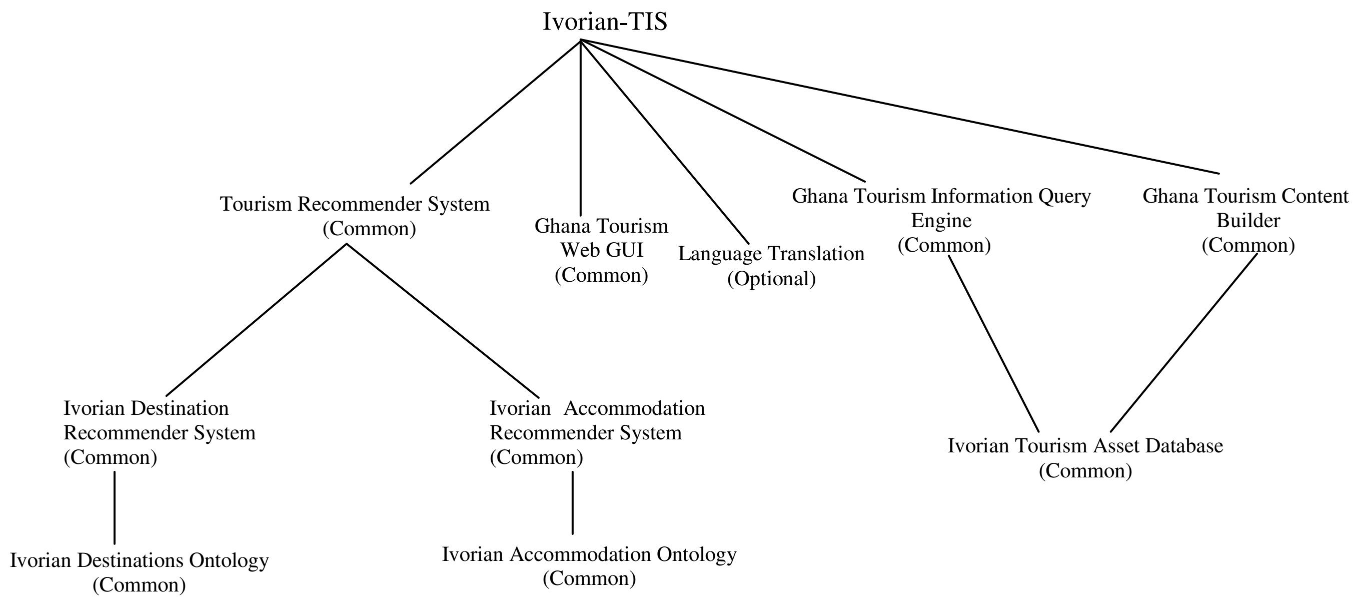

Ivorian-TIS
Tourism Recommender System (Common)
Ghana Tourism Web GUI (Common)
Language Translation (Optional)
Ghana Tourism Information Query Engine (Common)
Ghana Tourism Content Builder (Common)
Ivorian Destination Recommender System (Common)
Ivorian Accommodation Recommender System (Common)
Ivorian Tourism Asset Database (Common)
Ivorian Destinations Ontology (Common)
Ivorian Accommodation Ontology (Common)

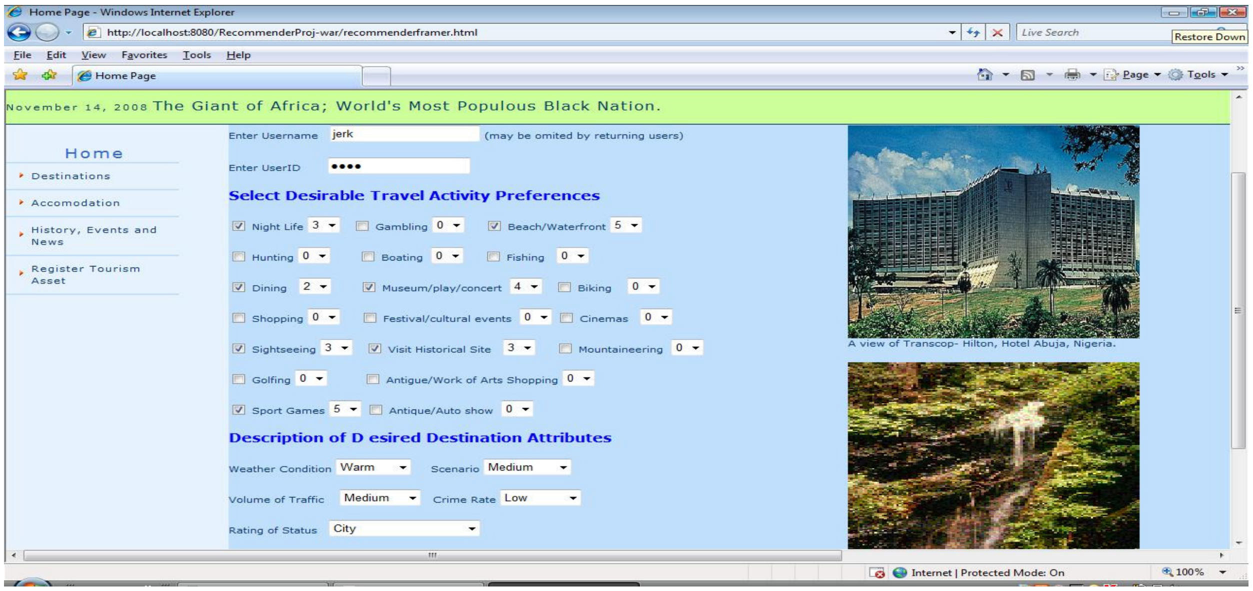
**Figure 4.26 Feature Tree Model of Ivorian-TIS**

### 4.5.3 Application Realization Process

Application realization commits to implementation of specific products in a product line leveraging the components developed during domain engineering. Hence, the focus of our implementation process shifted from a detailed development to customization and assembly of software components to realize specific product requirements.

The programming implementation platform used for the TIS products was Java 2 Enterprise Edition. The national tourism knowledge bases were created by populating the DCO and AO ontologies for each of the three countries with specific facts (individuals). Also, tourism information that pertains to each country was sourced and used to populate the respective tourism databases of the three countries. Java Servlet technology running on Sun Application Web Server 9.0 was employed to launch the functionalities of the recommender systems and information query components embedded in each of the national TIS products. The tourism databases were implemented in MySQL, using the JDBC Connector for connectivity. The web interfaces for each of the countries were implemented with Macro Media Flash and Dream Weaver web design tools leveraging a uniform web layout cascading style sheet template, while

Java Server Pages (JSP) scripting was used to provide the necessary supportive client-side scripting. Figures 4.27-4.29 are snapshots from the application realization of the three TIS products.


Home Page - Windows Internet Explorer
http://localhost:8080/RecommenderProj-war/recommenderframer.html
File Edit View Favorites Tools Help
Home Page
Home
Destinations
Accomodation
History, Events and News
Register Tourism Asset
Internet | Protected Mode: On
100%

November 14, 2008 The Giant of Africa; World's Most Populous Black Nation.

Enter Username jerk (may be omited by returning users)

Enter UserID ••••

**Select Desirable Travel Activity Preferences**

|  |  |  |
|---|---|---|
| ✓ Night Life 3 | Gambling 0 | ✓ Beach/Waterfront 5 |
| Hunting 0 | Boating 0 | Fishing 0 |
| ✓ Dining 2 | ✓ Museum/play/concert 4 | Biking 0 |
| Shopping 0 | Festival/cultural events 0 | Cinemas 0 |
| ✓ Sightseeing 3 | ✓ Visit Historical Site 3 | Mountaineering 0 |
| Golfing 0 | Antigue/Work of Arts Shopping 0 | |
| ✓ Sport Games 5 | Antique/Auto show 0 | |

**Description of D esired Destination Attributes**

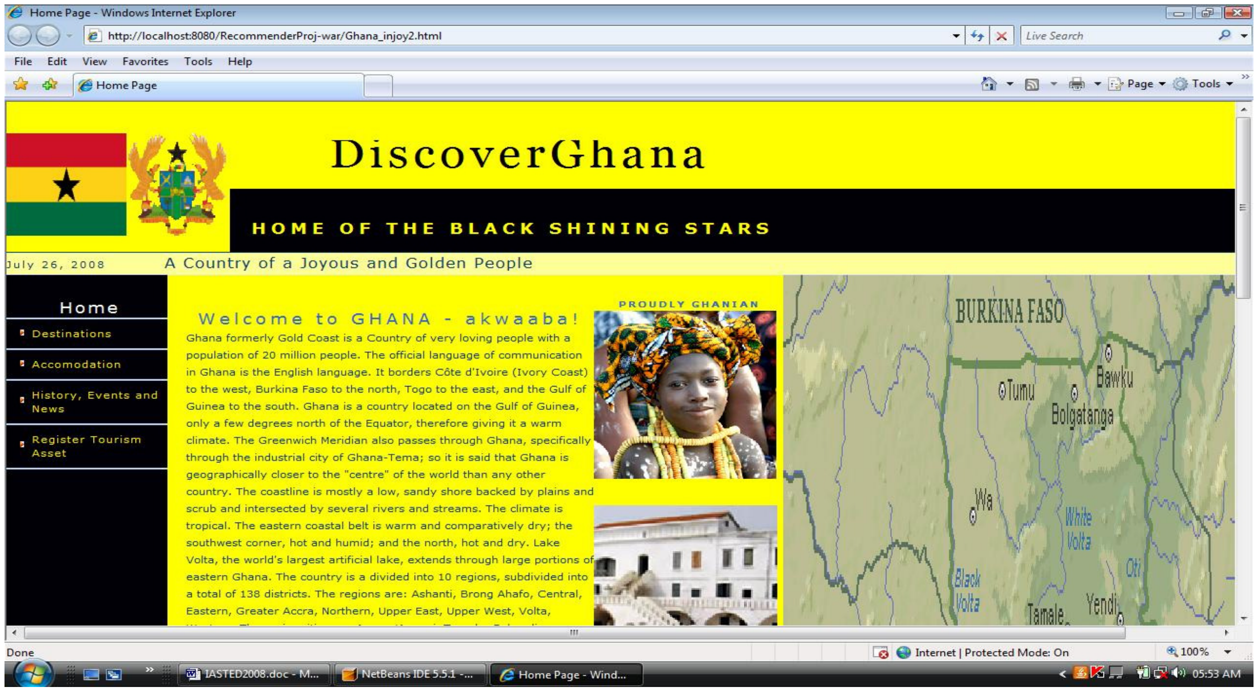
Weather Condition Warm Scenario Medium

Volume of Traffic Medium Crime Rate Low

Rating of Status City

A view of Transcop- Hilton, Hotel Abuja, Nigeria.

**Figure 4.27 Snapshot of the Destination Recommender System (DRS) in the Nigerian-TIS**



DiscoverGhana

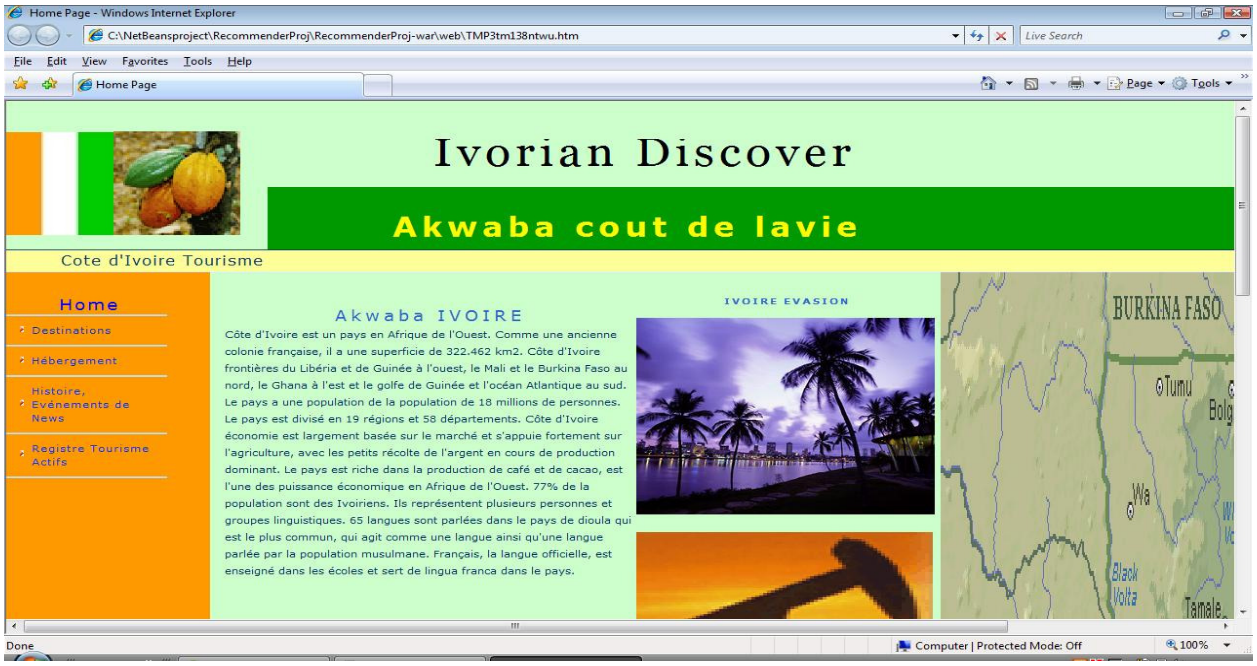
HOME OF THE BLACK SHINING STARS

July 26, 2008 A Country of a Joyous and Golden People

Welcome to GHANA - akwaaba!

Ghana formerly Gold Coast is a Country of very loving people with a population of 20 million people. The official language of communication in Ghana is the English language. It borders Côte d'Ivoire (Ivory Coast) to the west, Burkina Faso to the north, Togo to the east, and the Gulf of Guinea to the south. Ghana is a country located on the Gulf of Guinea, only a few degrees north of the Equator, therefore giving it a warm climate. The Greenwich Meridian also passes through Ghana, specifically through the industrial city of Ghana-Tema; so it is said that Ghana is geographically closer to the "centre" of the world than any other country. The coastline is mostly a low, sandy shore backed by plains and scrub and intersected by several rivers and streams. The climate is tropical. The eastern coastal belt is warm and comparatively dry; the southwest corner, hot and humid; and the north, hot and dry. Lake Volta, the world's largest artificial lake, extends through large portions of eastern Ghana. The country is a divided into 10 regions, subdivided into a total of 138 districts. The regions are: Ashanti, Brong Ahafo, Central, Eastern, Greater Accra, Northern, Upper East, Upper West, Volta,

**Figure 4.28 Snapshot of the Home Page of the Ghana- TIS (Discover Ghana)**


Home Page - Windows Internet Explorer
C:\NetBeansproject\RecommenderProj\RecommenderProj-war\web\TMP3tm138ntwu.htm
Live Search
File Edit View Favorites Tools Help
Home Page
Page Tools
Ivorian Discover
Akwaba cout de lavie
Cote d'Ivoire Tourisme
Home
Destinations
Hébergement
Histoire, Evénements de News
Registre Tourisme Actifs
IVOIRE EVASION
BURKINA FASO
Tumu
Wa
Black Volta
Tamale
Done
Computer | Protected Mode: Off
100%

Akwaba IVOIRE

Côte d'Ivoire est un pays en Afrique de l'Ouest. Comme une ancienne colonie française, il a une superficie de 322.462 km2. Côte d'Ivoire frontières du Libéria et de Guinée à l'ouest, le Mali et le Burkina Faso au nord, le Ghana à l'est et le golfe de Guinée et l'océan Atlantique au sud. Le pays a une population de la population de 18 millions de personnes. Le pays est divisé en 19 régions et 58 départements. Côte d'Ivoire économie est largement basée sur le marché et s'appuie fortement sur l'agriculture, avec les petits récolte de l'argent en cours de production dominant. Le pays est riche dans la production de café et de cacao, est l'une des puissance économique en Afrique de l'Ouest. 77% de la population sont des Ivoiriens. Ils représentent plusieurs personnes et groupes linguistiques. 65 langues sont parlées dans le pays de dioula qui est le plus commun, qui agit comme une langue ainsi qu'une langue parlée par la population musulmane. Français, la langue officielle, est enseigné dans les écoles et sert de lingua franca dans le pays.

**Figure 4.29 Snapshot of the Cote D'ivoire TIS (Ivorian-Discover)**

### 4.5.4 Details of Application Testing Process

The focus of our application testing process is to validate the quality of the TIS products generated in application realization. During this process, the TIS products were tested and validated using the domain requirements documents, application requirement documents, and domain test artifacts and application test artifacts. Some of the tests carried out include syntax checks (which was greatly boosted by the 'intellisence' and advanced debugging feature present in the NetBeans Java IDE that was used), unit testing (validating the individual created components), integration testing (checking the interaction of the components) and validation tests (ensuring that specific application requirements are fully satisfied). The fact that some measure of testing was carried out in domain testing using the Sample Application Strategy (SAS) and Commonality and Reuse Strategy (CRS), accelerated the application testing procedure. Also, the common parts were tested in a commonality test to see if they work correctly in different customized application contexts, all of these tests proved successful.

## 4.6 SUMMARY AND DISCUSSION

In this chapter the full scope of the application of the PLONTOREC life cycle has been discussed using a practical case study of TIS development. The sub-processes of PLONTOREC include: 1) Product Line Management - in which feasibility and risk assessment was undertaken prior to the commencement of the product line development activity; 2) Ontology Engineering – in which two OWL-KR ontologies were developed to enable a family of TIS products with knowledge-based recommendation capabilities; 3) Domain Engineering – in which specific domain reusable components such as tourism recommender systems for the TIS were developed; and 4) Application Engineering – in which 3 variant national TIS products for three countries in the West African sub-region were developed based on a specific product line feature model. The experience and observations gained from the application of these four aspects of PLONTOREC in a practical real-life scenario, demonstrates the potential viability of the PLONTOREC approach.

# CHAPTER FIVE
# EVALUTION OF THE PLONTOREC APPROACH

## 5.1 INTRODUCTION

This chapter reports the empirical evaluation of the PLONTOREC approach and its products. The usability evaluation of the two recommender systems was undertaken in order to capture users' impressions of the quality of their functionality and rate their efficiency. In addition, a usability evaluation of the TIS product platform was also conducted. Lastly, a comparative evaluation of the scenario of TIS development with PLONTOREC and without PLONTOREC was undertaken.

## 5.2 THE MOTIVATION FOR USABILITY EVALUATION OF TOURISM RECOMMENDER SYSTEMS

The accuracy metrics for evaluating recommender systems (See Section 2.3.5.1) involve measuring variables that are expected to affect the utility of a recommender system to the user and affect the reaction of the user to the system (Herlocker et al., 2004). Predictive accuracy metrics measure how close is the predicted ratings of a product by a system to true user ratings, while the decision support metrics evaluate the effectiveness of the system in helping users to distinguish between high-quality items and the rest of the product items. Precision and recall are the most popular decision-support metrics that have been used for evaluating recommender systems. The concepts of precision and recall were borrowed from Information Retrieval (IR), and have been used severally for recommender systems evaluation (Basu et al., 1998; Sarwar et al., 2000; Billsus & Pazzani, 2000; Sarwar, 2001). Precision is defined as the ratio of relevant products selected to the number of products selected. It represents the probability that a selected product is relevant. Recall is defined as the ratio of relevant product selected to the total number of relevant products available. It represents the probability that a relevant product will be selected.

However, while these metrics may be adequate for assessing the recommendation of commodity products like a book or a movie, they cannot be trusted in the case of the tourism product. This is because the tourism product is always unique to individuals, in that two people seldom have travel preferences that are exactly similar.  Peoples' tourism preferences are always unique and personal (unlike what obtains with commodity products). Recommender systems recommend products based on the likelihood that they will meet a specific user's taste or interest. However to determine whether a product meets the taste requirement of a user demand that we ask the user concerned. Thus, relevance is much more inherently subjective in tourism recommender systems and objective relevance does not exist.

Also, measuring recall in the context of tourism recommendation is almost always impractical. In the pure sense, measuring recall requires knowing whether each product is relevant; for a tourism recommender system, this would involve asking the opinion of many users to view all available products to measure how successfully a product has been recommended to each user. IR evaluations have been able to estimate recall by pooling relevance ratings across many users, but this approach depends on the assumption that all users agree on which items are relevant, which is inconsistent with the nature of the tourism product (Herlocker et al., 2004).

As an alternative to the traditional recall and precision metrics, Zanker et al. (2008) suggested that metrics along the dimensions of *efficiency, effectiveness* and *marketing intelligence* are more relevant to RS applications that will be featured in the commercial environment like tourism. Since the RS is expected to guide the user through a series of decision making steps without provoking him to quit the application, the usability of such RS then becomes the most approximate measure of its efficiency. Also, the fact that a direct usability evaluation of a system encapsulates several dimensions of users' perception of the system makes it a more realistic measure of efficiency (Zins et al., 2004a; Zins et al., 2004b).  These perspectives influenced the decision for a usability evaluation of the two recommender system components that were developed in the case study. The details of the empirical evaluation experiments are given next.

## 5.3 EMPIRICAL USABILITY EVALUATION OF RECOMMENDER SYSTEMS COMPONENTS

Usability evaluation is an attempt to measure the user's perception of a recommender system after an interaction experience. The essence of usability testing is to assess the quality of human-computer interaction properties of a system. According to ISO 924-11 (1998), usability is the extent to which specified users can use a system to achieve specified goals with effectiveness, efficiency and satisfaction. It is also, a perception of a system's ease of learning and use from both the experienced and inexperienced users' viewpoint (Lindgaard, 1994).

The reason for undertaking a prototype usability testing was to assess the performance of the DRS and ARS components and also to obtain timely feedback from potential users on possible future enhancements that are crucial for the recommender systems. This is based on our belief that the use of empirical testing with potential users is still the best way to find problems related to user's task and experiences (Zins et al., 2004a; Zins et al., 2004b; Riihiaho, 2003).

Herlocker et al. (2004) suggested the use of explicit (ask) and implicit (observe) feedback as the most appropriate for user evaluation of RS, and emphasised the need to clearly define the task that a recommender system is intended to support before its evaluation. Therefore, standard usability testing concepts (Nielsen, 1993) was used for evaluating the DRS.

### 5.3.1 Experiment Design for DRS

A trial experiment was undertaken with 20 users, including 5 non-Nigerian West Africans on short visit to Nigeria for the purpose of religious tourism. The rest of the sample user population comprises of staff and students of the Science and Technology faculty of Covenant University. All the participants gave their informed consent to participate in the experiment, and were taken through a 15 minutes tutorial session at the commencement of the experiment. Participants were requested to respond to a pre-experiment questionnaire which was specifically designed to evaluate the background of the participants particularly in terms of their IT skills, knowledge of the Internet, familiarity with recommender systems, e-Commerce portals, and general tourism and travel experience. They were asked to rate themselves on a scale of 100, which was graduated into 5 class categories. The specified task for the DRS is to provide intelligent recommendation to the user on the most probable Nigerian locations to spend the next vacation after it has been supplied with a list of travel activity preferences and social attributes description of a desirable destination. Participants were allowed to engage the system in as many sessions as they chose but were encouraged to randomly iterate between instances where social attribute preferences are included as input for recommendation and when they are not included.

The post-experiment questionnaire was a customisation of the Post-Study-Satisfaction-User-Questionnaire (PSSUQ) standard (Nielsen, 1993; Lewis, 1995; Zins et al., 2004a; Zins et al., 2004b). The PSSUQ has 26 questions, which were specifically adapted for a destination recommender system context (See Table 5.1). Items 16 and 17 in the questionnaire were specifically designed to capture users' impression of the system's recommendation when social attribute information is used and when not used, which is to be analysed to determine the potential influence of the inclusion of social attribute information of destination on the utility of recommendation. The participants were required to rate each item in the post-experiment question on a scale of 1-5 (1-Excellent, 2-Good, 3-Satisfactory, 2-Unsatisfactory, 1-Poor) while 'n/a' should be used for any questionnaire item they choose not to rate.

**Table 5.1 Usability and User Satisfaction Questionnaire for DRS**

| | **Items** | 5 | 4 | 3 | 2 | 1 | n/a |
|---|---|---|---|---|---|---|---|
| | **Design/Layout** | | | | | | |
| 1 | I liked using the interface of the system. | | | | | | |
| 2 | The organization of information presented by the system was clear. | | | | | | |
| 3 | The interface of this system was pleasant to use. | | | | | | |
| | **Functionality** | | | | | | |
| 4 | This system has all the functions and capabilities that I expect it to have to perform its task | | | | | | |
| 5 | The options listed by the system as a reply to my request were suitable for my travel. | | | | | | |
| 6 | I agree with the suggested recommendation of the system and believe it will be useful | | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| 7 | **Ease of Use** | | | | | | |
| 8 | It was simple to use this system. | | | | | | |
| 9 | It was easy to find the information I needed. | | | | | | |
| 10 | The information (such as online-help, on-screen messages, and other documentation) provided with this system was clear. | | | | | | |
| 11 | Overall, this system was easy to use. | | | | | | |
| | **Learnability** | | | | | | |
| 12 | It was easy to learn to use the system. | | | | | | |
| 13 | There was too much information to read before I can use the system. | | | | | | |
| 14 | The information provided by the system was easy to understand. | | | | | | |
| | **Satisfaction** | | | | | | |
| 15 | I felt comfortable using this system. | | | | | | |
| 16 | I am satisfied with recommendations when social attribute information of destination is used. (*) | | | | | | |
| 17 | I am satisfied with recommendations when social attribute information of destination is not used. (*) | | | | | | |
| 18 | Overall, I am satisfied with this system. | | | | | | |
| | **Outcome / Future Use** | | | | | | |
| 19 | I was able to complete the task quickly using this system. | | | | | | |
| 20 | I could not complete the task in the preset time frame. | | | | | | |
| 21 | I believe I could become productive quickly using this system. | | | | | | |
| 22 | The system was able to convince me that the recommendations are of value. | | | | | | |
| 23 | From my current experience with using the system, I think I would use it regularly. | | | | | | |
| | **Errors / System Reliability** | | | | | | |
| 24 | Whenever I made a mistake using the system, I could recover easily and quickly. | | | | | | |
| 25 | The system gave error messages that clearly told me how to fix problems. | | | | | | |
| 26 | In my opinion the system is somewhat fault tolerant | | | | | | |

The pre-experiment and post-experiment questionnaires were analysed and the following were the findings:

i) 80% of participants claimed to be expert Internet users (indicating a rating of 70-100).

ii) 60% of participants' claimed to have very good familiarity with RS and e-Commerce applications;

iii) 40% rated their travel and tourism experience as excellent;

iv) Another 40% rated their travel and tourism experience as above average.

v) While the remaining 20% claimed to have little or no travel and tourism experience.

Figure 5.1 is a chart showing a summary of the background of participants according to their familiarity with e-Commerce applications, RS and previous tourism experience.

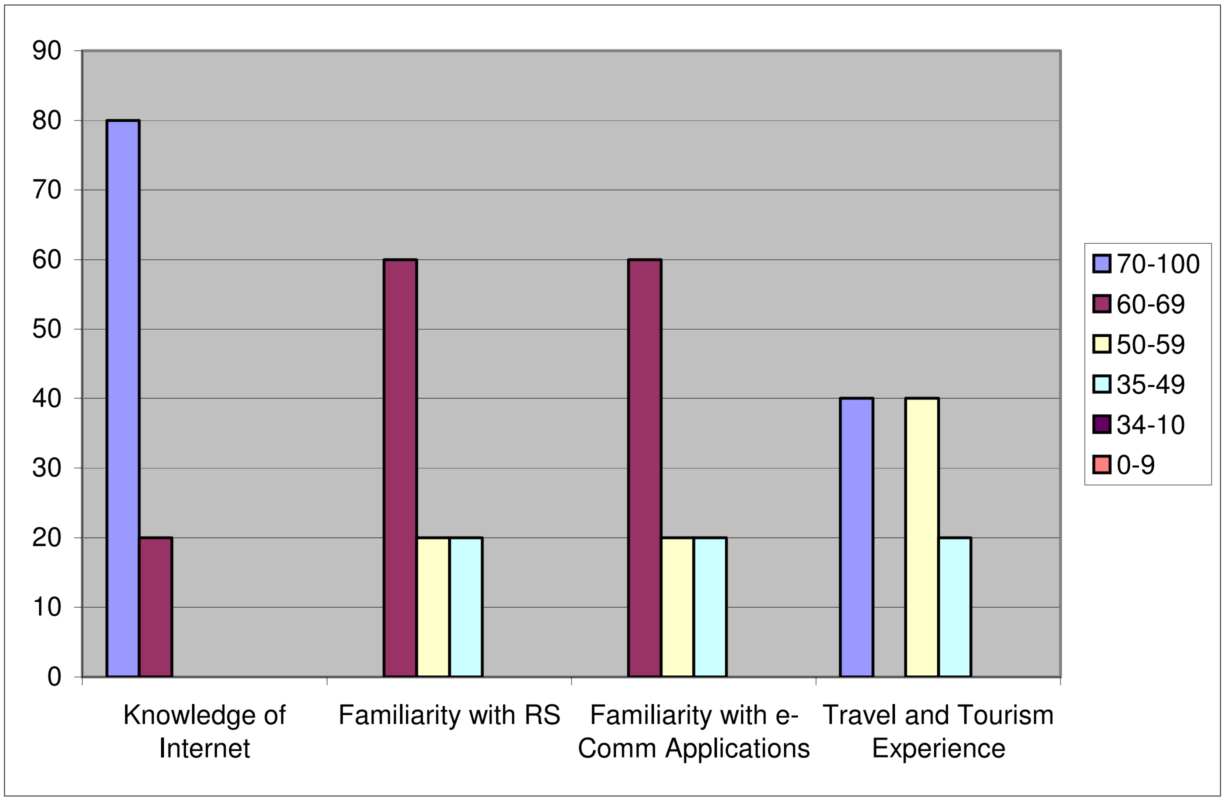


**Figure 5.1 Summary of Background of Participants**

### 5.3.2 Post –Experiment Results for DRS

The feedback obtained from users through the post-experiment questionnaire was analysed statistically to determine the mean scores of user ratings of the system based on the seven usability metric parameters used to evaluate the system. Table 5.2 shows the mean scores of the parameters used. These are: design/layout, functionality, ease of use, learnability, satisfaction (which was split into two, i.e. when social attribute information was used and when social attribute information was not used), future use (confidence), and reliability. From the result, the DRS had a mean score of above 4 in seven out of the 8 parameters used. Several usability studies have revealed that a system should have a mean score of 4 on a 1-5 scale to be rated as acceptably usable. Hence, it is sufficient to say that the DRS has a good usability. Also, from our experiment, it was discovered that most users expressed satisfaction; and showed preference for recommendations that were based on the use of social attributes information over when social attributes information was not used.

**Table 5.2: Means Scores of Usability Metrics for DRS**

| | Usability Metrics | Mean Scores | Std. Deviation |
|---|---|---|---|
| 1 | Design/Layout | 4.13 | 0.57 |
| 2 | Functionality | 4.19 | 0.63 |
| 3 | Ease of Use | 4.15 | 0.25 |
| 4 | Learnability | 4.00 | 0.76 |
| 5 | Satisfaction/Social attribute | 4.15 | 0.78 |
| 6 | Satisfaction/without Social attribute | 3.58 | 1.05 |
| 7 | Outcome/Future Use | 4.20 | 0.34 |
| 8 | Reliability | 4.02 | 0.68 |

Also, from our experiment, 80% of the sample population responded that they felt comfortable with the system by giving it a rating of 5(excellent) or 4(good). 20% of the participants gave the system a rating of 3(satisfactory) or 2(unsatisfactory). 60% of the sample population rated the recommendations of the system as excellent or good when social attributes information was used, 20% of participants rated the recommendations as satisfactory or unsatisfactory, while 40% chose not to comment. Also, 20% of participants rated recommendations of the system as 3(satisfactory) or 2 (unsatisfactory), when social attribute information is not used, 0% rated it as excellent or good, while 40% chose not to comment. 80% of participants felt generally satisfied with the system. Figure 5.2 is a visualization of user's satisfaction with the recommendation of the DRS prototype.

The results of the evaluation experiment clearly support the notion that use of contextual information such as the social attributes information of destinations as a factor in destination recommendation can indeed boost the dependability of destination recommendations.

**Figure 5.2 Summary of User's Satisfaction with the DRS**

### 5.3.3 Experiment Design for ARS

A trial experiment for evaluation similar to that of the DRS was engaged for the ARS. 10 willing and informed users were used for the evaluation even though it has been suggested that just five users are sufficient for a first cut usability study of any system (Nielsen, 1993). Half of the users in this experiment participated in the DRS evaluation which significantly reduced the need for prolonged preliminary introduction; nevertheless a 10 minutes tutorial session was given to users at the commencement of the experiment. The participants in this experiment shared largely the same background with those used for the DRS evaluation in terms of their IT skills, knowledge of the Internet, familiarity with recommender systems, e-Commerce portals, and general tourism and travel experience. The specified task for the ARS which participants are to evaluate is to see how well it recommends relevant accommodation types (e.g. hotel, guesthouse, chalet etc.) based on their selected preferences in terms of facilities, services, attractions, and gastronomy. Participants were allowed to engage the system in as many sessions as they desired.

A post-experiment questionnaire containing 24 questions that was designed based on the PSSUQ standard was used to capture user's impression of the ARS (See Table 5.3). The participants were required to rate each item in the post-experiment question on a scale of 1-5 (1-Excellent, 2-Good, 3-Satisfactory, 2-Unsatisfactory, 1-Poor) while 'n/a' should be used for any questionnaire item they choose not to rate.

**Table 5.3 Usability and User Satisfaction Questionnaire for ARS**

| | **Items** | 5 | 4 | 3 | 2 | 1 | n/a |
|---|---|---|---|---|---|---|---|
| | **Design/Layout** | | | | | | |
| 1 | I liked using the interface of the ARS. | | | | | | |
| 2 | The organization of information presented by the ARS was clear. | | | | | | |
| 3 | The interface of this system was pleasant to use. | | | | | | |
| | **Functionality** | | | | | | |
| 4 | This system has all the functions and capabilities that I expect it to have to perform its task | | | | | | |
| 5 | The options listed by the system as a reply to my request were suitable for my decision making. | | | | | | |
| 6 | I agree with the suggested recommendation of the system and believe it will be useful | | | | | | |
| 7 | **Ease of Use** | | | | | | |
| 8 | It was simple to use this system. | | | | | | |
| 9 | It was easy to find the information I needed. | | | | | | |
| 10 | The information (such as online-help, on-screen messages, and | | | | | | |

|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
|  | other documentation) provided with this system was clear. |  |  |  |  |  |  |
| 11 | Overall, this system was easy to use. |  |  |  |  |  |  |
|  | **Learnability** |  |  |  |  |  |  |
| 12 | It was easy to learn to use the system. |  |  |  |  |  |  |
| 13 | There was too much information to read before I can use the system. |  |  |  |  |  |  |
| 14 | The information provided by the system was easy to understand. |  |  |  |  |  |  |
|  | **Satisfaction** |  |  |  |  |  |  |
| 15 | I felt comfortable using this system. |  |  |  |  |  |  |
| 16 | I am satisfied with the recommendations. |  |  |  |  |  |  |
|  | **Outcome / Future Use** |  |  |  |  |  |  |
| 17 | I was able to complete the task quickly using this system. |  |  |  |  |  |  |
| 18 | I could not complete the task in the preset time frame. |  |  |  |  |  |  |
| 19 | I believe I could become productive quickly using this system. |  |  |  |  |  |  |
| 20 | The system was able to convince me that the recommendations are of value. |  |  |  |  |  |  |
| 21 | From my current experience with using the system, I think I would use it regularly. |  |  |  |  |  |  |
|  | **Errors / System Reliability** |  |  |  |  |  |  |
| 22 | Whenever I made a mistake using the system, I could recover easily and quickly. |  |  |  |  |  |  |
| 23 | The system gave error messages that clearly told me how to fix problems. |  |  |  |  |  |  |
| 24 | In my opinion the system is somewhat fault tolerant. |  |  |  |  |  |  |

### 5.3.4 Post –Experiment Results for ARS

The feedback obtained from users through the post-experiment questionnaire was analysed statistically to determine the mean scores of user ratings of the ARS based on the seven usability metric parameters used to evaluate the system. Table 5.4 shows the mean scores of the parameters used. From the result, the ARS had a mean score of above 4.3 in all seven parameters used, which is symbolic of the fact that the ARS has good usability. It is also important to note that the ARS is purely a semantic web application. Hence the evaluation of the ARS also directly corresponds to an application-based evaluation of the Accommodation Ontology (AO) that is at the base of the ARS knowledge-base. Therefore the positive result obtained certifies the good quality and completeness of the Accommodation ontology.

**Table 5.4: Means Scores of Usability Metrics for ARS**

| | Usability Metrics | Mean Scores | Std. Deviation |
|---|---|---|---|
| 1 | Design/Layout | 4.53 | 0.47 |
| 2 | Functionality | 4.50 | 0.32 |
| 3 | Ease of Use | 4.35 | 0.25 |
| 4 | Learnability | 4.50 | 0.56 |
| 5 | Satisfaction/Social attribute | 4.5 | 0.38 |
| 6 | Outcome/Future Use | 4.42 | 0.24 |
| 7 | Reliability | 4.30 | 0.58 |

Also, from our experiment, 80% of the sample population responded that they felt comfortable with the system by giving it a rating of 5(excellent) or 4(good). 20% of the participants gave the system a rating of 3(satisfactory) or 2(unsatisfactory). 80% of the participants also expressed confidence in the recommendations of the ARS, claiming that they believed it enough to act on it. The results of this evaluation experiment clearly support the notion that recommendations that are based on deep factual knowledge of a specific tourism domain are more dependable and have a greater propensity to foster users' confidence.

## 5.4 EMPIRICAL USABILITY EVALUATION OF TIS PRODUCTS

A usability evaluation of the e-Tourism portal for Nigeria was undertaken to assess user's impression of the TIS product. A post-experiment questionnaire was formulated based on the Post-Study-Satisfaction-User-Questionnaire (PSSUQ) standard. The PSSUQ had 26 questions, which were specifically adapted to fit the scenario of our case study. The participants were required to rate each item in the post-experiment question on a scale of 1-5 (1-Excellent, 2-Good, 3-Satisfactory, 2-Unsatisfactory, 1-Poor) while ‘n/a’ was used for any questionnaire item they choose not to rate. The questions addressed various aspects which include: design layout, functionality, ease of use, learnability, satisfaction, outcome/future use and reliability of the system. The post-experiment questionnaire was analysed statistically to determine the mean scores of user ratings of the system based on the seven usability metrics used for evaluation. Table 5.5 shows the mean scores obtained for each of the metrics used. From the result, the system had a mean score of above 4.0 in all of the 7 parameters used which suggests that the system is sufficiently usable and has a an acceptable performance level going by users’ ratings. In our experiment, we sought to know what users feel about the fact that the recommendations were knowledge-based. From the feedback, we discovered that most of the users felt that the recommendation were accurate enough to earn their trust, because of convincing evidences that they were based on some facts that they are also aware of.

**Table 5.5: Means Scores of Usability Metrics for e-Tourism System Prototype**

| | **Usability Metrics** | **Mean Scores** | **Std. Deviation** |
|---|---|---|---|
| 1 | Design/Layout | 4.13 | 0.57 |
| 2 | Functionality | 4.19 | 0.63 |
| 3 | Ease of Use | 4.15 | 0.25 |
| 4 | Learnability | 4.00 | 0.56 |
| 5 | Satisfaction | 4.15 | 0.28 |
| 6 | Outcome/Future Use | 4.20 | 0.34 |
| 7 | Reliability | 4.02 | 0.68 |

Summarily, 80% of the sample population responded that they felt comfortable with the system by giving it a rating of 5(excellent) or 4(good). 20% of the participants gave the system a rating of 3(satisfactory) or 2(unsatisfactory). 60% of the sample population rated the recommendations of the system as excellent or good and claimed to believe it, 20% gave it a rating of 3 or 2 while 20% chose not to comment. 80% expressed general satisfaction with all aspects of the system. Figure 5.3 is a visualization of user’s perception of the system. We consider the results of the evaluation experiment encouraging and supportive of our belief that development of semantic ontology-based platform will engender the delivery of knowledge-based recommendations and will command user's confidence is indeed viable.

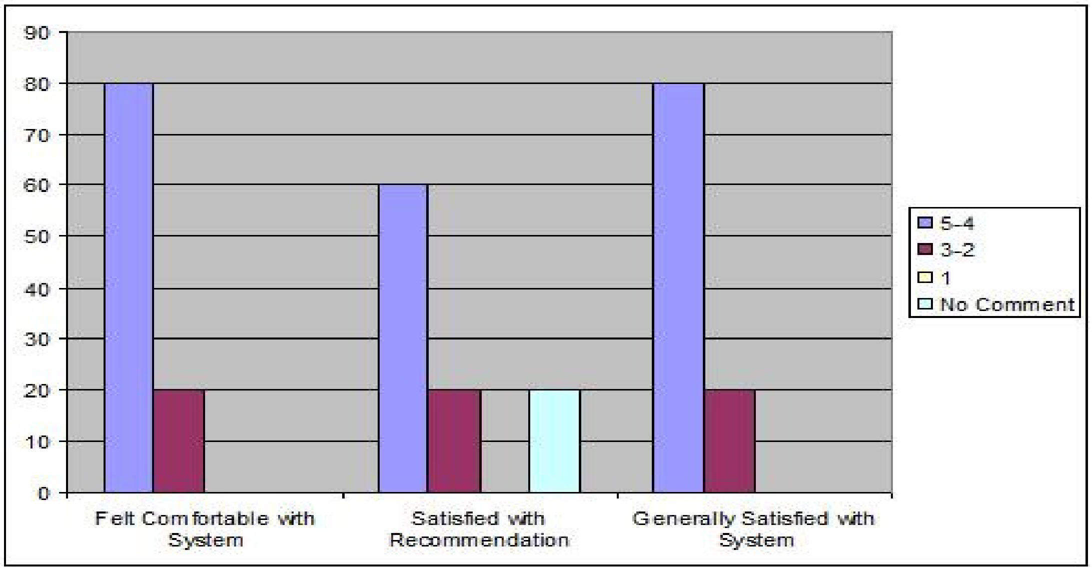


**Figure 5.3 A Graphical View of User's Satisfaction Index for the e-Tourism Portal**

## 5.5 EVALUATION OF THE PLONTOREC APPROACH

In order to evaluate the PLONTOREC approach, we compared our experiences in the scenario of the application of PLONTOREC and the situation where traditional software development approach was used in engineering the three TIS products. The details of the evaluation procedure are given next.

### 5.5.1 Estimating Effort of Developing TIS Products using COCOMO II

The Constructive Cost Model (COCOMO) is an algorithmic software budget estimation technique that is used to empirically determine the amount of effort required for the complete development of a software project (Boehm et al., 2000). It uses empirically derived formulas to estimate the cost of human resources (effort) as a function of the project size. The latest version of the COCOMO estimation technique is the COCOMO II which subsumes the previous version

COCOMO 81. COCOMO II consist of three different models, which are: 1) The Application Composition Model (which is to be used in the early analysis stage or during prototyping); 2) The Early Design Model (which is to be used after requirement analysis is completed); and 3) The Post-architecture Model (which is to be used after the software architecture design is known).

The COCOMO II Post-architecture model was used to estimate the effort in developing a TIS product using the traditional development approach. The motivation for using the Post-architecture model is to enable more accurate information for various cost drivers to be generated and thus ensure more accurate estimations. The estimation formula for COCOMO II model is given as (Boehm et al., 2000):

$$effort = c * size^k * m + autoeffort \qquad (5.1)$$

Based on the COCOMO II research, the value of constant coefficient *c* is set to 2.5. The Post-architecture model of COCOMO II makes use of seventeen cost drivers and five scale factors. The cost drivers are grouped into 4 categories (product, platform, personnel, and project) and it is not always necessary to consider all four categories. The values of the cost drivers are multiplied to obtain the value of the effort multiplier *m* in the formula. The value of cost drivers ranges from very low to very high. Cost drivers have a nominal value of 1. A value above 1 for a cost driver connotes negative effect on the effort multiplier while a value below 1 connotes a positive effect. A value of 1 does not affect the computation of *m*.

The scale factors are used to derive the value from exponent *k* applied to the *size* value. Each factor is rated with integer values from 5 to 0. The values are added, divided by 100, and the result added to the nominal value of the exponent (k) 1.01 to give the new value of the *k* to be used in the formula. The five scale factors were used to account for the relative economies or diseconomies of scale encountered for software projects of different sizes. The *size* value represents the total number of unadjusted function points in the project. The last term in the formula, *autoeffort*, is the effort put by developers in the automatic code generation and the integration of the code with the manually created programs. The details of our estimation experiment are presented next.

### 5.5.1.1 Determining Function Points

The function points metric is a measure of the size of the functionality associated with a software project. It is used to quantify the information processing functionality associated with major external data or control input, output, or file types in a software project. Function points are useful estimators since they are based on information that is available early in the project life cycle. The various components of function points are (http://linkinghub.elsevier.com/retrieve /pii/S0164121200000157):

**External Input (Inputs)**: This is the count of each unique user data or user control input type that (i) enters the external boundary of the software system being measured and (ii) adds or changes data in a logical internal file.

**External Output (Outputs)**: This is the count of each unique user data or control output type that leaves the external boundary of the software system being measured.

**Internal Logical File (Files)**: This is the count of each major logical group of user data or control information in the software system as a logical internal file type. This includes each logical file (e.g. each logical group of data) that is generated, used, or maintained by the software system.

**External Interface Files (Interfaces)**: These are files that are passed or shared between the software systems within each system.

**External Inquiry (Queries)**: This is the count of each unique input-output combination, where an input causes and generates an immediate output, as an external inquiry type.

An overview of the function points counting procedure used for the Nigeria-TIS is shown in Figure 5.4 as follows:

| **Data Functions:** | **Transactional Functions:** |
|---|---|
| List of Internal Logical Files (ILF):<br>1. Tourism Asset Table ( 19 fields)<br>2. Case Base Table ( 8 fields)<br><br>List of External Interface Files (EIF):<br>1. Destination Recommender Interface<br>2. Accommodation Recommender Interface<br>3. Query Interface<br>4. Update Asset Register Interface<br>5. Main Application Interface | List of External Inputs (EI):<br>1. User Commands (text boxes)<br>2. Command Buttons<br>3. User Login<br>4. Product Preferences Selection List<br><br>List of External Outputs (EO):<br>1. Destination Recommendations Result<br>2. Accommodation Recommendations Result<br>3. Query Result<br>4. Data Update feedback<br>5. User Login Result<br>List of External Queries (EQ):<br>1. Referencing Destination Context Ontology<br>2. Referencing Tourism Asset Database<br>3. Referencing Accommodation Ontology |

**Figure 5.4 Function Points Counting Procedure for Nigerian-TIS**

After the identification of the function points in the Nigeria-TIS, the COCOMOII standard tables (see Appendix I) were used for constructing the unadjusted function points as shown in Table 5.6. Twenty percent was added to all the obtained function points in order to cater for any missed component due to unspecified requirements (Boehm et al., 2000). The unadjusted function points for the Nigerian-TIS software in COCOMO II is 120.

**Table 5.6 Constructing Function Points**

| **Internal Logical Files (ILF)** | | | |
|---|---|---|---|
| Logical File | Data Element Types | Record Element Types | Complexity |
| 1 | 19 | 4 | Low |
| 2 | 8 | 3 | Low |
| **External Interface Files (EIF)** | | | |
| Logical File | Data Element Types | Record Element Types | Complexity |
| 1 | 16 | 4 | Average |
| 2 | 120 | 4 | High |
| 3 | 4 | 4 | Low |
| 4 | 16 | 4 | Low |

| | | | |
|---|---|---|---|
| 5 | 5 | 2 | Low |
| **External Inputs** | | | |
| External Inputs | File Types Referenced | Data Element Types | Complexity |
| 1 | < 2 | 15- Above | Low |
| 2 | < 2 | 5-15 | Low |
| 3 | < 2 | 1-4 | Low |
| 4 | < 2 | 15- Above | High |
| **External Outputs** | | | |
| External Outputs | File Types Referenced | Data Element Types | Complexity |
| 1 | 2 – 3 | 1-5 | Low |
| 2 | < 2 | 1-5 | Low |
| 3 | < 2 | 1-5 | Low |
| 4 | < 2 | 1-5 | Low |
| 5 | < 2 | 1-5 | Low |
| **External Oueries** | | | |
| External Outputs | File Types Referenced | Data Element Types | Complexity |
| 1 | < 2 | 5 | Low |
| 2 | < 2 | 4 | Low |
| 3 | < 2 | 5 | Low |
| **Unadjusted Function Points** | | | |
| | Low | Average | High |
| ILF | (2) X 7 | X 4 | X 6 |
| EIF | (3) X 7 | (1) X 10 | (1) X 15 |
| EI | (3) X 3 | X 4 | (1) X 6 |
| EO | (4) X 4 | X 5 | X 7 |
| EQ | (3) X 3 | X 4 | X 6 |
| Total Unadjusted Function Points: 100<br>Input for COCOMO II: 100* 1.2 = 120 (20% more due to other uncounted function points that might arise when more thorough requirements review has been conducted). | | | |

### 5.5.1.2 Determining the Scale Factors

The scale factors values used for the estimation are shown in Table 5.7, while the justifications for the values used are given as follows:

**Precedentedness (PREC)**: This reflects the similarities of a current project to projects that had been undertaken in the past. In our case, this factor is dimmed to be very low because the development team (actually only one person was involved) that will implement the system has no prior similar project experience at all.

**Development flexibility (FLEX)**: This refers to the level of suppleness associated with what actually must be developed versus the pre-established requirements and interface specifications of the software. This is assumed to be nominal since little information on strict adherence to conformance is specified.

**Architecture/Risk Resolution (RESL)**: This factor defines the need for the extension of the architecture being completely specified and major risks being eliminated. In the case at hand, the scope of the project does not demand going into details of risk management and thus, this factor is assumed to be very low.

**Team cohesion (TEAM)**: This accounts for the synergy factor or project turbulence factor due to ease or difficulty of synchronizing the views of project stakeholders like users, customers, developers, maintainers, interfacers, etc. This is assumed to be very high because only one person was involved in dictating the pace of the project with a few errand staffs.

**Process maturity (PMAT)**: This is a measure of the quality of software process used for development based on the SEI Capability Maturity Model Integration (CMMI) standard. Since the project is being handled as a pilot study, it is considered far from being fully professional, the PMAT factor that corresponds to CMMI – level 3 is assumed, which represents an averagely acceptable level of process maturity of the development.

**Table 5.7 Estimating k Exponent**

| **Scale Factors** | Rating (5-0) |
|---|---|
| Precedentedness (PREC) | 5 |
| Development Flexibility (FLEX) | 4 |
| Architecture/Risk Resolution (RESL) | 5 |
| Team Cohesion (TEAM) | 2 |
| Process Maturity (PMAT) | 2 |
| **Sum** | 18 |
| **K = 1.01 + (Sum/100) =** 1.19 | |

### 5.5.1.3 Determining Cost Drivers (Effort Multipliers)

Seventeen cost drivers that are divided into four categories were used to determine the effort in Person Months required for the software. The justifications for the values assigned to each of the cost drivers are presented as follows:

**Required System Reliability (RELY)**: This measures the extent to which the software must perform its intended function over a period of time. The system is expected to be reliable, however failure is not considered hazardous. The failure might cause some losses, but not considerable enough to be a major concern. Therefore, a nominal value less than 1 is assigned.

**Database size (DATA)**: This measure attempts to capture the effect that large data requirements have on product development. The rating is determined by dividing the database size (in bytes) by the program size (SLOC). The size of the database is important for consideration because of the effort required to generate the test data that will be used to test the program. This is determined to be much because of the data-intensive nature of the application.

**Product complexity (CPLX)**: This is the measure of the perception of how complex the product is. In this case a nominal value of is assumed.

**Reusability (RUSE)**: This is the level of reuse required in the project. This assumed to be nominal by implicit assessment since the project description is not emphatic on reuse.

**Documentation (DOCU)**: This is a measure of the documentation suitable for life-cycle needs. This is given a nominal value because right-sized documentation is assumed.

**Execution time (TIME)** and **Storage (STOR)**: These are platform constraints that are not considered as significant in this project context, the availability of capable hardware platform is assumed, hence nominal values have been assigned to both of them.

**Platform Volatility (PVOL)**: The platform volatility (PVOL) refers to the frequency of change of hardware/software that the services rely on to perform their own tasks. In this case, no major change is expected in a 12-month period that can radically alter the course of events of development although minor changes might occur along the line. Therefore, a nominal value is assigned.

**Personnel Continuity (PCON)**: This is the measure of stability of personnel in the project development team. This is given a nominal value since only one personnel is involved with an assumption that the project will be completed before any change of personnel takes effect.

**Analyst Capability (ACAP), Programmer Capability (PCAP)**: These are measures of the ability and competence of Analyst and programmers in the project team. Since the programmer has previous programming experience and has worked on other projects that required the use of analytical ability as well as programming ability, the analyst capability (ACAP) and programmer capability (PCAP) multipliers are assumed to be very high.

**Analyst Experience in Project Domain (AEXP), Programmer Experience in Project Domain (PLEX), Language Tool Experience (LTEX)**: These are measures of the experience of the analyst and the programmer in the project application domain. Although the analyst and programmer do not have prior experience in TIS development, some knowledge of web-based development is available, hence AEXP, PEXP and LTEX are considered relatively high.

**Toolsets (TOOL)**: This is the measure of tool-support available for the development process. It is expected that the development will leverage the rich tool-support available for the development. Hence this is assumed to be relatively high.

**Multisite Working and Quality of Communication (SITE)**: This is the measure of the extent of multi-site working and quality of site communication in development. The multisite development multiplier is set to be high because the degree of site collocation and communication support is relatively high.

**Development Schedule (SCED)**: This measure the schedule constraint imposed on the project team developing the software. The ratings are defined in terms of the percentage of schedule stretch-out or acceleration with respect to a nominal schedule for a project requiring a given amount of effort. In this case, there was no formal agenda with regard to stretch out and thus, we assumed a nominal value. Table 6.8 presents a view of the values for the different cost drivers.

**Table 5.8 Estimating Effort Multiplier (m)**

| | **Cost Drivers** | |
|---|---|---|
| **Product Attributes** | Required System Reliability (RELY) | 0.9 |
| | Complexity of System Modules (CPLX) | 1 |
| | Extent of Documentation Required (DOCU) | 1.2 |
| | Size of Data Used (DATA) | 1.5 |
| | Required % of Resuable Components (RUSE) | 1 |
| **Computer Attributes** | Execution Time Constraints (TIME) | 1 |
| | Volatility of Development Platform (PVOL) | 1 |
| | Memory Constraints (STOR) | 1 |

| | | |
|---|---|---|
| **Personnel Attributes** | Capability of Project Analysis (ACAP) | 0.3 |
| | Personnel Continuity (PCON) | 1 |
| | Programmer Experience in Project Domain(PEXP) | 0.8 |
| | Programmer Capability(PCAP) | 0.3 |
| | Analyst Experience in Project Domain (AEXP) | 0.5 |
| | Language and Tool Experience (LTEX) | 0.5 |
| **Project Attributes** | Use of Software Tools (TOOL) | 0.5 |
| | Development Schedule Compression (SCED) | 1 |
| | Extent of multi-site Working and Quality of Site Communication(SITE) | 0.5 |
| | **Effort Multiplier (M)** | 0.00729 |
| | **COCOMO II (in Person Months)** | **3.7** |

Given that the *autoeffort* is set to 0, using the formula in equation 5.1 then

COCOMO II for Nigerian-TIS = $2.5 * (120^{1.19})* 3.7 = 5.4$ Person Months.

Using the same procedure the effort in person months for the Ghana-TIS and Ivorian-TIS were computed to be 5.4 and 5.7 Person Months for unadjusted function point values (*size*) of 120 and 125 respectively. Hence, the total effort for the three products sums up to 16.5 person months.

### 5.5.2 Estimating Effort in PLONTOREC

The effort expended in PLONTOREC is computed in person months using an augmented SPL estimation model (see Section 3.2.3) given as:

$$\mathbf{E_{plontorec} = E_{org} + E_{dom} + E_{onto} + E_{ontoupdate} + N * (E_{reusewith} + E_{uniquewith} + J * E_{updatewith})}$$

Hence for,

Number of TIS products = 3

$E_{org} = 0.6$

$E_{dom} = 3.6$

$E_{onto}$ (for 2 ontologies) = 1.8

$E_{ontoupdate}$ (one cycle, estimated average) = 0.3

$E_{reusewith}$ (estimated average) = 0.5

$E_{uniquewith}$ (estimated average) = 0.3

$J = 4$ (for quarterly content updates)

$E_{updatewith} = 0.06$

Recall that

$\mathbf{E_{org}}$: Effort to introduce the product line, adapt the organization, train staff etc. (In this case, only one staff was involved assisted by available student support in non-technical areas such as data gathering)

$\mathbf{E_{dom}}$: Effort expended in domain engineering for the development of core assets, cost of commonality and variability analysis.

$\mathbf{E_{onto}}$: Effort expended in the development of relevant ontologies.

$\mathbf{E_{ontoupdate}}$: Effort expended in updating content of ontologies after initial development and its maintenance.

**N**: number of TIS products in the product line.

$\mathbf{E_{reusewith}}$: Average effort in application engineering for the reuse of existing core assets e.g. choosing, configuration, searching and integration of core assets.

$\mathbf{E_{uniquewith}}$: Average effort to extend core assets base with core assets unique to a product, effort with manual adaptations of core assets after creation.

**J**: Average planned number of content update cycles for one TIS product.

$\mathbf{E_{updatewith}}$: Average effort of updating the product-related core assets in the core asset base;

Therefore, estimate for PLONTOREC in person months

$E_{plontorec} = 0.6+3.6+1.8+0.3+ 3(0.24 + 0.5 + (4*0.06)) = 9.24$

### 5.5.3 Result and Discussion

The first advantage derived was the significant reduction in the effort expended on development. An expert-based estimate for the three TIS products used in our case study predicted between 4.5 - 5 *Person Months* for each product given that a minimum of Capability Maturity Model (CMM) Level-3 process standard is attained in development and other deciding factors remain stable. This represents a maximum of about *15 person months* for the three products. This budget estimate is also not so distant from the total estimate of *16.5 person months* was obtained by using the post architecture COCOMO II to estimate the required efforts for the three TIS products. However, with PLONTOREC *9.24 person months* was expended in the development of the three TIS products. The reduction in person months was due to the fact that relatively minimal effort was spent in application engineering. In our specific experience, 5.4 person months was expended on the combination of ontology engineering and domain engineering endeavours. Also, as expected the realization of the first variant product (Nigerian-TIS) took some time, because we needed to master the challenges of how best to customize our domain components. Also, the bulk of our domain testing was executed using the first product variant. Thereafter composing the two other products was relatively very fast. Overall, the difference in the effort in person months between the instance when PLONTOREC is used and when it is not used accounts for about 44% gain in development cost.

Secondly, the inherent benefits of reuse-oriented approach like PLONTOREC came to the fore in the course of our case study. During application testing, many of the bugs and required functionality adjustments that needed to be made in specific TIS products were traceable to specific core components. Therefore, all that we had to do was to fix the concerns that pertain to the respective core components and the effects of these corrections were automatically propagated to the various products within the product line. This obviously led to a reduction in the time and effort expended on maintenance. This also gives an indication that the PLONTOREC approach will allow most of future maintenance concerns to be centrally attended to.

PLONTOREC also created an avenue for proactive evolution of content and future extensions. In our specific case study the product line was based on a particular reference architecture derived from the requirements of three countries. However, the first set of working prototype implementations did not cover the full scope of the product line model, but there remain ample opportunities for future extensions and products evolution based on a predefined versioning scheme. Drawing from our implementation experience for example, it is obvious that to realize extended versions of the TIS products all that needed to be done is to add new domain content components such as travel ontology, restaurant ontology, travel recommender and restaurant recommender components that have been specified in the reference architecture but are currently missing. Also, other future additions could be made to the reference architecture based on the dynamics of user requirements in the specific domain which will in turn provide a basis for new product variants with added features and advanced functionalities to evolve.

The PLONTOREC approach enabled the generation of various kinds of knowledge-based tourism recommendations that pertain to the three countries on a relatively cheap platter compared to if a single product development approach had been adopted. Due to the similarity in requirements, it became possible to implement the core intelligent functionalities of recommendation as domain components and then populate products with contents that are unique to them. This ensured that such intelligent capabilities got systematically propagated to the generated products, which is obviously cheaper than pursuing a single product approach.

## 5.6 POSSIBILITIES FOR GENERALIZATION OF THE RESULTS

Having shown that PLONTOREC worked for one product line in the presented context, we therefore postulate that PLONTOREC can indeed be applied to create other product lines in the tourism domain, if sufficient commonalities exist and the variabilities of requirements among different tourism entities (service providers, support outfits etc.) or tourism organizations (continental, national, regional, local, enterprise etc.) are well known.

This connote that the aggregate of the tourism requirements in a domain can be represented by a conceptual product line model from which a set of core asset components, a feature tree model of each product, a set of relevant ontologies, the construction specification and a set of variant TIS products can be generated. PLONTOREC is particularly applicable in all cases where assorted kinds of intelligent recommendations on tourism objects are required. Instances of these include developing a product line of TIS for: 1) states, regional governments within a country; 2) a chain of hotels; 3) a group of religious organizations (promoting religious tourism); 4) a network of Destination Management Organizations (DMO); 5) a network of tour operators etc.
In all of these instances PLONTOREC holds the potential to create not only TIS platforms from which credible knowledge-based recommendations that foster user's confidence can be generated, but also that which will engender the proactive evolution of such TIS products in

tandem with  future emerging user requirements within the specific tourism domain concerned.

## 5.7 SUMMARY AND DISCUSSION

In this chapter a report of the procedure adopted for the evaluation of the PLONTOREC process and its products is presented. It is shown that the variant TIS products generated from the PLONTOREC approach had substantially favourable usability rating from users based on the empirical test conducted, which is very crucial for a people-oriented service delivery platform like e-tourism. Furthermore, the case study scenario has demonstrated the applicability of PLONTOREC in a real-life context and proved the viability of the PLONTOREC approach. This is because PLONTOREC produced measurable reduction in time and cost of development, demonstrated the potential to reduce maintenance cost, and facilitated significant improvements in the quality of recommendations obtained from the variant TIS products it generated.

The case study therefore, successfully validates PLONTOREC as platform for generating dependable and intelligent knowledge-based recommendation and one that has the potential to engender dynamic product evolution in the tourism.

# CHAPTER SIX
# SUMMARY AND CONCLUSION

This Chapter summarizes and discusses the contributions of the thesis, and presents an outlook of the opportunities for future research work. The thesis presented a specialized product line approach for ontology-based recommendations in e-Tourism Systems.

## 6.1 SUMMARY

The thesis has shown that tourism recommendation services are particularly important because of the information-intensive nature of the tourism industry where access to useful information guide brings immense benefits to all stakeholders within the tourism value chain.

However, tourism recommendation services are currently not prevalent on most of the existing e-tourism platforms (TIS), and where such exist, they need to be made more dependable in a way that fosters users' confidence. Also such e-tourism systems must be able to evolve with the dynamic nature of user requirements in tourism in order to maintain their relevance.

The thesis intervened by introducing a novel unified solution approach to these two concerns called **P**roduct **L**ine for **O***n*tology-based **T***o*urism **Rec**ommendations (PLONTOREC), which provides a process platform for the creation of variant e-tourism systems that can evolve proactively in response to dynamic user requirements and also offer dependable knowledge-based recommendations. PLONTOREC is a hybrid of software product line engineering and ontology engineering that is dedicated to the production of recommendation-intensive Tourism Information Systems.
The PLONTOREC approach consist of four main sub-processes which are: 1) Product Line Management (which provides the necessary managerial guide and organizational control that complements the technical aspects of PLONTOREC); 2) Ontology Engineering (which is concerned with the development of the reusable knowledge artifacts which are typically ontologies that are needed for knowledge-based recommendations); 3) Domain Engineering

(which is concerned with the construction of all reusable software assets that are used for building variant TIS products in the product line); and 4) Application Engineering ( which is concerned with the creation of specific TIS products through the reuse of core assets created in domain engineering. In Addition, PLONTOREC is based on a set of assumptions which defines the condition for its optimal applicability. These are:

- New products evolve by composition, using existing components in the common asset base;
- New versions of TIS products are variations of existing ones, having many things in common with the old versions; and also
- The points of variability are minimal and predictable.

The thesis provided a validation of the PLONTOREC approach by using a case study of TIS product line development involving three countries in the West African Sub-region (Nigeria, Ghana, and Cote D'ivoire) in order to demonstrate the applicability and viability of PLONTOREC in a real-life context.

The thesis made some significant contributions. Firstly, it has opened-up a new perspective on how to tackle the problem of dynamic user requirements in the e-tourism domain by offering a clear demonstration of the viability of software product line engineering as a solution approach to solving this problem. Secondly, an innovative 3-dimensional approach was introduced to destination recommendation with the use of ontological representation of contextual information on social attributes of prospective destinations in order to improve the dependability of destination recommendations. Thirdly, the creation of a suite of tourism ontologies as semantic web contents for the tourism value chain within the West African sub-region represents a first attempt at creating an interoperable platform for the sharing and reuse of tourism information, and tourism knowledge within the West African sub-region. Lastly, the thesis makes a first attempt to create a product line of recommendation-intensive TIS products using the PLONTOREC approach. PLONTOREC is a novel specialized and repeatable software product line process that engenders the generation of dependable and intelligent recommendations in e-tourism systems and facilitates the proactive evolution of such e-tourism systems in tandem with dynamic user requirements.

## 6.2 CONCLUSION

This research work tackled two concerns in the global e-tourism industry which is the need for more dependable recommendations and the need for e-tourism systems to proactively evolve with emerging users need. It has succeeded in providing a unified solution model for reasonably improved intelligence and dependability of recommendations of e-tourism systems and offered a way for developers of e-tourism systems to manage the dynamic nature of user requirements in tourism.

The research has also provided a theoretical and product-oriented framework that can be leveraged for the generation of recommendation-intensive tourism support systems in the geographic contexts of developing countries where none of such platforms is known to exist.

Finally, the results of this research endeavour if adopted will give the quality boost needed in most parts of Africa where tourism is largely undeveloped. For example the ontology artifacts created in the course of this research offers a potential platform for data interoperability, knowledge reuse, and business model standardization within the West African tourism value chain. Also the e-tourism portal prototypes if extended with more detailed requirements will provide a platform for increased publicity and promotion of tourism as a veritable tool for economic development in the countries concerned.

## 6.3 FUTURE WORK

The thesis provides several opportunities for further research in the immediate future. The PLONTOREC approach as implemented in this thesis directly inherited some limitations from its parent concepts of software product line engineering and ontology engineering. Notably, there exist ample research possibilities to enhance the concept in the following areas:

**i) Automated Tool Support**

There are different aspects of the PLONTOREC approach that may be enriched with automated tool support. Some of these include:

- **Domain Requirements Engineering**: The elicitation and validation of domain requirements is largely a human-centered activity, which is based on the experience of the experts involved. It will be interesting to see the possibilities of having an expert system that can reasonably emulate human capability in these areas or at least offer credible decision support in the elicitation and validation of domain requirements to minimize human effort. One candidate intelligent model that can be explored is Case-Based Reasoning (CBR). CBR entails solving new problems based on experience that have been gathered from previous episodes. One thinks that a CBR-based expert system that is fortified with a rich knowledgebase of relevant tourism domain knowledge looks very promising in this regard.
- **Architecture Creation and Evaluation**: The existing methods of architecture evaluation such as the Architecture Trade-off Analysis Method ATAM (Kazman et al., 1998), Scenario-based Architecture Analysis Method (SAAM) (Ionita et al., 2002) are largely human-centered; it will be interesting to see how an automated architecture evaluation scheme can be executed using tourism domain knowledge. Also, having a tool that can automatically generate the reference product line architecture from a given set of domain requirements is a fascinating dimension that could be introduced to fast track the PLONTOREC process.
- **Project Cost Prediction**: The Product Line Management activity of PLONTOREC could be greatly enhanced if an automated tool can be used to predict the effort required for the project based on some objective parameters extracted from the domain requirements and application requirements. The prediction model could be based on statistical techniques like regression or a machine learning concepts like Artificial Neural Networks.
- **Customization GUI Tool**: Most of the core assets used in this research work was implemented as parameterizable content components that were invoked from program codes. A better approach would be to create a tool that will enable the parameters for specific components in the core asset repository to be supplied from a GUI interface and have a customized version of the content component automatically created by calling relevant files and compilers.

**ii) Group Recommendations**

The work done in this thesis focused on provision of personalized recommendation services for individuals (i.e. single user recommendations). The aspect of recommendations that pertain to a group or team of people was not considered. This is one dimension that could be incorporated in the future, which have the capacity to enhance the quality of the products generated through the PLONTOREC approach.

**iii) Mobile Computing**

The scope of implementation of this research has been limited to web-based e-tourism systems. However, the possibilities of adapting or extending the PLONTOREC approach for the realization of mobile TIS product line looks very promising as future research endeavour.

**iv) Semantic Query Processing**

Ontology engineering has been engaged in this research to develop knowledgebase ontologies that contain facts about the specific tourism domain. In future research endeavours, domain ontologies that would serve as lexical databases for specific tourism domains could be included in the PLONTOREC framework. By doing this, the domain ontologies could be used as a basis for enabling the generated TIS with natural language processing capabilities which are currently lacking in existing e-tourism systems. Query reformulation techniques and content summarization techniques in the field of natural language processing are viable computational models that can be used to realize this objective. Other related aspects such as native language processing can also be explored on the PLONTOREC framework in future research work.

## REFERENCES

- Abowd, G., Atkeson, C., Hong, J., Long, S., Kooper, R. and Pinkerton, M., " Cyberguide: a mobile context-aware tour guide", *ACM Wireless Networks*, 5(3), pp. 421-433, 1997.
- Adomavicius G. and Tuzhilin, A., "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions", *IEEE Transactions on Knowledge and Data Engineering,* 7(6), pp. 734-749, 2005.
- Adomavicius, G. and Tuzhilin, A., "Expert-Driven Validation of Rule-Based User Models in Personalization Applications," *Data Mining and Knowledge Discovery*, Vol. 5, Nos. 1 and 2, pp. 33-58, 2001.
- Adomavicius, G., Sankaranarayanan, R., Sen, S. and Tuzhilin, "A: Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach", *ACM Transactions on Information Systems*, 23(1), pp. 103-145, 2005.
- Ali, K. and van Stam, W., "TiVo: Making show recommendations using a distributed collaborative filtering architecture", *In Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, Seattle, WA, USA, pp. 394–401, 2004.
- Alspector, J., Kolcz, A., and Karunanithi, N., "Comparing feature-based and clique-based user models for movie selection", *In Proceedings of the Third ACM Conference on Digital Libraries*. ACM Press, Pittsburgh, PE, USA, pp. 11–18, 1998.
- America P., Obbink H., Muller J., van Ommering R., "COPA: A Component-Oriented Platform Architecting Method for Families of Software Intensive Electronic Products", *In Proceedings of the First Conference on Software Product Line Engineering*, 2000.
- Arango, G., "Domain Analysis", In Marciniak, J. (Eds.), *Encyclopedia of Software Engineering*, Volume 1, pp. 423-434, Wiley, 1994.
- ArchStudio 4 – Software and Systems Architecture Development Environment (2007). Institute for Software Research, University of California, Irvine, Available at: *http://www.isr.uci.edu/projects/archstudio*
- Arpirez, J., Corcho, O., Fernandez-Lopez, M., Gomez-Perez, A., "WebODE in a nutshell", *AI Magazine*, 2003.

- Atkinson, C., Bayer, J., Bunse, C. , Kamsties, E, Laitenberger, O., Laqua, R., Muthig, D., Paech, B., Wüst, J., Zettel, J., "Component-based product line engineering with UML", Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 2002.
- Baeza-Yates, R. and Ribeiro-Neto, B., "Modern Information Retrieval", Addison-Wesley, Reading, MA, USA, 1999.
- Balabanovi´c, M. and Shoham, Y. "Fab: Content-based, collaborative recommendation", *Communications of the ACM*, 40(3), pp. 66–72, 1997
- Balzerani, L., Di Ruscio, D., Pireantonio, A., De Angelis, G. "A Product Line Architecture for Web Applications", *In Proceedings of the ACM symposium on Applied computing*, Santa Fe, New Mexico, 1689 – 1693, 2005.
- Bass, L. and Kazman, R., "Software Architecture in Practice", Addison-Wesley, M.A. pp. 19-23, 2003.
- Bastarrica, M. C., Hitschfeld-Kahler, N., Rossel, P. O., "Product Line Architecture for a Family of Meshing Tools", *International Conference on Software Reuse (ICSR)*, Torino, pp. 403-406, 2006.
- Basu, C., Hirsh, H. and Cohen, W. "Recommendation as Classification: Using Social and Content-Based Information in Recommendation," *Recommender Systems*, Papers from 1998 Workshop, Technical Report WS-98-08, AAAI Press 1998.
- Bechhofer S., Horrocks I., Goble C., Stevens R., "OilEd: a reason-able ontology editor for the Semantic Web", *In: Baader F, Brewka G, Eiter T (eds) Joint German/Austrian conference on Artificial Intelligence (KI'01).* Vienna, Austria. (LNAI 2174) Springer-Verlag, Berlin, Germany, pp. 396–408, 2001.
- Bernaras, A., Laresgoiti, I., Corera, J., "Building and reusing ontologies for electrical network applications", *Lecture Notes in Artificial Intelligence* (LNAI 1621) Springer-Verlag, Berlin, Germany, pp. 49-66, 1996.
- Bhargava, H. K., Sridhar, S. and Herrick, C., "Beyond Spreadsheets: Tools for Building Decision Support Systems", *IEEE Computer*, 32(3), pp. 31-39, 1999.
- Billsus, D. and Pazzani, M., "User Modeling for Adaptive News Access," *User Modeling and User-Adapted Interaction*, Vol. 10, Nos. 2-3, pp. 147-180, 2000.
- Blazquez, M., Fernandez-Lopez, M., Garcia-Pinar, J., Gomez-Perez, A., "Building Ontologies at the Knowledge Level using the Ontology Design Environment", *In:*

*Gaines B.R., Musen, M.A. (eds) 11th International Workshop on Knowledge Acquisition, Modeling and Management (KAW'98)*, Banff, Canada, SHARE 4: pp. 1-15, 1998.

- Bockle, G, Clements, P., McGregor, J.D., Mathig, D., Schmid, K: *"Calculating ROI for Software Product Lines", IEEE Software*, 21(3), pp. 23-31, 2004.
- Boehm, B., Abts, C., Winsor Brown, A., Chulani, S. Clark, B.K., Horowitz, E., Madachy, R., Reifer, D., Steece, B., "Software Cost Estimation with COCOMO II", Prentice Hall PTR, Upper Saddle River, N.J., 2000.
- Booch, G., Jacobsen, I. and Rumbaugh, J. OMG Unified Modeling Language Specification, 2000. Available at *www.omg.org/technology/documents /formal/unifiedmodelinglanguage.htm*
- Bosch J., "Design & Use of Software Architectures", Addison-Wesley, 2000.
- Bosch, J. and Svahnberg, M., "Evolution in Software Product Lines: Two Cases", *Journal of Software Maintenance*, 11(6), 391-422, 1999.
- Breese, J., Heckerman, D., and Kadie, C., "Empirical analysis of predictive algorithms for collaborative filtering", *In Proceedings of the Fourteenth Annual Conference on uncertainty in Artificial Intelligence*, Morgan Kaufmann, Madison, WI, USA, pp. 43–52, 1998.
- Burke, R., "Knowledge-based Recommender Systems:, *Encyclopedia of Library and Information Systems*, 69(32), pp. 180-200, 2000.
- Burke, R., Hammond, K. & Cooper, E., "Knowledge-based navigation of complex information spaces", *In Proceedings of the 13th National Conference on Artificial Intelligence*, pp. 462-468. Menlo Park, CA: AAAI Press, 1996.
- Burke, R., Hammond, K., and Young, B., "The FindMe Approach to Assisted Browsing", *IEEE Expert*, 12(4): 32-40, 1997.
- CAPITAL ITTS, *http://cordis.europa.eu/data/PROJ_FP5/ACTIONeqDndSESSIONeq112422005919nd DOCeq405ndTBLeqEN_PROJ.htm*. (Accessed 03/04/07)
- Cardoso, J**.**, "Approaches to Developing Semantic Web Services", *International Journal of Computer Science*, 1(1), pp. 8-21, 2004.

- Cheverst, K., Davies, N., Michell, K., Friday, A., Efstratiou, C., "Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences", *CHI Letters*, 2(1), pp. 17-24, 2000.
- Claypool, M., Gokhale, A., Miranda, T., Murnikov, P. Netes, D. and Sartin, A "Combining Content-Based and Collaborative Filters in an Online Newspaper," *Proc. ACM SIGIR '99 Workshop Recommender Systems: Algorithms and Evaluation*, Citeseer.ist.psu.edu/226009.html, 1999.
- Clement, P. and Northrup, L., "Software Product Lines, Practices and Patterns", Addison-Wesley, 2002.
- Clements, P., "SEI's Views and Beyond Approach for Documenting Software Architectures with ANSI-IEEE 1471-2000", 2005.
- Cleverdon, C., "The Cranfield tests on index language devices. *Aslib Proceedings 19*, pp. 173-192, 1967.
- Corcho, Ó, Gómez-Pérez, A., González-Cabero, R. and Suárez-Figueroa, M., "ODEval: a tool for evaluating RDF(S), DAML+OIL, and OWL concept taxonomies", *IFIP WG12.6 -- First IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI2004)*. Toulouse, France, 2004.
- Cortes, C., Fisher, K., Pregibon, D., Rogers, A., and Smith, F., "Hancock: A Language for Extracting Signatures from Data Streams," *Proc. Sixth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 9-17, 2000.
- Cote D'ivoire Tourism *http://www.tourisme.com* (accessed 19/07/08)
- CRUMPET, *http://www.eml-development.de/english/research/crumpet/index.php*
- Dahlen, B. J., Konstan, J. A., Herlocker, J. L., Good, N., Borchers, A., and Riedl, J., "Jumpstarting movielens: User benefits of starting a collaborative filtering system with dead data". TR 98-017, University of Minnesota, 1998.
- Daramola J.O, Adigun, M.O, and Ayo, C.K., "Building an Ontology-based Framework for Tourism Recommendation Services", *In W. Hopken and U. Gretzel, and M. Sigala (eds.) Information and Communication Technologies in Tourism* (pp. 135-147), Springer Wien New York, 2009.
- Daramola J.O, Adigun, M.O, and Olugbara, O.O., "A Product-Line Architecture for Evolving Intelligent Component Services in Tourism Information Systems", *In P. O'*

*Connor, W. Hopken and U. Gretzel (eds.) Information and Communication Technologies in Tourism (pp. 117-145)*, Springer Wien New York, 2008.

- Dashofy, E.M., van der Hoek, A. and Taylor, R.N., "A Highly-Extensible XML-Based Architecture Description Language", *Proceedings. Working IEEE/IFIP Conference on Software Architecture*, Amsterdam, Netherlands: pp. 103-112, 2001.
- Dashofy, E.M., van der Hoek, A. and Taylor, R.N., "A Highly-Extensible XML-Based Architecture Description Language", *Proceedings. Working IEEE/IFIP Conference on Software Architecture*, Amsterdam, Netherlands: 103-112, 2001.
- Dell'Erba, M., Fodor, O., Ricci, F., Werthner, H., "Harmonise: a Solution for Data Interoperability", In J.L. Monteiro, P.M.C. Swatman, and L.V. Tavares (eds.), *Proceedings of Second IFIP Conference on E-Commerce, E-Business, E-Government* (pp. 433-445), Kluwer, Boston, 2002.
- Dellarocas, C. "The Digitization of Word of Mouth: Promise and Challenges of Online Feedback Mechanisms", *Management Science*, Vol. 49, No. 10, pp. 1407-1424, 2003.
- Deng, G., Schmidt, D.C., Gokhale, A., Gray, J., Lin, Y., and Lenz, G., "Evolution in Model-Driven Software Product-line Architectures", available at: *http://www.cs.wustl.edu/~schmidt/PDF/MDE-PLA-BookChap-v10.pdf*, 2007
- Deshpande, M. and Karypis, G., "Item-based top-n recommendation algorithms", *ACM Transactions on Information Systems* 22(1), pp. 143–177, 2004.
- Dey, A., "Understanding and using context", *Personal and Ubiquitous Computing Journal*, Vol. 5, No. 1, pp. 4-7.
- Duineveld, A., Studer, S., Weiden, M., Kenepa, B., Benjamis, R., "WonderTools? A comparative study of ontological engineering tools", *In: Proc. 12th Knowledge Acquisition Workshop (KAW99)*, Banff, 1999.
- Dwyer, F., "Customer Lifetime Valuation to Support Marketing Decision Making", *Journal of Direct Marketing*, Vol. 3, No. 4, pp. 8-15, 1989.
- ECTRL Solutions, Trento, Italy, *http://www.ectrlsolutions.com*. (Accessed 04/03/08)
- Ehrgott, M., "Multicriteria Optimization", Springer Verlag, New York, 2000.
- Expedia Inc., *http://wwww.expedia.com* (Accessed 06/06/2007)
- Ezran M., Morisio M., Tully, C., "Practical Software Reuse", Springer, 2002.

- Farquhar, A, Fikes, R & Rice, J., "The ontolingua server: a tool for collaborative ontology construction", *Proceedings of the Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, 9–14 November, 1996.
- Fawcett, T. and Provost, F., "Combining Data Mining and Machine Learning for Efficient User Profiling," *In Proc. Second Int'l Conf. Knowledge Discovery and Data Mining (KDD-96)*, 1996.
- Felfernig, A. and Kiener, A., "Knowledge-based Interactive Selling of Financial Services using FSAdvisor", *In 17th Innovative Applications of Artificial Intelligence Conference (IAAI'05)*, pp. 1475–1482, Pittsburgh, Pennsylvania, 2005.
- Felfernig, A., Friedrich, G., Jannach, D., and Zanker, M., "An Integrated Environment for the Development of Knowledge-Based Recommender Applications", *International Journal of Electronic Commerce*, 11(2), pp. 11-34, 2006.
- Felfernig, A., Gordea, S., Jannach, D., Teppan, D., and Zanker, M., "A Short Survey of Recommendation Technologies in Travel and Tourism", *ÖGAI Journal*, 25(2), pp. 1-7, 2006.
- Fensel, D. and Perez, A.G., "OntoWeb deliverable 1.3: a survey on ontology tools", Technical report, IST OntoWeb Thematic Network, 2002.
- Ferman, M., Errico, J., van Beek, P., and Sezan, I., "Content-based filtering and personalization using structured metadata", *In Proceedings of the Second ACM/IEEE-CS Joint Conference on Digital Libraries*, ACM Press, Portland, OR, USA, pp. 393–393, 2000.
- Fernandez-Lopez, M., Gomez-Perez, A., Juristo, N., "METHONOLOGY: From Ontological Art Towards Ontological Engineering", Spring Symposium on Ontological Engineering of AAAI, Stanford University, California, pp. 33-40, 1997.
- Fernandez-Lopez, M., Gomez-Perez, A., Pazos, A, Pazos, J.: Building a Chemical Ontology Using Methontology and the Ontology Design Environment, *IEEE Intelligent Systems & their applications*, 4(1), pp. 37-46, 1999.
- Gamma, E., Helm, R., Johnson, R., Vlissides, J., "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley, 2005.

- Gangemi, A., Pisanelli, D., Steve, G.: An Overview of the ONIONS Project: Applying, Ontologies to the Integration of Medical Terminologies. *Data & Knowledge Engineering*, 31(2), pp. 183-220, 1999.
- Garg, A., Critchlow, M., Chen, P., Van der Westhuizen, C., van der Hoek, A.: "An Environment for Managing Evolving Product Line Architectures", *Proceedings of the International Conference on Software Maintenance*, Amsterdam, The Netherlands, pp. 358-367, 2003.
- Garlan, D., Monroe, R. and Wile, D., "ACME - An Architecture Description Interchange Language", *In Proceedings of CASCON '97*, ACM Press: 169-183, 1997.
- Ghana Online Forum. *http://www.viewghana.com* (accessed: 16/07/08)
- Ghana Tourism, *http://www.touringghana.com* (accessed: 16/07/08)
- Ghani, R. and Fano, A., "Building recommender systems using a knowledge base of product semantics", *In Proceedings of the Workshop on Recommendation and Personalization in E-Commerce (RPEC)*, Springer-Verlag, Malaga, Spain, citeseer.ist.psu.edu/697938.html, 2002
- Goldberg, D., Nichols, D., Oki, B. M., and Terry, D., "Using collaborative filtering to weave an information tapestry", *Communications of the ACM*, 35(12), pp. 61–70, 1992.
- Gomaa, H. and Farrukh, G. "A reusable architecture for federated client/server systems", *In Proc. of Fifth Symposium on Software Reusability*, pp. 113-121, 1999.
- Gomaa, H., "Designing Software Product Lines with UML", Addison-Wesley, 2005.
- Gomez-Perez, A., "A Framework to Verify Knowledge Sharing Technology. *Expert Systems with Application,* 11(4), pp. 519-529, 1996.
- Gomez-Perez, A., "Knowledge Sharing and Reuse", *In: Liebowitz J. (ed.) Handbook of Expert Systems*, CRC Chapter 10, Boca Raton, Florida, 1998.
- Gómez-Pérez, A., Corcho-García, O., Fernández-López, M., Lehtola, A., Taveter, K., Sorva, J., Käpylä, T., Tourmani, F., Soualmia, L., Barboux, C., Castro, E., Sallantin, J., Arbant, Bonnaric, G.," Multilingual Knowledge Based European Electronic Marketplace", *D-31: Requirement, Choice of a Knowledge Representation and Tools - V 2.0 – Public*, IST (Information Society Technologies), 2001.
- Gomez-Perez, A., Fernandez-Lopez, M. and Corcho, O., "Ontological Engineering", Springer-Verlag London, 2004.

- Gomez-Perez, A., Juristo, N., Pazos, J., “Evaluation and assessment of knowledge sharing technology”, *In: Mars N (ed.) Towards Very Large Knowledge Bases: Knowledge Building and knowledge Sharing (KBKS’95)*, University of Twente, Enschede, The Netherlands. IOS Press, Amsterdam, The Netherlands, pp. 289-296, 1995.
- Good, N., Schafer, B., Konstan, J., Borchers, A., Sarwar, B., Herlocker, J., and Riedl, J., “Combining collaborative filtering with personal agents for better recommendations”, *In Proceedings of the 16th National Conference on Artificial Intelligence and Innovative Applications of Artificial Intelligence*. American Association for Artificial Intelligence, Orlando, FL, USA, pp. 439–446, 1999.
- Goodaire, E., Parmenter, M., “Discrete Mathematics with Graph Theory”, Ed. Prentice Hall. 1998.
- Goren-Bar, D. "Overcoming mobile device limitations through adaptive information retrieval", *Applied Artificial Intelligence*, Vol. 18(6), pp. 513–532, 2004.
- Gruninger, M. and Fox, M., “Methodology for the design and evaluation of ontologies *In: Skuce, D. (eds.) IJCAI’95 Workshop on the Basic ontological Issues in Knowledge sharing*, Montreal, Canada, pp. 6.1-6.10, 1995.
- Hammersley, B. “Content Syndication with RSS”, O'Reilly Media Inc, California, 2003.
- Hartmann, J., Spyns, P., Giboin, A., Maynard, D., Cuel, R., Suárez-Figueroa, M., Sure, Y., “D1.2.3 Methods for ontology evaluation”, 2005. www.starlab.vub.ac.be/research/projects/knowledgeweb/
- Henriksson, R., “Semantic Web and E-Tourism”, *www.cs.helsinki.fi/u/glinskih/semanticweb/Semantic_Web_and_E-Tourism.pdf,* 2005. (Accessed 3/02/08).
- Herlocker, J., Konstan, J. and Riedl, J., “Explaining collaborative filtering recommendations”, 2000, *citeseer.ist.psu.edu/herlocker00explaining.html*.
- Herlocker, J., Konstan, J., Borchers, A., and Riedl, J., “An algorithmic framework for performing collaborative filtering”, *In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM Press, Berkeley, CA, USA, pp. 230–237, 1999.

- Herlocker, J., Konstan, J., Terveen, G. and Riedl, T., “Evaluating Collaborative Filtering Recommender Systems”, *ACM Trans. Information Systems*, Vol. 22, No. 1, pp. 5-53, 2004.
- Herlocker, J.L., “Understanding and Improving Automated Collaborative Filtering Systems”, PhD Thesis, 2000.
- Hill, W., Stead, L., Rosenstein, M., and Furnas, G.,” Recommending and evaluating choices in a virtual community of use”, ACM Press/Addison-Wesley Publishing Co., *http://doi.acm.org/10.1145/223904.223929*, pp. 210-217, 1995.
- Hi-Touch project at: *http://icadc.cordis.lu/fepcgi/srchidadb?CALLER=PROJ_IST&ACTION=D&RCN=63604&DOC=20&QUERY=3*
- Hofmann, T. “Latent Semantic Models for Collaborative Filtering”, *ACM Trans. Information Systems*, Vol. 22, No. 1, pp. 89-115, 2004.
- Hofmeister, C., Nord, R., “From Software Architecture to Implementation with UML”, *Proceedings of the Twenty-Fifth Annual International Computer Software and Applications Conference (COMPSAC 01)*, Chicago, Illinois, pp. 113-114, 2001.
- Intelligent Mobility Agents. IM@GINE IT, European Commission, Brussels, 2004, *http://dbs.cordis.lu/fepcgi/srchidadb?ACTION=D&SESSION=296320041126&DOC=53&TBL=EN_PROJ&RCN=EP_RPG:508008&CALLER=PROJ_IST*
- Intelligent system for Tourism, available at: *ftp://ftp.cordis.europa.eu/pub/ist/docs/transport environment/intelligentsystems_for_tourism_en.pdf*
- International Function Point Users’ Group, “Function Point Counting Practices: Manual Release 4.0”, *http://*linkinghub.elsevier.com/retrieve/pii/S0164121200000157
- Ionita, T., Hammer, K., Obbink ,H., “Scenario-Based Software Architecture Evaluation Methods: An Overview”, available at: *www.win.tue.nl/oas/architecting/aimes/papers/scenario-based%20scost%20evaluation%20methods.pdf.*, 2002.
- ISO 9241-11, “Ergonomic requirements for office work with visual display terminals (VDT’s) – Part 11”, *Guidance on usability*, International Organization of Standardization, 1998.

- Jennings, A. and Higuchi, H., “A personal news service based on a user model neural network”, *citeseer.ist.psu.edu/jennings92personal.html*, 1992.
- Jiang, B., Wang, W. and Benbasat, I., “Multimedia-Based Interactive Advising Technology for Online Consumer Decision Support”, *Communication of the ACM*, 48(9), pp. 93–98, 2005.
- Kang K., Kim S., Shin E., Huh M., “Form: A Feature-Oriented Reuse Method with Domain Specific Reference Architectures”, *Annals of Software Engineering*, Vol. 5, pp. 143-168, 1998.
- Kang, K., Cohen, J., Novak, W. and Peterson, S., “Feature-oriented domain analysis feasibility study”, Carnegie Mellon University, Software Engineering Institute, *Tech. Rep. CMU/SEI-90-TR-21,* pp. 1-82, 1990.
- Kang, K., Lee J., and Donohoe, P., “Feature-oriented product line engineering”, *IEEE Software*, 19(4), 58-65, 2002.
- Karypis, G., “Evaluation of item-based top-n recommendation algorithms”, *In Proceedings of the Tenth ACM CIKM International Conference on Information and Knowledge Management*”, ACM Press, Atlanta, GA, USA, pp. 247–254, 2001.
- Kazman, R., Klein, M., Barbacci, M., Lipson, H., and Carriere, J., “The Architecture Tradeoff Analysis Method”, Carnegie Mellon University, Software Engineering Institute, *Tech. Rep. CMU/SEI-98-TR-008*, pp. 1-83, 1998.
- Kietz, J., Maedche, A., Volz, R., “A Method for Semi-Automatic Ontology Acquisition from a Corporate Intranet”, *In: Aussenac-Gilles N, Biebow B, Szulman S (eds.) EKAW’00 Workshop on Ontologies and Texts,* Juan-Les-Pins, France, CEUR Workshop Proceedings, pp. 51:4.1-4.14. Amsterdam, The Netherlands (http://CEUR-WS.org/Vol-51/), 2000.
- Knublauch, H., Musen, M., Noy, N., “Tutorial: Creating Semantic Web (OWL) Ontologies with Protégé”, 2nd International Semantic Web Conference (ISWC2003), Sanibel, Island, Florida, U.S.A, 2003.
- Konstan, J., “Introduction to recommender systems: algorithms and evaluation”, *ACM Transactions on Information Systems*, 22(1), pp. 1–4, 2004.

- Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L. and Riedl, J., “GroupLens: Applying Collaborative Filtering to Usenet News”, *Communications of ACM*, Vol. 40, No. 3, pp. 77-87, 1997.
- Krösche, J., Baldzer, J., Boll, S., “MobiDENK-Mobile Multimedia in Monument Conservation”, *IEEE Multi-Media*, 11(2):72–77, 2004.
- Krutchen, P., “Architectural Blueprints- The “4+1” View Model of Software Architecture”, *IEEE Software*, 12(6), pp. 42-50, 1995.
- Lam, S. and Riedl, J., “Shilling recommender systems for fun and profit”, *In Proceedings of the 13th International Conference on World Wide Web,* ACM Press, New York, NY, USA, pp. 393–402, 2004.
- Lam, W., Mukhopadhyay, S., Mostafa, J., and Palakal, M., “Detection of shifts in user interests for personalized information filtering”, *In Proceedings of the 19th ACM SIGIR Conference on Research and Development in Information Retrieval,* ACM Press, Zurich, Switzerland, pp. 317–325, 1996.
- Lang, K., “NewsWeeder: Learning to filter netnews”, *In Proceedings of the 12th International Conference on Machine Learning*, Morgan Kaufmann, San Mateo, CA, USA, pp. 331–339, 1995.
- Lech, T.C. and Wienhofen, L.M., “AmbieAgents: A Scalable Infrastructure for Mobile and Context-Aware Information Services”, *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, The Netherlands: 625-631, 2005.
- Lenat, D., Guha, R., “Building Large Knowledge-based Systems: Representation and Inference in the Cyc Project”, Addison-Wesley, Boston, Massachusetts, 1990.
- Lewis, D. and Sparck-Jones, K., “Natural Language Processing for Information Retrieval”, Volume 39, No.1, *Communications of the ACM*, pp. 92-101, 1996.
- Lewis, J.R., “IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use”, *International Journal of Human Computer Interaction*, 7(1), 57-78, 1995.
- Lieberman, H., “Letizia: An Agent That Assists Web Browsing”, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, citeseer.ist.psu.edu/lieberman95letizia.html, pp. 924-929, 1995.

- Linden, G., Smith, B., and York, J., “Amazon.com recommendations: Item-to-item collaborative filtering”, *IEEE Internet Computing,* Vol. 4, No. 1 (January), pp. 76-80, 2003.
- Lindgaard, G., “Usability testing and system evaluation. A guide for designing useful computer systems”, London, Chapman and Hall, 1994.
- Lutz, R. and Gannod, G., “Analysis of a Software Product Line Architecture: An experience Report”, *Journal of Systems and Software*, 66(3), pp. 253-67, 2003.
- Mannila, H., Toivonen, H. and Verkamo, A.I., “Discovering Frequent Episodes in Sequences,” *Proc. First Int’l Conf. Knowledge Discovery and Data Mining (KDD-95)*, pp. 210—215, 1995.
- Mannion M., Lewis O., Kaindl H., Montroni G., Wheadon J., “Representing Requirements on Generic Software in an Application Family Model”, *Proceedings of the International Conference on Software Reuse* (*ICSR-6)*, pp. 153-196, 2000.
- Matinlassi M., “Comparison of Software Product Line Architecture Design Methods: COPA, FAST, FORM, KobrA and QADA”, *Proceedings of the 26th International Conference on Software Engineering (ICSE’04)*, pp. 127-136, 2004.
- Melville, P., Mooney, R., and Nagarajan, R., “Content-boosted collaborative filtering for improved recommendations”, *In Eighteenth National Conference on Artificial Intelligence. American Association for Artificial Intelligence*, Edmonton, Canada, pp. 187–192, 2002.
- Middleton, S., Shadbolt, N., and De Roure, D., “Ontological user profiling in recommender systems”, *ACM Transactions on Information Systems*, 22(1), pp. 54–88, 2004.
- Mooney, R. and Roy, L., “Content-Based Book Recommending Using Learning for Text Categorization”, *In: Proceedings of the Fifth ACM International Conference on Digital Libraries*, pp. 195-204, 2000.
- Mukherjee, R., Dutta, P., and Sen, S., “MOVIES2GO: A new approach to online movie recommendation”, *In Proceedings of the IJCAI Workshop on intelligent Techniques for Web Personalization*, Seattle, WA, USA, 2001.
- Necib, C. and Freytag, J, “Query Processing Using Ontologies”, *Lecture Notes in Computer Science*, Springer/Berlin, Heidelberg, pp. 167-168, 2005.

- Nielsen, J., “Usability engineering”, Boston, MA, Academic Press, Harcourt Brace & Company, 1993.
- Niemelä E., “QADA – Quality-driven Architecture Design and Architecture Analysis”, Available at: *http://www.vtt.fi/qada/*, (January 2006)
- Nigeria Tourism Development Corporation, *http://www.nigeriatourism.net*. (Accessed 04/03/08).
- Noy, F., Musen, M., “Anchor-PROMPT: Using Non-Local Context for Semantic Matching”, *In: Gomez-Perez A, Gruninger, M., Stuckenschmidt, H., Unschold, M. (eds) IJCAI’01 Workshop on Ontologies and Information Sharing*, Seattle, Washington, pp. 63-70, 2001.
- Noy, N and McGuinness, D., “Ontology Development 101: A Guide to Creating Your First Ontology”, available at: *journal.dajobe.org/journal/posts/2003/03/17/ ontology-development-101-a-guide-to-creating-your-first-ontology/.*
- Noy, N., Fergerson, R., Musen, M., “The knowledge model of Protégé-2000: Combining interoperability and flexibility”, *In: Dieng, R. Corby, O. (eds) 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW ’00*), Juan-Les-Pins, France (Lecture Notes in Artificial Intelligence LNAI 1937) Springer-Verlag, Berlin, Germany, pp. 17-32, 2000.
- Noy, N., Hafner, C., “The state of the art in ontology design”, *AI Magazine*, 18(3), pp. 53-74, 1997.
- Ogush, M., et al., “Template for Documenting Software and Firmware Architectures”, *available at http://www.architecture.external.hp.com.*
- ONTOPRISE n.d., How to work with OntoEdit, ONTOPRISE® Provider of Technology and Applications Enabling Semantic Solutions, Germany, *http://www.ontoprise.de/documents/tutorial_ontoedit.pdf* (Accessed: 15/10/05).
- OWL Documentation, *http://www.3WC.org*. (Accessed 5/05/08)
- OWL Web Ontology Language Overview, *http://www.w3.org/TR/owl-features/*
- Park, H., Yoo, J., and Cho, S., “A Context-Aware Music Recommendation System Using Fuzzy Bayesian Networks with Utility Theory”, *http://sclab.yonsei.ac.kr/publications/Papers/LNCS/FSKD2006 PHS.pdf*, 2006.

- Pashtan, A., Heusser, A., and Scheuermann, P., “Personal service areas for mobile Web applications”, *IEEE Internet Computing*, 8(6), pp. 34 – 39, 2004.
- Pazzani, M. and Billsus, D., “Learning and Revising User Profiles: The Identification of Interesting Web Sites”, *Machine Learning*, Vol. 27, pp. 313-331, 1997.
- Pazzani, M., “A framework for collaborative, content-based and demographic filtering”, *Artificial Intelligence Review*, 13, 5-6, pp. 393–408, 1999.
- Pazzani, M.J., Muramatsu, J. and Billsus, D., “Syskill & Webert: Identifying Interesting Web Sites”*, (AAAI)/(IAAI)*, Vol. 1, citeseer.ist.psu.edu/article/pazzani98syskill.html, 1996.
- Pellet DIG Reasoner, *http://pellet.owldl.com*
- Pennock, D. and Horvitz, E., “Collaborative Filtering by Personality Diagnosis: A Hybrid Memory And Model-Based Approach”, *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, San Francisco, 30 June-3 July, Morgan Kaufmann, San Francisco, CA, pp. 473-80, 2000.
- Perry, D., “Generic architecture descriptions for product lines”, *In Proc. Of ARES II: Software Architectures for Product Families (LNCS 1429)*, Springer-Verlag, pp. 51-56, 1998.
- Pohl, K., Bockle, G., van der Linden, “Software Product Line Engineering: Foundations, Principles and Techniques”, Springer-Verlag, 2005.
- Porter, B., “An exemplar-based learning apprentice”, *In Proceedings of the Fourth International Workshop on Machine Learning*, Irvine, Ca; Morgan Kaufmann, 12-23, 1987.
- Prankatius, V., Oberweis, A., and Stucky, W, “Product Lines for Digital Information Products”, *Information Systems*, 32(6), pp. 909-939, 2007
- Protégé: An Ontology Editor and Knowledge-Base Framework, *http://protege.stanford.edu/*
- Pühretmair, F., Rumetshofer, H., Schaumlechner, E., “Extended Decision Making in Tourism Information Systems”, *Lecture Notes In Computer Science*, *Proceedings of the Third International Conference on E-Commerce and Web Technologies*, Vol. 2455, pp. 57 – 66, 2002.
- Requirement Tools: *http://easyweb.easynet.co.uk/~iany/other/vendors.htm#Doors*

- Resnick, P. and Varian, H.R., “Recommender systems, Volume 40. ACM Press, *http://doi.acm.org/10.1145/245108.245121*, 1997.
- Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P. Riedl, J., “GroupLens: An Open Architecture for Collaborative Filtering of Netnews”, *Proceedings of ACM Conference on Computer Supported Cooperative Work*, citeseer.ist.psu.edu/resnick94grouplens.html, 1994.
- Riihiaho, S., “Experiences with Usability Evaluation Methods”, Thesis at the Helsinki University of Technology, Laboratory of Information Processing Science, 2000.
- Rosset, S., Neumann, E. Eick, U. Vatnik, N. and Idan, Y., “Customer Lifetime Value Modeling and Its Use for Customer Retention Planning,” *Proc. Eighth ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining*, pp. 332-340, 2002.
- Salton, G. and Buckley, C., “Term-Weighting Approaches in Automatic Text Retrieval”, *Information Processing and Management*, 25(8), pp. 513-523, 1988.
- Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. “Item-based collaborative filtering recommendation algorithms”, *In Proceedings of the Tenth International World Wide Web Conference,* Hong Kong, China, 2001.
- Sarwar, B., Karypis, G., Konstan, J., and Riedl, J, “Analysis of recommendation algorithms for ecommerce”, *In Proc. of ACM conference on Electronic Commerce*, pp. 158-167, 2000.
- Sarwar, B.M., Sparsity, Scalability, and Distribution in Recommender Systems, Ph.D. thesis, University of Minnesota, 2001.
- Satine project at *http://www.srdc.metu.edu.tr/webpage/projects/satine/*
- Schafer, J., Konstan, J. and Riedl, J., “E-Commerce Recommendation Applications”, *Data Mining and Knowledge Discovery*, 2001, citeseer.ist.psu.edu/schafer01ecommerce.html.
- Schein, A., Popescul, A., Ungar, L., and Pennock, D., “Methods and metrics for cold-start recommendations”, *In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* ACM Press, Tampere, Finland, pp. 253–260, 2002.
- Schmittlein, D.C., Morrison, D.G. and Colombo, R. “Counting Your Customers: Who Are They and What Will They Do Next ?”, *Management Science*, Vol. 33, No. 1, 1987.

- Schwanke, R. and Lutz, R.,"Experience with Architectural Design of a Modest Product Family", *Software Practice and Experience*, 34(13), 1273-1296, 2003.
- Shardanand, U. and Maes, P., "Social information filtering: Algorithms for automating "word of mouth". *In Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, ACM Press, Denver, CO, USA, pp. 210–217, 1995.
- Shaw, M., Garlan, D., "Software Architecture: Perspectives on an Emerging Discipline", Prentice-Hall, Upper Saddle River, NJ, USA, 1996.
- Soboroff, I and Nicholas, C. "Combining Content and Collaboration, in Text Filtering," *Proc. Int'l Joint Conf. Artificial Intelligence Workshop: Machine Learning for Information Filtering*, 1999. Citeseer.ist.psu.edu/259060.html
- Sollenborn, M. and Funk, P., "Category-based filtering and user stereotype cases to reduce the latency problem in recommender systems", *In Proceedings of the Sixth European Conference on Case-based Reasoning. LNCS*, Vol. 2416. Springer-Verlag, Aberdeen, UK, pp. 395–405, 2002.
- Staab, S., Schnurr H., Studer, R., Sure, Y., "Knowledge Processes and Ontologies, *IEEE Intelligent Systems*", 16(1), pp. 26-34, 2001.
- Staab, S., Werther, H., Ricci, F., Zipf, A., Gretzel, U., Fesenmaier, D.R., Paris, C., and Knoblock, C., "Intelligent systems for tourism", *IEEE Intelligent System*s, Volume 17, Issue 6, Nov/Dec, pp. 53-66, 2002.
- Statnikov, R. and Matusov, J., "Multicriteria Optimization and Engineering", Chapman & Hall, New York, 1995.
- Steinbauer, A., "Consumer Behaviour in e-Tourism", Doctoral Thesis, Department for Information Systems and e-tourism Ludwig-Franzens-Universität Innsbruck, 2005.
- Steiner, T, "Software Agents in Tourism Industry – Prototypes and Prospects", *Informatik.Informatique*, Vol. 1, pp. 33-38, 2002.
- Stumme, G., Maedche, A., "FCA-MERGE: Bottom-up Merging of Ontologies", *Bernhard Nebel (ed.) Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI, 2001)*, Seattle, Washington. Morgan Kaufmann Publishers, San Francisco, California, pp. 225-234, 2001.

- Su, X., Ilebrekke, L., "A Comparative Study of Ontology Languages and Tools", *In Proceeding of the 14th Conference on Advanced Information Systems Engineering (CAiSE'02)*, Toronto, Canada, 2002.
- Sure, Y., Erdmann, M., Angele J., Staab, R., Wenke, D., "OntoEdit: Collaborative Ontology Engineering for the Semantic Web", *Lecture Notes in Computer Science* (LNCS 2342), Springer-Verlag, Berlin, Germany, pp. 221-235, 2002.
- Svahnberg, M., Gurp, J., Bosch, J., "On the Notation of Variability in Software Product Lines", *Proceedings of the Working IEEE/IFIP Conference on Software Architecture,* pp. 45-55, 2001.
- Swartout, B. Ramesh, P., Knight, K, Russ, T., "Toward Distributed Use of Large-Scale Ontologies. *In: Farquhar, A. Gruninger, M, Gomez-Perez, A., Uschold, M, van der Vet, P (eds) AAAI'97 Springer Symposium on Ontological Engineering,* Stanford University, California, pp. 138-148, 1997.
- Szyperski C., Gruntz D., Murer S., "Component Software ─ Beyond Object-Oriented Programming", Second Edition, Addison-Wesley, 2002.
- Terveen, L. and Hill, W., "Beyond recommender systems: Helping people help each other", *In Human-Computer Interaction in the New Millennium*, J. Carroll, Ed. Addison-Wesley, Reading, MA, USA, 2001. citeseer.ist.psu.edu/417017.html
- The Eclipse Foundation, available at: *http://www.eclipse.org*. (Accessed: 13/03/2007).
- Thiel S., & Hein A., "Systematic Integration of Variability into Product Line Architecture Design", *Proceedings of the Second International Conference on Software Product Lines*, pp. 130-153, 2002.
- Thompson, C., Göker, M. and Langley. P., "A Personalized System for Conversational Recommendations", *Journal of Artificial Intelligence Research*, Vol. 21, pp. 393–428, 2004.
- Tichy, W.F., "Programming-in-the-Large: Past, Present and Future, *In Proceeding of 14th International Conference on Software Engineering (ICSE '92)*, NY, USA, ACM Press, pp. 362-367, 1992.
- Tiscover AG, *http://www.Tiscover.com*. (Accessed 6/06/07).
- UML, MDA Tools, *www.modelbased.net/mda_tools.html* (Accessed 15/05/2007)

- Uschold, M., King, M., “Toward a Methodology for Building Ontologies”, *In: Skuce, D. (eds.) IJCAI’95 Workshop on the Basic ontological Issues in Knowledge sharing*, Montreal, Canada, pp. 6.1-6.10, 1995.
- van Ommering, R. C., “Building product populations with software components”, *In Proceedings of the 22nd International Conference on Software Engineering, ICSE*, Orlando, Florida, USA, pp. 255–265, 2002.
- van Ommering, R. C., van der Linden, F., Kramer, J., and Magee, J.,”The Koala Component Model for Consumer Electronics Software”, *IEEE Computer*, 33(3): pp. 78–85, 2000.
- van Setten, M., Pokraev, S. and Koolwaaij, J., “Context-Aware Recommendations in the Mobile Tourist Application COMPASS”, *International Adaptive Hypermedia Conf., LNCS 3137*, pp. 235–244, 2004.
- Venturini, A. and Ricci, F., “Applying trip@dvice recommendation to visisteurope.com”, *In Brewka, G., Coradeschi, S., Perini, A., and Traverso, P., (eds.). Proceedings of the 17th European Conference on Artificial Intelligence* (pp. 607-611), Amsterdam, IOS Press, 2006
- Vozalis, E., & Margaritis, K., “Analysis of Recommender Systems’ Algorithms”. *In Proceedings of the 6th Hellenic-European Conference on Computer Mathematics and its Applications (HERCMA)*, Athens, Greece, pp. 732-745, 2003.
- W3C, http://www.w3.org/TR/owl-ref/
- Weiss D., Lai C., Tau R., “Software product-line engineering: a family-based software development process”, Addison-Wesley, 1999.
- Werthner, H. and Klein, S., “Information Technology and Tourism—A Challenging Relationship”, Springer-Verlag, New York, 1999.
- Whitehead Jr., E. J., Robbins, J. E., Medvidovic, N. and Taylor, R. N., “Software Architectures: Foundation of a Software Component Marketplace”, *In Proceedings of the First International Workshop on Architectures for Software Systems,* pp. 276—282, 1995.
- Wohltorf, J., Cissée, R. and Rieger, A., “BerlinTainment: An Agent-Based Context-Aware Entertainment Planning System”, *IEEE Communications*, 43(6), pp. 102-109, 2005.
- xADL 2.0 – A Highly-extensible Architecture Description Language for Software and Systems, *http://www.isr.uci.edu/projects/xarchuci/*

- Ye, H. and Liu, H., “Approach to modelling feature variability and dependencies in software product line”, *IEEE Proc-Software*, 152(3), pp. 101-109, 2005.
- Youngs, R., et al., “A Standard for Architecture Description”, *IBM Systems Journal*, 38(1), pp. 32-50, 1999.
- Zanker, M., Fuchs, M., Hopken, Tuta, M and Muller, N., “Evaluating Recommender Systems in Tourism – A Case Study from Austria”, *In P. O’ Connor, W. Hopken and U. Gretzel (eds.) Information and Communication Technologies in Tourism (pp. 24-34)*, Springer Wien New York, 2008.
- Ziegler, C. Towards, “Towards Decentralized Recommender Systems”, PhD Thesis, Albert-Ludwigs-Universitat, Freiburg, 2005.
- Zins, A., Bauernfeind, U., Missier, F., Mitsche, N., Ricci, F., Rumetshofer, H. & Schaumlechner, E., “Prototype testing for a destination recommender system: steps, procedures and implications”, *In Frew, A. J. (ed.): Information and Communication Technologies in Tourism (pp. 249-258)*, Vienna-New York, 2004.
- Zins, A., Bauernfeind, U., Missier, F., Venturini, A & Rumetshofer, H., “An Experimental Usability Test for Different Recommender Systems”, *In Frew, A. J. (ed.): Information and Communication Technologies in Tourism (pp 228-238)*, Springer, Vienna-New York, 2004.

# APPENDIX

**Table 1. Complexities in Function Point Components**

| Record Element Types | Data Element Types | Data Element Types | Data Element Types |
|---|---|---|---|
| | 1 - 19 | 20 - 50 | 51 - |
| <2 | Low | Low | Low |
| 2-5 | Low | Average | High |
| > 5 | Average | High | High |

Complexity of ILF and EIF

| File Types Referenced (FTRs) | Data Element Types | Data Element Types | Data Element Types |
|---|---|---|---|
| | 1 - 4 | 5 - 15 | 15 - |
| <2 | Low | Low | Low |
| 2 | Low | Average | High |
| > 2 | Average | High | High |

Complexity of EIs

| File Types Referenced (FTRs) | Data Element Types | Data Element Types | Data Element Types |
|---|---|---|---|
| | 1 - 5 | 6 - 19 | 20 - |
| <2 | Low | Low | Low |
| 2-3 | Low | Average | High |
| > 3 | Average | High | High |

Complexity of Eos

**Table 2. Unadjusted Function Points Table**

| | Low | Average | High |
|---|---|---|---|
| EI | X 3 | X 4 | X 6 |
| EO | X 4 | X 5 | X 7 |
| ILF | X 7 | X 10 | X 15 |
| EIF | X 5 | X 7 | X 10 |
| EQ | X 3 | X 4 | X 6 |

## List of Publications from the Thesis

### Publication of Chapters in Books / Refereed Conference Proceedings:

1. Daramola J.O, Adigun, M.O, and Olugbara, O.O. , "A Product-Line Architecture for Evolving Intelligent Component Services in Tourism Information Systems", In P. O' Connor, W. Hopken and U. Gretzel (eds.): Information and Communication Technologies in Tourism (pp. 117-145). Proceedings of the International Conference (ENTER '08) in Innsbruck, Austria, Springer Wien New York, 2008.
2. Daramola Olawande, Adigun Mathew and Ayo, Charles, "Building an Ontology-based Framework for Tourism Recommendation Services", In W. Hopken, M. Sigala, and U. Gretzel (eds.): Information and Communication Technologies in Tourism (pp. 135-147). Proceedings of the International Conference (ENTER '09) in Amsterdam, Netherlands, Springer Wien New York, 2009.
3. Daramola Olawande, Adigun Mathew, and Ayo, Charles, "A Hybridized Product Line Approach for Developing Recommendation-Intensive Tourism Information Systems", Proceedings of the International Conference on Enterprise Information Systems (ICEIS '09) Milan, Italy (May, 2009) - *Accepted.*

**Journal Publication**

1. Daramola J.O, Adigun M.O., Ayo, C.K. and Olugbara, O.O. (2009): Improving the Dependability of Destination Recommendations using Information on Social Aspects, TOURISMOS: An International Multidisciplinary Journal of Tourism *(Accepted; In Press; Vol. 5, No. 1)* – GREECE.