DEPARTMENT OF COMPUTER SCIENCE

CONTEXT-AWARE RECOMMENDATION-BASED EDUCATIONAL TOOL FOR ENHANCING THE HIGH LEVEL SOFTWARE MODELLING PROCESS WITH DESIGN PATTERNS

GEORGE A. SIELIS

A DISSERTATION

SUBMITTED TO THE UNIVERSITY OF CYPRUS

IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

NOVEMBER, 2016

VALIDATION PAGE

Doctoral Candidate: George A. Sielis

Doctoral Thesis Title: Context-aware Recommendation-based Educational Tool for enhancing the High Level Software Modelling Process with Design Patterns

The present Doctoral Dissertation was submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at the ***Department of Computer Science*** *and was approved on the 17th of November, 2016 by the members of the* ***Examination Committee****.*

Examination Committee:

Research Supervisor

Dr. George A. Papadopoulos, Professor, University of Cyprus

Research Co-supervisor

Dr. Aimilia Tzanavari, Associate Professor, University of Nicosia

Committee Member (Chairman)

Dr. George Pallis, Assistant Professor, University of Cyprus

Committee Member

Dr. George Samaras, Professor, University of Cyprus

Committee Member

Dr. Paris Avgeriou, Professor, University of Groningen

Committee Member

Dr. Spiridon Likothanassis, Professor, University of Patras

DECLARATION OF DOCTORAL CANDIDATE

The present doctoral dissertation was submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy of the University of Cyprus. It is a product of original work of my own, unless otherwise mentioned through references, notes, or any other statements.

George A. Sielis

CONTEXT-AWARE RECOMMENDATION-BASED EDUCATIONAL TOOL FOR ENHANCING THE HIGH LEVEL SOFTWARE MODELLING PROCESS WITH DESIGN PATTERNS

GEORGE A. SIELIS

University of Cyprus, 2016

ABSTRACT (IN GREEK)

Η παρούσα Διατριβή ασχολείται με την διερεύνηση της χρήσης Αλγορίθμων Υπολογισμού Συστάσεων, και συγκεκριμένα τη χρήση των συστημάτων δημιουργίας και προβολής συστάσεων που λαμβάνουν υπόψιν τις παραμέτρους περιεχομένου (Context Aware Recommendation Systems). Η διερεύνηση των συστημάτων αυτών διευρίνεται στην εξέταση της χρήσης Συστημάτων Υπολογισμού Συστάσεων σε Εργαλεία Δημιουργικής Ανάπτυξης και στο κατα πόσο επηρεάζουν την Δημιουργική διαδικασία. Μέσα από την εξέταση των πιο πρόσφατων ερευνητικών αποτελεσμάτων στις ερευνητικές περιοχές της Αναγνώρησης Περιεχομένου, Εργαλείων Δημιουργικής Ανάπτυξης και των Συστημάτων Υπολογισμού Συστάσεων, το πρώτο μέρος της Διατριβής, προσδοκεί να εντοπίσει και να περιγράψει την σχέση των πιο πάνω συστημάτων και μεθόδων με διαδικασίες που εφαρμόζονται στην Τεχνολογία Λογισμικού και συγκεκριμένα τον σχεδιασμό Υψηλού Επιπέδου Μοντέλων Λογισμικού. Με την εφαρμογή των ευρημάτων αυτών στη διδασκαλία της Τεχνολογιας Λογισμικού, θα διαφανεί η σημαντικότητα στη χρήση των Συστημάτων Συστάσεων και θα αποτελέσει σημαντικό εργαλείο για νέους Μηχανικούς Λογισμικού.

Το δεύτερο μέρος της Διατριβής επικεντρώνεται στο σχεδιασμό, την ανάπτυξη και την αξιολόγηση του πρωτότυπου λογισμικού που αναπτύχθηκε στα πλαίσια της Διατριβής αυτής και ονομάζεται ArchReco.Ένα εκπαιδευτικό εργαλείο που χρησιμοποιεί Συστάσεις παραγόμενες από παραμέτρους Περιεχομένου και συστήνει Σχεδιαστικά Πρότυπα για την υποστήριξη των χρηστών (φοιτητές ή επαγγελματίες Μηχανικούς) που θέλουν να βελτιώσουν τις σχεδιαστικές τους ικανότητες, και βοηθά να μάθουν τα υπάρχοντα Σχεδιαστικά Πρότυπα που εφαρμόζονται στην Τεχνολογία Λογισμικού. Το πρωτότυπο λογισμικό χρησιμοποιεί τεχνολογίες Σημασιολογικής ανάλυσης και αναπαράστασης καθώς και ανάλυση με βάση το περιεχόμενο για την παροχή «μη εξατομικευμένων» συστάσεων Σχεδιαστικών Προτύπων. Το λογισμικό προσβλέπει στην εύκολη πρόσβαση, και συνεπώς στην εκμάθηση των Σχεδιαστικών Προτύπων συνδυάζοντας την θεωρητική και πρακτική εφαρμογή των προτύπων σε διαγράμματα σχεδίασης. Παράλληλα, θέτει τις βάσεις για περαιτέρω ανάλυση και εφαρμογή πρόσθετων τύπων συστάσεων όπως οι συστάσεις για σύνθεση ομάδας ή οργάνωση εργασιών στον τομέα της Τεχνολογίας Λογισμικού. Η Διατριβή κλείνει με την αξιολόγηση του πρωτότυπου λογισμικού και την ανάλυση των αποτελεσμάτων, υπό το πρίσμα των ερευνητικών ερωτημάτων που έχουν τεθεί. Παρουσιάζει τα συμπεράσματα που προκύπτουν από την έρευνα αυτή, καθώς και τις προοπτικές της σε σχέση με άλλες ερευνητικές περιοχές.

ABSTRACT

The research described in this dissertation deals with the investigation of Recommendation Algorithms and in particular Context Aware Recommender Systems, in Creativity Support Tools and their influence on the creativity process. Through the analysis of the state of the art in Context Awareness, Creativity-Support Tools and Recommendation Systems, the first part of the dissertation aims to identify and describe the close connection to these with Software Engineering processes and more specifically the design of High Level Software Models. Applying the findings in the area of Software Engineering Education through the usage of Context-Aware Recommendations to support an Educational and Training tool for learning Design Patterns in High Level Software Models, proves the importance of such Recommendations in processes like these and potentially constitutes added value to modelling tools that target new Software Engineers.

The second part of the dissertation focuses on the design, development and evaluation of a software Prototype, named ArchReco, an educational tool that employs Context-aware Recommendations of Design Patterns, to support users (CS students or professionals) who want to improve their design skills when it comes to training on High Level Software models. The tool's underlying algorithms take advantage of Semantic Web technologies, and the usage of content-based analysis for the computation of non-personalized recommendations for Design Patterns. The recommendations' objective is to support users in functions such as find the most suitable Design Pattern to use according to the working context, as well as learn the objectives and usage of each Design Pattern. Moreover, it sets the basis for further analysis and implementation of additional types of Context Aware Recommendations, related to other fields of research, such as group composition or task scheduling in Software Engineering. The dissertation concludes with

the ArchReco prototype evaluation, and the results' analysis with respect to the defined research objectives. Moreover, it elaborates the conclusions of this research work and discusses the future research challenges.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor George A. Papadopoulos who provided his guidance and support throughout the progress of my dissertation.

I want to thank my co-advisor, Dr. Aimilia Tzanavari who supported and guided me from the beginning throughout the end of this dissertation. With her help, guidance and encouragement, Aimilia provided me with the tools to complete my dissertation.

I am grateful to my parents, Andreas and Xenia, and my sister Demetra for being always on my side, and always believe in me. Their love and support was always a strong motivation for me to complete my dissertation.

My friends who directly or indirectly helped to the completion of this thesis: Nearchos, Christos, Andrea and Nandia, who read the dissertation and provided feedback, comments, corrections and suggestions for the improvement of my thesis. My closest friends Antonis, Demetris, Marios, Makis, George, Yiannos, and Christos for their persistent support.

Finally, I want to thank my loving wife Polymnia for her continuous and tireless support, her endless faith and undoubted believe in me. My wonderful daughters Eva, Andrea and Danae who are lightening my life.

Dedicated to my loving wife Polymnia, my children Eva, Andrea and Danae

and also to

Constantinos Paraskevas and Socratis Kanaris.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

A creativity outcome is a sequence of thoughts and actions that lead to a novel adaptive production [97]. Plucker and Beghetto [131] define creativity as the interplay between ability and process by which an individual or a group produces an outcome or product that is both novel and useful as defined within some social context. Having assumed that creativity is an attribute that we all have, we reach the conclusion that it is necessary to find ways and means to assist in outsourcing this property. Creativity is a characteristic that can be cultivated, while being different for people in terms of the level of its development. Cultivating and expressing the creative ability can be achieved through help and guidance. One approach to achieve this is through the use of Creativity Support Tools (CSTs). Creativity Support Tools are software systems that can emulate a realistic creative process by offering users the ability to record thoughts and ideas with the use of ICT means. The utilization of the existing CSTs is bounded to the replacement of the conventional means of collecting ideas, such as the paper or the whiteboard, with corresponding virtual environments, but they also offer the users the possibility to collaborate with other people from a distance.

High Level Software Design is by definition a creative process. Software Engineering processes combine a set of competencies, knowledge background and creativity. Design and development of software tools is not always a straightforward procedure; especially during the last few decades when the Software Engineering processes and methodologies are rapidly changing. Fast internet and easy access to data retrieval forces Software Engineers to be up-to-date and well informed for new technological achievements, otherwise it is hard for them to be productive and, most importantly, innovative. This becomes more difficult for inexperienced Engineers who have recently acquired the degree in question. A common practice that new Engineers use is the web searching for finding new Software Engineering practices and methodologies. In most cases, this becomes time-consuming and confusing for them because of the plethora of information that exists, the uncertainty for the correctness, the lack of centralization and the unstructured form of the information in the web. This is what Recommendation Systems and in particular Context Aware Recommendation Systems are trying to solve.

This research aims to reinforce the importance of Creativity Support Tools (CSTs) and, more specifically, Software Engineering related CSTs with the addition of more advanced functionality. Generally, CSTs are the basis for strengthening the creative capacity and they can also be used for the cultivation of the creative ability. At this stage, these tools lack the mechanisms that can create the necessary stimulus to the user, which would make the user more productive in the process of creativity [157]. Such mechanisms are the Recommendation Systems for the support of the creativity process such as the recommendations of people to collaborate with, related problems or solutions, related ideas or related resources such as articles, images or videos. Recommendation systems are systems that belong to the information filtering systems family. In general, Recommendation systems seek to predict the rating or the preference that a user would give to an

item. The addition of "Context" as additional factor to be taken into account can become a useful enhancement for such systems.

1.1 Motivation

The motivation for this work is driven from the notion that Creativity and Innovation are keys to success for any business, or individuals who are able to learn and create innovative solutions. The rapid technological changes that can be seen during the last few years, as well as the new methodologies that are used in Software Engineering, make the continuous learning of these methodologies difficult, especially for new inexperienced Software Engineers.

The complexity of the Software Design process, in most cases is related to the incomplete requirements specifications or the lack of knowledge in specific design and programming methodologies, such as Design Patterns. In general, software designers can be classified into beginners, with little experience all the way through to the very experienced. This research work derives from the need of new Software Engineers and more specifically Computer Science or Engineering students, to be assisted in overcoming the feeling of uncertainty when designing software models using Design Patterns. Based on [113], "*a design pattern is a description of a recurring problem in an environment that includes a description of the solution to that problem, in such a way that the solution can be used many times over, without ever doing it the same way twice*". Therefore, it is believed that supporting Software Engineering students in learning the Design Patterns within a Creativity Supported environment can significantly enhance the Software Engineering design process.

Existing Software Design tools are lacking aiding mechanisms for the support of new designers in finding and applying Design Patterns to High Level Software models based on input

requirements written in physical language and taking into account the relevant context of a working problem. In the existing literature there are reported attempts to produce recommendations for Design Patterns such as [70],[65],[190] but none of the examined articles took the context into account, nor had they produced a complete prototype solution for specific target groups such as CS or SE students.

Shneiderman et al [153] identify the requirements that software tools must support for the enhancement of the creative process. Based on this study, software development and in particular development of user interfaces must pay attention to users' facilitation so that these can be productive and also more innovative. The enhancement of creative process can be achieved by offering users more effective searching of intellectual resources, improved collaboration among teams and more rapid discovery processes between the proposed requirements like the simplicity on design. Based on [117], the first is that creativity support tools must be "self-revealing" so what can be done is clear to users; the second is the need to enable users not only to compose artefacts, but to also think of what to compose as artefacts.

Based on the abovementioned statements, this thesis argues that Software Engineering Students can become more creative by using a creative environment, which can support them in learning and, at the same time, applying the Design Patterns. Due to the large amount of existing Design Patterns, the use of Recommendation mechanisms that can reveal and retrieve Design Patterns from more than one data source, for a given problem is higly important. This work examines the usage of Context Aware Recommendations with the usage of Semantic Web technologies for the development of a software prototype tool, which will follow the creativity principles in order to support the learning and training of Design Patterns and High Level Software Design to Software Designers.

1.2 Thesis Statement

This thesis contends that the existing Software Engineering Modeling tools do not support the training of Software Engineering or Computer Science students in learning the Design Patterns. They also lack creativity aiding mechanisms which would support users in learning and practicing the Design Patterns by taking the context of a given problem or a given requirement into account. The main contribution of this thesis is twofold: First, the definition of the necessary contextual models that can be used for the development of the Context Aware Recommendation system taking into consideration the Creativity and Software Engineering principles. Second, the development of a software prototype tool which will apply the developed Context Aware Recommendation system for assisting the students in learning and at the same time practicing the High Level Software design with the use of Design Patterns.

1.3 Approach

The work presented in this thesis has both theoretical and practical aspects. The theoretical aspect consists of the comprehensive examination of the topics that constitute the research statement which are the Context Awareness, Recommendation Systems and Creativity. The practical aspect consists of the development of the design and implementation of the Creativity Conceptual model, the Software Engineering conceptual model, their usage for the development of a Context Aware Recommendation Algorithm and finally the development of the software prototype that will apply the designed Recommendation System for Design Patterns for learning and practicing the High Level Software Design.

In particular, the research goals and the methods that will be used are listed in the rest of this section.

1.3.1 Research Goals

Based on the thesis statement that was set, the following research goals and objectives have been determined:

- Define the contextual elements of the creativity process and use the defined elements for the design of the Creativity conceptual model
- Confirmation of the Software Engineering needs regarding the Creativity and the Context Aware recommendations as aiding mechanisms from Software Engineering professionals
- Define the conceptual model for Software Engineering processes as an extension of the Creativity conceptual model
- Design and implement the Context Aware Recommendation engine
- Develop the Software Design training prototype with the support of Context Aware Recommendations of Design patterns
- Design the evaluation plan that will examine the results concerning:
  - The usability of the prototype
  - The usefulness of the prototype
  - The educational character of the prototype
  - The accuracy and validity of the Context Aware Recommendation System
  - User satisfaction and experience by using the existing recommendation algorithms and the proposed recommendation algorithm.
  - Impact of the prototype and the supported recommendations in regards to the user's creativity

Figure 1: Research Roadmap

In Figure 1 we depict the road-map followed for the completion of this thesis.

1.4 Definitions

In this section some of the most used terms that are used in this thesis are defined:

- High Level Software Design - Briand et al. [25] define High Level Software Design as "*a collection of module and subroutine interfaces related to each other by means of USES and IS COMPONENT OF relationships. Precise and formalized information on module or subroutine bodies is not yet available at the stage of High Level Design*".

- Context - Dey et. al [46], define context as "*any information that can be used to characterize the situation of an entity. An entity is any information that is considered relevant to the interaction between a user and an application including the user and applications themselves*"

- Creativity - A creativity outcome is a sequence of thoughts and actions that lead to a novel adaptive production [97]. Plucker [131] define creativity as the interplay between ability and process by which an individual or a group produces an outcome or product that is both novel and useful as defined within some social context.

- Creativity Support Tools - Tools that are considered as means for the enhancement of creativity beyond the classic creativity producing methods. Creativity Support Tools are software tools that can provide guidance and facilitate the creativity process for users, by monitoring the process and the produced results [144].

- Recommendation Systems - Recommender or Recommendation Systems (RS) are software tools in applications or websites that suggest information (e.g. items, people, news articles) that might be of interest to the end user, taking into account various types of knowledge and data, such as the user's preferences, actions, tasks and contextual information.

- Recommendation system for software engineering (RSSE) - Is a software application that provides information items estimated to be valuable for a software engineering task in a given context [61].

- Design Patterns - Alexander [5] defines Design Patterns as a well-known and frequently used software engineering problem-solving discipline, which has emerged from the object-oriented community.

- Semantic web - The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries [191].

1.5 Declaration and Credits

The work presented in this thesis is, to the best of my belief, original and has been written by the author, except as acknowledged below. Also, I declare that the material presented in this thesis has not been previously submitted for a degree at this or any other university.

1.5.1 Additional publications relevant to the thesis

A large part of this thesis consists of work that was authored or co-authored by myself and already published as well while some parts of this work are submitted and are reviewed for publication.

The work presented in chapter 3 was originally presented in [156] which was a joint work with Morpheus (Netherlands), University of Hildescheim (Germany), University of Aalbourg (Denmark) and University of Piraeus (Greece). The part that is presented in this thesis are the result of my personal work.

The work presented in chapter 5 contains input from the work presented in [157], while chapter 4 contains my personal work that was published in [160]. Part of the work presented in chapter 4 was authored by myself and co-authored by Christos Mettouris, Roger M. G. Dols and Quintin Siebers [155]. The parts presented in the chapter are the results of my personal work.

The work referred to the survey in Software Engineering Design Processes and Tools is work that was performed by myself and it is registered as a technical report [159]. The use of the creativity conceptual model and its extention by the Design Patterns model consist of personal work that was presented in [161]. The prototype that was developed for the aims of this work consist of personal work and it is presented in [158] while additional work regarding the Context Aware Recommendation for Design Patterns and the evaluation results from the user based evaluation related to the developed prototype of this work are submitted in ACM SAC International conference, and Journal for Software Engineering and Development respectively.

The papers I have authored or co-authored during the course of my doctoral dissertation were supervised by my research advisors, George A. Papadopoulos and Aimilia Tzanavari.

1.6 Structure of the Thesis

The rest of this thesis is organized as follows.

Chapter 2 presents the work related to this thesis. This chapter presents the most relevant research findings from the existing literature and concludes with the discussion of how this thesis goes beyond the current state-of-the-art.

Chapter 3 presents the literature with respect to context awareness, the methods and the tools for representing the context.

Chapter 4 presents the Recommendation Systems, the types of recommendation systems the existing tools and their applications.

Chapter 5 surveys the literature in regards to the creativity models and the several creativity models as they were defined through the years, and it also presents the examination of some of the most known creativity support tools examining whether they support context aware recommendations. Additionally, it presents the work that has been done for modeling the creativity and presents the work that was done for the development of a generic Context Aware Recommendation mechanism for creativity.

Chapter 6 presents the user based survey that was performed for the identification of the Software Engineering needs in regards to recommendations support and the ease of process creativity with the use of existing Software Modelling tools and processes.

Chapter 7 uses the creativity conceptual model that is presented in chapter 5 and the findings in chapter 6 to extend the conceptual model of creativity by the addition of the Software Engineering process concepts and define the model that will be used for the development of the Context Aware Recommendation tool for the recommendation of Design Patterns.

Chapter 8 presents the Context Aware Recommendation for Design Patterns the post and pre filtering techniques that were used in combination to the Semantic Web technologies.

Chapter 9 presents the ArchReco prototype as a complete solution including the Context Aware recommendation tool from chapter 8.

Chapter 10 presents the ArchReco user based evaluation setup and execution and depicts the evaluation results.

Finally, this thesis concludes with chapter 11. This chapter resumes results of this thesis, and it lists a number of directions for future work.

Chapter 2

Related Work

2.1 Introduction

This chapter examines the work that is related to the research topics that the current thesis deals with. The research presented in the current thesis involves three research topics, Context Awareness, Recommendation Systems and Creativity, while the domain of application of the combination of the three is Software Engineering. Therefore, the current chapter presents the related work found in the existing literature related to combinations of the aforementioned topics.

2.2 Context Awareness and Creativity

Context awareness in Computer Science could be defined as the recognition of the user's environment parameters, which will subsequently act as impulse that activates the corresponding function. Schilit et al. [144] attempted to define context by specifying three categories: computing context, user context and physical context.

Chen et al. [33] extended this definition by adding the time context element. The definitions given by [33] and [144] have a common limitation: they do not specifically state the boundaries for

considering information as context or not [74]. Context aware recommender systems concerned researchers extensively. Adomavicius [3] identifies two approaches for the recommendation systems content-based and collaborative recommendations. Park [125] proposes a context aware recommendation system which uses Bayesian networks and the Utility theory for the recommendation of appropriate music with respect to the context. Elahi [53] presents a recommender system which collects information from users' web searches for the improvement of content on visited pages. Very little work relating recommender systems to Creativity can be found in the literature, but several resources about Creativity and the modelling of Creativity process can be found, such as [168],[149],[8] and [152].

The second constructive part of this section is creativity which can be used in many disciplines. Many scientists have attempted to allocate meaning to the creative process and its potential outcomes. Plucker [131] defines creativity as the interplay between ability and process by which an individual or a group produces an outcome or product that is both novel and useful as defined within some social context. Schneiderman et al. [153] support that creativity is the development of a novel product that has some value to the individual and to a social group. Cougar [39] perceives creativity at three levels: as a discovery method through the idea generation, as an invention with the development of ideas, and as innovation with the transformation of ideas into services [87]. Atman [8] and Schneiderman [152] conceptualize creativity as a sequence of steps with variants. Simulating the creativity steps, along with applying creativity techniques, produces the software tools known as Creativity Support Tools (CST). In [131] creativity process perceived as a two stages process: "preparation" and "ideation". This two-stage process highlights the steps of creativity defined in [152] in a more concrete way. This grouping of the steps into two stages facilitates the specification of each step's context attributes and therefore their grouping in "primary"

and "secondary" entities, following the transformation proposed by [169]. This transformation has actual value for the design of context awareness ontology.

Tsatsou et al. [179] propose a hybrid recommendation system which combines ontological knowledge with content-extracted linguistic information. In a more domain specific application, [151] and [163] propose a recommender system in a learning platform which aims to the facilitation in finding resources and learning material. In the same way, an e-commerce recommender system aims to stimulate the curiosity of the user to view products that belong in the area of his interests. Research work presented by [176], presents seven advantages of the usage of recommender systems. The usage of a recommender system for the enhancement of creative process highlights two of the advantages identified by [176]: Effectiveness and efficiency. The work presented in [157] proved the lack of context awareness and particularly the absence of recommender systems, from the most known creativity support tools. Regarding knowledge dissemination and distribution at the workplace, many studies have shown that interpersonal help seeking is the most important strategy people employ so as to acquire knowledge at their workplaces. Beham et al. [17] presented the APOSDLE People Recommender Service, a service based on an underlying domain model and on the APOSDLE User Model to support interpersonal help seeking at the workplace. Similarly, Sie et al. [154] recommend knowledgeable persons for Creativity and networked innovation based on user profiles, position in the organization, power relationships and creativity. It is a utility-based recommendation approach.

2.3 Context Aware Recommendation Systems

Context Aware Recommendation systems are proven that they produce good quality recommendations, especially due to the personalization they provide. Taking into account contextual

elements that are associated with the user model of a particular domain offers a high level of personalization. Abbas et al. [1] surveyed the usage of Context Aware Recommendation systems that are using Computational Intelligence techniques such as Fuzzy sets, Artificial Neural Networks, Evolutionary programming, Swarm Intelligence and Artificial Immune Systems.

The key in Context Aware Recommendation Systems is the identification of the information that can be used as context, as well as to weigh how significant each contextual factor is for the recommendations. The dramatic growth of Social networks created a new perspective in personalized information as contextual information, since personalization does not only depend on a user's ratings but also on the similarities between the shared information within a network. The shared information is usually related to actions, interests, social behaviors or knowledge background that can be collected from the social network activities such as the "likes", "share", "join", "follow" etc. By using the collection of such information, the user interests are determined and an algorithmic framework for retrieving semantic data based on user interests from multiple sources is provided in [195]. The proposed framework retrieves the interests of a user from several social networks that the user belongs to, and acts as a recommender system that recommends interests based on what a user is doing at the time in the system. This way, user interests which may vary in several social networks are integrated and ranked. Berkovski et al. [19] support that the use of social network activities and the user interaction social networking can lead to more accurate personalized news feeds, but can also create similarity models that can personalize the users within cooperative mobile environments that incorporate different contextual conditions and individual user characteristics. Lane et al. [92] use the latter for the creation of similarity graphs extracted from the users' social networking activities in combination with the training data searching and classification models between individuals, in order to produce personalized classification models.

2.4 Design Patterns in Software Engineering

In the last decades Software Engineering became a hot topic for research and the research results taken from related works were directly applied by the industry. The rapid changes in technology and the way that Software Engineering principles and techniques are influenced by these changes, the specification on which tools should be used or not, as well as a complete analysis of Software Engineering life-cycle principles such as design, specification, verification, production and management are presented in [63]. Similar research such as [26] and [150] examines the Software Engineering methods techniques and applied tools for Software Engineering. One major challenge that concerned Software Engineering researchers but also professional engineers, was the usage of Design patterns in Software Design, especially the reverse engineering for the recovery of design patterns. Examples of such techniques can be found in [69],[178],[98],[50]. The recovery of Design Patterns was mainly focused on the identification, with high accuracy, and the usage of Design Patterns that were applied in existing software tools. Most importantly, research in Software Engineering focused on the Software Engineering life-cycle models like in [21], [115], [142] and [16]. The complexity of tasks that Software Engineering process, has led to the need of assisting tools and aiding functionality in the Software Engineering tools. Robilalard et.al in [137] give an overview on Recommender Systems for Software Engineering describing what are they, what can they do, and how are they used in the process of Software Engineering.

Design Patterns is a well known and frequently used subject in Software Engineerinng research. In the following paragraphs we elaborate how Design Patterns are used in the several disciplines of the Software Engineering research.

Design Patterns in Software Architecture Design - In the literature one can find two types of patterns related to Software Engineering, the Architectural Patterns and Design Patterns. Design Patterns were introduced by Alexander [5] as Software Engineering problems that may occur repeatedly, and they are associated with a solution that can be used to solve a problem every time it occurs within a current context that the problem exists. Architectural patterns are similar but with a wider scope. For example, more than one design patterns can be applied for specific Architectural patterns. Most of the publications related to Architectural Patterns focus in Architectural Styles and Views. Design patterns in terms of Software Architecture Design are generally met in papers related to the Architectural Decisions [198]. In a more general perspective, Architecture design [55] presents design patterns as part of the 4 views, which the Architecture Knowledge Management (AKM) is consisted. Zimmerman et al. [197] elaborates the combination of pattern and decision-centric design in Software Architecture Design while [72] presents methods for documenting decisions with patterns.

Appart from the fact that Design Patterns are part of Software Architecture Knowledge Management and considered a very important Architectural Decision point, this work is not focusing on defining the context of a design pattern through the very complex Architectural Decision Making process analysis, as this is presented in [198]. The context aware recommendation mechanisms of the current work are following the assumption that the Architectural decisions already exist and the Software Engineer is able to proceed to the High Level Modelling design and therefore be used by the developers for coding the designed components.

Design Patterns Recovery - After the Design Patterns usage in Software Engineering matured enough several frameworks and software platforms were developed with the use of Design Patterns. As mentioned in [134] the flexibility in software maintenance and re-usability motivated several researchers to develop Design Patterns recovery techniques. Examples of such techniques

can be found in [69],[178],[98],[50]. The recovery of Design Patterns mainly aims to identify with high accuracy the Design Patterns that were used in existing software tools. The current work aims to achieve the opposite result, which is to recommend the most suitable patterns to use for a Software tool. The recovery techniques, however, are useful for the current work especially for the evaluation of the current work's proposed tool. Scenarios from existing frameworks implemented following specific Design Patterns can be used by the proposed prototype of this thesis, as a proof for its accuracy and usefulness. In addition, Design Patterns recovery techniques can be used for the creation of datasets of Design Patterns for which the lack of such datasets consists of a known problem at the current stage of this work.

Design Patterns & Recommendation Systems - In the latest years, Design Patterns for Software Architecture design are increasing in number with new patterns appearing to cover general functionalities or more dedicated domains (e.g., mobile application design, or user interface design). Some previous research has focused on providing recommendations on the appropriate usage of Design Patterns. Gueheneuc et.al [70] proposed a methodology of recommending design patterns through the textual analysis of each pattern into the most important words and computing the similarity distance between those words and the words of the query given by the user. Gomes et al. [65] proposed a Case Based Reasoning (CBR) Recommendation system for the recommendation of Design Patterns based on previous experiences using a Design Patterns Knowledge Base and related taxonomies. A similar system developed by [190] that recommends patterns using the Implicit Culture Framework (ICF). The recommendations are produced based on the users' previous actions, based on conventional Information Retrieval and CBR methods. Palma et al. in [124] propose a Design Patterns Recommendation (DPR) framework, which recommends patterns based on predefined questions that the designers have to answer, and based on the given answer the framework has a weighting mechanism for the selection of the appropriate pattern. The initial

identification of patterns that can be used through the DPR framework [124] are selected through LUCENE indexing and Term Frequency - Inverse Document Frequency (TF-IDF) filtering of a given query and the intent description of each pattern.

Formalization and Reasoning of Design Patterns - Formalization of Design Patterns refers to the techniques and methodologies that were developed for the representation of Design patterns. In that aspect, in the existing literature several approaches can be found. In particular, there are research papers that refer to the ontological representation, graphical representation, UML representations of Design Patterns or to Design Patterns Specification Languages that are used for their representation. Bottoni et al. [24] present a visual and formal approach to the specification of patterns, supporting pattern analysis and pattern-based model completion. The approach is based on graphs, morphisms and operations from category theory and exploits triple graphs to annotate model elements with pattern roles. Dong et al. [49] provide a method of formalizing the representation of Design patterns with the use of extended UML language by adding UML annotations aiming to represent the roles that an operation/attribute plays in addition to the roles a class plays in a Design Pattern.

Software Engineering Educational and Training Tools - SimSE [118] is a computer-based environment that facilitates the creation and simulation of realistic game-based software simulation models. SimSE is used as an educational software environment providing the students with a platform through which they can interact with many different aspects of the software process in a practical manner. SimSE is a single-player game in which the user/player has the role of the project manager who must manage a team of developers for the completion of tasks for a software engineering project.

In Mancoridis et. al [103] work, a combination of two Software Engineering Educational tools are presented: The Object Oriented Turing (OOT) and Star. The software tool presented in

this work is targeted for Unix Users and specifically learners who are designing software using the Turing programming language. More specifically, it is a programming environment enhanced by a set of tools tools for editing, high-speed compiling, linking, executing, and debugging OOT programs, as well as browsing the Unix file system.

Another Game Based Educational tool for Software Engineering Training is presented in [184]. In the work presented in [184], the importance of games in education is emphasized and the education of the complex course of Software Engineering through a Game Development Framework (GDF) is being targeted. The difference between the GDF and other game-based educational tools is the fact that the framework is used to create games. This means that the students are learning the Software Engineering concepts through the game development by writing their own games for particular requirements that they are given.

2.5 Beyont the current State-of-the-Art

Through the examination of articles which are most relevant to this thesis, it becomes obvious that we can find related work to the partial topics that the current work is involved but we cannot find any work that combines all topics together. For example, the combination of Creativity with Context Aware recommendations in the topic of Software Engineering was not met in any of the examined articles. Although there is no direct literature work on the thesis statement, the existing literature has several research results regarding the individual research topics that the current thesis deals with.

With the examined research work it is confirmed that there is no previous work that presents the Contextual modelling of Creativity used for the production of Context Aware Recommendations, nor a combination of the Creativity Contextual model in relation to Software Engineering. In the existing literature, attempts for producing Recommendations of Design Patterns have been made

in the past [70], [65], [190], but they differ with this work as for the type of Recommendations, since the context is taken into account; the range of the recommended Design Patterns, since they are focusing on the GoF Design Patterns [59] only; and the lack of a complete solution with specific purpose of use, that targets Software Engineering and Computer Science students.

The originality of the current thesis is met by: 1. the design and definition of the Creativity Contextual model; 2. the design and definition of the Software Engineering contextual model as an extension of the creativity process; 3. the design and implementation of the Context Aware recommendations of Design Patterns; 4. the use of Semantic Web technologies for the retrieval of Design Patterns from multiple data sources and filter them based on the context of a working problem, and finally 5. design and development of a software solution that integrates the models and the algorithms of this thesis, aiming to contribute to the Software Engineering education and training topic.

In the next three chapters more comprehensive analysis for the three research areas, Context Awareness, Creativity and Recommendation Systems are presented, where each individual topic is analyzed regarding their supported tools, models and methodologies.

Chapter 3

Context Awareness

3.1 Introduction

The last few years, Context Awareness has been an active area of research. The importance of context awareness focuses on the automated services that can be offered from computing systems to users. In the literature, the word "context" is defined as the set of facts or circumstances surrounding a situation or event. Research in context aware systems approached in several disciplines and tested through the adjustment of its several research findings in several areas of application such as learning, mobile computing and ubiquitous computing. It is at least intriguing to notice that very few published articles relate context awareness with creativity. The design and implementation of context aware models usually aim to support and facilitate computing processes either by using adaptation methods or by providing recommendations to the end user, based on contextual parameters. In research areas such as e-learning, social networking and creativity the definition of contextual parameters are usually related to behavioral, social or psychological theories. The extensive use of e-learning systems and the evolution of the social network applications, during the last years, led to the study and implementation of several context aware models for learning and social networking that are currently used in such systems. Although context awareness was

studied extensively for learning and social networking, very little work is acknowledged regarding the use of context awareness in creativity and more specifically the use of context awareness in Creativity Support Tools. This verification became a motivation to us, for a further study of the Context Awareness, Creativity and Creativity Support Tools topics, in order to track the open issues and the challenges in combining the topics. In addition to that, this chapter examines whether contextual elements for creativity could be defined and used for modelling the context of Creativity. The fulfillment of this task will lead to future studies, answering research questions and hypothesis such as *"What is the impact of CA in collaborative creativity process","Is CA enhancing creativity?"*, *"Can we create adaptive collaborative creativity support interfaces and what contextual parameters do we have to take into account?"*. Therefore, it is important to define how Context Awareness and Creativity are perceived.

Context awareness in computer science could be defined as the recognition of the user's environment parameters, which will subsequently become the impulse for the activation of the corresponding functioning. Schilit, et al. [144] attempted to define context by specifying three categories: *computing context, user context and physical context*. Chen et al. [33] extended this definition by adding the *time context* element. The definitions given by [144] and [33] have a common limitation: they do not specifically state the boundaries for considering information as context or not [74]. A first solution to this problem was given in [46] by limiting the context to all information that is relevant to the interaction between user and application: "*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application including the user and applications themselves*". The problem with this definition was that it was not specific enough as to what can be considered as context, thus every application could be in a sense described as context-aware. Therefore, the definition of context required an upper

and a lower bound to only include information that is strictly relevant and necessary to perform context-awareness. This definition is given by [74]:

> "*Context characterizes the actual situation in which the application is used. The situation is determined by information which distinguishes the actual usage from others, in particular characteristics of the user (user's location task at hand, etc.) and interfering physical or virtual objects (noise level, nearby resources etc.). Thereby, we only refer to information as context that can actually be processed by an application (relevant information), but that is not mandatory for its normal functionality (auxiliary information).*"

This chapter aims to examine the area of context awareness in regards to the technologies, the methods and the several frameworks that they were developed. More specifically, the rest of this chapter refers to the examination of several context technologies, methods and techniques that can be found in the literature, with the perspective of the selection of the most suitable ones for their future application to the creativity context information management and creativity support implementations.

3.2 An overview in Context Awareness

From the definition of context and the categorization of context given in the introduction it is concluded that a context awareness application has two ways of using context. The first is with the automatic adaptation of context according to the discovered context, and the second is the dynamic "on the fly" updating of context in the user's profile for future use [32]. Most of the context awareness applications are using the human computer interaction and the interaction of the environmental conditions that the user is surrounded, to collect information and create or retrieve the corresponding context. Through the years several context awareness applications have been developed and in general, the context awareness systems can be implemented in several ways. The implementation approach depends on special requirements and conditions that characterize the

overall system on which the context aware system will be applied for. The use of context awareness systems is located in their ability to adapt a computer system to both the surrounding physical environment and the virtual environment of computing platforms and computing networks [189]. This is the approach of context awareness systems in *Ubiquitous computing*. One other approach is the Autonomic Computing [88] approach which refers to self-management systems that require only high-level human guidance, and they have the ability to manage and protect their own resources. The last, and more related to the functional architecture approach, is the traditional one which focuses on how, when and where humans and computers interact [109].

The notion of how, when and where humans and computers interact leads to the definition of the contextual elements which constitute the basic elements for the design and implementation of any kind of context aware application. The context elements are the necessary actions, events, or signals that a context aware application needs to start functioning. For example, [45] separates the context of learning in three kinds: *Digital context, Device context and Learner Information Context*, but each context type can be correlated to different context elements in order to function in a proper way.

The interaction between a context aware application and the physical environment must be expressed by a bridging model that describes the real world entities and their interactions [73]. Usual input/output methods, such as the keyboard and the mouse, are considered as simple forms of contextual elements. More complex context aware applications use hardware or software implementations as contextual elements, depending on the kind of context they represent. An example of two different approaches of context aware applications that use hardware and software contextual elements are presented in [73] and [146] respectively. For example, the platform presented in [73], uses sensors and telemetry software to collect environmental data. It contains five components, a location system, a data model for the real world entities description, a persistent distributed system

for the data model representation, resource monitoring for the communication between network equipment and information status, and also a spatial monitoring service which enables event based location-aware applications. The context elements for the contained components are implemented with sensors which are based on radio-based techniques (e.g. GPS), electromagnetic methods (e.g. interference from monitors and metal structures, imaging detectors). The elements implementations are described through three resource monitoring classes: *Machine activity (keyboard activity), Machine resources (CPU usage, memory usage), Network point-to-point bandwidth and latency*.

In yet another approach, [46] proposed four different categories, focusing on the user perspective: identity, location, status, and time. Schmidt [146] describes an e-learning ontology-based platform which accomplishes the context acquisition by defining *workflow systems, human resources systems, web browsers* and *office applications* as contextual elements. Wang et al. [186] refined the definition of context for the area of education, by specifying six dimensions: identity, spatio-temporal, facility, activity, learner and community.

The context elements can be characterized as the main context entities, which they are parted by several other entities. The contextual elements can be viewed as "Concepts" for a context aware approach and their containments the "Entities" that construct a detailed contextual model.

3.3 Context Representation

Based on the context definition, context is any information that determines and characterizes the situation of an entity. An entity can be a user/person, a place, an object or a virtual object like noise level, or a resource location. Context entities can be structured into three domains: the user domain, the computer domain and the environment domain [172]. In a context-aware application the three domains are interacting between each other and in particular, the context data that can

be collected as an ensemble of the context data for an entity. The disaggregation of a context domain into entities aims at the facilitation of setting rules for the representation of the context. In this case it is quite challenging to find a way to separate the context and organize it in a proper way. This way, the desired representations of all the kinds of context are able to interact with an entity. By the division of entities in domains and since the context data can be retrieved from various entities [172] the context information can be described in internal and external features that represent the context. The internal features are used to describe characteristics that exist inside the entity or its domain. The external features are those which describe the context information that can be retrieved. The features of an entity may have a significant type, e.g. string, real, vector, etc. The value of an entity's feature can also represent one other entity. Usually mark-up languages with standard notations are used to represent the context to the clients, through the usage of categorization and classification of the contextual information. For example, e-sticky [27] is a context aware application that uses the Standard Generic Mark-up Language (SGML). In the latter work, it is mentioned that it is necessary to make representation of contextual information as easy as the development of a web page in HTML. It also emphasizes the importance in taking advantage of the syntax used within mark-up languages as well as the usage of DTD (Document Type Definition), that defines the tags and the way they "fit" together. The DTD is an easy method that allows the author to define new tags, or augment existing ones [27] and can be used for the representation of particular contextual fields.

In the rest of this section, the types of human knowledge are presented. Additionally, some of the well-known knowledge representation languages, also used as contextual representation models, are defined.

3.3.1 Types of Human Knowledge

Knowledge is the understanding of a subject area [51]. It includes concepts and facts about a subject area, relations between them and mechanisms of how to combine them to solve problems in specific areas. The types of human knowledge became an object of research in Cognitive psychology. Based on the acknowledged identifications of the human types, while several AI systems implemented many of them.

Procedural knowledge is the knowledge of how to accomplish a task or solve a problem. Typical types of procedural knowledge are the rules, problem-solving strategies, agendas and procedures. An example of Procedural knowledge is a Java programmer. An expert may have knowledge of a specific algorithm and how this can be implemented in multiple languages. However, a Java programmer knows how to implement the algorithm using the Java programming language only. Procedural knowledge is commonly referred as "know-how".

Declarative knowledge describes what is known about a topic or about a problem. This description is expressed in declarative sentences or indicative propositions. Declarative knowledge describes concepts and objects using statements. These statements may express facts or specific attributes like true or false.

Meta-knowledge is the knowledge of pre-existed knowledge. It is used to decide what other knowledge is best suited to solve an already solved problem, or what other knowledge is relevant or irrelevant to the problem.

Heuristic knowledge is the set of rules that guide the problem solving process on the basis of previous experience in solving problems, individual intuition and skills, and a good understanding of the problem.

Structural knowledge describes mental models and the organization of problems, solutions and their respective spaces. In other words, it contains the relationships between different pieces of knowledge from other categories.

Inexact and uncertain knowledge characterizes problems, topics and situations in which information is imprecise, unavailable, incomplete, random or ambiguous. It is often described in terms of a priori, a posteriori and conditional probabilities of events.

Commonsense knowledge is the collection of information and facts that an ordinary person is expected to know. McCarthy [108] defines commonsense knowledge as the term which is used to denote a vast amount of human knowledge about the world which cannot be put easily in the form of precise theories. Humans usually rely on this kind of knowledge when they face an incomplete characterization of a problem they are trying to solve or they lack more appropriate knowledge.

Ontological knowledge is an essential supplement to knowledge about a specific domain, describing the categories of things of a domain and the terms that people use to describe them [166]. The types of ontological knowledge overlap with the types of the other categories of knowledge, like i.e. declarative and structural knowledge.

3.3.2 Knowledge Representation languages and models

***Key-Values Languages*:** Key-value coding is a mechanism for accessing objects' properties indirectly. They use strings to identify properties, rather than through invocation of an accessor method or through direct access using instance variables. Key-Value models are the simplest data structure for context modelling and they are frequently used in various service frameworks. Usually, key-value pairs are used to describe the capabilities of a service. Schilit et al. [144] used the key- value representation for modelling context.

An example of key-values implementation is given for the Redis (http://redis.io/) project which supports lists as values. A key-value model stores data to caching systems, where they are represented in form of a map between keys and values. For the better understanding, a coding example using key-values values (listing 3.1) demonstrates the method of modeling the social news in Reddit (http://www.reddit.com/) project. The key-value store contains keys representing objects with a unique reference (identifiers) and these objects represent news from social news sites.

Listing 3.1: Example of Key-values supporting lists.

```
id = incr NextId => 1
set news_url_<id> ''http://foobar.org''
set news_title_<id> ''My foobar story''
push myList 1


id = incr NextId => 2
set news_url_<id> ''http://antirez.net''
set news_title_<id> ''The blog you reading now''
push myList 2
```

Key-values models are useful for the implementation of dimensional models, and their usability is located due to their ability to insulate dimensions from changes to source systems and enabling historical versions of dimension members. Key-values have poor query performance especially for dimensional queries in RDBMS (Relational Data Base Management Systems). Key-Values Modelling is simple, but not very efficient for more sophisticated data structuring purposes. They require the exact matching in order to support retrieval context algorithms and they do not support inheritance.

The key-values strengths are located in their simplified high performance joins, their reduced I/O operations, and the RDBMS optimizations. They can be used to simplify high performance joins by using a key value to simplify the join between a fact and a dimension table. The use of natural language keys, increases the width of a fact table, so a data page which is used by an RDBMS to store indexes, stores fewer fact table rows. This increases the I/O operations. Key-values help the enabling of optimizing mechanisms of an RDBMS and specifically for dimensional RDBMS models, e.g. bitmap indexing.

Key-values modeling methods can become useful for models that need high implementation performance. Their optimization mechanisms are good, but at the same time very complex. In context aware applications the performance issues are considered as secondary. Context aware systems, usually put in priority the management of data and knowledge information so that a system can identify its environment parameters with accurate computations.

***Markup Scheme Modeling Languages*:** All mark-up based models use a hierarchical data structure consisting of mark-up tags with attributes and content. The content of the mark-up tags is defined within other mark-up tags. The mark-up schemes are usually used to collect information for profiles. The context information profile building is usually using the Standard Mark-up Language (SGML), the super class of all the mark-up languages like XML. Multiple examples of such profiles exist like the Composite Capabilities / Preferences Profile (CC/PP) and User Agent Profile (UAProf) [171]. The Mark-up Scheme models are well-structured and well-formed models. The tags containing the data information and the data attributes are represented as a node tree under one root element. Mark-up schemes are characterized for the well hierarchical structure form they have.

Listing 3.2: XML example.

```
<people>
  <person>
    <name>Joe</name>
    <age>30</age>
  </person>
  <person>
    <name>Rob</name>
    <age>29</age>
  </person>
</people>
```

Modelling the context using mark-up languages has advantages and disadvantages. The use of mark-up languages offers the flexibility to a programmer to write her own mark-up language and there are not limitations in a restricted set of tags. The freedom offered in writing new tags, facilitates the representation of a model. The power of mark-up scheme models is not located only in the freedom in creating new tags but also in the ability of setting rules within the tags for the representation of data descriptions and data relationships. Human readability of the models can also be included as an advantage of the Mark-up models. They can be used as validation tools, define exchange formats and be used for the development of exchange data applications.

However, the mark-up scheme models have several weaknesses that make their selection for modelling the context prohibitive. Firstly mark-up schemes are very tied to the logic and language of HTML. The presentation of mark-up scheme model's data with other languages (e.g.java) is more difficult and complex. Secondly, searching for information in the data is tough. The fact that a mark-up language is content-based language and case sensitive, states difficult the information

searching. For example, if we search for "Joe" in the example provided (listing 3.2), to find the result, there must be an extract matching of the search keyword and the model's data fields. If the model is very large, then the processing is becoming more difficult. Third, the GUI is embedded in the data, and therefore a change on the presentation method of the data requires its complete recoding. The most important weakness of mark-up scheme models is tracked to the difficulty to express non-hierarchical relationships, making the data management of such model hard.

***Graphical Models*:** Murphy [116] states that Graphical models are a marriage of probability theory and graph theory. Probabilistic graphical models are graphs in which nodes represent random variables, and the arcs represent conditional independence assumptions. Therefore, they provide a compact representation of joint probability distributions. In the literature we can find two types of graphical models. The directed (e.g. Bayesian Networks) and the undirected (Markov Random Fields) graphical models. Directed is the graph in which a direction is shown for every arc. On the contrary, undirected is the graph in which a direction is not shown for every arc. One other type of graphical modeling is the Unified Modelling Language (UML). UML is a modelling context language which is structuring the context modelling based on UML diagrams. UML diagram is the partial graphical representation of a system's model which also contains semantics documentation, such as written use cases that drive the model elements and diagrams. UML diagrams represent two different views of a system model [80], the static and the dynamic view. The static view, also known as structural, emphasizes the static structure of the system using attributes, objects, operations and relationships. The structural view includes class diagrams and composite structure diagrams. Dynamic view, also known as behavioral, emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects. This view includes sequence diagrams, activity diagrams and state machine diagrams.

The graphical context representation models are more or less related to computational intelligence algorithms, data mining and classification algorithms. These types of modeling are facilitating the use of computational parameters for the production of statistical results. In the area of context awareness they have been used extensively especially for context awareness applications that are supported by artificial intelligence adaptation or reasoning methods. The UML graphical models are somehow differentiated from the statistical use of these models. The representation of concepts in the form of classes, objects and attributes to represent conceptual models, as well as the representation of models using semantics offer a different perspective of the graphical models. UML diagrams in combination to the UML language can be used as a mixture of mark-up scheme models and the ontology-based models that they will be described later in this subsection.

A context graphical model was introduced by [78] as an ORM extension (Object Role Extension) with differences from the classic ORM. Its differences are the basic modelling concepts, facts, and the involvement of the fact types and roles of a domain that an entity belongs. Henricksen et al. [78] extended the ORM to allow fact types to be categorized, according to their persistence and source, either as static or as dynamic. The latter ones are further distinguished depending on the source of the facts as either profiled, sensed or derived types [171].

***Object Oriented Models*:** An Object Oriented model is based on a collection of objects like for example the E-R model. Object oriented context modelling approaches are using the benefits of the object oriented programming possibilities: encapsulation and re-usability. With these characteristics of object oriented programming, problems raised with the dynamic context in ubiquitous environments are covered and the details of context processing are encapsulated on an object level and hence hidden to other components. We can see the use of an OOM in figure. 2 The figure represents an Object Oriented Database Model which forms an Object Oriented Database Management System. The OO Management System is the connection between the programming and the

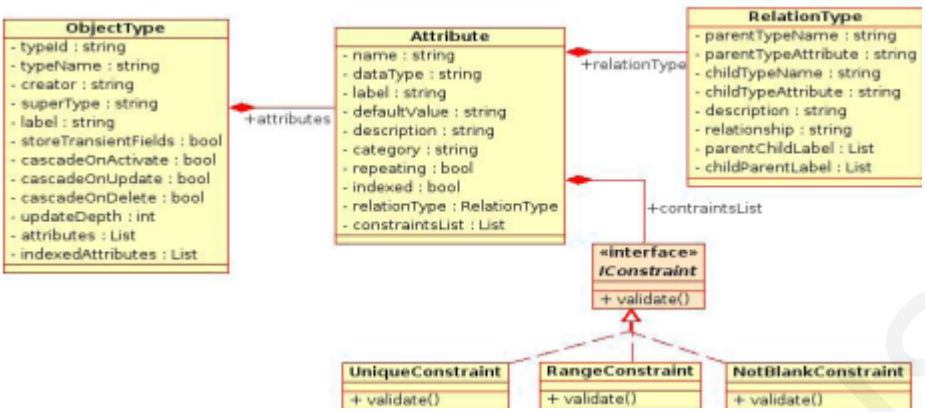

Figure 2: Object Oriented Model Example

data modelling within an OO environment. Object database permits developers to model complex data easily and capture the relationships in a natural way.

As it is shown in figure 2 the data are modeled as object types. An object represents a class with associated attributes and the meta-data like relationships, constraints, etc. of the class. Every object inherits the parent object type. The inheritance of the object types allows the evolution of types easier and increase the application scalability.

***Logic-Based Modelling languages*:** Logic-based models have a high degree of formality. Typically, facts, expressions and rules are used to define a context model. In these systems the context/knowledge is defined as facts expressions and rules. The contextual information is usually added to, updated in and deleted from a logic based system in terms of facts. Every formal logic has a clearly defined syntax that determines how sentences are built in the language, and semantics that determines the meanings of sentences, and an inference procedure that determines the sentences that can be derived from other sentences. Propositional logic, First Order logic, Knowledge Interchange Format (KIF) and Description logics are some of the Logic based modelling languages or group of languages. In table 3 we provide an example of KIF language for better understanding.

Listing 3.3: KIF example.

```
(forall((?x P)) (Q ?x)); ''All P are B''
(exist ((?x P)) (not(Q ?x)) ; ''Some P is not Q''
President 802 Thomas Jefferson ;
a record in the database of presidents
```

In the example shown in listing 3.3 we can see that KIF includes both its syntax and its semantics.

Modelling the context refers to two kinds of modelling, modelling the data and modelling the Knowledge representation. When talking about data and knowledge modelling it is not difficult to understand that we refer to repository mechanisms in which data are organized and controlled with the use of modelling languages. At this point we will describe the modelling data schemas and the ontology models, two of the most known and commonly used models Ontologies and Database Schemas.

***Ontology-Based Models*:** According to [67] Ontology is a specification of a conceptualization. Hendler [77] defines ontology as a set of knowledge terms, including the vocabulary, the semantic interconnections and some simple rules of inference for some particular topic. In a more general definition, ontology is the representation of the knowledge over a specific domain using the concepts that describe the domain and associations between the concepts with the use of semantics. Ontologies provide a number of useful features for intelligent systems, knowledge representation and knowledge engineering processes such as, usage of vocabularies, taxonomies, content theory, knowledge sharing and reuse.

Vocabulary is the list of terms in a subject area. A vocabulary in ontologies contains a finite list of terms that they are denoted with the same identifier. Hence the terms of a vocabulary can be easily processed by a machine. Part of a vocabulary in ontologies is the thesauri which provide additional semantics in the form of synonym relationships between terms.

Taxonomy is a hierarchical categorization or classification of entities within a domain based on common ontological characteristics. Ontologies provide taxonomies in a machine readable and machine processable form [60].

Content theory is the identification of classes of objects, their relations and concept hierarchies in an elaborative way using ontology representation languages.

Knowledge sharing and Reuse refers to the ability of the ontologies to be reused in several applications. Every ontology provides a description of the concepts and relationships that can exist in a domain and that can be shared and reused among intelligent agents and applications [60].

Ontologies are a very promising instrument for modeling contextual information due to their high and formal expressiveness and the possibilities for applying ontology reasoning techniques. Based on an evaluation of an ontology-based context model in [171], it is concluded that ontologies can be characterized as expressive models that fulfill most of the designers' requirements. Examples of such requirements are: simplicity, flexibility, extensibility, generality, and expressiveness.

3.4 Context Frameworks

Bardram [14] presents the JCAF – Java Context Awareness Framework which is a java based context-awareness infrastructure and programming API for creating context-aware computer applications. JCAF is a distributed, service oriented, event based and secure infrastructure. JCAF

relies on having a set of distributed context services that cooperate in a loosely coupled peer-to-peer or hierarchical fashion. JCAF' s architecture is modifiable, event based and secure. It is modifiable and extensible at runtime and not at design and compilation time. JCAF services, monitors, actuators, and clients can be added to the JCAF runtime infrastructure while running. Although JCAF is based on a peer-to-peer model, it does not support automatic discovery of peers or a super-peer. The communication is based on Java RMI (Remote Method Invocation), but it however supports various sensors for monitoring locations and base classes for describing relevant entities used in context awareness applications [168].

Salber et al. [140] present the Context toolkit. The Context toolkit addresses the distinctions between context and user inputs. It enables application developers to build context aware applications, by introducing three main abstractions, widgets, aggregators and interpreters. A widget is a component that is responsible for acquiring directly from a sensor. The aggregators can be considered as meta-widgets, taking all capabilities of widgets and they also aggregate context information of real world entities and act as a gateway between applications and widgets. Interpreters transform low-level information into higher level information that is more useful to applications. All of the components share a common communications mechanism (XML over HTTP) that supports the transparent distribution.

In Burkle et al. [30] an agent-based infrastructure is presented which provides integration and collaboration of autonomous, context aware services in a heterogeneous environment. It describes the Computers in the Human Interaction loop (CHIL) agent infrastructure and how the objectives of the proposed architecture were achieved. Kasim et al. introduce an architecture based on the Model-View-Controller for the development of interactive context aware applications. With the MVC architecture and its capabilities the architecture attempts to make the communication between the user and application more intelligible in several ways.

The Context Fusion Networks (CFN) was proposed in [34]. It allows context aware applications to select distributed data sources and compose them with customized data-fusion operators to a directed acyclic information fusion graph. Such a graph represents how an application computes high-level understandings of its execution context from low level sensory data. CFN was implemented by a prototype named Solar, a flexible, scalable, mobility-aware, and self-managed system for heterogeneous and volatile ubiquitous computing environments which first described in [139] and it consists of a set of functionally equivalent nodes, coded Planets, which peer together to form a service overlay using a distributed hash table (DHT) based P2P routing protocol such as Pastry.

RCSM (http://dpse.asu.edu/rcsm/RCSM-software.html) is a middle-ware supporting context sensitive applications based on an object model on which context sensitive applications are modeled as objects. It supports a special language for the support of situation awareness requirements and thus it provides situation awareness functionality. Based on the runtime situation analysis RCSM generates application-specific objects based on the context data that RCSM retrieves from its sources.

A project which provides infrastructure for the development of context-aware and pro-active applications is the AWARENESS. Awareness framework refers to health-care domain applications and more specifically applications in mobile networks for the health-care applications domain. It provides generic components and manages context, security and identity using web services infrastructure.

Gu et al. [68] present the SOCAM framework which is a middle-ware which supports context modeling and reasoning based on OWL and its implementation is based on RMI.

The ESCAPE [177] framework is a web services-based context management system for team work and disaster management. ESCAPE services are designed for a front-end of mobile devices

and the back-end of high end systems. The front-end part includes components support for context sensing and sharing that are based on web-services and executed in an ad-hoc network of mobile devices. The back-end includes a web service for storing and sharing context information among different front-ends.

3.5 Functional Architecture for Context Awareness Systems

Context reasoning is referred to as the task of using context data in an intelligent way [119]. The reasoning for context aware applications has been approached from multiple perspectives and views. Through the research in reasoning problems, one of the main tasks was to define a functional architecture for the context reasoning procedures in context aware systems. The attempt to create systems which deduce the relevant information, from a bigger pool of context data led to the creation of a formal model of the context reasoning architecture. The reasoning in context aware applications is performed by an individual engine which is adapted to a context aware system. This engine can have the form of a prototype, a middle-ware application or can be a component of a system.

To explain the functional architecture of a reasoning engine we will describe it as a single component within an overall architecture for context aware computing system. This is the same way [123] are exploring a reasoning engine named "ReaGine" through a general overview of it. Padovitz et al. [123] tracked five basic functional steps during the reasoning process taken by "ReaGine" as illustrated in Figure 3.

Reasoning starts when information arrives to the reasoning engine either as raw data or as basic reasoned context, then the information is checked for low level discrepancies. Following this, the data are synthesized according to concepts drawn in the conceptual model, and then the engine checks for conflicts (e.g. ambiguous situations or situations that cannot coexist) at the

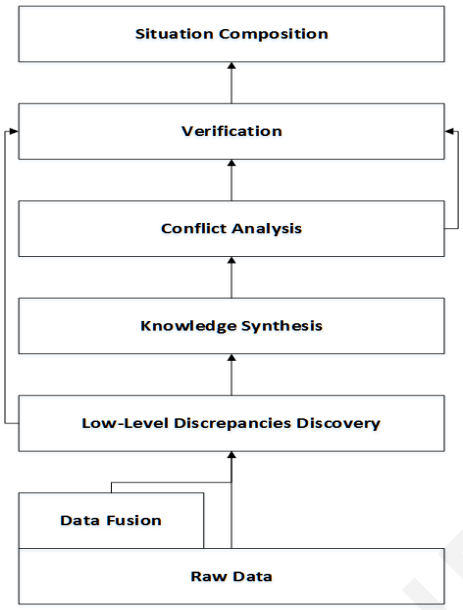

Figure 3: Reasoning Engine Functional Stages

conceptual level. Conflicting situations are dealt with in an additional verification phase for the resolution or the verification of the true situation's nature. At the final stage, the verified situations are composed of more complex situations and compared with policies predefined by the system designers.

In a more general view the reasoning engine receives the context data as raw data which is called "low level context" and the engine transforms it to "higher level contexts", which are combinations of lower level data sources [119]. The low level data are collected according to the actions or events coming from sensors or middle-ware software APIs defining the context data sources. The low level data are processed and mapped to high level context which is the data output of the engine.

The Event Collector engine is the engine which collects events and generates event notifications described according to an event model [93]. According to [93] event notification engines are using matching approaches like the matching based on simple flat, the hierarchical topic-based

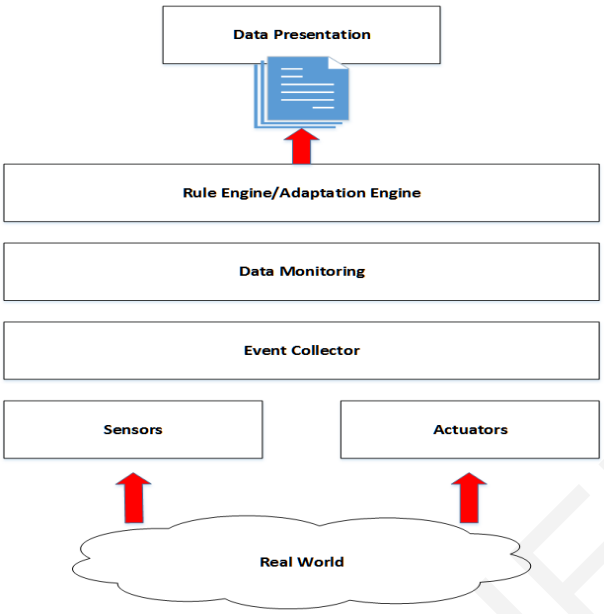

Figure 4: Context Reasoning Engine-Functional Architecture

event models or a graph structured event model. The simple flat matching and the hierarchical topic-based event models are fast but they do not allow expressive subscriptions and sophisticated content-based matching. Graph-structured event model represents the different subscriptions as a direct acyclic graph in which the set of nodes in the graph is the set of possible subscription and notification categories [91].

The Context Data Monitoring engines are related with the context inference engines that are described in the following sections. These engines use context reasoning techniques for the retrieval of context information.

The Adaptation Engine contains the functions and the formulas used by the system to make decisions for the adaptations results. The adaptation results are computed and extracted from the Adaptation Engine based on the user's profile, history or any other parameter influencing the adaptation result.

3.6 Context Reasoning Techniques

In the literature there are numerous references for the role of context reasoning. Context reasoning, as it is described by the five functional stages of the reasoning engine, could be built as an ordered list composed of each event followed by any created reasoning events, including any predictions created by the engine as part of its process [106]. Thus, context reasoning has a very important role as part of the architecture of a context system. The context reasoning is responsible for the detection of possible errors, for the completion of missing values and for the decision of the quality and the validity of the sensed data. Its importance is focused on the responsibility of transformation of raw context data into meaningful information, and the extraction of decisions that may lead to actions.

In order to make the context reasoning tasks achievable, researchers deployed context reasoning techniques such as Ontological Reasoning, Rule Based Reasoning, Distributed Reasoning, Case Based Reasoning, Offline Reasoning and Probabilistic Reasoning. These techniques are separated and distinguished in Non-Symbolic reasoning Techniques and Symbolic Techniques which will be presented in the following sections. The classification of these techniques to symbolic or non-symbolic is depended on the methods of representation of the reasoning events and situations. The methods of representation might be deterministic by using symbols and regular expressions (Symbolic) or non deterministic using probabilistic methods (Non-Symbolic).

3.6.1 Non-Symbolic Context Reasoning Techniques

Non-symbolic reasoning techniques use probabilistic methods and machine learning algorithms for the prediction of the proposed data. The architecture requirements of systems supporting learning algorithms for reasoning are not determined only by algorithms; they are also

determined by data sets and the interaction of algorithms within a larger system [40]. The architectural drivers based on the algorithm characteristics given by [40] are:

- Probabilistic Relational Model: Probabilistic Computation, large densely connected graphs, indirection over graph nodes, varying granularity, load balancing, trade off error for latency
- SATisfiability – based planner: Parallel tree traversal symbolic matching, partial results sharing and communication, any time solutions etc.
- Support Vector Machine Classification: Variable precision arithmetic on sparse vectors, flexible caching, computational density etc.

Non-Symbolic reasoning techniques are commonly used for the development of cognitive systems which are used to solve problems that are too difficult to be solved optimally or exactly so the use approximations and no zero error rates [40].

3.6.2 Symbolic Context Reasoning Techniques

Symbolic reasoning techniques are expressed with symbols. A very simple form of symbolic reasoning can be expressed as a single-case based simulation where each variable can have a single constant value only, and the association of an expression with each variable at each point of execution [20]. Consider listing 3.4 as an example of a symbolic simulation. When introducing symbolic simulation techniques, Blank et al. [20], mention that the symbolic expressions (like the example of listing 3.4) presume a large number of cases for the provision of faster and more reliable results. In the same work they also give a number of the ingredients and problems of symbolic simulation:

1. the number of simulation steps is finite and limited

2. for the process of substitution the particular semantics of the function symbols involved do not care

Ontological Reasoning, Action-based Reasoning, Distributed Reasoning and Case-based Reasoning techniques can all be considered as Symbolic forms of reasoning.

Listing 3.4: Example of Symbolic Simulation [41].

```
int test (int x, int y)
{
int z=x*y;
if (z<0)
{
//Throw exception
}
if (x<y)
   {
int temp=x
x=y;
y=tmp;
   }
int sqry=y*y;
return z*z- sqry* sqry;
}
```

An **Ontological Reasoning** example is CARE [4]. CARE is a framework supporting on-line ontological reasoning for Context Aware Internet Services. The contextual data of the framework is managed by different entities (i.e. user, network operator, service provider). The context data is collected and managed by a certain entity and it is expressed by means of references to ontological classes and relations. The context management and reasoning is achieved by the declaration of policies in the form of rules over the collected data. Other examples of ontological reasoning are PELLET [165] and Racer Pro [71].

Action-based Reasoning - Action-based policies are undoubtedly the most popular ones and are used in different domains related to networks and distributed systems, such as computer networks, active databases, and expert systems. An action policy consists of situation-action rules, which specify exactly what to do in certain situations.

Fuzzy Rules - A possibility to introduce a more human-like way of thinking in rule-based reasoners is enabled by fuzzy logic [185] and fuzzy rules. Instead of having rules that are evaluated to either true or false (binary logic), fuzzy logic allows for multi-valued logic with different degrees of truth. Thus, instead of having IF-THEN-ELSE rules we have IF-THEN rules (without ELSE), or equivalent constructs, such as fuzzy associative matrices that allow multiple rules to be evaluated and assigned different degrees of truth for the same case. A typical fuzzy rule is defined in the following format: IF variable IS property THEN decision, where variables are linguistic rather than mathematical and properties take fuzzy rather than numerical values. Thus a fuzzy variable for temperature may take values such as very hot, hot and cold rather than a specific number in Celsius or Fahrenheit. The fuzziness of properties along with the AND, OR and NOT operations of Boolean logic that also exist in fuzzy logic, allow as to assign scalar values to adaptation decisions depending on the antecedent of our rule.

Goal-based Reasoning - The benefit of goal-based policies for system administrators is that they are relieved from the task of defining actions (required in Action Based techniques), usually requiring a very detailed knowledge of the system. Under goal-based policies, the system itself must reason about a sequence of actions able to satisfy the goal.

Utility Functions - In the context of self-adaptive computing, one of the main needs is a way to detect the best variant among each possible state that the system is allowed to assume at a given time. Utility functions can be used in order to map each possible state of the system into a scalar value, so that it is possible to select automatically a new state for the system by selecting the configuration that provides the best utility value. They are mathematical equations that based on some input from the environment can assess the suitability of different adaptation alternatives.

Case-based reasoning - Case-based reasoning is the process of solving problems based on the solutions of similar past problems. The process is formalized as a four-step process:

1. Retrieve: Given a target problem, retrieve cases from memory relevant to solving it. A case consists of a problem, its solution, and, typically, annotations about how the solution was derived.

2. Reuse: Map the solution from the previous case to the target problem.

3. Revise: Having mapped the previous solution to the target situation, test the new solution in the real world (or a simulation) and, if necessary, revise.

4. Retain: After the solution has been successfully adapted to the target problem, store the resulting experience as a new case in memory.

Several Case Based reasoning applications were built, like in [101], which provides guidance for architectural design and adapts existing designs for new buildings. “AskJef” is an application using

multimedia technology for storing and presenting cases to the user. Other Case Based Reasoning applications are Casecad [102] and Archie-II [48].

Distributed reasoning - Distributed reasoning systems are the systems composed of separate modules (agents) and asset of communication paths between them. The separate modules might be individual reasoning sub paths implemented as Rule Based, Case Based or Ontological based reasoning systems. These modules can communicate between them and their combination is creating a distributed reasoning system. An example of Distributed reasoning system DRAGO [147] a distributed reasoning Architecture for semantic web, which is implemented with multiple semantically related ontologies presented with different methods (Description Languages).

3.6.3 Hybrid Context Reasoning Techniques

Hybrid reasoning techniques combine two or more of the presented Symbolic or non-Symbolic Techniques. A very common combination of techniques is the integration of rule-based and case-based reasoning. The benefit of this combination is the creation of a new scheme which is derived from the individual schemes, e.g. rules derived from cases and vice versa [132]. Prentzas [132] supports that the integration of hybrid reasoning techniques is distinguished in two categories: efficiency improving and accuracy improving methods. A hybrid approach is presented in [132], but also by [58] and [62]. Prentzas [132] proposed a hybrid reasoning approach integrating Symbolic rules with neuro-computing giving preeminence to the symbolic component. This approach is based on the example of hybrid approach given by [64] which refers to the creation of an effective scheme combining three types of knowledge representation formalisms: symbolic rules, neural networks and cases. Another hybrid reasoning approach was proposed in [52]. Their proposal is based on a relational model and an ontology-based approach for the creation of rules. The

proposed hybrid context model, HCoM, suggests separating context data management from context knowledge management, process them separately and then put the results together for better reasoning and decision support in a context aware environment.

3.7 Existing Context Reasoning Technologies

The various context reasoning techniques that were presented in the previous section can be implemented using a number of different technologies such as:

- The rule-based reasoning engine technologies for context reasoning
- The ontology-based reasoning technologies
- The Topic Maps technology

3.7.1 Context abstraction using rule-based reasoning engines

Rule-based reasoning is performed through the conversion of context data into ''facts" and the creation of rules to obtain new facts to be converted back as new context information. The functioning of a rule-based reasoning engine can be summarized as follows:

1. A Context Reasoning Provider is used for the creation of specific context reasoning sessions and their activation on specific context reasoning situations. It manages "WHEN" a context session is created.
2. Context Abstraction Rules are created to capture "HOW" the abstract context information collected by the Context Reasoning Provider is used to make the context data information more specific.
3. The context information is translated in facts to represent knowledge and to activate context abstraction rules.

4. The new rules derived from the context abstraction rules are converted back to the context representation and made available as new context information

Taveter [174] in an attempt to give a definition for the Business Rules they specified the types of Business Rules that exist in the literature. These types are extracted and applied to a model and formalize the types of rules. The types mentioned in [174] can be used to define the types of rules used for the definition of an abstract rule based reasoning engine. The types of rules are:

- Integrity constraints: Integrity constraint rule is an assertion that is based on the evolved states and the transition histories of a system viewed as a discrete dynamic system [183] Constraints can be expressed as IF-THEN statements in programming languages or as explicit assertion statements supported by programming languages such as C++, or Java 2.

- Derivation rules: Derivation rules allow the derivation of knowledge from other knowledge by an inference or a mathematical equation [183]

- Reaction rules: The Reaction Rules define the behaviour of a system by stating the conditions under which actions must be taken. These conditions are expressed in response to perceived environment events and communication events [183].

Wagner [183] gives examples of the methods of modelling and representing the rules with UML (Unified Modelling Language), SQL database queries and XML based languages. For the implementation of the rule based reasoning, several rule based engines and technologies are available, like Mandarax [47], ILOG (http://www.ilog.com/) and Jess [57] Mandarax [47], is an open source java library for business rules, including the representation, persistence, exchange, management and processing of rule bases. In Mandarax rules are presented as clauses. The clauses consist of a body which is the prerequisite of a rule and a head which is the consequence of a rule.

Both of the clauses are facts which themselves consist of terms and predicates associating those terms. Terms can be constants, variables or complex terms.

ILOG (http://www.ilog.com/) is a rule engine and programming library used for the combination of rule based and object oriented programming for the adjustment of business rules to new and existing applications. ILOG rule is composed of a header, a condition part and an action part. The header defines the name of the rule, its priority and packet name. The condition part of the rule defines the conditions that must be met such that the rule is eligible for execution. The action part specifies the activities to be carried out when the rule is fired. The ILOG engine can directly parse and output XML representation allowing the management of rules by standard XML tools.

Jess [57] is one of the most popular rule engines. It is a Java library which offers different levels of APIs for the creation of reasoning sessions, load facts and rules and run reasoning algorithms. Jess implements the RETE reasoning algorithm to fire rules and derive new facts.

3.7.2 Ontology-based inference engines

In this subsection a number of ontology-based inference engines will be described. The described engines will be F-OWL, OWL inference engine based on XSLT and JESS, Jena 2, SWRL rules from Protégé, BaseVIsor, RacerPro, FaCT++, Pellet and KAON2. F-OWL is an ontology inference engine for the Web Ontology Language OWL. The engine is using the Flora2 advanced object oriented knowledge base language which translates a unified language of F-Logic, HiLog and translation Logic into the XSB deductive engine. FOWL is reasoning using the ontology model defined by the standard OWL language. FOWL has been used as ontology reasoner in a number of intelligent prototypes like CoBra, TAGA and REI. OWL inference engine based on XSLT and JESS is an inference engine which is using the JESS rule engine and XSLT style sheets. JESS has the ability to “reason” using knowledge supplied in the form of declarative rules, and

for the translation to Jess rules using the JESS for OWL some kind of "adapters" are needed. The code provided by the project helps to load OWL ontologies and annotations into JESS, transforming them into rules and facts. Then using CLIPS (syntax) inference rules for the ontology are created. The code is organized as follows:

- OWL Meta model: description of OWL meta-model in JESS language and directly loaded into the JESS engine.
- Ontology Stylesheet: an XSLT style sheet that transforms an OWL schema into a set of JESS assertions based on the OWL Meta model. The resulted assertions can be loaded into the JESS engine.
- Annotation Stylesheet: an XSLT style sheet which transforms an OWL annotations file into a set of JESS assertions based on OWL meta-model. The resulted assertions can be loaded into the JESS engine.

Jena 2 is an inference subsystem which allows a range of inference engines or reasoners to be plugged into Jena. The engines are used to derive additional RDF assertions which are entailed from some base RDF together with any optional ontology information and the axioms and rules associated with the reasoner. With this mechanism, languages like RDFS and OWL can be supported. With the support of these languages, the additional facts to be inferred from instance data and class descriptions is allowed. Jena2 includes the predefined reasoners for RDFS, OWL-Lite, DAML and other generic reasoners for user defined rules.

SWRL rules from Protégé is an inference engine with a more complete solution related with Semantic Web. SWRL Bridge which provides the necessary the infrastructure to incorporate rule engines into Protégé - OWL for the execution of SWRL rules, is an add-in to the engine. The interaction with this bridge is achieved through a user interface called SWRLJessTab.

BaseVIsor is a forward-chaining inference engine based on the Rete network optimized for the processing of RDF triples. It has been developed to process RuleML and R-Entailment rules. It is a Rule Based system implementing a Rete based algorithm solution. The difference with JESS and CLIPS is the fact that the user-defined types cannot be arbitrary list structures but simple data structures in order to increase efficiency of rules pattern matching.

RacerPro is a core inference engine for the semantic web. It has two APIs that are used by network clients like OilEd, the visualization tools RICE and the ontology development environment Protégé 2. The Racer server supports the standard DIG protocol via HTTP and a TCP based protocol with extensive query facilities. It supports the web ontology languages DAML+OIL, RDF and OWL.

FaCT++ is a description logic reasoner written in C++. It can be used with applications supporting OWL-DL via the DIG standard; it uses a tableaux decision procedure to solve SHOIQ description logic and supports simple data types.

3.7.3 Topic Maps Technologies

Topic Maps technology is part of the ISO standards of the semantic web technologies. The ISO standard of Topic Maps is formally known as ISO 13250. In general, topic maps are used in semantic web applications for finding and exchanging information using topics. Topic Maps are designed for the enhancement of navigation and information retrieval. This is achieved by the addition of semantics into the resources and their representation as context data sets [75].

Wrightson [192], through the description of Topic Maps mentions the distinguished differences of a topic map and ontologies. Ontologies are used for the description of shared common understanding aspects, like objects or relations between them. A topic map is used for the representation of the ontologies by linking the resources belonging to them. In other words, a topic

map is the collection of topics, associations and scopes [170]. The consisting parts of a topic as they are introduced by [170] are the subject which gives the generic sense of a topic, the reification which is the act for the creation of a topic, the subject identity which is used to specify the relation between subjects, the subject indicator which is a resource indicated by the topic and finally, the topic characteristics which are the name, the role the occurrence or the associations of a topic. Topic Maps technology is used for the integration of several Topic Maps standards. Topic Maps APIs, Query standards, Constraint standards [165], are:

- TMAPI (Topic Maps Application Programming Interface)
- ISO:1848 TMQL (Topics Map Query Language)
- ISO 19756: TMCL (Topic Maps Constraint Language)
- XTM (XML Topic Maps)

In the research work of [75], three Topic Maps Tools categories were identified: *Topic Maps Engines, Topic Maps Navigators* and *Topic Maps Editors*.

3.8 Context Storage and Retrieval

Advances in context awareness research made the creation and development of context storage and retrieval, mandatory. The need for adaptation methods and the generation of recommendations within a computing system indicated that such a system is context aware. The dynamic change of data, on runtime, the storage and retrieval of data guided the research world to find solutions based on the context technologies they integrated. Siljee et al. [162], divide the concept of context awareness in three phases:

1. Monitoring of the environment,

2. Interpretation of the data (being monitored) through the context model and

3. Adaptation of the system to the changed context.

These three phases predicated the interaction with a repository of the context data and the integration of mechanisms for the update, storage and retrieval of them. In a more detailed analysis, Siljee et al. [162], are tracking the different aspects in the process of retrieving the context information. These aspects are identified as *initiative, timing, history,* and *information presentation*.

Initiative is the process of context information retrieval using *context-push* and *context-pull*. With context-push, the context information is retrieved without the need of send request to the system every time the information is needed. With context-pull, when the system needs context information it explicitly request for the needed information.

Timing includes the *event driven* and *periodic* methods of context information retrieval. Event driven method requests for context information when some event occurs. Periodic method requests or retrieves context information in certain scheduled points in time.

History includes the *absolute* and *relative* methods. Absolute method context information is taken in absolute points without using previous information. Relative is based on the difference of context information over time, thus the previous stored information is used.

Information presentation is divided in *explicit* and *implicit*. The explicit method does not need any context model and it uses the needed information as it is. By implicit method, the system refers to the context information based on its input data and thus a context model is needed.

3.9 Examples of Context Awareness Applications

In this section we will attempt to show how the context awareness technologies described in section two enable the production of Context Aware applications. Context Aware applications

cover a wide spectrum of different research areas of computer science and information technology. Therefore, we will attempt to show how the context awareness technologies are enabled through context awareness applications in the several research areas.

Based on a list of features that a context awareness application must support [13] [14] [128], the following three categories of context are identified: 1) presentation of information and services to a user; 2) automatic execution of a service; and 3) tagging of context to information for later retrieval. Bontas [23] introduced a list of examples that show the variety of context usage forms in computer science like Domain Classification, Natural Language Processing, Information Integration and Mobile Computing. Taking into consideration the examples given by [23] for each form of context usage in Computer Science and the categorization of context, it becomes apparent that context-awareness is not solely a characteristic of computer applications of a particular domain exclusively; rather it can be applied in several domains in different ways.

The area of health care research appears as one of the interests for a potential testing area of tools and frameworks which support context awareness. Examples of such tools are Vocera communication System [167] and MobileWORD—Mobile Electronic Patient Record [89]. Vocera [167] is a communication badge system for mobile users has from a push to call button and a small text screen, as well as voice dialing capabilities based on voice recognition. It allows hands-free conversations, voice messages and it is biometrically secured with speaker verification. It delivers the data directly to the users without the need to use a distance device, like a phone or a PC. The MobileWARD is a hospital prototype which supports the morning tasks in a hospital ward and it is able to display patients' profiles and information. The information and the functionality presented by the application depend on the nurse's location and the time of the day. Other context awareness examples in the health-care sector are Context-aware mobile communication—CICESE

[56], Intelligent hospital software [111]. CICESE is a context aware mobile communication application which is used by hospital workers to carry out their tasks. Its contextual elements include: location, delivery timing, role reliance, artefact location and state.

As ubiquitous computing is a post-desktop model of human computer interaction, this area of computing research deals with the mobile computing, wearable computing, with the aim of such research applications to locate and serve the user. The basic mechanism of context awareness computing in this area is summarized in [2] into four steps:

1. Collect information on the user's physical, informational or emotional state.
2. Analyze the information, either by treating it as an independent variable or by combining it with other information collected in the past or present.
3. Perform some action based on the analysis.
4. Repeat from Step 1, with some adaptation based on previous iterations.

An example application presented by [2] is the CyberGuide application which uses the capabilities of the Personal Digital Assistant (PDA) to locate the position of a user, and give them directions and information for sights, restaurants or hotels that are close to them. The CyberGuide is an event-driven model where components act as event sources.

Tarasewich [173], presents the principles of mobile commerce and states that designing successful m-commerce applications and their interfaces, is dealing with context. As a step beyond the traditional wired web-commerce today the commerce research is taking advantage of the mobile applications, like PDAs, Bluetooth technologies, etc. With mobile computing, people might be anywhere, any time. The e-commerce area could solve the problem of mobile users by just determining the location of users. But this is something more complex. Mobile application's use can vary continuously because of changing circumstances and differing user needs [173]. This is the

reason behind the need of creating context models in mobile commerce research area. Chen et al. [32] give a summary of a context awareness application. They describe the application proposed by [7] which determines the location of a customer within a store, and gives the customer information about the items, such as how to locate them, point out items on sale, make comparative price analysis.

In learning, the adoption of context awareness is not a new idea. It has been demonstrated in learning systems for quite some time. Classical methods, as those encountered in early intelligent tutoring systems [54] and student modeling [29] can all be regarded as context-aware approaches used as adaptation methods [29]. They support the idea that the problem in adaptive interfaces is located in the complexity of the interfaces used in such applications. Adaptive interfaces can be the starting point for the determination of the significance of the existence of context awareness in e-learning applications. The usual e-learning applications are not supporting adaptation methods, thus users with different abilities, web experience, knowledge and background get the same pages in the same context [29]. Recent Adaptive Educational Systems, most of them web-based [28], promise to offer adaptation with respect to the presentation of the learning material, the navigation support, the curriculum sequencing and support in problem solving. One of the methods used for establishing adaptivity and adaptability is context awareness. However, the proposed exploitation of contextual information from a broad spectrum as a means of enabling the best possible support in collaborative learning that aims at creative knowledge building is a novel and promising area.

3.10 Conclusions

The chapter presented a comprehensive literature review analysis for the topic of Context Awareness. The analysis presented the current most known and mot used Context based technologies. It presented the methods for modelling and representation of Context. It presented the

frameworks and tools that are commonly used for modelling and representation of Context and depicted several examples of Context Aware research articles. The findings of the chapter include the Context reasoning methods and examples of Context reasoning that can be found in the existing literature.

Chapter 4

Recommender Systems Review: Types, Techniques and Applications

4.1 Introduction

Recommender or Recommendation Systems (RS) are software tools in applications or websites that suggest information (e.g. items, people, news articles, etc.) that might be of interest to the end user, taking into account various types of knowledge and data, such as the user's preferences, actions, tasks, context, etc. In most cases these systems use computational methods to analyze users' past actions and decisions, along with other user-related or task-related information, to offer useful, usually personalized recommendations for an individual user. The motivation behind this is to alleviate the information overload problem, by bringing to surface what is most relevant, interesting to the user, filtering out anything what is not. Examples can be seen in many well-known e-commerce websites such as Amazon.com, which promotes products, such as products that were last examined or purchased by a user, or products that have been rated or reviewed by other users. In addition to that, there are cases where the recommendations are the result of a combination of factors that are difficult to determine accurately. In such cases, a variety of alternative methods are employed to generate recommendations.

Recommendation systems belong to the information filtering systems family and, therefore, seek to predict the rating or the preference that a user would give to an item. Thus the common methodology that the recommendation systems follow, is to determine the relation between three types of modeling, user, rating and item, in order to produce recommendations. All recommendation algorithms and their variations are following this model for the computation of recommendations.

This chapter includes a comprehensive critical review of the different types of recommender systems, their typical architecture and the algorithms used for generating recommendations, while at the same time identify their strengths and weaknesses. Some of the most popular recommendation systems are described, and finally, the challenging topic of evaluation of RS is discussed. It outlines the possible approaches to assess the accuracy, usefulness and user satisfaction from recommendations. In existing literature, Recommendation Systems, are classified into three categories based on the type of recommendation filtering method they use: Content-based filtering recommendation systems, Collaborative recommendation systems and Hybrid approaches. For each one of them, several algorithms, filtering techniques and applications can be found. The recommendation systems research community is one of the largest and most active ICT communities, with continuous inputs and findings in the topic.

Internet is a huge source of information where almost everything can be found. Nevertheless, this information is not structured and it does not have any kind of organization to facilitate users to find exactly what they are searching for. The expansion of the web made the information structure and consequently the information filtering more complex. This complexity occurs because more factors related to the information have been introduced. Such factors are the social, psychological, behavioral and other factors related to the users who receive or create the information. Recommendation systems are mechanisms that are used for filtering and removing the irrelevant

information based on how each user perceives the information. In other words, recommendation systems take into account the preferences of a particular user, they compare it to what other users with similar preferences liked or disliked, and try to predict the information that satisfies user the most. Based on this logic, several recommendation algorithms have been implemented and used in commercial as well as research recommendation tools.

4.2 Recommendation Systems

The current chapter approaches recommendation systems through the presentation of recommendation types, the filtering methods, the well-known recommendation frameworks and the most common recommendation algorithms' evaluation metrics.

4.3 Recommendation Filtering Techniques/Algorithms

In general, recommendation systems refer to the production of recommendations for a user, where these recommendations are useful to the user for the accomplishment of a task. This task might be related to web pages navigation, find items to buy, explore learning resources, find people to socialize with or collaborate with. The types of recommendations are usually domain and task depended and consequently context depended. In this section, the high level recommendation systems architecture is described, as well as the several recommendation techniques that can be used for the development of recommendation systems. The defined techniques are later used to describe how they can be adopted by each particular recommendation filtering method.

4.3.1 Types of Recommendation Systems

Content-based are the recommendations that a user receives based on their own past preferences. Content-based recommendation systems produce recommendations by filtering data, taking

into account mainly textual parameters. These systems use the words of the saved texts as filtering features and they learn user's profile based on the presence of features in objects that the user has rated. A content-based filtering system selects items, based on the relation between the content of the items and the user's preferences. These preferences are filtered by a collaborative filtering system that chooses items based on the relevance between people with similar preferences.

Collaborative recommendations are recommendations given to the user based on the similarity of the taste and preferences that other people have in relation to an active user. Collaborative filtering systems try to predict the utility of items for a particular user, based on the items previously rated by other users [3].

Hybrid recommendations are the recommendations produced by combining the collaborative and content-based methods. The hybrid approaches aim to overcome certain limitations that the first two approaches may have when they are applied individually [11] [15].

4.3.2 Functional Architecture of Recommender Systems

A recommender system consists of cyclic functioning procedures that are divided in the following four steps:

1. The collection of data
2. Filtering the data
3. Rank the recommended items
4. Presentation of the data to the user

By the completion of the last four procedural steps a recommender system aims at two tasks. Firstly, the production of recommendations and secondly, use the users feedback after the delivery

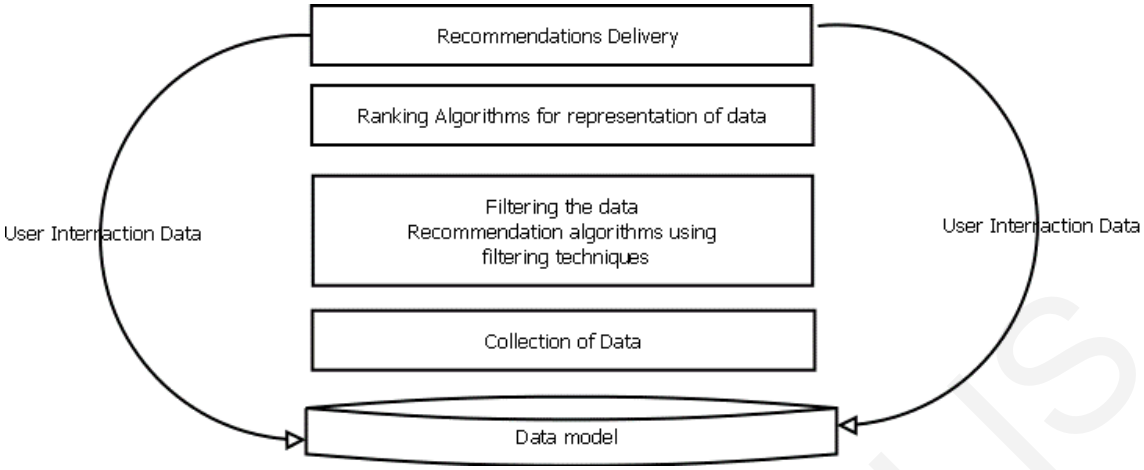

Figure 5: Functional Recommendation Systems Architecture

of the recommendations to them, so the process can be repeated and produce new recommendations as it is depicted in Figure 5.

Collection of data - The collection of data correlates directly with the data model that is used within a software application. The data is usually defined based on the overall design of a software application and the contextual information that a software application collects and processes into further computations. The data model usually depends on the domain that a recommender system is built for. The storage and representation handling methods of the data model can be designed using "knowledge representation" or "context representation" methods, such as relational databases or semantic web representation methods.

The collection of data depends on the type of data, and the recommendation filtering techniques that a recommendation system uses. For example, for the recommendation of items, such as products, the data model is usually designed in the form of statistical analysis that takes into account the frequency of views, likes or dislikes of products. On the contrary, for the recommendation of articles or news-feeds, data model usually collects semantic meta-data relations, such as keywords. Thus, data model design is directly related to the recommendation filtering techniques

that a recommendation system adopts. Based on the selected techniques the recommendation systems are classified in the corresponding type that they belong.

Filtering the data - As mentioned, above the recommendation filtering techniques depend on the type of the processing data and the type of the produced recommendations. The recommendation filtering techniques are divided into three categories according to the type of data they use for the computation of recommendations, as well as the used computational algorithm methods. The three categories are the Content-based Filtering (CBF), the Collaborative Filtering (CF) and the Hybrid Filtering (HF) techniques.

Ranking Algorithms and Representation of recommendations - Data collection and process by the recommendation filtering techniques, produce a set of recommendations that fulfill the design rules that are set, based on the design requirements, of a recommendation system. The last that a recommendation system has to complete is the presentation of the produced set of recommended items to the final recipients, the users. At this point, recommended items must be ranked based on the computed users' preferences. A recommendation is considered successful when the highest in priority recommendation offered to a user is closer to the user's interests or preferences, and when the recommended item is actually useful to the user.

4.3.3 Recommendation filtering techniques

TF-IDF - One of the most-known and most-used content-based filtering techniques is the TF-IDF (Term Frequency - Inverse Document Frequency) measure [141]. TF-IDF measure is defined as follows: For total number of documents N that can be recommended to users keyword k_i

appears in n_i of them. If $f_{i,j}$ is the number of times that k_j appears in document d_j then $TF_{i,j}$ is the term frequency of keyword k_i in document d_j and it is defined as

$$TF_{i,j} = \frac{f_{i,j}}{\max_z f_{z,j}} \tag{1}$$

Keywords used in equation 1 may appear in many documents, and keywords in that case are not useful for the distinction between a relevant and an irrelevant document. For that reason, the inverse document frequency (IDF_i) is used in combination with the simple term frequency ($TF_{i,j}$). The IDF_i is defined as

$$IDF_i = \log\left(\frac{N}{n_i}\right) \tag{2}$$

Finally, TD-IDF weight for each keyword k_j in each particular document d_j is defined as

$$w_{i,j} = TF_{i,j} \times IDF_i \tag{3}$$

and the content for each document d_j is defined as

$$Content(d_j) = (w_{1j}, ..., w_{kj}) \tag{4}$$

Naive Bayes - Naive Bayes algorithm is a machine learning probabilistic algorithm and belongs to the general class of Bayesian Classifiers. Bayesian Classifiers construct their models based on previous observations, which are used as training data. The Bayesian method estimates the a-posteriori probability of document d belonging in class c which is the probability $P(c|d)$. The a-posteriori probability is computed based on the a-priori probability $P(c)$ which is the probability of observing a document in class c, the probability $P(d|c)$, which is the probability of

observing the document d given the class c and the probability $P(d)$, which is the probability of observing the instance d. The Bayesian theorem is expressed as:

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)} \tag{5}$$

and a document is classified using the class with the highest probability

$$c = argmax_{c_i} \frac{P(c_j)P(d|c_j)}{P(d)} \tag{6}$$

Rocchio's algorithm - Rocchio's Algorithm applies the *relevance feedback technique*, which helps users to incrementally refine queries based on previous search results. Rocchio's algorithm represents documents as vectors, so that documents with similar content have similar vectors. Each component of such a vector corresponds to a term in the document, typically a word. The weight of each component is computed using the TF-IDF term weighting scheme. Learning, is achieved by combining document vectors (of positive and negative examples) into a prototype vector for each class in the set of classes C. To classify a new document d, the similarity between the prototype vectors and the corresponding document vector that represents d are calculated for each class (for example by using the cosine similarity measure). Then d is assigned to the class whose document vector has the highest similarity value. A formal representation of Rocchio's algorithm is depicted in [96] by the following equation:

$$\omega_{ki} = \beta \sum_{|d_j \in POS_i|} \frac{\omega_{kj}}{|POS_i|} - \gamma \sum_{|d_j \in NEG_i|} \frac{\omega_{kj}}{|NEG_i|} \tag{7}$$

Where ω_{kj} is the TF-IDF weight of the term t_k in document d_j, POS_i and NEG_i are the set of positive and negative examples in the training set for the specific class c_j, β and γ are control parameters that allow setting the relative importance of all positive and negative examples.

To assign a class c to a document d_j, the similarity between each prototype vector c_i and the document vector d_j is computed, and c will be the c_i with the highest value of similarity.

Decision Trees - A decision tree is a collection of nodes arranged as a binary tree. The arcs of the tree represent decisions and the nodes contain the classified objects. For example, for the classification of an item, the root is the starting point where the value of the item is set, with the possible decision values as the arcs coming out of this value. The decision arcs can take values, such as true or false. For each decision value the tree expands containing new children nodes, which contain new decisions until a leaf is reached. For the case of a true - false decision tree, if the decision referring to the root node is true, then the right child path is followed, and if it is false, the left child path is followed. This is repeated for each child node until a leaf is reached. A leaf is the last child node of the path and this contains the classification value of the item.

Construction of a decision tree requires the selection of values for each node that a tree includes. The selection of the node values can be achieved in many ways, and the task of the selection is based on the arrangement of nodes into positive and negative values related to the decisions. The division of nodes into positive and negative can be done in multiple levels, where each level consists of an individual decision that routes to the final one that classifies an item.

The decision tree classifiers tend to take a long time to construct. For example, the usage of decision trees in recommendation systems presumes one tree per user. This makes the construction and the classification of an item a high-complexity process, since each tree has to look through all item profiles and consider many different decision values. Therefore, decision trees method is usually used for small size problems.

4.3.4 Similarity Distance

The recommendation systems tend to use similarity measures to classify or group recommendation results together (clustering). In existing literature, several recommendation techniques that apply different similarity algorithms either to classify or to cluster objects, can be found. The selection of such algorithms usually differs in the type of recommendation that an engine produces. For example, the selection regarding the expected recommendations are user-based or item-based. In the part that follows, the most commonly used similarity distance methods are presented.

k-NN nearest Neighbors similarity - The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples. In the classification phase, k is a user-defined constant, and an unlabelled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point.

Pearson correlation based similarity - measures the tendency of two series of numbers, paired one to one to move together. The Pearson correlation is a number that corresponds to each pair of numbers and its value belongs to the range between -1 and 1. The Pearson correlation measure shows how a number in one series is relatively large when the corresponding number in the other series is high, and vice versa. It measures the tendency of the numbers to move together proportionally, such that there is a roughly linear relationship between the values in one series and the other [122]. The Pearson correlation value is close to 1 when this tendency is high and when it appears to be little relationship, the value is close to 0. When it appears to be an opposing relationship, for example when one series of numbers are high and the other series of numbers are low, then the value is close to -1.

Euclidean distance - The similarity between items is measured based on the Euclidean distance between the items. In this case, the items are represented as points in a space of many dimensions whose coordinates are the defined preference values. This similarity measures the Euclidean distance d between two points. The use of this value alone does not consist of a valid similarity metric since larger values means more distant and, therefore, less similar items [122].

Cosine similarity measure - In the multidimensional space the preferences related to an item are the coordinates of a point. The cosine similarity measure uses the geometric representation of the similarity between the items by measuring the angle between lines that start from the $(0, 0, 0, ..., 0)$ point of the multidimensional space to the point that corresponds to items. So, for instance, for two items there will be two lines having the $(0, 0, 0, ..., 0)$ as common point and their similarity is computed based on the angle which is created by the two lines. Two items are similar when the angle is relatively small and the similarity becomes weaker while the angle is growing [122].

Spearman correlation distance - is a variant of the Pearson correlation distance. Spearman correlation computation is based on the relative rank rather than the original preference values. For example, in a list of recommended items the least preferred item's preference by a user is overwritten with 1 and the next least preferred item's preference value is changed to 2, and so on. With this method, the recommendation system preserves the essence of the preference values using their ordering, but removes information about how much more each item was liked than the previous one [122].

Tanimoto coefficient - The Tanimoto coefficient completely ignores the preferences values. The method computes the similarity distance as the ratio of the size of intersection to the size of the union of their preferred items. In other words, when two users' items completely overlap, the result is 1.0 and when they have nothing in common, it is 0.0. The value is never negative.

Similarly to Tanimoto coefficient, there is another similarity metric called log-likelihood. Log-likelihood is more of an expression of how unlikely it is for two users to have so much overlap, given the total number of items and the particular items that each user has a preference for [122].

4.4 Categories of Recommendation Systems

The recommendation filtering techniques and the method that each recommender system employs in the several domains of application, can divide recommendation systems into categories. In this section, the most common categories of recommendation systems are presented, as well as examples of applications where they are used. In addition, an attempt to identify each recommendation type's main strengths and limitations, which are summarized at the end of the section at Table 2 .

4.4.1 Content-Based Recommendations

As already mentioned, content-based recommendations are taking into account the textual information that is used to describe an item. Thus, content-based methods are using text comparison algorithms. The usual methodology used for implementing content-based recommendations is the combination of content-based filtering techniques and collaboration-based filtering techniques. Although this combination classifies those recommendations into the hybrid recommendation models, the use of content-based techniques will be discussed in the current paragraph in particular. Pazzani [130] separates the content-based recommendations into a two-phase methodology; firstly, is the recommendation of items to a user based on the description analysis of the item in relation to the interests of a particular user and secondly, the implementation of strategies for the representation of items, the creation of user profiles that contain description of the types of

items that the user likes or dislikes and finally strategies for the comparison of the user profile to some referenced characteristics.

Lops et. al [96] provides a high level of architecture that the content-based recommender systems can adopt. Based on this architecture, the content-based recommender systems follow a three-step process, in which for every step, specific components must be developed. The three steps of the process are *the Content Filtering, the Profile Learner* and *the Filtering Component*.

Strengths

User Independence - The computation of content-based recommendations is taking into account the active user' s profile and the individual ratings provided by the active user. Collaborative recommendation methods use other users' ratings to provide recommendations to the active user based on what other users liked or viewed. In Content-based Recommendations exploit solely rating results and these results are used for the construction of each particular user' s own profile.

Transparency - Content-based recommendations are produced in a transparent way. The methods, the factors or the rules that are used to the computation of the recommendations are usually provided to the user in the form of questions or interactive tools that they collect information from the user in order to adapt the recommendations on her preferences. Thus, recommendation results are usually delivered to users with additional information explaining why and how an item was recommended.

New Item - Content-based recommender systems can recommend items that have not been rated by other users yet. The content-based recommendations can analyze the new item's semantics and use them in the recommendation filtering methods directly.

Weaknesses

Limited Content Analysis - The content-based recommendations are directly connected to the information that describes an item. This information might be the item's description, the

item's meta-data or any other information that can be used as item's "profile" information. The more information data an item contains, the more accurate recommendations can be produced. In reality, this is very difficult to achieve because of the unlimited existing data sources, such as web resources or digital libraries. This makes defining the level of relevance or identifying the structure of the content information, which is retrieved from different sources, hard.

Over-Specialization - The content-based recommendations are usually produced based on the comparisons between items that they have already rated. This causes the limitation in producing recommendations of unexpected items. This is also called the serendipity problem [96] to highlight the tendency of content-based recommendation systems to produce recommendations with a limited degree of novelty.

New User - Content-based recommendation systems are using the user profile attributes and the ratings collected by each particular user to produce their recommendations. Therefore, when just a few ratings are available for a new user, it is very difficult for the system to use the new user's profile to produce reliable recommendations.

4.4.2 Collaborative Recommendations

Terveen [175] defines Collaborative Filtering as the exploration of techniques for matching people with similar interests and then produce recommendations on this basis. This approach prerequisites the participation of many people, the existence of an easy way in which people are able represent their interests and finally, the adopted algorithms must have the ability to match people with similar interests. Collaborative Filtering Recommendation systems are commonly applied to systems that are used by communities of users who are able to express preferences for items. The recommendations produced by those systems are based on the users' preference similarities. For example, for an on-line bookstore user A and user B have similar tastes in

books because they rated a particular book highly. Then if a user C rates the same book with a high score then user C most probably likes the books that A and B liked as well. In other words, Collaborative Filtering is based on relationships between users, where these relationships are created based on their preferences.

Collaborative recommendation algorithms are classified in two categories based on the recommendation technique each algorithm is applied with. The two categories are the *memory-based* and the *model-based* recommendation algorithms. In memory-based algorithms, the entire set of data of a space S is processed in order to create recommendations. On the contrary, model-based recommender algorithms use a subset of an entire set of data to produce recommendations. Memory-based algorithms in their majority correlate with Collaborative Filtering in combination with Content-based filtering techniques, and model-based algorithms are usually used in Rule-based or Hybrid filtering recommendation systems which are presented later in this chapter.

Schafer [143] mentions that Collaborative Filtering is more effective in domains with certain properties and groups. Their properties are defined regarding the distribution of data, the underlying meaning and the data persistence. These properties are considered necessary for the implementation of Collaborative Filtering algorithms. This, however, does not presume the existence of all properties as a whole.

Data distribution contains properties related to the amount of data that surrounds the items or the users and which they can be used for the computation of recommendations based on them. For example, the number of items, the number of ratings for items, the number of users rating the items and the ratings of multiple items by users are properties belonging in the Data distribution group [143].

The grouping of properties based on the Underlying meaning, contains properties that are related to the taste and the preferences of users, in relation to items. It contains properties that

represent data regarding the commodity of tastes between users within a community of users, the similarity of tastes/preferences of users in relation to items and, finally, it contains properties related to the homogeneity of the items that are used by the recommendation systems.

The Data Persistence grouping is taking into account properties that are related to the time period that the data become relevant and, consequently, valid for producing relevant recommendations. These properties are related to the items persistence since, in some cases, items are changing rapidly, e.g. the news that are changing every day and also the taste persistence, like in example the changing of taste in books or music.

The use of properties that a recommendation technique is using, in combination with the selected technique can distinguish the Collaborative Filtering in Memory-based and Model-Based Collaborative Filtering recommendation systems. As mentioned above, the Memory-based Collaborative Filtering uses the entire or sample of the user-item relationship to generate a prediction. Every user is part of a group of people with similar interests, and thus in memory-based Collaborative Filtering a usual methodology to use is the finding of users similar to the active user (called neighbors). Based on the preferences of the users who belong in the neighborhood Collaborative Filtering, then generate preference relationships and recommendations. From the described techniques in the previous paragraph, in memory-based recommendation filtering techniques the similarity distance techniques used to measure the distances between the users' preferences can be included. By including these, it is possible to cluster or classify users into groups based on their preferences. Model-based recommendation techniques use more complex computations, which are based on data modelling and probabilistic models. The model-based techniques use machine-learning algorithms and training data, to produce intelligent predictions based on the trained data-models. Examples of model-based recommendation techniques are the Bayesian recommendations and Rocchio's algorithm based recommendations.

A comparison between Model-based and Memory-based recommendation algorithms is depicted in Table 1, which summarizes the strengths and weaknesses between the two categories of Collaborative Filtering techniques.

Strengths

One of the strengths of the Collaborative Filtering recommendations is its ability to collect information about the produced recommendations. The Collaborative Filtering techniques use all the data that is related to an item and thus, after the recommendation is computed the data related to each item could be used as explanatory information presented to the user. This helps the user understand why an item is recommended to them and how the recommended item can be used.

Easy creation and usage - Collaborative Filtering algorithms in some cases might be complex especially in the probabilistic and AI ones. In contrast to that, the development of a Collaborative Filtering Recommendation system is more easily created and used. This is because there is no need to transform or adjust the data that will be analyzed by the recommender system since this is done by the algorithms applied. Thus, a Collaborative Filtering algorithm can be very easily applied to a recommendation engine.

New data can be added easily and incrementally. In CF the data are used as input to the CF algorithms to be used for the computations of the recommendations. The collection of the data is based on the data model that a system is using and it has nothing to do with the CF algorithms. Therefore, new data can be easily added in a recommendation system and the new data will participate in the CF executions without the need for modifications to the algorithms.

Weaknesses

Data Sparsity - Collaborative Filtering recommendation systems usually analyze large datasets for the production of recommendations. In many cases, the relations between users and items appear to be extremely sparse especially in the cases of a new item or a new user. This problem is

Table 1: Strengths and Weaknesses of Collaborative Filtering techniques categories

Collaborative filtering techniques	**Strengths**	**Weaknesses**
Model Based	Is highly scalable regardless of the amount of data Most calculations are done off-line so that may be highly sophisticated and scalable A list of recommendations is produced very quickly	The results are not so accurate because the data used in the processing is a subset of the complete dataset New data cannot easily be used in the computations because the large amount of data that is already involved in the calculations can rarely overlap
Memory Based	High level of accuracy because of the use of the complete dataset The implementation of these algorithms is simple because the results of these algorithms must be produced fast All the data correspond to ”real time” situation	Sensitive to profile injection attacks All calculations must be made on-line The cold start situations are very common They appear to have performance problems because of the growing volume of data which are stored in memory.

also known as Cold Start problem [194] [3]. In such cases it is very difficult to find similar users or similar items because of the lack of information of the new entered user or item which can cause the lack of recommendations or the inaccurate recommendation results.

Scalability - Collaborative Filtering is using the whole set of data for generating recommendations. The continuous growth of data repositories is usually causing scalability problems since for large repositories the filtering is done through millions of items. The problem of scalability is mostly observed in web-based recommendation systems where the rabid increment of data repositories is huge and, therefore, the scalability problems reflect in problems related to the time of execution and problems related to the accuracy of the recommendations.

Grey Sheep - Refers to the users whose opinions do not agree or disagree with the groups of users belonging in a system. Therefore, those users cannot benefit from collaborative filtering since the lack of relationships between a Grey Sheep and the other users of the system makes the production of recommendations difficult.

Synonymy - Refers to the tendency of some items that are similar or the same, to appear with different names. In these cases, it is noticed that these items are stored as different items and they are treated as individual items instead of one common item.

Shilling Attacks - In a recommendation system where everyone can give the ratings, people may give lots of positive ratings for their own items and negative ratings for their competitors. It is often necessary for the collaborative filtering systems to introduce precautions to discourage such kind of manipulations.

4.4.3 Knowledge-based recommendations

Although that the most used recommendation systems are the collaborative filtering recommendations and the content-based systems (especially for commercial purposes), there are cases

that the two types of recommender systems are not so convenient to use. More specifically, in cases where a lack of ratings exists or in cases where the recommendations are expected to be the outcome of explicit requirements from complex domains, it is difficult to use the two recommendation systems mentioned above. For example, five year-old-ratings for computer systems might be rather inappropriate for content-based recommendations, or a complex query for car searching with explicit requirements such as "black car with maximum price x" cannot be used by the typical, collaborative or content-based recommendation systems [82]. These limitations can be dealt by using recommendation systems known as Knowledge-based recommendation systems which do not directly use the ratings given by the users, but try to predict the ratings through similarity metrics between the users and the items related with them, or with the use of recommendation rules that take into account the requirements that the user explicitly inputs. Knowledge-based recommendation systems produce results through interactive methodologies. The usual methodology used is the presentation of recommendations produced by content-based similarities based on the input given by the users and then reproduction of recommendations based on refined input given by the users. Burkle [30] characterizes these systems as conversational systems that guide the user in a personalized way to interesting objects or useful objects in a large space of possible options or that produce such objects as output. Knowledge-based recommendation systems are characterized by the "short term" information retrieval. Their common attribute is the fact that the recommendations generated are filtered through existing databases or knowledge bases which are not dynamically changed because of ratings or recent preferences. Therefore, they are mostly used in systems where the users are visiting to get the information needed and once they get the expected results, they might not use the recommendation system again (e.g. find a car and after buying the car they do not search again for cars).

Knowledge-based recommendation systems are divided into two subcategories: *1) Case Based Reasoning Systems and 2) Constrained Based Systems*. Both types are based on producing recommendations by filtering existing items in a knowledge base, with different attributes by limiting the recommended subset to include items that are closer to the user's preferences. The two knowledge-based algorithms differ in the interaction methods they use to collect the information needed to filter the main set of items into the most relevant subset of items. For a better understanding, the two types of knowledge-based recommender systems are described below.

Case Based Systems - Case-Based Recommendation systems are based on the case-based reasoning. Their reasoning relies on the similarity between an existing case and the solutions that already exist in a database. The interaction with a CBR System consists of four steps cycle Retrieve, Reuse, Revise and Retention [43]. The interaction starts by the definition of a Case which is the description of the requested target with the use of query attributes. The query attributes are defined by the user. After the definition of the case, the next step is the retrieval of previous cases that can solve the current case. Then the user can select one of the recommended solutions as the one which is closer to their initial preferences (reuse) and then revise their query based on the last selection. This can be repeated until the recommendation results are closer to the initial case and the user can retain the best solution provided by the system.

Constrained-Based Systems - Based on a given set of preferences by the user, Constraint Based Systems provide a set of possible solutions including explanations as to why these solutions were selected. A Constrained Satisfaction Problem as it is defined in [82], is described by three sets of variable V, D and C. V is a set of the customer requirements combined with a set of product properties. For example, customer requirements can be the maximum and minimum price of a product which denotes the price range of a product that the user is searching for. A product

property can be the CPU speed which denotes a computer system property or the max-storage which denotes a property for a hard drive, usb or any other storage device.

As mentioned in [82], Constraints set can be of three types: Compatibility Constraints, Filter conditions and Product constraints. Compatibility Constraints define the allowed property definitions that a user can set; Filter conditions define the relationships between the user properties and the product properties and Product constraints define the currently available product assortment.

Strengths

- The repetitive interaction helps the user to get the most relevant to the initial task results
- Provision of recommendations without being necessary to create user profile

Weaknesses

- Cold Start is one of the problems. Without an existing database it is not possible to ably a knowledge based recommender system
- The results are not ranked. The priority of the results can be based on explicit properties given by the user and not based on the similarity between the results or based on the user preferences.
- Re-use of existing solutions is not used to build user-to-user relations because of the short term usage of the systems

4.4.4 Trust-based Recommendations

Trust-based recommendations (TR) are an enhancement of the classical recommendation techniques that attempted to improve the accuracy of the recommendation results taken from the well-known collaborative filtering techniques. The logic behind TR is the use of graphs representing

the relation between users and items based on their connection on particular attributes. TRs are commonly used in social networks where a huge number of users are connected within the network and usually the users are connected because of a reason or an attribute. O'Doherty [120] mentions "trust has been shown important improvements to the traditional techniques. The differentiation between the trust-enhanced methods is the acquisitions of trust values between pairs." The enhancement of trust-based techniques in relation to the traditional collaborative techniques considers the replacement of the similarity measure used in the formulas, with the trust factor between pairs of users. This factor is used as a weight factor for the ratings of items during the computation of recommendations $r(u, i)$ where u is the user and i is the item.

For better understanding of the Trust Recommender Systems we present some of the trust enhanced techniques and recent research works related to TR systems.

Trust – Based weighted mean:

$$r_{u,i} = \frac{\sum_{v}^{v \in R^T} t_{u,v} r_{v,i}}{\sum_{v}^{v \in R^T} t_{u,v}} \tag{8}$$

The Trust-Based weighted mean present the trust between users' u and v as a weight value in place of the similarity measure between users' u and v. Trust-based Collaborative Filtering

$$r_{u,i} = mean(r_u) + \frac{\sum_{v}^{v \in R^T} t_{u,v}(r_{v,i-mean(r_u)})}{\sum_{v}^{v \in R^T} t_{u,v}} \tag{9}$$

This formula is a refinement of the Resnick's formula where the Pearson's Correlation Coefficient is replaced be the trust factor t as a ratings weight factor. Trust filtered mean

$$r_{u,i} = \frac{i}{|R^T|} \sum_{v}^{v \in R^T} r_{v,i} \tag{10}$$

This equation presents a trust-filtering method, whereby the users who rated item i are filtered according to their trust values, where only the users who are trusted above a certain threshold are used in the computation of predicting a rating. Using these users, we then take a simple average of their ratings for item i. This method provides results according to the idea that "*users are more likely to accept recommendations from their most trusted friends*" [120].

In the existing literature more techniques and trust-based enhancement can be found such as "*Ensemble Trust*" [181] and Trust-filtered collaborative filtering [121].

Strengths

- High accuracy recommendations, better than the usual collaborative filtering techniques
- Better results in relation to the satisfaction and coverage

Weaknesses

- Cold start is one of the problems that Trust algorithms can face, since there must be a network with the corresponding associations between the users.
- The evaluation results are analogous to the size of the network and the associations between the members of the network. So the evaluation results differ according to the dataset size.

4.4.5 Context-Aware Recommendation Systems

Context Awareness in Recommendation Systems involves the use of data that characterizes an entity in order to use them as contextual information for the computation of recommendations, wherever this is needed. Through the general overview in Recommender Systems, it became noticeable that the major effort in building recommendation systems is the well-modeled information that the Recommendation Systems for their computations. Using the latter as a principle for the development of Recommendation Systems, Context Awareness is used as a means for the

collection of the valuable information that characterizes entities within a system. Adomavicius [3] correlates the context with topics that are related with Recommender Systems, such as Data Mining, e-Commerce Personalization, Ubiquitous and mobile Context Aware Recommender Systems and Information Retrieval aiming to demonstrate the need of Context within Recommender Systems.

Context Aware Recommendation Systems (CARS) depend on how well the context is modeled and, for that reason, there are several context modeling and context representation techniques. These techniques are used in modeling the contextual information in Recommender Systems but with the modifications that each particular recommender system requires in order to use this information to generate recommendations for the domain that the system will be applied to. Recommendation Systems, in general, produce their recommendations with the use of relations between users and items by measuring the similarity distances between the users or by taking into account the ratings that the users give to items. CARS include additional information in their computations, that which is related to the users or the items but that is not visible by the users. Such information is the user's demographic data, the time of data retrieval and any other information that might influence the recommendation results. To facilitate understanding, an example that describes the contextual model of a Collaborative Filtering Recommendation System is given by [3]. In this example, a common Collaborative Filtering Recommendation System for the recommendation of movies uses the ratings of the movies given by the users. In that simplified form the Ratings of the movies are modeled in a two-dimensional space as the cross product between Users and Movies as

$$Users \times Movies \rightarrow Rating \tag{11}$$

If we consider that there is additional information that can influence the ratings such as the Theater where the movies are shown, the time that the movies were produced or viewed by the users, the social background of the user, the language that a user prefers and any other information that is related to the movies and the users, then this additional information is considered as context and transforms the rating computation from a two dimensional space into a multidimensional space where the Rating of a movie is changing into:

$$Users \times Movies \times Context \rightarrow Rating \tag{12}$$

In other words, the key in CARS is the identification of the information that can be used as context, as well as the estimation of how significant each contextual factor is for the recommendations.

Strengths

- Context Aware Recommendation systems can use or remove particular contextual factors that can improve the quality of the recommendations.
- Context filtering produces more targeted recommendations. For example recommendations that take into account contextual factors, such as location and time will be modified accordingly.

Weaknesses

- High complexity that depends on the number of contextual factors involved in the computations.

4.5 Popular Recommendation Systems

Collaborative filtering approaches are the most commonly used in existing commercial tools. In general, the correlation of items to users or the correlation between users' preferences offers qualitative and accurate recommendations. Two of the most known commercial tools that use collaborative filtering methods are the Amazon.com and the Netflix.com. The two of them consist of (probably) the largest user-items rating databases and their recommendation systems is always state of the art in the current topic. The common attribute in the collaborative filtering recommendations is the fact that the recommendations are using ratings for the computation of recommendations. The ratings vary to the rating correlation they use for example item-to-item or user-to-item correlations which are used to predict a single user's preferences. In the existing literature, several content-based recommendation systems, also exist. The application of content-based recommendations usually differs on the domain that they are developed for and based on the domain the produced recommendations is also different. In this paragraph we will depict some of the existing recommendation systems:

Amazon.com - Amazon has one of the best recommendation engines. It uses the so-called correlation matrix to generate associations between users and items and thus associations between items to items. Amazon introduced the recommendation methods "Users who bought this they also bought... " or "Users who viewed this also viewed this...". Amazon is a commercial platform that trades products of several domain contexts such as books, electronics, housing material, etc.

Netflix.com - is a movie-rating platform on which users can rate movies they have already seen and get recommendations of movies they will probably like. The main association attribute that the Netflix recommendation engine is using is the rating that each user gives for each particular movie.

Based on the user-item collaborative filtering, Netflix applies other variations of collaborative filtering algorithms aiming to the accuracy of the recommendation results.

Letizia [95] is a web browser agent which tracks the user's browsing behavior such as the links, searches or requests for help, and tries to anticipate what items may be of interest to the user. Letizia builds a personalized model consisting of keywords related to the user's interests. The keywords are collected from the meta-data that each visited HTML page contains. Using the collected keywords it then relies on implicit feedback given by the user to infer the user's preferences. A problem on Letizia is the lack of natural language capabilities that can extract grammatical and semantic information. The use of such language capabilities would improve the recommendations accuracy.

Personal WebWatcher [112] is a system that observes users in the web and suggests pages they must be interested in. It also uses the "visited documents" included in the 'visited pages' to create a user model by considering the "visited" documents as a successful suggestion and the 'not visited' ones as unsuccessful. Personal WebWatcher uses two Machine Learning classification algorithms for modeling the users' interests, the Bayesian and the k-Nearest. By the experiments described in [112] a problem that was found using the Personal WebWatcher was that the increment of the data vector size influences the accuracy of the recommendations negatively.

Syskill & Webert, [130] [129] identifies informative words from Web pages to use as Boolean features, and learns a Naive Bayesian classifier to determine the interestingness of pages. Syskill & Webert system was designed as an aiding tool that determines whether a page on a topic is interesting for the user or not.

ifWeb [6] is a content-based recommender system which supports the user in executing specific tasks without imposing specific solutions or decisions. ifWeb uses two recommendation

operations, the navigation and document search operation. For the navigation operation the system collects web documents, it analyzes and classifies them in a structured representation of the documents that have been accessed by the user. For the document search the system autonomously performs an extended navigation in the web retrieves and classifies web documents based on a specific document pointed by the user. The model used by ifWeb is constituted by a set of attribute-value pairs corresponding to the structured part of the documents such as host, size, number of images, etc. and a weighted semantic network whose nodes correspond to terms (concepts) found in documents and where arcs link terms which co-occurred in some document.

Amalthea [114] is a multi-agent ecosystem that assists users in coping with information overload in the World Wide Web and, in particular, it tries to identify potential sites of interest to the user based on the user's model. Amalthea uses machine-learning algorithms to learn the user's habits and interests, and adapt its recommendation results to the user's interests. Amalthea uses a combination of two keyword based representation of documents, which are the weighted vector representation and evolutionary mechanisms which they are used for ranking the results based on the fitness measurements of the system.

4.6 Recommendation Frameworks-Engines

The development of a recommendation engine is a highly complex process. A recommendation system depends on the type, the domain of its application and the context that each recommender system is applied to. The development of a single algorithm and its recommendations results presentation is only one part of the overall process and probably the most important one. Yet when a recommendation algorithm is designed with the aim to be applied in a commercial application is, it really needs the development of a complete recommendation framework to support all the partial development pieces that such an engine includes. From the practical point of

view, this is not necessary because of the existence of frameworks that contain all the necessary parts of a recommendation system and the developer's work is only focused on the implementation of the premised recommendation algorithm. The most common methodology that a recommendation system engineer must follow is to design a recommendation algorithm and apply it in an existing recommendation framework or engine, in order to evaluate it and modify it according to the final needs of the application. The next step is to reduce the designed algorithm and apply it into a real world application using the appropriate modifications based on the context of the application. Therefore, for the development of a recommendation system of any type, it is a good practice to find an existing recommendation framework on which the algorithm will be developed and applied, and then find the appropriate transformation method in order to apply it to the final application. In this section, some of the most-known recommendation engines and frameworks will be depicted.

Apache Mahout (http://mahout.apache.org/)– is an open source machine learning library. Mahout is mainly implements recommendation algorithms (in particular collaborative filtering algorithms), clustering and classification algorithms. Mahout is written in java and it is open and scalable so developers extend or use the existing algorithms.

Lenskit (http://lenskit.grouplens.org/) – is an open source toolkit for building, researching and studying recommendation systems. It is written in java and it is a very flexible tool for applying a new recommendation algorithm. Lenskit is a great tool for experimenting and evaluating algorithms using different evaluation strategies. One of the minors that lenskit has is the lack of documentation especially for new developers in the topic of recommendation systems.

EasyRec (http://easyrec.org/)– EasyRec is a web based recommendation engine by which any web page can add recommendations through integration to easyrec via web services. EasyRec as an open source application can be setup on a dedicated server where a web application can

be built as an extension of EasyRec or integrate the web page with EasyRec via predefined web services. EasyRec supports collaborative filtering recommendations and gives the flexibility to the developer to define and set customized recommendation filtering rules.

IwebProj [105] – IwebProj is a java based recommendation engine which offers to the developer the flexibility to develop recommendation applications. It guides the developer through already implemented applications of different kinds such as collaborative based and content-based applications. For the development of a new recommendation application IwebProj offers the corresponding API and libraries to accomplish that.

jCollibri (http://gaia.fdi.ucm.es/research/colibri/jcolibri) – jCollibri is an open source java based Case Based Reasoning framework. jCollibri provides a reference platform for the development of CBR applications. It is well structured and scalable so developers can develop CBR applications easy and fast. It also provides the CBR studio which offers to the developers standardized CBR templates which guide developers to setup a CBR application with minimum coding effort. jCollibri divides the lifecycle of a CBR application into standard coding parts for the facilitation of the programming of each particular phase and it has in its core application an extensible recommendation engine with a number of implemented recommendation algorithms.

By the selection of a recommendation algorithm or filtering method it is a good practice to also select a good recommendation framework to develop the algorithm. In the existing literature, there is a variety of recommendation algorithms and methods which are different in the quality of their recommendations, which lead to the satisfaction or un-satisfaction of the end users according to the task they have to accomplish. Therefore, the above recommendation methods support functionality for the evaluation of recommendation algorithms using specific evaluation metrics.

4.7 Evaluation for Recommendation Systems

Shani [148], separate the recommendation systems evaluation in three experimental levels. The separation of the evaluation into levels facilitates the evaluation of Recommender Systems by comparing them to each other. The evaluation levels depend on attributes and characteristics that each system has. Moreover, the type of evaluation that can be applied to recommender systems depend on the data model that a recommender system is using, the domain that the Recommendation System refers to and the type of the expected results. Thus, Shani [148] separate the evaluation of Recommendation Systems into the following three levels: off-line experiments, user studies and online evaluation. In the same work it is highlighted that for each recommendation system evaluation experiment setup it is important to follow the following basic guidelines:

- Before running an experiment a hypothesis must be formed
- When comparing candidate algorithms on certain hypothesis, all variables that are not tested will stay fixed
- When drawing conclusions from experiments, the extracted conclusions must be able to generalize beyond the immediate context of the experiments.

The evaluation of Recommendation Systems is the proof that the developed system and its recommendation algorithm produce the desirable recommendations. Thus, the evaluation process must be taken into account from the design phase of a Recommendation System to its final development. The awareness of how a Recommender System will be applied, aids to design the appropriate evaluation experiment that corresponds to what the Recommendation System is focused on. For example, there are Recommendation Systems that target the accuracy of their results and others that target the users' satisfaction. This is why the above three guidelines are important for the facilitation of the Evaluation of the Recommendation Systems.

4.7.1 Evaluation Metrics for Recommendation Systems

Recommendation systems aim at offering alternative selections related to the context of the working environment or application that is the recommendation system is applied on, as well as based on the user preferences and actions. A successful recommendation is the recommendation that got selected by the user but at the same time satisfied her/him. The satisfaction of the user can vary according to what the user wants to achieve. For example, on a movies recommendation system a user may select to see the description of several recommended movies but only one or two of them can satisfy their taste. In this example, the task is to find a movie to see. In a project management software a recommendation system can be used to produce recommendations of people for the formulation of a working group. In this case, the task is obviously different and the satisfaction factor becomes more complex, since satisfaction factor becomes a variable that can change in the future. In this section, several evaluation metrics that are traditionally used for the recommender systems evaluation, are examined.

4.7.1.1 Prediction based metrics - Accuracy metrics

Para et al. define the prediction metrics as the metrics which are used to compare which Recommender System makes fewer mistakes when inferring how a user will evaluate a proposed recommendation. The common usage of recommendation algorithms is to analyze the user's data and produce recommendations based on the similarity of an active user and others, or based on the similarity of the user's preferences and the preferences of other users. Considering that this production of recommendations, in this manner, are based on numerical computations, the recommendation results are the outcome of several equations which are applied to specific data models. In such cases, the interest in evaluating the algorithms is focused on the validity of the computations, and the closeness of the recommendation results according to the user preferences.

We can consider these results to be the most "logical results" that a recommender system can produce. Therefore, the evaluation metrics which are used are straightforward since the evaluation can be done using the same datasets as input into different similarity algorithms.

Based on [44] "*accuracy metrics measure the quality of nearness to the truth or the true value achieved by a system*" and usually accuracy is measured using the following expression.

$$accuracy = \frac{number\ of\ successful\ recommendations}{number\ of\ recommendations} \tag{13}$$

A more flexible metric used as a prediction metric is the Mean Absolute Error (MAE). MAE measures the average absolute standard deviation between each predicted rating and each user's real selections. For better understanding we depict the MAE equations as they are used in [127].

$$MAE = \sum_{i=1}^{N} \frac{|p_i - r_i|}{N} \tag{14}$$

Where p_i is the predicted rating r_i is the actual rating and N is the total number of predictions. For cases with larger deviances from the actual ratings the Means Squared Error MSE is used instead of MAE.

$$MSE = \sum_{i=1}^{N} \frac{(p_i - r_i)^2}{N} \tag{15}$$

4.7.1.2 Information Retrieval Related Metrics

Information Retrieval refers to the recommendation of items based on their content. Such recommendations might be documents which are recommended with the use of tag based filtering methods. The metrics used for the evaluation of such recommendations does not rely on the user's preference through ratings but in most of the cases the metrics which are used rely mostly on the

usefulness of the recommendation or the user's satisfaction. The most commonly used metrics are the recall and precision or the Discounted Cumulative Gain (DCG).

According to [104], precision is the fraction of recommended items which are relevant and it is expressed as

$$Precision = \frac{|relevant\ items\ recommended|}{|items\ in\ the\ list|} \tag{16}$$

Recall is defined in [104] as the fraction of relevant recommendations that are presented to the user and it is expressed as

$$Recall = \frac{|relevant\ items\ recommended|}{|relevant\ items|} \tag{17}$$

The recommender systems produce a list of recommendations which are subsets of a larger set of data. The precision is the rate of the recommended subset in relation to the overall list of items and recall is the rates the actual relevance of the recommended items in comparison to items which are already known or characterized as relevant. The necessity of knowing the relevant items in order to be able to compute the recall metric led [79] to characterize recall as useless for the evaluation of a Recommender System. Overcoming the latter characterization for recall, recall is useful in cases where the evaluation of Recommender Systems is done with the use of datasets for which the recommended items and the user preferences already exist. Therefore, in such cases recall in combination with precision can give valid results according to the accuracy of the recommended items but not so valid results regarding the user's satisfaction and the actual relevance of the recommended items for a particular user.

Discounted Cumulative Gain (DCG) [83][84] - Discounted Cumulative Gain is used for measuring the effectiveness of recommendation items based on their order of appearance. Usually, the

recommended items are presented ranked from the most to the less relevant. DCG is using the ranked recommended items by using two assumptions; firstly that the highly relevant items are more useful for the user (of any relevance level) when appearing at the top of the recommended items list and, secondly that highly relevant documents are more useful than marginally relevant documents [83].

Discounted Cumulative Gain is defined as:

$$DCG = \sum_{i}^{p} \frac{2^{rel_i - 1}}{\log_2(1+i)} \tag{18}$$

Where p is the position of the item in the ranked list and rel_i is the graded relevance of the item at the position i.

Maximum Margin Matrix Factorization (MMMF) - MMMF is an effective method used for the estimation of the rating functions by taking advantage of the collaborative effects such as rating patterns from other users which are used to estimate ratings for the current user [188]. The features extraction with the use of the MMMF method is domain specific. For instance, the extraction of features used for the recommendation of books is not the same as the features used for the recommendation of movies.

The definition of the MMMF method given by [136] is: "Given a partially observed $n \times m$ matrix Y, let us find a matrix X of the same size that provides 'best' approximation for unobserved entries of Y with respect to a particular loss function, such as sum squared loss for real-valued matrices, 0/1 loss or its surrogates such as hinge loss for binary and ordinal matrices, and so on."

4.7.1.3 Diversity, Novelty and Coverage

The evaluation of recommender systems is usually related to accuracy. Thus, most recommender systems are measuring their success using precision and recall to prove the correctness of

the results, but this is not always what a user expects from a recommender system. A well-designed recommender system with high accuracy in its results usually gives recommendations that the user is aware of. For example, in a movie recommendation system if the user likes Steven Spielberg movies, then the recommender system will include in the recommendation list all movies directed by Steven Spielberg. From the accuracy point of view, this is correct and the recommendation list can be considered successful. From the user's perspective this is not always the expected result. The user will consider a recommendation list as successful if it contains movies that they are not aware of but at the same time are satisfying her tastes [110]. Therefore, apart from accuracy, it is also very important for a recommendation system to produce useful and satisfying recommendations. Measuring usefulness and satisfaction regarding the recommendation results is a challenging research topic since both metrics are linked to the subjective opinion of each particular user and also depend on the context supporting these recommendations.

For the differentiation between accuracy and satisfaction or the accuracy and usefulness, three new metrics have been introduced. These are the Diversity, Novelty and Coverage. The novelty of a piece of information generally refers to how different is the piece with respect to "what has been previously seen", by a specific user, or by a community as a whole [31]. Diversity generally applies to a set of items, and is related to how different the items are with respect to each other. Coverage refers to the percentage of items, part of the problem domain, that a recommender system can produce for a user or a group of users. Ziegler et. al [196] propose the Intra-List Similarity for measuring the diversity of a recommendation list. Intra-List Similarity takes into account any kind of features and it is defined as:

$$ILS(P_{w_i}) = \frac{\sum_{b_k \in \vartheta P_{w_i}} \sum_{b_e \in \vartheta P_{w_i}, b_k \neq b_e} c_0(b_k, b_e)}{2} \tag{19}$$

Where $c_0(b_k, b_e)$ is the similarity between the two items b_k and b_e and P_{w_i} is the intra-list similarity of a list. The algorithm presented in the same work denote that by using the Intra-List similarity formula, if the positions of recommendations in a Top-N list P_{w_i} rearranged the P_{w_i} Intra-List recommendation is not affected.

4.8 Conclusions

By reviewing the recommendation systems types and filtering techniques we can see that research in the topic of recommendation systems is challenging and at the same time very interesting. The plurality of algorithms and recommendation filtering methods can be used for different types of recommendations based on the domain they are applied for. The selection of particular methods to apply in a recommendation system is based on the desired recommendation results. Thus, the development of a recommendation algorithm must be accompanied by the corresponding evaluation metrics. This chapter gave readers an introduction to the area, through the overview of the types of recommendation systems, the frameworks they can use for development and the metrics that can be used for recommendations evaluation. Specifically, it addressed the following main topics: presentation of the functional architecture of a recommendation system; the types of recommendation systems that exist; the filtering and similarity methods used by recommender systems; some of the most-known frameworks that can be used for the development of recommendation systems and also the most important evaluation methods and metrics that can be used for evaluating the recommendation systems. We concluded the chapter with a subjective selection of topics considered as emerging challenges, elaborating on possible future directions of recommender systems research.

Table 2: Strengths and Weaknesses of the Different Types of Recommendation Systems

RS Type	**Strengths**	**Weaknesses**
Collaborative filtering	Participation of all data in the filtering process Easy Creation and usage Addition of new data easily and incrementally	Data sparsity Scalability Grey Sheep Synonymy Shilling Attacks
Knowledge Based	The repetitive interaction with the user helps the user to get the most relevant to the initial task results Provision of recommendations without being necessary to create user profile	Cold Start Recommendation results ranking Re-use of existing solutions is not used to build user to user relations.
Trust Based	High accuracy recommendations, better than the usual collaborative filtering techniques Better results in relation to the satisfaction and coverage	Cold start The evaluation results are analogous to the size of the network and the associations between the members of the network.
Context Aware	Quality of the recommendations. Targeted recommendations.	High Complexity

Chapter 5

Creativity and Creativity Support Tools

5.1 Introduction

From the beginning of 19th century, psychology researchers were concerned with the investigation of creativity. The artistic evolution and the scientific achievements of the epoch generated an abundance of questions about creativity, such as where creativity comes from; why some people are more creative than others; whether creativity can be modelled; what the relation between creativity and intelligence is. These questions established creativity as a research topic which continues to generate extensive investigation. The psychological approach to creativity research produced theoretical creativity models. Today, researchers use these psychological creativity fundamentals to link the study of creativity with other research areas such as education and computer sciences. One of the challenges in current creativity research is to further explore the impact that technology can have in this area. It is necessary to understand creativity and its dimensions in order to construct computational models.

In order to define computational methods that can support and enhance creativity with the use of the computer, it is important to track down the creativity types, creativity levels, steps of the creative process, and human factors influencing creativity. The development of cooperative

computer systems and the facilitation in creating social networks using the Internet, in conjunction with the aforementioned, can identify creativity techniques. The creativity support tools can easily simulate these techniques.

5.2 Creativity Models

One of the aims of the current chapter is to define and to design a generic conceptual model for creativity. To achieve this task, it is necessary to examine the several creativity models from the literature and to clarify its aspects in order to identify its constructive parts.

Dewey (1910) modelled creativity through an empirical state of view as a combination of five actions: Wallas (1926) modelled the process of creativity as a four-step process *preparation, incubation, illumination* and *verification*; Guildford (1950) argued that the Wallas model could not satisfactorily describe the creativity process; Barron (1988) gave a psychological perspective to creativity and his model consisted of four-step definition of creativity.

Dewey's (1926) described one of the earliest models for creativity. He described the process of problem-solving as a combination of five-action steps [168]. In his attempt to give an emotional tone to the steps, he defines the first step as the feeling of difficulty, the second as the identification and the localization of the difficulty, third step as the consideration of the possible solutions, fourth step as the evaluation of the consequences of the found solutions, and the fifth step as the selection of a solution as the accepted one.

Later Wallas studied the writings of creative people and generated a model for the process of creativity, which consisted of four steps. The Wallas model until today,is concerned with description of the creative process. Wallas goes beyond the Dewey's sequencing to include unconscious processing and the experienced "Aha" described by many creators. The first step of the model is the preparation, the incubation, the illumination and the verification. During the preparation phase,

the creator is able to gather information about the problem in order to come up with the best ideas. During the incubation, the creator does not consciously think about the problem and goes about other activities, while at some level the mind continues to consider the problem or question. The illumination is associated with the "Aha" experience. At this point, the ideas suddenly fit together and the solution becomes clear. The verification step is used to check the practicality, effectiveness and appropriateness of the solution.

Guildford (1950) believes that creativity is in all of us and can be measured with a psychometric approach using paper and pencil tasks. Based on Guildford's work, Torrance (1974) developed the Torrance Tests of Creative Thinking. The creative tests are of two types; those that involve cognitive affective skills and those that attempt to tap into a personality syndrome [149]. A model of creativity closed to the Wallas model is defined by Barron (1988). Barron gives a psychological perspective to creativity and his model consists of a four-step definition of creativity: Conception, Gestation, Parturition and Bringing up the baby.

In most recent research outcomes related to creativity, Atman et al. [8] proposed nine design steps that are more relevant to engineering and that are being increasingly used in the commercial-product design life-cycle:

- Problem definition
- Information gathering
- Generation of ideas
- Modelling
- Feasibility Analysis
- Evaluation

- Decision
- Communication
- Implementation

Finally Schneiderman [152] proposed the eight steps of creativity:

- Searching & browsing digital libraries
- Consulting with peers & mentors
- Visualizing data & processes
- Thinking by free associations
- Exploring solutions, What if tools
- Composing artifacts & performances
- Reviewing & replaying session histories
- Disseminating results

5.3 Types of creativity

The examination of the different creativity models reveals the multidimensional character of creativity. Thus, it is hard to describe creativity with an explicit definition and several definitions can be found in the existing literature, with variations. Therefore, creativity is a sequence of steps or actions, and the type of creativity depends on the strategy or pattern which is used by people in order to achieve creativity.

Cougar [39] perceives creativity at three levels: as discovery - idea generation, as invention - development of ideas, and as innovation - transformation of ideas into services [87]. Sternberk

[169] correlates creativity to investment theory. This work points to the requirements of creativity in relation to the investment theory. Thus, creativity requires a confluence of six distinct, but interrelated resources: intellectual abilities, knowledge, styles of thinking, personality, motivation, and environment. Sternberk [169] identifies the intellectual skills as follows: a) the synthetic skill to see problems in new ways and to escape the bounds of conventional thinking, b) the analytic skill to recognize which of one's ideas are worth pursuing and which are not, and c) the practical contextual skill to know how to persuade others to acknowledge the value of one's ideas. Knowledge about a field can result in a closed and entrenched perspective of past knowledge. This can help or it can hinder creativity.

Thinking styles are decisions about how to deploy the skills available to a person. It helps a person to become a major creative thinker. It is a person's ability to think globally as well as locally, distinguishing among the important questions and the non-important ones. Personality attributes have a great importance for creative functioning. Personality attributes, according to [169], can be considered as the willingness to take sensible risks, the willingness to tolerate ambiguity and self-efficacy.

Motivation is also essential for creativity. Motivation is not something inherent in a person. Therefore, the motivation can relate to the domain and the interests of a person. When a person has the background knowledge and is interested on a topic, working in a certain area, can be considered as motivation. The final resource for creativity given by [169] is the environment. Environment which is supportive and rewarding of creative ideas can enhance creativity. The environment can offer the creative thinker the internal resources and environmental support that can help the person become more creative.

Unsworth [180] defines 4 types of creativity based on two dimensions the "driver type" and the "problem type": Expected creativity, Proactive creativity, Responsible creativity, Contributory

creativity. “The driver type” is based on the desire of people to be creative. The initialization of the wish for creativity is either a personal choice or is due to external demands. For example, the wish to be creative or to achieve a goal represents an internal driver for creativity. The “Problem type” is related to the categorization of a problem. The examination in problem-finding research elaborates to the degree to which the problem has been formulated before the creator begins the process. These two dimensions, according to [180], define the types of creativity that may exist.

- Expected creativity: Expected creativity is the creativity that is brought via an external expectation but with a self-discovered problem.
- Proactive creativity: This occurs when individuals, driven by internal motivators, actively search for problems to solve.
- Responsible creativity: The category in which the individual has the least control over problem-solving choices.
- Contributory creativity: A type of creativity that is self-determined and is based upon a clearly formulated problem.

Kirton [100] discovered that people fall into two broad families which represent creative style:

1. People who prefer to take ideas and improve them. These people are fairly cautious, practical and use standard approaches. They prefer incremental innovation. Their motto is to do things better.
2. People who prefer to find new ideas by sometimes overturning concepts. These people challenge and can be risky and abrasive. They are into “big bang” innovation. Their motto is to do things differently.

These two families are characterized as Innovators and Adaptors respectively. The Kirton KAI tool [100] determines the creativity style, not the ability of being creative. The Kirton's theory is based on a psychometric approach which classifies the people in the two teams. The categorization is based on the range of the psychometric tests scores. There is a range of continuum scores from 32 to 160 with a population average 96. People who score above 96 are innovators, while those who score less than 96 are adaptors. The degree of each depends on how far the individual is away from the average.

Innovators tend to view problems and novel stimuli in unconventional frameworks and thereby see those problems in a new way. They tend to search a broader cognitive space in problem-solving, decision-making, and other creative activities, and prefer generating multiple possibilities in those activities. Moreover, they tend to be more ready to accept and to initiate change, especially if the change arises from unexpected sources.

Adaptors have a weak tendency to incorporate a broader framework into their approach to a problem, decision, or other creative challenge. Adaptors tend to cope with novelties (new unexpected stimuli) by assuming that the outset that a relevant paradigm has the power to resolve the problem posed by the perception of such novelties. Adaptors tend to dominate mature organizations at any given time and a change like this can become a threat to them.

5.3.1 Computational Creativity Research

As already mentioned, creativity research is related to the Computers Science research. In the existing research articles, there are several references for computational creative projects. Domains of research such as the interactive music creation, collaborative media creation and exploration, creative writing and the domain of scientific visualization, include paradigms of computational research in combination with creativity.

In the domain of interactive music creation MySong [164] automatically chooses chords to accompany a vocal melody. MySong research project generates music using an alternative methodology compared to the usual artistic method. MySong application trains a Hidden Markov Model, using a music database and uses that model to select chords for new melodies. The added value of the application regarding the creativity is localized on the facilitation in creating music without any musical instrument knowledge background.

Chan and Chew [36] present an automatic style-specific accompaniment. This work proposes general quantitative methods and metrics for the evaluation and visualization of machine-generated style-specific accompaniment results. ChuckCk [36] programming language combines music with machine learning algorithms. It is a high-level audio-programming language in which the programming model promotes a strong awareness of time and concurrency and provides low-level access over time and concurrency.

The domain of creative writing [81] presents RiTa toolkit. RiTa Toolkit for Computational Literature is a suite of open source components, tutorials and examples providing support for a range of tasks related to the practice of creative writing and programmable media. Riedl [135] presents a story-generating application, which uses artificial intelligence to model the story generation as a model. The application views the story generation as a problem-solving activity in which the problem is to create an artifact that achieves particular desired effects on an audience.

Studying the research articles describing the above tools, what is noticeable is that all of the above are focusing into Artificial Intelligence techniques in order to produce creative results. None of them is approaching creativity with methods that can simulate end user's creativeness. The outcome of those tools is mainly facilitating users without the appropriate knowledge background in a specific domain to produce a creative result. For example, MySong [164] generates chords for a given vocal melody and makes it unnecessary for the users to have musical instruments

knowledge. The current research work approaches creativity as an interactive process. That means that creative result must be formulated by the end user. The role of a creativity support tool must be supportive. It is based on the factors and the concepts of creativity knowledge background of the user. Thus, creative support tool usage is based on the user's input in order to generate recommendations that will make the user become more creative when they lack ideas.

In addition to the creativity research projects analysis, we tried to examine some other of the most known commercial creativity support tools. The usual utilization of creativity support tools is the simulation of the creativity process in the way that it is defined by a creativity technique. Usually this constitutes a straight forward procedure, not for the support or the enhancement of creativity, but to make the computer behave as a substitute of paper or white-board. The most known creativity support tools like Comapping, Mind Meister, etc. substantiate the latter.

5.4 Creativity Techniques

Today, several creativity techniques are used for the development of the creativity support tools (http://www.mycoted.com). There exist more than 170 known creativity techniques; however, many of them are rarely used in existing creativity support tools. The existing techniques can be grouped according to the result which can be produced from their use. A simple grouping of the techniques is given in (http://www.mycoted.com), where techniques are used for*Problem Definition, Idea Generation, Idea Selection, Idea Implementation, Processes*.

- Problem Definition is used to clearly define a problem, redefine a problem or specify all the aspects describing a problem. Creativity techniques used for Problem Definition are Assumption Busting, Backwards Forwards Planning, Chunking, Five Ws and H and Multiple redefinition.

- Idea Generation is the process of creating ideas. Creativity techniques used for Idea Generation are Brainstorming and Taking Pictures.
- Idea Selection is the process of converting the ideas into solutions. Creativity techniques used for Idea Selection are Anonymous voting, Consensus Mapping, Idea Advocate and Sticking dots.
- Idea Implementation is the process of making the ideas reality.
- Processes are the schemes and techniques which look at the overall process from start to finish like Free Writing, Creative Problem Solving, Synectics and Thinkx.

An alternative grouping for creativity techniques is proposed in [66]. Based on the fact that one of the aspects characterizing a creativity technique is the applicability to the current usage context, in [66] it is proposed that the grouping of creativity techniques based on the usability of a technique according to the current context. Two criteria categories of the context factors are defined; hard criteria and soft criteria. The hard criteria include the physical requirements, single and group technique and emotions. Soft criteria include the web usable, time and data/technique.

5.5 Creativity Support Tools

Research in creativity led to the development of the creativity support tools as means for the enhancement of creativity beyond the classic psychometric methods. Creativity support tools can provide guidance and facilitate the creativity process for the users by monitoring the process and the produced results [145]. Creativity support tools are used to simulate the creativity techniques and to create environments which guide the users in order for them to become more creative. Throughout the years, a lot of creativity techniques have been proposed. Based on these

techniques, or a combination of them, a lot of creativity tools were proposed and developed. Considering the fact that people react differently to each creativity technique and aiming to find how creativity support tools could enhance the creativity of a person, in this work an evaluation of the existing creativity tools was made. With the evaluation of the existing tools we aim to track the characteristics supported, and find the missing points which would enhance the creativity process. The following creativity support tools were evaluated:

Comapping (http://www.comapping.com): Comapping is a web-based application that is mostly used for quick and intelligent problem-solving. It supports team collaboration for solving a problem, even if the teams are separated by geographical or time-zones. It supports real-time collaboration and asynchronous sharing. The supported techniques of comapping are: Brainstorming, Mind Mapping (left to right) and Problem Solving. It supports the importing and exporting of data from and to other creativity tools, in formats like Freemind, Mindmanager and MeadMap, as well as Microsoft office formats like rich text, presentations, excel, etc. The maps can be downloaded and stored for offline usage. The meta-model supported is the left to right mapping.

Mind Meister (http://www.mindmeister.com): Mind Meister is a web-based application that supports Mind mapping and Collaborative Brainstorming techniques. It supports importing and exporting from Freemind and Mind Manager Creativity tools. It exports data in GIF, JPG and PNG format images, in rich text and PDF formats. It also publishes password protected maps. The meta-model used in Mind Meister is traditional directional maps. It also supports real-time collaboration.

Google Docs (http://docs.google.com): Google docs are used for the creation of text documents within the web browser. The user interface resembles the typical word processor. The created documents can be shared with other users and can be collaboratively authored. The

documents can be saved on the local machine of the user, or uploaded from the local hard drive to the user's account. They can be exported as PDF, HTML, csv, ods, txt and xls formats. Through the Google docs tool the user can discuss with other team members, as well as publish the document on a given URL.

MindManager 8 (http://www.mindjet.com): MindManager 8 uses the mind-mapping creativity technique. Several features are supported by MindManager 8 thus making it an interactive visual creativity application. It interacts with Microsoft office tools like MS Word and MS Outlook, and offers the user the ability to view and edit Microsoft files directly within the tool. Web Services for the addition of data to the created maps are supported. The user can collect data from Google, Yahoo, Amazon and more. It supports embedded web browser and a database linker to Oracle, MySQL, DB2, MSSQL Server, Access, Excel, CSV and text files. It exports the map as PDF, image, web page, mpx and Mindjet Player file format. It gives the user the ability to create a group and organize a meeting with other participants. The selected users are selected from the user's contact list. MS Word files, MS Outlook Tasks and MPX files can be imported and processed. It can be used as a single-user application or as a real-time collaborative application using file exchange.

Thinkature (http://thinkature.com/): It supports Mind Mapping, Collaborative Brainstorming and Idea Organization. With Thinkature the user is able to use a white-board to write ideas. He/she is able to take notes during the creativity process. The composition of a group of users for collaboration is supported. The selected users can be found from the user's contact list. The members of a group can communicate via chat. A user can add images to the map and use various coloring and fonts formatting, drag and drop topics. The meta-model used is Mind Map structure. Importing and exporting functionality is not supported.

TRIZ (http://en.wikipedia.org/wiki/TRIZ): TRIZ is a methodology, tool set, knowledge-based and model-based approach for generating innovative ideas and solutions for problem solving. It provides tools and methods for use in problem formulation, system analysis, failure analysis, and patterns of system's evolution. The TRIZ approach for problem-solving is based on finding the best previously well-solved problem and propose the analogous solutions with the minimum harmful effects. It includes several methods and tools such as, 40 Inventive principles which are used for the solution of contradictions, contradiction Matrix, which is a database of known solutions (principles), Technical contradiction method, Physical contradiction method etc. TRIZ is using ARIZ (a program for the exposure and solution of contradictions); Su-Field Analysis (produces a structural model of the initial technological system, exposes its characteristics, and with the help of special laws, transforms the model of the problem).

5.6 Contextual Elements for Creativity

From the existing studies in the area of creativity, it is possible to identify the context of creativity. Based on the definition given for creativity, the significance of the user (or group of users), social environment and task as contextual elements in formulating a creativity process are transparent. Each one of these elements constitutes information and includes attributes that can be perceived as individual entities. The combination of these entities leads to the overall context of the creativity process.

5.6.1 Description of Contextual Elements

The following entities are considered as "primary" [157] context entities. A description for each "primary" context entity is given as follows:

User: This may include someone's competences, preferences, etc. This information defines the profile of a user. This profile can, for example, be used in creating balanced teams or in establishing the qualification to perform a task. User modeling follows an approach of standard based modeling suggested in [74] based on a combination of open specifications for learner profiles such as IEEE PAPI and IMS LIP, where the user's learning activities are recorded in performance measurements and portfolios. User modeling is mainly used to formulate a profile. The profile defines the user's role in the creative process and thereby the context in which someone functions. The context for user in the recommendation system can be the combination of the user's actions, attributes as well as their associations with the other context entities that are subsets of the contextual elements.

Social Environment: This concerns the social background of users and the social environment in which the learning takes place. This possibly includes information such as group composition, roles played in the group, etc. The generation of an idea is usually an individual process followed by knowledge transfer to other people, or knowledge received by others. The collaborative process is often used as an internal process in team-groups, companies or organizations. Therefore, the conceptualization of Social Environment demands the formulation of the appropriate associations between other entities in regards to the social background of the user. The social background of a user can be constructed based on the knowledge background, in domain specific subjects/areas, the social role, the expertise and social attributes such as the language and the location. The social role and the social background of a user constitute important context factors that influence the final recommendation of a user to be included in a team's creative session.

System: This may include information such as the software or platform used at a given time. Mixing tools and automatically tweaking system functions should result in a platform which can be used in various settings. Some settings require a more formalized environment and other settings

require an informal environment. In both cases, a creative process will be facilitated. The system context element can be defined by the following attributes:

- Connection Speed: the connection speed influences the collaborative procedure, and is important to the proposed resources type. (E.g. multimedia, real time Skype conversations, etc.)
- Type of device accessing: The user may access Creativity Support Tool from a PC, tablet, or smart phone.
- 3rd Party applications: possible plug-in requirements or possible integration with external web services

Task (also referred to as "ideation"): Information about a task including which project it concerns, the specific activity, the objective, the owner and the stakeholders of a task. Ideation is the most important element of the creative process. The overall model of contextualization of creativity aims to facilitate successful ideation. The task can be defined through the associations of the aforementioned context elements with contextual entities which are influencing the "ideation".

The enhancement of the creative process with the use of Context Awareness of the creative process converges to the Task. The analysis and identification of the creativity contextual elements guided the design of the creativity conceptual model in the form of ontology schema. The designed ontology model used for the development of a context aware recommender tool in a generic form that could be applied in creativity support tools. The designed ontology for Creativity process is depicted in figure 6.

In this attempt additional to the aforementioned contextual elements, two more context entities, "Keywords" and "Domains" were formalized.

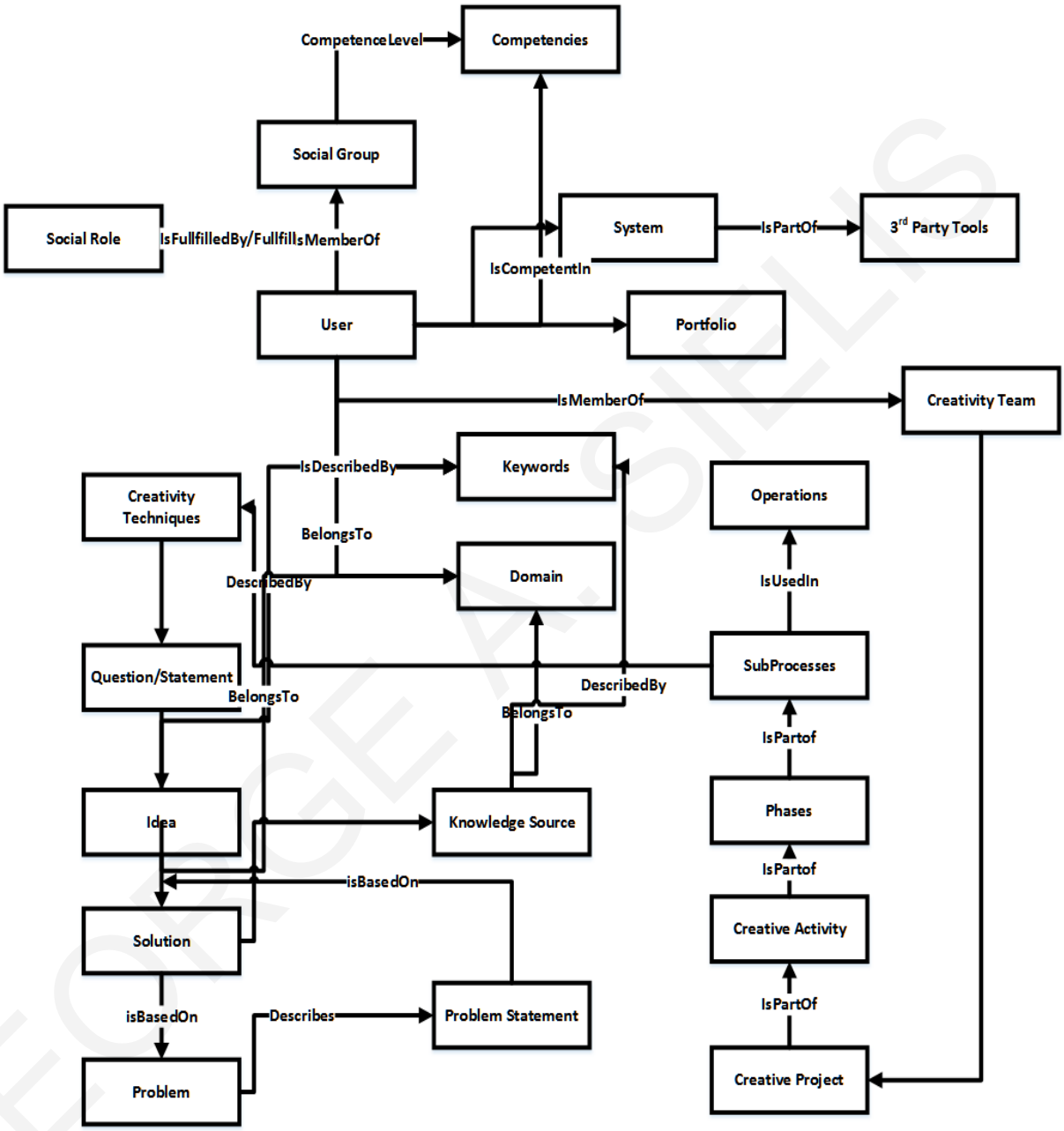

Figure 6: Creativity Contextual Model

In figure 6 the importance of these two context entities can be realized by noticing the several semantic association types they have. **Keywords:** This is an entity used to semantically describe and characterize other entities. For example, a user may be characterized as a "programmer" or "doctor", while an idea may be characterized as "SOS" or "Theoretical". **Domains:** This is an entity used to provide the field(s) to which the entity belongs. For example, a user that holds a PhD in Human Computer Interaction may have "HCI" as his domain.

5.7 A Generic Context Aware Recommender System

In addition to the creativity analysis and creativity conceptual model, the current chapter, also refers to the implementation of a generic Context Aware Recommendation system. The system is used for the generation of recommendations (e.g. people to collaborate with, relevant resources, relevant ideas, related projects etc.) during the creativity process and it was applied and tested within a creativity-support platform, named idSpace. The aim of the recommendations was the enhancement of the users in their learning ability and the collaboration with experts, access relevant resources or elaborate ideas from previous related creativity sessions.

The context awareness recommender system is designed based on the concepts of creativity as they were defined at the beginning of this chapter. The contextual elements are analyzed into several contextual entities that influence or stimulate the creativity of a user. For the design of the contextual model of the creativity process, the recommendation system is supported by the ontology schema of figure 6.

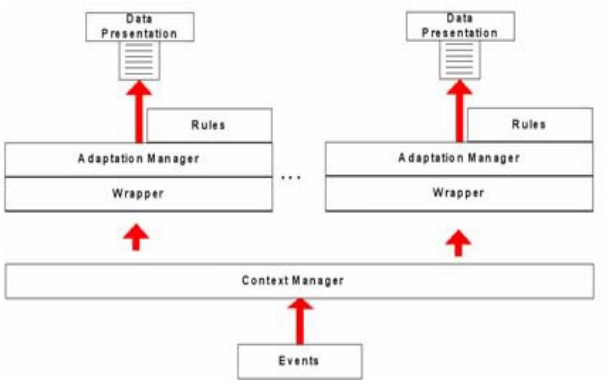

Figure 7: Context Aware Recommender System Architecture

5.7.1 System Architecture

The context awareness recommender system described in this work acts as an implicit context awareness recommender system, in the aspect that it needs input from the user to function. It provides recommendations upon user's demand or automatically; either way, the recommendations are based on data the user has inputted and which are relevant to the context of the project. The user is able to select among a number of recommendation types, which are made available according to the phase of the creativity process that is active at the given time. For example, during the phase of formatting a creativity group, the "recommendation of users" is enabled. After selecting the preferred recommendation type, the user has the option to provide additional input in the form of keywords to facilitate the recommendation process. The recommender system calculates the recommendations based on the data in the topic map concerning the project and the additional keywords provided by the user, if any.

The four recommendation types supported by the recommender system are: Recommendation of users, Recommendation of resources, Recommendation of solutions, and Recommendation of ideas.

The reasoning method used relies on the semantics. It is based, on one hand, on the keywords given as input by the user upon requesting recommendations and, on the other hand, on the keywords and domains that describe the current session. The latter data are provided by the moderator during the "project creation" phase and are considered to characterize the entire project. By using the aforementioned dataset, the Wrapper queries the topic map to find all entities of the requested type that are associated with this dataset. The wrapped data are then forwarded to the Adaptation Manager, which is responsible for making adaptation decisions according to the refined context. More specifically, in the Adaptation Manager the context information is retrieved, based on the predefined context factors, and the overall wrapped data are filtered. The filtered context data are parsed according to the adaptation method, ranked and finally presented to the end user as ranked recommendations.

5.7.2 Reasoning Method

The reasoning method described in this paragraph is an approach that the current thesis introduces. It uses a combination of technologies and methods such as the Topic Maps technology [126] and the Utility Theory [86]. Each recommendation type's software package receives all necessary input from the topic map and the user, computes the recommendations and presents them to the user. The packages, although providing recommendations of different type each, were designed based on the same structure, the same architecture and the same reasoning method. Each recommendation type has the following characteristics:

- Its recommendations are based on a set of factors. Each factor utilizes one or more ontology entities such as keywords, domains and problem statements (figure 7) to determine a set of recommendations.

- Each recommendation x is evaluated in regards to a factor i by using a Fitness Function $f_i(x)$. The Fitness Function shows how relevant a recommendation is in respect to a factor.

- To allow end-users specify the importance of each factor, according to their personal preferences, weighted values were used. One weight W_i applies to each factor i, determining the factor's relevance or, in other words, significance in the recommendation. The user can choose among three possible values: "high relevance", "medium relevance" or "low relevance".

- The Relevance Function $R(x)$ [86] is an equation of contextual factors and relevance weights that computes the relevance score of each recommendation, based on which their rank of appearance is determined:

$$R(x) = \frac{W_1 f_1(x) + W_2 f_2(x) + W_3 f_3(x) + ...W_N f_N(x)}{\sum_{i=1}^{N} W_i N} \tag{20}$$

where N: number of Factors and W: weights with values 1 (low importance), 2 (medium importance) or 3 (high importance)

As it was previously explained, each recommendation type is based on a set of factors. These factors are ontology entity types, for example "keywords", "problems", "domains", etc. Depending on the recommendation type, the recommender system examines all instances within the appropriate entity types and measures their relevance to the given problem to be solved, in order to opine which are the most relevant to be recommended. The relevance is being measured by using ontology associations. If an instance of such an entity type is highly associated with an active session, then it is highly recommended by the recommender system. The instance and the current idSpace session are highly associated if they have a number of common keywords and domains. The more common data they have, the more associated they are considered to be.

Table 3: Set of factors for each recommendation type

Recommendation type	**Factors**
Recommendation of users	Related keywords Related domains User's previous work User's competences User's user role User's social role
Recommendation of resources	Related keywords Related domains Related Problems Related Solutions Related Ideas
Recommendation of Solutions	Related Keywords Related Domains Related Problem Statements Related Problems
Recommendation of Ideas	Related Keywords Related Domains Related Problems Related Solutions Related Resources User's Competence

For example, let us consider the recommendation of users. In this type of recommendation, the factors that play a role are related keywords, related domains, user's previous work, user's competences, user's user role and user's social role. For each of these factors a fitness function exists that indicates how relevant that factor is to the overall recommendation. For the purposes of this example let us consider only the keywords and domain factors with weights 3 and 2 respectively for two different users, user A and user B. The fitness functions for these factors indicate the percentage of common keywords and domains that these users have with the current project. Thus, the more common data a user has with the project, the higher the fitness will be, resulting in a higher score in the Relevance Function $R(x)$. Let's suppose that the project has 10 keywords and 4 domains and that the number of common data user A has with the project is 4 keywords and 1 domain and user B 3 keywords and 2 domains. The fitness functions for the keywords (f_{key}) and the domains (f_{domain}) will be:

$f_{keyuserA} = 40\%\ or\ 0.4$

$f_{domainuserA} = 25\%\ or\ 0.25$

$f_{keyuserB} = 30\%\ or\ 0.3$

$f_{domainuserB} = 50\%\ or\ 0.5$

The relevance function will be:

$$User\ A:\ R(x) = \frac{3 * 0.4 + 2 * 0.25}{5 * 2} = 0.17 \tag{21}$$

$$User\ B:\ R(x) = \frac{3 * 0.3 + 2 * 0.5}{5 * 2} = 0.19 \tag{22}$$

The result is that user B is more relevant to the project than user A, and thus the recommender system will recommend user B.

5.8 Conclusions

This chapter elaborated on the concepts of Creativity through the various models for creativity as they were shaped trough time. The most important models that are relevant to the current research are the models defined by Atman [8] and Schneiderman [152]. The steps defined by the last two, consist of the basis for the design and development of the Creativity conceptual model. Moreover, in this chapter the Creativity techniques were presented and as an extension to the techniques a set of the most known Creativity Support Tools was examined in terms of the supporting and aiding mechanisms offered to the users. Through the examination of the tools, the lack of supporting mechanisms and, in particular, the lack of recommendation mechanisms was noticed. Based on Creativity analysis and the observation that the existing Creativity Support Tools are lacking of recommendation systems, this chapter also presented the design of the Creativity Ontology based on the Creativity contextual elements that were extracted from the analysis, as well as the development of a generic purpose Context Aware Recommendation System.

Chapter 6

Framing the problem - A Survey in Software Design Process and Tools

6.1 Introduction

Software engineering projects are complex processes which, apart from the implementation of a project, require proper planning and management of a complete project life-cycle. The cycle of planning and implementation of such software tools became one of the most important Software Engineering research subjects, where the Software Design process phases and models were individually studied. Boehm et al. [22] present the several Software Engineering process models from the 1950's to twentieth century models such as *SAGE, Software Crafting, Waterfall* models and *Agile* methods. Ideally, the proposed models should be the combined result of research and actual experience; however in reality this does not always happen. This is due to many reasons, the study of which has led to the survey that this chapter presents. In particular, the objective of this work is the collection of information regarding possible needs that Software Engineers may have through the usage of Software Engineering Design Tools. It will be possible to examine possible

enhancements for these tools, especially in terms of the Software Engineering life-cycle processes for Design, Management and Execution of Software Engineering projects.

The survey aimed towards the collection of opinions from experts in Software Engineering regarding the existing Software Engineering Design tools that they use. The survey also aimed towards finding possible flaws of the tools regarding the Collaborative Design. The work presented in this chapter was motivated by the notion that applying Context Aware Recommendations to the tools that professional Software Engineering Designers use, would enhance the overall process of preparation and execution of a Software Design project. Therefore, the survey objective was to track the experiences regarding the designers' needs, as well as additions they would like to have in the existing tools; for example, Social Networking features or Context Aware Recommendations support.

.

6.2 Software Engineering Design Tools - Overview

This section examines some of the most known Software Engineering Design Tools. The examined tools were evaluated in regards to their supported modules, the social attributes they have and the types of recommendations they possibly support. In this overview, a list of commercial and open source tools were examined, the most of which were tools that the experts who participated in the survey stated that they use.

Eclipse - Eclipse is an Open Source IDE which provides a universal complete toolset for development. Several of the tools/modules included in eclipse are used specifically for Software Architecture Design projects. One of the most complete Eclipse sub-projects is the Eclipse Modeler which promotes the evolution and promotion of model-based development technologies, by the provision of a unified set of modeling frameworks, tooling, and standards implementations. It

supports team-working tools, giving the flexibility to a team of developers to share their code and automatically commit code in version-control systems. Eclipse can be used for Software Architecture Design through the sub-projects and the plugins that can be installed to it, but is mostly used as a development IDE. It supports Code Generation for Java, C++ and C programming languages and it can use additional modeling tools as separate eclipse installation modules. Such examples are UML tools, Object Constraint Language (OCL) expression tools and Eclipse Modeling Framework (EMF) - based implementations. Eclipse is a desktop application which can connect to code repositories for sharing and version-controlled code projects.

Microsoft Visio - is a 2D drawing tool on which the end user is able to draw diagrams. It supports drawing of shapes for several types of diagrams such as flowcharts and ER diagrams. It does not support any type of recommendations or social modules. It is a commercial desktop application which offers the flexibility to the user to extract the diagrams in several format types such as xml files and images.

Enterprise Architect - A UML-based software which can dynamically simulate software behavior, state models, confirm process design and specify triggers, events and constraints. Enterprise Architect offers the tools to create and debug embedded solutions or build domain specific custom solutions. It is based on open standards like UML, BPMN and SysML, and supports enterprise architecture frameworks like TOGAF and UPDM. It offers team based repositories and version control tools.

MagicDraw - MagicDraw supports the UML 2 meta-model, the latest XMI standard for data storage and the most popular programming languages for implementation such as Java, C++, Csharp, CL (MSIL) and CORBA IDL programming languages), database schema modeling, DDL generation and reverse engineering facilities. MagicDraw offers tools which can be used to deploy a Software Development Life Cycle (SDLC) environment that best suits the needs of a business. It

offers an Open API giving the flexibility to developers to integrate MagicDraw with applications that work together. It can also be integrated with other products such as IDEs, requirements design tools, testing tools, estimation tools, Multi-valued Decision Diagrams, databases, and others. MagicDraw offers a teamwork server so developers can simultaneously work on the same model.

Table 4: Software Engineering Design Tools Attributes

Software	**C/ OS**	**Supported Design functionality**	**Supported recommendations**	**Supported Social Attributes**	**D/ W**
Eclipse	OS	UML tools, EMF based projects, OCL expressions	Code Generation for Java, C++ and C	Code Sharing between a team	D
Microsoft Visio	C	Diagram design with the use of shapes	None	None	D
Enterprise Architect	C	UML, BPMN and SysML, Enterprise architecture frameworks like TOGAF and UPDM	None	Team based repositories and version control tools	D
Enterprise Architect	C	UML, BPMN and SysML, Enterprise architecture frameworks like TOGAF and UPDM	None	Team based repositories and version control tools	D

Magic Draw	C	UML 2 metamodel, the latest XMI standard for data storage and programming languages development, database schema modeling, DDL generation and reverse engineering facilities	Data Definition Language (DDL) generation	Server for real-time modeling by team	D
ArgoUML	OS	UML for Class diagrams, Statechart diagrams, Activity diagram, Use Case diagrams, Collaboration diagrams, Deployment diagrams and Sequence diagrams	Design Critics	None	D, W
IBM rational	C	UML based on role, model, life-cycle phase, and current task, BPMN2, Model reporting, etc.	None	Hosted by IBM Cloud	D, W

Yaoqiang BPMN	C	BPMN 2.0 diagrams	Spell checks, automatically generates BPMN2.0 diagram interchange information	None	D
ER Studio	C	UML 2.0 diagrams	Pre-defined patterns and templates for new projects	None	D

StarUML - StarUML is an open source project that aims to develop fast, flexible, extensible, featureful, and freely-available UML/MDA platform running on Win32 platform. StarUML project is used for the development of software models. StarUML is mostly written in Delphi; however it supports multi-lingual projects without being tied to specific programming language. Any programming languages can be used to develop StarUML such as C/C++, Java, Visual Basic, Delphi, JScript, VBScript, C# and VB.NET.

ArgoUML - ArgoUML is an Open Source UML modeling tool. It can be used to design Class diagrams, Statechart diagrams, Activity diagram (including Swimlanes), Use Case diagrams, Collaboration diagrams, Deployment diagrams (includes Object and Component diagram in one) and Sequence diagrams. It offers XMI and OCL support, reverse engineering module, as well as design critics. Design critics are generated by embedded agents. The agents analyze the design as the designer is working, and suggest possible improvements such as syntax errors, reminders to return to parts of the design that need finishing, style guidelines and advices of expert designers.

IBM rational - General modeling with the industry's most robust support for UML 2.2 and a configurable modeling environment that can be tailored to expose "just enough" UML based on

role, model, life-cycle phase, and current task. It supports BPMN2, including interchange with products supporting BPMN2 standards compliance, like WebSphere Business Modeler. It supports the following: Requirements integration, with end-to-end traceability and impact analysis; Model analysis and metrics with the use of extensible query and analysis frameworks; Model reporting with BIRT; Rich SCM integrations; model compare-merge, and extensible team modeling framework; rapid development of custom software factories; fast creation of UML-based, domain-specific modeling languages tailored to a problem and solution domains; use of graphical mapping tools for quick development of model-to-model transformations; and finally, use of exemplar-driven pattern capture tools combined with a rich JET authoring environment to quickly develop model-to-text/code transformations. It can be integrated with several products that belong to the IBM Rational products group.

A summary of the attributes and the tools' supported functionalities are depicted in Table 4. Analyzing the attributes and the characteristics of the examined Software Design tools, the lack of aiding mechanisms, such as Recommendations that can help engineers in fulfilling Software Engineering tasks, and the lack of social collaborative functionality were revealed. A usual argument that is commonly used to excuse the lack of these attributes, is that since the existing tools do not support them then most probably, there is no need for them to exist. However, the survey results proved the exact opposite result. The rest of this chapter describes the overall survey design and presents its outcomes that lead to the conclusion that aiding functionalities and features are necessary for Software Design tools.

6.3 Survey Design and Analysis

The survey was designed for a particular target group, the Software Engineers. Software Engineering is an applied by professionals topic, with a strong relation between industry and

academic research. Therefore, the collection of data given by experts in Software Engineering is of paramount importance. The survey design and the most important outcomes of the survey are presented in the following sections.

6.3.1 Survey Design

The survey was designed as a web-based survey, while the overall design was structured based on the following four axles: *1. Software Design Experience, 2. Project management, 3. Software Design Tools* and *4. Development of new Ideas*. The first axle contained a group of questions for the collection of data that demonstrate the participants' experience in Software Engineering. The second axle contained a group of questions related to the preparation and management of Software Design projects based on the participants' working experiences. The third group of questions refers to the Software Design Tools that the participants currently use and the tools that they were aware of (Table 4). It contains questions that can lead to conclusions regarding the modules they have in supporting the process of design, composition of group, the supporting resources and the social attributes they support or not. In this question group, participants were also asked questions related to their perception on how the inclusion of specific additional modules, related to the objective of this work, such as Context Aware Recommendations and Social functionality. The fourth group of questions contained questions regarding the flexibility the participants have in developing or promoting new ideas within their working environment. The complete question-set that was given to the participants can be found in appendices A.1,A.2,A.3 and A.4

This survey was sent via web to professionals from the industry and experienced Computer Science Researchers. In total, 28 professionals from both industry and academia responded to the survey, from which 14 belong to the private sector (11 International Private Corporate - NCR Corporation, 1 Small Medium Enterprise - KResearch Ltd. and 2 Self-employed), 1 belongs to

the public Sector and 13 belong to the Research/Academic sector. The age range of the participants was 26 years old to 55 years old. The mean age between all participants, was 35.3 years old. In particular, the survey was performed by 12 Software Developers, 1 Software Architect, 3 Software Development Team Leaders, 2 Project Managers, 1 Manager of Software Development Department, 5 Research Associates, 3 Academic Professors and 1 System Administrator.

In the rest of this section the responses received per question-group are based on a selection of the most important ones. For each question-group the most valuable statistical results are elaborated.

6.3.2 Software Architecture Design Experience

For the confirmation of the participants' experience in the topic of Software Design, the first part of the survey asked the participants to provide their demographic data, their affiliation and their role in the organization they are working.

As expected, the analysis of the demographic data in relation to the participants' experience showed that their age is linear to the experience they have in Software Design projects. The experience threshold age seems to be the age of 35. Participants with age less than 35 seem to have 1 or 2 years of experience, while users with age greater than 35 have more than 4 years of experience in the topic. This is also analogous to the active participation of participants in Software Architecture Design projects. The participants who stated that they have more than 4 years of experience in the topic, participated actively in fewer than 5 projects while the rest participated in more projects. The younger participants who stated that they have 1 to 2 years of experience in the topic, actively participated in 1 to 3 projects. However, the participation in projects, in relation to the experience appeared as not linear or analogous. This probably happens because each project has unknown factors that cannot be taken into account for the comparisons

and produce safe conclusions. Example of such factors could be the unknown time duration of each project. In total, 60% of the participants have more than 4 years of experience and 80% of the participants have more than 3 years of experience, which gives an added value to the participants' opinions.

6.3.3 Project Management

The project Management group of questions contained 10 questions which aimed towards the collection of information regarding the management of project-handling. The project handling includes the project setup in regards to the resource allocation, the assistance and the factors that each participant takes into account for the team formulation before the beginning of a new project.

The main task of the current question-group was the examination of how a project is designed and structured, in real-working environments in comparison to the theoretical project management models. Additionally, identification of which factors are taken into account for the formulation of a working team and the collection of the supporting material, can facilitate the progression of a new Software Design project. The participants were also asked how they would perceive the existence of context-related recommendations during the process of design and preparation of a new project. Moreover, participants were asked to give their opinions as to what would they consider as the most important recommendations that would facilitate the design process.

Participants were also asked how many projects they usually run at the same time. As expected the professionals' responses showed their huge effort of work. The 75% of the participants declared that they run 2-4 projects the same time and 17.86% of them run 5 and more projects at the same time.

The participants were asked to prioritize the team members' attributes which are taken into account in order to assign them a new Software Design project. They had 5 options to prioritize:

Experience, Knowledge Background, Social Background, Development competencies and Teamwork mentality. The prioritized results are as follows: As first priority attribute 35.71% of the participants selected “Experience” while the 32.14% selected the “Knowledge Background”. It is noticeable that 21.43% believe that “Development competencies” is the most important attribute and must be the first in the list. A smaller percentage 10.71% believe that team work mentality is the most important one. None of the participants believe that Social Background is the most important attribute. More specifically “Social Background” classified by the 89.29% as the least important attribute, 3.57% as the fourth attribute in the list 3.57% as third and 3.57% as second. In a very similar way, 35.71% of the participants selected “Experience” as a second attribute in the list and 25% the “Knowledge Background” attribute. 21.43% of the participants believe that the second most important attribute is the “Development competencies” and 14.2% the “Teamwork mentality”.

Most of the participants believe that “Development competencies” must be the third in priority attribute with the percentage of 39.29%, 25% classify “Knowledge Background” as the third most significant attribute and 14.29% the “Experience”. As fourth in the list, the 53.57% of the participants classify the “Teamwork mentality” while 17.86% and 14.29% the “Knowledge Background” and “Development competencies” responsively. 10.71% classified “Experience” as fourth in the list. Having the above results the priority list based on the complete set of answers given is the following:

1. Experience
2. Knowledge Background
3. Development Competencies
4. Teamwork mentality

5. Social Background

The above-mentioned priority list raises several discussion issues. Firstly, it is obvious that the opinions vary. A surprisingly huge percentage of participants classified Social Background as the less important attribute. Indeed Experience can be considered as the most important factor that someone takes into account before deciding to include someone in a project. But, what happens if a person is highly experienced but cannot communicate because of language (Social background attributes) or cannot accept other colleagues opinions (teamwork mentality attributes)? In addition to these, some of the listed attributes are encapsulated. Experience and Development Competencies prerequisite the Knowledge Background, Development Competencies are improved through Experience.

6.3.4 Further Analysis

Aiming to correlate the formal Software Engineering Life-cycle models with the real activities taken by professionals during the setup of a project, participants were asked to describe three steps they take for the preparation of the project. All participants completed the three steps questions; however answers differed. The steps they described are all correct but the sequence was different. Based on that, it is clear that the participants are aware of the process life-cycle steps but are not following a formal process for the preparation of a new project.

Based on this, we can conclude that even if the experts in Software Architecture Design are aware of the steps for a Software Engineering life-cycle process, they do not follow the correct sequence or the "formal" sequence of steps.This probably happens because professionals do not give attention in following typical procedures in setting up a new project. They rather pay attention in executing tasks fast. In most of the cases this can cause problems in the process of the execution

because of errors. The typical procedure can be considered time-consuming, but at the same time it is necessary for the correct design and setup of a new project.

The ovarall analysis of the results used the Pearson's correlation method [18] in finding the most significant relations between the answers given for particular questions. A sample of the correlated questions showing the correlation coefficient (r) and the significance value p is shown on Table 5.

Using the full set of correlated questions and the most statistically significant ones ($r < 0.05$) the following outcomes were reached:

Collaboration during the design process

- The participants usually collaborate with other people during the Software Design process but in most cases they prefer to collaborate with people they know. At the same time, the plurality of them believe that it is very important for them to get recommendations of people to collaborate with.

- During the Software Design process they involve people or teams from abroad, but they usually work with local teams. (Lack of real time collaboration tools)

- The characteristics taken into account for the selection of colleagues to collaborate with in priority order are: 1. Experience, 2. knowledge Background, 3. Development competencies, 4. Teamwork Mentality and 5. Social Background (Contextual factors for the recommendation of people to collaborate with).

Access to other projects to use them as reference

- Most of the participants denoted that it is difficult for them to find related projects in order to use them as references. Others mentioned that it is easy for them to find related projects.

Table 5: Pearson' s correlations between questions -Subset table

Question 1	**Question 2**	**p**	**r**
Years of experience	In how many projects had active role	0.632	.000
Position in the company	Use similar projects recommendations	0.470	.042
Type of organization	How easy id to access related projects	0.378	.047
People who worked in projects remotely	Collaboration with other experts	0.396	.037
How easy is to find other people	Do you always know the people involved	0.683	.000
Recommendation of web url's	Recommendation of Similar projects	0.474	.004

When the participants were asked how they find related projects the answers were: web (Google), asking other people, email to professional teams.

- When participants were asked if they want to have access to other projects, almost all of them answered positively. Contrary to that, almost all of them hesitated to share their

projects in public networks, but are positive to share their projects in a closed network (i.e. Intranet)

Software Engineering project setup

- The participants were asked to write in free text three steps of setting up a new Software Design project. The conclusion that came out is that there is a need to formalize the steps following a project lifecycle model. The answers given can be considered as correct but vary in regards to the sequence of steps as they were given by each individual.

Ease of developing a new idea as new project

- More than 50% of the participants answered that it is difficult for them to develop a new idea as a new project or to promote a new idea within their organization.

Helpful resources

- The participants believe that the following additional resources would be helpful during the Software Design process: Documentation related to the project (75%), Similar/Related projects (67,86%), Experts in the area of Software Design (60,71%), Recommended web resources (32,14%), Images (17,86%) and Other (7,14%).

6.4 Discussion

The results of the survey presented in this work lead to the conclusions that there the tools that are currently used by Software Engineering experts lack the aiding and assisting functionality such as Context Aware Recommendations and Social related functionality. From the analysis of the results it is obvious that the Software Engineering professionals support the development and integration with the existing tools of such functionalities. The usage of Context Aware Recommendation tools in Software Design software can become a great enhancement for the Software

Design projects setup, design and management by taking into account the prioritized results for the recommendation types that the participants mentioned. Additionally, the accessibility and availability of executed projects raised social concerns that can be investigated in more depth in the future. This should take into account responses like “I would like to access other projects” but “I would not like others to access my projects”. The outcomes of this survey revealed challenges worth investigating for future research in the topic of Software Engineering, and in particular in the Software Design assisting software tools.

Chapter 7

Design Patterns Ontology Model - Design, Analysis, Implementation

7.1 Introduction

From the analysis for Creativity in Chapter 5 it is obvious that creativity process is not a straight-forward process. On the contrary, it is characterized by dimensions that may vary depending on the domain that it is applied to. Creativity tools integrated with context aware recommender systems can positively influence the creativity process. Context aware recommendations such as recommendations of users for collaboration or resources related to the context of a creative project, but also other recommendations related to particular creativity phases, can facilitate the creativity process. Therefore, the creativity process can be approached in two ways: first the development of a generic Creativity Support Tool for the development of professional networks targeted in solving problems and producing innovative ideas; second, the development of a context aware recommendation algorithm, which will have the flexibility to analyze the most important contextual factors for each phase of the creativity process, and therefore, produce useful recommendations that foster the professionals' creative ability. Nevertheless, wideness of creativity topic, makes the design of a research plan very complex and in most of the cases difficult to implement. For example, a creativity process may be an art session, a book conversation or the development of a web site.

Using the survey analysis that was presented in chapter 6, a decision to frame the problem into a particular creativity-related problem was taken. This topic is the Software Design (SD) process, on which users must select the most suitable Design Pattern in order to describe a Software Design problem in a High Level Software diagram model. Software Design is by definition a creative process while it is executed in phases by individuals or groups of experts. More specifically, the problem of designing High Level Software Models by using Design Patterns. It is approached as part of the Software Engineering Education and Training point of view, were the target group is limited to students who can creatively learn the Design Patterns through practice.

This chapter focuses on the design of an ontology model that represents the process of High Level Software Diagrams design, with the use of Design Patterns. The objective of the current chapter is met through the extension of the already defined Creativity ontology model, with the addition of new entities and relations for its adjustment to the Software Design process. The new ontology model represents the Software Design process and also contains the Design Patterns attributes. The new design, aims to the usage of the model by recommendation algorithms for providing recommendations of the most relevant Design Patterns, based on the user's input. Using the Design Patterns recommendations, users will be able to learn, discover and apply the Design patterns into diagrammatic design models.

7.2 Modeling Software Design Model as Creativity process

Modelling the Software Design process as Creativity process enhanced by Context Aware Recommendations consists of a combination of the following two models: *1. The General model for Creativity* and *2. The Software Design model*, which demonstrates the Software Design life-cycle. For the first model a comprehensive analysis was done in Chapter 5. Therefore, the rest of this section is focused on the design of the Software Design Model and the representation of

Design Patterns. Moreover, it describes how the designed model is integrated to the Creativity model that was defined in chapter 5.

7.2.1 The Software Design Model

In the existing literature, several software development models can be found such as the Waterfall [12], ETVX [9], Prototype, Spiral [182], V-model [187], Agile methodologies such as Scrum [138] and Unified Process Model. Following the comparison between models that was presented in [115], it is supported that the most commonly used models are the Waterfall and the Spiral while Agile models are also increasingly applied during the last years. The difference between models such as Waterfall and Spiral, in comparison to Agile models is mainly observed on the sequence of application of their phases during a Software Design process life-cycle. On the one hand, Waterfall and Spiral models use the strict sequential completion of phases, setting as a necessary condition to proceed into a next phase, the completion of the previous one. In Agile approaches there is more freedom in the completion of tasks, which belong to the several phases of a Software Design life-cycle. For modelling the Software Design process, the Waterfall model was selected as a guiding model. It was used for the specification of the process phases and the semantic representation for the Software Design life-cycle. The waterfall process phases are listed as follows:

- Requirements specification
- Design
- Construction (implementation or coding)
- Integration
- Testing and debugging (validation)

- Installation
- Maintenance

Each particular phase of the Software Design model can be considered as an individual creative process, that can be executed by individuals or through collaboration of peers. Considering each particular phase as individual creative process, makes the definition of context for each phase possible. Following the contextual factors as they were defined in Creativity model (Chapter 5), Table 6 is formulated.

Using the Creativity contextual entities and the table 6, it is possible to analyze in a more comprehensive way a new Contextual model through the revision of the general Creativity model enhanced with the Software Design attributes (entities). The initial Contextual model for Creativity is depicted in a visualized form in Appendix C.1.

Table 6: Software Design phases as Creativity processes

Phase	**Users**	**Types of ideas expected**	**Resource Material**	**Tools**
Requirements specification	Individual or Group	Text based ideas. New ideas or modify ideas entered by other team members.	Resources entered by group members. Requirements from similar projects. Links, Videos, Images	Resource repository, Chat, Recommendations of users, Recommendations of related projects, Recommendations of related requirements. Recommendation of actions.

Design	Individual or Group	Schematic design with comments e.g. UML diagrams	Images, Tutorials, White papers, Design Patterns	Recommendations of resources, recommendation of experts, Drawing tool, UML editor, canvas designer etc.
Coding	Individual or Group	Classes and methods of implementation, coding Design Patterns		Programming language IDEs.
Testing	Individual or Group	Testing methodologies, testing scenarios, unit-tests		
Maintenance	Individual or Group	Maintenance actions		

Figure 8 depicts an example of how Creativity model interprets between the phases of Software Design model.

Following the intersection of the creativity model between each phase, the conceptual model that combines the above-mentioned partial models was designed. Through the conceptual model and using the following attributes and entities relations the ontology model was designed. Table 7 shows the triples that were used for the semantic representation of the Software Design - Creativity Design model. The extension of the base Creativity Model to the final Ontology

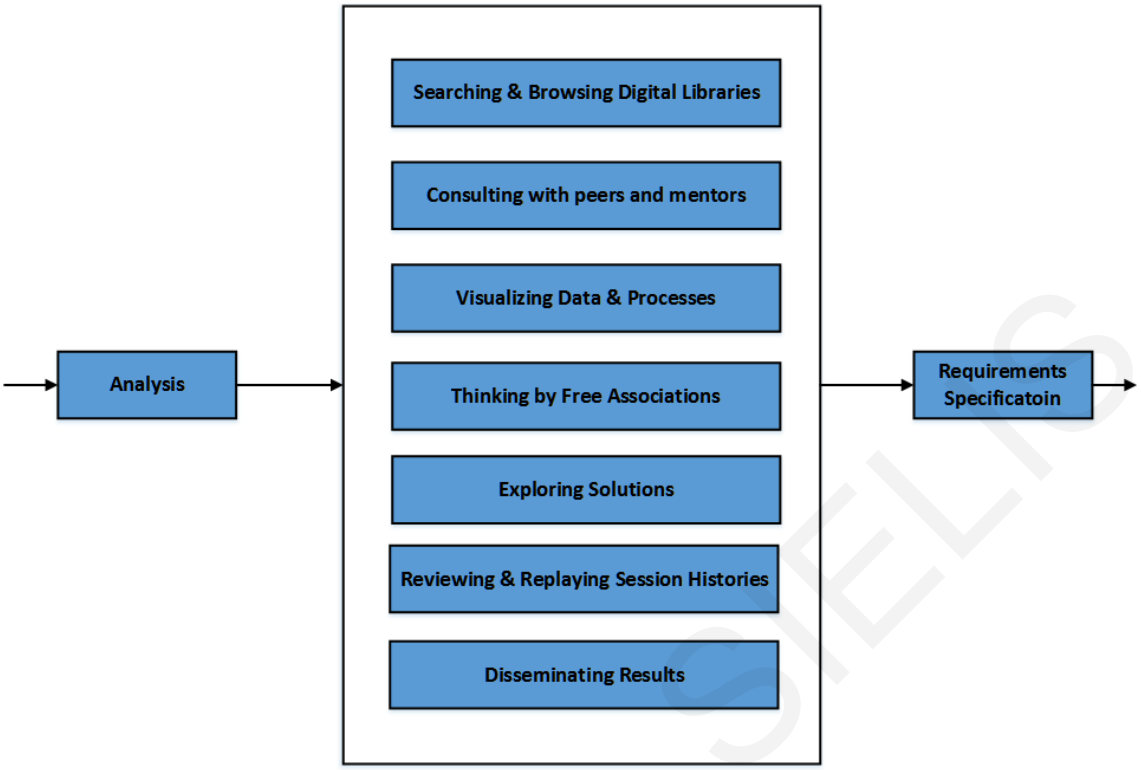

Figure 8: Sample of the intgrated model between Creativity and Software Design

model that includes the Software Engineering entities and relations are depicted through visualizations in Appendices C.1 and C.2. Appendix C.3 shows the overall ontology structure through the classes/entities hierarchy and the the data and object properties while in appendix C.4 the final visualization of the ontology model including the data and object property relations is shown.

Table 7: Semantic Analysis of the Creativity-Software Design model (triples)

Domain	**Property**	**range**
ProjectCreator	createProject	Project
Project	projectCreatedBy	ProjectCerator
ProjectGroupMember	isMemberOf	ProjectGroup

Project	isDividedInto	Phases
RequirementsDefinition	isPhaseOf	Project
Design	isPhaseOf	Project
Implementation	isPhaseOf	Project
Project	IsConsideredAs	CreativityProject
RequirementsDefinition	IsConsideredAs	CreativityProject
Design	IsConsideredAs	CreativityProject
CreativityProject	isRelatedTo	DigitalResources
CreativityProject	ContainIdeas	Ideas
Ideas	areContainedBy	GroupMembers
GroupMembers	isExecutedBy	GroupMembers
CreativityProject	createIdeas	Ideas
Ideas	areCreatedby	GroupMembers
CreativityProject	isExecutedBy	GroupMembers
GroupMembers	Execute	CreativityProject
Group	hasmembers	GroupMembers
GroupMembers	areMembersof	Group
Project	hasTitle	String
Project	hasDescription	String
Project	belongsToTopic	String
ProblemStatement	hasStatement	String
Project	isDigitalResource	DigitalResource

Idea	isDigitalResource	DigitalResource
Book	isDigitalResource	DigitalResource
ResearchPaper	isDigitalResource	DigitalResource
Idea	hasContent	String
CreativityTechnique	hasName	String
CreativityProject	usesTechnique	creativityTechnique
Project	hasProjectGroup	Group
Group	isGroupOf	Project
Requirements	hasrequirement	String
Design	hasDesignParts	String
Testing	hasTestingComments	String

Definition and design of the Software Design Ontology Model as an extension of the Creativity Contextual model, in combination with the definition of the Software Design phases analysis, and the survey results that were presented in previous chapter, led to the decision to approach the following examination: *how context aware recommendations affect the particular creativity process in software design and particular the Design phase*. The approach had two tasks to accomplish: First, the development of a Design module where users would be able to design high level Software Design models, and second, the recommendation of Design patterns as a separate model.

7.3 Design Patterns Ontology Model

For the definition of the Design Patterns Ontology Model, Design Patterns were analyzed into contextual elements that characterize and define them. A helpful resource that was used for this,

was the GoF description template, used in [59]. In this template, each Design Pattern is described based on its *Intent, Consequences, Implementation, Known uses, Motivation* and *Collaborators*. For each Design Pattern additional features were used as contextual elements, such as the image of a Design Pattern, a generic descriptive UML diagram for the pattern, urls with information for the pattern and documents related to a pattern. Combining the Design Patterns ontology model with the Creativity model, made also possible the collection of additional contextual information, such as, the problem/project that is used for, the user who used it, the type of application that the model was designed for etc.

Definition of the Design Patterns characteristics and attributes was the first step for definition of the model's concepts and contextual information. For the Design Patterns semantic representation, the specification of the semantic relations between the entities should also be defined. The triples, used for the model definition and representation are shown in table 7. A sample of the Design Patterns model is shown in figure 9.

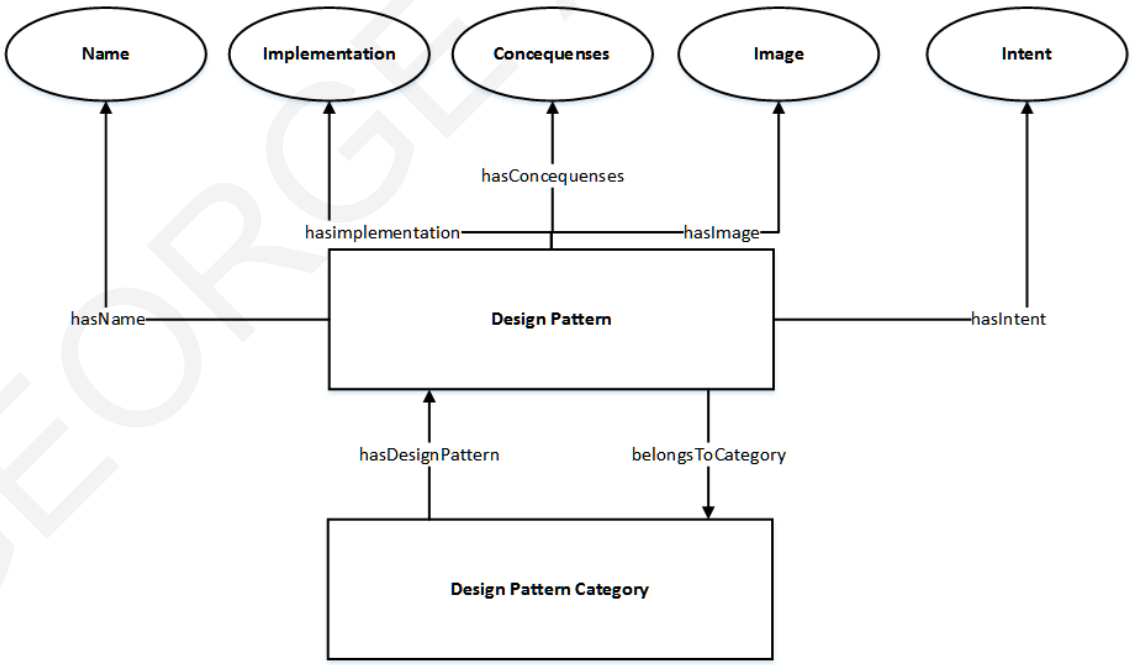

Figure 9: Design Patterns Ontology Model Based on the GoF templating

Moreover, the design patterns are not limited to the GoF [59], but there are several categories of Design Patterns which are used for specific application types. The application types are also defining the Architectural patterns that are available, and by the selection of an Architectural Pattern, Design Patterns are filtered based on the selected Architecture.

7.3.1 Categorization of Design Patterns

Last years, technology trends change rapidly and the development methodologies are also influenced by these changes. In general, there is a notion that any coding method, which is used more than three times in the same way, can be considered as Design Pattern. Increment of technologies, and development of new applications of different types, makes identification of coding methodologies as Design Patterns, difficult to handle and hard to follow. In particular, it is very difficult for young engineers who do not have the necessary experience in coding and more specifically, usage of Design Patterns. Taking this into consideration, categorization of Design Patterns becomes necessary. Filtering the patterns based on Architectural pattern or the type of application they can be applied to, becomes a useful tool for Software Developers. Until recently, the only set of patterns that was used were the GoF Design Patterns [59], which are categorized in this thesis, as "Operational" Design Patterns. Nowadays more design patterns exist, and they are exponentially increased since coding Patterns for smart phone applications, web services, web applications, new data storage methods (e.g. NoSQL) and also UI Design Patterns exist.

Taking into account the several Architectural Patterns that exist, associations between Design Patterns and Architectural patterns or associations between Design Patterns and design principles have been identified.

The Architecture types are defined based on the application type they are usually applied for. For example, this work defines four Architecture types based on the well-known application

types: *Desktop, Mobile, Service Oriented* and *Web Based applications*. Each Architecture type can have one or more architectures associated to corresponding application type. For example, for web applications the Architectural patterns of MVC, MVP, MVVM can be applied, for Mobile application the MVVM pattern is commonly used, and for web services the Model-Controller (MC) may be used. The architectural patterns can be related with more than one Architecture types. The differentiation of Architectural patterns based on the associated application types, also influences the availability of Design Patterns per Architectural pattern or application type. For example, UI Design patterns are not used for Web Service implementations but they can be used for mobile applications that use web services.

Therefore, the designed model uses five categories of Design Patterns *Antipatterns, Classic (GoF), Enterprise Application Architecture Patterns, MVC* and *User Interface* Design Patterns.

For each category there are subcategories of Design Patterns such as:

- Classic (GoF) ***hasSubcategories*** Behavioral, Creational, and Stractural
- Antipatterns ***hasSubCategories*** Software Design Antipatterns
- Enterprise Application Architecture Patterns ***hasSubcategories*** Data Source Architectural Patterns, Object Relational Behavioural Patterns, Object Relational Metadata Mapping Patterns, Object Relational Structural Patterns
- User Interface ***hasSubcategories*** Yahoo UI and similar repositories of UI Design Patterns

7.4 Implementation of the model - Used Semantic Web Tools

Modeling the Design Patterns and in general the Software Design process as a continuation of the Creativity process in the form of ontologies, was a selection made after the analysis of the different types of modeling the context that was analyzed in previous chapter. Modeling with the

use of semantics and in particular with the use of ontologies offer a big advantage, which is the retrieval of data from several, more than one, data sources, with different data structures and it becomes possible to map them into a single structure model which is the ontology model. This makes easier the finding and filtering of data from several sources with semantic web based query language and the use of semantics. That means that by adding new data sources into our model will only need the mapping of a given schema to the corresponding entities of our model and it will not be necessary to adjust the used queries to filter and retrieve the data that are passed to the implemented algorithms.

Therefore, a set of Semantic web technologies and frameworks were used for the design and implementation of the models presented in this chapter. Firstly, for the design of the ontology model Protege Semantic web tool was used. Protege is an ontology builder with a rich tool-set with visual and text based representation of the ontology models. With Protege an ontology model can be built by defining the conceptual entities (classes), the object attributes, functional attributes and the semantic relations between them. Protege exports the designed models in OWL (Ontology Web Language), which then can be parsed and used based on the needs of any application that uses the model (sample is given in listing 7.1). Second, the Jena Framework was used. Jena is an open source Semantic Web framework written in Java language. Jena framework is probably the most known, complete and well defined framework for the development of Semantic Web Applications. It includes build-in reasoners and semantic web query languages for the development of Semantic Web based applications. As a third tool for the development of model SPARQL (Semantic Parsing Query Language) was used. SPARQL used as the semantic Web Query Language for the retrieval and filtering of data that are parsed through the designed model. The Semantic Web Cycle is as as follows: Design of the ontology Model; Consume data via web services or direct connection to internal or external Databases; Map the consumed data to the corresponding Entities of the model,

Reasoning of the retrieved data and generation of a unified data model in-memory (using Jena); Query the data with SPARQL queries using semantics.

Listing 7.1: Sample of the Ontology Model in OWL

```
<owl:ObjectProperty rdf:about="&design_patterns;CategorizedIn">
 <rdfs:domain rdf:resource="&design_patterns;pattern"/>
 <rdfs:range rdf:resource="&design_patterns;pattern_category"/>
 <rdfs:range rdf:resource="&design_patterns;pattern_subcategory"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&design_patterns;SubCategoryOf">
  <rdfs:range rdf:resource="&design_patterns;pattern_category"/>
  <rdfs:domain rdf:resource="&design_patterns;pattern_subcategory"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&design_patterns;architectureIsTypeOf">
  <rdfs:domain rdf:resource="&design_patterns;architecture"/>
  <rdfs:range rdf:resource="&design_patterns;architecture_type"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&design_patterns;hasStyle">
 <rdfs:domain rdf:resource="&design_patterns;pattern"/>
 <rdfs:range rdf:resource="&design_patterns;pattern_styles"/>
</owl:ObjectProperty>
```

7.4.1 Implementation of the Semantic Interoperability library

The implementation of the semantic analysis, data retrieval and filtering is done with the use of Jena framework [85]. The implementation mechanism is as follows: The designed ontologies, used by ArchReco prototype, are loaded in memory and a definition of the vocabulary used by each ontology, and their defined uri's are declared. For any external data-source, an endpoint url is used to access the included data (i.e Web Services, XML files or external Databases). The external retrieved data are loaded in the memory of the system, and the data are transformed into "in-memory models" with the use of a "data-to-rdf" mechanism that is implemented in the system. The created virtual rdf's are mapped to the Ontology models through configurations (for the time being, by the developer) and a set of SPARQL queries that parse the Ontology models retrieve and filter the data from the defined external sources. Figure 10 depicts the representation of the overall life-cycle of the data exchange in regards to the semantic mapping and representation.

For the evaluation of the Semantic model and the usage of the multiple data-sources it was necessary to import data from more than two sources and execute the implemented SPARQL queries to ensure that data are retrieved properly. During the evaluation a limitation was extracted, the lack of open, public accessible web services for existing Design Patterns repositories. In order to test the designed semantic model, data were collected from multiple Design Patterns repositories, either using crawling functionality or by hand, and the collected data stored in local database and xml files. The testing data sources applied in the designed model and a set of SPARQL queries were executed. For the current version of the prototype, the collected data were transferred in the ontology file, as part of the release package making possible the off-line data retrieval, filtering and reasoning, and thus, avoiding cold-start problems.

Examples of dynamic in memory creation from database and xml files as two different data-sources are depicted in B.1 and B.2. The semantic web data retrieval and usage life-cycle is depicted in figure 10

7.5 Semantic web Data Mapping

Data may exist in different data repositories having different structure and identification labels. The commonality between the data that exist in the repositories is the content and more specifically the scope of the content. For the better understanding the following example is given: Most of the systems have users registered in their databases. The name used for the entity "Users" may differ between the systems and it depends on the developer who defined the corresponding table (if it is a relational DB), the collection (if it is a NoSQL db), the tag (if it is an xml notation). Common names given to the entities are "User" without the ending "s", "SystemUsers", "UsersData" etc. Independently to the storage engine and the name that was used to define the Users the content and the scope of their existence is the same, they store the users' data. Therefore, for the creation of a system that collects the user data from all the repositories that contain users' data this can be achieved with the use of an ontology model, the data retrieval mechanism from the several end-points and the corresponding mapping of the retrieved datasets to the unified ontology model. Continuing the "Users" example, the modeling and mapping will be as shown in Listing 7.2.

Listing 7.2: Mapping sample

```
//Definition of the repositories uri's

@prefix myapp: <http://www.mynewsystemontology.com/ontology#> .
@prefix datafromsql: <http://www.sqldata.com/sql-data-mapping-model#> .
```

```
@prefix datafrommongo: <http://www.mongodata.com/mongo-data-
mapping-model#> .
@prefix datafromxml: <http://www.xmldata.com/xml-data-mapping-model#> .

//Definition of the semantically defined properties that
// all models contain

myapp:hasName a owl:DatatypeProperty .
datafromsql:hasName a owl:DatatypeProperty .
datafrommongo:hasName a owl:DatatypeProperty .
datafromxml:hasName a owl:DatatypeProperty .

datafromsql:UsersData rdfs:subClassOf myapp:Users .
datafromsql:hasFirstName rdfs:subPropertyOf myapp:hasName .
.
.
```

Using the example in 7.2 it is shown that the data coming from external sources are transformed into virtual models and for each one of them a uri is defined. Then we define a prefix for each particular Uri and we map the data objects and the data properties to the ones that the main ontology contains. With the above transformations and mapping it becomes possible to create queries that refer to the main ontology model and retrieve data from all external data sources that were transformed in the system.

Listing 7.3 depicts part of the mapping configuration that was used for the data retrieval from MySql DB and XML files from the Appendix examples B.1 and B.2.

Listing 7.3: Mapping sampl used in ArchReco Semantic Interoperability Layer

```
# Standard import statements
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

# SWRL imports
#@prefix ruleml: <http://www.w3.org/2003/11/ruleml#> .
@prefix swrl: <http://www.w3.org/2003/11/swrl#> .
@prefix swrlb: <http://www.w3.org/2003/11/swrlb#> .

# Domain imports
@prefix sadLatest: <http://scst.com/ontology#> .
@prefix cstapp: <http://www.cstapplication.com/ontology#> .

@prefix : <http://http://www.scst.com/sad-cst-mapping#> .

sadLatest:hasName a owl:DatatypeProperty .
sadLatest:hasEmail a owl:DatatypeProperty .

cstapp:hasName a owl:DatatypeProperty .
cstapp:hasEmail a owl:DatatypeProperty .
cstapp:hasProfession a owl:DatatypeProperty .
cstapp:hasEducation a owl:DatatypeProperty .
cstapp:hasUsername a owl:DatatypeProperty .
```

```
cstapp:hasShortCV a owl:DatatypeProperty .
cstapp:speaksLanguages a owl:DatatypeProperty .
cstapp:livesInCountry a owl:DatatypeProperty .
cstapp:worksInCompany a owl:DatatypeProperty .

cstapp:projectTitle a owl:DatatypeProperty .
cstapp:hasDescription a owl:DatatypeProperty .
cstapp:hasStatement a owl:DatatypeProperty .
cstapp:belongsToTopic a owl:DatatypeProperty .
cstapp:hasKeywords a owl:DatatypeProperty .
cstapp:hasRelatedTopics a owl:DatatypeProperty .
cstapp:hasMember a owl:DatatypeProperty .



cstapp:Person rdfs:subClassOf sadLatest:Person .
cstapp:hasName rdfs:subPropertyOf sadLatest:hasName .
cstapp:hasEmail rdfs:subPropertyOf sadLatest:hasEmail .
cstapp:hasProfession rdfs:subPropertyOf sadLatest:hasProfession .
cstapp:hasEducation rdfs:subPropertyOf sadLatest:hasEducation .
cstapp:hasUsername rdfs:subPropertyOf sadLatest:hasUsername .
cstapp:hasShortCV rdfs:subPropertyOf sadLatest:hasShortCV .
cstapp:speaksLanguages rdfs:subPropertyOf sadLatest:speaksLanguages .
cstapp:livesInCountry rdfs:subPropertyOf sadLatest:livesInCountry .
cstapp:worksInCompany rdfs:subPropertyOf sadLatest:worksInCompany .

cstapp:Project rdfs:subClassOf sadLatest:Project .
cstapp:hasTitle rdfs:subPropertyOf sadLatest:hasTitle .
```

```
cstapp:hasDescription rdfs:subPropertyOf sadLatest:hasDescription .
cstapp:hasStatement rdfs:subPropertyOf sadLatest:hasStatement .
cstapp:belongsToTopic rdfs:subPropertyOf sadLatest:belongsToTopic .
cstapp:hasKeywords rdfs:subPropertyOf sadLatest:hasKeywords .
cstapp:hasRelatedTopics rdfs:subPropertyOf sadLatest:hasRelatedTopics .
```

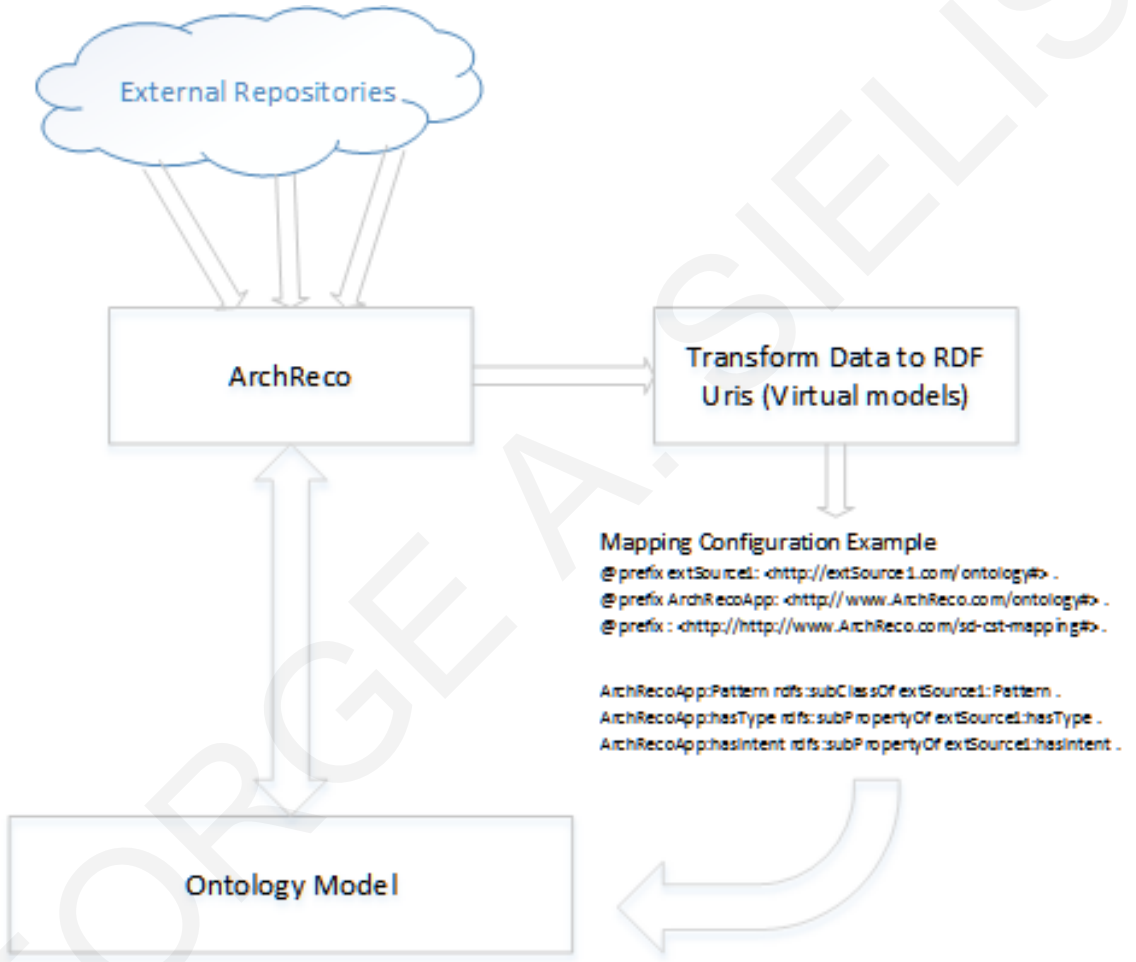

Figure 10: Semantic Web Usage

7.5.0.1 SPARQL queries - samples for context based data filtering

In this section, some of the functions that created for the retrieval of Design Patterns data using category as contextual information are depicted. In this case, "category" is used as input parameter in these functions. In listings B.3 and B.4 that can be found in Appendix B, the related SPARQL queries are depicted. In the examples shown, the definition of a SPARQL query is based on the uri prefixes, which point the models on which the SPARQL query refers to. In our case two prefixes are used: the "dp" prefix that defines the ontology model of the Design Patterns and the second one, "rdf", a semantic web standard prefix that points to an rdf schema which can be used for the semantic representation of relations between ontology entities/classes. The SPARQL queries are very similar to SQL queries but with different syntax and they are based on the semantic representations that a semantic model has. The SPARQL queries can be expressed in physical language in the way it is written, since the semantic relations can be used to translate a query into a physical language expression and vice versa. For example, in listing B.3 *the pattern name is known and based on this it is possible to retrieve the category name that the particular pattern belongs to*. The structure of the model is as follows: The Design Pattern belongs to a subcategory of a category of Design patterns. Therefore the Design Pattern's name is used as the basis for the query construction. The SPARQL query in listing B.3 can be expressed as:

"The pattern with name z which is categorized in a subcategory that is subcategory of a category and has category name x, Return the x."

7.6 Conclusions

In this chapter the Design Patterns contextual model was defined. The produced model integrated with the Creativity contextual model that was presented in chapter 5. With this integration,

the created model (in the form of Ontology) consists of the main conceptual model on which the Context Aware Recommendation system of ArchReco prototype, is built. In addition to that, the current chapter presented the "semantic web interoperability mechanism" through samples of semantic web queries. The chapter that follows, describes how the defined conceptual model is used for the development of the Context Aware Recommendation system for the computation of Design Patterns recommendations.

Moreover, the semantic interoperability mechanism that is presented consists of a reusable and extensible component. For the aims of the current thesis the component was developed as java library component but for future usage it is worth noticing, that the semantic interoperability mechanism is also implemented in the form of a web service, which can be adjusted and used for other projects on different research topics.

Chapter 8

Design and development of the Design Patterns Context Aware Recommendation System

8.1 Introduction

Reusability based on existing knowledge, assists Software Engineers in designing well-structured and well-defined High Level Software (HLS) Design Models [25]. This thesis deals with reusability through Design Patterns [5]. The plethora of Design Patterns and the several combinations between Design Patterns, makes Design Patterns training mandatory, and their correct usage in real problems, a challenging task. The objective of this chapter, is to introduce the thesis's approach for the Design Patterns recommendations. The recommendation system, belongs to the Context Aware Recommendation Systems that were described in chapters 3 and 4 respectively. The current chapter describes the developed Context Aware Recommendation engine, its architecture design and its filtering/reasoning methods. The design and implementation of the Context Aware Recommendation System is done for its integration with the ArchReco software prototype. Its usage of the system aims to the users' assistance to find the most suitable Design Patterns that

can be used for a given problem during the High Level Software Design modeling. With the recommendation of Design Patterns, users will be able to learn their content and select and apply them.

8.2 Context Aware Recommendations

The definition of the contextual elements in combination with their semantic representation as ontology models is used. For the development of the recommendation engine, for the ArchReco prototype, implements two recommendation methodologies, pure Text-based and Utility-based Context Aware Recommendations. For the design and development of the system, findings from previous chapters were used. In particular, the recommendation methods that applied were presented in chapter 4; the contextual model and the semantic interoperability component that were presented in chapter 7; designed and developed the Context Aware Recommendation System using the experience gained in chapter 5, from the generic Recommendation System for Creativity development.

8.2.1 Comparison with the Generic Recommendation System (Chapter 5)

The generic Recommendation System that was presented in chapter 5, was a proof that providing recommendations based on the designed Creativity Contextual model was valid. However, its development was also a chance to identify and track a number of problems and flaws, that should be avoided in a future development attempt. Some of the found problems were related to usability and also related to several development dependencies. First, semantics and semantic modeling was not properly used. Even the fact that the system was built based on a semantic model (designed using the Topic Maps technology), it was not taking advantage of the semantic technology and its advantages. For example, the system was not able to access and retrieve data

from multiple data-sources due to the semantic web platform that was used (Ontopia platform). The designed ontology, was used as a repository for storing input data and the developed queries for reasoning and filtering of data were executed with direct access to the stored content. In general it is not recommended to "write" data into ontology from the application level. The wrong operations and the limitations described, happened for two reasons: due to the specific Topic Maps technology and the Ontopia platform that was used as a Content Management System. In addition to the data access and retrieval, from external data sources, limitations there were problems with the design of the model, due to lack of experience at the time of the development. The overall filtering and reasoning that the Recommendation system was using was designed based on the "keywords" and "domains" that should describe any input that was referred to an entity. The keywords and domains should be defined from two parts, the users who were adding content and the users who were requesting recommendations. The lack of keywords or domains for an input would make data of the repository unreachable. Since, the recommendations were produced based on keywords and domains that the user should provide upon the recommendation request, caused usability problems and accuracy problems. User should be aware of keywords or domains in order to receive any kind of recommendations.

The experiences earned from the development and usage of the generic recommendation system (chapter 5), led to the requirements definition for the ArchReco prototype's Context Aware Recommendation System. The objectives for the ArchReco Context Aware recommendation Systems were based on three axes: *1. Minimization of the users' input, 2. Automatic recommendation triggering* and *3. Ability to provide Recommendation of items coming from more than one data-sources*.

8.2.2 Requirements definition

Using the outcomes of the generic implementation that was attempted in chapter 5 and the needs that were extracted in chapter 6, it became possible to define the requirements for the implementation of the Context Aware recommendation system of Design patterns. The requirements for the recommendation system are as follows:

- Development of a **content based** Context Aware Recommendation System.
- Delivery of recommendations of Design patterns coming from more than one repository
- Give the flexibility to the user to interract with the recommender in order to receive better results.
- Development of an extensible recommendation system on which more than one algorithms will be possible to be applied.

The particular requirements were met as follows:

- The developed recommendation system is focusing on the content than in producing recommendations based on ratings (collaborative filtering). The system is using the content of Design patterns that is coming from external or internal sources as individual documents which are indexed and analyzed for the computation of recommendations. Additionally, the system takes into account the input given by the users in free text and not only based on "keywords" and "domains" like in 5. The use of semantic web technologies in combination to the automatic indexing and filtering makes the input that the user has to create more creative and more simplified.
- The system uses the semantic interoperability component which was adjusted to the needs of the current thesis as it was described in chapter 7.

- The recommendation system implements the Utility based Context Aware Recommendation algorithm as post-filtering method, where the user is able to define weights of importance for specific contextual elements and adjust the recommendation results based on the corresponding preferences.

- The current system implements two recommendation methods and it can apply other algorithms in the case that it is needed. The semantic model that is used can be extended without modifying or influencing the current semantic queries and computations that the current system uses.

8.2.3 Contextual Elements Used the Software Engineering Training prototype

The context elements that were selected and used by the recommendation engine are defined as follows:

"Project" - Project refers to the description of the project that a diagram is created for. The project is defined by a "Title" and a "Description". The definition of the project with its title and description is a high level definition of the problem that the diagram solves.

"Application type" - The type of application is used for the selection of the most suitable set of Architectural templates that can be applied in the diagram. The type is used for the selection of the most suitable Architecture generic diagram that can be applied and the Architectural layers are used as contextual filtering elements when a diagrammatic shape is added in each layer.

"Architecture types" - Architecture types and their diagrammatic generic templates are defined based on the application type they are usually applied for. Like for example we identify four Architecture types based on the well known application types: *Desktop, Mobile, Service Oriented* and *Web Based applications*. Each Architecture type can have one or more architectures that can be applied for the development of each corresponding type. For example for web applications the

Architectural patterns of MVC, MVP, MVVM can be applied, for Mobile application the MVVM pattern is too common and for web services the Model-Controller may be used. The Architectural patterns can be related with more than one Software Architecture type.

“Shapes” - Each particular shape that exists in the palette has a specific type. The type of the shape defines the type of Design Patterns that may be referred to based on the categorization of the Design Patterns that was described in this section. For each shape user is able to add text describing a requirement and the recommended Design Patterns for the specific shape will be the most suitable for solving the described requirement.

“Structure of shapes” - As contextual information the structure of the shapes within a diagram is also taken into account. For a particular shape description the system is also able to collect the descriptions that were given for “parent” or “child” shapes that are connected with it. This is manly used for the Utility based algorithm only when the user requires the participation of other shapes in the recommendation filtering.

“User” - User is the main contextual entity of the system. User is the person who defines the overall design of a diagram; gives the input text that is used for the recommendation filtering; and is the entity that takes the decisions regarding the selection of a recommended Design Pattern or not, to apply in the diagram.

8.3 Recommendation Methods

As already mentioned, ArchReco prototype is integrated with two different Recommendation methods that use contextual information to produce recommendations of Design Patterns.

8.3.1 Text-Based Recommendations for Design Patterns

Text Based recommendations are triggered when user writes a description into a diagram shape. The input is based on physical language text, which is processed in the following way: the shape has a specific type such as Data operation, Server Side operation or User Interface operation. Based on the shape's type a pre-filtering [37] [38] on the Design Patterns is done with the use of SPARQL queries and the produced subset of Design Patterns is processed into an indexing mechanism (using LUCENE Framework [107]) which handles each pattern's attribute as indexed document. Then with the use of the TF-IDF [193] algorithm and the cosine similarity the input text is analysed and compared with the indexed attributes of the Design Patterns subset and produce the recommendation ranked list, based on the calculated similarity.

8.3.2 Utility Based Recommendation for Design Patterns

Similar to the first method, Utility-based Recommendation uses text filtering for the computation of the recommendations but not only the text is taken as input for a shape but also the text of the connected parent shapes, the connected children shapes, the title and the problem definition/description of the diagram. Additionally, the recommendations are computed and ranked based on the utility of each Design Pattern for the context that it is retrieved. The utility is changing dynamically since its computation depends on the weight of each contextual factor. The weights for each factor are defined by the user and thus the user may adjust the recommendations on her own preferences.

For the Utility Based method, four contextual factors were selected; the given title of the project, the general description (problem to solve), the parent shapes and the child shapes of the selected shape that the recommendations refer to. For each contextual factor the users are able to define weight of importance and also can modify the text for every factor until the results satisfy

the user's preferences. The Utility function is computed by equation 23 and the form by which the users can define the contextual factors weights of importance.

$$U = \sum_{i=1}^{N} \frac{w_i * f_i}{N} \tag{23}$$

The two methods have commonalities in regards to the pre-filtering methodology but differ in the post filtering, since the post filtering mechanism is applied only for the Utility-based method. Therefore the context in each particular case differs and the output result is changed regarding the items as well as the ranked list of presentation. In the rest of this section the pre-filtering and post-filtering methods are described.

8.4 Architecture of the Context Aware Recommendation System

The Context Aware Recommendation System developed as individual component that accepts input based on a specific request and returns a result-set based on the algorithm and filtering method that it is triggered. Following that design, it offers the flexibility for extensibility of the component as well as the transferability of the component for its integration to other applications. The component designed and developed following the functional Architecture that was presented in chapter 4, defined by the following four layers:

1. Collection of data
2. Filtering data
3. Ranking of the recommended items
4. Presentation of recommended items to the user

The recommendation system's architecture is depicted in figure 11.

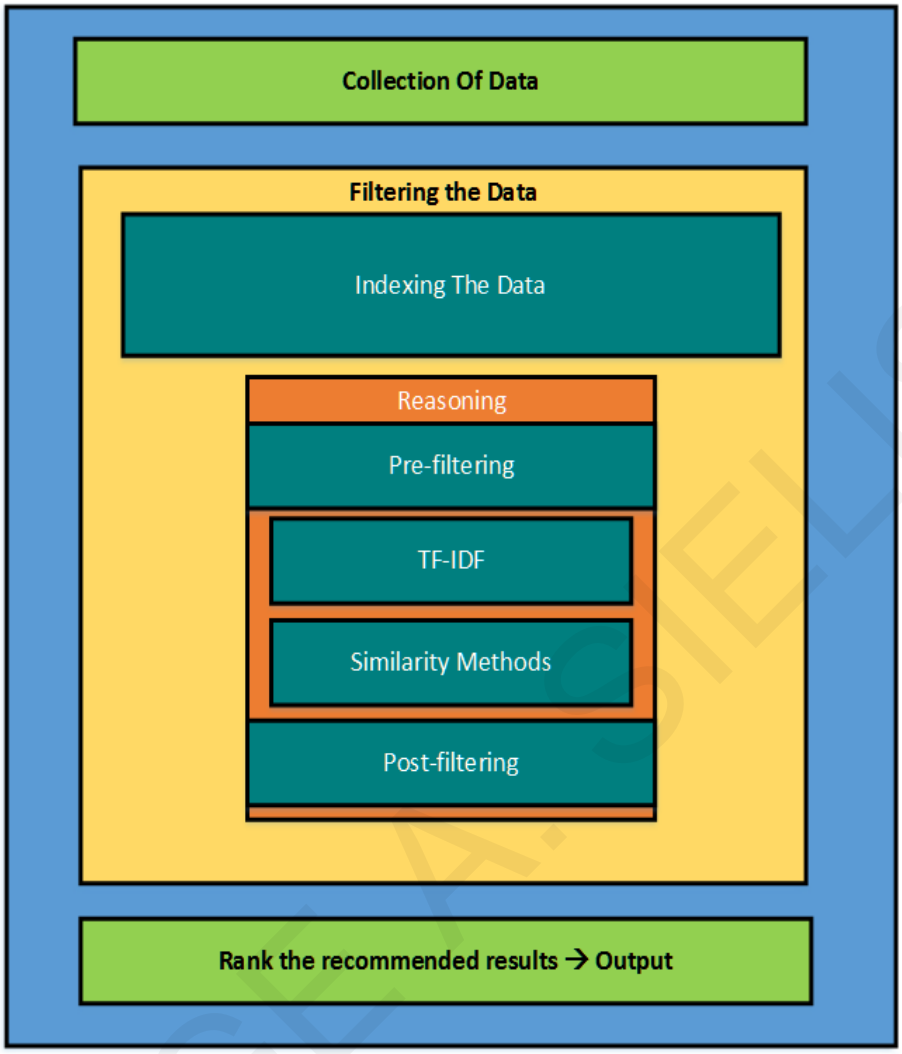

Figure 11: Context Aware Recommendation System Component Architecture

8.4.1 Collection of Data

The collection of data is a result between the interaction of the users' input in combination to the Semantic Interoperability layer that collects data from the defined data sources through the designed ontology model. The data are collected through the ontology model and the input of the user defines the context that will be taken into account by the component in order to trigger the corresponding filtering method that the system supports. The collection of the data and the way the system retrieves the data through the Semantic Interoperability layer is described in chapter 7.

8.4.2 Filtering the Data

For the filtering of the data the system supports two filtering methods: the pre-filtering and post-filtering. Each particular method is triggered based on the selected by the user method and they are functioning as follows:

8.4.2.1 Pre-filtering

For the pre-filtering method the filtering factor that is taken into account is the type of the shape that it is used for the description of a requirement within the diagram. Using a specific type for a shape defines which subset of Design Patterns categories or subcategories should be retrieved using the Ontology model. For example for a "Server Side" requirement shape, after adding it to the canvas and giving to it a requirement description, a SPARQL query is executed retrieving from all defined data-sources through the ontology model all categories that contain Server Side Design Patterns, such as the "Classic (Gang of Four - (GoF)), the Antipatterns" and the "Enterprise Application Architecture Patterns" and their subcategories. The retrieved subset of Design Patterns is then indexed as virtual documents having as content the descriptions of their attributes descriptions as they are defined by the corresponding source of retrieval. Then, the text that was given by the user as a requirement description is filtered by removing the stop words and vectorized. The created vector is used as input to the TF-IDF algorithm and using the cosine similarity it returns the most relevant Design Patterns ranked based on the number of common words found in each Design Pattern's attributes.

In the case of the Text-based method, the text that is taken into consideration during the computation, is only the shape's description. In the case of Utility-based method additional text is used, taken from the additional contextual elements that the user defined from the interface, such as the Title of the diagram, its description, the parent shapes text and the children shapes text. For

the Text-based filtering method the results are directly presented to the user. For the Utility-based method the results are passed into a post filtering mechanism that performs additional filtering based on the weights of importance that the user defined for each participating contextual factor.

8.4.2.2 Post-filtering

The pre-filtering results in combination with the LUCENE indexing and the TF-IDF algorithm return a list of results ranked based on the number of common words that were found for each Design Pattern. With the Utility based method that is used in ArchReco prototype the user is able to define for a set of pre-defined contextual factors the weight of importance that each factor may have in the computation and if the factor should be included in the computation or not. The weights of the context factors and their text inputs are passed into the recommendation engine and then the content is used for the pre-filtering and the weights are used for the post-filtering computations.The list of Design Pattern results that is produced by the pre-filtering is now re-ranked based on the weights of importance and thus the utility value of each Design Pattern. The computation used for the utility given in equation 23 defines a new rank for the Design Patterns results which reflect to the user's opinion regarding the utility of each Design Pattern.

8.4.3 Ranking and presentation of the Data

After the filtering methods are executed the data are ranked based on the number of word "hits" of words and the similarity method that is used by the TF-IDF algorithm in the case of pre-filtering; and based on the utility function results in the case of post filtering. The results are returned by the component in the form of a ranked list to the presentation layer, the ArchReco software, and presented to the users.

8.5 Conclusions

his chapter described the Context Aware Recommendation Component and how this is used in ArchReco prototype. It described the high level architecture of the component which is designed following the recommendation Systems functional architecture that was presented in chapter 4. It describes the methods that were used for both pre-filtering and post-filtering methods and how the analysis of the content is performed.

Chapter 9

ArchReco Software Prototype

9.1 Introduction

The thesis statement that was set in chapter 1 defined two objectives for this thesis: “First the definition of the necessary contextual models that can be used for the development of the Context Aware Recommendation system that will take into account the Creativity and Software Engineering principles”, and “second the development of a software prototype tool which will apply the developed Context Aware Recommendation System (CARS) for assisting the students in learning and at the same time practicing the High Level Software design with the use of Design Patterns”. From the previous chapters the first objective was met through the definition and design of the contextual model for Creativity and the Creativity in combination to Software Engineering concepts, and more specifically the design patterns. The second objective is a combination between the defined models, the implemented semantic interoperability layer (chapter 7), the Context Aware recommendation system for Design patterns (chapter 8) and the findings from chapters 3, 4 and 5.

This chapter consists of the work related to the second objective of this thesis. It presents the ArchReco software prototype which is the result of the work presented in the previous chapters.

ArchReco is a software prototype that was implemented following the creativity models given by [8] and [152] and integrating the models and components that were presented in chapters 7 and 8.

9.2 Requirements definition

The requirements that were set for the design and development of the ArchReco prototype were set based on the thesis statement objectives. The requirements that were set are the following:

- Development of a Creative environment were user will be able to design High Level Software Diagrams.
- Support of the user by providing Context Aware Recommendations of Design patterns.
- Offering the ability to extract the designed model in transferable formats (e.g. xml based files)

9.3 ArchReco prototype

ArchReco is a Software Design prototype tool which can be used for the design of HLS Diagrams. High Level Software Design is defined in [25] as "*a collection of module and subroutine interfaces related to each other by means of USES and IS COMPONENT OF relationships. Precise and formalized information on module or subroutine bodies is not yet available at the stage of High Level Design*". The tool provides components such as the design canvas, design shapes palette, and the Context Aware Recommendation mechanism that provides recommendations of Design Patterns during the design process. ArchReco is a java based application supported by Semantic Web technologies such as Jena Framework, SPARQL query language, MySql Database engine (whenever it becomes necessary) and OWL for the representation of the conceptual models that are used by the prototype.

9.3.1 ArchReco usage

ArchReco prototype is a software tool that can be used for the design of diagrams with the use of shapes (drag ‘n drop to the canvas component) and for each shape users are able to provide descriptions to the purpose of use describing their functional purpose in physical language text. Each shape has a type which is related to the requirement type that it can be used for. The prototype also provides a set of pre-defined Architectural templates, with generic information and structure so the user is able to enrich the design by adding shapes of different types on particular architectural layers. The shape types in combination to the Architectural templates (different subsets of templates per application type i.e. Web, Desktop, Mobile application) and the provided input text, by the user, consist part of the contextual information that is used for the data filtering and the production of the most relevant Design Pattern recommendations. ArchReco was designed and developed for particular target group, CS or Computer Engineering students, who want to learn how to design HLS Diagrams and in particular, learn the Design Patterns and their usage. For that reason, CARs of Design Patterns is an important enhancement that aims to the provision of support to the students who use the tool in training and learning by practice the Design Patterns coming from multiple repositories that are defined.

Retrieval of different Design Patterns types, known or new ones, presumes the accessibility to several Design Pattern repositories for the collection of their surrounding information in a structured and unified form. For that reason a number of ontologies have been introduced (Design Patterns, user, software project ontologies), for the modeling of the Design Patterns, and the contextual elements in a structured form. Using the ontology models, a semantic analysis engine was built for querying the retrieved data with the use of SPARQL query language. The most important ontology model for the computation of Design Patterns recommendation is the Design Patterns

ontology. Design Patterns ontology contains data following the Design Patterns attributes structure that is commonly used for the Gang of Four (GoF) [59] patterns. More specifically, Design Patterns are structured in terms of the Intent, Motivation, Applicability, Structure, Participants, Collaborations, Consequences, Implementation, Known Uses and Related Patterns. Design Patterns are grouped in categories based on their usual usage type like for example “Data”, “UI” or “Operational” Design Patterns.

9.3.2 ArchReco Prototype as a Design Patterns Training Tool

The objective that the ArchReco prototype aimed to meet is the training of Computer Science and Computer Engineering students in learning the Design Patterns through practice. The Context Aware Recommendations is the aiding tool that was used for achieving the task. The overall design of the prototype was done based on the assumption that the users of the system do not know UML, since UML design is used for functional design and demands a good level of programming language knowledge and experience in similar modelling tools. Additionally, the tool is designed to offer aiding tools other than the Context Aware recommendations for Design Patterns. By the initialization of a diagram model/project users are able to define the problem they want to transform into a model, define the type of application that the High Level Design is about, such as Web, Mobile, Web Service or Desktop Application and the nature of the application such as if it will be a static or dynamic content application. Using the input given by the beginning of a new design the initializing input is analyzed and a list of Architectural templates for the defined application types is presented. For example, for a dynamic Web-based application the list with the Architectural Patterns contains the MVC (Model View Controller), MVVM (Model View - View Model), MVP (Model View Presenter) and other Architectural Patterns that are commonly used

for Web Applications are presented. For each Architectural Pattern there is a complete description coming from the existing related literature and an image that describes its high level design.

The prototype consists of four areas/panels. In these panels the following modules exist: A palette with custom shapes (not UML shapes), a canvas, the diagram model details and panel with three tabs that contain the Design Patterns recommendation results, the Architectural Patterns and a set of Design Principles. The shapes of the palette have names that help the user to understand the purpose that they can be assigned for and each shape triggers a specific semantic pre-filtering mechanism based on their usage. The recommendation tabs on the right panel are automatically expanded and provide information according to the performed actions. For example, when the user defines the type of the application, the Architectural Patterns tab is expanded containing the selected type related Architectural Patterns, and when a shape is dragged in the canvas, the Recommendations of Design Patterns tab expands, containing the recommended patterns that are produced based on the user's textual input. The automatic expansion of the tabs aimed to attract the user's attention when an action is performed and make the user to select the recommended patterns and study their provided content. For each Pattern Architectural or Design, the user is able to add them in the canvas by pressing a button.

The ability to design diagrams on predefined Architectural diagrams over the canvas in combination with the Context Aware Recommendations for the Design Patterns offer the users an environment where they can design a high level software diagram based on a given problem without the need for advanced knowledge in Software Design modelling. The given environment offers accessibility to Design Patterns information for learning without being necessary to search the web. Additionally, the use of context for the recommendation of Design Patterns gives the opportunity to the users to examine a minimized set of Design Patterns that is produced based on the problem description, the specific task description and the type that a shape of the diagram

has. The designed diagram then can be extracted in several formats from which one of them is in *.XMI an XML based modelling language which can later be transformed into a uml diagram. The transformation of the XMI language into UML is not included in the scope of this work. A sample of the expected diagram is depicted in figure 12. A sample XMI exported file is shown in listing 9.1. ArchReco prototype is depicted in figure 13.

Listing 9.1: XMI diagram

```
<mxGraphModel>
        <root>
                <mxCell id="0"/>
                <mxCell id="1" parent="0"/>
                <mxCell id="Model" parent="1" style="shape=hexagon" value="Model"
                 vertex="1">
                        <mxGeometry as="geometry" height="100.0" width="580.0"
                         x="40.0"/>
                </mxCell>
                <mxCell id="View" parent="1" style="Interface;swimlane" value="View"
                 vertex="1">
                        <mxGeometry as="geometry" height="180.0" width="580.0"
                         x="40.0" y="130.0"/>
                </mxCell>
        </root>
</mxGraphModel>
```

9.4 Prototype Implementation

The prototype design tool was developed as a graph editor that is built on the basis of the jGraph framework [10]. The prototype enhanced by functionality that supports recommendations for Design Patterns in runtime, during the design of Software Architecture Diagrams. The recommendation algorithms and the methods of presenting the recommendation results were built as individual components and integrated with the basic graph editor tool, which exists in the jGraph package. Generally speaking, the Software Architecture prototype is a complete graph editor consisted of the following modules/components:

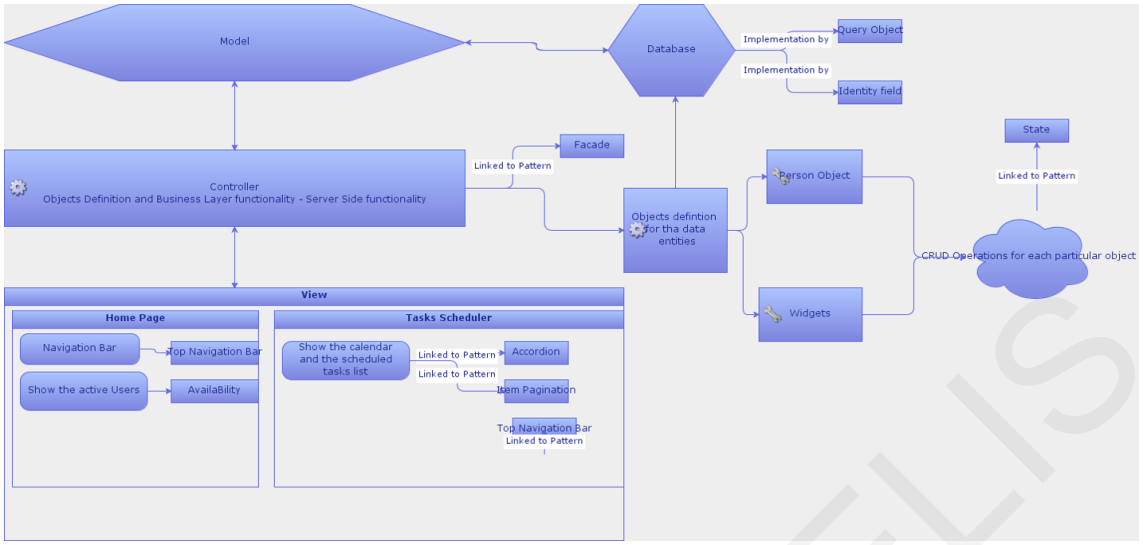

Figure 12: ArchReco Diagram Sample

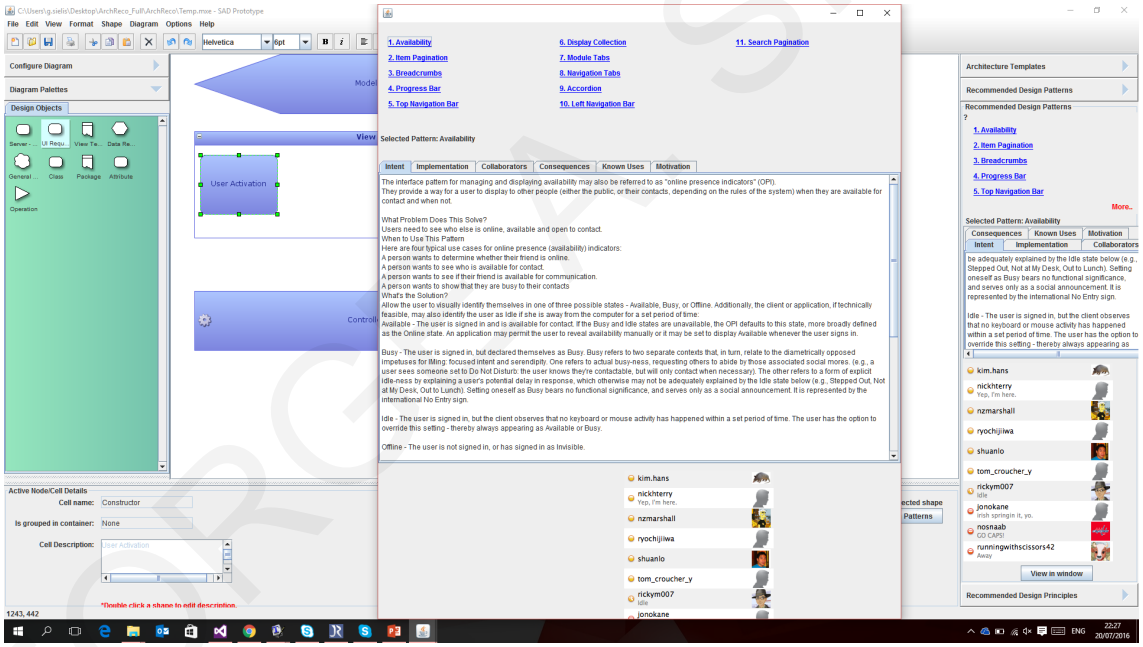

Figure 13: ArchReco Prototype

- Graph Editor Canvas where the designer is able to design Architecture Diagrams (Support of styling and editing the diagrams).

- Palette of diagram components where all the diagram components exist and the designer is able to drag to and drop the components in the canvas.

- Architecture Diagram Information Area where the designer can write the meta-data information of the working model such as a description or a title.

- Diagram Actions Area module, which is changing according to the selected recommendation algorithm. In this area the designer is able to see information of the selected nodes (from the designed diagram) or interact with the tool according the recommendation algorithm requirements. The input parameters that are required based on the selected algorithm are described in a following section with the description of each particular context-aware recommendation method.

- Recommendation of Design Patterns Area module: In this area the Design Patterns Recommendation results are shown. It is structured in such a way that the designer by selecting a recommended pattern can acquire full knowledge of its characteristics and its generic UML diagram, and he is also able to insert the selected pattern in the diagram by pressing a button.

For the development of the implemented components that were integrated with the prototype software a combination of methods were used. For example, for modeling the data, two ontologies were defined and used in combination with the database engine SPARQL ontology query language, as well as dynamic transformation of data into an ontology based description language, whenever this was necessary. The use of semantic web technologies, especially for modeling the data, was done for two reasons: Firstly, for re-usability reasons, since the conceptual model that was defined can be reused for the development of similar applications. Secondly, semantic web technologies offer the flexibility to retrieve data from multiple sources, handle and structure the way it suits the application and easily query the data to retrieve the necessary information. For the

aim of this work, two ontologies were defined and designed; the Design Patterns Ontology model and a second ontology for modeling the context based on the contextual elements identified for the current domain of work: Software Architecture Design.

9.4.1 System Architecture

As mentioned before the ArchReco prototype was implemented in Java programming language using the JSwing framework. The application is developed as a desktop application that is consisted of three parts, the Core, Integration Layer (Integration of the partial components) and the presentation layer. The overall architecture is depicted in figure 14.

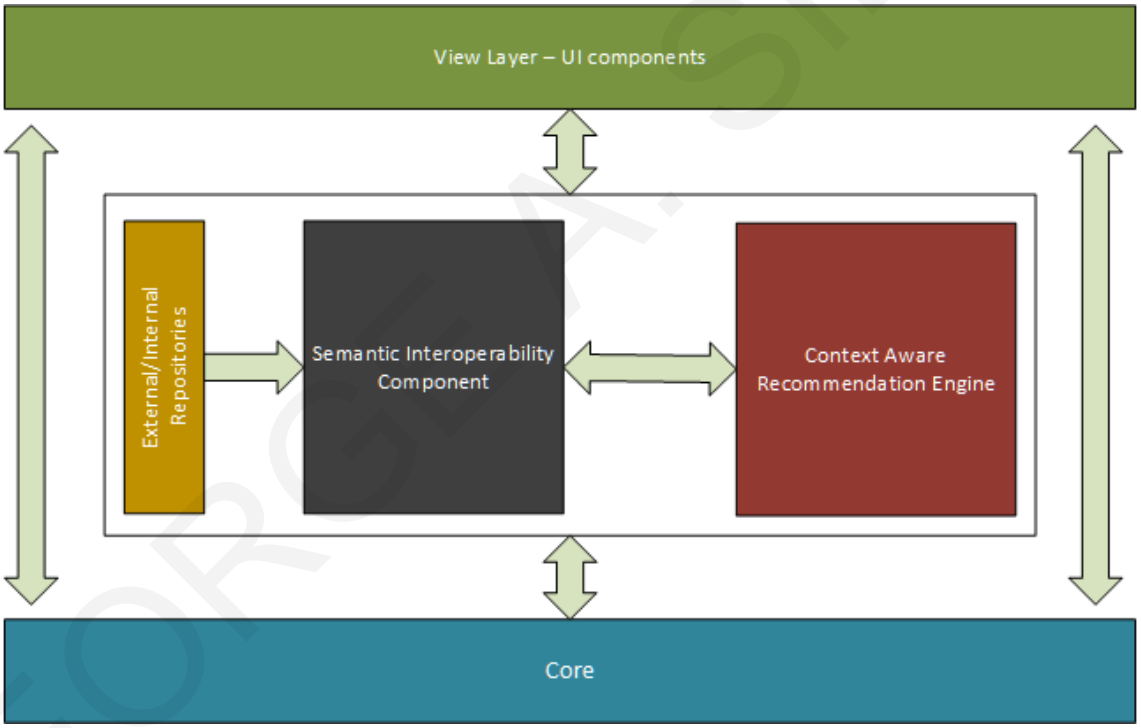

Figure 14: ArchReco System Architecture

The **Core layer** contains the core functionality of the system, such as the data bus of the system for transferring messages and data between the components or between the layers. The Core also contains the Object Definitions for entities that exist in the system.

The **Integration Layer** contains the Semantic Interoperability Component and the Context Aware Recommendation engine. Each component is responsible for the processing of the data and the operations they perform according to the role they have in the system as they were described in the previous chapters. The Context Aware recommendation engine is also able to exchange data between the view layer and the core, especially in the case of the post-filtering process.

The **View layer** contains the JSwing components that are used for the presentation of the software to the user. With the presentation layer users are able to provide input to to the system which then will be processed by the other two architectural layers.

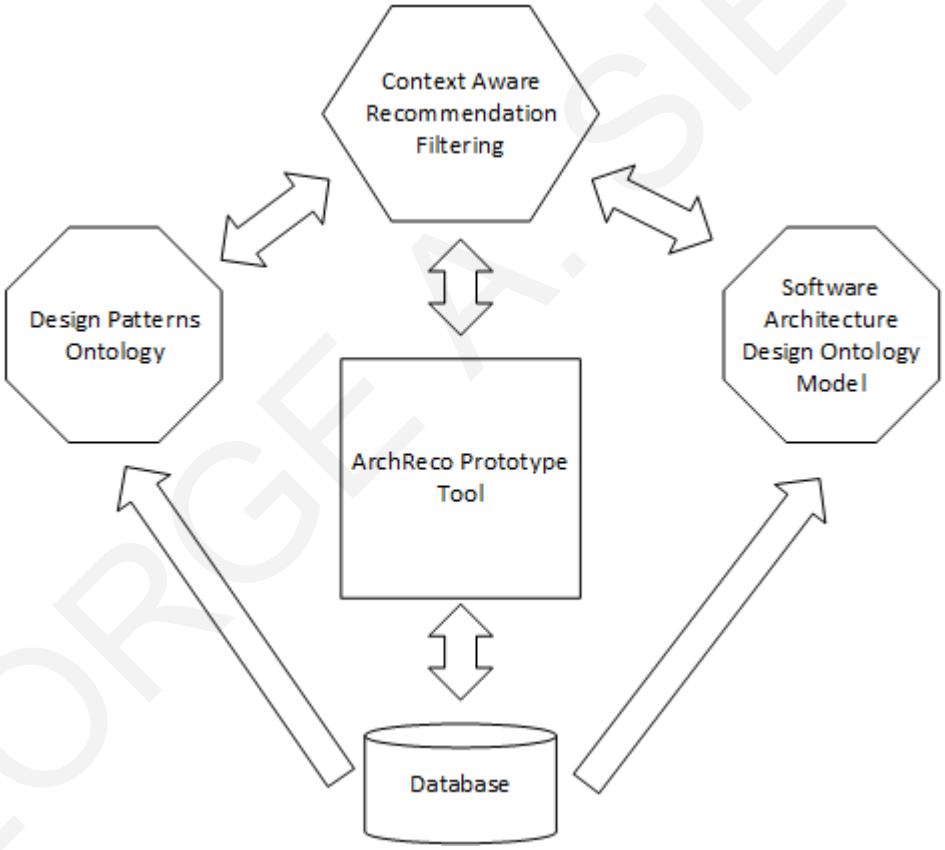

Figure 15: ArchReco Components Communication

In figure 15 the communication between the components is depicted. As it is shown the individual components or modules are able to exchange data between each other and all functionality

is passed through the ArchReco prototype. In this figure a database is also used for the storage of persistent data related to the interaction between the prototype and the users, but database is not a dependency for the software so it can also operates offline.

9.5 ArchReco prototype description

This section aims to present the ArchReco prototype through its interface modules and how users are able to interact with each particular module.

ArchReco's interface is divided in 4 main panels (figure 16) which are implemented as individual modules: *the canvas, the left panel/module, the righ panel/module* and *the bottom panel.*

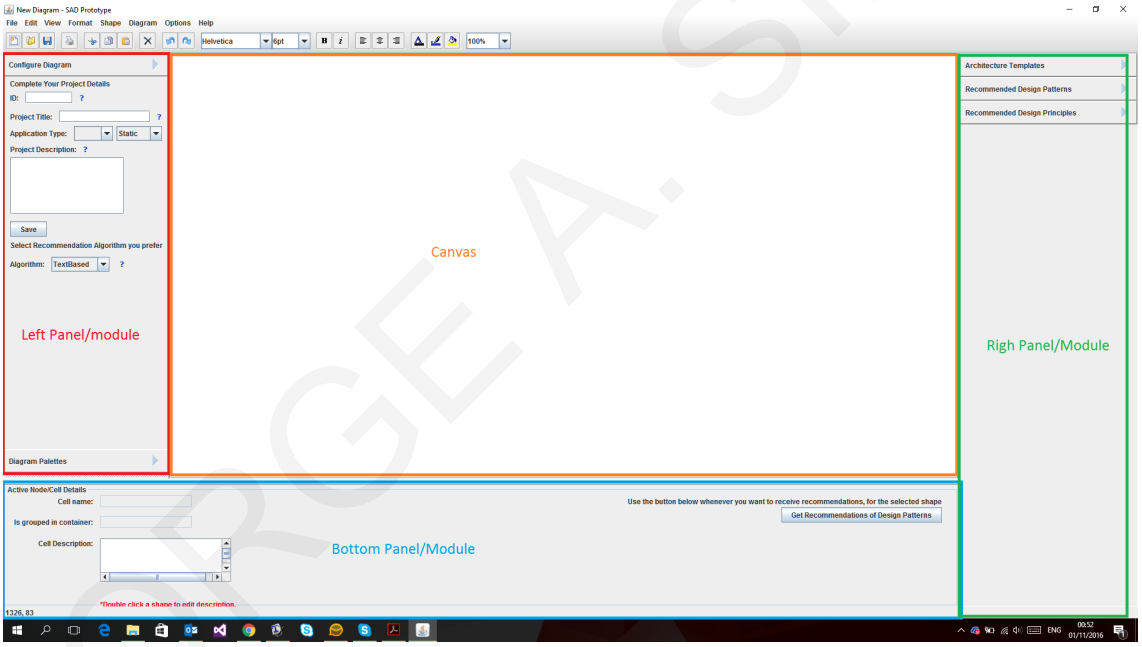

Figure 16: ArchReco interface divided in areas

9.5.1 Canvas

In the central panel there is the canvas module on which the user is able to draw diagrams using the diagramming palette on the left side. From the right side module additional automated designs

can be added to the canvas through specific actions according to the provided recommendations in each case.

9.5.2 Left Panel/Module

The Left module (figure 17) contains a set of input controls that the user has to complete before or during the design of a diagram.

9.5.2.1 Configure Diagram

In the left panel there are 2 tabs. In the first tab "Configure Diagram" the user can describe the pattern to be created. A diagram is considered as a new project for which the title has to be declared, the type of application (whether it is web, mobile application etc.) and general description.

The input given in the provided controls are taken into account in the recommendations filtering. These given input can be modified any time during the design process by pressing the "*Edit information*" button and "*Save*".

In the field "Algorithm" there are three options from which two of them are active at this stage. Default option is "TextBased" and the second is the active selection "UtilityBased". This field addresses the algorithm used by the system for the creation of recommendations of Design Patterns while creating a diagram. The recommendations are generated based on the description given by the designer in every shape that the user adds in the chart. The descriptions of shapes depend on how designer describes a shape regarding the purpose the shape serves in the diagram. By "UtilityBased" algorithm, the user is able to determine the weight of importance for certain factors which are taken into account for the creation of recommendations of Design Patterns. By choosing the "UtilityBased" the lower panel displays the corresponding controls that the user can

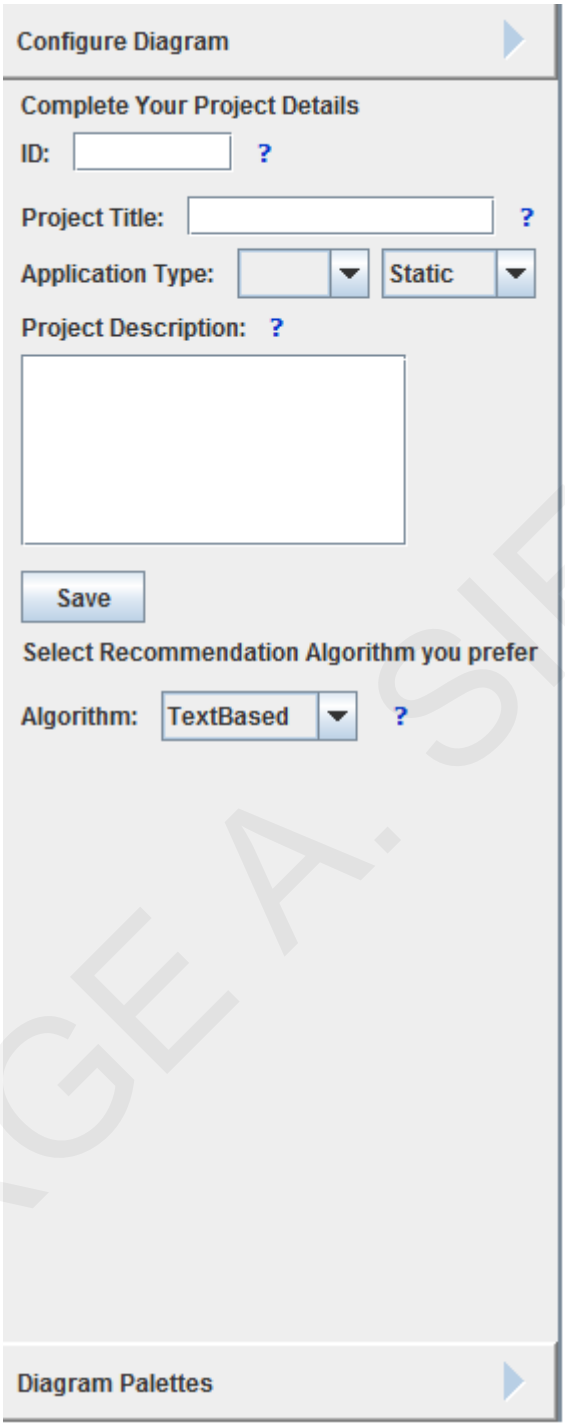

Figure 17: ArchReco Left Side Panel/Module

interract with, and affect the results of recommendations which are given for a selected shape on the canvas.

9.5.2.2 Diagram Palettes

As shown in the figure 18 the palette contains nine shapes that can be used for the design of a diagram. Each shape is different depending on its name and what each one of them represents. The type of shape plays a role in what kind of design patterns will be recommended, which are related to the requirement that the shape describes. For example a “server side requirement” may have the description of a server side operation and the given recommendations for Design Patterns will be related to the server side operations of the High Level Diagram design. In a “UI requirement” Design patterns related to the on the appearance of the system will be recommended such as - UI patterns design (client side).

In particular the shapes are described as follows:

- **Server Side Requirement** - Used to describe features that will run the server. Recommended Design patterns are related design patterns used to implement functions in the sever plane as e.g. GoF Design patterns.
- **UI Requirements** - Is used to describe functions will be performed in the client. It may be a description of a simple user interface widget or module such as horizontal menu.
- **View Template** - Used as a container for more than one UI Requirements. It can be considered as a separate page which can contain many UI elements.
- **Data Requirement** - Used for requirement description related to data.The system proposes patterns design on methods used to design databases or writing data to static files. According to the given description a list of Data related Design Patterns is recommended.
- **General** - This shape can be used for the description of a general requirement that is related to all types of design patterns such as Data, UI, Server Side related Design Patterns.

Figure 18: ArchReco Left Side Panel/Module - Shapes Palette

- **Class** - Used when we want to show in the diagram a representation of a specific class which will help the developer understand a particular function/operation on the diagram.
- **Package** - General-purpose container which can be used for grouping shapes or partial diagrams
- **Attribute** - It can be used for the description of features that will be supported by the designed system. It can also be used to mark or remark will help the developer to better understand some design functionalities.
- **Operation** - Description of a function derived from external program or web service.

9.5.3 Right panel/module

The right panel (figure 19) contains three tabs. Architecture templates, Recommended Design Patterns and Recommended Design Principles.

- **Architecture templates** - It contains descriptions Architectural patterns which correspond to the several types of applications. In the left panel in the tab "Configure Diagram" when choosing application type and selecting the application type, Architecture Templates automatically correspond to each type. For each architecture template the user is able to read and understand the description of the architecture or is able to apply it to the canvas. In some cases architecture.
- **Recommended Design Patterns** - This tab presents the recommended Design Patterns. Recommendations are presented automatically depending on the selected shape in the diagram. The selection criteria of the proposed design patterns is the type of the selected shape, the description of each shape, as to what it serves, and in case of use of Utility-Based Algorithm, the criteria and weights specified by the user. For each design pattern, the user can

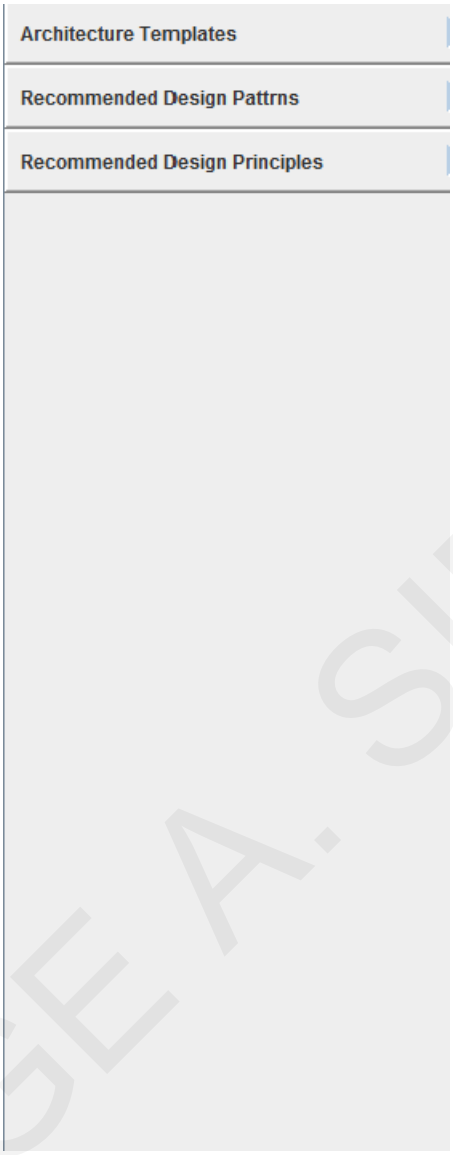

Figure 19: ArchReco Right Side Panel/Module

read the description of available features and design representation if any. Also a pattern design can be automatically added to the diagram by connecting edge in design element that is pre-selected by the user pressing the “Add button pattern”.

- **Recommended Design Principles** - Design principles that are registered in the system for optional usage and learning.

9.5.4 Bottom Panel

At the bottom panel (figures 20 and 21) the selected shapes information is shown. For each selected item apart from the automated received recommendations, the user can request to receive recommendations by pressing the button “Get Recommendation of Design Patterns”. The button is commonly used for shapes such as e.g. General or Class which does not automatically trigger the mechanism proposals for patterns design.

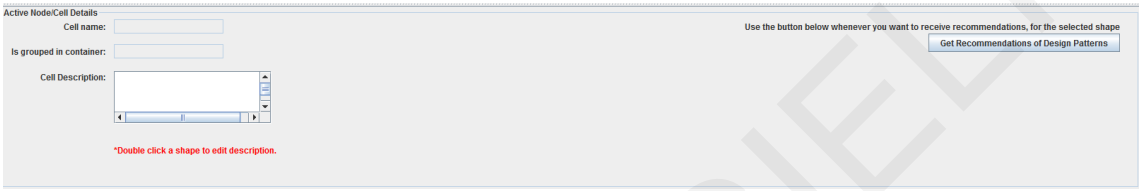

Figure 20: ArchReco bottom Panel/Module (Text - Based Recommendation Algorithm)

By changing the recommendation algorithm to Utility-Based a list of contextual controls is displayed (figure 21). The controls are used by the users to modify the recommended results based on the weights that each contextual element may have.

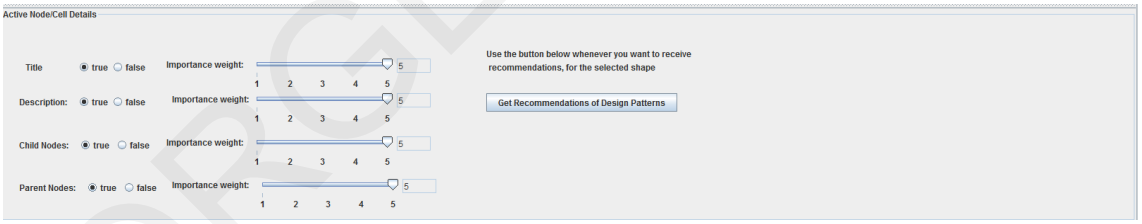

Figure 21: ArchReco bottom Panel/Module (Utility - Based Recommendation Algorithm)

Chapter 10

Evaluation

10.1 Introduction

The development of the ArchReco prototype and its integration with the Context-Aware Recommendations for Design Patterns was given to representative users for testing, aiming to collect feedback and at the same time be able to track possible limitations for future improvements. Additionally, the evaluation's objective was to examine how users perceive the prototype in terms of its educational character, its usefulness and usability. Therefore, the software was given to users who were asked to execute a specific task remotely (i.e. not in a lab environment), capturing their sessions in video form, for further analysis regarding their interaction while they were performing the task. In particular, with the evaluation process and the questionnaires given, the evaluation resulted in a significant set of results regarding the objectives set by the evaluation planning. The evaluation plan was designed based on relevant evaluation methodologies that can be found in [94] [35] [42] [99].

The ArchReco prototype was developed and designed as a Software Engineering Educational and Training tool for the support of users in learning Design Patterns by practice as part of the process of the design of high level software diagrammatic models. The evaluation of the prototype

was setup as a remote test and targeted students and researchers who actively involved in Software Engineering and Software Design. The prototype developed as a stand-alone application but that was not a restrictive factor carrying out the test remotely. To make sure that the task given to the participants was completed in a proper way and to be able to track their reactions while they were using the software, the test sessions were captured in video form. The participants were asked to submit the video upon completion of their individual test sessions. Additionally, related help material was provided to them such as tutorial for the prototype and also help functionality such as "Help" button and "Information" links internally in the software. With the session videos captured, it became feasible to measure times of reactions, possible difficulties related to the interface design and examine the overall usage of the tool and the produced Context Aware Recommendations in designing the High Level Software Diagram with the usage of Design Patterns.

For the design, planning and execution of the evaluation, a number of related evaluation frameworks were examined. The examined frameworks that guided the current evaluation design are presented in the following subsection.

10.2 Evaluation Frameworks

ResQue [133], is an evaluation framework that stands for recommender System's Quality of user experience. It is a complete evaluation framework for the evaluation of recommender systems from user's perspective. It measures the quality of the recommended items based on the system's usability, usefulness, interface and qualities, user satisfaction and the influence of these qualities on users' behavioral intentions. The framework contains 20 questions measuring the latter metrics and those questions are the result of a large survey that determined them as the most important questions that user preferences depend on.

Hayes et al. [76] propose an evaluation framework which is based on the idea of system utility by comparing how a recommendation strategy performs against another. With this framework the setup evaluation presumes the existence of a common dataset, a common interface and the mechanism to change the recommendation strategy, aiming by this to measure the user's satisfaction according to the used recommendation strategy.

Knijnenburg [90], believe that user experience is an ill-defined concept that lacks well-developed assessments methods and metrics. Therefore, they suggest an evaluation framework which distinguishes between objective system aspects (algorithms, user interface features etc.), subjective system aspects (user's perceptions of the objective system aspects) and interactions (user behavior). The subjective system aspects are measured with questionnaires and they are expected to mediate the influence of the objective system aspects on the user experience. The framework is focused on the distinction between attitude and behavior and more specifically the experience and interaction. The experience signifies the users' evaluation of the system and it is measured with the use of questionnaires that are divided into the evaluation of the system, as *system experience*, the evaluation of the decision process *process experience* and the evaluation of the final decisions *outcome experience*. *Interaction* is the observable behavior of the user.

10.2.1 Evaluation Methodology

The methodology used for the ArchReco prototype evaluation was mostly based on the ResQue framework [133]. The questionnaire presented in [133], was the most complete and relevant to the objectives set for the ArchReco evaluation, due to the evaluation of the Context Aware Recommendations as individual component of the system, but also the usage of the prototype as a

complete solution supported by the recommendation systems. The design of the evaluation questionnaires, for both, pre-test and post-test were designed using 5-likert scale type questions [94], grouped based on the guidelines that were revealed from [133].

The evaluation methodology, designed based on the collection of opinions from Computer Science, Computer Engineering students and researchers for the qualitative analysis of the results regarding usefulness, functionality and the recommendation tool's usage. The lack of a satisfactory number of valid users at the University of Cyprus led to the design and setup of the evaluation as remote evaluation (non-controlled) by which the sessions were monitored through screen capturing videos during the evaluation process. For the confirmation that the evaluation participants were valid for the evaluation a pre-test questionnaire for the collection of data regarding their profession, year of studies and Design Patterns knowledge background was prepared. A valid evaluation result would be considered a session which would be complete the given task, by designing a complete High Level Diagram based on the task's requirements and most importantly include in the diagram a valid number of Design Patterns.

10.3 Evaluation setup

The evaluation process designed as follows: The participants were asked to sign a Non-Disclosure form by adding their emails in an on-line form and by accepting the terms of the evaluation they were able to proceed to a web-page where they had to complete 5 steps. *1.Complete the demographics data questionnaire, 2.Download and open the prototype, 3. Perform a given task with the prototype, 4. Answer a post-test questionnaire* and *5. Send the screen capturing video to us*. The evaluation request was given to Computer Science and Computer Engineering students of the Computer Science Department of University of Cyprus and the Computer Engineering and Informatics of the University of Patras. Additionally Computer Engineering researchers of the

Institute of Information Technology (ITI/CERTH) in Greece were also participated. In total 28 responses were received where 23 (83.9%) were from men and 5 (17.1%) were from women. The task that the participants asked to perform was "*to design the high level diagram for a web based electronic bookstore showing the CRUD operations of the system using the MVC architecture and definition of the most suitable Design Patterns at the three architectural layers*". The evaluation can still be accessed at the following URL: http://www.cs.ucy.ac.cy/~sielis.

10.3.1 Pre-test questionnaire

As mentioned above the evaluation was performed by 28 people from which 23 were men and 5 were women. The participants were mostly students and researchers with Computer Science (75%) or Computer Engineering (25%) background, from Cyprus (71.4%) and Greece(28.6%). Their ages were between 22 and 38 years and more specifically 53.6% had ages between 20-25, 35.7% between 26-31 years old, 7.1% between 32-37 years old and 3.6% between 38-43 years old. More detailed descriptions for the participants are shown in Table 8. (See also Appendices D.1 and D.2)

Before the execution of the task the participants were also requested to rate their experience in using relevant tools, Design Patterns knowledge and experience in using Design patterns. A summary with the given responses and the standard deviations are depicted in Table 9. As it is shown in Table 9 the experience in using relevant tools and in general their experience in Design Patterns and Software Engineering design was low. The high standard deviation for the three questions lead to the conclusion that the responses were spread in relation to the calculated mean value. Therefore, using the three questions we tried to correlate the responses with some individual variables and explore how the results of these questions are correlated and possibly influence results from the post test questionnaire that we analyze later in this section. We examined the

Table 8: Profile of participants (N=28)

	Item	Frequency	Percentage
Gender	Male	23	82.1%
	Female	5	17.9%
Age	20-25	15	53.6%
	26-31	10	35.7%
	32-37	2	37.1%
	38-43	1	3.6%
Profession	Student (Bsc) / (Msc)	12	42.9%
	Computer Scientist	6	21.4%
	Computer Engineer	8	28.6%
	Researcher	2	7.1%
Nationality	Greek	8	28.6%
	Cypriot	20	71.4%
Location	Nicosia(CY)	16	57.1%
	Patra(GR)	3	10.7%
	Salonica(GR)	3	10.7%
	Larnaca(CY)	4	14.3%
	Limassol(CY)	1	3.6%
	Leeds(UK)	1	3.6%
Educational Level	Bsc	12	42.9%
	Msc	11	39.3%
	PhD	5	17.9%

correlations between the questions from table 9 based on the Individual grouping variables such as the gender, age, profession,nationality, location and year of studies.

Table 9: Pre-Test responses for the participants' experience

Question	**Mean**	**St. Deviation**
PreTest-Q1. Please rate your experience with Software Design tooling, understood as systems that promote, accelerate and facilitate the design of Software Design Models?	1.75	0.441
PreTest-Q2. Please rate your level of experience in Software Design	2.821	0.8630
PreTest-Q3. Please rate your knowledge of Design Patterns	2.571	1.26
PreTest-Q4. Please rate your experience in using Design Patterns	2.821	1.105

The Pearson's coefficient correlation showed with statistical significance 0.024 at the $p<0.05$, relation between profession and the "*Experience using Design Patterns*". Moreover, statistical significance 0.0001 at the $p<0.01$ level between profession and "*Knowledge of Design Patterns*" was also found. There is also statistical significance 0.048 between location and "*Experience in Software Design*", "*Nationality*" and "*Experience in Software Design*" (sig = 0.021 and $p<0.05$), "*Nationality*" and "*Knowledge of Design Patterns*" (sig=0.031 and $p<0.05$), "*Gender*" and "*Experience using Design Patterns*" (sig=0.037 and $p<0.05$) and finally correlation between "*Educational background*" and "*Experience using Design Patterns*" (sig=0.003 and $p<0.01$). In table 9, one would expect to see the correlation between "*Age*" and "*Experience*". Surprisingly we

notice that there is not a statistical significance between age and experience, but there is a statistical significance between educational background and experience. Additionally, profession is also correlated with experience in using Design Patterns, which is normal since the participants were mostly B.Sc. and M.Sc. students and researchers who most probably use Design Patterns more frequently. A very important correlation regarding the experience in using Design Patterns as well as the knowledge of Design Patterns is the location in combination with the educational background. Most of the participants with Cypriot nationality have Computer Science background and most of the Greek participants have Software Engineering background. Therefore, it is possible to excuse this due to the difference in the methodologies used in teaching the Design Patterns in Computer Science and Computer Engineering schools and therefore this is reflected in the usage of Design Patterns in their practical usage.

10.3.2 Post-task questionnaire

After the execution of the given task, the participants were asked to rate how they perceived the given scenario, in order to reach into conclusions whether the task was easy and understandable. The participant asked to rate 5 questions regarding the given task. The means and standard deviations for each question are shown in Table 10. (See also appendix D.3)

The post-task results give an initial indication of how the users perceived the task and give the direction on how to proceed with the rest of the analysis taking into account the pre-test and post-test data results. For more clarity in the analysis of the results the mode values, the rate with the highest frequency for each question, is also depicted. From the mean and mode values it becomes obvious that the ease of the task has a mean value above the average rating. The mode values give us a hint on what was the actual tendency of how the users perceived the task. In questions Qa and Qb we notice that the average is 3.714 and the mode is 3. Having a standard

Table 10: Post-Task questions

Question	Mean	St. Deviation	Mode
Qa. Overall, I am satisfied with the ease of completing the task	3.714	1.013	3
Qb. Overall, I am satisfied with the amount of time it took to complete the task	3.250	1.076	3
Qc. Overall, I believe I learned new Design Patterns with the use of the software	3.286	0.089	4
Qd. Overall, I believe I learned where and how Design Patterns can be used	3.536	0.838	4
Qe. I believe I learned new things for Design Patterns	3.321	0.9049	4

deviation close to 1, denote the existence of responses that have some distance from the mean value and probably need further investigation. Responses for question Qa are dependent on the experience and the confidence of the users in the Software Design topic and the knowledge they have in Design Patterns. Therefore, an examination of the relation between the Qa and profession or experience will be done. Question Qb has to do with the availability of the users in time to execute the task. On the evaluation setup there was not a time limitation since the evaluation was done remotely. But the completion of the task by reading and understanding the content that is provided to the user for each particular Design Pattern needs time. To reach into safe conclusions regarding the time and what was the mean time of executing the tasks will be analyzed in the rest of this section through the analysis of the screen capturing videos that received by the users. By the execution of a single task it was not expected from the users to learn the Design Patterns but

the evaluation test was a mean to get familiar with the tool and identify its training character. The average mean value for Qe 3.324 and with mode value 4 is very encouraging but the high standard deviation attracts the attention for further analysis

To reach into more safe conclusions that will help analyzing the post-test results a grouping of the users based on their profession was made. Due to the small sample of users, for the analysis of the results, the users were grouped in students and professionals since the participants who declared profession other than students can be considered as professionals. With this grouping it is manageable to compare the mean values for the experience they, using the pre-test and the post-task questionnaire responses. The same comparison is also used for the post-test questionnaire and the evaluation of each particular question group that will be described for the post-test questionnaire responses.

The results comparing the means of the corresponding groups, students and professionals, regarding their experience in using similar tools and the experience they have in using Design Patterns in comparison to the post-task responses means are shown in Table 11.

Table 11: Pre-Test and Post-Task means comparison based on profession

Profession	**Value**	Pre Test-Q1	Pre Test-Q2	Pre Test-Q3	Pre Test-Q4	Qa	Qb	Qc	Qd	Qe
Students	Mean	1.833	2.500	2.417	2.333	3.750	3.250	3.417	3.583	3.500
	St.deviation	0.389	0.674	1.083	0.651	1.215	1.138	0.900	0.793	1.000
Professionals	Mean	1.687	3.063	3.188	2.688	3.688	3.250	3.188	3.500	3.188
	St.deviation	0.479	0.929	0.910	1.400	0.873	1.065	0.910	0.834	0.834

10.3.3 Post-test questionnaire

The post test questionnaire (see appendix D.4) was organized to have questions which would return results in regards to the *Usefulness, Satisfaction, Usability* and *the training of the Design Patterns through the recommendations*. The post-test analysis begins with the presentation of the results regarding the Usefulness of the prototype in table 12. From the results it is shown that the general perception for the ArchReco is positive and the majority of the users find the software Useful. From table 12 it is shown that the tool was not evaluated with high scores on questions related to its effectiveness in completing a task fast (Q1 and Q4) but it received high scores in questions such as Q2, Q5 and Q7 where the users identified the usefulness in understanding the Design Patterns and enhancement of the produced diagrams since the tool gives the possibility for additions to the diagram that goes beyond the existing knowledge in the topic. The low scores in the speed of completing tasks may be created for three reasons. The first is the unfamiliarity with the software and the completion of the task in the first use may caused delays, the second reason is because ArchReco as a training tool provides content for the Design patterns that need time to read and understand, and third in most of the cases professionals who have adequate background in Software Design and Design Patterns may prefer to apply their knowledge directly in a designed model instead of reading, learning and afterwards applying the Design Patterns into a Diagram. Taking into account the character of the tool and the latter considerations, it can be concluded that the prototype is generally perceived as useful first as a training tool but without the limitation to be used as a professional tool too. It is important to see how the perception of the usefulness of the tool was rated by the two groups of users, the Students and the professionals which will declare the aforementioned thoughts.

Table 12: Post-Test questions for Usefulness

Question	**Mean**	**St. Deviation**	**Mode**
Q1. Using the tool in Designing Software models would enable me to accomplish tasks more quickly	3.464	1.036	3
Q2. Using the tool would improve my understanding in using Design Patterns in a high level software design model	3.643	0.911	4
Q3. Using the tool in Designing Software models would increase my productivity	3.643	0.869	4
Q4. Using the tool to identify the most suitable Design Patterns would enhance my effectiveness on the job	1.714	0.713	2
Q5. Using the tool would make easier the process of Software Design	3.643	0.989	4
Q6. I would find the tool useful in Designing Software diagrams	3.929	0.940	4
Q7. The outcome of the tool would be beneficiary for the software developers who will implement the diagram into an actual application	3.786	0.7868	4

From table 13 it is noticeable that both groups have similar mean values in their responses. Small differences are noticed in questions Q4 and Q6. For Q4 the closeness in mean values from both groups shows that the low rating of the tool as a mean for completing task faster is most probably because of the recognition of the tool as training tool that takes time for studying the recommended content and not as a productive tool that will limit the time for completing tasks. At the same time both teams believe that ArchReco would be a useful tool for designing Software Design Models (Q6) with the highest mean coming from the Students group.

Table 13: Usefulness means comparison for Students and Professionals

Profession	**Value**	Q1	Q2	Q3	Q4	Q5	Q6	Q7
Students	Mean	3.417	3.583	3.583	1.833	3.833	4.000	3.833
	St.deviation	0.900	0.793	0.668	0.834	1.029	0.954	0.577
Professionals	Mean	3.500	3.687	3.687	1.625	3.500	3.875	3.750
	St.deviation	1.155	1.014	1.014	0.619	0.966	0.957	0.939

Table 14 presents the results that were collected in regards to the functionality supported by ArchReco and how the users rated the supported functionality. The most important results from table 14 are coming from two questions the Q10 and Q13. The users are describing their experience with ArchReco as positive while the context sensitive support is considered crucial. Moreover the users believe that ArchReco supports them in being more creative during the design process. In many cases during the design process users are searching the web for finding content and documentation for the creation of their models. The recommendations offered by ArchReco reduce the time of searching by giving the necessary knowledge material for well known design patterns within the environment and the users can decide how to use them in a creative way within the working environment of the software.

Table 14: Post-Test questions for Functionality

Question	Mean	St. Deviation	Mode
Q8. Archreco can effectively support the creation of High Level Software design model	3.750	0.645	4
Q9. ArchReco can effectively support the representation and management of Software Design components	3.643	0.869	4
Q10. The context-sensitive support (i.e. recommendations) is crucial to the ArchReco process	3.571	0.742	4
Q11. Using ArchReco tool supports me in being more creative during the design process	3.536	0.838	4
Q12. ArchReco tool enhances the outcome of the High Level diagram design	3.500	0.638	4
Q13. I describe my experience with ArchReco tool in general as positive	3.679	0.904	4

From the table 15 the results show that the users recognized the training value of the ArchReco prototype and the enhancement of the design process by the Context Aware Recommendations support. This is tracked on Question Q23 where almost all testers recognized the tool and the usage of the recommender system as educational. Comparing the mean values between students and professionals we notice that on Q23 there is an equality on their means (4.083 and 4.063

respectively). This shows that the tool is recognized as a training tool with training enhancements for the design process from all users experienced and not experienced. We noticed though, some unexpected results by comparing the means between Students and Professionals. The mean value for Students group (3.417) was lower than the Professionals (4.000) for question 15 where the same noticed for question Q20 where Students mean value was 3.083 and the Professionals mean rating value was 3.438. One would expect the opposite results. The most logical explanation on that result is maturity in recognizing new valuable knowledge due to the experience they have in working with similar tools. Moreover the students in most of the cases need more multimedia designs to learn something fast and less content to study. That was also part of the comments we received from some students that we were able to talk with. ArchReco though is a research prototype and these comments are tracked down to be taken into account fin future releases of the software.

Table 15: Post-Test questions for Design Patterns Training & Educational character of the Recommender system

Question	**Mean**	**St. Deviation**	**Mode**
Q14. AerchReco offers Stimulating possibilities to explore new Design Patterns	3.536	0.792	4
Q15. ArchReco helps in choosing useful Design patterns to apply in a Software Design diagram	3.750	0.799	4
Q16. I feel that I learned to work creatively using the ArchReco tool	3.571	0.836	4

Q17. I found ArchReco tool helpful to support us go over and over new Design Patterns until I found a suitable one to apply in my model	3.750	0.881	4
Q18. I found the recommendation of Design Patterns useful	3.750	0.881	4
Q19. The information provided for each pattern was sufficient	3.643	0.678	4
Q19. The information provided for each pattern was sufficient	3.643	0.678	4
Q20. Recommendations for Design Patterns helped me learn new patterns	3.286	0.976	4
Q21. Some of the Design Patterns are familiar to me	3.321	0.863	3
Q22. The items recommended to me are novel and interesting	3.357	0.731	4
Q23. The recommender system is educational	4.071	0.899	5
Q24. The recommendation System provides an adequate way for me to express my preferences	3.429	0.836	4
Q25. I became familiar with the recommendation system very quickly	3.536	0.999	4

10.3.4 Screen Capturing Videos - Results

From the 28 evaluation participants 11 of them sent their screen capturing videos for analysis. The videos that were not sent were mostly because they could not send the file due to their

large size, since the received files are approximately 28-30MB each. The received files though were thoroughly analysed in order to examine the actions of the users while they were executing the tasks. From the videos it was managed to confirm that the participants followed the task instructions and that they used both Context Aware recommendation algorithms to receive recommendations of Design Patterns and they used the recommendation results to take decisions for applying the recommendations in their diagram design models. The mean time for executing the tasks was 35.06 minutes with maximum 47 minutes and minimum 17 minutes. Each one of the participants, followed the instructions that were given by the evaluation web-page and they started by the configuration of the diagram that was asked to design. By the definition of the diagram type the architectural patterns were appearing on the right site of the software and from the videos we saw that almost all of them went through the descriptions of all web based architectural patterns description, even if the requested one was clearly mentioned in the task that was the MVC. They all used the add "Architectural pattern to canvas" button and they continued to the diagram design without further investigation of the Architectural Pattern Design. Then they started building the diagram using the palette shapes for each particular Architectural layer using the provided shapes from the palette and when the recommendations were appearing users were going through all the list of patterns that it was provided to them. Some of them were pressing the "*More*" button and they were continuing their study with more Design Patterns. From the videos it was possible to see the time that they were spending for reading the information for each Design Pattern. The mean time was calculated to be between 2-5 minutes per Design Pattern and in all videos it is shown that participants applied in their diagrams 2 (minimum) to 9 (maximum) Design Patterns. In most of the Design Patterns the testers were reading the descriptions provided in the small placeholder that by default presents the Pattern's data and for specific patterns they were opening the large window to read the content. In most of the cases the Patterns that were viewed in large window

were the ones that were selected to be included in the diagrams. For the UI related diagrams the image provided seemed to be enough for selection since they had the minimum time of reading before they add them in the diagram. Further analysis on the videos, also gave us some indications for improvements where most of them were related to the Interaction with the software and the User Experience. Some of the users were trying to drag the Design Patterns in the canvas which is currently not supported. The Recommended Patterns are currently presented in the form of links that the user has to select to see the information and their addition to the canvas is done with the use of a button. It was noticed that in some cases this was confusing for the users and it will be an improvement that will be changed in future releases. Also some attributes of specific Design Patterns like "*implementation*" or "*code example*" was not containing enough information and the users were reloading the specific info tab to get content. The lack of content for specific attributes of patterns happened because of the lack of this information from the source that was initially retrieved and therefor in cases like this it is better to hide the empty Design Pattern attributes when this is happening.

10.3.5 Evaluation of the Context Aware Recommendation Algorithms

The overall evaluation of the recommendation algorithms was done through the opinions collected by the users via the questionnaires and more specifically the results presented in table 15. The accuracy of the algorithms is not a representative measurement especially for software tools that the most important factor for evaluation is the subjective opinion of the users. Accuracy though remains an important factor to take into account. In the current release of the prototype the work focused on the usage of non-personalized content based recommendations. The reason for not proceeding into personalized recommendations at the current phase of the work was a decision that should be taken in order to focus this evaluation on the training impact of the prototype and

its design functionality and not confuse the users by asking them to provide ratings for the Design Patterns. Therefore, for the non-personalized Context Aware Recommendations at the current release of ArchReco prototype we used samples of input data that was collected by the evaluation participants and we used them as comparison dataset for the computation of Precision – Recall measurements for the two recommendation algorithms.

For the Text-Based algorithm the precision is estimated close to 60% and the recall is estimated to be 8.57%. the high precision is showing that there is a high percentage of relevant recommended Design Patterns but the selected (recall) value is not so high. The result varies based on the wording of the input text. The more descriptive is the text then the results are also more accurate. This is shown by the computation of the precision recall of the Utility based algorithm using four contextual factors. For the Utility based algorithm the algorithm run respectively and for every repetition on factor was removed from the calculation. The minimum number of factors that were used was two. The results received are: with four factors were p=100%, r=14.286% respectively, with three factors p=66.67% r= 11.429%, two factors p=33.33%, r=2.857%. It is obvious that the more contextual factors used in the computations the better results are received. Additionally, the weights for each particular factor influence the ranking of the results and therefore the recall value of the results.

10.4 Conclusions

The methodology used for the evaluation of ArchReco software prototype was based on the evaluation framework as it was defined by ResQue evaluation framework presented in [133]. The evaluation framework was used as a guideline for the collection of opinions regarding the usefulness, the supported functionality and the educational character of the context aware recommendation tool. The evaluation of the software was difficult to be performed in a controlled environment

due to the lack of an adequate number of users who could perform the tests. Therefore, the evaluation test performed remotely to specific user groups, students and researchers from Computer Science and Computer Engineeering University Departments in Greece and Cyprus.

The collected results were analyzed and described based on the pre-test questionnaires that evidenced the validity of the participants and the screen capturing videos that showed the design process in the way each user used ArchReco to complete the given task. In particular, through the demographic data it was confirmed that users who performed the evaluation test were Computer Science or Computer Engineering, students and researchers. Using the same data, the prior level of knowledge in Design Patterns and usage of similar Software design or modelling tools was also confirmed. Based on these data, knowledge level and experiences in using Design Patterns and similar tools created two groups of users, professionals and students. The two groups were used for the collection of qualitative results that were used to reach into safe conclusions, through comparisons of the responses to the pre and post-test (5-likert scale questions) questionnaires.

Additionally, the screen capturing videos were used to extract conclusions regarding the usage and the creative outcomes of the evaluation, through the observations of how each user interacted with the system. The observations were mainly based on the time needed to perform the given task, the time that each pattern was studied, whether the recommended design patterns were used and finally what was the final diagram result.

10.4.1 Results summary

The collection of the results in combination with the qualitative analysis that the current chapter presented, are summarized as follows:

- The development of such tools using Creativity principles and models is valid and at the same time important. Using the Creativity model providing creativity support tools such as

a graphical representation of diagrams, as well as the provision of existing knowledge in the form of recommendations can help and guide the users to create High Level Software diagrams independent to the prior knowledge on the Design Patterns but mostly based on the background they have in Software Requirements definition. Using the screen capturing videos during the evaluation sessions it was obvious that the selection of the shapes and the descriptions given by the users were valid requirements definitions based on the evaluation task that the users had to perform. The mean time of reading the recommended Design Patterns as well as the selection of patterns for the designed diagrams showed that users were helped on the creation of diagrams and at the same time on the selection of appropriate Design Patterns to use in the designed diagrams.

- ArchReco software is Useful. The ratings given by the users, professionals and students regarding the usefulness of the tool produced results higher than the average rating values. The results showed that the tool perceived as useful, and with added value for the training of Design Patterns. ArchReco supported functionality facilitates the learning and understanding of Design Patterns as well as the creative usage of the patterns.

- The Recommendation tool has educational character and its context sensitive functionality in combination with the semantic interoperability support saves time from searching Design Patterns. All participants perceived the recommendation tool as helpful and training tool. The participants denoted that they learned new Design Patterns using the ArchReco Software and in particular the Context Aware Recommendation System.

The usage of the creativity model and the computation of Design Patterns recommendations based on the creativity principles showed that the combination of Context Aware Recommendation tools

and Creativity can produce knowledge that can facilitate the design of High Level Software models. At the same time, it created an infrastructure for the creation of additional types of recommendations that can be created and measured in future experiments within a controlled environment i.e. laboratory sessions with specific creativity metrics. The results received by ArchReco evaluation confirmed the validity of the software as a training and supportive tool for the topic of Software Engineering topic. It proved that it can be used as a comparison tool for further investigations regarding Creativity measurements and the selection of more contextual factors that can enhance the creativity ability of the users who will use ArchReco or similar Software Design tools. Moreover, the small dataset of Design Patterns that was used to measure the accuracy of the non-personalized recommendations, based on precision and recall metrics, are motivating the development of personalized Context Aware Recommendation system for the improvement, not only the usefulness of the tool, but also the accuracy of the recommended Design Patterns, while designing Software models.

Chapter 11

Discussion - Future Research Challenges

This chapter concludes the thesis by summarizing its research contributions and by providing a discussion for the future work and the key research challenges.

11.1 Summary of contributions

The main goal of this thesis was the development of a software prototype that could be used by Computer Science or Computer Engineering students for the learning of the Design Patterns through practice. The aim was to provide a creative environment for the users which would provide to them context aware recommendations of design patterns so they would facilitated in learning the patterns, design a High Level Software diagram and at the same time stimuli their creativity during the design process. In this respect this thesis presented the procedure followed to reach into the result which was the design and development of the ArchReco software prototype that meets the requirements that were set for this thesis. Through the user based evaluation and the qualitative analysis of the results, ArchReco can be considered as the basis for further research. The research contributions of this thesis are summarized by chapter in the remainder of this section.

Chapter 2 presented the related to this thesis work. It examined the related work in regards to the topics that this thesis deals with. It revealed the lack of relevant work that combines the combination of the topics of Creativity, Context Awareness, Recommendation systems and their application in Software Engineering processes such as the recommendation of Design Patterns. Furthermore the related work showed that there are not Software Engineering education and training tools that use these topics and technologies in the form of a complete solution like ArchReco prototype.

Chapter 3 presented the literature with respect to context awareness, the methods and the tools for representing the context. The chapter went through the most well known methods, tools and frameworks that are used for modeling and applying the context. It provided a set of methodologies and techniques from which the most suitable to the current research were selected and applied. Based on the methods of modeling the context the Semantic representation with the use of Ontologies was selected. The examination of the frameworks that are commonly used for the representation of context led to the selection of Jena framework and the use of Protege tool that were used for defining and representing the models of this thesis.

Chapter 4 presented the Recommendation Systems, the types of recommendation systems and some of the most known recommendation tools and algorithms. The chapter helped in specifying the type of recommendations that ArchReco should use, for the time being, the non personalized content based recommendations. With the selection of the type of recommendations ArchReco should provide the algorithms and methods were also provided and defined through the analysis of this chapter. The analysis of the filtering techniques and the importance of analyzing the context highlighted the TF-IDF algorithm and the content based filtering methods that were used for the pre-filtering and post-filtering methods that were applied in chapter 8.

Chapter 5 surveyed the literature in regards to the creativity models and the several creativity models as they were defined through the years. It also presented the examination of some of the most known creativity support tools in regards to the possibility of support of any kind of context aware recommendations. The chapter presented the work that was done for modeling the creativity and the work that was done for the development of a generic Context Aware Recommendation mechanism for creativity. The chapter showed that the current, most known, Creativity Support tools lack of recommendation mechanisms. It also used for the definition of the most important contextual entities of creativity that were used for the composition of Creativity contextual model. The outcomes of the creativity analysis led to the development of a generic recommendation tool, based on the Creativity contextual model, which was used as reference for the development of the Context Aware recommendation tool for the ArchReco prototype.

Chapter 6 linked the theory of the first chapters with the applied work that would follow. It presented the user based survey that was performed for the identification of the Software Engineering needs in regards, to recommendations support and the ease of process creativity with the use of existing Software Modelling tools and processes. It showed that the current tools do not support the recommendation mechanisms not only for educational but also for the commercial usage of thees tools. From the participants responses it was concluded that recommendation mechanisms would enhance the existing tools and processes, which contributed in the decision for implementing the ArchReco software for the support of fresh Engineers in learning the Design Patterns within a creative environment. The survey framed the wide range of creativity scope, into the specific need for a specific creative problem related to the Software Design process.

Chapter 7 used the creativity conceptual model of the chapter 5 and in combination with the findings of chapter 6 it presented an extended conceptual model which included the Software

Engineering process concepts. It defined the model that would later be used for the development of the Context Aware Recommendation tool for the recommendation of Design Patterns.

Chapter 8 presented the work that has been done for the development of the Context Aware Recommendation for Design Patterns. It explained the the post and pre-filtering techniques that were used in combination to the Semantic Web technologies. It elaborated the techniques and the algorithms that were used and it depicted a set of examples of how the recommendation system takes into account the context with the use of semantics in order to produce recommendations of Design Patterns, coming from more than one data repository.

Chapter 9 presented the ArchReco software prototype as a complete solution including the Context Aware recommendation tool from chapter 8. It presented how the prototype can be used providing its usefulness as an educational and training tool for Engineering students, it presented the overall system architecture and it described the user interface and the functions it supports. This chapter consists of the integration of the overall work presented in this thesis and it describes how the ArchReco software prototype meets the objectives that were set by the thesis statement.

Chapter 10 presented the ArchReco user based evaluation setup and execution. The results collected by the evaluation are analyzed and presented using a qualitative results analysis. With the evaluation analysis the usefulness, usability, and educational character of the tool in combination with the Context Aware Recommendation tool were extracted. The results become an evidence of the validity of the work presented in this thesis and they state the need for further research in the topic and opens the route for future work.

The achievements and contributions of this thesis are summarized in the following list:

- Identification of the Creativity most important entities based on the various creativity models literature review.

- Design of the Creativity contextual model and representation of the model as an Ontology.
- Development of a generic context aware recommendation tool, a first attempt for producing recommendations for a Creative process.
- Identification of the needs of Software Design process (in terms of creativity) by surveying professionals in the topic of Software Design.
- Adjustment of the generic creativity ontology model to the Software Design process and design of a new contextual model for Software Design as a Creative process model.
- Design of a Design Patterns Context Aware Recommendation System, which accepts as input physical language text, analyzes the context of a working problem and provides recommendations of Design Patterns.
- Design and development of a Semantic Interoperability mechanism which can retrieve Design Patterns from multiple repositories using the created ontology models.
- Design and development of the ArchReco prototype, a solution that integrates the aforementioned outcomes into a complete solution. The prototype consists of a complete Software Design tool that can be used for the education and training of CS or SE students for learning the Deign Patterns within a creative context.

11.2 Future work

The current version of Archreco Software supports the provision of non-personalized content based context aware recommendations of Design Patterns. It is important for the future to also examine the personalized context aware recommendations of Design Patterns. From the evaluation

that was performed the users perceived in a positive way the tool and its supported recommendations, which is a hint for improving the recommendations and provide more personalized ones. The personalized recommendations presume the inclusion of the "rating" factor for each Design Pattern. The challenge here is tracked n the fact that a Design Pattern can be recommended several times for different types of diagrams and for different context and the user may rate the recommended pattern with a different value every time it is recommended. Taking that into consideration the focus of a future evaluation moves from the usefulness, and satisfaction of a set of recommendations, to the accuracy of recommendations. Therefore, in the future personalization must be included in the overall recommendation mechanism and using the ratings of the users attempt different kinds of personalized algorithms for the computation of personalized context aware recommendations. Moreover the use of personalized recommendations for different types of context opens new challenges in relation to the optimal contextual elements that can be used by the recommendation mechanism. For example a Design Pattern was rated by 5 for diagram A and by 2 for diagram B. What was the difference between the two? Is there any rating pattern in regards to the context or the content? Part of the personalization task is work that is began and it is work in progress. A personalization component is built where the users are able to provide personal data, diagrams they created and Design Patterns they used for each diagram. The component is still under development and will soon be integrated with ArchReco software.

In addition to the above, for the future it is planned to create related taxonomies and thesaurus to be used for the content analysis during the semantic filtering of the users input. The input required from the users is free text in most of the cases and the input is processed through indexing and TF-IDF analysis. The existence of taxonomies or thesaurus containing domain specific words, it is assumed that would improve the accuracy of the recommendations.

Furthermore, future work will also be focused on the creation and delivery of more recommendation types other than the Design Patterns like for example users that can take responsibility of a task, group composition and supporting reading material like documents or research papers.

Finally, as future work it is planned to improve the appearance of the software and more specifically it is planned to transfer to software into an on-line version. ArchReco was build as a desktop application for the aims of this research and used as a proof of concept. The tool will be more usable and easily accessible in the case of an on-line version which will support all the aforementioned objectives.

11.3 Research Challenges

Using the future work objectives of the previous section as the basis for the research challenges, the remaining section will approach the possible challenges in two axes: *Algorithmic* and *Applications*. The research challenges are identified based on the current work and the possibility of using the current software prototype for further investigations.

11.3.1 Algorithmic

For the algorithmic the following research topics can be further investigated and examined:

- Examination of more recommendation algorithms for the content based recommendations taking into account the contextual information of entities for a given problem. That means the definition of the context and analysis of methods that prove that the selected contextual information is trustfully valuable for the recommendations.

- Examination of using Genetic Algorithms and Evolutionary programming for the optimization of the Utility Function in terms of the weights of the contextual elements that participate in the Utility function equation.

- Examination of using Matching Algorithms, a research topic that belongs to the Game Theory area of research, for generating recommendations based on voting between users within a working group. It can be considered as a transformation of the existing methodology that this thesis currently used, with the traditional Recommendation Algorithms, into gamification methods and comparison between the two approaches.

11.3.2 Applications

- User Experience and UI principles is an active research topic in Recommendation Systems. The combination of Recommendation Systems with Creativity can be further investigated in terms of the User Interfaces design, Recommendations Delivery (Information provided by the recommendation systems) and the interaction between user and Recommendation System.

- Examination of using Context Aware Recommendation algorithms and the Creativity Models in other domains such as Multimedia design and Graphics.

- Cloud-based applications is one of the most used research topics during the last few years. Data collection and data transformation for the usage with Data Mining algorithms is a challenging area for further investigation. The usage of Archreco or similar tool and by taking advantage of the collected data in the cloud can create a huge repository for Design Patterns, their usage, and by taking advantage of the users preferences based on their contextual information, apply algorithms for domain specific recommendation algorithms,

training data sets or Computational Intelligence algorithms that would be applied in related Software Engineering research topics.

- IoT applications is an other commonly used topic of the last years research. One of the challenges that IoT applications deal today is the existence of several communication protocols between the "Things" and the "Cloud". The challenge is mainly located on the lack of a common communication language between the two layers and thus the difficulty in transferring data from a physical network to multiple cloud applications. The semantic interoperability service can be used for this transformation and create a common communication language through the convergence of the transferred messages through a unified ontology model. The applicability of such mechanism can be accompanied with a tool like ArchReco software that through the graphic representation of a communication network topology can provide Design Patterns recommendations for the implementation of the communication protocols through a semantic web layer in order to reach specific cloud-based applications.

- A challenging approach for further research would be the examination of emotions as contextual factors for the production of recommendations based on the user's emotions. The design of emotions based model in combination with the creativity model and the development of emotions-based Context Aware recommendation system is challenging task that can be further investigated.

Bibliography

[1] A. Abbas, L. Zhang, and S. U. Khan, "A survey on context-aware recommender systems based on computational intelligence techniques," *Computing*, vol. 97, no. 7, pp. 667–690, 2015.

[2] G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton, "Cyberguide: A mobile context-aware tour guide," *Wireless networks*, vol. 3, no. 5, pp. 421–433, 1997.

[3] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 6, pp. 734–749, June 2005.

[4] A. Agostini, C. Bettini, and D. Riboni, "Experience report: ontological reasoning for context-aware internet services," 2006.

[5] C. Alexander, S. Ishikawa, and M. Silverstein, *A Pattern Language: Towns, Buildings, Construction*, ser. Center for Environmental Structure Berkeley, Calif: Center for Environmental Structure series. OUP USA, 1977. [Online]. Available: https://books.google.com.cy/books?id=hwAHmktpk5IC

[6] F. A. Asnicar and C. Tasso, “ifweb: a prototype of user model-based intelligent agent for document filtering and navigation in the world wide web,” in *Sixth International Conference on User Modeling*, 1997, pp. 2–5.

[7] A. Asthana, M. Cravatts, and P. Krzyzanowski, “An indoor wireless system for personalized shopping assistance,” in *Mobile Computing Systems and Applications, 1994. Proceedings., Workshop on*. IEEE, 1994, pp. 69–74.

[8] C. J. Atman, J. Turns, M. Cardella, and R. S. Adams, “The design processes of engineering educators: Thick descriptions and potential implications,” in *Expertise in Design: Design Thinking Research Symposium*, vol. 6. Citeseer, 2003.

[9] M. A. Awais, “Requirements prioritization: Challenges and techniques for quality software development,” *Advances in Computer Science: an International Journal*, vol. 5, no. 2, pp. 14–21, 2016.

[10] J. Bagga and A. Heinz, in *Graph Drawing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2002, vol. 2265, pp. 459–460. [Online]. Available: http://dx.doi.org/10.1007/3-540-45848-4_45

[11] M. Balabanović and Y. Shoham, “Fab: Content-based, collaborative recommendation,” *Commun. ACM*, vol. 40, no. 3, pp. 66–72, Mar. 1997. [Online]. Available: http://doi.acm.org/10.1145/245108.245124

[12] S. Balaji and M. S. Murugaiyan, “Waterfall vs. v-model vs. agile: A comparative study on sdlc,” *International Journal of Information Technology and Business Management*, vol. 2, no. 1, pp. 26–30, 2012.

[13] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, no. 4, pp. 263–277, 2007.

[14] J. E. Bardram, "The java context awareness framework (jcaf)–a service infrastructure and programming framework for context-aware applications," in *Pervasive computing*. Springer, 2005, pp. 98–115.

[15] C. Basu, H. Hirsh, and W. Cohen, "Recommendation as classification: Using social and content-based information in recommendation," in *In Proceedings of the Fifteenth National Conference on Artificial Intelligence*. AAAI Press, 1998, pp. 714–720.

[16] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries *et al.*, "Manifesto for agile software development," 2001.

[17] G. Beham, B. Kump, T. Ley, and S. Lindstaedt, "Recommending knowledgeable people in a work-integrated learning system," *Procedia Computer Science*, vol. 1, no. 2, pp. 2783–2792, 2010.

[18] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*. Springer, 2009, pp. 1–4.

[19] S. Berkovsky, J. Freyne, and G. Smith, "Personalized network updates: Increasing social interactions and contributions in social networks," in *User Modeling, Adaptation, and Personalization*, ser. Lecture Notes in Computer Science, J. Masthoff, B. Mobasher, M. Desmarais, and R. Nkambou, Eds. Springer Berlin Heidelberg, 2012, vol. 7379, pp. 1–13. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31454-4_1

[20] C. Blank, H. Eveking, J. Levihn, and G. Ritter, "Symbolic simulation techniques-state-of-the-art and applications," in *hldvt*. IEEE, 2001, p. 45.

[21] B. W. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, May 1988.

[22] B. Boehm, "A view of 20th and 21st century software engineering," in *Proceedings of the 28th International Conference on Software Engineering*, ser. ICSE '06. New York, NY, USA: ACM, 2006, pp. 12–29. [Online]. Available: http://doi.acm.org/10.1145/1134285.1134288

[23] E. P. Bontas, "Context representation and usage for the semantic web: A state of the art," Technical Report B-04-30, 2004.

[24] P. Bottoni, E. Guerra, and J. de Lara, "Formal foundation for pattern-based modelling," in *Fundamental Approaches to Software Engineering*, ser. Lecture Notes in Computer Science, M. Chechik and M. Wirsing, Eds. Springer Berlin Heidelberg, 2009, vol. 5503, pp. 278–293. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-00593-0_19

[25] L. C. Briand, S. Morasca, and V. R. Basili, "Defining and validating measures for object-based high-level design," *IEEE Trans. Softw. Eng.*, vol. 25, no. 5, pp. 722–743, Sep. 1999. [Online]. Available: http://dx.doi.org/10.1109/32.815329

[26] S. Brinkkemper, "Method engineering: engineering of information systems development methods and tools," *Information and software technology*, vol. 38, no. 4, pp. 275–280, 1996.

[27] P. J. Brown, J. D. Bovey, and X. Chen, "Context-aware applications: from the laboratory to the marketplace," *Personal Communications, IEEE*, vol. 4, no. 5, pp. 58–64, 1997.

[28] P. Brusilovsky, "Developing adaptive educational hypermedia systems: From design models to authoring tools," in *Authoring tools for advanced technology Learning Environments*. Springer, 2003, pp. 377–409.

[29] P. Brusilovsky, L. Pesin, and M. Zyryanov, "Towards an adaptive hypermedia component for an intelligent learning environment," in *Human-computer interaction*. Springer, 1993, pp. 348–358.

[30] A. Burkle, W. Muller, U. Pfirrmann, M. Schenk, N. Dimakis, J. Soldatos, and L. Polymenakos, "An agent-based architecture for context-aware services supporting human interaction," in *Proceedings of the 2006 IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology*. IEEE Computer Society, 2006, pp. 146–152.

[31] P. Castells and S. Vargas, "Novelty and diversity metrics for recommender systems: Choice, discovery and relevance," in *In Proceedings of International Workshop on Diversity in Document Retrieval (DDR)*, pp. 29–37.

[32] C.-T. Chen, Y. C. Cheng, and C.-Y. Hsieh, "Towards a pattern language approach to establishing personal authoring environments in e-learning," in *Proceedings of the Sixth Conference on IASTED International Conference Web-Based Education - Volume 2*, ser. WBED'07. Anaheim, CA, USA: ACTA Press, 2007, pp. 13–18. [Online]. Available: http://dl.acm.org/citation.cfm?id=1323159.1323162

[33] G. Chen, D. Kotz *et al.*, "A survey of context-aware mobile computing research," Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, Tech. Rep., 2000.

[34] G. Chen, M. Li, and D. Kotz, "Design and implementation of a large-scale context fusion network," in *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*. IEEE, 2004, pp. 246–255.

[35] J. P. Chin, V. A. Diehl, and K. L. Norman, "Development of an instrument measuring user satisfaction of the human-computer interface," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '88. New York, NY, USA: ACM, 1988, pp. 213–218. [Online]. Available: http://doi.acm.org/10.1145/57167.57203

[36] C.-H. Chuan and E. Chew, "Evaluating and visualizing effectiveness of style emulation in musical accompaniment." in *ISMIR*, 2008, pp. 57–62.

[37] V. Codina, F. Ricci, and L. Ceccaroni, "Semantically-enhanced pre-filtering for context-aware recommender systems," in *Proceedings of the 3rd Workshop on Context-awareness in Retrieval and Recommendation*. ACM, 2013, pp. 15–18.

[38] V. Codina, F. Ricci, and L. Ceccaroni, "Distributional semantic pre-filtering in context-aware recommender systems," *User Modeling and User-Adapted Interaction*, vol. 26, no. 1, pp. 1–32, 2016.

[39] J. D. Couger, *Creative problem solving and opportunity finding*. boyd & fraser publishing company, 1995.

[40] S. Crago, J. McMahon, C. Archer, K. Asanovic, R. Chaung, K. Goolsbey, M. Hall, C. Kozyrakis, K. Olukotun, U. OReilly *et al.*, "Cearch: Cognition enabled architecture," in *Proceedings of the Tenth Annual High Performance Embedded Computing Workshop, Lexington, MA*, 2006.

[41] C. Csallner, N. Tillmann, and Y. Smaragdakis, "Dysy: Dynamic symbolic execution for invariant inference," in *Proceedings of the 30th international conference on Software engineering*. ACM, 2008, pp. 281–290.

[42] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS Q.*, vol. 13, no. 3, pp. 319–340, Sep. 1989. [Online]. Available: http://dx.doi.org/10.2307/249008

[43] R. L. De Mantaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M. L. Maher, M. T COX, K. Forbus *et al.*, "Retrieval, reuse, revision and retention in case-based reasoning," *The Knowledge Engineering Review*, vol. 20, no. 03, pp. 215–240, 2005.

[44] F. H. del Olmo and E. Gaudioso, "Evaluation of recommender systems: A new approach," *Expert Systems with Applications*, vol. 35, no. 3, pp. 790–804, 2008.

[45] M. Derntl, K. Hummel *et al.*, "Modeling context-aware e-learning scenarios," in *Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on*. IEEE, 2005, pp. 337–342.

[46] A. K. Dey, G. D. Abowd, and D. Salber, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications," *Hum.-Comput. Interact.*, vol. 16, no. 2, pp. 97–166, Dec. 2001. [Online]. Available: http://dx.doi.org/10.1207/S15327051HCI16234_02

[47] J. Dietrich, "The mandarax manual," *Institute of Information Sciences & Technology, Massey University, New Zealand*, 2003.

[48] E. A. Domeshek, J. L. Kolodner, and C. M. Zimring, "The design of a tool kit for case-based design aids." Springer, 1994, pp. 109–126.

[49] J. Dong, Y. Sheng, and K. Zhang, "Visualizing design patterns in their applications and compositions," *Software Engineering, IEEE Transactions on*, vol. 33, no. 7, pp. 433–453, July 2007.

[50] J. Dong, Y. Zhao, and Y. Sun, "A matrix-based approach to recovering design patterns," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 39, no. 6, pp. 1271–1282, Nov 2009.

[51] J. Durkin and J. Durkin, *Expert systems: design and development*. Prentice Hall PTR, 1998.

[52] D. Ejigu, M. Scuturici, and L. Brunie, "Hybrid approach to collaborative context-aware service platform for pervasive computing," *Journal of computers*, vol. 3, no. 1, pp. 40–50, 2008.

[53] M. Elahi, "Context-aware intelligent recommender system," in *Proceedings of the 15th international conference on Intelligent user interfaces*. ACM, 2010, pp. 407–408.

[54] W. Etienne, "Artificial intelligence and tutoring systems," 1987.

[55] R. Farenhorst and R. de Boer, "Knowledge management in software architecture: State of the art," in *Software Architecture Knowledge Management*, M. Ali Babar, T. Dingsøyr, P. Lago, and H. van Vliet, Eds. Springer Berlin Heidelberg, 2009, pp. 21–38. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02374-3_2

[56] J. Favela, A. I. Martinez-Garcia *et al.*, "Context-aware mobile communication in hospitals," *Computer*, no. 9, pp. 38–46, 2003.

[57] E. Friedman-Hill *et al.*, "Jess, the rule engine for the java platform," 2008.

[58] S. I. Gallant, *Neural network learning and expert systems*. MIT press, 1993.

[59] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-oriented Software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[60] D. Gaševic, D. Djuric, and V. Devedžic, *Model driven architecture and ontology development*. Springer Science & Business Media, 2006.

[61] M. Gasparic and A. Janes, "What recommendation systems for software engineering recommend: A systematic literature review," *Journal of Systems and Software*, vol. 113, pp. 101 – 113, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121215002605

[62] A. Z. Ghalwash, "A recency inference engine for connectionist knowledge bases," *Applied Intelligence*, vol. 9, no. 3, pp. 201–215, 1998.

[63] C. Ghezzi, M. Jazayeri, and D. Mandrioli, *Fundamentals of Software Engineering*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2002.

[64] A. R. Golding and P. S. Rosenbloom, "Improving accuracy by combining rule-based and case-based reasoning," *Artificial Intelligence*, vol. 87, no. 1, pp. 215–254, 1996.

[65] P. Gomes, F. Pereira, P. Paiva, N. Seco, P. Carreiro, J. Ferreira, and C. Bento, "Using cbr for automation of software design patterns," in *Advances in Case-Based Reasoning*, ser. Lecture Notes in Computer Science, S. Craw and A. Preece, Eds. Springer Berlin Heidelberg, 2002, vol. 2416, pp. 534–548. [Online]. Available: http://dx.doi.org/10.1007/3-540-46119-1_39

[66] P. Grube and K. Schmid, "idspace d2. 1 state of the art in tools for creativity," 2008.

[67] T. R. Gruber, “Toward principles for the design of ontologies used for knowledge sharing?” *International Journal of Human-Computer Studies*, vol. 43, no. 5â6, pp. 907 – 928, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1071581985710816

[68] T. Gu, H. K. Pung, and D. Q. Zhang, “A middleware for building context-aware mobile services,” in *Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th*, vol. 5. IEEE, 2004, pp. 2656–2660.

[69] Y.-G. Gueheneuc and G. Antoniol, “Demima: A multilayered approach for design pattern identification,” *Software Engineering, IEEE Transactions on*, vol. 34, no. 5, pp. 667–684, Sept 2008.

[70] Y.-G. Gueheneuc, Y.-G.heneuc and R. Mustapha, “A simple recommender system for design patterns.”

[71] V. Haarslev, K. Hidde, R. Möller, and M. Wessel, “The racerpro knowledge representation and reasoning system.” *Semantic Web*, vol. 3, no. 3, pp. 267–277, 2012.

[72] N. Harrison, P. Avgeriou, and U. Zdun, “Using patterns to capture architectural decisions,” *Software, IEEE*, vol. 24, no. 4, pp. 38–45, July 2007.

[73] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster, “The anatomy of a context-aware application,” *Wireless Networks*, vol. 8, no. 2/3, pp. 187–197, 2002.

[74] M. Hartmann and G. Austaller, “Context models and context awareness,” in *Handbook of Research on Ubiquitous Computing Technology for Real Time Enterprises*, 2008, pp. 235–256. [Online]. Available: http://www.igi-global.com/Bookstore/Chapter.aspx?TitleId=21771

[75] A. Hatzigaidas, A. Papastergiou, G. Tryfon, and D. Maritsa, "Topic map existing tools: a brief review," in *ICTAMI 2004 (International Conference on Theory and Applications of Mathematics and Informatics)*, 2004.

[76] C. Hayes, P. Massa, P. Avesani, and P. Cunningham, "An on-line evaluation framework for recommender systems," in *In Workshop on Personalization and Recommendation in E-Commerce (Malaga*. Springer Verlag, 2002.

[77] J. Heflin and J. Hendler, "Dynamic ontologies on the web," in *AAAI/IAAI*, 2000, pp. 443–449.

[78] K. Henricksen, J. Indulska, and A. Rakotonirainy, "Generating context management infrastructure from high-level context models," in *In 4th International Conference on Mobile Data Management (MDM)-Industrial Track*. Citeseer, 2003.

[79] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 5–53, 2004.

[80] J. Holt, *UML for Systems Engineering: watching the wheels*. IET, 2004, vol. 4.

[81] D. C. Howe, "Rita: creativity support for computational literature," in *Proceedings of the seventh ACM conference on Creativity and cognition*. ACM, 2009, pp. 205–210.

[82] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender systems: an introduction*. Cambridge University Press, 2010.

[83] K. Järvelin and J. Kekäläinen, "Ir evaluation methods for retrieving highly relevant documents," in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2000, pp. 41–48.

[84] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.

[85] A. Jena, "A free and open source java framework for building semantic web and linked data applications," *Available online: jena. apache. org/(accessed on 28 April 2015)*, 2015.

[86] K. Kakousis, N. Paspallis, and G. A. Papadopoulos, "Optimizing the utility function-based self-adaptive behavior of context-aware systems using user feedback," in *On the move to meaningful internet systems: OTM 2008*. Springer, 2008, pp. 657–674.

[87] A. Karapidis, A. Kienle, and H. Schneider, "Creativity, learning and knowledge management in the process of service development–results from a survey of experts," in *Proceedings of I-Know*, 2005, pp. 432–440.

[88] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[89] J. Kjeldskov and M. B. Skov, "Supporting work activities in healthcare by mobile electronic patient records," in *Computer Human Interaction*. Springer, 2004, pp. 191–200.

[90] B. P. Knijnenburg, M. C. Willemsen, Z. Gantner, H. Soncu, and C. Newell, "Explaining the user experience of recommender systems," *User Modeling and User-Adapted Interaction*, vol. 22, no. 4-5, pp. 441–504, Oct. 2012. [Online]. Available: http://dx.doi.org/10.1007/s11257-011-9118-4

[91] J. Kulik, "Fast and flexible forwarding for internet subscription systems," in *Proceedings of the 2nd international workshop on Distributed event-based systems*. ACM, 2003, pp. 1–8.

[92] N. D. Lane, Y. Xu, H. Lu, A. T. Campbell, T. Choudhury, and S. B. Eisenman, "Exploiting social networks for large-scale human behavior modeling," *IEEE Pervasive Computing*, vol. 10, no. 4, pp. 45–53, 2011.

[93] G. Lee, T. Naganuma, and S. Kurakake, "Efficient matching in a context-aware event notification system for mobile users," *Proceedings Distributed Event-Based Systems*, 2005.

[94] J. R. Lewis, "Ibm computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use," *Int. J. Hum.-Comput. Interact.*, vol. 7, no. 1, pp. 57–78, Jan. 1995. [Online]. Available: http://dx.doi.org/10.1080/10447319509526110

[95] H. Lieberman *et al.*, "Letizia: An agent that assists web browsing," *IJCAI (1)*, vol. 1995, pp. 924–929, 1995.

[96] P. Lops, M. de Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Springer US, 2011, pp. 73–105. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-85820-3_3

[97] T. I. Lubart, "Models of the creative process: Past, present and future," *Creativity Research Journal*, vol. 13, no. 3-4, pp. 295–308, 2001. [Online]. Available: http://dx.doi.org/10.1207/S15326934CRJ1334_07

[98] A. D. Lucia, V. Deufemia, C. Gravino, and M. Risi, "Design pattern recovery through visual language parsing and source code analysis," *J. Syst. Softw.*, vol. 82, no. 7, pp. 1177–1193, Jul. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2009.02.012

[99] A. M. Lund, "Measuring usability with the use questionnaire," *Usability interface*, vol. 8, no. 2, pp. 3–6, 2001.

[100] K. M., Accessed: 2016. [Online]. Available: http://pubs.acs.org/subscribe/archive/ci/31/i11/html/11hipple_box3.ci.html

[101] M. L. Maher and A. G. de Silva Garza, "Developing case-based reasoning for structural design," *IEEE Intelligent Systems*, no. 3, pp. 42–52, 1996.

[102] M. L. Maher and A. G. de Silva Garza, "Case-based reasoning in design," *IEEE Intelligent Systems*, no. 2, pp. 34–41, 1997.

[103] S. Mancoridis, R. C. Holt, and M. W. Godfrey, "Tool support for software engineering education," Department of Computer Science, University of Toronto, Tech. Rep., 1994.

[104] C. D. Manning, P. Raghavan, H. Schütze *et al.*, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1.

[105] H. Marmanis and D. Babenko, *Algorithms of the intelligent web*. Manning Greenwich, 2009.

[106] B. E. Mastenbrook and E. G. Berkowitz, "Representing symbolic reasoning." in *MAICS*. Citeseer, 2003, pp. 96–101.

[107] M. McCandless, E. Hatcher, and O. Gospodnetic, *Lucene in Action, Second Edition: Covers Apache Lucene 3.0*. Greenwich, CT, USA: Manning Publications Co., 2010.

[108] J. McCarthy, "Concepts of logical ai," in *Logic-based artificial intelligence*. Springer, 2000, pp. 37–56.

[109] P. K. McKinley, S. M. Sadjadi, E. P. Kasten, and B. H. Cheng, "A taxonomy of compositional adaptation," *Rapport Technique numéroMSU-CSE-04-17*, 2004.

[110] S. M. McNee, J. Riedl, and J. A. Konstan, "Being accurate is not enough: How accuracy metrics have hurt recommender systems," in *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '06. New York, NY, USA: ACM, 2006, pp. 1097–1101. [Online]. Available: http://doi.acm.org/10.1145/1125451.1125659

[111] S. Mitchell, M. D. Spiteri, J. Bates, and G. Coulouris, "Context-aware multimedia computing in the intelligent hospital," in *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*. ACM, 2000, pp. 13–18.

[112] D. Mladenic and B. A. P. Webwatcher, "Machine learning used by personal webwatcher," 1999.

[113] A. Mouasher and J. M. Lodge, "The search for pedagogical dynamism–design patterns and the unselfconscious process," *Educational Technology & Society*, vol. 19, no. 2, pp. 274–285, 2016.

[114] A. Moukas, "Amalthaea information discovery and filtering using a multiagent evolving ecosystem," *Applied Artificial Intelligence*, vol. 11, no. 5, pp. 437–457, 1997.

[115] N. M. A. Munassar and A. Govardhan, "A comparison between five models of software engineering," *IJCSI*, vol. 5, pp. 95–101, 2010.

[116] K. Murphy, "A brief introduction to graphical models and bayesian networks," http://www.cs.ubc.ca/ murphyk/Bayes/bayes.html, 1998, Último acesso: 08/12/05.

[117] K. Nakakoji, Y. Yamamoto, M. Akaishi, and K. Hori, "Interaction design for scholarly writing: Hypertext representations as a means for creative knowledge work," *New Review of Hypermedia and Multimedia*, vol. 11, no. 1, pp. 39–67, 2005.

[118] E. O. Navarro and A. van der Hoek, "Comprehensive evaluation of an educational software engineering simulation environment," in *Software Engineering Education Training, 2007. CSEET '07. 20th Conference on*, July 2007, pp. 195–202.

[119] P. Nurmi, M. Przybilski, G. Lindén, and P. Floréen, "A framework for distributed activity recognition in ubiquitous systems." in *IC-AI*, 2005, pp. 650–655.

[120] D. O'Doherty, S. Jouili, and P. Van Roy, "Towards trust inference from bipartite social networks," in *Proceedings of the 2nd ACM SIGMOD Workshop on Databases and Social Networks*. ACM, 2012, pp. 13–18.

[121] J. O'Donovan and B. Smyth, "Is trust robust?: an analysis of trust-based recommendation," in *Proceedings of the 11th international conference on Intelligent user interfaces*. ACM, 2006, pp. 101–108.

[122] S. Owen, R. Anil, T. Dunning, and E. Friedman, *Mahout in action*. Manning Shelter Island, 2011.

[123] A. Padovitz, S. W. Loke, A. Zaslavsky, and B. Burg, "Towards a general approach for reasoning about context, situations and uncertainty in ubiquitous sensing: Putting geometrical intuitions to work," in *2nd International Symposium on Ubiquitous Computing Systems (UCS'04), Tokyo, Japan*, 2004.

[124] F. Palma, H. Farzin, Y.-G. Guéhéneuc, and N. Moha, "Recommendation system for design patterns in software development: An dpr overview," in *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*, ser. RSSE '12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 1–5. [Online]. Available: http://dl.acm.org/citation.cfm?id=2666719.2666720

[125] H.-S. Park, J.-O. Yoo, and S.-B. Cho, "A context-aware music recommendation system using fuzzy bayesian networks with utility theory," in *Fuzzy systems and knowledge discovery*. Springer, 2006, pp. 970–979.

[126] J. Park, S. Hunting, and D. C. Foreword By-Engelbart, *XML Topic Maps: creating and using topic maps for the Web*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[127] D. Parra and S. Sahebi, "Recommender systems: Sources of knowledge and evaluation metrics," in *Advanced Techniques in Web Intelligence-2*. Springer, 2013, pp. 149–175.

[128] J. Pascoe, "Adding generic contextual capabilities to wearable computers," in *Wearable Computers, 1998. Digest of Papers. Second International Symposium on*. IEEE, 1998, pp. 92–99.

[129] M. Pazzani and D. Billsus, "Learning and revising user profiles: The identification of interesting web sites," *Machine learning*, vol. 27, no. 3, pp. 313–331, 1997.

[130] M. J. Pazzani, J. Muramatsu, D. Billsus *et al.*, "Syskill & webert: Identifying interesting web sites," in *AAAI/IAAI, Vol. 1*, 1996, pp. 54–61.

[131] J. A. Plucker and R. A. Beghetto, "Why creativity is domain general, why it looks domain specific, and why the distinction does not matter." 2004.

[132] J. Prentzas and I. Hatzilygeroudis, "Integrating hybrid rule-based with case-based reasoning," in *Advances in case-based reasoning*. Springer, 2002, pp. 336–349.

[133] P. Pu, L. Chen, and R. Hu, "A user-centric evaluation framework for recommender systems," in *Proceedings of the Fifth ACM Conference on Recommender Systems*, ser. RecSys '11. New York, NY, USA: ACM, 2011, pp. 157–164. [Online]. Available: http://doi.acm.org/10.1145/2043932.2043962

[134] G. Rasool and D. Streitferdt, "A survey on design pattern recovery techniques," *J. Comp. Sci*, vol. 8, no. 2, 2011.

[135] M. O. Riedl, "Vignette-based story planning: Creativity through exploration and retrieval," in *Proceedings of the 5th International Joint Workshop on Computational Creativity*, 2008, pp. 41–50.

[136] I. Rish and G. Tesauro, "Active collaborative prediction with maximum margin matrix factorization." *ISAIM*, vol. 2008, p. 20, 2008.

[137] M. Robillard, R. Walker, and T. Zimmermann, "Recommendation systems for software engineering," *IEEE Software*, vol. 27, no. 4, pp. 80–86, July 2010.

[138] P. Rola, D. Kuchta, and D. Kopczyk, "Conceptual model of working space for agile (scrum) project team," *Journal of Systems and Software*, vol. 118, pp. 49–63, 2016.

[139] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware 2001*. Springer, 2001, pp. 329–350.

[140] D. Salber, A. K. Dey, and G. D. Abowd, "The context toolkit: aiding the development of context-enabled applications," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 1999, pp. 434–441.

[141] G. Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[142] W. Scacchi, "Process models in software engineering," *Encyclopedia of software engineering*, 2001.

[143] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The adaptive web*. Springer, 2007, pp. 291–324.

[144] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, ser. WMCSA '94. Washington, DC, USA: IEEE Computer Society, 1994, pp. 85–90. [Online]. Available: http://dx.doi.org/10.1109/WMCSA.1994.16

[145] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*. IEEE, 1994, pp. 85–90.

[146] A. Schmidt and C. Winterhalter, "User context aware delivery of e-learning material: Approach and architecture," *Journal of Universal Computer Science*, vol. 10, no. 1, pp. 28–36, 2004.

[147] L. Serafini and A. Tamilin, "Drago: Distributed reasoning architecture for the semantic web," in *The Semantic Web: Research and Applications*. Springer, 2005, pp. 361–376.

[148] G. Shani and A. Gunawardana, "Evaluating recommendation systems," in *Recommender systems handbook*. Springer, 2011, pp. 257–297.

[149] P. Sharma, "Teaching creativity-a systematic viewpoint," *Enhancing curricula*, pp. 330–342, 2002.

[150] M. Shaw and D. Garlan, *Software architecture: perspectives on an emerging discipline*. Prentice Hall Englewood Cliffs, 1996, vol. 1.

[151] L.-p. Shen and R.-m. Shen, "Ontology-based learning content recommendation," *International Journal of Continuing Engineering Education and Life Long Learning*, vol. 15, no. 3-6, pp. 308–317, 2005.

[152] B. Shneiderman, "Creating creativity: user interfaces for supporting innovation," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 7, no. 1, pp. 114–138, 2000.

[153] B. Shneiderman, G. Fischer, M. Czerwinski, M. Resnick, B. Myers, L. Candy, E. Edmonds, M. Eisenberg, E. Giaccardi, T. Hewett *et al.*, "Creativity support tools: Report from a us national science foundation sponsored workshop," *International Journal of Human-Computer Interaction*, vol. 20, no. 2, pp. 61–77, 2006.

[154] R. L. Sie, M. Bitter-Rijpkema, and P. B. Sloep, "A simulation for content-based and utility-based recommendation of candidate coalitions in virtual creativity teams," *Procedia Computer Science*, vol. 1, no. 2, pp. 2883–2888, 2010.

[155] G. A. Sielis, C. Mettouris, A. Tzanavari, G. A. Papadopoulos, R. M. G. Dols, and Q. Siebers, "A context aware recommender system for creativity support tools," *J. UCS*, vol. 17, no. 12, pp. 1743–1763, 2011. [Online]. Available: http://dx.doi.org/10.3217/jucs-017-12-1743

[156] G. A. Sielis, A. Tzanavari, R. Dols, G. Hopmans, P. Dolog, K. Schmid, P. Grube, A. Kouloumbis, and S. Papavasiliou, "D3. 1–description of context awareness in idspace," 2008. [Online]. Available: http://dspace.ou.nl/bitstream/1820/1662/1/idSpace%20D3.1%20final%20%20EC%2028-11-2008.pdf

[157] G. A. Sielis, A. Tzanavari, and G. A. Papadopoulos, "Enhancing the creativity process by adding context awareness in creativity support tools," in *Universal Access in*

Human-Computer Interaction. Applications and Services, 5th International Conference, UAHCI 2009, Held as Part of HCI International 2009, San Diego, CA, USA, July 19-24, 2009. Proceedings, Part III, 2009, pp. 424–433. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02713-0_45

[158] G. A. Sielis, A. Tzanavari, and G. A. Papadopoulos, "Archreco: Software architecture design tool enhanced by context aware recommendations for design patterns," in *Proceedings of the 2015 European Conference on Software Architecture Workshops*. ACM, 2015, p. 67.

[159] G. A. Sielis, A. Tzanavari, and G. A. Papadopoulos, "Investigating the state-of-the-art in software architecture design tools and their support in personalized recommendations: A survey in software architecture process, tools and practices," University of Cyprus, Tech. Rep. UCY-CS-TR-15-1, 2015.

[160] G. A. Sielis, A. Tzanavari, and G. A. Papadopoulos, "Recommender systems review of types, techniques, and applications," *Encyclopedia of Information Science and Technology, Third Edition, Mehdi Khosrow-Pour (ed), IGI Global, 2015, chapter 714*, pp. 7260–7270, 2015.

[161] G. A. Sielis, A. Tzanavari, and G. A. Papadopoulos, "A social creativity support tool enhanced by recommendation algorithms: The case of software architecture design," in *Knowledge, Information and Creativity Support Systems*. Springer, 2016, pp. 457–466.

[162] B. Siljee, I. Bosloper, and J. Nijhuis, "A classification framework for storage and retrieval of context," in *KI-04 Workshop on Modelling and Retrieval of Context, CEUR*, vol. 114. Citeseer, 2004.

[163] B. Simon, Z. Miklós, W. Nejdl, M. Sintek, and J. Salvachua, "Smart space for learning: A mediation infrastructure for learning services," in *Proceedings of the Twelfth International Conference on World Wide Web*, 2003, pp. 20–24.

[164] I. Simon, D. Morris, and S. Basu, "Mysong: automatic accompaniment generation for vocal melodies," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2008, pp. 725–734.

[165] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," *Web Semantics: science, services and agents on the World Wide Web*, vol. 5, no. 2, pp. 51–53, 2007.

[166] J. F. Sowa, "Knowledge representation: logical, philosophical, and computational foundations," 1999.

[167] V. Stanford, "Beam me up, doctor mccoy [pervasive computing]," *Pervasive Computing, IEEE*, vol. 2, no. 3, pp. 13–18, 2003.

[168] A. J. Starko, *Creativity in the classroom: Schools of curious delight*. Routledge, 2013.

[169] R. J. Sternberg, *Handbook of creativity*. Cambridge University Press, 1999.

[170] G. M. Steve Pepper, "Xml topic maps (xtm) 1.0, topicmaps.org specification," August 2001.

[171] T. Strang and C. Linnhoff-Popien, "A context modeling survey," in *In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*, 2004.

[172] J.-Z. Sun and J. Sauvola, "Towards a conceptual model for context-aware adaptive services," in *Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference on*. IEEE, 2003, pp. 90–94.

[173] P. Tarasewich, "Designing mobile commerce applications," *Communications of the ACM*, vol. 46, no. 12, pp. 57–60, 2003.

[174] K. Taveter and G. Wagner, "Agent-oriented enterprise modeling based on business rules." Springer, 2001, pp. 527–540.

[175] L. Terveen and D. W. McDonald, "Social matching: A framework and research agenda," *ACM transactions on computer-human interaction (TOCHI)*, vol. 12, no. 3, pp. 401–434, 2005.

[176] N. Tintarev and J. Masthoff, "A survey of explanations in recommender systems," in *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*. IEEE, 2007, pp. 801–810.

[177] H.-L. Truong, L. Juszczyk, A. Manzoor, and S. Dustdar, *Escape–an adaptive framework for managing and providing context information in emergency situations*. Springer, 2007.

[178] N. Tsantalis, A. Chatzigeorgiou, G. Stephanides, and S. Halkidis, "Design pattern detection using similarity scoring," *Software Engineering, IEEE Transactions on*, vol. 32, no. 11, pp. 896–909, Nov 2006.

[179] D. Tsatsou, F. Menemenis, I. Kompatsiaris, and P. C. Davis, "A semantic framework for personalized ad recommendation based on advanced textual analysis," in *Proceedings of the third ACM conference on Recommender systems*. ACM, 2009, pp. 217–220.

[180] K. Unsworth, "Unpacking creativity," *Academy of management review*, vol. 26, no. 2, pp. 289–297, 2001.

[181] P. Victor, M. De Cock, and C. Cornelis, *Trust and Recommendations*. Boston, MA: Springer US, 2011, pp. 645–675. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-85820-3_20

[182] K. Wagatsuma, T. Harada, S. Anze, Y. Goto, and J. Cheng, "A supporting tool for spiral model of cryptographic protocol design with reasoning-based formal analysis," in *Advanced Multimedia and Ubiquitous Engineering*. Springer, 2016, pp. 25–32.

[183] G. Wagner, "How to design a general rule markup language," in *In Invited talk at the Workshop XML Technologien für das Semantic Web (XSW 2002*. Citeseer, 2002.

[184] A. I. Wang and B. Wu, "An application of a game development framework in higher education," *Int. J. Comput. Games Technol.*, vol. 2009, pp. 6:1–6:12, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1155/2009/693267

[185] P. Wang, "The interpretation of fuzziness." *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics: a publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 26, no. 2, pp. 321–326, 1995.

[186] Y.-K. Wang, "Context awareness and adaptation in mobile learning," in *Wireless and Mobile Technologies in Education, 2004. Proceedings. The 2nd IEEE International Workshop on*. IEEE, 2004, pp. 154–158.

[187] T. Weilkiens, J. G. Lamm, S. Roth, and M. Walker, "B: The v-model," *Model-Based System Architecture*, pp. 343–352, 2016.

[188] M. Weimer, A. Karatzoglou, Q. V. Le, and A. Smola, "Maximum margin matrix factorization for collaborative ranking," *Advances in neural information processing systems*, pp. 1–8, 2007.

[189] M. Weiser, "Ubiquitous computing," *Computer*, no. 10, pp. 71–72, 1993.

[190] M. Weiss and A. Birukou, "Building a pattern repository: Benefitting from the open, lightweight, and participative nature of wikis."

[191] *W3C Semantic Web Activity*, World Wide Consortium (W3C) Std., 2016. [Online]. Available: https://www.w3.org/2001/sw/

[192] A. Wrightson, "Topic maps and knowledge representation," February 2001.

[193] H. C. Wu, R. W. P. Luk, K. F. Wong, and K. L. Kwok, "Interpreting tf-idf term weights as making relevance decisions," *ACM Transactions on Information Systems (TOIS)*, vol. 26, no. 3, p. 13, 2008.

[194] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H.-P. Kriegel, "Probabilistic memory-based collaborative filtering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 1, pp. 56–69, 2004.

[195] Y. Zeng, N. Zhong, X. Ren, and Y. Wang, "User interests driven web personalization based on multiple social networks," in *Proceedings of the 4th International Workshop on Web Intelligence & Communities*, ser. WI&C '12. New York, NY, USA: ACM, 2012, pp. 9:1–9:4. [Online]. Available: http://doi.acm.org/10.1145/2189736.2189749

[196] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, "Improving recommendation lists through topic diversification," in *Proceedings of the 14th international conference on World Wide Web*. ACM, 2005, pp. 22–32.

[197] O. Zimmermann, U. Zdun, T. Gschwind, and F. Leymann, "Combining pattern languages and reusable architectural decision models into a comprehensive and comprehensible design method," in *Software Architecture, 2008. WICSA 2008. Seventh Working IEEE/IFIP Conference on*, Feb 2008, pp. 157–166.

[198] O. Zimmermann, "Architectural decision identification in architectural patterns," in *Proceedings of the WICSA/ECSA 2012 Companion Volume*, ser. WICSA/ECSA '12. New York, NY, USA: ACM, 2012, pp. 96–103. [Online]. Available: http://doi.acm.org/10.1145/2361999.2362021

Appendix A

Survey Questions

A.1 Demographic Data

1. What is your age?
2. How do you define the type of organization you are working for? Answer Count Percentage
3. What is your position / role in the organization?
4. How many years of experience in Software Design do you have?
5. In how many projects you had active role in Software Design?
6. In the projects that you participate, do you collaborate with other experts?
7. Have you ever worked with a team remotely on Software Design?

A.2 Project Management

1. How many projects do you usually run at the same time?
2. Please arrange in a priority order the attributes of your employees that you take into account before you assign to them a new project.

3. When a new project is about to start can you roughly describe three steps for the preparation of the project?

4. Do you usually involve resources from other departments of other teams in abroad into your projects?

5. Do you always know the people you invite for collaboration?

6. During the design and preparation of the project how important would be for you to receive recommendations of people that can get involved into your project, by the system?

7. How easy is it for you to find other people to ask them to join a project that you manage?

8. Please describe how you can find other people to join your projects.

9. What kind of additional resource material would you consider as helpful during the process of the Software Architecture Design?

10. How easy is it for you to have access to other projects created by other teams so you can use them as reference?

A.3 Software Design

1. Do you use Software Design modeling tools?

2. Can you name some of the Software Design Modeling Tools that you are aware of?

3. Do you usually use the Software Modeling Tools in cooperation with other colleages?

4. Please name some Software Modeling Tools which support collaboration of peers

5. Would you agree in sharing your Software Design projects with other people in your company?

6. Would you agree in sharing your Software Design projects with other people in your company?

7. Would you agree in sharing your Software Design projects with other people in a public network?

8. How important is it for you to have access to Software Design projects that other people created?

9. Do you usually use other Software Design projects as reference to your project?

10. How easy is for you to find projects related to yours to use them as reference source?

11. Which methods you use to find related projects?

12. Are you satisfied with the methods you use to find reference projects?

13. In the process of collaboration between you and your team, which functionality do you believe that would facilitate the process of Software Design?

A.4 Creation of new Ideas

1. How easy is for you to create a new project based on a new idea that you have?

2. How easy is for a member in your team to process a new idea as a new project?

Appendix B

Semantic Web Code-Samples

B.1 Sample of Dynamic Model Creation using data from MySQL DB

Listing B.1: Sample dynamic Model creation from mysql DB

```
ResultSet res = conn.SelectQuery("SELECT * FROM CST_users ");


    while(res.next())
    {
          String id= res.getString("id");
          String User=res.getString("firstname") +"_"+
           res.getString("lastname");
           String usrName=res.getString("username");
           String profession = res.getString("profession");
           String education = res.getString("education");
           String languages=res.getString("languages");
           String email=res.getString("email");
           String shortcv=res.getString("shortcv");
           String country=res.getString("country");
           String company=res.getString("company");


           writer.openIndividual("", id, "j", "Person");


   if(null != userName)
   {
     writer.addLiteral("rdfs", "label", userName, "xsd:string");
     writer.addLiteral("j", "hasName", userName, "xsd:string");
     writer.addLiteral("rdfs", "label", usrName, "xsd:string");
```

```
    writer.addLiteral("j", "hasUsername", usrName, "xsd:string");
    writer.addLiteral("j", "hasProfession", profession, "xsd:string");
    writer.addLiteral("j", "hasEducation", education, "xsd:string");
    writer.addLiteral("j", "speaksLanguages", languages, "xsd:string");
    writer.addLiteral("j", "hasEmail", email, "xsd:string");
    writer.addLiteral("j", "hasShortCV", shortcv, "xsd:string");
    writer.addLiteral("j", "livesInCountry", country, "xsd:string");
    writer.addLiteral("j", "worksInCompany", company, "xsd:string");
  }
     writer.closeIndividual();

  }
```

B.2 Sample of Dynamic Model Creation using data from XML file

Listing B.2: Sample dynamic Model creation from XML File

```
File fXmlFile = new File("allDataUpToProjectTasks.xml");
        DocumentBuilderFactory dbFactory =
         DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
        Document doc = dBuilder.parse(fXmlFile);
        doc.getDocumentElement().normalize();
        NodeList nList = doc.getElementsByTagName("row");

        for (int temp = 0; temp < nList.getLength(); temp++)
        {
                Node nNode = nList.item(temp);
                if (nNode.getNodeType() == Node.ELEMENT_NODE)
                {
                Element eElement = (Element) nNode;
           writer.openIndividual("", eElement.getElementsByTagName("user_id").item(0).getTextContent()
.replaceAll("[-+.^:,\"]",""), "j", "Person");
           if(null != eElement.getElementsByTagName("user_id").item(0).getTextContent().replaceAll("[-+.^:,]",""))
           {
            writer.addLiteral("rdfs", "label", eElement.getElementsByTagName("user_name").item(0).getTextContent()
.replaceAll("[-+.^:,\"=]",""), "xsd:string");
            writer.addLiteral("j", "hasName", eElement.getElementsByTagName("realname").item(0).getTextContent()
.replaceAll("[-+.^:,\"=]",""), "xsd:string");
            writer.addLiteral("rdfs", "label", eElement.getElementsByTagName("user_name").item(0).getTextContent()
.replaceAll("[-+.^:,\"=]",""), "xsd:string");
            writer.addLiteral("j", "hasUsername", eElement.getElementsByTagName("user_name").item(0).getTextContent()
.replaceAll("[-+.^:,\"=]",""), "xsd:string");
```

```
            writer.addLiteral("j", "hasProfession", "", "xsd:string");
            writer.addLiteral("j", "hasEducation", "", "xsd:string");
            writer.addLiteral("j", "speaksLanguages", "en", "xsd:string");
            writer.addLiteral("j", "hasEmail", "", "xsd:string");
            writer.addLiteral("j", "hasShortCV", eElement.getElementsByTagName("people_resume").item(0).getTextContent()
.replaceAll("[-+.^:,\"=\n]",""), "xsd:string");
            writer.addLiteral("j", "livesInCountry", "", "xsd:string");
            writer.addLiteral("j", "worksInCompany", "", "xsd:string");
            }
           writer.closeIndividual();
                }
        }
        writer.close();
```

B.3 SPARQL examples

Listing B.3: SPARQL query for retrieval of Design Pattern's Category

```
protected String getCategoyOfPattern(String pattern)
   {

      String cat="";
            String queryString =
                     "PREFIX dp: <" + ontVocabulary.Base.getString() + "> \n" +
                     "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> \n" +
                     "SELECT ?x "
             + "WHERE { "

              + "?pat dp:hasPatternName  \""+pattern+"\" . \n"
              + "?pat dp:CategorizedIn  ?subcategory . \n"
              + "?subcategory dp:SubCategoryOf  ?category . \n"
              + "?category dp:hasCategoryName ?x . \n"
                     + "}";
             Query query;
                     QueryExecution queryExecution;

                     ResultSet rs;

              query = QueryFactory.create(queryString);
              queryExecution = QueryExecutionFactory.create(query, _model);

              rs = queryExecution.execSelect();
```

```
            while(rs.hasNext()){
                    QuerySolution solution = rs.nextSolution();

                    RDFNode category;
                    //Resource architecture,architecture_type;


                    category = solution.getLiteral("x");
                    cat= extractValueFromNode(category);
            }

            return cat;

    }
```

Listing B.4: SPARQL query for retrieval of Design Patterns for specific category

```
protected List<Pattern> getDesignPatternsByCategory(String[] categoryName)
    {
        List<Pattern> toReturn = new ArrayList<Pattern>();
        String queryString="";
        for(int i=0; i<categoryName.length;i++)
        {
        if(categoryName[0]=="All")
        {
                queryString =
                        "PREFIX dp: <" + ontVocabulary.Base.getString() + "> \n" +
                        "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> \n" +
                        "SELECT ?subcategory "
                            + "?category "
                            + "?pattern "
                            + "?pattern_name "
                            + "?intent "
                    + "WHERE { "
                    + "?category dp:hasCategoryName ?x . \n"
                    + "?subcategory dp:SubCategoryOf  ?category . \n"
                    + "?pattern dp:CategorizedIn  ?subcategory . \n"
                    + "?pattern dp:" + ontVocabulary.hasPatternName.toString() + "  ?pattern_name . \n"
                    + "?pattern dp:" + ontVocabulary.hasIntent.toString() +"  ?intent . \n"
                    + "}";


        }
        else
        {
```

```
queryString =
                "PREFIX dp: <" + ontVocabulary.Base.getString() + "> \n" +
                "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> \n" +
                "SELECT ?subcategory "
                    + "?category "
                    + "?pattern "
                    + "?pattern_name "
                    + "?intent "
        + "WHERE { "

         + "?category dp:hasCategoryName \""+categoryName[i]+"\" . \n"
         + "?subcategory dp:SubCategoryOf  ?category . \n"
         + "?pattern dp:CategorizedIn  ?subcategory . \n"
         + "?pattern dp:" + ontVocabulary.hasPatternName.toString() + "  ?pattern_name . \n"
         + "?pattern dp:" + ontVocabulary.hasIntent.toString() +"  ?intent . \n"

         + "}";

}
String patternName;
String implementation;
String _intent;
String _AlsoKnownAs;
String _Collaborators;
String _Consequences;
String _KnownUses;
String _Motivation;
String _Participants;
String _RelatedPatterns;
String _SampleCode;
String _Structure;
String _Example;
String _Image;
String _Category;
String _SubCategory;



                Query query;
                QueryExecution queryExecution;


                ResultSet rs;

          query = QueryFactory.create(queryString);
          queryExecution = QueryExecutionFactory.create(query, _model);
```

```
rs = queryExecution.execSelect();

while(rs.hasNext()){
        QuerySolution solution = rs.nextSolution();
        RDFNode pattern_name;
        RDFNode implement,AlsoKnownAs,collaborators,consequenses,knownUses,motivation,participants,
                relatedPatterns,sampleCode,structure,example,image,intent;
        Resource pattern,subcategory,category;

        category = solution.getResource("category"); //Literal means the property value
        subcategory = solution.getResource("subcategory");
        pattern = solution.getResource("pattern");
        pattern_name = solution.getLiteral("pattern_name");



        _Category= stripType(extractValueFromNode(
                        getFirstNode(
                                _model.listObjectsOfProperty(
                                                category,
                                                getProperty(ontVocabulary.hasCategoryName)))));
        _SubCategory= stripType(extractValueFromNode(
                        getFirstNode(
                                _model.listObjectsOfProperty(
                                                subcategory,
                                                getProperty(ontVocabulary.hasSubCategoryName)))));

        patternName= extractValueFromNode(pattern_name);
        Pattern p=extractPattern(pattern);
        implementation=p.getImplementation();
         _intent = p.getIntent();
        _AlsoKnownAs=p.getAlsoKnownAs();
        _Collaborators=p.getCollaborators();
        _Consequences=p.getConsequences();
        _Example=p.getExample();
        _Image=p.getImage();
        _KnownUses=p.getKnownUses();
        _Motivation=p.getMotivation();
        _Participants=p.getParticipants();
        _RelatedPatterns=p.getRelatedPatterns();
        _SampleCode=p.getSampleCode();
        _Structure=p.getStructure();
```

```
            toReturn.add( new PatternImpl(patternName, _intent,implementation,_AlsoKnownAs,_Collaborators,_Consequences,
        _KnownUses,_Motivation,_Participants,_RelatedPatterns,_SampleCode,
        _Structure,_Example,_Image,_Category,_SubCategory));

                }
        }

                return toReturn;
    }
```

Appendix C

Ontology Models - Visualizations

C.1 Creativity Ontology Visualised Model using Protege

The visualized ontology model is shown in figure 22

C.2 Creativity merged with SE entities Ontology Visualised Model using Protege

The visualized ontology model is shown in figure 23

C.3 Class Hierarchy of the merged ontology model

The ontology model structure and class hierarchy is shown in figure 24

C.4 Completed ontology model with data properties and relations included

The final visualized ontology model is shown in figure 25

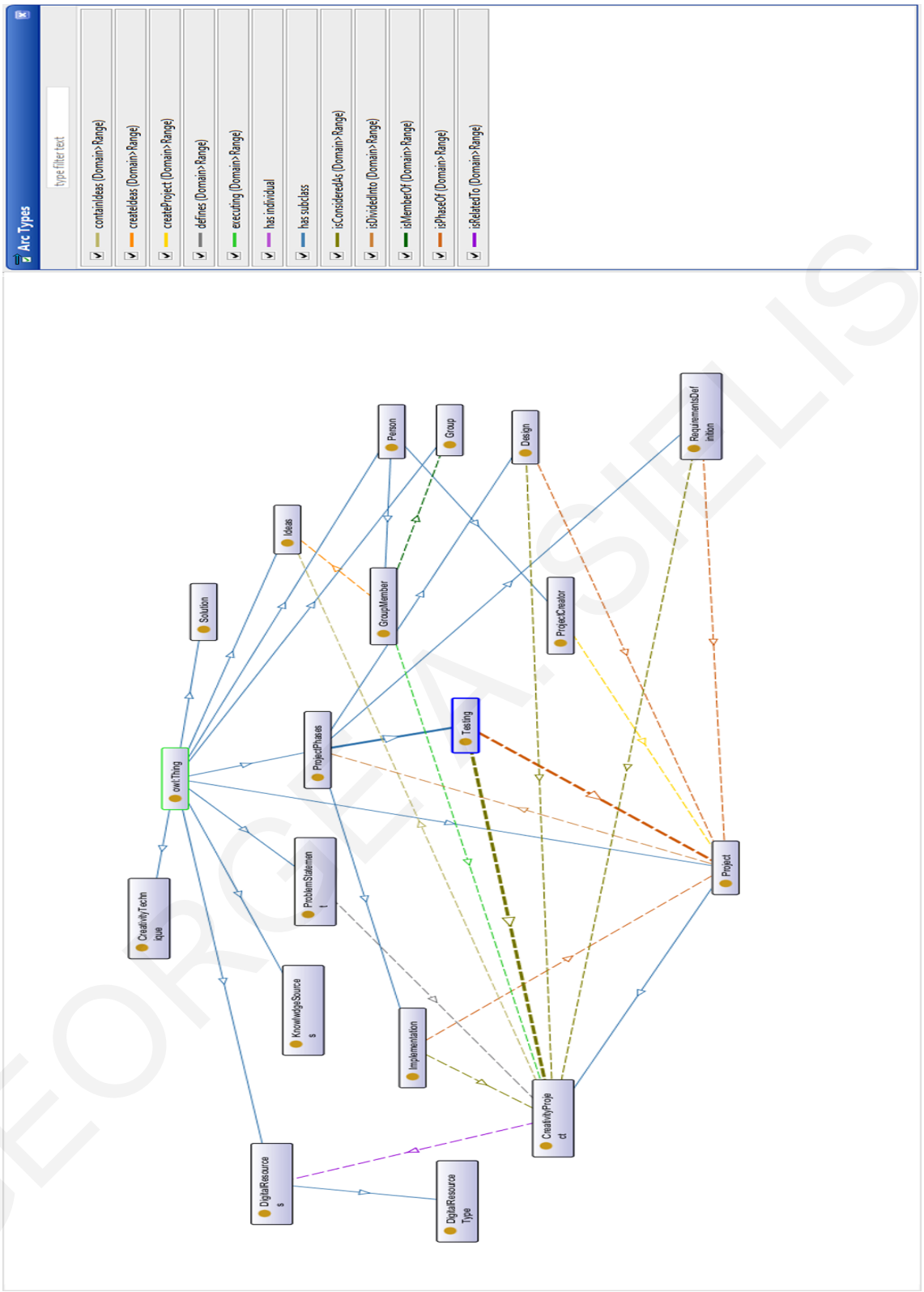

Figure 22: Creativity Ontology Model Created in Protege Ontology Editing tool

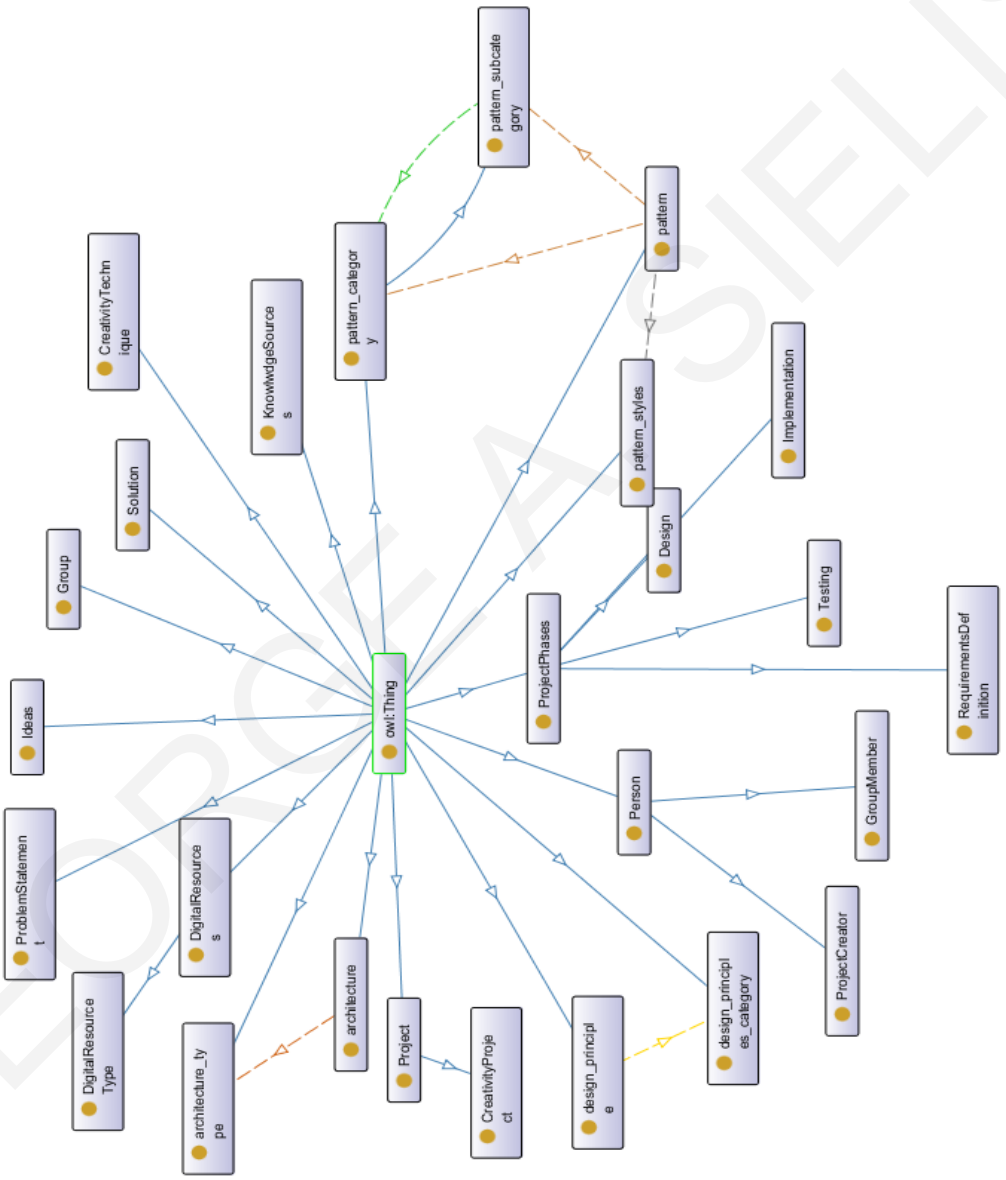

Figure 23: Creativity merged with SE entities Ontology Model Created in Protege Ontology Editing tool

Figure 24: Class Hierarchy of the merged ontology model in Protege Ontology Editing tool

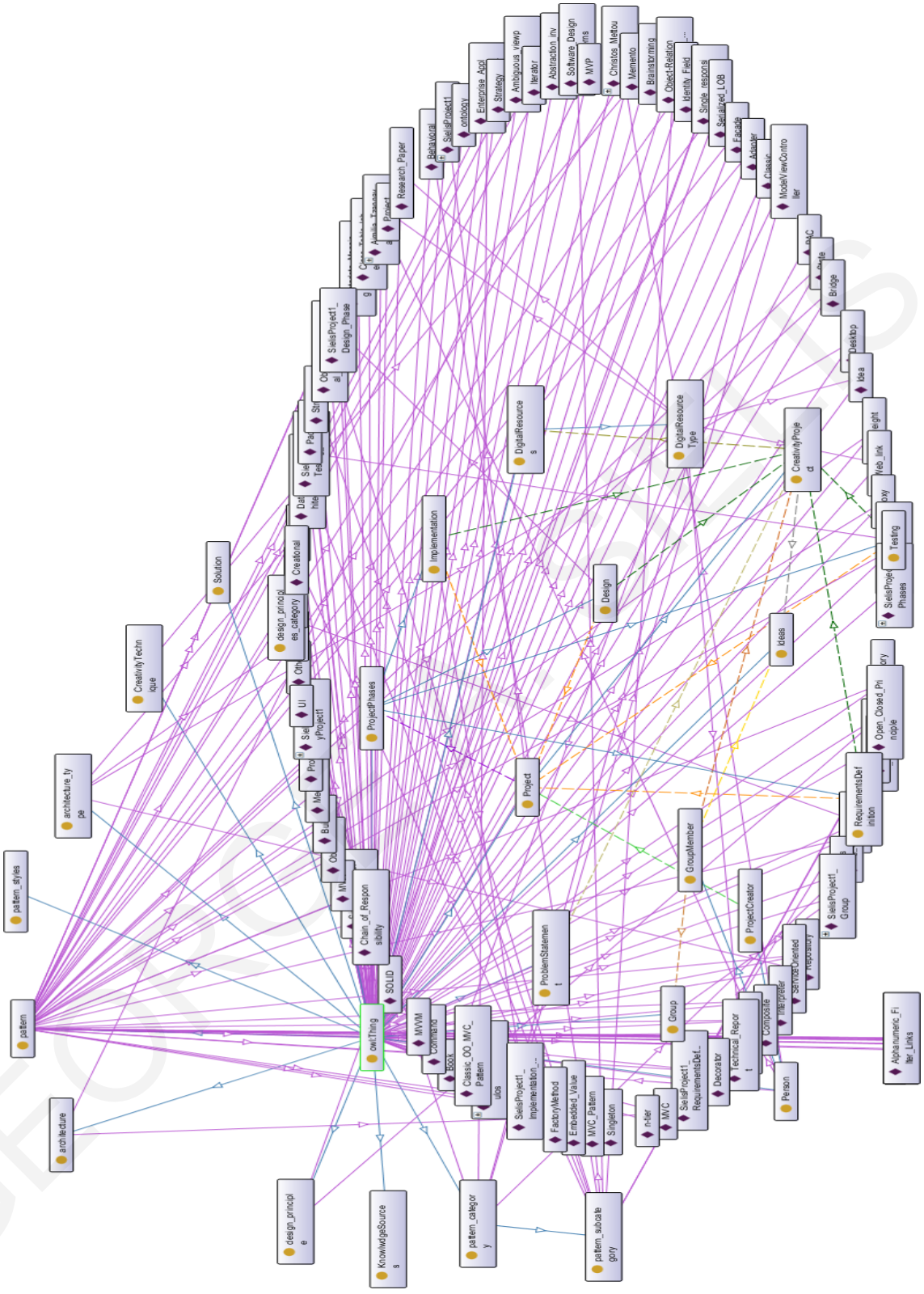

Figure 25: Completed ontology model with data properties and relations visualized in Protege Ontology Editing tool

Appendix D

ArchReco Evaluation Questionnaire

D.1 Demographic Data

1. Please write your Student Card Registration Number - AM. (It will be used only as a reference to the evaluation session logs for comparison of the results)
2. Profession
3. Location
4. Nationality
5. Gender
6. Age
7. Educational Background
8. Year of Studies

D.2 Pre-test questionnaire

1. Please rate your experience with Software Design tooling, understood as systems that promote, accelerate and facilitate the design of Software Design Models?
2. Please rate your level of experience in Software Design
3. Please rate your knowledge of Design Patterns
4. Please rate your experience in using Design Patterns

D.3 Post-task questionnaire

1. Overall, I am satisfied with the ease of completing the task
2. Overall, I am satisfied with the amount of time it took to complete the task
3. Overall, I believe I learned new Design Patterns with the use of the software
4. Overall, I believe I learned where and how Design Patterns can be used
5. I believe I learned new things for Design Patterns

D.4 Post-test questionnaire

1. Using the tool in Designing Software models would enable me to accomplish tasks more quickly
2. Using the tool would improve my understanding in using Design Patterns in a high level software design model
3. Using the tool in Designing Software models would increase my productity

4. Using the tool to identify the most suitable Design Patterns would enhance my effectiveness on the job

5. Using the tool would make easier the process of Software Design

6. I would find the tool useful in Designing Software diagrams

7. The outcome of the tool would be beneficiary for the software developers who will implement the diagram into an actual application

8. ArchReco tool can effectively support the creation of high level Software Design Model.

9. ArchReco tool can effectively support the representation, and management of Software Design Components.

10. The context-sensitive support (i.e. recommendations) is crucial to the ArchReco design process.

11. Using ArchReco tool supports me in being more creative during the design process

12. ArchReco tool enhances the outcome of the High Level diagram design.

13. I describe my experience with ArchReco tool in general as positive.

14. ArchReco tool offers stimulating possibilities to explore new Design Patterns.

15. ArchReco tool helps in choosing useful Design Patterns to apply in an Software Design diagram.

16. I feel that I learned to work creatively using the ArchReco tool.

17. I found ArchReco tool helpful to support us to go over and over new design patterns till we found a suitable one to apply in our design model.

18. I found the recommendation of Design Patterns useful

19. The information provided for each pattern was sufficient

20. Recommendations of Design Patterns helped me learn new patterns.

21. Some of the recommended items are familiar to me

22. I am not familiar with the items that were recommended to me

23. The items recommended to me are novel and interesting

24. The recommender system is educational.

25. The recommender system helps me discover new Design Patterns

26. I could not find new design patterns through the recommender

27. The items recommended to me took my personal context requirements into consideration.

28. The recommender provides an adequate way for me to express my preferences

29. I became familiar with the recommender system very quickly.

30. The recommended items effectively helped me find the ideal Design Patterns