Sielis *et al. Journal of Software Engineering Research and Development* (2017) 5:2
DOI 10.1186/s40411-017-0036-y

Journal of Software Engineering Research and Development

RESEARCH Open Access

ArchReco: a software tool to assist software design based on context aware recommendations of design patterns

George A. Sielis[1*], Aimilia Tzanavari[2] and George A. Papadopoulos[1]

*Correspondence:
sielis@cs.ucy.ac.cy
[1]Department of Computer Science, University of Cyprus, University of Cyprus, P.O. Box 20537, 1678 Nicosia, Cyprus
Full list of author information is available at the end of the article

Abstract

This work describes the design, development and evaluation of a software Prototype, named ArchReco, an educational tool that employs two types of 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 for 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 finding the most suitable Design Pattern to use according to the working context, learn the meaning, objectives and usages of each Design Pattern. The current work presents the Semantic Modeling of the Software Design process through the definition of the context that defines the Software Design process and in particular the representation of the Design Patterns as Ontology model, the implemented Context Aware Recommendation Algorithms and the evaluation results extracted from a user based testing for the ArchReco prototype.

Keywords: Software design, Context awareness, Semantic web, Recommendation algorithms, Software engineering educational tools, Design patterns learning

Content

Methods

In this article, the ArchReco Software prototype tool is presented, which supports Design Patterns learning and practicing through Semantic Web based Context Aware Recommendations of Design Patterns.

Results

ArchReco provides a design environment where users can draw diagrams, with the use of pre-defined shapes that exist in a palette. The description of the shapes and the purpose of use for each shape, are used as part of a contextual elements set that is processed by the system for the computation of the most suitable Context-based recommended Design Patterns. The recommended Design Patterns are retrieved from multiple data sources and filtered based on the contextual information that is processed, when recommendations are requested. The recommendation results can be used by users, to read the content of the recommended Design Patterns and decide whether a Design Pattern should be

included in a Diagram or not. The tool was evaluated by users and the qualitative results of the evaluation are presented in the article.

Conclusions

The evaluation results revealed the tool's usefulness, usability and most importantly proved the educational character of the Context Aware Recommendation tool regarding the recommendation of Design Patterns.

Background

Design Patterns, as defined in (Alexander et al. 1977), are a well-known and frequently used software engineering problem-solving discipline, which has emerged from the object-oriented community. Design Patterns are "templates" that intend to solve particular problems in a specific context. The solution of such problems can adopt derived models, provided by one or more individual Design Pattern or their combination, depending on the kind or type of the designed module/component. Therefore, the satisfaction of new software requirements and specifications, with the use of Design Patterns can facilitate the overall Software Design.

Buschmann et al. (1996) states that "*Patterns help you build on the collective experience of skilled software engineers. They capture existing, well-proven experience in software development and help to promote good design practice*". Following this statement, the issue that needs to be investigated is which patterns are the most suitable to use for the design of a module/component under specific conditions. In reality, there is no single answer since the selection of Design Patterns that can be applied in a design process depends primarily on the subjective opinion of the designer, and it is based on which design patterns better match the design requirements. The adoption of a design pattern or a set of patterns relies on how knowledgeable the designer is, related to the existing design model.

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 (Cross 2004; Walz et al. 1993). In order to target the fresh Software Engineers and more specifically Computer Science or Engineering students, and assist them in overcoming the feeling of uncertainty in designing software models using Design Patterns, we have created ArchReco, a Software Architecture Design prototype tool, which supports Context-Aware recommendations for Design Patterns. Context as it is defined by (Dey et al. 2001), is "*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*". With ArchReco prototype, the current research work foresees the examination not only of the impact that the recommendations of Design Patterns may have on Software Design and more specifically the High Level Software Design, but also in terms of the impact that Context Awareness, as a recommendation mechanism, has on proposing Design Patterns. High Level Software Design is defined in (Briand et al. 1999) 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*".

ArchReco is designed and implemented to be an educational tool that employs two types of 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 for High Level Software models.

The motivation of this work is based on the lack of Software Design tools that support new designers in finding and applying Design Patterns in High Level Software models based on input requirements written in natural 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 (Gomes et al. 2002; Guéhéneuc and Mustapha 2007; Weiss and Birukou 2007) but none of them was taking into account the context or has produced a complete prototype solution for specific target groups such as CS or SE students which ArchReco supports.

The rest of the paper is structured as follows: Section Motivation presents the motivation for the current work; Section ArchReco: a design patterns recommendation tool presents the related work and how this work is different than the related work reported in the literature; Section A use case scenario presents the ArchReco prototype, the context analysis and its Architecture. It also presents the Semantic Web Interoperability and Context Aware Recommendation components; Section Methods presents a Use Case scenario describing how ArchReco can be used; Section Related work presents the ArchReco prototype evaluation and presents the results through a qualitative analysis. The article concludes with section Conclusions, where the conclusions and further work are discussed.

Motivation

Over the last few years, research papers related to the Design Patterns and their applications to real life software tools were significantly increased. Design Patterns are examined from many different perspectives such as the recovery of Design Patterns from existing software tools (Rasool and Streitferdt 2011), Formalization and Reasoning Design Patterns (Bayley and Zhu 2008; Hou and Hoover 2006), or generate Design Patterns repositories other than the well-known GoF patterns (Gamma et al. 1995) like for example the yahoo design patterns repository https://developer.yahoo.com/ypatterns/. It is worth noting that in publications related to the recommendation algorithms for Design Patterns (Gomes et al. 2002; Guéhéneuc and Mustapha 2007; Weiss and Birukou 2007) we have not found applied recommendation algorithms that were used in software tools. It is also worth noting that there is not reported work where context was used as part of the recommendation mechanisms for the computation of recommendations for Design Patterns.

Moreover, in the last decades, a rapid growth of new design patterns is noticed. Studying the Patterns Languages of Programs (PLOP) conferences proceedings, it is noticeable that every year a huge number of Design Patterns are presented. In total, there are at least 10 PLOP conferences, which are organized every year (a full list of the PLOP conferences can be found in http://hillside.net/conferences). Moreover, a big number of Design Patterns can be found in repositories such as in https://sourcemaking.com/ where 101 Design Patterns are presented or Enterprise Architecture Management Pattern (EAM). EAM patterns catalog v1.0 that was published in 2008 contained approximately 154 Design Patterns and more recently EAM Catalog V2 was published containing much more Design

Patterns including all recent Enterprise Architecture Management patterns that were defined until 2016. Undoubtedly, the number of Design Patterns increases every year and it is almost impossible for Software Engineers, especially students without experiences in the topic of Software Engineering to follow and learn all Design Patterns, existing or new ones.

Based on the latter ascertainment and with the aim to identify additional needs that Software Engineers may have in design and modelling processes, we performed an on-line survey with 28 participants from which *12 were 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*. The survey aimed to collect opinions from experts in Software Engineering regarding the existing tools that they use, and the lack of such tools in terms of Collaborative Design, Social Creativity Enhancement and the aiding mechanisms like Recommendation Systems that the known tools could possibly support. The comprehensive analysis of the survey is out of the scope of this work, but its results motivated the current research. The survey results related to the needs they have in accessing helpful resources and the accessibility to existing solutions that other people performed motivated the study of Design Patterns as helpful resources and reusable components that can be applied in new solutions. Design Patterns, as reusable components can be used in the form of training resources. However, the plethora of Design Patterns and the lack of a unified structured repository that all Patterns can be found, motivated the need for the development of methodologies for finding and retrieving them from multiple data sources in a structured form. Through the survey, Software engineers were also asked to define the most well-known Software Design Tools that they use. Table 1 presents the list of the tools that the participants defined. The examined tools shown in Table 1, lack of recommendation mechanisms that would support the software design process. The tools appeared in Table 1 are professional tools that consider existing knowledge as fact, which is not always reflecting the reality. From the one hand, professionals usually have short deadlines and do not have time to see all new patterns being proposed every year, and on the other hand, students must learn existing Design Patterns and at the same time develop the skills so they will become able to follow and learn new ones.

Therefore, this work attempts to examine the development of a High-Level Software design tool which would provide existing Design Patterns during the design process, enhanced by Context Aware recommendations. The tool aims to examine how users, students and professionals, perceive the usage of Design Patterns recommendations during the process of designing a High-Level software diagram, using recommended Design Patterns from multiple heterogenous data sources, for learning or applying them in High-level Software design diagrams.

ArchReco: a design patterns recommendation tool

For the appropriate design of ArchReco we have introduced a number of ontologies for modelling the Design Patterns and the contextual elements in a uniformed way. A few years ago, the most commonly used designed patterns were the GoF Design Patterns (Gamma et al. 1995), which are still the fundamental Design Patterns that software engineers use. Over the last few years, Design Patterns became a common practice for several companies and individual engineers developed new Design Patterns and categorized

Table 1 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

them according to the domain the patterns are applied to. For example, repositories such as the yahoo (https://developer.yahoo.com/ypatterns/) repository of Design Patterns for the UI/UX domain contain a number of patterns that can be applied in web based interfaces. Additionally, there are specific Design Patterns related to Data collection, Web Services, Mobile development and several other domain specific Design Patterns.

The coverage of all of the above domains within a unified model that would recognize all Design Patterns for each domain, is done through the context analysis of the

overall Software Design Process and the design of a Semantic Web Ontology model that would make possible the collection of Design Patterns information from several sources, in a structured way. At the same time, such a model would offer accessibility to specific information for its usage for further computational analysis.

Software design process - context analysis

In the existing literature, several software development models can be found such as the Waterfall (Balaji and Murugaiyan 2012), ETVX (Awais 2016), Prototype, Spiral (Wagatsuma et al. 2016), V-model (Weilkiens et al.), Agile methodologies such as Scrum (Rola et al. 2016) and Unified Process Model. Based on comparison of the existing software development models in (Munassar and Govardhan 2010), the most commonly used are the Waterfall and the Spiral models. However, the last years Agile models are increasingly applied. The difference between models like Waterfall and Spiral in comparison to Agile models is mainly on the sequence of applying the phases that are defining the Software Design 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 the modelling of the Software Design process, the Waterfall model was selected as a guiding model for the specification of the process phases and the semantic representation of the Software Design life-cycle. The phases used and analyzed based on Waterfall model are the Requirements specification, Design, Construction (implementation or coding), Integration, Testing and debugging (validation), Installation and Maintenance (Table 2).

Table 2 Software design phases contextual elements

Phase	Users	Types of ideas expected	Resource material	Tools
Requirements specification	Individual or Group	Text based ideas. New ideas or modified 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	High Level Design, Low Level Design, Web	Programming language IDEs.
Testing	Individual or Group	Testing methodologies, testing scenarios, unit-tests	Revision of requirements	
Maintenance	Individual or Group	Maintenance actions		

Table 3 Semantic analysis of the software design model (triples)

Domain	Property	Range
ProjectCreator	createProject	Project
Project	projectCreatedBy	ProjectCreator
ProjectGroupMember	isMemberOf	ProjectGroup
Project	isDividedInto	Phases
RequirementsDefinition	isPhaseOf	Project
Design	isPhaseOf	Project
Implementation	isPhaseOf	Project
Project	IsConsideredAs	CreativityProject
RequirementsDefinition	IsConsideredAs	CreativityProject

As it is shown in Table 2, Design Patterns is an important resource for the design phase and therefore a partial Ontology model specific for Design patterns was designed. Table 3 demonstrates a sample of the semantic relational triples that were applied for semantic representation of the entities and relations of the Software Design process.

The ontology model defined and designed as a Contextual model using the following contextual elements as the main "entities"/"concepts":

"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. 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 applications, 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.

"Diagram Shapes" - Each shape within a diagram may have specific type and purpose of use. The description of a shape, the type of Design Patterns that are related to its type as well as the connections with other shapes in the diagram, consist of contextual information that can be analyzed and processed.

"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.

Design patterns ontology model

The definition of the Design Patterns Ontology Model was done through the analysis of the Design Patterns attributes as contextual elements that characterize and define the

Design Patterns. The design of the model is following the Design Patterns description template as it is given by (Gamma et al. 1995) for the definition of the GoF Design Patterns. In this template, each particular Design Pattern is described based on its Intent, Consequences, Implementation, Known uses, Motivation and Collaborators attributes. Additionally, in the designed model more features were added such as a descriptive image that visually describes a Design Pattern, a generic descriptive UML diagram for the pattern (as image), url's with information for the pattern, documents related to the pattern. Those attributes were added for the usage of Design Pattern as a training material if this is needed by the applications that will apply the designed models. Figure 1 depicts the Semantic Web Ontology Model representation.

ArchReco prototype - design patterns training tool

The objective that the ArchReco prototype aimed to meet is the training of Computer Science and Computer Engineering students or professionals in learning new or existing 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

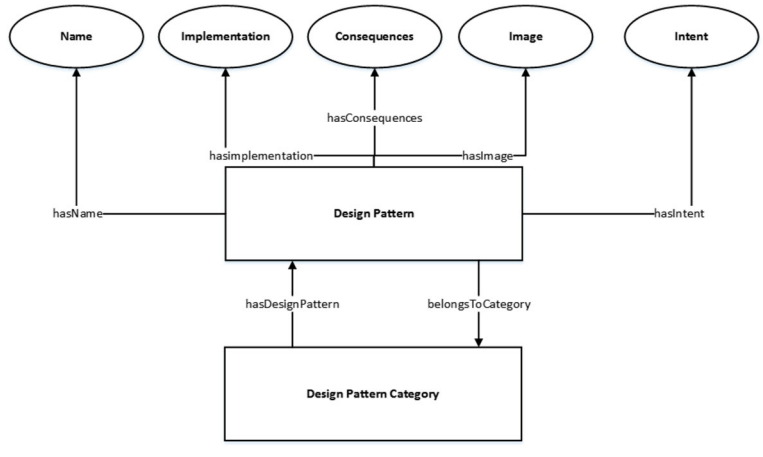

Fig. 1 Design Patterns Ontology Model Based on the GoF template Description of the ontology model representing the semantic relations between the Design Pattern entity and its attributes as data objects

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, in order to select the recommended patterns and study their provided content. An Architectural or Design Pattern, can be added to the canvas as individual shape or set of shapes, with the corresponding labels, which describe the Pattern(s) that the shape(s) refer to.

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.

Shape type is used as context information and processed as semantic filtering information using the Design Patterns Ontology model. A structured representation of Design Patterns using categories and sub-categories of Patterns is described in the following sample list:

- Classic (Gang of Four - GoF) **hasSubcategories** Behavioral, Creational, and Structural
- 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

The types of shapes that were defined and used for the ArchReco prototype are the following:

- Server Side Requirement shape - Used to describe features that will run the server. Recommended Design patterns are related design patterns used to implement functions in the server plane as e.g. GoF Design patterns.

- UI Requirement shape - Is used to describe functions that 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 shape - 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 shape - 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 shape - 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.

System architecture

ArchReco prototype was implemented in Java programming language using the JSwing framework. The application is developed as a desktop application that consists of three parts: the Core, Integration Layer (Integration of the partial components) and the presentation layer. The overall architecture is depicted in Fig. 2.

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 component. Each component is responsible for the processing of the data and the operations they perform according to the role they have in the system. 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.

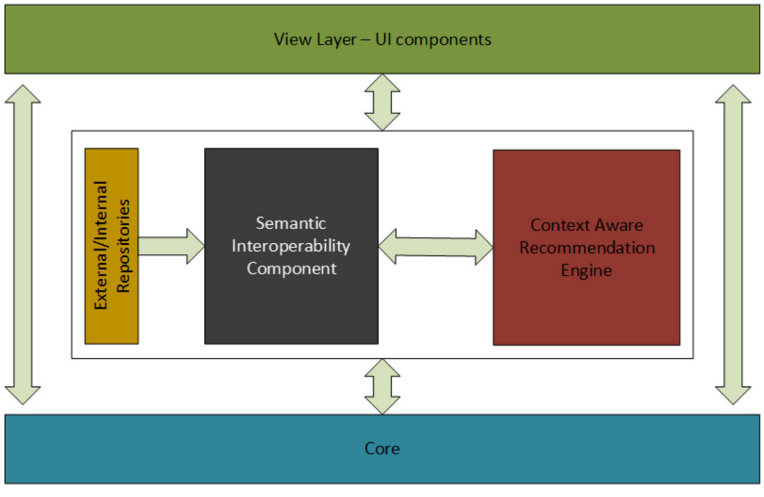

Fig. 2 ArchReco System Architecture Representation of the system architectural layer and the interaction between them

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.

In Fig. 3 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. Therefore, it can also operate offline.

Semantic interoperability component

The implementation of the semantic analysis, data retrieval and filtering is done with the implementation of the semantic interoperability component which was developped on top of Jena framework (Jena 2015). The implementation mechanism is as follows. The Designed ontologies described (Listing 1) above, are loaded in memory and a definition of the vocabulary of the defined URIs are created internally using Java code. For all external data-sources that contain data that can be retrieved for the aims of the system, an endpoint URL is used to define the data source (i.e Web Services, XML files or external Databases) internally. By accessing the data from external sources, the data retrieved 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" (Listing 2) mechanism that was implemented internally. The created virtual rdf's are mapped to the Ontology model and the defined SPARQL

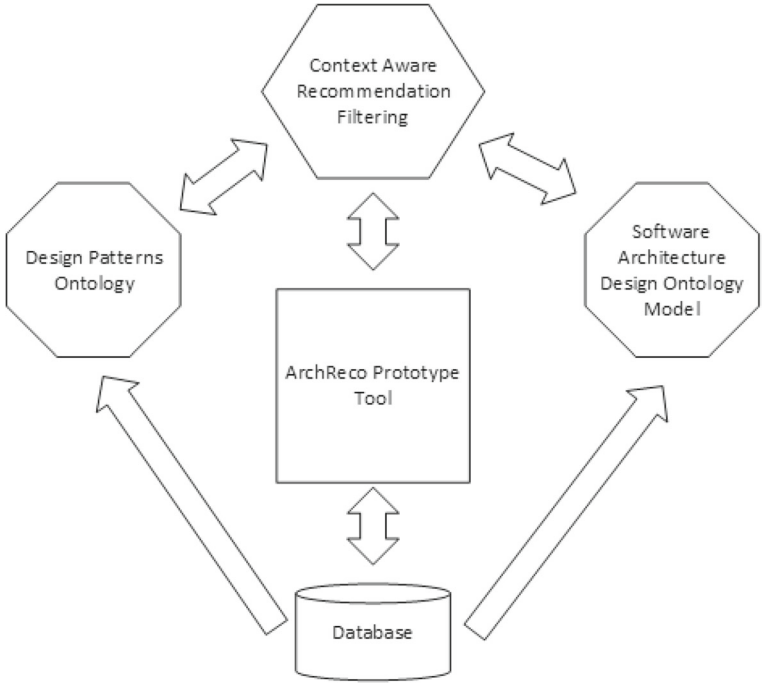

Fig. 3 ArchReco Prototype System Architecture The High Level Architecture of the prototype demonstrating the lifecycle of the components communications. Demonstration of how the semantic module intersects between the modules using the semantic mapping for the retrieval of data from multiple data sources

queries that parse the Ontology model are able to retrieve and filter the data from all the defined external sources.

Listing 1 OWL Example

```
<rdf:RDF xmlns="www.scst.com/ontology#"
     xml:base="&scst;ontology"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:scst="http://scst.com/"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:ontology="www.scst.com/www.scst.com/ontology#"
     xmlns:ontology2="&scst;ontology#">
    <owl:Ontology rdf:about=""/>

    <owl:ObjectProperty rdf:about="#areContainedBy">
        <owl:inverseOf rdf:resource="#containIdeas"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:about="#areCreatedBy">
        <owl:inverseOf rdf:resource="#createIdeas"/>
    </owl:ObjectProperty>
```

Listing 2 Data to Rdf Example

```
writer.openIndividual(" ", id,"j" , "Person");
if(null != userName )
{
writer.addLiteral("rdfs" , "label", userName , "xsd:string" ) ;
writer.addLiteral("j", "hasName" , userName , "xsd:string") ;
.
.
.
```

For the evaluation of the Semantic model and the usage of the multiple datasources, it was necessary to import data from more than two sources and execute the implemented SPARQL to make sure 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. To solve this problem and test the semantic model that was developed, we collected Data from multiple Design Patterns repositories, either using crawling functionality or by hand, and stored the collected data in local database and xml files. We applied the created testing data sources in our model and we confirmed that the retrieval of information using SPARQL queries through the designed model was properly delivering the expected results. For the release of the ArchReco software that was delivered to users for evaluation, the collected data were all transferred in the ontology file that is included in the release package, in order to avoid connectivity problems that would cause cold-start problems.

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 as it is shown in Fig. 4.

The example in Listing 3 shows that the data coming from external sources are transformed into virtual models and for each one of them a URI is defined. A prefix for each

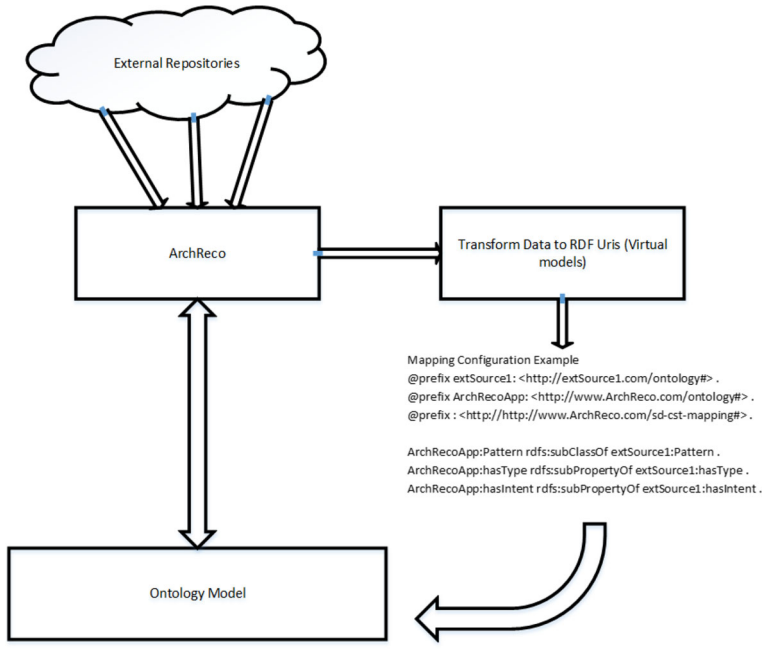

Fig. 4 Data retrieval and semantic mapping Communication of ArchReco with external Data Sources and the Semantic Mapping to the Ontology Model for the retrieval of data from multiple data sources

URI is defined and it is used to map data objects and data properties to the ones that the main ontology contains. Using the transformations and the corresponding mapping, it is possible to create queries to the main ontology model, and through this, retrieve data from the defined external or internal data sources.

Listing 3 Semantic Mapping Example

```
datafrommongo:hasName a owl:DatatypeProperty.
datafromxml:hasName a owl:DatatypeProperty.
datafromsql:UsersDatardfs:subClassOf myapp : Users .
datafromsql:hasFirstName rdfs:subPropertyOf myapp : hasName .
```

The designed diagram then can be extracted in several formats from which one of them is in *.MXE an XML based modelling language which can later be transformed into a UML diagram. The transformation of the MXE language into UML is not included in the scope of this work. A sample of the expected diagram is depicted in Fig. 5 while Fig. 6 depicts how the recommendations' content is presented to user. A sample MXE exported file is shown in Listing 4.

Listing 4 MXE 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>
```

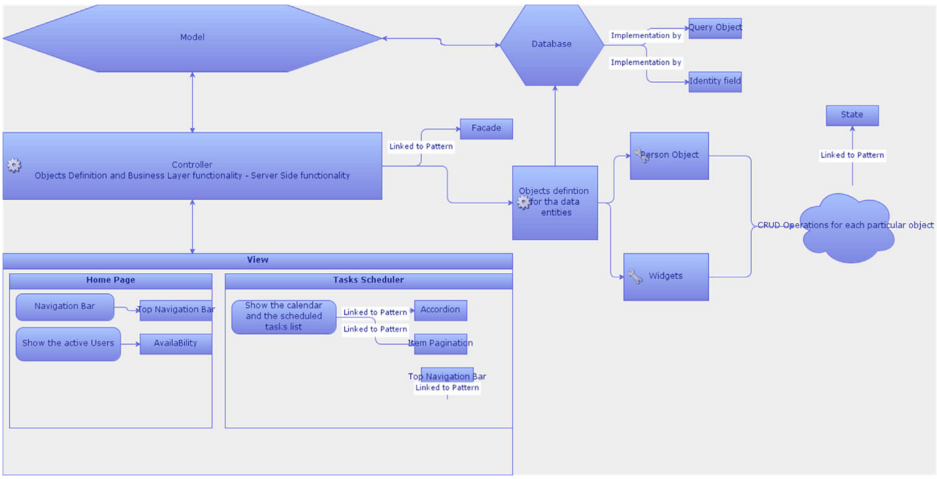

Fig. 5 Designed High Level Software Diagram with Design Patterns sample Sample of a designed High Level software diagram designed by the ArchReco prototype. The sample diagram represents an MVC based architecture where each layer of the architecture contains requirements description and the related Design Patterns that were selected for the implementation of the Requirements

Context aware recommendations component for design patterns

Taking into account existing approaches (Gomes et al. 2002; Guéhéneuc and Mustapha 2007; Weiss and Birukou 2007), this work suggests two methodologies from which one is a variation of an existing methodology and the other is used first time for the recommendation of Design Patterns.

The common attribute of the two methods is the use of context as the decision filtering factor but with different context elements in each case. Context Awareness in Recommendation Systems involves the use of data that characterizes an entity to be used as contextual information for the computation of recommendations, wherever this is needed.

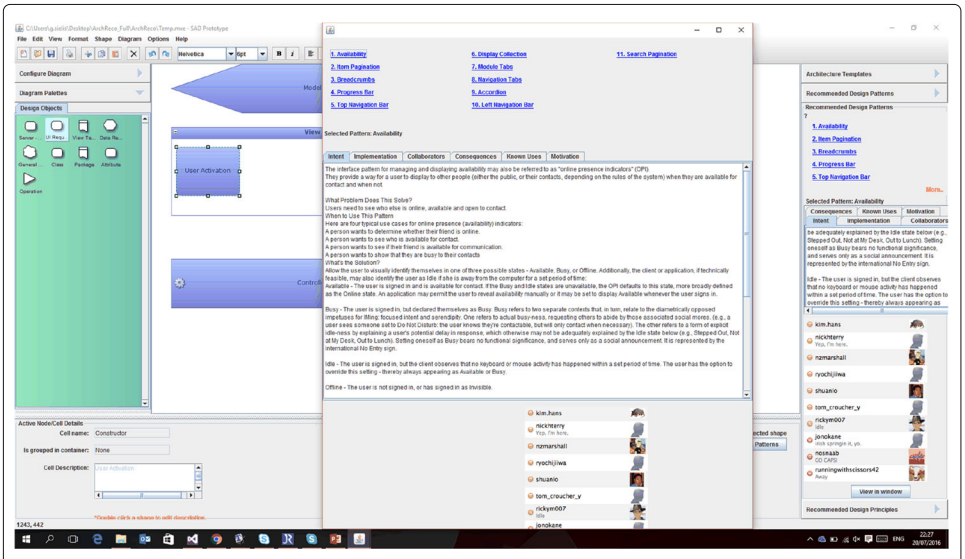

Fig. 6 ArchReco Prototype Sample of the ArchReco Interface with a Design Pattern content open after the recommendation was produced

Text-based recommendations for design patterns

Text Based recommendations are triggered when the user writes a description into a diagram shape. The input is natural 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 (Codina et al. 2013; 2016) 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 (McCandless et al. 2010)) which handles each pattern's attribute as indexed document. Then with the use of the TF-IDF (Wu et al. 2008) algorithm and the cosine similarity the input text is analyzed and compared with the indexed attributes of the Design Patterns subset and produce the recommendation ranked list, based on the calculated similarity.

Utility based recommendation for design patterns

Similar to the first method, Utility-based Recommendation uses text filtering for the computation of the recommendations. However, 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 Algorithm, 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 Eq. 1 and the form by which the users can define the contextual factors weights of importance is depicted in Fig. 7.

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

Where w_i is the weight and f_i is the factor.

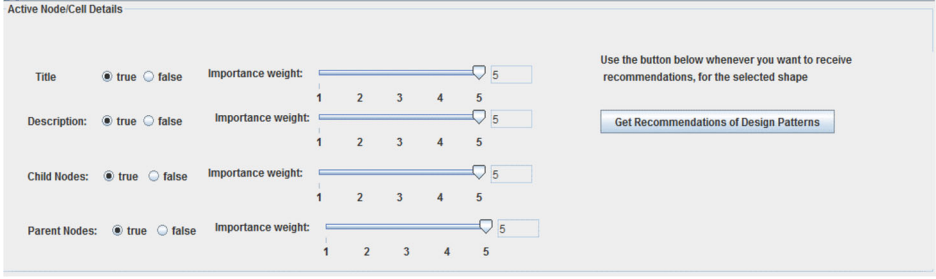

Fig. 7 Utility Based Recommendation System Ű Weights for Contextual factors input form Filters form that it is used for the definition of the Utility function input, defining the contextual elements and their importance weight values

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 (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 provided requirement description is processed and filtered by removing the stop words from the text and creating a words vector. 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 but 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, but 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 user defined for each participating contextual factor.

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 Patterns results that is produced by the pre-filtering is now reranked based on the weights of importance and thus the utility value of each Design Pattern. The computation used for the utility given in Eq. 1 defines a new rank for the Design Patterns results which reflect to the user's opinion regarding the utility of each Design Pattern.

A use case scenario

The adoption of a Design Pattern or a set of patterns relies on how knowledgeable the designer is regarding the existing design model. Based on the experience of designers, the following two cases may occur:

- An experienced designer who has a good knowledge of Design Patterns uses solely specific ones based on his prior knowledge. The designer applies patterns in the designed model based on experience from previous work, but does not apply any

alternative patterns that might be better suited for the model that is being currently developed.

- A junior designer without adequate knowledge of existing Design Patterns wants to use them, but does not know where to start from or which pattern is the right one to apply for a design model.

The usage of ArchReco prototype can be described through a use case scenario describing how ArchReco can be used as a Design Patterns training tool.

A group of students attend the Software Engineering course, at the second year of studies for the Computer Science Bachelor's degree. The course tutor gives an assignment to the students asking them to design a High Level Software Diagram for a web based application for an electronic book store. The application will support the books management (Add new book, edit existing books information and delete books). Additionally, the user management mechanism should exist. The application must be supported by a database for storing data as well as a logging mechanism for the collection of warning and error logs. The assignment description asks from the users to deliver a High Level Software design on which the Architecture should be shown and described and for each Architectural layer a set of Design Patterns should be defined and documented so the High Level software Design could be easily analyzed in a Low Level Software design which would be the next assignment. A student named Joe, uses the ArchReco software to produce the High Level software design and use its produced model to write the assignment report.

As a second year student, Joe has little experience in web design or web applications architecture models, and a very limited knowledge regarding Design Patterns. Joe opens a new blank diagram project using ArchReco and configures the new project by adding a general description that the model will be designed for. He defines the application type as a dynamic web-based application and saves the given information. By saving the configuration data Joe notices that on the right panel of ArchReco the "Architectural Templates" tab is automatically collapsed and a set of Architectural Pattern names are shown. The provided list contains Architectural patterns related to web based applications, such as MVC, MVP, MVVM and other. Joe, clicks on each name to read the content of the Architectural patterns, description, examples, images with UML samples. He examines all provided patterns and he decides that MVC Architectural Pattern is the most suitable for the aims of his assignment. He selects the MVC pattern and he presses the button "Add to Canvas". Automatically the MVC Architectural template appears in the canvas. Three shapes appeared with the Model - View - Controller names on each shape. Joe selects each shape-layer of the template and by each selection a set of Design Patterns is shown on the right panel, under the tab recommended Design Patterns. The recommended patterns at this stage are produced by taking into account the type of the selected layer. For example, with the selection of the View layer a set of UI Design Patterns is shown and with the selection of Model layer, Data-related Design patterns are produced. Joe decides to start the High Level design by the View layer. He double clicks on the View layer and writes a general description for the View layer which generally describes the view requirement for the assignment. Using the palette on the left side, Joe selects a View Template and drags the shape in the diagram. He connects the shape to the View layer and double

clicks the shape to add a more specific description for the View Template shape. He writes in the shape the phrase "Home Page". A new set of recommended UI design patterns appears. Joe examines one by one the recommended UI Design Patterns by reading their description, their intent, their coding examples (if exists) and preview the image with the UI Design Pattern as this is delivered from its source repository (i.e. yahoo UI Design patterns repository). Then Joe selects from the palette a "UI Requirement" shape and he adds it in the "Home Page" View Template. He double clicks on the shape and write in it "Main Menu". A set of Main Menu Design Patterns is produced and Joe reviews the recommended patterns one by one. He concludes to one of them and he adds the Design Pattern shape in the canvas by pressing the button "Add to Canvas". Automatically a new shape is added to the canvas and the new shape is connected to the "Main menu" UI requirement.

Joe repeats the process for the other two Architectural layers and he creates a High Level Diagram with the specified requirements described in natural language and a set of Design Patterns associated with each requirement. Joe extracts the designed diagram in .*png and *.xme file formats to use them in his report. The designed diagram contains the selected Architecture, the requirements and the Design Patterns that will be used for the Low-level analysis and later the development of the application. Joe learned the web-application related Architectures and he also learned about the Design Patterns he could use not only for the particular diagram but also other Design Patterns that he did not use in the current design. Now he is able to write his assignment report and explain his selections.

Using the above use-case scenario, the usefulness for a junior designer is obvious. However, the tool can also be useful for more experienced users since new Design Patterns can be recommended and be selected for further study. Without ArchReco users must search the web for examples and similar projects which in many cases becomes confusing and time consuming.

Methods

The following section describes the methods used for the evaluation of the ArchReco prototype. The development of the ArchReco prototype and its integration with the Context-Aware Recommendations for Design Patterns was given to real users for evaluation aiming to collect feedback from them and at the same time be able to track possible deficiencies for future improvements. Additionally, the evaluation's objective was to examine how the users perceive the prototype in terms of its educational character, the usefulness and usability. Therefore, the software was given to users who were asked to execute a specific task remotely, but during the execution of the task a screen capturing video was running for further analysis regarding their actual activities, while they were performing the task. In particular, with the evaluation process and the given questionnaires, the evaluation gave a significant set of results regarding the objectives that were set by the evaluation planning and follow the directions from other relevant evaluation methodologies (Chin et al. 1988; Davis 1989; Lund 2001; Lewis 1995).

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 evaluation and delivered to students and

researchers who are actively involved with Software Engineering and Software Design. The prototype developed as a stand-alone application but that was not a restrictive factor in executing the evaluation remotely. To make sure that the task given to the testers was completed in a proper way and to be able to track their reactions while they were using the software, a screen capturing of the evaluation sessions was running when the software was starting until its closure. The participants were asked to deliver the created video by the completion of their individual evaluation sessions. Additionally, related helping 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 screen capturing videos, 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 and development 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.

Evaluation frameworks

ResQue (Pu et al. 2011), 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' behavioural 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. (2002) 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 et al. (2012), 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 behaviour of the user.

Evaluation of the ArchReco prototype

The methodology used for the ArchReco prototype evaluation was mostly based on the ResQue framework (Pu et al. 2011). The questionnaire presented in (Pu et al.

2011), 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 (Lewis 1995), grouped based on the guidelines that were revealed from (Pu et al. 2011).

Evaluation setup

The evaluation process was designed as follows: The participants were asked to sign a Non-Disclosure form by adding their emails in an on-line form. The acceptance of the terms was redirecting the users to the evaluation page where the evaluation was described through the completion of a five steps process. *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 have also participated. From the total of 28 responses were received, 23 (83.9%) were from men and 5 (17.1%) were from women. The task that the participants were 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 is still open and accessible at the url: http://www.cs.ucy.ac.cy/~sielis.

Results and discussion

Pre-test questionnaire

As mentioned above the evaluation was performed by 28 people from which 23 were men and 5 were women. The responders were basically 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 4.

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 5 and the corresponding chart is shown in Fig. 8. As it is shown in Table 5 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 correlations between the questions from Table 5 based on

Table 4 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 Students	12	42.9%
	Msc Students	11	39.3%
	PhD Candidates & PhD holders	5	17.9%

the Individual grouping variables such as the gender, age, profession,nationality, location and year of studies.

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 participants were asked to rate 5 questions regarding the given task. The means and standard deviations for each question are shown in Table 6 (Fig. 9).

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

Table 5 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

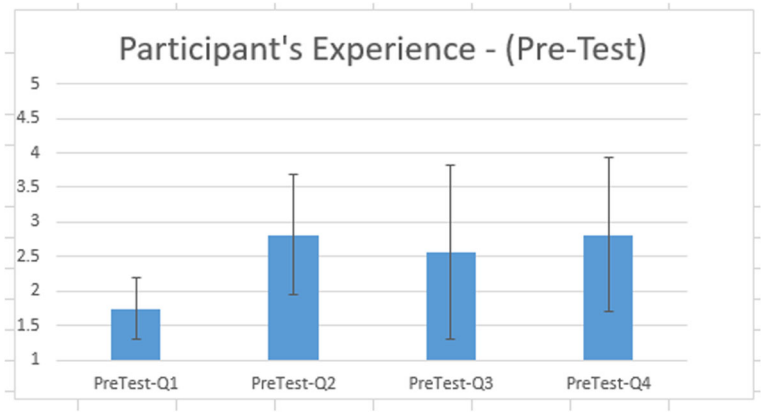

Fig. 8 Pre-Test responses for the Experience. Responses based on the 5-likert scale questions (1-lowest (negative) and 5-highest (positive) responses)

how the users perceived the task. In questions "*for the satisfaction in completing the given task*" (Qa) and the "*satisfaction with the amount of time needed for the completion of task*" (Qb), we notice that the average is 3.714 and the mode is 3. Having a standard 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 analysed in the rest of this section through the analysis of the screen capturing videos that we 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 the question "*if participants learned new things for Design Patterns*" (Qe) was 3.324, with mode value 4 is very encouraging but the high standard deviation attracts the attention for further analysis.

Table 6 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

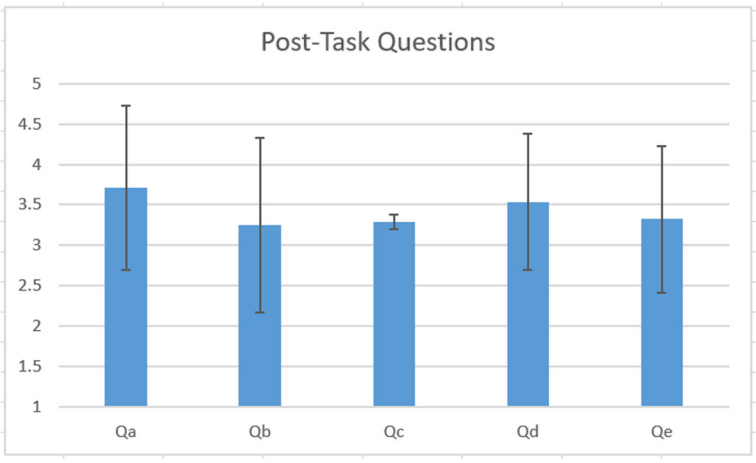

Fig. 9 Post-Task responses. Responses based on the 5-likert scale questions (1-lowest (negative) and 5-highest(positive) responses)

To reach into more safe conclusions that will help analysing 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, in commbination with the pre-test and post-test questionnaires, it is possible to extract and compare the mean values of the participants' experience. 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 7 and Fig. 10.

Post-test questionnaire

The post test questionnaire was organized to have questions, which would return results related 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 8 (Fig. 11). 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 8 it is shown that the tool was not evaluated with high scores on questions related to its effectiveness in completing a task fast ("*Using the tool in Designing Software models would enable me to accomplish tasks more quickly*" (Q1) and "*Using the tool to identify the most suitable Design Patterns would enhance my effectiveness on the job*" (Q4)). However, high scores received in questions such as "*Using the tool would improve my understanding in using Design Patterns in a high level software design model*" (Q2), "*it would make easier the process of design process*" (Q5) and "*the outcome of the tool would be benefic for the software developers who will implement the diagram's defined concepts*" (Q7), where the users identified the usefulness in understanding the Design Patterns and enhancement of the produced diagrams. High ratings for Q2,

Table 7 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

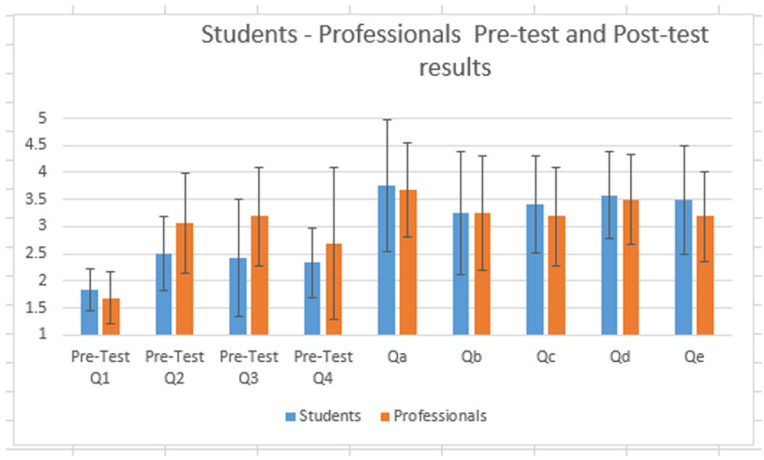

Fig. 10 Students & Professionals responses for pre and post-test questionnaires. Responses based on the 5-likert scale questions (1-lowest (negative) and 5-highest (positive) responses)

Q5 and Q7 are due to the flexibility that the tool offers to the user to add in the diagram Design Patterns that go beyond the existing knowledge in the topic. The low scores in the speed of completing tasks may be coused by three reasons. Firstly, due to the unfamiliarity with the software, which may have caused delays in completing the task; secondly due to the large amount of content that ArchReco provides for each Design Pattern that requires time for reading it; and third due to the fact that in most of the cases, professionals who have adequate background in Software Design and Design Patterns, prefer to apply the known Design Patterns in a designed model instead of reading new Design Patterns. Taking into consideration the prototype's training character and the usefulness questionnaire results, it can be concluded that the prototype is generally perceived as useful 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 above-mentioned thoughts.

From Table 9 and Fig. 12 it is noticeable that both groups have similar mean values in their responses. Small differences are noticed in questions about the effectiveness of the

Table 8 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 2	1.714	0.713	
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

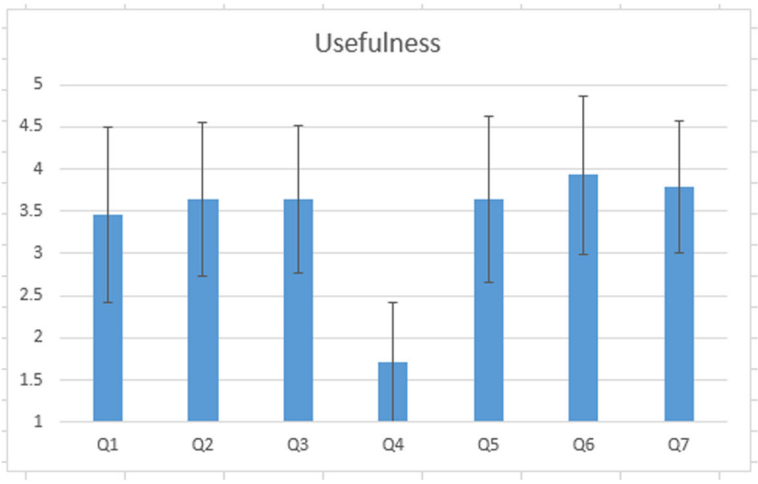

Fig. 11 Post-test responses for Usefulness. Responses based on the 5-likert scale questions (1-lowest (negative) and 5-highest (positive) responses)

tool in work (Q4) and usefulness of the tool in designing software diagrams (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 10 and Fig. 13 presents the results that were collected related to the functionality supported by ArchReco and how the users rated the supported functionality. The most important results from Table 10 are coming from two questions: "*The context-sensitive support (i.e. recommendations) is crucial to the ArchReco process*" (Q10) and "*I describe my experience with ArchReco tool in general as positive*" (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.

From the Table 11 and Fig. 14 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 "*The recommender*

Table 9 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

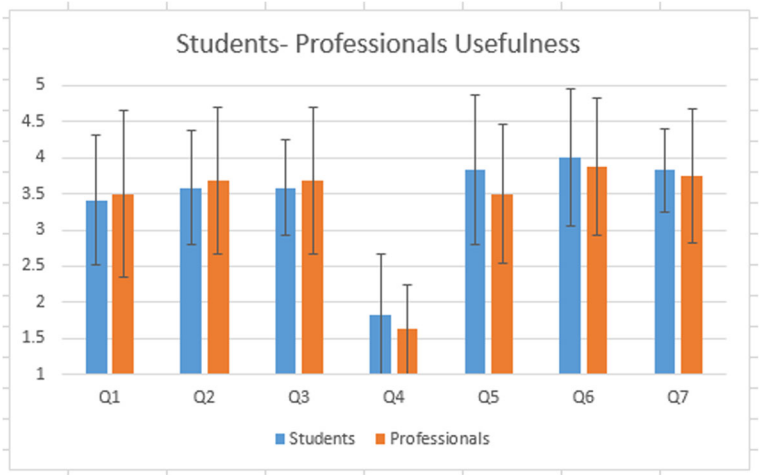

Fig. 12 Students & Professionals responses for Usefulness. Responses based on the 5-likert scale questions (1-lowest (negative) and 5-highest (positive) responses)

system is educational" (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 "*Recommendations for Design Patterns helped me learn new patterns*" (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 for future releases of the software.

Table 10 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	3.679	0.904	4
Q13. I describe my experience with ArchReco tool in general as positive	3.679	0.904	4

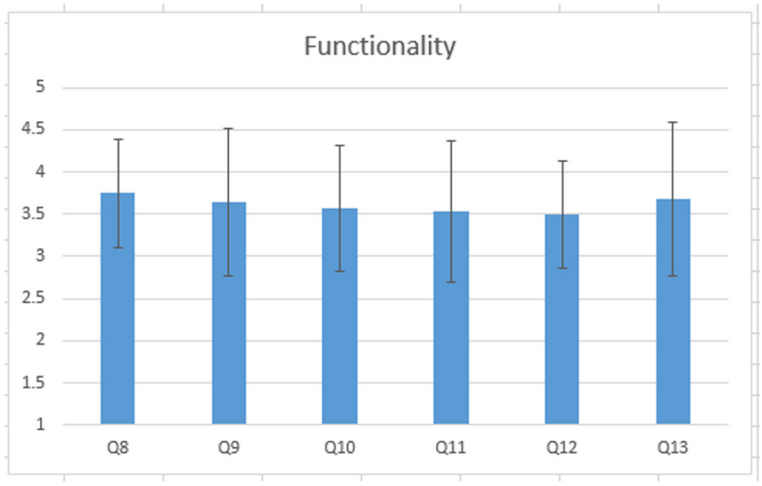

Fig. 13 Post-test responses for Functionality. Responses based on the 5-likert scale questions (1-lowest (negative) and 5-highest (positive) responses)

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

Table 11 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
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

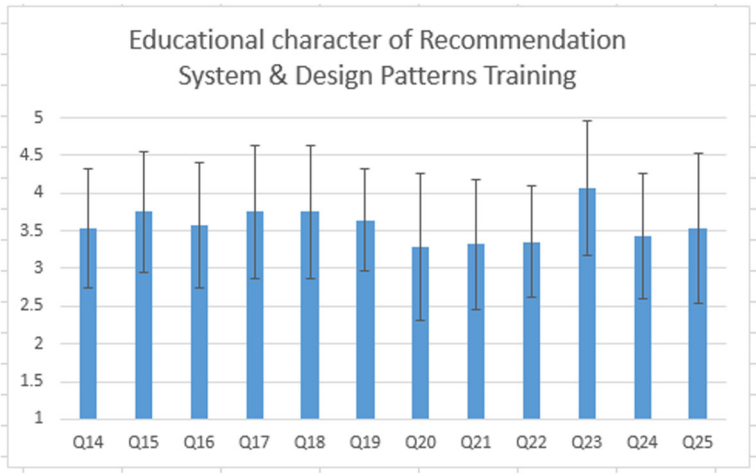

Fig. 14 Post-test responses for for Design Patters training and Educational Character of Recommendation System. Responses based on the 5-likert scale questions (1-lowest (negative) and 5-highest (positive) responses)

that were given by the evaluation web-page and they started by the configuration of the diagram that they were 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 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 space-holder 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*" were 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 therefore, in cases like this, it is better to hide the empty Design Pattern attributes when this is happening.

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 11. 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 (Eq. 2) - Recall (Eq. 3) measurements for the two recommendation algorithms.

Precision (Manning et al. 2008) is defined as the fraction of recommended items which are relevant and it is expressed as

$$Precision = \frac{|relevant\ items\ recommended|}{|items\ in\ the\ list|} \tag{2}$$

Recall (Manning et al. 2008) as the fraction of relevant recommendations that are presented to the user and it is expressed as

$$Recall = \frac{|relevant\ items\ recommended|}{|relevant\ items|} \tag{3}$$

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.

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 low. 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 executed repeatedly and for every repetition, one of the defined contextual factors 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.

Related work

The examination of Context Aware Recommendations for Design patterns related literature revealed a few research papers that refer to Recommendations for Design Patterns and less that refer to the usage of context as a means to compute recommendations for Design Patterns. There are several topics though related to the Design patterns research and their usage, which were examined in separate and they are presented in the rest of this section.

Design patterns in software architecture design

Hohpe et al. (2013) refers to the difficulty but also the necessity of finding good patterns while at the same time they identify the need for additional work in organizing and structuring the large body of existing Design Patterns. In the same work it is also mentioned that pattern centric design tools promise to be more appealing to the software engineers than mere component and connector drawing tools. In the literature one can find two types of patterns related to Software Engineering, the Architectural Patterns and Design Patterns. Design Patterns were introduced in (Alexander et al. 1977) as Software Engineering problems that may occur repeatedly, and they are associated with a solution that can be used to solve the problem every time it occurs within the current context that the problem exists. Architectural patterns are similar but with a wider scope. For example, more than one Design Pattern can be applied for specific Architectural patterns. Most of the publications related to Architectural Patterns are focusing in Architectural Styles and Views. Design patterns in terms of Software Architecture Design are generally met in papers related to Architectural Decisions (Capilla et al. 2016; Zimmermann 2012). In a more general perspective of Architecture design (Farenhorst and de Boer 2009) presents Design Patterns as part of the 4 views that consist of the Architecture Knowledge Management (AKM). Zimmerman et al. (2008) elaborates the combination of pattern and decision centric design in Software Architecture Design while (Harrison et al. 2007) presents methods for documenting decisions with patterns.

Beyond the fact that Design Patterns are part of Software Architecture Knowledge Management and considered an important Architectural Decision point, the current work is not focusing in defining the context of a Design Pattern by analyzing the very complex Architectural Decision Making process as it is presented by (Zimmermann 2012). 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 which will be used by the developers for coding the designed components.

Design patterns recovery

The last decade Design Patterns are commonly accepted and used in Software Engineering. After the Design Patterns usage in Software Engineering is matured enough, several frameworks and software platforms were developed with the use of Design Patterns. As mentioned in (Rasool and Streitferdt 2011) the flexibility in software maintenance and reusability motivated several researchers to develop Design Patterns recovery techniques. Examples of such techniques can be found in (Dong et al. 2009; Gueheneuc and Antoniol 2008; Lucia et al. 2009; Tsantalis et al. 2006) and in more recent work, such as Elaasar et al. (2015). The recovery of Design Patterns mainly aims to identify with high accuracy

the Design Patterns that were used in existing software tools. The recovery techniques are useful for the current work and especially for the evaluation of the current work's proposed tool in the future. In addition, Design Patterns recovery techniques can be used for the creation of datasets of Design Patterns for which the lack of such datasets is a known problem at the current stage of this work.

Design patterns & recommendation systems

Design Patterns for Software Design are recently 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 works have focused on providing recommendations on the appropriate usage of Design Patterns. Gueheneuc et al. (2007) 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. (2002) 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 was developed by (Weiss and Birukou 2007) 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. (2012) 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 is selected through LUCENE indexing and TF-IDF filtering of the query given 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 works 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. (2009) 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. (2007) 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 (Navarro and van der Hoek 2007) is a computer-based environment that facilitates the creation and simulation of realistic game-based software simulation models. SimSE is used as 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. 1994) a combination of two Software Engineering Educational tools is presented: The Object Oriented Turing (OOT) and Star. The software tool presented in (Mancoridis et al. 1994) is a targeted work 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 for editing, high speed compiling, linking, executing, and debugging OOT programs, as well as for browsing the Unix file system.

Another Game Based Educational tool for Software Engineering Training is presented in (Wang and Wu 2009), where the importance of games in education is emphasized and the work is targeting the education of the complex course of Software Engineering through a Game Development Framework (GDF). The difference between the GDF and other game based educational tools is the fact that the framework is used to create games. That 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.

The current work differs from the above approaches in the following aspects: *1. The recommendations methods used for the recommendation of Design Patterns*, *2. the development of a prototype tool that applies the proposed methods* and *3. Use of Semantic Web Technologies for modeling the context of Software Design process and for the retrieval of Design Patterns from multiple Design Patterns repositories*. In particular, Recommendation methods that were implemented for the aims of this work differ from the existing methods as per the type of recommendations, the implementation and the final usage. In particular, a text-based method was already used in similar research works (Gomes et al. 2002; Guéhéneuc and Mustapha 2007; Weiss and Birukou 2007) with the difference that in these works the text filtering was parsing only specific Design Patterns attributes such as the Implementation or the Intent. Utility based recommendations were not applied in other Design Patterns related work. The use of Utility-based recommendations is a well known method in Recommendation Systems research area but not commonly used for the current topic and context, which is the Context Aware Recommendations for Design Patterns for the training in using the Design Patterns when designing High Level Software Diagrams. Moreover, the combination of Semantic Web Technologies with Context Aware recommendations and their integration as a complete prototype solution for the training of students in learning the Design Patterns by practice consists of an original work in relation to the reported related work.

Conclusions

The work presented in this article describes the ArchReco software and how the combination of Semantic Web Technologies for modelling the Context of Software Design Process and specifically the representation of Design Patterns contextual information is used for the training by practice in the topic of Software Design. Context Aware Recommendations are used as enhancement tool for making that possible. The implementation of the described models and the development of the Semantic based Context Aware Recommendations within High Level Modelling Tools can become a useful training tool for

Computer Science and Software Engineering students. The work presented the semantic analysis and implementation of the examined models as well as the two Context Aware Recommendation algorithms that were used. It presents the ArchReco prototype environment and presented the results from a user based evaluation that was performed. The evaluation's results were analysed and this analysis showed the Usefulness of the tool, the perception of the users regarding the supported functionality and finally the impact of the tool and particularly the usage of the recommendation system as a training support system. The positive results as well as comments such as the following are motivating the continuation of the current work and the development of additional enhancements that will make ArchReco a professional training tool. "*I have studied many Design Patterns through a different way (reading a book from cover to cover and write notes on my Evernote account, which includes intent, description, UML diagrams etc. so that I can have a quick and accurate reference when it comes to decide how to solve a problem). I found that the recommendation system can be really useful. I think, it will encourage many people to use design patterns which they forget when they design a software*".

The results received from the current evaluation are also defining new objectives as future work. The most important one is the integration of the current system with personalized Context Aware Recommendations that will take into account prior knowledge of the users for the computation of the recommendations. This will be achieved with the usage of more sophisticated user based contextual analysis and collaborative recommendation filtering. Using the personalized Context Aware Recommendation results and the user based feedback for the produced recommendations, it will be possible to evaluate the produced results using false-positive analysis for the improvement of the results accuracy and the contextual refactoring in the cases that this might be necessary. Moreover, in the future more attention will be given to the improvement of the user interface design that will facilitate the overall user experience by using ArchReco. Finally, a transformation of the tool into a web based application is routed as well as the exposure of Design Patterns as a publicly accessible repository that currently do not exist and identified as a necessity for similar developments or additions to similar training modelling tools.

Abbreviations

AKM: Architecture Acknowledgement management; CBR: Case based reasoning; DPR: Design patterns recommendation; DB: Data-base; GDF: Game development framework; GoF: Gang of Four; ITI: Institute of information technology; MVC: Model view controller; MVP: Model view presenter; MVVM: Model view - view model; OOT: Object oriented turing; OWL: Web Ontology Language (W3C); SPARQL: Simple Protocol and RDF (Resource Description Framework) Query Language; TF-IDF: Term frequency inverse document frequency; UML: Unified modelling language; UI: User interface URI: Uniform resource identifier

Funding

No funding sources.

Availability of data and materials

The results of the survey mentioned in the Motivation Section, The evaluation responses and the screen capture videos are available from http://www.cs.ucy.ac.cy/~sielis/Results.zip. The ArchReco software is available in http://www.cs.ucy.ac.cy/~sielis/ArchReco.zip. The ontologies and the source code can be provided by communicating the authors.

Authors' contributions

GAS is the main contributor of the research work presented in this article. He performed the research, development and evaluation that is presented, and thus he is the correspondant author of the work. AT co-supervised the research work and contributed in the evaluation setup and evaluation results analysis. GAP co-supervised the overall research and contributed by guidelining the research process from begining to end. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Author details
[1]Department of Computer Science, University of Cyprus, University of Cyprus, P.O. Box 20537, 1678 Nicosia, Cyprus.
[2]Department of Design & Multimedia, University of Nicosia, University of Nicosia, P.O. Box 24005, 1700 Nicosia, Cyprus.

Received: 28 September 2016 Accepted: 5 April 2017
Published online: 26 April 2017

References

Alexander C, Ishikawa S, Silverstein M (1977) A Pattern Language: Towns, Buildings, Construction. Center for Environmental Structure Berkeley, Calif: Center for Environmental Structure series. USA, OUP USA. https://books.google.com.cy/books?id=hwAHmktpk5IC

Awais MA (2016) Requirements prioritization: challenges and techniques for quality software development. Adv Comput Sci Int J 5(2):14–21

Bayley I, Zhu H (2008) On the composition of design patterns. In: Quality Software, 2008. QSIC '08. The Eighth International Conference On. pp 27–36. doi:10.1109/QSIC.2008.32

Balaji S, Murugaiyan MS (2012) Waterfall vs. v-model vs. agile: a comparative study on sdlc. Int J Inf Technol Business Manag 2(1):26–30

Bottoni P, Guerra E, de Lara J (2009) Formal Foundation for Pattern-Based Modelling(Chechik M, Wirsing M, eds.). Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-00593-0_19, http://dx.doi.org/10.1007/978-3-642-00593-0_19

Briand LC, Morasca S, Basili VR (1999) Defining and validating measures for object-based high-level design. IEEE Trans Softw Eng 25(5):722–743. doi:10.1109/32.815329

Buschmann F, Meunier R, Rohnert H, Sommerlad P, Stal M (1996) Pattern-oriented Software Architecture: A System of Patterns. John Wiley & Sons Inc., New York

Capilla R, Jansen A, Tang A, Avgeriou P, Babar MA (2016) 10 years of software architecture knowledge management: Practice and future. J Syst Softw 116:191–205. doi:10.1016/j.jss.2015.08.054

Chin JP, Diehl VA, Norman KL (1988) Development of an instrument measuring user satisfaction of the human-computer interface. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '88. ACM, New York, pp 213–218. doi:10.1145/57167.57203, http://doi.acm.org/10.1145/57167.57203

Codina V, Ricci F, Ceccaroni L (2013) Semantically-enhanced pre-filtering for context-aware recommender systems. In: Proceedings of the 3rd Workshop on Context-awareness in Retrieval and Recommendation. ACM, pp 15–18

Codina V, Ricci F, Ceccaroni L (2016) Distributional semantic pre-filtering in context-aware recommender systems. User Model User-Adapted Interact 26(1):1–32

Cross N (2004) Expertise in design: an overview. Des Stud 25(5):427–441. doi:10.1016/j.destud.2004.06.002

Davis FD (1989) Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Q 13(3):319–340. doi:10.2307/249008

Dey AK, Abowd GD, Salber D (2001) A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Hum-Comput Interact 16(2):97–166. doi:10.1207/S15327051HCI16234_02

Dong J, Sheng Y, Zhang K (2007) Visualizing design patterns in their applications and compositions. Softw Eng IEEE Trans 33(7):433–453. doi:10.1109/TSE.2007.1012

Dong J, Zhao Y, Sun Y (2009) Syst Man Cybernet Part A Syst Humans IEEE Trans 39(6):1271–1282. doi:10.1109/TSMCA.2009.2028012

Elaasar M, Briand LC, Labiche Y (2015) Vpml: an approach to detect design patterns of mof-based modeling languages. Softw Syst Model 14(2):735–764. doi:10.1007/s10270-013-0325-9

Farenhorst R, de Boer RC (2009) Knowledge management in software architecture: State of the art. Softw Archit Knowl Manag:21–38. doi:10.1007/978-3-642-02374-3_2

Gamma E, Helm R, Johnson R, Vlissides J (1995) Design Patterns: Elements of Reusable Object-oriented Software. Addison-Wesley Longman Publishing Co. Inc., Boston

Gomes P, Pereira F, Paiva P, Seco N, Carreiro P, Ferreira J, Bento C (2002) Using cbr for automation of software design patterns. In: Craw S, Preece A (eds). Advances in Case-Based Reasoning. Lecture Notes in Computer Science, vol. 2416. Springer, Berlin Heidelberg, pp 534–548. doi:10.1007/3-540-46119-1_39, http://dx.doi.org/10.1007/3-540-46119-1_39

Guéhéneuc YG, Mustapha R (2007) A simple recommender system for design patterns. In: Proceedings of the 1st EuroPLoP Focus Group on Pattern Repositories

Gueheneuc YG, Antoniol G (2008) Demima: A multilayered approach for design pattern identification. Softw Eng IEEE Trans 34(5):667–684. doi:10.1109/TSE.2008.48

Harrison N, Avgeriou P, Zdun U (2007) Using patterns to capture architectural decisions. Softw IEEE 24(4):38–45. doi:10.1109/MS.2007.124

Hayes C, Massa P, Avesani P, Cunningham P (2002) An online evaluation framework for recommender systems. In: Workshop on Personalization and Recommendation in E-Commerce (Malaga). http://www.tara.tcd.ie/handle/2262/13178

Hou D, Hoover HJ (2006) Using scl to specify and check design intent in source code. IEEE Trans Softw Eng 32(6):404–423. doi:10.1109/TSE.2006.60

Hohpe G, Wirfs-Brock R, Yoder JW, Zimmermann O (2013) Twenty years of patterns' impact. Softw IEEE 30(6):88–88. doi:10.1109/MS.2013.135

Jena A (2015) A free and open source java framework for building semantic web and linked data applications. Available online: jena.apache.org. Accessed 28 Apr 2015

Knijnenburg BP, Willemsen MC, Gantner Z, Soncu H, Newell C (2012) Explaining the user experience of recommender systems. User Model User-Adapted Interaction 22(4-5):441–504. doi:10.1007/s11257-011-9118-4
Lewis JR (1995) Ibm computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. Int J Hum Comput Interact 7(1):57–78. doi:10.1080/10447319509526110
Lund AM (2001) Measuring usability with the use questionnaire. Usability Interface 8(2):3–6
Lucia AD, Deufemia V, Gravino C, Risi M (2009) Design pattern recovery through visual language parsing and source code analysis. J Syst Softw 82(7):1177–1193. doi:10.1016/j.jss.2009.02.012
Manning CD, Raghavan P, Schütze H (2008) Introduction to Information Retrieval. Cambridge University Press, New York
Mancoridis S, Holt RC, Godfrey MW (1994) Tool support for software engineering education. Technical report, Department of Computer Science, University of Toronto
McCandless M, Hatcher E, Gospodnetic O (2010) Lucene in Action, Second Edition: Covers Apache Lucene 3.0. Manning Publications Co., Greenwich
Munassar NMA, Govardhan A (2010) A comparison between five models of software engineering. IJCSI 5:95–101
Navarro EO, van der Hoek A (2007) Comprehensive evaluation of an educational software engineering simulation environment. In: Software Engineering Education Training, 2007. CSEET '07. 20th Conference On. pp 195–202. doi:10.1109/CSEET.2007.14
Palma F, Farzin H, Guéhéneuc YG, Moha N (2012) Recommendation system for design patterns in software development: An dpr overview. In: Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering. RSSE '12. IEEE Press, Piscataway, pp 1–5. http://dl.acm.org/citation.cfm?id=2666719.2666720
Pu P, Chen L, Hu R (2011) A user-centric evaluation framework for recommender systems. In: Proceedings of the Fifth ACM Conference on Recommender Systems. RecSys '11. ACM, New York, pp 157–164. doi:10.1145/2043932.2043962, http://doi.acm.org/10.1145/2043932.2043962
Rasool G, Streitferdt D (2011) A survey on design pattern recovery techniques. J Comp Sci 8(2)
Rola P, Kuchta D, Kopczyk D (2016) Conceptual model of working space for agile (scrum) project team. J Syst Softw 118:49–63
Tsantalis N, Chatzigeorgiou A, Stephanides G, Halkidis ST (2006) Design pattern detection using similarity scoring. Softw Eng IEEE Trans 32(11):896–909. doi:10.1109/TSE.2006.112
Wang AI, Wu B (2009) An application of a game development framework in higher education. Int J Comput Games Technol 2009:6–1612. doi:10.1155/2009/693267
Walz DB, Elam JJ, Curtis B (1993) Inside a software design team: knowledge acquisition, sharing, and integration. Commun ACM 36(10):63–77. doi:10.1145/163430.163447
Wagatsuma K, Harada T, Anze S, Goto Y, Cheng J (2016) A Supporting Tool for Spiral Model of Cryptographic Protocol Design with Reasoning-Based Formal Analysis(Park JJJH, Chao H-C, Arabnia H, Yen NY, eds.). Springer, Berlin, Heidelberg. doi:10.1007/978-3-662-47895-0_4, http://dx.doi.org/10.1007/978-3-662-47895-0_4
Weilkiens T, Lamm JG, Roth S, Walker M B: The v-model. Model-Based Syst Archit:343–352
Weiss M, Birukou A (2007) Building a pattern repository: Benefitting from the open, lightweight, and participative nature of wikis. In: International Symposium on Wikis (WikiSym). ACM, pp 21–23
Wu HC, Luk RWP, Wong KF, Kwok KL (2008) Interpreting tf-idf term weights as making relevance decisions. ACM Trans Inf Syst (TOIS) 26(3):13
Zimmermann O, Zdun U, Gschwind T, Leymann F (2008) 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. pp 157–166. doi:10.1109/WICSA.2008.19
Zimmermann O (2012) Architectural decision identification in architectural patterns. In: Proceedings of the WICSA/ECSA 2012 Companion Volume. WICSA/ECSA '12. ACM, New York, pp 96–103. doi:10.1145/2361999.2362021, http://doi.acm.org/10.1145/2361999.2362021