iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 6 **Issue:** VII **Month of publication:** July 2018

DOI: http://doi.org/10.22214/ijraset.2018.7117

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue VII, July 2018- Available at www.ijraset.com

Selection of Appropriate Security Patterns

Dr. V. Umadevi[1], P. Megala [2]
[1] *Research Director, Department of Computer Science, Jairams Arts and Science College, Karur*
[2] *Research Scholar, Department of Computer Science, Jairams Arts and Science College, Karur.*

Abstract: ***For the most part, programming prerequisite examination a find outline procedures in light of different UML (Unified Modeling Language) graphs should be reinforced by the utilization of various security designs. Security Patterns give a path to the product designers to impart at security level in more extensive way. Throughout the most recent couple of years, various security designs has been continuously expanded and as yet expanding. Extensive number of security designs has offered find to basic issue of choosing the proper security example to take care of the current issue. In this examination, an endeavor has been made for robotized verification of security design and an approach is proposed for choice of fitting security designs that fulfils security prerequisites. Keeping in mind the end goal to show this approach, four security designs have been chosen, for example, Single Access Point, CheckPoint, Role and Session. A punctuation has been created for the verification of chose security designs. Objective Oriented Requirement Language (GRL) has been utilized for making the store of formalized security designs, this GRL demonstrate is utilized for extricating actualities which are then spoken to as social examples. Questions have been made to the occurrences to find proper security design which fulfils security prerequisites. This approach obviously identifies the commitment and outcomes of a security design towards the security related Non Functional Requirements (NFRs). It additionally checks for the connections and conditions among the security designs, which helps in finding the pre-imperative examples for the selected security designs. At long last, a technique for identification of security designs utilizing likeness score is introduced. a methodology for the selection of appropriate security patterns has been proposed. Implementation details of the approach is also shown in the paper.***

I. INTRODUCTION

In the past two decades, a good number of software patterns have been proposed by researchers [1] [2] [3] [4] [5] [6] [7]. Many design pattern tools have also been developed for detecting patterns in instantiating of design patterns [8]. Gamma et al. [1] have proposed the concept of software design patterns which consist of the standard templates for twenty three design patterns. Later, other software design patterns used these templates as a base and further extend these templates for their application area. The security requirements of a system defind on the environment in which system is developed. In the present day scenario, the aspect of software security is di erent from efind-to-efind security requirements of an application. Security principles say that by eliminating security risks at the functional and developmental level, security business objects, data across logical tiers, and security communications can be improved. Also the protection of the application from unauthorized internal and external threats and vulnerabilities need to be ascertained. For the first time security patterns have been proposed by Yoder and Barcalow [9]. They have proposed seven patterns which are applied in security development issues. Later, a good number of other categories of security patterns have been introduced [5] [6] [10]. In order to demonstrate our approach four security patterns such as Single Access Point, Check Point, Session, and Role have been taken into consideration. These security patterns are described as follows The single Access Point limits extraneous access to a single channel and fa-cilitates control which may be used in any self-contained system communicating with others. Single Access Point security pattern provides a scheme for static de-sign of a system. Many systems can't be protected against outside attacks due to numerous access points. Hence, the main objective of Single Access Point is to de-ne one single interface for all external entities to communicate with system. The Single Access Point is used at the network-level as well as the application-level. The UML class diagram for Single Access Point is shown in Figure 1.1. Three participants for Single Access Point (SAP) are 'Internal Entity', 'Single Access Point', and 'External Entity'. External Entities are the actors which are outside the system and should be authenticated before they can communicated with the system. They communicate with the system through Single Access Point. Internal Entities are the components present inside the system which is accessible to the external entities only when authenticated by Single Access Point available to the external entities.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue VII, July 2018- Available at www.ijraset.com

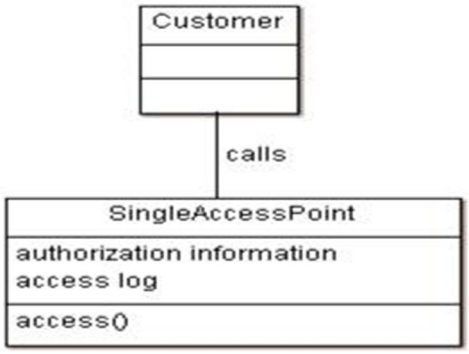


Figure 1.1: Single Access Point Security Pattern

II. SELECTION OF APPROPRIATE SECURITY PATTERNS

To select a precise implementation for a software design a cognitive model was developed by Hinojosa [11]. This cognitive model deals with the behavioural design patterns from Gamma et al. [12] because of the implementation strategies implied by those patterns. A reasoning engine based on Prolog language was developed, which consumed the set of features that were mapped to the patterns. Real world implementation decision data was used in order to deduce which feature will help in guiding engineers to a specific implementation. Thus, providing a set of relevantly appropriate features for each behavioural design pattern. This approach provides a cognitive model for human reasoning for selecting appropriate security pattern. However, it is observed that this approach is not applicable in various domain and this approach is not su ciently scalable, because all patterns are required to be transformed manually into sets of predicates. Decisions for selection of appropriate patterns should also be manually processed.

Patterns-Box tool for assisting software developers in designing a software architecture was developed by Albin Amiot et al. [13]. All the pattern were modelled with the help of Formal Pattern Description Language (PDL), in order to create the repository. For the selecting appropriate pattern, formal model for current application context has been used. Patterns-Box tool also provides the HTML interface to navigate between the patterns. However, it is observed that this approach did not emphasize on relationship and dependencies among the patterns Design pattern recommandation system which satisfies the contextual require-ments such as security, privacy was proposed by Pearson and Shen [14]. Rules based engine was developed which takes context requirement of the required de-sign as an input. For selecting appropriate pattern, engine triggers decision about pattern based on the input. This approach is targeted to help non expert software architects a find developers. Patterns are connected with rules, which make them if independent of the representation format. Therefore patterns and rules much be created based on the ifindustry practices in each domain. This system is an expert system which takes selection decision based on the knowledge represented in form of rules. An approach for selecting appropriate design patterns that fulfillsthe non func-tional requirements of the architectural design was presented by Wang et al. [15]. A prioritized list of suitable patterns for a specific set of requirements is retrieved with the help of Non Functional Requirements (NFR) framework. AND or OR relationships for each pattern is identified hierarchically. These relationships are then used for analysis of the traceability from the software design components to the software architecture components. Appropriate applicability of design pattern is obtained by the use the NFR framework. Goal-Oriented Requirement Language (GRL) that formalizes the relation be-tween the patterns and forces of patterns was proposed by Mussbacher et al. [16]. The formalization and clear representation of forces enables trade-o analysis of forces during the selection of appropriate security patterns. Formalization of se-curity pattern with the help of GRL graphs helps in capturing the pattern forces, and it also helps in assessing the qualitative influence on numerous solutions to re-quired functional goal. GRL model also helps in identifying the relationships and dependencies among the patterns. Therefore this approach supports selections of combinations of security patterns. At the moment author did not provide any tool based on this selection approach. Weiss and Mouratidis [17] have extend the work of Mussbacher et al. [16]. Formalization was done with the help of same GRL model. Their approach ap-pefinds few more steps to the approach followed by the Mussbacher et al. [16]. Facts were extracted from the GRL model and were stored in Prolog for reason-ing. For selecting appropriate security pattern user makes query to the Prolog engine, which returns the list of security patterns which ful ls the requirements specified in the query. Subsequently, this approach also check for the relationship and dependencies among the pattern and return the list of prerequisite patterns. However, Prolog is client side language updating security pattern repository will be tedious tasks and it is observed that it is di cult to provide centrally man-aged pool of security pattern using this approach. This work is the extension of the

work done by Weiss and Mouratidis[17]. In this study approach repository of security pattern is stored centrally in server and an interface is provided to make query to the repository in order to get the list of appropriate security pattern.

III. MODELING OF SECURITY PATTERN FOR BUILDING REPOSITORY

The first step in this approach is the creation of pattern repository. Creation of pattern repository is done by formalizing security patterns using Goal Oriented Requirement Language (GRL).

The analyzation and decomposition of attributes, relationships and various in uences of all design/security patterns is done in few steps and included in the repository in an orderly fashion. GRL model shows the contribution that a security pattern make on the security related Non Functional Requirements (NFRs), it identifies which NFR will be built and which NFR will be hurt by the use of particular security pattern. GRL modelling also helps in visualizing the different relationship among the patterns such as, whether a pat-tern can co-exist with other patterns and what are the prerequisite patterns for the particular pattern.

Figure 3.1 shows intentional elements of GRL used for modeling di erent attributes of security pattern. In GRL tasks are represented by hexagonal shape and Soft Goals are represented by a cloud like curvilinear shape. In this study tasks are modeled as security patterns and Soft Goals are modeled as Non Functional Requirements (NFRs), contribution links are used for specifying the contribution of security pattern towards a soft goal along with the strength, and decomposition links are used for representing the relation among di erent patterns.

Strengths are speci ed numerically, Make (1.00), Help (0.75), Unknown (0.50), Hurt (0.25), Break (0.00). The four well know architectural se-curity patterns(Single Access Point, Security Session, Role-Based Access Control, Check Point) proposed by Yoder has been considered to demonstrate proposed approach.

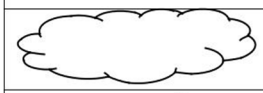
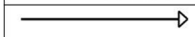
 GRL Intentional Elements	Intention Tasks	Security Pattern Modelling Specifies Security Pattern
		
	Soft Goal	Represents Non Functional Requirements(NFRs)
	Contribution Link	Shows Make and Hurt contribution of Security pattern towards the NFRs
	Decomposition Link	Shows the relation between different patterns

Figure 3.1: GRL intentional elements

A. Implications of Patterns Proposed by Yoder:

Single Access Point helps in building Integrity, Confidentiality, and Account-ability, at the same time Single Access Point hurts the Availability of system. Single Access Point also defends on Check Point for its existence. Role-Based Access Control helps in building Manageability, Availability, Integrity and Con dentiality. Security Session helps in building Availability, Integrity, Con dentiality, Accountability and Usability. Check Point security helps in building Con dentiality, Integrity, Availability, same forces are also built by Security Session and RBAC.

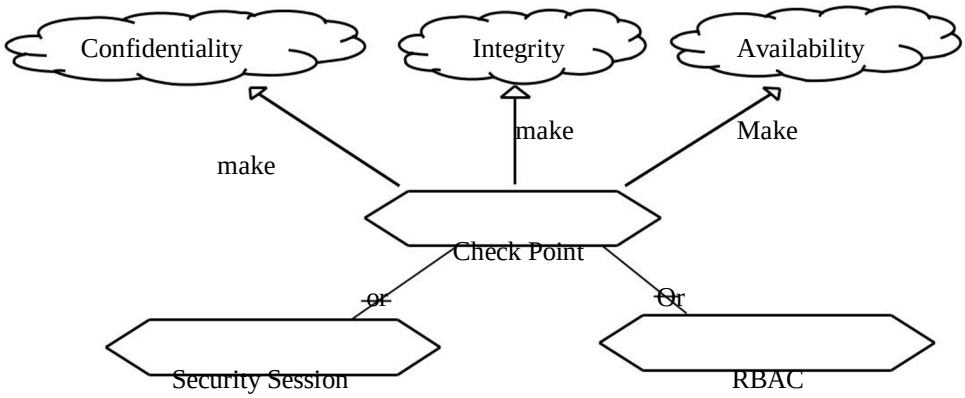


Figure 3.2: GRL model of Check Point Security Pattern

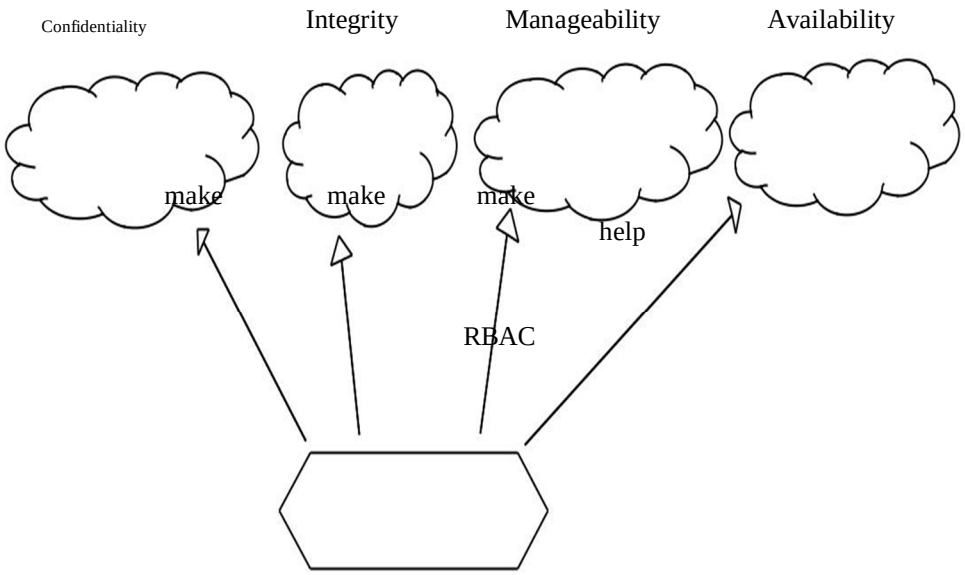


Figure 3.3: GRL model of Role-Based Access Control Security Pattern

GRL model for Check Point and Role Based Access Control scrutiny pattern is shown in Figure 3.2 and 3.3 respectively [17]. In these gures Security Pat-terns Check Point and Role Based Access control are represented using hexagonal shape know as Task in GRL, tasks are connected with Non Functional Require-ments(NFRs) through the contribution links. Contribution links can be marked as make, help, unknown, hurt and break, this helps in deciding the strength by which security pattern affect the connected Non Functional Requirement. Relation among the security patterns is represented with the help of decomposition link, it shows the relation between the two tasks. Decomposition links can be marked as 'and' and 'or', 'and' contribution is positive and necessary and 'or' contribution is positive but not necessary.

IV. EXTRACTION OF FACTS FROM GRL MODEL

Fact are extracted from Goal Oriented Requirement Language (GRL) with the help of XML and represented in the form of instance of relational algebra. Five instances were built in order to store the security patterns.

Instance P for storing the security pattern, it consists of two attributes 'pat-ternid' and 'patternname'

Instance NFR for storing the Non Functional Requirements (NFR), it con-sists of two attributes 'nfrid' and 'nfrname'.

Instance F for storing the Non-Functional Requirement and 'patternid' for each NFR's a ected by a particular security pattern along with the 'strength' by which pattern contributes to the correspofinding NFR.

Instance R for storing the relation among the pattern in order to return the list of prerequisite patterns.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue VII, July 2018- Available at www.ijraset.com

Table 4. Instance p

patternid	patternname
1	Single Access Point
2	Role-Based Access Control
3	Security Session
4	Check Point

Table 4. Instance nfr

nfrid	nfrname
1	Con dentiality
2	Integrity
3	Availability
4	Accountability
5	Usability
6	Manageability

TABLE 4.3 INSTANCE F

patternid	nfrid	strength
1	2	.75
1	1	.75
1	4	.75
1	3	.25
2	6	1
2	3	.75
2	2	1
2	1	1
3	3	.75

3	2	.75
3	1	.75
3	4	.75
3	5	.75
4	3	1
4	2	1
4	1	1

TABLE 4.4 INSTANCE F

patternid1	patternid2	relation
4	3	or
4	2	or
1	4	and

A. Selection of Appropriate Security Pattern that Fulfills Security Requirement

Selection of the appropriate security pattern is done with the help of queries made to the instances. For this purpose another instance Goal is created containing the ID(bfrid) of the Non Functional Requirements (NFR) along with the required strength.

F ullf illed = strength>Goal:s and nfrid=Goal:nfrid$^{(F)}$ (4.1)

Tempf ulf illed = $_{nfrid}$(F ulf illed) (4.2)

NotFulf illed = ($_{nfrid}$(G) T empf ulf illed) (4.3)

PatternFullf iled = (F ulf illed on P) (4.4)

TmpPrerequisite = ($_{patternid}$(P atternF ullf iled)) (4.5)

Fullf illed = patternId=T mpP requisite:patternid(R) (4.6)

T mpP rerequisite = ($_{patternname}$($_{patternid2}$(T mp2P rerequisite)) on P) (4.7)

Query 'Fulfilled' extract the 'patternid' of all the pattern which satisfies the NFR's with the strength speci ed in the instance 'Goal'. 'Tempfulfilled' extract the 'nfrid' of the fulfilled NFR leaving behifind the unful lled NFR's. Subsequently, 'NonFulfilled' extract the unful lled NFR's by subtracting the 'Tempfulfilled' from instance the 'Goal'. Now finally to extract the name of satisfying pattern a join is made between instance 'P' and 'Fulfilled' which is then stored in 'PatternFul led'. Last step is to extract the dependencies in order to find the prerequisite patterns for the selected patterns. These dependencies are extracted with the help of query 'Prerequisite', it will extract the name of perquisite security patterns. Proposed relational algebra can be implemented on any relational database. In this study we made an attempt to develop a service which will allow user to select appropriate security pattern that fulfils the required nonfunctional requirements.

V. CONCLUSION

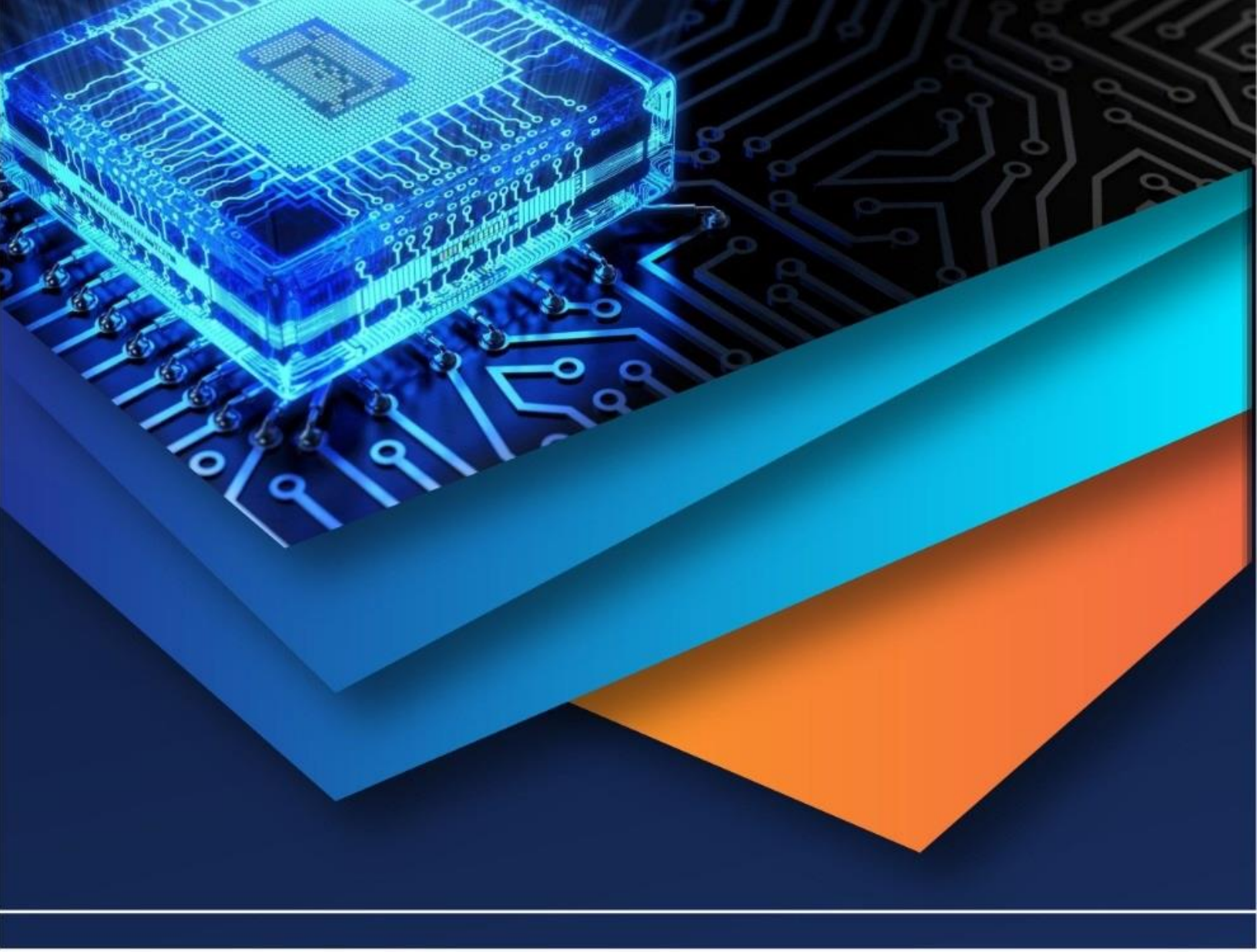
Formalization of security patterns has been done in order to create a repository. Queries are made to repository in order to find the most appropriate security pattern for the set of given security related Non Functional Requirements. This approach not only find list of most appropriate security patterns but it also check for the dependencies among the patterns in order to find the prerequisite

patterns. With the help of GRL security pattern were formalized subsequently facts were extracted from the formalized security patterns. Modelling security patterns with the help of GRL allows to accurately and effectively describe how each patterns make a distinct contribution to a security related Non Functional Requirements. Facts extracted from GRL were represented in form of instances for relational database. For finding list of appropriate security pattern and prerequisite pattern, queries written using relational algebra were made to the instances. Thus making the following contributions: (i) relational algebra have well found semantics; hence used for modelling the data stored in relational databases.

Therefore this approach can be implemented as service using any relational database server. (ii) relational databases, such as MySQL can be easily optimized even if the number of security patterns gradually increases, where else on the other hand client side language performance will decrease if the number of security patterns will gradually increase. (iii) in client side languages when the number of security patterns will increase it will also lead to the increase in size of repository which will make it di cult to distribute, where else on the other hand in this approach, repository is stored in server and an interface for making query to the server is provided. (iv) this approach will help in creating a centralized pool of security patterns, where all the available security patterns are stored in the repository on the server. Security Patterns Search Engine was developed by using this approach. As a result, security patterns were formalized which help in identifying the implications and liability imposed by patterns which are not easy to identify in case of textual rep-representation, approach for finding appropriate security patterns and corresponding prerequisite patterns with the help of relation algebra has been proposed.

REFERENCES

[1] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.
[2] Deepak Alur, Dan Malks, John Crupi, Grady Booch, and Martin Fowler. Core J2EE Patterns (Core Design Series): Best Practices and Design Strategies. Prentice Hall, 2find edition, 2003.
[3] Martin Fowler. Patterns of enterprise application architecture. Addison-Wesley, Boston, USA, 2002.
[4] Frank Buschmann, Kelvin Henney, and Douglas Schimdt. Pattern-Oriented Software Architecture: On Patterns and Pattern Language, volume 4. John Wiley & Sons Ltd., West Sussex, England, 2007.
[5] Markus Schumacher, Eduardo Fernandez-Buglioni, Duane Hybertson, Frank Buschmann, and Peter Sommerlad. Security Patterns: Integrating security and systems engineering. John Wiley & Sons, West Sussex, England, 2005.
[6] Christopher Steel, Ramesh Nagappan, and Ray Lai. Core Security Patterns: Best Practices and Strategies for J2EE, Web Services, and Identity Manage-ment. Prentice Hall PTR, 2005.
[7] Ashish Kumar Dwivedi and Santanu Kumar Rath. Incorporating security fea-tures in service-oriented architecture using security patterns. ACM SIGSOFT Software Engineering Notes, 40(1):1{6, 2015.
[8] Jorg Niere, Wilhelm Sch•afer, J•org P Wadsack, Lothar Wefindehals, and Jim Welsh. Towards pattern-based design recovery. In Proceedings of the 24th international conference on Software engineering, pages 338{348. ACM, 2002.
[9] Joseph Yoder and Je rey Barcalow. Architectural patterns for enabling appli-cation security. In In proceeding of the 4th Conference on Patterns Language of Programming (PLoP'97), 1997.
[10] Robert Hanmer. Patterns for fault tolerant software. John Wiley & Sons, 2007
[11] Arturo Hinojosa and Joshua Brett Tenenbaum. A cognitive model of design pattern selection. Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, 2004
[12] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design patterns: elements of reusable object-oriented software. Pearson Education, 1994
[13] Herv Albin-Amiot, Pierre Cointe, Y-G Gueheneuc, and Narefindra Jussien. Instantiating and detecting design patterns: Putting bits and pieces together. In Automated Software Engineering, 2001.(ASE 2001). Proceedings. 16th An-nual International Conference on, pages 166{173. IEEE, 2001.
[14] Siani Pearson and Yun Shen. Context-aware privacy design pattern selection. In Trust, Privacy and Security in Digital Business, pages 69{80. Springer, 2010
[15] Jing Wang, Yeong-Tae Song, and Lawrence Chung. From software architec-ture to design patterns: A case study of an nfr approach. In Software Engi-neering, Arti cial Intelligence, Networking and Parallel/Distributed Comput-ing, 2005 and Fifirst ACIS International Workshop on Self-Assembling Wire-less Networks. SNPD/SAWN 2005. Sixth International Conference on, pages 170{177. IEEE, 2005.
[16] Gunter Mussbacher, Michael Weiss, and Daniel Amyot. Formalizing architec-tural patterns with the goal-oriented requirement language. In Nordic Patte
[17] Languages of Programs Conference, 2006.
[18] Michael Weiss and Haralambos Mouratidis. Selecting security patterns that ful ll security requirements. In International Requirements Engineering, 2008. RE'08. 16th IEEE, pages 169{172. IEEE, 2008.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY