SecREAD: Security-aware Requirements Elicitation, Assessment and Design Methodology

Ph.D. Thesis

RAJAT GOEL
ID No. 2013RCP9052

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY JAIPUR

October 2018

SecREAD: Security-aware Requirements Elicitation, Assessment and Design Methodology

Submitted in

fulfillment of the requirements for the degree of

Doctor of Philosophy

by

RAJAT GOEL

ID No. 2013RCP9052

Under the Supervision of

Prof. Mahesh Chandra Govil

Dr. Girdhari Singh

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY JAIPUR

October 2018

Declaration

I, **RAJAT GOEL**, declare that this thesis titled, **"SecREAD: Security-aware Requirements Elicitation, Assessment and Design Methodology"** and the work presented in it, are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this university.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this university or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself, jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Date: Rajat Goel

(ID: 2013RCP9052)

Certificate

This is to certify that the thesis entitled **"SecREAD: Security-aware Requirements Elicitation, Assessment and Design Methodology"** being submitted by **RAJAT GOEL (2013RCP9052)** is a bonafide research work carried out under our supervision and guidance in fulfillment of the requirement for the award of the degree of **Doctor of Philosophy** in the Department of **Computer Science and Engineering**, Malaviya National Institute of Technology, Jaipur, India. The matter embodied in this thesis is original and has not been submitted to any other University or Institute for the award of any other degree.

Supervisiors

(Prof. Mahesh Chandra Govil)	**(Dr. Girdhari Singh)**
MNIT Jaipur	Associate Professor
(Presently Director, NIT Sikkim)	MNIT Jaipur

Place : Jaipur

Date:

Acknowledgment

I express my sincere gratitude to my supervisors, Prof. M.C. Govil and Dr. Girdhari Singh for providing valuable guidance throughout my research work. It is pertinent to mention that Prof. Govil spared countless hours for me, out of his busy schedule, throughout my research journey. Moreover, the co-operative and serene attitude of Dr. Girdhari Singh facilitated me to accomplish my task.

I owe great thanks to the DREC members Dr. E.S. Pilli, Dr. Dinesh Gopalani and Dr. Yogesh Kumar Meena for their useful insights to shape my work.

I extend my thanks to the DPGC convener Dr. Namita Mittal, faculty members and staff of Computer Science and Engineering department for their cooperation.

I am thankful to MNIT for providing me the necessary infrastructure and conducive environment to expedite my work.

I may also like to thank my fellow scholars and friends for fruitful discussions as also for keeping me rejuvenated during this period.

It would also not be out of place to mention the names of Dr. Nidhi Govil, Devang & Prajjwal for providing me a homely atmosphere, necessary for any researcher.

Finally, I would like to earnestly reveal that my parents showered on me countless blessings for my success and my sister provided necessary moral support all throughout. The sincere role of my wife, during the latter part of my research work is also worth mentioning, who kept me free from all domestic responsibilities so that I may concentrate on my goal.

Date: Rajat Goel

(ID: 2013RCP9052)

Dedication

Dedicated to my beloved late Grand Father

Abstract

Software is now a part and parcel of our lives and involves sensitive and personal information. This makes security an integral part of software. There exist a multitude of models and methodologies to develop a software but a very few of them concerntrate on integrating security in the development process. The software today are in a dire need of such security-based development process models as the general practice of treating security as an add-on can jeopardize the success of software. The research has been carried out to address this critical issue. The expansive literature survey conducted during the course of the research has revealed that security measures ought to be imbibed in the early phases of developemnt *i.e.* requirement and design. From a systematic and critical analysis of the prior works, their positive aspects are assimilated and enhanced to fill in the research gaps by introducing novel concepts. This has led to the contemplation of a holistic process, "Security-aware Requirements Elicitation, Assessment and Design (SecREAD) Methodology", proposed in this research work.

Requirements and Design phases pose many challenges while integrating security. Obtaining the precise and clear requirement specification is indispensable but quite difficult to achieve. Similarly, modeling the requirements or their diagramatic representation is essential for unambiguous communication between the users and the development team. Further, for modeling the requirements effectively, an efficient modeling language is required. But, the existing modeling languages do not suffice in this regard. In the light of all the above factors, SecREAD methodology is proposed that attempts to overcome all the above problems.

SecREAD puts forward a formal process to elicit and model the requirements. It provides a range of diagrams to model the security concerns of the software. The methodology allows gathering of requirements in natural language in the form of Stories, facilitating the involvement of all kinds of stakeholders that may or may not be technically conversant. These stories are represented graphically. For this a notion of Story Conversion Diagram (SCD) has been introduced that provides a clear and unambiguous view. From the stories entities viz. stakeholders, assets and functionalities are identified. The entities undergo rigorous refinement process to remove inconsistencies and ambiguities, if any. Refinement is conducted at different levels and every time the SCDs simplify the process. A mapping mechanism then establishes the associations among the entities. The previously obtained SCDs are modified to reflect these associations. The stakeholders rank the assets for security on pertinent parameters. Mapping ensures that the stakeholders rank their relevant assets only. This makes the ranking more realistic. Following the ranking process, empirical analysis is conducted to consolidate the ranks given by different stakeholders and produce final security ranks for every asset and functionality of the software. This rank information is accomodated in the SCDs to produce the rank diagrams. All diagrams are coupled with textual templates which further minimize the ambiguities. In this manner, SecREAD attempts to formalize the development process. Furthermore, SecREAD envisages the continual involvement of client and experts with the developers to smoothen the process.

The concepts of mapping, empirical analysis and such type of diagrams is unique to this methodology. These diagrams are incremental in nature that grow with the information obtained at various levels, making them easy to draw and understand.

In this research work, the proposed methodology has been applied on two case studies namely, Internet Banking and Smart Building to demonstrate its applicability in varied domains.

Contents

List of Tables

List of Figures

Acronyms and Abbreviations

SecREAD	Security-aware Requirements Elicitation, Assessment and Design
INB	Internet Banking
UML	Unified Modeling Language
SRS	Software Requirements and Specification
RE	Requirements Engineering
COM	Component Object Model
CBSE	Component-based Software Engineering
COTS	Components-off-the-Shelf
RAD	Rapid Application Development
WYSIWYG	What You See is What You Get
RUP	Rational Unified Process
ICSM	Incremental Commitment Spiral Model
CBA	Cots-based Application
FISMA	Federal Information Security Management Act
OMG	Object Management Group
XP	Extreme Programming
CLASP	Comprehensive Lightweight Application Security Process
SDL	Security Development Lifecycle
SaSDLC	Security-aware Software Development Life Cycle
IRE	Iterative Requirement Elicitation
GSD	Global Software Development
RiskREP	Risk-based Security Requirements Elicitation and Prioritization
CC	Common Criteria
EAL	Evaluation Assurance Level

PREview	Perspective Requirements Engineering View
CORE	Controlled Requirement
UC	Use Cases
BP	Business Process
FODA	Feature Oriented Domain Analysis
IBIS	Issue-based Information System
JAD	Joint Application Design
REVU	Requirements Visualization of UML
FGDRE	Focus Group Discussion for Requirements Elicitation
ADORA	Analysis and Description of Requirement and Architecture
TSP	Team Software Process
SCD	Story Conversion Diagram
CSCD	Consolidated Story Conversion Diagram
FRD	Functionality Rank Diagram
CRD	Comprehensive Rank Diagram
ARD	Authorization Rank Diagram
WP	Write with Permission
FD	Fixed Deposit
OTP	One Time Password
NEFT	National Electronic Funds Transfer
RTGS	Real Time Gross Settlement
IoT	Internet of Things
CCTV	Close Circuit Television
HVAC	Heating, Ventilation and Air Conditioning

Chapter 1

Introduction

“Our civilization runs on software.”

- Bjarne Stroustrup [1]

In today’s era of ever growing digitization, software is becoming a part of every aspect of human lives. Software are applied in varied domains like banking, bill payments, atomic energy, medical, ticket-booking, shopping, entertainment etc. They may involve flow of finances and sensitive information. Undoubtedly, software play a significant role and in this context it is indispensable to make them secure. Researchers have put in a lot of efforts in this regard and have concluded that security ought to be imbibed in the development process itself.

According to B. W. Boehm [2], Software engineering is the application of science and mathematics by which the properties of software are made useful to people. Although, numerous models and methodologies are available today but to develop a software, very few of them deal with security concerns. The developers usually focus on functional requirements considering nonfunctional requirements like security at the end. This approach compromises the security. The extensive literature survey has revealed that security is best imbibed in the early phases of Software Development Life Cycle (SDLC) *i.e.* requirements and design. In the proposed Security-aware Requirements Elicitation, Assessment and Design (SecREAD) Methodology, the requirements engineering is formalized by eliciting requirements by all kinds of stakeholders. The stakeholders can be very varied in their technical knowledge. To accommodate this, the proposed methodology allows narration of requiremnts even in natural language. From these requirements the various entities are obtained and mapped to each other according their relevance. The design process is based on an intricate ranking mechanism and empirical analysis. The diagrams are developed as per the analysis. These diagrams are easy to draw and they very lucidly illustrate the security concerns of the software. The diagrams are incremental in nature that grow with the information obtained at various levels. SecREAD attempts to overcome the limitations of existing methodologies while assimilating their positive aspects and imbibes the good practices as suggested by different researchers. To substantiate and demonstrate the methodology, it has been applied on the two case studies viz. Internet Banking (INB) and Smart Building.

In the upcoming sections, an introduction to various conventional and popular Software Development Life Cycle (SDLC) models is presented to make better understanding of security-oriented models and methodologies that follow in Chapter 2. Unified Modeling Language (UML) too is described along with its advantages and drawbacks. The chapter also includes the problem & motivation and the objectives.

1.1 Software Development Life Cycle (SDLC)

There are many definitions of SDLC. According to Janczura and Golińska [3], "It covers the time span from the point of realizing the necessity for creating a system, to the moment of its decommission." A Software Development Life Cycle (SDLC) is a systematic process that helps to ensure the successful development, operation, and retirement of information systems [4]. R. S. Pressman [5] considers it a methodology to build a software of high quality. A software development process is the collection of activities to be performed to build a software system. The organisation of software process is given by a process model. It broadly consists of the requirements analysis, design, development, testing and maintenance steps. According to I. Sommerville [6] the fundamental activities of software development process are the Software Specification that defines its functionalities, Design, Implementation, Validation to check whether the software does what the customer wants and Evolution that concerns with the evolution of software to accommodate customer's changing needs.

Earlier researchers were not convinced on whether adherence to the existing process models leads to the development of a secure product. Several security-oriented models have been proposed by various researchers, discussed in Chapter 2. However, for easier understanding of those, becoming conversant with the classical and established models will be useful. Hence, these models have been discussed briefly in the following sub-sections.

1.1.1 Waterfall Model

Waterfall model is a well-known classical sequential model. In a sequential model of this kind firstly the requirements are gathered and analyzed followed by design, implementation & testing, installation and maintenance phases. The phases are clearly separated and well-organized. The description of the different phases is as follows.

- Requirements Elicitation, Analysis and Specification: Requirements are gathered correctly from the customer and communicated to the developers of the system. They are then specified and the outcome of this phase is a Software Requirements and Specification (SRS) document which serves as a baseline for rest of the development process. A.V. Lamsweerde [7] calls it Requirements Engineering (RE) that deals with the elicitation of objectives to be achieved, specifying them, assigning responsibilities to the software, devices and/or humans and the evolution of requirements over the time.
- Design: The software system abstractions and their relationships are identified and detailed. Design may be depicted graphically by various diagrams using a modeling language.
- Implementation and Unit Testing: The design is converted into system parts called program units. Each of these units is tested individually to check whether it meets the requirement specifications.
- Integration and System Testing: After testing the individual units, all the units are integrated to form the system and tested again to ensure that it meets the requirement specifications.
- Deployment and Maintenance: The system is deployed or installed at the customer's site and its usage begins. During maintenance, those errors are corrected which remained undiscovered earlier. The system may be enhanced for newer technologies.

This methodology is easy to use with clear milestones that are good for planning, staffing and tracking. However, all the requirements are frozen at the very inception itself. For changing requirements in later stages, one has to go back to the previous phase(s). It is well-known that if the error is detected at a later stage in the development cycle, more expensive it becomes to handle. Moreover, there is no provision for consulting the client with a working version at an early stage. Also, many members of the project team remain unoccupied for long periods. Still this kind of sequential models are used because of the simplicity they offer in managing a project. The model is suitable for small or middle-sized projects with well-understood technology where changes are limited. It is a good option for developing a new version, porting to a new platform and quality is the priority over schedule or cost. [3][8]

Iterative Waterfall Model is an improved and practical variation of Waterfall model that provides option for backtracking to previous phases.

1.1.2 V-Shaped Model

V-Shaped Model [9] emphasizes over the validation and verification of the software. It emphasizes planning for verification and validation in early stages of development and the testing is conducted in parallel with the development. Again this model is good for tracking the progress. However, handling dynamic requirements and concurrent events is difficult. Systems that require high reliability usually suit this kind of development. Steps of the model are enlisted in Table 1.1

Table 1.1: Steps of V-Shaped Model [5]

Steps	Description
Project and Requirements Planning	Allocate resources
Product Requirements and specification Analysis	Complete specification of the software system
Architecture or high level design	Defines how software functions fulfil the design
Detailed Design	Develop algorithms for each architectural component
Coding	Transform algorithms into software
Unit Testing	Checks that each module acts as expected
Integration and Testing	Checks that modules interconnect correctly
System and Acceptance Testing	Checks the entire software system in its environment
Production operation and Maintenance	Provides enhancement and correction

1.1.3 Prototyping Development Model

Prototyping Development Model [10] belongs to the class of Evolutionary development models. These kinds of models allow the update of functionalities with each update of the product [5]. Other models of this class include the Spiral model, the WinWin model, the Parallel model, the Incremental build model and the Component Object Model (COM), discussed later. Developers build a prototype after gathering initial requirements and quick design. Feedback from the user is sought over the prototype and if the user is not satisfied the prototype is refined. This may be repeated several times. At the end, the code of the prototype code is elevated to the level of the final product to be released. Each prototype is released as a version of the software. Using this model, a more accurate end product can be developed assimilating the unexpected requirements. Moreover, all members of development team remain involved. However, Kumar et al. [11] calls it is a “Quick and Dirty” development as the system may be corrupted by continuous changes compromising the overall maintainability. The process is not well supported by documentation. Many times, the client may get the impression that the initial prototype is very close to the final product and may want it to be delivered instead. Such models may prove to be time consuming, when market forces faster release of products as the process seems unending. Prototype Development Model is applicable when the system is new and requirements are not well understood, or while developing user interface or while developing parts of large systems.

1.1.4 Incremental Development Model

Incremental Development Model is a combination of the evolutionary and the waterfall models. The phases of requirement specification, design, and implementation are broken into smaller increments. A partial system is constructed and then functionality is added one by one. In other words, every subsequent release of the system adds function to the previous release. The delivery of product takes place in increments instead of a single delivery. After developing an increment it is validated, integrated into the previous partially built system and delivered. Each delivered product is operational. After integration system is again validated. This goes on until the whole system is constructed. The higher priority functionalities of the user are included in the early increments facilitating their delivery early. In this manner, the functionality with the highest priority is tested the most. Customer can give their feedback for each increment. However, the model requires good planning, design and well-defined module interfaces. It is applicable when it is possible to deliver the system in small parts which is not easy always owing to the fact that it requires an early specification of the whole system. The methodology is inconvenient due to frequent meetings with the users. It is a good option when there is a need to release the prioritized functionality early in the market when the project has a lengthy development schedule or when project is based on a new technology.

1.1.5 Spiral Development Model

The Spiral Model introduced by B.W. Boehm [10] represents the development activities in a spiral fashion rather than as a sequence. Throughout the process, risks are considered, assessed and resolved. It can be seen as a combination of prototype and iterative methodologies. It allows development of critical functions first. Feedback from users can be sought. There are no fixed phases. Each loop in the spiral represents a phase in the process. One activity is performed in each of the four quadrants per loop. These include:

- Determination of objectives, constraints and alternatives
- Evaluation of alternatives, and identification and resolution of risks
- Development of next-level product
- Planning of the next phase

Although effective, it is a complex process and difficult to adhere to. Risk evaluation and resolution requires expertise and a lot of time. A lot of time is spent in resetting objectives, planning and prototyping. It has high human resource requirements and keeps much of that idle during non-developmental phases. This model is suitable when the requirements are unclear and expected to change, when creation of prototype is possible, for projects that represent new product line and when costs and risk evaluation is important.

1.1.6 Win-Win Spiral Model

WIN-WIN spiral model [12] is a more practical variant of Spiral Model. It includes negotiation activities at the beginning of each spiral where the developer simply asks the customer what is required. The customer may have to balance functionality and performance cost and time. It can be said that the customer wins by getting the product that satisfies his/her majority of requirements and the developer wins by achieving deadlines within the budget.

1.1.7 CBSE and COTS-based Application Process Decision Framework

Component-based Software Engineering (CBSE) is a reuse-oriented methodology that emphasizes on making intensive use of existing components or Commercial-off-the-shelf (COTS) systems instead of building components from the scratch. The COTS-based Application Process Decision Framework [13] enables the development teams to determine course of actions based on the appropriate combinations of Assessing, Tailoring, Glue Coding and Custom Coding process elements that best fit their project situation and dynamics. There are five principles for development:

- The team should start with flexible win conditions while assessing the alternative products
- Development teams should spend more time in assessing the alternative products and spend little time in tailoring, Glue Coding and Custom Coding
- Buy information early to reduce risk and rework
- Tailor a process to accommodate the process of COTS selection, integration and maintenance
- A good amount of time and effort should be spent in assessing the market and products since average upgrade time is ten months

COTS facilitate faster development and reliable product. However, trustworthiness of components is doubtful as source code is not available; compromise in requirements is quite likely. The model is applicable only when a pool of existing components is available. However, according to S. Koolmanojwong [14], while using COTS, issues concerning tailoring and interoperability may arise.

1.1.8 Rapid Application Development (RAD) Model

The RAD is an incremental software development process model that emphasizes on extremely short development cycle by using component-based construction. Each function is handled by a separate team and then integrated to form a whole. The focus is on code *i.e.* What You See is What You Get (WYSIWYG) rather than documentation. This approach encompasses modeling of data, process and business generating the testing, application and turnover [15]. RAD facilitates high productivity, low cost and risk mitigation. Involvement of customer throughout the process ensures their satisfaction. This model is a good choice when there is a pressing deadline [16]. If requirements are well understood and project scope is constrained, the RAD process enables a development team to create a “fully functional system” within a very short time [17]. The RAD model faces difficulty for large but scalable projects as they require enough human resources to create a number of teams. Both the developers and the customers ought to be committed to the rapid-fire activities [18]. It can be said that RAD needs well-known requirements, low risks and scope of modularization.

1.1.9 Rational Unified Process (RUP)

The Rational Unified Process (RUP) [19] is a hybrid process model that brings together elements from all of the generic process models, illustrates good practice in specification and design and supports prototyping and incremental delivery. It emphasizes on accurate documentation. The RUP is described from three perspectives and includes four phases. The perspectives include the dynamic perspective which shows the phases of the model over time, a static perspective which shows the process activities including the workflows and a practice perspective which suggests good practices to be used during the process. The phases in the RUP are firstly, Inception that identifies all external entities and their interactions with the system. Secondly, Elaboration, to understand the problem domain. Thirdly, Construction, involving system design, programming, testing and integrating system parts. Fourthly, Transition or deployment of the system at the client side. It resolves the project risks. The highlight of the model is the recognition that deploying software in a user’s environment is a part of the process. The development process is very complex and hard to understand requiring an expert software developer. Integration adds the confusion that causes more issues during the stages of testing. These problems are recognized by Booch et al. [20] and has expressed the requirement of tailoring in this method. The RUP is not a suitable process for all types of development e.g. embedded systems.

1.1.10 The Incremental Commitment Spiral Model (ICSM)

The ICSM [14] emphasizes early verification and validation, but allows for multiple incremental interpretations and alleviates sequential development. ICSM focuses on risk-driven activity prioritization, but offers an improvement by adding well-defined milestones. It provides adaptability to unexpected changes and allows scalability. In the Exploration phase, scope of the system is defined and alternatives are explored. In the Valuation phase operational concept is developed, requirements are prioritized, non-developmental products are assessed and business case is studied. In the Foundation phase, the development team focuses on building the system and creating a development plan. Finally, the project is delivered and deployed in the Operation phase. In ICSM, at every milestone risk is assessed and decision is taken to skip or repeat a phase.

1.2 Modeling and UML

Modeling is a fundamental activity within the requirements engineering process and concerns the construction of abstract descriptions of requirements. The choice of a modeling technique is critical whenever it is necessary to discuss the interpretation and validation of requirements, particularly the functional requirements and when stakeholders have divergent goals and different backgrounds and experience [21]. The model then lets you record, communicate and analyze the important aspects of the design [22].

Unified Modeling Language (UML) [20] is an industry-standard graphical modeling language for visualizing, specifying, documenting and constructing the artifacts of software systems. It is appropriate for modeling systems of varied types. It is not only quite expressive language but easy to understand and use as well. It is independent of implementation language. It supports diverse application areas. It simplifies the complex process of software design. It uses graphical notation to communicate more clearly than natural language (imprecise) and code (too detailed). The Object Management Group (OMG) released the UML in 1997 to provide the development community with a stable and common design language that could be used to develop and build computer applications. The primary authors were Booch, Jacobson and Rumbaugh. [23]

1.2.1 Diagrams in UML

Following are the major diagrams in UML:

- Use Case: A use case captures and describes the system's behavior under various conditions as it responds to a request from a stakeholder called an actor. Scope identifies the system and the preconditions tell what must be true before and after the use case runs. A success scenario is a case in which nothing goes wrong. The extensions describe what can happen differently during that scenario. The numbers in the extensions refer to the step numbers in the main success scenario at which each different situation gets detected [24]. The use cases together (also called the use case model) describe the whole functionality of the system.
- Sequence Diagram: Sequence diagrams show a detailed flow for a specific use case. The horizontal dimension shows the object instances to which the messages are sent while the vertical dimension shows the sequence of messages/calls in the time order that they occur.
- Class Diagram: The Class diagrams show how the different entities (people, things, and data) relate to each other.
- Activity Diagram: The Activity diagrams show the procedural flow of control between two or more class objects while processing an activity.
- State Chart Diagram: The State Chart diagrams model the different states that a class can be in and how that class transits from state to state.
- Deployment Diagram: The Deployment diagrams show how a system will be physically deployed in the hardware environment. Its purpose is to show where the different components of the system will physically run and how they will communicate with each other.
- Component Diagram: The Component diagrams show the dependencies that the software has on the other software components in the system.

1.2.2 Advantages of UML

It provides different views on a system and a high degree of abstraction possible. UML notation is widespread, well-known and acceptable [25] [26]. The UML offers an unprecedented opportunity for high-quality critical systems development that is feasible in an industrial context. A large number of developers are trained in UML.

Compared to previous notations, UML is relatively precisely defined [27]. B. Selic [26] believes that UML is suitable to model real time systems (and other domains/parameters) as well because many tools from different vendors support UML and there is an excellent conceptual match between the object paradigm and real- time systems. You can harness tool support to make your models much more useful than mere pictures.

1.2.3 Limitations of UML

Researchers have argued the abilities and usefulness of UML and its diagrams. According to E. Woods [22], the UML diagrams generally don't provide much information but only relation between things, which is not enough. Though UML is taught widely in academia but not used so often in industry because it has limited building blocks to describe the design. B. Selic [26] says UML needs modification to efficiently specify environments that support multiple paradigms and frameworks that support multiple views. According to Konrad et al. [28] UML lacks a precise and formally defined semantics. While it provides a nice variety of constructs but they may prove inconsistent and choosing an appropriate construct is difficult. Moreover, the UML semantics is both informal and problematic [25]. Dobing and Parsons [29] say that the UML needs customization as per the context of the project. M. Glinz [30] has found many deficiencies in UML. According to him Use cases offer limited support for eliciting security threats and requirements [31]. C. Kobryn [32] says that Use Cases are not well integrated with the rest of the language. Sindre and Opdahl [33] have stated that Use Cases are not suitable for requirements specification. These are really too vague and offer limited support for eliciting security threats and usually neglect extra-functional requirements, such as security.

UML is still used despite its several shortcomings because it is universally known and understandable in the designer community. Such a widespread acceptance is not enjoyed by any other modeling language.

1.3 Stakeholders and Assets

R.S. Pressman [5] describes a stakeholder as anyone in the organization that has a direct business interest in the system or product to be built and will be rewarded for a successful outcome or criticized if the effort fails. D'Souza and Wills [34] more lucidly elaborate that stakeholders of any system include the end user, administrator, developer, customer, maintainer and so on. The overall requirements and conflicting objectives are frequently much broader and vary among the different stakeholders *i.e.* roles of people who will be involved in the construction of the system. Table 1.2 describes these concerns and objectives of different stakeholders.

Table 1.2 : Concerns and Requirements of Various Stakeholders [4]

Stakeholder	**Concerns and Requirements**
Customer	within budget, on time and stable
End-user	intuitive and correct behavior, helps to do tasks, performance, reliability
System Administrator	intuitive behavior, tools to aid monitoring and administration
Marketer	competitive features, time to market, scores over existing products
Architect	familiar domain, infrastructure, architecture, buildable, meet requirements
Developer	clear requirements, simple design
Development Manager	predictability and tracking of project, schedule, productive use of resources including existing or familiar code, cost
Maintainer	documented, understandable and consistent design approach, easy to modify

Assets are parts of the system that are valuable for the organization, e.g. information, software, or hardware. They need to be protected from malicious activities in order to achieve business goals. Within this research work data items have been considered as assets. [35]

1.4 Principles of Secure Software Development

Saltzer and Schroeder [36] have put forward some principles to help guide secure software development. These include designs that are simple and not secret, access decisions not based on exclusion but permission, checking every access right to every object, granting least possible privileges to every user or program that is enough to complete the task, providing a robust access mechanism with two keys, and an easy-to-use human interface. Goertzel et al. [37] manifests three security principles for software. Firstly, it should be secure by design. Secondly, it should be secure by default *i.e.* the default or supplier's configuration should be as restrictive as possible. Thirdly, it should be secure in deployment *i.e.* even after the software has gone for production its security can be maintained.

1.5 Sources causing insecurity in software

Yoder and Barcalow [38] say that usually software is developed without keeping in mind the security aspects because the programmer is busy in learning the domain, building prototypes or eliciting the customer's requirements. Some major sources of insecurity in software, as pointed out by Goertzel et al. [37], are as follows:

- Inadequate Development Principles and Practices
- Incorrect or misunderstood developer assumptions
- Insufficient capture of requirements for security properties
- Design mistakes
- Inadequate documentation
- Insufficient tools and resources for developers
- Project management decisions that undermine secure software production

1.6 Problem and Motivation

As mentioned in the Introduction (Section 1) and the discussion presented in Goel et al. [31], the requirements and design phases are the most appropriate phases for integrating security in the development life cycle. A proper elicitation and specification of security requirements is obviously necessary but their precise and clear elicitation is difficult [7]. Similarly, modeling the requirements or their diagrammatic representation is essential for unambiguous communication between the users and the development team [39]. Further, to model the security requirements effectively, the existing popular language known as Unified Modeling Language (UML) [40] is not suitable. The above issues have served as a motivation to develop Security-aware Requirements Elicitation, Assessment and Design (SecREAD) Methodology which is a novel and improved methodology for eliciting specifying and modeling security requirements.

1.7 Objectives of the Research

The research work has been carried out to achieve the following major objectives:

- Address the need of integrating security into the development lifecycle
- Critically analyze the existing security-oriented development models to find appropriate development phase(s) to imbibe security and deduce effective practices propounded by various researchers in this regard
- Develop an efficient requirements engineering process to elicit, analyze, specify and model security requirements that reckons the concerns and aspirations of all stakeholders of the software
- Introduce an effective security ranking process followed by empirical analysis
- Model the above requirements based through easy but effective diagrams based on the rankings and the empirical analysis.
- All in all, to propose a new improved secure software development methodology that overcomes the limitations of the prior works but includes their positive aspects and other best practices advocated by different researchers.

1.8 Organization of the Thesis

The thesis consists of six chapters where Chapter 1 throws light on general concepts of software development along with a description of some classical and established process models, and Unified Modeling Language (UML). Chapter 2 incorporates an expansive literature survey that highlights the concerns raised by various researchers over imbibing security in SDLC, justifying its need and relevance in the present scenario. Furthermore, it critically analyses different methods and methodologies proposed so far to cater to these concerns. Chapter 3 illustrates the SecREAD methodology in detail. The methodology has been validated through its application on two case studies. In Chapter 4 the case study of Internet Banking is taken up while in Chapter 5, the methodology is applied on the case study of Smart Buildings. Finally, the conclusions are drawn in Chapter 6. The thesis also contains two appendices A and B that supplement the case studies presented in Chapter 4 and Chapter 5 respectively.

Chapter 2

Literature Survey

"…so essential is security that no software should ever be released without these requirements being met"

- Microsoft [41]

Both academia and industry have raised concerns on the security in software and its inclusion in SDLC. According to Swiderski and Snyder [42] "Security is an aspect most customers expect and all customers want." Failures of security mechanisms may cause very high damage. However, security is mainly taken as an add-on to the common system development [43].

In the previous chapter, certain principles of secure software development have been discussed along with some major sources that bring insecurity in software. This chapter presents a discourse on the need for imbibing security in SDLC, certain challenges or issues faced by research community, finding the most suitable phases of the SDLC to imbibe security and certain relevant prior works in this direction.

2.1 Why Security in SDLC?

Security has lagged behind maintainability in seeking attention of developers. Software systems are based on insecure technologies and are marked by faults and vulnerabilities which should be avoided. There are many reasons for requirement of security in software. Firstly, high-consequence software systems which involve sensitive information are increasing and being exposed to the internet. Secondly, security lags far behind correctness and maintainability as far as its application throughout the SDLC is concerned. Thirdly, security attacks often lead to huge losses. Fourthly, very few software engineers are aware of security concepts. Fifthly, there is lack of software engineering tools and techniques for security. [37] [44]

Meier et al. [45] has explicitly stated that for secure design and deployment of software, security ought to be integrated into the development life cycle. R. Anderson

[46] says that integration of security into the system development is necessary to build secure systems. Similar view is taken by On-Point Organization [4] who believes that incorporating security into the SDLC is one of the most effective ways to protect the assets of the organization. Furthermore, adherence to an SDLC model increases the likelihood of project success by meeting the requirements of stakeholders in a better way. N. Davis [47] goes on to say that a very few of the many available processes and methodologies specifically support secure software development from the ground up. Software security is an inevitable issue in an increasing networked world.

Goertzel et al. [37] attributes the most critical difference between secure software and insecure software to the nature of the development process used. Lindvall et al. [48] says that the selection of development model is significant in decreasing risk levels. Similarly, D. Shreyas [44] stresses for an urgent integration of security policies in the development process. However, J. J¨urjens [27] says that security mechanisms cannot be “blindly” inserted into a security-critical system, but the overall system development must take security aspects into account.

Any support to aid secure systems development is thus dearly needed. Such an inclusion will reduce cost and effort. It can be understood by some of the studies conducted. Detecting and repairing an error during the testing phase costs 10 to 100 times more than the cost of fixing it in the earlier phases. Further, this cost may grow up to 40 to 1,000 times if it is found after the releasing the software [49]. The return on investment for secure software engineering can be as high as 21 percent [37].

2.2 Security in SDLC: Suitable Phase(s)

Many researchers [37][50][51][52][53][54][55] advocate that considering security from the very start in the SDLC will be more beneficial. In this regard, [33] stresses upon inclusion of security in requirements phase and considers that its postponement to design and implementation phases can cause security issues which may be forgotten or ignored. Security is seen as a qualitative or non-functional requirement by [56][57] and [58] that must be engineered into the product rather than being added on at the last minute.

However, various other researchers [38][59][60][61] recommend the inclusion of security in the design phase. In general it seems that the customer/user community finds diagrammatic expression more acceptable than words and symbols [62].

Some researchers [16][27][44][63][64] believe that security should be considered throughout the life cycle. The intent is not to disturb or add more phases to the SDLC, rather incorporate security activities into an existing SDLC methodology [4]. The next sub-sections discuss the significance and characteristics of requirements and design phases respectively, as also how challenging it is to imbibe security in them.

2.2.1 Requirements Engineering

D. Shreyas [44] says that RE is an area of primary concern in software engineering. Sharif et al. [65] considers it to be the most important phase in the development life cycle. According to Fellir et al. [66] and Kumari and Pillai [67], the success of the software depends on requirements engineering. Researchers [68][69][70] attribute the software quality on the specification of requirements.

Being so important, Requirements Engineering however poses several challenges. Breu et al. [16] and Futcher and Solms [63] consider requirement elicitation to be the most critical and ambiguous process. A small mistake at this stage can make the system unacceptable by the customer and may require a lot of rework, time and cost. Babur et al. [71] says that lack of clarity in objectives makes the development of software cumbersome. The process can be facilitated by taking into consideration the requirements of the user and other stakeholders. This will result in greater customer satisfaction and will enhance business value. In his research, R. Snijders [72] has concluded that user involvement has a large potential for improving the quality of RE and thereby the quality of software. Sabahat et al. [73] explicitly states that the best approach is to get the correct requirements initially but Wäyrynen et al. [50] argues that customers are usually not in a position to freeze all requirements at the very beginning. E.R. Keith [61] says that when the environment is not stable and requirements keep changing, it usually becomes difficult for the customers to realize what they really want until a system is functioning. Haley et al. [74] is of the view that there is no satisfactory integration of security requirements into requirements engineering. Chua et al. [75] has underlined the importance of complete and correct requirements in the development of a correct system in accordance with users' wants and needs. However, the process of eliciting business user requirements is quite time-consuming for both business analysts and users. Methods such as prototyping and use cases, according to them, are insufficient for outlining requirements completely and correctly. Problems with requirement elicitation highlighted by them are incorporated in Table 2.1. Some elicitation methods have been proposed keeping security in mind which will be discussed in Section 2.6.

Table 2.1: Problems with Requirements and Their Effect

Problems with Requirements	**Effect of the Problems**
Incomplete domain knowledge	Poor user collaboration
Incomplete requirements	Incomplete understanding of needs
Inconsistent requirements	Non-solid intentions of requesters
Ambiguous requirements	Synonymous and homonymous terms
Excessive requirements	Unorganized bulky information sources
Fluctuating requirements	Continuous acceptance of additional
Overlooking tacit assumptions	Incorrect requirements
Ill-defined system boundaries	Misunderstanding of system purpose
Un-testable terms	Unnecessary design considerations
Different views of different users	Unfixed requirements
Too many requesters	Over-commitment by sales staff

2.2.2 Security in Design

The practice of simple design will make the software easy to be evaluated from a security perspective. Complex systems with many interactions are difficult to analyze and understand which will have an impact on other security areas. According to B.H. Wu [76] systematic presentation of design fragments and techniques can produce effective results. High quality software needs great designers which are rare. This is also seconded by Redwine and Davis [77]. According to N. Coblentz [78], design flaws amount to 50% of security problems. UML is the standard designing language that has been discussed in Section 1.2.

2.2.3 Human Factors and Team Composition for security

Sawyer and Guinan [79] believe that social process affects Software production more than the technological process. Similarly, according to Birk et al. [80] the activity of software development is a human and knowledge intensive activity. Basri and O’Connor [81] emphasize that software development depends on quality of communication within the different team members and among different teams. Further, they consider people as the greatest asset in any software organization who are critical to the success of software development.

Certain improvements in team composition are required for security enhancement in development of the software. According to D. Shackleford [53], the two teams of development and security ought to work together. This has several benefits like efficient development operations, better code and quality processes. However, there is a basic problem. Developers and security teams have different priorities. The security team is more concerned of integrity and confidentiality of data, which can slow down the development team. But, developers are governed by the business requirements to produce and revise code as quickly as possible. In this way developers focus on what works best instead of what is most secure. To remove bugs the developer has to write more code and since code is directly proportional to bugs, this again results in bugs. The management personnel are governed by market forces because of which they are more concerned in quick delivery of code and consider security team responsible for slowing down the development process. In this manner not only programmers but security personnel and upper management as well share the brunt of security problems. So, a right balance between the conflicting roles of security experts, developers and management is desired. This is summed up aptly by Viega and McGraw [82] that the team is the basis of every software project and that achieving a tradeoff between speedy development and satisfying security requirements is a challenge. Other modifications suggested are the inclusion of testers in the team by J. Rasmussen [83]. Redwine and Davis [77] have suggested the inclusion of personnel with substantial education, training, and experience to meet the demands of modern development process requiring various specialties. Researchers [68] and [84] have advocated the inclusion of customers and end-users to understand their needs and context consequently, increasing the probability of their satisfaction. In the light of this, I. Sommerville [55] has underlined the issue that stakeholders and end-users are reluctant to adopt new notations imposed on them but want to use their own notations and terminology to describe their requirements. Though they have their own ideas, they are not always competent in representing them.

2.3 Security Parameters

Various researchers judge the security of the system on some parameters. The parameters are more or less the same with very minute difference in perception. Talukder and Prahalad [51] take confidentiality, integrity, authentication, authorization to be important parameters and performs ranking of assets over them by giving values 1, 2 and 3. Breu et al. [16] also considers the same parameters. Authenticity of each actor in an activity is a critical requirement. Authentication should be handled using standard protocols and components, if available. Authentication involves verifying that people are who they claim to be, by using username-password scheme, biometric authentication based or voice recognition, fingerprint scans or retinal scans. Non-repudiation means that for each important activity the actor cannot deny that he/she executed the activity. Integrity is about preventing unauthorized alteration. Authorization is about determining what resources an authenticated person has access to. Redwine and Davis [77] take session management separately as a security measure and availability as parameter for preventing unauthorized destruction or denial of access or service. D. Shreyas [44] thinks security of a software system depends on Authentication, Confidentiality, Integrity and Non-repudiation and treats Access Control as separate from authorization. Goertzel et al. [37] considers only authentication, access control and authorization as parameters. Federal Information Security Management Act (FISMA) [85] defines three security objectives for information and information systems namely, Confidentiality, Integrity and Availability. Their impact is rated as low, moderate and high. D’Souza and Wills [34] lay stress on better team management and flexibility to cater to different domains along with integrity. Department of Homeland Security [85] has elaborated security parameters. One of them is availability of software to its authorized users whenever needed. Further they take Integrity, Confidentiality, Non-repudiation and Accountability as other parameters. They specify elaborately integrity to keep safe from unauthorized modifications like corruption, overwriting, tampering, insertion of unintended logic, deletion or destruction. Accountability is somewhat like authorization where responsibilities of stakeholders are set for their actions. B.A. Forouzan [86] considers Authentication, Integrity, Non-repudiation and Confidentiality as important elements of security.

When security is referred to, it may imply to one or more of the above dimensions. For example in e-mail communication, security might involve integrity, non-repudiation and confidentiality while in online shopping it may involve integrity, non-repudiation, confidentiality as well as authentication. Ranking is performed within the context of software domain and interest of client organization. Pohl and Rupp [87] have found ranking process suitable but according to them it should involve the stakeholders of the software. S. Hatton [88] has advised simplicity in ranking process for the stakeholders. Goertzel et al. [37] suggests assessment of security throughout the development process by experts, internal or external. This idea of experts is seconded by Saripalli and Walters [89]. Ranking in some way or the other is conducted in diffrrent methodologies for secure software development like SecSDM [63], Threat Modeling [77] and SaSDLC [90]. All of these are described in Section 2.5.

2.4 Empiricalness in Software Engineering

In software engineering research, unlike other fields of study, there is lack of empirical analysis [91]. Redwine and Davis [77] have raised concerns about the effectiveness of existing empirical practices for producing secure software. Further, they have summarized the strengths and weaknesses of empirical study. Their structural approach contains certain steps *i.e.* designing better studies, collecting data in a more effective manner, and involving other stakeholders. Fenton and Pfleeger [92] have pointed out that several empirical studies are not fit for large systems and do not have proper statistical designs. As far as qualitative analysis is concerned it uses data that is less readily quantified through the techniques like interviews, observations etc. which help in understanding the people's perspectives. At the end, the researchers must carefully analyze how their biases affect the data.

2.5 Related Works: Existing Secure Lifecycle Models & Methodologies

To address the issues elucidated in the previous sections, various researchers have proposed some models and methodologies. These are described and critically analyzed in this section.

2.5.1 Agile Software Development Methodology

These processes are named so because of their adaptability unlike traditional processes. Agile Processes focus on early development of code and people interactions rather than documentation and planning [61]. The manifesto of the "Agile Alliance" [93] has outlined certain principles for this kind of development. These include, giving the highest priority to the satisfaction of customers and developers working together, which is in line with the views of various researchers already highlighted. It is suitable for frequent alterations, catering to the changing requirements and minimizing risk by developing software in short time boxes or iterations. Adhering to Agile methodology, critical issues are stated early and addressed as also continual feedback is provided to the development team. The steps for development are project initialization, setting the project time-box, determining the right number of cycles and the time-box for each, setting an objective for each cycle, assigning basic components to each cycle, developing a list of tasks, reviewing the success of a cycle and planning of next cycle. To enhance security, the development team includes security experts in this methodology.

However, Agile processes link requirements to the code but lack documentation. Therefore, if the customer does not possess enough clarity, the development process can go off the track [61]. Over-dependence on tacit knowledge makes the transfer of software to other organizations difficult [94]. Lindvall et al. [48] and Turk et al. [93] do not consider it to be a good choice for the development involving subcontracting, distributed environments, large teams, reusable artefacts, and development of large, complex, safety-critical or reliable software. The practice of pushing code quickly and object re-use can compromise security.

Agile is applicable to the software that can be built quickly, especially that are maintenance-intensive, time-critical applications and in the organizations which have disciplined methods [48]. An important Agile method is Extreme Programming (XP), detailed in the next section.

2.5.2 Extreme Programming (XP)

XP development methodology [6] is based on developing and delivering functionality in very small increments. The practices of simple designing, testing, reviewing and short iterations are taken to extreme levels. It is simple and implements what is actually needed according to the situation. It emphasizes rapid feedback from the customer through frequent deliveries. It expedites development, increases customer satisfaction, lowers the chances of defects rates and handles frequently changing requirements. It is applicable to small-to-medium-sized software with rapidly changing or vague requirements.

XP involves the following practices:

- ‘Pair-programming’ means that code is written by two programmers on one machine
- Metaphor or a story about the working of the system
- Simple design
- Testing by programmers and customers
- Refactoring or continuously restructuring the system
- Collective Ownership of the code *i.e.* anyone can change any code anywhere in the system at any time
- Continuous integration of the system every time a task is completed
- Small Releases of new versions in a very short time
- Planning Game or determining the scope of the next release
- Working 40 hours a week
- Following the coding standards – programmers write all code in accordance with rules emphasizing communication through the code
- Customer On-site, always to answer the questions

XP has certain limitations as well. Frequent meetings with customers amount to enormous expenses. An exact estimation of work cannot be made initially since scope and requirements of the project are not known. Usually the cost of changing the requirements at a later stage in the project can be very high. K. Beznosov [8] doubts the successful application of XP to security engineering projects. Lack of documentation renders it unsuitable for secure development. Viega and McGraw [82] go as far as saying that XP has a negative impact on software security. XP lacks a comprehensive consultation team to aid and advise the developers thus, specialized inputs may not be obtained.

Wäyrynen et al. [50] has proposed some modifications to XP to make it suitable for developing projects securely. Firstly, a security engineer should be included in the team. Secondly, the security architecture must be documented before the security review. Thirdly, complement pair programming with verification.

2.5.3 Comprehensive, Lightweight Application Security Process (CLASP)

CLASP introduces security in the software development in the early stages. It includes the usage of instructions, guidance, and checklists. Thirty specific activities are expressed in CLASP to make the development team security-aware. The activities are assigned to eight roles viz. requirements specifiers, project managers, security auditors, software architects, implementers, designers, testers, test analysts, and integrators and assemblers. For each activity its implementation, associated risk on its omission and estimation of risk is defined. CLASP claims that it can be adapted for any development process. [37]

2.5.4 Security Development Lifecycle (SDL)

Microsoft [53] introduced SDL to create a more secure software. Security is considered throughout the cycle. It is claimed that while using the SDL, the vulnerabilities can be reduced by 50%. The model is depicted in Figure 2.1. The model has some unique features. Developers are imparted security training at the very inception and there are separate phases for verification (pre-release) and response (post-release).

This model is applicable to software that involve large development teams, long development cycles and extensive resources [78] like such as Windows and Microsoft Office [41]. Figure 2.1 given by R. Labbe [95] shows how SDL aligns with the traditional DLC.

SDLC	Envision	Design	Develop/ Purchase	Test	Release/ Sustainment
SDL	Application Entry/ Risk Asessment	Threat Model/ Design	Internal Review	Pre-production Assessment	Post-production Assessment

Figure 2.1: Alignment of SDL with tradition SDLC

2.5.5 Secure Software Development Methodology (SecSDM)

SecSDM [63] is a risk-based methodology that advocates integration of security throughout the SDLC. It performs identification of assets and their prioritization based on the threats based on a ranking mechanism assessing the likelihood, frequency and impact of threats. Security concerns implemented are associated with possible risks that are identified and traceable. It emphasizes on security training. SecSDM is claimed to be developed keeping in mind that it adds no additional time, cost and skill overheads. It has 10 steps grouped under six phases. Its first phase is the investigation phase in which the possible risks or threats are investigated. The later phases are analysis, design, implementation and maintenance. Risks are identified and correspondingly security services are provided which include identification, integrity, confidentiality, authorization, authentication, access control and non-repudiation, but there is no provision of eliciting stakeholders' views. However, assets are identified and ranked. Data items are considered as assets. Their study shows that security concerns attributed to the two phases of requirements and design are more than that for other phases.

2.5.6 Security-aware Software Development Life Cycle (SaSDLC)

Talukder and Prahlad [51] believe that in future security ought to be considered as a significant functional requirement during development owing to inter-linking of applications accessible by varied users. Applications, running on variety of devices and platforms, need to be security-aware. In SaSDLC first the assets are identified. In later steps functional requirements are captured and analyzed followed by identification of security requirements. Later each risk is rated on a scale of 0 to 10 using DREAD [23] which uses the formula depicted in Eq. (2.1). Each threat is ranked at three levels. These rankings are compared with value of assets as measured in the first step. Here, assets mean data items.

$$DREAD = (D + R + E + A + D) / 5 \qquad (2.1)$$

where, D = Damage Potential, R = Reproducibility, E = Exploitability,

A = Affected Users and D = Discoverability

The method, however, does not produce a comprehensive list of stakeholders for each different project but brings all stakeholders under one single umbrella of ‘user’ along with an ‘administrator’. It also considers an ‘attacker’ and solicits ranking from it as well. Furthermore, the ranking is not governed by any set of rules and is done regardless of domain. [90]

2.6 Security Techniques/Models for Requirements Engineering

This section describes some of the existing techniques and models for secure requirements engineering.

2.6.1 Iterative Requirement Elicitation (IRE) for Global Software Development

Sabahat et al. [73] has proposed the IRE approach for effective requirement elicitation of Global Software Development (GSD) projects *i.e.* software development at geographically separate locations. For such type of development, efficient coordination and synchronous interaction between distributed groups is desired which makes the task more complex. Due to lack of personal meetings with the client, developers may assume the requirements. There are cultural, language, legal and social barriers in requirement elicitation of such systems along with the issues of distance and time-difference. In the existing approaches of requirement elicitation the requirements engineers may assume requirements in later phases when they encounter any ambiguity leading to poor customer satisfaction. To avoid this, requirements are elicited from customers in an iterative manner. The techniques of interviews, prototyping, questionnaires and scenarios are used to elicit requirements. The results show that IRE approach is quite effective in satisfying the customer requirements. But, only the customers are involved in the elicitation process, leaving out other stakeholders.

2.6.2 Risk-based security Requirements Elicitation and Prioritization (RiskREP)

RiskREP [35] is an extension of misuse case-based methods. Risks are assessed countermeasure are defined and prioritized according to business goals, cost and their effectiveness. The model contains technical, user and business perspectives. User perspective specifies quality attributes of the system to be protected e.g. confidentiality. Business perspective specifies business goals, expressed as quality requirements like “confidentiality of password”. Assets include information, software and hardware which are to be protected to achieve business goals. Steps of the method are Quality goal analysis, Risk analysis, Countermeasure definition and Countermeasure prioritization. The information is elicited from IT manager, security officer and business owner who represent the IT, user and business perspectives respectively. A risk expert and an RE expert elicit the information from other stakeholders.

2.6.3 Common Criteria

The Common Criteria (CC) [96] describes security related functionality to be included into the development process, like assurance, secrecy and authentication. After successful evaluation of the CC, a certificate is issued depicting its trustworthiness. The CC improves the quality of the system and reduces the cost and effort of instilling security. The methodology is iterative. An iteration consists of planning, analysis, design, implementation, operation and delivery phases. After every iteration, a fully functional product is obtained that satisfies a set of requirements and is ready for evaluation. During the entire process, security measures are carried out as early as possible ensuring that discovery and mitigation of security problems early resulting in cost reduction. However, the requirements have highly complex dependencies and no guidance is provided to deal with them. All in all, CC is highly sophisticated for a security critical system with seven Evaluation Assurance Levels (EALs) [50].

2.6.4 The Security Analysis Process

Breu et al. [16] has proposed Security Analysis Process to explore requirements and measures at the proper level of detail. It consists of steps namely Security Requirements Elicitation, Threat Modeling, Risk Analysis, Measures Design and Correctness Check. Apart from these an elaborate access policy is formed during the requirements specification with the help of customers and/or end-users. Systematic checks are conducted with respect to Authentication, Confidentiality, Integrity, Availability and Non-Repudiation.

2.6.5 PREview

I. Somerville et al. [55] has introduced a multi-perspective requirements engineering approach (PREview) for industrial use. PREview allows incremental requirements elicitation in spiral fashion. It is a flexible model of viewpoints with no particular notation to organize requirements acquired from different sources. Viewpoints are used in the early stages as a structuring mechanism for requirements elicitation and analysis. Identifying viewpoints and organizing information around them reduces the possibility of missing any critical information and it provides a mechanism for linking requirements with their sources. A viewpoint has a meaningful name, a focus *i.e.* a definition of its perspective, the requirement sources, the history of the changes over time and a list of the applicable concerns that are used to elicit requirements and formed into questions to be answered by the stakeholders. In summary, PREview helps improve the quality of requirements specification. However, there are certain drawbacks like managing the customer information due to gathering of requirements from a number of viewpoints. It is not clear when the process of elicitation should cease.

2.6.6 CORAS

CORAS [97] [98] is a model-based method for analyzing security risks. It provides a customized language to model risks and threats. For security all aspects of integrity, confidentiality, non-repudiation, availability, reliability, authenticity and accountability of IT systems are considered. The activities involved are identification of assets, identification of important security requirements that are discussed with the experts, identification of the risks to assets and analyze them. Lastly, evaluate risks level as low, moderate and high over the parameters namely, rare, unlikely, possible, likely and almost certain. The steps of CORAS are a meeting with the client followed by another meeting with the client where the analysts will present their understanding of the system by studying the documents. In the third step a more refined description of the system is made. This step is terminated once all this documentation is approved by the client. The client then defines the criteria of evaluating risks. In step 4 a workshop is organized with the experts aimed at identifying unwanted incidents. A workshop is again organized in the fifth step focusing on evaluating risks that is presented in the treatment workshop in the last step.

2.6.7 Controlled Requirement (CORE)

CORE Specification Method [39] is also a viewpoint based approach. It considers views of all stakeholders, supported by diagrammatic notation that can be applied to the description of both requirement and design. The diagrams can be separated into dynamic diagrams and static diagrams. Dynamic diagrams represent flow of time and connections between temporally ordered items while Static diagrams present the hierarchic structure connecting dynamic diagrams. However, CORE does not lay emphasis on non-functional requirements, rather it focuses on information flow. Analysts have a passive role which makes proper elicitation of requirements difficult.

2.6.8 Use Processes

Use Processes [68] is a methodology for requirements elicitation. A Use Process Diagram is used that is based on Use Case and the OMG Business Process Modeling Notation. According to them, Use Cases (UC) [6] do not present a business process (BP) oriented approach for eliciting requirements. The main customers of the software are business people who generally prefer flow charts to visualize business processes. So Use Processes allow participation of users and customers in the requirement elicitation process and results in their better satisfaction. The steps of the methodology are: Defining a problem statement, Modeling the Business Process, Defining the System Boundary, Describing the Activities and Involved Roles, and Identifying and Describing the System Functionalities.

2.6.9 Quality Function Deployment (QFD)

QFD [99] is a technique for quality management that converts the customer needs into technical requirements for software. QFD spans the entire development process. It aims at maximizing customer satisfaction. For this, QFD lays stress on what is valuable to the client and then deploys these values throughout the software development process. QFD classifies requirements into three kinds *i.e.* Expected, Normal and Exciting. For elicitation of requirements QFD uses the techniques of observation, interviews, surveys and examination of historical data. To extract expected requirements, matrices and diagrams are used. Its steps are to identify customers and requirements, identify technical features and then relate the previous two steps to develop different kinds of architectures. The best out of these is selected.

The major disadvantage is the heavy initial investment. The usage of QFD may slow-down the development process. If direction changes mid-way a lot of re-work will be done. Moreover, management commitment is essential. [62]

2.6.10 Miscellaneous Methodologies

Core Set of Factors has been proposed by J. Steven [24] which can be addressed independently of the development methodology. These factors are in the form of questions, answers of which can aid in secure development. Broadly the questions are related to the identification of stakeholders and their concerns, how is design verified to cater to security requirements, how adherence of code to design is proven, how are threats weighed against risks and how attacks are handled.

Feature Oriented Domain Analysis (FODA) [100] is a process where relevant information is elicited about the system's domain iteratively. To achieve this, analysts, domain experts and users work together to suggest the required features and according to them the designers develop the architecture. The method is iterative in nature. Users are interviewed to gather requirements. However, this method is suitable for well-understood domains only that don't change rapidly. Issue-based Information System (IBIS) [101] is a formal method of requirement elicitation with underlying features of Question, Idea and Argument. However, it lacks in graphics and iteration. Joint Application Design (JAD) [62] is another elicitation method which involves users throughout the SDLC but it works well with committed top management only.

2.7 Secure Design Techniques/Models

This section describes some of the existing techniques or models for secure software design.

2.7.1 Requirements Visualization of UML (REVU)

Using the REVU [28] process, the functional requirements can be visualized through a UML model. In the first step the developer specifies the properties of a witness scenario in natural language. It is a sequence of steps to fulfill a functional requirement. This declarative specification facilitates a developer to discover scenarios. In the second step, a model checker generates witness scenarios that adhere to the previously specified properties. Finally, each witness scenario is viewed by the developer as per the original UML model.

2.7.2 Security Design Patterns

In this methodology various security design patterns [38] may be applied to develop a software. These are listed as follows:

- ‘Check Point’ to authorize and authenticate users
- ‘Single Access Point’ to validate users and collect information about them
- ‘Roles’ to club users that possess same privileges
- ‘Limited View’ that allows users to view only what they have access to
- ‘Full View with Errors’ that gives users a complete view with exceptions
- ‘Secure Access Layer’ that allows secure communication with external systems
- ‘Session’ in an environment with many users to maintain information

2.7.3 Focus Group Discussion for Requirements Elicitation (FGDRE)

Kasirun and Salim [54] have proposed FGDRE for eliciting requirements iteratively, which is suitable when there are many stakeholders who keep changing their requirements. The requirements are represented in both graphical and text forms for better understanding. A consultation is held among all stakeholders. The session is guided by a facilitator who ensures that the discussion is within the context. Feedbacks and responses from the discussion are taken into consideration in the next activity. The session starts with identification, followed by elaboration and refinement. In the final step of integration all the agreed concerns and viewpoints are integrated.

2.7.4 Misuse Cases

One of the techniques for introducing security in requirements elicitation is proposed by Sindre and Opdahl [102]. It is a slight modification in the Use Cases to overcome their limitations, already mentioned in Section 1.2. Misuse case actually means a negative use case that specifies behavior not required in the proposed system. These can help in eliciting security requirements properly. Corresponding to every actor in a use case there is a misuse and a mis-user. The Misuse Cases are denoted by black ovals against every use case, denoted by white ovals. Similarly, a crook is denoted as an actor but with a black head. For example ‘Register Customer’ is a use case involving customer as a user. Correspondingly, ‘Flood System’ may be a misuse case initiated by a mis-user namely crook.

For textual description different templates are given by A. Cockburn [24], and Kulak and Guiney [103]. They contain fields like Use Case Name, Name of Author, Basic course of events, Preconditions, Post-conditions, Iteration, Exception paths, Alternative paths, Extension points, Triggers, Assumptions, Business rules, Summary and Date of writing. Sindre and Opdahl [102] has proposed some modifications to cater to security requirements more effectively like identifying important assets, define security goals for every asset, identifying threats to each security goal, identifying risks for the threats and specifying security requirements to mitigate risks. Misuse cases are beneficial due to early focus on security. They can be understood by the non-technical stakeholders. They help in linking the functional and non-functional requirements [102]. But, the misuse may not be identifiable and it may not be a result of an identifiable sequence of actions [33].

2.7.5 Threat Modeling

As per N. Sportsman [59] threat modeling is a security control activity performed during the design phase. Here, a meeting is organized among the security and development teams. Threat modeling [77] is used to identify risks and consequently support the decisions of design, coding, and testing. It identifies the key assets of an application, identifies and categorizes the threats to each asset, ranks the threats and then develops strategies for mitigating threats that are then implemented. Microsoft [104] has developed a robust technique of threat modeling that is implemented during design to identify potential vulnerabilities. Since it is applied in the early phases, it reduces the cost by identifying mistakes early on. It has several benefits for improving security by finding vulnerabilities, threat analysis and reducing or minimizing the impact of risk. Apart from this it also aids in testing and reducing cost. Threat Modeling Process (shown in Figure 2.2) is iterative and comprises of five steps namely Identify Security Objectives, Application Overview, Decompose Application, Identify Threats (and countermeasures) and Identify Vulnerabilities.

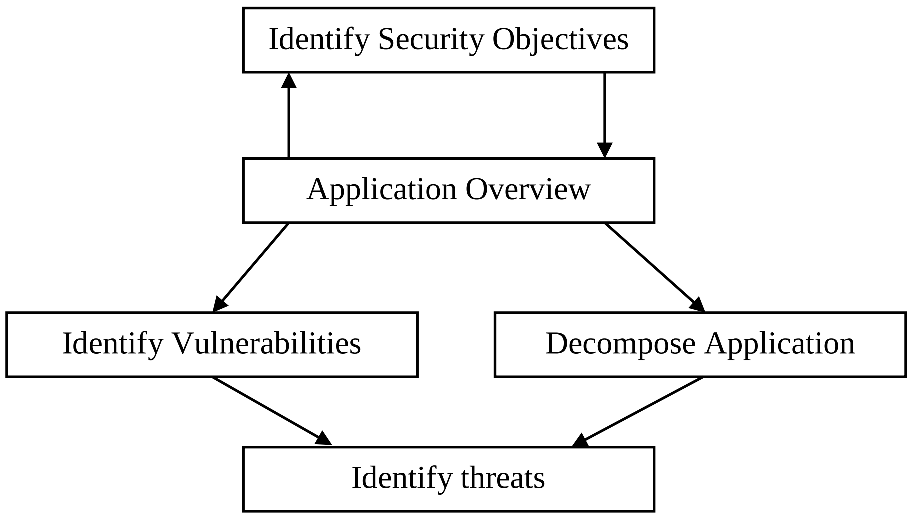


Figure 2.2: Threat Modeling Process

Threat Modeling is asset-centric and iterative that evolves over time, adapting to changing business requirements and new threats. Scenarios are used to identify threats. Use cases stemming from security requirements and abuse cases are understood. If cases are not available, new are created and analyzed with developers. S.F. Burns [60] believes that threat modeling helps in managing, acknowledging and communicating security risks throughout the application. Creating a threat model consists of three steps. The first step is to view the system as an adversary and identify assets and entry/exit points. Secondly, characterizing the system based on the background information. Thirdly, a threat profile is created that includes identifying the threats, investigating and analyzing them, and mitigating the vulnerabilities caused by the threats. However, it is costly rendering it suitable for only the most critical applications. It usually needs outside security expertise [59]. It cannot predict novel threats and attack patterns causing failure of software to operate correctly throughout its lifetime [37].

2.7.6 STRIDE

STRIDE [105] model is used to classify the identified threats which makes them easier to understand and helps in determining their priority. STRIDE is an acronym for five categories in which the threats are classified namely Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilige.

2.7.7 UMLsec

Jan Jürjens developed UMLsec [75], an extension of UML. It is motivated from the understanding that high quality development of security critical systems is not easy. Many flaws are found in design and implementation of such systems. UMLsec endorses the fact that security mechanisms cannot be inserted “blindly” into security-critical software, but security should be imbibed in the overall development process. Notions of tags, stereotypes and constraints have been used to extend UML [27]. UMLsec facilitates static security modeling. Information related to security is added to class diagrams [43]. However, UMLsec has no elicitation and assessment procedure. It considers less parameters and deals with physical aspects also [68].

2.7.8 Miscellaneous Methodologies

SysML or Systems Modeling Language [106] has been developed to extend and customize UML in order to support the specification, analysis, design, validation and verification of systems that include both software and hardware. Analysis and Description of Requirement and Architecture (ADORA) [30] language is based on the hierarchy of abstract objects which integrate the behavior, structure, functionality and user interaction. ADORA removes some of the problems pertaining to UML structure like decomposition and aspect interaction. Formalization of UML by Chanda et al. [107] considers that UML is an informal model which may lead to ambiguity in designs pertaining to different aspects of the same system. They have proposed a formal model for Use Case, activity & class diagrams, the three widely used models which represent static and behavioral aspects. A context free grammar is proposed for the UML 2.0 standard. A set of verification criteria composed of correctness rules, consistency rules and traceability rules are defined and verified. The Context View [21] describes the relationships, interactions and dependencies between the system and the environment it interacts with. UML has its limitations in representing such element types. For this, the Context View is proposed. It is equipped to render representations to the connected external entities and a set of relationship types indicating a connection's characteristics. Although it represents more detail but not the non-functional requirements like security.

2.8 Security oriented Improvements in development team

Lindvall et al. [48] has concluded that for the success of any project experienced people ought to be involved in the development process. Such a team comprises of those who have some experience in the concerned domain, have been involved in building similar kinds of systems and have considerable communication skill. It was understood that their percentage must be 25%-33% of the total number of development team members.

2.8.1 Team Software Process (TSP)

TSP [77] is an operational process for development teams. The process is very effective for producing almost defect-free software within the budget and on schedule. The software developers and managers are trained to introduce the methods into an organization at all levels. The TSP for Secure Software Development (TSP-Secure) imbibes the security practices of TSP throughout the development life cycle. Software developers receive additional training in security issues. Using TSP is difficult owing to large initial investment and the requirement of management support for the technical work and also for empowering, coaching, and motivating teams.

2.8.2 SDLC with Developers Working with Security

D. Shackleford [53] has proposed an SDLC in which the development teams and security groups work together. Security is dealt with in all phases by adding a toll-gate or review at the en;d of each phase. Moreover, security is prioritized during the requirements specification phase. This process aids in secure application development and maintaining a reasonable development schedule. All of the programmers, security personnel and the upper management share responsibility for the security issues occuring during the development.

2.9 Inferences drawn from the literature survey

This section summarizes the literature survey and lists the present state of security in SDLC and the good practices which need to be followed to achieve it effectively. The proposed SecREAD methodology attempts to assimilate all these and introduces more novel concepts for further improvement.

- Imbibing security in SDLC is the need of the hour.
- For ensuring security in the development life cycle the traditional models are no longer adequate.
- Security is best if introduced early *i.e.* requirements and design phases.
- Requirements engineering is the most crucial activity for any kind of software.
- Iteration is a good approach for the Requirements Engineering process.
- Aspirations of all types of stakeholders ought to be elicited and analyzed.
- The software and its assets must be ranked on security parameters.
- The client and experts should be involved and remain in close touch with the developers
- The ranking process should be simple for all kinds of stakeholders.
- Empirical analysis of the security ranking should be conducted.
- The present design languages do not suffice the modeling of security requirements, hence a new design language is required

Chapter 3

SecREAD Methodology

In modern software systems security is a major concern and as discussed in the previous chapter, various researchers and organizations advocate that imbibing security in the software development life cycle is necessary to plug the security vulnerabilities and to produce secure software. Moreover, researchers [37][49] have found it more economical. As already discussed, in Section 2.2 several researchers have believed early phases of development to be more suitable *i.e.* either requirements or design or both. The need of security within software development life cycle has also been established through the discussion provided in Section 2.1. After discussing and critically analyzing several methodologies that concern with imbibitions of security in the software development process in some way or the other, it has been concluded that there is an ample space to improve or develop new methodologies to address the gap or limitations. All this motivated us to propose a new methodology named Security-aware Requirements Elicitation, Assessment and Design Methodology (SecREAD). The methodology is an attempt to enhance the methodologies and overcome some of their limitations by incorporating the best practices, drawn from the literature and listed in Section 2.9.

The proposed methodology is an 'Asset-Functionality-Stakeholder' based that envisages a well-structured process to gather requirements, extract entities, find the associations among them, represent them graphically, rank them on pertinent parameters and then reflect these rankings graphically in a meaningful way with clarity for later phases in development life cycle.

In any methodology correct and complete identification of stakeholders, assets and functionalities is very important to ensure correctness, quality and other non-functional requirements of the software. Assets, stakeholders and functionalities have already been described in Section 1.3. Within this research work, data items have been considered as assets similar to SaSDLC [90] methodology described in Section 2.5.6. The potential stakeholders of any software are the members of the development team, client organization and the users. The client organization may consist of the senior management and members of different departments which will be catered to by the different functionalities of the system. There will be a chief administrator designated, who will be responsible for the working of the system at the client side. The users may be many, belonging to different groups. Some may be technically conversant while others not. Within the user community there may be several groups that will use different functionalities of the system with varied authorization rights to the assets.

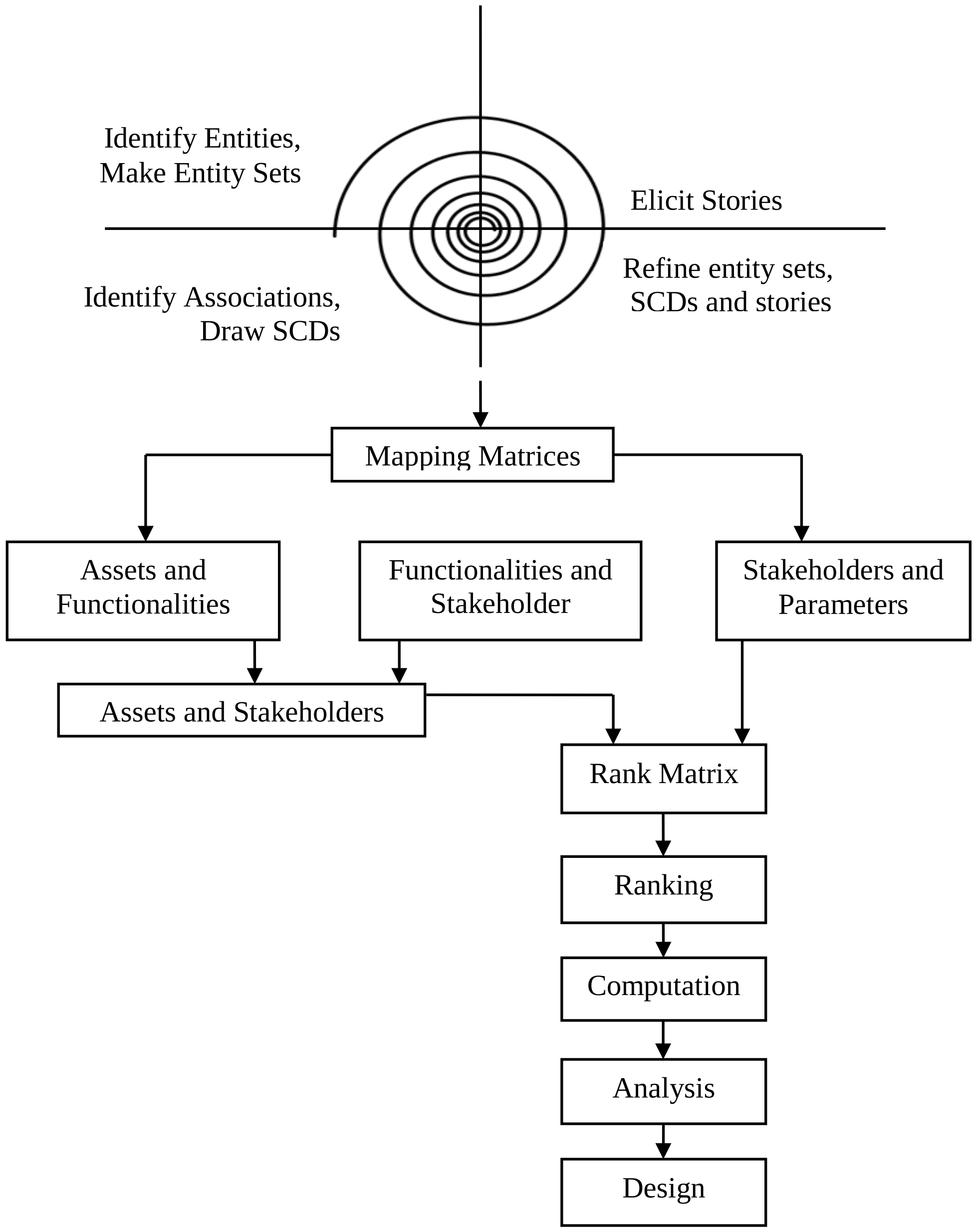
A notion of core group is proposed which consists of the client representative(s), developers, the ethical hackers to find vulnerabilities and the experts. The experts may be appointed from the client side and technical experts or experienced people from the developer organization. Experts belonging to a third party can also be included. This group advises the developers all through the development process, aids in clarification & conflict resolution, analyzes each phase of the methodology and finds pitfalls and best possible solutions through mutual discussions.

In this work five security parameters- authorization, authentication, integrity, confidentiality and non-repudiation are considered. These have already been discussed in detail in Section 2.3. The above five parameters are considered because these are general and belong to almost all software development domains. However, the methodology can easily be extended to include other parameters.

SecREAD Methodology integrates security in both requirements and design phases of software development. This methodology is unique as requirements elicitation can be conducted in natural language effectively. SecREAD Methodology involves all kinds of stakeholders of a system and at the same time consults a very competent core group. Like some prior works, it performs ranking process but unlike any of them it is formalized by a thoughtful mapping mechanism that brings in the idea of relevance. Various entities of the software are ranked over these parameters. After empirical analysis of ranks, different kinds of diagrams are drawn to give a clear and unambiguous picture of security concerns in the software that can easily be understood by all stakeholders.

3.1 Proposed Methodology: SecREAD

The proposed methodology consists of phases namely, Identification & Refinement, Mapping, Ranking, Analysis and Design. The assets and functionalities of the software are ranked by the stakeholders over parameters namely, authentication, confidentiality, authorization, integrity and non-repudiation. As it is difficult to gather the requirements for any big system in one go, the iterative approach is more practical and desirable to produce good software. Therefore, this method is used for both Identification and Refinement phases and thus, spiral method is adopted. Figure 3.1 illustrates the phases of the methodology which are discussed subsequently.



Identify Entities, Make Entity Sets
Elicit Stories
Identify Associations, Draw SCDs
Refine entity sets, SCDs and stories
Mapping Matrices
Assets and Functionalities
Functionalities and Stakeholder
Stakeholders and Parameters
Assets and Stakeholders
Rank Matrix
Ranking
Computation
Analysis
Design

Figure 3.1: SecREAD Methodology Process Flow

3.2 Identification

The identification and refinement phases are probably the most important as in this phase each and every element required to build software is to be identified. This phase is divided in four quadrants to facilitate the development process as shown in Figure 3.1. Various methods can be adopted to ensure identification or gathering of complete requirements. The two most intuitive methods are directly identifying all three entities – assets, stakeholders and functionalities, or use story-telling approach. The latter looks more practical and effective approach in designing of large software systems, as narrating stories is easier than directly recognizing the entities. Since, the system has both kinds of stakeholders *i.e.* conversant with the domain & technology and non-conversant. Further, it is mentioned that it is not possible to gather all the system requirements in one go and therefore the process moves in a spiral fashion.

As depicted in Figure 3.1 each spiral is divided into four quadrants. In the first quadrant stories are elicited from the client side and recorded by the development team. In the second quadrant the stories are analyzed and three entities- assets, stakeholders and functionalities are identified and corresponding entity sets are constructed. In the third quadrant of the spiral, the associations among the entities are identified and based on those, Story Conversion Diagrams (SCDs) are developed. Lastly, in the fourth quadrant the entity sets, the stories and the SCDs are refined. After each spiral more stakeholders, assets and functionalities are identified and added to the corresponding sets.

The Identification phase encompasses the first three quadrants. This phase is explained in the following sub-sections while the fourth quadrant deals with the refinement process which is explained in Section 3.3. The identification activities are initiated by the development team in close interaction with the client side.

3.2.1 Story and Story Conversion Diagram (SCD)

A story is a natural language narration of requirements/process by the stakeholders. Now, it is the responsibility of the development team and the core group to correctly identify the assets, functionalities and stakeholders from these stories. To formalize the design process, a notion of sets of these important entities is also introduced. The stakeholder set is denoted by S, asset set by A and functionality set by F and are defined by Eq. (3.1), Eq. (3.2) and Eq. (3.3) respectively.

$$S = \{S_1, S_2, \ldots\ldots S_i\} \quad \text{where, } 1 \leq i \leq m \tag{3.1}$$

$$A = \{A_1, A_2, \ldots\ldots A_j\} \quad \text{where, } 1 \leq j \leq n \tag{3.2}$$

$$F = \{F_1, F_2, \ldots\ldots F_k\} \quad \text{where, } 1 \leq k \leq g \tag{3.3}$$

$i, j, k, m, n, g \in I$ where, I is the set of integers

m, *n* and *q* denote the cardinality of the corresponding sets.

A new kind of diagrammatic representation, namely Story Conversion Diagram (SCD) is introduced in SecREAD methodology to facilitate the design process. These diagrams provide a pictorial representation of stories created to define the identified entities and associations among them. These diagrams are the first of their kind owing to their evolutionary and highly orthogonal nature, ease of drawing & understanding and the amount of non-functional requirements they deliver. Their conception is inspired from the famous English idiom, “A picture is worth a thousand words.” In an SCD a stakeholder is represented by a human stick diagram, functionality by an oval, asset through a rectangle and association by a line. It is established that a diagram is better than text for communication. Diagrams are highly useful in understanding the system and help in effective design process. As a story is elicited, it is analyzed to find the entities as mentioned above and associations among them. Based on the stories narrated to the developers, there can be many types of SCDs. Efforts are made to classify the stories for the ease of understanding and preparing SCDs to facilitate complete secure design of a software. Different types of stories and their SCDs are discussed next.

3.2.1.1 Single-Entity Story

This is a type of story containing any one entity. Examples are: For internet-banking system,
“The system will have a branch manager” (branch manager is a stakeholder)

For smart-home system,
"Temperature should be detected" (temperature is an asset)

For internet-banking system,
"The system should provide money transfer facility" (money transfer is a functionality)

The generalized SCDs for these examples/cases are shown in Figures 3.2(a), 3.2(b) and 3.2(c) respectively where, $S_i \in S$, $A_j \in A$ and $F_k \in F$.

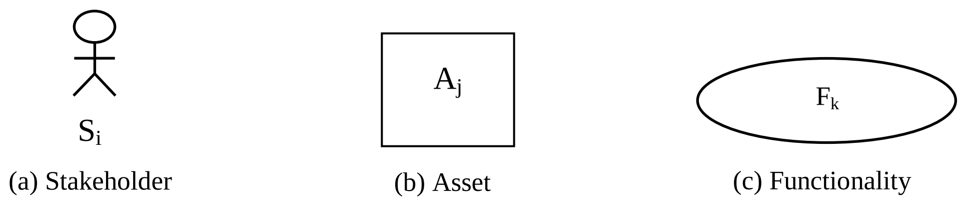


Figure 3.2 Generalized SCDs for single-entity stories

3.2.1.2 Two-Entity Story

Such stories define association among any two of the three entity types. Depending on the stories there can be three possible cases:

- stakeholder associated with asset (or vice versa),
- asset associated with functionality (or vice versa) or
- stakeholder associated with functionality (or vice versa).

The corresponding SCDs are illustrated in Figure 3.3, Figure 3.4 and Figure 3.5 along with the examples. Again in these figures, $S_i \in S$, $A_j \in A$ and $F_k \in F$.

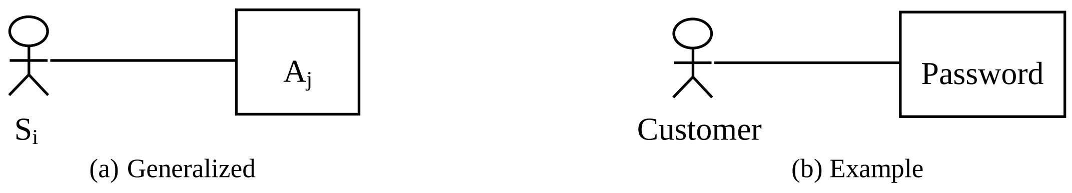


Figure 3.3: SCD for Association between Stakeholder and Asset

Figure 3.3(a) denotes association between stakeholder and asset. An example story is "Customer will have password". Here, customer is a stakeholder, password is an asset and corresponding SCD is shown in Figure 3.3(b).

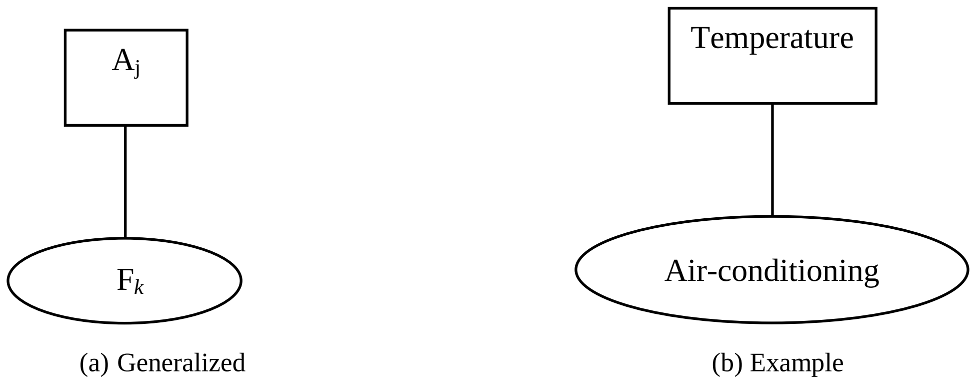


Figure 3.4: SCD for Association between Asset and Functionality

Figure 3.4(a) denotes association between asset and functionality. An example story is "Temperature will be maintained by the air-conditioning system". Where, temperature is an asset while air-conditioning is functionality. An SCD for the story is shown in figure 3.4(b).

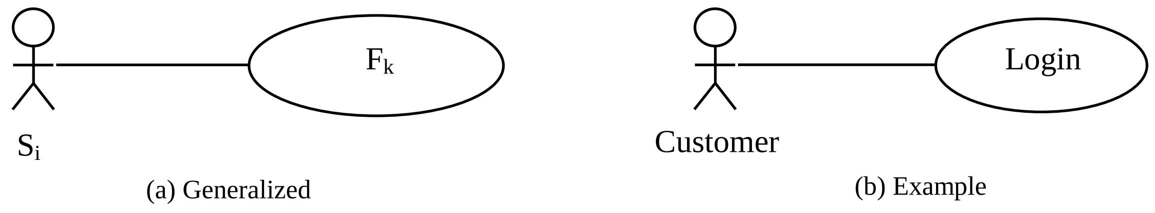


Figure 3.5: SCD for association between Stakeholder and Functionality

Figure 3.5(a) depicts association between stakeholder and functionality. An example story is "Customer will login." Where, customer is a stakeholder and login is functionality. An SCD for the example is shown in Figure 3.5(b).

3.2.1.3 Three-Entity Story

A story may contain all the three entities- stakeholders, assets and functionalities. This kind of story generally depicts:

Stakeholder can [do/perform] functionality [on/through] Asset

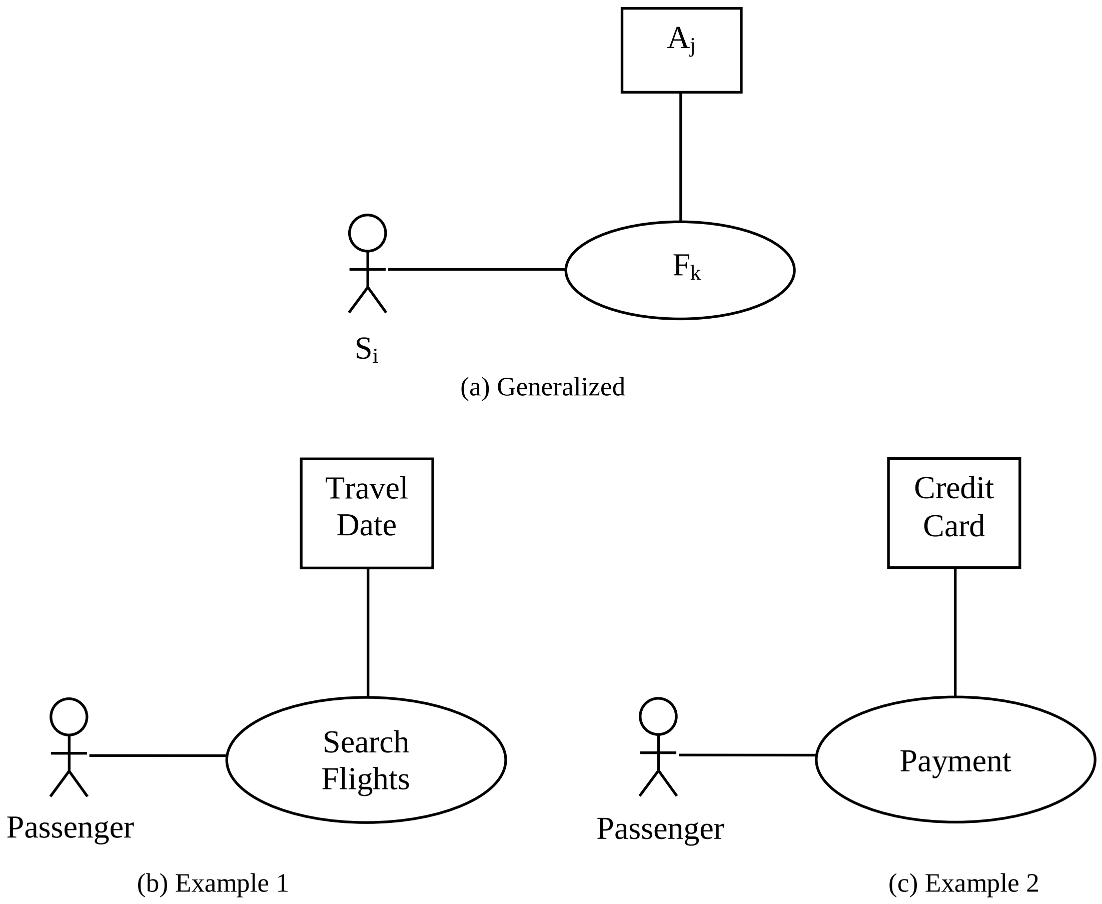


Figure 3.6: SCD for Association between Stakeholder Functionality and

Figure 3.6(a) depicts association between functionality F_k, asset A_j and stakeholder S_i. Example stories for a three-entity story from Airline System are, “Passenger can search flights selecting the travel date” and “Passenger can make payment through credit card”. SCDs for these two stories are given by Figures 3.6(b) and 3.6(c) respectively.

3.2.1.4 Multi-instance Story

In a system there can be stories which contain more than one instance of an entity type associated with other entity. SCDs need to be created for such stories. There are two possible ways for creating SCDs for such stories. One is to create a complete SCD from a multi-instance story itself. Second, is to identify the related basic SCDs already created during the development process or create new and integrate them to obtain the aggregate SCD for the multi-instance story. This method may be easier as a diagrammatic representation makes it easier to visualize the multiple instances than a story.

There can be ten possible cases of associations, illustrated in Table 3.1 along with the examples. The corresponding SCDs are presented in Figure 3.7 to Figure 3.16 respectively. For simplicity in representation, the maximum number of assets, stakeholders and functionalities is restricted to two.

Here, $S_a, S_b \in S$; $A_c, A_d \in A$ and $F_e, F_g \in F$. The examples are drawn from the Internet Banking System where, INB officer stands for Internet Banking officer or the administrator of the system.

Table 3.1: Multi-instance Stories

Case	Associations in Stories	Example
1	Many stakeholders associated with one functionality	Customer and INB Officer associated with Login
2	Many assets associated with one functionality	Username and Password associated with Login
3	Many stakeholders, one asset associated with one functionality	Customer, INB Officer, Username associated with Login
4	One Stakeholder, many assets associated with one functionality	Customer, Username, Password associated with Login
5	Many stakeholders and many assets are associated with one functionality	Customer, INB Officer, Username, Password associated with Login
6	One asset associated with many functionalities	Password associated with Login and Money Transfer
7	Many assets associated with many functionalities	Password, Username associated with Login and Money Transfer
8	One stakeholder associated with many functionalities	Customer associated with Login and Money Transfer
9	Many stakeholders associated with many functionalities	Customer and INB Officer associated with Login and Money Transfer
10	Many assets, many stakeholders associated with many functionalities	Customer, INB Officer, Username, Password associated with Login and Money Transfer

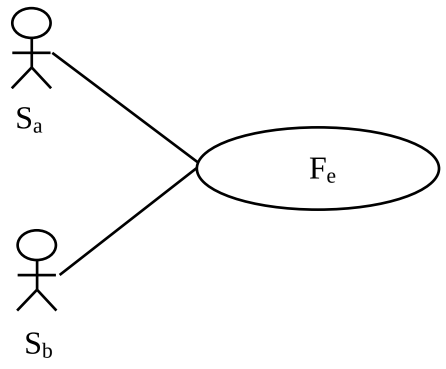


Figure 3.7: SCD for many stakeholders associated with one functionality (case 1)

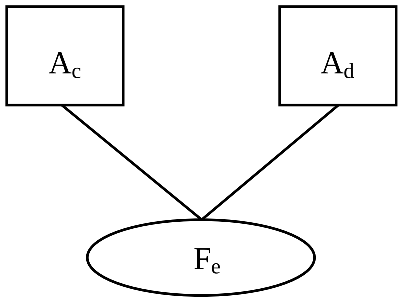


Figure 3.8: SCD for many assets associated with one functionality (case 2)

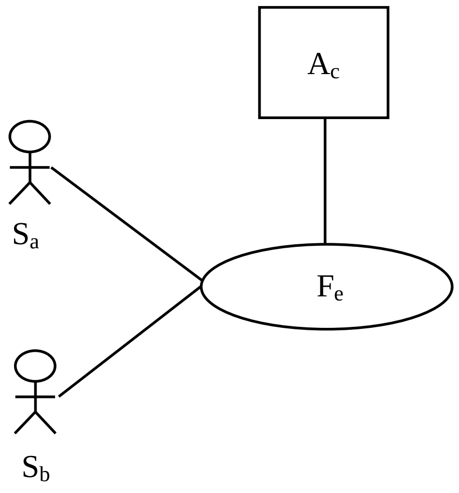


Figure 3.9: SCD for many stakeholders, one asset associated with one functionality (case 3)

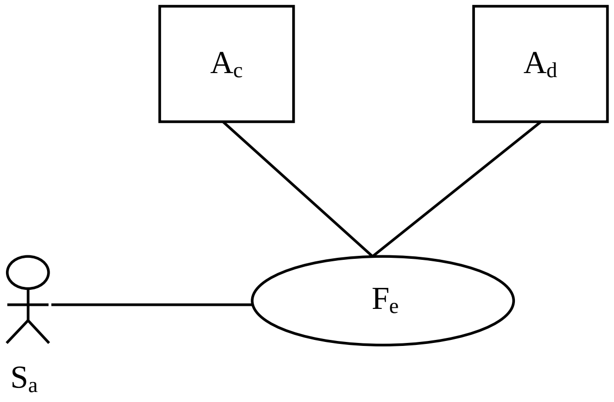


Figure 3.10: SCD for one stakeholder, many assets associated with one functionality (case 4)

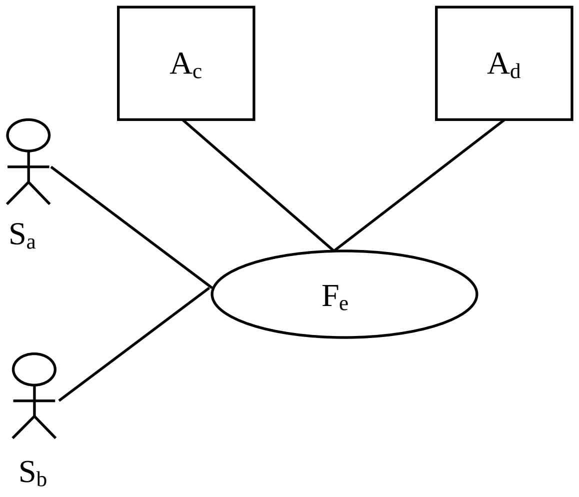


Figure 3.11: SCD for many stakeholders, many assets associated with one functionality (case 5)

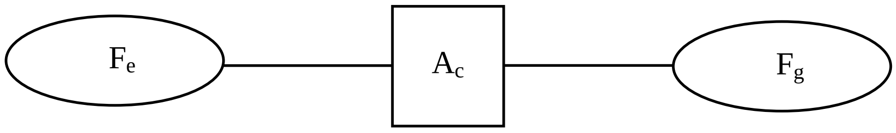


Figure 3.12: SCD for one asset associated with many functionalities (case 6)

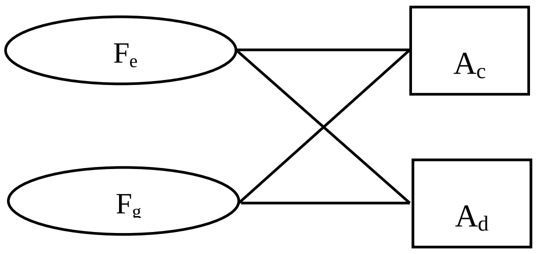


Figure 3.13: SCD for many assets associated with many functionalities (case 7)

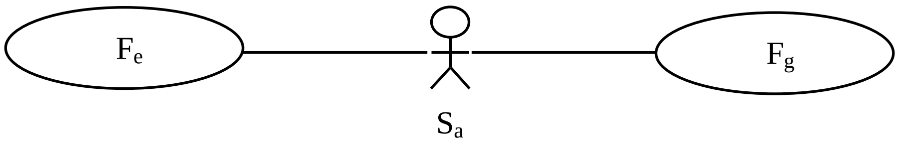


Figure 3.14: SCD for one stakeholder associated with many functionalities (case 8)

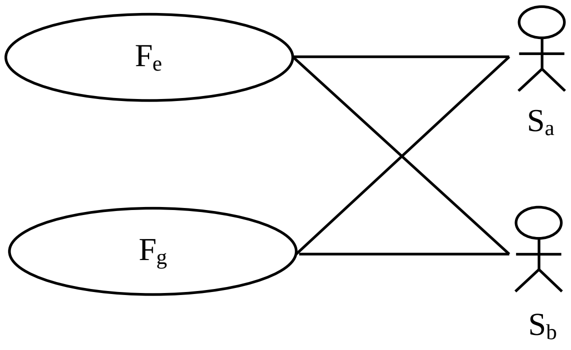


Figure 3.15: SCD for many stakeholders associated with many functionalities (case 9)

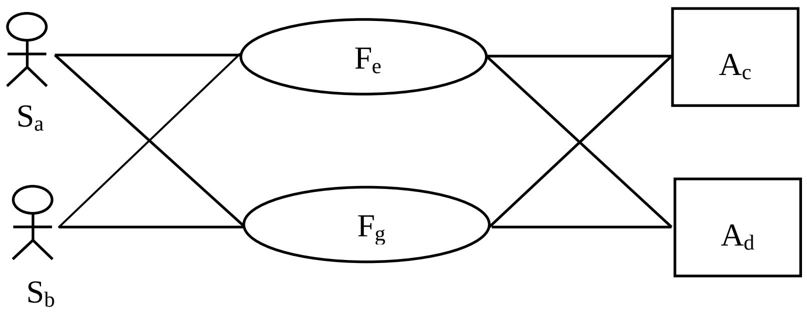


Figure 3.16: SCD for many assets, many stakeholders associated with many functionalities (case 10)

In the next section the Refinement process is discussed. Refinement is necessary to check redundancy and perform aggregation and decomposition, followed by updation of sets and SCDs, if necessary. The Identification with Refinement completes one cycle of spiral.

3.3 Refinement

In the proposed methodology as depicted in Figure 3.1, the refinement is performed in the last quadrant of each spiral. The refinement phase consists of the removal of redundancy from the entity-sets, aggregation of entities and SCDs, decomposition of entities or SCDs or both aggregation and decomposition. The refinement is performed at two levels *i.e.* at the level of the development team and the level of the core group.

It has already been mentioned that identification and refinement are iterative and continue till all the information is exhaustively compiled to ensure quality of software developed. In the identification phase, requirements are elicited from the stakeholders. As soon as a story is obtained, the entities and their relationship are extracted from it and the corresponding SCD is developed. The identified entities are added to the respective sets. The refinement phase follows the identification phase where the information gathered in the identification is cleansed and refined to check the flow of any ambiguity and redundancy to later phases. For refinement SCDs play a vital role as they exhibit the associations lucidly. Each time the refinement action is taken, the respective entity sets and the affected SCDs are revised. After refinement, final asset set A, stakeholder set S and functionality set F are obtained.

For refinement, all the entity sets and SCDs are checked and if any inconsistency is found, it is solved. First, the development team tries to solve the solution and if required they may consult the core group. The objective is to simplify the design as much as possible.

The refinement process needs refinement- redundancy removal, aggregation and decomposition which are detailed in the following sub-sections.

3.3.1 Redundancy Removal

During the identification phase, redundancy of entities may creep in. The elicited stories are the major cause of redundancy which can be large in number, collected from many stakeholders. These stakeholders have different levels of technical understanding and may perceive the system quite differently. Further, each stakeholder may be related to some and not all entities.

This difference in perception and relations can lead to the usage of more than one name for a single entity at the level of system as a whole. For example, assets like ‘Phone No.’ and ‘Mobile No.’ can create ambiguity. Likewise, for a single stakeholder two terms may be used. For example, in smart building system, ‘Administrator’ and ‘Building Manager’. Any one name ought to be decided upon and used throughout the system. If ‘Building Manager’ is selected then ‘Administrator’ should be deleted from the stakeholder set and consequently, all SCDs that contain ‘Administrator’ need to be modified. Similarly, for functionalities, two names may be elicited as ‘Fire Safety System’ and ‘Fire Alarm System’.

Algorithm 3.1: Redundancy_Removal

A = ϕ, S = ϕ, F = ϕ; *a*: cardinality of A, *b*: cardinality of S, *c*:cardinality of F

```
1.   while: final A, S, F not obtained
2.   {
3.       scan A
4.       {
5.         $\forall A_i \in A$ where, $1 \le i \le a$
6.         if $\exists A_j$ similar to $A_i$ and $j \neq i$      // similar means denoting same entity
7.             discard either $A_j$ or $A_i$        // decision taken by the core group
8.         update A
9.        }
10.      scan S
11.      {
12.        $\forall S_i \in S$ where, $1 \le i \le b$
13.         if $\exists$ $S_j$ similar to $S_i$ and $j \neq i$        // similar means denoting same entity
14.             discard either $S_j$ or $S_i$             // decision taken by the core group
15.         update S
16.         }
17.       Scan F
18.       {
19.         $\forall$ $F_i \in$ F where, $1 \leq i \leq c$
20.         if $\exists$ $F_j$ similar to $F_i$ and $j \neq i$        // similar means denoting same entity
21.             discard either $F_j$ or $F_i$             // decision taken by the core group
22.           update F
23.       }
24.   }
```

3.3.2 Decomposition and Aggregation

This sub-section, deals with the application of decomposition and aggregation techniques of refinement on stories and entities.

3.3.2.1 Aggregation

When two or more stories convey similar information these are aggregated into one and so do their SCDs. For a single functionality F_k, all such cases are enlisted in Table 3.2. Here, for simplicity in representation, the maximum number of assets and stakeholders in a story is restricted to two. The aggregated stories render the same meaning as the multi-instance stories in Sub-section 3.2.1.4. The five cases correspond to the first five cases of the multi-instance stories and their respective SCDs *i.e.* Figure 7 to Figure 11. Same examples are applicable here. S_a, $S_b \in$ S and A_c, $A_d \in$ A.

Table 3.2 : Aggregated Stories

Case	Multiple Stories	Aggregate Story	Example
1	"S_a is associated with F_k" and "S_b is associated with F_k"	"S_a, S_b are associated with F_k"	Customer and INB Officer associated with Login
2	"A_c is associated with F_k" and "A_d is associated with F_k"	"A_c, A_d are associated with F_k"	Username and Password associated with Login
3	"S_a and A_c are associated with F_k" and "S_b and A_c" are associated with F_k	"S_a, S_b, A_c are associated with F_k"	Customer, INB officer, Username associated with Login
4	"S_a and A_c are associated with F_k" and "S_a and A_d are associated with F_k "	"S_a, A_c, A_d are associated with F_k"	Customer, Username, Password associated with Login
5	"S_a and A_c are associated with F_k" and "S_b and A_d are associated with F_k"	"S_a, S_b, A_c, A_d are associated with F_k"	Customer, INB Officer, Username, Password associated with Login

3.3.2.2 Decomposition

To facilitate understanding and cater to the specific requirements of software, it may be required to deal with entities differently. Sometimes an asset, functionality or stakeholder needs to be decomposed into its constituents. If decomposition takes place, the entity is replaced by its constituents. For example, entity 'Customer Details' may be decomposed into Customer Name, Customer ID, Customer Address and Customer Phone No. Conversely, the latter four may be aggregated into the former. Functionalities may also be required to be decomposed or aggregated. For example in an airline booking system, 'Payment for baggage' and 'Payment for food and beverages' can be aggregated as 'Payment for services'.

3.3.2.3 Addition of entities

There is also a provision for addition of entities for operational, technical or security reasons. Usually it is done on the advice of the client. However, inputs may be provided by the developers and experts out of their experience. For example, in internet banking system, 'User Profile Password' is added for further securing money transfer to a third party for the first time.

3.3.3 Culmination of Spiral

The entire spiral moves once for each story elicited. In the next cycle, the newly identified stakeholders sit together with the previous stakeholders to identify more stakeholders, functionalities and assets. In this manner, the participants in the meetings increase incrementally and more entities are found. At the culmination of this phase asset, stakeholder and functionality sets are obtained exhaustively.

3.4 Mapping

In the previous phase, we have determined three entities *i.e.* stakeholders, assets and functionalities. These entities are associated with each other as depicted by the SCDs. In this phase, an effort is made to formalize these associations or in other words, determine the relevance of entities with each other. The mapping is a process in which such relevance is presented explicitly. For this, relevance matrices are developed. Four such matrices are proposed namely, Asset-Functionality (X), Functionality-Stakeholder (Y), Asset-Stakeholder (Z) and Stakeholder-Parameter (W). These matrices are developed by the core group. All of these matrices are described in the following sub-sections. In every matrix, the shaded cells denote relevance and un-shaded cells denote irrelevance. For the sake of explanation four stakeholders, five functionalities and six assets are taken as follows:

$S_1, S_2, S_3, S_4 \in S$

$F_1, F_2, F_3, F_4, F_5 \in F$

$A_1, A_2, A_3, A_4, A_5, A_6 \in A$

3.4.1 Asset-Functionality and Stakeholder-Functionality Relevance Matrices

Asset-Functionality and Stakeholder-Functionality relevance matrices are created by the analysis of the requirements gathered or the SCDs developed in the identification phase. The relevance matrices map the related entities.

In Asset-Functionality relevance matrix X Assets and Functionalities are mapped to each other where each row represents assets and columns represent functionalities. It is found that all assets are not related to all functionalities (or vice versa). Therefore, it is necessary to distinguish the cells that denote with the ones that do not show any relationship. The cells that show relationship are shaded. Similarly, matrix Y is developed where, rows contain functionalities and columns contain stakeholders.

$$X = (x_{jk}) \tag{3.4}$$

where, j represents assets and k represents functionalities;

$1 \leq j \leq n$; $1 \leq k \leq g$;

$$Y = (y_{ki}) \tag{3.5}$$

where, k represents functionalities and i represents stakeholders;

$1 \leq i \leq m$;

n is cardinality of A; g is cardinality of F; m is cardinality of S

X and Y are given by Tables 3.3 and 3.4 respectively

Table 3.3: Asset – Functionality Relevance Matrix X

Assets	**Functionalities**				
	F_1	F_2	F_3	F_4	F_5
A_1	▨				
A_2	▨		▨	▨	▨
A_3	▨				
A_4	▨				
A_5	▨				
A_6					

Table 3.4: Functionality – Stakeholder Relevance Matrix Y

Functionality	**Stakeholders**			
	S_1	S_2	S_3	S_4
F_1	▨	▨	▨	
F_2				▨
F_3				▨
F_4				▨
F_5		▨	▨	▨

3.4.2 Asset – Stakeholder Relevance Matrix

It is required to find relation between assets and stakeholders in order to get the assets ranked by the relevant stakeholders. Asset – Stakeholder Relevance Matrix Z is determined by the cross product of matrix X and matrix Y, given by Eq. (3.6). Z is described by Eq. (3.7). It is developed to ensure that the assets are ranked by the relevant stakeholders only.

$$Z = X * Y \tag{3.6}$$

$$Z = (z_{ji}) \tag{3.7}$$

where, j represents assets and i represents stakeholders;

$1 \leq j \leq n; 1 \leq i \leq m;$

To obtain the cross product of X and Y it is necessary to fill their cells by some integer value. To facilitate the product, 'one' is used to denote the filled cell and 'zero' for empty cells. Table 3.5 and Table 3.6 are X and Y respectively. Table 3.7 is their cross product Matrix Z.

Table 3.5: Asset – Functionality Relevance Matrix X with Integral Values

Assets	**Functionalities**				
	F_1	F_2	F_3	F_4	F_5
A_1	1	0	0	0	0
A_2	1	0	1	1	1
A_3	1	0	0	0	0
A_4	1	0	0	0	0
A_5	1	0	0	0	0
A_6	0	0	0	0	0

Table 3.6: Functionality – Stakeholder Relevance Matrix Y with Integral Values

Functionality	**Stakeholders**			
	S_1	S_2	S_3	S_4
F_1	1	1	1	0
F_2	0	0	0	1
F_3	0	0	0	1
F_4	0	0	0	1
F_5	0	1	1	1

Table 3.7: Matrix Z or Cross product of X and Y

Assets	**Stakeholders**			
	S_1	S_2	S_3	S_4
A_1	1	1	1	0
A_2	1	2	2	3
A_3	1	1	1	0
A_4	1	1	1	0
A_5	1	1	1	0
A_6	0	0	0	0

Any non-zero value denotes relevance between the particular stakeholder and asset, while zero denotes irrelevance. Matrix Z is a very important matrix on which the final rank matrix, to be filled by stakeholders, is constructed. Through the cross product it is ensured that the relatively complex transitive relationship between assets and stakeholders is represented easily and correctly. Z actually signifies that the stakeholders related to a particular functionality ultimately relate to the assets of the functionality. How Z represents the said association can be understood through the following example.

In matrix X (Table 3.5), $x[A_1, F_1] = 1$ or asset A_1 is related to functionality F_1;

In matrix Y (Table 3.6), $y[F_{1,} S_1] =1$ or functionality F_1 is related to stakeholder S_1;

Consequently, in matrix Z (Table 3.7), $z[A_{1,} S_1]$ = (some non-zero value)

or asset A_1 is related to stakeholder S_1

For the sake of generalization, the cells containing non-zero values are shaded to show the relevance. Table 3.8 denotes Z with shaded cells.

Table 3.8: Asset – Stakeholder Relevance Matrix Z with Shaded Cells

Assets	**Stakeholders**			
	S_1	S_2	S_3	S_4
A_1	■	■	■	
A_2	■	■	■	■
A_3	■	■	■	
A_4	■	■	■	
A_5	■	■	■	
A_6				

3.4.3 Consolidated Story Conversion Diagram (CSCD)

It is seen in the previous sub-sections that matrix X relates to assets and functionalities while matrix Y relates to functionalities and stakeholders. The combination of the information in these two matrices results in the presentation of associations among assets, stakeholders and functionalities. From these associations one functionality is selected at a time and its associated assets and stakeholders are found. CSCD is a pictorial representation of this relation. This diagram evolves from the SCDs. In other words, it is a consolidation of the information presented by all the SCDs for certain functionality. A CSCD is obtained through the information presented by matrices X and Y, using Algorithm 3.2. All the relevant assets and stakeholders are attached to the functionality. It provides a holistic view of a functionality lucidly. Figure 3.17 shows a CSCD for functionality F_1.

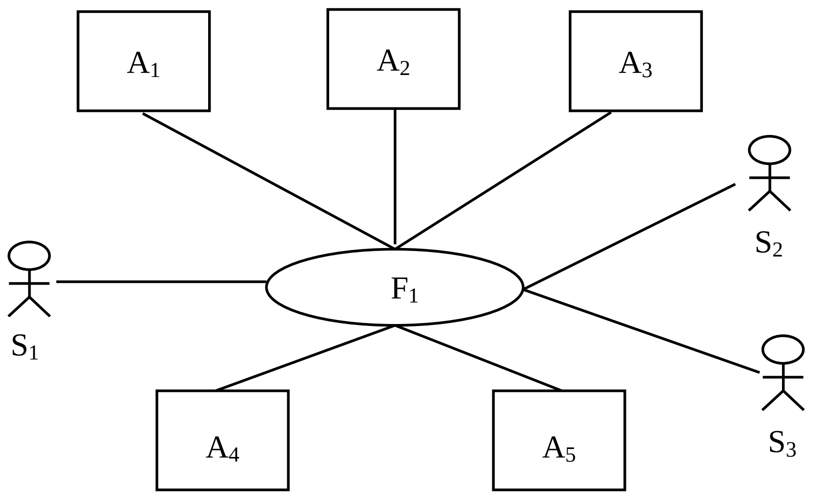


Figure 3.17: CSCD for Functionality F_1

Algorithm 3.2: Obtain_CSCD

S = ϕ, A = ϕ, F = ϕ; *q*:cardinality of F

```
25.  while: final S, F, A not obtained
26.  {
27.      elicit Story
28.         identify asset(s), functionality(s) and/or stakeholder(s) from the story
29.         if found functionality $F_k$ and $F_k \notin F$
30.           insert $F_k$ in F
31.         endif
32.         if found stakeholder $S_i$ and $S_i \notin S$
33.           insert $S_i$ in S
34.         endif
35.         if found asset $A_j$ and $A_j \notin A$
36.           insert $A_j$ in A
37.         endif
38.         identify the association among entities
39.         draw preliminary SCDs
40.         if ambiguities, inconsistencies and/or redundancy exist in S, F and/or A
41.         {
42.           Conduct Refinement: decomposition, aggregation and/or redundancy
      removal
43.           update S, F, A
44.           update SCDs
45.         }
46. }
47. Mapping (final S, F, A)
48. {
49.    obtain X i.e. map assets and functionalities
50.    obtain Y i.e. map functionalities and stakeholders
51.    cross product X and Y and obtain Z              //map assets and stakeholders
52. }
53. Draw_CSCD (F)
54. {
55.  use X and Z
56.   $\forall F_i \in F$, attach all relevant stakeholders and assets where $1 \le i \le q$
57. }
```

3.4.4 Stakeholder-Parameter Relevance Matrix

It has already been stated that our aim is to develop a methodology to create secure software. To achieve this goal we need to rank the assets on the relevant parameters and a Stakeholder-Parameter Relevance Matrix W is defined which explicitly relates the stakeholders to their relevant parameters. W is described by Eq. (3.8).

$$W = (w_{ih}) \tag{3.8}$$

where, i represents stockholders and h represents parameters;

$1 \leq i \leq m;$ m is cardinality of set S

$1 \leq h \leq p;$ p is cardinality of parameter set P

P = {Authentication, Integrity, Confidentiality, Non-repudiation, Authorization}

For ease of representation, each parameter is assigned an ID as shown in Table 3.9. Table 3.10 represents matrix W with S_1, S_2, S_3 and S_4 as stakeholders.

Table 3.9: Parameter IDs

Parameter ID	**Parameter**
Pr_1	Authentication
Pr_2	Integrity
Pr_3	Confidentiality
Pr_4	Non-repudiation
Pr_5	Authorization

Table 3.10: Stakeholder-Parameter Relevance Matrix W

Stakeholders	**Parameters**				
	Pr_1	Pr_2	Pr_3	Pr_4	Pr_5
S_1	▨	▨	▨	▨	
S_2	▨	▨	▨	▨	
S_3		▨	▨		
S_4			▨		

3.4.5 Rank Matrix: A practical approach to ranking

Ranking is an important task that not only defines the security level of the software but also the development cost. It requires sincere efforts at the level of stakeholders.

At the same time ranking should be easier for all kinds of stakeholders, then only it can reap significant benefits. Keeping this tradeoff in mind rank matrix R is proposed. It is unique in the sense that it allows the ranking process to be performed, for the entire system, through a single customized sheet for a particular stakeholder.

R is developed by the combination of matrices Z and W. Z maps assets and stakeholders while W maps stakeholders and parameters. Consequently, R relates assets, parameters and stakeholders. It is three-dimensional. The matrix is defined by Eq. (3.9). Figure 3.18 denotes its structure.

$$R = (r_{jhi}) \tag{3.9}$$

where, j represents assets, i represents stakeholders, h represents parameters;

$1 \leq j \leq n$; where, n is cardinality of set A;

$1 \leq h \leq p$; where, p is cardinality of set P;

$1 \leq i \leq m$; where, m is cardinality of set S;

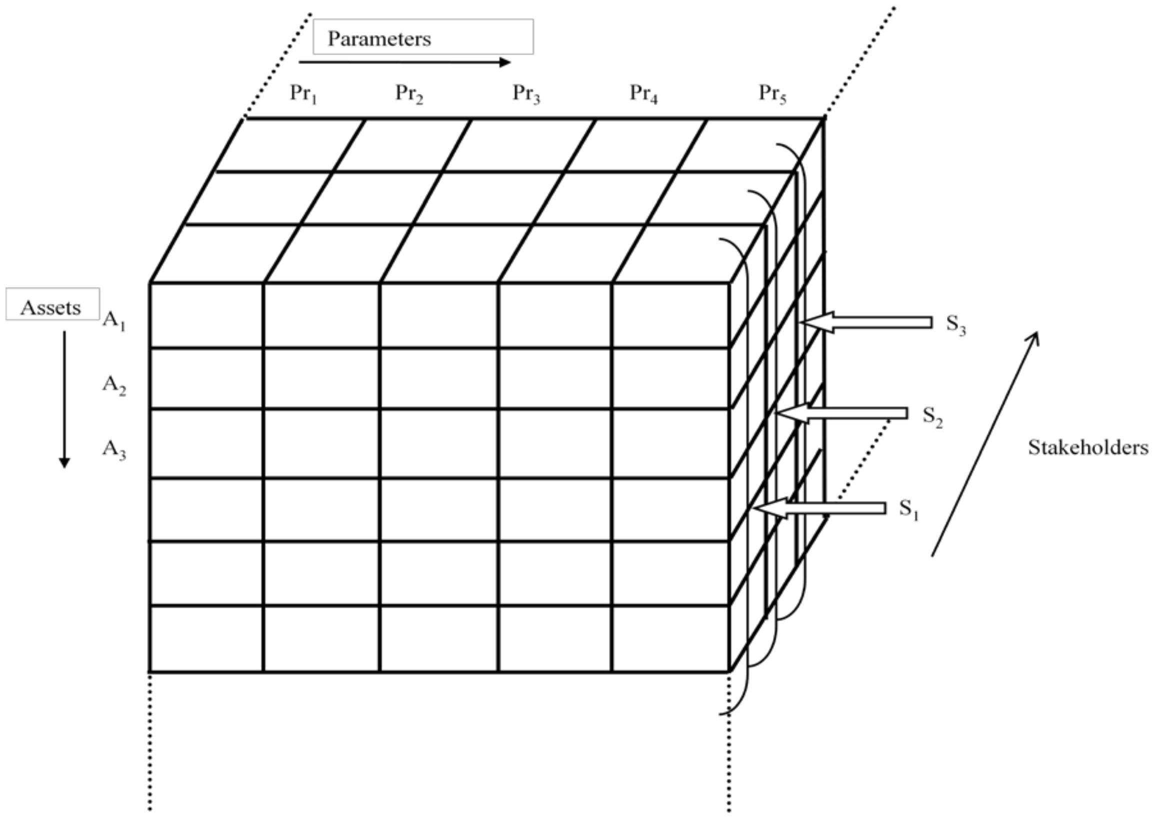


Figure 3.18: Structure of the rank matrix R

3.5 Ranking

Ranking of assets is performed over five parameters viz. authorization, confidentiality, non-repudiation and integrity through the rank matrix R. Ranking helps in determining the level of security of each asset and accordingly takes measures. In this work, three levels of High, Medium and Low are taken for ranking. Each level is assigned a numeric value *i.e.* 3, 2 and 1 respectively. This is presented in Table 3.11

Table 3.11: Ranks and Numeric Values

Rank	**Numeric Value**
High	3
Medium	2
Low	1

Through matrix R, the effort in ranking is minimized. Functionalities are not considered directly. However, it relates to the stakeholders indirectly through their relevant assets. This removal of functionalities aids in obtaining a single sheet for one kind of stakeholder. In this way, the stakeholder can perform ranking for the entire software through one personalized sheet only. Table 3.12 is one sheet of the rank matrix meant for one stakeholder S_1. This is formed as per the matrices Z and W given by Table 3.8 and Table 3.10 respectively.

Table 3.12: Rank sheet for stakeholder S_1

Assets	**Parameters**				
	Pr_1	Pr_2	Pr_3	Pr_4	Pr_5
A_1					
A_2	▨	▨	▨	▨	
A_3	▨	▨	▨	▨	
A_4					
A_5	▨	▨	▨	▨	
A_6	▨	▨	▨	▨	

Stakeholder ranks the assets relevant to him/her over the parameters relevant to him/her. The matrix given by Table 3.12 is specifically for the stakeholder S_1 where the cells valid to him/her are shaded that can be ranked as 3, 2 or 1. The un-shaded cells denote irrelevance and contain null values. The ranks sheet may differ for different stakeholders due to which they may furnish ranks in some other cells and some other cells may contain null values. As it will obviously be incongruous for stakeholders to decide their authorization rights own-self, the Authorization ranks for every stakeholder are ranked by the client in consultation with the domain expert. Sub-section 3.5.5 throws more light on how this parameter is dealt with.

The core group can assign fixed rank to a parameter or an asset considering the domain of the software like ‘High’ for ‘Integrity’ parameter in the Railway Reservation System or ‘High’ for ‘User profile Password’ under Confidentiality parameter.

After completing the ranking process, the ranks furnished by all stakeholders need to be consolidated to obtain individual ranks for assets, parameters, functionality and authorizations in the software. Based on these final values, design will be conducted as will be seen later in Section 3.7. Priority lists can be obtained which aid the development team in prioritizing effort, time and cost. For this, firstly, the 3-d R is reduced to a 2-d matrix T with assets as rows and parameters as columns. The cells in T contain the consolidated values of ranks given by all stakeholders. Hence, the third dimension or z-axis of Stakeholders is removed. T is given by Eq. (3.10). The consolidation is done through the Cell Computation and the ranks are determined using four other kinds of computations, all of these are described in next sub-sections.

$$T = (t_{jh}) \tag{3.10}$$

where, j represents assets and h represents parameters;
$1 \leq j \leq n; n$ is cardinality of set A
$1 \leq h \leq p$; p is cardinality of set P

3.5.1 Cell Computation

To obtain the rank-value of an asset under a parameter, the mode of all values contained in the same cell position in all the sheets of R is computed. It is noteworthy that values in some cells will be null due to irrelevance. Since the data stored in the matrices is ordinal *i.e.* rank-based, mean and methods of central tendency are not applicable [108]. So, mode has been used for ranking. This will also produce exactly one of the three integral values *i.e.* 1, 2 and 3 providing clear demarcation of the ranks. This mode is stored in the same cell position in the consolidated matrix T. This value of mode is taken as the rank of jth asset under hth parameter. In case of multiple modes, that with the highest value is allotted. It is done reckoning the spirit of methodology which is security that can not be compromised. Same is followed for all other computations. Algorithm 3.2 illustrates the cell computation. Here, number of assets is denoted by n, number of stakeholders by *m* and number of parameters by *p*. Null values in the cells denoting irrelevance are excluded in the calculation of mode in all the computations.

Algorithm 3.2: computation_cell

n is cardinality of A; *p* is cardinality of P; *m* is cardinality of S

j=1; h=1; i=1

r is an element of R, t is an element of T

```
1.  while $j \le n$
2.  {
3.      while $h \le p$
4.      {
5.          $t_{jh}$ = mode ($r_{jhi}, r_{jhi+1}, r_{jhi+2}, \ldots\ldots, r_{jhm}$)
            // z-axis or the stakeholders are incremented by keeping rows i.e. asets and columns i.e.
            parameters constant; null values are excluded
6.          $h=h+1$          // incrementing parameters
7.      }
8.      $j=j+1$              //incrementing assets
9.  }
```

3.5.2 Calculating Asset Rank

As already mentioned, after obtaining matrix T, the rank of every asset is calculated. For this, row computation is performed. The mode of all values of an asset under all relevant parameters (row of T) is calculated which serves as the rank of that asset. The rank of the jth asset under p number of parameters is calculated using asset_rank algorithm given by Algorithm 3.3. These ranks are stored in a single-column matrix U. Matrix U is defined by Eq. (3.11).

$$U = (u_j) \tag{3.11}$$

where, $1 \le j \le n$; n is cardinality of A

Algorithm 3.3: asset_rank

n is cardinality of A; p is cardinality of P;

j=1, h=1,

t is an element of T; u is an element of U

```
1.  while $j \le n$
2.  {
3.      $u_j$ = mode ($t_{jh}$, $t_{jh+1}$, …..$t_{jp}$)
            // columns (parameters) are incremented keeping rows i.e. assets constant; null values are
            excluded
4.  $j=j+1$                //incrementing assets
5.  }
```

3.5.3 Calculating Parameter Rank

To calculate the rank of a parameter in the whole system, column computation is performed. For this, mode of all relevant asset ranks under each parameter (column of T) is obtained and stored in a single row matrix V. The rank of the hth parameter with n number of assets is calculated using algorithm 3.4. Matrix V is defined by Eq. (3.12).

$$V = (v_h) \qquad (3.12)$$

where, h represents parameters, $1 \le h \le p$; p is cardinality of set P

Algorithm 3.4: parameter_rank

n is cardinality of A; p is cardinality of P;

j=1, h=1,
t is an element of T; v is an element of V

```
1.  while $h \le p$
2.  {
3.      $V_h$ = mode ($t_{jh}$, $t_{j+1h}$, $t_{j+2h}$, ……$t_{nh}$)
          // rows (assets) are incremented keeping columns (parameters) constant; null values are
          excluded
4.      $h=h+1$     //incrementing assets
5.  }
```

Figure 3.19 illustrates the structure of the matrix T and how matrices U and V are derived from it.

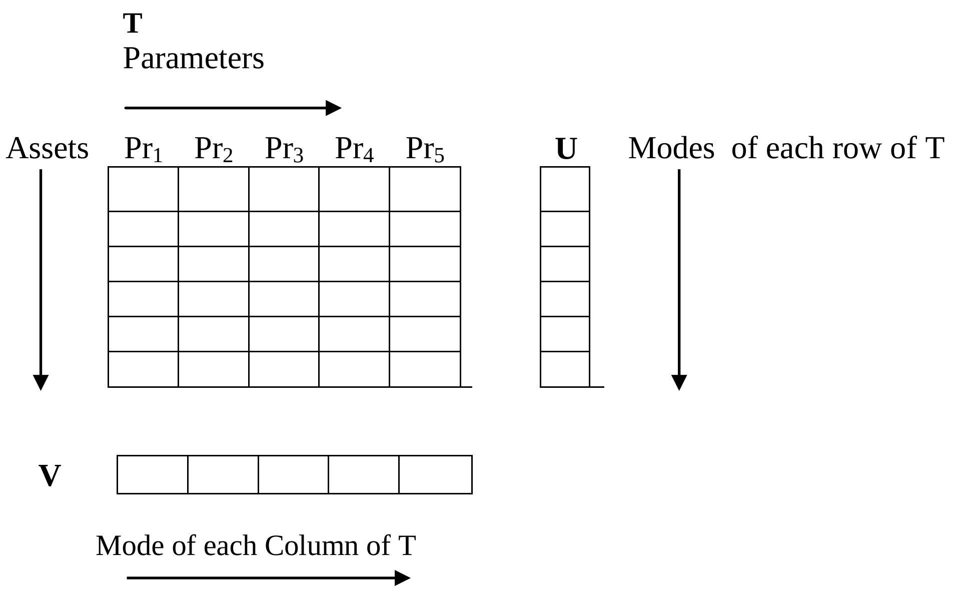


Figure 3.19: Structure of T, U and V matrices

3.5.4 Calculating Functionality Rank

The mode of the ranks of a functionality's relevant assets is taken as its rank. For this, matrix X, that maps assets and functionalities, is used along with asset rank matrix U. The rank of the functionalities are stored in single column matrix Q. Matrix Q is defined by Eq. (3.13). Algorithm 3.5 computes the functionality rank.

$$Q = (q_k) \qquad (3.13)$$

where, k represents parameters, $1 \leq k \leq g$ is cardinality of set F or the number of functionalities in the software.

Algorithm 3.5: functionality_rank

n is cardinality of A; *g* is cardinality of F

j=1, *k*=1

j denotes assets; *k* denotes functionalities

x is an element of X; q is an element of Q

```
1.  while $k \leq g$
2.  {
```

```
3.        for every functionality $F_k \in F$
4.        {
5.            $q_k$= mode of all assets $A_j \in A$, if $x[A_j, F_k] = 1$
6.            if $j \le n$
7.            {
8.                $j=j+1$
9.            }
10.           $k=k+1$
11.       }
12.   }
```

3.5.5 Authorization Computation

The authorization parameter needs a different treatment since authorization rights are determined for a stakeholder with his/her relevant assets individually. Calculating its mode in cell computation will result in same authorization right of all stakeholders over an asset, which is obviously not feasible. The ranks of this parameter are furnished by the client in consultation with core group, if required. For this, the rank matrix of each stakeholder has to be considered individually. Here, the consolidated matrix T will not work but 3-dimensional matrix R will be used, as through that the value for each stakeholder can be obtained individually. It means the particular sheet of a particular stakeholder will be used. The rank of authorization parameter (5th column in R) for jth asset with respect to ith stakeholder is given by Eq. (3.14).

$$\text{Authorization rank of } A_j \text{ w.r.t. to } S_i = r_{j5i} \tag{3.14}$$

where, $1 \le j \le n$; n is cardinality of set A; $1 \le i \le m$; m is cardinality of set S;

5 is fifth column or Authorization parameter in R; r is element of R;

3.6 Analysis

To find the rank of any functionality, the final rank of only its relevant assets is considered from matrix U. These relevant assets with respect to the functionality can be found through the matrix X (Table 3.3).

A Functionality Template is prepared for each functionality. The templates used in SecREAD are inspired from those used in Misuse Cases [33][102] described in Sub-section 2.7.4. A template is a textual representation of the diagram. It helps in avoiding any ambiguity or inconsistency. From the template, diagram can be drawn and vice versa. The template includes the names of assets with an optional brief description along with their ranks and the rank of the functionality with the suggested security measure.

Table 3.13 is a template for functionality F_1 which is made with associations as per the CSCD in Figure 3.17. Table 3.14 gives the functionality rank matrix Q that shows the ranks of all the functionalities of the system. The description and ranks have been assumed for explanation.

Table 3.13: Functionality Template for F_1

Functionality: F_1		
Stakeholders		
Name	Description/Role	
S_1	Administrator	
Assets		
Asset name	Description	Rank
A_1	(Description of A_1)	Low
A_2	(Description of A_2)	Medium
A_3	(Description of A_3)	High
A_4	(Description of A_4)	Medium
A_5	(Description of A_5)	Medium
Security Rank: Medium		
Security Measure: (as suggested)		

Table 3.14: Functionality Rank Matrix Q

Functionality	**Security Rank**
F_1	Medium
F_2	Medium
F_3	Low
F_4	High
F_5	High

3.7 Design

In the design phase of SecREAD, ranks of every functionality with its relevant assets and stakeholders is represented diagrammatically. Also, diagrams are drawn to represent authorization rights. These are described in the following sub-sections.

3.7.1 Functionality Rank Diagram (FRD)

An FRD, for a functionality, is an extension of its CSCD. It has evolved when rank information is added to a CSCD. The rank of a functionality is denoted through the number of concentric elongated ovals around it *i.e.* a single oval for low rank, double for medium rank and triple for high rank. The relevant assets are connected with their ranks, depicted by concentric rectangles *i.e.* one rectangle for low, two for medium and three for high. Figure 3.20 is the FRD for the F_1 functionality. It can be seen in the figure that functionality F_1 has medium rank, it is associated with stakeholder S_1 , S_2 and S_3 along with five assets A_1, A_2, A_3, A_4 and A_5. Asset A_1 has low security rank while A_3 has high security rank. A_2, A_4 and A_5 are of medium security rank. Assets with the same ranks can be clubbed together as seen in Figure 3.21 to avoid cluttering. Similar to the above diagrams, diagrams of other functionalities of the system are developed.

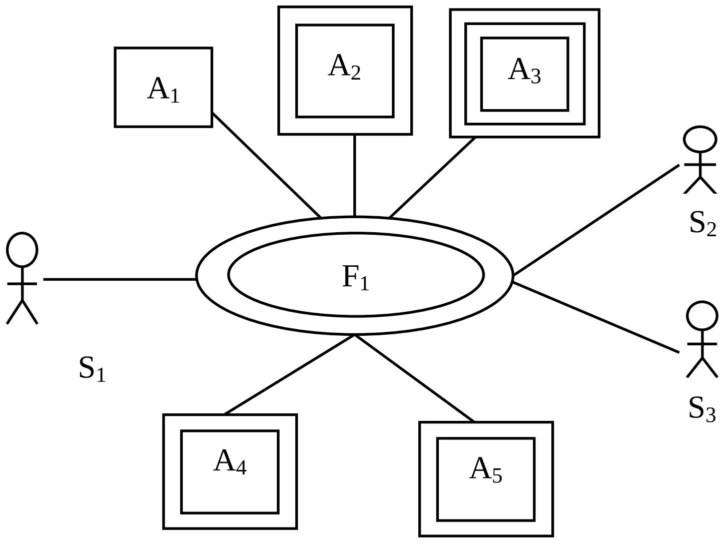


Figure 3.20: FRD for functionality F_1

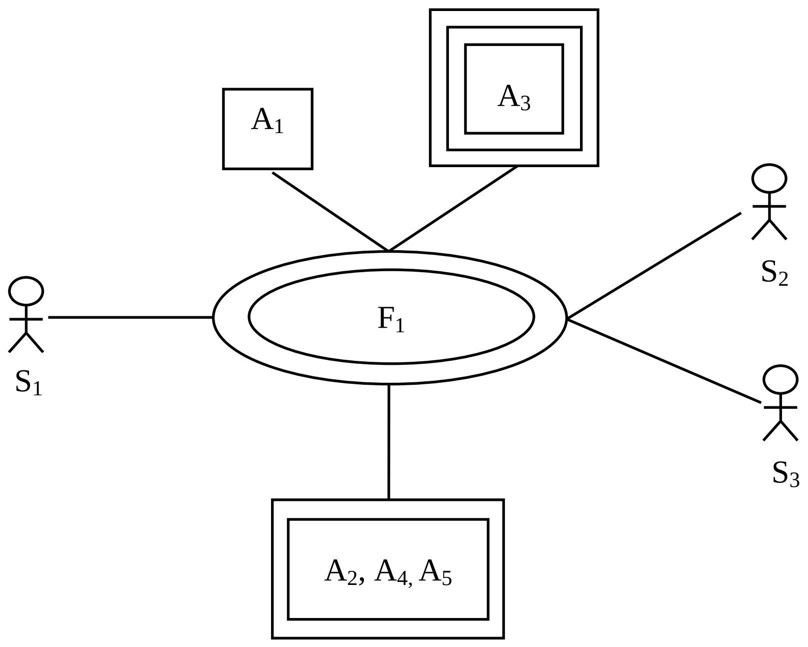


Figure 3.21: FRD for functionality F_1 with clubbed assets

3.7.2 Comprehensive Rank Diagram (CRD)

Figure 3.22 is an example CRD that summarizes the complete system and depicts all functionalities of the system with their ranks and stakeholders.

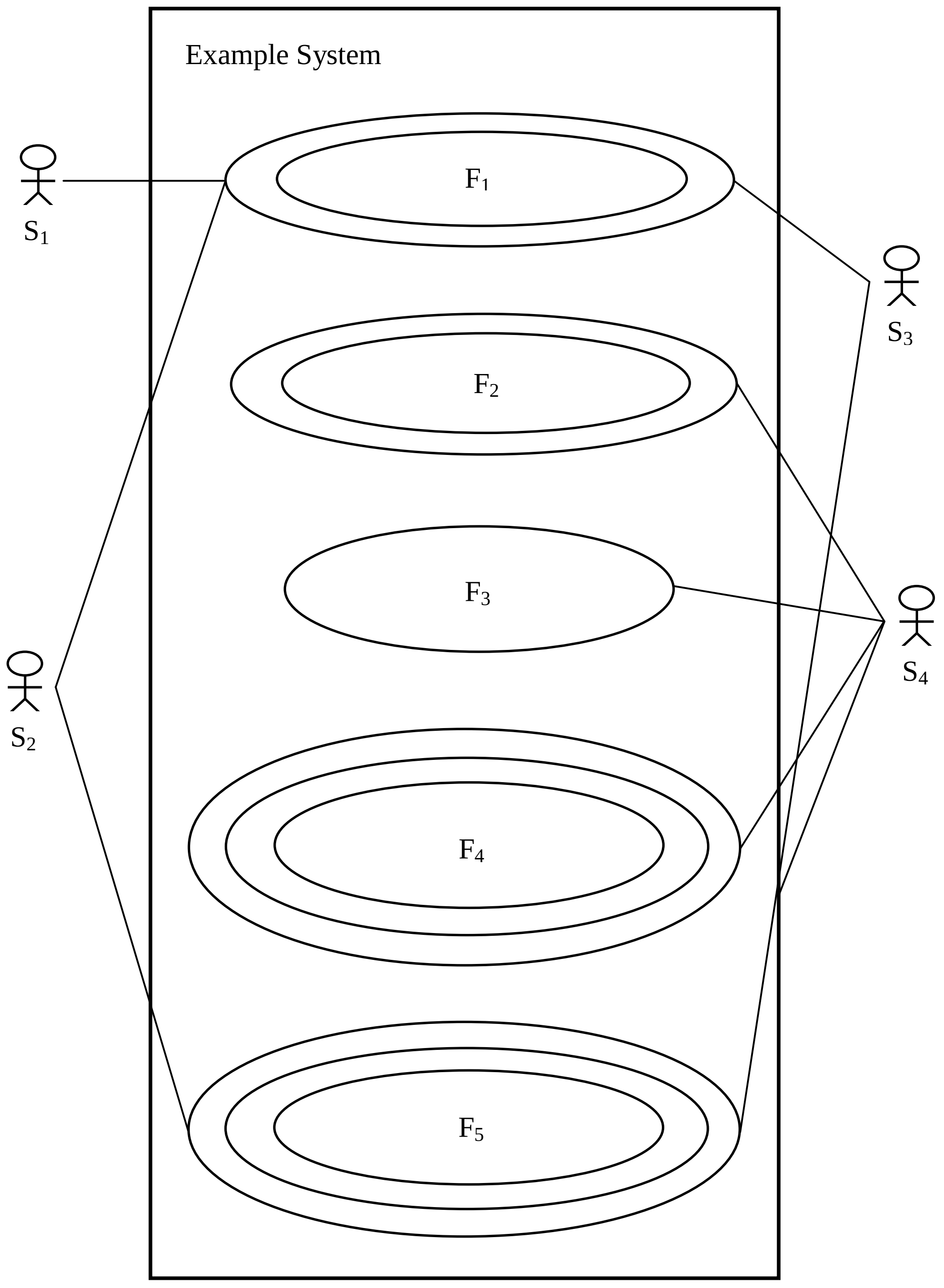


Figure 3.22: CRD for an example system

A CRD is formed through a combination of all FRDs. However, the assets are dropped to avoid cluttering. It presents an overview of the system at a glance. It evolves when all FRDs are obtained. Table 3.15 is the template for this CRD.

Table 3.15: Template for CRD

Name of System: Example		
Functionality	Rank	Relevant Stakeholders
F_1	Medium	S_1, S_2,S_3
F_2	Medium	S_4
F_3	Low	S_4
F_4	High	S_4
F_5	High	S_2, S_3,S_4

3.7.3 Authorization Rank Diagram (ARD)

Apart from showing associations, an Authorization Rank Diagrams (ARD) denotes the authorization or access rights of a stakeholder with respect to his/her relevant assets. Three ranks, high medium and low of authorization transform into ‘Write’ (W), ‘Write with Permission’ (WP) and ‘Read’ rights. There are two flavors of authorization rank diagram namely, asset-centric and stakeholder-centric [109]. Figure 3.23 is a stakeholder-oriented ARD. Assets are shown with their rankings. The authorization right of the stakeholder to a particular asset is written on the connector. In this diagram, it is seen that the Stakeholder S_1 has a right to ‘Write with Permission’ on asset A_4. The permission will be granted by S_2. S_1 can read A_1 and A_2, and write on A_3. Table 3.16 is the template for the diagram.

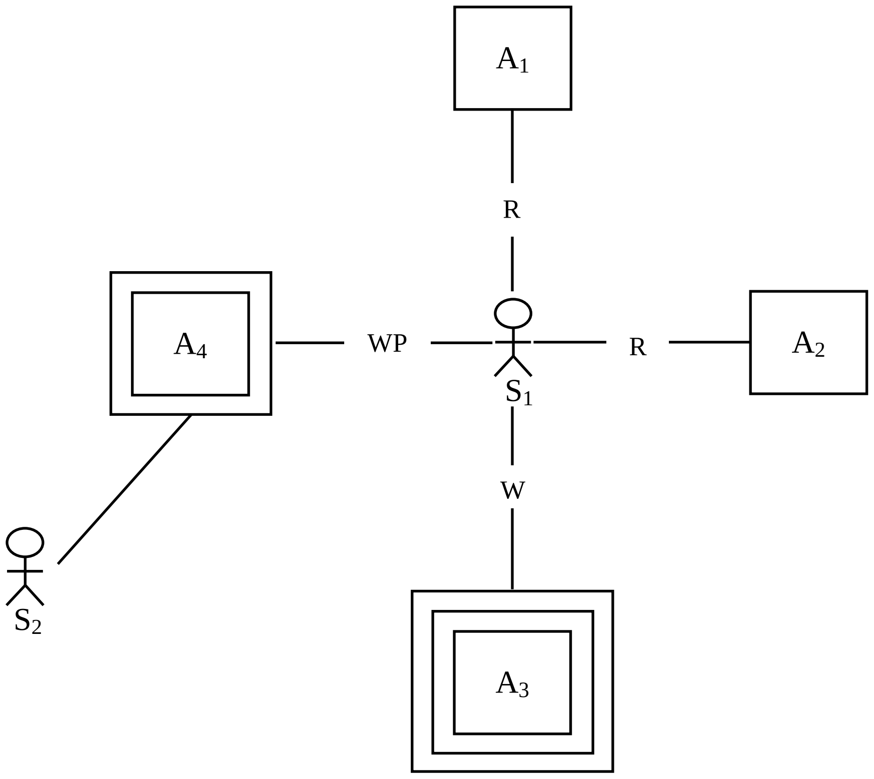


Figure 3.23: Stakeholder-oriented ARD

Table 3.16: Stakeholder-oriented ARD Template

Stakeholder: S_1			
Assets	**Security Rank**	**Authorization Rights**	**Permitting Stakeholders**
A_1	Low	R	
A_2	Low	R	
A_3	High	W	
A_4	Medium	WP	S_2

To avoid cluttering, assets with same authorization rights and same security rating can be clubbed. Figure 3.24 is equivalent to Figure 3.23 but clubs similar assets.

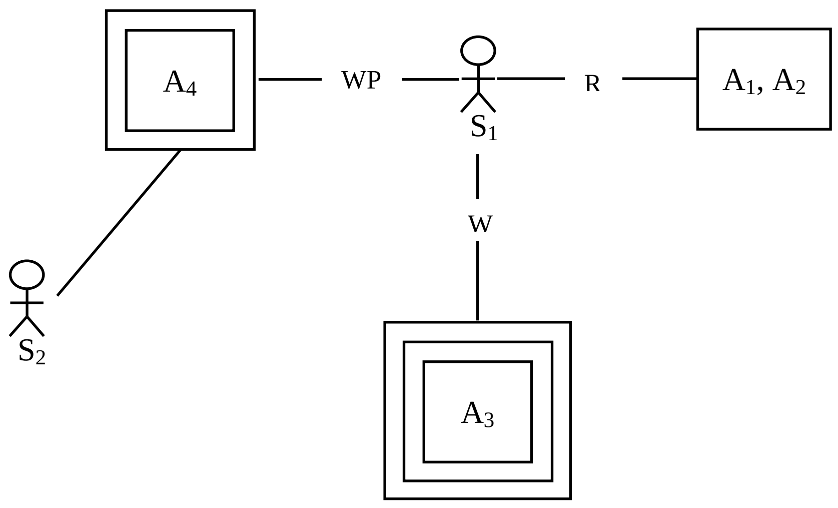


Figure 3.24: Stakeholder-oriented ARD with clubbed assets

Figure 3.25 is an Asset-oriented Authorization-Security Diagram where an asset is taken at the centre with its security rating and connected to all of its related stakeholders. The authorization right possessed by a particular stakeholder on the asset is written on the respective connector. In this diagram, it is seen that stakeholders S_1, S_2 and S_3 have Read, Write and Write with Permission rights on asset A_1 respectively. Table 3.17 is the template for this ARD.

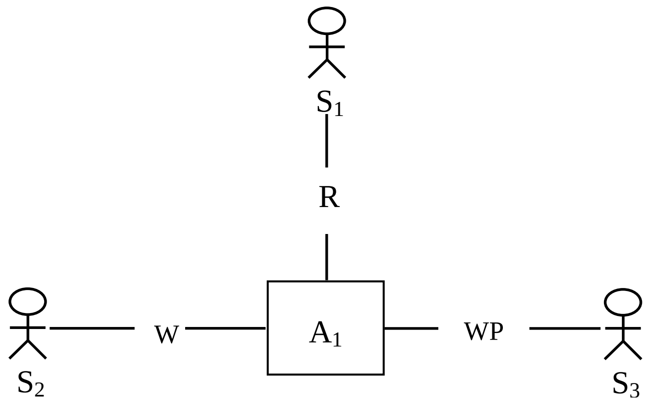


Figure 3.25: Asset-oriented ARD

Table 3.17: Template For Asset-oriented ARD Template

Asset A_1	
Stakeholder	**Authorization Rights**
S_1	R
S_2	W
S_3	WP

3.8 Development of the Tool

As a part of this research work a tool was developed to facilitate the elicitation, mapping, ranking and design processes of SecREAD methodology. The tool has been developed using PHP as front-end and MySQL as the back-end. The entities of the software to be developed, as identified from the stories, are entered through the interface and stored in three tables, one each for asset, functionality and stakeholder by the core group. The assets and functionalities are mapped by check boxes to obtain relevance matrix X. In the database, the checked values are stored as '1' and unchecked as '0' (zero). These denote relevance and irrelevance respectively. Same is followed while mapping functionalities and stakeholders in developing matrix Y. The tool then performs the cross-product of X and Y to produce matrix Z. Matrix W is formed by mapping stakeholders and parameters. The stakeholders can now login the system and furnish the ranks. The personalized rank sheets for every stakeholder are produced by the tool. Consolidating all these sheets, the three-dimensional matrix R is developed. The tool then performs all the computations, as described in Section 3.5 to produce matrices T, U, V and Q are developed. The information is passed on to the drawing module where the diagrams are generated.

3.9 Summary

SecREAD methodology takes into account the views of all the stakeholders with varied understandings. Moreover, the requirements and design phases are coupled tightly. In this methodology, both the assets and the functionalities are ranked. The concept of ranking is used to some extent in prior methodologies. However, in these methodologies ranking process does not involve all stakeholder types explicitly. The notion of relevance is unique to all prior methodologies. The proposed methodology is diagram-oriented as diagrammatic expression is more suitable than the text. It is easier to comprehend and has less chances of ambiguity. While developing these diagrams, the mindset had been to tinker as less as possible with the conventional modeling design (UML). The diagrams proposed in this methodology are evolving in nature that stem from initial narration of simple stories to SCDs, then to CSCDs, FRDs and finally CRDs as information keeps on adding. Use of very less but orthogonal structures renders these diagrams simplicity in drawing and understanding without affecting lucidity. FRD happens to be the only diagram of its kind that illustrates single-handedly, for functionality, such a large number of aspects including security level, stakeholders involved and assets involved along with their criticalness. CRD summarizes the complete system. All the diagrams are coupled with templates to provide an unambiguous view. SecREAD combines all the best practices considered by prior researchers to achieve a better software product like iteration in requirement elicitation, involvement of client and stakeholders in the development of software which increases the probability of their satisfaction. The methodology can be extended by adding more parameters. All in all, the hallmark of the methodology lies in active involvement of stakeholders, ranking by only relevant stakeholders and empirical analysis, which is rare in this domain of research. SecREAD can cater to software of varied domains, as will be demonstrated through its application on two case studies in next two chapters.

Chapter 4

Case Study: Internet Banking

In the previous chapter SecREAD methodology is proposed to address the security aspects in the development process of the software. This chapter illustrates the application of the SecREAD methodology and its validation. The most commonly used Internet Banking (INB) application is chosen as a case study because it is highly security intensive and involves financial transactions.

It is evident that in the modern era, cyber space contains a large number of applications where security is a major concern and security breach in these software systems can lead to catastrophic situations. Security requirements have been augmented as these systems ought to be safeguarded from cyber attacks. Earlier, these attacks were not so severe but since more and more services are becoming online and ubiquitous, present systems have become more vulnerable, forcing research community to consider security imbibitions in the developed software systems right at the beginning of the development process.

An INB system provides customers complete control over almost all banking demands online. It also caters to corporate customers or non-personal accounts. An INB system offers convenience in accessing banking services anytime at any place. INB system is accessible through computer, mobile phones or other hand-held computing devices. The INB service can also be called Online banking, e-banking or Virtual banking. Online banking was introduced in various parts of the world in early 1980s. In New York, USA the four banks, Citibank, Chase Bank, Chemical Bank and Manufacturers Hanover were the forerunners. In the United Kingdom, Bank of Scotland started this service in 1983. In 1984, online banking was introduced in France using terminals called Minitels. In 1995, Wells Fargo Bank of the USA added account services to its website. Presidential Bank opened accounts over the internet.

In earlier days the access to INB systems was limited and cyber attacks were also not wide spread. However, due to tremendous development in the computer industry as a whole the cyber attacks have also increased in variety, severity and volume. It has forced the banking industry to relook the security aspects and the research community to rethink the software development process to satisfy not only the banking domain but the end user customer as well.

Pervasiveness of the modern INB systems is the paramount need of the day. This property adds to quality of service and convenience of the varied users at the same time poses various security threats. Vulnerabilities present in the software can lead to huge financial loss to the customers as well as to the banks. There are numerous examples where security vulnerabilities in the systems have been exploited and have led to huge financial losses - attacks on South Korean Banks using malware ‘DarkSeoul’ in 2013, data breach in J P Morgan bank in 2014 and the SWIFT Hack affected several countries in 2016. This was another motivation to choose the INB application as a case study.

INB has several functionalities like transfer of money within the country as well as foreign countries, payment of bills, cheque clearance, issuing demand drafts, printing account statements, creating and breaking fixed deposits (FDs), and various levels of Login and authentication mechanisms for users as per their authorization rights and functionalities they wish to access etc. All of these need security at some level or the other. Authentication of users is of utmost importance and their authorization rights ought to be defined explicitly. Since a bank is answerable to its customers as a custodian of their finances, any leniency in security is unacceptable. This motivated us to take the INB as a case study to demonstrate the application and effectiveness of the methodology. In the subsequent subsections phase-wise application of the methodology on the case study is considered.

4.1 Identification

As discussed in Chapter 3, the stakeholders for any software system to be developed are categorized as members of the development team, client, users and the domain experts. In case of the INB system, stakeholders from the client side or the bank are the INB officer or the administrator, an important functionary *i.e.* Rule Authorizer and Branch staff that caters to the customers. Users are the customers of the bank who may be both in-house customers and drop-in customers. The in-house customers have their accounts in the bank while the drop-in customers do not hold account in the bank but use the services of the bank like creating DD. The stakeholders also include those to whom payment of bills is made or to whom money is transferred. The stakeholders falling in the above broad categories will be identified after Identification and Refinement phases.

Identification and Refinement phases run in spiral. To gather information during identification phase the notions of ‘Story’ and Story Conversion Diagrams (SCDs) have been introduced in the previous chapter as it is an easier method to collect requisite information more correctly and accurately. The same approach is used in this case study. As it is not possible to list all the stories in the thesis for INB system, some of the stories are listed to demonstrate this phase and the sets S, F, and A for stakeholders, functionality and asset respectively, are generated.

This phase consists of four quadrants as shown in Figure 3.1 and on each story the activity mentioned in each quadrant is to be performed strictly. In the first quadrant, requirements are elicited from the core group either directly or in the form of stories. As per the second quadrant, the stakeholders, functionalities and assets are identified and put into stakeholder set S, functionality set F and asset set A respectively. In the third quadrant, associations among the entities are identified and corresponding SCD is developed. Lastly, in the fourth quadrant, the refinement process is performed that has already been discussed at length in Section 3.3. The spiral continues till the phase is over or entities are identified exhaustively.

Table 4.1 contains some of the stories obtained from various stakeholders. With every story its type and identified entities are mentioned. The sets S, F and A are created and updated incrementally whenever a new entity is encountered in a newly elicited story.

Table 4.1: Identification Phase

S. No.	Stories	Type of Story	Entities Identified from the Story			Set of Entities obtained from the story	Incremented Set
			Stake-Holders	Functionalities	Assets		

1	Account Holder can print statement of his/her account.	Multi-Instance Story	Account Holder	Print Statement	Statement, Account	S = {Account Holder}
						F = {Print Statement}
						A = {Statement, Account}
							S = {Account Holder}
							F = {Print Statement}
							A = {Statement, Account}

2	Generate statement for any specified period	Two-Entity Story	(none)	Print Statement	Period	S = {ϕ}
						F = {Print Statement}
						A = {Period, statement}
							S = {Account Holder}
							F = {Print Statement}
							A = {Statement, Account, Period}

3	Account holder can login using username and password	Multi-Instance Story	Account Holder	Login	Username, Password	S = {Account Holder}
						F = {Login}
						A = {Username, Password}
							S = {Account Holder}
							F = {Print Statement, Login}
							A = {Statement, Account, Period, Username, Password}

4	SWIFT code and destination address are required for funds transfer	Multi-Instance Story	(none)	Funds Transfer	SWIFT code, Destination Address	S = {ϕ}
						F = {Funds Transfer}
						A = {SWIFT code, Destination Address}
							S = {Account Holder}
							F = {Print Statement, Login, Funds Transfer}
							A = {Statement, Account, Period, Username, Password, Swift Code, Destination Address}

5	Branch staff can also transfer amount on behalf of the customer	Multi-Instance Story	Branch Staff, Customer	Money Transfer	Amount	S = {Branch Staff, Customer}
						F = {Money Transfer}
						A = {Amount}
							S = {Account Holder, Branch Staff, Customer}
							F = {Print Statement, Login, Funds Transfer, Money Transfer}
							A = {Statement, Account, Period, Username, Password, Swift Code, Destination Address, Amount}

6	Customer may choose one account to print statement	Multi-Instance Story	Customer	Print Statement	Statement, Account	S = {Customer}
						F = {Print Statement}
						A = {Statement, Account No.}
							S = {Account Holder, Branch Staff, Customer}
							F = {Print Statement, Login, Funds Transfer, Money Transfer}
							A = {Statement, Account, Period, Username, Password, Swift Code, Destination Address, Amount}

7	Demand Draft (DD) Issue facility may be provided online	One-Entity Story		DD Issue		S = {ϕ}
						F = {DD Issue}
						A = {ϕ}
							S = {Account Holder, Branch Staff, Customer}
							F = {Print Statement, Login, Funds Transfer, Money Transfer, DD Issue}
							A = {Statement, Account, Period, Username, Password, SWIFT Code, Destination Address, Amount}

8	An account should always contain required amount for bill payment	Multi-Instance Story		Bill Payment	Account, Amount	S = {ϕ}
						F = {Bill Payment}
						A = {Account, Amount}
							S = {Account Holder, Branch Staff, Customer}
							F = {Print Statement, Login, Funds Transfer, Money Transfer, DD Issue, Bill Payment}
							A = {Statement, Account, Period, Username, Password, SWIFT Code, Destination Address, Amount}

9	Money can be transferred only by the account holder	Three-Entity Story	Account Holder	Money Transfer	Money	S ={Account Holder}
						F = {Money transfer}
						A = {Money}
							S = {Account Holder, Branch Staff, Customer}
							F = {Print Statement, Login, Funds Transfer, Money Transfer, DD Issue}
							A = {Statement, Period, Account, Money, Username, Password, SWIFT Code, Destination Address, Amount}

10	Confirm transfer of funds to Payee	Two-Entity Story	Payee	Funds Transfer		S = {Payee }
						F = {Funds Transfer}
						A = {ϕ}
							S = {Account Holder, Branch Staff, Customer, Payee}
							F = {Print Statement, Login, Funds Transfer, Money Transfer, DD Issue}
							A = {Statement, Period, Account, Money, Username, Password, SWIFT Code, Destination Address, Amount}

Figures 4.1 to 4.10 denote the SCDs for the stories 1 to 10 in Table 4.1 respectively.

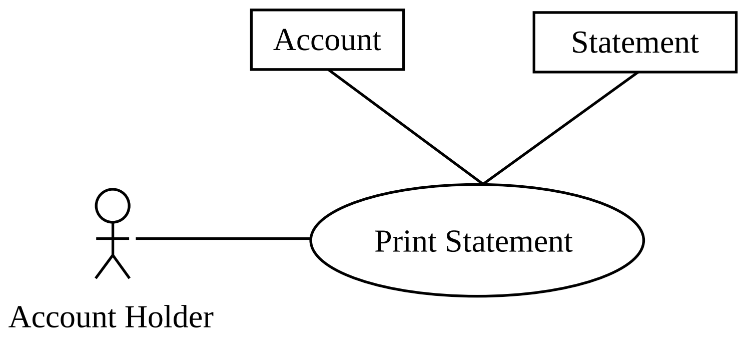


Figure 4.1: SCD for Story 1

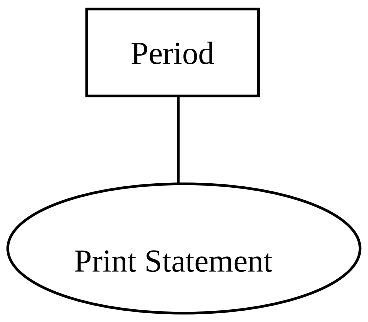


Figure 4.2: SCD for Story 2

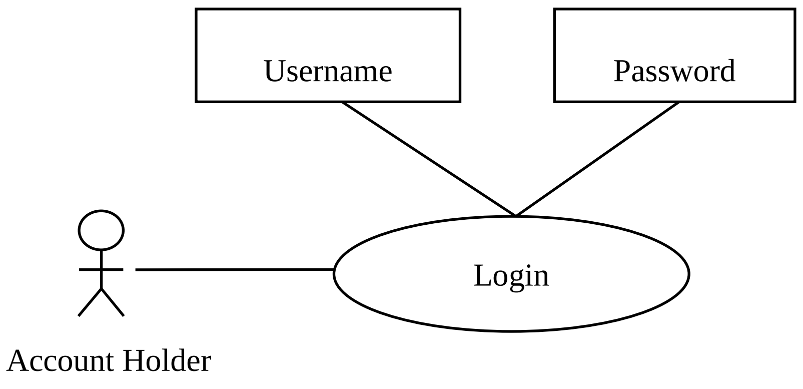


Figure 4.3: SCD for Story 3

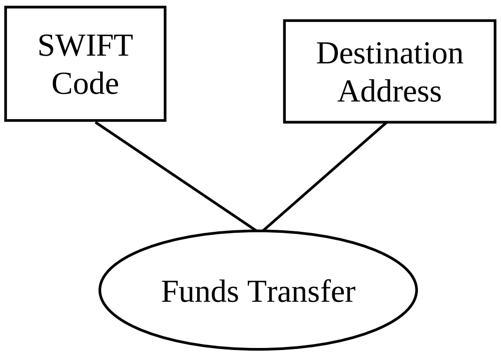


Figure 4.4: SCD for Story 4

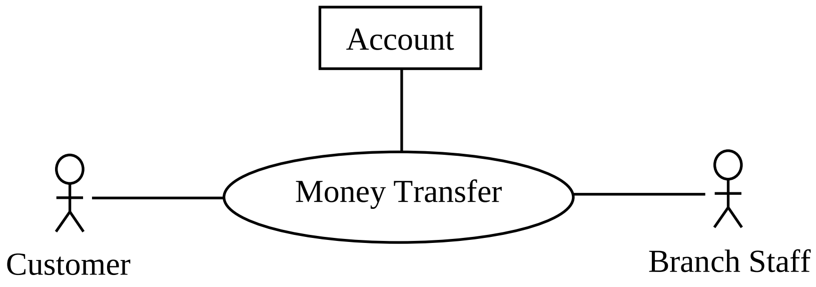


Figure 4.5: SCD for Story 5

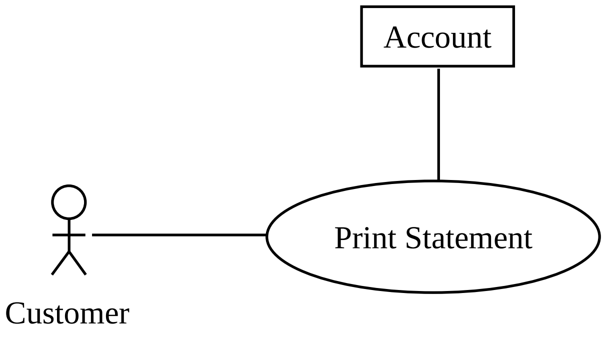


Figure 4.6: SCD for Story 6

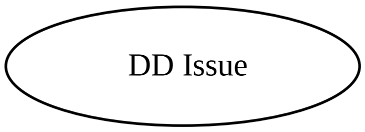


Figure 4.7: SCD for Story 7

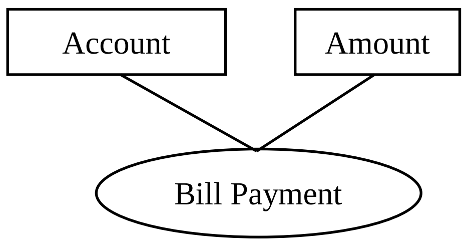


Figure 4.8: SCD for Story 8

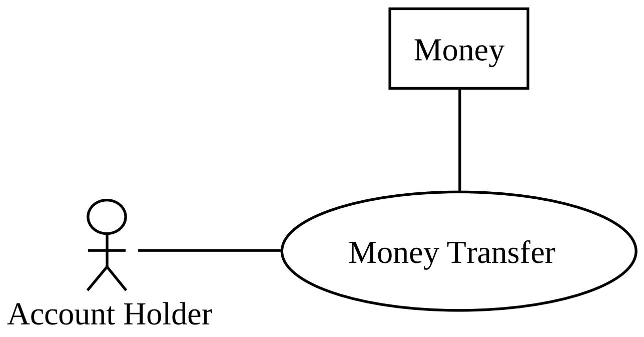


Figure 4.9: SCD for Story 9

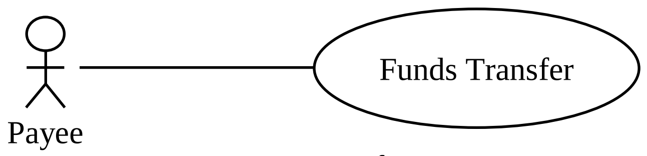


Figure 4.10: SCD for Story 10

4.2 Refinement

Once the stories are elicited and entities are identified, refinement procedure is conducted. Refinement is performed in the last quadrant of each spiral of this phase. The refinement process consists of redundancy removal, aggregation and decomposition activities. In this phase, the information gathered in the identification phase is refined to ensure correctness, non-ambiguity, efficiency, quality etc. in a software system being developed. Refinement is explained at length in Section 3.3. It makes use of SCDs for redundancy removal, aggregation and decomposition as per need.

Sets S, F and A are obtained based on the primary stories in Table 4.1 and are given here as Eq. (4.1), Eq. (4.2) and Eq. (4.3) respectively.

S = {Account Holder, Branch Staff, Customer, Payee} (4.1)

F = {Print Statement, Login, Funds Transfer, Money Transfer, DD Issue} (4.2)

A= {Statement, Period, Account, Money, Username, Password,
SWIFT Code, Destination Address, Amount} (4.3)

4.2.1 Redundancy Removal

In set S given by Eq. (4.1) two entities 'Customer' and 'Account Holder' represent same entity so a common term is selected to avoid ambiguity in developing the software system. Here, customer is a more general and appropriate term as already stated in Section 4.1, it is not necessary that every person who uses bank services is an account holder. Similarly, in functionality set F, given by Eq. (4.2), 'Money Transfer' and 'Funds Transfer' are found to be synonymous and hence decision is to be made to keep only one. Money Transfer is discarded while Funds Transfer is accepted.

It is the responsibility of the development team to carefully scan all the entity sets to remove the redundancy that may be repetitive. Algorithm 3.1 Redundancy_removal has been followed to scan the different entity sets. For instance the elements of stakeholder set S are scanned and when two similar stakeholders 'Payee' and 'Beneficiary' are encountered then the decision is made to discard 'Payee' and keep 'Beneficiary' that encompasses all parties to whom any kind of funds transfer is made. Then the sets are updated. In set A given by Eq. (4.3) it is found that two terms 'Money' and 'Amount' convey same meaning. Here also, the development team has to decide which term is more logical. The term 'Amount' is accepted. Another instance from the asset set is the term 'Account' that seems to be correct but it is ambiguous from the point of view of software system as it carries no meaning. The system identifies an account uniquely by its number. Therefore, it is replaced by the term 'Account No.' After ever redundancy removal the particular set is updated

Figures 4.11 to 4.17 are the SCDs developed after removing redundancies. These SCDs have been derived easily and quickly by scanning the previously developed SCDs *i.e.* Figures 4.1 to 4.10.

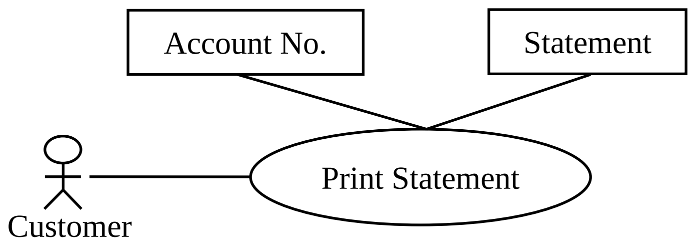


Figure 4.11: SCD for story 1 after refinement

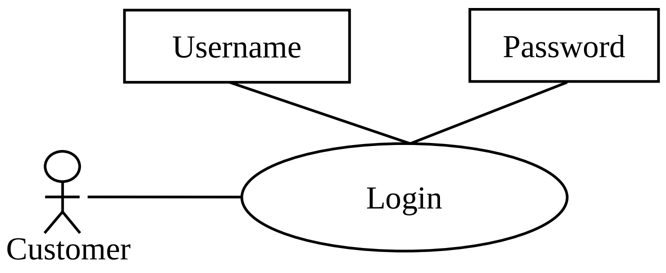


Figure 4.12: SCD for story 3 after refinement

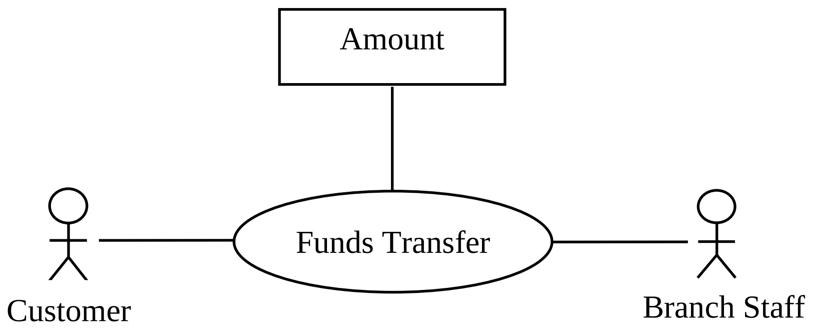


Figure 4.13: SCD for story 5 after refinement

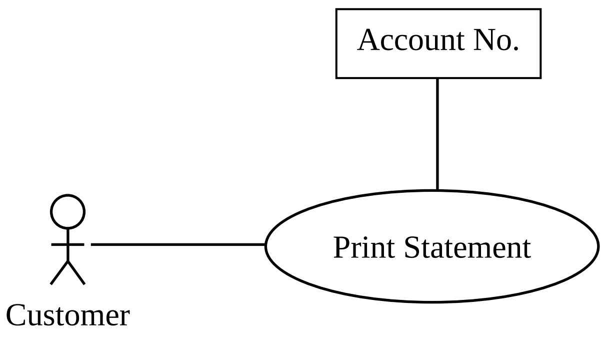


Figure 4.14: SCD for story 6 after redundancy removal

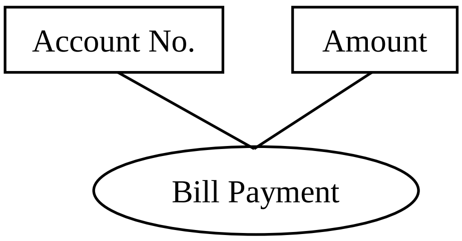


Figure 4.15: SCD for story 8 after refinement

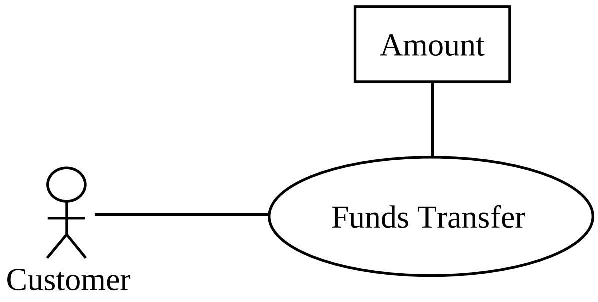


Figure 4.16: SCD for story 9 after redundancy removal

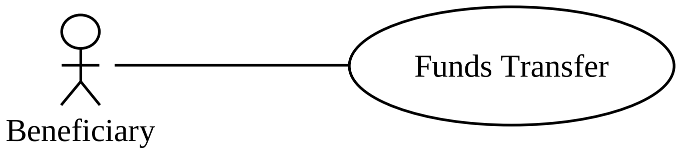


Figure 4.17: SCD for story 10 after refinement

After the redundancy removal process, the sets obtained are as follows:

S = {Customer, Branch Staff, Beneficiary} (4.4)

F = {Print Statement, Login, Funds Transfer, DD Issue} (4.5)

A = {Statement, Period, Account No., Username, Password, SWIFT Code, Destination Address, Amount} (4.6)

4.2.2 Aggregation

Aggregation is an activity to combine stories conveying similar information or containing commonalities (like related to same functionality). This similarity or commonality can easily be consolidated and represented through an aggregation of such SCDs and generating an ‘Aggregated SCD’. It is developed by combining the knowledge obtained through preliminary SCDs and redundancy-removed SCDs. If we look carefully it is found that functionality ‘Funds Transfer’ is common in Figures 4.4, 4.10, 4.13, 4.16 and 4.17. An aggregated SCD (Figure 4.18) is developed that summarizes associations pertaining to ‘Funds Transfer’ functionality.

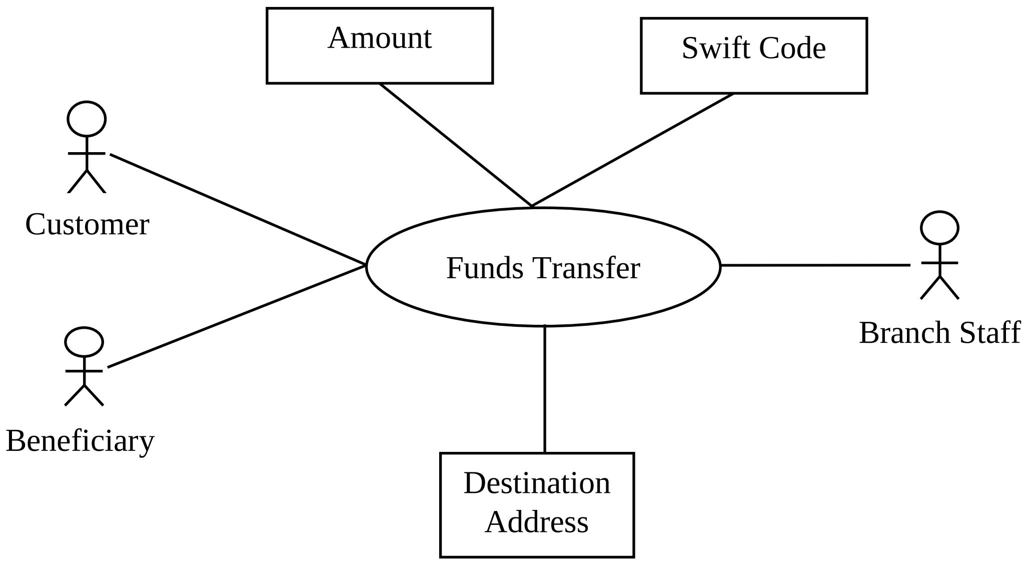


Figure 4.18: Aggregated SCD for Funds Transfer

Similarly, it is seen that the SCDs for stories 1, 2 and 6 contain the same functionality ‘Print Statement’. Story 1 is refined in Figure 4.11 and story 6 in Figure 4.16 (after redundancy removal). So by combining the Figures 4.2, 411 and 4.16 an aggregated SCD is obtained as shown in Figure 4.19.

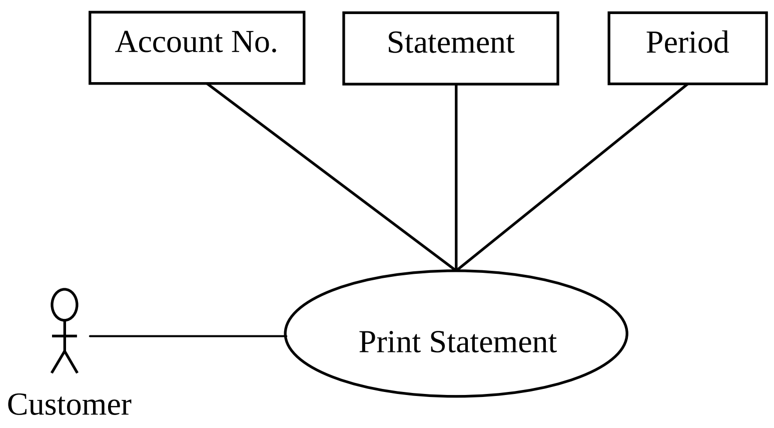


Figure 4.19: Aggregated SCD for Print Statement

4.2.3 Decomposition

Decomposition is an important phase to obtain more precision in requirements. It provides more granularities in entities by decomposing them into constituents or sub-parts. For example, destination address in set A shown by Eq. (4.6), is to be decomposed into Destination Country, Destination Bank Name and Destination Bank Address. These details are required explicitly for funds transfer in different cases. Figure 4.20 is the new SCD for 'Funds Transfer' functionality with decomposed assets. It is derived from Figure 4.18. Similarly, Funds Transfer functionality can also be broken into three types *i.e.* intra-bank, inter-bank and international. SCDs for these cases will be developed and presented subsequently.

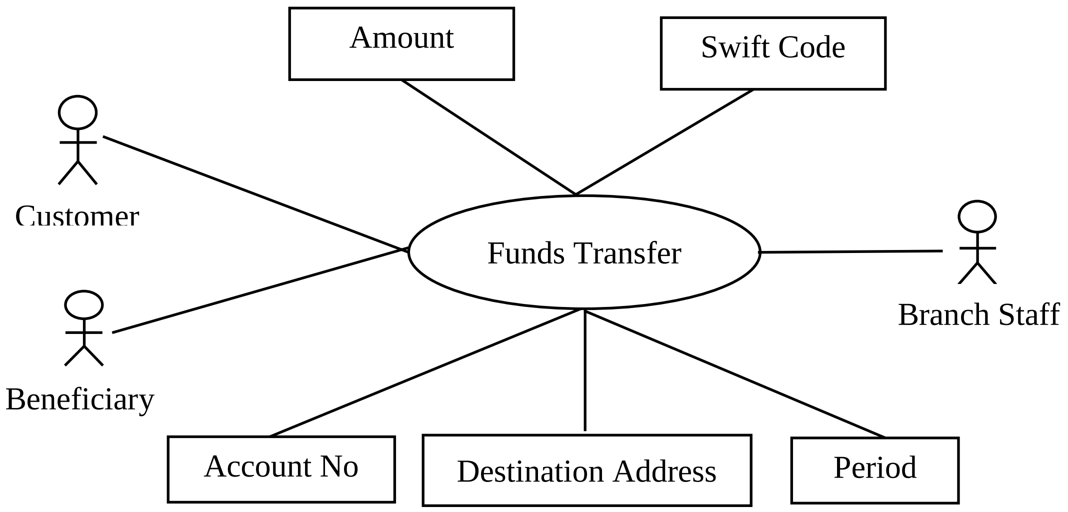


Figure 4.20: Decomposed SCD for Funds Transfer

Evidently, the notion of SCD plays a pivotal role in the refinement because these provide a graphical view of stories and it is always easier to identify redundancy. This notion also helps in developing new refined SCDs quickly.

4.2.4 Culmination of Spiral

When the identification and refinement activities were applied on the complete INB system, more entities were obtained incrementally. Final entity sets are obtained after the spiral ends. Based on our survey the functionality, stakeholder and asset sets were obtained. These have been enlisted below along with their brief description or role in the system. Detailed description of some significant entities along with the rationale of their inclusion in the system follows the sets. Sincere efforts have been made to identify the entities exhaustively. However, some may have still left out unintentionally.

It has been underlined in the previous chapter that the comprehensive listing of stakeholders is crucial for the success of the proposed SecREAD methodology. The basic stakeholders are the developers, the domain experts, the client and the users. A client representative is made a part of the development team who advises and provides clarifications to the developers whenever required. Most importantly he/she conveys the client’s intensions and aspirations with the system to the developers. There can be several sub-groups within the client and the user community. The iterative nature of the identification and refinement phase ensures that all such sub-groups are discovered. There are a large number of stakeholders in the banking industry. Apart from developers and domain experts, the stakeholders within the client side are enlisted in Table 4.2. The different user groups are enlisted in Table 4.3.

Table 4.2: Cilent-side Stakeholders

Stakeholders	**Brief Description/Role**
INB officer	Issue kit, re-issue username and password
Rule Authorizer	Activate kit, approve the request for username and password
Branch Staff	The front-end staff members of the bank that cater to the needs of the customers
Client Representative	A person that advices the development team

Table 4.3: User-Group Stakeholders

Stakeholders	Brief Description/Role
Customer	Customer of the bank
Biller	Person or organization that receives the payment of the bill like a telecom company or electricity department
Beneficiary	Person or organization to whom money is transferred

INB system performs several functions. The different functionalities that the system offers are enlisted in Table 4.4.

Table 4.4: Functionality Set F

Functionality	Brief Description/Role
Issue INB Kit	A kit containing user name and password is issued by Bank to the customer to use INB facility
Login	Customer has to login by giving the user name and password provided in kit to use INB facility. Later these credentials can be changed.
View Account	After logging-in Customer can view all details of his all accounts of the Bank linked in INB
Print Statement	Customer can take printout of the statement of his/her account(s)
Intra-Bank Funds Transfer	Customer can transfer funds between accounts of the same bank
Inter-Bank Funds Transfer	Customer can transfer funds between accounts of different banks.
International Funds Transfer	Customer can transfer funds between accounts internationally.
DD Issue	Customer can make a request to issue a DD and may collect the DD from branch or through post.
Bill Payment	Customer pay the bill to billers listed in the INB system on a scheduled date
FD/RD Creation	Customer can make FD or RD online and break the same prematurely.
Cheque book issue	Customer can ask to issue a new cheque book. Cheque book is delivered to his/her address
Cheque Payment Stop	Stop cheque payment by the customer

Similarly, there are different groups of assets used by different functionalities and stakeholders though they may cut across the boundaries. Assets can roughly be grouped into customer details, security credentials & other assets for security, account related assets, functionality-specific and miscellaneous assets enlisted in Tables 4.5, 4.6, 4.7 and 4.8 respectively.

Table 4.5: Assets Pertaining to Customer

Asset	Brief Description
Customer Name	Name of the customer
Address	Address of the customer
Mobile No.	Mobile number of the customer
e-mail	e-mail address of the customer
PAN	Permanent Account Number of the customer
Aadhar No.	It is a 12-digit unique identification number for Indian citizens issued by government of India that is mandatory to be linked with the bank account.
Kit no.	No. of the kit issued to the customer
CIF no.	It stands for Customer Information No. which is unique for every customer with respect to one bank. It is used in identifying how many accounts he/she holds, in which branches, the type of accounts, the balance in the accounts, his/her deposits like FDs and RDs. These details are useful in providing information to government, income tax department or other investigation agencies, if required for.

Table 4.6: Assets for Security and as Credentials

Asset	Brief Description
Username	Username of the customer to login into INB
Password	Password of the customer to login into INB
User Profile Password	A password provided to customer for enhancing security in money transfer – to add a payee/biller
OTP	One Time Password for authentication during online transaction, it is sent on the customer's mobile phone
INB Officer Username	Username of INB officer
INB Officer Password	Password of the INB Officer
Rule Authorizer Username	Username of Rule Authorizer
Rule Authorizer Password	Password of the Rule Authorizer
Fingerprint Pattern	Fingerprint Pattern of the customer/staff member
Voice Sample	Voice Sample of the customer

Table 4.7: Account Related Assets

Asset	Brief Description
Account No.	Account no. of the customer
Account Type	Type of account like current or savings
Account Balance	Balance amount in the account
Statement	Statement of account
Dates- from and to	Specifies the period following between these dates, for which the statement is to be printed
Month	Month of which statement is to be printed

Table 4.8: Functionality-specific and Miscellaneous Assets

Asset	Brief Description
Branch Name	Name of the branch, usually its address and/or area
IFSC	It stands for Indian Financial System Code that is a alphanumeric code of 11 characters that uniquely identifies each bank branch
SWIFT Code	It stands for Society for Worldwide Inter-bank Financial Telecommunication used when the transfer between two banks happens internationally.
DD No.	Number of the DD issued
DD Amount	Amount of DD
DD payable at	Place where DD is payable at
DD issued at	Place where DD is issued at
Delivery mode	Mode of delivery of DD *i.e.* by post or collected in person
Biller Name	Name of the party to which the bill is to be paid
Biller ID	ID of the party to which the bill is to be paid
Scheduled Date	A later date specified at which the money is to be paid to the biller or transferred to the payee
FD/RD No.	No. of FD or RD
FD/RD Amount	Amount of FD or RD
Maturity Date	Maturity date of FD
Nominee Details	Name of the nominee of the FD and account including his/her relationship with the customer
Transfer amount	Amount to be transferred
Max. Limit	Maximum limit of money that can be transferred to a payee at a time
Destination Bank Details	Name and address of the of the beneficiary bank
Beneficiary Name	Name of the beneficiary
Cheque no.	The no. of the cheque. It is also used to specify the start and end number of cheques for stopping payments
Standing Instructions	Standing instructions given to the bank by the customer to perform a task automatically at future date(s)

As more global financial activity becomes digitally-based, many banks are utilizing new technologies to develop next-generation identification controls to combat fraud, make transactions more secure, and enhance the customer experience. Several security measures are used like username-password based (something you know), smart card based (something you have) and biometric based (something you are). Additional security layer is provided by One Time Password (OTP) and user profile password. The schemes can be categorized as low, medium and high level of security measures. Their usage varies with functionalities, domains and client's aspirations.

The biometric mechanisms considered in the case study are fingerprints and voice control used for authentication of users. The SecREAD methodology adjudges the level of security for entities and based on those rankings appropriate security measures are suggested to the developers and client to select.

Funds Transfer is an important feature of INB system. It is of three types:

- Intra-bank money transfer allows transfer of money between accounts of the same bank.
- Inter-bank money transfer allows transfer of money between accounts of different banks but within the country.
- International money transfer allows transfer of money between accounts of the different banks/branches but internationally.

For the inter-bank money transfer in India, two applications viz. National Electronic Funds Transfer (NEFT) and Real Time Gross Settlement (RTGS) are used. NEFT has a limit of Rs. 2,00,000 while RTGS is boundless. The amount desired to be transferred is represented by the asset 'Money to be transferred'. The 'Payee' to which money is to be transferred ought to be added first. A limit of maximum amount to be paid, at a time, is set for each payee. The asset 'Max Limit' signifies this limit. The 'IFSC' is used to identify any bank branch in India. It contains 11 alphanumeric characters where, first four characters represent bank code, the 5th character is 0 reserved for future use and last six characters are branch code. All the bank branches within the country are assigned an IFSC by the Reserve Bank of India. For example, IFSC for State Bank of India, Jaipur City, Sanganeri Gate branch is SBIN0000656.

'SWIFT Code' is required for International Money Transfer. It is a unique identification code assigned to a specific bank to perform money transfer between banks internationally. The code is a combination of 8 or 11 alphanumeric characters where, the first four characters represent bank code. (Letters only, *i.e.* AAAA), next two characters represent country code. (Letters only, *i.e.* BB), next, two characters represent location code (letters and digits, *i.e.* 1C). Last three characters are optional that represents branch code (letters and digits (DDD). For example, SWIFT code for the above mentioned branch is SBININBB154.

As we are aware that due to modern cyber attacks only username and password scheme does not suffice alone so an asset like One Time password (OTP) is used especially, for all kinds of money transfer. One Time Password (OTP) is a password that is valid for only one login session or transaction. It is generated randomly and it further enhances security associated with static password scheme as it is not susceptible to replay attack [86] which is the resending of a message that has been intercepted by an intruder. While using critical functionalities like funds transfer, OTP is sent on the registered mobile number of the customer and has to be keyed in as an additional security feature. 'User Profile Password' is another security feature provided to the customer which is required at the time of updation of personal details and addition of payee or biller. Cheque book is issued to the customer on his/her request usually, using 'Cheque Book Issue' functionality when all the leaves are over. It has the 'Customer Name' and 'Account no.' written over it. If a customer wishes to stop the payment of a cheque, it can be done through 'Cheque Payment Stop' functionality. In case of multiple cheques, the start and end nos. of the cheques are entered.

The 'Issue Kit' functionality is invoked once a customer opens an account with the bank. The INB kit is issued by the INB officer for the customer at the time of account-opening. It is collected by the customer through the branch in which he/she has the account. At the time of account opening, the customer's Aadhar Number and PAN are linked with the account. The kit contains the initial Username and Password of the customer which can be changed. In 'Intra-bank Funds Transfer' functionality the 'account no.' denotes the account of the customer from which the money is to be transferred while the 'Beneficiary Account No.' is the account no. in which the money is to be transferred. In inter-bank funds transfer functionality, IFSC is used to transfer funds within India while SWIFT code is used for funds transfer in a foreign country. Similarly, in foreign funds transfer, 'Destination Bank Details' are required which include the name of the bank and its country. The 'DD Issue' functionality requires the name and IFSC of the issuing branch and the branch where DD is payable at. The DD can be collected in person or via courier.

As per the SecREAD methodology flow chart shown in Figure 3.1, when the complete entity sets are obtained, they are mapped to each other in the mapping phase which is explained in the next section.

4.3 Mapping

The entities are mapped to each other for which three relevance matrices are developed for this, as already explained in Section 3.4. Asset-Functionality Matrix (X) maps assets to functionalities, Functionality-Stakeholder Matrix (Y) maps functionalities to stakeholders and Asset-Stakeholder Matrix (Z) maps assets to stakeholders. In X and Y, a cell-value '1' indicates that corresponding row and column entities are relevant to each other. This mapping is performed by the core group. The zero value indicates irrelevance. Z is obtained through the cross-product of X and Y. As already mentioned in Sub-section 3.4.2, the product values in the cells of Z are significant only to show relevance or association among assets and stakeholders, so these cells are shaded. Another matrix W is produced by the core group that maps stakeholders to parameters. It has been described in Sub-section 3.4.4. A tool has been developed in this research work, as described in Section 3.8, to facilitate the processes.

For the case study in question, the X, Y and Z matrices are shown by tables 4.9, 4.10 and 4.11 respectively. It can be seen in matrix X that the asset 'Kit No.' is mapped to 'Issue Kit' functionality and in Y matrix the same functionality 'Issue Kit' is mapped to stakeholder 'INB Officer'. Consequently, in matrix Z which is the cross product of X and Y it is seen that 'Kit No.' is relevant to INB Officer. This verifies the development of Z.

Only parts of matrices X and Z are shown here. Complete matrix X is given by Table A.2 and matrix Z is given by Table A.1, as generated by the tool. It is noteworthy, that since the tool is based on arithmetic calculations, the matrix Z (as presented in Table A.1) produced by it contains actual numeric values of the cross product that may be treated merely for denoting relevance.

Table 4.9: Asset-Functionality Relevance Matrix X

Assets	**Functionality**											
	Issue Kit	Login	View Account	Print State-ment	Intra-Bank Funds Transfer	Inter-Bank Funds transfer	Inter-national Funds Transfer	DD Issue	Bill Payment	FD / RD Creation	Cheque Book Issue	Cheque Payment Stop
Customer Name	1	0	0	0	0	0	0	0	0	0	1	0
Address	1	0	0	0	0	0	0	0	0	0	1	0
PAN	1	0	0	0	0	0	0	0	0	1	0	0
Aadhar No.	1	0	0	0	0	0	0	0	0	1	0	0
Mobile No.	1	1	0	0	1	1	1	1	1	1	0	1
e-mail	1	1	0	1	0	0	0	0	1	1	1	1
CIF no.	1	0	0	0	0	0	0	0	0	0	0	0
Kit No.	1	1	0	0	0	0	0	0	0	0	0	0
Username	1	1	1	1	1	1	0	1	1	1	1	0
Password	1	1	1	1	1	1	0	1	1	1	1	0

Table 4.10: Functionality-Stakeholder Relevance Matrix Y

Functionality	**Stakeholder**								
	INB Officer	Rule Authorizer	Branch Staff	Customer	Biller	Beneficiary	Client	Developer	Expert
Issue Kit	1	1	1	0	0	0	1	1	1
Login	0	0	0	1	0	0	1	1	1
View Account	0	0	0	1	0	0	1	1	1
Print Statement	0	0	0	1	0	0	1	1	1
Intra-Bank Funds Transfer	0	0	0	1	0	1	1	1	1
Inter-Bank Funds Transfer	0	0	0	1	0	1	1	1	1
International Funds Transfer	0	0	1	1	0	1	1	1	1
DD Issue	0	0	1	1	0	1	1	1	1
Bill Payments	0	0	0	1	1	0	1	1	1
FD/RD creation	0	0	1	1	0	0	1	1	1
Cheque Book Issue	1	1	1	1	0	0	1	1	1
Cheque Payment Stop	0	0	0	1	0	0	1	1	1

Table 4.11: Asset-Stakeholder Relevance Matrix Z

Assets	**Stakeholders**								
	INB Officer	Rule Authorizer	Branch Staff	Customer	Biller	Beneficiary	Client	Developer	Expert
Kit No.	▨	▨	▨	▨			▨	▨	▨
Customer Name	▨	▨	▨	▨			▨	▨	▨
Address	▨	▨	▨	▨			▨	▨	▨
PAN	▨	▨	▨	▨			▨	▨	▨
Aadhar No.	▨	▨	▨	▨			▨	▨	▨
Mobile No.	▨	▨	▨	▨	▨	▨	▨	▨	▨
e-mail	▨	▨	▨	▨			▨	▨	▨
CIF no.	▨	▨	▨				▨	▨	▨
Username	▨	▨	▨	▨	▨	▨	▨	▨	▨
Password	▨	▨	▨	▨	▨	▨	▨	▨	▨

Table 4.12: Stakeholder-Parameter Relevance Matrix W

Stakeholder	**Parameters**				
	Authentication	Confidentiality	Integrity	Non-repudiation	Autho-rization
INB Officer	1	1	1	1	1
Rule	1	1	1	1	0
Branch Staff	1	1	1	1	0
Customer	1	1	1	1	0
Beneficiary	0	0	1	1	0
Biller	0	0	1	1	0
Client	1	1	1	1	1
Developer	1	1	1	1	1
Domain Expert	1	1	1	1	1

The X and Y relevance matrices, both depict the association of assets and stakeholders with a functionality. CSCDs can be developed using these two matrices. CSCDs are described in Sub-section 3.4.3 and are developed using Algorithm 3.2. CSCD for 'Issue Kit' is given by Figure 4.21.

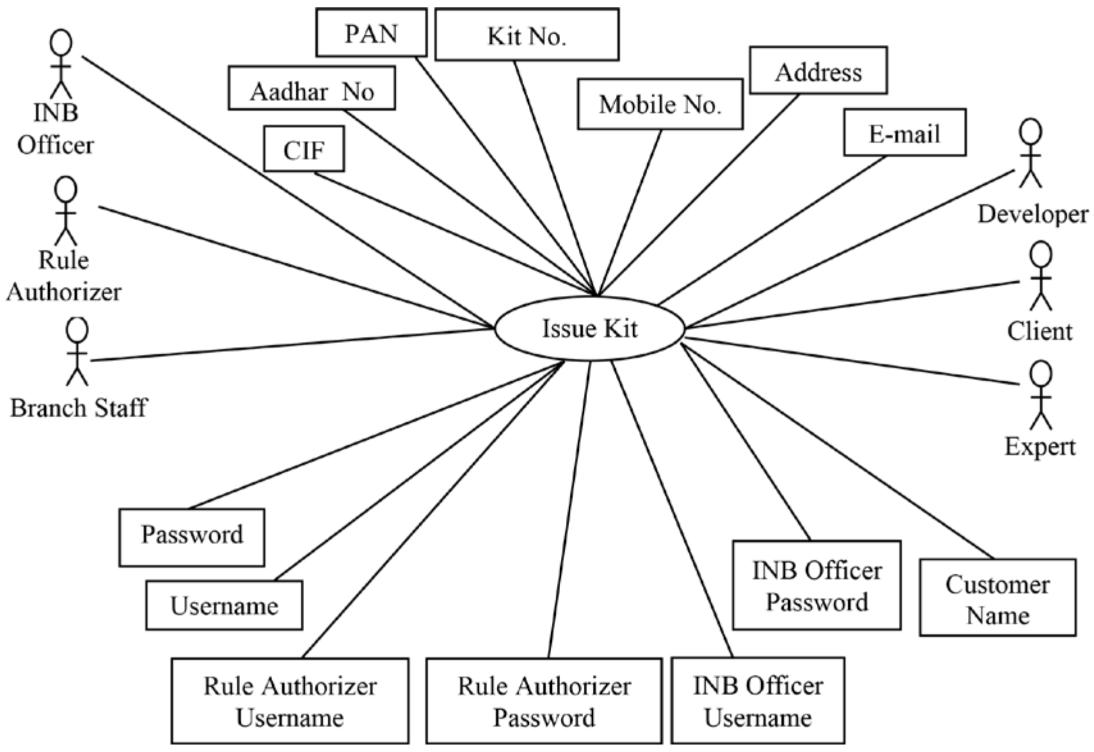


Figure 4.21: CSCD for Issue Kit functionality

4.4 Ranking and Analysis

Based on the relevance matrices, the rank matrix R is developed. Through this matrix ranks for assets are collected by different stakeholders as already explained in the previous chapter.

4.4.1 Rank Matrix

The process of ranking is conducted by stakeholders. A 3-d Rank matrix R is developed by combining the information contained in relevance matrices W and Z, as defined in Sub-section 3.4.5. The z-dimension of R is made of rank sheets for every stakeholder. A 3-d view for the INB case study is given by Figure 4.22. It shows the rank-sheet of the stakeholder 'Customer', and the ranks furnished by him for his/her relevant assets. Some of the assets shown here are Mobile No., E-mail, Username and Password. The assets are ranked over his/her relevant parameters, which as per matrix W (Table 4.12), are Confidentiality, Integrity and Non-Repudiation (denoted by Pr_2, Pr_3 and Pr_4 respectively) as described in Chapter 3. Also, in this figure, the rank sheets of other stakeholders can be seen behind each other or on the z-axis of R.

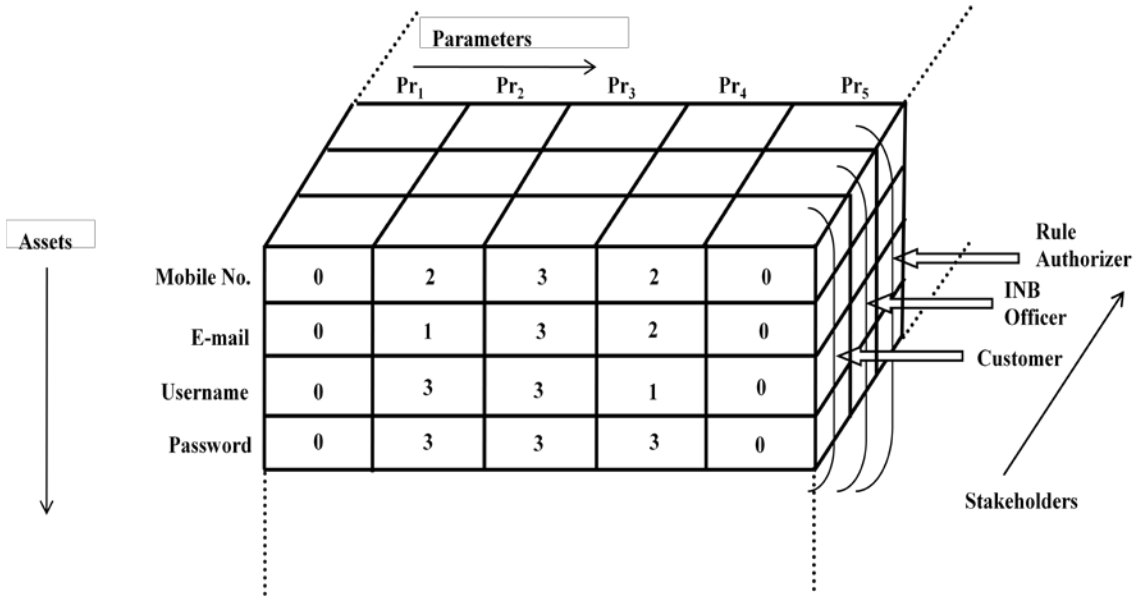


Figure 4.22: 3-d View of Rank Matrix R for INB System with Rank Sheet for Customer

4.4.2 Results and Discussion

The input data of all stakeholders *i.e.* the compilation of all rank sheets is contained in a 3-d matrix R, already defined in Section 3.6. Computations are performed on R once all the stakeholders have performed ranking of their relevant assets under their relevant parameters. Table 4.13 shows a portion of matrix R obtained by the tool which shows the ranking (in the form of 1, 2, 3 for low, medium and high respectively) given by all the stakeholders over their relevant assets under their relevant parameters. In this way matrix R represents three dimensions of assets, stakeholders and parameters. In the portion of R provided in Table 4.13 only a partial view pertaining to stakeholder 'Customer' is provided. The ranks of Password and Account No.(s) in R match with that of rank sheet of beneficiary (Figure 4.22). The complete matrix R contains the entries of all the stakeholders *i.e.* over 800 rows. A larger view of this matrix, in SQL table form, is given by Table A.3.

After performing cell computation on matrix R, matrix T is obtained. It is derived from matrix R, removing its third dimension of stakeholder. Thus it is a 2-dimensional matrix with first dimension as asset and second as parameter. In other words, it gives a consolidated rank of each asset over different parameters. A part of the matrix is given by Table 4.13 where the first column contains assets repeated five times for all five parameters listed in the second column. Assets Account Balance and Address are shown here. The third column contains the mode of all the rank values given by all relevant stakeholders for that asset. In this way matrix T provides a 2-dimensional view of assets and parameters. The larger view of this matrix is given by Table A.4.

Table 4.13: 2-d Matrix T

Asset	**Parameter**	**Mode**
Account Balance	Authentication	3
Account Balance	Authorization	3
Account Balance	Confidentiality	2
Account Balance	Integrity	3
Account Balance	Non-repudiation	3
Address	Authentication	2
Address	Authorization	2
Address	Confidentiality	1
Address	Integrity	3
Address	Non-repudiation	2

Matrices U and V are derived from computations on matrix T, as discussed in Section 3.5, which aid in determining the ranks of assets and parameters respectively. Using Algorithm 3.3, matrix U is obtained that depicts the ranks of assets. It is obtained by taking the mode of all the mode values under the different parameters for each asset in matrix T. In matrix T for INB system, given by Table 4.13, the asset Account Balance is ranked under five parameters as 3, 3, 2, 3 and 3. In matrix U, given by Table 4.14, the mode of these values is considered and the asset is given the value 3 or high rank. This verifies the creation of matrix U by the tool. Complete matrix U is given by Table A.5.

Table 4.14: Matrix U or Asset Rank

Asset	Mode	Rank
Account Balance	3	High
Address	2	Medium
CIF	3	High
Customer Name	2	Medium
Biller Name	1	Low

Matrix V is obtained from matrix T by using Algorithm 3.4 and depicts the ranks of every parameter in the system. In order to verify this matrix obtained from the tool, the complete matrix ought to be considered. Mode of all the modes values of one particular parameter in the matrix T is obtained to find the rank of that parameter in the software. Table 4.15 gives matrix V.

Table 4.15: Matrix V or Parameter Rank

Parameter	Mode	Rank
Authentication	3	High
Confidentiality	3	High
Integrity	3	High
Non-repudiation	2	Medium
Authorization	3	High

The security ranks of individual functionalities in the INB system are given by the matrix Q (Table 4.16).

Table 4.16: Matrix Q or Functionality Rank

Functionality	**Mode**	**Rank**
Issue Kit	3	High
Login	3	High
View Account	1	Low
Print Statement	1	Low
Intra-Bank Funds Transfer	2	Medium
Inter-Bank Funds Transfer	3	High
International Funds Transfer	3	High
Bill Payment	3	High
Cheque Payment Stop	1	Low
Cheque Book Issue	1	Low
DD Issue	2	Medium
FD/RD creation	3	High

4.5 Design

As per the ranks obtained in the analysis phase, designing is conducted here. Functionality Rank Diagram (FRD), Comprehensive Rank Diagram (CRD) and Authorization Rank Diagrams (ARD) are developed. These diagrams have already been discussed in Section 3.7. Each of these three diagrams for the Internet Banking are elaborated in the following sub-sections.

4.5.1 FRD and CRD

An FRD for the ‘Issue Kit’ functionality is depicted by Figure 4.23. It is an extension of the CSCD for the same functionality given by Figure 4.21 with the rank information embedded. However, the assets with the same security rank are clubbed together. Assets in three rectangles like Kit No. and Password have high rank while those within two rectangles like Address and e-mail have medium security rank. FRDs for other functionalities are given in Appendix A.

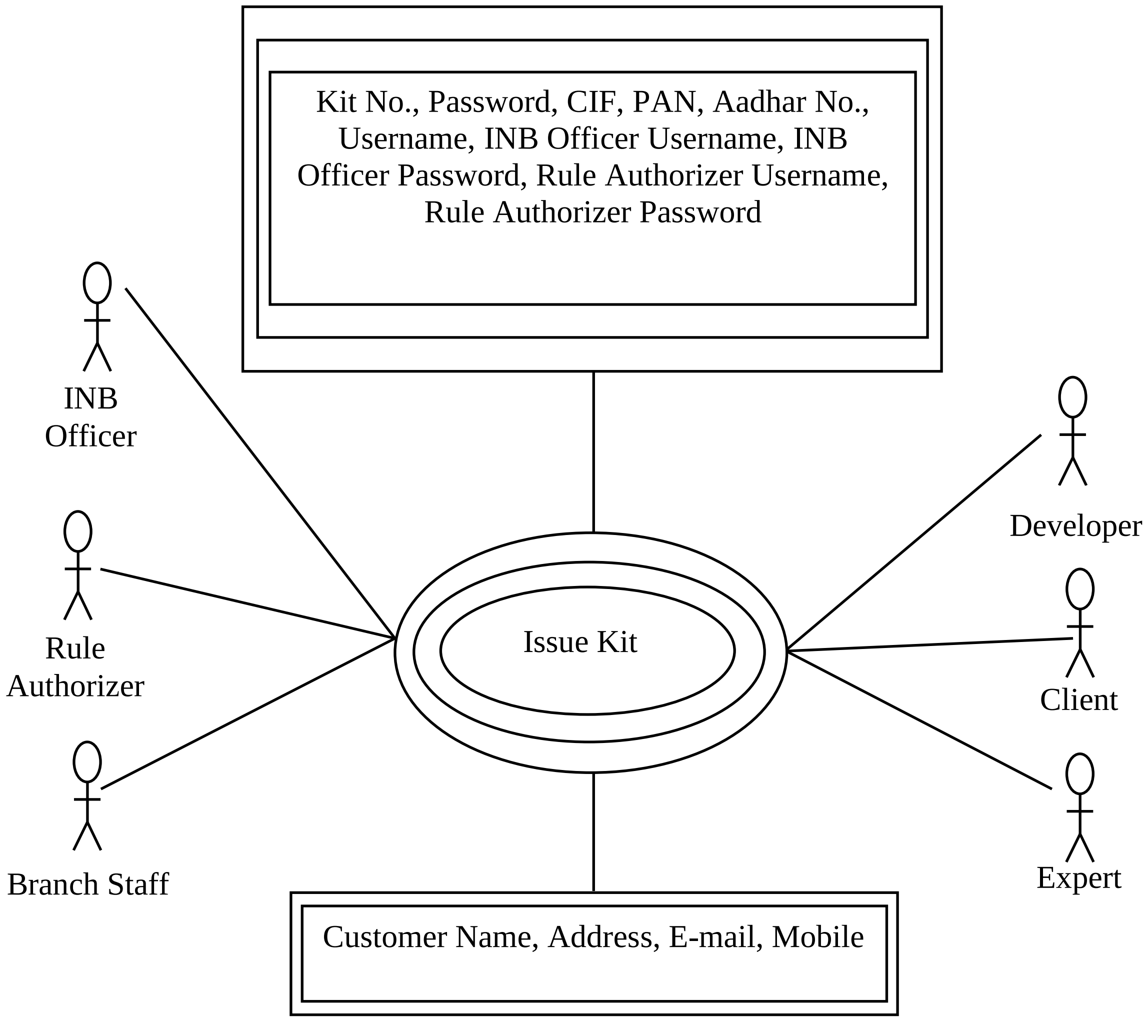

Kit No., Password, CIF, PAN, Aadhar No., Username, INB Officer Username, INB Officer Password, Rule Authorizer Username, Rule Authorizer Password
INB Officer
Rule Authorizer
Branch Staff
Issue Kit
Developer
Client
Expert
Customer Name, Address, E-mail, Mobile

Figure 4.23: FRD for Issue Kit

Table 4.17: Template of FRD for Issue Kit

Functionality: Issue Kit	
Stakeholders	
INB Officer	
Branch Staff	
Rule Authorizer	
Client	
Developer	
Expert	
Assets	
Name	**Rank**
Kit No.	High
Customer Name	Medium
Address	Medium
PAN	High
Aadhar No.	High
E-mail	Medium
Mobile No.	Medium
Username	High
Password	High
INB Officer Username	High
INB Officer Password	High
Rule Authorizer Username	High
Rule Authorizer Password	High
CIF	High
Functionality Rank: High	
Measure: Username, Password	

Figure 4.24 is a CRD that summarizes the complete software and Table 4.18 is its template.

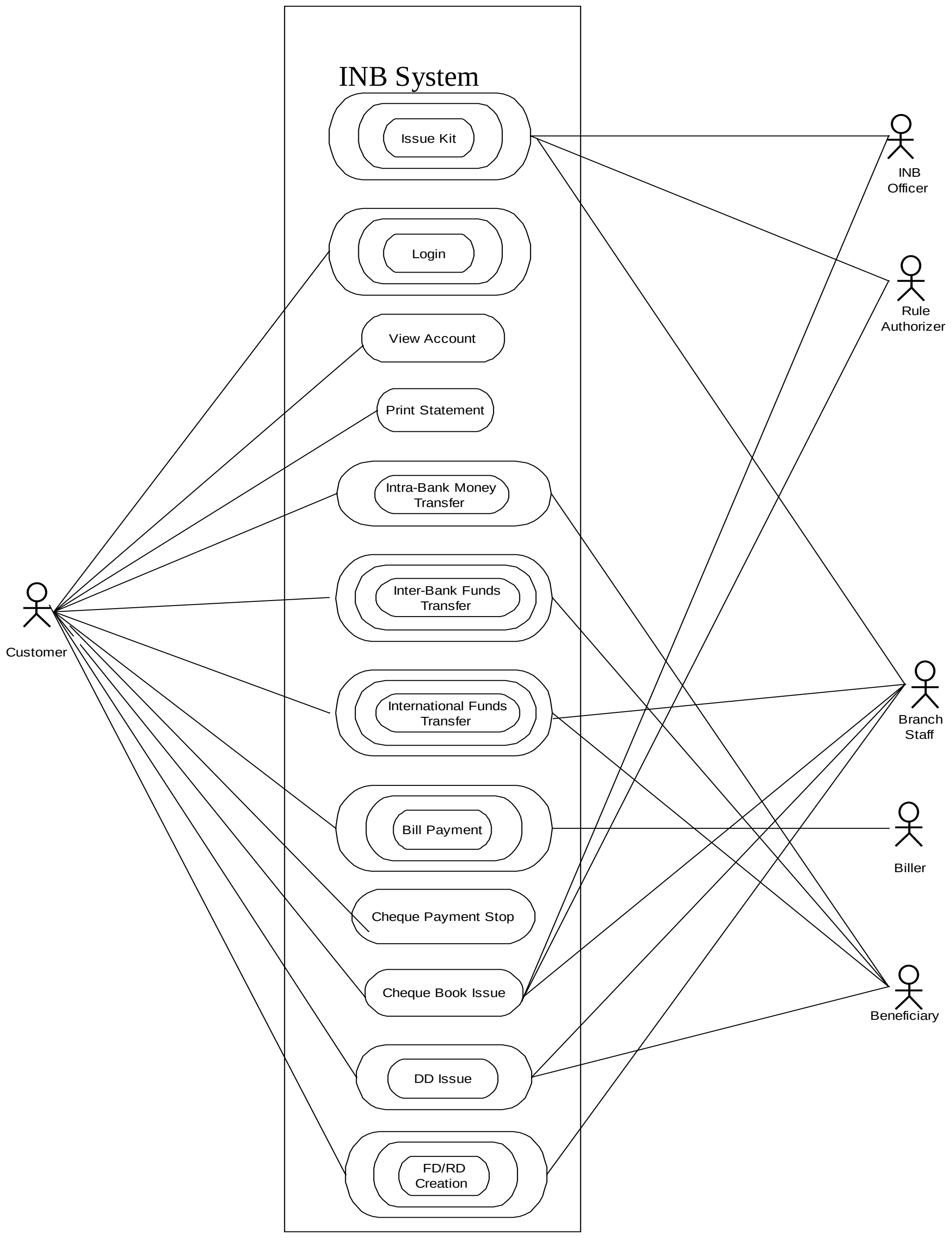


Figure 4.24: CRD for the complete INB Software

Table 4.18: Template for CRD

Name of Software: Internet Banking (INB)		
Function-List	**Rank**	**Stakeholders Associated**
Issue INB Kit	High	INB Officer, Rule Authorizer, Branch Staff, Customer, Client, Developer, Expert
Login	High	Rule Authorizer, Customer, Client, Developer, Expert
View Account	Low	Customer, Client, Developer, Expert
Print Statement	Low	Customer, Client, Developer, Expert
Intra-Bank Funds Transfer	Medium	Beneficiary, Customer, Client, Developer, Expert
Inter-Bank Funds Transfer	High	Beneficiary, Customer, Client, Developer, Expert
International Funds Transfer	High	Beneficiary, Customer, Client, Developer, Expert
Bill Payment	High	Beneficiary, Customer, Client, Developer, Expert
Cheque Payment Stop	Low	Branch Staff, Customer, Client, Developer, Expert
Cheque Book Issue	Low	Branch Staff, Customer, Client, Developer, Expert
DD Issue	Medium	Branch Staff, Beneficiary, Customer, Client, Developer, Expert
FD/RD Creation	High	Branch Staff, Beneficiary, Customer, Client, Developer, Expert

4.5.2 Authorization Rank Diagram (ARD)

Authorization Rank Diagrams (ARDs) have been developed which show the authorization right a stakeholder possesses over his/her relevant assets. The ARDs have been explained in Sub-section 3.7.3. Figure 4.25 represents a stakeholder-oriented ARD for INB officer. It can be seen that the INB officer has Low rank or ‘Read’ right on assets Customer Name and PAN, high rank or ‘Write’ right on CIF no., INB Officer Username, INB Officer Password and Kit No. Furthermore, he/she has medium rank on ‘Write with Permission (WP)’ right provided by customer on username, password and user profile password assets [109]. Table 4.19 is the template for the diagram.

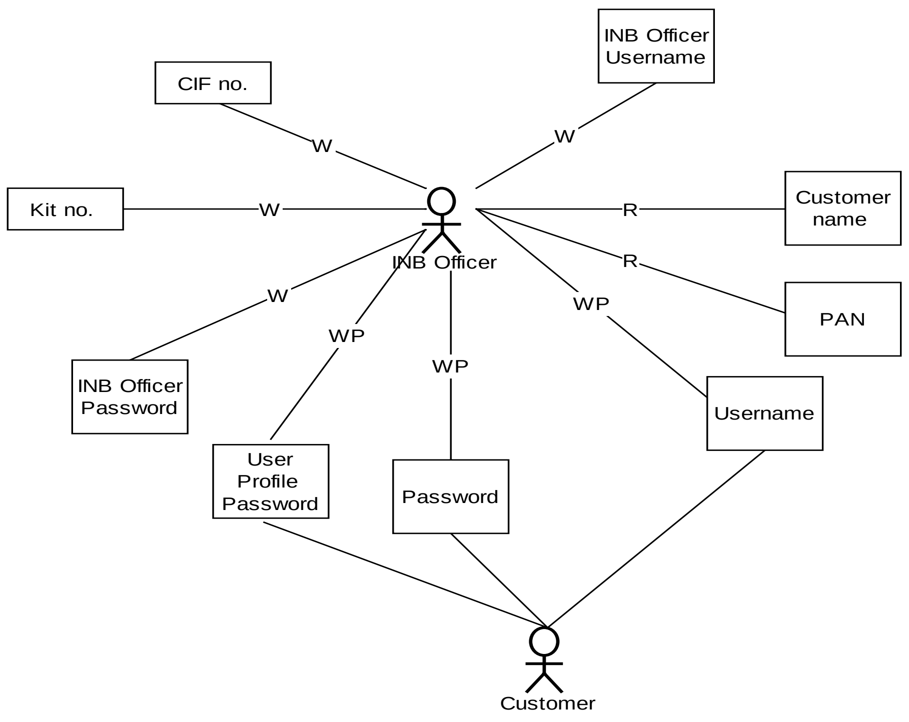


Figure 4.25: Authorization rights of a stakeholder INB officer

Table 4.19: ARD Template for INB Officer

Stakeholder: INB Officer			
Assets	**Security Rank**	**Authorization Rights**	**Permitting Stakeholders**
Customer Name	Low	R	
PAN	Low	R	
CIF No.	High	W	
INB Officer Username	High	W	
INB Officer Password	High	W	
Kit No.	High	W	
Username	Medium	WP	Customer
Password	Medium	WP	Customer
User Profile Password	Medium	WP	Customer

Figure 4.26 shows an Asset-oriented ARD for asset DD no. It can be seen that the stakeholders associated with it include Customer and Beneficiary with ‘Read’ right, and Branch Staff with ‘Write’ right. Table 4.20 is the template for this diagram. More ARDs are presented in Section A.4

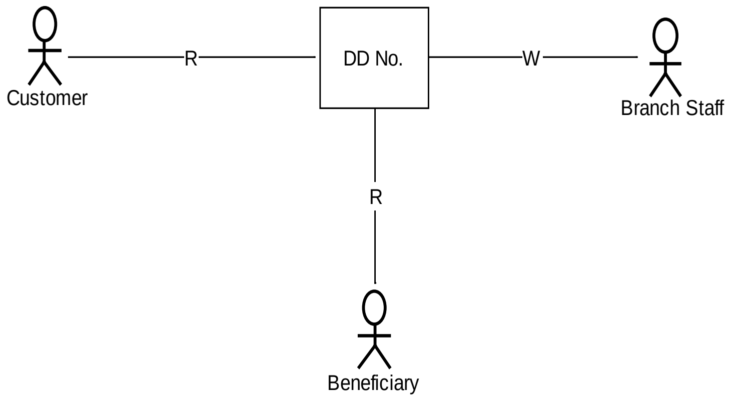


Figure 4.26: Asset-Oriented ARD for DD no.

Table 4.20: Template for Asset-oriented ARD for DD no.

Asset: DD No.	
Stakeholder	**Authorization Rights**
Customer	R
Beneficiary	R
Branch Staff	W

4.6 Summary

This chapter successfully demonstrates the application of the proposed methodology in Internet Banking. Banking is a very security-intensive and challenging domain because it performs several functions, involves financial transactions and has several stakeholders. The application of SecREAD in this industry has been successful in capturing the aspirations of these stakeholders through the system. This has been achieved by providing a better model for eliciting security requirements from stakeholders of different backgrounds through stories. This facilitates induction of security in the development process leaving least chance of any security issue arising at a later stage which can be disastrous in this domain. As well as the methodology aids in making better sense of those requirements to the developers through the notion of SCDs. The usefulness of the SCDs is proved in Identification and Refinement phases. To our idea, for INB system, graphical representation has come out to be better than textual representation. Furthermore, the methodology well-defines the two-way authorization rights between stakeholders and assets that are of paramount importance in the application of this domain. The application of SecREAD to INB is quite close to the actual implementations available. The coupling of empirical analysis with the designing of the system to this level has made SecREAD more realistic. The tool developed for this research work provided flawless computations and automated generation of diagrams.

The proposed SCDs have come out to be very useful since textual representation of requirement is not free of ambiguities. Through these diagrams stories were visualized and redundancy was removed easily. The FRDs present very clearly security levels of individual functionalities in the internet banking. The ARDs represent the access rights of every stakeholder over different assets involved in the system. The computations given by Eq. (3.1), Eq. (3.2) and Eq. (3.3), and Algorithm 3.2, are implemented to develop the diagrams. The aim is to illustrate the development of systems by adding security in the development lifecycle early.

Chapter 5

Case Study: IoT-Enabled Smart Building

This is an era of Internet of Things (IoT) which encompasses smart applications like smart buildings, transportation systems, healthcare, industrial automation, smart city, smart home and various smart electronic devices like watches, goggles etc. In this research work, smart building has been chosen as a case study because it contains safety and security critical tasks to be taken care of. Security requirements are paramount in design and development of control software for such buildings. The security levels of Smart Buildings vary with their types. A residential building will have less security threat than a military or industrial building. Also, every kind of building is prone to some mishap or the other like earthquake, flood, fire, burglary etc. Any damage to a building can lead to severe loss of life and property. These situations demand installation of safety and security measures of the highest order. At the same time, the comfort of the residents is also one of the priorities. Keeping all this in mind SecREAD has been developed in Chapter 3.

In this chapter, Section 5.1 includes a brief description of IoT. Section 5.2 introduces smart buildings while Section 5.3 elaborates their major functionalities. Each of the Sections, from 5.4 to 5.8, deal with one phase of the SecREAD applied on the case study. Finally conclusion is drawn in Section 5.9. Appendix B contains tables and figures pertaining to this chapter, providing more detail and clarity.

5.1 Internet of Things

In the year 1999, Kevin Ashton of Procter & Gamble, coined the term "Internet of Things". It is the interconnection of various computing devices, used in everyday life, through internet facilitating them to exchange data. It is the inter-networking of physical devices embedded with electronics, software, sensors, actuators, and network connectivity that enable these objects to collect and exchange data. According to Laya and Bratu [110] IoT is "the infrastructure of the information society." Using IoT, the objects can be controlled and/or sensed remotely causing more direct integration of the computer-based systems with the physical world. This leads to improved efficiency, accuracy and economic benefit along with reduction in human intervention. According to Al-Fuqaha et al. [111], IoT is expected to bridge diverse technologies to enable new applications by connecting physical objects together in support of intelligent decision making. The IoT enables physical objects to see, hear, think and perform jobs by letting them share information and coordinate decisions. IoT talks about remote control of applications or functions of any Cyber-Physical system. Thus, IoT can be applied in various facets of life like industries, health, transportation, home, buildings and cities altogether. Designing software for cyber physical systems is difficult as they cut across varied aforesaid domains.

The rate at which physical objects are being developed and connected to the internet, IoT has surfaced as an attractive field for research equally lucrative for business. The annual business of IoT will be around \$6.2 trillion by 2015 [111]. Exhaustive literature survey reveals that no comprehensive method is proposed or developed for such kind of systems. IoT based physical systems consist of a large number of safety critical tasks and failure due to one or the other software error in the system will lead to catastrophic consequences. The demand of increased automation and need of security requirements in the development of such systems motivated us to think for the development of secure software development methods. The methodology presented in Chapter 3 is based on the concept of integration of security aspects in standard software development life cycle from the first phase. As already mentioned, the applications of IoT are far and wide, one very significant application of Smart Building has been chosen as a case study.

5.2 Smart Buildings

A Smart Building is a physical domain comprising both hardware and software devices owned by one or more individuals that can be monitored and controlled via the Internet [112]. Robles and Kim [113] define a smart building to be a building that is equipped with special structured wiring to enable occupants to remotely control or program an array of automated home electronic devices by entering a single command.

Building is a complex system and contains large number of ongoing activities. The input acquired from sensors of such systems for even small sized buildings is tremendous. A distinct feature of intelligent buildings is that their behavior constantly changes over time as they are supposed to continually adapt to the changing preferences and requirements of the inhabitants. Such an adaptation should be completely autonomous so that the inhabitants of the building are not required to configure it, repeatedly.

Security requirements in Smart Buildings have augmented in present times as more and more services are becoming online and people want to access them remotely. A smart building can lower energy and maintenance costs by fine-tuning its processes. This has necessitated their development in present times owing to increasing urban population which is expected to increase from 2.9 billion in 2015 to 4.3 billion by 2025. This population is responsible for the consumption of three-fourth of world's energy production and 80% of carbon dioxide emissions. The smarter the building is the more efficient is its operation and maintenance.

The Middle-East region is a forerunner in adopting smart buildings. The headquarters of Private Equity Bank in Bahrain and the building of Ministry of Higher Education in Riyadh are some good examples that render the building management and the residents the ability to control remotely a variety of subsystems like security, lighting, temperature control, fire alarm system, and elevator control system. However, in the literature survey, so far, no comprehensive methodology has been found that, from the very inception, takes into account the security concerns of all kinds of stakeholders involved and models them effectively in order to design a secure system. This may lead to vulnerabilities in these highly critical systems where a great volume of life and property is at stake. SecREAD methodology is an attempt to fill this void.

Some major components of smart buildings are Surveillance, Maintenance, Heating and Ventilation, Energy Conservation, Security, Disaster Management etc. It is not possible to address all tasks, so the scope of the discussion has been limited to those important tasks that are safety and security intensive. These are discussed in the next section.

5.3 Major functionalities of Smart Buildings

Some major functionalities of the Smart Building System are discussed in the next sub-sections.

5.3.1 Security System

This functionality includes authentication, access control, granting authorization rights, surveillance and intruder detection. Authentication is the first line of defense. This can only be achieved by using adequate technologies to ensure that safety and security of the inhabitants of the building. In order to detect and authenticate any entrant in the building premises, numerous technical solutions are available [42]. Various access control systems are used in the system such as smart cards and biometrical identification systems along with conventional username-password scheme. Video cameras are commonly applied in security surveillance systems. Retina scanners are also used for the purpose, which use a low-intensity light source and a delicate sensor to scan the unique pattern of blood vessels at the back of the retina. The intruder detection is necessary for any security system prevailing in the building. Again, as stated in previous case study, SecREAD methodology is restricted to only suggest a security measure while its implementation is a prerogative of client based on his/her constraints and the type of building in question.

5.3.2 Fire Safety System

Fire is undoubtedly one of the most disastrous calamity and its aversion and subversion both should be a prime objective in any building. The smart fire safety system is an answer to this menace. The extreme criticalness of this application has made it a part of our case study.

Generally when a fire breaks out the available information is either less or not systematic. Fire fighters independently observe the fire-ground and analyze the immediate situation and build a mental model. If the model is incorrect, problems could escalate and lead to severe loss of life and property. IoT can be used to create a fire safety system that may provide actionable information.

The fire is detected by the carbon monoxide (CO) levels, temperature and smoke level in the building. The mitigation action may be any one or combination of starting the sprinklers, turning off the electricity, shutting down elevators and escalators and sending messages to the fire department, the nearest hospital and the ambulance service. Inside the building the announcement system must be activated to alert the residents and they should be guided to the nearest exit. With the fire department, the information of the presence of residents through the close circuit television (CCTV) images and body motion, and detailed building plans (stairs, exits, utilities, standpipes, construction) should be shared.

5.3.3 Heating, Ventilation, and Air Conditioning (HVAC)

Buildings are among the largest consumers of electricity in the world. These account for 70% of total electricity and also are responsible for huge amounts of greenhouse gas emissions. This need for energy conservation has attracted the attention of researchers. It has lead to the development of HVAC system.

HVAC is an energy saving functionality. It is the technology of indoor and vehicular environmental comfort. Its goal is to provide thermal comfort and acceptable indoor air quality. HVAC constitutes of Heating, Ventilation, and Air Conditioning. Heating should be performed when the temperature fluctuates from the comfortable range and also on the basis of occupancy control. Ventilation is the process of exchanging or replacing air in any space to provide high indoor air quality which involves temperature control, oxygen replenishment, humidity control, removal of carbon dioxide etc. Air conditioning system provides cooling and humidity control for the building.

HVAC is now an important part of any building, like residential structures such as single family homes, apartment buildings, hotels and senior living facilities, medium to large industrial and office buildings such as skyscrapers and hospitals. Due to its wide range of applications it has been an obvious choice for our study.

Through the HVAC system installed, a resident may control the temperature of his/her home/building from a GUI system or even remotely from his car through a smart phone before he reaches. The HVAC system also adjusts itself to the resident’s preferences in past-history, the time of day and room temperature providing better comfort. Also, a device like an AC can be shut down remotely if it has been left on mistakenly before leaving the home.

5.3.4 Lighting

IoT enabled Lighting system is designed for energy efficiency. 19% of energy consumption and 6% of greenhouse emissions in the world are attributed to lighting. Optimizing this energy consumption is the need of the hour and smart lighting is instrumental in this. It allows the householder to control remotely the lighting. It also involves utilizing sunlight to reduce the use of man-made lighting which means that the lights can be switched on and off as per the outside sunlight luminance. This system also uses the technique of occupancy sensing or motion detection *i.e.* turn on light only when a person is within a particular space.

Apart from these four main sub-systems there are a large number of smaller sub-systems where safety is important. Some of them are burglar alarm system, access control system, authentication system, fire fighting system, temperature control system etc. These have been aggregated to form a Security System. From the next section essential phases of the proposed methodology in IoT based smart building system are discussed.

5.4 Identification and Refinement

In this phase the task is to critically identify all requirements of the system to be designed for smart building. The goal of the proposed methodology is different from the standard SDLC model as the task of the proposed methodology is to critically analyze the security concerns and provide a way to imbibe the same in the development of secure software. As stated above, the identification phase is restricted to list some of the important functionalities of the smart building to demonstrate the application of the proposed methodology in achieving the goal of the secure software development. Some of the important sub-systems of the smart building are Heating, Ventilation, and Air Conditioning (HVAC) system, Fire Safety System, Lighting System, Water Distribution System, Electricity Supply System, Drainage System etc. Each of these sub-systems is studied and functionalities, stakeholders and assets associated with them are identified.

During the development of the process of requirement gathering, it is observed that the proposed notion of story-telling and SCDs has proved its effectiveness as it is easier for stakeholders to narrate requirements in the form of stories.

Further, the requirements of the IoT enabled building are different as automation concept is little different. Here, it is desired that all applications/tasks should be controlled remotely through any form of internet connectivity. Another major difference is the use of smart sensors and smart actuators to fulfill the need of the system’s intelligence. It raises many questions in every phase of the system design. In this phase one has to take decision about the control data security and device access security, making secure software design methodologies more relevant. As discussed earlier these systems are more vulnerable to attack and therefore integrating security right from the inception is utmost essential. In this way the proposed methodology, SecREAD, becomes more important and the contribution is novel in the sense that it is the first methodology which is addressing IoT requirements of a system and formalizing the design and development process.

The IoT compliance of any software developed for big applications like smart building demands more attention towards security as it includes a number of critical tasks along with their priority of execution, to be accomplished by various devices to ensure correct functioning of the systems. As a smart building is to comply with the IoT requirements, the IoT related functionalities need to be identified. Several stakeholders or people are associated with the building like residents, workers, guests etc. Further, in an IoT based application, sub-systems or devices ought to communicate with each other and with the external world, bringing in a variety of data. The stories narrated by stakeholders consist of IoT features of various sub-systems. Therefore, it can be said that coupling the IoT technology with buildings adds a number of unique features to them making the requirement gathering, design as well as the development of the system quite complex. SecREAD methodology is perfectly aligned to address these intricacies of the Smart Building System as its foundation is the identification of functionalities, stakeholders and data assets. For every story, as soon as it is elicited, the entities are extracted and put into sets segregating the functionalities, stakeholders and assets. The sets are incremented whenever a new entity is encountered. Furthermore, the highly intertwined associations between the entities of the Smart Building System, derived from the stories, can be expressed very clearly through the SCDs.

To identify the entities comprehensively the spiral phase of Identification comes in very handy. It is initiated by the core group members constituting experienced developer(s), client representative(s) and domain expert(s). In this case, the Owner of the building is the client and experts include, among others, urban planner(s) and representative(s) from local government who will ensure the compliance with standards, adherence to policies and obtaining licensing approvals. The refinement process follows each identification phase. The spiral activity continues till all the stakeholders, assets and functionalities are identified.

As seen in Figure 3.1 this phase runs in spiral with the elicitation of story in the first quadrant followed by identification of entities, drawing of SCDs, refinement and updation of entity-sets in the second, third and fourth quadrants respectively. Some example stories are listed in Table 5.1 to demonstrate the way the stakeholder set S, asset set A and functionality set F are obtained.

Table 5.1: Stories Elicited and Entities Identified for Smart Building

S.No.	Stories	Type of Story	Entities Identified from the Story			Set	Incremented Sets
			Stakeholders	Functionalities	Assets		
1	The Password of the inhabitants must be kept confidential	Three-Entity Story	Inhabitant	Authentication System	Password	S = {Inhabitant} F = {Authenti-cation System} A = {Password}	S = {Inhabitant} F = {Authentication System} A = {Password}
2	If a burglar tries to enter the building message police	Multi-Instance Story	Burglar, Police	Burglar Alarm System	Message	S = {Burglar, Police} F = {Burglar Alarm System} A = {Message}	S = {Inhabitant, Burglar, Police} F = {Authentication System, Burglar Alarm System} A = {Password, Message}
3	Only the Building Manager can assign username, password to every resident	Multi-Instance Story	Building Manager, Resident	Access Control System	Username, Password	S = {Building Manager, Resident} F = {Access Control System} A = {Username, Password}	S = {Inhabitant, Burglar, Police, Building Manager, Resident} F = {Authentication System, Burglar Alarm System, Access Control System} A = {Password, Message, Username}
4	Every entrant must be authenticated	Two-Entity Story	Entrant	Authentication System		S = {Entrant} F = {Authenti-cation System} A = {ϕ}	S = {Inhabitant, Burglar, Police, Building Manager, Resident, Entrant} F = {Authentication System, Burglar Alarm System, Access Control System} A = {Password, Message, Username}
5	Lights should be switched off when inhabitants are not present	Three-Entity Story	Inhabitant	Lighting System	Body Motion	S = {Inhabitant} F = {Lighting System} A = {Body Motion}	S = {Inhabitant, Burglar, Police, Building Manager, Resident, Entrant} F = {Authentication System, Burglar Alarm System, Access Control System, Lighting System} A = {Password, Message, Username, Body Motion}
6	Ring alarm when some indication of smoke is	Two-Entity Story		Fire Safety	Smoke Level	S = {ϕ} F = {Fire Safety	S = {Inhabitant, Burglar, Police, Building Manager, Resident, Entrant}

	found.			System		System}
						A = {Smoke Level}
							F = {Authentication System, Burglar Alarm System, Access Control System, Lighting System, Fire Safety System}
							A = {Password, Message, Username, Body Motion, Smoke Level}

7	Adjust the room temperature automatically for the resident	Three-Entity Story	Resident	HVAC	Room Temper-ature	S = {Resident}
						F = {HVAC}
						A = {Room Temperature}
							S = {Inhabitant, Burglar, Police, Building Manager, Resident, Entrant}
							F = {Authentication System, Burglar Alarm System, Access Control System, Lighting System, Fire Safety System, HVAC}
							A = {Password, Message, Username, Body Motion, Smoke Level, Room Temperature}

8	Maintain appropriate level of moisture	Two-Entity Story		HVAC	Moisture Level	S = {ϕ}
						F = {HVAC}
						A = {Moisture Level}
							S = {Inhabitant, Burglar, Police, Building Manager, Resident, Entrant}
							F = {Authentication System, Burglar Alarm System, Access Control System, Lighting System, Fire Safety System, HVAC}
							A = {Password, Message, Username, Body Motion, Smoke Level, Room Temperature, Moisture Level}

9	The Humidity Level should be comfortable.	Two-Entity Story		HVAC	Humidity Level	S = {ϕ}
						F = {HVAC}
						A = {Humidity Level}
							S = {Inhabitant, Burglar, Police, Building Manager, Resident, Entrant}
							F = {Authentication System, Burglar Alarm System, Access Control System, Lighting System, Fire Safety System, HVAC}
							A = {Password, Message, Username, Body Motion, Smoke Level, Room Temperature, Moisture Level, Humidity Level}

10	Building Manager is the chief administrator of the system	One-Entity Story	Building Manager			S = {Building Manager}
						F = {ϕ}
						A = {ϕ}
							S = {Inhabitant, Burglar, Police, Building Manager, Resident, Entrant}
							F = {Authentication System, Burglar Alarm System, Access Control System, Lighting System, Fire Safety System, HVAC}
							A = {Password, Message, Username, Body Motion, Smoke Level, Room Temperature, Moisture Level, Humidity Level}

Figures 5.1 to 5.10 depict the SCDs for the stories 1 to 10.

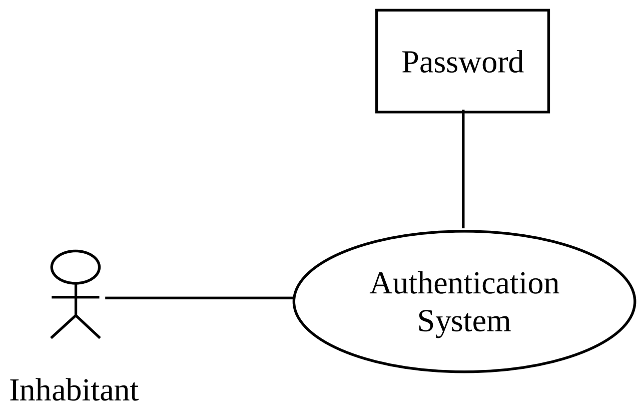


Figure 5.1**:** SCD for story 1

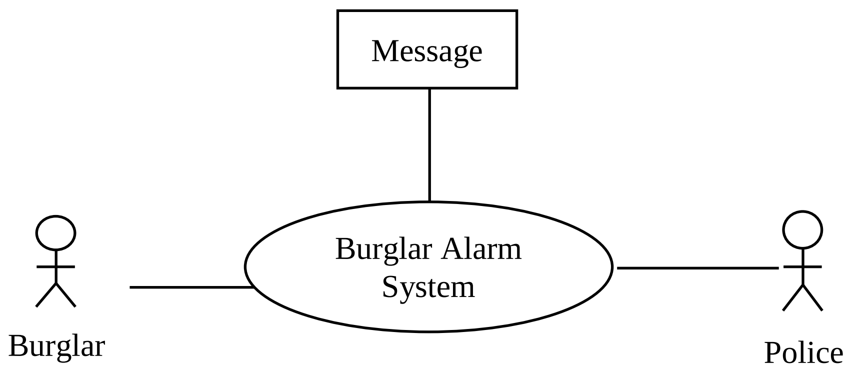


Figure 5.2: SCD for story 2

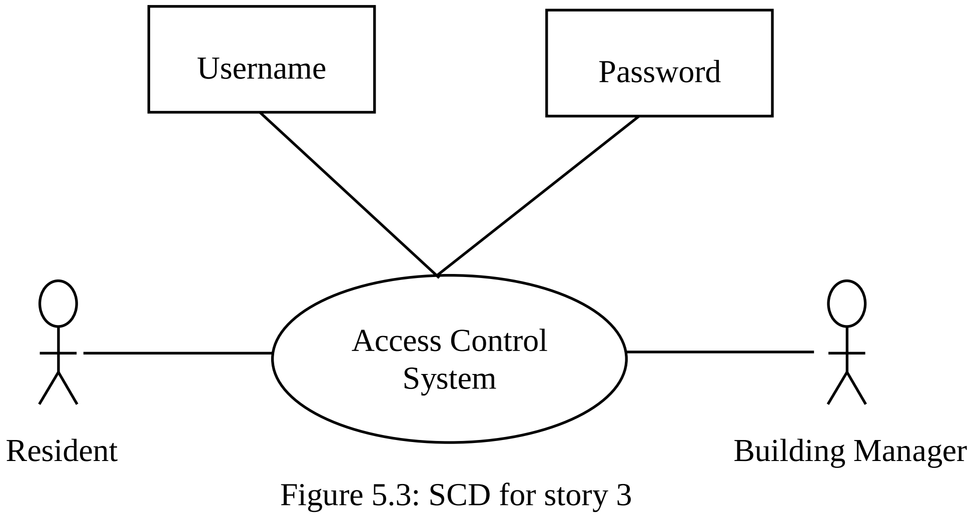


Figure 5.3: SCD for story 3

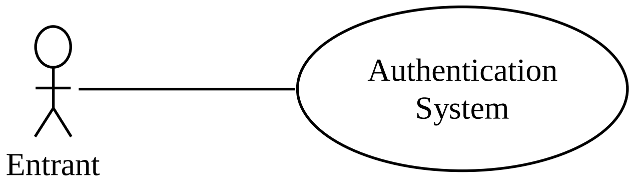


Figure 5.4: SCD for story 4

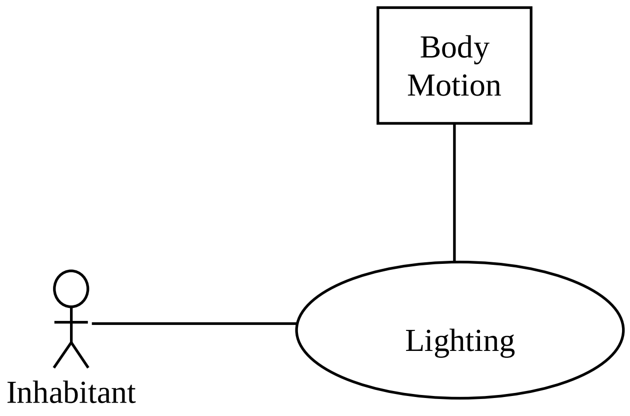


Figure 5.5: SCD for story 5

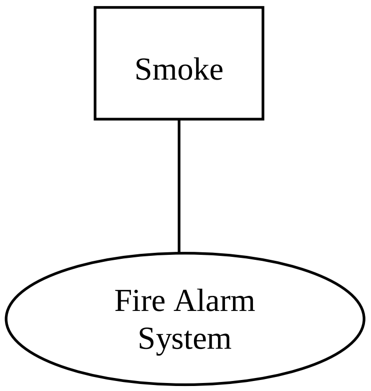


Figure 5.6: SCD for story 6

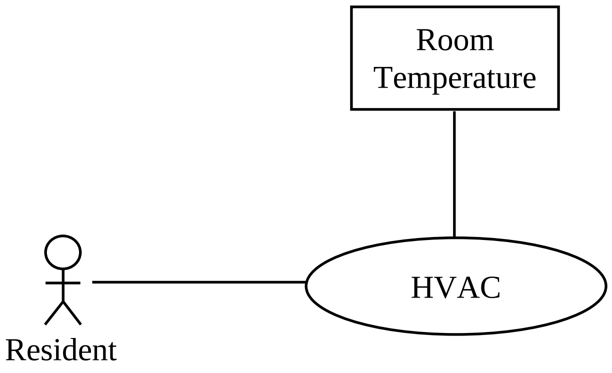


Figure 5.7: SCD for story 7

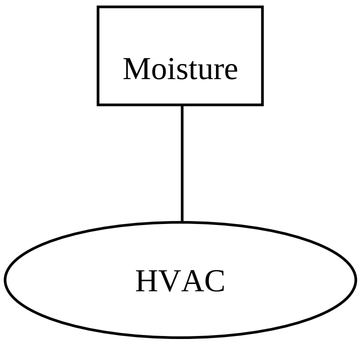


Figure 5.8: SCD for story 8

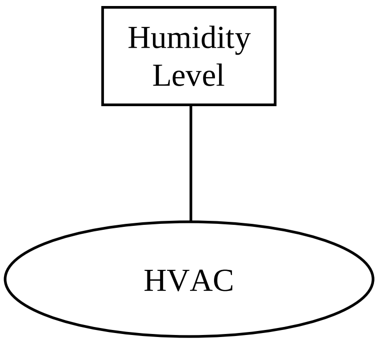


Figure 5.9: SCD for story 9

Building Manager

Figure 5.10: SCD for story 10

Every time, after eliciting a story and identifying its entities the refinement phase is conducted. As described in Section 3.3 it includes the activities of redundancy removal, decomposition, aggregation and addition of entities, as required. All these activities are discussed below with examples related to this case study as per the example stories taken in Table 5.1. Since there are varied stakeholders of the system from whom requirement stories are elicited from which the entities are to be identified, it is of utmost importance that information entities are unambiguous and remain consistent throughout the system. To fulfill this necessity, the Refinement phase of SecREAD with the use of SCDs is very apt and can serve the purpose quite effectively, as will be seen next.

Sets S, F and A are obtained based on the primary stories in Table 5.1 and are given here as Eq. (5.1), Eq. (5.2) and Eq. (5.3) respectively.

S = {Inhabitant, Burglar, Police, Building Manager, Resident, Entrant} (5.1)

F = {Smart Authentication System, Smart Burglar Alarm System, Smart Access Control System, Smart Lighting System, Smart Fire Safety System, HVAC}(5.2)

A = {Password, Message, Username, Body Motion, Smoke Level, Room Temperature, Moisture Level, Humidity Level} (5.3)

5.4.1 Redundancy Removal

Redundancy in entity sets is removed by using the algorithm. As explained in section 4.2.1 for INB here also in set S given by Eq. (5.1), two entities ‘Inhabitants’ and ‘Residents’ are redundant so a common term is selected to avoid ambiguity in developing the software system. Here, ‘Resident’ is a more appropriate term as, according to Oxford dictionary, it specifically applies to human beings (not including animals as in ‘Inhabitant’). So, the term ‘Inhabitant’ is discarded. ‘Inhabitant’ is found in Figure 5.1 and Figure 5.4. The figures are modified as Figure 5.11 and Figure 5.12 respectively. It is seen in Eq. (5.2) that set A contains two assets namely ‘Moisture Level’ and ‘Humidity Level’ that convey the same meaning. Therefore, only ‘Moisture Level’ is accepted of the two. In this light, it is deduced that Figures 5.7 and 5.9 are redundant. Accordingly, Figure 5.9 is discarded.

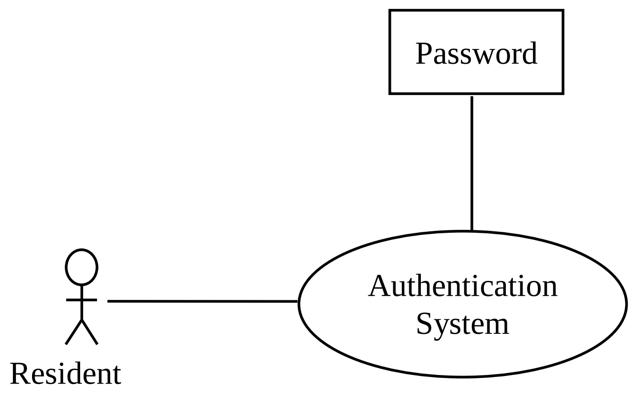


Figure 5.11: SCD for story 1 after redundancy removal

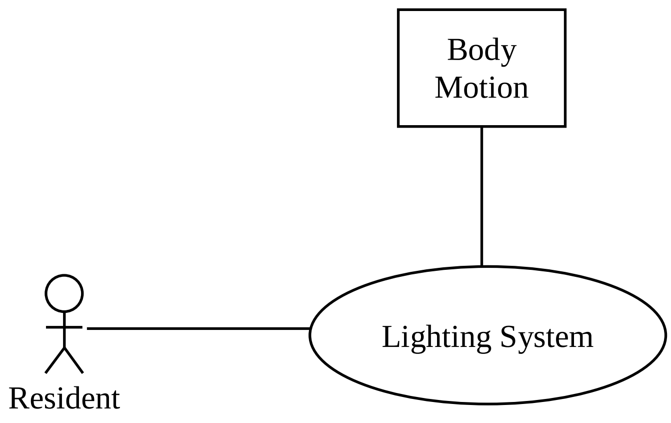


Figure 5.12: SCD for story 5 after redundancy removal

5.4.2 Decomposition

Decomposition is an activity to break an entity into its constituent entities if they possess different properties. For example, the stakeholder ‘Entrant’ is a very generalized term. Actually, it signifies three kinds of persons namely ‘Residents’ who live in the building, ‘Guest’ who visit or are temporary residents or workers and ‘Intruders’ who are illegitimate entrants. Any person that enters without authentication with malicious intentions is named as intruder. The decomposition is performed in the SCD that contains ‘Entrant’ as stakeholder *i.e.* SCD given by Figure 5.4. Figure 5.13 shows the SCD obtained after the decomposition. Similarly, other entities can be identified and decomposed as per need.

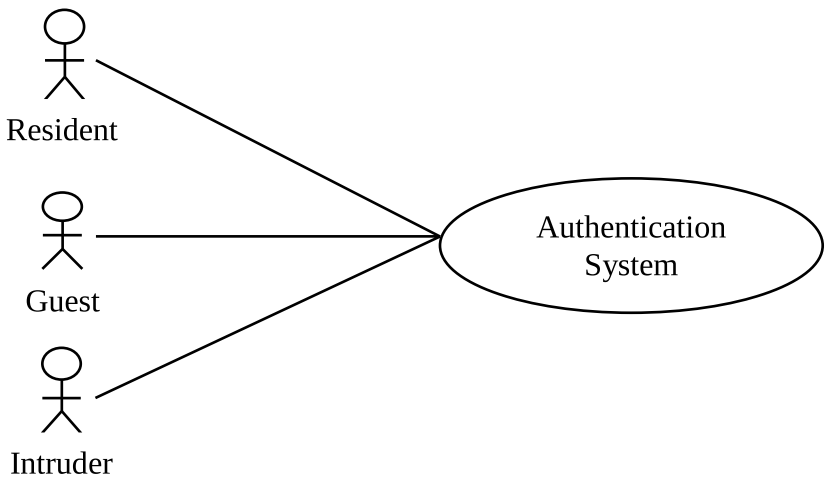


Figure 5.13: Decomposition of Figure 5.8

5.4.3 Aggregation

Through Aggregation similar information is combined. For this, the previous SCDs are scanned and aggregated based on the common functionality they represent. It is found that the functionality 'HVAC' is common in Figures 5.7 and 5.8 (Figure 5.9 has already been discarded in Redundancy Removal). An aggregated SCD (Figure 5.14) is developed that summarizes associations pertaining to this functionality.

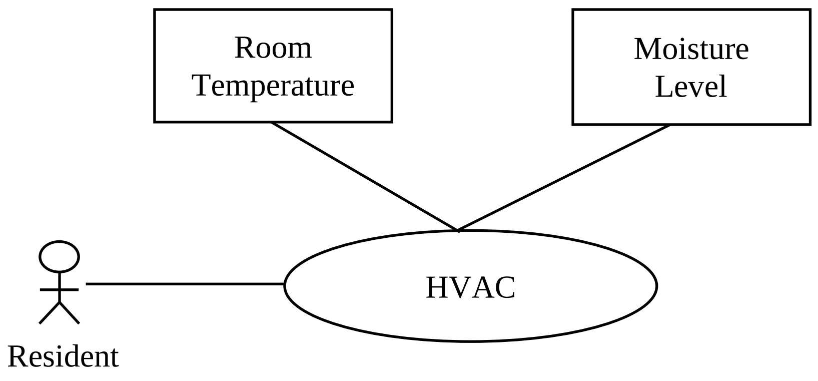


Figure 5.14: Aggregated SCD for HVAC System

In Figures 5.2, 5.3, 5.11 and 5.13 we encounter the functionalities of 'Smart Burglar Alarm System', 'Smart Access Control System' and 'Smart Authentication System'. These three functionalities are inseparable constituents of 'Security' and hence, these are aggregated into one 'Security System' by consolidating the said SCDs. It is noteworthy that Figures 5.1 and 5.4 are not considered since they have already been modified to Figures 5.11 and 5.13 respectively. Again SCDs come in handy to attach all entities associated with different functionalities to one functionality. The 'Burglar' is dropped as the 'Intruder' term is inclusive of Burglar or any other person with malicious intentions like terrorists. The aggregated SCD is given by Figure 5.15. In similar fashion aggregated SCDs can be obtained for every functionality from the preliminary SCDs.

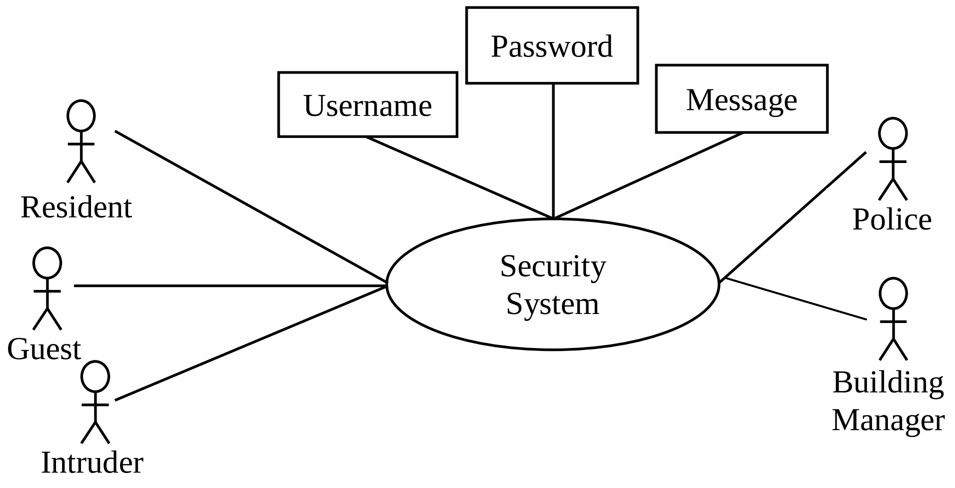


Figure 5.15: Aggregated SCD for Security System

5.4.4 Culmination of Spiral

When the identification and refinement activities were applied on the complete system, more entities were obtained incrementally. Final entity sets were obtained after the spiral ends. Based on our survey the functionality, stakeholder and asset sets were obtained. These have been enlisted below. Sincere efforts have been made to identify the entities exhaustively. However, some might have been still left out unintentionally.

It has been underlined in the third chapter that the comprehensive listing of stakeholders is crucial for the success of the proposed SecREAD methodology. The basic stakeholders are the developers, the domain experts, the owner and the users. The Fire Department and the Police are the outside agencies that act in emergency situations. Building Manager is the most important functionary of a smart building system that monitors every function and is primarily responsible for the maintenance and smooth functioning of the building. The iterative nature of the identification and refinement phase helps in discovering all the stakeholders. They are enlisted in Table 5.2.

Table 5.2: Stakeholder Set S for Smart Building

Building Manager
Resident
Guest
Intruder
Fire Department
Police
Owner
Developer
Expert

A smart building has a number of sub-systems. Each of these functionalities have a number of sub-functionalities which also need to be identified and analyzed. For example, in a smart building. the lighting system consists of lights switch on/off, controlling intensity of light, reporting faulty equipments etc. Similarly, Fire safety system may consist of detecting smoke level, raising an alarm, informing concerned agencies etc. The HVAC system consists of heating, cooling, controlling moisture in air etc. Security System may consist of ringing burglar alarm, informing police, informing owner etc. However, only broad functionalities are considered in which the related sub-functionalities are aggregated. These broad functionalities are listed in Table 5.3.

Table 5.3: Functionality Set F for Smart Building

Lighting System
Fire Safety System
Security System
HVAC

Similarly, the finally obtained asset set A is presented in Table 5.4. The set includes credentials falling under different security schemes *i.e.* 'Something-you-know' (Username and Password), 'Something-you-have' (Smart-card) 'Something-you-are' (Fingerprint and Iris Image), and CCTV Footage. These credentials facilitate authentication, access control and defining authorization rights of stakeholders. Their usage varies with functionalities, domains, owner's aspirations and economic constraints. The SecREAD methodology adjudges the level of security for entities as low, medium and high and based on the rankings assists the developers and owner to select appropriate security measures.

Day-light Luminance is measured in candela per square meter (cd/m^2) and according to this the light inside the building is maintained by the Lighting System. As per normal Human requirement, the HVAC system maintains the Humidity Level in the air in the range of 30% to 50%. Also, the HVAC and the Smart Lighting Systems aim at minimizing the Electricity Consumption. The SMS and e-mail are the preferred means of communication and notifications to and fro residents and the system, with the outside agencies like Police and Fire Department. The smoke, CO and room temperature, beyond the preset levels, raise an alarm of fire. Residents' preferences to various attributes of internal climate, at different times of the day, are recorded for servicing the residents better. Body Motion of residents is sensed to on-off lights.

Table 5.4: Asset Set A for Smart Building

Body Motion
Day-light Luminance
Room Temperature
Humidity Level
A/C Fan Speed
Electricity Consumption
Time of Day
CO Level
Smoke Level
SMS
E-mail
Username
Password
Fingerprint image
Iris image
CCTV Footage
Smart Card

As per the SecREAD methodology described in Chapter 3, when the complete entity sets are obtained, the entities are mapped to each other in the Mapping phase, explained in the next section.

5.5 Mapping

As propounded in Chapter 3, in this phase the entities are mapped to each other using Relevance Matrices. All these matrices have already been defined in Section 3.4. Matrix X maps Assets to Functionalities and matrix Y maps Functionalities to Stakeholders. For Smart Building, X is shown in Table 5.5 and Y is shown in Table

5.6. In these matrices the relevance or association is shown by ‘1’ and irrelevance by ‘0’. Only a portion of matrix X is presented here. The complete matrix is presented in Table B.1.

Through the cross product of X and Y, matrix Z is obtained that associates assets with stakeholders. A portion of Z is given by Table 5.7. Since, product values are insignificant, associations in Z are shown by shaded cells. Table B.2 gives the detailed view of the Z matrix for the Smart Building System, generated through the tool developed for this purpose that has been described in Section 3.8. It is noteworthy that since the tool performs arithmetic calculations, that table contains numeric values. ‘Intruder’ is not considered in the mapping process since he is not supposed to participate in the ranking process.

Table 5.5: Asset-Functionality Relevance Matrix X for Smart Building

Assets	**Functionality**			
	Lighting	Fire Safety System	Security System	HVAC
Fingerprint Pattern	0	0	1	0
Password	0	0	1	0
Electricity Consumption	1	0	0	1
Smoke Level	0	1	0	1

Table 5.6: Functionality-Stakeholder Relevance Matrix Y for Smart Building

Functionality	**Stakeholders**							
	Building Manager	Resident	Guest	Fire Department	Police	Client	Developer	Expert
Lighting	1	1	1	0	0	1	1	1
Fire Safety System	1	1	0	1	0	1	1	1
Security System	1	1	1	0	1	1	1	1
HVAC	1	1	1	0	0	1	1	1

Table 5.7: Asset-Stakeholder Matrix Z for Smart Building

Assets	**Stakeholders**							
	Building Manager	Resident	Guest	Fire Depart-ment	Police	Client	Developer	Expert
Fingerprint Pattern	▨	▨	▨		▨	▨	▨	▨
Password	▨	▨	▨		▨	▨	▨	▨
Electricity Consumption	▨	▨				▨	▨	▨
Smoke Level	▨	▨		▨		▨	▨	▨

In matrix X it is seen that asset 'Fingerprint Pattern' is mapped to functionality 'Smart Security System' and in Y matrix 'Smart Security System' is mapped to stakeholder 'Police'. Consequently, in matrix Z which is the cross product of X and Y it is seen that 'Fingerprint Pattern' is relevant to Police. In this manner the concept of developing Z is verified.

The relevance matrices, particularly X and Y, together are instrumental in showing the association of assets and stakeholders with a functionality. Therefore, through these matrices the CSCDs are developed. These are made through the Algorithm 3.2. CSCDs are elaborated in Sub-section 3.4.3. The CSCD for Security System can be seen in Figure 5.16. The associations shown in the figure can be verified by the matrices given by Table B.1 and Table 5.6.

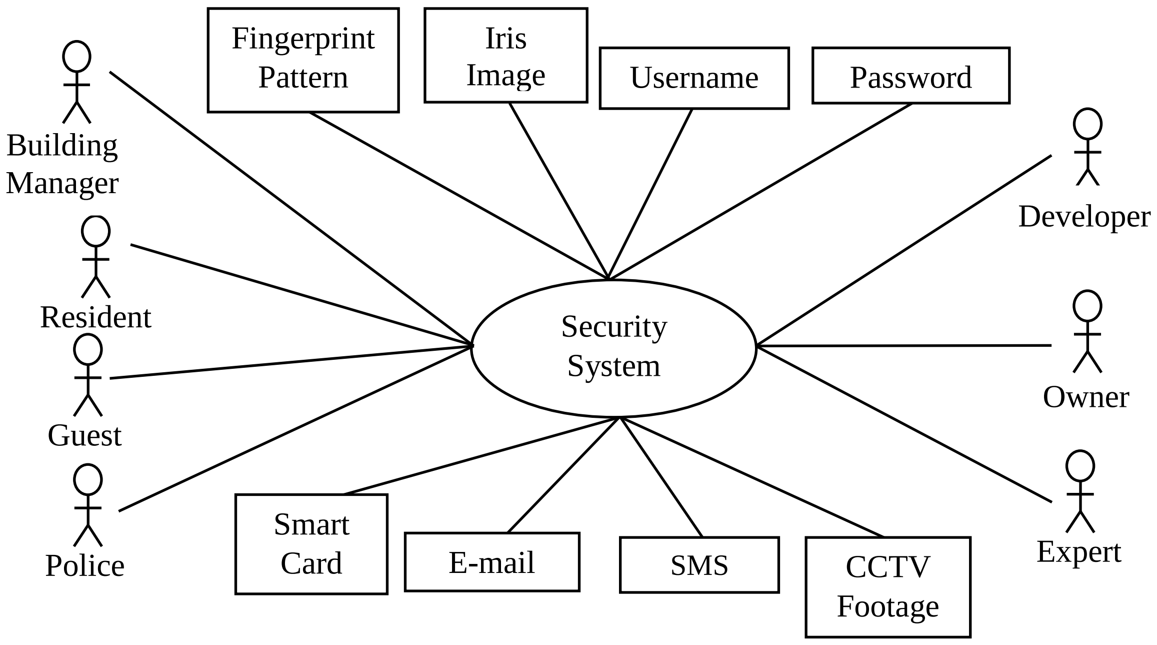


Figure 5.16: CSCD for Smart Security System

Another relevance matrix is W presented by Table 5.8 that maps stakeholders to their relevant parameters. It is developed by the core group. This matrix is utilized later in the ranking phase given in the next section.

Table 5.8: Stakeholder- Parameter Matrix W for Smart Building

Stakeholders	**Parameters**				
	Authentication	Confidentiality	Integrity	Non-repudiation	Authorization
Building Manager	▨	▨	▨	▨	▨
Resident		▨	▨		
Guest			▨		
Fire Department			▨	▨	
Police		▨	▨	▨	
Owner	▨	▨	▨	▨	▨
Developer	▨	▨	▨	▨	▨
Expert	▨	▨	▨	▨	▨

5.6 Ranking and Analysis

According to the matrices obtained out of the mapping phase a rank matrix R is created for the complete system in which the ranks are elicited by the stakeholders and later analyzed to find final ranks of assets and functionalities.

5.6.1 Rank Matrix

The process of ranking is conducted by stakeholders. A three dimensional Rank matrix R is developed by the combination of matrices Z and W. This matrix is defined in Sub-section 3.4.5. Every z-dimension of R denotes a rank sheet for one stakeholder. It contains the ranks furnished by the said stakeholder for his/her relevant assets some of which shown here are Fingerprint Pattern, Smart Card, Electricity Consumption, Humidity Level and Room Temperature furnished by the Resident. The assets are ranked over his/her relevant parameters which, as per matrix W (Table 5.8), are Confidentiality and Integrity. The complete R matrix has over 1000 rows where every asset is listed with values under all parameters supplied by all stakeholders. The irrelevant cells are marked by zeros. Figure 5.18 shows a portion of R for this case study in the 3-d format. In this figure the rank-sheet for the stakeholder 'Building Manager' is shown that contains the rank furnished by him/her for the assets Fingerprint Pattern, Body-Motion, Username and Password. The ranks are given under the five security parameters that are denoted, due to space constraints, as Pr_1, Pr_2, Pr_3, Pr_4 and Pr_5 for Authentication, Confidentiality, Integrity, Non-repudiation and Authorization respectively. Behind this rank sheet are the rank sheets for Resident, guest and so on for all other stakeholders. A larger portion of R, showing the rank values given by Building Manager and Resident is given in Table B.3. The structure of matrix R as obtained by the tool is already explained in Section 3.8.

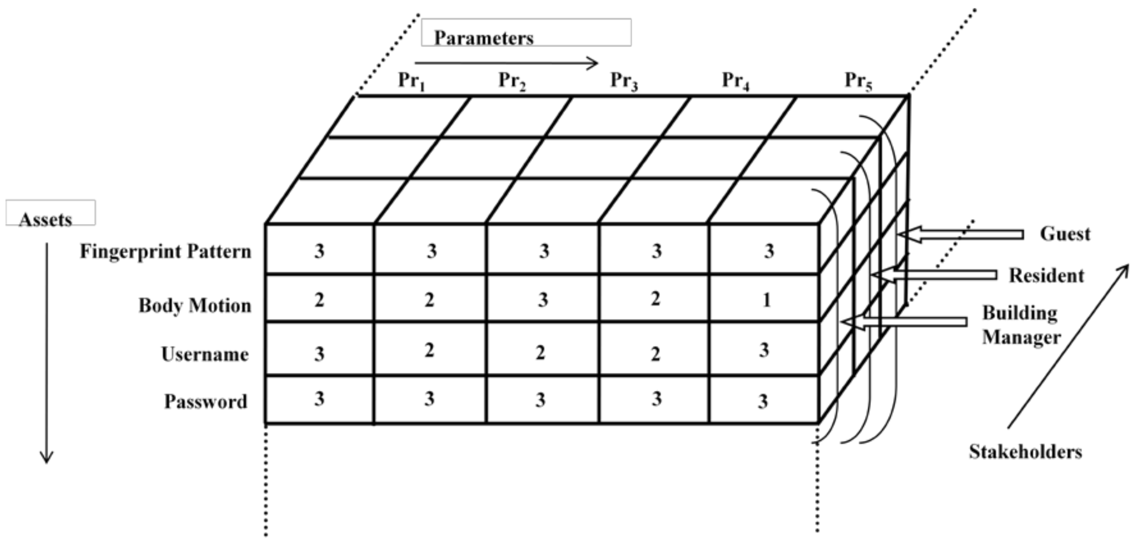


Figure 5.17: 3-d View of R for Smart Building

It is seen that the ranks are provided under the parameters of Confidentiality and Integrity only as only these two are defined as relevant for the said stakeholder in the matrix W, given by Table 5.8. The ranks of ‘Fingerprint Pattern’ and ‘Iris Image’, for the two parameters, are high which match with that of rank sheet of Resident, given by Figure 5.17. The complete matrix R contains the entries of all the stakeholders. The detailed view of R is presented by Table B.3.

5.6.2 Results and Discussion

Computations are performed on R once all the stakeholders have performed ranking of their relevant assets under their relevant parameters. After performing cell computation (Algorithm 3.2) on matrix R, matrix T is obtained. It is derived from matrix R and defined in Section 3.5. It has rows as assets and columns as parameters. A part of the matrix T is given by Table 5.9 that shows mode values for all the five parameters for the asset ‘Room Temperature’. Complete matrix T is given by Table B.4.

Table 5.9: Matrix T for Smart Building

Assets	Parameters	Modes
Room Temperature	Authorization	1
Room Temperature	Authentication	2
Room Temperature	Confidentiality	1
Room Temperature	Integrity	2
Room Temperature	Non-repudiation	2

Matrices U and V are defined in Section 3.5. These are derived by performing row computation (Algorithm 3.3) and column computation (Algorithm 3.4) respectively on matrix T. Table 5.10 is a partial view of matrix U for the Smart Building System that gives the ranks of assets. It can be seen in T (Table 5.9) that the asset Body Motion is ranked under five parameters as 1, 2, 1, 2 and 2. The mode of all these values is 2 and consequently rank is medium. This is visible in matrix U. In this way the creation of matrix U by the tool is verified. The complete matrix U is given by Table B.5.

Table 5.10: Asset Rank or Matrix U for Smart Building

Asset	**Mode**	**Rank**
Room Temperature	2	Medium
Fingerprint Pattern	3	High
Password	3	High
Electricity Consumption	1	Low
Humidity Level	1	Low

Matrix V presents the ranks of parameters. Mode of all the modes of one particular parameter in the whole matrix T obtained to find the rank of that parameter in the software. Matrix V for this case study is given by Table 5.11.

Table 5.11: Parameter Rank or Matrix V for Smart Building

Parameter	**Mode**	**Rank**
Authentication	3	High
Confidentiality	2	Medium
Integrity	3	High
Non-repudiation	2	Medium
Authorization	3	High

The security ranks of functionalities of the system are given by matrix Q (Table 5.12).

Table 5.12: Functionality Rank or Matrix Q for Smart Building

Functionality	**Mode**	**Rank**
Security System	3	High
Fire Safety System	3	High
Lighting System	2	Medium
HVAC	2	Medium

5.7 Design

As per the ranks obtained in the analysis phase designing is conducted here. Rank diagrams Functionality Rank Diagrams (FRD), Comprehensive Rank Diagram (CRD) and Authorization Rank Diagrams (ARD) are developed. These diagrams have already been discussed in Section 3.7. Each of these diagrams, for the Smart Building System, is elaborated in the following sub-sections.

5.7.1 FRD and CRD

An FRD for the 'Smart Security System' functionality is depicted by Figure 5.18. It is an extension of the CSCD for the same functionality given by Figure 5.16. The assets and stakeholders associated are the same. However, three concentric ovals show that the functionality has high security rank. Similarly, assets in three rectangles like Username and Password have high rank. Table 5.13 is the template for this FRD. The FRDs for all other functionalities are presented by Figures B.1, B.2 and B.3 along with their templates given by Tables B.6, B.7 and B.8 respectively.

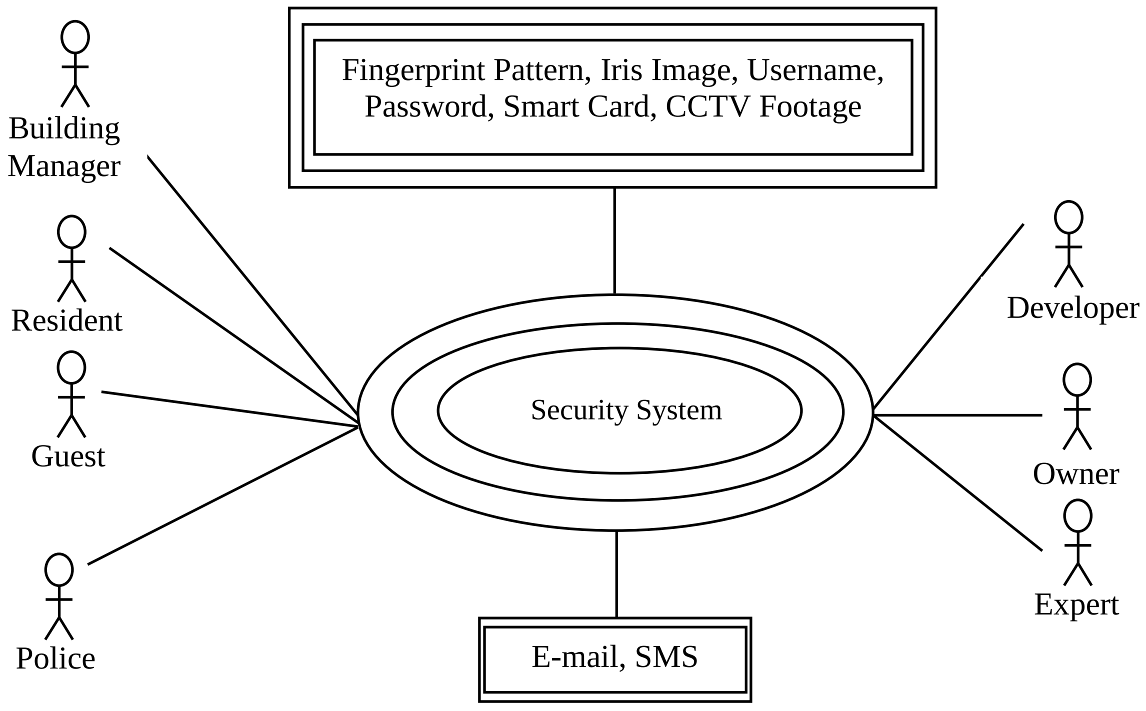


Figure 5.18: FRD for Security System

Table 5.13: Template of FRD for Security System

Functionality: Security System	
Stakeholders	
Name	**Description**
Building Manager	Administrator of the building
Resident	
Guest	Entrant with malicious interests
Police	
Owner	Owner of the building
Developer	
Expert	
Assets	
Name	**Rank**
Fingerprint Pattern	High
Iris Image	High
CCTV Footage	High
Smart Card	High
Username	High
Password	High
SMS	Medium
e-mail	Medium
Functionality Rank: High	
Measure: Fingerprint or Iris Recognition, CCTV Footage, Smart Card for Residents and guests, Username and Password for Guest, block intruders	

Figure 5.19 is a CRD that summarizes the complete software and Table 5.14 is its template.

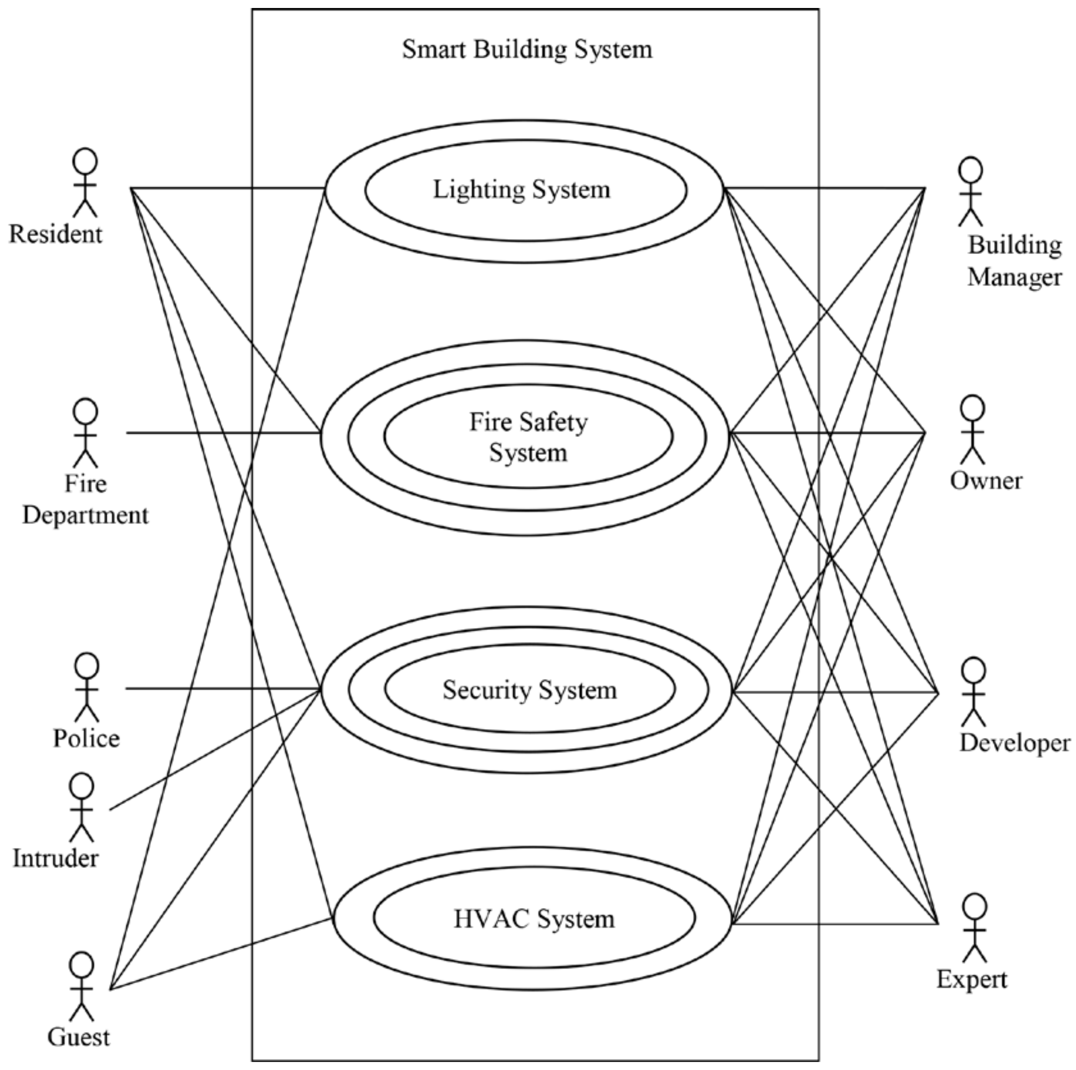


Figure 5.19 : CRD for Smart Building System

Table 5.14: Template for CRD

Smart Building System		
Function List	**Rank**	**Stakeholders Associated**
Security System	High	Building Manager, Resident, Guest, Intruder, Police, Owner, Developer, Expert
Fire Safety System	High	Building Manager, Resident, Fire Department, Owner, Developer, Expert
Lighting System	Medium	Building Manager, Resident, Owner, Guest, Developer, Expert
HVAC System	Medium	Building Manager, Resident, Guest, Owner, Developer, Expert

5.7.2 Authorization Rank Diagram (ARD)

Authorization rank diagrams (ARDs) have been developed which show the authorization right a stakeholder possesses over his/her relevant assets. The ARDs have been explained in Sub-section 3.7.3. Figure 5.20 represents a stakeholder-oriented ARD for Resident. It can be seen in the figure that the Resident has Low rank or ‘Read’ right on asset Room temperature, high rank or ‘Write’ on Password and medium rank or ‘Write with Permission (WP)’ right on Username provided by the Building Manager. Table 5.15 is the template for the diagram.

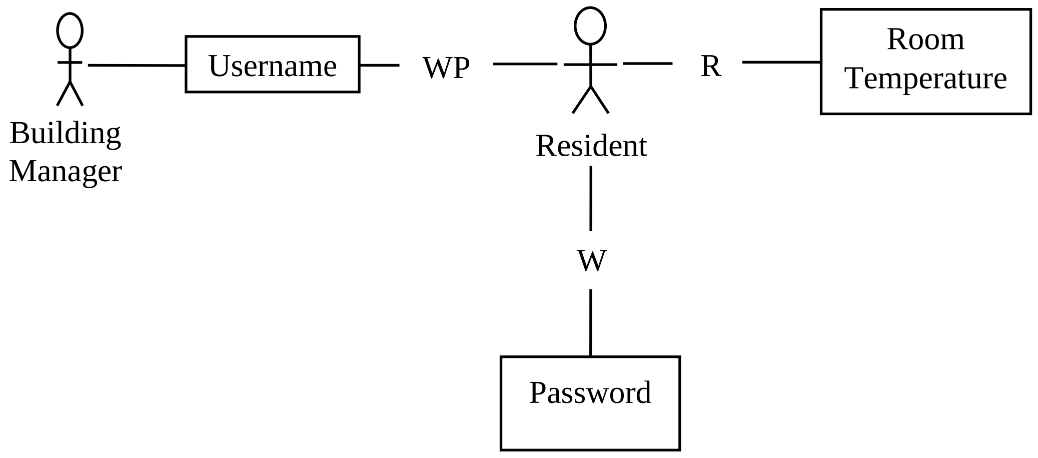


Figure 5.20 : Authorization rights of a stakeholder Resident

Table 5.15: ARD Template for Resident

Stakeholder: INB Officer			
Assets	**Security Rank**	**Authorization Rights**	**Permitting Stakeholders**
Room Temperature	Low	R	
Username	Medium	WP	Building Manager
Password	High	W	

Figure 5.21 shows an asset-oriented ARD for Username. It can be seen that over this asset the stakeholder Building Manager has 'Write' right, Resident has 'Write with Permission (WP)' right provided by Building Manager and Owner has 'Read' right. Table 5.16 is the template for this diagram. More ARDs are presented in Section B.4.

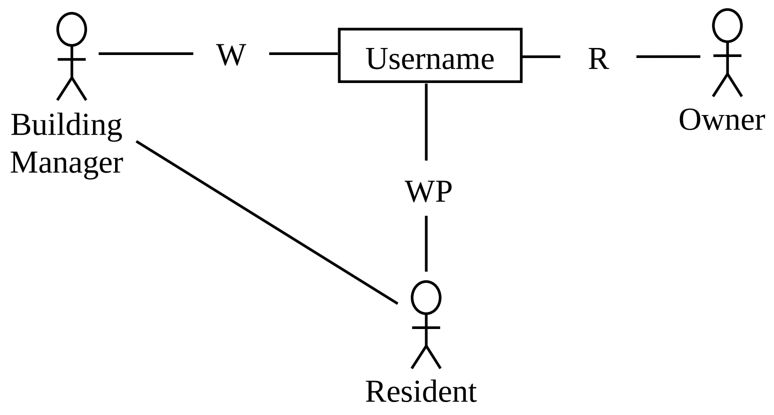


Figure 5.21: Asset-Oriented ARD for Username

Table 5.16: Template for Asset-oriented ARD Template for Username

Asset: Username		
Stakeholder	**Authorization Rights**	**Permitting Stakeholders**
Building Manager	W	
Owner	R	
Resident	WP	Building Manager

5.8 Summary

This chapter successfully demonstrates the application of the proposed methodology in Smart Building System. It is relatively new domain and unique in the sense that the nature of the building and its functionalities are quite diverse. Therefore, putting the security challenges in perspective is a challenge. The residents or the main stakeholders of the building have variety of aspirations from the owner or the manufacturer. The SecREAD methodology is a successful attempt to address these issues through segregation of entities and then dealing with them. At the very beginning of the development process, it identifies all possible stakeholders and then precisely captures and also makes sense of their raw requirements using the concepts of stories and their graphical representations or the SCDs. SecREAD is a systematic way of offering a secure model of Smart Building by considering all facets of security. It provides a robust first line of defense of a smart building *i.e.* authentication of entrants and other stakeholders, and then allotting the minimum authorization rights to the valid entrants on various assets of their use. The confidentiality and integrity level of every asset in the system is established. Also, the level of non-repudiation required for assets in transition is also defined. Rigorous mathematical analysis delivers the overall ranks of the assets and based on that the ranks of different functionalities. The notion of mapping makes the furnishing of ranks by the stakeholders very realistic. At every step, elaborate matrices and diagrams are produced which aid in proper understanding of the flow of the methodology and the final diagrams lucidly put forward the picture of all the intricacies of the Smart Building System. The tool was developed for this research work that automates the process of computations and diagram formation.

Chapter 6

Conclusion

In this modern era, software is an indispensable and inseparable part of our lives. The modern society relies on software. Therefore, it is utmost necessary that software is secure. In this regard requirements elicitation, specification and modeling are the most important elements. Many solutions have been developed to address these issues but have not sufficed. Hence, it is necessary to devise an improved and wholesome methodology for developing secure software that collects the positive aspects of all the prior works and enhances them.

This thesis provides a detailed account of SecREAD, a novel methodology which integrates security in the requirements and design phases. The proposed methodology allows gathering of requirements in natural language or stories that facilitates participation of even the least technical conversant stakeholder in the requirement elicitation process. From these stories different entities are extracted and then mapped to each other to clearly identify the associations between them. To perform this, a concept of Story Conversion Diagrams (SCD) has been envisaged. The methodology is founded on the premise that diagrammatic expression is more suitable than the text for easy understanding by both users and the developers and reduces the chances of ambiguity. Ranking is conducted only by relevant stakeholders. The concepts of mapping and relevance are unique. The diagrams are drawn to model the security requirements based on empirical analysis of ranks. Many useful diagrams have been proposed in this methodology that are evolving in nature that stem from basic SCDs. Use of very less but orthogonal structures render these diagrams simplicity in drawing and understanding without affecting lucidity. While developing these diagrams, the mindset had been to tinker as less as possible with the popular Unified Modeling Language (UML) to make them easy to understand by all.

In the beginning the tone of the research is set and the present state of security in software is underscored. Some basic but important concepts and terms, that are the foundation blocks of the methodology, are introduced. Also, some major conventional or popular development models and methodologies are described in general and critically analyzed to make it easier for the reader to understand the security-oriented models that follow in the next chapter.

As part of research work an intensive and extensive literature survey was carried out. This justifies the need of security in software and how important and useful it is to integrate security in the development process. Then we move on to find the appropriate phases for such an integration and for this the authors deliberate on the views and ideas of many researchers. It is observed that more or less they are in harmony that security should be imbibed in the early phases of development process. However, as discussed in Chapter 2, some advocate such an inclusion in the requirement phase and others in the design phase. In this light, the authors have followed a pacified approach and included security in both the requirements and design phases. Further, many prior methodologies and models pertaining to security in software development are studied and critically analyzed, defining their advantages and limitations. Finally, the inferences are drawn from the literature survey to find the good practices that should be adhered to develop a better methodology.

The Security-aware Requirements Elicitation, Assessment and Design Methodology (SecREAD) is unfolded in the third chapter. The different phases of the methodology are elaborate. In this methodology we have proposed the concept of stories which are the requirements stated in natural language by the stakeholders. This facilitates and ensures that the requirements are elicited easily by all kinds of stakeholders who may or may not be technically conversant. To formalize the requirements process, a concept of entity-sets is introduced. All entities of the software *i.e.* functionalities, assets and stakeholders are identified and refined to remove inconsistencies iteratively and incrementally using a spiral model. To model these requirements effectively a unique diagram is proposed, known as Story Conversion Diagram (SCD). It presents the entities extracted from the stories and the associations among them graphically, which is easily understood by all. Each time any inconsistency or ambiguity in entities is found, it can easily be corrected through these diagrams.

The mapping activity is another unique feature of this methodology. The entities are mapped to each other in order to find the relevance among them. For this, Relevance Matrices are developed. First matrix maps the assets with functionalities and the second maps functionalities with stakeholders. However, the third matrix is produced mathematically by the cross product of first two matrices, resulting in the mapping of assets with stakeholders. In this way, the chances of error are nullified. A fourth relevance matrix maps stakeholders with their relevant parameters. The SCDs are then consolidated, with respect to functionality, to form Comprehensive Story Conversion Diagram (CSCD). The mapping paves the way for a novel ranking method. This ranking method contemplates that the stakeholders rank only their relevant assets on the parameters they are competent for. For example a guest in the smart building is deprived of ranking on the Authorization parameter. These parameters are the outcome of the survey presented in Chapter two. In the ranking process, the stakeholders supply ranks in three levels *i.e.* low, medium and high represented by value 1, 2 and 3 respectively. For this the information in the relevance matrices is combined to form a three dimensional matrix. Using this, a stakeholder can perform the ranking for the entire system through a single customized sheet. Based on the mapping the SCDs are consolidated and CSCDs are developed for each functionality. Further, a rigorous empirical analysis is performed on the ranks which is a novelty that is absent is any of the prior works. Empirical analysis is performed on the ranks to obtain the ranks of individual assets, functionality and the parameters in the software. This information is then added to the CSCDs to obtain the Functionality Rank Diagram (FRD) for each functionality. The FRD alone represents several aspects for a functionality including authentication level, stakeholders and assets involved, and its criticalness. Authorization rank Diagrams (ARD) are also developed. These diagrams are also a novelty that tell the rights a stakeholder holds over an asset. We developed a tool as a part of this research work that inputs the list of entities, develops rank sheets, saves the rank values furnished by different stakeholders, computes the rank values and develops the corresponding diagrams. In other words it automates the mapping, ranking and empirical analysis and generation of diagrams.

The application of the proposed methodology in Internet Banking (INB) software is successfully shown in chapter four. INB system has large security requirements. It is shown that the methodology addresses the security aspects of such a system and it is effective in design and development. The application of SecREAD to INB is quite close to the actual implementations available. FRDs present very clearly security levels of individual functionalities in the internet banking. The ARDs represent the two-way access rights of every stakeholder over different assets involved in the system which are of paramount importance in this domain.

Another case study is taken from a different domain of Internet of Things (IoT). To validate the proposed methodology, Smart Building application is chosen and elaborated in chapter five. IoT is very important in modern era and software should be IoT compliant. It is successfully demonstrated that proposed methodology can ensure safety requirements of such systems and can help in reducing the risk of security breach.

SecREAD is unique in several ways than its predecessors. The formation of a comprehensive consultation group takes SecREAD ahead from the existing methodologies. SecREAD ranks both assets and functionalities over security parameters which has not been done so far. Furthermore, use of very less but orthogonal structures renders these diagrams simplicity in drawing and understanding without affecting lucidity. All in all, the hallmark of the methodology lies in active involvement of stakeholders, ranking by only relevant stakeholders and empirical analysis, which is quite rare in this domain of research. The methodology is quite flexible to cater to software of different domains. It embraces both technically-aware and common stakeholders. Ranking is done by only those stakeholders relevant to them. This notion of relevance is unique to all prior methodologies.

The diagrams are coupled with templates to provide an unambiguous view. Furthermore, these are simple to draw and understand. All the good practices, of related prior works, have been incorporated in SecREAD. The provision of assigning weights to stakeholders or parameters makes it more realistic and flexible.

6.1 Contributions of SecREAD Methodology

Security-aware Requirements Elicitation, Assessment and Design (SecREAD) Methodology has following contributions:

- The methodology formalizes the processes suggested by the existing related works.
- The methodology is unique as it takes into account both the popular views of researchers *i.e.* security is integrated in both requirements and design phases.

- A method is proposed to elicit requirements in the form of stories in natural language from all kind of stakeholders.
- The notion of Story Conversion Diagrams (SCDs) is proposed which formally represent the requirements elicited from users. It is the basic diagram from which complex diagrams evolve incrementally. Diagrams are drawn in stages as more and more information is gathered.
- Effectiveness of SCDs is demonstrated even in identification and refinement.
- The diagrams are coupled with textual templates to avoid any ambiguity.
- The diagrams proposed are quite close to the popular UML diagrams, making them easier to draw and understand.
- It assimilates the views of all kinds of stakeholders of the system.
- A ranking method is proposed in which entities are ranked over apt security parameters.
- The ranking process is based on an elaborate mapping mechanism based on the concept of relevance *i.e.* software entities are ranked by relevant stakeholders only on the parameters of their expertise.
- Although ranking process includes an intricate mapping procedure, ranking itself is easy and practical for the stakeholders of all kind, both technically conversant and naive.
- Empirical analysis is proposed and included in the framework of this methodology.
- Design process to reflect security concerns is in unison with the ranking process.
- SecREAD combines all of the best practices like iteration in requirement elicitation [54][73], involvement of a representative from client side in the development team [50], communication within the development team and between teams [81] and involvement of stakeholders in the development of software [69].

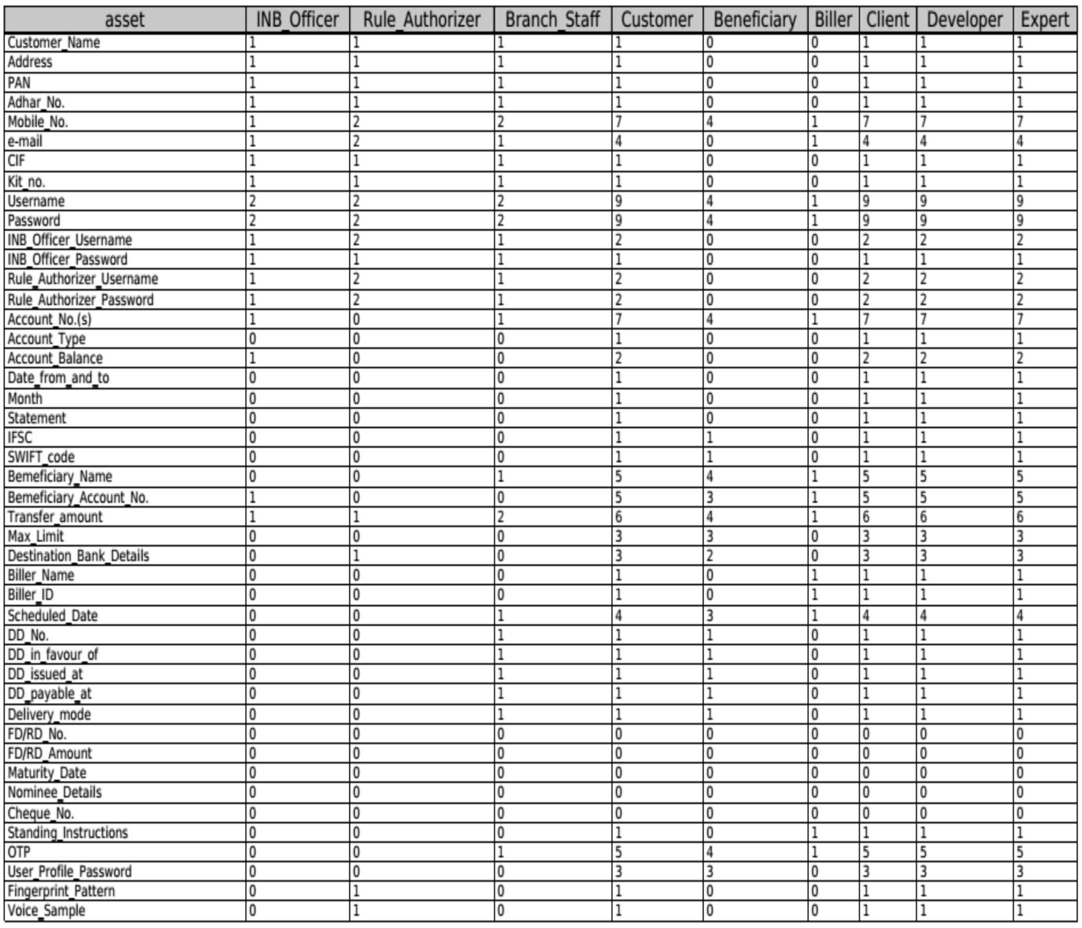
6.2 Future Work

In future, the methodology can be improved by adding more parameters. The methodology can be made more domain specific by assigning weights to rank values furnished by certain stakeholders. The tool developed can be enhanced. The methodology can be validated for different applications by considering case studies from other domains.

Appendix A: Results of Internet Banking Case Study

A.1 Relevance Matrices

Table A.1: Asset-Stakeholder Relevance Matrix Z for INB System



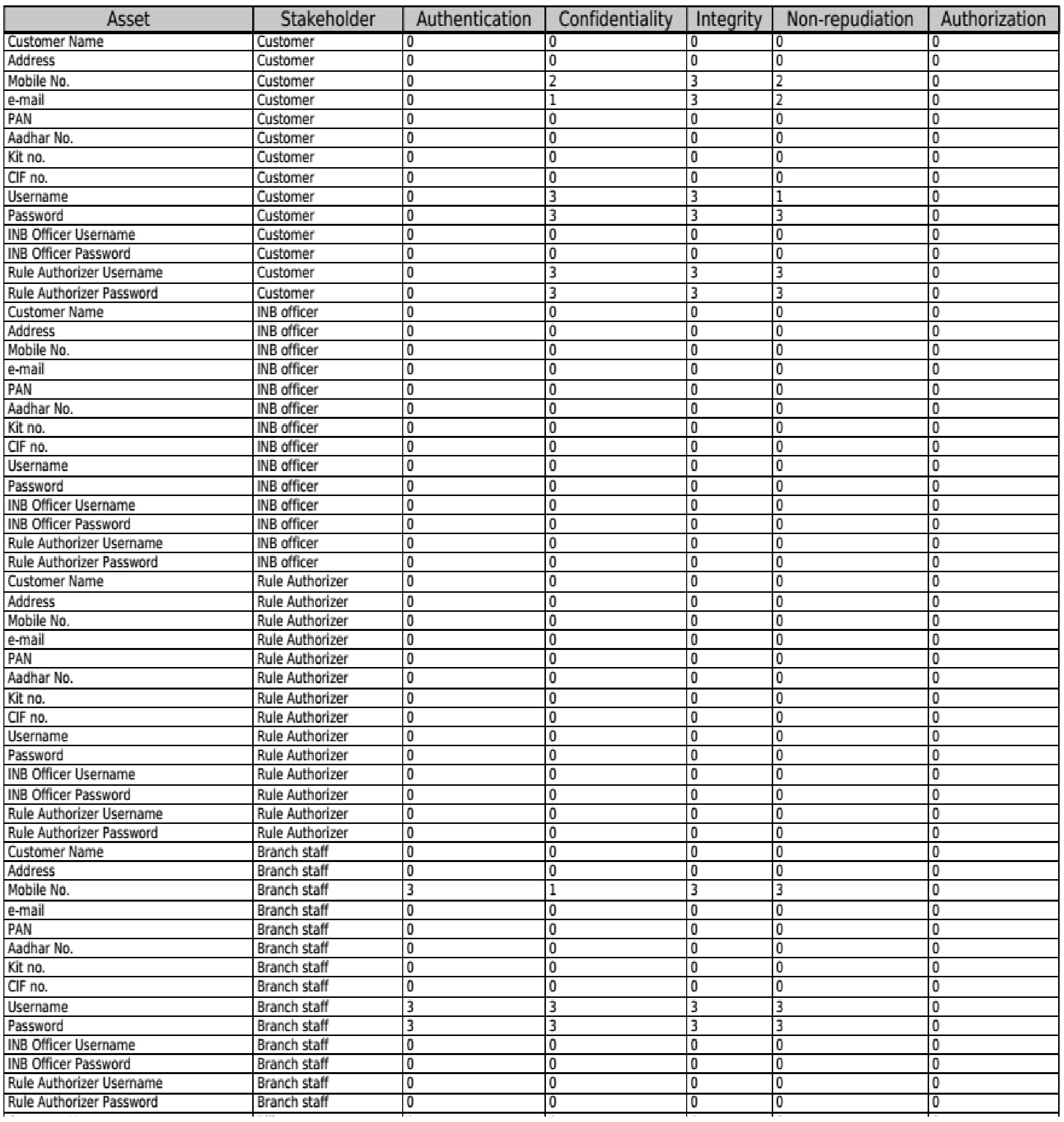
asset	INB_Officer	Rule_Authorizer	Branch_Staff	Customer	Beneficiary	Biller	Client	Developer	Expert
Customer_Name	1	1	1	1	0	0	1	1	1
Address	1	1	1	1	0	0	1	1	1
PAN	1	1	1	1	0	0	1	1	1
Adhar_No.	1	1	1	1	0	0	1	1	1
Mobile_No.	1	2	2	7	4	1	7	7	7
e-mail	1	2	1	4	0	1	4	4	4
CIF	1	1	1	1	0	0	1	1	1
Kit_no.	1	1	1	1	0	0	1	1	1
Username	2	2	2	9	4	1	9	9	9
Password	2	2	2	9	4	1	9	9	9
INB_Officer_Username	1	2	1	2	0	0	2	2	2
INB_Officer_Password	1	1	1	1	0	0	1	1	1
Rule_Authorizer_Username	1	2	1	2	0	0	2	2	2
Rule_Authorizer_Password	1	2	1	2	0	0	2	2	2
Account_No.(s)	1	0	1	7	4	1	7	7	7
Account_Type	0	0	0	1	0	0	1	1	1
Account_Balance	1	0	0	2	0	0	2	2	2
Date_from_and_to	0	0	0	1	0	0	1	1	1
Month	0	0	0	1	0	0	1	1	1
Statement	0	0	0	1	0	0	1	1	1
IFSC	0	0	0	1	1	0	1	1	1
SWIFT_code	0	0	0	1	1	0	1	1	1
Bemeficiary_Name	0	0	1	5	4	1	5	5	5
Bemeficiary_Account_No.	1	0	0	5	3	1	5	5	5
Transfer_amount	1	1	2	6	4	1	6	6	6
Max_Limit	0	0	0	3	3	0	3	3	3
Destination_Bank_Details	0	1	0	3	2	0	3	3	3
Biller_Name	0	0	0	1	0	1	1	1	1
Biller_ID	0	0	0	1	0	1	1	1	1
Scheduled_Date	0	0	1	4	3	1	4	4	4
DD_No.	0	0	1	1	1	0	1	1	1
DD_in_favour_of	0	0	1	1	1	0	1	1	1
DD_issued_at	0	0	1	1	1	0	1	1	1
DD_payable_at	0	0	1	1	1	0	1	1	1
Delivery_mode	0	0	1	1	1	0	1	1	1
FD/RD_No.	0	0	0	0	0	0	0	0	0
FD/RD_Amount	0	0	0	0	0	0	0	0	0
Maturity_Date	0	0	0	0	0	0	0	0	0
Nominee_Details	0	0	0	0	0	0	0	0	0
Cheque_No.	0	0	0	0	0	0	0	0	0
Standing_Instructions	0	0	0	1	0	1	1	1	1
OTP	0	0	1	5	4	1	5	5	5
User_Profile_Password	0	0	0	3	3	0	3	3	3
Fingerprint_Pattern	0	1	0	1	0	0	1	1	1
Voice_Sample	0	1	0	1	0	0	1	1	1

Table A.2: Asset-Functionality Relevance Matrix X for INB System

asset	Issue_Kit	Login	View_Account	Print_Statement	Intra_Bank_Funds_Transfer	Inter_Bank_Funds_Transfer	International_Funds_Transfer	DD_Issue	Bill_Payment	FD/RD_Creation	Cheque_Payment_Stop
Customer_Name	1	0	0	0	0	0	0	0	0	0	0
Address	1	0	0	0	0	0	0	0	0	0	0
PAN	1	0	0	0	0	0	0	0	0	1	0
Adhar_No.	1	0	0	0	0	0	0	0	0	1	0
Mobile_No.	1	1	0	0	1	1	1	1	1	1	1
e-mail	1	1	0	1	0	0	0	0	1	1	1
CIF	1	0	0	0	0	0	0	0	0	0	0
Kit_no.	1	0	0	0	0	0	0	0	0	0	0
Username	1	1	1	1	1	1	1	1	1	1	1
Password	1	1	1	1	1	1	1	1	1	1	1
INB_Officer_Username	1	1	0	0	0	0	0	0	0	0	0
INB_Officer_Password	1	0	0	0	0	0	0	0	0	0	0
Rule_Authorizer_Username	1	1	0	0	0	0	0	0	0	0	0
Rule_Authorizer_Password	1	1	0	0	0	0	0	0	0	0	0
Account_No.(s)	0	0	1	1	1	1	1	1	1	1	0
Account_Type	0	0	0	1	0	0	0	0	0	0	0
Account_Balance	0	0	1	1	0	0	0	0	0	0	0
Date_from_and_to	0	0	0	1	0	0	0	0	0	0	0
Month	0	0	0	1	0	0	0	0	0	0	0
Statement	0	0	0	1	0	0	0	0	0	0	0
IFSC	0	0	0	0	0	1	0	0	0	0	0
SWIFT_code	0	0	0	0	0	0	1	0	0	0	0
Bemeficiary_Name	0	0	0	0	1	1	1	1	1	0	0
Bemeficiary_Account_No.	0	0	1	0	1	1	1	0	1	0	0
Transfer_amount	1	0	0	0	1	1	1	1	1	0	0
Max_Limit	0	0	0	0	1	1	1	0	0	0	0
Destination_Bank_Details	0	1	0	0	0	1	1	0	0	0	0
Biller_Name	0	0	0	0	0	0	0	0	1	0	0
Biller_ID	0	0	0	0	0	0	0	0	1	0	0
Scheduled_Date	0	0	0	0	1	1	0	1	1	0	0
DD_No.	0	0	0	0	0	0	0	1	0	0	0
DD_in_favour_of	0	0	0	0	0	0	0	1	0	0	0
DD_issued_at	0	0	0	0	0	0	0	1	0	0	0
DD_payable_at	0	0	0	0	0	0	0	1	0	0	0
Delivery_mode	0	0	0	0	0	0	0	1	0	0	0
FD/RD_No.	0	0	0	0	0	0	0	0	0	1	0
FD/RD_Amount	0	0	0	0	0	0	0	0	0	1	0
Maturity_Date	0	0	0	0	0	0	0	0	0	1	0
Nominee_Details	0	0	0	0	0	0	0	0	0	1	0
Cheque_No.	0	0	0	0	0	0	0	0	0	0	1
Standing_Instructions	0	0	0	0	0	0	0	0	1	1	0
OTP	0	0	0	0	1	1	1	1	1	0	0
User_Profile_Password	0	0	0	0	1	1	1	0	0	0	0
Fingerprint_Pattern	0	1	0	0	0	0	0	0	0	0	0
Voice_Sample	0	1	0	0	0	0	0	0	0	0	0

A.2 Ranking and Results

Table A.3: Rank Matrix R for INB System

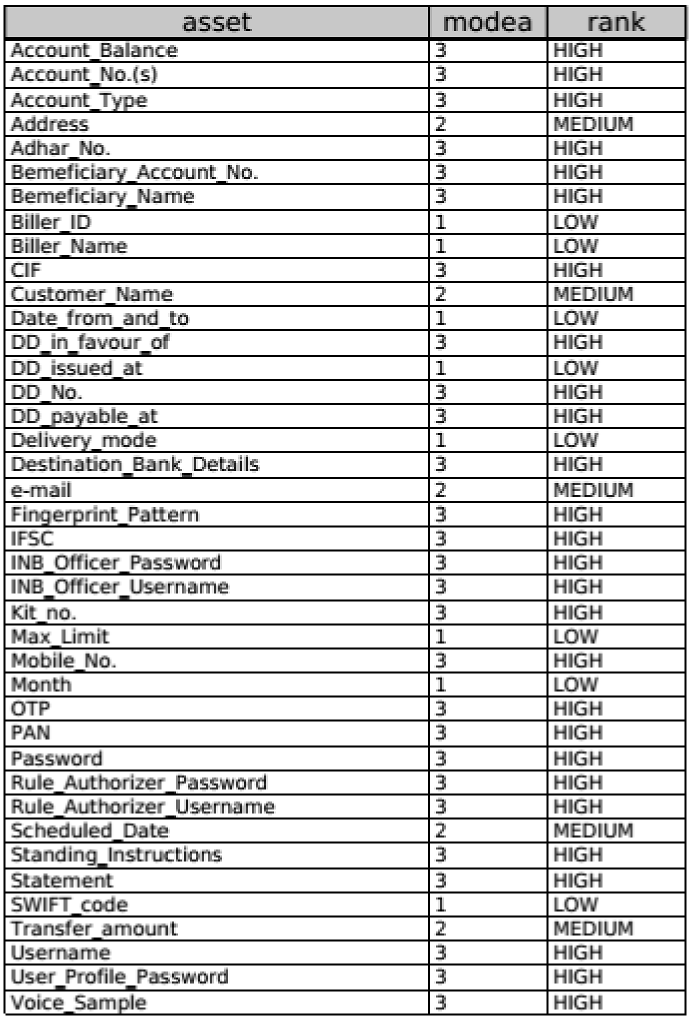


Asset	Stakeholder	Authentication	Confidentiality	Integrity	Non-repudiation	Authorization
Customer Name	Customer	0	0	0	0	0
Address	Customer	0	0	0	0	0
Mobile No.	Customer	0	2	3	2	0
e-mail	Customer	0	1	3	2	0
PAN	Customer	0	0	0	0	0
Aadhar No.	Customer	0	0	0	0	0
Kit no.	Customer	0	0	0	0	0
CIF no.	Customer	0	0	0	0	0
Username	Customer	0	3	3	1	0
Password	Customer	0	3	3	3	0
INB Officer Username	Customer	0	0	0	0	0
INB Officer Password	Customer	0	0	0	0	0
Rule Authorizer Username	Customer	0	3	3	3	0
Rule Authorizer Password	Customer	0	3	3	3	0
Customer Name	INB officer	0	0	0	0	0
Address	INB officer	0	0	0	0	0
Mobile No.	INB officer	0	0	0	0	0
e-mail	INB officer	0	0	0	0	0
PAN	INB officer	0	0	0	0	0
Aadhar No.	INB officer	0	0	0	0	0
Kit no.	INB officer	0	0	0	0	0
CIF no.	INB officer	0	0	0	0	0
Username	INB officer	0	0	0	0	0
Password	INB officer	0	0	0	0	0
INB Officer Username	INB officer	0	0	0	0	0
INB Officer Password	INB officer	0	0	0	0	0
Rule Authorizer Username	INB officer	0	0	0	0	0
Rule Authorizer Password	INB officer	0	0	0	0	0
Customer Name	Rule Authorizer	0	0	0	0	0
Address	Rule Authorizer	0	0	0	0	0
Mobile No.	Rule Authorizer	0	0	0	0	0
e-mail	Rule Authorizer	0	0	0	0	0
PAN	Rule Authorizer	0	0	0	0	0
Aadhar No.	Rule Authorizer	0	0	0	0	0
Kit no.	Rule Authorizer	0	0	0	0	0
CIF no.	Rule Authorizer	0	0	0	0	0
Username	Rule Authorizer	0	0	0	0	0
Password	Rule Authorizer	0	0	0	0	0
INB Officer Username	Rule Authorizer	0	0	0	0	0
INB Officer Password	Rule Authorizer	0	0	0	0	0
Rule Authorizer Username	Rule Authorizer	0	0	0	0	0
Rule Authorizer Password	Rule Authorizer	0	0	0	0	0
Customer Name	Branch staff	0	0	0	0	0
Address	Branch staff	0	0	0	0	0
Mobile No.	Branch staff	3	1	3	3	0
e-mail	Branch staff	0	0	0	0	0
PAN	Branch staff	0	0	0	0	0
Aadhar No.	Branch staff	0	0	0	0	0
Kit no.	Branch staff	0	0	0	0	0
CIF no.	Branch staff	0	0	0	0	0
Username	Branch staff	3	3	3	3	0
Password	Branch staff	3	3	3	3	0
INB Officer Username	Branch staff	0	0	0	0	0
INB Officer Password	Branch staff	0	0	0	0	0
Rule Authorizer Username	Branch staff	0	0	0	0	0
Rule Authorizer Password	Branch staff	0	0	0	0	0

Table A.4: Matrix T for INB System

Asset	Parameter	Mode
Account_Balance	Authentication	3
Account_Balance	Authorization	3
Account_Balance	Confidentiality	2
Account_Balance	Integrity	3
Account_Balance	Non-repudiation	3
Account_No.(s)	Authentication	3
Account_No.(s)	Authorization	3
Account_No.(s)	Confidentiality	2
Account_No.(s)	Integrity	3
Account_No.(s)	Non-repudiation	3
Account_Type	Authentication	3
Account_Type	Authorization	2
Account_Type	Confidentiality	3
Account_Type	Integrity	3
Account_Type	Non-repudiation	1
Address	Authentication	2
Address	Authorization	2
Address	Confidentiality	1
Address	Integrity	3
Address	Non-repudiation	2
Adhar_No.	Authentication	3
Adhar_No.	Authorization	2
Adhar_No.	Confidentiality	1
Adhar_No.	Integrity	3
Adhar_No.	Non-repudiation	2
Bemeficiary_Account_No.	Authentication	3
Bemeficiary_Account_No.	Authorization	3
Bemeficiary_Account_No.	Confidentiality	1
Bemeficiary_Account_No.	Integrity	3
Bemeficiary_Account_No.	Non-repudiation	3
Bemeficiary_Name	Authentication	2
Bemeficiary_Name	Authorization	3
Bemeficiary_Name	Confidentiality	3
Bemeficiary_Name	Integrity	3
Bemeficiary_Name	Non-repudiation	3
Biller_ID	Authentication	1
Biller_ID	Authorization	1
Biller_ID	Confidentiality	1
Biller_ID	Integrity	3
Biller_ID	Non-repudiation	3
Biller_Name	Authentication	1
Biller_Name	Authorization	1
Biller_Name	Confidentiality	1
Biller_Name	Integrity	3
Biller_Name	Non-repudiation	3
CIF	Authentication	3
CIF	Authorization	3
CIF	Confidentiality	3
CIF	Integrity	3
CIF	Non-repudiation	3
Customer_Name	Authentication	2
Customer_Name	Authorization	2
Customer_Name	Confidentiality	1
Customer_Name	Integrity	3
Customer_Name	Non-repudiation	1

Table A.5: Matrix U or Asset Rank for INB System



asset	modea	rank
Account_Balance	3	HIGH
Account_No.(s)	3	HIGH
Account_Type	3	HIGH
Address	2	MEDIUM
Adhar_No.	3	HIGH
Bemeficiary_Account_No.	3	HIGH
Bemeficiary_Name	3	HIGH
Biller_ID	1	LOW
Biller_Name	1	LOW
CIF	3	HIGH
Customer_Name	2	MEDIUM
Date_from_and_to	1	LOW
DD_in_favour_of	3	HIGH
DD_issued_at	1	LOW
DD_No.	3	HIGH
DD_payable_at	3	HIGH
Delivery_mode	1	LOW
Destination_Bank_Details	3	HIGH
e-mail	2	MEDIUM
Fingerprint_Pattern	3	HIGH
IFSC	3	HIGH
INB_Officer_Password	3	HIGH
INB_Officer_Username	3	HIGH
Kit_no.	3	HIGH
Max_Limit	1	LOW
Mobile_No.	3	HIGH
Month	1	LOW
OTP	3	HIGH
PAN	3	HIGH
Password	3	HIGH
Rule_Authorizer_Password	3	HIGH
Rule_Authorizer_Username	3	HIGH
Scheduled_Date	2	MEDIUM
Standing_Instructions	3	HIGH
Statement	3	HIGH
SWIFT_code	1	LOW
Transfer_amount	2	MEDIUM
Username	3	HIGH
User_Profile_Password	3	HIGH
Voice_Sample	3	HIGH

A.3 Functionality Rank Diagrams (FRD)

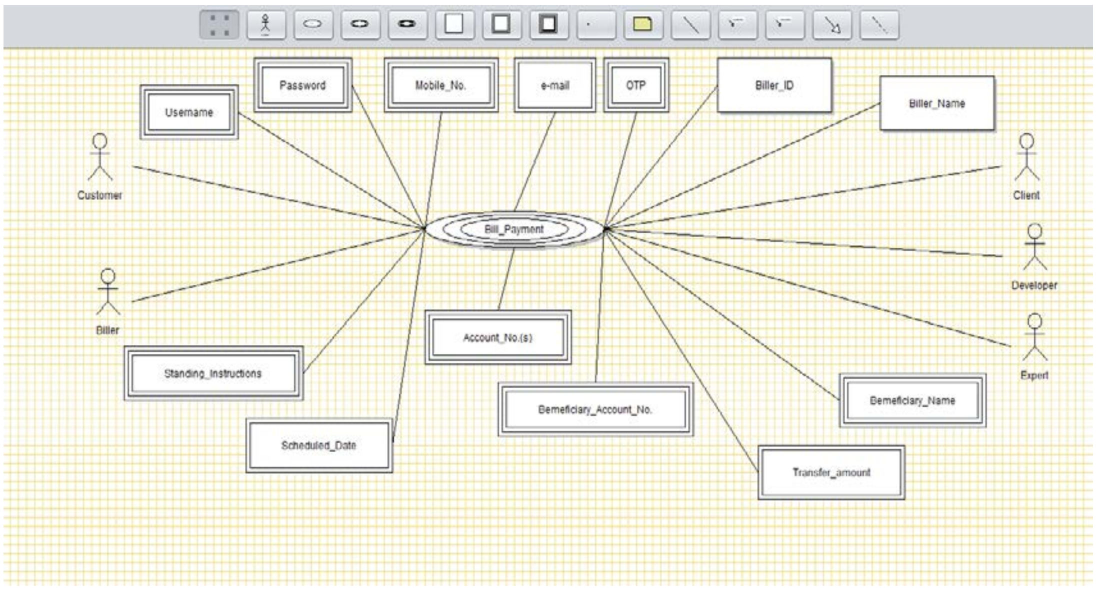


Figure A.1: FRD for Bill Payment

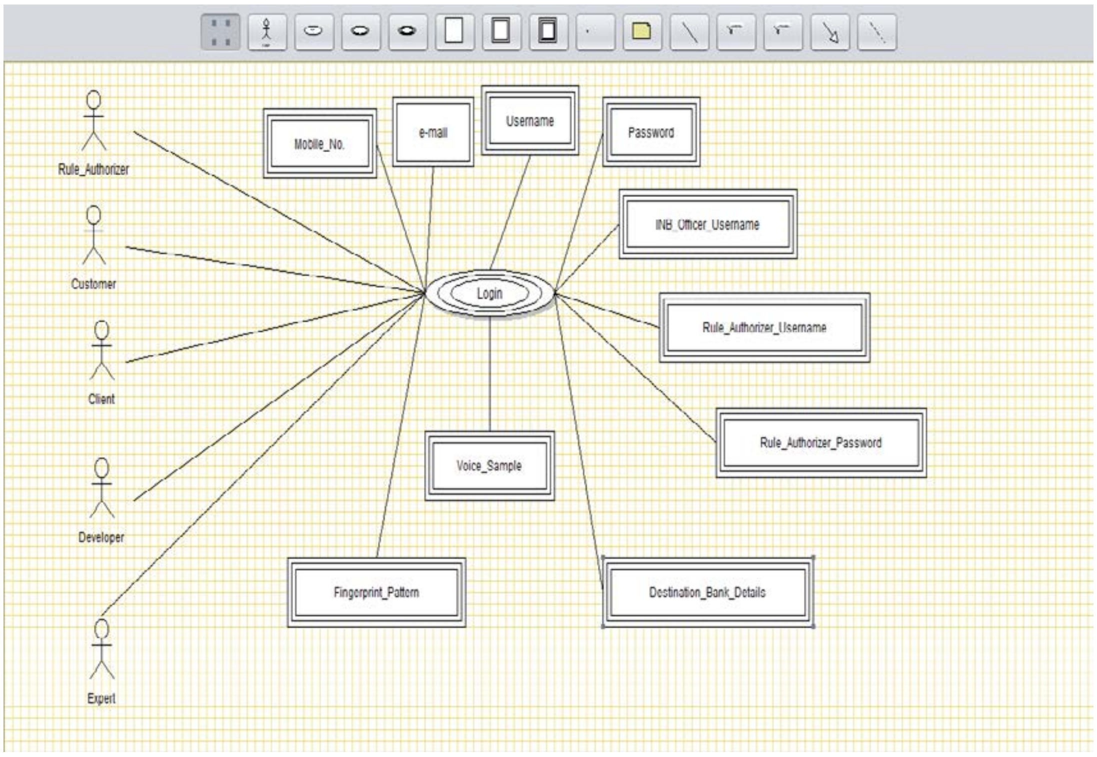


Figure A.2: FRD for Login

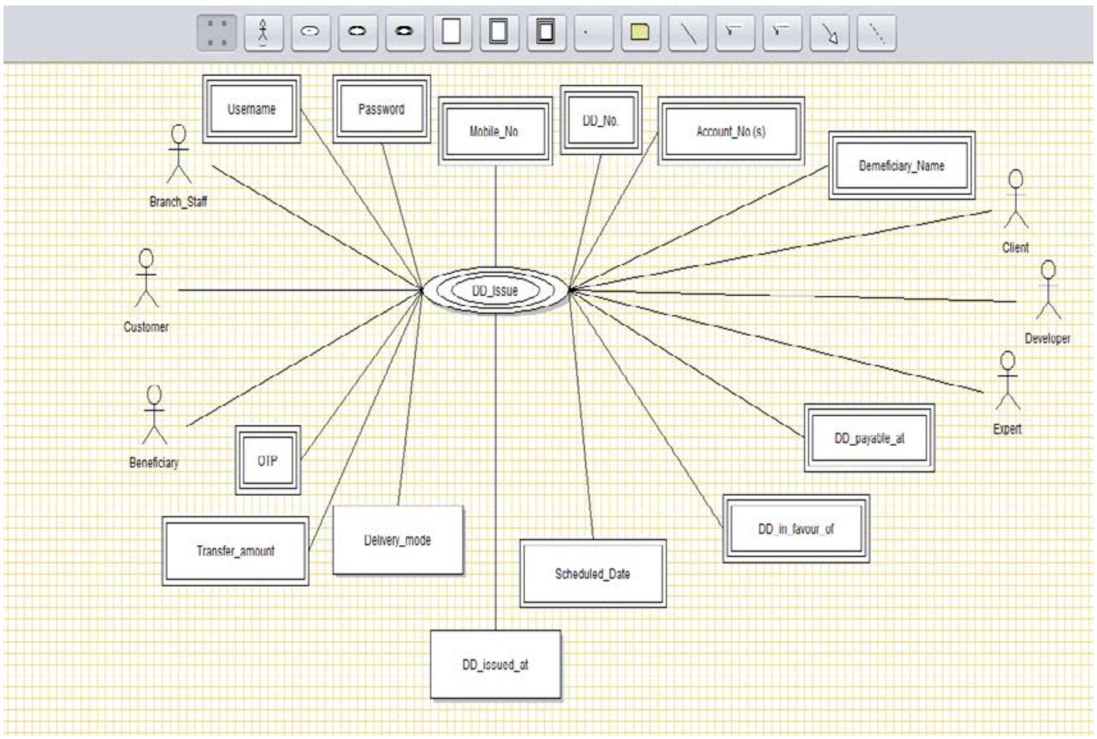


Figure A.3: FRD for DD Issue

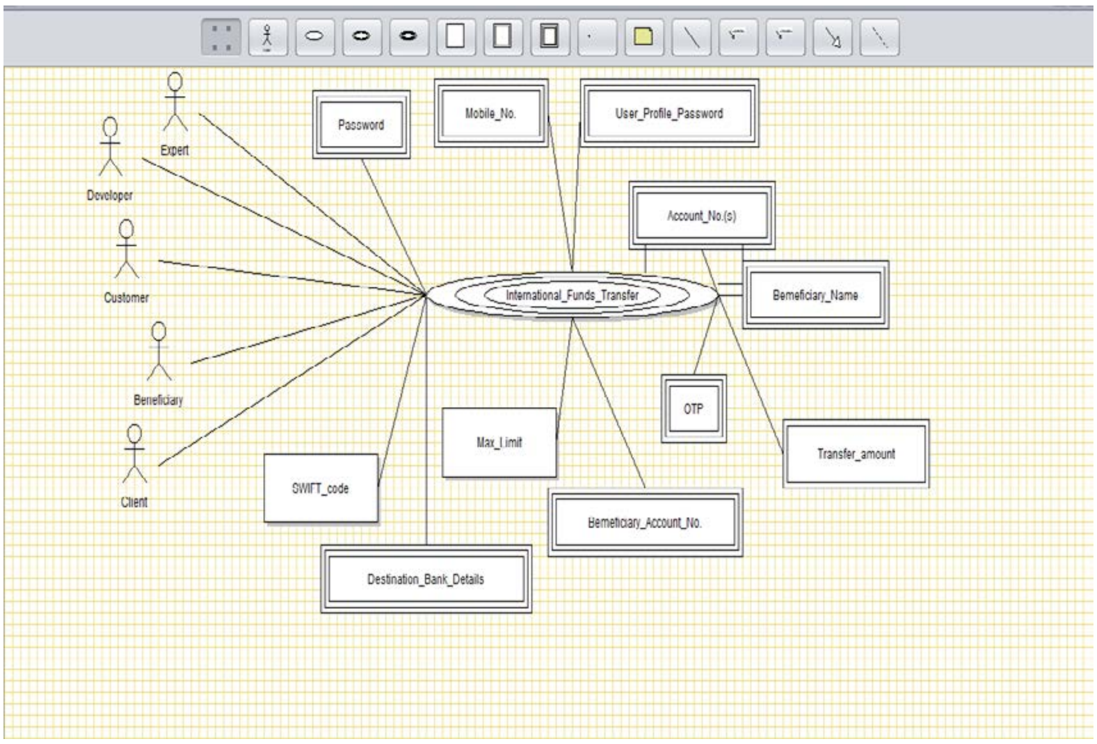


Figure A.4: FRD for International Funds Transfer

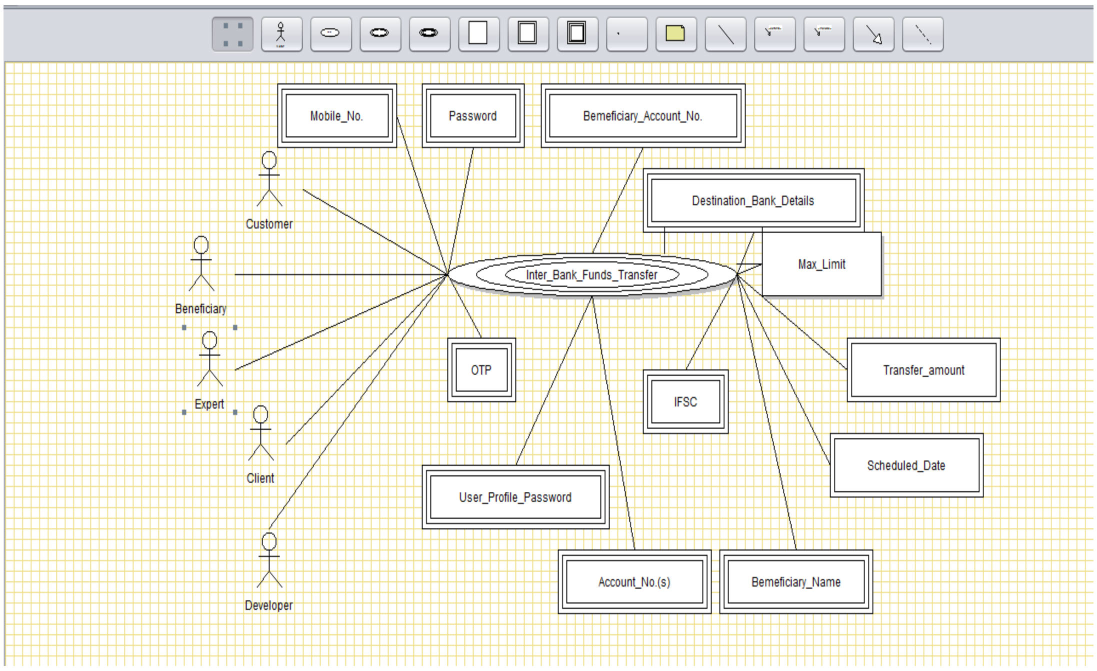


Figure A.5: FRD for Inter-bank Funds Transfer

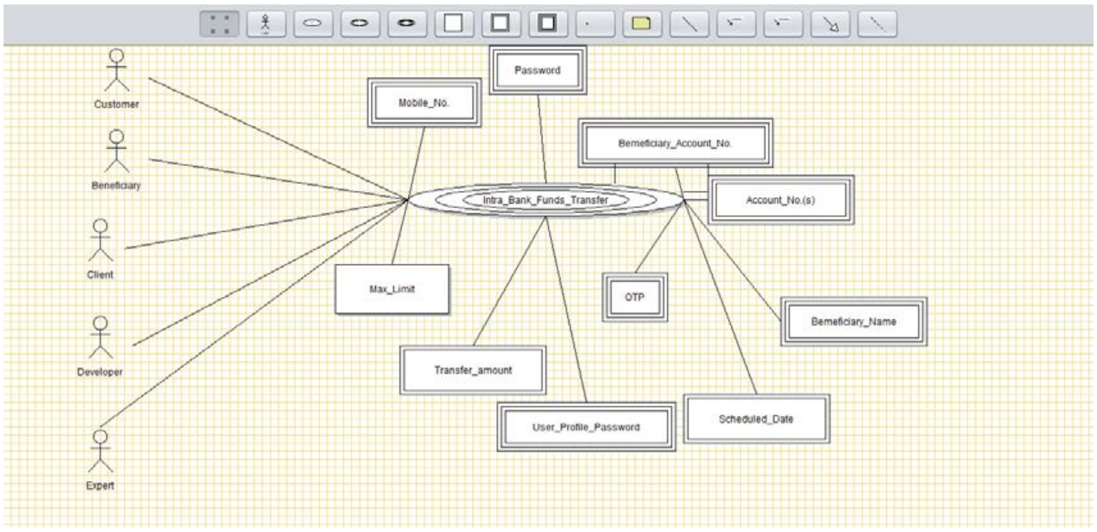


Figure A.6: FRD for Intra-bank Funds Transfer

A.4 Authorization Rank Diagrams (ARD) for INB System

Figure A.8 is a stakeholder-centric ARD for stakeholder ‘Customer’. It has Write with Permission (WP) right on asset ‘Password’ permitted by ‘Rule Authorizer’. He holds ‘Read’ right on CIF and ‘Write’ right on Address. Table A.6 is the template for this diagram.

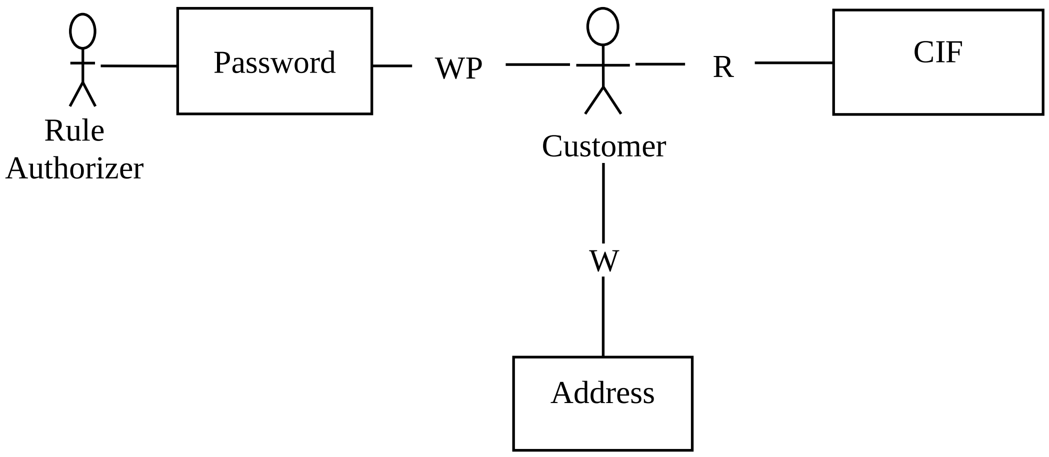


Figure A.7: ARD for Customer

Table A.6: ARD Template for Customer

Assets	**Authorization Rights**	**Permitting Stakeholders**
Password	WP	Rule Authorizer
CIF	R	
Address	W	

Figure A.9 shows an asset-oriented ARD for Username. It can be seen that over this asset the stakeholder Building Manager has ‘Write’ right, Resident has ‘Write with Permission (WP)’ right provided by Building Manager and Owner has ‘Read’ right. Table A.7 is the template for this diagram.

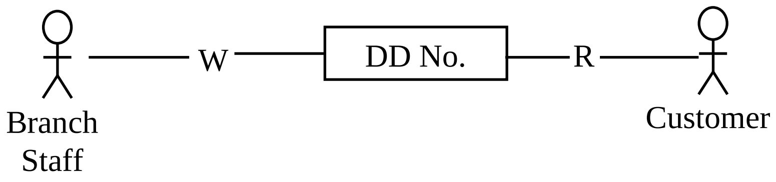


Figure A.8: Asset-Oriented ARD for Username

Table A.7: Template for Asset-oriented ARD Template for Username

Asset: Username		
Stakeholder	**Authorization Rights**	**Permitting Stakeholders**
Building Manager	W	
Owner	R	
Resident	WP	Building Manager

Appendix B: Results of Smart Building Case Study

B.1 Relevance Matrices

Table B.1: Asset-Functionality Relevance Matrix X

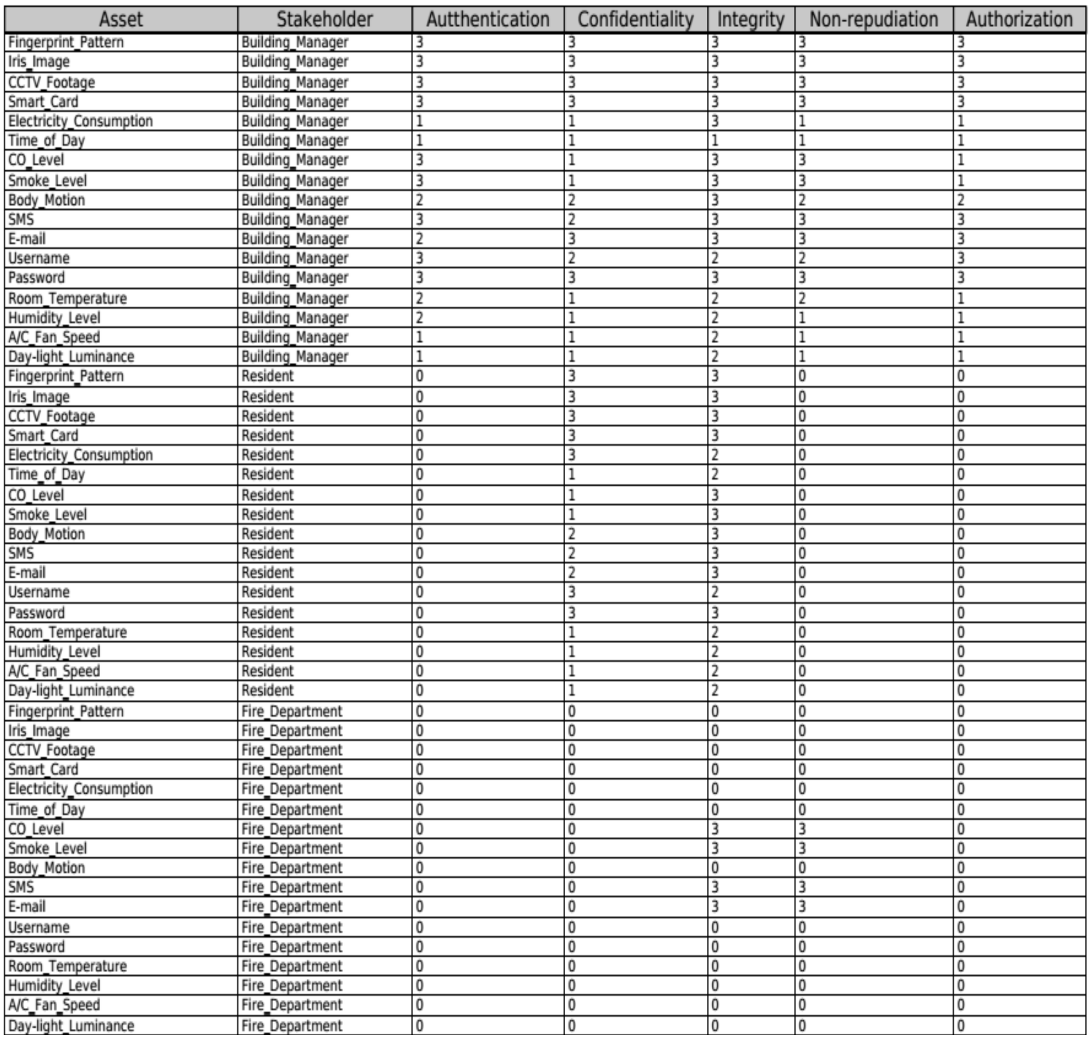
Asset	Smart_Lighting	Smart_Fire_Alarm_System	Smart_Security_System	HVAC
Fingerprint_Pattern	0	0	1	0
Iris_Image	0	0	1	0
CCTV_Footage	0	0	1	0
Smart_Card	0	0	1	0
Electricity_Consumption	1	0	0	1
Time_of_Day	1	0	0	1
CO_Level	0	1	0	0
Smoke_Level	0	1	0	1
Body_Motion	1	0	0	0
SMS	1	1	1	1
E-mail	1	1	1	1
Username	0	0	1	0
Password	0	0	1	0
Room_Temperature	0	0	0	1
Humidity_Level	0	0	0	1
A/C_Fan_Speed	0	0	0	1
Day-light_Luminance	1	0	0	0

Table B.2: Asset-Stakeholder Relevance Matrix Z

Asset	Building_Manager	Resident	Intruder	Fire_Department	Police	Owner	Developer	Expert
Fingerprint_Pattern	1	1	1	0	1	1	1	1
Iris_Image	1	1	1	0	1	1	1	1
CCTV_Footage	1	1	1	0	1	1	1	1
Smart_Card	1	1	1	0	1	1	1	1
Electricity_Consumption	2	2	0	0	0	2	2	2
Time_of_Day	2	2	0	0	0	2	2	2
CO_Level	1	1	0	1	0	1	1	1
Smoke_Level	2	2	0	1	0	2	2	2
Body_Motion	1	1	0	0	0	1	1	1
SMS	4	4	1	1	1	4	4	4
E-mail	4	4	1	1	1	4	4	4
Username	1	1	1	0	1	1	1	1
Password	1	1	1	0	1	1	1	1
Room_Temperature	1	1	0	0	0	1	1	1
Humidity_Level	1	1	0	0	0	1	1	1
A/C_Fan_Speed	1	1	0	0	0	1	1	1
Day-light_Luminance	1	1	0	0	0	1	1	1

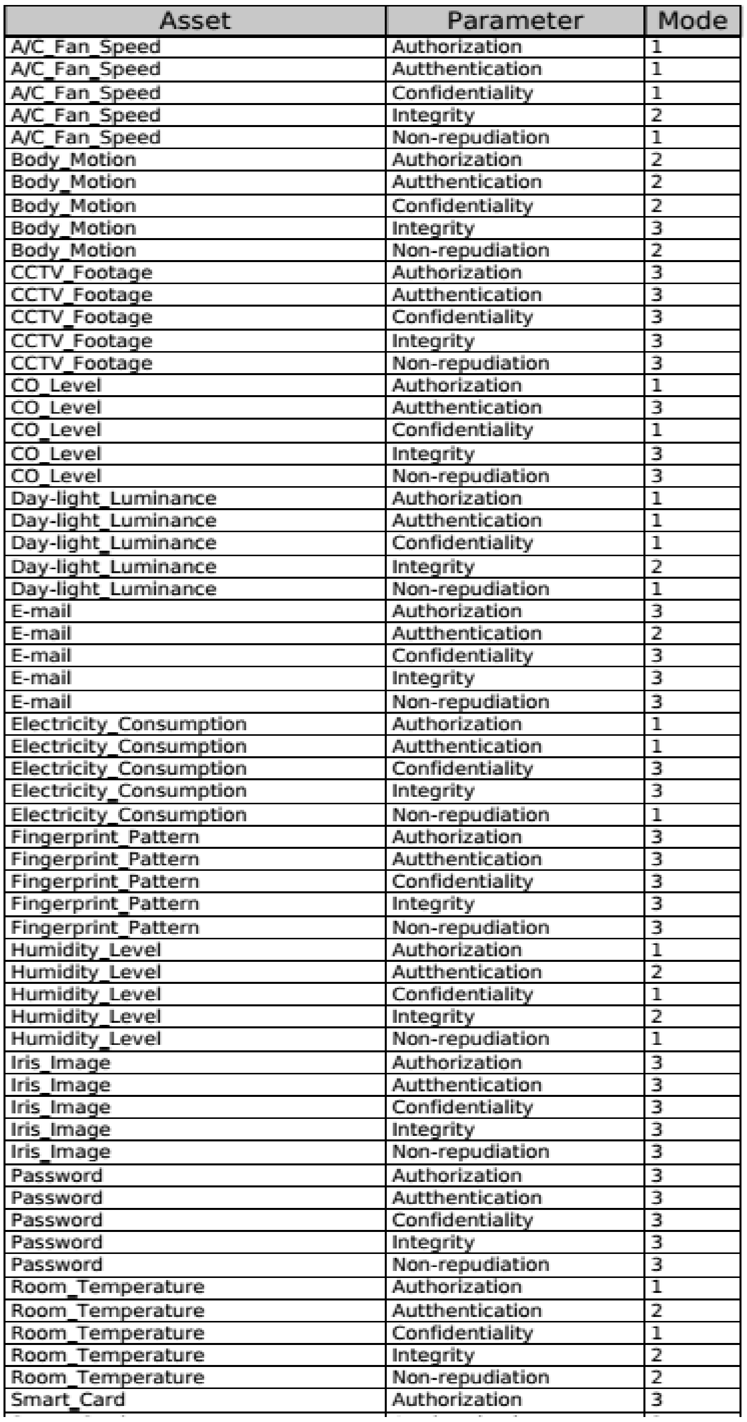
B.2 Ranking and Results

Table B.3: Rank Matrix R for Smart Building System



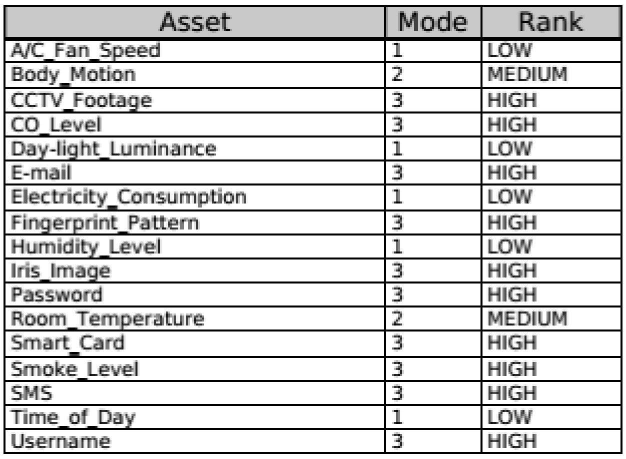
Asset	Stakeholder	Autthentication	Confidentiality	Integrity	Non-repudiation	Authorization
Fingerprint_Pattern	Building_Manager	3	3	3	3	3
Iris_Image	Building_Manager	3	3	3	3	3
CCTV_Footage	Building_Manager	3	3	3	3	3
Smart_Card	Building_Manager	3	3	3	3	3
Electricity_Consumption	Building_Manager	1	1	3	1	1
Time_of_Day	Building_Manager	1	1	1	1	1
CO_Level	Building_Manager	3	1	3	3	1
Smoke_Level	Building_Manager	3	1	3	3	1
Body_Motion	Building_Manager	2	2	3	2	2
SMS	Building_Manager	3	2	3	3	3
E-mail	Building_Manager	2	3	3	3	3
Username	Building_Manager	3	2	2	2	3
Password	Building_Manager	3	3	3	3	3
Room_Temperature	Building_Manager	2	1	2	2	1
Humidity_Level	Building_Manager	2	1	2	1	1
A/C_Fan_Speed	Building_Manager	1	1	2	1	1
Day-light_Luminance	Building_Manager	1	1	2	1	1
Fingerprint_Pattern	Resident	0	3	3	0	0
Iris_Image	Resident	0	3	3	0	0
CCTV_Footage	Resident	0	3	3	0	0
Smart_Card	Resident	0	3	3	0	0
Electricity_Consumption	Resident	0	3	2	0	0
Time_of_Day	Resident	0	1	2	0	0
CO_Level	Resident	0	1	3	0	0
Smoke_Level	Resident	0	1	3	0	0
Body_Motion	Resident	0	2	3	0	0
SMS	Resident	0	2	3	0	0
E-mail	Resident	0	2	3	0	0
Username	Resident	0	3	2	0	0
Password	Resident	0	3	3	0	0
Room_Temperature	Resident	0	1	2	0	0
Humidity_Level	Resident	0	1	2	0	0
A/C_Fan_Speed	Resident	0	1	2	0	0
Day-light_Luminance	Resident	0	1	2	0	0
Fingerprint_Pattern	Fire_Department	0	0	0	0	0
Iris_Image	Fire_Department	0	0	0	0	0
CCTV_Footage	Fire_Department	0	0	0	0	0
Smart_Card	Fire_Department	0	0	0	0	0
Electricity_Consumption	Fire_Department	0	0	0	0	0
Time_of_Day	Fire_Department	0	0	0	0	0
CO_Level	Fire_Department	0	0	3	3	0
Smoke_Level	Fire_Department	0	0	3	3	0
Body_Motion	Fire_Department	0	0	0	0	0
SMS	Fire_Department	0	0	3	3	0
E-mail	Fire_Department	0	0	3	3	0
Username	Fire_Department	0	0	0	0	0
Password	Fire_Department	0	0	0	0	0
Room_Temperature	Fire_Department	0	0	0	0	0
Humidity_Level	Fire_Department	0	0	0	0	0
A/C_Fan_Speed	Fire_Department	0	0	0	0	0
Day-light_Luminance	Fire_Department	0	0	0	0	0

Table B.4: Matrix T for Smart Building System



Asset	Parameter	Mode
A/C_Fan_Speed	Authorization	1
A/C_Fan_Speed	Autthentication	1
A/C_Fan_Speed	Confidentiality	1
A/C_Fan_Speed	Integrity	2
A/C_Fan_Speed	Non-repudiation	1
Body_Motion	Authorization	2
Body_Motion	Autthentication	2
Body_Motion	Confidentiality	2
Body_Motion	Integrity	3
Body_Motion	Non-repudiation	2
CCTV_Footage	Authorization	3
CCTV_Footage	Autthentication	3
CCTV_Footage	Confidentiality	3
CCTV_Footage	Integrity	3
CCTV_Footage	Non-repudiation	3
CO_Level	Authorization	1
CO_Level	Autthentication	3
CO_Level	Confidentiality	1
CO_Level	Integrity	3
CO_Level	Non-repudiation	3
Day-light_Luminance	Authorization	1
Day-light_Luminance	Autthentication	1
Day-light_Luminance	Confidentiality	1
Day-light_Luminance	Integrity	2
Day-light_Luminance	Non-repudiation	1
E-mail	Authorization	3
E-mail	Autthentication	2
E-mail	Confidentiality	3
E-mail	Integrity	3
E-mail	Non-repudiation	3
Electricity_Consumption	Authorization	1
Electricity_Consumption	Autthentication	1
Electricity_Consumption	Confidentiality	3
Electricity_Consumption	Integrity	3
Electricity_Consumption	Non-repudiation	1
Fingerprint_Pattern	Authorization	3
Fingerprint_Pattern	Autthentication	3
Fingerprint_Pattern	Confidentiality	3
Fingerprint_Pattern	Integrity	3
Fingerprint_Pattern	Non-repudiation	3
Humidity_Level	Authorization	1
Humidity_Level	Autthentication	2
Humidity_Level	Confidentiality	1
Humidity_Level	Integrity	2
Humidity_Level	Non-repudiation	1
Iris_Image	Authorization	3
Iris_Image	Autthentication	3
Iris_Image	Confidentiality	3
Iris_Image	Integrity	3
Iris_Image	Non-repudiation	3
Password	Authorization	3
Password	Autthentication	3
Password	Confidentiality	3
Password	Integrity	3
Password	Non-repudiation	3
Room_Temperature	Authorization	1
Room_Temperature	Autthentication	2
Room_Temperature	Confidentiality	1
Room_Temperature	Integrity	2
Room_Temperature	Non-repudiation	2
Smart_Card	Authorization	3

Table B.5: Matrix U or Asset Rank Smart Building System



Asset	Mode	Rank
A/C_Fan_Speed	1	LOW
Body_Motion	2	MEDIUM
CCTV_Footage	3	HIGH
CO_Level	3	HIGH
Day-light_Luminance	1	LOW
E-mail	3	HIGH
Electricity_Consumption	1	LOW
Fingerprint_Pattern	3	HIGH
Humidity_Level	1	LOW
Iris_Image	3	HIGH
Password	3	HIGH
Room_Temperature	2	MEDIUM
Smart_Card	3	HIGH
Smoke_Level	3	HIGH
SMS	3	HIGH
Time_of_Day	1	LOW
Username	3	HIGH

B.3 Functionality Rank Diagrams (FRD) for **Smart Building System**

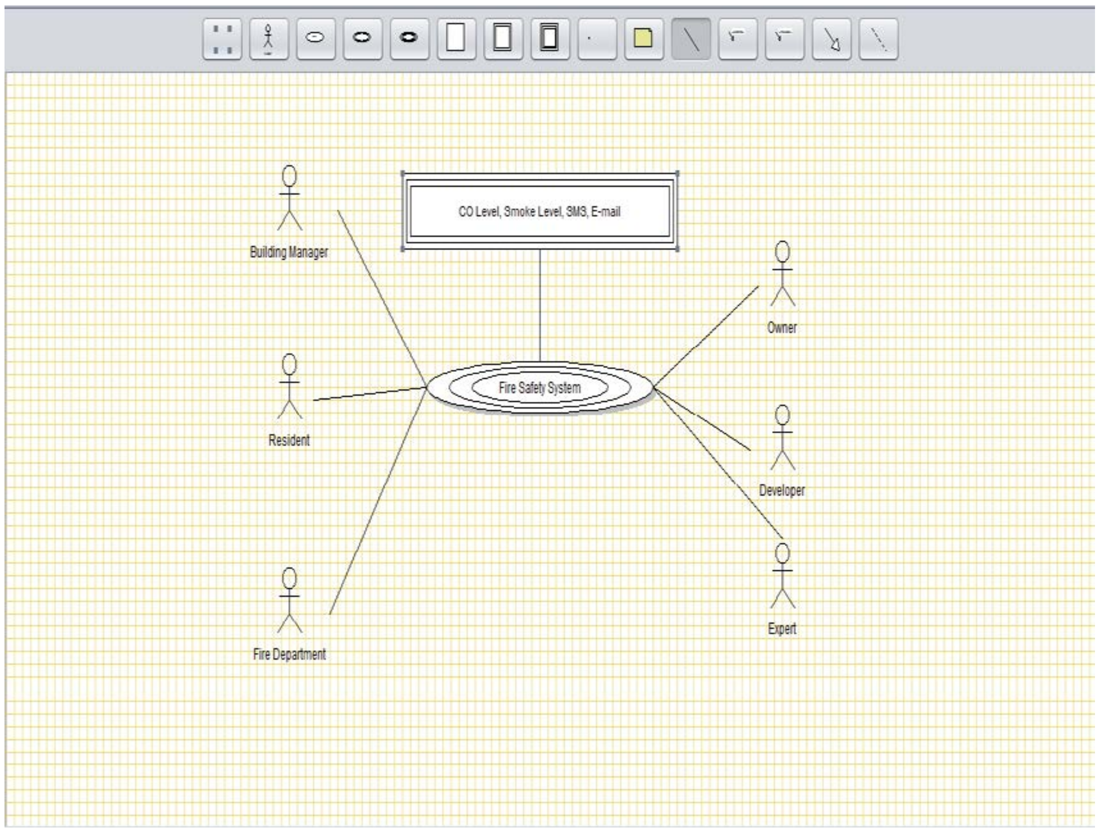


Figure B.1: FRD for Fire Safety System

Table B.6: Template for Fire Safety System

Functionality: Fire Safety System	
Stakeholders	
Building Manager	
Resident	
Fire Department	
Owner	
Developer	
Expert	
Assets	
Name	**Rank**
CO Level	High
SMS	High
E-mail	High
Functionality Rank: High	
Measure: SMS and E-mail with highest precedence and integrity to authorized stakeholders with no confidentiality	

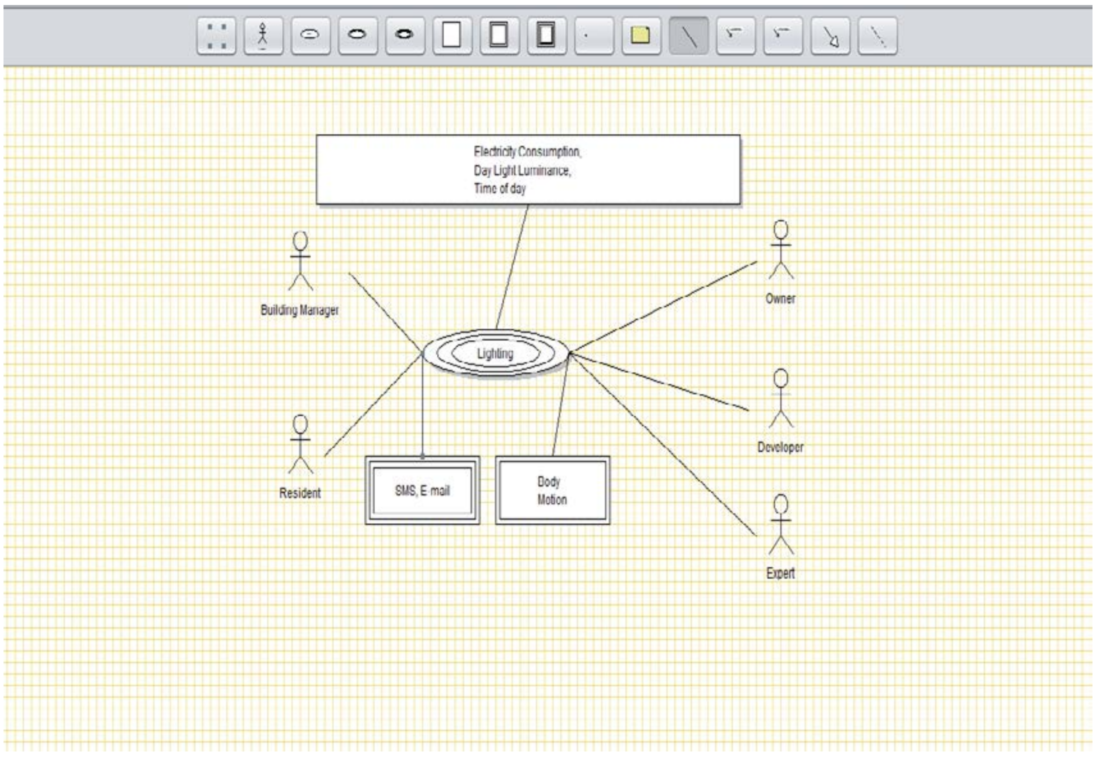


Figure B.2: FRD for Lighting System

Table B.7: Template for Lighting System

Functionality: Lighting System	
Stakeholders	
Building Manager	
Resident	
Owner	
Developer	
Expert	
Assets	
Name	**Rank**
Electricity Consumption	Low
Day-Light Luminance	Low
Time of Day	Low
Body Motion	Medium
SMS	High
E-mail	High
Functionality Rank: Medium	
Measure: Username and Password	

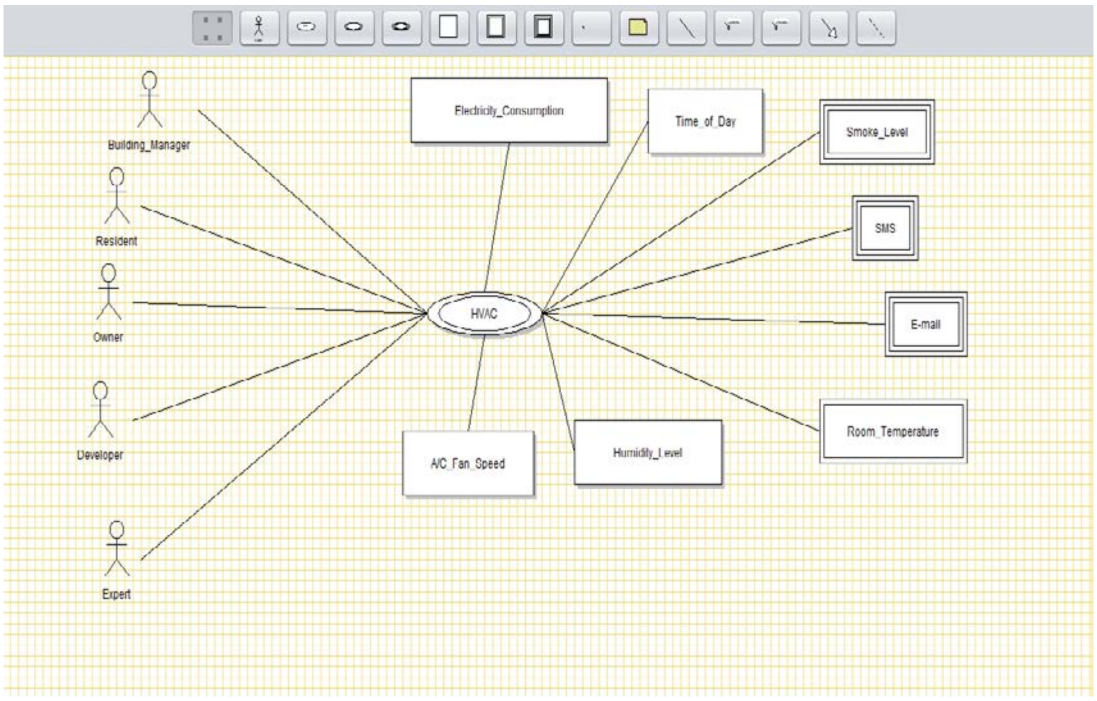


Figure B.3: FRD for HVAC System

Table B.8: Template for HVAC System

Functionality: HVAC System	
Stakeholders	
Building Manager	
Resident	
Owner	
Developer	
Expert	
Assets	
Name	**Rank**
Electricity Consumption	Low
Time of Day	Low
Smoke Level	High
SMS	High
E-mail	High
Room Temperature	Medium
Humidity Level	Low
AC/Fan Speed	Low
Functionality Rank: Medium	
Measure: Username and Password	

B.4 Authorization Rank Diagrams (ARD) for Smart Building System

Figure B.4 is a stakeholder-centric ARD for stakeholder ‘Resident’. It has Write with Permission (WP) right on asset ‘Fingerprint Pattern’ permitted by ‘Building Manager. He holds ‘Read’ right on Day-light Luminance and ‘Read’ right on CO Level. Table B.11 is the template for this diagram.

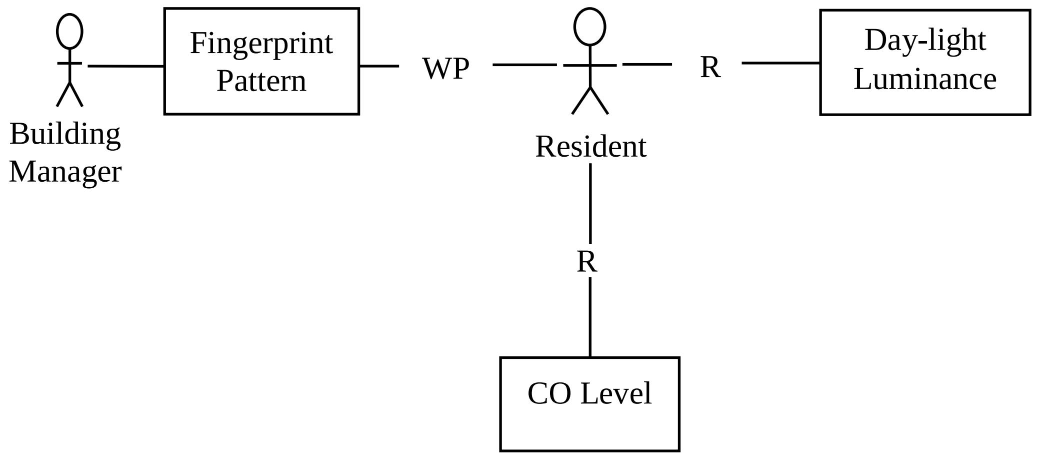


Figure B.4: ARD for Resident

Table B.9: ARD Template for Resident

Assets	**Authorization Rights**	**Permitting Stakeholders**
Fingerprint Pattern	WP	Building Manager
Day-Light Luminance	R	
CO Level	R	

Figure B.5 shows an asset-oriented ARD for Username. It can be seen that over this asset the stakeholder Building Manager has 'Write' right, Resident has 'Write with Permission (WP)' right provided by Building Manager and Owner has 'Read' right. Table B.10 is the template for this diagram.

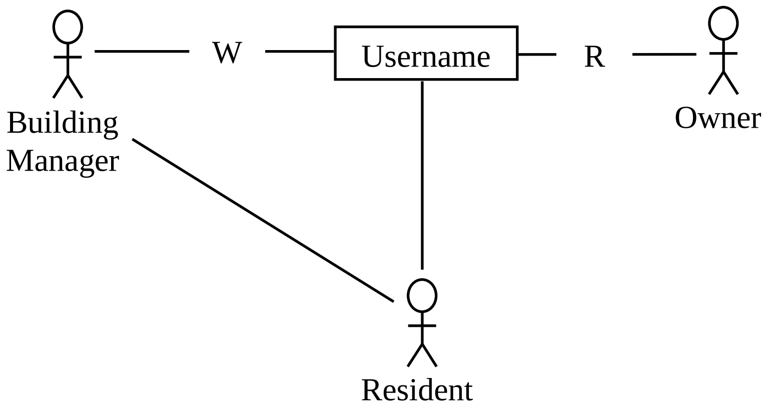


Figure B.5: Asset-Oriented ARD for Username

Table B.10: Template for Asset-oriented ARD Template for Username

Asset: Username		
Stakeholder	**Authorization Rights**	**Permitting Stakeholders**
Building Manager	W	
Owner	R	
Resident	WP	Building Manager

Appendix C:Publications

- R. Goel, M. C. Govil, G. Singh, “Eliciting, Analyzing and Modeling Software Security Requirements”, *International Journal of Software Engineering and Its Applications*, Vol. 11 no. 5, pp. 34-57, 2016.
- R. Goel, M. C. Govil, G. Singh, “Security in Requirements and Design Phases”, *IIOABJ*, Vol.7 , no. 1, pp. 585 -589, 2016.
- R. Goel, M. C. Govil, G. Singh, "A Novel Methodology for Effective Requirements Elicitation and Modeling“, In Proc. International Conference on Computational Science and Its Applications, 2018, pp. 474-487.
- R. Goel, M. C. Govil, G. Singh , “Security Requirements Elicitation and Modeling Authorizations”, in Communications in Computer and Information Science, P. Mueller, S. Thampi, M.A. Bhuiyan, R. Ko , R. Doss, C.J. Alcaraz, (eds) Singapore: Springer, vol. 625, 2016, pp. 239-250.
- R. Goel, M. C. Govil, G. Singh , “Modeling Software Security Requirements through Functionality Rank Diagrams”, in Lecture Notes in Computer Science , O. Gervasi et al. (eds) , Singapore: Springer, Vol. 9790, 2016, pp. 398-409.
- R. Goel, M. C. Govil, G. Singh, "A Secure Software Design Methodology", In Proc. International Conference on Advances in Computing, Communications and Informatics, 2016 pp. 2484-2488.
- R. Goel, M. C. Govil, G. Singh, “Security Requirements Elicitation and Assessment Mechanism (SecREAM)”, In Proc. International Conference on Advances in Computing, Communications and Informatics, 2015, pp. 1862-1866.
- R. Goel, M. C. Govil, G. Singh, “Imbibing Security in Software Development Life Cycle: A Review Paper”, In Proc. Afro - Asian International Conference on Science, Engineering & Technology, 2015, pp. 593 – 599.

References

[1] B. Stroustrup, “Software development for infrastructure,” *IEEE Comput.*, vol. 45, no. 1, pp. 47–58, 2012.

[2] B. W. Boehm, “Value-based software engineering: Overview and Agenda,” in *Value-based Software Engineering*, G. P. Biffl S., Aurum A., Boehm B., Erdogmus H., Ed. Springer Berlin Heidelberg, 2006, pp. 3–14.

[3] G. Holodnik-Janczura and I. Golinska, “Decision Support System for Choosing a Model for a Software Development Life Cycle,” *Oper. Res. Decis.*, vol. 20 no. 1, pp. 61–77, 2010.

[4] On Point Corporation, “Incorporating Security into the System Development Life Cycle (SDLC),” Report, Arlington, VA, 2010.

[5] R. S. Pressman, *Software Engineering A Practitioner’s Approach*, 5th ed. New York: Mc Graw Hill, 2001.

[6] I. Sommerville, *Software Engineering*, 9th ed. Addison-Wesley, 2010.

[7] A. Van Lamsweerde, “Goal-oriented requirements engineering: from System Objectives to UML Models to Precise Software Specifications,” In Proc. 25th International Conference on Software Engineering, 2003, pp. 744–745.

[8] K. Beznosov, “Extreme Security Engineering: On Employing XP Practices to Achieve ‘Good Enough Security’ without Defining It,” In *First ACM Work. on Business Driven Security Engineering*, vol. 31, 2003.

[9] A. Mishra and D. Dubey, “A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios,” *Int. J. Adv. Res. Comput. Sci. Manag. Stud.*, vol. 1, no. 5, pp. 64–69, 2013.

[10] B. W. Boehm, “A Spiral Model of Software Development ans Enhancement,” *IEEE Comput.*, vol. 21, no. 5, pp. 61–72, 1988.

[11] N. Kumar, A. S. Zadgaonkar, and A. Shukla, “Evolving a New Software Development Life Cycle Model SDLC-2013 with Client Satisfaction,” *Int. J. Soft Comput. Eng.*, vol. 3, no. 1, pp. 216–221, 2013.

[12] B. Boehm, A. Egyed, J. Kwan, D. Port, A. Shah, and R. Madachy, “Using the WinWin spiral model: A case study,” *Computer (Long. Beach. Calif).*, vol. 31, no. 7, pp. 33–44, 1998.

[13] Y. Yang, “Composable Risk-driven Processes for Developing Software Systems from Commercial-off-the-shelf (COTS) Products,” USC Graduate School, University of Southern California, 2006.

[14] S. Koolmanojwong, “The Incremental Commitment Spiral Model Process Patterns for Rapid-Fielding Projects,” Ph. D. Thesis, USC Graduate School,

University of Southern California, 2010.

[15] J. Kerr and R. Hunter, *Inside RAD*, McGraw-Hill, 1994.

[16] R. Breu, K. Burger, M. Hafner, J. Jurjens, G. Popp, G. Wimmel, and V. Lotz, “Key Issues of a Formally Based Process Model for Security Engineering,” In Proc. *16th International Conference on Software & Systems Engineering and their Applications*, 2003, pp. 1–15.

[17] J. Martin, *Rapid Application Development*. Indianapolis: Macmillan publishing company, 1991.

[18] J. Butler, “Rapid Application Development in Action,” *Manag. Syst. Dev. Appl. Comput. Res.*, vol. 14, no. 5, pp. 6–8.

[19] P. Krutchen, *The Rational Unified Process—An Introduction*, 3rd ed. Reading, MA: Addison-Wesley., 2003.

[20] G. Booch, I. Jacobson, and J. Rumbaugh, *Unified Software Development Process,* Pearson Education India, 1999.

[21] S. Abrahão, C. Gravino, E. Insfran, G. Scanniello, and G. Tortora, “Assessing the Effectiveness of Dynamic Modeling in the Comprehension of Software Requirements : Results from a Family of Five Experiments,” *IEEE Trans. Softw. Eng.*, vol. 39, no. 3, pp. 327–342, 2013.

[22] E. Woods, “Harnessing UML for Architectural Description —The Context View,” *IEEE Softw.*, vol. 31, no. 6, pp. 30–33, 2014.

[23] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, 1st ed. Reading, Massachusetts: Addison Wesley, 1998.

[24] A. Cockburn, *Writing Effective Use Cases*. Boston: Addison-Wesley, 2001.

[25] C. Choppy and G. Reggio, “Requirements capture and specification for enterprise applications: A UML based attempt,” In Proc. *Aust. Softw. Eng. Conf.* pp. 19–28, 2006.

[26] B. Selic, “Framework A Generic Resources with UML for Modeling with UML,” *Computer,* vol. 36, no. 5., pp. 64-69 2000.

[27] J. Jurjens, “UMLsec: Extending UML for Secure Systems Development,” In Proc. *5th International Conference on The Unified Modeling Language*, 2002, pp. 412–425.

[28] S. Konrad, H. Goldsby, K. Lopez, and B. H. C. Cheng, “Visualizing requirements in UML models,” In *First Int. Work. Vis. Requir. Eng.*, 2007.

[29] B. Dobing and J. Parsons, “How UML is Used,” *Commun. ACM*, vol. 49, no. 5, pp. 109–113, 2006.

[30] *M. Glinz,* “Problems and Deficiencies of UML as a Requirements Specification

Language," In Proc. *Tenth International Workshop on Software Specification and Design*, 2000, pp. 11–22.

[31] R. Goel, M. C. Govil, and G. Singh, "Imbibing Security in Software Development Life Cycle : A Review Paper," In Proc. *3rd Afro - Asian International Conference on Science, Engineering & Technology*, 2015, pp. 593–599.

[32] C. Kobryn, "UML 3 . 0 and the Future of Modeling," *J. Softw. Syst. Model.*, vol. 3, no. 1, pp. 4–8, 2004.

[33] G. Sindre and A. Opdahl, "Capturing Security Requirements through Misuse Cases," In P*roc. 14th Norwegian Informatics Conference, Troms, Norway*. 2001.

[34] D. D'souza and A. Wills, *Objects, components, and frameworks with UML: The catalysis approach*. Reading, Massachusetts: Addision-Wesley, 1999.

[35] A. Herrmann, A. Morali, S. Etalle, and R. Wieringa, "RiskREP: Risk-based Security Requirements Elicitation and Prioritization," in *1st International Workshop on Alignment of Business Process and Security Modelling* , 2011, pp. 155-162.

[36] J. Saltzer and M. Schroeder, "The Protection of Information in Computer Systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.

[37] K. M. Goertzel, T. Winograd, H. McKinley, and P. Holley, "Security in the Software Lifecycle Making Software Development Processes — and Software Produced by Them — More Secure," 2006.

[38] J. Yoder and J. Barcalow, "Architectural Patterns for Enabling Application Security," *Urbana*, vol. 51, p. 61801, 1998.

[39] G. P. Mullery, "CORE-A Method for Controlled Requirement Specification," In Proc. *4th International Conference on Software engineering*, 1979, pp. 126–135.

[40] G. Booch, J. Rumbaugh, and I. Jacobson, *Unified Modeling Language User Guide*, vol. 2nd. Reading, Massachusetts: Addison Wesley, 2005.

[41] Microsoft, "Security Development Lifecycle for Agile Development," vol. 2013, no. October 15. Microsoft Press, 2012.

[42] F. Swiderski and W. Snyder, *Threat Modeling*, 1st ed. Microsoft Press, 2005.

[43] G. W. Gerhard Popp, Jan Jürjens, "Use Case Oriented Development of Security-Critical Systems," *Inf. Secur. Bull.*, vol. 8, no. 2, pp. 55–60, 2003.

[44] D. Shreyas, "Software Engineering for Security - Towards Architecting Secure Software," *ICS*, vol. 221, pp. 1–12, 2001.

[45] J. D. Meier, A. Mackman, B. Wastell, P. Bansode, and C. Bijwe, "Security

Guidelines: .NET Framework 2.0.” Microsoft, 2005.

[46] R. Anderson., *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 2001.

[47] N. Davis, “Secure Software Development Life Cycle Processes : A Technology Scouting Report,” No. CMU/SEI-2005-TN-024. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 2005.

[48] M. Lindvall, V. R. Basili, B. W. Boehm, P. Costa, K. Dangle, F. Shull, R. Tesoriero, L. Williams, and M. V Zelkowitz, “Empirical Findings in Agile Methods,” In Proc. *2nd XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods*, 2002, pp. 197–207.

[49] O. Korkmaz, “System simulation for Software Quality Assurance,” MS Thesis, Department of Computer Engineering, Atilim University, Ankara, 2007.

[50] J. Wäyrynen, M. Bodeen, and G. Boström, “Security Engineering and eXtreme Programming: An Impossible Marriage?,” In Proc. *Extreme Programming and Agile Methods*, 2004, pp. 117–128.

[51] A. K. Talukder and H. A. Prahalad, “,” In Proc. *Internet Multimed. Serv. Archit. Appl. Int. Conf.*, 2009, pp. 1-4.

[52] J. J. Pauli and D. Xu, “Misuse case-based design and analysis of secure software architecture,” *Inf. Technol. Coding Comput.* vol. 2, no. C, pp. 2005–2010, 2005.

[53] D. Shackleford, “Integrating Security into Development, No Pain Required,” *SANS Whitepaper*, September, 2011.

[54] Z. M. Kasirun and S. S. Salim, “Focus Group Discussion Model for Requirements Elicitation Activity,” In Proc. *International Conference on Computer and Electrical Engineering,* 2008, pp. 101–105.

[55] I. Sommerville, P. Sawyer, and S. Viller, “Viewpoints for requirements elicitation: a practical approach,” In P*roc. IEEE Int. Symp. Requir. Eng. RE ’98*, 1998, pp. 74-81.

[56] J. D. McGregor, “Secure Software.,” *J. Object Technol.*, vol. 4, no. 4, p. 33, 2005.

[57] A. Sachitano and R. Chapman, “Security in software architecture: a case study,” In *IEEE Work. Inf. Assur.*, June, 2004.

[58] P. Zave, “Classification of Research Efforts in Requirements Engineering York, England. IEEE Press,” In Proc. *Second IEEE International Symposium on Requirements Engineering,* 1995, pp. 214–216.

[59] N. Sportsman, “Threat Modeling,” Praetorian, Information Security Provider and Research Center, 2007.

[60] S. F. Burns, “Threat Modeling : A Process to Ensure Application Security,” *GIAC Security Essentials Certification (GSEC) Practical Assignment,* SANS Institute InfoSec Reading Room, 2005.

[61] E. R. Keith, “Agile Software Development Processes A Different Approach to Software Design”, A White paper available online at

http://cf. agilealliance. org/articles/system/article/file/1099/file. pdf., 2003.

[62] S. Kishore and R. Naik, *Software Requirements and Estimation*, 1st ed. New Delhi: Tata McGraw-Hill Education, 2001.

[63] L. Futcher and R. von Solms, “SecSDM: A Usable Tool to Support IT Undergraduate Students in Secure Software Development,” In Proc. *Human Aspects of Information Security & Assurance,* 2012, pp. 86–96.

[64] D. Burley and M. Bishop, “Summit on Education in Secure Software: Final report,” 2011.

[65] N. Sharif, K. Zafar, and W. Zyad, “Optimization of Requirement Prioritization Using Computational Intelligence Technique,” In Proc. *International Conference on Robotics and Emerging Allied Technologies in Engineering,* 2014, pp. 228–234.

[66] F. Fellir, K. Nafil, and R. Touahni, “System Requirements Priorization based on AHP,” *Inf. Sci. Technol. (CIST), IEEE Int. Colloq.*, no. 978–1–4799–5979–2/14, pp. 163–167, 2014.

[67] S. Neetu Kumari and A. S. Pillai, “A survey on global requirements elicitation issues and proposed research framework,” In Proc., *IEEE Int. Conf. Softw. Eng. Serv. Sci.*, pp. 554–557, 2013.

[68] U. I. Hernández, F. J. Á. Rodríguez, and M. V. Martin, “Use Processes - Modeling requirements based on elements of BPMN and UML Use Case Diagrams,” In Proc. *2nd Int. Conf. Softw. Technol. Eng. , 2010,* pp. 36–40.

[69] M. I. Kamata and T. Tamai, “How Does Requirements Quality Relate to Project Success or Failure?,” In Proc. *Requirements Engineering Conference,* 2007, pp. 69–78.

[70] F. Sher, D. N. A. Jawawi, R. Mohamad, and M. I. Babar, “Requirements Prioritization Techniques and Different Aspects for Prioritization a Systematic Literature Review Protocol,”In Proc. *8th Malaysian Software Engineering Conference*, 2014, pp. 31–36.

[71] M. I. Babar, M. Ramzan, and S. a. K. Ghayyur, “Challenges and Future Trends in Software Requirements Prioritization,” In Proc. *International Conference on Computer Networks and Information Technology,* 2011, pp. 319–324.

[72] R. Snijders, “Crowd-Centric Requirements Engineering,” In Proc. *International Conference on Utility and Cloud Computing*, 2014, pp. 614–615.

[73] N. Sabahat, F. Iqbal, F. Azam, and M. Y. Javed, “An iterative approach for global requirements elicitation: A case study analysis,” In Proc. *International Conference On Electronics and Information Engineering*, 2010, pp. 361–366.

[74] C. B. Haley, R. Laney, and J. D. et al. Moffett, “Security Requirements Engineering : A Framework for Representation and Analysis,” *IEEE Trans. Softw. Eng.*, vol. 34, no. 1, pp. 133–153, 2008.

[75] B. B. Chua, D. V. Bernardo, and J. Verner, “Understanding the use of elicitation approaches for effective requirements gathering,” In Proc.*5th Int. Conf. Softw. Eng. Adv., 2010*, pp. 325–330.

[76] B. H. Wu, “On software engineering and software methodologies a software developer’s perspective,” In Proc. *Int. Conf. Inf. Sci. Technol.* 2011, pp. 155–162.

[77] S. T. Redwine and N. Davis, “Processes to Produce Secure Software: Towards More Secure Software,” National Cyber Security Summit, Vol.1, 2004.

[78] N. Coblentz, “Microsoft SDL : Agile Development”, OWASP Foundation ” 2010.

[79] S. Sawyer and P. J. Guinan, “Software Development: Processes and Performance,” *IBM Syst. J.*, vol. 37, no. 4, pp. 552–569, 1998.

[80] A. Birk, D. Surmann, and K. D. Althoff, “Applications of Knowledge Acquisition in Experimental Software Engineering,” In Proc. *11th EuropeanWorkshop on Knowledge Acquisition, Modeling, and Management,* 1999, pp. 67–84.

[81] S. Basri and R. V. O’Connor, “The impact of software development team dynamics on the Knowledge Management Process,” In Proc. *23rd International Conference on Software Engineering and Knowledge Engineering*, 2011, pp. 339–342.

[82] J. Viega and G. McGraw, *Building Secure Software: How to Avoid Security Problems the Right Way*. Addison-Wesley, 2002.

[83] J. Rasmussen, “Introducing XP into Greenfield Projects: Lessons Learned,” *IEEE Softw.*, vol. 20, no. 3, 2003.

[84] A. Knauss, “On the usage of context for requirements elicitation: End-user involvement in IT ecosystems,” In Proc. *IEEE International Requirements Engineering Conference*, 2012, pp. 345–348.

[85] F. I. P. Standards, “Standards for Security Categorization of Federal Information and Information Systems,” *Fed. Inf. Process. Stand. Publ.*, vol. FIPS PUB 1, no. February, 2004.

[86] B. A. Forouzan, *Data Communications and Networking*, 4th ed. McGraw-Hill, 2007.

[87] K. Pohl K and C. Rupp, *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam Foundation Level-IREB Compliant*. Santa Barbara, CA: Rocky Nook Inc, 2015.

[88] S. Hatton, "Early Prioritisation of Goals," In Proc. *26th International Conference on Conceptual Modeling Foundations and Applications*, 2007, pp. 235–244.

[89] P. Saripalli and B. Walters, "QUIRC: A Quantitative Impact and Risk Assessment Framework for Cloud Security," In Proc. *3rd International Conference on Cloud Computing*, 2010, pp. 2005–2010.

[90] A. K. Talukder, V. K. Maurya, B. G. Santhosh, E. Jangam, S. V Muni, K. P. Jevitha, S. Saurabh, and A. R. Pais, "Security-aware Software Development Life Cycle (SaSDLC)-Processes and Tools," In Proc. *IFIP International Conference on Wireless and Optical Communications Networks*, 2009, pp. 1–5.

[91] D. E. Perry, A. a. Porter, and L. G. Votta, "Empirical studies of software engineering," In Proc. *Conf. Futur. Softw. Eng., 2000*, pp. 345–355.

[92] G. N. Fenton, S.L. Pfleeger, "Science and Substance: A Challenge to Software Engineers," *IEEE Softw.*, vol. 11, no. 4, pp. 86–95, 1994.

[93] D. Turk, R. France, and B. Rumpe, "Limitations of agile software processes," In Proc. *Third Int. Conf. Extrem. Program. Agil. Process. Softw. Eng. 2002*, p. 4.

[94] K. Beznosov and P. Kruchten, "Towards agile security assurance," In Proc. *Work. New Secur. Paradig.*, 2005, p. 47.

[95] R. Labbe, "Microsoft Security Development Lifecycle for IT." Microsoft, 2006.

[96] M. Vetterling, G. Wimmel, and A. Wisspeintner, "Secure Systems Development Based on the Common Criteria: The PalME Project," pp. 129–138, 2002.

[97] Y. C. Stamatiou, E. Henriksen, M. S. Lund, and E. Mantzouranis, "Experiences from using model-based risk assessment to evaluate the security of a telemedicine application," In Proc. *Telemedicine in Care Delivery*, 2002, p. 115–119.

[98] K. Stolen, F. D. Braber, T. Dimitrakos, R. Fredriksen, B.A. Gran, S.H. Houmb, M.S. Lund, Y. Stamatiou, "Model-based Risk Assessment - the CORAS approach," In Proc. *NIK Informatics Conference*, 2002.

[99] W. Xiong, X.-T. Wang, and Z.-X. Wu, "Study of a customer satisfaction-oriented model for outsourcing software quality management using Quality Function Deployment (QFD)," In Proc. *4th International Conference on Wireless Communications, Networking and Mobile Computing*, 2008.

[100] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, “Feature-oriented Domain Analysis (FODA) Feasibility Study,” No. CMU/SEI-90-TR-21, Carnegie-Mellon Univ Pittsburgh Pa, Software Engineering Inst, 1990.

[101] N. Douglas and H. W. J. Rittel, “Issue Based Information System for Design,” *Association for Computer Aided Design in Architecture*, pp. 275-286, 1989.

[102] G. Sindre and A. L. Opdahl, “Eliciting Security Requirements with Misuse Cases,” *Requir. Eng.*, vol. 10, no. 1, pp. 34–44, 2005.

[103] D. Kulak and E. Guiney, *Use cases: requirements in context*. New York: ACM Press, 2000.

[104] S. Swigart, Scott, Campbell, “Evolution of the Microsoft Security Development Lifecycle,” *SDL Series - Article #7*, no. May. Microsoft, pp. 1–8, 2009.

[105] M. Howard and D. LeBlanc, *Writing Secure Code*, 2nd ed. Microsoft Press, 2004.

[106] R. S. Sanford Friedenthal, Alan Moore, *Practical Guide to SysML: Systems Modeling Language*, 3rd ed. San Francisco: Morgan Kaufmann Publishers, 2015.

[107] J. Chanda, A. Kanjilal, S. Sengupta, and S. Bhattacharya, “Traceability of requirements and consistency verification of UML use case, activity and Class diagram: A Formal approach,” In Proc. *Int. Conf. Methods Model. Comput. Sci.*, 2009, pp. 1-4.

[108] C. R. Kothari, *Research methodology: Methods and Techniques.*, 2nd ed. Delhi: New Age International, 2004.

[109] R. Goel, M. C. Govil, and G. Singh, “Security Requirements Elicitation and Modeling Authorizations,” In Proc. *International Symposium on Security in Computing and Communication*, 2016, pp. 239–250.

[110] J. M. Andres Laya, Vlad-loan Bratu, “Who is investing in machine-to-machine communications?,” In Proc. *24th European Regional Conference of the International Telecommunication Society*, 2013.

[111] A. Al-fuqaha, M. Guizani, and M. Mohammadi, “Internet of Things : A Survey on Enabling Technologies, Protocols, and Applications,” *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[112] B. Sridharan and A. P. Mathur, “Infrastructure for the Management of SmartHomes,” *Software Engineering Research Center Tech Report SERC-TR-177-P*, 2002.

[113] R. J. Robles and T. Kim, “A Review on Security in Smart Home Development,” *Int. J. Adv. Sci. Technol.*, vol. 15, pp. 13–22, 2010.

Author’s Biography

Rajat Goel joined Ph.D. at Department of Computer Science and Engineering, Malaviya National Institute of Technology (MNIT) Jaipur in 2013. He obtained M.Tech. in Computer Science and Engineering from Central University of Rajasthan, Ajmer in 2012 with specialization in information security. He received B.E in Computer Engineering from University of Rajasthan, Jaipur in 2008. He worked as system developer in IBM India Pvt. Ltd. and later chose to make his career in academics. He has taught at various institutions of repute. He is member of IEEE. His areas of interests include Software Engineering, Information Security and Database Management Systems.