*I.J. Information Technology and Computer Science,* **2015, 01, 1-11**
Published Online December 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2015.01.01

# TSSR: A Proposed Tool for Secure Software Requirement Management

Prof. **Mohammad Ubaidullah Bokhari**
Department of Computer Science, Aligarh Muslim University, Aligarh, India
Email: mubokhari@gmail.com

**Shams Tabrez Siddiqui**
Department of Computer Science, Aligarh Muslim University, Aligarh, India
Email: stsiddiqui.rs@amu.ac.in, stabrezsiddiqui@gmail.com

***Abstract*—** This paper provides a unified framework in which entire design of the project can be captured right from the beginning of the software development. This paper discusses about the requirements which should be included in the development of the requirement management tools. As the requirements, criteria which have been discussed, we introduce a requirement management tool known as TSSR (Tool for Secure Software Requirement). This tool manages risk analysis, system requirements, security of the system and project, users/group restriction, encrypted database, traceability and extension of the tool to interact with external requirement management tools. The aim of this paper is to describe the TSSR framework and its four components: Planner, Modeller, Prover and Documenter which will be helpful in interacting and managing requirements with arbitrary number of external tools for secure software development.

***Index Terms*—** Software Development, Requirement Management Tools, Secure Software Requirement Management, Risk Analysis, Encrypted Database and Traceability

## I. Introduction

Requirements are developed based on the project; as the size of the project increases, they become more concrete, more elaborate, more detailed and additionally more complex. The effort on requirements does not end even when the project enters the design, integration and implementation phases. Requirements changes, tracing to other development artifacts should be handled and established.

The requirements for a system are mostly large in numbers and too complex to be captured at once by humans and this is not a new problem. To handle the complexity of the project, a systematic tool supported requirements management (RM) should be established. The option of a RM tool is a significant and sensitive issue [1]. We have to select RM Tool on the basis of our requirements, the size of the group, and size of the project [2]. Since the requirements are used and changed virtually throughout the development process. A developer knows that changing tool within the middle of a project is an expensive and time-consuming business [3].

Requirement management tools are general argument for security requirement engineering such tools is based on the spreadsheet metaphor [4]. In metaphor a table is used to enter the attributes of the natural language requirement. Certain heavyweight requirement management tools contain its own extension Language (EXL), scripting frameworks which allowing augmentation with additional functionality for extending tool to interact with external tools [5]. Unfortunately, the generic strength of the requirements management tools is also its weakness; due to the lack of distinct semantics means analysts annually maintain traceability links between requirements and non requirements artifacts. Security is considered as a non-functional requirement which is more seriously taken into account nowadays [6] [7]. A good requirement requirements specification documents includes both functional and non-functional requirements [8] [9].

Expertise in different areas of a requirement engineering, security, we need to define useful requirements from the abstract common security goals. There are many RMT that uses low level formal verification. The security problem varies in almost all of the Requirement Management Tools. Just a few of the tools have solved security problem by providing centralized repository. The requirement included with the proposed tool are Risk analysis, impact analysis, restricted users/groups and encrypted database that will be helpful in developing secure projects.

This paper is structured in V sections, Section I is Introduction which discusses the requirements of Requirement Management Tools, security aspects of it and how to secure your project using RM Tools. Section II discusses Requirements for developing RMT from the system developers point of view, users point of view as well as from tool administrator's point of view. Section III is about the new tool known as Tool for Secure Software Requirement (TSSR) Management which discusses its components and framework in detail, based on the requirements which have been discussed in Section II. Section IV is Tool Interaction in which, the discussion is on how to import and export this tool with the third party tool in the project if required. Section V is a conclusion which concludes that developing a secure software right from the beginning of project development.

 

## II. Requirements for Developing RMT

This section describes the criteria and their requirements that needed from the system developers' point of view as well as tool administrators' point of view while developing RM Tool.

### *A. Requirement Management Information Model*

The tool must allow the users to freely define a Requirements managements Information model (RMI) that must be independent of process and method [10].

- The RMI must be easily changeable during the entire project.
- Database objects must be uniquely identifiable over its lifetime of the project.
- Reuse and Inheritance ought to be available for all classes, types and attributes.
- If required RMI could be defined graphically.

### *B. Views*

Various views of the same data must be supported by the tool.

- The tool must enable views and should be defined centrally as well as with a user-specific manner.
- In the current view the objects must be changeable and freely configurable
- The requirements of the users must be viewed in an information-model-oriented and document-oriented manner. i.e. tables, forms
- The RM tool must provide suitable views of the enormous amount of information accessible in the tool which will be helpful for the users to accept that tool [11].

### *C. Formatting and Multimedia files*

The tool should enable the requirements to be enriched with formatting. Specifications that are created with text processing tools like Word, which contain lots of graphics or other different multimedia elements, must be directly visible within the RM Tools user interface.

- The tool should support basic text formatting.
- The tool should enable mathematical formulas to be utilized in the description of the texts.
- The tool should also support scientific and foreign-language character sets.
- In the database non-text objects should be saved directly or at least in a configuration management tool which is tightly coupled.

### *D. Change Management and Comments*

The tool must handle formal change requests, systematically to reduce errors. This function must be customizable and integrated into rights management to change the process of the users.

- A restrictive change management is important in the later phases of the project development.
- The change requests should have accepted, rejected or pending like public status information.
- The users might add comments and changes to requirements in the comment or discussion function.

### *E. History Documentation*

- All changes made to the requirements should be tracked and kept within the database.
- The tool objects should be versioned, and distinction between them as a major and minor version.
- The tool should have a facility to allow a requirement to be changed back if required to any previous state anytime. Older versions and changes should always be available.
- The tool should generate freely configurable change reports and these reports should moreover relate to views, baselines and document generation.

### *F. Baseline*

The tool must support baselines that are used to save the specified set of requirements objects state, before a larger development step of the document or project, which is fixed at a given point of time. Baseline management features provide additional support for comparing and merging changes [11]. The status saved in a baseline is the starting point for further development of the project from that given point. A baseline is a database consisting of various objects, each in a certain version.

### *G. Traceability*

The tool must facilitate traceability through links between requirements. The linking ought to be highly user-friendly in a manner so that it helps only if it is relatively complete. It has an ability to provide a central, secure repository for project requirements [12]. Linking is not popular among developers because its benefit is mostly visible in later phases of project development.

- The tool must be able to impose the creation or change of certain links upon creation or change of a requirement [13].
- Links should be directed an object to a source and target at the same time and follow the links directly in both directions.
- Links should connect any objects within the database, not only in the same module and project.
- The tool must feature a concise graphical representation and navigation of the traces which is user-friendly [13].

### *H. Analysis Functions*

The tool should analyze requirements such as analysis of project progress, risk management and analysis of the link structure [14].

- The tool should provide latest status and progress information about the project.
- The tool should analyze inconsistencies in the link structure approximating finding gaps in the traces.
- The tool must scan the unsuitable/inexact language or wrongly used terminology in the texts description.

### *I. Tool Integration*

The tool should have open interfaces; the information stored in them should be visible and linkable to other tools which are used within the development process. It is one of the significant features of the RMT. RM tools must be integrated tightly to improve consistency

 

between development phases. It should also allow complete traceability over the entire product life cycle for existing tool environments [15].

- The connection should be transparent with 3rd party tools.
- Linking must not lead to redundant data.
- A Tool must manage the links to external objects in the same way as internal links.
- Recognized access rights and target the smallest possible formation of the external objects.
- The tool could support tool integration platforms and interface should occur automatically for active, synchronized or change notifications [13].

*J. Import/Export*

The tool should be able to import/export requirements specification documents from 3rd party tools. The simplest method to import text files into a spreadsheet and export this text as a CSV (common separated values), or tab delimited format files [11].

- The tool should recognize text marks, formatting, grammatical structure, line ends or keywords to interpret them right from the beginning to end of requirements texts and documents [15].
- The tool should support a semiautomatic import of requirements from existing documents [16].

*K. Document Generation*

The tool needs a document generator to generate official and internal documents with current data from the database. The generated documents are connected to the database; an independent document file is created. Document generator of the RM Tools is one of the main productivity-enhancing applications. Developers will generate documents at the push of a button and they don't have to carry out the detailed formatting before and after document generation [13].

- All information which is available in the tool must be included in the document generator.
- The document should be flexibly configurable, compared to views, formatting and positioning of the subset of data, too.
- The document generator should create documents in certain standard template formats.
- The generated document must be included in non-textual object and in the Meta information in the vein of the change history or ownership.
- With including external objects the tool must generate very large documents about 5000 pages.
- The document generator must be extensible when required via a programming interface provided by the tool and the document generated automatically as a background task.

*L. Checking out for Offline Use*

The tool ought to have a facility to work offline and whatever changes or work done during this should be stored. When users come online the changes should be enabled and send to the data repository so that other users can check out the changes which are made offline [14].

*M. Web Access*

The tool ought to have a web interface or browser-based client that avoids unnecessary installation of a client application for sporadic users. They are interested with the internal users that uses the tool occasionally collaboration with external partners. Web interfaces offer a reliable as well as easily manageable opportunity to work with the requirements. Most of the users are "power users" for whom the native clients provide a smoother user experience on the web, some managers and administrators has headaches while opening the tool to the web.

*N. Database*

A RM tool database failure can vanish all the data which is stored in the database. If developers can't work within the deadlines of the project, then it will be very expensive.

- The tool should use an appropriate database technology, which must be secure, scalable and reliable.
- Maintenance work of the database must be done the system is running.
- The tool must consistently provide the safe transaction of the database. The database must be available 24 hours in a day and 365 days in a year.
- The database must have a data-integrity check and consistency-analysis. In near future if needed, it must be able to repair such errors.
- The tool must use a database that can be administered independently so that availability and security can be improved.
- It must be possible to export all project data and to import them again at a distinct place or time for/with different tool.

*O. Size Restrictions*

The database must able to handle large projects. The size restriction of the database and the number of requirements, users, groups as well as upper limit should not be there. The database fields should not have a fixed size restriction and if such size restriction exists they must be known exactly.

- Unlimited number of requirements
- Unlimited size of a requirement
- Unrestricted number of users, user groups and roles
- Unrestricted size of database

*P. Encryption*

The tool must store information in the database in encrypted form so that it should not be readable to system administrators. The tool ought to allow all communication which is in between client and server must be in encrypted form. Suppliers have a strong interest in data security and this is a major problem in this higher competitive sophisticated market. The cryptography ought to protect the database from unauthorized users.

 

*Q. Central Installation*

The tool should be centrally installed so that all project-wide information should be controlled and changed from one place. Associated changed history should be available, viewed to all users as well as project administrator. The RM tool implements a devoted group of persons to take the responsibility for the exact mapping of the process specification.

*R. Extensibility*

The tool ought to be adaptable and extensible according to the needs and requirements of the software, project or organization.

- The tool should provide an open and well-documented object models, whereas API that makes all data and functions available as well as accessible to extensions.
- The tool should use standard programming languages.
- The tools user interface must be customizable and extensible with the use of a standard script language [16].

*S. Users, Roles and Rights*

The tool should allow fine-grained administration of users, user groups, roles, and rights. A history of changes to those should be available within the database [14].

- The administrator should assign central the roles to manage user accounts and groups.
- Centrally users should be defined for all projects.
- API programming or scripting extensions should not compromise with security concept.
- Users must perform more than one role at a time.
- Access rights should be grantable and a distinction ought to be made between rights to view and propose changes and make changes.

## III. Tool for Secure Software Requirement (TSSR) Management

TSSR is a new requirement management tool used for managing the requirement which will be helpful in developing secure software. It has four components and each component has their own functionalities, which has been discussed underneath.

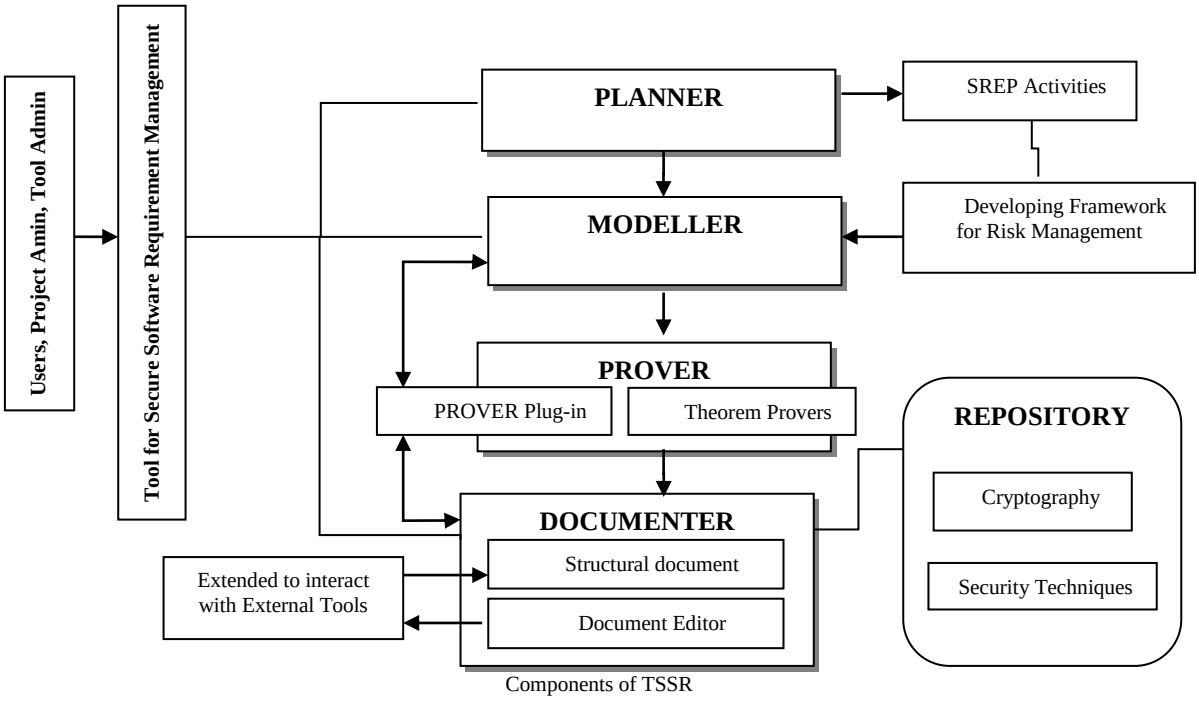


Fig. 1. Framework of Tool for Secure Software Requirement Management

The main components of the TSSR are:

1) The DOCUMENTER supports the powerful indexing, structured view, editing of technical documents as well as it provides an interface to external tools. Within a single project artifact the DOCUMENTER manages all associated documents and design constructs. The database is secure by using cryptography for protecting the database from unauthorized users. The database also uses a Cipher Hill security technique for encryption and decryption of data in the database.
2) The PROVER is used as a plug-in to the DOCUMENTER that provides intelligent theory management, literate reasoning, proof support and theorem-proving in Higher-Order Logic for checking of mathematical models [17].
3) The MODELLER is a TSSR module supporting system modeling, System Architecture and verification using the DOCUMENTER/PROVER system. It also provides intelligent graphical visualization of design, reasoning about concurrency, animation and proof for tasks, real-time properties and sophisticated hierarchical design for dataflow and state machine [17].
4) The PLANNER supports pre- requirements and requirement phase for Identify Security Requirements,

Plan and Develop Framework for Risk Management and system requirements using DOCUMENTER / PROVER / MODELLER shown above in Fig 1. It also supports in calculating task analysis, impact analysis and risk analysis of requirement management on the basis of Multilevel Security Spiral (MSS) Model [18]. Iteration model is considered in this module for the requirement, time limit, cost expenditure and other project details.

### *A. The DOCUMENTER*

The DOCUMENTER supports the design and preparation of structured technical documents. In a standard file structure a project is just equivalent to a folder or directory. The project is built as an extension of existing projects through collecting related design constructs and documents.

The principal sub-components of DOCUMENTER are the *Document structure, Datastore*, the *Normative Design Document* (NDD), *traceability matrix and Tool Interface* (TI).

#### *1) Document structure*

The constructors of a formal document language help the DOCUMENTER to create documents. Documents are built through creating a hierarchy of blocks which is nested within each other. Fig 2, is an example which shows a mathematical expression, x + y = 3 ^ 2x - y = 3.

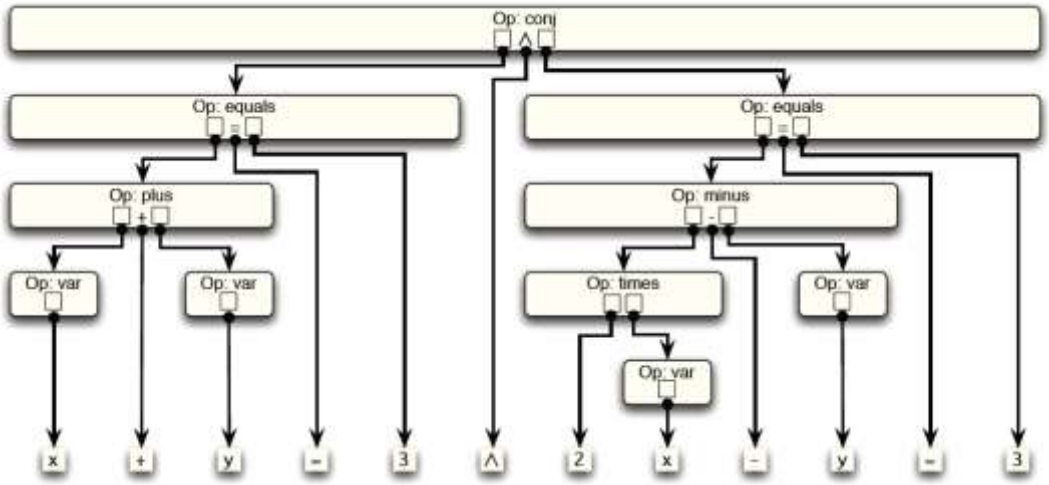


Fig. 2. The tree-like operational structure of a mathematical expression (Reprinted [14]).

In the TSSR, document production operators are used at all levels, the document is written through the constructor of algebra. The document language grammar is derived from that given algebra for each TSSR module. Branching is one of the examples of the mechanism which builds the block hierarchy. Blocks are collected into sorts; algebra is a collection of many sorts, similarly as a typed programming language structure. This provides meta-information through which the DOCUMENTER can "understand" the content of a TSSR document and there are a variety of ways in which this context-sensitivity is utilized [19]. The user selects a dummy block at each stage and inserts an operator, or string/number from the Datastore. The document language includes mechanisms for modifying the document's appearance, and each block in a document has attributes and it inherits the attribute of parents, as the standard WYSIWYG formatting style. The style collects presentation for each artifact together, allowing document modification in the same manner as cascading style sheets (CSS) used in HTML documents for modification.

#### *2) The Datastore*

Normally a project has one Datastore, or repository, data can be imported/exported from other projects to the Datastore. It is a standard database; all elements are collected in a single table, which is used to records their result sort and the sorts of their operator arguments [20]. Each row of a table records information of an element and referred as a *fact* about that element. In a given TSSR module or tool plug-in the tables are generated for information. The Column has a fixed sort and in the table all data are structured text. The Datastore has three functions Data Security, Data view and Data input, which is crucial to the user interaction with the DOCUMENTER. Data can be entered into TSSR documents directly from the Datastore, using drag-and-drop method into dummy blocks. TSSR has a central storage that makes effective handling of multiple documents that are probable to share data, which having different technical documents about the project. Every single change in the Datastore will search and replace that element in every document of the project. Data Security is done in two ways:

- *Cryptography*

Cryptography is one of the oldest techniques involved with security and confidentiality of the data. One of the essential aspects of secure communication, protecting passwords and user details from unauthorized users is known as cryptography. As cryptography is necessary for secure communication, protecting passwords is not sufficient by itself, there are some security requirements, including Authentication of the users, Privacy/

confidentiality of users data, integrity of data, Non-repudiation etc.

- *Security Techniques*

In TSSR Hill Cipher encryption and decryption method is used for database security. This cryptography approach provides additional level of security to the database system. The basic concept of this algorithm is that each letter of the alphabet is allotted a number starting from 0 in a continuous sequence. Alphabets are numbered as A=0, B=1, C=2, D=3, E=4……, Z=25 but this is not a fixed requirement of the cipher. In this the encryption takes n successive plain text letters and substitutes them for n cipher text letters [21].

In case of n=3, the encryption expressed in terms of matrix multiplication.

$$\begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} mod\ 26$$

Mathematical model:

Abbreviations used in this model are;

P = plain text,

C = cipher text,

K = key used for encryption of plain text to cipher text,

K-1 = Inverse of key K used for decryption of cipher text to plain text.

C = Ek(P) = KP mod 26

P = Dk(P) = K-1C mod 26 = K-1KP = P

K is the key matrix

K-1 is the matrix inverse. The inverse matrix can be calculated as K.K-1 = I where I is the identity matrix.

*3) The Normative Design Document*

A project contains exactly one NDD which is a structured document within the block hierarchy. NDD helps to enter all Datastore table entries into the project [22]. The NDD is literately programmed script through that the user controls all interactions with tools. A command supports the uniformity of tool interaction and inserted in the NDD to initiate the appropriate tool action. The Command is a special sort of structured text. The NDD is a linear script which allows the user to keep contextual focus whilst controlling several tools. In NDD the theorem prover tool will not accept or consider constants that have not been declared on associate earlier stage of the software development life cycle.

*4) Traceability*

Traceability is primarily used in software development projects to trace application, business, security or any other requirements for their implementation, testing or completion [23].

- *Bi-directional Traceability*

The direction of traceability information is called bi-directional traceability and even forward or backward traceability. Forward traceability is the ability to trace a requirement to components of a design or implementation. Backward traceability is the ability to trace a requirement to its source [24].

- *Traceability Matrix*

Traceability Matrix is primarily used in software development projects to trace business, application, security or any other requirements for their implementation, testing or completion. A Traceability Matrix is a worksheet type document consisting of tables.

- *Requirements Traceability Matrix*

The Requirements Traceability Matrix (RTM) captures the user with complete system requirements for the system, or a portion of that system. RTM captures all requirements with their traceability in a single document. It also provides bi-directional traceability between numerous associated requirements. Requirements tracing is the process of documenting the links between the user requirements for the system and the work products developed to implement and verify those requirements [25]. Document Unique ID is given which is associated with a specific requirement to easily trace that specific document shown in Table 1.

Table 1. Template of Requirements Traceability Matrix [25].

| Requirement ID | Requirement Type | Requirement description | Risks | Trace from user Requirement/ Trace to system Requirement | Trace to Design Specification | Unit Test case Integration Test Case | System Test cases | User Acceptance Test case | Trace to Test Script |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | | | | | | | | | |

- *Requirements Traceability Techniques*

There are so many requirements traceability techniques, but that were identified in the systematic review are briefly described below [26].

✓ *Information Retrieval (IR)*

Information Retrieval (IR) [27, 28, 29] approach is used to automate the generation of traceability links. Commonly IR methods include, Probabilistic Models (PM), Vector Space Model (VSM) and Latent Semantic Indexing (LSI). IR methods are generally based on similarity comparison and probabilistic value of two artifacts.

✓ *Event Based Traceability (EBT)*

Event Based Traceability (EBT) approach was to provide accurate maintenance of traceability relationships as publisher-subscriber relationship [30, 31]. This

relationship depends on object whenever a requirement change occurs; an event message is published, which is then notified to all.

✓ *Rule Based (RB) Approach*

Rule-based (RB) approach is to automatically generate traceability links using rules [32]. There are two requirement traceability rules: inter-requirement traceability rule and requirement to- object model traceability rule. The two rules are used for three specific documents such as use case documents (UCD), requirement statement document (RSD, and analysis object model (AOM).

✓ *Scenario-Based Approach (SB)*

According to [32, 33] scenarios are used to model system functionality and to generate functional test cases. Scenarios-based test cases create a mapping between requirements and other artifacts like design and code. The traceability is established by mapping scenarios with the design elements. Scenarios are created to trace only the interesting cases, therefore they might not provide complete coverage.

✓ *Goal Centric Traceability (GCT)*

The traceability of Non-Functional Requirements (NFR) is difficult. This is due to the fact that extensive interdependencies and trade-offs exists between them. Goal Centric Traceability (GCT) technique has been successfully evaluated in a case study with a real world project [27].

*4) The Tool Interface*

The TSSR interfaces are used for a broad range of external tools. Just an abrupt list would include: requirements analysis tools; theorem provers and model checkers; programming support tools; algebraic analysis tools and numerical analysis tools; system simulation tools; drawing tools; project management tools; and version control tools. The TSSR assumes that tools apply some function or commands to input data and produces some diagnostic output data. The NDD command scripts are used to input data to tools from the datastore, and are sent to the appropriate syntactic form rear to the tool. The TSSR received the tool output, inserted as appropriate into the Datastore and NDD. The NDD provides the linear program script which sequences the computations to be beatific and sent to the tool. The tool computation is controlled by the user through the NDD. Error messages are handled properly and displayed appropriately to the user. The TSSR architecture is abundant, flexible to handle a variety of different external tools through means of plug-ins, where as in the tool the plug-in encapsulates the operator syntax. Plug-ins interfaces with disambiguates and identical syntax has different meaning in different tools.

*B. The PROVER*

The PROVER will be a TSSR plug-in which provides theorem prover tools for high assurance of semantic checking of mathematical models [17]. Isabelle is a generic theorem prover, designed for interactive reasoning in a variety of formal theories [34]. Isabelle provides advantageous proof procedures for First-order logics, Constructive Type Theory, set theory, and higher-order logic. We briefly discuss the use and choice of Isabelle, what is involved in defining the plug-in, and also the benefits of the TSSR during this setting.

*1) Theorem provers*

A theorem prover tool provides a safe mechanism for applying set Theory and inference rules of an accustomed logic to accustomed axioms. The user extends a generic theorem prover to core meta-theory in which object logics can be constructed. Such tools provide an absolute powerful environment for acumen reasoning in complex or complicated settings without having the possibility of making the absent minded mistakes which are so prevalent, expensive and costly in human endeavors. Isabelle tool plug-in is considered for the project. The TSSR MODELLER modules principal application will be considered for high assurance system modelling. Isabelle is a generic theorem prover. It is written in standard XL. In a programmable meta-language inference rules are expressed as functions and the true propositions they construct are known as theorems. By writing programs in a meta-language, users automate the proof process as much as desired.

A recent development of a “front-end” Isar language which provides sophisticated support to the mathematician or developers for the formal definition of mathematical objects. The working environment is very close to the ‘natural’mode of proof for the practicing mathematician and it is machine-assisted and human readable [35].

Isabelle allows users to define theories which are a collection of named objects and theorems of principal interest in the usual mathematics, especially for system modelling, and the most advanced, is Higher Order Logic (HOL) [36]. It is also observed that the meta-logic (higher-order logic) has simple models and helpful for informal mathematical reasoning. The user can extend HOL to model a system of interest as required in named theories files. Due to this it will be the core inclusion in the theorem PROVER module.

Isabelle is an interactive proof tool. There are many significant reasons to select an interactive tool because it is well suited to the task of generic modelling. Isabelle also features efficient automatic reasoning tools.

The user must select the implemented proof tactics at each stage of the evolving proof. One of the reasons for selecting Isabelle is that it has no rules for inputs or outputs; they generally describe a relationship between premises bounds and conclusion that must hold in a valid proof [34].

There is no general algorithm for constructing a formal proof of any assets; therefore an automatic tool is needed to target a subclass of subgoals tightly. The interactive proof tool clearly provides the user a lot of power so that the user can extend the logical structure almost indefinitely.

The user often needs to accumulate and keep track of the logical changes involved; it is not necessarily an acceptable idea to have tactics which accomplish an

ample change from one proof state to the next. The machine-generated proof is unlikely to be very intuitive or useful for the users, when the automatic proof attempt does not succeed. TSSR modules target highly desirable automatic theorem provers and model checkers in a later phase.

*2) The PROVER plug-in*

Isabelle tool has a plug-in for the PROVER which has five effects on the DOCUMENTER:

- With appropriate menus it can extend the user interface to the tool;
- With appropriate tables it extends the DataStore to the Isar language;
- The Prover Plug-in incorporates and manages the files which is required by the tool, and producing them from the natural input to the NDD;
- Prover Plug-in provides the appropriate code for the tool interface which understands how to return from a given Isar command initiated via the user's NDD.
- Full traceability is provided by it, so that after completion the user can see all the links.

On the basis of Isar experience the PROVER's graphical interface improves significantly. The PROVER will allow significant automation of frequent activities. The PROVER will accomplish it accessible for users to analyze and inspect the new Isabelle code in powerful ways by storing a simple model of an Isabelle development in the Datastore. It will even be possible and accessible to include the results of such analyses directly in their documents to accept and update them automatically with changes to the code.

*C. The MODELLER*

In Fig1, the framework of the tool has a MODELLER module which is using the PROVER plug-in, that provide a comprehensive, high-assurance environment for the specification and design of critical engineering projects.

The MODELLER targets the design, modelling stages of system development, verification, intelligent graphical visualization of design, animation proof, and system architecture for the critical importance for successful acquisition of the projects. An extensive reasoning environment has been developed in Isabelle/Isar to support the TSSR approach to high-assurance developments [37]. A sophisticated GUI foreground end to this acumen reasoning environment which will be developed using the DOCUMENTER and PROVER infrastructure.

*1) Requirements capture and analysis*

The MODELLER requirements language has a feature like Z- schema calculus [38]. As we have known that the Z- schema calculus has proved to be an able and powerful tool for naming, structuring, grouping and reusing complex system requirements [17]. Moreover, for a formal notation, it has an almost advanced relatively user base. As such, it is a natural basis from which the TSSR requirements language is built. In the TSSR, the schema calculus is extended in the Object Z style with schema types [39]. This brings the schema calculus benefits of structuring and reuse to the description of complex component hierarchies and is a critical enabler for the treatment of very large systems [17].

*2) Horizontal Design Hierarchies*

Facilities for imposing a hierarchical structure which allows reuse of common components that are critical to the efficient specification and design of complex systems. In addition, evocative visualizations of designs can provide a powerful pedagogical aid [17]. In the TSSR, a familiar block diagram is adopted as the basal tool for system design. The TSSR block diagrams depict simple input/output components. They have well defined input and output interfaces; and allows the unrestricted use of internal interfaces, including feedback loops. An able and powerful typing discipline is imposed on all interfaces that include the use of structured schema types and polymorphism.

*3) Vertical Design Hierarchies*

To describe a system at different levels of abstraction and to demonstrate corresponding arguments between the different abstractions is an important mechanism for developing and communicating system assurance. Testing and reasoning must be used for appropriate correspondences of the demonstrated abstraction levels. Such a hierarchy of abstract system descriptions is alleged as a vertical design, because it spans a number of abstraction levels [17].

*D. The PLANNER*

*1) Security Requirements Engineering Process (SREP)*

The security requirements engineering process is a risk driven and assets based methods which focuses on security concepts write from the early phase of the software development lifecycle [8].

*2) Risk Management*

Risk management is the identification, assessment, and prioritization of risks. Risks can appear from uncertainty or ambiguity in financial markets, credit risk, threats from project failures, legal liabilities, accidents, natural or accustomed causes and disasters as able-bodied as a deliberate attack from an adversary, or events of uncertain or ambiguous or unpredictable root-cause [49].

The strategies to manage threats typically include transferring the threat to third parties, avoiding the threat, reducing or abbreviating the negative effect or probability of the threat, or even accepting some or all of the potential or absolute consequences of a particular threat, and the opposites for opportunities [40].

*3) Identify security objectives, dependencies and threats*

To determine the security objectives of the organization as well as legal requirements, we will take into account of security policies of that organization [51]. Assumption of security objectives for the environments are retrieved and made in this process [8]. Treats target asset which prevents security objective from being achieved. It is better to develop artifacts to develop new specific, generic threats or requirements and it is necessary to look for threats that are not related to or linked with the assets of the repository.

 

*4) Task Analysis*

Task analysis defined as the study of what a user is required to do to achieve a system goal in terms of actions and/or cognitive processes [41, 42]. Task analysis covers several techniques, but particular relevance to

- *Hierarchical Task Analysis (HTA)* is most commonly used task analysis technique. HTA technique is used to represent the relationships between tasks and subtasks [43].
- *Time Analysis (TA)* is used to document the temporal aspects of the tasks. TA technique is well suited for possible resource problems in performing tasks and represents task dependencies [43].
- *LINK analysis* is the simplest techniques, which is simply demonstrating the frequency of linkage between tasks. Most often link analysis is based on observational data that is why it can be easily performed [43].

Data about how target users plan to use the system-to-be is modelled in TSSR using personas and tasks. Although there is no agreed way of measuring the usability of a task with respect to its participating personas, persona and task, usability attributes match attributes found in the ISO 9241-11 framework [44][45].

$$U_T = \frac{\overline{t+f}}{2} + \bar{m} + \bar{c} \quad (1)$$

Each property has an associated value x, maps in the range $0 \leq x \leq 3$ and corresponds to the quantitative value of none, low, medium and high respectively.

Usability of a task T, written as $U_T$ and computed using formula 1

$\frac{\overline{t+f}}{2}$ is the mean task efficiency,

$\bar{m}$ is the mean task satisfaction and $\bar{c}$ is the mean tack effectiveness.

Variables t, f, m and c refer to task duration, frequency, mental demands and goal conflicts respectively.

*5) Risk Analysis*

A risk analysis involves identifying the most probable threats to project and analyzing the related vulnerabilities of the project to these threats [53]. The risk analysis process provides the foundation for the intact recovery planning effort [46]. Threats are synonymous with attacks, and vulnerabilities are properties of a system, making it liable to exploitation [45].

$$L_R = L_{T_R} - \overline{m_{T_R}} \quad (2)$$

Each element n is scored $0 \leq n \leq 3$ based on the value none, low medium or high.

$L_R$ is the likelihood of the thread, computed using formula 2, where

$L_{T_R}$ is the likelihood of the thread T associated with risk R.

$\overline{m_{T_R}}$ is the mean likelihood value for the thread TR

To score risks with respect to the perceived value of the assets threatened, we model a security property using a row vector [ciao], where c, i, a, and o represent the values held for confidentiality, integrity, availability and accountability respectively [47].

*6) Impact Analysis*

An impact analysis involves identifying the critical functions of the project and determining the impact of not performing the function beyond the maximum acceptable outage [48]. Impact as a part determined to be affected, and therefore worthy of inspection. Activity of identifying what to modify to accomplish a change, or the potential consequences of changes [50].

Risk impact *I* is illustrated by a security property, representing the value of the assets at risk from threat *T* [45].

$$I_R = \left(I_{T_R} \times I_{A_R}\right) - \overline{mI_R} \quad (3)$$

Risk impact $I_R$ is computed using formula 3, where

$I_{T_R}$ is the security property of associated with risk R,

$I_{A_R}$ is the security property of the vulnerable at risk.

$\overline{mI_R}$ is the mean security property for risk threat.

To perform impact analysis different search algorithms are identified. Each of these algorithms has been presented in at least one paper [52]:

- **Stochastically guided:** probabilities used for performing analysis and derived from the situation, using characteristics of the artifacts.
- **Unguided/exhaustive:** traceability information is used to identify a brute force solution. E.g transitivity closure algorithms.
- **Semantically guided:** the analysis is performed for objects and relationships using predetermined semantics.

## IV. Tool Interaction

TSSR used to read third party tool format and export to that format. As it is discussed earlier in this paper that TSSR is having plugin architecture, this option will be helpful in writing a TSSR plugin that recognizes files of third party tools. The third party's tool module will read TSSR means, or more likely the whole projects. A plugin for third party tool could be written in TSSR extension language (EXL) to input and allow transparent, export of the TSSR project into that tool. The Plugin would also be written in the third party tool's extension language, so that it will allow them to import TSSR project into them.

## V. Conclusion

We know that there are thousands of requirement management tools that are available in the markets in which few of them are free of cost. But for secure software development only a few tools are helpful.

 

In this paper, we introduce a framework of a new tool which is helpful for developing secure projects. The tool has four components; MODELLER, PLANNER, PROVER and DOCUMENTER, the tool is known as TSSR. The PLANNER module includes risk analysis, task analysis, which will be helpful in identifying, analyzing the security aspects of the project. The MODELLER module supports system modeling, system architecture and verification of the projects. It also provides intelligent graphical visualization of design and reasoning about concurrency. The PROVER provides intelligent management theory, literate reasoning, proof support and theorem proving for checking of mathematical models. The last module is the DOCUMENTER which supports the power indexing, structure view, editing of technical documents as well as it provides an interface to external tools. It also manages a central repository database which is using hill cipher encryption and decryption security technique so that the database will be secure. The Planner and Documenter modules are useful in developing secure software right from the initial stage in the vein of risk analyzing to secure central repository system.

## REFERENCES


[1] Anthony Finkelstein, Wolfgang Emmerich, "The Future of Requirements Management Tools," *In: Information Systems in Public Administration and Law*, 2000.

[2] Daniyal M Alghazzawi, Shams Tabrez Siddiqui, Mohammad Ubaidullah Bokhari, Hatem S Abu Hamatta, "Selecting Appropriate Requirements Management Tool for Developing Secure Enterprises Software," *IJITCS*, vol.6, no.4, pp.49-55, 2014. DOI: 10.5815/ijitcs.2014.04.06

[3] Rajat R Sud, James D Arthur, "Requirement Management Tools-A Qualitative Assessment," *Technical Report 03-10*, February 01 2003.

[4] G. Kotonvy and I. Sommerville, Requirements Engineering, *John Wiley & Sons, New York*, 1998.

[5] Shams Tabrez Siddiqui, "Needs, Types and Benefits of Requirements Management Tools," *International Journal of Trends in Computer Science*, Volume 2, Issue 11, 2013 ISSN: 7462 – 8452.

[6] Vineet Kumar Maurya, "Suraksha: A Security Designers' Workbench," *Presented at Hack.in 2009, IIT Kanpur, India*, 17-19 March 2009.

[7] Raimundas Matulevicius, "Process Support for Requirement Engineering: A Requirement engineering Tool Evaluation Approach," *Ph.D (Thesis), Department of Computer and Information science, Norwegian University of science and Technology. Trondheim*, 2005

[8] Daniel Mellado, "A common criteria based security requirements engineering process for the development of secure information systems," *Computer Standards & Interfaces 29*, 244–253. Elsevier, 2007.

[9] K. Wiegers, Software Requirements, *Microsoft Press, Redmond, Wash.*, 1999.

[10] Jun Han, "TRAM: A Tool for Requirements and Architecture Management," *in the Proceedings of the 24th Australasian Computer Science Conference, Gold Coast, Australia*, 2001.

[11] "Requirements Management with Enterprises architect," *by Sparx system*, 2010 version1.3 Website: www.sparxsystems.com

[12] M U Bokhari, Shams T Siddiqui,"A Comparative study of software requirements tools for secure software Development," *BVICAM"S International Journal of IT (BIJIT)*, 2010: 207-216.

[13] Matthias Weber and Joachim Weisbrod, "Requirements Engineering in Automotive Development: Experiences and Challenges," *IEEE Software*, 2003.

[14] M. Hoffmann, N. Kuhn, M. Weber, and M. Bittner, "Requirements for requirements management tools," *in RE '04: Proceedings of the 12th IEEE International Conference on Requirements Engineering, Washington, DC, USA*, 2004, p. 301–308.

[15] Rajat R Sud, "Requirement Management Tool: Assessment and Comparison," *Report*, February 15, 2012- Version 10.

[16] Raimundas Matulevičius1, Patrick Heymans1, and Guttorm Sindre2, "Comparing Goal-Modelling Tools With The Re-Tool Evaluation Approach", *ISSN 1392 – 124X Information Technology And Control*, 2006, Vol.35, No.3A

[17] Tony Cant, Jim McCarthy and Robyn Stanley, "Tools for Requirements Management: a Comparison of Telelogic DOORS and the HiVe", *Defence Science and Technology Organisation*, 2006

[18] Shams Tabrez Siddiqui, "Multilevel Security Spiral (MSS) Model: NOVEL Approach", *International Journal of Computer Applications* (0975 – 8887) Volume 65– No.20, March 2013

[19] Unicode. http://www.unicode.org/.

[20] Shams-ul-Arif, Qadeem Khan, S. A. K. Gahyyur, "Requirements Engineering Processes, Tools/Technologies, & Methodologies", *International Journal of Reviews in Computing*, 2009.

[21] Jasdeep Singh Bhalla, "A Database Encryption Technique to Enhance Security Using Hill Cipher Algorithm," *International Journal of Engineering and Advanced Technology (IJEAT)* ISSN: 2249 – 8958, Volume-2, Issue-4, April 2013.

[22] Francesca Ricciardi, "Design and Normative Claims in Organization Studies: A Methodological Proposal," *Lecture Notes in Information Systems and Organisation 1*, DOI: 10.1007/978-3-642-33371-2_2, _ Springer-Verlag Berlin Heidelberg 2013

[23] V Kumar, R Thareja, "Goal Structured Requirement Engineering and Traceability Model for Data Warehouses," *International Journal of Information Technology and Computer Science*, vol. 12, pg. 78-85. 2013

[24] R. J. Wieringa, "An Introduction to Requirements Traceability," *Vrije Universiteit, Faculty of Mathematics and Computer Science, Amsterdam*, 1995.

[25] Software Testing-Requirements Traceability Matrix.Website:http://www.etestinghub.com/requirements_traceability_matrix.php

[26] Uzair Akbar Raja and Kashif Kamran, "Framework for Requirements Traceability- TLFRT supporting pre-RS & post-RS traceability," *School of Engineering Blekinge Institute of Technology. Master Thesis*, 2008

[27] J.Cleland-Huang, R. Settimi, O.B. Khadra, E. Berezhanskaya, and S. Christina, "Goal-Centric Traceability for Managing Non-Functional Requirements," *Proceedings of the 27th international conference on Software engineering ICSE '05, ACM*, pp. 362-371. 2005

[28] J. Huffman Hayes, A. Dekhtyar, and J. Osborne, "Improving Requirements Tracing via Information

Retrieval", *Proceedings of 11th IEEE International Requirements Engineering Conference*, IEEE CS Press, 2003, pp.138-147.
[29] J. Cleland-Huang, R. Settimi, C. Duan, and X. Zou, "Utilizing supporting evidence to improve dynamic requirements traceability," *Proceedings of IEEE Intentional Requirement Engineering Conference*, 2005, pp. 135–144.
[30] J. Cleland-Huang, C. Sethi, G. Javvaji, and K. Xia, "Automating speculative queries through event-based requirements traceability*," Proceedings of the IEEE Joint International Requirements Engineering Conference (RE'02),* 2002, pp. 289- 296.
[31] J. Cleland-Huang, C.K. Chang, and M. Christensen, "Event-Based Traceability for Managing Evolutionary Change," *IEEE Transactions on Software Engineering*, vol. 29, no. 9, IEEE, 2003, pp. 796-810.
[32] F. Blaauboer, K. Sikkel, M.N. Aydin, "Deciding to Adopt Requirements Traceability in Practice," *Springer Lecture Notes in Computer Science*, vol. 4495/2007, pp 294-308
[33] J.Cleland-Huang, "Toward Improved Traceability of Non-Functional Requirements", *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering TEFSE'05*, ACM, 2005, pp. 14-19.
[34] L. C. Paulson and T. Nipkow, "Isabelle: A Generic Theorem Prover," *in volume 828 of LNCS.Springer-Verlag*, 1994.
[35] Markus Wenzel, "Isar — a generic interpretative approach to readable formal proof documents," *In Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, and L. Thery, editors, Theorem Proving in Higher Order Logics: TPHOLs '99, volume 1690 of LNCS*, 1999.
[36] T. Nipkow, L. C. Paulson, and M. Wenzel. Isabelle's Logics: HOL, 2001. *Part of the Isabelle distribution*, http://isabelle.in.tum.de/doc/logics-HOL.pdf.
[37] B. P. Mahony. The DOVE approach to the design of complex dynamic processes. 2002. In 'TPHOLs 2002 (Track B)', http://techreports.larc.nasa.gov/ltrs/PDF/2002/cp/NASA-2002-cp211736.pdf.
[38] J. M. Spivey. The Z Notation: A Reference Manual. *Second edn*, Prentice Hall International.
[39] G. Smith. Logic for object-Z, *Technical Report 94-48, Software Verification Research Center, The University of Queensland*, 1994.
[40] Online on: http://en.wikipedia.org/wiki/Risk_management, March 14, 2014.
[41] B. Kirwan, and L (eds). Ainsworth, K. A Guide to Task Analysis. *Taylor & Francis Ltd*. 1992.
[42] J Richardson, T C Ormerod, A Shepherd, "The role of task analysis in capturing requirements for interface design," *Interacting with Computers*, 1998.
[43] Pedro J. Valderas Aranda," A Requirements Engineering Approach for the Development of Web Applications," *Department of Information Systems and Computation Technical University of Valencia, Thesis*, November 2007.
[44] ISO. ISO 9241-11. Ergonomic requirements for office work with visual display terminals (VDT) s - Part 11 Guidance on usability. Technical report, 1998.
[45] Shamal Faily, "A framework for usable and secure system design," *Ph.D(Thesis) Wolfson College University of Oxford*, Hilary Term 2011
[46] Geoffrey H. Wold and Robert F. Shriver, "Risk Analysis Techniques," *Disaster Recovery Journal*© 1997.
[47] IEC. IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems. Parts 1-7. International Electro technical Commission, Switzerland, 1998-2005.
[48] B.J.M. Abma, "Evaluation of requirements management tools with support for traceability-based change impact analysis," *Master's Thesis*, September 10, 2009.
[49] Tousif ur Rehman, Muhammad Naeem Ahmed Khan, Naveed Riaz, "Analysis of Requirement Engineering Processes, Tools/Techniques and Methodologies," *In I.J. Information Technology and Computer Science*, 2013, 03, 40-48.
[50] S.A Bohner, and R.S Arnold, "Software change impact analysis," *IEEE Computer Society Press, Los Alamitos, Calif.*, 1996.
[51] M.U.Bokhari, "Metrics for Requirement Engineering and Automated Requirement Tools," *Proceedings of the 5th National Conference; INDIACom-2011, New Delhi*, 2011
[52] S. A. Bohner, "Software Change Impacts - An Evolving Perspective," *Proc. IEEE International Conference on Software Maintenance, Montreal, Canada*, pp. 263-272, 3-6 October, 2002.
[53] Muhammad Naeem Ahmed Khan, Muhammad Khalid, Sami ul Haq, "Review of Requirements Management Issues in Software Development," *IJMECS*, vol.5, no.1, pp.21-27, 2013.DOI: 10.5815/ijmecs.2013.01.03

## Authors' Profiles

**Prof (Dr) Mohammad Ubaidullah Bokhari** is currently working as a Professor and Chairman, Department of Computer Science, AMU, Aligarh. He has published more than 90 research papers in different reputed journals and conference proceedings. He has also authored 5 books on different fields of Computer Science. His current research interests are Requirement Engineering, cryptography, Software Reliability, Wireless Network Security and Database.

**Shams Tabrez Siddiqui** received hisB.Sc and MCA degree from Aligarh Muslim University, Aligarh, India in 2003 and 2007. He is pursuing Ph.D in Software Requirement Engineering from AMU, Aligarh. He is also working as a counselor for IGNOU. He has published 12 research papers in different reputed international/ national journals and conference proceedings. His research interest includes Requirement Engineering, Software Engineering and Software Security. He is a member of the Computer Society of India, ACM, IAENG and IACSIT.

**How to cite this paper:** Mohammad Ubaidullah Bokhari, Shams Tabrez Siddiqui,"TSSR: A Proposed Tool for Secure Software Requirement Management", International Journal of Information Technology and Computer Science(IJITCS), vol.7, no.1, pp.1-11, 2015. DOI: 10.5815/ijitcs.2015.01.01