Int Jr of Advanced Computer Engineering and Architecture
Vol. 1, No. 1, June 2011

www.ijacea.yolasite.com

A Framework to Detect and Analyze Software Vulnerabilities: Analysis Phase in SDLC

Mr.Manoj Ashok Wakchaure[1], Prof. Dr. Shashank D. Joshi[2]
PG Student[1] and Professor[2], Department of Computer, B.V.U.C.O.E, Pune
Email: [1]manoj13apr@gmail.com, [2]sdj@live.in

***ABSTRACT* – Security is a major problem in software development. Software defects lead to security Software is an important part of everyday life. Large number of peoples using internet, ATM, Mobile phones for their need. So for developing any software security is very much important things. Better way providing security from the initial stage of software development life i.e. security from requirement stage to the deployment of any software product. Security enhancement of the SDLC process mainly involves the adaptations or augmentations of existing SDLC activities, practices,and checkpoints, and in a few instances it may also entail the addition of new activities, practices .Secure software is not easily achieved and the actual scenario is that investments in software development process improvement do not assure software that resist from attacks or do not current security vulnerabilities. It can be demonstrated that changes to the software Development process can help to minimize the number of vulnerabilities in new or developing software. Colleges and Universities play a important role in the education and training of software engineers. This paper gives new way of checking Different conditions which are helpful for detecting the vulnerabilities in requirement phase of software development life cycle.**

***Index Terms:* Security, SDLC, Requirement, Analysis, Vulnerabilities.**

I. INTRODUCTION

The vulnerabilities are being exploited at an increasing rate, costing businesses millions of dollars each year and threatening the security of the Nation and of individuals. Large Number of cyber vulnerabilities can be traced to defects in software that are because of bad Analysis, bad design and poor development method. Defective software is largely the result of poor software engineering techniques that often “add on” security considerations rather than include them from the beginning.

If software vulnerability is to be improved, universities must change the way that software design and development is being taught and industry must demand software that has fewer defects and is less prone to exploitation. This paper presents a discussion of the Vulnerabilities Detection by using certain checklist. That means particularly taken the example of Geographical Database. This paper only focused on The Requirement Analysis Phase of software Creating Methodology. Software Engineering Institute (SEI) at Carnegie Mellon University analyzed thousands of programs written by thousands of programmers and concluded that even experienced professionals unintentionally include defects when writing software. These defects result from the determination of requirements, designing, developing or testing the software. According to this study done by the SEI, over 90% of security vulnerabilities are the result of known software defects. This study demonstrated that 10 common defects accounted for about 75% of all security vulnerabilities [1].

Another study of more than 45 business applications showed that 70% of security defects were caused by poor software design and perhaps poor Analysis [3].Many Times developers are preceding their work without considering the security issues or vulnerabilities detection. The most part, software design and development errors that lead to security breaches include a basic lack of security knowledge, errors declaration, and failure to input validation, buffer overflows, and logical errors. The Director of Research at the Sans Institute observed that all of the conditions listed on the Sans Institute High Level Internet Security vulnerabilities is because of poor coding, Testing and bad software engineering. But all of these flaws are avoidable and preventable. The cause of the problem has been identified as coming from multiple sources; education of programmers, business practices of software development firms and application deployment practices of end-users [4].

This paper addresses the role of Detecting and preventing Different vulnerabilities at the analysis stage with respect to the problems of poor software development practices. Programmers are from Different areas means both Computer Science and Information Systems programs, as well as a wide range of other technical programs. we also be able to develop a system which is useful for detecting vulnerabilities of any domain (like financial, hospital, company etc.).So that we are able for finding report on requirements and it will classify that vulnerability. The application of these concepts and exposure of the number of objectives for a geographical database and the design that meets the objectives have been described. The prototype of the database using a simplified strictly tree structured hierarchical classification scheme has been implemented using the commercial database management system.

II. BACKGROUND

Application security means many different things to many different people. Security should be explicitly at the requirement level. Security requirements must cover both overt functional security (say the use of applied cryptography) and emergent characteristics. One great way to cover the emergent security space is build abuse cases. Similar to use cases ,abuse cases describe the systems behavior under attack,buiding them requires explicit coverage of what should be protected from whom and far how long[14].

The software engineering techniques currently being taught at colleges and universities and being used by industry do not Supply the Required secure software, changes to the common methods must be implemented.

A recent study at SEI found that defects can be reduced to an average of 0.06 defects per KLOC when rigorous processes are followed [1]. This compares to an industry average of 1 to 7 defects per KLOC [2].There are Different number of tools and techniques for making secure software, But all of them are applied in an ad-hoc way. While developing secure software Risk assessment factor is also necessary. The vulnerability analysis database contains all vulnerable data.

The question is if the secure software engineering principles are being taught in the majority of these colleges, why is there still a problem with defects in released software? There are different points to discuss. There are millions of lines of code that have existed in industry for generations. Second, the quality of the programs was not assessed. Also the sample for the survey included only NSA schools of academic excellence. The vast majority of colleges and universities were not included. No community colleges were included in the survey. So that there is a need to change the method for teaching the functions of Software development [15].

Different methods for teaching software development begin with an initial step of programming class. Time is not given to this security issues. Rather than this other courses contains computer network, data communication database management, analysis and design. Security methods are under the high level class and are considered to be an add in to the original software. The habits formed from initial programming can be for a long time. Having students focus repeatedly on issues of syntax, and primitive details of data structures, control structures, etc. forms habits associated with this level of concern. Higher level issues, testability, requirements, security and maintainability may be covered late in coursework, but never to the degree to form strong work habits. To change programmer behavior will continually run against initial habits formed early in their educational experience [5].

III.Integrating security into SDLC

Security enhancement of the SDLC process mainly involves the adaptation or augmentation of existing SDLC activities, practices, and checkpoints, and in a few instances, it may also entail the addition of new activities, practices, or checkpoints. In a very few instances, it may also require the elimination or wholesale replacement of certain activities or practices that are known to obstruct the ability to produce secure software.

Organizations can insert secure development practices into their software life cycle process either by adopting a codified secure software development methodology, such as Enhancing the Development Life Cycle to Produce Secure Software , and the SDLC Process content area of Build Security In, or through the evolutionary security enhancement of their current practices, as described in Sections of Enhancing the Development Life Cycle to Produce Secure Software and in the Best Practices and Knowledge sections of Build Security In.

These, as well as the other Best Practices, Knowledge, and Tools articles on Build Security In support organizations in making progress toward achieving these goals. Those responsible for ensuring that software and systems meet their security requirements throughout the development life cycle should review, select, and tailor BSI guidance as part of normal project management activities. Additional Resources on BSI and the references below provide additional, experience-based practices and lessons learned that development organizations need to consider.

Secure software is software that is in and of itself robust against attack. This means that software will remain dependable even when that dependability is threatened. Secure software cannot be subverted or sabotaged. In practical terms, this software lacks faults or weaknesses that can be exploited either by human attackers or by malicious code. Compared with other software properties, security is still not well understood: at no point in the software development life cycle (SDLC) are developers or users able to determine with 100 percent certainty that the software is secure nor, to the extent that it is considered secure, what makes it so. Software security is a dynamic property—software that is secure in a particular environment within a particular threat landscape may no longer be secure if that environment or threat landscape changes or if the software itself changes. In terms of “testability,” security is also difficult to gauge. Software testers can run 10,000 hours of testing and ultimately be very confident that the software that passes those tests will operate reliably.

Currently there is a lack of standardization over what constitutes an application security assessment. With no single set of criteria being referenced, it is suggested that OWASP

establish a set of standards defining and establishing a baseline approach to conducting differing types/levels of application security assessment. The standards should be flexible in design to accommodate a range of security assurance levels. The standards should not be viewed as placing requirements on any party. Rather, the standards should make recommendations about what should be done to be consistent with what the OWASP community believes is best practice. Adhering to the standards should help increase end user organization confidence that assessments meet an industry agreed-upon approach.

IV. IMPLEMENTATION

If we are taking the example of Geographical information system i.e. GIS. In this database features are like spatial and Non-spatial. Again Non-Spatial having Arbitrary and Classifications .In spatial ,Chain,Region,Feature.Another part of Geographical database is Topology ,It is again divided into incidence ,containment and exclusion.Cultural Features are Aviation and Urban.Aviation means Construction ,Airport etc..In Urban building, park likewise [16].

Fig 1 shows geographical information's for Analysis. it gives an informal graphical overview of the design of the database. the tree structure depicts each entity in the database, as well as the components of which it consists. In figure curly bracket denotes sets of entities, whereas angle bracket denotes arrays of entities that mean order is important. In given database consists of two components i.e. set of features and a collection of topologies on the spatial attributes of the features.

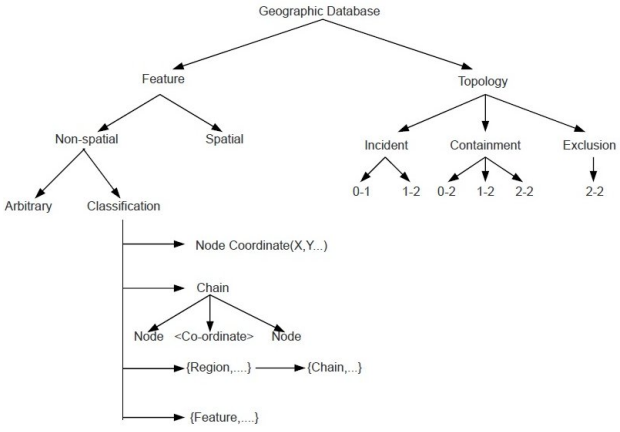

Fig 1: Geographical Database Domain.

Figure 2 shows an example of a partially ordered set defining some classes for cultural features. The vertices are distinct feature classes, and arcs denote a major class and subclass

relation. The topology of a GIS contains information about the structure of spatial attribute.Incidence; Containment and exclusion are topological relationships.

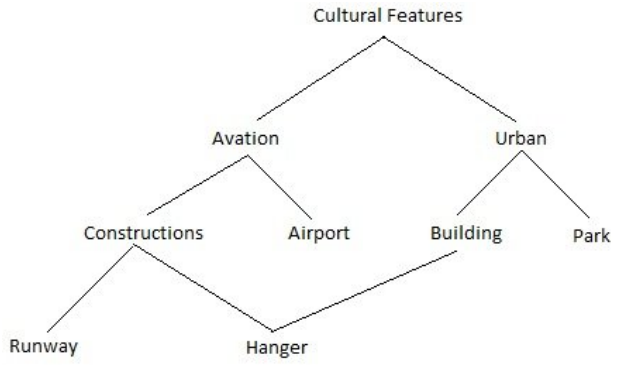

Fig 2: Cultural Features of Geographical Database.

In our developed framework this databases will show in report. So that no requirement vu In this work different points are taken just like cheaklist.this snapshot important points are cheak out one by one.

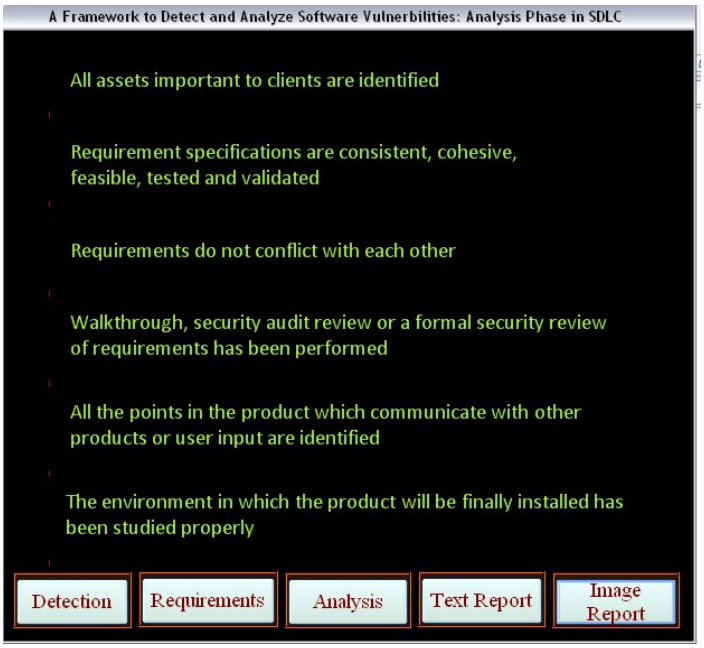

Fig 3:Detect & Analyze software Vulnerabilities.

Next step go through the Requirements option and put the requirements ,sub requirements and so on .In this snapshot only taking requirements and some sub requirements .After putting requirements go on requirement done option.

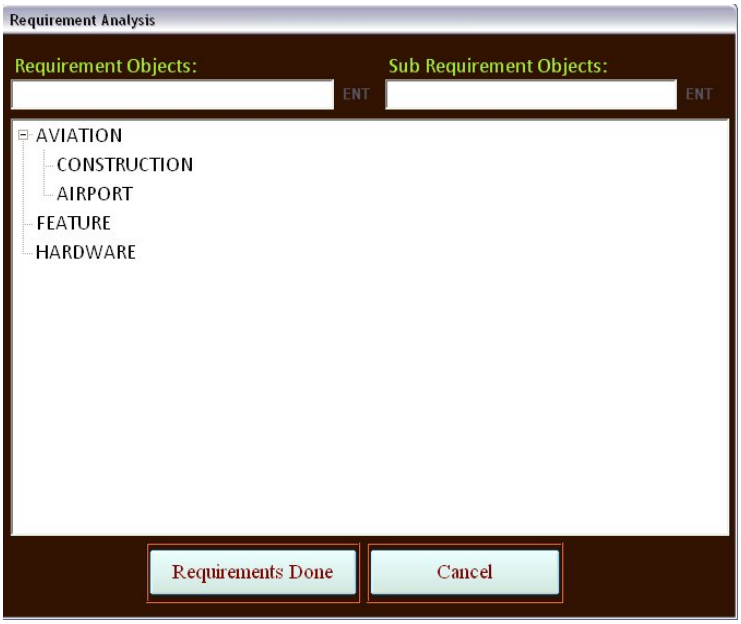

Fig 4:Requirement Analysis with missing requirements

For the detection of several points go on detection option .it is showing that True and False option depending on requirements. In following snapshot first two checkpoints are not fulfill. Third to fifth checkpoints are fulfill but not last.

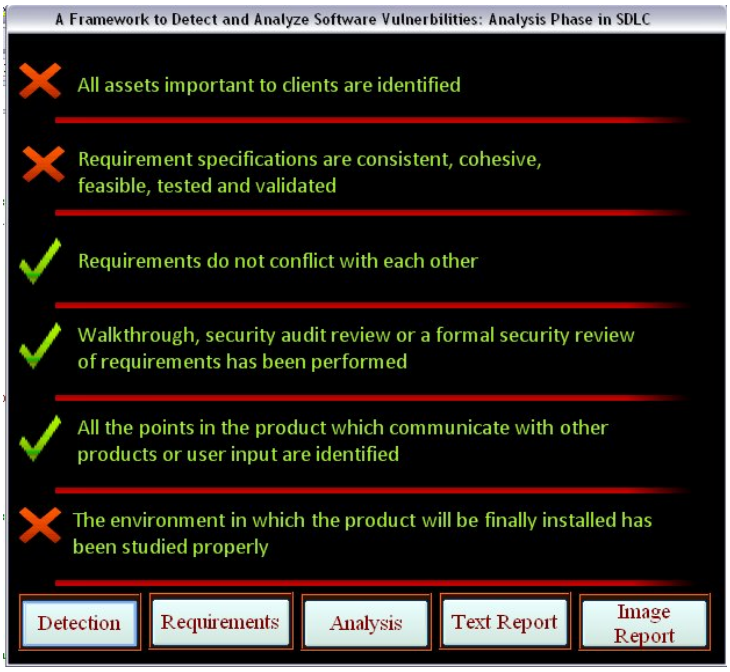

Fig 5:A framework for detect software Vulnerabilities

Then click on Analysis it shows that the missing requirements and that is called as Vulnerability Analysis.

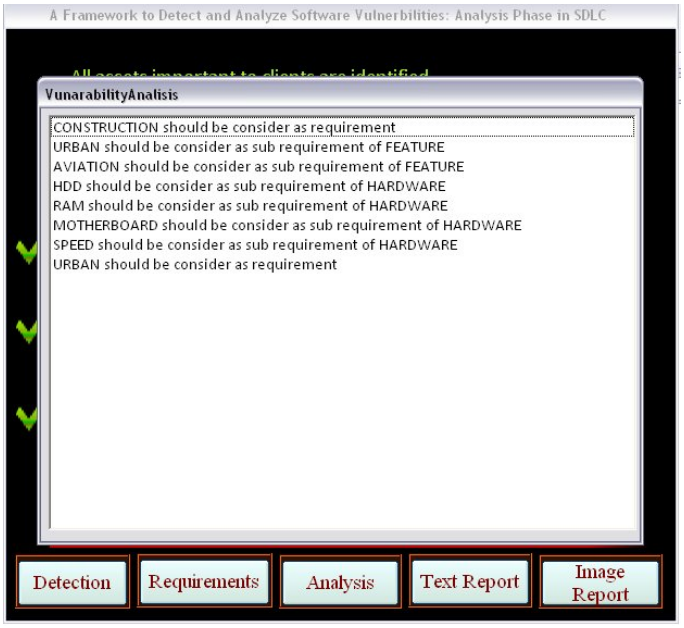

Fig 6:Vulnerabilities Analysis.

If we put all requirements with its sub requirements of certain geographical database system.

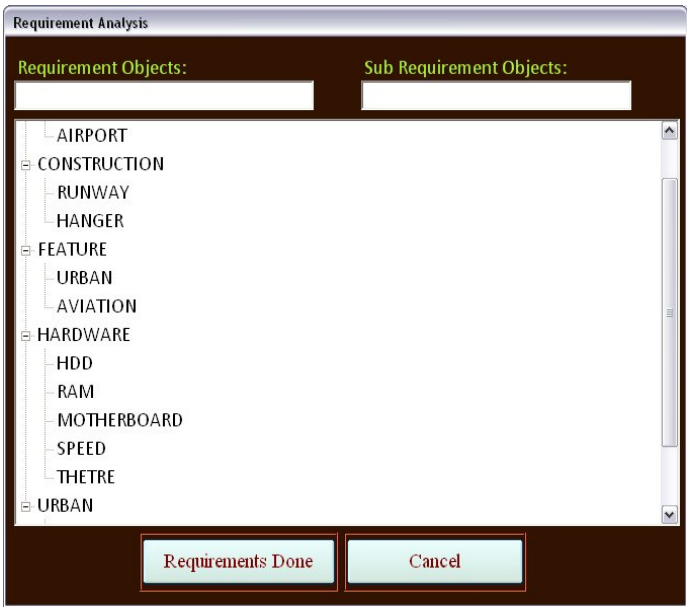

Fig 7:Requirement Analysis with all Requirements

Then it shows all checkpoints are correct. Means this system find out the missing requirements and errors .

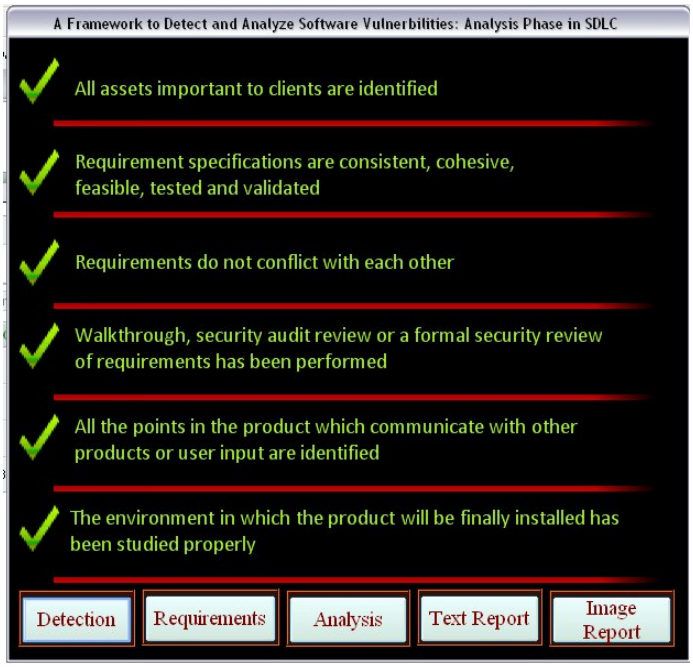

Fig 8: A framework with all Points Fulfill.

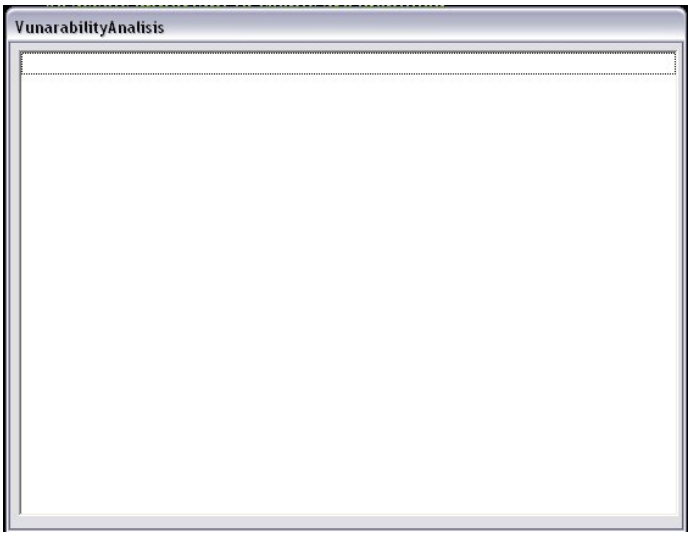

Fig 9: Vulnerabilities With No Suggestions

V. CONCLUSION

Before you start trying to solve a problem it is very much important to study the existing system before embarking on major changes. It has been demonstrated that software defects lead to security vulnerabilities. Analysis: It is nothing but Requirements review. This allows for definition of security requirements to ensure that security is well defined and included in the design phase.

Finally a framework it can be detect and analyze different vulnerabilities.Requirment Analysis will checkout the different vulnerabilities by using that condition criteria. we can

just applying that conditions on geographical database i.e. geographical information system .the system will collecting the all requirement of the particular Domain which is required for achieving secure product. Now actually the secure product means the product which is not having vulnerabilities or very less number of vulnerabilities. In our software development life cycle we are just carried out the development step by step from requirement gathering up to deployment of software product. But In this system Our major target is only on Analysis Phase of SDLC.Until and unless the given conditions are not valid or satisfy for the particular domain it will shows lacking things in it. Secure Software engineering in the same manner that schools teach mechanical engineering or Different Basic engineering discipline. The main difference between the current development and this one is to integrate the security while initial phase of the software development process.

This work shows that mapping of activities to vulnerabilities. Only problem is that Applying security for every phase is costly and Time consuming. Requirements analysts, designers, programming, testing and management functions. It is not intended that this methodology change will remove the errors or vulnerabilities in new software of in changed software.

REFERENCES

[1] Davis, N. and J. Mullaney, The Team Software Process in Practice: A Summary of Recent Results. 2003, Software Engineering Institute,Carnigie Mellon University.

[2] Jones, C., Software Assessments, Benchmarks, and Best Practices. 2000, Reading, MA: Addison-Wesley. 659.

[3] Jacquith, A., The Security of Applications: Not All Are Created Equal. 2002, @Stake Research. p. 12.

[4] Bishop, M. and S. Engle. The Software Assurance CBK and University Curricula. in Proceedings of the 10th Colloquium for Information Systems Security Education. 2006. University of Maryland, Adelphi, MD

[5] Conklin, W.A. Bottom-Up meets Top-Down: A new Paradigm for Software Engineering Instruction. in Proceedings of the 10th Colloquium for Information Systems Security Education. 2006. University of Maryland, University Collage, Adelphi, MD

[6] Howard, M., D. LeBlanc, and J. Viega, 19 Deadly Sins of Software Security 2005: McGraw-Hill Osborne Media. 304.

[7] Howard, M. and D.C. LeBlanc, Writing Secure Code. Second Edition ed. 2002: Microsoft Press. 650.

[8] Institute, S.E., Build Security In. 2006, Strategic Initiatives Branch of the National Cyber Security Division (NCSD) of the Department of Homeland Security (DHS) https://buildsecurityin.uscert. gov/portal/.

[9] Schneider, B., Applied Cryptography: Protocols, Algorithms, and Source Code in C. 1995: Wiley.784.

[10] Schneider, B., Secrets and Lies: Digital Security in a Networked World 2004: Wiley. 448.

[11] Bishop, M. and B.J. Orvis. A Clinic to Teach Good Programming Practices. in Proceedings of the 10th Colloquium for Information Systems Security Education. 2006. University of Maryland, University , Adelphi, MD

[12] ACM, Computing Curricula Information Technology Volume. 2005, ACM. p. 115.

[13] ACM/IEEE Computer Society, Software Engineering 2004.

[14] J.Viega"The CLASP Application Security Process ", Volume 1.1, Training Manual.

[15] Wm.Arthur Conklin and Glenn Dierich: Secure "software engineering a new paradigm", Proc. Of the 40th annual Hawaii international conference, IEEE,2007.

[16] Willem van billion, "A geographical database system "National research institute for math. Science.