International Journal of Engineering Sciences & Emerging Technologies, Dec. 2012.
ISSN: 2231 – 6604 Volume 4, Issue 1, pp: 133-143 ©IJESET

SECURE SOFTWARE DEVELOPMENT PROCESS FOR EMBEDDED SYSTEMS CONTROL

Sanjai Gupta[1], Md Faisal[2], Mohammed Hussain[3]

[1]Department of Computer Science & Engineering, CMJ University, Meghalaya, India
[1]guptasanjay3@gmail.com

[2]Department of Information Technology, Nizwa College of Technology, Oman
[2]faisal31621@gmail.com

[3]Department of Computer Science & Engineering, M G Institute of Management & Technology, Lucknow, UP, India
[3]mohd.husain90@gmail.com

ABSTRACT

Security is becoming a major concern in embedded system designing and development. Security requirements must be tackled early in software design and embedded in corresponding business process models. The purpose of this paper is to present modified process model and overview information about existing processes, standards, life cycle models, frameworks, and methodologies that support or could support secure software development for embedded System Control. Where applicable and possible, some evaluation or judgment is provided.

KEYWORDS— *Software Process Models, Embedded Systems Control, Security Assurance, Standard.*

I. INTRODUCTION

Embedded systems are vital to industry and society. Today, security in one form or another is a requirement for an increasing number of embedded systems, ranging from low-end systems such as PDAs, wireless, handsets, networked sensors, and smart cards, to high-end systems such as routers, gateways, firewalls, storage servers, and web servers. Technological advances that have spurred the development of these electronic systems have also ushered in seemingly parallel trends in the sophistication of security attacks. It has been observed that the cost of insecurity in electronic systems can be very high. With the evolution of the Internet, information and communications security has gained significant attention.

The purpose of this paper to integrate security into their standard software development processes. It is also relevant for developers and managers looking for information on existing software development life cycle (SDLC) processes that address security.

In this paper we have described existing frameworks and standards such as the Capability Maturity, the Systems Security Engineering Capability Maturity Model (SSE-CMM), in addition to existing processes such as the Microsoft Trustworthy Computing Software Development Lifecycle, the Team Software Process for Secure Software Development (TSPSM-Secure), and Agile Methods. In the last we have extended the Systems Security Engineering Capability Maturity Model (SSE-CMM) by using USING MS-TCSDL, TSP-Secure, and Agile Method.

This paper is documented in seven sections: the introduction section describes the history, purpose, and the general concepts and principles of security evaluation, and describes the model of evaluation. The second section defines common term used in this paper. The third section of the document describes some common software process model, which includes various methods of assuring that a product is secure. The fourth section explores Systems Security Engineering Capability Maturity

Model (SSE-CMM) and in the fifth section we have extended SSE-CMM. The sixth section of the document contains Component Structure Diagram of Extended SSE-CMM and in the last we have conclude this paper.

II. PROCESS, PROCESS MODEL, STANDARD, SECURITY ASSURANCE

A. Process

The Institute of Electrical and Electronics Engineers (IEEE) defines a process as "a sequence of steps performed for a given purpose" [1]. A secure software process can be defined as the set of activities performed to develop, maintain, and deliver a secure software solution. Activities may not necessarily be sequential; they could be concurrent or iterative.

B. Process Model

A process model provides a reference set of best practices that can be used for both process improvement and process assessment. Process models do not define processes; rather, they define the characteristics of processes. Process models usually have architecture or a structure. Groups of best practices that lead to achieving common goals are grouped into process areas and similar process areas may further be grouped into categories. Most process models also have a capability or maturity dimension that can be used for assessment and evaluation purposes.

There is no guarantee that even when organizations conform to a particular process model, the software they build is free of unintentional security vulnerabilities or intentional malicious code. However, there is probably a better likelihood of building secure software when an organization follows solid software engineering practices with an emphasis on good design, quality practices such as inspections and reviews, use of thorough testing methods, appropriate use of tools, risk management, project management, and people management.[2]

C. STANDARD

Standards are established by some authority, custom, or by general consent as examples of best practices. Standards provide material suitable for the definition of processes. [2]

D. SOFTWARE PROCESS

Although the term "security assurance" is often used, there does not seem to be an agreed-upon definition for this term. In the Capability Maturity Model for Software (SW-CMM), the purpose of "software assurance" is described as providing appropriate visibility into the process being used by the software projects and into the products being built [2]. The SSE-CMM describes "security assurance" as the process that establishes confidence that a product's security needs are being met. In general, the term means the activities, methods, and procedures that provide confidence in the security-related properties and functions of a developed solution.

In the Security Assurance section of its Software Assurance Guidebook, the National Aeronautics and Space Administration (NASA) defines a minimum security assurance program is one that ensures the following:

i. A security risk evaluation has been performed.
ii. Security requirements have been established for the software and data being developed and/or maintained.
iii. Security requirements have been established for the development and/or maintenance process.
iv. Each software review and/or audit includes evaluation of security requirements.
v. The configuration management and corrective action processes provide security for the existing software and that the change evaluation processes prevent security violations.
vi. Physical security for the software and the data is adequate.

Security assurance usually also includes activities for requirements, design, implementation, testing, release and maintenance phases of a SDLC. [2]

III. MS-TCSDL, TSP-SECURE, AND AGILE METHOD

A. MS-TCSDL-Microsoft Trustworthy Computing Security Development Lifecycle

Microsoft's Trustworthy Computing Security Development Lifecycle (SDL) is a process that they have adopted for the development of software that needs to withstand security attacks. The process adds a series of security-focused activities and deliverables to each phase of Microsoft's software development process. These security activities and deliverables include definition of security feature requirements and assurance activities during the requirements phase, threat modelling for security risk identification during the software design phase, the use of static analysis code-scanning tools and code reviews during implementation, and security focused testing, including "fuzz testing" during the testing phase. [3] An extra security activity includes a final code review of new as well as legacy code during the Verification phase. Finally, during the release phase, a Final Security Review is conducted by the Central Microsoft Security team, a team of security experts who are also available to the product development team throughout the development life cycle, and who have a defined role in the overall process [4].

Microsoft has augmented the SDL with mandatory security training for its software development personnel, security metrics, and with available security expertise via the Central Microsoft Security team. Microsoft is reporting encouraging results from products developed using the SDL, as measured by the number of critical and important security bulletins issued by Microsoft for a product after its release.

B. TSP-Secure-Team Software Process for Secure Software Development

The SEI's Team Software Process (TSP) provides a framework, a set of processes, and disciplined methods for applying software engineering principles at the team and individual level [5]. Software produced using the TSP has one or two orders of magnitude fewer defects than software produced with current practices—that is, 0 to .1 defects per thousand lines of code, as opposed to 1 to 2 defects per thousand lines of code [6].

TSP for Secure Software Development (TSP-Secure) extends the TSP to focus more directly on the security of software applications. The TSP-Secure project is a joint effort of the SEI's TSP initiative and CERT® program. The principal goal of the project is to develop a TSP- based method that can predictably produce secure software. TSP-Secure addresses secure software development in multiple ways:

• Secure software is not built by accident – TSP-Secure addresses how to plan for security

• Schedule pressures and inter-personal issues can get in the way of implementing best practices – TSP-Secure helps to build self-directed development teams and then expects these teams to manage their own work

• Security and quality are closely related – TSP-Secure helps manage quality throughout the product development life cycle.

• People developing secure software must have an awareness of software security issues – TSP-Secure includes security awareness training for developers

Teams using TSP-Secure create their own plans. Initial planning is conducted during a series of meetings called a "project launch," which takes place over a three– to four–day period. The launch is led by a qualified team coach. During a TSP-Secure launch, the team reaches a common understanding of the security goals for the work and the approach they will take to do the work, produces a detailed plan to guide the work, and obtains management support for the plan. Typical tasks included in the plan are identifying security risks, eliciting and defining security requirements, secure design and code reviews, use of static analysis tools, unit tests, and fuzz testing. (Fuzz testing is thought to enhance software security and software safety because it often finds odd oversights and defects which human testers would fail to find, and even careful human test designers would fail to create tests for [7].)

Each team member of a TSP-Secure team selects at least one of nine standard team member roles (roles can be shared). One of the defined roles is a Security Manager role. The Security Manager leads the team in ensuring that product requirements, design, implementation, reviews, and testing address

security. He or she ensures that the product is statically and dynamically assured, provides timely analysis and warning about security problems, and tracks any security risks or issues to closure. The Security Manager works with external security experts when needed.

After the launch, the team executes its plan and ensures all security-related activities take place. Security status is presented and discussed during every management status briefing.

Visits to web sites such as the SANS Institutes Top 20 list of security vulnerabilities, the MITRE Common Vulnerabilities and Exposures (CVE) site, the US-CERT Technical Cyber Security Alerts site, and the Microsoft Security Advisory site show that common software defects are the leading cause of security vulnerabilities (buffer overflows have been the most common software defect leading to security vulnerabilities) [8]. Therefore, The TSP-Secure quality management strategy is to have multiple defect removal points in the software development life cycle. The more defect removal points there are, the more likely one is to find problems right after they are introduced, enabling problems to be more easily fixed and their root causes more easily determined and addressed.

Each defect removal activity can be thought of as a filter that removes some percentage of defects that can lead to vulnerabilities from the software product, as shown in Figure 1. The more defect removal filters there are in the software development life cycle, the fewer defects that can lead to vulnerabilities remain in the software product when it is released. More importantly, early measurement of defects enables the organization to take corrective action early in the software development life cycle.

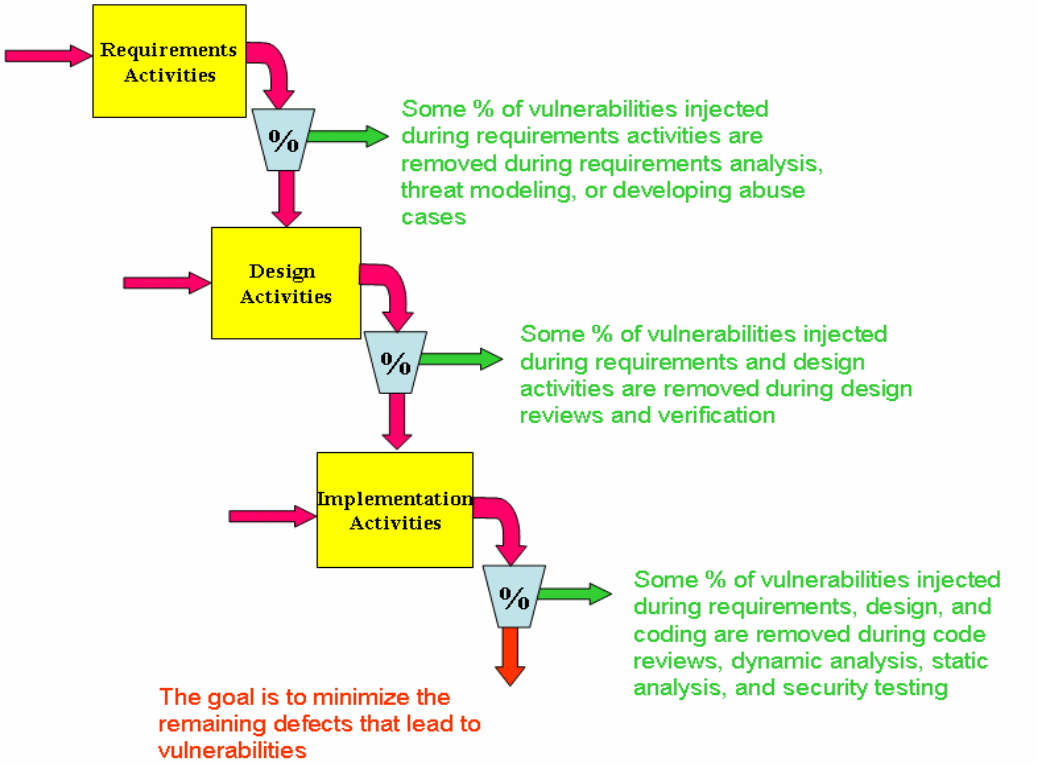

Figure-1: Vulnerability Removal Filters

Each time defects are removed, they are measured. Every defect removal point becomes a measurement point. Defect measurement leads to something even more important than defect removal and prevention: it tells teams where they stand with regard to their goals, helps them decide whether to move to the next step or to stop and take corrective action, and indicates where to correct their process in order to meet their goals.

The TSP-Secure team considers the following questions when managing defects:

- What type of defects lead to security vulnerabilities?
- Where in the software development life cycle should defects be measured?

- What work products should be examined for defects?
- What tools and methods should be used to measure the defects?
- How many defects can be removed at each step?
- How many estimated defects remain after each removal step?

In addition, the TSP-Secure method includes training for developers, managers, and other team members that specifically focuses on security awareness. [8]

C. Agile Methods

The individual Agile Methods include Extreme Programming (the most well-known), Scrum, Lean Software Development, Crystal Methodologies, Feature Driven Development, and Dynamic Systems Development Methodology. While there are many differences between these methodologies, they are based on some common principles, such as short development iterations, minimal design up front, emergent design and architecture, collective code ownership and ability for anyone to change any part of the code, direct communication and minimal or no documentation (the code is the documentation), and gradual building of test cases. Some of these practices are in direct conflict with secure SDLC processes. For example, a design based on secure design principles that addresses security risks identified during an upfront activity such as Threat Modelling, is an integral part of most secure SDLC.

Processes, but conflicts with the emergent requirements and emergent design principles of Agile Methods.

In their article Towards Agile Security Assurance, Beznosov and Kruchten address this issue and make some proposals as to how security assurance activities could be merged into Agile development methods [9]. They classified existing software assurance activities into four categories: those that are a natural match for Agile methods, those that are independent of any development methodology, those that can be automated or semi-automated so that they could be incorporated into Agile methods, and those that are fundamentally mismatched with Agile methods. Table 1 (included with permission from the authors) shows that almost 50% of traditional security assurance activities are not compatible with Agile Methods (12 out of 26 mismatches), less than 10% are natural fits (2 out of 26 matches), about 30% are independent of development method and slightly more than 10% (4 out of 26) could be semi-automated and thus integrated more easily into the Agile Methods. [9]

Table 1: Agile Methods – Compatibility with Security Assurance Practices

Security assurance method or technique		**Match (2)**	**Independent (8)**	**Semi-automated (4)**	**Mismatch (12)**
Requirements	Guidelines		**X**		
	Specification Analysis				**X**
	Review				**X**
Design	Application of specific architectural approaches		**X**		
	Use of secure design principles		**X**		
	Formal validation				**X**
	Informal validation				**X**
	Internal review	**X**			
	External review				**X**

Implementation	Informal requirements traceability				**X**
	Requirements testing			**X**	
	Informal validation				**X**
	Formal validation				**X**
	Security testing			**X**	
	Vulnerability and penetration testing			**X**	
	Test depth analysis				**X**
	Security static analysis			**X**	
	High-level programming languages and tools		**X**		

IV. SYSTEMS SECURITY ENGINEERING CAPABILITY MATURITY MODEL (SSE-CMM)

The SSE-CMM is a process model that can be used to improve and assess the security engineering capability of an organization. The SSE-CMM provides a comprehensive framework for evaluating security engineering practices against the generally accepted security engineering principles. By defining such a framework, the SSE-CMM, provides a way to measure and improve performance in the application of security engineering principles [11]. The SSE-CMM has been adopted as the ISO/IEC 21827 standard.

The stated purpose for developing the model is that, although the field of security engineering has several generally accepted principles, it lacks a comprehensive framework for evaluating security engineering practices against the principles. The SSE-CMM, by defining such a framework, provides a way to measure and improve performance in the application of security engineering principles. The SSE-CMM also describes the essential characteristics of an organization's security engineering processes.

The model is organized into two broad areas: Security Engineering, and Project and Organizational processes. Security Engineering in turn is organized into Engineering Processes, Assurance Processes, and Risk Processes. There are 22 Process Areas distributed among the three categories. Each Process Area is composed of a related set of process goals and activities. The International Systems Security Engineering Association (ISSEA) maintains the SSE-CMM. [11]

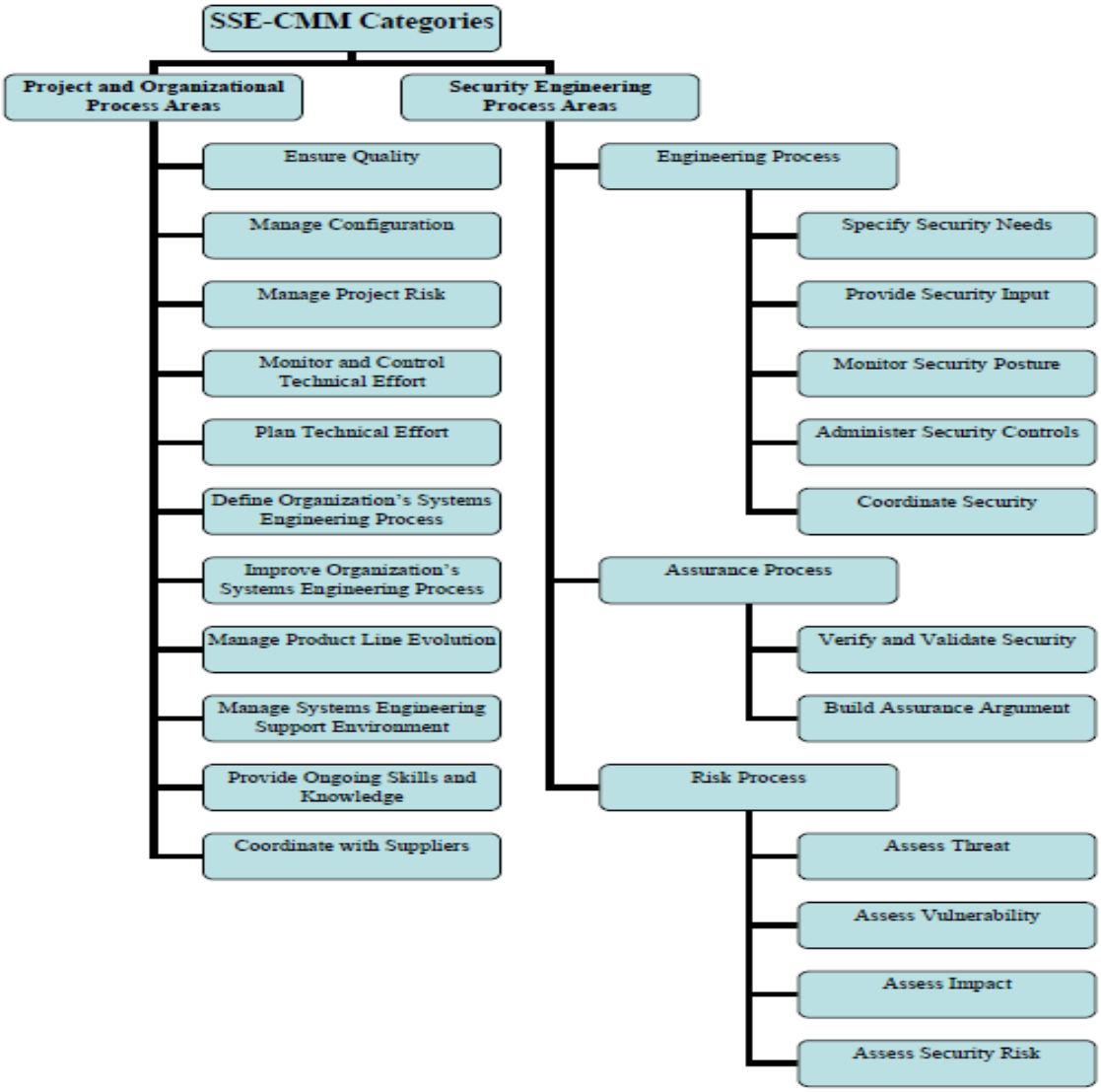

Figure-2: *Process Areas of the SSE-CMM*

V. EXTENDED SSE-CMM BY USING MS-TCSDL, TSP-SECURE, AND AGILE METHOD FOR EMBEDDED SYSTEMS CONTROL

a. What Is Needed?

Because of the integration of process disciplines and coverage of enterprise issues, the SSE-CMM is used by more organizations; yet this model contains gaps in their coverage of safety and security.. The proposed Safety and Security additions to the SSE-CMM identify standards-based practices expected to be used as criteria in guiding process improvement and in appraising an organization's capabilities for providing safe and secure products and services.

b. PROPOSED SAFETY AND SECURITY ADDITIONS TO THE SSE-CMM:

1. Goal 1 – An infrastructure for safety and security is established and maintained.
   a. Ensure safety and security awareness, guidance, and competency.
   b. Establish and maintain a qualified work environment that meets safety and security needs.

c. Ensure integrity of information by providing for its storage and protection, and controlling access and distribution of information.

d. Monitor, report, and analyse safety and security incidents and identify potential corrective actions.

e. Plan and provide for continuity of activities with contingencies for threats and hazards to operations and the infrastructure.

2. Goal 2 – Safety and security risks are identified and managed.

a. Identify risks and sources of risks attributable to vulnerabilities, security threats, and safety hazards.

b. For each risk associated with safety or security, determine the causal factors, estimate the consequence and likelihood of an occurrence, and determine relative priority.

c. For each risk associated with safety or security, determine, implement and monitor the risk mitigation plan to achieve an acceptable level of risk.

3. Goal 3 – Safety and security requirements are satisfied.

a. Identify and document applicable regulatory requirements, laws, standards, policies, and acceptable levels of safety and security.

b. Establish and maintain safety and security requirements, including integrity levels, and design the product or service to meet them.

c. Objectively verify and validate work products and delivered products and services to assure safety and security requirements have been achieved and fulfil intended use.

d. Establish and maintain safety and security assurance arguments and supporting evidence throughout the life cycle.

Goal 4 – Activities and products are managed to achieve safety and security requirements and objectives.

a. Establish and maintain independent reporting of safety and security status and issues.

b. Establish and maintain a plan to achieve safety and security requirements and objectives.

c. Select and manage products and suppliers using safety and security criteria.

d. Measure, monitor and review safety and security activities against plans, control products, take corrective action, and improve processes.

VI. COMPONENT STRUCTURE DIAGRAM OF EXTENDED SSE-CMM

In this section we construct an Extended SSE-CMM structure of component from a structural point of view. The structure of the components of Project and Organizational Process Areas and Security Engineering Process Areas are the basic elements.

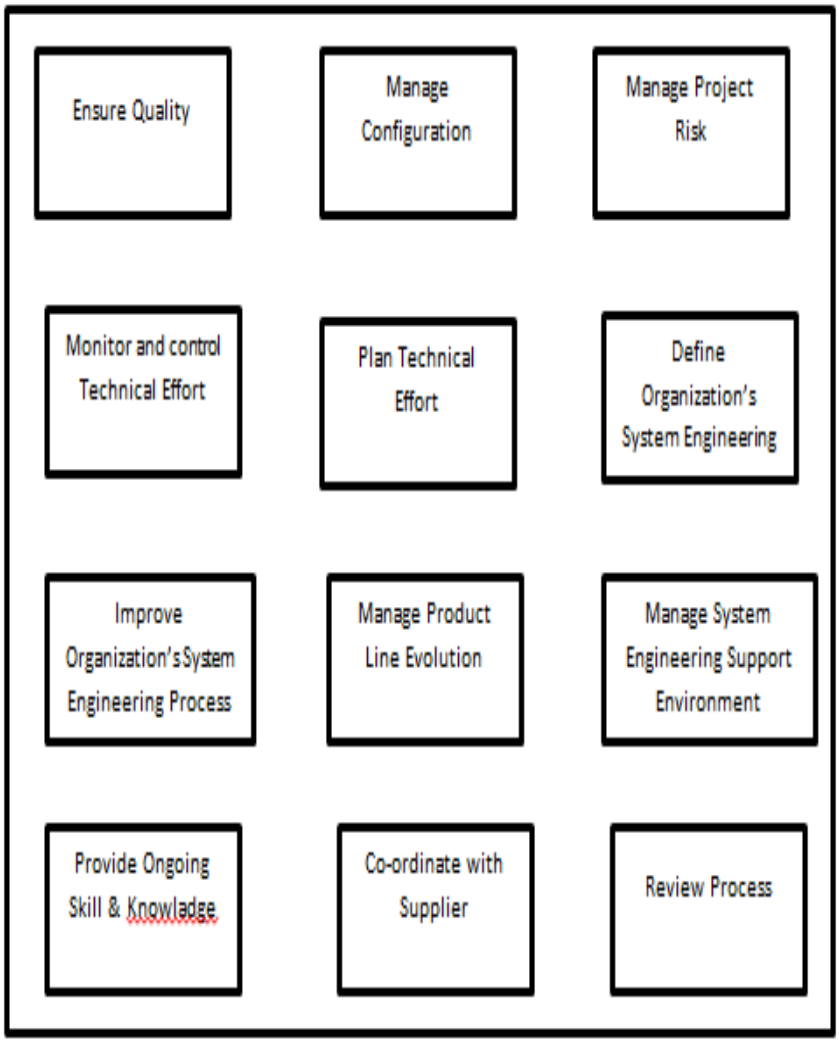

Figure-3: Project and Organisational Process Areas

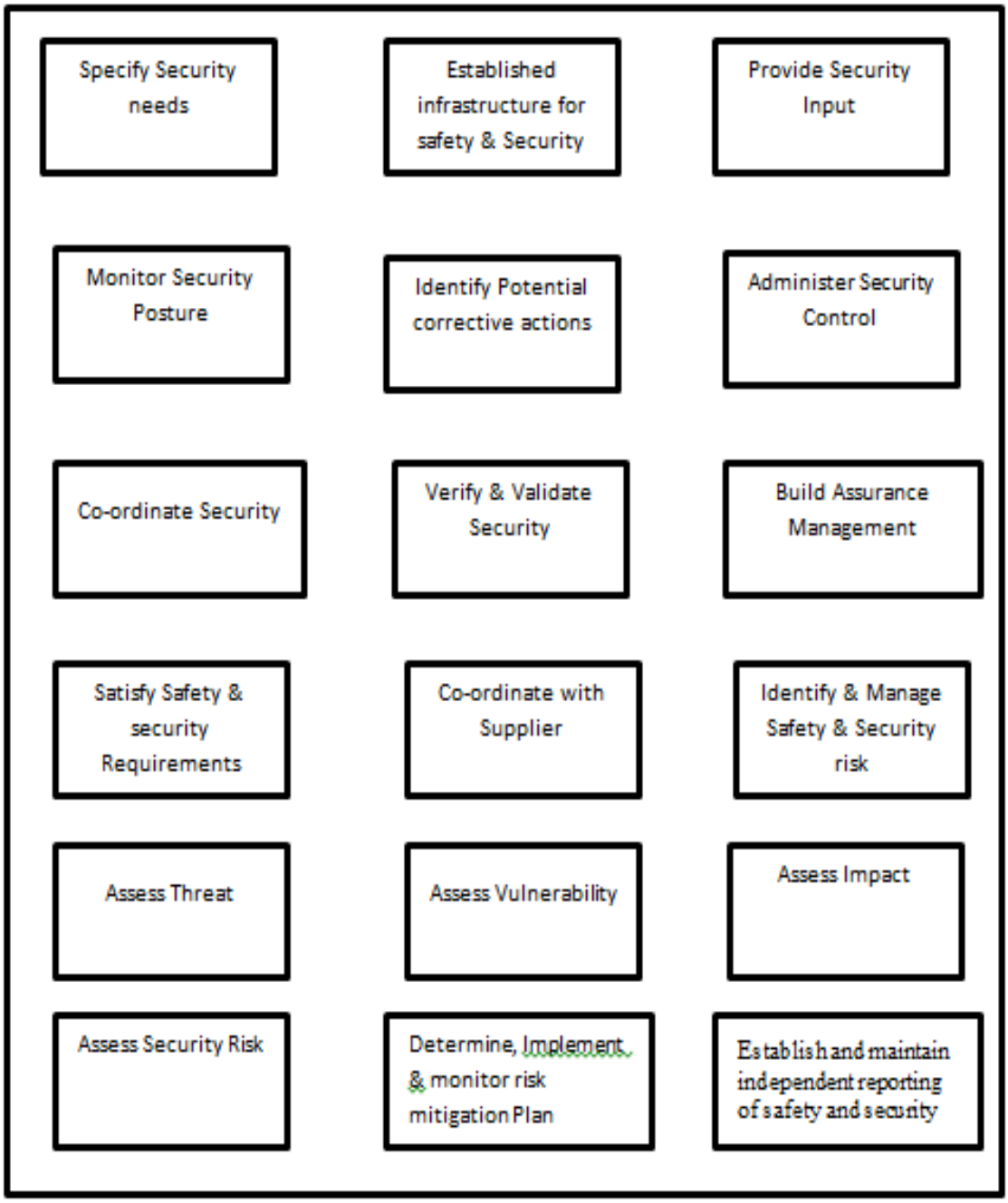

Figure- 4: Security Engineering Process Areas

VII. CONCLUSION

This research work explores different software process namely MS-TCSDL, TSP-Secure, And Agile Method and SSE-CMM which further extended together for secure software development for Embedded System Control. Although there are several processes and methodologies that could support secure software development, very few are designed specifically to address software security from the ground up. The notable exceptions are Microsoft's Trustworthy Computing SDL and the SSE-CMM. As software security becomes a more important issue in an increasingly networked world, more processes that explicitly address the four focus areas identified in this paper security engineering activities, security assurance activities, and security organizational and project management activities, and security risk identification and management activities) should achieve visibility.

In this paper we have discussed the most important issues concerning security in embedded systems. Dealing with both security and safety constraints when designing software process model is a challenging task that we propose to address with a new process model called Extended SSE-CMM. The Extended SSE-CMM process is based on the strong belief that each step should serve a clear purpose and be carried out using the most rigorous techniques available to address that particular problem. Specifically, the process almost always uses formal methods to specify behavioural, security, and safety properties of the software.

REFERENCES

[1] IEEE. IEEE Standard Glossary of Software Engineering Terminology. ANSI/IEEE Std 610.12-1990. February 1991.
[2] Somerville. -Software Engineering. 9th Edition 2011.
[3] Steve Lipner Michael Howard Security Engineering and Communications Security Business and Technology Unit Microsoft Corporation. March 2005.
[4] Lipner, Steve & Howard, Michael. The Trustworthy Computing Security Development Lifecycle. http://msdn.microsoft.com/security/default.aspx?pull=/library /en-us/dnsecure/html/sdl.asp (2005).
[5] Humphrey, Watts S. Winning with Software: An Executive Strategy. Boston, MA: Addison Wesley, 2002 (ISBN 0201776391).
[6] Davis, Noopur & Mullaney, Julia. The Team Software Process (TSP) in Practice: A Summary of Recent Results (CMU/SEI-2003-TR-014, ADA418430). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. http://www.sei.cmu.edu/publications/documents/03.reports /03tr014.html .
[7] The SANS Institute. The Twenty Most Critical Internet Security Vulnerabilities (Updated) – The Experts Consensus. http://www.sans.org/top20/ (2005).
[8] Team software process for secure system development by James W Over, Software engineering institute. www.sei.cmu.edu.March 2002.
[9] The Agile Alliance. Manifesto for Agile Software Development. http://agilemanifesto.org (2001).
[10] A Survey of Agile Development Methodologies by Laurie Williams 2007.
[11] SSE-CMM) developed by the International Systems Security Engineering Association (ISSEA). www.sse-cmm.org/

AUTHORS BIOGRAPHY

Sanjai Gupta has received his graduation degree as B.Sc. in Mathematics (Hon's) from Lucknow University, and post-graduation degree as Master of Computer Application (MCA) from VBS University, Varanasi with distinction and pursuing his PhD. in computer Science from CMJ University, Meghalaya, India. Currently he is working as Lecturer in IT department at Nizwa College of Technology, Oman on contractual basis. His area of research is Software Engineering-Software Development Process.

Md. Faisal has received his graduation degree as B.Sc. in Computer Application (Hon's) from Ranchi University, and post-graduation degree as Master of Computer Application (MCA) from Vishveswaraya Technological University, Belgaum with distinction and he has published many research papers in different international journals in his research area. Currently he works as Lecturer in IT department at Nizwa College of Technology, Oman on contractual basis. His area of research is Software Engineering- Requirement Engineering, Security system and Process Models.

Mohammed Husain has received his PhD in Computer Science from Integral University, Lucknow and currently working as Director of M G Institute of Management & Technology, Lucknow, UP, India.