

Secure Software Diversity

By Jason Grembi

Secure Software Diversity

- What is Secure Software Diversity
- Understand Principles and Concepts of Secure Software Diversity
- How Software Diversity is applied in the organization

Secure Software Diversity

- What is Software Diversity?
 - Software Engineering Body of Knowledge (SWEBOK) defines Software Diversity as “the ability to anticipate change in software” .
 - The principle to anticipate change is applied in the approach, design, and effort in software development that will prolong the shelf life and reusability of any application.

Secure Software Diversity

- Why Software Diversity?
 - All things change during software development:
 - Requirements
 - Options
 - names
 - values

Application must be designed to seamlessly anticipate these changes so that little to no additional coding efforts are required

Secure Software Diversity

- How does Software Diversity impact Security?
 - Every time a software component or module is touched, a security vulnerability could be introduced.
 - Security vulnerabilities come from:
 - Human forgetfulness/error
 - Insufficient requirements
 - Complexity confuses understanding
 - “I did not think of that” realization

Secure Software Diversity

- Software Diversity an Polymorphic approach.
 - Creating a polymorphic software application allows for a more secure way of developing, deploying, and maintaining software applications.
 - One change in property manager table
 - Building sensing activities in code to change according to rules and policies.
 - Building applications that can rest and redeploy themselves
 - Highly reusable and scalable applications
 - Creates software that stands the test of time

Secure Software Diversity

- Software Diversity is a security principle that incorporates two goals:
 - Produce secure code
 - Polymorphic applications rely on rules and settings (parameters) to change behavior. Securing these rules and settings with network topology (i.e. firewalls) and database security (i.e. views).
 - Produce quality code
 - In order to be a diverse, the application must have all quality aspects inherently coded so that the write once and use anywhere concept can be applied.

Secure Software Diversity

- Goals of Secure Code
 - Secure software means keeping all input parameters and code base C.I.A.
 - Confidentiality: Attained by keeping unauthorized users (humans or software) from accessing confidential information
 - Integrity: The way in which the application accepts, transmits, and stores data from the point of entry all the way to the database and back
 - Availability: System is available during its normally scheduled hours of operation
 - Swa-CBK defines the principles of security

Secure Software Diversity

- Goals of Quality Code
 - Software construction as an engineering process that creates working, meaningful software through a combination of coding, self-validation, and self-testing
 - SWEBOK, Software Engineering Body of Knowledge (BOK), is the industry standard for quality code. This BOK defines two techniques used for software development: manual and automated
 - Manual techniques are the human ability to logically break down complex problems into smaller ones
 - Total automation is a tool-intensive process

Secure Software Diversity

- How are those techniques applied in Software?
 - Manage complexity through Clean Code
 - Total Tool Automation

Secure Software Diversity

- Three ways to manage complexity in software is through clean code:
 - *Removal of complexity:*
 - *Automation of complexity:*
 - *Localization of complexity:*

Secure Software Diversity

- *Removal of complexity:*
 - Some requirements are just add-ons that are really not needed
 - Complexity of processing data can be ‘outsourced’ by using frameworks, third-party processing, and/or web services.
 - Complex algorithms and business logic are duplicated in many applications.
 - Complex algorithms can be simplified
 - The principle of Software Diversity is to eliminate redundancy and secure the perimeter.

Secure Software Diversity

- *Automation of complexity:*
 - There are many complex transactions that take place within a software application. Every transaction needs to be automated and that automated process needs to be secure.
 - Complex Transactions include
 - Compiling code
 - Encrypting code
 - Transferring files from server to server
 - Archiving code and logs

Secure Software Diversity

- *Localization of complexity:*
 - If the complexity can not be removed or automated, then the code function must be localized and contained.
 - Place application logic on one spot (refactoring)
 - Do not use static code or hard coded values with the function
 - Do use static variables as input parameters
 - Build small modules

Secure Software Diversity

- Total Tool Automation
 - SWEBOK defined two techniques used for software development: manual and automated
 - Manual techniques are the human ability to logically break down complex problems into smaller ones
 - Total automation is a tool-intensive process

Secure Software Diversity

- Total Tool Automation
 - Every task that a developer takes during application development must be replaced by a tool.
 - Tools can and will be integrated with other tools so that all movements of code and it's surrounding dependencies are tool driven.

Secure Software Diversity

- An example of Total Tool Integration: Auto Deployment:
 - ANT (another neat tool) can integrate with code that exists in a repository tool to pull a specific code version.
 - ANT will then pull the code down into server A; compile code on server A, and deploy the code base on production server B.
 - Ant will then reboot production server B after the installation
 - ANT will them erase all code from server A

Secure Software Diversity

- So far:
 - Software Diversity is a principle that lends to building application that stand the test of time because they were already built to change.
 - Software Diversity relies on a development process that promotes a ‘clean code’ approach. That means, code that is simple, reusable, module and dynamic.
 - Software Diversity is achieved by the use of automation and tools throughout the development process. Automation simplifies complex tasks, human forgetfulness, and consistency.
 - Software Diversity relies on the use of tools

Secure Software Diversity

- How does such a principle get enforced on a development team?
 - Why would programmers want to use tools for everything?
 - Why would programmers want to create ‘clean code’?
 - How does project management encourage and support Software Diversity?

- Answer: Application Guide

The Application Guide

- Application Guide is a written contract by programmers for other programmers that hold accountability.
- The following is what you need to know about starting this guide:
 - Defining the Application Guide
 - Creating an Application Guide
 - Risks of not using an Application Guide
 - Risks of using an Application Guide
 - Why you should not hoard secrets

The Application Guide

- What does the Application Guide look like?
 - It is a tool that provides the blueprint of how a programmer codes software on a specific application
 - It provides working “how-to” instructions for programmers
 - Documents project details such as standards, rules, and coding conventions
 - Tool that holds the team together and ensures that all code is created equal

The Application Guide

- Things listed in an Application Guide
 - step-by-step instructions on how to create the development environment and process
 - Where to acquire development tools
 - Where are the tools located (on hard drive or other)
 - How to configure tools
 - Where to find license information
 - Where to find deployment instructions
 - Where to push code into production (server names, IP addresses)

The Application Guide

- When to Create an Application Guide
 - Creation of an Application Guide starts on day one and never finishes until the software is fully matured and in maintenance mode
 - Creating the Application Guide requires detailed specifics of the project
 - Some of these specifics might not be defined at the time that the document is initially created
 - Normally takes several iterations of “trial and error” before an Application Guide starts to work for a team

The Application Guide

- Creating an Application Guide (continued)
 - Who Creates the Application Guide (continued)
 - It is the lead developer's responsibility to initiate and maintain this guide
 - Written agreement created by the developers for the developers
 - Team decides which tools to use, which conventions to uphold, and any frameworks that need to be followed
 - Guide is created in a democratic fashion

The Application Guide

- A successful implementation of the Application Guide will have:
 - Benefits for the organization
 - Benefits in the cube
 - Produces a consistent development process. A higher degree of confidence in application Security is achieved once the process is consistent and repeatable.

The Application Guide

- Benefits to the Organization
 - Other developers, managers, and departments will be interested in what made your project so successful
 - As each application uses an Application Guide and a standard approach to software development, all code will start to look and act the same
 - The Application Guide is a powerful mechanism that can be used to streamline a lot of redundant processing

The Application Guide

- Benefits to the Organization
 - One company can have multiple departments, with each department having multiple applications

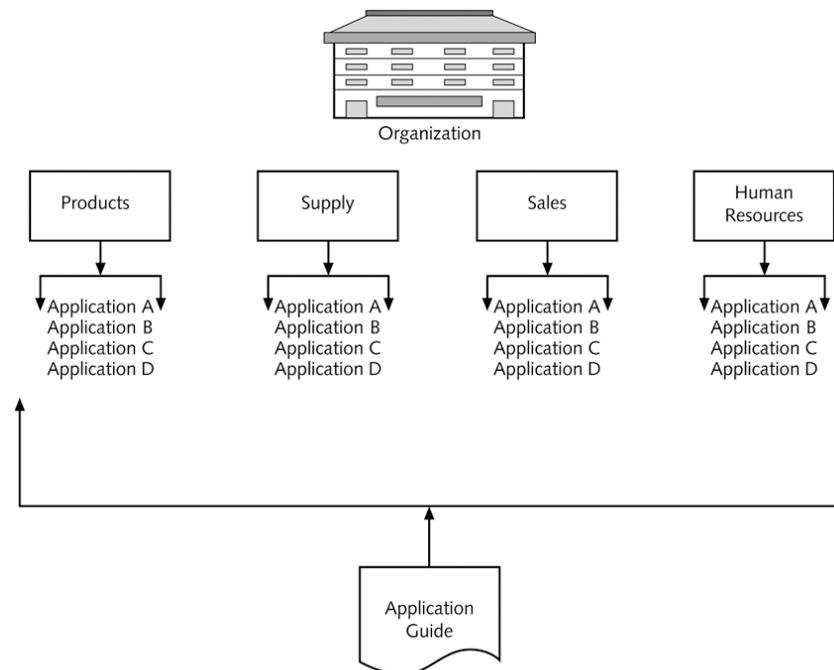


Figure 4-1 Application Guide inside an organization

The Application Guide

- Benefits to the Organization (continued)
 - Code Reuse
 - If all the code on a given application was created with the same characteristics and design across all modules, there would be a lot of reusable components
 - Cross-Training
 - A developer will be able to move from one project to another with little to no learning curve
 - Awareness
 - Knowing how the application's code is written, what the code can do, and what the code cannot do is half the battle in analysis
 - Security Awareness
 - Secure code starts with quality, and the Application Guide definitely strives for quality

The Application Guide

- Benefits in the Cube
 - Using an Application Guide will bring many benefits to you and the development team
 - All developers working from the same page
 - Provide them with one common vision of how the code will be developed (frameworks)
 - Where the code is going to be programmed, archived, and deployed
 - How and when security will be applied to the software artifacts

The Application Guide

- Benefits in the Cube (continued)
 - The Application Guide in the cube will do the following:
 - Improve the code review process
 - Create a common language and understanding

The Application Guide

- Benefits in the Cube (continued)
 - Improving the Code Review Process
 - The Application Guide lays out the *where* and the *how* of the developers
 - It leaves no room for guessing or assuming
 - Eliminating guesswork saves the developer time and energy from analyzing rudimentary items
 - Eliminating guesswork also improves the peer review process because every developer knows exactly what reviewers are looking for

The Application Guide

- Benefits in the Cube (continued)
 - We All Speak One Language
 - Geek speak can be cute and funny, but to people who are not in the know, it can be annoying
 - Everyday verbiage will begin resembling that of the other developers on your team
 - Other developers will soon be speaking (and meaning) the same jargon and references

The Application Guide

- Coding conventions define how the code is written and how it looks and the Application Guide states these conventions
- If the group helps decide, it is more likely that the group will cooperate
- Using an Application Guide creates a repeatable development process. When a process is consistently repeated, security is easier to apply and control.

The Application Guide Summary

- The Application Guide is a tool that provides the blueprint of how the software application is written
- The guide serves as a recipe for how to build the developer's environment on the fly with little or no help so that all developers can build software using the same environment, tools, and standards

The Application Guide Summary

- The Application Guide is a living document that should be created by the developers for the developers
- The Application Guide will go through different iterations of changes and refinements as the project evolves; this is a natural progression of a living document that improves over time
- The Application Guide needs to be tested from time to time to ensure that all the supporting documentation is still accurate