Lesson Proper for Week 5

CONFIGURATION MANAGEMENT

When we look at the Agile process, CM methods are not referenced for any specific routines. These methods are a supporting discipline and not directly involved in creating executable code. If Agile processes have a lack of configuration control, then Lean principles are a bust or a big waste of time and lead to a chaotic activity. You see no progress in software design/development.

The members who make up the Agile team are focused on processes and tools that would imply configuration control is not important, but CM disciplines aim to trim the process and provide more automation in the tools, bringing back focus to configuration control objectives. The software tools common to other team members are adapted to the processes.

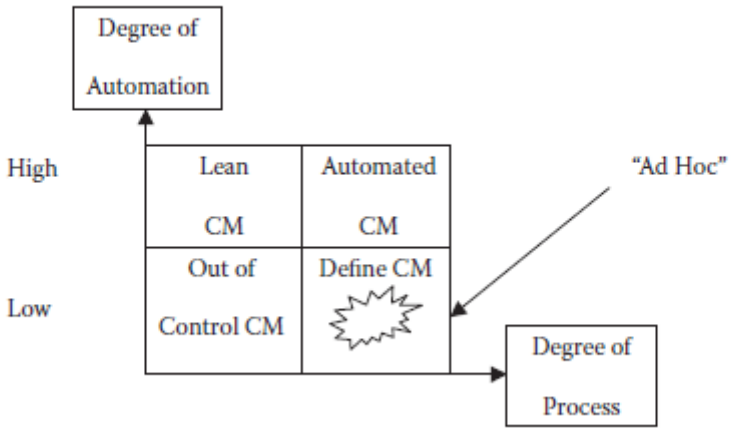


FIGURE 1: Lean CM performance.

CM is a support discipline to the Agile team, but there are times in software design/development whether work products are comprehensive to plan and process documentation. Controlling change is the foundation to ensure software design/development activities are important when change is in the picture. Make sure CM methods do not limit change and become a stumbling block and a nuisance for program and project plans.

By adapting Agile practices in the CM process, teams can have a leaner concept in programs and projects by:

- Ensuring change is common with configuration control objectives
- Having a clear distinction for changes to requirements
- Embracing that change is possible with appropriate routines for configuration activities
- Providing the development and enhancement of Agile ideas

My experience has been that common changes from Lean CM create chaos in working toward heavy and out-of-control processes. An example is defined in Figure 1.

SOFTWARE STANDARDS

It is and will always be a requirement that software design engineers follow required software standards to ensure that development processes are in accordance with specific process models (i.e., CMMI). The software design engineers are required to show defined, managed, and consistent improvements during life-cycle development. At minimum, software standards consist of:

- Documented and maintained plans and procedures
- Peer reviews to eliminate defects and prevent future occurrences
- Standard software tools for design, code and unit test, and configuration control

CAPABILITY MATURITY MODEL INTEGRATION

CMMI provides the best opportunity to address software design or development with ongoing support to customers after delivery. The design/ development process could be a complex activity that benefits the accomplishment of correct software tasks during the life cycle. Software engineering and processes require an association with each other when it comes to software engineering. CMMI does provide a systematic, disciplined approach to all software engineering tasks in the affected program and projects. The development of software work products using CMMI will enhance the knowledge base for software designers.

To implement CMMI processes provide the content for performance during the software life cycle for change, development, installation, integration, and maintenance. Complete software design practices provide the basic concepts (i.e., form, fit, function, interface, integration process, etc.) and other sound concepts. Some software

engineering tasks defined by CMMI are:

- The identification of internal and external interfaces
- Software design to establish infrastructure abilities during software design/development
- Development of plans, processes, and procedures
- Reuse of capabilities for software identified for use

With CMMI implemented and used by the program and projects, this model represents processes in two different scenarios: continuous and staged. Every organization should work toward the achievement of a third level: defined. The process standards apply to:

- Requirements development
- Technical solution (see following discussion)
- Product integration
- Verification
- Validation
- Organizational process focus
- Organizational process definition
- Organizational training
- Integrated project management
- Integrated supplier management
- Risk management
- Decision analysis and resolution
- Organizational environment for integration
- Integrated teaming

Software engineering methods have the capability to utilize many tasks and activities along with described approaches. The method is effective when how much support is needed. The engineering process area of technical solution can be the template for software engineering to design, develop, and implement solutions to requirements supporting software and systems integration. The processes that are related to this process area concept receive requirements for managed processes.

Technical solution processes support each other and service products related to the software life cycle. A technical data package provides the software design/development team complete understanding for effective design and development for the next phase of integration activities.

CMMI Version 1.3

In 2009, CMMI version 1.3 was initiated. The release was to provide a new approach to software companies and military and aerospace programs and projects to improve performance in appraisals and training. The focus is on:

- High maturity
- More effective processes
- Conducting and performing effective appraisals
- Commonality in all product suites
-

Major elements implemented allow appraisal teams to become more effective in reflecting organizational high maturity using a staged approach.

The high-maturity practices were not understood and were unclear, leading to mixed views by organizations for how objectives are related and lead to high levels. Modernized practices include improvements in:

- Agile environments
- Functional requirements during software design/development
- Subcontractor agreements pertaining to COTS (commercial off-the- shelf) and NDI (non-development item) software
- Organizational training

CMMI version 1.3 coverage will add updated information currently supported by Lean and Six Sigma and customer satisfaction for software design/development life-cycle tasks.

Lean Six Sigma

Lean and Six Sigma tools and philosophies have helped thousands of software companies and military and aerospace programs and projects dramatically improve processes, customer satisfaction, on-time delivery, and other measurable results. But, do these same tool sets apply to the processes of software design/development? The answer is “yes” when the correct tools are applied in the right way and to the right process.

The Six Sigma methodology attempts to reduce process variation, resulting in fewer errors and defects. A software defect is defined by customer requirements, whether formally documented or an expectation that is not met. A defect may be detected during the software design/development phase by team members or later when the customer is using the delivered software. The Six Sigma process does show fewer defects per opportunities or zero defects in a software work product.

Opportunities for defects abound, including but not limited to macro functional requirements allowing the end user to enter wrong data.

To accomplish the goal of zero defects, team members must have highly structured and robust processes for each step in a software life cycle. In Six

Sigma, the steps are:

- Define
- Measure
- Analyze
- Improve

The control software teams often use a form of these requirements: gathering, design, implementation, verification, and maintenance. The formal processes program and projects' scope are customer requirements to have effective methods for software and systems integration. Data flow analysis and feature breakdown structures ensure fewer opportunities for errors. During the software design/development process, the Six Sigma philosophy is applied for building quality through mistake-proofing methods.

The creation of effective charts of when and where defects were detected and code had to be rewritten, added, or reused can assist the team in evaluating which steps in the process have the most variation and are candidates for Six Sigma process improvement. The Lean production of software and systems integration work products focuses on the elimination of waste from defined processes. The eight wastes are easily remembered with the acronym DOWNTIME:

- Defects
- Overproduction
- Waiting
- Non-utilized talent
- Transportation
- Inventory

- Motion
- Excess processing

Software design/development is a complex process integrated with wastes that include defects as discussed and resulting in rework or reuse, which is another waste for excess processing. Waiting on waste occurs when programmers, project leaders, and team members require information such as customer requirements and parameters from code and unit tests that could delay their development of software work products. Excess processing of waste also occurs with numerous review cycles rather than having a robust process for designing quality and being right the first time. An example of overproduction and the inventory wastes may be the creation of features that were not requested or are not needed.

The Lean philosophy of problem solving uses simple and straightforward tools to achieve fast yet powerful results. Utilizing the software design/development process creates a process flow or a value stream map that shows each step. Look for each of the eight wastes as you walk through the process. Identify the “hidden factory” or tasks that are regularly performed but are not documented or do not add value. If an activity does not change the functionality of code or the software programming activity, it is waste. If the activity does require waste, such as some phase quality gate code reviews, do minimize the wasteful activity by creating a sound process for performing the task and minimizing rework or reuse.

When you have identified waste in your process flow, drill down to the root cause using a simple technique. Develop a future state process that eliminates or minimizes waste. Finally, implement process changes, putting in place standardized work products and program and project leaders to follow up and ensure improved processes function the intended way. Make progress so the achievement of program and project milestones is visible to ensure team members can see progress and are accountable for meeting their goals. Do not forget to track waste by moving forward so you can continuously improve processes. A schedule showing the number of times a section of code has to be rewritten or reused is an example of how easy it is to identify, track, and eliminate waste.

Combining the data-driven approach of Six Sigma and the waste-eliminating tools of Lean streamlines the design/development process and produces better software in less time. The goal is to satisfy your customers with amazing work products and services. Creating a defect- and waste-free process during the software life cycle does make excellent programs and projects and the customer happy.

SOFTWARE COMPANIES

Many software companies and military and aerospace programs and projects previously and currently build the concept of specializing in software design/development to boost competitiveness in the software industry. Many global companies have always gained ground to integrate the market with strong software and hardware. Countries have been in competition in software development for years, nurturing technicians with software expertise at an early stage.

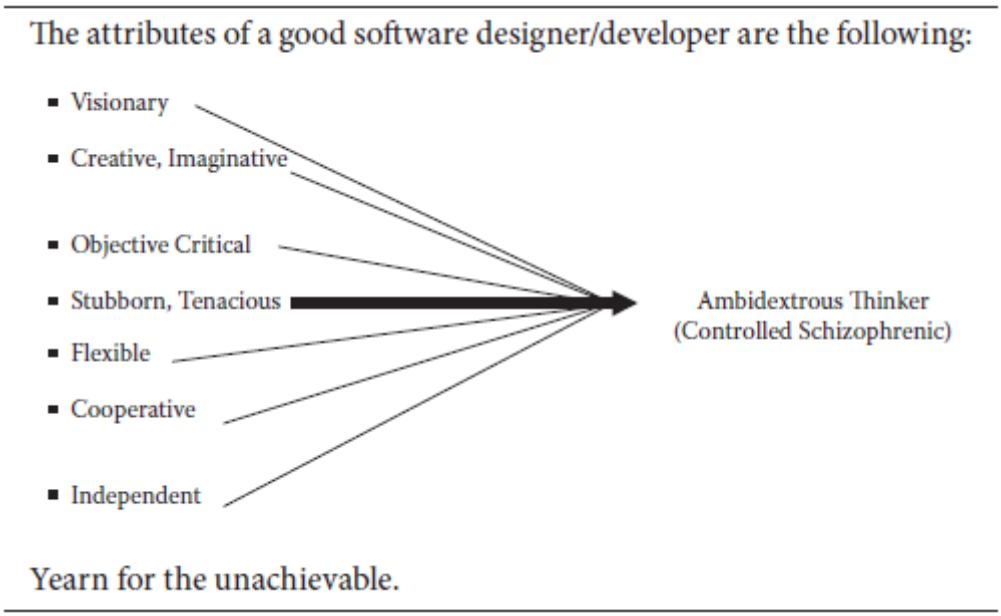
Software Design/Development

The software design/development opportunities secure databases related to software technology owned by companies, governmental research facilities, and even universities to provide technology concepts to customers. The software technology consultant provides customized consulting and technology to companies, government, and individual and small firms.

The lower limits of software integration restrict the consultant from participating. The software companies have increased the dollars for winning business and sales. Effective methods for software and systems integration efforts inject profit from inside the software design/development sector. Profits will increase once actions for consulting come into play for software companies all over the world.

CONCLUSION

The pairs of software design/development attributes shown cannot be exhibited simultaneously without circulating the brain. One can (and must) learn to switch, change, and be flexible from one mode to another as needs arise. This can be done, and one can learn how to do it.



◄ Preliminary Activity for Week 5

Jump to...

Analysis, Application, and Exploration for Week 5 ►

Navigation

Home

Dashboard

Fair Warning

Activities

- Assignments
- Forums
- Quizzes
- Resources

Bestlink College of the Philippines
College Department

Powered by eLearning Commons