Journal of Information Technology Management

ISSN #1042-1319

A Publication of the Association of Management

A CASE STUDY ON THE CHALLENGES AND TASKS OF MOVING TO A HIGHER CMMI LEVEL

DAN TURK
COLORADO STATE UNIVERSITY
Dan.Turk@Colostate.Edu

CHARLES BUTLER
COLORADO STATE UNIVERSITY
Charles.Butler@Colostate.Edu

KRISTI MULDOON
BRIGHTON SCHOOL DISTRICT 27J
kmuldoon@sd27j.org

ABSTRACT

A study was performed of a Federal organization (referred to as Federal Systems in this paper) that provides government agencies with business application services. Recently, Federal Systems carried out an initiative to move its software development capability from CMMI Level 1 to CMMI Level 3. The analysis of this study assesses how extensively Federal Systems projects adopted process areas, identifies some of the challenges they faced, and makes recommendations for organizations that are attempting similar initiatives.

Keywords: software engineering, process models, software management, software development methodology, process maturity, Capability Maturity Model Integration (CMMI)

INTRODUCTION

Software development methodologies, including project governance, receive prominent attention in today's information technology (IT) research and practice. At a high level, a software development methodology is a conceptual framework that assists organizations with the understanding of their own processes and the way they work. On an applied level, it provides a road map of the software development effort and is a planning tool for business and technology implementation. A software development methodology is a framework that is used to structure, plan, and control the process of developing an information system when aligning it with business goals. Software development projects establish the decision-making framework within which design decisions are made. Companies look to software development methodologies as means of formulating tactics for implementing activities and constructing deliverables that integrate into an organization's infrastructure, applications, and data. Governance provides the framework through which decisions are formulated, made, communicated, and supported regarding how information technology resources are utilized.

In this paper a case study is presented of an organization's process improvement efforts in its attempt to move up the Capability Maturity Model Integration (CMMI) staged capability maturity hierarchy [3; 7] from

Level 1 to Level 3. First, the study's objectives are summarized. Then, the organization is briefly described. Next, the case study methodology used in this study is explained, and CMMI's two views on process capability are described. Analysis of the organization's Systems Development Lifecycle (SDLC) is then performed, including an assessment of how completely the organization's methodology covers CMMI process areas (PAs). The analysis concludes with an assessment of how extensively the developers comply with the organization's intended practices. The paper finishes with findings and conclusions.

STUDY OBJECTIVES

The purpose of this research was to analyze Federal Systems' initiative to implement enterprise project management and software development processes, and the activities and deliverables required for CMMI appraisal. The initiative used CMMI Level 3 as a benchmark for success. From this analysis, the outcomes were assessed to determine the level of success attained.

The objective of this study included four elements. The first element was to describe the architectural framework of the Federal Systems SDLC model. (This is done in the first two parts of the Analysis section.) The second element was to assess the state of enterprise project management and software development processes. This element involved ranking the success of these processes within a CMMI Level 3 framework. (This is done in the first part of the Federal Systems Mapping section.) The third element examined Federal Systems' software development project practices to determine the level of successful methodology implementation. (This is done in the second and third parts of the Federal Systems Mapping section.) Finally, recommendations for improving implementation of project management and software development processes in line with the CMMI appraisal criteria were formulated as a result of this study. (This is done in the Findings section.)

ORGANIZATIONAL BACKGROUND

Federal Systems, a US federal government agency, provides other federal agencies with business application services. These services are mission critical for business information technology performance. Federal Systems provides a full spectrum of professional services for developing, maintaining, and supporting enterprise-class business applications. Its goal is to be the provider of choice for enterprise web services, custom applications (both mobile and desktop), access management services, and leading technology eLearning solutions.

Federal Systems provides a full range of professional services for developing, maintaining, and supporting enterprise-class business application portfolios including the following:

- Application and database hosting and maintenance services.
- Program management and software engineering services.
- Information technology solutions.
- Access management, providing secure, logical access to information systems.
- Information on future developments within information technology and its application to business processes.
- Software as a Service delivering citizen-centric, online information and services.

Federal Systems deploys a systems development lifecycle for its software development projects. The SDLC is an important component of its enterprise framework for software development methodology. Underlying this methodology is a strategy to evolve to a more mature IT organization as measured by CMMI. The systems development lifecycle is composed of processes associated with a CMMI maturity level, specifically Defined (Level 3). With the SDLC, these processes are mostly contained in the project management and software development areas. Federal Systems utilizes Process Action Teams (PATs) to implement the processes needed for each process area. After utilizing their software development lifecycle for funded projects, Federal Systems sought to assess progress, measure compliance, and determine maturity growth toward CMMI Level 3 maturity level rating.

CASE STUDY METHODOLOGY

For the study described in this paper, the Federal Systems SDLC was examined to identify successes and failures which resulted in mature or immature software development project execution. This examination included identifying common factors relative to successful SDLC usage and barriers which resulted in deviation from the methodology. In order to complete this examination, this study methodology included the following five steps:

1. Describe an architectural framework for four major elements of the Federal Systems SDLC model.

2. Complete a high level critique of the Federal Systems SDLC. A critique was conducted of the deliverables and events elements of the lifecycle. This critique identified inconsistencies and architectural weaknesses that constrain best practices.
3. Construct a mapping between Federal Systems' SDLC and CMMI. SDLC phase events were used as process proxies. The mapping generated satisfied and omitted CMMI process requirements.
4. Create a descriptive statistical summary of Federal Systems SDLC practices. In order to accomplish this step, a baseline benchmark of 16 projects was completed measuring how frequently the processes were implemented. The baseline was accomplished by gathering information from project leads via surveys, personal interviews, and an examination of the document repository.
5. Analyze compliance of Federal Systems project practices with the enterprise methodology as measured by required and optional elements of the methodology.

The results of these steps are shown in the following sections.

Interview Process

Thirteen[1] project leads from Federal Systems were interviewed for this study. Project leads included individuals with many years in their respective positions as well as individuals with only a few months in their current positions. Project leads were interviewed individually. The interview process consisted of identifying projects which met the scope requirements described below in the Examined Projects section. These projects were then examined individually with the project lead [8]. Interview questions followed survey questionnaire format that was the basis for quantifying the events and deliverables for projects. During each interview, the interviewer verified the existence of each available artifact by means of having the project lead identify and open electronic files. In this study, an artifact is a required or optional deliverable. As part of this study, these artifacts were document and rated.

Selected project leads resisted meeting to review projects and deliverables. The reluctance was overcome by explaining how the study would proceed and how the study results would help the project leads. It was explained that the study results will help determine methodology elements to which process improvement could be applied. Simplification and streamlining would improve current processes and decrease the amount of time needed to document current processes.

[1] The discrepancy between there being thirteen project leads and sixteen projects is because several project leads managed multiple projects.

Project Definition

One problem encountered during this study step centered on the definition of a project. There was no project portfolio. When project lists were requested from the division chiefs, lists with program and customer names were provided instead of projects. To compensate for this shortcoming, program and customer names were gathered from each project lead and compiled into a comprehensive project list. Project leads did not having a standard definition for a project. One project lead was adamant that his resources did not work on projects. Rather, they worked on "tasks" representing customer requests. This definition required additional probing to gather the information defining the project scope, duration, and costs. It was determined that the project lead's task definition was synonymous with other project leads' project definition.

Examined Projects

The number of projects used for this study was limited to those completed within the first quarter of the fiscal year in which this study occurred and open projects during the second and third quarters of that fiscal year [8]. This sample was determined to find the most projects that would have the benefit of using the new enterprise processes. Projects meeting these requirements were further limited to those not in a "steady-state". The term "steady-state" is used by Federal Systems to describe the activities for sustaining projects already in production environments and generally refers to maintenance of existing applications. Sixteen projects were found which met the above requirements.

CMMI

The head of Federal Systems had outlined a strategic initiative to establish an improved software development methodology aimed to gain CMMI Level 3 maturity level rating and streamline the methodology to support successful software development projects. Capability Maturity Model Integration is a process-improvement framework for directing organizations to increase their software development performance. CMMI is the successor of Capability Maturity Model (CMM) and was developed at Carnegie Mellon University. The goal of the CMMI project was to integrate many different maturity models into one framework, thereby improving usability. When working within a CMMI framework, one can take a continuous (capability) perspective for each activity or process area within an organization, and at the same time one can take a staged (maturity) view of the whole organization. While specific activities or process areas may be continuously improving, the overall organization progresses through discretely-identifiable stages of maturity over time as all process areas as a whole obtain increasing levels of capability. Federal systems' goal was to achieve CMMI level 3 – a staged maturity perspective.

Process Areas

CMMI provides guidance for organizations wishing to improve their processes. CMMI does not actually contain any processes, procedures, or work instructions; rather, it defines process areas that should be addressed, at increasing levels of capability, as an organization matures its software development process practices. CMMI defines a process area as "a cluster of related practices in an area that, when implemented collectively, satisfies a set of goals considered important for making improvement in that area" [3, p. ii]. While there are different implementations of CMMI models, each focuses on its particular area of interest; all contain the core process areas. These process areas contain best practices including concepts in project and process management, support, and engineering. In addition to these core process areas, each model also contains other process areas specific to that model.

Goals and Practices

CMMI process areas have associated goals and practices, generic and specific. Each process area has between one and three generic goals, which in turn include generic practices. As the name indicates, generic goals and practices are the same between all process areas. Each process area also has at least one specific goal which contains at least two specific practices. These specific goals and practices are unique to each process area. A process area is considered satisfied when actual work practices of an organization cover all the goals and practices in that process area.

CMMI Process Areas

As illustrated in Figure 1, there are three types of CMMI models: CMMI for Development, CMMI for Services, and CMMI for Acquisition [3]. The three CMMI models share 16 core process areas. In addition, each model has process areas unique to its functional activity. CMMI for Development, which was the target for Federal Systems, has seven unique process areas. These models contain generic goals and generic processes that apply across multiple process areas. As one could conclude from the process area names, goals and practices are essential (hence, required) to achieving improvement in each process area. These include the general and specific goals [3].

CMMI Model for Development

A maturity level is "a defined evolutionary plateau for organizational process improvement" [3; 7, p. 2]. There are five maturity levels designated by numbers 1 through 5:

1. Initial: The software process is described as ad hoc. Sometimes the software process is referred to as chaotic. Processes are generally not defined, and success depends on individual effort.
2. Managed: Project-oriented management processes are implemented to build functionality, track cost, and schedule. A process discipline is established to replicate earlier successes on new projects with similar characteristics.
3. Defined: The processes for both project-oriented management and software engineering activities are documented, standardized, and integrated into an organization-wide software process. Projects use a documented and approved version of the organization's processes for developing and maintaining software.
4. Quantitatively Managed: Quantitative measures of the software process and product quality are collected. The software process and product quality are quantitatively analyzed, understood and controlled using detailed measures.
5. Optimizing: Continuous feedback improvement is established through quantitative feedback from the processes and from innovative ideas and technologies.

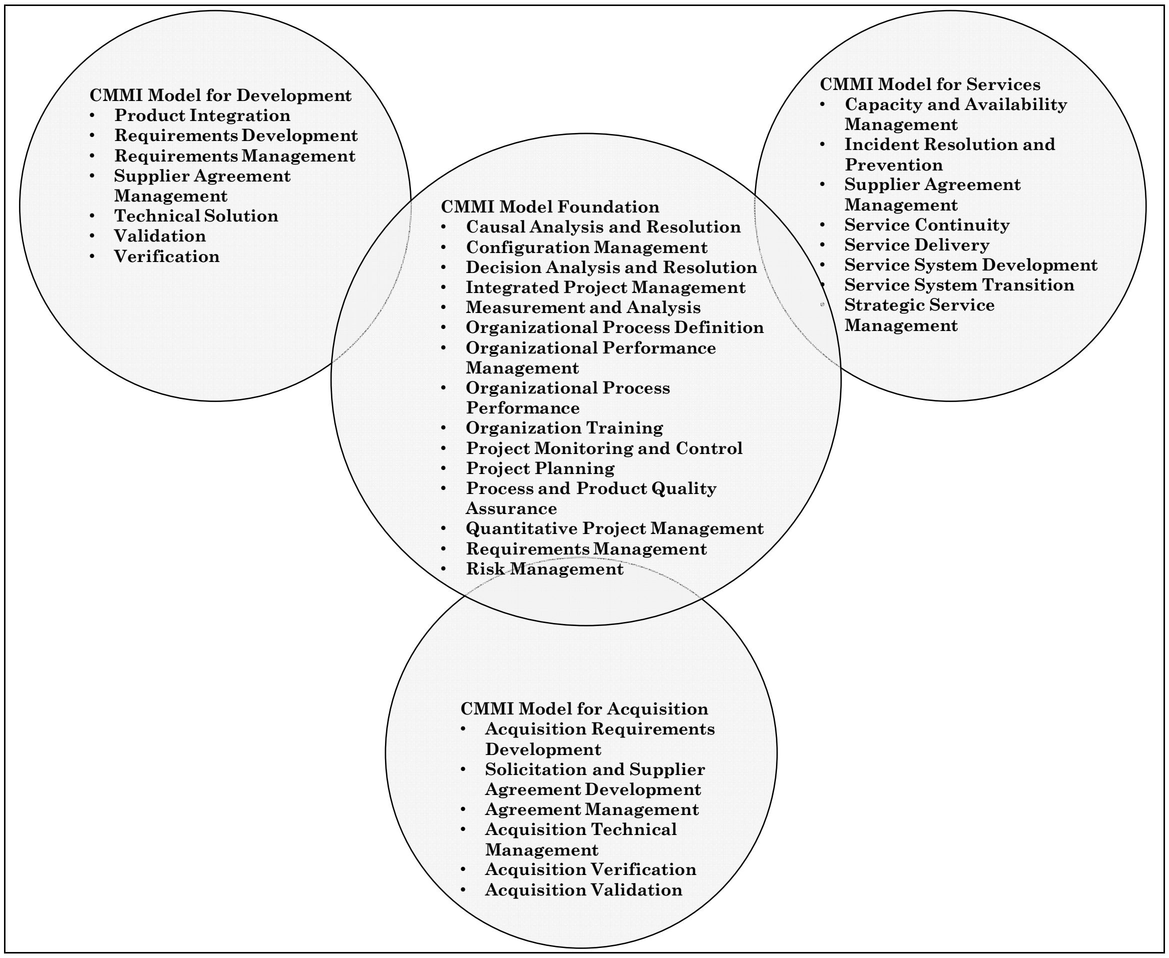
CMMI Model for Development
• Product Integration
• Requirements Development
• Requirements Management
• Supplier Agreement Management
• Technical Solution
• Validation
• Verification
CMMI Model Foundation
• Causal Analysis and Resolution
• Configuration Management
• Decision Analysis and Resolution
• Integrated Project Management
• Measurement and Analysis
• Organizational Process Definition
• Organizational Performance Management
• Organizational Process Performance
• Organization Training
• Project Monitoring and Control
• Project Planning
• Process and Product Quality Assurance
• Quantitative Project Management
• Requirements Management
• Risk Management
CMMI Model for Services
• Capacity and Availability Management
• Incident Resolution and Prevention
• Supplier Agreement Management
• Service Continuity
• Service Delivery
• Service System Development
• Service System Transition
• Strategic Service Management
CMMI Model for Acquisition
• Acquisition Requirements Development
• Solicitation and Supplier Agreement Development
• Agreement Management
• Acquisition Technical Management
• Acquisition Verification
• Acquisition Validation

Figure 1: CMMI Models

Each level is a foundation for the next. As such, each level measures the achievement of the specific and generic goals within a set of associated process areas. Improvement is thus measured using maturity levels. Federal Systems sought to achieve maturity level 3 in CMMI for development. Maturity levels 2 and 3 for CMMI for Development consist of the below process areas:

Maturity level 2 – Managed (seven processes areas)

- Configuration management (core)
- Measurement and analysis (core)
- Project monitoring and control (core)
- Project planning (core)
- Process and product quality assurance (core)
- Requirements management (core)
- Supplier agreement management (dev)

Maturity level 3 – Defined (eleven process areas)

- Decision analysis and resolution (core)
- Integrated project management (core)
- Organizational Process Definition (core)
- Organizational process focus (core)
- Organizational training (core)
- Product integration (dev)
- Requirements development (dev)
- Risk management (core)
- Technical solution (dev)
- Validation (dev)
- Verification (dev)

ANALYSIS

Federal Systems SDLC Architectural Framework

Federal Systems has a systems development lifecycle model as an architectural foundation for its software development projects. The lifecycle model is composed of four elements: 1) phases, 2) required deliverables, 3) "when needed" deliverables, and 4) events. As shown in Figure 2, the lifecycle contains six phases including 1) Plan, 2) Analyze, 3) Design, 4) Build, 5) Verify/Validate, and 6) Deploy. Each phase has 1) required deliverables, 2) "when needed" deliverables, and 3) "events" [5].

Each of these six phases has a number of required deliverables (a total of 30) that provide essential project management and system engineering knowledge. In Phase 1 (Plan), the project charter and project plan are key project management deliverables. In Phase 2 (Analyze), the key deliverables, requirements document and use case document, are software engineering requirements. Phase 3 (Design) concentrates on design and software testing with a high-level design document and test plan. As the lifecycle model transitions to Phase 4 (Build), the key deliverables provide coded components that will be tested in Phase 5 (Verify/Validate). Both verification and validation testing deliverables are produced in Phase 5. Last, Phase 6 (Deploy) results in the constructed and tested software being deployed into production. Each phase also has "when needed" deliverables as shown in Figure 2. A "when needed" deliverable (a total of 20) is completed at the discretion of the project team. If the breadth, depth, and complexity of a project justifies a "when needed" deliverable, it is included as a project deliverable [5].

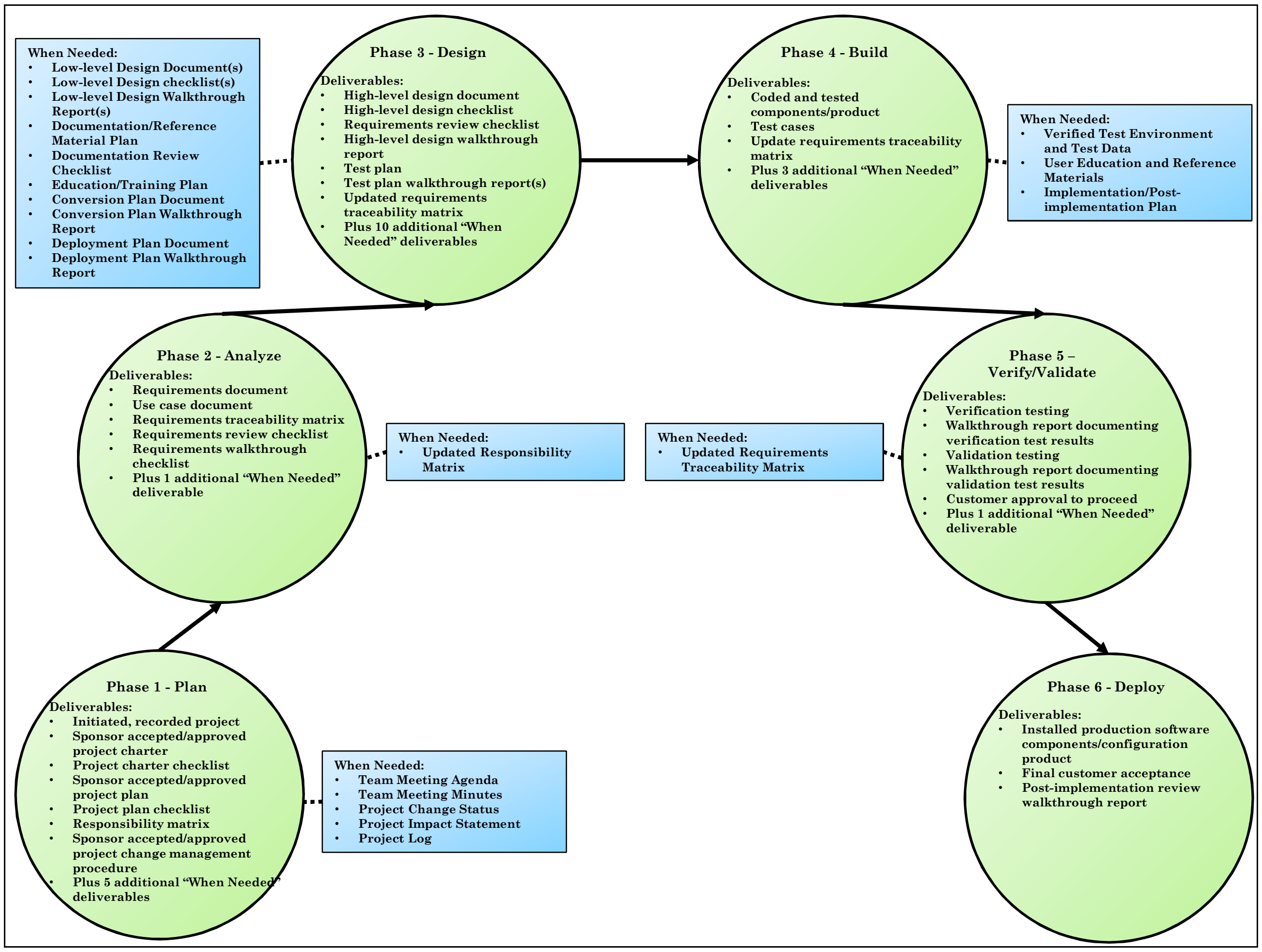
When Needed:
• Low-level Design Document(s)
• Low-level Design checklist(s)
• Low-level Design Walkthrough Report(s)
• Documentation/Reference Material Plan
• Documentation Review Checklist
• Education/Training Plan
• Conversion Plan Document
• Conversion Plan Walkthrough Report
• Deployment Plan Document
• Deployment Plan Walkthrough Report
Phase 3 - Design
Deliverables:
• High-level design document
• High-level design checklist
• Requirements review checklist
• High-level design walkthrough report
• Test plan
• Test plan walkthrough report(s)
• Updated requirements traceability matrix
• Plus 10 additional "When Needed" deliverables
Phase 4 - Build
Deliverables:
• Coded and tested components/product
• Test cases
• Update requirements traceability matrix
• Plus 3 additional "When Needed" deliverables
When Needed:
• Verified Test Environment and Test Data
• User Education and Reference Materials
• Implementation/Post-implementation Plan
Phase 2 - Analyze
Deliverables:
• Requirements document
• Use case document
• Requirements traceability matrix
• Requirements review checklist
• Requirements walkthrough checklist
• Plus 1 additional "When Needed" deliverable
When Needed:
• Updated Responsibility Matrix
When Needed:
• Updated Requirements Traceability Matrix
Phase 5 – Verify/Validate
Deliverables:
• Verification testing
• Walkthrough report documenting verification test results
• Validation testing
• Walkthrough report documenting validation test results
• Customer approval to proceed
• Plus 1 additional "When Needed" deliverable
Phase 1 - Plan
Deliverables:
• Initiated, recorded project
• Sponsor accepted/approved project charter
• Project charter checklist
• Sponsor accepted/approved project plan
• Project plan checklist
• Responsibility matrix
• Sponsor accepted/approved project change management procedure
• Plus 5 additional "When Needed" deliverables
When Needed:
• Team Meeting Agenda
• Team Meeting Minutes
• Project Change Status
• Project Impact Statement
• Project Log
Phase 6 - Deploy
Deliverables:
• Installed production software components/configuration product
• Final customer acceptance
• Post-implementation review walkthrough report

Figure 2: Federal Systems Software Development Lifecycle

Federal Systems' fourth lifecycle model element is an "event". Federal Systems defines an event similarly to a task in that it is work assigned and completed as a responsibility of one of the project's roles such as a stakeholder, project manager, team member, business analyst, or product assurer or integrator. Many of the methodology's 28 events correspond to knowledge area processes in PMI's PMBOK® [10, p. 526]. When mapping phases, deliverables, and events shown in Figure 3, the lifecycle model demonstrates integration among these elements [5]. For example, in Phase 1 (Plan), the phase events, Create project charter and Create project plan, define integrated work activities (see the dashed arrow in Figure 3) that generates these four required deliverables. Architecturally, this integration is retained in each lifecycle phase in which one or more events result in required deliverables [5].

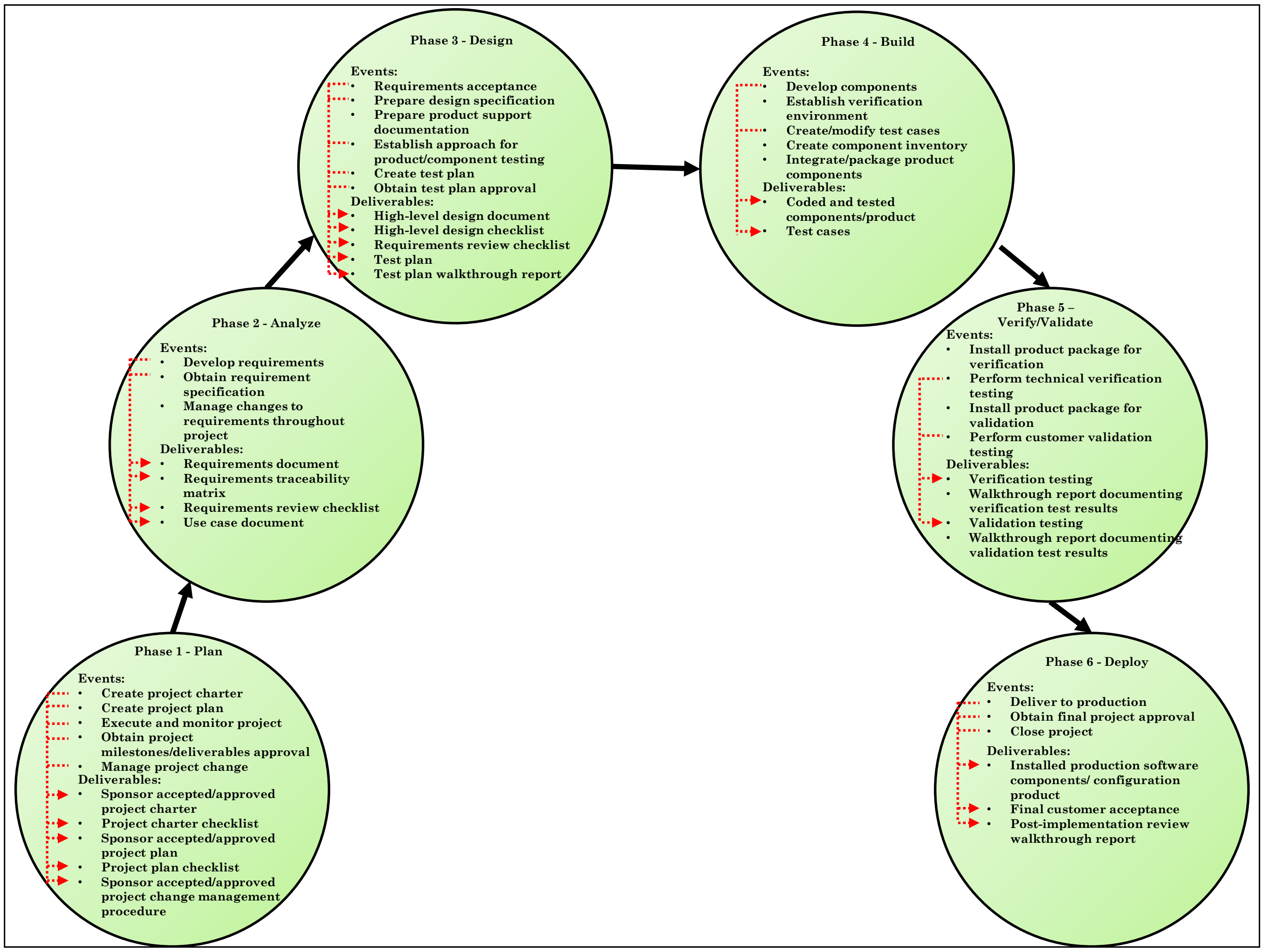
Phase 3 - Design
Events:
Requirements acceptance
Prepare design specification
Prepare product support documentation
Establish approach for product/component testing
Create test plan
Obtain test plan approval
Deliverables:
High-level design document
High-level design checklist
Requirements review checklist
Test plan
Test plan walkthrough report
Phase 4 - Build
Events:
Develop components
Establish verification environment
Create/modify test cases
Create component inventory
Integrate/package product components
Deliverables:
Coded and tested components/product
Test cases
Phase 2 - Analyze
Events:
Develop requirements
Obtain requirement specification
Manage changes to requirements throughout project
Deliverables:
Requirements document
Requirements traceability matrix
Requirements review checklist
Use case document
Phase 5 – Verify/Validate
Events:
Install product package for verification
Perform technical verification testing
Install product package for validation
Perform customer validation testing
Deliverables:
Verification testing
Walkthrough report documenting verification test results
Validation testing
Walkthrough report documenting validation test results
Phase 1 - Plan
Events:
Create project charter
Create project plan
Execute and monitor project
Obtain project milestones/deliverables approval
Manage project change
Deliverables:
Sponsor accepted/approved project charter
Project charter checklist
Sponsor accepted/approved project plan
Project plan checklist
Sponsor accepted/approved project change management procedure
Phase 6 - Deploy
Events:
Deliver to production
Obtain final project approval
Close project
Deliverables:
Installed production software components/ configuration product
Final customer acceptance
Post-implementation review walkthrough report

Figure 3: Federal Systems Deliverable and Event Mapping

As described, the Federal Systems SDLC suggests a waterfall approach, but supports iteration by permitting repetitive development of key deliverables within a project or a project phase. With this practice, Federal Systems relaxes the limitation of sequential phases and promotes the benefits of iterative development.

High Level Federal Systems Software Development Lifecycle Critique

Table 1 contains the detail of the Federal Systems Software Development Lifecycle critique. In general, the categories of weaknesses were identified including event and project phase timing, misleading deliverable and event names, and missing milestones. One of the most significant weaknesses in the lifecycle is event and project phase timing. Three events, 1) Execute and monitor project, 2) Manage project change, and 3) Manage changes to requirements throughout project, are mapped to a single SDLC phase each – Plan, Plan, and Analyze, respectively. In practice, these events are conducted during at least five lifecycle phases including Plan, Analyze, Design, Build, and Verify/Validate. Architecturally, it is incorrect to include these lifecycle elements in only one phase.

Table 1: Federal Systems Software Development Lifecycle Critique

SDLC Phase	Life Cycle Model Element	Comment
1-Plan	Deliverables	
	• Initiated, recorded project	There is no event to generate this deliverable.
	• Sponsor accepted/approved project charter	
	• Project charter checklist	
	• Sponsor accepted/approved project plan	
	• Project plan checklist	There is no event to generate this deliverable.
	• Responsibility matrix	
	• Sponsor accepted/approved project change management procedure	
	• 5 additional "When Needed" deliverables	• There is no criteria for when these deliverables are needed. • There are no events to generate these deliverables.
	Events	
	• Create project charter	
	• Create project plan	
	• Execute and monitor project	This event's time reference spans more than Phase – 1 Plan. What is the implication in the other phases?
	• Obtain project milestones/deliverables approval	
	• Manage project change	This event's time reference spans more than Phase – 1 Plan. What is the implication in the other phases?

Table 1 (continued)

2-Analyze	Deliverables • Requirements document • Use case document • Requirements traceability matrix • Requirements review checklist • Requirements walkthrough checklist • Updated responsibility matrix • 1 additional "When Needed" deliverable	This deliverable is not a different deliverable than "responsibility matrix" in Phase 1 – Plan. Why is it defined again? Any of the deliverables, not just the responsibility matrix, can be updated via integrated change control. • There is no criteria for when this deliverable is needed. • There are no event to generate this deliverable.
	Events • Develop requirements • Obtain requirement specification • Manage changes to requirements throughout project	• Verbose – suggest "Manage requirements". • This event's time reference spans more than Phase – 2 Analyze. What is the implication in other phases?
3-Design	Deliverables • High-level design document • High-level design checklist • Requirements review checklist • High-level design walkthrough report • Test plan • Test plan walkthrough report(s) • Updated requirements traceability matrix • 10 additional "When Needed" deliverables	This deliverable is not a different deliverable than "responsibility matrix" in Phase 1 – Plan. Why is it defined again? Any of the deliverables, not just the responsibility matrix, can be updated via integrated change control. • There is no criteria for when these deliverables are needed. • There are no events to generate these deliverables.
	Events • Requirements acceptance • Prepare design specification • Prepare product support documentation • Establish approach for product/component testing • Create test plan • Obtain test approval	Note no verb in this event. Suggest "Accept requirements" or "Conduct requirements review". What is an approach? Is there a test approach deliverable?

Table 1 (continued)

4-Build	Deliverables • Coded and tested components/product • Test cases • Update requirements traceability matrix • 3 additional "When Needed" deliverables	There is inadequate definition of test levels including system, integration, and unit level testing. This deliverable is not a different deliverable than requirements traceability matrix in Phase 2 – Analyze. Why is it defined again? Any of the deliverables, not just the requirements traceability matrix; can be updated via integrated change control. • There is no criteria for when these deliverables are needed. • There are no events to generate these deliverables.
	Events • Develop components • Establish verification environment • Create/modify test cases • Create component inventory • Integrate/package product components	
5-Verify/ Validate	Deliverables • Verification testing • Walkthrough report documenting verification test results • Validation testing • Walkthrough report documenting validation test results • Customer approval to proceed • 1 additional "When Needed" deliverable	Deliverable implies activity more than a deliverable. Suggest this deliverable be named "verification testing findings". Suggest "Validation testing results". Suggest "customer verification/validation approval". Is this customer approval the first time the customer has participated in a phase review? • There is no criteria for when this deliverable is needed. • There are no event to generate this deliverable.
	Events • Install product package for verification • Perform technical verification testing • Install product package for validation • Perform customer validation testing	

Table 1 (continued)

6-Deploy	Deliverables • Installed production software components/configuration product • Final customer acceptance • Post-implementation review walkthrough report	Suggest "Lessons learned".
	Events • Deliver to production • Obtain final project approval • Close project	Is there an organizational software configuration management process?

The naming of one of the lifecycle elements is a weakness in the methodology. An event is defined as follows:

- A stimulus that causes "a transition from one state to another", that causes "the system to change state" [11, p. 245; 9, p. 317].
- "A noteworthy occurrence that has a location in time and space. It occurs at a point in time; it does not have duration" [6].

The event names such as Create project charter and Create project plan correlate with the work activity conducted to gain, organize, and finalize the deliverable content, not the point in time when the task is completed [Project Management Institute 2013, 526]. This definitional inconsistency persists through the lifecycle. Other events, such as Execute and monitor project, Develop requirements, Obtain requirements specification, Develop components, and Create/modify test cases exhibit similar definitional inconsistency. Each of these events is more a task required to be done as a duty of a project role rather than an arrival of a significant point in time.

As element naming is a lifecycle weakness, the naming of deliverables is also a flaw. Consider the Verification testing deliverable in Phase 5 – Verify/Validate. Testing connotes an activity rather than a deliverable. By definition, software testing is the process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software item [1]. Note the word "process" in the definition. Thus, the deliverable name Verification testing implies the process of verification testing rather than a deliverable associated with verification testing. A more understandable name for this deliverable might be Verification test findings. This confusing deliverable naming convention is also seen in the deliverable, Validation testing.

Best practices should include important milestones. In particular, quality assurance milestones could enhance the Federal Systems lifecycle. Useful milestones for system requirement review (SRR), preliminary design review (PDR), critical design review (CDR), production readiness review (PRR), test readiness review (TRR), and operational readiness review (ORR) might provide benefit to project success [12].

FEDERAL SYSTEMS LIFECYCLE PHASE AND CMMI LEVELS 2 AND 3 MAPPING

CMMI defines seven process areas for Level 2 rating and eleven process areas for Level 3 rating. In Table 2, the process areas are presented in the horizontal rows and the SDLC events are shown in the vertical columns. SDLC defines six processes for the Plan phase, three processes for Analyze phase, and six processes for the Design phase. Each SDLC process maps to one or more CMMI process areas. For example, the SDLC Develop requirements event maps to two CMMI process areas, 1) Requirements Management and 2) Requirement Development. Overall, the SDLC events in the first three SDLC life cycle phases map to and address four of the seven Level 2 process areas. Three are gaps between SDLC and CMMI in the Supplier Agreement Management, Measure and Analysis and Configuration Management process areas. With respect to CMMI Level 3 process areas, five SDLC events in the first three life cycle phases map to and address five of the eleven process areas [3].

It was noted that Federal Systems events are architected to achieve an appropriate goal for CMMI process areas by recommending generic practices. The recommended practices might be categorized guidelines and were generally "inferred or verbal" rather than "written." As a result, practice compliance was difficult to evaluate. In selected practice areas (such as Process and Product Quality Assurance), practices for product quality were observed (such as peer reviews and desk audits) while process quality practices were not as verifiable. This lack of accountability results in confusion regarding whether the mapping accurately validates compliance with generic or specific practices as targeted by CMMI.

Table 2: Mapping of Federal Systems SDLC Phases 1, 2 and 3 to CMMI Levels 2 and 3 Process Areas

CMMI Capability Level	Process Area	Phase 1 – Plan						Phase 2 - Analyze			Phase 3 - Design					
		Create Project Charter	Obtain project charter approval	Create project plan	Obtain project plan approval	Obtain project milestones/deliverables approval	Manage project change	Develop requirements	Obtain requirements specification	Manage changes to requirement	Requirements acceptance	Prepare design specification	Prepare product support documentation	Establish approach for project/component testing	Create test plan	Obtain test approval
Capability Level – 2 Managed	Requirements Management							X	X	X	X					
	Project Planning	X	X	X	X	X				X	X					
	Project Monitoring and Control						X									
	Supplier Agreement Management															
	Measure and Analysis															
	Process and Product Quality Assurance													X	X	X
	Configuration Management															
Capability Level 3 - Defined	Requirements Development							X	X	X	X					
	Technical Solution											X				
	Product Integration												X			
	Verification													X	X	X
	Validation													X	X	X
	Organizational Process Focus															
	Organizational Process Definition															
	Organizational Training															
	Integrated Project Management															
	Risk Management															
	Decision Analysis and Resolution															

When the last three lifecycle phases of Federal Systems' SDLC are examined, the SDLC events provide improved mapping to CMMI Level 2 and 3 process areas. As shown in Table 3, two more SDLC events map to CMMI process areas, Measure and Analysis and Configuration Management. In total, SDLC events address six of the seven process areas for CMMI Level 2 rating. In the later SDLC lifecycle phases, denser mapping is found between the same five process areas successfully mapped in the earlier SDLC lifecycle phases. The five CMMI Level 3 process areas including Requirements Development, Technical Solution, Product Integration, Verification, and Validation successfully map to all SDLC lifecycle phases except Planning. Unfortunately, the later SDLC events do not provide improved coverage of existing gaps from the earlier SDLC phase events, and only five of eleven CMMI Level 3 process areas are addressed [3].

Table 3: Federal Systems SDLC Phases 4, 5 and 6 to CMMI Levels 2 and 3 Process Area Mapping

		Phase 4 - Build				Phase5 – Verify/ Validate				Phase 6 – Deploy		
CMMI Capability Level	Process Area	Develop components	Establish verification environment	Create/modify test cases	Integrate package /product components	Install product package for verification	Perform customer verification testing	Install product package for validation	Perform customer validation testing	Deliver to production	Obtain final project approval	Close project
Capability Level – 2 Managed	Requirements Management											
	Project Planning											
	Project Monitoring and Control										X	X
	Supplier Agreement Management											
	Measure and Analysis						X		X			
	Process and Product Quality Assurance		X	X		X	X	X	X			
	Configuration Management		X									
Capability Level 3 – Defined	Requirements Development											
	Technical Solution	X										
	Product Integration	X			X					X		
	Verification		X	X		X	X					
	Validation							X	X			
	Organizational Process Focus											
	Organizational Process Definition											
	Organizational Training											
	Integrated Project Management											
	Risk Management											
	Decision Analysis and Resolution											

Descriptive Summary of Federal Systems SDLC Practices

Table 4 shows that of the sixteen projects examined, four were completed projects, two of which took place completely outside of Federal Systems' SDLC processes.

Table 4: Latest Phase by Project

Project	Latest Phase Project In or Completed
J	1 (Plan)
K	3 (Design)
O	3 (Design)
P	3 (Design)
E	3 (Design) - 4 (Build)
M	4 (Build)
I	4 (Build)
H	4 (Build)
A	4 (Build)
D	5 (Verify/Validate)
B	4 (Build) - 6 (Deploy)
L	6 (Deploy)
G	Completed
C	Completed
F	Completed, and used methodology other than SDLC
N	Completed, and used methodology other than SDLC

The remaining projects covered all six phases of the SDLC, ranging from one project still in Phase – 1 Planning to two projects still in Phase – 6 Deploy. Artifacts for deliverables by phase were found at very low rates across all six phases, as can be seen in Table 5. The percentages indicate the number of deliverables found divided by the number of opportunities for deliverables to have been created. Recall that the SDLC defines deliverables as either required or optional ("When Needed"). Table 5 displays the comparison for total deliverables (both required and optional combined) and required deliverables. As seen in the table, the usage rates by phase are very similar between the two groups (required and "when needed" deliverables vs just required deliverables). Because of uncertainty regarding the data from which this summary was derived, it is unclear whether these low rates are truly because there was an issue with the defined process being followed, or simply because the phase being summarized was not yet fully complete. This issue is addressed in the analysis shown in Table 8 below.

Table 6 shows artifacts for deliverables for each project surveyed. The % of deliverables is based upon the latest phase the project was in or had completed. The variance by project was even more enlightening. Four projects had no artifacts. On the more productive end of the spectrum, the project with the most artifacts completed had 42% of all deliverables and 54% of required deliverables. The average for overall deliverables was 11% with the average for required deliverables slightly higher at 14%.

One variable not taken into consideration above was the duration of the project. Of the sixteen projects, approximate duration was available on eleven. It is interesting to note however, duration of the project does not seem to correlate with the usage rates. Note in Table 7 that the longest duration project, Project N, has the highest delivery rate. However, the project with the second longest duration, Project J, had the lowest % of deliverables. Of the 11 projects with known duration information, the average duration was just over six months (6.02) and the average percentage of deliverables was 12% for overall deliverables and 15% of required deliverables.

Table 5: Number of Required and "When Needed" Deliverables Produced by Phase

SDLC Phase (Latest Phase Project In or Completed)	# Projects	% of Required and "When Needed" Deliverables	% of Required Deliverables
1 – Plan	1	16%	18%
2 – Analyze	0	16%	19%
3 – Design	3	4%	7%
4 – Build	5	15%	14%
5 – Verify/Validate	1	0%	0%
6 – Deploy	2	11%	11%

Table 6: Number of Required and "When Needed" Deliverables Produced by Project

Project	% of Required and "When Needed" Deliverables	% of Required Deliverables
J	0%	0%
K	0%	0%
O	0%	0%
P	0%	0%
E	2%	4%
M	4%	6%
I	7%	4%
H	9%	4%
A	12%	17%
D	13%	13%
B	15%	21%
L	16%	23%
G	17%	19%
C	20%	33%
F	24%	26%
N	42%	54%
Average	11%	14%

Table 7: Number of Deliverables Produced by Project Duration

Project	Approximate Project Duration (Months)	% of Required and "When Needed" Deliverables	% of Required Deliverables
E	3.00	2%	4%
M	3.00	4%	6%
B	3.50	15%	21%
A	3.75	12%	17%
D	4.00	13%	13%
F	4.00	24%	26%
K	5.00	0%	0%
H	6.00	9%	4%
G	7.00	17%	19%
J	13.00	0%	0%
N	14.00	42%	54%

Compliance Summary

In order to achieve CMMI Level 2 rating, Federal Systems must comply with the specific practices of process areas. CMMI Level 2 process areas focus on project management. Recall that The Federal Systems SDLC successfully mapped to five of the seven process areas with only two process gaps, Supplier Agreement Management and Measure and Analysis. In order to determine if projects were utilizing generic or specific practices within Federal Systems events, the required deliverables were examined. As shown in Table 8, the completion rate of required project management deliverables varied from 0% to 31% with the Sponsor accepted/approved project charter and the Sponsor accepted/approved project plan being the two most frequently completed project management deliverables. There were five project management deliverables not produced in any project. Two of these not completed deliverables included the Customer approval to proceed and the Final customer acceptance. Overall, the low rate of completed required deliverables indicated low compliance with SDLC events associated with CMMI process areas. With respect to optional project management deliverables, several projects elected to produce Team meeting agendas, Team meeting minutes, Project change requests, and Project impact statements. The completion rate of these optional project management deliverables were 31%, 19%, 6%, and 6%, respectively. As with required project management deliverables, the low rate of completed optional deliverables seem to indicate low compliance with SDLC events associated with CMMI process areas.

Table 8: Required Project Management Deliverables

SDLC Phase	# Projects Through Phase	Project Management Artifact Name	# of Existing Complete Artifacts	Existing Complete Artifacts % of Total	# of Existing Incomplete Artifacts	Existing Incomplete Artifacts % of Total	# Missing Artifacts	Missing Artifacts % of Total
1 - Plan	16	Initiated, recorded project	3	19%	0	0%	13	81%
		Sponsor accepted/ approved project charter	5	31%	0	0%	11	69%
		Project charter checklist	0	0%	0	0%	16	100%
		Sponsor accepted/ approved project plan	5	31%	0	0%	11	69%
		Project plan checklist	1	6%	0	0%	15	94%
		Responsibility matrix	3	19%	0	0%	13	81%
		Sponsor accepted/ approved change management procedure	1	6%	0	0%	15	94%
2 - Analyze	15	Requirements review checklist	0	0%	1	7%	14	93%
3 - Design	15	Requirements review checklist	0	0%	0	0%	15	100%
4 - Build	12	None						
5 - Verify/ Validate	7	Customer approval to proceed	0	0%	0	0%	7	100%
6 - Deploy	6	Final customer acceptance	0	0%	0	0%	6	100%
		Post-implementation review walkthrough report	1	17%	0	0%	5	83%
Grand Total			19	11%	1	1%	141	80%

CMMI Level 3 requires that Federal Systems must comply with the specific practices of process areas focused on software engineering as well as process and project management. Recall that the Federal Systems SDLC successfully mapped to fewer process areas with six process gaps (See Tables 2 and 3). In order to determine if projects were utilizing generic or specific practices when conducting the five SDLC events mapped to Level 3, the required software engineering deliverables were examined. As shown in Table 9, the completion rate of required software engineering deliverables varied from 0% to 40% representing a slight improvement over required project management deliverables. The two most frequently completed required deliverable are the Requirements and the High level design documents. There were nine software engineering deliverables not produced in any project. Two of these not completed deliverables included the Verification and Validation testing documents. As with required project management documents, the low rate of completed required deliverables seems to support the idea of low compliance of SDLC events associated with CMMI process areas. With respect to optional project management deliverables, several projects elected to produce Low-level design, Deployment plan document, Verified test environment and test data, and User education and reference materials. The completion rate of these optional software engineering deliverables were 20%, 7%, 8%, and 33%, respectively. As with required software engineering deliverables, the low rate of completed optional deliverables also seems to support the idea of low compliance of SDLC events associated with CMMI process areas.

Table 9 : Required Software Engineering Deliverables

SDLC Phase	# Projects Through Phase	Software Engineering Artifact Name	# of Existing Complete Artifacts	Existing Complete Artifacts % of Total	# of Existing Incomplete Artifacts	Existing Incomplete Artifacts % of Total	# Missing Artifacts	Missing Artifacts % of Total
1 - Plan	16	N/A						
2 - Analyze	15	Requirements document	6	40%	0	0%	9	60%
		Use case document	4	27%	0	0%	11	73%
		Requirements traceability matrix	1	7%	0	0%	14	93%
		Requirements walkthrough report	2	13%	0	0%	13	87%
3 - Design	15	High-level design document	6	40%	0	0%	9	60%
		High-level design checklist	0	0%	0	0%	15	100%
		High-level design walkthrough report	2	13%	0	0%	13	87%
		Test plan	1	7%	0	0%	14	93%
		Test plan walkthrough report(s)	0	0%	0	0%	15	100%
		Updated requirements traceability matrix	0	0%	0	0%	15	100%
4 - Build	12	Coded and tested components/ product	2	13%	1	7%	9	60%
		Test cases	2	13%	0	0%	10	67%
		Updated requirements traceability matrix	0	0%	0	0%	12	80%
5 - Verify/ Validate	7	Verification testing	0	0%	0	0%	7	100%
		Walkthrough report documenting verification testing results	0	0%	0	0%	7	100%
		Validation testing	0	0%	0	0%	7	100%
		Walkthrough report documenting validation testing results	0	0%	0	0%	7	100%
6 - Deploy	6	Installed production software components/ configuration/ product	0	0%	0	0%	6	100%
Grand Total			26	12%	1	0%	193	88%

FINDINGS

Federal Systems has made progress toward implementing a more well-defined enterprise software development methodology. The methodology architecture consisting of phases, events, required deliverables, and "when needed" deliverables is defined and being utilized in funded software development projects. When mapped to process areas for CMMI Level 2 and Level 3 rating, the methodology falls short of the requirements to meet CMMI best practices. Specifically, the SDLC does not meet the CMMI Level 2 and Level 3 requirements for supplier management, organization process management and risk management and SDLC elements for these CMMI requirements should be defined. A small percentage of the required deliverables (no more than 30%) were produced by the sampled 16 products, and an equally small percentage of the "when needed" deliverables (no more than 40%) were produced. This low percentage represents low enterprise deployment and acceptance of the methodology as an enterprise methodology.

For improved understandability and usability, the SDLC should be revised for consistent definitions and life cycle implementation. Events, such as Execute and monitor project, which are positioned in a single lifecycle phase and are activities in other lifecycle phases should be clarified so that project team members can accurately implement and manage them during project execution. When improved understanding can be achieved through simplification, methodology element names should be simplified. For example, a verbose event such as "Manage changes to requirements throughout project" can be renamed "Manage requirements" and retain its effectiveness in defining project plan activities.

Simplification is more valuable than volume. Deliverables do not substitute for processes. Thirty required deliverables, twenty "when needed" deliverables and thirty events are voluminous. Clear architectural structure is needed among required deliverables, "when needed" deliverables, and events. The structure should define what event is associated with which deliverable. The most important contribution of a deliverable is its value-added content. A single document such as a requirements document can be defined with internal content (in an outline format) to support functional requirements, nonfunctional requirements, business use cases, and requirements traceability. A narrow definition of a key software engineering deliverable in each phase would reduce the required deliverable volume and promote consistent deliverable content which can be customized to the breadth and depth of the project.

While simplification provides improved understanding and promotes consistency, specifying a required development approach would improve the process framework. Given there are twenty "when needed" deliverables, it is not possible to determine if the SDLC is being utilized since a project does not identify which "when needed" deliverables have been determined to be elements of a project. To complement this element, the SDLC needs a development approach deliverable. This deliverable would be completed at the beginning of the project by stating which "when needed" deliverables are selected as deliverables. For example, if a project selected the "when needed" deliverable, low-level design documents, this deliverable would be a required deliverable in the project. Project execution and compliance would ensure that the event associated with work activity is included and the produced deliverable is reviewed and approved. In this role, a required development approach would be an important deliverable identifying the role of each of the twenty "when needed" deliverables, and accountability would be established during project execution to ensure the selected "when needed" deliverables are produced.

Another important recommendation pertaining to SDLC effectiveness is the quality review of deliverables. Deliverable review points should be established for required and selected "when needed" deliverables. Deliverable review points could coincide with phase ends or could align with events which result in deliverables. No matter which approach is implemented, the quality review points could utilize checklist and walkthrough reports established in the current methodology. Two additional improvements can be realized with deliverable review points [12]. First, artifact storage can be standardized and possibly automated so that project artifacts are stored in a standard enterprise repository. When stored in a standard repository, compliance is achieved and audits can be completed in a more economical manner. Second, deliverable review points and standard artifact storage processes can be implemented using automated workflow. Data repository technologies, such as Microsoft SharePoint and Documentum EMC, have integrated workflow functions that can be used to create, route, review, and approve documents. Automated workflow contributes to improved timing and accountability among project team members. Automated workflow functions also integrate an enterprise development methodology into the daily work environment of the project team making it more attractive to the team and improving the acceptance and utilization during project execution.

CONCLUSIONS

Analysis of the Federal Systems initiative to implement an enterprise software development methodology confirmed the use of their SDLC in the deployment of its phases, events, required deliverables, and "when needed" deliverables but it also revealed shortcomings in the success of its deployment as an enterprise standard. First, the Software Development Lifecycle processes, represented by its events, did not fully map to CMMI Level 2 and Level 3 process areas. Secondly, the execution of the processes with their associated work for and completion of the required deliverables was not compliant, since a large percentage of required deliverables were either missing or incomplete. Finally, if "when needed" deliverables were selected to be used during a project, the associated work for and completion of the "when needed" deliverables were at a fairly low completion rate. It is not possible from the data available to determine whether this outcome is a result of them simply not being needed or because they were just not completed "when needed", this would be an area that would benefit from further investigation. Based on the low completion rates of the required deliverables, it may be likely that the "when needed" deliverables are also not being completed very frequently even when they are "needed". The SDLC software development methodology elements that were implemented are valuable ones but the utilization of the elements was not at an enterprise level.

A closer examination of sixteen software development projects showed that the success of the enterprise software development methodology was limited at both CMMI Level 2 and CMMI Level 3. Using CMMI benchmarks, SDLC project management and software engineering practices were weak. For project management practices, the measurement of compliance is the number of completed required project management deliverables. Since the percentage of required project management deliverables was low, the agency's adoption rate of CMMI Level 2 process areas was low. In the examination of software engineering practices, it was found that these practices were also weak, as measured by the number of completed software engineering deliverables. From these measurements, the Chief of Federal Systems can determine the tactical steps necessary to eliminate gaps between SDLC practice and CMMI requirements and to ensure compliance with these requirements through improved software development practices.

Federal Systems has subsequently determined that their attempt to move from CMMI Level 1 to CMMI Level 3 was too big of a step. Too many new process changes were attempted all at once, possibly leading to the low completion rate of required and "when needed" deliverables discussed above. The low completion rate serves as an important factor that Federal systems must consider. It is not enough for Federal Systems events to map to CMMI process areas. There must also be generic or specific goals and practices for these events in order to satisfy a CMMI appraisal resulting in a Level 2 or 3 rating. Because of this, Federal Systems simplified its process-improvement approach, focusing on basic project management tasks such as reporting and tracking. It hopes that this simplified approach will allow it to build a base of more mature processes, from which it can then extend to other CMMI process areas as it attempts to move to Level 3.

Because the Software Development Lifecycle elements worked successfully in selected process areas, its architectural structure lends itself toward enterprise standardization and serves as a foundation for CMMI appraisal. Using the current SDLC elements, Federal Systems can identify the barriers and constraints to enterprise adoption and formulate structural and process adjustments which obtain the targeted goal of CMMI best practices. Instead of utilizing a high volume of required deliverables, Federal Systems should implement several enhancements including a development approach deliverable, deliverable review points, central document repository and automated work flow. These methodology enhancements would further strengthen its architectural structure, maximize its best practices, and minimize the risk associated with non-compliance.

REFERENCES

[1] Beizer, B., *Software System Testing and Quality Assurance*, Van Nostrand Reinhold Company Inc., New York, NY, 1984.

[2] Carnegie Mellon Software Engineering Institute, *Introduction to the Architecture of the CMMI Framework*, Carnegie Mellon Software Engineering Institute, Pittsburgh, PA, 2007, pp. 1-17.

[3] Carnegie Mellon Software Engineering Institute, *CMMI for Development, Version 1.3*, Technical Report, Carnegie Mellon Software Engineering Institute, Pittsburgh, PA, 2010.

[4] Chrissis, M., Konrad, M., and Shrum, S., *CMMI for Development: Guidelines for Process Integration and Product Improvement*, Addison-Wesley Professional, Upper Saddle River, New Jersey, 2011.

[5] Federal Systems, *Federal Systems Engineering Life Cycle Documentation*, Federal Systems, 2011.

[6] Jacobson, I., Booch, G., and Rumbaugh, J., *The Unified Software Development Process*, Addison-Wesley, Boston, MA, 1999.

[7] Marcal, A., de Freitas, B., Soares, F., and Belchior, A., "Mapping CMMI Project Management Process Areas to SCRUM Practices", *Proceedings of the 31st IEEE Software Engineering Workshop* , Loyola College, Washington, DC, March 6-8, 2007, pp. 13-22.

[8] Muldoon, K., *Federal Systems Interview Process*, Interviews carried out by Kristi Muldoon at the Federal Systems organization, 2012.

[9] Pressman, R., *Software Engineering: A Practioner's Approach*, McGraw-Hill, New York, NY, 2005.

[10] Project Management Institute, *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, Project Management Institute, Inc., Newtown Square, PA, 2013.

[11] Sommerville, I., *Software Engineering*, Addison-Wesley, Boston, MA, 2011.

[12] U.S. Department of Defense, *Department of Defense Standard Practice MIL-STD-882D*, U.S. Department of Defense, Washington, DC, 2010.

AUTHOR BIOGRAPHIES

Charles W. Butler is Professor of Computer Information Systems in the College of Business at Colorado State University. His research explores the software development, software metrics and project management. He earned an MS from the University of South Florida and Ph.D. from Texas A&M University.

Kristi Muldoon is a Student Information System Application Specialist and Database Analyst at Brighton School District 27J in Brighton, Colorado. She is a nationally credentialed Infinite Campus Certified Administrator. She earned an MS from Colorado State University.

Dan Turk is Associate Professor of Computer Information Systems in the College of Business at Colorado State University. His research explores software development, software development metrics, and the value of various software development processes. He earned an MS from Andrews University and Ph.D. from Georgia State University.