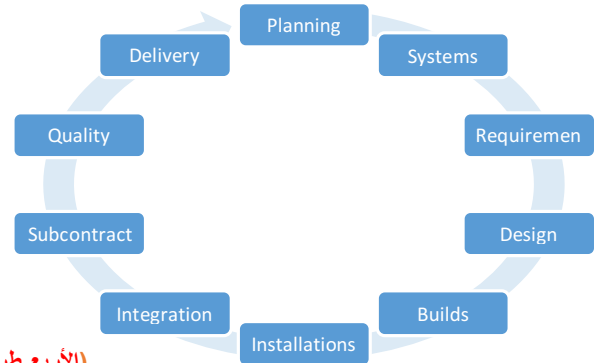


Chapter 1 (Introduction)

INTEGRATION METHODS (مهم و تُحفظ بالترتيب)

The success of software and systems integration capabilities is dependent upon a major discipline supporting the entire software life cycle.

To develop, operate, and maintain software and systems integration capabilities inside work product facilities, there must be a major discipline in supporting the entire software life cycle →



THE RIGHT DISCIPLINES (الأربع طرق هذي مهمه في كل مرحلة من مراحل التطوير)

Methods: effective methods improve productivity and help better prepare for future challenges that could impact integration environments

Software: design, code, plans, and test procedures integrated with applied systems ensure the software developed is done right, “Peer reviews” are key.

Systems: for systems to be deemed ready for the combination of software and systems integration they must have accomplished allocation of software design and engineering practices

Integration: software, systems, firmware, and hardware must all work together as on

PROGRAM AND PROJECT PLANNING (ماهو الغرض من هالمرحلة و ماهي النتيجة)

Purpose: to provide the necessary process steps to plan for systems and software design and development for integration.

Result: This type of planning will ensure effective results for performing the disciplines necessary to implement supporting software and systems integration activities.

SYSTEMS DESIGN (معرفة الطريقة في تصميم السستم و ما أهميتها)

Method: analyze customer requirements and develop a software design/development plan for defining the essential elements for a designed system to meet the specified requirements.

Importance: provides the disciplines required and implemented during software design/development life cycles.

هنا يتكلم عن تصميم السستم نفسه ، ف طريقة التصميم مهمه وهي تحليل متطلبات العميل و وضع خطه لتطوير التصميم وهذا كله يرجع على ان يتم تحديد architecture, components, modules, interfaces, and necessary data لتوافق متطلبات العميل.

SOFTWARE REQUIREMENTS (ماهو الغرض منها و كيف تستخدم و ماهي الطريقة لاستخدامها)

Purpose: the basis for software design and/or development

Use: provides a systematic approach to development from multiple resources.

Method: used for initial development of software requirements and changes to requirement baselines.

SOFTWARE DESIGN / DEVELOPMENT (يتكلم عن تعريفه و اين يتم و الطريقه)

Purpose: a systematic approach used to create software design and its development to reflect design and software definitions related to the work product. Provides traceability according to software-defined processes and procedures.

Requirements: established between the elements of the design/development. The documented program and project plan.

Method: defines details about the product construction, behavior, components, and interfaces.

SOFTWARE IMPLEMENTATION

Purpose: provides assurance that engineering builds function as expected.

Use: enables smooth execution for verification and test activities.

Method: incremental software and test approach adds the functions incrementally in a series of engineering builds. As software is tested, build plans are modified for succeeding builds based results previously demonstrated during troubleshooting, and checkout.

Importance: requirement for integration testing in a development, integration facilities, or the software systems integration facility (S/SIF).

طبعاً الشرط في هالمرحلة الي هي التنفيذ او البناء انوه يختبر التكامل في المشروع و يضمن بأن هندسة البرمجية تُبنى كما هو متوقع و مخطط لها.

SOFTWARE AND SYSTEMS INTEGRATION

Purpose: provides a consistent approach to integration to ensure that the software and systems elements are assembled properly.

Method: determines if created elements are prepared and subject to verification or validation

تتم معالجة جميع البرامج التي يتم تسليمها أو تنفيذها من خلال تكامل أو اختبار البرامج من خلال نظام مكتبة برامج يتم ضبطه والتحكم فيه والذي يحافظ على المحاسبة الرسمية لكل عملية تسليم. تتطلب مهام الدمج أو التكامل أن تكون عمليات تصميم وتطوير واختبار البرامج جاهزة لضمان أن التكامل جاهز لاستكشاف الأخطاء وإصلاحها.

SOFTWARE SUBCONTRACTOR

Role:

- to describe how a programs and projects will benefit from outside resources.
- Provides required software/hardware products to be under contract and effective.
- presentation to the customer must be understood from start of the presentation to the finish.

SOFTWARE AND SYSTEMS INTEGRATION DELIVERY

Purpose: to ensure that units tested are complete and documented prior to official delivery.

Requirement: integration testing to ensure both software and systems are integrated and working mutually.

في هالمرحلة يقوم باختبار التكامل للتسليم، بحيث ان يتم التأكد من التكامل بين الانظمة والبرامج قبل التسليم الفعلي للعميل.

PRODUCT EVALUATION

Purpose: provides the necessary process steps to conduct and perform continual evaluations of software products during the design/development life cycle and integration activities.

- evaluation tools and checklists are developed with related scheduled processes to conduct the mandatory audits and evaluations.

هنا يتكلم عن تقييم المنتجات بشكل مستمر خلال software life cycle.

Quality First is the most important method. In order to have an impact on the integration and quality of software in addition to meeting scheduled delivery deadlines, hardware and software designers MUST work together to solve issues.

Chapter 2 (The First Step For life cycle)

Program and project planning is important as it describes the necessary planning for software and system efforts during software design/development life cycles. the definitions of systems design, software requirements and design, configuration control, systems and software integration, subcontractor involvement, deliveries, and product quality evaluations are critical to effective planning efforts.

ESSENTIAL ELEMENTS

- Definitions of systems design
- Configuration control
- Subcontractor involvement
- Product quality evaluations
- Software requirements and design
- Systems and software integration
- Deliveries

Before a **program can require a plan**, program objectives are defined and **technical** and **management disciplines** are identified. this information defines a reasonable estimate or cause for:

PROGRAM PLAN DEVELOPMENT

- Cost evaluations
- Risk management assessments
- Defined and documented tasks
- Manageable schedules
- Progress reports

Effective programs that perform to defined **objectives** and within the scope are successful due to implementing:

EFFECTIVE PROGRAMS

- Required data
- How the work product performs
- Tasks or functions
- Quantitative mechanisms

Software processes provide the **framework and effective planning** when it is time for deliveries to software and systems integration facilities and the customer.

ESTABLISHED FRAMEWORK

- Effective planning entails multiple tasks, scheduled milestones, and quality aspects for everyone involved from management to employees.
- Configuration management personnel monitor the framework process.

The main reason that **software projects** are planned and controlled is to eliminate any confusion that could occur. There are times when systems and software designers do **not communicate**, so defined requirements are not discussed in relation to the developed work product. To eliminate this lack of communication, **guidelines must be established**, such as:

PROJECT GUIDELINES

- Daily meetings
- Idea sharing
- Keep project managers informed
- Complaint resolution

Communication planning principals define **goals and objectives** during the course of program and project planning. Also, Define and understand quality.

COMMUNICATION PLANNING PRINCIPALS

- Establish a set of managers who:
 - Understand the technical practices that support systems and software engineering
 - Can clearly define and provide a scope for the team defining the development stages
 - Provide a scope for the team to know what is ahead
 - Involve systems and software teams to help with delivery schedules
 - Can accommodate change and identify potential risks that impact on program and project planning
 - Track the progress daily and adjusting if needed

At the **senior management level**, program and project managers are required to provide **effective planning** and **focus** so teams can be effective during software design/development activities.

SENIOR MANAGEMENT

Role:

- provide the common framework for program and project planning to address engineering tasks

Responsibilities:

- Communicate efficiently and manage a team wisely
- Implement and use reasonable schedules
- Oversee the development of a quality work product that meets the needs of the customer
- Demand the best from designers and developers

PLANNING ACTIVITIES INCLUDE:

- Lessons learned from previous programs and projects
- Cost/schedule estimates and plans for staffing
- Definitions for software and system requirement
- Requirements for safety and security
- Selection of software subcontractors
- Engineering documentation and historical data impacts
- Objectives for program/project
- Contract interpretation of necessary requirements

There are **three critical factors** in many software design/ development programs and projects. The **scope, schedule, and budget** combined affect the quality of work products.

PLANNED SCHEDULES

- **Purpose:** defines tasks and processes to be conducted for implementation
- **Importance:** planned schedules affect team capabilities for risk assessment, configuration control, and quality.

CRITICAL ITEMS IN A DEVELOPMENT PLAN

- Planned schedules
- Engineering information
- Software production direction
- A process consistent with system-level planning
- Consistency in agreement with the steps outlined

DEVELOPMENT PLANNING STEPS

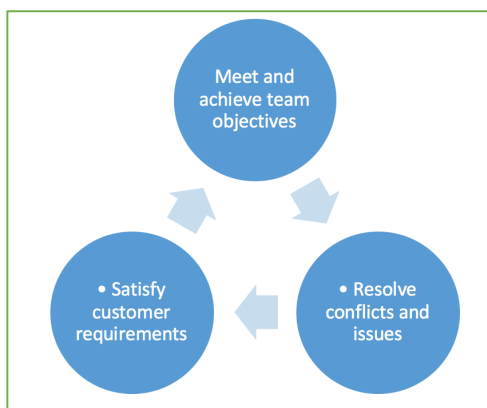
- Define entry and exit criteria for the software design/development
- Review and assessment of the work product/task requirements
- Define/update the process for each software activity
- Develop/update the estimating process
- Develop initial cost with schedule estimation and potential risks included
- Prepare detailed implementation plans

An **important element** in all software programs and projects is **teamwork**, the coordination and communication within teams applied to meet work expectations. **Managers do not control change but manage change.**

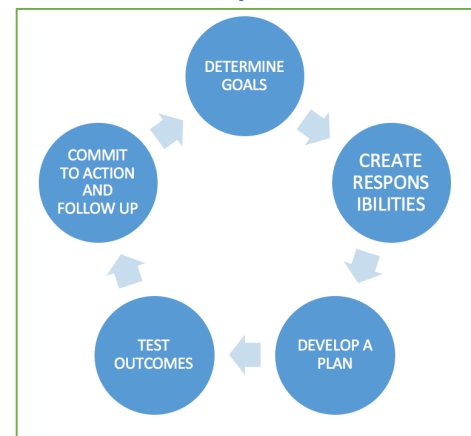
THE IMPORTANCE OF TEAMWORK

- Energy and consistency influence high-performance goals. Therefore, trust and cohesiveness must be maintained in the work environment.
- A plan is successful when a team delivers a high-quality work product, meets the defined schedule and maintains budget.

Team Responsibilities



Team Action Cycle



Chapter 3 (Second Step for Life Cycle)

The system/subsystem requirements reviewed by program and project personnel ensure accurate and complete understanding of the restrictions of systems design and applied work products. If program or project plans include reusable software interfaces; requirements are identified and evaluated for use. the term **reusable software** is commonly used in military and aerospace programs or projects.

SYSTEM DESIGN

Purpose: to ensure an accurate and complete understanding of the restrictions of a system or subsystem that affect work products.

- The external software interface is defined and verified for completeness.
- External interfaces based on software architecture definitions are identified as part of consequent software requirements.

The **systems engineering team** for programs and projects is responsible for the development of **software requirements** and **analyzes the system architecture** and **design** and **allocates system requirements**. The **main purpose** of the **SEP** is to address upgraded processes from a systems engineering point of view. This **plan is organized into three main sections: systems engineering, technical program processes, and engineering integration**.

SYSTEM ENGINEERING PLAN

Purpose: to develop software requirements, analyze the system architecture, design and allocates system requirements.

- A systems engineering plan (SEP) can be written to establish system-level technical reviews

MAJOR TECHNICAL REVIEWS & AUDITS AFFECTING SOFTWARE AND SYSTEMS INCLUDE: (44)

- Initial requirements (IR)
- Final design meeting (FDM)
- First-article inspection (FAI)
- Physical configuration audit (PCA)
- Incremental design review (IDR)
- Test readiness (TR)
- Functional configuration audit (FCA)

SYSTEMS ENGINEERING TEAM

Purpose: describes an orderly and structured approach to the overall system design, software design/development, required formal reviews, and audits.

❖ Importance of team:

- documents/provides the technical expertise to execute activities throughout a software design/development life cycle.
- enables performance to be more effective
- enables technical planners to spend more time planning to ensure the customer assurance and satisfaction in addressing the technical challenges

The purpose of **software architecture evaluations** is to provide a common approach to developing the work product architecture. this evaluation applies to the implementation of enhancements for change or corrections to existing software architectures.

SOFTWARE ARCHITECTURE EVALUATION

- Provides the viability of software architecture definitions to be applied
- Conflicts in requirements, architecture, or program and project plans are reported to product teams for resolution
- Program and project are analyzed to determine the impacts on architecture development.

Objectives: operational scenarios and system or subsystem requirements

Scope: does use interface requirements to analyze operational designs, software risks, and plans to determine the objectives of the architecture.

Development: identified during development and made available to be understood before beginning a software design/development life cycle.

EVALUATIONS PROVIDE

- Operational scenarios for revision
- Defined system and subsystem requirements for analysis
- Defined system/subsystem interfaces for analysis

THE SYSTEM/SUBSYSTEM ARCHITECTURE REQUIREMENTS DETERMINE IMPACTS

- Influences to quality
- Functional necessities for determination of the software architecture

Chapter 4 (Third Step for Life Cycle)

Defined **software requirements** provide programs and projects with a systematic approach to the development of software requirements provided by various ideas and solutions.

Software requirements establish the principals for software design and integration test activities for both software and systems integration. the generation and execution of software requirements are created as a standalone item or as an item embedded in higher-level assemblies.

DEFINITION OF SOFTWARE REQUIREMENTS

Stage One: review of the functional or performance requirements to identify the constraints on software. Defined into greater detail to define derived software requirements.

System Requirement: used to determine accuracy, completeness, and applicability of the requirements for product.

Tool Purpose: (dynamic object-oriented requirements system [DOORS], matrix worksheets, etc.) used to understand potential architectures and associated software requirements.

Work Product: the execution and the knowledge of what flows from the start of requirements analysis to verification and validation drives software requirements development.

ANALYSIS

Include: a step-by-step process to develop requirements for software work products.

Use: fulfillment of high-level user requirements, allocated system requirements, and ideas for system operational concepts.

Reports: produced as run procedures verified and validated to support test and evaluations.

USE CASE

Purpose: to describe a flow of operations for the performance of systems and software implementation.

- Defines the limitations/technical considerations established on target computers, work product execution strategy, and operating systems.

Includes: functionality, performance, maintenance, and support considerations, in addition to the work product's operational environment

- Incorporates boundaries and constraint

FUNCTION

Analysis Purpose: to guarantee an accurate and thorough understanding of software performance.

- An ongoing activity during the software requirements definition process.

ARCHITECTURE

Identified/ defined as part of derived software requirements.

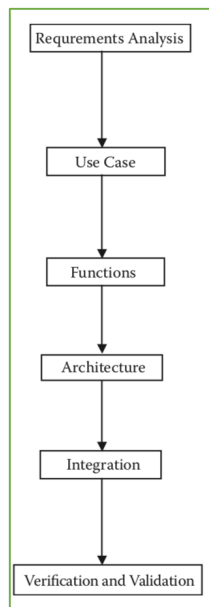
Function: software design/development life cycle states and modes are established.

Application: timing, sequence, conditions, and execution probability define and redefine functional interface requirements for system architectures.

INTEGRATION

Purpose: transforms functional architecture into optimal design solutions.

Planning Resource Component: critical for implementation of disciplined interface management principles is critical in order to perform systems build integration activities for the execution.



VERIFICATION AND VALIDATION

Verification and validation begins with the review of software requirements.

Requirement Priority Purpose: determines the extent of verification and validation for each defined requirement.

- Verification and validation is identified, a list of techniques that includes analysis is developed, inspections made, demonstrations run, and software integration tests conducted.

ACCOMPLISHED SYSTEMS VERIFICATION AND VALIDATION OF REQUIREMENTS

Performance: plan, evaluate, and record software work product compliance with established requirements.

Risk Reduction Assessment: ensures software design/development activities satisfy user needs in a manner that is efficient and cost-effective for integration of validation and requirement verification.

REQUIREMENTS DOCUMENTATION

Includes: software requirements and related information.

Use cases and derived software requirements are the source of each requirement.

Definition: documented based upon development plans, software processes, and product standards.

REQUIREMENTS TRACEABILITY

Documented according to developing planning, defined processes, and software work product instructions.

Purpose: enters information into the traceability system according to the program/ project for traceability standards.

- Traceable from system requirements or user requirements
- Clearly lead to software architectural component.

FORMAL REVIEW PREPARATION

Critical Requirement: defined and complete software requirements must be in place before formal review.

- **FCA Involvement:** reviews requirements connected to the release of software documentation and procedures that trace to hardware drawings configured in work products. (**Functional Configuration Audit = FCA**)
- Customer involved in both audits.
 - After customer satisfaction and audit approval the customer has the deliverable work product in possession.

MANAGING A REQUIREMENTS TOOL

Senior program and project managers should look for **software requirements tools** that meet the following:

- Have ability to impose multiple format requirements
- Support traceability and impact analysis
- Support software baselines and releases
- Alert modifications of requirements database

RELEASED SOFTWARE REQUIREMENTS

- aligns business goals and objectives
- reduces rework
- increases productivity
- ensures that requirements lead directly to program and project success and effective software delivery

Chapter 5 (Fourth Step for Life Cycle)

Software design is a consistent approach and method for the development of software requirements in defined designs of a work product. The software architecture definition provides a framework for the creation of the product design and at times can provide constrictions. The software design definition implements details about a software product's architecture, components, and interfaces.

DEVELOPMENT PLAN

The **development plan (DP)** for software is a documented and well-defined process useful for implementation and applicable standards. The **tasks** for the development of top-level software design architecture include the identification of major software functions (components), functional hierarchy diagrams, and hardware/software interfaces.

SOFTWARE DESIGN DECISIONS

Purpose: provides design concepts and decisions for a work product.

- Analysis and integration of software requirements definition and the software operational concepts identify the capabilities and characteristics required to make key design decisions.
- Software designer uses **software design tools** for requirements, code development, configuration management (CM), and software documentation.

SOFTWARE REQUIREMENTS EVALUATION

Purpose: defines software operational scenarios to ensure problems affecting software design are identified, evaluated, and resolved.

- A risk analysis using prototype software is performed to support early requirements evaluations and design feasibility.
- If requirement is proven unusable and not to be implemented for use, data from evaluations is fed back into the output of the software requirements development phase.

PEER REVIEWS (مرهه مهم)

Purpose: to find and correct as many errors as possible before test team integration or customers find problems.

- Starts with requirements, design models, and uninterrupted code and unit tests for the software designer.
- Applied at various stages during the software design/development life cycle
- Creates clean software work products and provides assurance that issues or errors are discovered and resolved.

EXAMPLE OF PEER REVIEW METHODS	PEER REVIEWS CRITERIA
<ul style="list-style-type: none"> • Inspections • Structured walk-throughs • Deliberate refactoring • Pair programming 	<ul style="list-style-type: none"> • Schedule the peer review at a convenient time • Assign reviewers • Prepare/update materials • Provide checklists • Introduce training materials • Select software work products • Provide entry and exit criteria

SOFTWARE DESIGN/DEVELOPMENT METHODS

Concurrent Software/Design Development

- **Requirement:** software design expertise to anticipate where the defined design is going.
- **Con:** possible to delay commitment until the last moment when failure to make a decision eliminates an important alternative or decision.

Lean Software Design/Development

- **Objective:** to move as many changes as possible from the top curve to the bottom curve.
- Delays the freezing of all design decisions as long as possible.
- Emphasizes designing and managing changes throughout the life cycle.
- Provides a better understanding of software engineering and quick delivery to customers.

AGILE SOFTWARE PROCESSES

- Provides fewer defects.
- Supports numerous initiatives
- Provides a program and project with a manager's approach to emphasize short-term program and project planning.
- Adopts values that are consistently depicts processes and makes decisions that may reject a software design.
- More effective than the traditional models due to perfection versus good-enough concepts for software design practices.
- Provides capability to understand information first before jumping into software design and development.

FOUR KEY ELEMENTS OF AGILE SOFTWARE ENGINEERING

1. The team has control of work assignments
2. Communication with team members and customers is needed
3. Change is good: "Think outside the box"
4. Customer satisfaction and expectations are achieved

CONFIGURATION MANAGEMENT (CM)

CM methods are a supporting discipline not directly involved in creating executable code. In the Agile process, CM methods are not referenced for specific routines.

CM Purpose: trim process and provide more automation in tools

- Brings back focus to configuration control objectives.
- Software tools common to other team members are adapted to the process
- ❖ When Agile processes lack configuration control, Lean principles are a waste of time and lead to chaos. As a result, there is no progress in software design/development.
- ❖ The ability to control change is the foundation of design/development activities. CM methods should not limit change or it will become a barrier for program and project plans.

SOFTWARE STANDARDS

Purpose: ensures development processes are in accordance with identified process models.

Minimum software standards consist of the following:

- Documented and maintained plans and procedures
- Peer reviews
- Standard software tools

CAPABILITY MATURITY MODEL INTEGRATION (CMMI)

Purpose: provides opportunity to address software design/development with support to customers after delivery.

- Provides a systematic approach to software engineering tasks in programs and projects.
- Enhance the knowledge base for software designers.
- Provides content for performance during the software life cycle.

CMMI SOFTWARE ENGINEERING TASKS

- Identify internal and external interfaces
- Establish infrastructure abilities with software design
- Develop plans, processes, and procedures
- Reuse capabilities for identified software

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.

LEAN SIX SIGMA

Purpose: reduces process variation, resulting in fewer errors and defects.

Goal: Zero defects

Six Sigma Process:		LEAN GOAL: ELIMINATE EIGHT WASTES	
• Define	• Measure	1. Defects	2. Overproduction
• Analyze	• Improve	3. Waiting	4. Nonutilized talent
		5. Transportation	6. Inventory
		7. Motion	8. Excess processing

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.

SOFTWARE DESIGN /DEVELOPMENT

Purpose: secure databases related to software.

- Effective software and systems integration methods create profit from inside the software design/development sector.

تم الحمد لله ..

الملخص يشمل السلايدات و الكتاب ، المربعات الي باللون الأخضر تحتوي على نقاط اكثر وضوحاً في الكتاب و اضيفتها هنا.

كل الشكر و التقدير لهما العتيبي ♥ ..

و دعواتكم .