SE Life Cycle Models

Lecture 45, v03

John G. Artus

BSEE

MSSE

INCOSE ESEP

© Copyright 2023-2024 John G. Artus

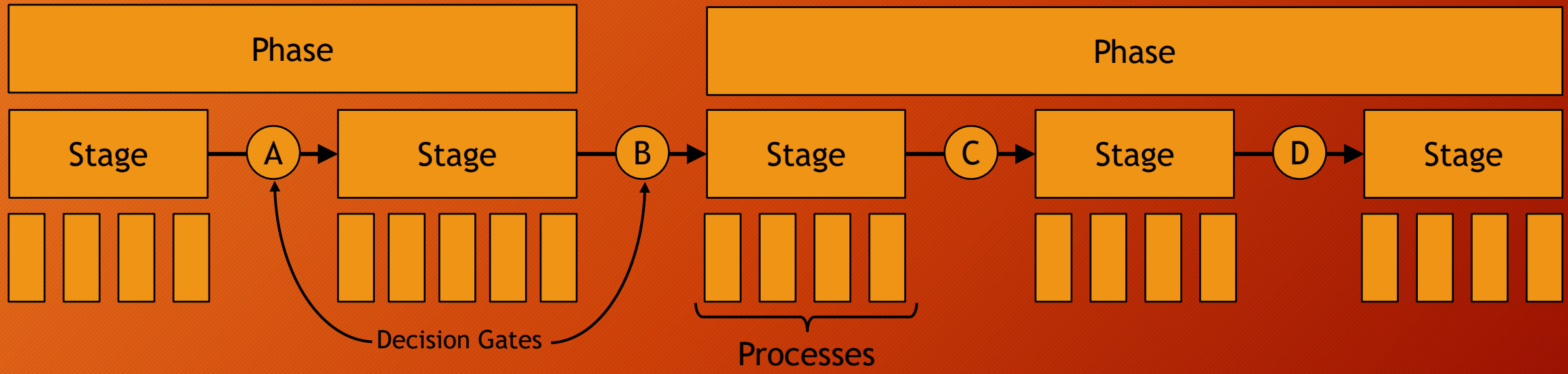
About This Courseware

- The majority of the material presented in this course is sourced from the textbook “Visualizing Project Management” by Kevin Forsberg, Hal Mooz, and Howard Cotterman
- I, John Artus, make no claim of ownership of the material sourced from this textbook
- I, John Artus, am using the material sourced from this textbook, and other indicated sources, as content for this courseware for educational purposes only
- This courseware lecture material has been sourced, interpreted, assembled, formatted, and copyrighted by John G. Artus for use in this educational context
- Anyone may freely access, and reuse this material in an educational context provided the copyright owner, John G. Artus, is recognized as the interpreter, assembler, and formatter of the source material used in the generation of this courseware, and provided that Kevin Forsberg, Hal Mooz, and Howard Cotterman are recognized as authors of the textbook “Visualizing Project Management” from which the majority of the content of this courseware has been sourced

© Copyright 2023-2024 John G. Artus

What is a Life Cycle Model?

- A life cycle model is used in engineering to describe the complete life of an instance of a System-of-Interest (SoI)
  - It consists of a set of phases, stages, and processes that the SoI goes through, from its inception to its retirement and disposal



- The stages are terminated by decision gates, where the key stakeholders decide whether to
  - Proceed to the next stage
  - Proceed, but address open actions from the previous stage while in the new stage
  - Remain in the current stage until ready to proceed
  - Terminate the project or re-scope the project

© Copyright 2023-2024 John G. Artus

Life Cycle Model Terminology

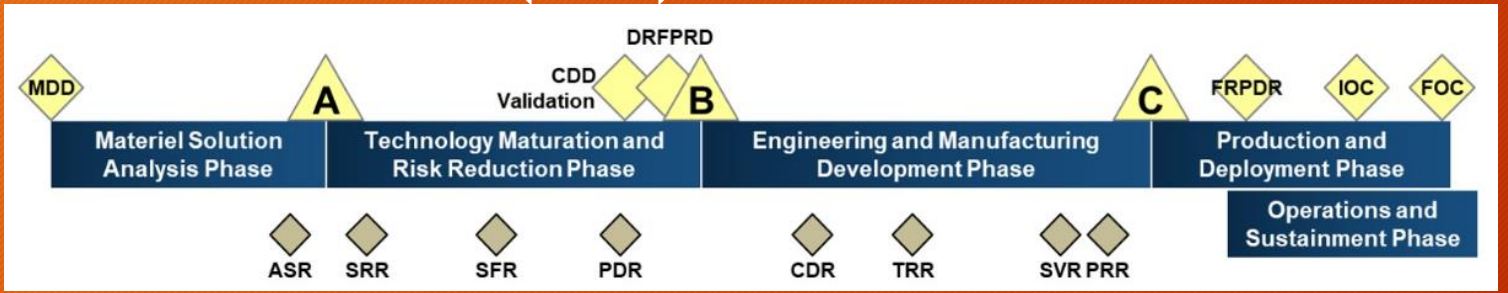
- Different organizations use different terms
- But in general, life cycle model terms are:
  - Phases – Represents major periods of performance on a program
  - Stages – Minor periods of performance within a program phase
  - Processes - A series of technical processes performed within stages, during which technical and management activities are performed
    - The organization must establish the work that is to be performed during each process
    - The organization must establish which artifacts are required to do the work in each phase and which artifacts will be produced as deliverables

© Copyright 2023-2024 John G. Artus

Examples of Specific Life Cycle Models

- US Department of Defense (DoD)

https://aaf.dau.edu/aaf/mca/tech-reviews/

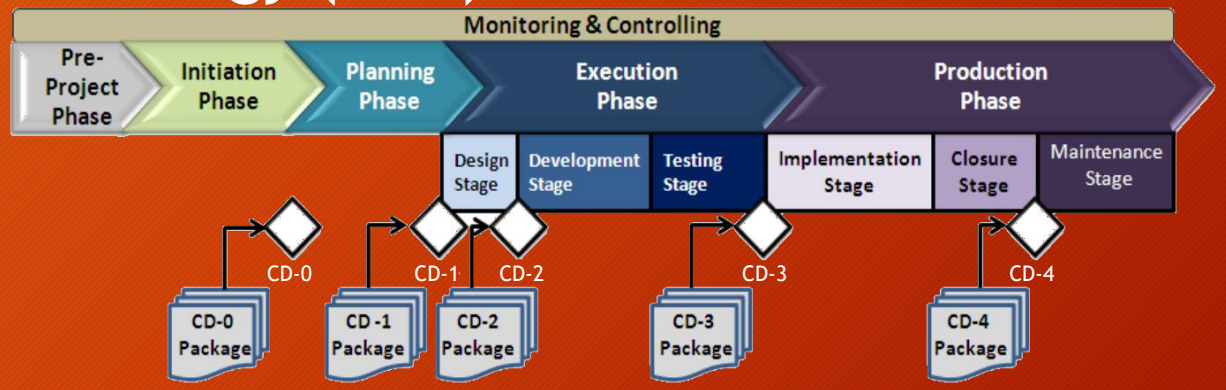


- US National Aeronautics and Space Administration (NASA)

https://www.nasa.gov/seh/3-project-life-cycle

NASA Life-Cycle Phases	Approval for Formulation — FORMULATION — Approval for Implementation				IMPLEMENTATION		
Project Life-Cycle Phases	Pre-Phase A: Concept Studies	Phase A: Concept and Technology Development	Phase B: Preliminary Design and Technology Completion	Phase C: Final Design and Fabrication	Phase D: System Assembly, Integration & Test, Launch & Checkout	Phase E: Operations and Sustainment	Phase F: Closeout
Project Life-Cycle Gates, Documents, and Major Events	KDP A, FAD, FA, Preliminary Project Requirements	KDP B, Preliminary Project Plan	KDP C, Baseline Project Plan	KDP D	KDP E, Launch	KDP F, End of Mission	Final Archival of Data

- US Department of Energy (DOE)



https://www.directives.doe.gov/directives-documents/400-series/0415.1-EGuide-1/@@images/file

It can be seen here that different organizations employ somewhat different life cycle models

But in the final analysis, they all accomplish similar objectives

The models are different to accommodate the specific types of projects worked and the interests of specific stakeholders

© Copyright 2023-2024 John G. Artus

Typical Phases of the SE Life Cycle

- Feasibility or Study Phase
  - Stakeholder requirements and system requirements are identified, viable solutions are identified and studied, and virtual prototypes can be implemented
  - The feasibility of alternative concepts reaching a second decision gate before initiating the execution stage is studied
- Execution Phase
  - The execution phase includes activities related to four stages of the system life cycle: development, production, utilization, and support
  - Typically, there are two decision gates and two milestones associated with execution activities
    - The first milestone provides the opportunity for management to review the plans for execution before giving the go-ahead
    - The second milestone provides the opportunity to review progress before the decision is made to initiate production
  - The decision gates during execution can be used to determine whether to produce the developed SoI and whether to improve it or retire it

© Copyright 2023-2024 John G. Artus

Typical Stages of the SE Life Cycle (continued)

- Concept Stage
  - Define the user (and stakeholder) requirements and constraints
  - Establish the feasibility of meeting the user requirements, including technology readiness assessment
  - Stakeholder needs and requirements are revisited as new information becomes available
- Development Stage
  - The selected concept(s) identified in the concept stage are elaborated in detail down to the lowest level to produce the solution that meets the stakeholder requirements
  - Continue with user involvement through in-process validation (the right-side upward arrow on the Vee model)
- Production Stage
  - The SoI is built or manufactured
  - Product modifications may be required to resolve production problems, to reduce production costs, or to enhance product or SoI capabilities
  - Any of these modifications may influence system requirements and may require system re-qualification, re-verification, or re-validation
  - All such changes require SE assessment before changes are approved

© Copyright 2023-2024 John G. Artus

Typical Stages of the SE Life Cycle (continued)

- Utilization Stage
  - Provide supporting systems which help sustain operation of the product
  - These supporting systems should be seen as system assets that, when needed, are activated in response to a situation that has emerged in respect to the operation of the SoI
  - The collective name for the set of supporting systems is the Integrated Logistics Support (ILS) system
- Support Stage
  - The SoI is provided services that enable continued operation
  - Modifications may be proposed to resolve supportability problems, to reduce operational costs, or to extend the life of a system
  - These changes require SE assessment to avoid loss of system capabilities while under operation
- Retirement Stage
  - The SoI and its related services are removed from operation
  - Planning for disposal is part of the system definition during the concept stage
  - SE focus is on ensuring that disposal requirements are satisfied

© Copyright 2023-2024 John G. Artus

Systems Engineering Technical Processes

- The INCOSE Handbook Identifies the Following 14 SE Technical Processes
  - Business or Mission Analysis Process
    - Define the business or mission problem or opportunity, characterize the solution space, and determine potential solution class(es) that could address a problem or take advantage of an opportunity
  - Stakeholder Needs and Requirements Process
    - Define the stakeholder requirements for a system that can provide the capabilities needed by users and other stakeholders in a defined environment
  - System Requirements Definition Process
    - Transform the stakeholder, user-oriented view of desired capabilities into a technical view of a solution that meets the operational needs of the user
  - Architecture Definition Process
    - Generate system architecture alternatives, to select one or more alternative(s) that frame stakeholder concerns and meet system requirements, and to express this in a set of consistent views
  - Design Definition Process
    - Provide sufficient detailed data and information about the system and its elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture
  - System Analysis Process
    - Provide a rigorous basis of data and information for technical understanding to aid decision-making across the life cycle
  - Implementation Process
    - Realize a specified system element

See Section 2.3.5 of the INCOSE Handbook v5 for further details of each of these technical processes

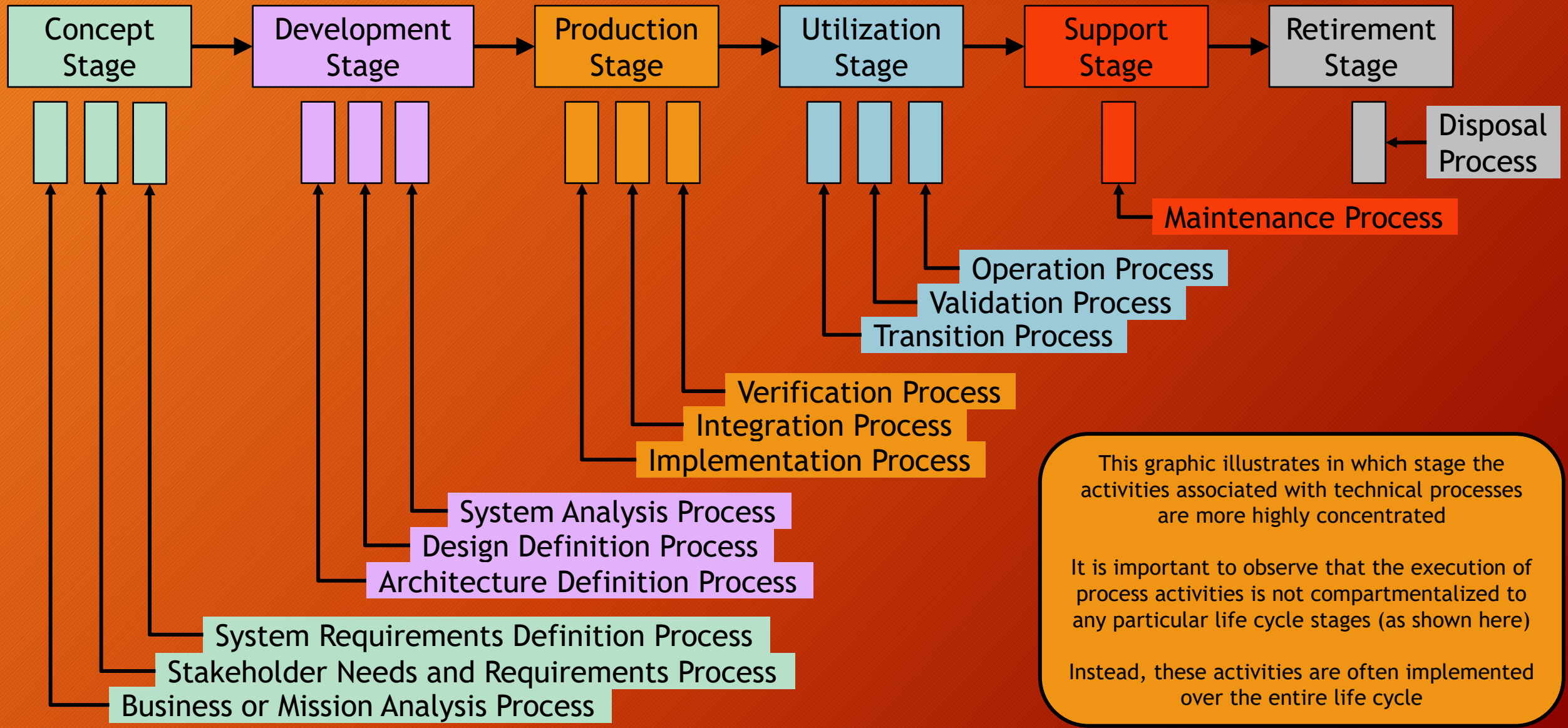
© Copyright 2023-2024 John G. Artus

Systems Engineering Technical Processes (continued)

- Integration Process
  - Synthesize a set of system elements into a realized system (product or service) that satisfies system requirements, architecture, and design
- Verification Process
  - Provide objective evidence that a system or system element fulfils its specified requirements and characteristics
- Transition Process
  - Establish a capability for a system to provide services specified by stakeholder requirements in the operational environment
- Validation Process
  - Provide objective evidence that the system, when in use, fulfills its business or mission objectives and stakeholder requirements, achieving its intended use in its intended operational environment
- Operation Process
  - Use the system to deliver its services
- Maintenance Process
  - Sustain the capability of the system to provide a service
- Disposal Process
  - End the existence of a system element or system for a specified intended use, appropriately handle replaced or retired elements, and properly attend to identified critical disposal needs

© Copyright 2023-2024 John G. Artus

Mapping the 14 SE Technical Processes to a Generic Life Cycle Model



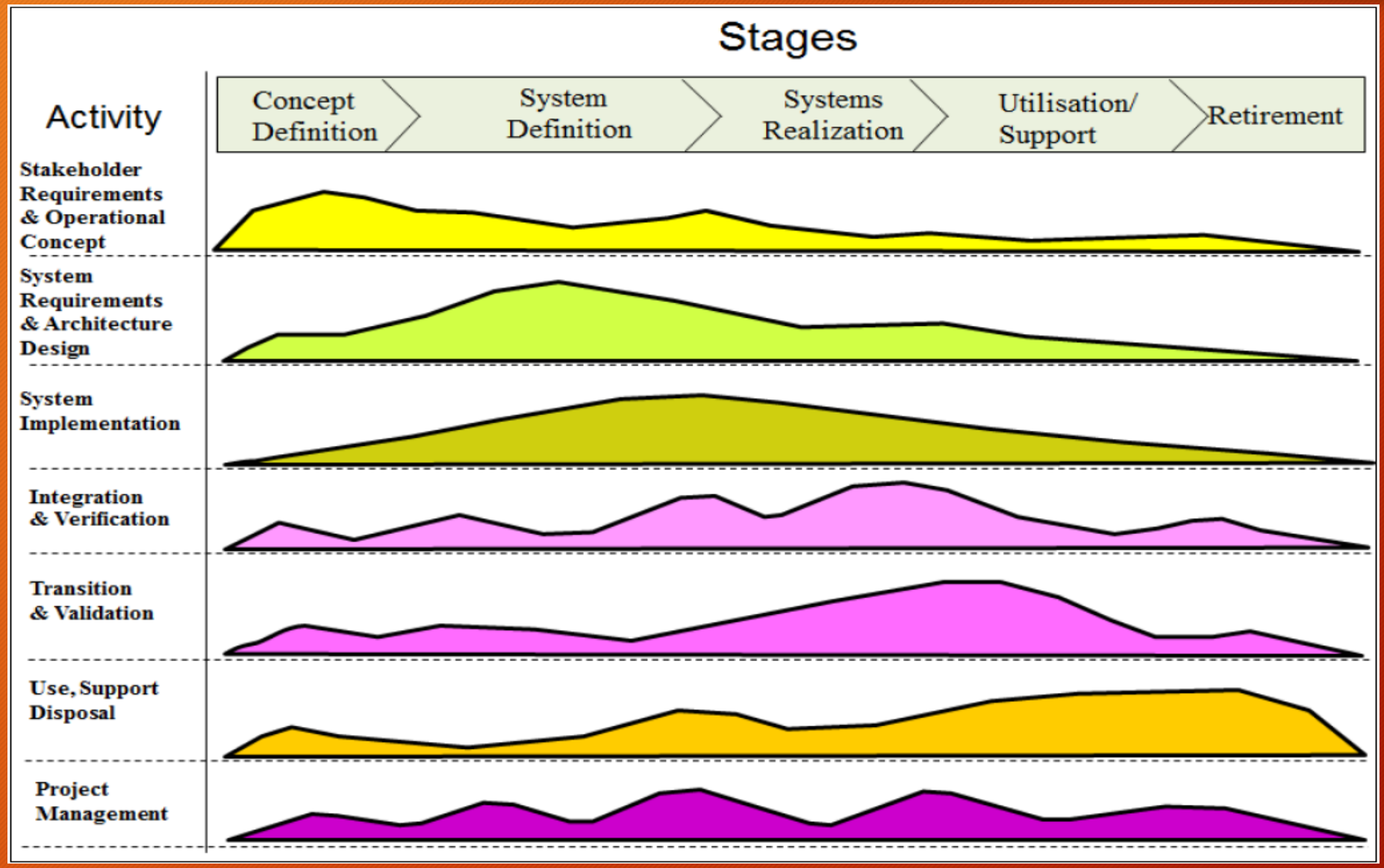
This graphic illustrates in which stage the activities associated with technical processes are more highly concentrated

It is important to observe that the execution of process activities is not compartmentalized to any particular life cycle stages (as shown here)

Instead, these activities are often implemented over the entire life cycle

© Copyright 2023-2024 John G. Artus

More Realistic Representation of Technical Process Activities



As shown, these activities may have peaks of effort at certain times, but are often implemented over the entire life cycle of the SoI

SEBok v2.8, page 285

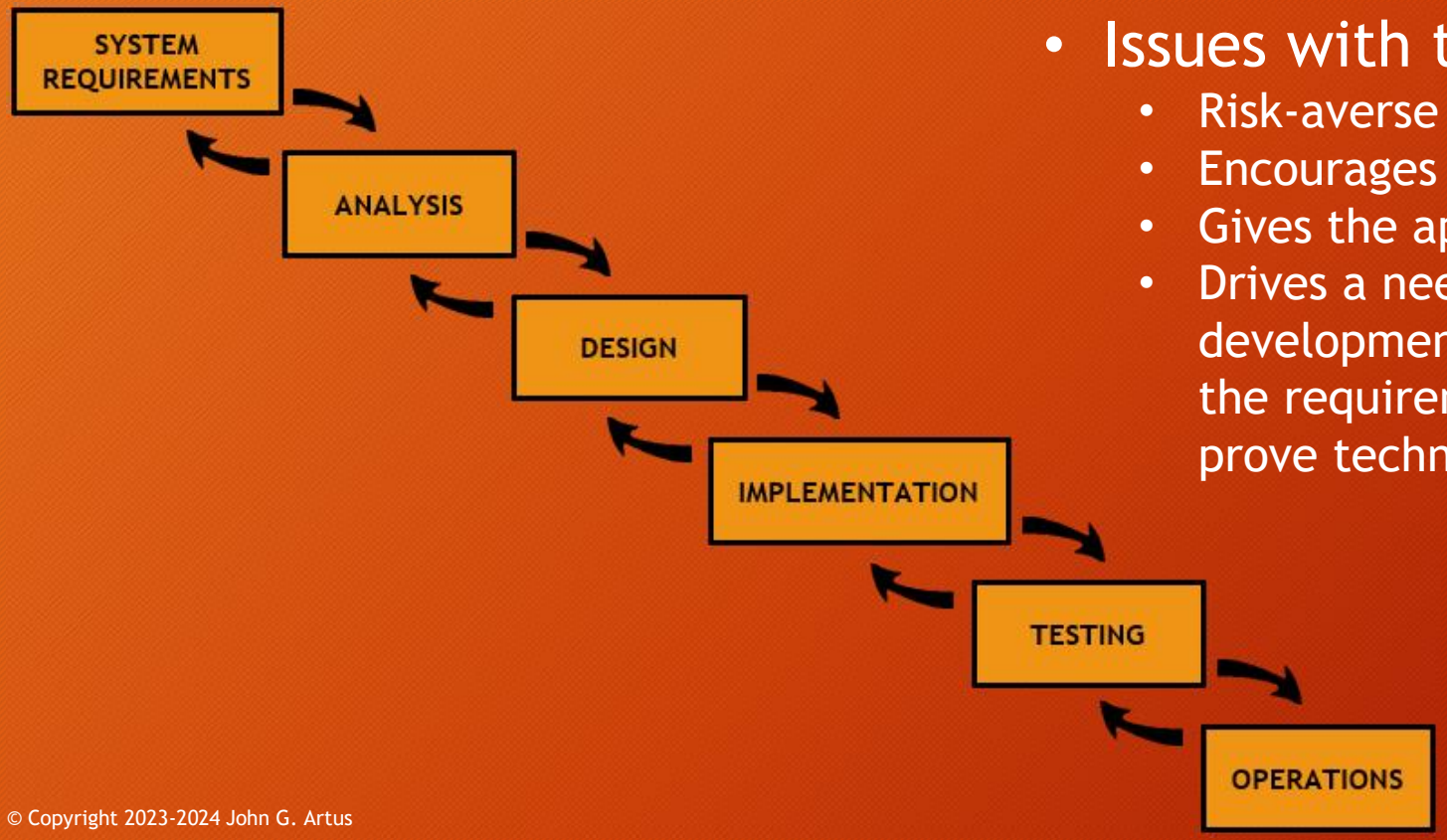
© Copyright 2023-2024 John G. Artus

Major Types of Life Cycle Models

© Copyright 2023-2024 John G. Artus

Waterfall Model

- The Waterfall Model depicts development activities as a series of steps progressing diagonally from upper left to lower right in discrete, sequential, linear phases
- It requires that work downstream should not begin until up-stream uncertainties are resolved and major reviews (decision gates) have been satisfied

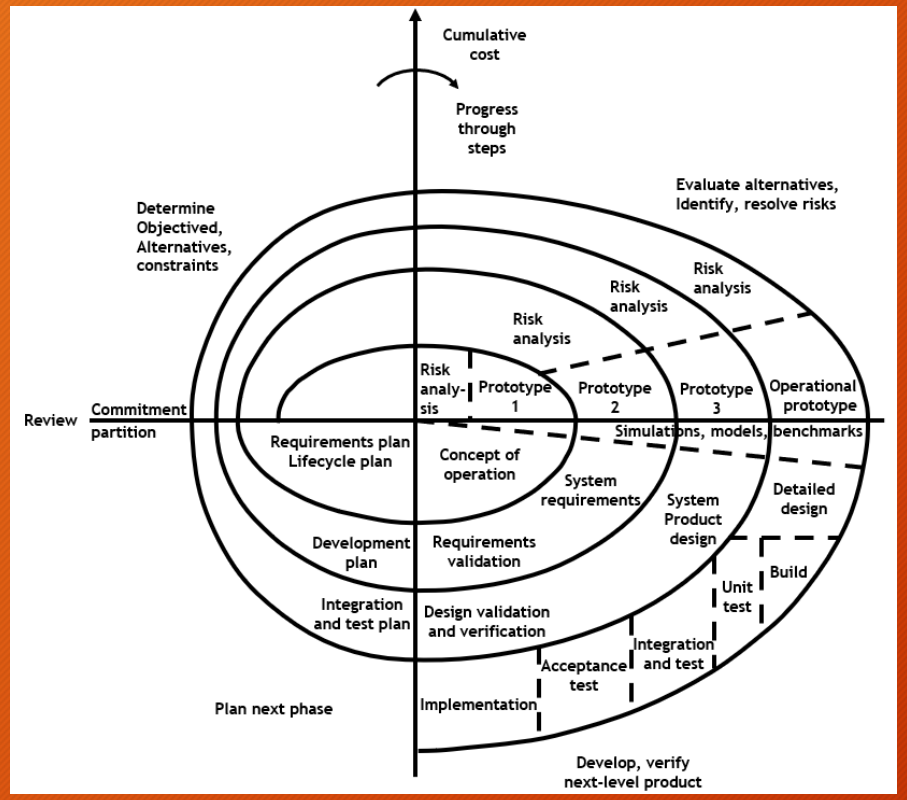


- Issues with the waterfall model
  - Risk-averse
  - Encourages unrealistic cost and schedule estimates
  - Gives the appearance of problem-free development
  - Drives a need to initiate system design earlier in the development cycle than appropriate to ensure that the requirements are properly understood and to prove technical feasibility

© Copyright 2023-2024 John G. Artus

Spiral Model

- An excellent risk-driven model that attempts to address the shortcomings of the Waterfall Model
- Addresses the need for early requirements understanding and feasibility modeling including operational scenario modeling
- The Spiral is another view of the technical aspect of the project cycle that emphasizes early risk analysis and system prototyping



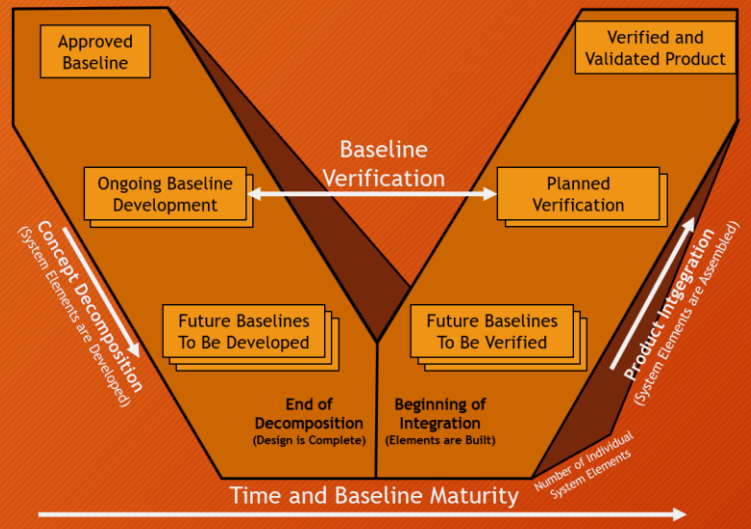
- Issues with the spiral model
  - The circular time representation is inconsistent with traditional left-to-right time representations
  - Risk management is portrayed as a sequence of serial analyses preceding and delaying low-risk product development rather than offering the option of performing risk management as an ongoing, parallel part of the development process
  - All risk management is shown to cease once the concept represented by the operational prototype is available, giving the impression that the following detail design and build-up will be risk free

© Copyright 2023-2024 John G. Artus

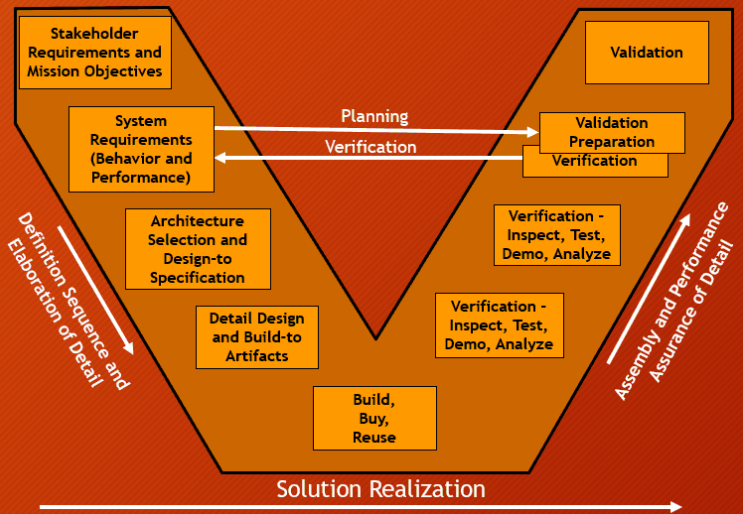
The SE Vee Life Cycle Models

- There exist many different interpretations of the SE Vee Model
- Three main interpretations that are addressed in this lecture are those based on
  - The expanding development and verification of product baselines
  - The sequential flow of SE Technical Processes that result in the development and verification of product baselines
  - A "Dual-Vee" that incorporates the features of the two above interpretations

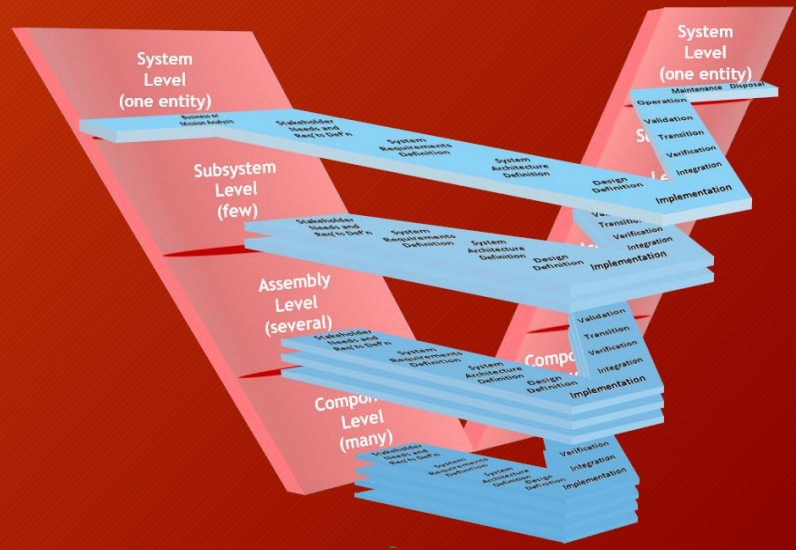
"Architecture Vee"
Based on Expanding Baselines



"Entity Vee"
Based on Sequential Flow of SE Technical Processes



"Dual Vee"
Incorporates the features of Architecture Vee and Entity Vee



© Copyright 2023-2024 John G. Artus

The Architecture Vee Model

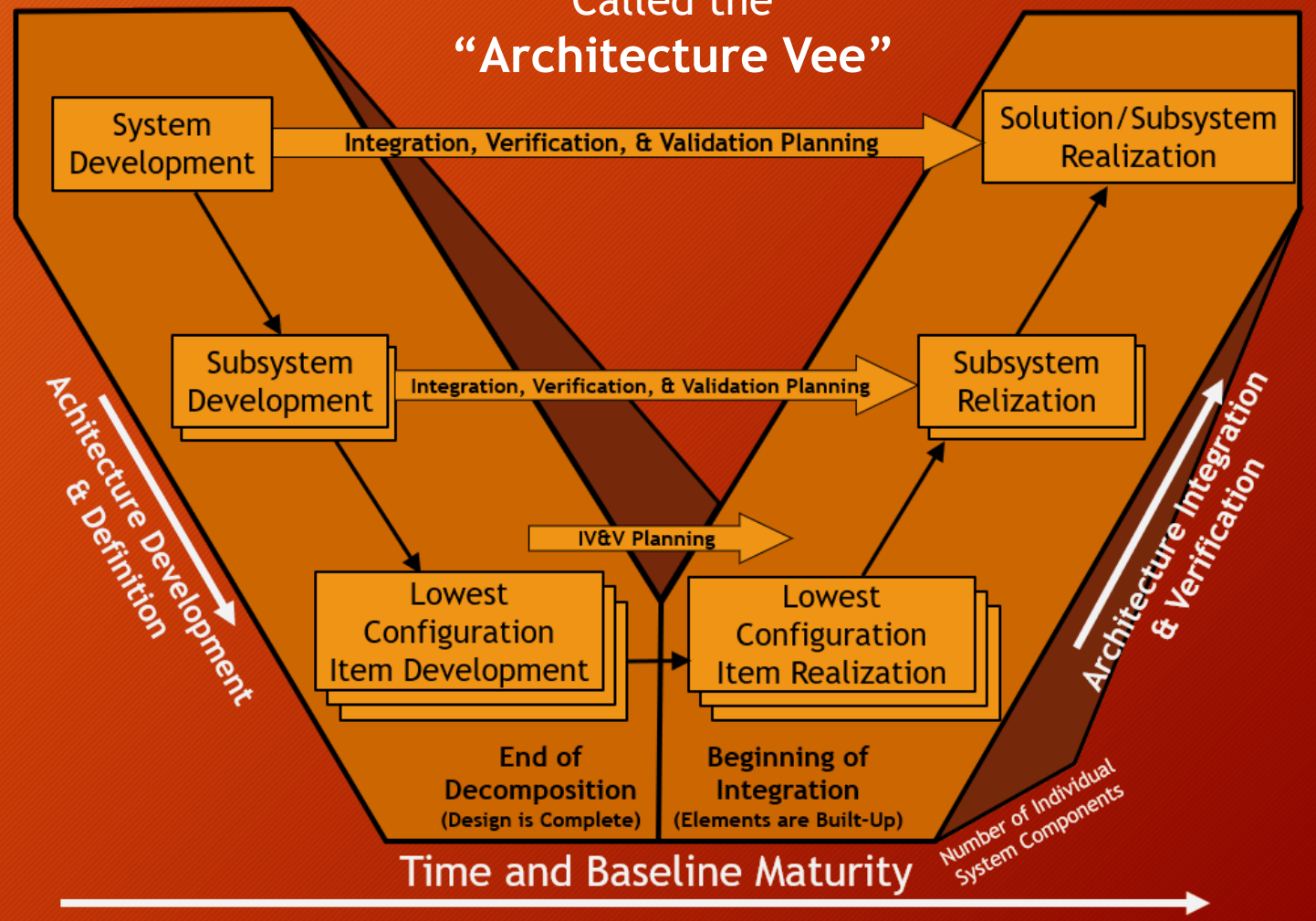
© Copyright 2023-2024 John G. Artus

Architecture Vee is Based on Expanding Baselines

- The Architecture Vee model is based on expanding development and verification of product baselines
- This is a primary example of a model based on pre-specified and sequential processes
- Its core involves a sequential progression of plans, specifications, and products that are baselined and put under configuration management

A baseline is a fixed reference in the development cycle or an agreed-upon specification of a product at a specific milestone in the project

Called the
"Architecture Vee"



In Systems Engineering, a component is given the technical term "configuration item" or CI

© Copyright 2023-2024 John G. Artus

Baselining of a Product Configuration

- A Configuration Baseline is a fixed reference in the development cycle or an agreed-upon specification of a product at a specific time which can only be changed through a change control process
- It aims to identify major changes and non-compliance to the performance of a configuration item throughout system development so that the Program Manager / Engineers can take appropriate action
- It consists of the performance documentation and standards that comprise a product at a certain moment during its development
  - When a certain program milestones is met, the product configuration at that point in time is preserved - this is called a Baseline
  - The milestone chosen could be any one of
    - Time (Schedule)
    - Maturity (Progress)
    - Budget (Termination of an activity based on expenditures)
    - Any combination or some other criteria
- The Baseline is formally examined and agreed upon at a given time and can only be amended via change control procedures
- Throughout the development lifecycle, the baseline is employed to measure, monitor, and manage changes

© Copyright 2023-2024 John G. Artus

Types of Configuration Baselines

- Functional Baseline
  - Describes the functional and interface characteristics of the system in a detailed functional performance specification
  - Describes the verification procedures to be performed to demonstrate the achievement of the specified functional characteristics
  - The functional baseline is normally established and put under configuration control at the System Functional Review (SFR)
  - Usually verified during a System Verification Review (SVR) and/or a Functional Configuration Audit (FCA)

"Configuration Item" is a formal term for system elements under configuration management and can include hardware items as well as software items

- Allocated Baseline
  - Describes the configuration items (CIs) that make up a system
  - Describes how system function, performance, and interface requirements are allocated across lower-level CIs
  - Describes the verification procedures required to demonstrate the traceability and achievement of the allocated requirements
  - Usually established at each configuration item's Preliminary Design Review (PDR), culminating in a system-allocated baseline established at the system-level PDR

"Allocation" refers to the assignment of responsibility to some configuration item for implementing a function or performance requirement

- Product Baseline
  - Describes the functional and physical characteristics of a configuration item
  - Describes the selected functional and physical characteristics designated for production acceptance testing
  - Describes the tests necessary for deployment/installation, operation, support, training, and disposal of the configuration item
  - The initial product baseline includes "build-to" specifications for hardware and software
  - Usually established at each configuration item's Critical Design Review (CDR), culminating in an initial system product baseline established at the system-level CDR

A "Build-To Spec" includes all the instructions needed to fabricate a part

© Copyright 2023-2024 John G. Artus

Technical Baselines

- Technical baselines describe product functions, performance, and interfaces
  - Technical baselines are formally controlled definitions of the characteristics of a system
  - Includes user requirements, program and product information, and related documentation for all configuration items
- The technical baselines consists of the Functional, Allocated, and Product Baselines
  - As development proceeds, the program establishes its functional baseline, allocated baseline, and product baseline
- Each of the technical baselines is placed under configuration management when they are established
  - Once a baseline is established, change becomes a formalized process which provides stability during design
  - Technical baselines enable the underlying design to progress using a common reference
- Management of technical baselines is typically part of the configuration management process

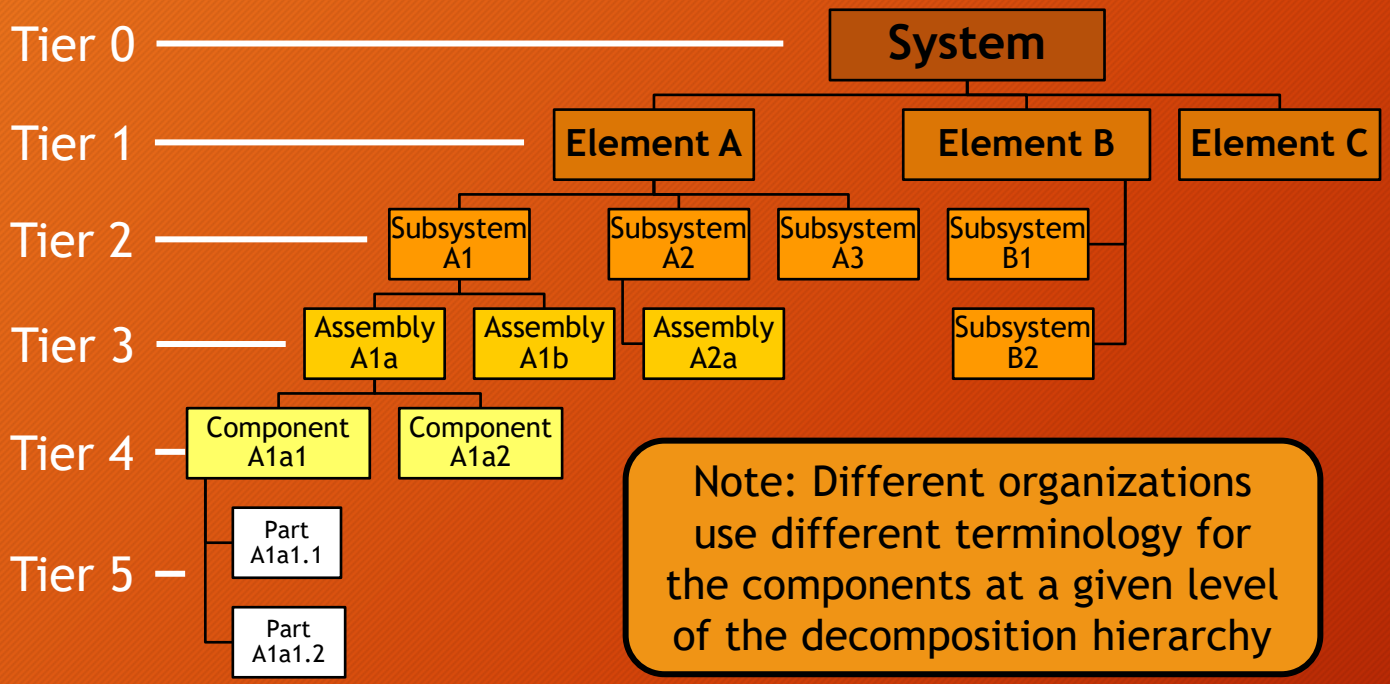
Major Documents in the Technical Baseline	Functional Baseline	Allocated Baseline	Product Baseline
System Performance Specifications	X		
Item Performance Specifications		X	
Item Details Specifications			X
External Interfaces Specifications / Interface Control Documents	X		
Internal Interfaces Specifications / Interface Control Documents		X	
Functional Architecture	X		
Physical Architecture		X	X
Technical Architecture			X

© Copyright 2023-2024 John G. Artus

Progress of Configuration Baselining

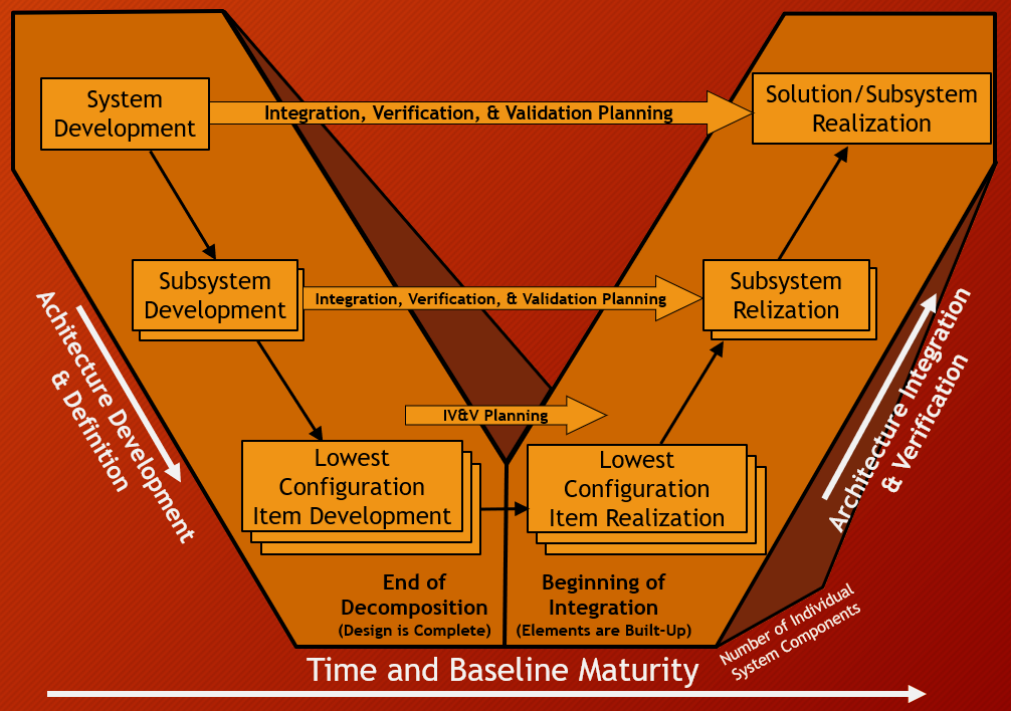
- As the system design progresses from conceptual to developmental to product, the system description is typically formalized through a series of increasingly detailed baselines
- Baselines are often established when the structural and functional decomposition of a system at a given tier level is complete
- Baselining continues as successive verification proceeds as the system components are integrated towards delivery of the final product

Example System Component Decomposition



Note: Different organizations use different terminology for the components at a given level of the decomposition hierarchy

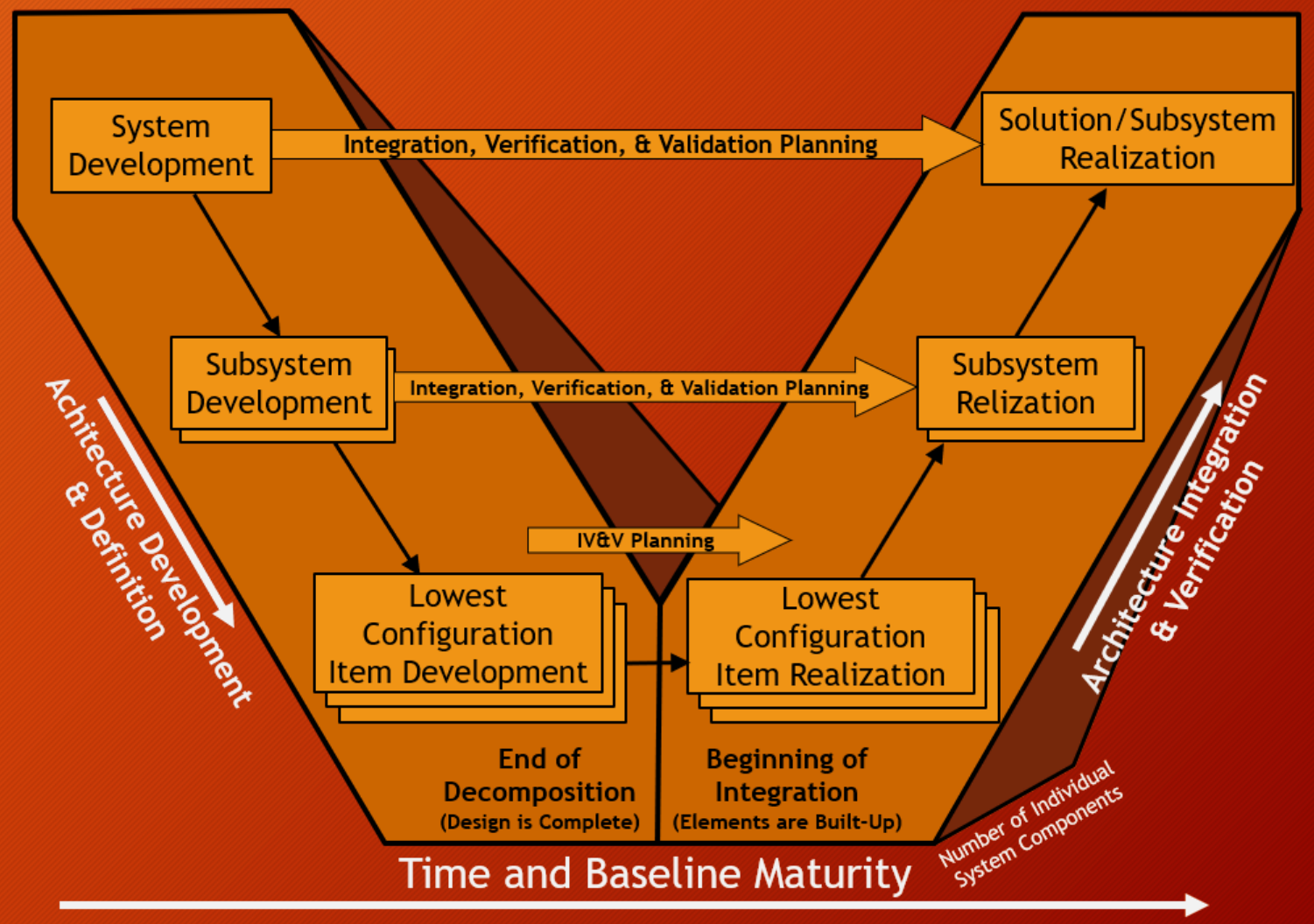
Architecture Vee



© Copyright 2023-2024 John G. Artus

Features of the Architecture Vee

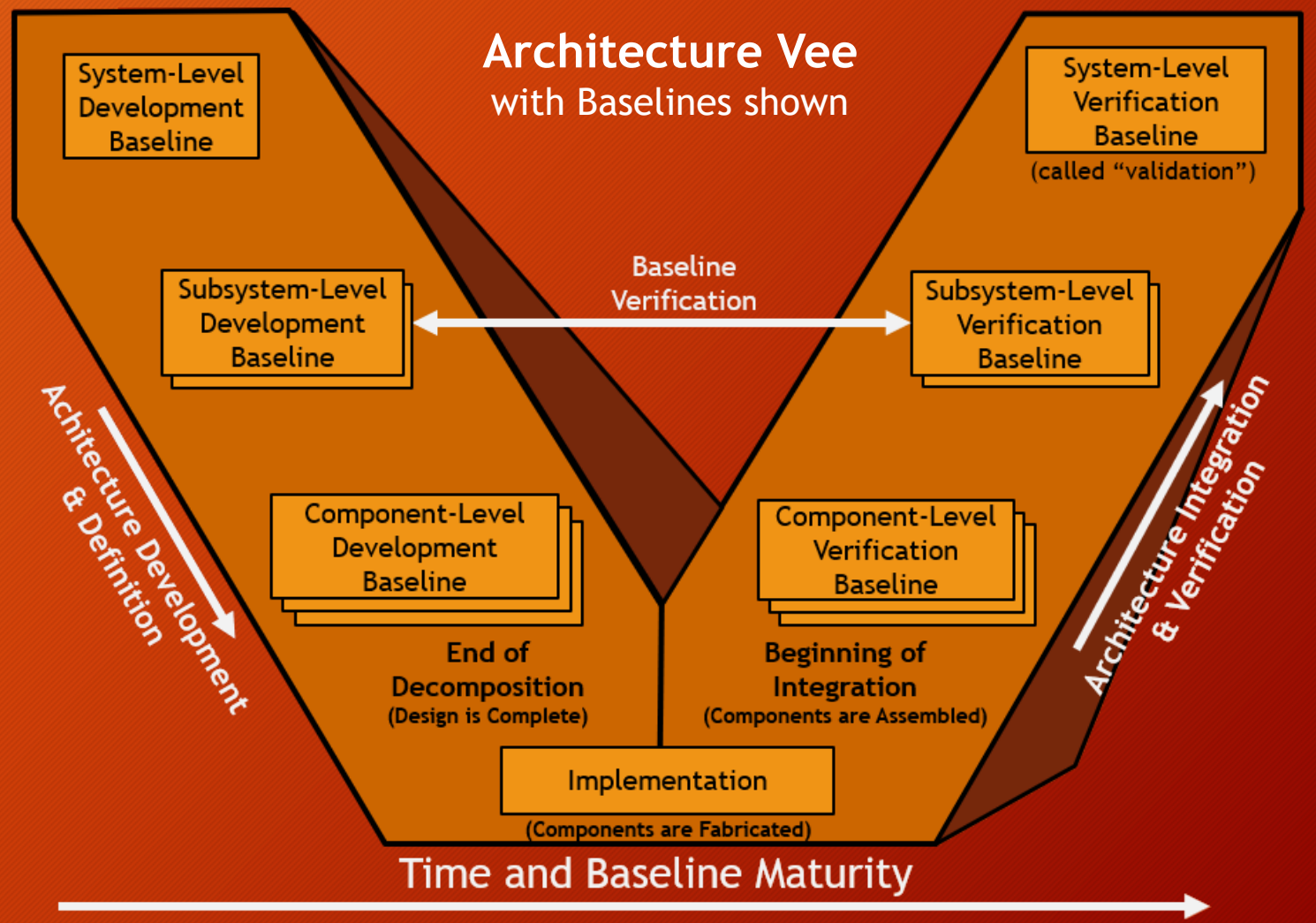
- The left side of this Vee represents the decomposition of the system at multiple levels, ending at the final level which represents the “Lowest Configuration Items” (LCIs)
- The Vee widens going from top to bottom to indicate the increasing number of defined components
- The right side of the Vee represents the integration (assembly) of components into larger assemblies until the final product is assembled at the top of the Vee
- The graphic is in the form of a “V” since the left side processes perform planning for the integration and verification activities that are performed by the right side processes



© Copyright 2023-2024 John G. Artus

Architecture Vee showing Baseline Development

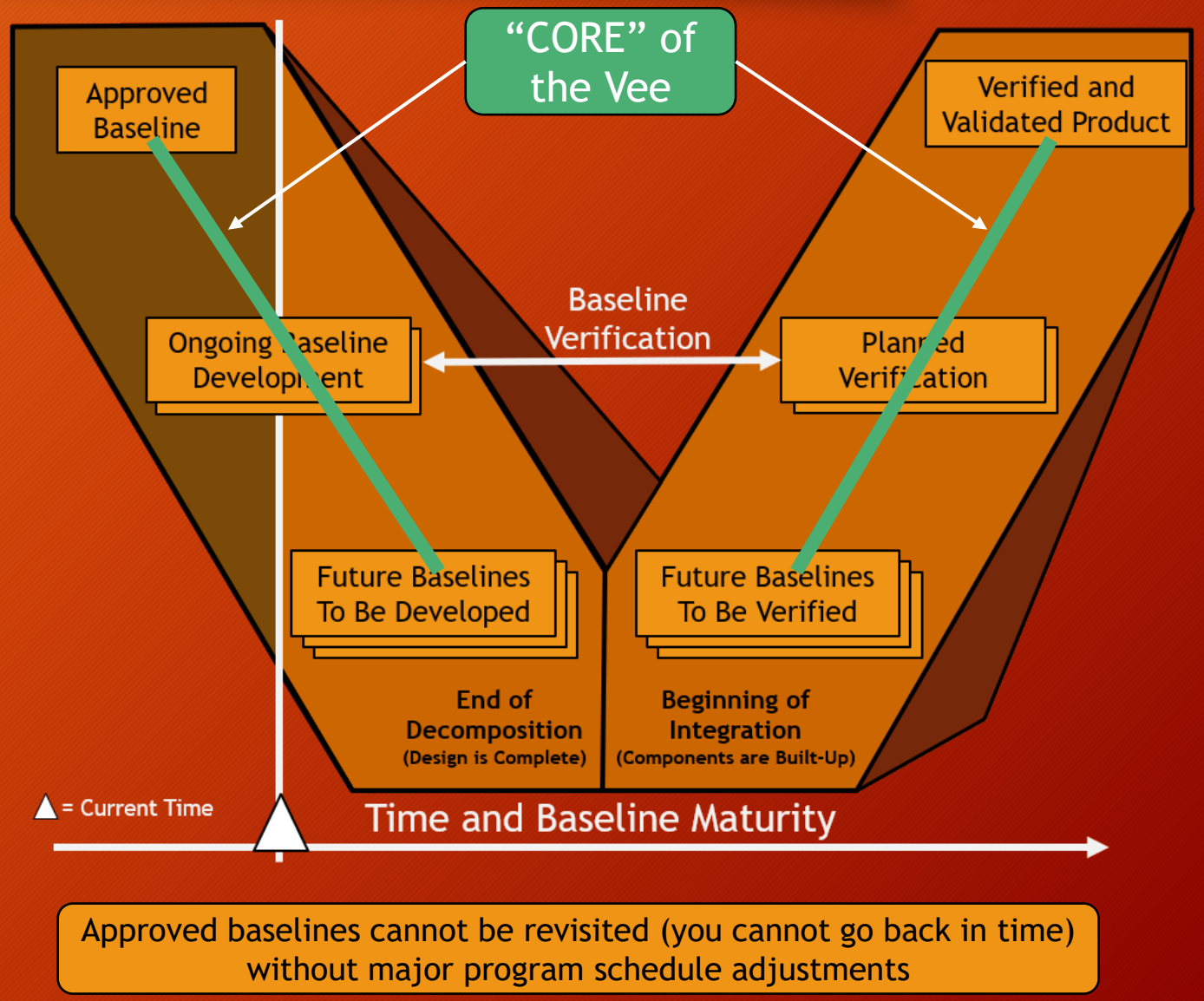
- This view of the Architecture Vee shows the ongoing development of product baselines
- In the figure, as you go down the left side of the vee, decomposition of the system proceeds, and developmental baselines are produced
- As you go up the right side of the vee, integration of the components proceeds, and verification baselines are produced
- At each vertical level of the vee, developmental baselines established on the left side of the vee eventually define and support verification activities on the right side of the vee
- When verification activities at a given vertical level of the vee on the right side are concluded, a verification baseline is established



© Copyright 2023-2024 John G. Artus

Architecture Vee Core Activities and Baseline Maturation

- As time progresses in the project, new baselines are approved and placed under configuration control
- In the figure, the vertical line represents the current state of the project in time
- At the time shown, decomposition of the system is continuing, representing the ongoing development of the current baseline
- Activities on the right side of the vee culminate with integration milestones that produce baselines as well
- Decomposition and Integration activities that produce baselines are considered to be “core” activities of the vee

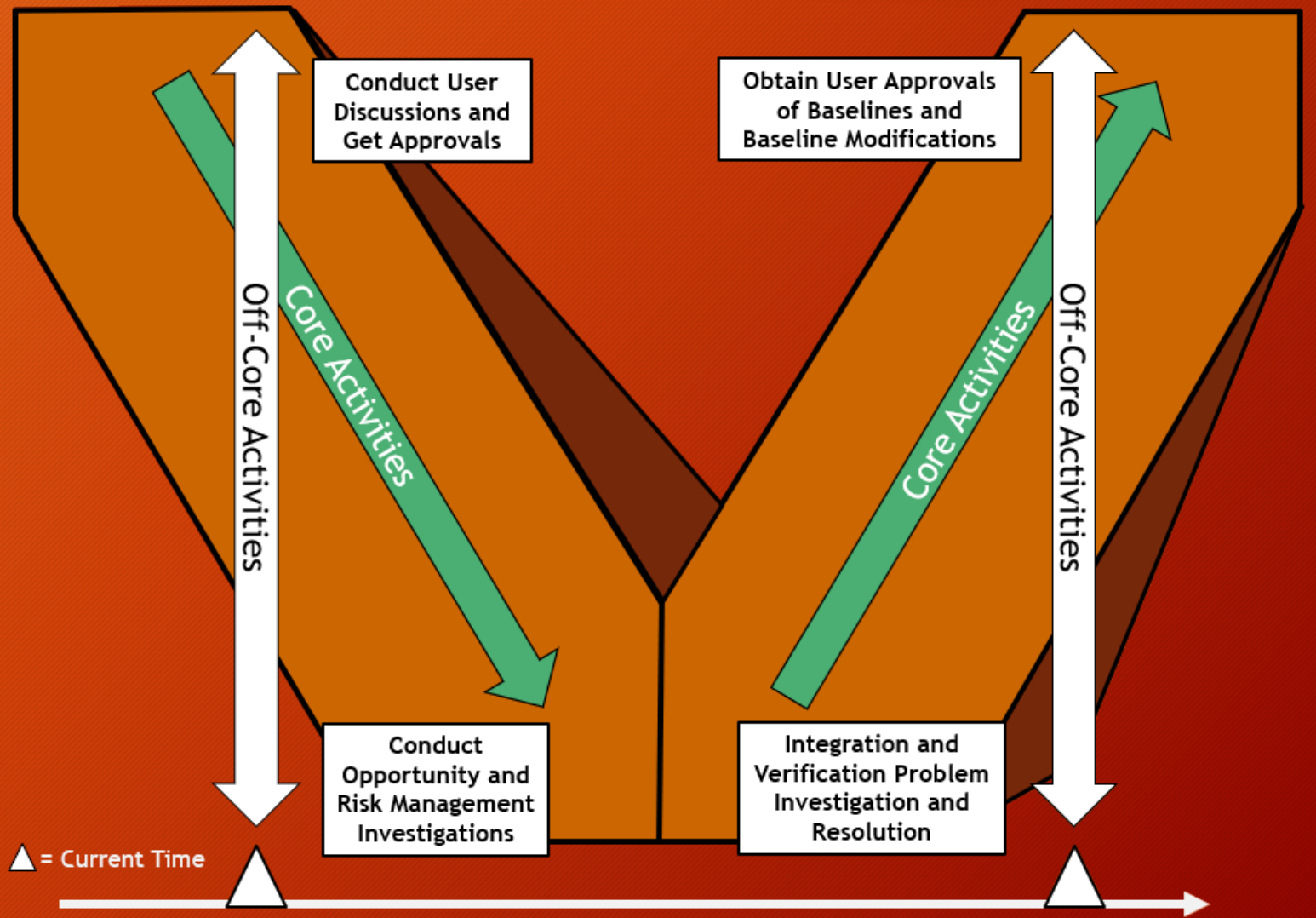


Approved baselines cannot be revisited (you cannot go back in time) without major program schedule adjustments

© Copyright 2023-2024 John G. Artus

Architecture Vee Off-Core Activities

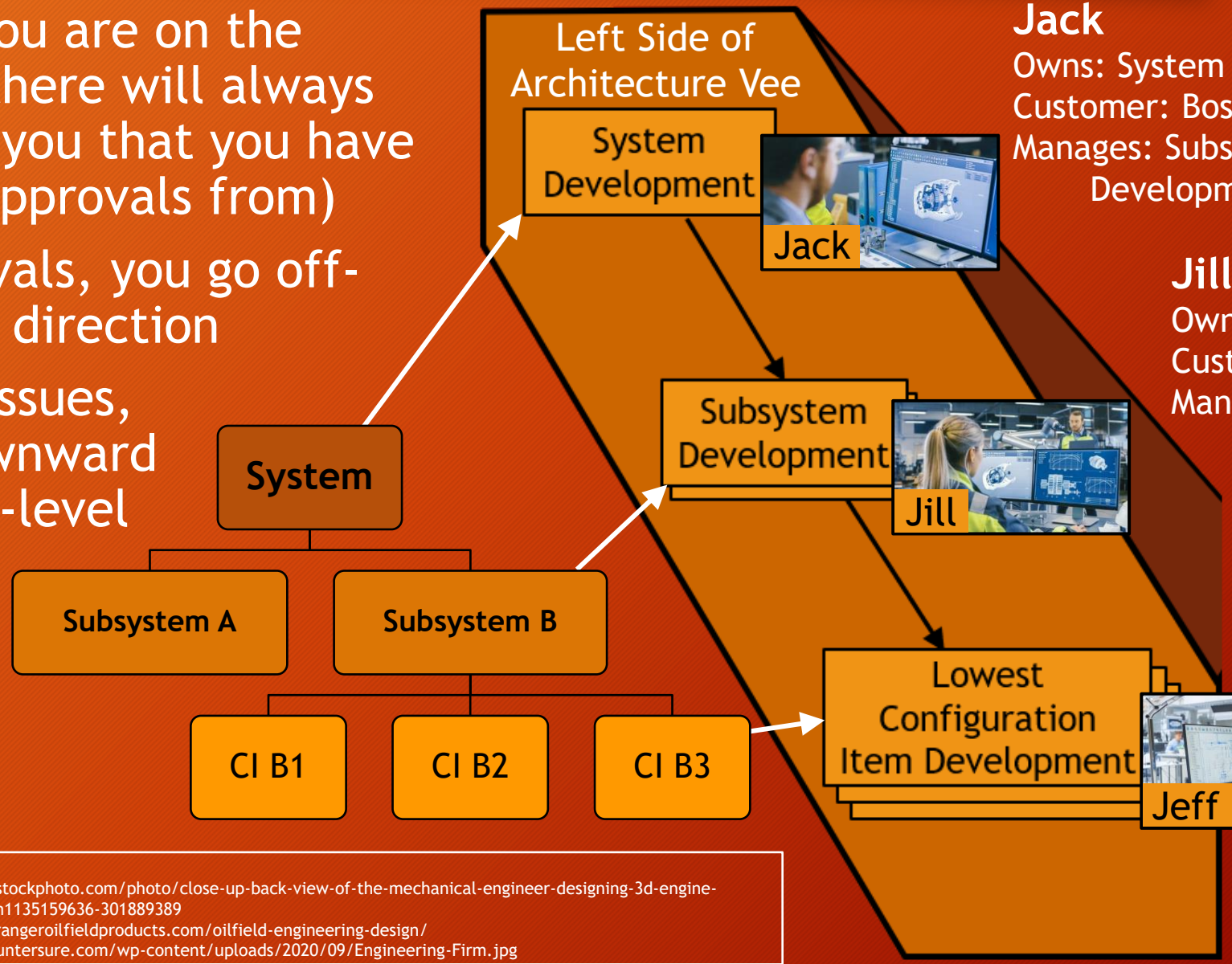
- In addition to the core activities performed during decomposition and integration, off-core activities can be conducted vertically at any time during baseline maturation
- Systems engineers traverse down the hierarchy to resolve unknowns by examining lower level details
- They can also traverse up the hierarchy to obtain user approals as necessary
- This enables projects to perform concurrent opportunity and risk analyses, as well as continuous in-process validation



© Copyright 2023-2024 John G. Artus

Everybody has a Customer

- No matter where you are on the Architecture Vee, there will always be someone above you that you have to answer to (get approvals from)
- To get those approvals, you go off-core in the upward direction
- To solve technical issues, you go off-core downward towards your lower-level developers



Jack
Owns: System
Customer: Boss or Stakeholder
Manages: Subsystems A and B Development

Jill
Owns: Subsystem B
Customer: Jack
Manages: CIs B1, B2, and B3

Jeff
Owns: CI B3
Customer: Jill
Manages: Nobody

Images:
Jack: https://www.istockphoto.com/photo/close-up-back-view-of-the-mechanical-engineer-designing-3d-engine-model-on-personal-gm1135159636-301889389
Jill: https://www.rangeroilfieldproducts.com/oilfield-engineering-design/
Jeff: https://www.huntersure.com/wp-content/uploads/2020/09/Engineering-Firm.jpg

© Copyright 2023-2024 John G. Artus

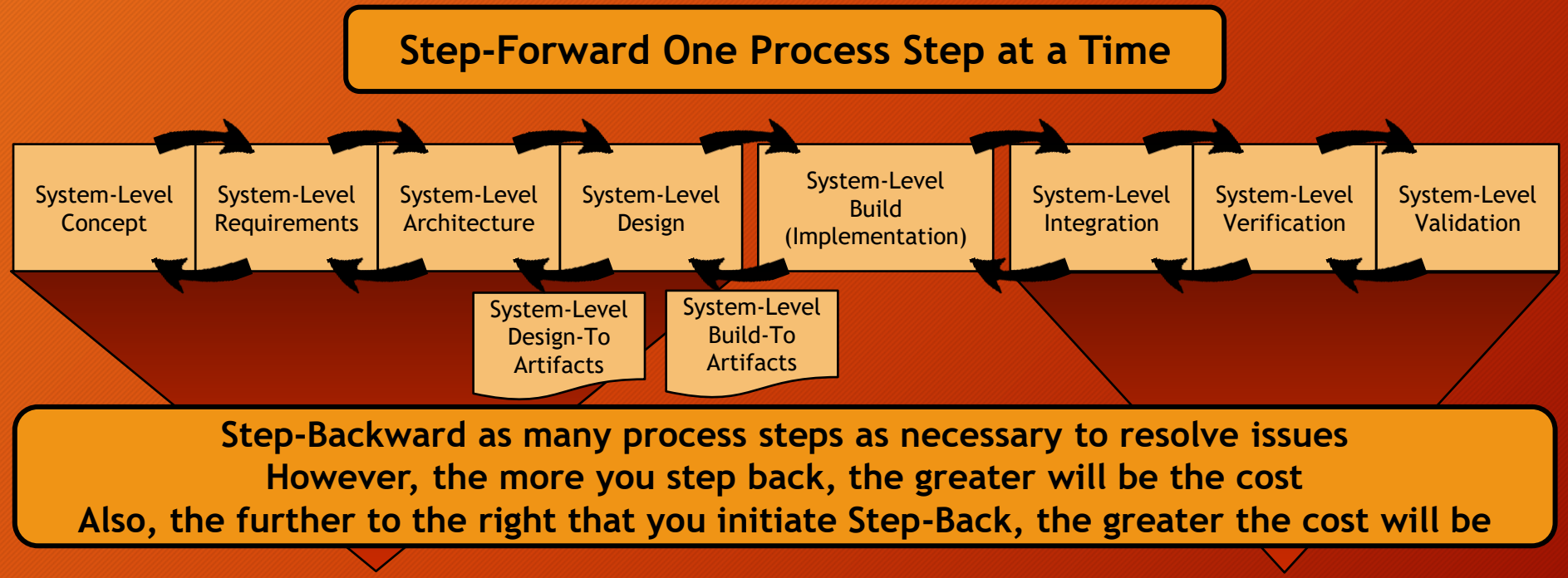
The Entity Vee Model

Called the Entity Vee because the SE Technical Process are applied to every Entity (every component) of the system at all levels of the system

© Copyright 2023-2024 John G. Artus

SE Activities at Each Level of Decomposition

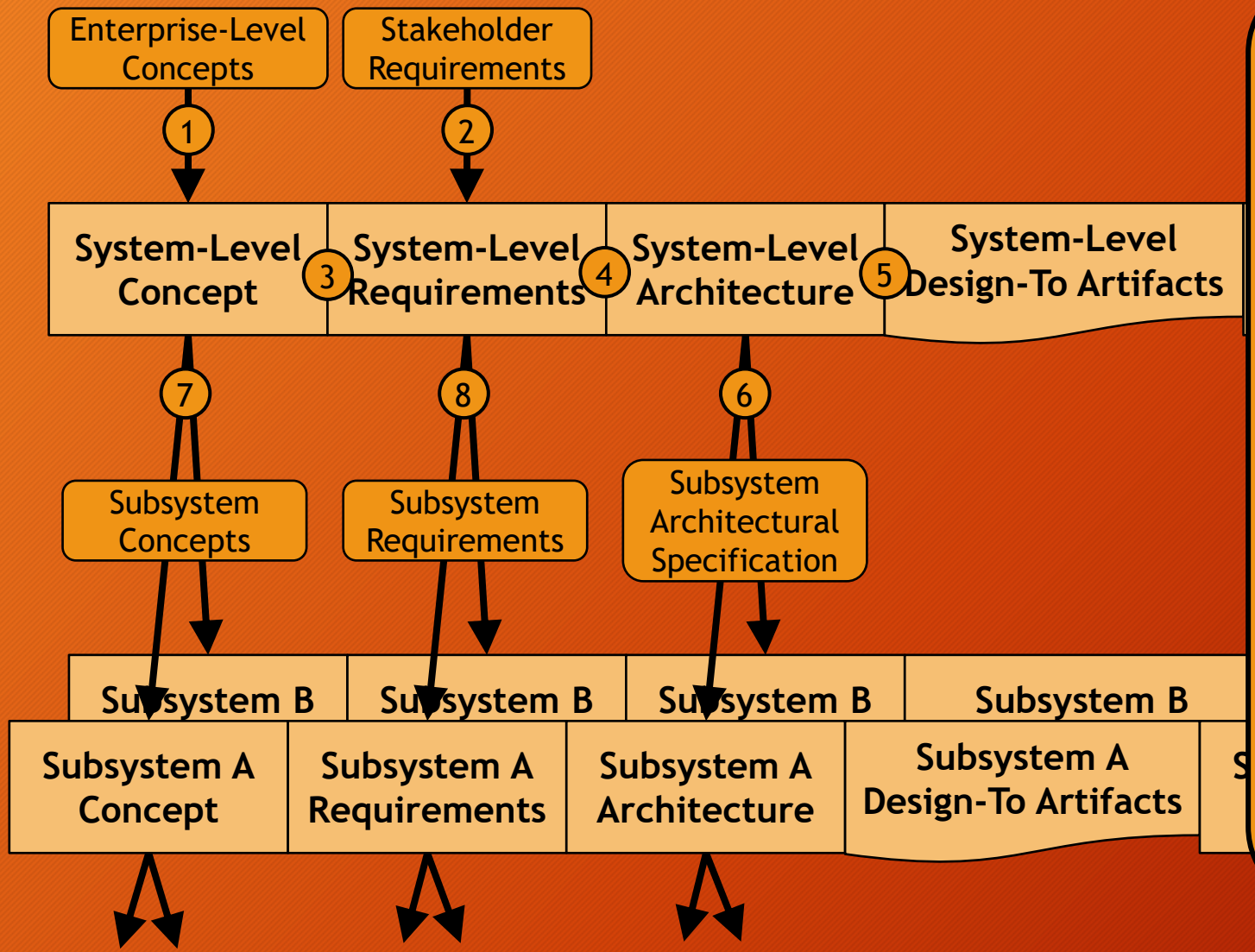
- To understand the essence of the Entity Vee, we need to focus on the SE activities that occur at each level of decomposition
- To develop a sound architectural solution, the SE Technical Processes are performed at each level of architecture decomposition and for each entity at that level
- At the first level (system level), the only entity is the system
- Nevertheless, the SE Technical Processes are performed for that one entity



To save space, not all 14 Technical Processes are shown

© Copyright 2023-2024 John G. Artus

How DAR is Implemented Among Levels (Part 1)



How DAR is Implemented Among Levels
(Part 1)

1. The Enterprise-Level concept is received to establish the context for the System-level concept
2. Stakeholder requirements are received and transformed into System Requirements
3. The System-level concepts serves as input for developing the System-level requirements
4. Together, the System-level concept and the System-level requirements serve as inputs to developing the System-level architecture
5. All three of these system-level artifacts form the basis for the System-level design artifacts which will establish the System-level Baseline at the PDR
6. The System-level architecture decomposes and establishes the structural elements that constitute the Subsystem-level entities (Subsystem A and Subsystem B)
7. The process defined in steps 1 thru 7 is repeated recursively, starting with the System-level concept being received to establish the context for the Subsystem-level concepts (A and B)
8. The System requirements are received and transformed into derived requirements for Subsystem A and Subsystem B

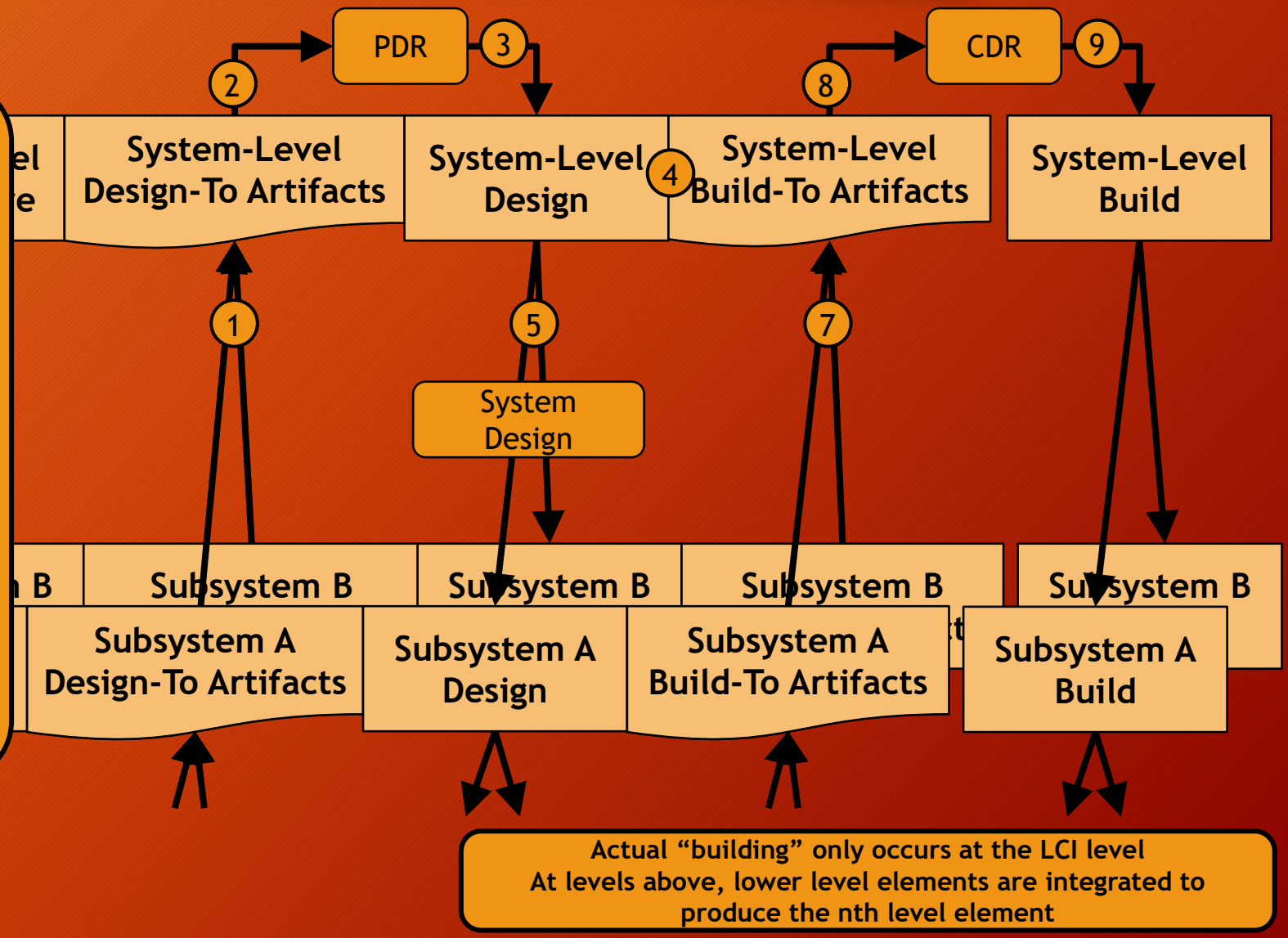
This process repeats for each entity at each subsequent lower level

© Copyright 2023-2024 John G. Artus

How DAR is Implemented Among Levels (Part 2)

How DAR is Implemented Among Levels
(Part 2)

1. All Design-To Artifacts from all levels are rolled up to form the Design-To Baseline
2. The Design-To Baseline is reviewed at the PDR
3. The successful PDR opens the gate to proceed with System-Level design
4. The product of the System Design activity is the System-level artifact that contributes to the Build-To Baseline at the CDR
5. The System-level design is received at the Subsystem-level to perform the Subsystem-level design
6. Steps 4 and 5 are repeated recursively for each entity at each subsequent lower level
7. All Build-To Artifacts from all levels are rolled up to form the Build-To Baseline
8. The Build-To Baseline is reviewed at the CDR
9. The successful CDR opens the gate to proceed with System-Level build

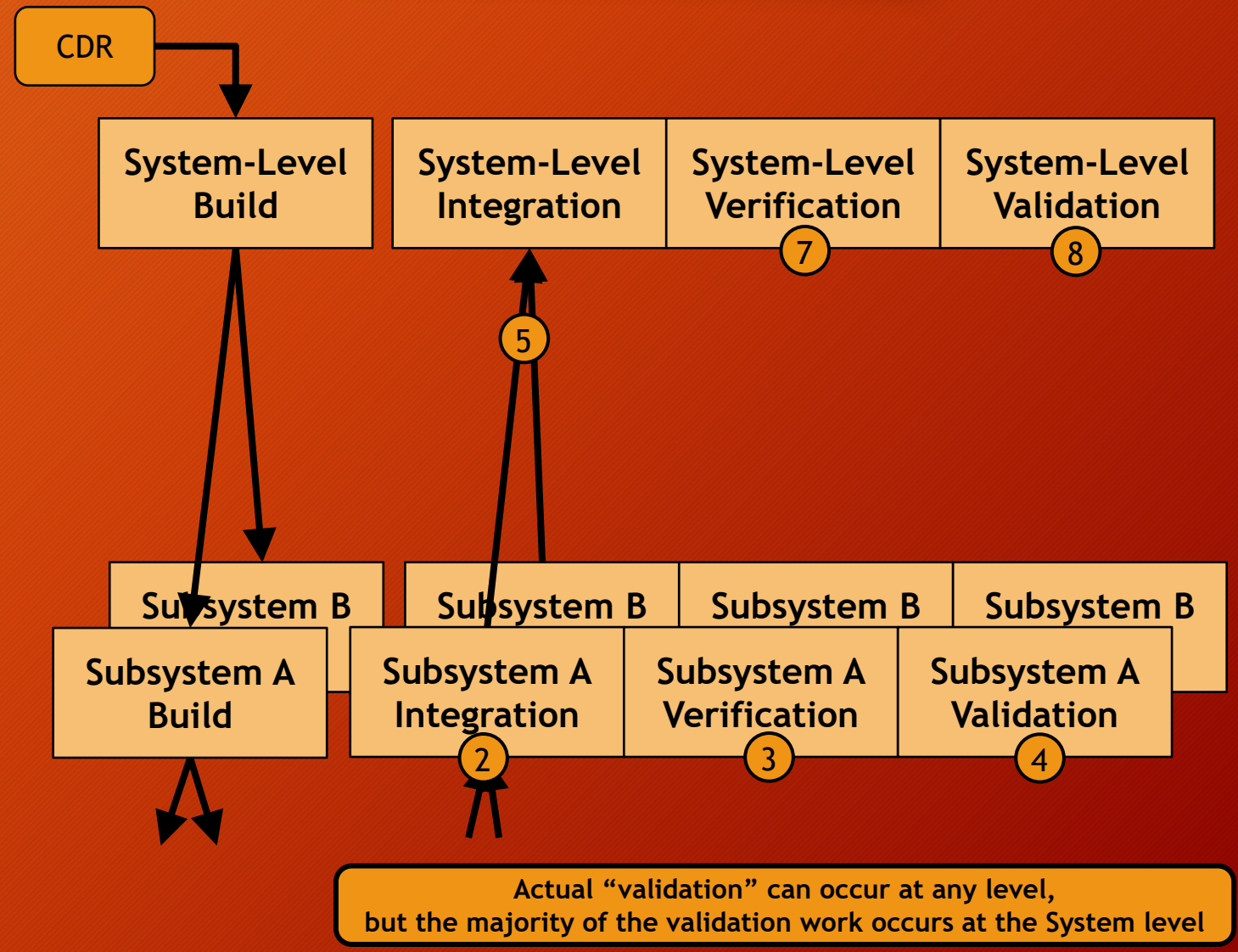


© Copyright 2023-2024 John G. Artus

How IAR is Implemented Among Levels

How IAR is Implemented Among Levels

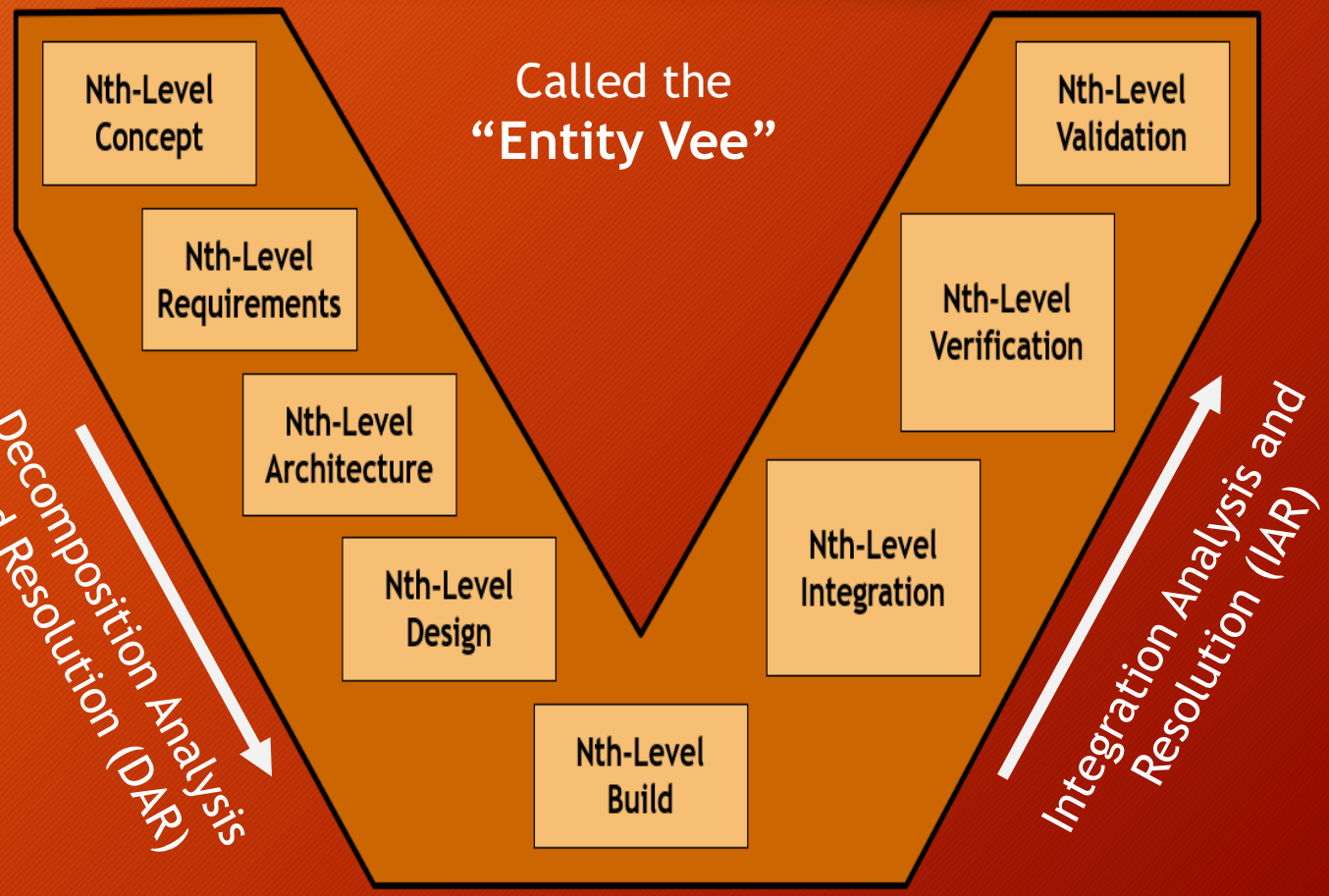
1. The process starts at the Lowest Configuration Item (LCI) level
2. Note: There is no Integration step at the LCI level
3. Starting with the LCIs, LCIs are verified to prove that they satisfy their requirements
4. If required, LCIs are validated to prove their useability in the operational environment
    a. LCIs are normally not validated
    b. Validation normally occurs for components at the upper 2 or 3 levels of the component hierarchy
5. The LCI entities are flowed up to the next highest level where they are integrated together to form entities at the new higher level
6. Steps 3 thru 5 are repeated recursively until the System-level is reached
7. The System is verified to prove that it satisfies its requirements
8. The System is validated to prove its useability in the operational environment



© Copyright 2023-2024 John G. Artus

The Entity Vee

- This model is based on the sequential flow of SE Technical Processes that result in the development and verification of a single system entity (component or configuration item)
- The left leg represents the sequence of definition elaboration, called Decomposition Analysis and Resolution (DAR)
- The right leg represents the sequence of assembly and performance assurance, called Integration Analysis and Resolution (IAR)
- The Entity Vee is repeated for every entity of the architecture from the system, down to the Lowest Configuration Items (LCIs)
- At each elaboration level (tier), the method of verification and integration to be used on the right leg of the Vee must be defined at the time that requirements and architecture (respectively) are developed on the left side
- Otherwise, requirements could be created that might never be verified, and system components could be designed that might never integrate together properly

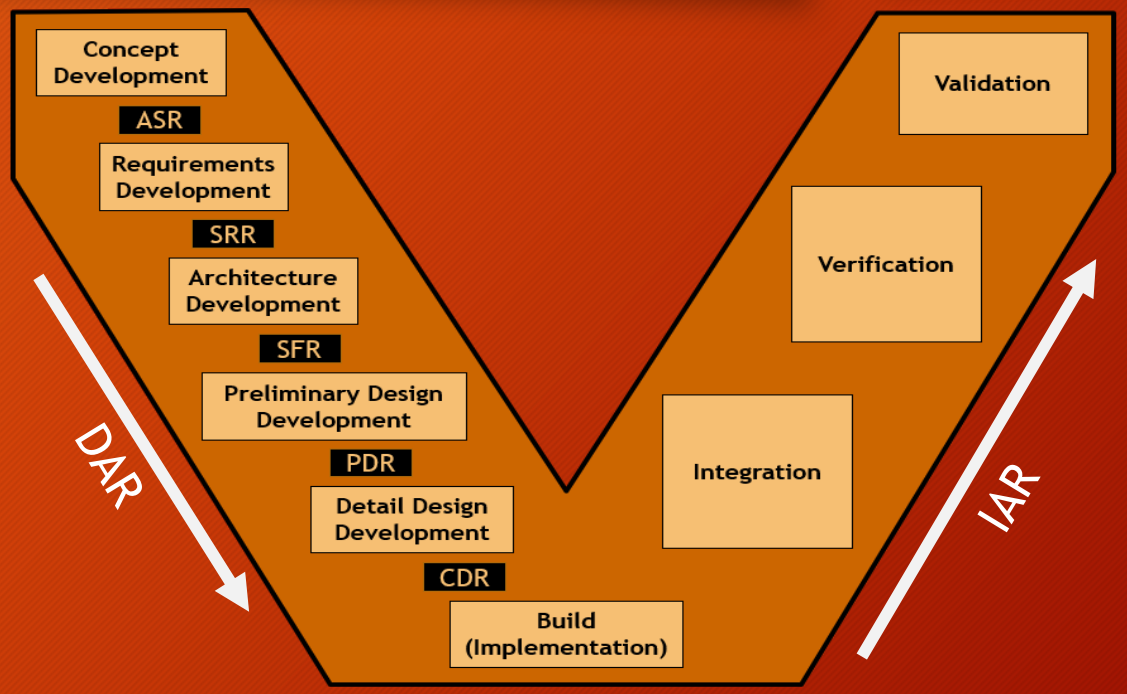


The SE Technical Processes are applied to every component at every level of the system decomposition hierarchy, from the System Level (Level 0) down to the lowest decomposed component

© Copyright 2023-2024 John G. Artus

The Role of Developmental Reviews

- Alternative Systems Review (ASR)
  - A technical review of the proposed conceptual solutions that selects the solution that has the best potential to be cost-effective, affordable, operationally effective, and suitable
  - Successful completion of the ASR determines that work to develop the system requirements may proceed
- System Requirements Review (SRR)
  - A formal review conducted to ensure that system requirements have been completely and properly identified and that a mutual understanding between the stakeholders and developer (contractor) exists
  - Successful completion of the SRR determines that work to develop the initial system design may proceed
- System Functional Review (SFR)
  - A technical review to establishes whether the defined system functionality can satisfy the system requirements, whether the system's lower-level performance requirements are fully defined and consistent with the system concept, and whether lower-level systems requirements trace to top-level system performance requirements
  - Successful completion of the SFR establishes the Functional Baseline and determines that the Integrated Product Teams (IPTs) are prepared to start preliminary design work
- Preliminary Design Review (PDR)
  - A technical review that is the first opportunity for stakeholders to closely observe the contractor's hardware and software design
  - The review establishes that each function in the Functional Baseline has been allocated to one or more system configuration items, establishes the existence and compatibility of the physical and functional interfaces among the configuration items and other items, and ensures that the system will be operationally effective
  - Successful completion of the PDR establishes the Allocated Baseline and determines that the Integrated Product Teams (IPTs) are prepared to start detail design work



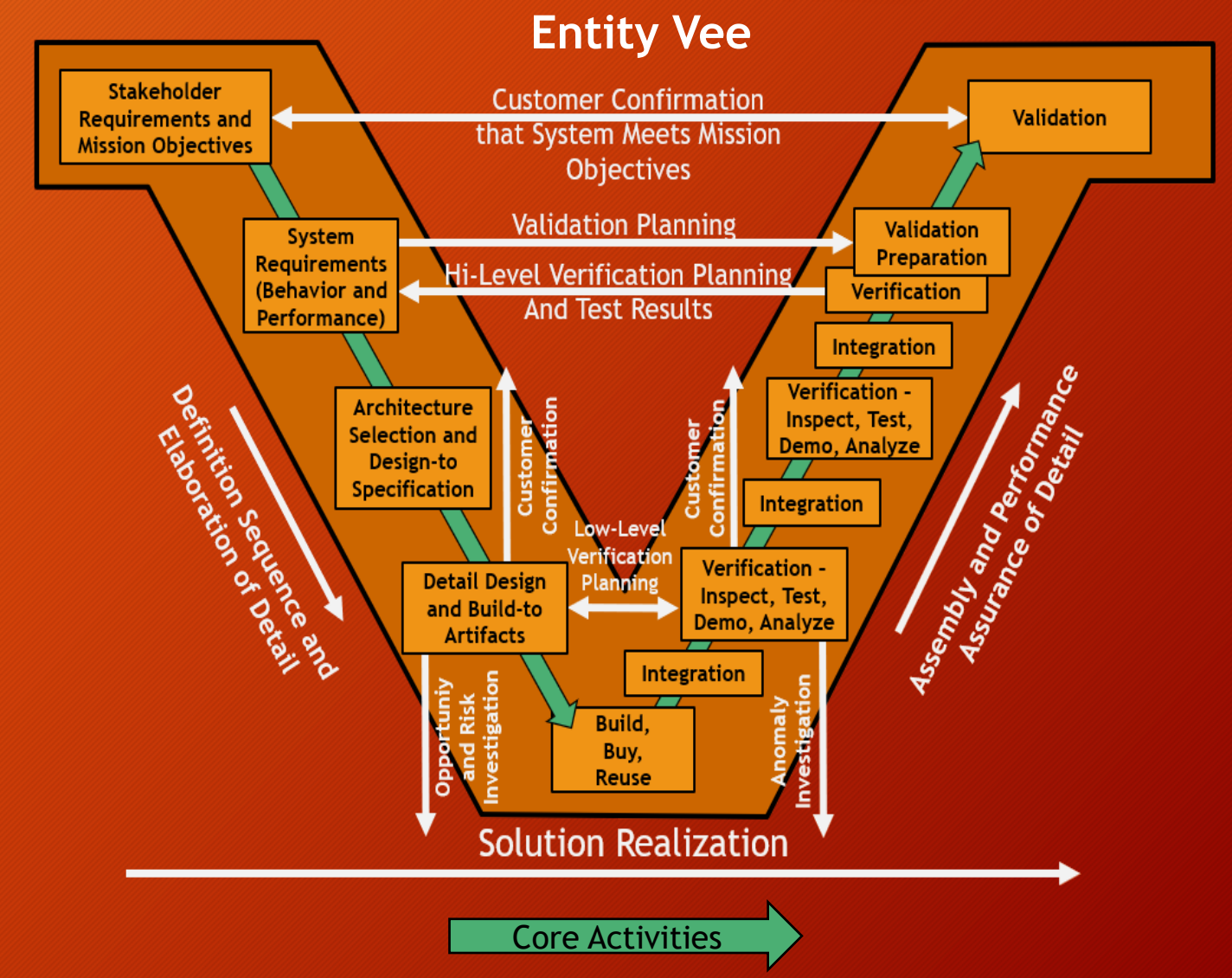
- Critical Design Review (CDR)
  - A technical review that ensures that each Configuration Item (CI) has been captured in the detailed design documentation (a set of detailed drawings and specifications), and ensures that that the subsystem requirements, subsystem detailed designs, and plans for test and evaluation form a satisfactory basis for proceeding into system implementation and integration, thus establishing the Initial Product Baseline
  - Successful completion of the CDR establishes that the system can proceed into system implementation (fabrication) and integration, demonstration, and test and can meet stated performance requirements within cost, schedule, and risk

These review types are not standard across all industries

© Copyright 2023-2024 John G. Artus

Off-Core Activities of the Entity Vee

- Baseline elaboration of a single entity is performed within the core of the Entity Vee
- Off-core activities associated with opportunity and risk management are pursued downward to the level of detail necessary for issue evaluation and resolution
- Opportunity and risk investigations are performed either in series or in parallel with the on-core development work
- Exploratory design and analysis can be performed at any point in the project cycle to investigate or prove performance or feasibility concepts
- Off-core activities associated with customer confirmation of the entity definition and verification are pursued upward to the level necessary for the required approvals



© Copyright 2023-2024 John G. Artus

Value of Off-Core Activities of the Entity Vee

- As an example, to evaluate two competing concepts, technical feasibility of the two concepts would be modeled
- Selection might be based on performance versus cost and complexity of the system
- Customer confirmation can provide valuable in-process validation of the preferred approach
- In the right leg, off-core investigations are used to resolve assembly and verification anomalies
- This may require descending vertically to examine design errors, or operator error, etc
- Upward off-core user interactions obtain stakeholder confirmation or rejection of the realized performance
- At any level of decomposition, the customer of an entity is the manager of the next higher level of decomposition

In the Entity Vee, off-core interactions address individual entity solutions and not the integration of the whole architecture
Integration of the whole architecture is modeled by the Architecture Vee

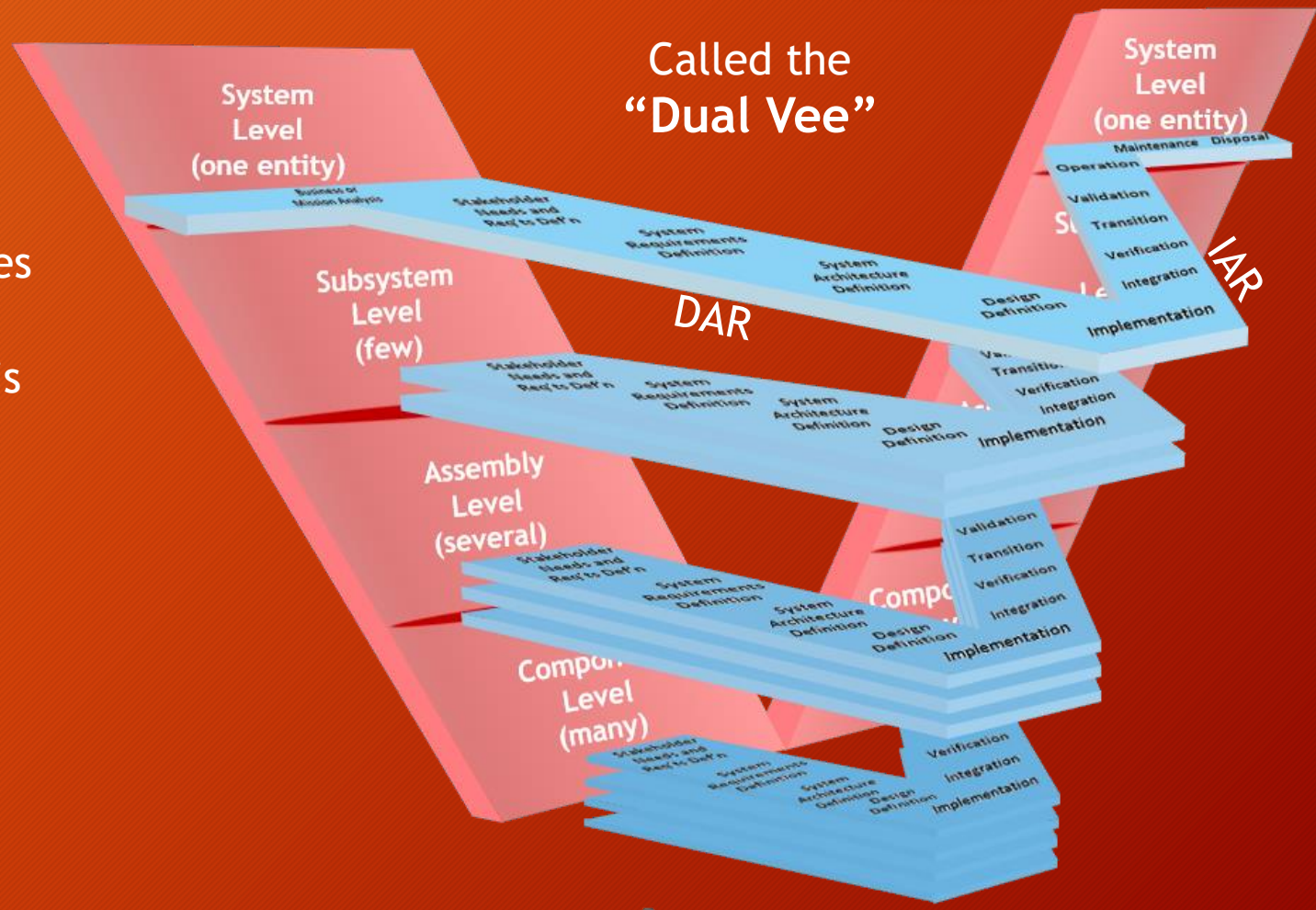
© Copyright 2023-2024 John G. Artus

The Dual-Vee Model

© Copyright 2023-2024 John G. Artus

Combines the Architecture Vee with the Entity Vee

- In order to convert a set of user needs into a deployed system that satisfies those needs requires that a solution be found for each entity at each level of architecture decomposition
- This can be visualized by positioning Entity Vees orthogonal to the Architecture Vee
- For each entity of the Architecture Vee there is a corresponding Entity Vee that addresses the entity development
- For example, the Architecture Vee here shows two subsystems (there could be more)
- The two Entity Vees shown represent the process for creating those two subsystems
- This figure reiterates the relationship of the DAR and IAR processes to the Architecture Vee
- It elaborates further on the interrelationships by superimposing the DAR and IAR on the Entity Vees that they support



Architecture Vee	Addresses the architecture component hierarchy (the hierarchy of system entities)
Entity Vee	Addresses the application of technical processes for each architectural entity

© Copyright 2023-2024 John G. Artus

Applications of the Vee Model

Considerations for Technical Development Tactics when Applying the Vee Model

© Copyright 2023-2024 John G. Artus

Tactical Development and Delivery Approach

Development and delivery decisions are driven by the business case in response to the demands of the stakeholders

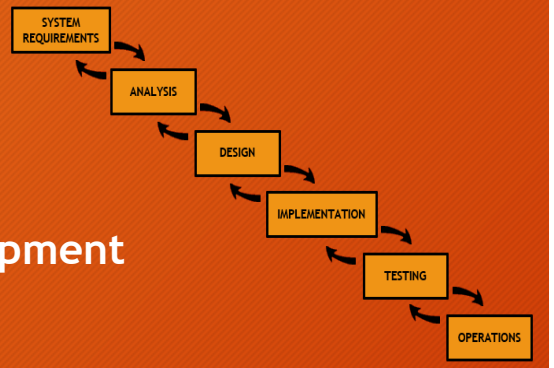
This results in a business strategy that is achieved through implementation tactics

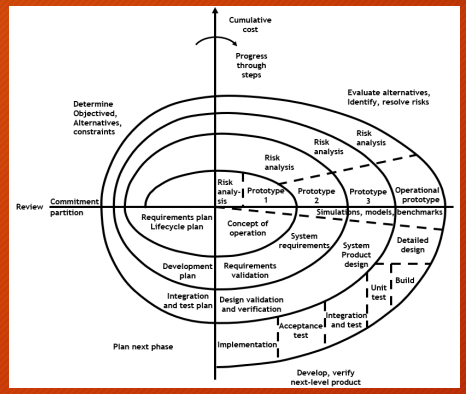
The systems engineer needs to fully appreciate the flexibility of the project to accommodate and benefit from the various tactical development and delivery approaches

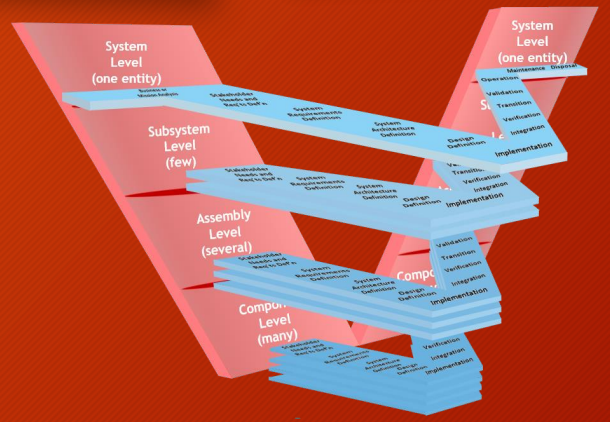
To arrive at the best tactical decision, the project manager and the systems engineer must collaborate on a development approach

This decision is then baselined and communicated to the project team so that the tactics can be built into all planning

Development Model







Waterfall | Spiral | Dual-Vee

Primary Development Method

- Unified
- Incremental

Secondary Development Method

- Unified: Linear, Evolutionary
- Incremental: Linear, Evolutionary

Delivery Method

- Unified / Linear: Single
- Unified / Evolutionary: Single, Multiple
- Incremental / Linear: Single, Multiple
- Incremental / Evolutionary: Single, Multiple

© Copyright 2023-2024 John G. Artus

Primary Development and Delivery Approaches

- Primary development methods
  - Unified
    - Effective for systems in which decomposition into an architecture with separate deliverable elements or modules is not practical
    - Example: the physical structure of a spacecraft
  - Incremental
    - Decompose the concept into an architecture having entities to be developed incrementally (i.e., separately for later integration)
    - Allows parallel development, assigning experts to each increment
    - Exhibits flexibility to accommodate funding and schedule constraints
    - Incremental development can plan for subsequent upgrading by increment

- Example of incremental approach: Automobile Product Line
  - Engines, chassis, and transmissions are separately developed
  - Then integrated into a complete automobile at the final assembly plant
  - Increments that are later discovered to be faulty can be recalled and replaced in the field
- Example of incremental approach: Software Development
  - Incremental development can start with the most important requirement
  - The increment is complete when the increment satisfies the requirement
  - Then building on that verified increment, the thrust would be to satisfy the second requirement and so on
  - With this incremental approach, each increment is built on the previous set resulting in one single delivery
  - However, later upgrades to internal increments are not possible
  - In this case, the entire integrated set of increments must be upgraded as a whole

© Copyright 2023-2024 John G. Artus

Secondary Development and Delivery Approaches

- Secondary development methods
  - Linear
    - A single-path approach
    - The requirements and the solution are sufficiently well understood
    - Allows straight-forward design and implementation
    - No iteration or experimentation is required or desired
    - Example: Installation of electrical and plumbing systems in home construction is a linear approach developed over years of experience
  - Evolutionary
    - Experimentation or investigation is necessary to determine the best solution
    - Works well for
      - Uncertain requirements
      - Pursuit of opportunities and risks
      - Pursuit of alternate concepts and solutions
    - The evolutionary approach is common to research projects
    - Disadvantage: unpredictability of progress
    - As a result, cost and schedule estimates are guesses and are rarely met

© Copyright 2023-2024 John G. Artus

Delivery Method

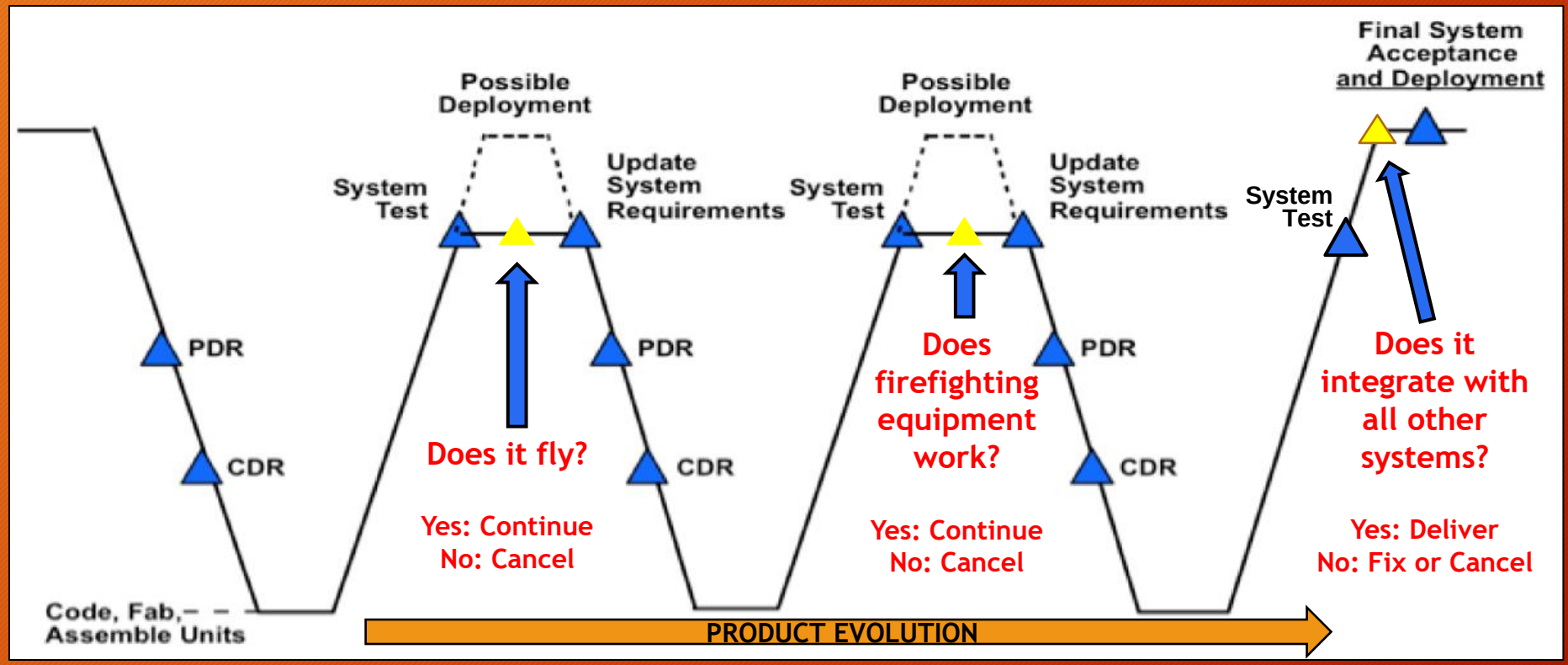
- For unified, linear development
  - Only a single delivery occurs
- Incremental, with or without evolutionary development, requires a decision
  - Field the system in a single delivery
  - Or deliver increments and versions of increments to gradually increase solution capability over time
- This decision for incremental can be driven by
  - The urgency for a solution to be fielded
  - The staggered availability of functional capability, funding limitations, regulatory constraints, or any other factors making staged fielding beneficial

© Copyright 2023-2024 John G. Artus

Example: Unified – Evolutionary Development with Multiple Version Deliveries

Example: Large Special-Purpose Firefighting Aircraft Program

- Evolution 1: Does it fly?
- Evolution 2: Does firefighting equipment work?
- Evolution 3: Does it integrate well with all other firefighting systems?



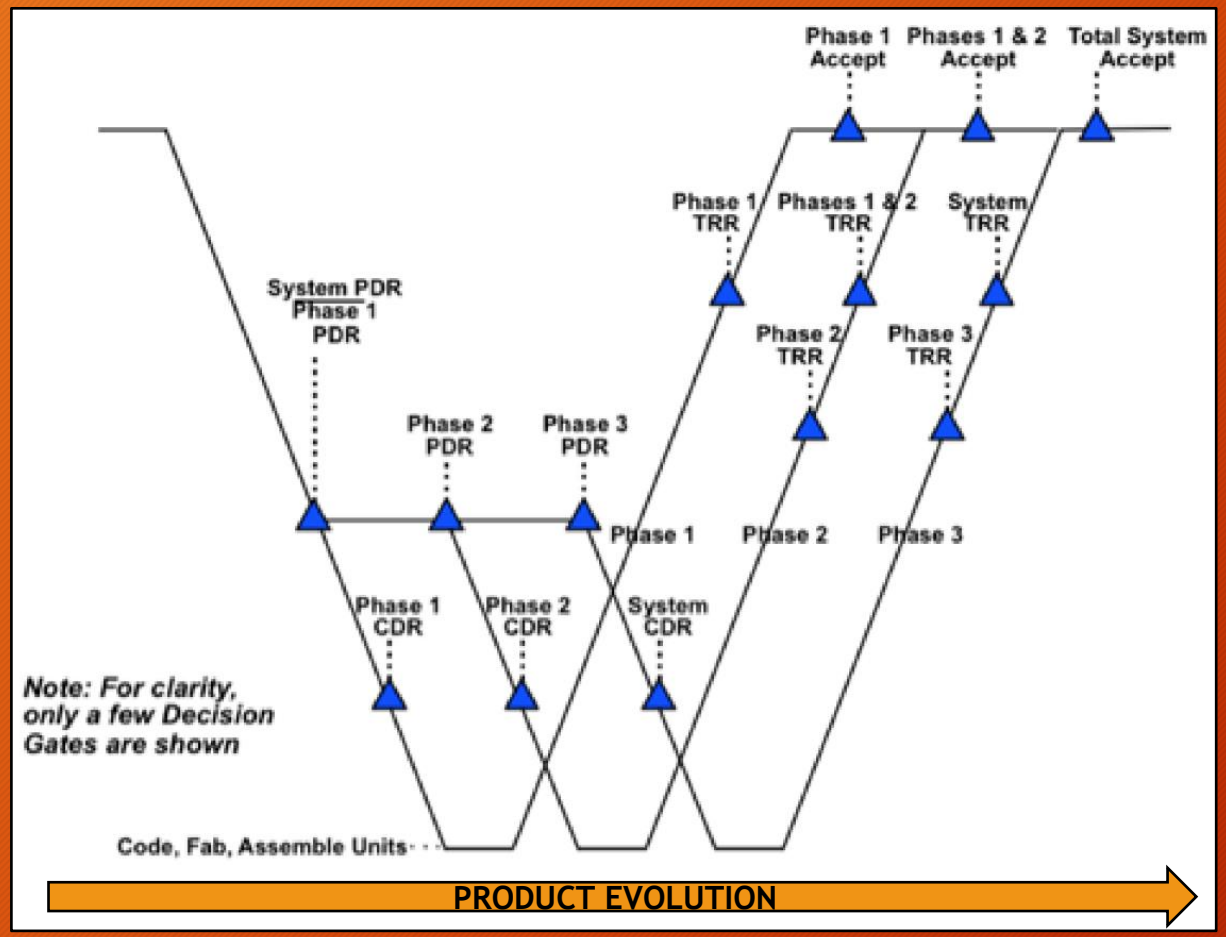
https://www.kcrw.com/news/articles/firefighters-are-fuming-about-drones-over-wildfires

© Copyright 2023-2024 John G. Artus

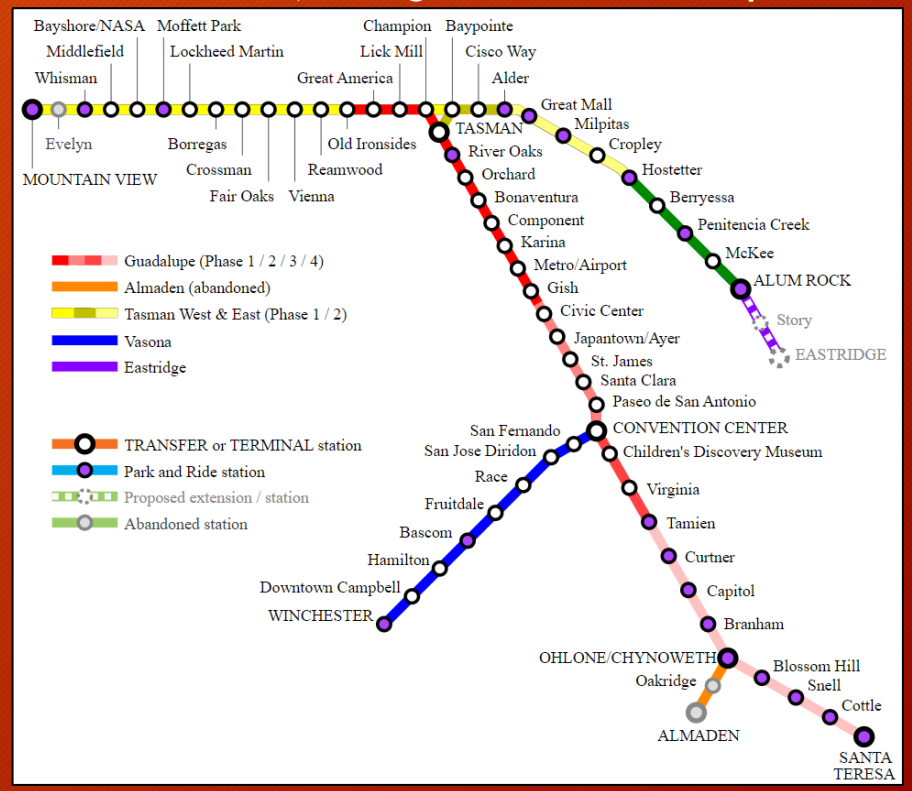
Example: Incremental – Linear Development with Multiple Incremental Deliveries

Example: San Jose, CA Light Rail Program

- Phase 1: First segment of track (10-miles in 1990)
- Phase 2: Second segment of track (18-miles in 1993)
- Phase X: Final segment(s) of track to additional cities (to be completed in 2027)



San Jose, CA Light Rail Stations Map



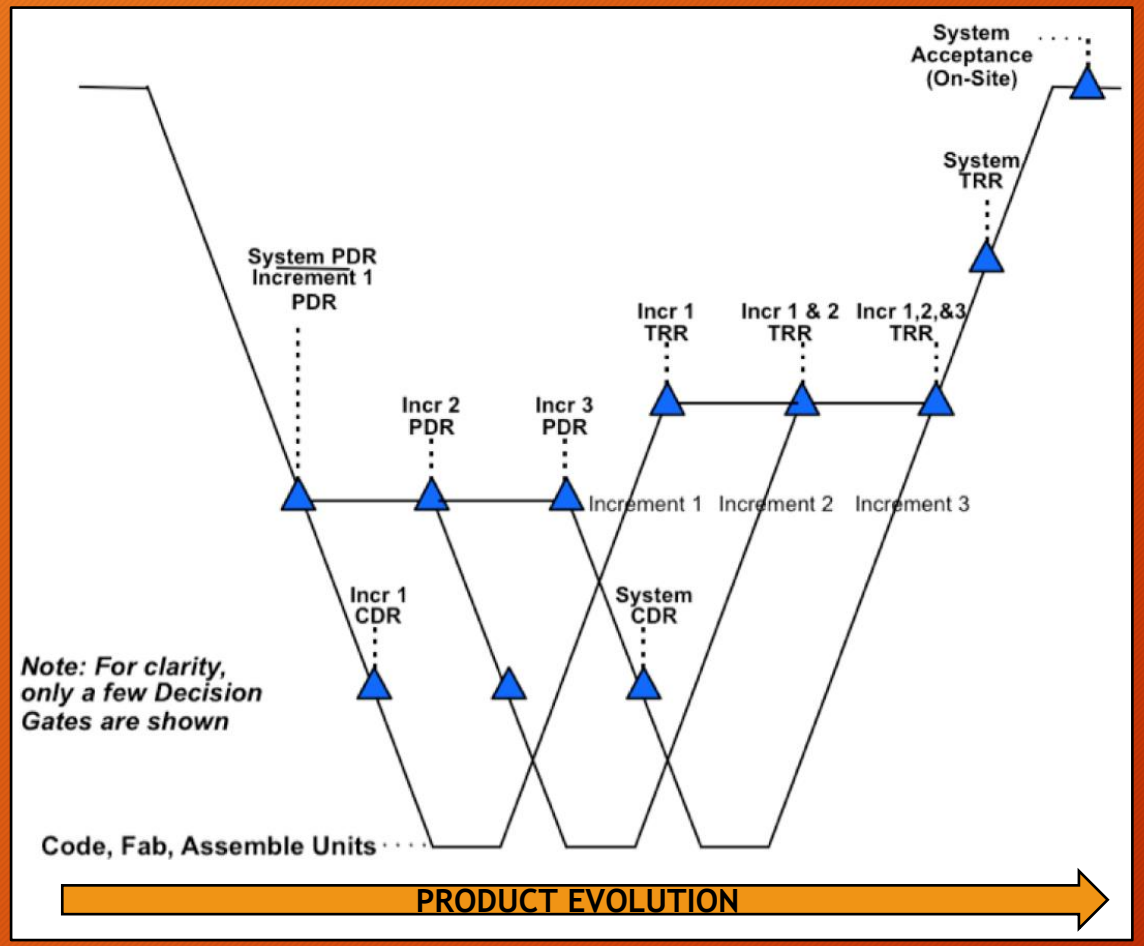
https://en.wikipedia.org/wiki/File:VTA_Light_Rail_map_line_history.svg

© Copyright 2023-2024 John G. Artus

Example: Incremental – Linear Development with Single Delivery

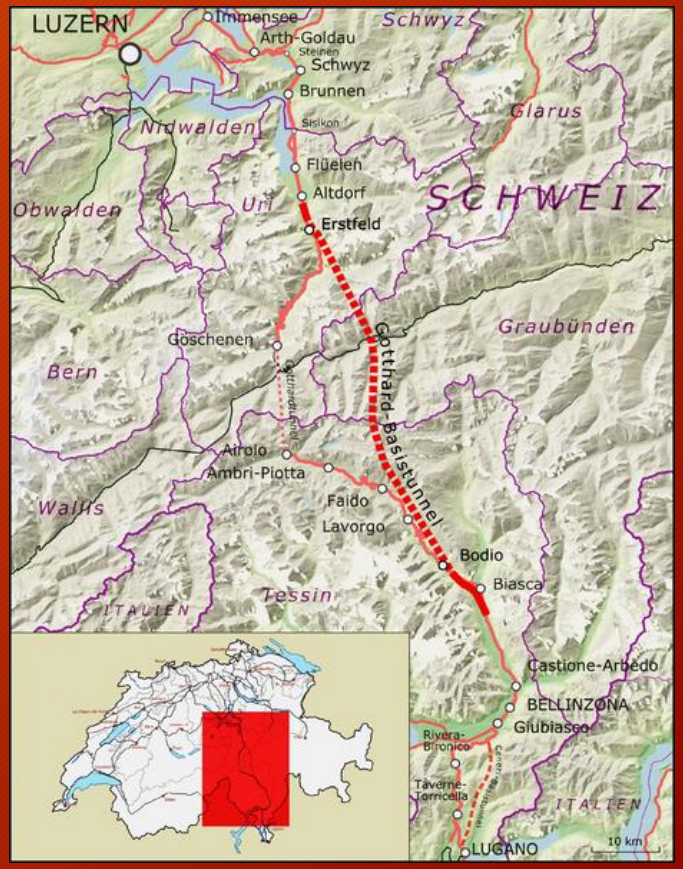
Example: St. Gotthard Alps Tunnel Program

- Phase 1: First section of tunnel (Sedrum in 1996)
- Phase 2: Second section of tunnel (Amsteg in 1999)
- Phase X: Final section of tunnel (Commissioned in 2016)



Route could only be used when fully completed

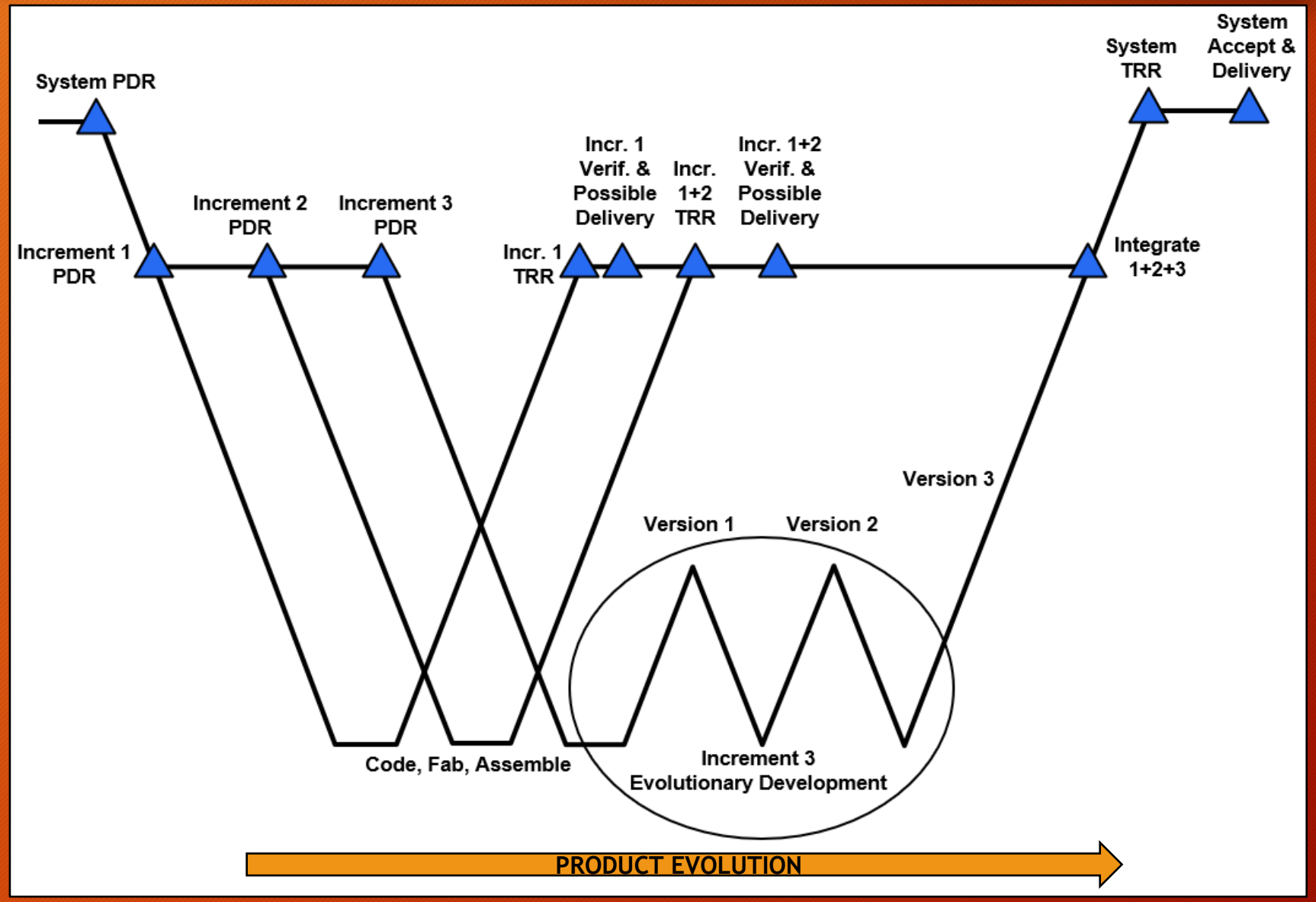
Map Showing Route of St. Gotthard Alps Tunnel



https://en.wikipedia.org/wiki/File:Map_Gotthard-Basistunnel.png

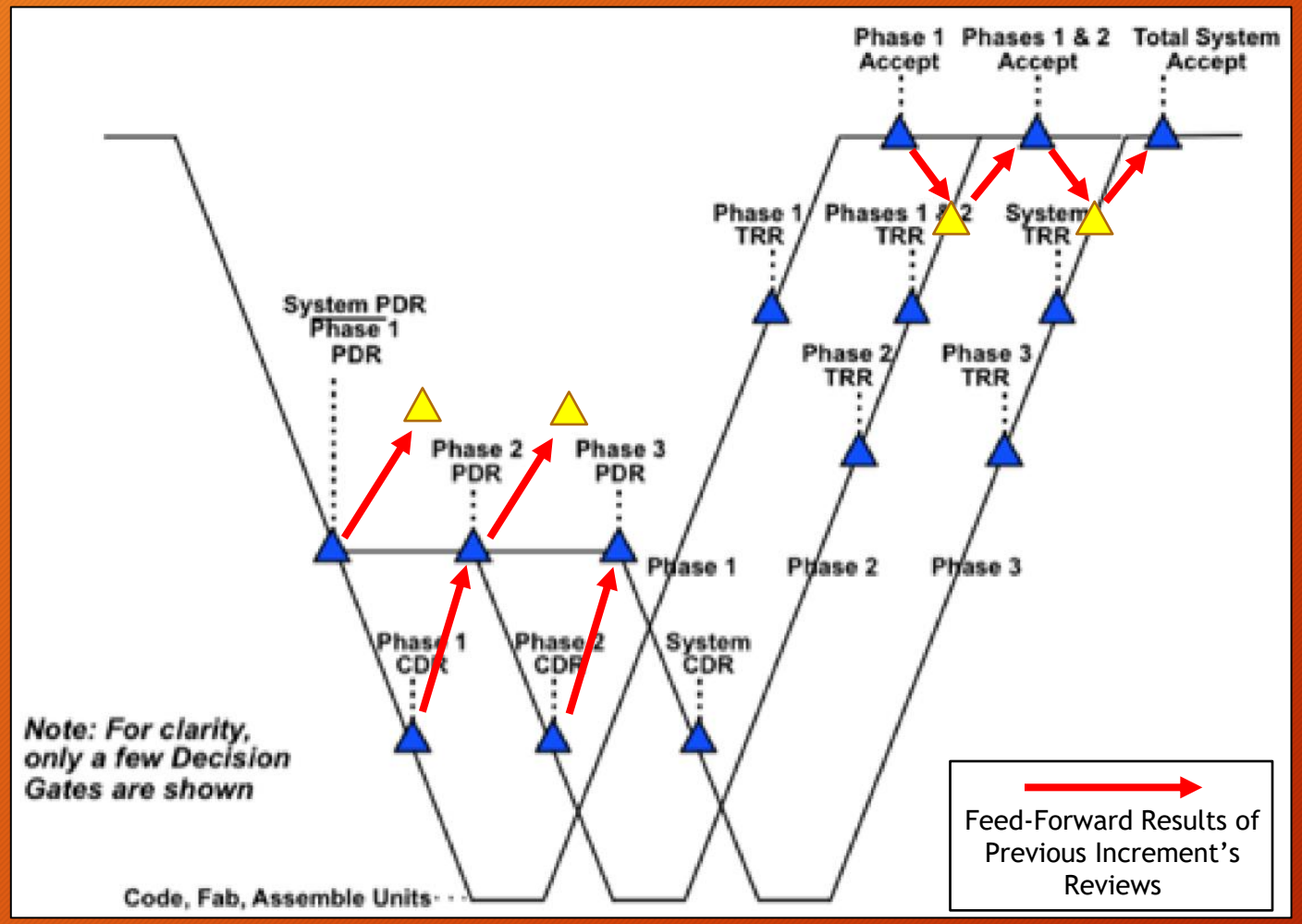
© Copyright 2023-2024 John G. Artus

Example: Incremental-Linear and Evolutionary Development with Single or Multiple Version Deliveries



© Copyright 2023-2024 John G. Artus

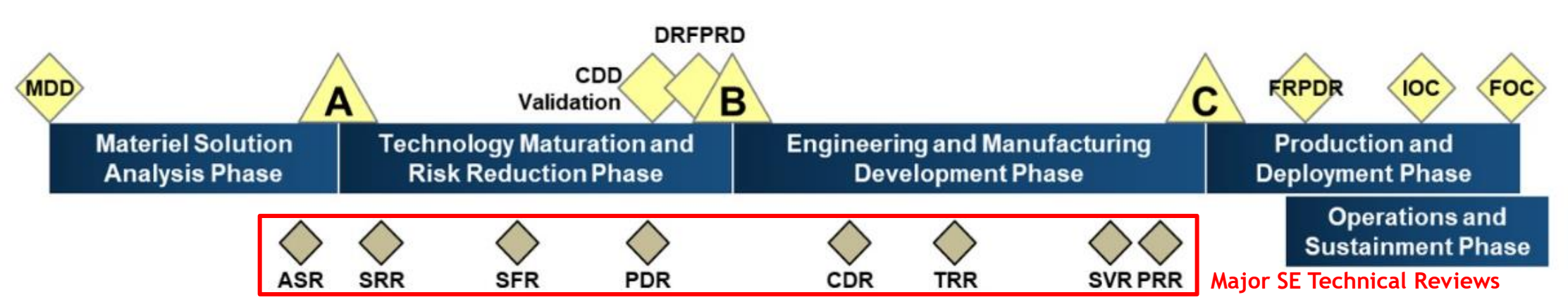
Evolutionary Development - Applying Lessons Learned



Programs should take every opportunity to learn from one review, and apply lessons learned to the next cycle prior to the next cycle's next review

© Copyright 2023-2024 John G. Artus

Example Lifecyle Model (DoD)



- MDD - Materiel Development Decision
  - Decision to proceeed with system development
- CDD - Capability Development Document
  - Specifies the operational requirements for the system that will deliver the capability that meets the operational performance criteria
- DRFPRD - Development RFP Release Decision Point
  - Ensures that an executable and affordable program has been planned using a sound business and technical approach
- FRPDR - Full-Rate Production Decision Review
  - Assess the results of initial OT&E and initial manufacturing to determine whether to proceed to FRP
- IOC – Initial Operational Capability
- FOC – Full Operational Capability

© Copyright 2023-2024 John G. Artus

References

- Forsberg, Kevin; Mooz, Hal; Cotterman, Howard (2005). *Visualizing Project Management: Models and Frameworks for Mastering Complex Systems*, 3rd Edition, John Wiley & Sons, Inc., Hoboken, New Jersey
- Systems Engineering Body of Knowledge (SEBoK) v2.7 https://sebokwiki.org/wiki/Guide_to_the_Systems_Engineering_Body_of_Knowledge_(SEBoK)
- INCOSE G2SEBOK v3.30 http://g2sebok.incose.org/app/qualsys/view_by_id.cfm?ID=INCOSE%20G2SEBOK%203.30&ST=F
- Walden, D., Roedler, G., Forsberg, K., Hamelin, R., Shortell, T. (2015). *INCOSE Systems Engineering Handbook*, 4th Edition, John Wiley & Sons, Inc., Hoboken, New Jersey
- Royce, W. (1970). *Managing the Development of Large Software Systems* http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf
- Boehm , B. (1987) *A Spiral Model of Software Development and Enhancement* https://www.cse.msu.edu/~cse435/Homework/HW3/boehm.pdf
- AcqNotes page on Alternative System Review https://acqnotes.com/acqnote/acquisitions/alternative-systems-review
- AcqNotes page on System Requirements Review https://acqnotes.com/acqnote/acquisitions/system-requirements-review-srr
- AcqNotes page on System Functional Review https://acqnotes.com/acqnote/acquisitions/system-functional-review
- AcqNotes page on Preliminary Design Review https://acqnotes.com/acqnote/acquisitions/preliminary-design-review
- AcqNotes page on Critical Design Review https://acqnotes.com/acqnote/acquisitions/critical-design-review
- AcqNotes page on Configuration Baseline https://acqnotes.com/acqnote/careerfields/configuration-baselines
- AcqNotes page on Technical Baseline https://acqnotes.com/acqnote/careerfields/technical-baseline

© Copyright 2023-2024 John G. Artus