Derivation: the creation of requirement based on an analysis of higher-order requirements (ex. availability requirements)

Allocation: the assignment of a requirement to one or more system elements within the architecture. (ex. thrust requirement > propulsion requirements)

Flow Down: assigning requirement to one or more lower-level elements. (ex. water resistant requirement > all elements exposed to outside environment)

Traceability: the creation of linkage from one system elements to others (ex. mtbf requirements > availability requirements)

System engineer must know how:

- test able

- Produced

- packaged

- transported

Pre-planned product improvement requirements (P3I)

Sys Engg mush work with reliability SMEs

RFA - request for action

RFI - request for information

RVTM - requirements verification and traceability matrix

How to write requirement statement:

* Specific intent
* Desired capability
* Desired level of performance
* Functions or behavioral characteristics
* Not specifying how functions/behaviors are performed (calling for specific components)
* requirement should not specify specific piece of hardware of software into the system
* Determining proper hardware or software is not job description of the systems engineer. It is job of SMEs.
* each requirement should be unambiguous (all parties have same understanding of the requirements)
* requirement should be complete (stand alone, independent of other requirements, avoid cross referencing other requirements, avoid pronoun)
* requirement must be feasible (should not request impossible or unrealistic behavior or feature, some time all req are individually feasible but collection of them are not feasible)
* requirements should be verifiable (analysis, demonstration, test or inspection)
* requirement can use illustration/graphics

Architecture Definition

* it is abstract representation of system elements
* architecture should be implementation free
* is being performed as enabled for other process
* It is primary artifact for the system requirements specifications etc.

Primary goal for architecture definition:

* produce alternate concepts
* compare concepts in trade studies
* select one or more concept alternative(s)

Inputs to architecture definitions:

* Life cycle concepts
* system function definition
* system requirements
* system functional interface identification
* System requirements traceability
* Updated RVTM
* Design traceability
* Interface definition update identification
* Life cycle constraints.

Goes into architecture definition

* Prepare for architecture definition
* Develop architecture viewpoints
* Develop models and views
* Relate the architecture to the design
* Assess architecture candidates
* Manage the selected architecture

Output of architecture definition

* Architecture definition strategy
* System architecture description
* System architecture rationale
* Documentation tree
* Preliminary interface definition
* Technical performance measure needs and data
* Architecture traceability
* Architecture definition record.

Architecture views

* operational (user's perspective)
* logical (acquirer/customer's perspective)
* Physical (designer's perspective)

Architecting is abstract/inductive. More art than science designing is deductive. More science than art

Architecture is not a modeling language.

Model based systems engineering (MBSE)

* systems engineering (requirement development, functional and mission development, physical definition)
* MBSE tool
* MBSE language (sysML)

A thing that performs a function is called physical including software

System model domains -

* Requirement domain: description of mandatory things your widget needs to do
* Behavioral domain: all the ways your widget should behave goes
* Structural domain: what your widget is made out of is in structural domain

Interfaces: (forces, energy, data, signals or materials)

SysML - Systems modeling language and UML - unified modeling language:

* Modeling language: artificial language of communicating concept in standardized way
* agreed upon grammar and vocabulary
* mostly graphical

Unified modeling language:

* general purpose modeling language for software engineer that is intended to provide a standard way to visualize the design of a software system

System modeling language:

* general purpose modeling language for systems engineer that is intended to provide a standard way to visualize the design of a system (hardware and software)
* it adds two new diagram (requirements and parametric)

Type of diagrams:

* Structure Diagram types
  + Bdd - block definition diagram
  + ibd - internal body diagram
  + par - parametric diagram
  + pkg - package diagram
* Behavior diagram types
  + act - activity diagram
  + seq - sequence diagram
  + stm - state machine diagram
  + ucd - use case diagram
* Requirement diagram type
  + req - requirement diagram

BDD (block definition diagram)

* it is a structure of system showing system blocks, their attributes, property and interactions
* analogue to class diagram in UML
* good for systems hierarchy

IBD (internal block diagram)

* internal structure of single structure of the block
* inputs on the perimeter of diagram are brought in

Activity diagram

* Depicts system behaviors, sequencing, decision and other logic between system behaviors
* functional interfaces (force, energy, data, signal, material)
* control flow is dotted line and object flow is solid line
* decision node needs logic

Sequence diagram (never saw using this diagram)

* blocks
* relation between blocks and actors
* synchronous messages or call messages

State machine

* behavioral diagram
* on state, off state, degraded state
* nested state (substate)
* start point and end point are important

Parametric diagram

* performance, reliability and SWaP analysis (size weight and power)

Package diagram

* hierarchy diagram
* architecture diagram
* requirement domain, behavior domain and structural domain\

UML diagrams:

* Behavioral Diagram
* Activity diagram
* State machine diagram
* Interaction diagram
* Timing diagram
* Interaction overview diagram
* Communication diagram
* Sequence diagram
* Use case diagram

Structure diagram

* component diagram
* class
* composite
* deployment
* package
* object

Behavioral/functional domain:

* function/behavior and activity are synonymous in sysengg

Function is an action performed by system element that converts inputs to output. Function:

* uses input
* generates output
* based on requirements/context/scenario
* decomposes/decomposed by sub-function
* performs/performed by component

Activities vs action

* Activities are overarching system behavior, composed of actions. These are decomposed into lower-level actions.
* Actions are leaf level system behavior. Actions are not decomposed. Actions are basic units of functionality.

Activity must have at-least one input and one output.

Outputs are always different than inputs

Sequence diagram always draws relation between physical elements from functional perspective.

Functional block diagram:

- it depicts relation between internal elements to external entity

Use case:

it is a list of steps user will most likely deploy, operate, support or dispose of a system to achieve a desired goal.

* helps refine requirements.
* foster top-down design
* preempt errors and faults
* save time and money preventing faults/ errors from existing
* help decompose the problem into more manageable segment

Elements of an usecase:

* Actors: a stakeholder / one or more actors.
* Summary: description of an use case in paragraph from
* goal: primary outcome desired by an actor
* pre-condition: things that must be true prior to starting the use-case
* gurantees: the least the system can do for the actor, if the goal is not met.
* trigger: step 1 in course of an environment
* course of event: numbered list of actions the actor/s and system takes to accomplish the goal.
* alternative path:

Primary source of function:

* Requirement specificaitons
* Context Diagram
* Life Cycle Concepts
* Legacy and predecessor system functions
* Parent functions

Top-down systems engineering:

* The behavioral domain is bases on
  + the requirements domain
  + the mission domain
  + itself
* the behavioral domain is not based on
  + the physical domain

Top-down sys engg

* Needs, technology driver, context basis of:
  + life cycle concepts and requirements basis of:
    - functions basis of:
      * hardware/ software

Hardware and software are based on functions. Functions are not based on hardware and software.

Stakeholder needs and requirements definition focus:

* high-risk systems ands subsystem entities
* defining systems and subsystem performance requirements/MOEs
* defining the lifecycle concepts

Architecture definition focus:

* Decompose system and subsystem functions until makes/buy decision can be made at component level
* Mitigate function-related risks into the green
* Complete the rest of the functional architecture for non-primary-mission scenarios, fail modes and recovery operations
* Heavy use of analysis, modeling and simulations

Interfaces:

* Force
* Energy
* Data
* Signal
* material