How to write good requirements
Module 9 of 10
The use of requirements in the rest of the system development process

Version 1.2.5

Course Module topics

1. Introduction to requirements
2. Stakeholders and their importance
3. Identifying the stakeholders' wants
4. Converting stakeholder wants to needs
5. Documenting stakeholders' needs
6. Converting stakeholder needs to requirements
7. Converting requirements to well-written requirements
8. Converting well-written requirements to good requirements
9. **The use of requirements in the rest of the system development process**
10. Summary and closeout

Objectives of Module 9

1. To expose participants to the consequences of poorly-written requirements in the SLC, by explaining the use of requirements in the
   1. Needs identification and System Requirements States
   2. System Design State
   3. Subsystem Construction State
   4. Subsystem Test State
   5. System Integration State
   6. System Test State
   7. System Operations, Maintenance and Upgrade States
2. To explain the requirements and change management processes once the initial set of system requirements have been accepted
3. To discuss the place of requirements in the Agile paradigm (software)
4. To practice what has been taught
5. To provide the opportunity to exercise the 5 levels of knowledge in the updated Blooms taxonomy

Knowledge components

- Lecture
  - Sets the context and provides overview
- Readings
  - 0902 Improving monitoring of technical performance by using CRIP charts, https://youtu.be/5AUafacJ5AU, (24:27), 2015
- Resources (with respect to the Agile methodology)
  - 0950 Kasser J.E., The Cataract Methodology for Systems and Software Acquisition, *proceedings of the SETE 2002,* Sydney, Australia, 2002, the paper
  - 0951 Kasser J.E., The Cataract Methodology for Systems and Software Acquisition, *proceedings of the SETE 2002,* Sydney, Australia, 2002, the presentation slides
- Exercises
  - 9-1 Impact of a change request
  - 9-2 Knowledge reading 0902

Module topics

- **The system development process**
- The consequences of poorly-written requirements
- The use of requirements in the
  - A. Customer Needs Identification States
  - B. System Requirements State
  - C. Subsystem Design State
  - D. Subsystem Construction State
  - E. Subsystem Testing State
  - F. System Integration and System Testing States
  - G. Operations, Maintenance and Upgrade (O&M) States
  - H. System Disposal State
- The requirements and change management processes once the initial set of requirements have been accepted
- The place of requirements in the Agile paradigm (software)
- Exercises

The Hitchins-Kasser-Massie-Mabelo Framework - (HKM²F)

Complexity: Layer of complexity		A	B	C	D	E	F	G	H
Global (Planetary)	7								
Regional	6								
Socio-economic	5								
Supply chain	4								
Business (multiple)	3								
System (single)	2		System Development Process						
Product	1								
Component	0								

Lifecycle States

A – Customer Needs Identification	**B** – System Requirements	**C** – Subsystem Design	**D** – Subsystem Construction	**E** – Subsystem Testing
F - Systems Integration and Test	**G** - Operations and Maintenance	**H** – System Disposal		

States of the System LifeCycle (SLC)

	State
A	Customer Needs Identification
B	System Requirements
C	Subsystem Design
D	Subsystem Construction
E	Subsystem Testing
F	System Integration & System Test
G	Operations, Maintenance & Upgrade
H	System Disposal

J2

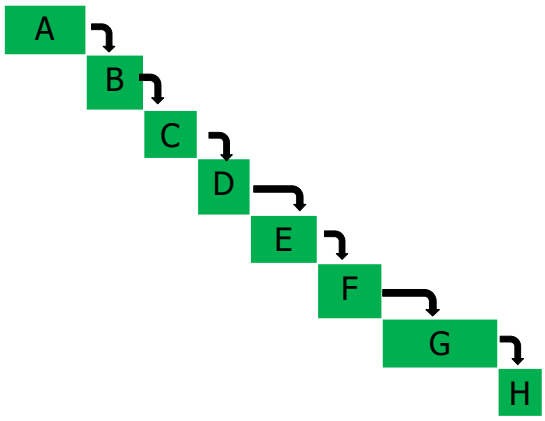
States of the SLC (Gantt chart)

	STATE	T_A	T_B	T_C	T_D	T_E	T_F	T_G	T_H
A	Customer Needs Identification								
B	System Requirements								
C	Subsystem Design								
D	Subsystem Construction								
E	Subsystem Testing								
F	System Integration & System Test								
G	Operations, Maintenance & Upgrade								
H	System Disposal								

Slide 7

J2 Joseph, 31/10/2023

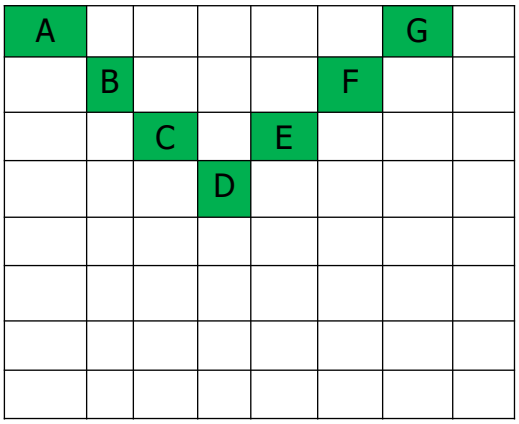
States of the SLC (Waterfall)



States of the SLC (N^2 chart)

A	X						
	B	X					
		C	X				
			D	x			
				E	x		
					F	x	
						G	x
							H

States of the SLC (V view)



Summary of SDP Milestone reviews

- (Start of project)
- ***OCR*** at the end of the Needs State
- ***SRR*** at the end of the Requirements State
- ***PDR*** at the end of the Preliminary design sub-State of the Design State
- ***CDR*** at the end of the Design State
- ***TRR*** at the end of the Construction State
- ***IRR*** at the end of the Subsystem-Testing State
- ***DRR*** at the end of the System Integration and Test States
- (End of project)

- Engineering
  - Technical reporting
    - Progress, problems and risks
- Management
  - Progress, problems and risks
  - Short term future in detail
    - E.g. SRR to Next Formal Milestone
  - Distant future in less detail
    - E.g. SRR to completion
    - Must be feasible
  - Tasks, what, who, where, when
  - Process risks
  - Excuses
  - Lessons Learned
  - Etc.

Framing the SDP with reference the 9-system model

1. **The undesirable situation** (S1)
   - A need to transition the undesirable situation to a FCFDS
2. **Assumptions**
3. **The FCFDS** (S3)
   - S1 but without the undesirable aspects of S1 with added desirable functionality
4. **The solution system (S6) operating in its context (S7) sometime in the future**
   - When S1 has evolved into S7
5. **The problem**
   1. To validate the remedy provided by the solution system (S6) (Do this in S8)
   2. To realize the solution system (S6) according to the SDP (Do this in S5)
   3. To plan the solution SDP (S5) (Do this in S4)
   4. To gain an understanding of the causes of undesirability of the undesirable situation (S1) (Do this in S2)

Framing the SDP with reference the 9-system model

1. **The solution**
   1. Gain an understanding of the causes of undesirability of the undesirable situation (S1) (Do this in S2)
   2. Create the CONOPS (Do this in S2)
      - OCR
   3. To plan the solution SDP (S5) (Do this in S4)
   4. Create the system requirements specification (SRD) (Do this in S4)
      - SRR
   5. Realize the solution system (S6) according to the SDP (Do this in S5)
      - PDR, CDR, TRR, IRR
   6. Validate the remedy provided by the solution system (S6) in its operational context (Do this in S8)
      - DRR
   7. Transition into service

CRIP charts

- In the traditional paradigm, we can tell the customer
  - How much money has been spent and if we expect to overrun
  - How much time has elapsed and if we expect to overrun
- BUT, we can't tell the customer how much of the project has been completed
- CRIP charts provide a way to answer the question "*how much of my project has been completed*?"
- Show change in the state of requirements as the SDP progresses
- Reading/video 0902

	Identified			In process			Completed			In test			Accepted		
Range	P	E	A	P	E	A	P	E	A	P	E	A	P	E	A
1															
2															
3															
4															
5															
6															
7															
8															
9															
10															

Module topics

- The system development process
- **The consequences of poorly-written requirements**
- The use of requirements in the
  - A. Customer Needs Identification States
  - B. System Requirements State
  - C. Subsystem Design State
  - D. Subsystem Construction State
  - E. Subsystem Testing State
  - F. System Integration and System Testing States
  - G. Operations, Maintenance and Upgrade (O&M) States
  - H. System Disposal State
- The requirements and change management processes once the initial set of requirements have been accepted
- The place of requirements in the Agile paradigm (software)
- Exercises

Meetings

- To discuss the meaning of the poorly-written requirement
- Each meeting
  - Takes one hour
  - Has five people (average)
- Multiply that by the number of poorly-written requirements
- Cost of each meeting = five people hours * (salary + overhead)
- Multiply that cost by the number of meetings over the SDP to discuss all the poorly-written requirements
- Note those people are not working on what they should be working on

Lessons learned from NASA on system design

- Lack of clear definition of requirements early in the detailed design phase leads to poor project outcomes, including:
  - Starting designs before requirements were known
  - Incomplete documentation of requirements
  - Vague specifications and changing requirements that resulted in many iterations
  - Lack of early thorough requirements analyses prior to start of design
  - Premature design freezes that underwent many changes
  - Late design freezes that permitted many changes
  - Inability to freeze design
  - Inadequate consideration of design margins
  - Designing from the "bottom-up" instead of from the "top-down"

In the days before software intensive projects
Contrast these with "Agile software development"

Some of the problems caused by poorly written requirements

- Requirements fail to reflect the real needs of the customer
- Requirements are inconsistent, incomplete and/or contradictory
- Increases in cost, delays in schedule
  - It is expensive to make changes to requirements after they have been agreed (contractual impediment)
- Misunderstandings and/or misinterpretations of the requirements between customers, those developing system requirements and the people developing or maintaining the system
- Lack of technical knowledge can lead to requirements that are unreasonably expensive to implement

Module topics

- The system development process
- The consequences of poorly-written requirements
- **The use of requirements in the**
  - **A. Customer Needs Identification States**
  - B. System Requirements State
  - C. Subsystem Design State
  - D. Subsystem Construction State
  - E. Subsystem Testing State
  - F. System Integration and System Testing States
  - G. Operations, Maintenance and Upgrade (O&M) States
  - H. System Disposal State
- The requirements and change management processes once the initial set of requirements have been accepted
- The place of requirements in the Agile paradigm (software)
- Exercises

The Needs Identification State

- Is missing in the 'B' paradigm
  - Starts in column B of the HKM²F
- Begins with an undesirable situation (and the funding to do something about it)
- Addresses the undesirability and causes
- Determines the FCFDS
- Comprises the following sub-States:
  1. The exploring the problem sub-State
  2. The conceptualizing the solution sub-State
  3. The system architecting sub-State.
- Ends at the Operations Concept Review (OCR)

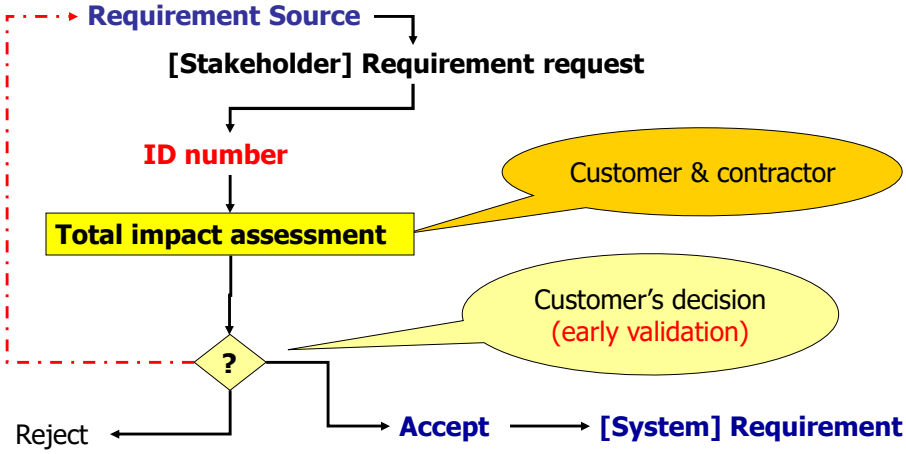
Modules 2 - 5

2. Stakeholders and their importance
3. Identifying the stakeholders' wants
4. Converting stakeholder wants to needs
5. Documenting stakeholders' needs

Module topics

- The system development process
- The consequences of poorly-written requirements
- **The use of requirements in the**
  - A. Customer Needs Identification States
  - **B. System Requirements State**
  - C. Subsystem Design State
  - D. Subsystem Construction State
  - E. Subsystem Testing State
  - F. System Integration and System Testing States
  - G. Operations, Maintenance and Upgrade (O&M) States
  - H. System Disposal State
- The requirements and change management processes once the initial set of requirements have been accepted
- The place of requirements in the Agile paradigm (software)
- Exercises

Requirements processing



The System Requirements State

- **'A' Paradigm**
- Develops an understanding of the situation (S1) [in Needs State]
  - Undesirability and causes
- Develops the FCFDS (S3)
- Creates the CONOPS from the FCFDS
- Creates
  - **Requirements from the CONOPS**
    - matched set of system and subsystem specifications

- **'B' Paradigm**
- Elicits and elucidates requirements
  - To (try to) develop an understanding of the situation
- Creates
  - **The requirements from information provided by stakeholders**
    - matched set of system and subsystem specifications
  - **The CONOPS from requirements**

- Creates (as applicable) the information in
  - The Systems Engineering Plan (SEP)
  - The Systems Engineering Management Plan (SEMP)
  - The Test and Evaluation Master Plan (TEMP)
- Ends at System Requirements review (SRR)

Modules 6-8

6. Converting stakeholder needs to requirements
7. Converting requirements to well-written requirements
8. Converting well-written requirements to good requirements

Module topics

- The system development process
- The consequences of poorly-written requirements
- **The use of requirements in the**
  - A. Customer Needs Identification States
  - B. System Requirements State
  - **C. Subsystem Design State**
  - D. Subsystem Construction State
  - E. Subsystem Testing State
  - F. System Integration and System Testing States
  - G. Operations, Maintenance and Upgrade (O&M) States
  - H. System Disposal State
- The requirements and change management processes once the initial set of requirements have been accepted
- The place of requirements in the Agile paradigm (software)
- Exercises

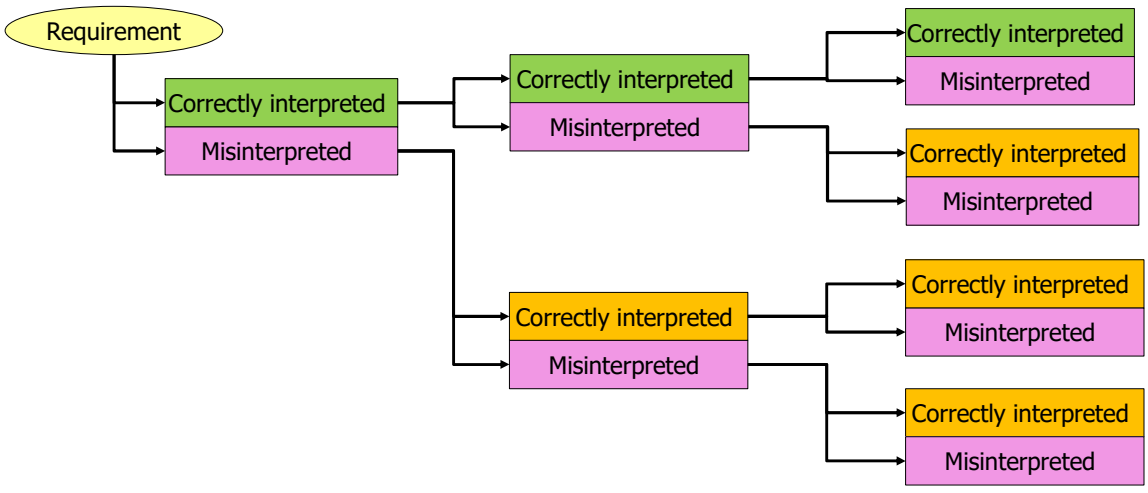
The System Design State

- Begins at end of the System Requirements Review (SRR)
- Converts the matched set of requirement specifications to a preliminary design
  - According to the process conceived in the SEMP and TEMP
  - Derive or elaborate subsystem requirements
  - Create candidate solution system architectures (designs)
  - Selects best one
- Intermediate milestone
  - Preliminary Design Review (PDR)
- Creates a more detailed design of the whole solution system
  - through a combination of people, doctrine, parts, subsystems, interactions, etc., including configuration, architecture and implementation criteria
- Ends at Critical Design Review (CDR)

Opportunities for misinterpretation

- When interpreting the requirements
- Software
  - Converts functional requirements to scenarios (use cases)
- Hardware
  - Creates several conceptual designs in parallel
  - Selects the best one
- Updates matched set of subsystem specifications
- Testing
  - Develops system test plans to verify the system meets its acceptance criteria

Error propagation in the SDP



Update Requirements Traceability Matrix (RTM)

- Add information in RTM to show traceability to requirements, including
  - Planned Hardware part numbers
  - Planned Software identification
  - Schematics
  - Draft system integration and test plans
  - Impact of approved changes

Module topics

- The system development process
- The consequences of poorly-written requirements
- **The use of requirements in the**
  - A. Customer Needs Identification States
  - B. System Requirements State
  - C. Subsystem Design State
  - **D. Subsystem Construction State**
  - E. Subsystem Testing State
  - F. System Integration and System Testing States
  - G. Operations, Maintenance and Upgrade (O&M) States
  - H. System Disposal State
- The requirements and change management processes once the initial set of requirements have been accepted
- The place of requirements in the Agile paradigm (software)
- Exercises

The Subsystem Construction State

- Starts at the end of the Critical Design Review (CDR)
- Creates the individual parts, subsystems, interactions, etc. *in isolation.*
  - Consequently the set of activities are mainly engineering, training, etc., not systems engineering.
- The role of the system systems engineer is to make sure
  - that the parts or subsystems contribute to the whole or system
  - when completed and integrated, the system will remedy the undesirable situation that will exist when the system is placed into service.
- Ends at the Test Readiness Review (TRR)
  - when the Subsystems are ready for testing

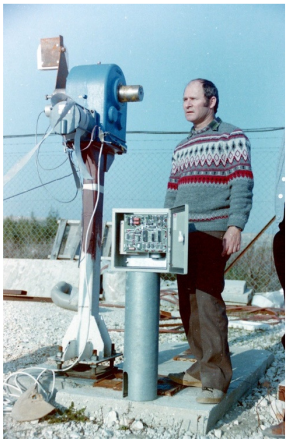
Update Requirements Traceability Matrix (RTM)

- Update information in RTM to show traceability to requirements, including
  - Actual hardware part numbers
  - Actual software identification
  - Schematics
  - Actual (approved) system integration and test plans
  - Draft system integration and test procedures
  - Impact of approved changes

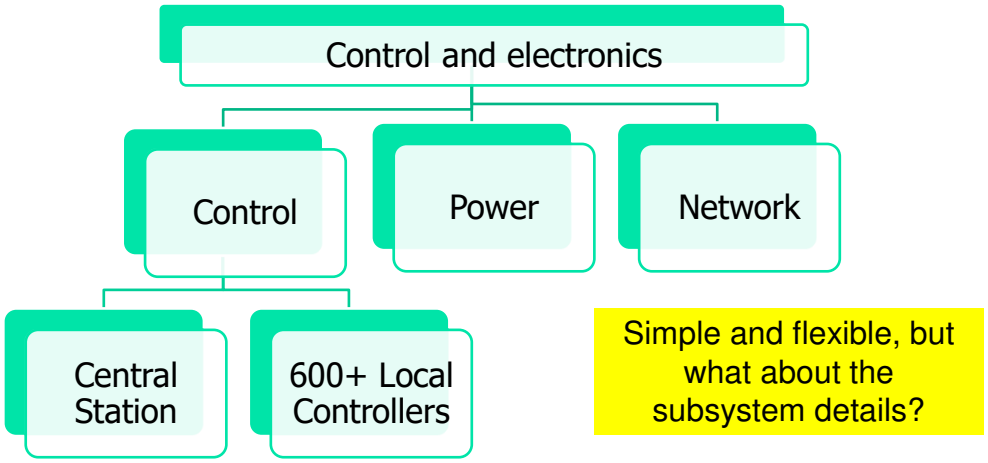
Luz SEGS-1 during constructions/installation

Field of mirrors under construction

Size it



System architecture



Requirements (written text statements)

- Mirror pointing accuracy = ±0.2 degrees
- System level
  1. In operation, SEGS-1 shall generate more power than it uses
  2. SEGS-1 shall generate the maximum possible amount of power each day
- Subsystem level
  1. Cable interface specifications
  2. AC power

Module topics

- The system development process
- The consequences of poorly-written requirements
- **The use of requirements in the**
  - A. Customer Needs Identification States
  - B. System Requirements State
  - C. Subsystem Design State
  - D. Subsystem Construction State
  - **E. Subsystem Testing State**
  - F. System Integration and System Testing States
  - G. Operations, Maintenance and Upgrade (O&M) States
  - H. System Disposal State
- The requirements and change management processes once the initial set of requirements have been accepted
- The place of requirements in the Agile paradigm (software)
- Exercises

The Subsystem-Testing State

- Begins at the end of the TRR
  - Subsystems ready to be tested
- Validates the performance of the individual parts, subsystems, etc. *in isolation* against their requirements
- Ends at the Integration Readiness Review (IRR)
  - when the Subsystems have passed Subsystem testing and are ready to be integrated into a system

Update Requirements Traceability Matrix (RTM)

- Update design information in RTM to show traceability to requirements, including
  - Impact of approved changes
  - Impact of subsystem test results
    - Possible changes to subsystem requirements

Module topics

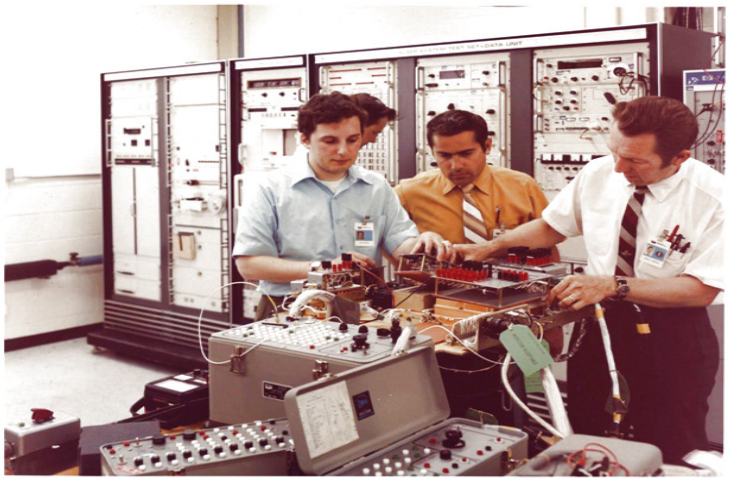
- The system development process
- The consequences of poorly-written requirements
- **The use of requirements in the**
  - A. Customer Needs Identification States
  - B. System Requirements State
  - C. Subsystem Design State
  - D. Subsystem Construction State
  - E. Subsystem Testing State
  - **F. System Integration and System Testing States**
  - G. Operations, Maintenance and Upgrade (O&M) States
  - H. System Disposal State
- The requirements and change management processes once the initial set of requirements have been accepted
- The place of requirements in the Agile paradigm (software)
- Exercises

The System Integration and Testing State

- Begins at the end of the IRR
- Combines the parts, subsystems, interactions, etc., to constitute the solution system
- Establishes, under test conditions, the performance of the whole solution system, with optimum effectiveness, in its operational context
- Requirements for test equipment
- Subsystem requirements for test points to facilitate testing
- Ends at the Delivery Readiness Review (DRR)
- May be sequenced if subsystems are ready at different times

Testing ALSEP prototype

- Requirements to test integrated system
- Requirements for test equipment (which also have to be tested)



Update Requirements Traceability Matrix (RTM)

- Update information in RTM to show traceability to requirements, including
  - Actual system integration and test procedures
  - System integration and test result documents
  - Test results
  - Impact of approved changes

Testing poorly written requirements

"204.1 DADS shall automatically maintain *statistics* concerning the number of times and the most recent time that each data set has been accessed. These same statistics shall be maintained for each *piece of media* in the DADS archive" (ST-DADS 1992)

- The requirement contains:
  - ***Undefined terms:*** statistics, piece of media.
  - ***Multiple requirements:*** and, two sentences.

Example 2 [ST-DADDS]

"204.1 DADS shall automatically maintain *statistics* concerning the number of times and the most recent time that each data set has been accessed. These same statistics shall be maintained for each *piece of media* in the DADS archive" (ST-DADS 1992)

- The requirement contains:
  - ***Undefined terms:*** statistics, piece of media.
  - ***Multiple requirements:*** and, two sentences.

Creating four atomic requirements

204.1a DADS shall automatically maintain statistics concerning the number of times ~~and the most recent time~~ that each data set has been accessed. ~~These same statistics shall be maintained for each piece of media in the DADS archive~~.

204.1b DADS shall automatically maintain statistics concerning the ~~number of times and~~ the most recent time that each data set has been accessed. ~~These same statistics shall be maintained for each piece of media in the DADS archive.~~

204.1c DADS shall automatically maintain statistics concerning the number of times ~~and the most recent time~~ that ~~each data set has been accessed. These same statistics shall be maintained for~~ each piece of media in the DADS archive has been accessed.

204.1d DADS shall automatically maintain statistics concerning the ~~number of times and~~ the most recent time that ~~each data set has been accessed. These same statistics shall be maintained for~~ each piece of media in the DADS archive has been accessed.

Four atomic requirements

204 Access to data sets

204.1a DADS shall automatically maintain statistics concerning the number of times that each data set has been accessed.

204.1b DADS shall automatically maintain statistics concerning the most recent time that each data set has been accessed.

204 Access to each piece of media

204.1c DADS shall automatically maintain statistics concerning the number of times that each piece of media in the DADS archive has been accessed.

204.1d DADS shall automatically maintain statistics concerning the most recent time that each piece of media in the DADS archive has been accessed.

Four atomic requirements

204 Access to data sets

204.1a DADS shall automatically maintain statistics concerning the number of times that each data set has been accessed.

204.1b DADS shall automatically maintain statistics concerning the most recent time that each data set has been accessed.

Statistics

204.1c DADS shall automatically maintain statistics concerning the number of times that each piece of media in the DADS archive has been accessed.

204.1d DADS shall automatically maintain statistics concerning the most recent time that each piece of media in the DADS archive has been accessed.

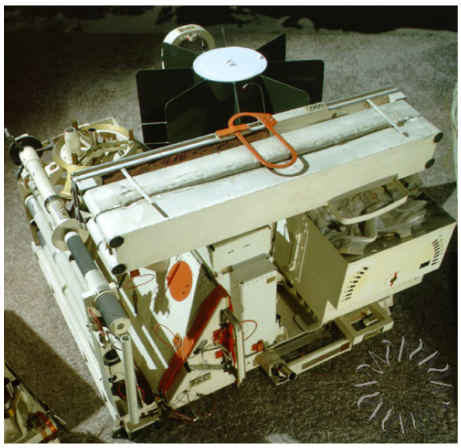
Module topics

- The system development process
- The consequences of poorly-written requirements
- **The use of requirements in the**
  - A. Customer Needs Identification States
  - B. System Requirements State
  - C. Subsystem Design State
  - D. Subsystem Construction State
  - E. Subsystem Testing State
  - F. System Integration and System Testing States
  - **G. Operations, Maintenance and Upgrade (O&M) States**
  - H. System Disposal State
- The requirements and change management processes once the initial set of requirements have been accepted
- The place of requirements in the Agile paradigm (software)
- Exercises

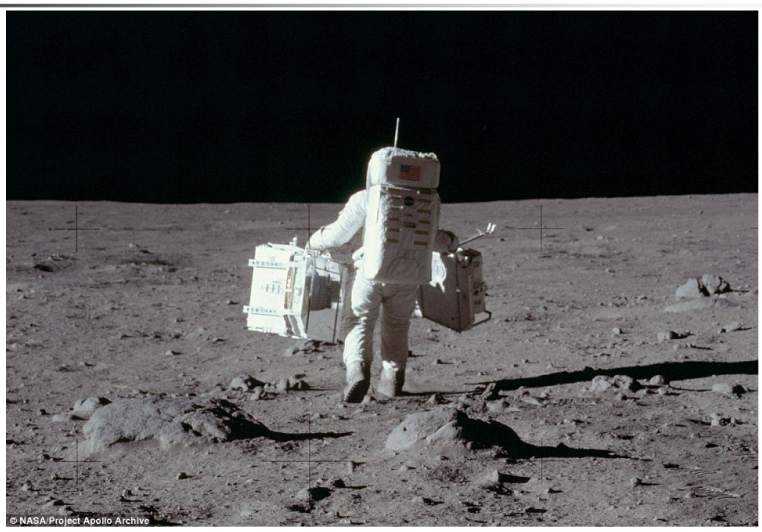
The Operations and Maintenance State

- Begins after the DRR
- Contains the Delivery, Acceptance Test Review (ATR) and Deployment sub-States
- Actively provides a remedy to the undesirable situation for which the whole system (S6) was created including:
  - Operating the system
  - Support to maintain operations
  - Improving the whole to enhance effectiveness, and to accommodate changes in the nature of the undesirable situation over time
- Ends when:
  - The solution system is no longer capable of remedying the undesirable situation effectively or economically

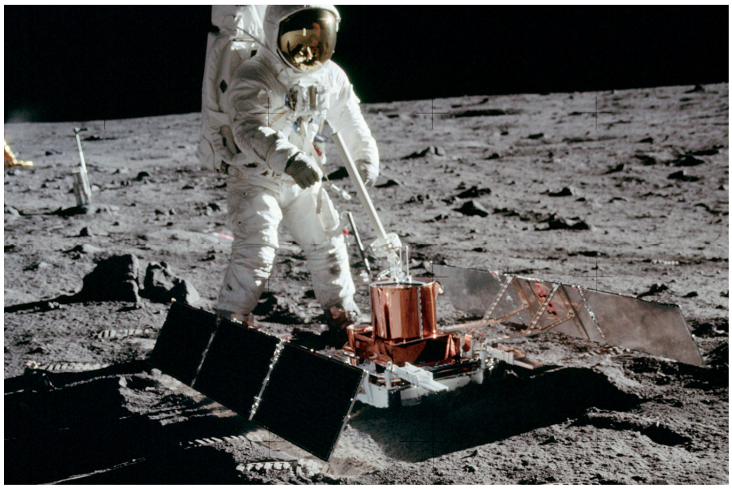
Apollo Lunar Surface Experiment Package (ALSEP)



Carrying the ALSEP to deployment location

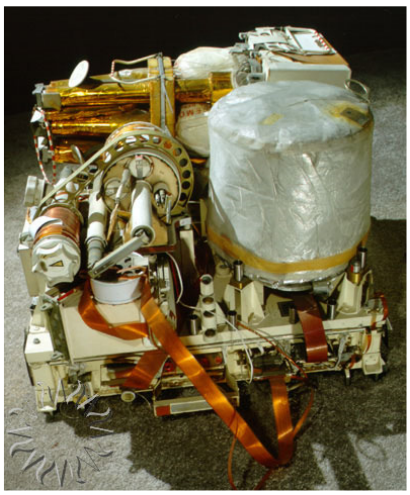


Seismometer experiment



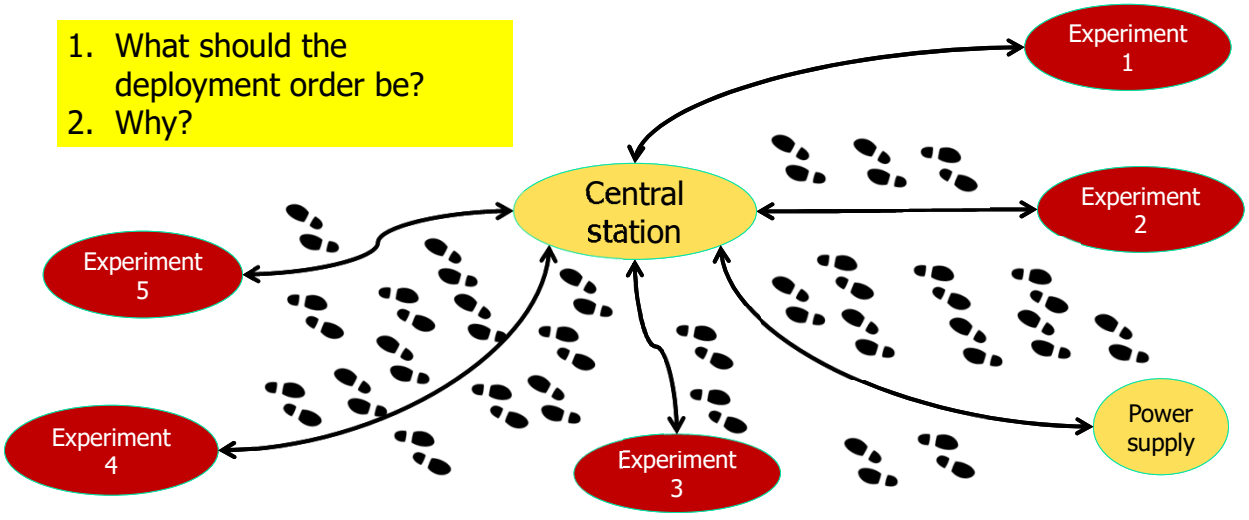
Traditional view of systems integration

- Each experiment is integrated into the assembly without impacting another
- Each experiment can be removed from the assembly without impacting another
- Each experiment is then deployed by being carried to its location trailing the cable



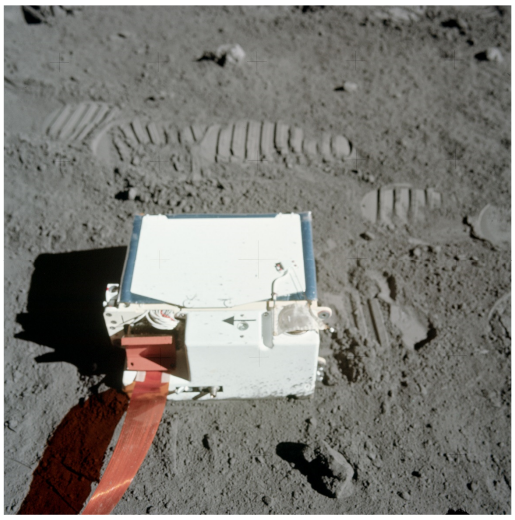
ALSEP deployment?

1. What should the deployment order be?
2. Why?



Astronaut broke a cable

- Misuse deployment scenario
  - Tripped on cable and broke it
- Risk, if identified
  - High severity of impact (5)
  - Low probability of occurrence (1)
  - Users of traditional risk matrix would ignore the risk
- It happened
- Problem
  - How to repair the cable?



Astronaut broke a cable

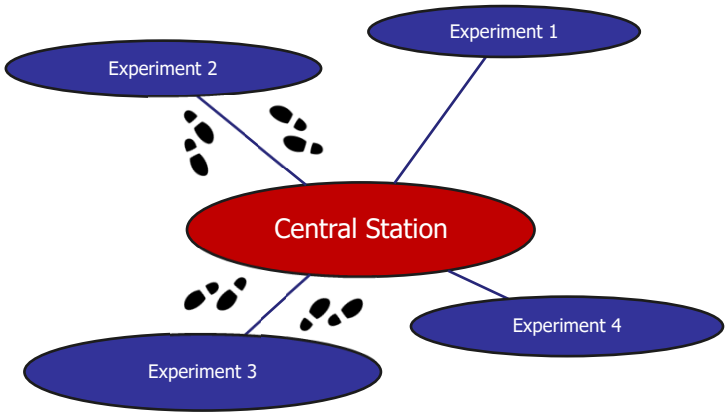
- Misuse deployment scenario
  - Tripped on cable and broke it
- Risk, if identified
  - High severity of impact
  - Low probability of occurrence
  - Users of traditional risk matrix would ignore the risk
- It happened
- Problem
  - How to repair the cable?
- Constraint
  - With equipment on the moon
- Solution
  - Could not be achieved in time

The deployment requirement

- ~~When deploying the experiments, the astronaut shall not step over the cable~~
- Do not write a requirement to walk around a deployed cable to avoid tripping on it and breaking the cable
  - It probably won't be met in the event the astronaut needs to do it
- Write requirements (procedure) for integration to minimize probability of tripping over the cable
  - Visualize the scenario
- One way to turn a negative need into a set of positive requirements

ALSEP integration scenario

- Unpack from palette shown earlier
- Deploy experiments on surface location per layout plan
- Write procedure to connect cables in a sequence that avoids stepping over a connected cable
- Walk out on one side of cable, walk around (set up) experiment, walk back on other side
- Note simple layout plan representation for stakeholder communications



Module topics

- The system development process
- The consequences of poorly-written requirements
- **The use of requirements in the**
  - A. Customer Needs Identification States
  - B. System Requirements State
  - C. Subsystem Design State
  - D. Subsystem Construction State
  - E. Subsystem Testing State
  - F. System Integration and System Testing States
  - G. Operations, Maintenance and Upgrade (O&M) States
  - **H. System Disposal State**
- The requirements and change management processes once the initial set of requirements have been accepted
- The place of requirements in the Agile paradigm (software)
- Exercises

The System Disposal State

- Begins when:
  1. The solution system is no longer capable of meeting the stakeholders' needs effectively or economically
- Contains the activities that dispose of the system
- Reverse of the SDP
- Ends when the system has been disposed

Requirements ?

- It depends
- Complexity of disposal project
- Options (different requirements)
  - Abandon
    - Walk away and leave in place
  - Dump
    - Transport to a facility that can salvage or store components
  - Sell
    - Find a third party who will purchase the system
  - Contractor
    - Pay someone to dispose of system
      - outsource problem
  - Others
    - Not mentioned above, combinations, etc.

Module topics

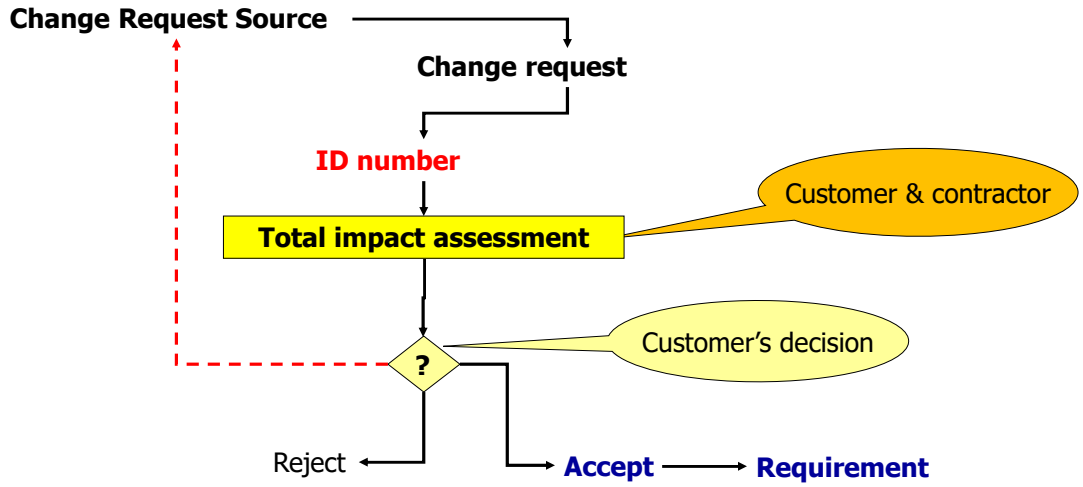
- The system development process
- The consequences of poorly-written requirements
- The use of requirements in the
  - A. Customer Needs Identification States
  - B. System Requirements State
  - C. Subsystem Design State
  - D. Subsystem Construction State
  - E. Subsystem Testing State
  - F. System Integration and System Testing States
  - G. Operations, Maintenance and Upgrade (O&M) States
  - H. System Disposal State
- **The requirements and change management processes once the initial set of requirements have been accepted**
- The place of requirements in the Agile paradigm (software)
- Exercises

The three effects of a change

1. New functions and requirements are added
2. Existing functions and requirements are deleted
3. Both of the above

- Each impacts
  1. Technical performance
  2. Cost
  3. Schedule (time to complete)

Change Processing



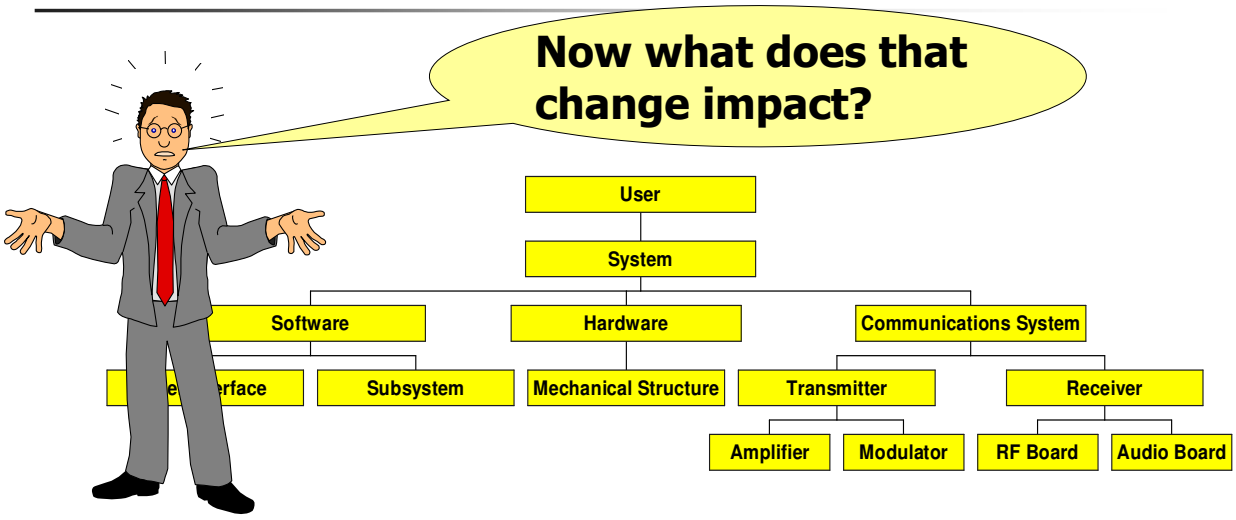
Impact Assessment

- *Determine if the request has been rejected before and if those reasons are still applicable*
- Determine if a conflict or contraction exists
- Estimate cost to implement
- Determine cost drivers
- Perform sensitivity analysis
- Discuss cost drivers with customers
  - Are cost drivers really necessary?
  - Document decisions in requirement repository

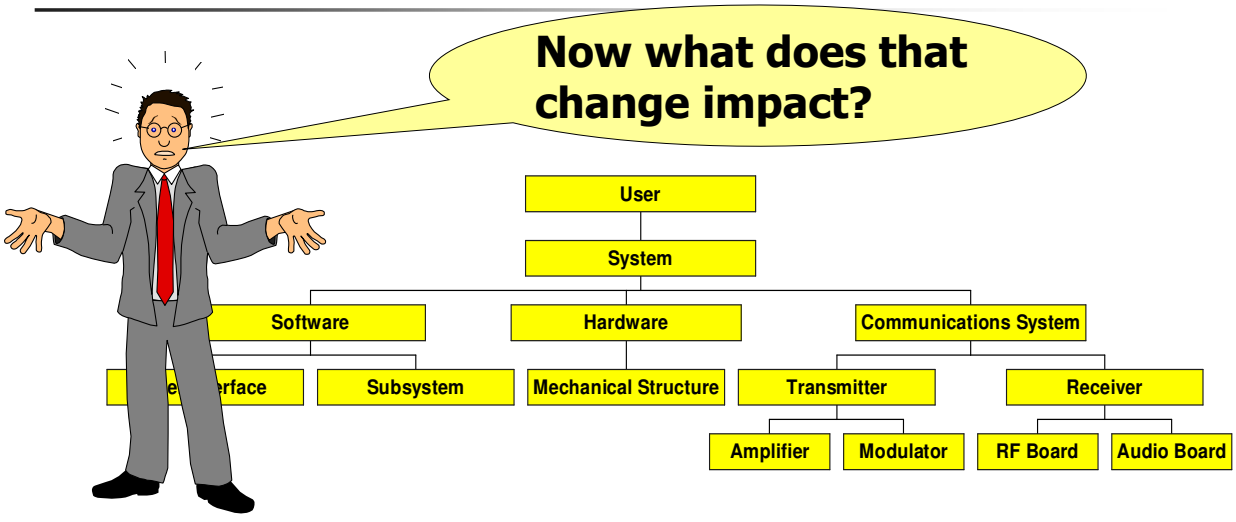
Typical impact assessment questions

- Why do we need the change?
- What if we don't accept it?
- What are the alternatives?
- **What will the modified system do? (CONOPS)**
- How will the change contribute to mission objectives?
- How will the change impact existing **and planned** adjacent systems?
- What are the risks and their probability of occurrence?
- What are the required resources required to implement the change?
- How long will it take to implement the change?

Importance of the requirements documentation tree



Importance of the Requirements Traceability Matrix



Information System Perspective

- **Programs do not fail because the requirements change**
  - Tasks, products and processes exist.
- **Programs fail because of poor requirements management**
  - The ability to write a good requirements document is not enough
  - The ability to meet stakeholders' needs during the SDP is not enough

Module topics

- The system development process
- The consequences of poorly-written requirements
- The use of requirements in the
  - A. Customer Needs Identification States
  - B. System Requirements State
  - C. Subsystem Design State
  - D. Subsystem Construction State
  - E. Subsystem Testing State
  - F. System Integration and System Testing States
  - G. Operations, Maintenance and Upgrade (O&M) States
  - H. System Disposal State
- The requirements and change management processes once the initial set of requirements have been accepted
- **The place of requirements in the Agile paradigm (software)**
- Exercises

Waterfall vs. Agile (*Continuum* HTP)

Waterfall

- (Traditional SLC)
- Used in bespoke (single customer) system development
- Builds to specification
- Works well, when the baseline needs are well-defined and the needs do not change (or change slowly) during the SDP
- Some needs may change during O&M state
  - Waterfall is iterative once system is deployed (cataracts) in O&M state

Agile

- Used in mass market (many customer) (systems and) software product development
- Builds to speculation (hope people will buy it)
- No O&M state once product is released
- Instead, product is upgraded
  - To fix defects
  - Add functions (features)
- Kano model applies
- May also be misused for physical system development

Agile paradigm (with variations, seems to be)

1. Identify a subset of the stakeholders (traditional)
2. Identify some stakeholders' wants ('B' paradigm)
3. Begin
   1. While within cost and schedule constraints for a software Build, do Begin
      1. Model known stakeholder wants (stakeholder requirement requests) using scenarios
      2. Convert the scenarios (use cases) to software
      3. Test the software Build (against known stakeholder wants)
         1. If the functions in the software meet stakeholder wants, keep them
         2. If the functions in the software do not meet stakeholder wants, throw them away, change them or just leave them in the software Build
      4. Package operational (executable) software Build as a product release
   2. End
4. Release product to mass market
5. Accept new customer wants (real new need or changes based on use of the Build)
   1. Does not differentiate between the baseline needs "I forgot" and the real changes in needs during the software development
6. End

The agile paradigm (from the HTPs)

- Worthy attempt to cut out the bumph or unnecessary documentation for a specific type of system/product
- Originated by software engineers (working in the 'B' paradigm?) published in 2001*
- To focus on delivering a product for customers buying a commercial software product
  - Wide range of customer needs
  - No maintenance on existing software, just feature upgrades and bug fixes in new release
- Does not address the problem of systematically and systemically finding a complete set of stakeholders
  - Seems to assume a representation of customers have been found
- Has no way of knowing if the needs are complete (no reference model)
  - May not be needed since products are upgraded
  - (but a more complete set, might reduce time to market and development costs)
- Does not seem to address the Operations and Maintenance States of the traditional SLC
  - Which needs information which may not be developed in the software Agile methodology
  - However, O&M states not applicable since mass market software products are upgraded not maintained

* Sarah Laoyan, What is Agile methodology? (A beginner's guide), October 15th, 2022, https://asana.com/resources/agile-methodology, accessed 8 November 2023

Agile software does not need functional requirements

- Traditional process (part of)
  - Create a set of scenarios (from generic and specific stakeholder requirement requests)
  - Write a set of good functional requirements based on scenarios
  - Create use cases (scenarios) from the set of good functional requirements
  - Create the software from the use cases (scenarios)
- Agile process
  - Create a set of scenarios (from generic and specific stakeholder requirement requests)
  - ~~Write a set of good functional requirements based on scenarios~~
  - ~~Create use cases (scenarios) from the set of good functional requirements~~
  - Create the software from the use cases (scenarios)

Agile alternative: for systems that will be maintained*

- Conventional wisdom
  - Waterfall approach does not cope well with changing requirements
  - Change the software development process from the waterfall approach to some type of rapid, spiral, or other methodology
- Result
  - Not much of an improvement
- Cataract approach for systems that will have an O&M state
  - Mini waterfalls under configuration control

* 0950/1 Kasser J.E., The Cataract Methodology for Systems and Software Acquisition, *proceedings of the SETE 2002,* Sydney, Australia, 2002

Module topics

- The system development process
- The consequences of poorly-written requirements
- The use of requirements in the
  - A. Customer Needs Identification States
  - B. System Requirements State
  - C. Subsystem Design State
  - D. Subsystem Construction State
  - E. Subsystem Testing State
  - F. System Integration and System Testing States
  - G. Operations, Maintenance and Upgrade (O&M) States
  - H. System Disposal State
- The requirements and change management processes once the initial set of requirements have been accepted
- The place of requirements in the Agile paradigm (software)
- **Exercises**

Exercise 9-1 impact of a change request

- **The change request (just before PDR)**

  R513.2 At least 60% of the replacement system shall be constructed from parts incorporated in the current system

  Source of request: Mr Bean Counter in finance department

  Reason for request : Cost savings by using components already developed

- **Affected requirement**

  503.1 After the replacement system is put into operation, the current system shall be operated in parallel with the replacement system for at least 30 continuous days

Exercise 9-1

1. Consider the two requirements in the previous page (503.1 and 513.2)
2. You are the Change Control Board (CCB), what is the impact of the change request?
3. Prepare a <5 minute presentation containing
   1) This slide and version number of Module
   2) The assumptions you made
   3) A summary of the impact of the change
   4) Accepted or rejected
   5) If accepted, the changed requirement(s)
   6) A compliance matrix for the exercise
   7) Lessons learned from exercise
   8) The problem posed by the exercise formulated per COPS problem formulation template
4. Save as a PowerPoint file in format Exercise9-1-abcd.pptx
5. Post/email presentation as and where instructed

Exercise 9-2 knowledge reading

1. Prepare a brief on two main points on reading 0902 (< 5min)
2. Presentation to contain
    1. Formulated problem per COPS problem formulation template
    2. A summary of the content of the reading (<1 minute)
    3. The compliance matrix
    4. This slide and version number of Module
    5. The main points
    6. The two briefings
    7. Reflections and comments on reading (<2 minute)
    8. Comparisons of content with other readings and external knowledge
    9. Why you think the reading was assigned to the module
    10. Lessons learned from module and source of learning e.g. readings, exercise, experience, etc. (<2 minutes)
3. Save as a PowerPoint file as Exercise9.2-abcd.pptx
4. Post/email presentation as, when and where instructed
5. Brief on one main point

Meeting the objectives

#	Objectives	Met
1	To summarize the System Development Process (SDP and the System LifeCycle (SLC)	12-21
2	To expose participants to the consequences of poorly-written requirements in the SLC,	23-24
3	by explaining the use of requirements in the Needs Identification and System Requirements States	26-27
4	by explaining the use of requirements in the System Design State	29-31
5	by explaining the use of requirements in the Subsystem Construction and Subsystem Test States	33-47
6	by explaining the use of requirements in the System Integration and System Test States	49-51
7	by explaining the use of requirements in the System Operations, Maintenance and Upgrade States	53-62
8	To explain the requirements and change management processes once the initial set of system requirements have been accepted	66-72
9	To discuss the place of requirements in the Agile paradigm (software)	75-79
10	To practice what has been taught	81-82
11	To provide the opportunity to exercise the 5 levels of knowledge in the updated Blooms taxonomy	81-83

Any questions ?

1. Best
2. Worst
3. Missing

Email: beyondsystemsthinking@yahoo.com
Subject: <class title> BWM Module #