2010

Software Engineering Strengths and Weaknesses in Systems Engineers (presentation)

Shebalin, Paul

October 2010

NAVAL
POSTGRADUATE
SCHOOL

Software Engineering Strengths and Weaknesses in Systems Engineers

Dr. Paul Shebalin, Director
The Wayne E. Meyer Institute, Naval Postgraduate School
pshebali@nps.edu, 831-656-1047

"Applied and Basic Research in Systems Analysis, Modeling and Engineering

Monterey, California
WWW.NPS.EDU

NAVAL POSTGRADUATE SCHOOL

Topics

- SE at the Naval Postgraduate School
- SE4003 Systems Software Engineering
- The Need for Systems Engineers to Serve as Software Engineers
- Software Engineering Capability Evaluation
- Observations
- Recommendations

NAVAL POSTGRADUATE SCHOOL

Mission of the Naval Postgraduate School

- NPS provides high-quality, relevant and unique advanced education and research programs that increase the combat effectiveness of the Naval Services, other Armed Forces of the U.S. and our partners, to enhance our national security.

NAVAL POSTGRADUATE SCHOOL

NPS Summary

- A Department of the Navy graduate school founded in 1909 located in Monterey, California
- Four schools and 65 Curricula in engineering, science, business, public policy, operations research, information sciences, international studies, national security studies and homeland security.
- Four research institutes and 20 centers
- Faculty: 248 Tenure-Track, 429 Non-TT
- Graduate Students: 1500 Resident, 750 Non-Resident
  - 44% Navy, 12% USMC, 23% other US Services, 14% International, 7% Civilian

NPS Systems Engineering Programs

SE Certificate	MSSEA	MSSE	MSSE	MSSEM (PD21)
282	308	311	580	721
DL and resident	Resident	DL	Resident	DL
4 quarters	8 quarters (with embedded refresher)	8 quarters	8 quarters (with refresher)	8 quarters
4 courses	32 courses	16 courses	36 courses	16 courses
Integrated Project (in courses)	Project	Project	Thesis	Thesis
2 cohorts/year	1 cohorts/year	12 cohorts/year	1 cohort/year*	1 cohort/year
30 per cohort	20 per cohort	~30 per cohort	18 per cohort	20-25 per cohort
23 students	34 students	314 students	43 onboard	41 students

NAVAL POSTGRADUATE SCHOOL

MSSE Core Courses

- Resident and non-resident programs share a common nine course core curriculum
- Informed by INCOSE and DOD reference curricula
- DAU Equivalencies
- Burnt orange courses compose the certificate
- Degree requirements met by core, 4 course track, and 3 course project
- P-codes can impose additional requirements

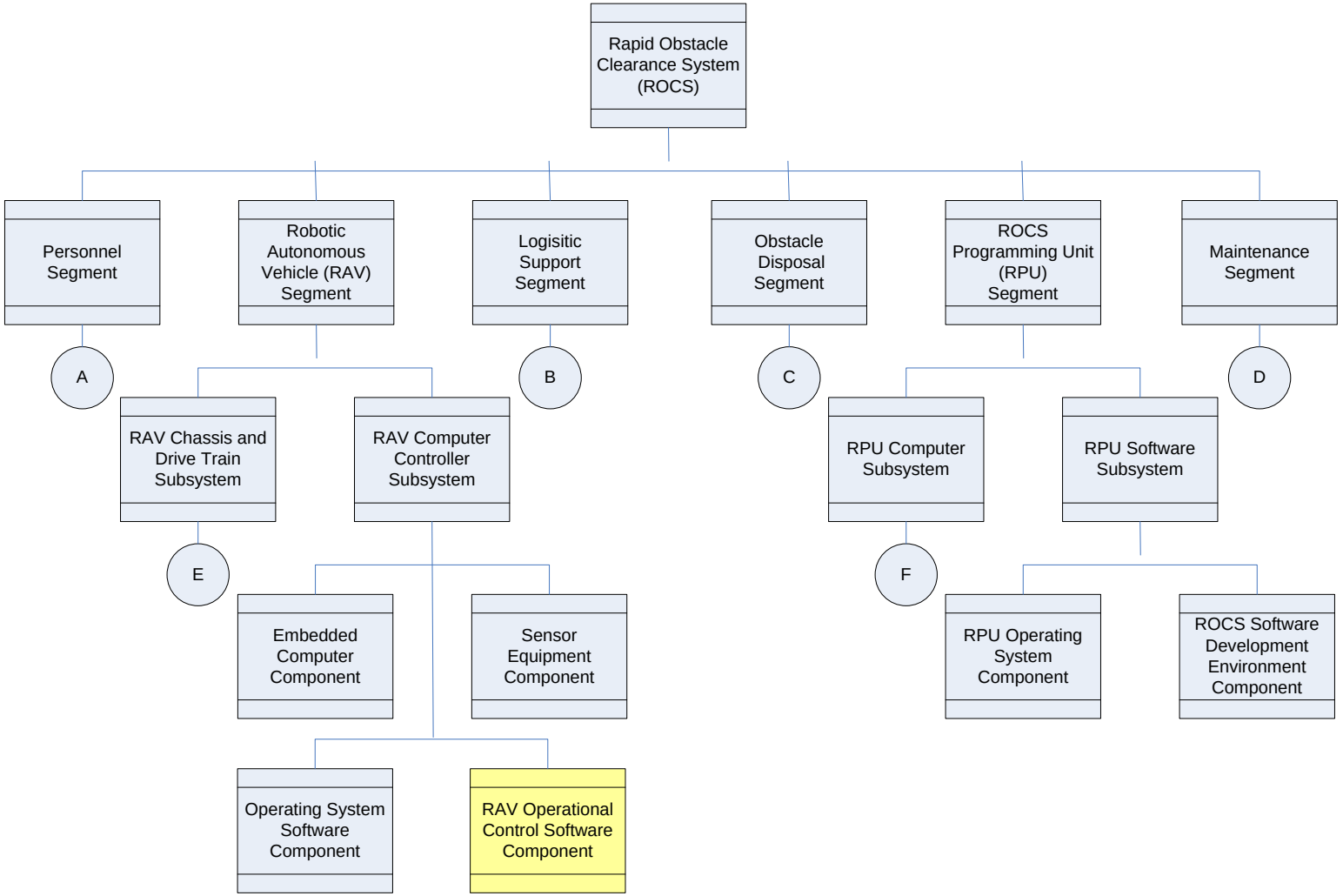
Fundamentals of Systems Engineering
Systems Suitability
Systems Assessment
Fundamentals of Engineering Project Management
Engineering Economics and Cost Estimation
Capability Engineering
System Architecture and Design
Systems Software Engineering
Systems Integration and Development

NAVAL POSTGRADUATE SCHOOL

SE4003 Systems Software Engineering

- Course objective: *teach students the basic concepts of software engineering and methods for requirements, definition, design and testing of software.*
- Course framework:
  - 10-week quarter
  - Prerequisite: Computer Programming Course
  - Text by Pressman: *Software Engineering: A Practitioner's Approach (7th Ed.)* (Chapters 1-10, 17-19)
  - Assigned readings and class presentations, exercises and discussions complement hands-on project experience.
  - Embedded System Software Project:
    - Team of 3-5 members
    - Software development for Lego NXT robot using NXC (Not eXactly C) or Java
    - Deliverable and non-deliverable software products
- Basis for identifying SWE strengths and weaknesses

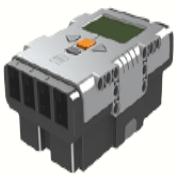
ROCS System Hierarchy



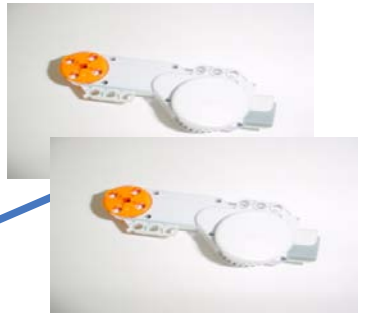
Hands-On Project Hardware

Robotic Autonomous Vehicle (RAV)
Prototype

LEGO NXT Ultrasonic Sensor



LEGO NXT Brick



Two (2) LEGO NXT Servo Motors

LEGO NXT Light Sensor

LEGO NXT Light Sensor

Systems Engineers as Software Engineers?

- Why do systems engineers need software engineering knowledge, skills, and abilities?
  - To be better systems engineers?
  - To be better software engineers?
- Many systems engineers will be called to serve as software engineers on a software project – development, maintenance, V&V, T&E, …
- How software-engineering-capable are the systems engineers we're graduating?
- How can we measure SWE capability?

Measuring SWE Capability

- Formal testing
- Peer evaluation
- Process quality
- Product quality
- Timeliness and appropriateness
- Repeatability and long-term performance
- Observation and evaluation 

NAVAL POSTGRADUATE SCHOOL

Observation and Evaluation Method

- Define evaluation criteria using
  - "Knowledge, Skills, and Abilities" (KSAs)
  - SWEBOK (IEEE) breakdown of SWE topics
- Consider the results of five offerings of SE4003 with 82 students and 27 project teams
- Subjectively, evaluate the student performance through the use of a Pugh Matrix
  - -1, 0, 1 representing Worse, Same, Better of average student performance compared with that expected of a competent software engineer

KSA's

- Skills
  - Effectiveness in using tools and technology
  - E.g., effectiveness in using a specific software development environment (SDE)

- Abilities
  - The wherewithal to perform specific tasks
  - E.g., the ability to create a suitable Software Requirements Analysis Document

- Knowledge
  - Mental data representing information, facts, relationships, concepts, logical implications, etc.
  - E.g., object-oriented software engineering concepts

NAVAL POSTGRADUATE SCHOOL

SWEBOK Topics Breakdown

	Area	Subareas	Topics	
A.	SW Requirements	7	28	
B.	Software Design	6	25	Areas Evaluated
C.	Software Construction	3	13	
D.	Software Testing	5	16	Subtopics
E.	Software Maintenance	4	15	13
F.	SW Configuration Mgt	6	17	15
G.	SW Engineering Mgt	6	24	
H.	SW Engineering Process	4	16	9
I.	SWE Tools and Methods	2	14	
J.	Software Quality	3	11	14

A. Software Requirements

1. Software requirements fundamentals
   a. Definition of software requirement
   b. Product and process requirements
   c. Functional and non-functional requirements
   d. Emergent properties
   e. Quantifiable requirements
   f. System requirements and software requirements
2. Requirements process
   a. Process models
   b. Process actors
   c. Process support and management
   d. Process quality and improvement
3. Requirements elicitation
   a. Requirements sources
   b. Elicitation techniques
4. Requirements analysis
   a. Requirements classification
   b. Conceptual modeling
   c. Architectural design and requirements allocation
   d. Requirements negotiation
5. Requirements specification
   a. System definition document
   b. System requirements specification
   c. Software requirements specification
6. Requirements validation
   a. Requirements reviews
   b. Prototyping
   c. Model validation
   d. Acceptance tests
7. Practical considerations
   a. Iterative nature of requirements process
   b. Change management
   c. Requirements attributes
   d. Requirements tracing
   e. Measuring Requirements

B. Software Design

1. Software design fundamentals
   a. General design concepts
   b. Context of software design
   c. Software design process
   d. Enabling techniques
2. Key issues in software design
   a. Concurrency
   b. Control and handling of events
   c. Distribution of components
   d. Error and exception handling and fault tolerance
   e. Interaction and presentation
   f. Data persistence
3. Software structure and architecture
   a. Architectural structures and viewpoints
   b. Architectural styles (macroarchitectural patterns)
   c. Design patterns (microarchitectural patterns)
   d. Families of programs and frameworks
4. Software design quality analysis and evaluation
   a. Quality attributes
   b. Quality analysis and evaluation techniques
   c. Measures
5. Software design notations
   a. Structural descriptions (static)
   b. Behavioral descriptions (dynamic)
6. Software design strategies and methods
   a. General strategies
   b. Function-oriented (structured) design
   c. Object-oriented design
   d. Data-structure centered design
   e. Component-based design (CBD)
   f. Other methods

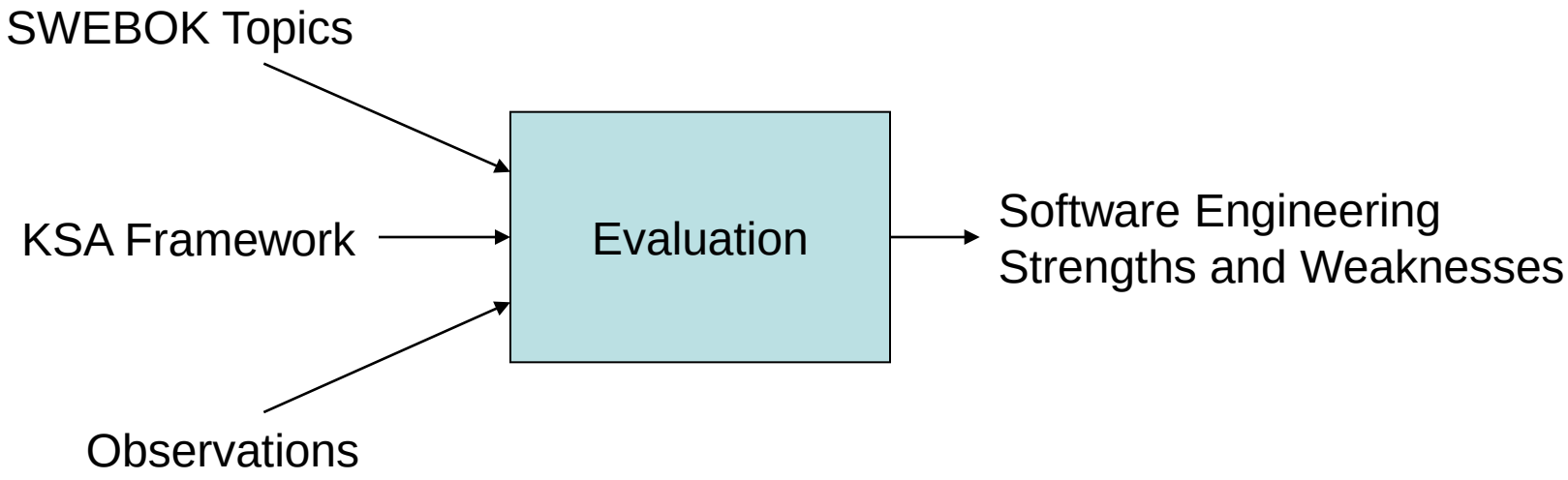
C. Software Construction

1. Software construction fundamentals
   a. Minimizing complexity
   b. Anticipating change
   c. Construction for verification
   d. Standards in construction
2. Managing construction
   a. Construction methods
   b. Construction planning
   c. Construction measurement
3. Practical considerations
   a. Construction design
   b. Construction languages
   c. Coding
   d. Construction testing
   e. Construction quality
   f. Integration

D. Software Testing

1. Software testing fundamentals
   a. Testing-related terminology
   b. Key issues
   c. Relationships of testing to other activities
2. Test levels
   a. The target of the tests
   b. Objectives of testing
3. Test techniques
   a. Based on tester's intuition and experience
   b. Specification-based
   c. Code-based
   d. Fault-based
   e. Usage-based
   f. Based on nature of application
   g. Selecting and combining techniques
4. Test-related measures
   a. Evaluation of the program under test
   b. Evaluation of the tests performed
5. Test process
   a. Management concerns
   b. Test activities

Process Summary



When assigned a software engineering role:

What level of software engineering knowledge would the systems engineer have with regard to a SWEBOK topic?

What level of software engineering ability would the systems engineer have with regard to a SWEBOK topic?

NAVAL POSTGRADUATE SCHOOL

A. Software Requirements: Knowledge

1. Software requirements fundamentals
   a. Definition of software requirement
   b. Product and process requirements
   c. Functional and non-functional requirements
   d. Emergent properties
   e. Quantifiable requirements
   f. System requirements and software requirements
2. Requirements process
   a. Process models
   b. Process actors
   c. Process support and management
   d. Process quality and improvement
3. Requirements elicitation
   a. Requirements sources
   b. Elicitation techniques
4. Requirements analysis
   a. Requirements classification
   b. Conceptual modeling
   c. Architectural design and requirements allocation
   d. Requirements negotiation
5. Requirements specification
   a. System definition document
   b. System requirements specification
   c. Software requirements specification
6. Requirements validation
   a. Requirements reviews
   b. Prototyping
   c. Model validation
   d. Acceptance tests
7. Practical considerations
   a. Iterative nature of requirements process
   b. Change management
   c. Requirements attributes
   d. Requirements tracing
   e. Measuring Requirements

Strength

Weakness

NAVAL POSTGRADUATE SCHOOL

A. Software Requirements: Ability

1. Software requirements fundamentals
   a. Definition of software requirement
   b. Product and process requirements
   c. Functional and non-functional requirements
   d. Emergent properties
   e. Quantifiable requirements
   f. System requirements and software requirements
2. Requirements process
   a. Process models
   b. Process actors
   c. Process support and management
   d. Process quality and improvement
3. Requirements elicitation
   a. Requirements sources
   b. Elicitation techniques
4. Requirements analysis
   a. Requirements classification
   b. Conceptual modeling
   c. Architectural design and requirements allocation
   d. Requirements negotiation
5. Requirements specification
   a. System definition document
   b. System requirements specification
   c. Software requirements specification
6. Requirements validation
   a. Requirements reviews
   b. Prototyping
   c. Model validation
   d. Acceptance tests
7. Practical considerations
   a. Iterative nature of requirements process
   b. Change management
   c. Requirements attributes
   d. Requirements tracing
   e. Measuring Requirements

Strength

Weakness

NAVAL POSTGRADUATE SCHOOL

B. Software Design: Knowledge

1. Software design fundamentals
   a. General design concepts
   b. Context of software design
   c. Software design process
   d. Enabling techniques
2. Key issues in software design
   a. Concurrency
   b. Control and handling of events
   c. Distribution of components
   d. Error and exception handling and fault tolerance
   e. Interaction and presentation
   f. Data persistence
3. Software structure and architecture
   a. Architectural structures and viewpoints
   b. Architectural styles (macroarchitectural patterns)
   c. Design patterns (microarchitectural patterns)
   d. Families of programs and frameworks
4. Software design quality analysis and evaluation
   a. Quality attributes
   b. Quality analysis and evaluation techniques
   c. Measures
5. Software design notations
   a. Structural descriptions (static)
   b. Behavioral descriptions (dynamic)
6. Software design strategies and methods
   a. General strategies
   b. Function-oriented (structured) design
   c. Object-oriented design
   d. Data-structure centered design
   e. Component-based design (CBD)
   f. Other methods

Strength
Weakness

NAVAL POSTGRADUATE SCHOOL

B. Software Design: Ability

1. Software design fundamentals
   a. General design concepts
   b. Context of software design
   c. Software design process
   d. Enabling techniques
2. Key issues in software design
   a. Concurrency
   b. Control and handling of events
   c. Distribution of components
   d. Error and exception handling and fault tolerance
   e. Interaction and presentation
   f. Data persistence
3. Software structure and architecture
   a. Architectural structures and viewpoints
   b. Architectural styles (macroarchitectural patterns)
   c. Design patterns (microarchitectural patterns)
   d. Families of programs and frameworks
4. Software design quality analysis and evaluation
   a. Quality attributes
   b. Quality analysis and evaluation techniques
   c. Measures
5. Software design notations
   a. Structural descriptions (static)
   b. Behavioral descriptions (dynamic)
6. Software design strategies and methods
   a. General strategies
   b. Function-oriented (structured) design
   c. Object-oriented design
   d. Data-structure centered design
   e. Component-based design (CBD)
   f. Other methods

Strength
Weakness

C. Software Construction: Knowledge

1. Software construction fundamentals
   a. Minimizing complexity
   b. Anticipating change
   c. Construction for verification
   d. Standards in construction
2. Managing construction
   a. Construction methods
   b. Construction planning
   c. Construction measurement
3. Practical considerations
   a. Construction design
   b. Construction languages
   c. Coding
   d. Construction testing
   e. Construction quality
   f. Integration

Strength
Weakness

C. Software Construction: Ability

1. Software construction fundamentals
   a. Minimizing complexity
   b. Anticipating change
   c. Construction for verification
   d. Standards in construction
2. Managing construction
   a. Construction methods
   b. Construction planning
   c. Construction measurement
3. Practical considerations
   a. Construction design
   b. Construction languages
   c. Coding
   d. Construction testing
   e. Construction quality
   f. Integration

Strength
Weakness

D. Software Testing: Knowledge

1. Software testing fundamentals
   a. Testing-related terminology
   b. Key issues
   c. Relationships of testing to other activities
2. Test levels
   a. The target of the tests
   b. Objectives of testing
3. Test techniques
   a. Based on tester's intuition and experience
   b. Specification-based
   c. Code-based
   d. Fault-based
   e. Usage-based
   f. Based on nature of application
   g. Selecting and combining techniques
4. Test-related measures
   a. Evaluation of the program under test
   b. Evaluation of the tests performed
5. Test process
   a. Management concerns
   b. Test activities

Strength
Weakness

D. Software Testing: Ability

1. Software testing fundamentals
   a. Testing-related terminology
   b. Key issues
   c. Relationships of testing to other activities
2. Test levels
   a. The target of the tests
   b. Objectives of testing
3. Test techniques
   a. Based on tester's intuition and experience
   b. Specification-based
   c. Code-based
   d. Fault-based
   e. Usage-based
   f. Based on nature of application
   g. Selecting and combining techniques
4. Test-related measures
   a. Evaluation of the program under test
   b. Evaluation of the tests performed
5. Test process
   a. Management concerns
   b. Test activities

Strength
Weakness

NAVAL POSTGRADUATE SCHOOL

Evaluation Summary

- Strengths in all four evaluated SWEBOK areas
- Weaknesses in first two SWEBOK areas
  - A. Software Requirements
  - B. Software Design
- Knowledge may be satisfactory, but Abililties to perform are the real “proofs of the pudding”
- In general, the typical systems engineering student did well as a “competent software engineer”.

Classroom Observations (1 of 4)

1. Some discussion required to clarify the differences between functional and non-functional requirements (system and SW).
2. Initial definition of top-level use case framework was weak, but then detailed use case descriptions were well done.
3. Allocation of system requirements to software requirements was not easily done.
4. Concept of a “CSCI” required discussion and project experience to understand.

NAVAL POSTGRADUATE SCHOOL

Classroom Observations (2 of 4)

5. Actually scripting a CSCI requirement was surprisingly difficult.
6. Object-oriented class modeling was weak – some students got it quickly, most did not.
7. Structured analysis (Data Flow Diagrams) was not easy, but most students quickly learned.
8. Constructing state transition diagrams (STDs) was surprisingly difficult.

Classroom Observations (3 of 4)

9. Experience with Requirements Traceability Matrices was mixed
   - General understanding and use was strong, but
   - "Atomically" capturing and listing software (CSCI) requirements was weak.
10. Following architecture and design patterns was strong, but creating new patterns was weak.
11. Object-oriented design and construction was bi-modal – some got it, some didn't.

NAVAL POSTGRADUATE SCHOOL

Classroom Observations (4 of 4)

12. Behavior modeling (UML) was mixed
    - Strong: *Activity* and *Swim Lane* Diagrams
    - Weak: *Control Flow*, *Sequence*, and *State Transition* Diagrams.
13. Algorithm design (using PDL) was weak.
14. Construction – coding and debugging – was strong in general.
15. System integration and testing was strong.
16. Project management and team work was strong.

NAVAL POSTGRADUATE SCHOOL

Conclusions and Recommendations

- NPS MSSE and SE4003 are, in general, good preparation for students to act as software engineers, but weaknesses may exist.
- Recommendations:
  - Prioritize the observed weaknesses and do a causality analysis.
  - For the high-priority (critical) weaknesses, determine which core courses (including SE4003) might need changes.
  - Evaluate impact of proposed changes and determine a balanced curriculum update package.
  - Brief to curriculum sponsor, as required, and implement approved changes.
  - Re-evaluate after several course offerings.