Stakeholder Needs and Requirements

Lecture 11, v01

John G. Artus

BSEE

MSSE

INCOSE ESEP

Flow of System Specifications

- For most system development undertakings, the initial specification of needs for a system solution come from a set of stakeholders
- These need statements get transformed into technical specifications that will provide a level of technical rigor needed by the system development team

Stakeholder Need Statements → Stakeholder Requirement Statements → System Requirement Statements → Derived Requirement Statements

- Many people can end up getting their hands in the process of generating need statements or requirements
  - From stakeholders not skilled at writing technical specifications to engineers that specialize in preparing unambiguous technical specifications
- Requirements engineers are the specialists that have experience in soliciting inputs from stakeholders and converting raw need statements into unambiguous technical specifications
- System requirements are the principle system specification inputs to the System Architecture Definition Process
- The purpose of this lecture is to make the system architect aware of the issues associated with the development of these critical inputs to the architecture development process

Stakeholder Needs and Requirements

- Stakeholder need statements represent the views of users, acquirers, customers, and other stakeholders
- They are often stated in common language (not engineered to be unambiguous)
- They often simply express the desires of stakeholders as they relate to the problem (or opportunity), for a solution that can provide the services needed by the stakeholders in a defined environment
- Often, stakeholders are led through a structured process to elicit these need statements
- Stakeholder needs are then transformed by Requirements Engineers into a defined set of Stakeholder Requirements, which may be documented in the form of a model, a document containing textual requirement statements or both
  - Ambiguity may likely still remain in the Stakeholder Requirements as these requirements have not yet been fully analyzed within the context of a holistic solution

System Requirements

- System requirements result from the translation of stakeholder requirements (expressing stakeholder needs) into engineering-oriented language in order to enable proper architecture definition, design, and verification activities
- The system requirements are based around identification and synthesis of the functions required of the solution system associated with performance and other quality measures
- They provide the basis for the assessment of candidate solutions and verification of the completed system
- The system requirements are expressed in technical language that is useful for architecture and design: unambiguous, consistent, coherent, exhaustive, and verifiable
- Close coordination with the stakeholders is necessary to ensure the translation is accurate and traceability is maintained
- This results in a set of system functions and requirements specifying measurable characteristics which can form the basis for system realization
- Requirements traceability is used to track information from the original stakeholder requirements, to the requirements and other system definition elements at all levels of the system hierarchy

Constraints

- Constraints are limiting factors imposed upon the design by stakeholders, or other persons of authority
- Constraints can also be established during design work as new information is developed as the design unfolds that locks down certain design characteristics
  - Constraints can be imposed during within the design activities to distribute allocated budgets within the design
  - Example: Distributing a “not-to-exceed” weight or power-consumption level among components
  - These are also known as “allocated requirements”
- Sometimes constraint are dictated by external factors such as natural physics (gravity) or the limitations of the current state of technology
- Constraints are typically not able to be changed based on trade off analyses as are other system characteristics not placed under constraint

Assigned Requirements

During architecture definition and design, the assignment of requirements from one level to lower levels in the system hierarchy can be accomplished using several methods, as appropriate

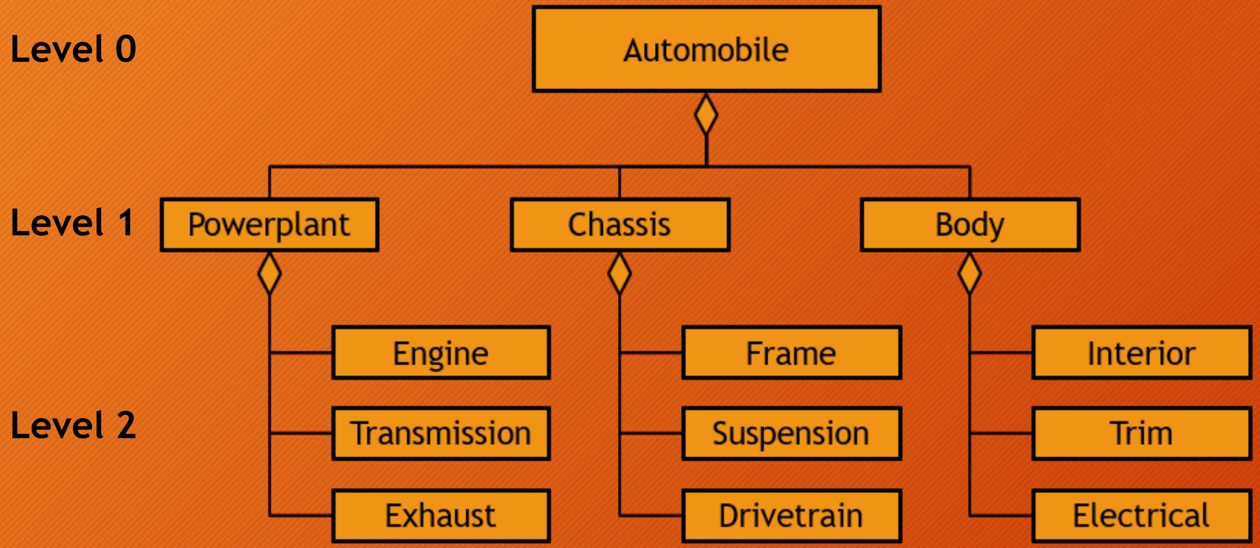
- Direct Assignment
  - The system requirement from the higher level is directly assigned to a system or a system element for a lower level
  - Example: the color used to paint visible parts of the product
- Indirect Assignment - Simply Decomposed
  - The system requirement is distributed across several systems or system elements and the sum of a more complex calculation for distribution is equal to the requirement of higher with sufficient margin or tolerance level
  - Example: budget allocation to components, such as mass, power, reliability, etc.
- Indirect Assignment - Modeled and Decomposed
  - Same as above, except in this case, the allocations are made using an analysis or mathematical modeling technique
  - Example: a radar detection requirement that is analyzed, with lower-level parameters for output power, beam size, frequencies, etc. assigned to the appropriate hardware and software elements of the radar system
- Derived Requirement
  - The result of decomposing top level system requirements into a set of child requirements that are then allocated to child components that, when satisfied, will result in the full satisfaction of the parent requirement
  - Developed during the design activities as a result of the decision of the design team, not the stakeholder community
  - May not directly trace to a stakeholder requirement, but they do not conflict with a stakeholder requirement or a constraint

Derived Requirements

- Derived requirements are those that are created by the design team in order to specify lower-level details of the design that the stakeholders either
  - Do not have the expertise to specify themselves
  - Do not have the vision into the system details to know what to specify
  - Do not have interest in being involved in the specification of these details
- Derived requirements are requirements that aren’t explicitly defined in higher-level requirements, but which are necessary for meeting the higher-level requirements
- Derived requirements are needed by lower-level architects and designers in order to have the information necessary to do their work at their level of design
- Derived requirements can be created from other derived requirements, throughout the system structural hierarchy, down to the lowest level

Because derived requirements are often crested a a result of engineering analysis based on constraints imposed by (perhaps many) system requirements, it is often not possible to trace derived requirements back to just one or a few system requirements

Derived Requirements Example



System Requirements

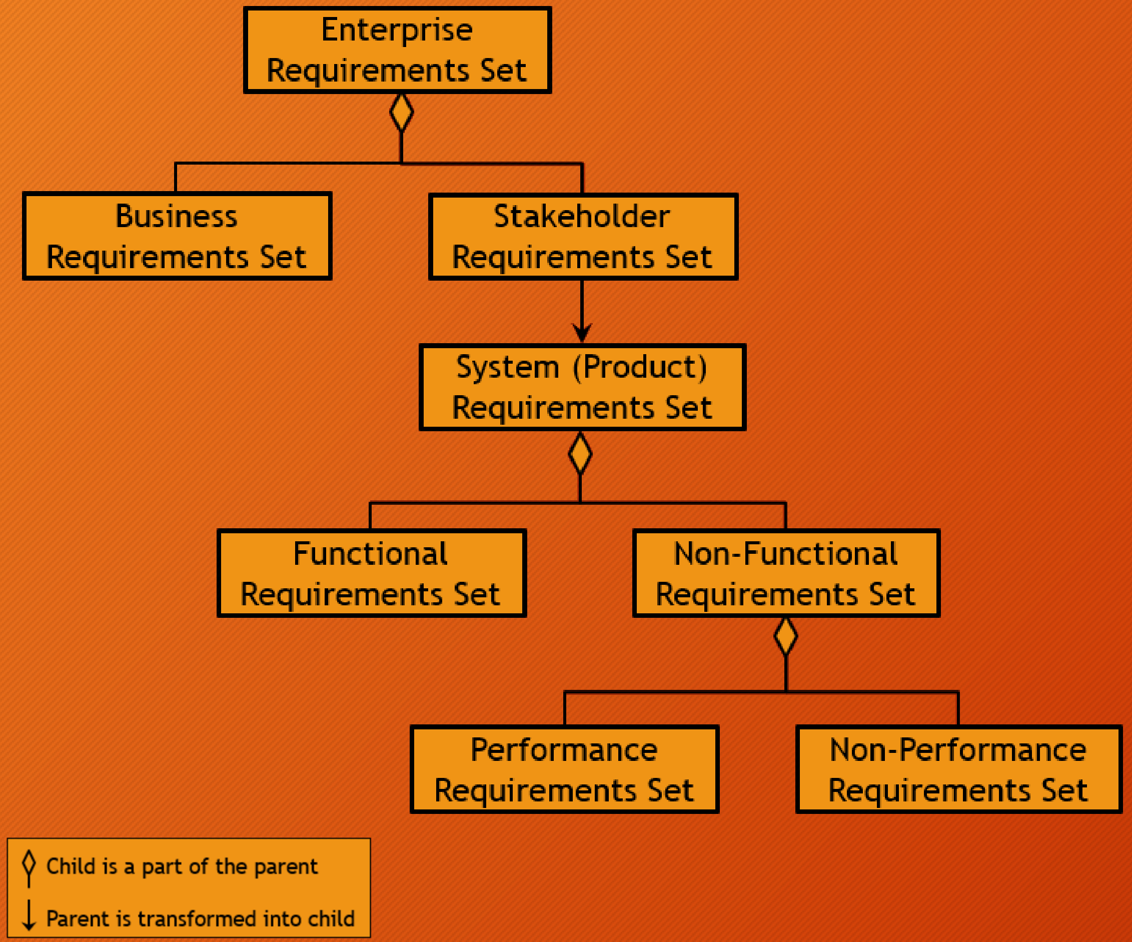
(as transformed from Stakeholder Requirements)

- Level 0 (System) The Automobile System shall achieve a maximum acceleration rate of 0 to 60 mph in 5.5 seconds.
- Level 1 (Powerplant) The Automobile System Powerplant shall be front-mounted longitudinally in the Chassis.
- Level 2 (Engine) The Automobile System Engine shall achieve a maximum sustained fuel efficiency of 25 mpg at 60 mph under road conditions defined in Document AUTO-1234.
- Level 2 (Drivetrain) The Automobile System Drivetrain shall deliver power to the rear axle via a driveshaft.

Derived Requirements

- Assuming that the System Requirements trace back to Stakeholder Requirements, these end up being constraints on the design (we have no flexibility to adjust these system characteristics)
- Anything not specified by the Stakeholder provides the architect with design space
- The design team has several degrees of freedom to play with to meet the system requirements:
  - Vehicle weight
  - Engine power
  - Vehicle aerodynamics
  - Vehicle component friction
- These can be established by creating derived requirements based on analysis using the system requirements as constraints
- Let’s say that the company imposes the constraint that an existing engine must be used in this product; then as a result of analysis, the design team selects an engine from among the stable of available designs of various power levels, volume, etc.
- Level 2 (Engine) The Automobile System shall use the current Model 346xi-A6 Internal Combustion Engine as the powerplant.

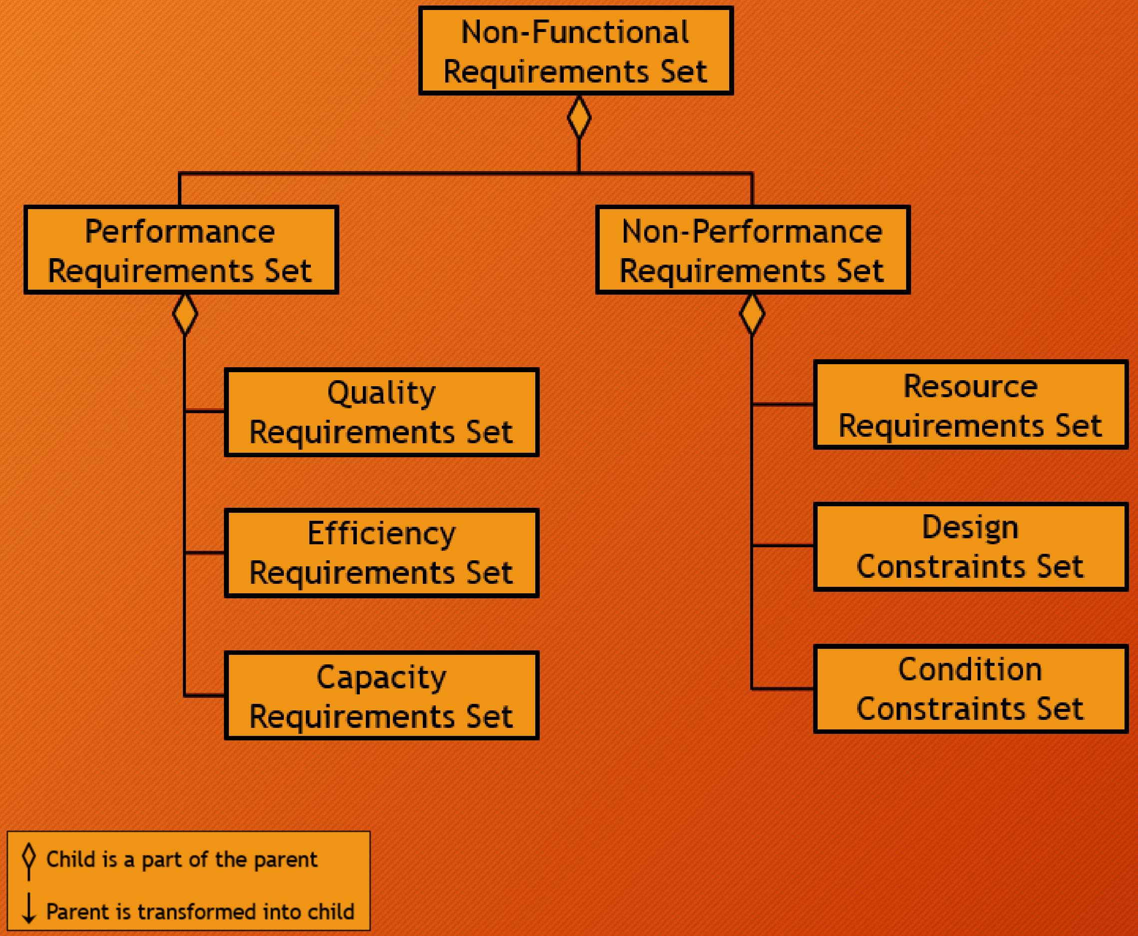
Taxonomy of Requirement Types



Categories and category names presented here are not standard
Other categorizations have been suggested by other people
Purpose is simply to highlight the main concerns of the architect

Requirement Type	Significance to Architecture
Enterprise Requirements Set	Consider this to be the complete set of requirements from all perspectives
Business Requirements Set	These are the requirements that have to do with the business goals for the project (sales figures, profit estimates, etc)
Stakeholder Requirements Set	This is the complete set of stakeholder needs statements from any and all stakeholders (including procurer interests, user interests, etc)
System Requirements Set	The stakeholder requirements are transformed by Requirements Engineers into unambiguous System Requirements
Functional Requirements Set	This is the subset of System Requirements that address what the system will do (what functions it will perform)
Non-Functional Requirements Set	This is a catch-all category to capture any requirement that does not specifically address system functionality
Performance Requirements Set	This is the subset of Non-Functional Requirements that specifically address the performance goals that the system is to demonstrate for the functions that it performs
Non-Performance Requirements Set	This is a catch-all category to capture any requirement that does not specifically address system functionality or performance (including appearance, style, aesthetics, etc.)

Taxonomy of Non-Functional Requirement Types



Constraints are a type of requirement where a design parameter is fixed by the architect or other senior design staff member and generally cannot be traded (adjusted)

Requirement Type	Significance to Architecture
Quality Requirements Set	How well the system performs (such as usability, availability, customer satisfaction, etc)
Efficiency Requirements Set	Specifies the required improvement in resource utilization compared to defined benchmarks (such as energy, material, data, etc)
Capacity Requirements Set	The required capacity of the system processes (such as system peak processing volume, speed of execution, data storage capacity, etc)
Resource Requirements Set	The levels of resources that stakeholders plan to expend to develop and operate a system (such as mileage performance, water consumption rate, etc)
Design Constraints Set	Firm and fixed design characteristics that are imposed upon the system design, usually with no room for compromise
Conditional Constraints Set	Any additional constraints to those imposed by the function requirements, the performance requirements, the resource requirements and the design constraints that have some conditional dependency (such as ‘the system shall auto-adjust the clock if daylight savings time is in force’)

Characteristics of Good Requirements

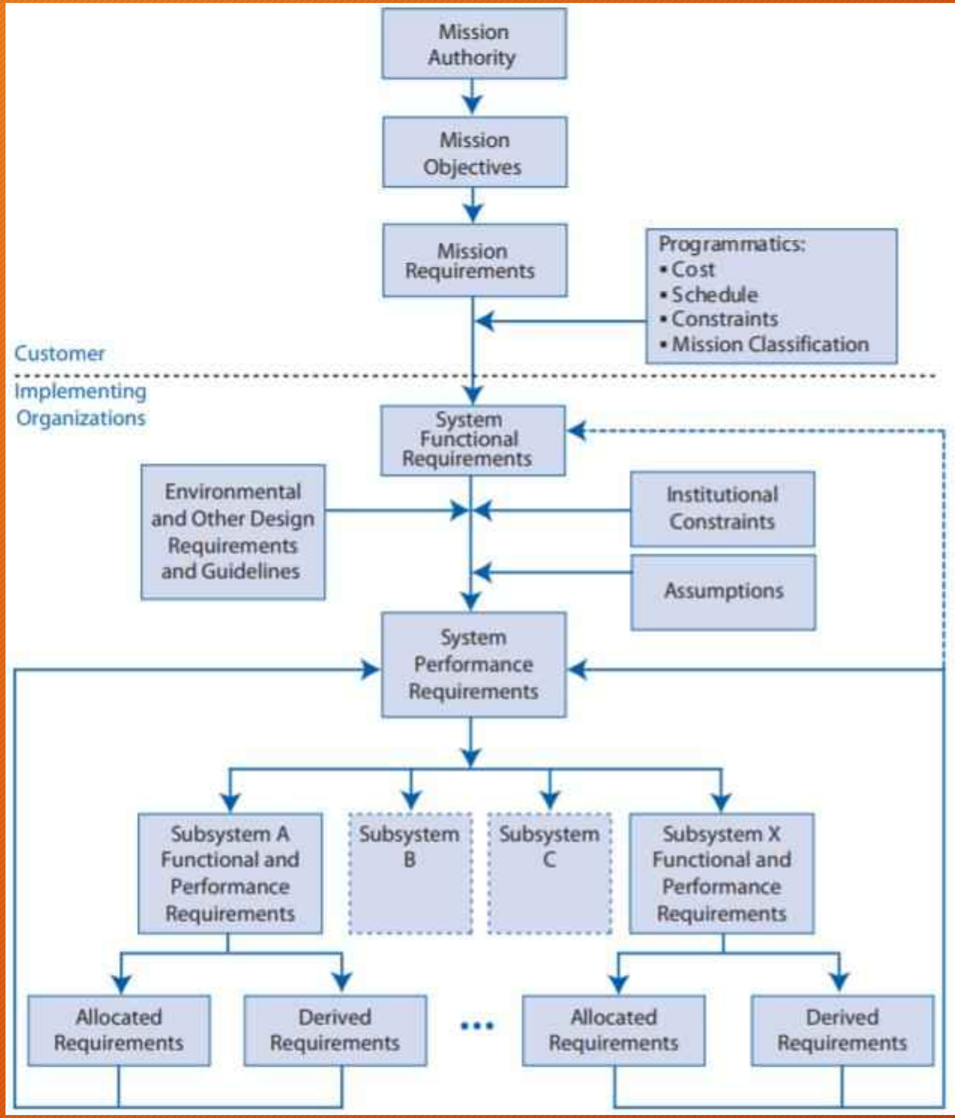
Well-written requirements are essential for project success, including the following key characteristics

- Necessary
  - State only what is determined to be necessary for achieving the client's mission within regulatory constraints
  - These statements are derived through analytical means to the depth necessary for design, procurement, implementation and verification
- Implementation Free
  - State what is required and how well it needs to be done without bias for how it will be done
  - The design team should be allowed to choose the best means of accomplishing the requirements. This helps to provide stable requirements and to control cost
- Unambiguous
  - The intent of a requirement must be understood in the same way by the writer, the designer, and those doing verification and validation
  - Avoid using language which could lead to multiple interpretations – ensure that there is sufficient precision that there can be no debate about what is the intention
- Clear
  - Convey what must be achieved in a manner that can be understood by those who are expected to implement the requirement, without having to ask the author what was meant

Characteristics of Good Requirements (continued)

- Achievable
  - Confirm with the implementer that the requirement can be affordably achieved either by previously developed means, or within a reasonable period of development
- Traceable
  - Ensure that system requirements can be traced to a user need or a higher-level specification, and to a supporting analytical basis
  - Ensure that derived requirements can be traced to the implementing item
- Verifiable
  - Requirements must be stated in a manner that compliance can be objectively confirmed
  - Typical methods of confirmation include analysis, inspection, demonstration, and test
- Complete
  - A set of requirements needs to be complete, such that if all are met then the resulting system will successfully achieve the client’s need for the system
  - Ensure that the needs of other stakeholders will be addressed to the agreed extent, and that applicable regulations will be met

Flowdown of Requirements Summary



https://www.arxterra.com/realistic-constraints-and-engineering-standards/

- This diagram provides a larger perspective of the way project specification information flows down from the highest, most authoritative sources of such information, down to the lower-level implementors of the architectural design
- The areas of concern for the system architect from this graphic are below the line demarcating the Implementing Organization (specifically for us, the architecture team)
- Feedback loops are indicated to account for the refinement of requirements as the design proceeds deeper into the structural hierarchy and reveals the need to make adjustments to system requirement in order to reach a feasible solution

Measures of Effectiveness and Performance and Key Performance Parameters

Measures of Effectiveness and Performance

- A Measure of Effectiveness (MOE) is a criterion used to assess changes in system behavior, capability, or operational environment, tied to measuring the attainment of an end state, an objective, or the creation of an effect
- Measures of Effectiveness (MOEs) are designed to correspond to the accomplishment of mission objectives and achieving desired results
- MOEs measure the relevance of one or more actions performed to produce a desired effect
  - The desired effect is generally not the direct result of the individual action(s) performed
- An MOE is generally an aggregation of MOPs
  - The terms “MOP” and “MOE” are not interchangeable since they answer completely different sets of questions
- The necessity to relate MOPs and MOEs is embedded in the definition of a MOE itself
  - Since the achievement of a certain mission effect depends on the performance of the one or more actions that result in the desired effect
  - It is often difficult to establish direct relationships between the performance of these activities and their immediate effects
  - However, it is necessary to explore possible relationships between the activities being performed and the impact of those activities on the operational environment
  - For instance, if a flight student is bombarded with hours and hours of mandatory classroom training evolutions and power-point presentations and subsequently passes all required exams (MOPs) but is unable to satisfactorily complete an actual flight (MOE), then perhaps the company should reconsider the amount of resources being spent on classroom training and presentations as opposed to alternative, more effective activities such as simulation training

Measure of Effectiveness

- MOEs quantify the results to be obtained by a system and may be expressed as probabilities that the system will achieve the desired effects
  - This measure may be quantitative or qualitative and it allows comparable systems to be ranked
  - These effectiveness measures are defined in the problem-space (in the requirements)
- MOEs should be easily understood and evaluated
  - A complicated MOE is more difficult to evaluate and can lead to confusion and a lack of understanding of the real issue
  - Implicit in the meeting of MOE requirements is that threshold values must be exceeded
- Measures of Effectiveness should use assessment indicators that are
  - Relevant, measurable, responsive, and resourced
  - There should be no false impression of what is needed to accomplish a task or objective
  - A mix of quantitative and qualitative indicators is used to evaluate MOEs to mitigate the risk of misinterpretation and overcome the limits of raw data in understanding complex situations

Why are MOEs Important?

- Regardless of what planning process or paradigm is used, actions create effects, whether they are positive, negative, or neutral
  - These effects encompass the full range of possible outcomes (or consequences of actions) across the full spectrum of system operations
  - There has to be an attempt to measure those effects by doing more than just measuring performance of individual actions
- Within the operational environment, we are trying to determine causation (what actions produce the desired effects) in order to develop those actions and reach the desired outcome (called the "end state")
- Our planning process demands that we know as much as possible about the environmental situation (the state of the operational environment when actions are taken) if we are to develop the actions necessary to create the conditions of the desired end state
  - Part of that planning process is to predict the outcome from the actions taken
  - This can be an extremely complex task when each problem is distinctive unto itself
  - Yet, together they shape the operational environment and can make it difficult to predict effects from individual actions
    - Due to the potentially indirect nature of the relationship of the outcome of the actions to the desired effects
    - Due to the complex nature in which the outcomes of multiple actions can possibly interact with each other

The Scientific Basis for MOEs

- While we can rarely be certain of any outcome, we make assumptions based on existing facts to establish causation between actions and results
  - These facts should be based on an analysis of past inputs and their outcomes
- During this process, we must be careful to distinguish between correlation and causation
  - Correlation means that two events tend to occur together with some frequency, but this does not necessarily imply causation
  - Causation can only be determined by developing a hypothesis, which will attempt to find the correct way of linking our actions to the desired effects
  - Put simply, causation indicates that if we do “X,” we expect to get “Y” result
- As with any hypothesis, there has to be a method for determining if we were correct (by measuring something)
  - Here, it becomes important to include MoEs within the planning process to help assess success
- A planner of system operational effectiveness must consider causation and correlation, and then attempt to predict effects on the operational environment
  - This is similar to the scientific method in that they both attempt to establish a relationship between inputs and outputs
  - An input is simply what goes into the action taken (what are we doing); the output is the direct result of our input
  - For example, we can hypothesize that if classroom training increases (input), then the students will be more proficient pilots (output)
- Some form of MOP can typically measure both the input and output (classroom training results), but neither of these can determine if there has been an increase in pilot proficiency
- To determine this, we measure the operational outcome (pilot proficiency)
  - Which should reflect whether there was a change (positive, negative, or neutral) due to of the output of the action taken
  - Within the operational environment, the outcome is usually determined by the effect on human behavior, since stakeholders are the ultimate judge of system value

MOE Example

- Typical system-level MOEs are
  - Emergency response time
  - False alarm rate
  - Operational availability
  - Total cost of ownership
  - etc
- Target values for each of these MOEs are established so as to achieve a competitive advantage
- These MOEs cannot practicably be determined until the system is operating in the field (in its operational location)

MOE Example for a Chemical Processing Plant

- Short Title: False Alarm Rate
- Definition: This MOE measures the percentage of system-generated alarms which are found to be false following some analysis of the system operational status/situation
- Unit of Measure: Percentage of system-generated false alarms per month = (Total number of alarms generated by the system that were later deemed to be false alarms / total number of system generated alarms) * 100; captured over a period of 30 days
- Benchmark: Less than 10% false alarms over a 30 day period
- Formula: Less (lower percentage number) is better

Measure of Performance (MOP)

- MOPs are a measure of a system's performance and are expressed as distinctly quantifiable performance features
  - MOPs quantify the set of selected parameters and measure the attributes of system behavior
- Several MOPs may be related to the achievement of a particular Measure of Effectiveness (MOE)
  - MOPs are necessary for correlation to specific MOEs in order to determine the optimal levels of effort for objective achievement
- Example MOPs for sensor systems include detection probability, probability of false alarm, and probability of correct identification

MOP Example

- The architect is often more concerned about MOPs that support MOEs
- MOPs can determined by measuring the performance of system components in the laboratory
- Typical system-level MOPs are
  - Reliability (As determined by failure rate)
  - Throughput (Output volume or rate)
  - Uptime (Inverse of downtime due to maintenance)
  - etc
- Target values for each of these MOEs are established so as to achieve a competitive advantage
- These MOEs cannot practicably be determined until the system is operating in the field (in its operational location)

MOP Example for a Chemical Processing Plant

- Short Title: Liquid Chemical Flow Pump Failure Rate
- Definition: This MOP measures the percentage of times that a pump fails to achieve its advertised performance in terms of starting, and reaching pressure and flow volume metrics
- Unit of Measure: Percentage of pump failures = (Total number of times the pump fails to achieve performance levels / 10000 laboratory test runs) * 100
- Benchmark: Less than 5% failure rate
- Formula: Less (lower percentage number) is better

Key Performance Parameter (KPP)

- KPPs are key system capabilities that must be met for a system to meet its operational goals set by the stakeholders
- These key capabilities form the foundation of any system and are deemed vital to its performance, function, design, and operations
- Any changes to the KPP would have a significant impact on the performance of the system along with the development cost and schedule
- When stakeholders identify KPPs in a system design, this indicates that they have a specific interest in meeting these performance goals, with the potential for serious contractual consequences if the goals are not met
  - The number of KPPs identified by a Sponsor should be kept to a minimum to maintain program flexibility
- It is imperative for contractors to
  - Ensure that effort is being put into the system design to achieve the goals set by KPPs
  - Collect data that will illustrate that system design activities are on track to meet KPPs by expected due dates
  - Such data is reported to stakeholders on a regular basis to assure them of certainty of meeting established KPP goals

References

- Gilb, T. (2005), *Competitive Engineering*, Butterworth-Heinemann Publishers
- Systems Engineering Body of Knowledge (SEBoK) v2.7
  - https://sebokwiki.org/wiki/Guide_to_the_Systems_Engineering_Body_of_Knowledge_(SEBoK)
- Bayhill, T. (2008), *Decomposed Requirements versus Derived Requirements*,
- INCOSE. (2012), *Guide for Writing Requirements*, Document Number: INCOSE-TP-2010-006-01
- INCOSE. (2015), *Managing Requirements for Design*, , Document Number: INCOSE-PI-2015-003-1.0, Version 1.0
- Arxterra, *Engineering Design*
  - https://www.arxterra.com/realistic-constraints-and-engineering-standards/
- AcqNotes, *Measures of Effectiveness*
  - https://acqnotes.com/acqnote/tasks/measures-of-effectivenessrequirements
- AcqNotes, *Measures of Effectiveness*
  - *https://acqnotes.com/acqnote/acquisitions/key-perfrormance-parameter*
- Westphal, T., Guffey , J. (2019), *Measures of Effectiveness in Army Doctrine*
  - *https://www.benning.army.mil/armor/eARMOR/content/issues/2014/OCT_DEC/Westphal.html*