Functionality, Behavior, and Effects

Lecture 47, v02

John G. Artus

BSEE

MSSE

INCOSE ESEP

About This Courseware

- I, John Artus, am using the material sourced from various textbooks, research papers, and other indicated sources, as content for this courseware for educational purposes only
- I, John Artus, make no claim of ownership of the source material used to produce the educational material contained in this document
- This courseware lecture material has been sourced, interpreted, assembled, formatted, and copyrighted by John G. Artus for use in this educational context
- Anyone may freely access, and reuse this material in an educational context provided the copyright owner, John G. Artus, is recognized as the interpreter, assembler, and formatter of the source material used in the generation of this courseware

Functionality Versus Emergent Behavior

- For System Architects, the most important concept to be understood is that

The functionality of the system (as expressed by its behavior) is greater than the sum of the functionality of the individual entities

- Architecting a system such that the Anticipated system behavior emerges is the goal — and the art — of system architecting
- The difference between the functionality that is designed into the system to produce the anticipated behavior and the actual, observed behavior that emerges from the system is that
  - While we can systematically analyze and predict the behavior that we anticipate will emerge
  - There are often unforeseen, and possibly unintended interactions among the components of the system that produce emergent behavior that is not exactly what we anticipated
  - This is especially true in today's highly complicated systems with extremely large numbers of interacting components

What is Emergent Behavior?

- Functionality is what a system is designed to do: which will hopefully result in useful and valuable outcomes or outputs for the stakeholder
- Emergent behavior refers to what actually is observed to appear, materialize, or surface when a system operates
- We architect systems so that the Anticipated, desirable behavior emerges
- Systems are intentionally designed to produce behavior that is of benefit and/or value to the stakeholder
  - This is anticipated, desirable behavior
- Occasionally some system behavior emerges that is expected, but undesirable
  - This is an anticipated, undesirable behavior
- At other times, and often by surprise, some unexpected but valuable behavior emerges
  - This is a unanticipated, desirable behavior
- Unfortunately, some behavior can sometimes emerge that was not expected and is wholly unwanted
  - This is an unanticipated, undesirable behavior

Types of Emergent System Behavior

Classification of the Types of Emergent System Behavior

		ANTICIPATION	
		Anticipated	Unanticipated
DESIRABILITY	Desireable	***Anticipated-Desirable*** *Behavior that we expected the system to demonstrate and that has value to us*	***Unanticipated-Desirable*** *Behavior that we did not expect the system to demonstrate but that has value to us*
	Undesireable	***Anticipated-Undesirable*** *Behavior that we expected the system to demonstrate but that has detrimental value to us*	***Unanticipated-Undesirable*** *Behavior that we did not expect the system to demonstrate and that has detrimental value to us*

Examples of Types of Emergent Behavior

- Anticipated-Desirable
  - This is the most common type of emergent behavior since it is what we design systems to produce
- Anticipated-Undesirable
  - There are somethings that we cannot avoid due to technological limitations or cost, etc.
  - A common example is bodily damage resulting from an automobile accidents
    - Seatbelts and other safety devices are installed in automobiles to help deal with the undesirable, but anticipated occurrence of accidents
- Unanticipated-Desirable
  - These types of behavior occur when a surprise event is discovered
  - In 1928, Alexander Fleming serendipitously discovered penicillin after accidentally leaving a Petri dish open while taking vacation away from his laboratory, that then formed a blue-green mold that contained the powerful antibiotic
- Unanticipated-Undesirable
  - This is the worst type of emergent behavior to experience
  - Experiencing an excessive amount of this type of behavior in a system may indicate that the developer of the system may be in the wrong line of business

How Does Behavior Emerge?

- Emergence is the magic produced by a system
- As the entities of a system are brought together, a new kind of functionality emerges as a result of the combination of
  - The functions performed by the individual entities
  - The functional interactions among the entities
- The idea that the functionality of a system is greater than the sum of functionality of the individual entities is an idea that focuses purely on emergence
  - Emergent behavior is a result of this “greater functionality”
  - Greater functionality (obviously) is realized in the domain of the function
  - Nothing “emerges” in the domain of the form (the physical components) of the system
  - But function depends on form to host (or to “implement”) that functionality

The Role of Form and Function in Producing Emergent Behavior

- Form enables the function of the entities
  - Formal relationships are instrumental in functional interactions
- This implies that both form and formal relationships (structure) are important to consider in predicting emergence
- The formal relationships are important in guiding a certain specific functional interaction that leads to a specific system-level emergence of desired behavior
- The emergence of behavior from a system depends on the function of the entities and their functional interactions
  - The form enables the function of the entities
  - The formal relationships are instrumental in functional interactions
  - This implies that both form and formal relationships (structure) are important to consider in predicting emergence
- Formal relationships are critical to emergence
  - They enable functional interactions that lead to system-level emergence

The Significance of Emergence

- The property of emergence gives systems their power
- Striving to understand and predict emergence is the primary goal (and challenge) of system architecting
- System success occurs when the anticipated, desirable behaviors emerge that result in achieving performance goals
  - Their interaction will implement allocated functionality
  - Resulting in behavior (and other properties), that can be measured
  - To determine whether performance goals have been met
- Dealing with change in complex systems can be problematic since change can often propagate in unpredictable ways
  - Especially true for not-well-thought-out, or poorly planned changes
  - Without proper analysis, it is difficult to predict how a change in one entity will influence the emergent properties of a system

Emergence and System Failure to Achieve Objectives

- System success and system failure often hinge on emergence of behavior and the resulting measure of system performance
- Another way to understand the importance of emergence is to think what might happen if the behavioral results are not as expected
- This can happen in two ways:
  - The anticipated, desirable behavior fails to emerge
  - And/or unanticipated, undesirable behavior does emergence
- Trying to understand and predict such system failures is also a goal of system architecting
- Contingency plans (often planned and implemented by operators) can sometimes deal with the appearance of anticipated but undesirable emergent behavior

Predicting Emergence

It is hard to predict a-priori what kind of behavior will emerge from the combinations of the functions of the various entities of a system

There are three ways to predict emergence of behavior

- Based on similar experiences in the past
  - This is prediction based on precedent (past experiences)
  - Look for identical or very similar solutions in our experience
  - Implement them with at most small changes
- By conducting experiments
  - Try putting together the entities with the proposed relations to see what emerges
  - This can range from tinkering to very highly structured prototyping
  - Spiral development is a form of experiment in which some of the system is first built to check emergence before the rest of the system is built in later spirals
- By performing modeling and simulation exercises
  - If the function of the entities and the functional interaction can be simulated by executing (running) models of the system, then it may be possible to predict emergence from a model

Predicting Emergence (continued)

- Predicting emergence is needed for a system that is
  - Without precedent
  - Cannot be experimented on
  - Cannot be reliably modeled
- Predicting the emergence of desired behavior is at the crux of system architecting
- The need to predict emergence arises routinely in new product development (unprecedented systems) for which experimentation and modeling are not easy
  - In these situations, humans must reason about emergence based on available information
- This reasoning may be informed
  - Partially by precedent (observing results in similar but not identical systems)
  - Partially by experiments and incomplete modeling
  - But the projection about emergence ultimately depends on human judgment

Concepts Related to Emergent Behavior

- System Performance
  - Performance is how well a system operates or executes its Anticipated functionality
  - Performance is an attribute of the functionality of the system
  - Emergence of desired behavior and the achievement of performance goals tends to create stakeholder value immediately
- Emergence of Quality Attributes
  - There are other quality attributes of operation that emerge from a system, such as reliability, maintainability, operability, safety, and robustness - called “-ilities”
  - The value created by these “-ilities” tends to emerge over the lifecycle of the system
- Unanticipated / Undesirable Emergence
  - Often, this type of behavior has a severe, negative impact on system value
  - It is so important that it often merits special attention and control measures
  - The appearance of behavior of this type is often characterized as an “emergency”

The Importance of System Performance

- System performance can affect system operational effectiveness, and thus can influence stakeholder satisfaction
- System performance can be measured to determine how effectively a system accomplishes its Anticipated functions under specific operating conditions
  - Examples of Anticipated functions include: processing workloads, responding to requests, and utilizing resources
- Performance measures establish quantifiable characteristics that collectively determine a system's ability to meet functional requirements within defined constraints

The Importance of System Performance (continued)

- Examples of performance measures that collectively determine overall performance include:
  - Response time (how quickly the system reacts)
  - Throughput (how much work can be processed)
  - Resource utilization (how efficiently resources are consumed)
  - Scalability (how performance changes with increasing load)
- Rather than pursuing generic system improvements that may not align with stakeholder business priorities, system architects should instead
  - Pursue performance optimization so as to address the specific attributes of most importance to stakeholders for particular use cases

Quality Attributes (“-ilities”)

- System requirements that address quality attributes are largely representative of the system as a whole rather than just some constituent parts
- They are often called quality attributes, non-functional requirements, specialty requirements, and “-ilities”
- The term specialty engineering refers to the collective engineering of all quality attributes of a system

Quality Attributes ("-ilities") (continued)

Incomplete list of typical quality attributes:

- Affordability
- Agility
- Alignment With Business Goals & Strategies
- Assurance
- Autonomy
- Availability
- Behavior
- Business Impact
- Capability
- Complexity
- Compliance To Regulation
- Concurrency
- Control
- Cost
- Customer Experience
- Data Accessibility
- Deadlock
- Disposability
- Environmental Impact
- Feasibility
- Flexibility
- Functionality
- Information Assurance
- Interoperability
- Inter-process Communication
- Known Limitations
- Maintainability
- Misuse
- Modifiability
- Modularity
- Openness
- Performance
- Privacy
- Quality Of Service
- Reliability
- Resilience
- Resource Utilization
- Schedule
- Security
- Shortcomings
- State Change
- Structure
- Subsystem Integration
- System Features
- System Properties
- System Purposes
- Usability
- Usage
- Viability

Observed Effectiveness of the System During Operations

- The effectiveness of a system (how effective a system is, while operating in its operational environment, at producing the effects desired by the stakeholder) is, obviously, up to the stakeholder to determine
  - The effects that are to be achieved in the operational environment will be influenced by factors beyond the architect's control, such as how the stakeholder employs and uses the system in the operational environment
  - Once in operations, it is the stakeholder that defines
    - What the desired effects are
    - How to employ the system during operations to achieve the desired effects
    - Whether the achieved effects are desirable or not
  - The effect that a system is to have during operations cannot be exactly predicted during development
  - What architects do have control over is the behavior of the system that leads to desired or undesired effects as determined by the stakeholder during operations

The US Military is well known for utilizing systems in ways not anticipated during development in order to achieve effects on the battlefield that were not considered during development

Observed Effectiveness of the System During Operations (continued)

- However, during development, in order to better plan for the eventual use of a system by a stakeholder, the effects that the system is to achieve are predicted (assumed)
  - Establishing assumptions is an effective engineering technique to allow development to proceed without definitive and exact data to support
  - Assumptions need to be eventually resolved by replacing them with data supported by evidence (see following slides that discuss engineering assumptions)
  - Whether the system produces the desired effects predicted during development are achieved or not will not be known until an assessment of the achieved effects following operation of the system in its operational environment
- During development, the demonstrated behavior of the system is under the direct control of the architect
  - Thus, the predicted effectiveness of the system can be achieved by making the appropriate design decisions
  - However, predicted effectiveness is not the same as the actual effectiveness desired by the stakeholder during operations of the system in its operational environment
- Effects that are observed in one release of the system could later become the predicted effects for the next release of the system

The effects that are predicted during development may not be the same as the effects that are desired by the stakeholder during actual operations

Measure of Effectiveness (MOE)

- MOEs are the stakeholder's key indicators of achieving the mission needs for performance, suitability, and affordability across the life cycle of the system
  - MOEs focus on the system's capability to achieve mission success (achieve the desired effects) within the total operational environment
  - MOEs represent the stakeholder's most important evaluation and acceptance criteria against which the quality of a solution is assessed
  - MOEs are the overall operational success criteria to be used by the acquirer for the delivered system, services, and/or processes
- MOEs are independent of any particular solution
  - MOEs are specific properties that any alternative technical solution must exhibit to be acceptable to the stakeholder
  - In addition to using MOEs to compare and evaluate alternatives, they can also be used for sensitivity analysis of performance from variations of key assumptions and parameters of the potential alternatives
- MOEs are important for test and evaluation because they determine how test results will be judged
  - Since test planning is directed toward obtaining these measures, it is important that they be defined early

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 environment)

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

Making Engineering Assumptions During Design

- In engineering design, it is necessary to make many data-driven decisions in order to create a product that meets system requirements
  - With limited time, resources, and a lack of sufficiently detailed information, engineers often find themselves having to make assumptions
  - One reason engineers make assumptions is to enable the design process to move forward
- Assumptions can reduce the complexity of a problem, allowing engineers to focus on the most critical aspects of a design
  - Assumptions are often based on past experiences or limited data that supports the assumption
  - Assumptions should only be established to substitute for data that is currently unavailable or unknown
- Another reason engineers make assumptions is to save time and resources
  - Sometimes, it is not practical or feasible to conduct extensive testing or research to gather all the necessary data to make a more informed decision
- Assumptions can also introduce errors or biases
  - Engineering assumptions are often just educated guesses about what the actual data or property value may eventually turn out to be, and therefore can sometimes be wrong
- Thus, establishing assumptions incurs risk if the assumptions are not eventually converted to data supported by engineering evidence

Proper Management of Assumptions

- While assumptions may be necessary and useful, they need to be carefully managed and resolved in order to avoid creating incorrect or incomplete designs
- Proper management of assumptions requires that
  - Engineers keep formal track of assumptions made during design activities
  - With the goal of eventually replacing assumptions with real engineering data
  - Any remaining assumptions that cannot be resolved with actual data should be reported to stakeholders in order to obtain agreement on the use of the assumptions in the final design
- One way to do this is to test assumptions as soon as possible in the design process
  - Engineers can conduct preliminary tests, simulations, or construct prototypes to validate their assumptions and identify potential issues with established assumptions early on in the design process
- Another approach is to involve a diverse team of experts with different backgrounds and perspectives
  - By incorporating different viewpoints, engineers can identify and address potential blind spots and biases that may arise from making assumptions
- Where possible, engineers should eventually replace assumptions with engineering data supported by evidence when that data becomes available

References

1. Crawley, Edward; Cameron, Bruce; Selva, Daniel (2016). *System Architecture: Strategy and Product Development for Complex Systems*, Pearson Higher Education Inc, Hoboken, NJ
2. "*System Performance*" page at capstera.com. Retrieved from https://www.capstera.com/glossary/system-performance/
3. Web Article: "*Science and serendipity: famous accidental discoveries*". Retrieved from https://newhumanist.org.uk/articles/4852/science-and-serendipity-famous-accidental-discoveries
4. Engineering Technology website article "*Engineering Design Assumptions*". Retrieved from https://engineeringtechnology.org/engineering-graphics/the-engineering-design-process/assumptions-a-double-edged-sword/
5. Martin, J. (2017). *Overview of an Emerging Standard on Architecture Evaluation - ISO/IEC 42030*; International Council on Systems Engineering (INCOSE)
6. Systems Engineering Body of Knowledge (SEBoK) v2.11. Retrieved from https://sebokwiki.org/wiki/Guide_to_the_Systems_Engineering_Body_of_Knowledge_(SEBoK)
7. Artus, J. (2025). Unpublished comments to INCOSE Architecture Working Group on review of Committee Draft 2 of ISO/IEC/IEEE 42024, "*Enterprise, systems and software -Architecture fundamentals*"