Role of the Systems Architect

Lecture 10, v01

John G. Artus

BSEE

MSSE

INCOSE ESEP

About This Courseware

- The majority of the material presented in this course is sourced from the textbook “System Architecture” by Edward Crawley, Bruce Cameron, and Daniel Selva
- I, John Artus, make no claim of ownership of the material sourced from this textbook
- I, John Artus, am using the material sourced from this textbook, and other indicated sources, as content for this courseware for educational purposes only
- 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, and provided that Edward Crawley, Bruce Cameron, and Daniel Selva are recognized as authors of the textbook “System Architecture” from which the majority of the content of this courseware has been sourced

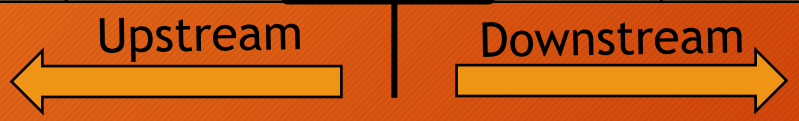
Role of the System Architect

- The system architect is not a generalist, but rather a specialist in simplifying complexity, resolving ambiguity, and focusing creativity (Eberhardt Rechtin)
- The system architect crafts the vision of the system and communicates that vision among the stakeholders and the extended project team
- Principal roles of the system architect:
  1. Reduce ambiguity
     - By defining the boundaries, goals, and functions of the system
  2. Employ creativity
     - By creating the concept
  3. Manage complexity
     - By decomposing the system into smaller, more manageable parts

Life Cycle Technical Process Flow Stream

- Upstream Activities
  - Before the architect is engaged, there has been an “upstream process” that has defined issues, opportunities, and needs for the new system
  - The upstream activities are full of ambiguity
- Downstream Activities
  - These are things that are done after the architecture is created but which must be factored into the architectural design prior to delivery of the architecture

Business or Mission Analysis	Stakeholder Needs and Requirements	Architecture Definition	Design Definition	System Analysis	Implementation	Integration	Verification	Transition	Validation	Operation	Maintenance	Disposal



http://www.clipartpanda.com/clipart_images/winding-river-graphic-45893222

Architecture sits astride the upstream activities and downstream activities
Ambiguity arises from both sides

Ambiguity from Upstream Processes

- Some examples of ambiguity present in the upstream influences are
  - Strategy: How much risk is corporate, the board, or the government willing to take?
  - Marketing: Will it fit with marketing's plans for product line positioning and distribution?
  - Customers: What do the customers want/need? Will their needs change with time?
  - Manufacturing: Will manufacturing be ready to produce it?
  - Operations: What failures must the product endure and still operate?
  - R&D: Is the technology infusible?
  - Regulations: What are applicable regulations? Are they likely to change?
  - Standards: What are the applicable standards, and will they change?
- The initial information that the architect receives could be fuzzy, uncertain, missing, conflicting, or false

Identifying Sources of Ambiguity

- The architect's job is to provide more certain, less fuzzy inputs to the team
  - In some cases, the ambiguity can be pared down by analysis
  - In other cases, one may choose to impose constraints simply to make the problem more tractable
  - Sometimes it is necessary to make an assumption so that work can proceed, being careful to mark the assumption for future resolution/verification
- The architect is responsible for creating boundaries and concretizing goals
  - Interpreting corporate and functional strategies
  - Interpreting competitive marketing analyses
  - Listening to users, beneficiaries, customers, or their representatives
  - Considering the competence of the enterprise and its extended supply chain
  - Considering the operations and operational environment of the system
  - Infusing technology where appropriate
  - Interpreting regulatory and legal influences
  - Recommending standards, frameworks, and best practice
  - Developing goals for the system based on the upstream influences

It is the job of the architect to drive ambiguity out of the products generated in the upstream processes before they get into the architectural design

Driving Ambiguity Out

- The architect must realize that many of the upstream influences just occur, often with incomplete, overlapping, or conflicting outcomes
- It is easy to get trapped in the thinking that the upstream influences (corporate strategy, marketing, the board, the customer, the government, and so on) have knowledge that can reduce the ambiguity, or that they have a responsibility to inform the architecture by commenting on early drafts
- The architect must engage these upstream influences and drive the ambiguity out to create a vision and plan for a successful product
- This requires knowledge of what the upstream influences are, who has “control” of them, and how they are best engaged
- The architect must keep in mind the objective: a consistent, complete, attainable, clear, and concise set of goals for the product
- Removing, reducing, and resolving ambiguity is the main role of the architect at the interface to the upstream process

Ambiguity is a Fact of the Architect's Life

- The early phase of a system design is characterized by great ambiguity
- It is the job of the architect to resolve this ambiguity in order to produce (and continuously update) concrete and feasible goals for the architecture team
- However, ambiguity is a fact of life in architecture work
  - Development is possible only with the acceptance of some level of uncertainty
- Ambiguity is handed to the architect by upstream processes
  - In general, the architect has no control over the processes upstream of architecting, so there should be no naive expectation of a lack of ambiguity in information handed to the architect
  - There will always be uncoordinated, incomplete, and conflicting inputs
  - Sometimes uncertainty can create opportunities; it is not always bad
- Ambiguity is especially apparent at the interface to the upstream influences, because there is generally little formal design discipline in the upstream processes
  - Ambiguity contains known unknowns and unknown unknowns, as well as conflicting information and false assumptions
- Ambiguities should be identified and prioritized so that they can be managed properly within the architectural design

Reducing Ambiguity

- Early development activities are challenging because the goals are unclear
  - A defined customer base cannot articulate what it wants next
  - The end customer is unclear
  - The underlying technology development and delivery progress is unclear
- The task of the architect is particularly challenging because
  - The architect sits between corporate goals and product/system definition quality
  - The architect therefore must competently bridge both worlds
  - To these ends, it is important for the architect to recognize and be able to deal with the various types of ambiguity
- Ambiguity is composed of two ideas
  - Fuzziness
  - Uncertainty
- However, ambiguity can also connote
  - Incorrect information
  - Missing information
  - Conflicting information

Fuzziness and Uncertainty

- Fuzziness occurs when an event or state is subject to multiple interpretations
  - Fuzziness is rampant in the statements of needs of customers
  - For example, when they say they want a “smooth” finish or “good” gas mileage
- Fuzziness is also influenced by context
  - What is good gas mileage will depend on the background of the customer
- Uncertainty occurs when an event’s outcome is unclear or the subject of doubt
  - Example: the outcome of a coin flip is uncertain
  - We can clearly articulate the potential states that could happen, but we don’t know which state will occur
  - For example, a new technology may or may not be ready at the date needed to support a new product
  - This produces uncertainty in its availability when needed in the design

Unknown, Conflicting, and False Information

- Compounding the ambiguity that arises from fuzziness and uncertainty are the related phenomena of Unknown, Conflicting, and False information
- Unknown information occurs when information is under-determined or unavailable
  - This is a very common case early in the product development cycle
  - You may know nothing about the product/market intentions of your competitors
  - Known Unknows: You may know about the absence of information
    - Example: You know that you don't know whether your existing competitor will launch a product or not
  - Unknown Unknowns: Even more dangerous, you may not know about the absence of information
    - Example: You are blindsided by a new competitor that introduces a new competing product
- Conflicting information occurs when two or more pieces of information offer opposing indications
  - The problem is over-determined
  - Example: you may have information from one source that the government will impose a new regulation affecting your product, but information from another source may suggest that the government will not do so
- False information occurs when you are presented with incorrect inputs
  - In this case you believe you have all the information, but some of it is wrong
  - Example: you have been told by someone that a supplier will be ready on a certain date, while the supplier actually has no plan whatsoever to be ready on that date

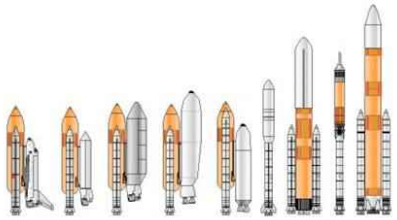
Dealing with Downstream Information

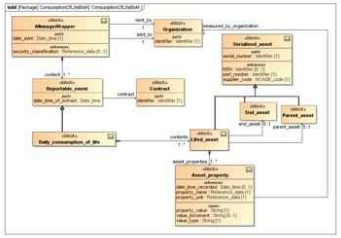
- One of the roles of the architect is moving relevant downstream information upstream into the architecting phase
  - One approach for accomplishing this is to analyze the downstream needs, and thus establish design constraints that will sevice the downstream needs
- Experience would suggest that it is easy to move constraints upstream
  - For example, say, repairability is an important attribute of the product
  - Therefore an applicable design constraint is that all parts that wear shall be colored differently, quickly accessible, and easily diagnosed
- Information to move upstream can be plentiful
  - But knowing which information will meaningfully differentiate between candidate architectures is more difficult

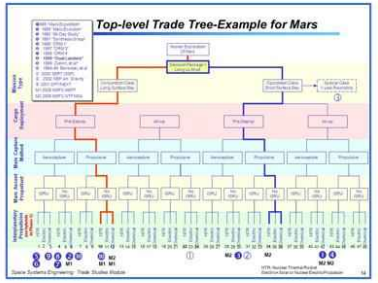
Developing a Concept for the System

- Once the goals for the system have been defined, the next task is that of defining a concept for the system
  - The architect employs creative thinking to create the concept for the product
  - The system concept captures the essential vision of the system, and differentiates the product relative to previous concepts
  - A good concept does not ensure system success, but a bad choice of concept almost certainly dooms a system to failure
  - In the moment when the concept is chosen, there is a relatively small amount of information available to define the system
  - But very rapidly the information explodes
    - Team members start defining external interfaces
    - The first-level design and decomposition take place
    - The considerations for downstream factors begin to be defined
- Several of the tasks in defining the concept are:
  - Proposing and developing concept options
  - Identifying key metrics and drivers
  - Conducting highest-level trades and optimization
  - Selecting a concept to carry forward, and perhaps a backup
  - Thinking holistically about the entire product life cycle
  - Anticipating failure modes and plans for mitigation and recovery

Concept of Operations


Concept Alternatives


Concept Architectures



Trade Study

Concept Down-Select Report & Model

Selecting a viable concept is critical as the rest of the program will be built around the chosen concept

Managing Explosion of Information as Design Develops

- The architect must manage investment in complexity and the evolution of complexity in the system, in order to ensure that the goals are delivered
  - A key component of that task is ensuring that the system is comprehensible to all
  - This aim is achieved by decomposing form and function of the system
- Decomposing form and function means
  - Clarifying the allocation of functionality to elements of form
  - Defining interfaces between subsystems and the surrounding context
  - Configuring the subsystems
  - Managing flexibility vs. optimality
  - Defining the degree of modularity
  - Articulating vertical vs. horizontal strategies
  - Balancing in-house vs. outsourcing design and manufacturing
  - Controlling product evolution
- These three principal roles of the architect all center on information
  - Identifying the necessary, consistent, and important information by reducing ambiguity
  - Adding new information through creativity
  - Managing the explosion of information in the final architecture

Managing Competing Forces Present in Design Work

- Traditional systems engineering conceives of a project as a three-way tradeoff among performance, schedule, and cost
- The systems engineer must identify the tensions among these competing objectives by
  - Setting (fixing) one of these three variables
  - Managing to a second variable
  - Allowing the third variable to float
- The dynamics that couple performance, cost, and schedule have been largely defined through project management models
- But correct in-house estimation of the three variables is rare, given the difficulty of obtaining the data and the expertise required to calibrate these models
- For the architect, complexity, creativity, and ambiguity are not as directly coupled as performance, cost, and schedule, but they are far more amorphous and difficult to manage
- The metaphor of tension in systems engineering is directly applicable to the role of the architect
- One of the primary mechanisms by which the architect acts is recognizing, communicating, and resolving tensions in the system

Deliverables of the Architect

- Deliverables are very different from tasks in that they are the end result, not the procedure by which the state is achieved
- The architectural deliverables include:
  - A clear, complete, consistent, and attainable (with 80–90% confidence) set of goals (with emphasis on functional goals)
  - A description of the broader context in which the system will sit, and the whole product context
  - A concept for the system
  - A concept of operations for the system, including contingency and emergency operations
  - A functional description of the system, with at least two layers of decomposition, including
    - Description of primary and secondary externally delivered functions
    - Process flow with internal operands and processes, including non-idealities, supporting processes, and interface processes
  - The decomposition of form to two levels of detail, the allocation of function to form, and the structure of the form at each level
  - Details of all external interfaces and a process for interface control
  - A notion of the developmental cost, schedule, and risks, and the design and implementation plans

One of the many challenges the architect faces is to represent the Architectural Deliverables such that they can be easily absorbed and used

Revisiting the Role of the Architect

- The role of the architect is to resolve ambiguity, focus creativity, and simplify complexity
- The architect seeks to create elegant systems that create value and competitive advantage by
  - Defining goals, functions, and boundaries
  - Creating the concept that incorporates the appropriate technology
  - Allocating functionality
  - Defining interfaces, hierarchy, and abstractions to manage complexity
- In view of the ambiguity and complexity present in most systems, and the need for balance, it is often desirable to have the architecture creation led by a single individual (Chief Architect) or by a small group (Lead Architecture Team)
- The architect maintains a holistic view but always maintains focus on the small number of issues critical to the design
- The architect may need to adopt different frameworks, views, and paradigms as appropriate to the issue at hand and as allowed by contract

Influences of Modern Industrial Practices

- Modern product development process, with concurrency, distributed teams, and supplier engagement, places even more emphasis on having a good architecture
- Some of the impacts this trend has on the architecting process are:
  - Accelerating the development process through the practice of concurrency increases the importance of initial conceptual decisions
  - Driving decision making to the lowest appropriate level (empowerment) and the use of distributed or non-collocated teams increases the importance of well-coordinated and visible high-level design guidance
  - Involving suppliers early in the process increases the importance of having an architectural concept and baseline; suppliers can define or defy decomposition

Product Development Process

- Nearly every large firm today has defined an internal product development process (PDP)
- This captures the methodology of product development, including the terminology, phases, milestones, schedules, and lists of tasks and outputs
- The intent of the PDP is to provide an enterprise framework that
  - Captures the wisdom of previous development efforts
  - Provides standard processes and approaches
- A principal advantage of such a PDP is that it can reduce ambiguity by defining tasks and responsibilities
- The architect should be cautioned not to be lulled into the notion that the PDP resolve the upstream ambiguities, leaving the architect with a clear understanding of strategy, market, customer needs, and technology - this is not always entirely true
- The PDP can do a disservice to the architect if it presumes more certainty than is actually present
- The PDP can be used to examine upstream and downstream influences and as a method to drive early ambiguity from the system

Summary

- The role of the architect includes reducing ambiguity, employing creativity, and managing complexity
- Architecture sits astride the upstream activities (things that are done before the architecture is created) and downstream activities (things that are done after the architecture is created but which must be factored into the architecture)
  - Ambiguity arises from both the upstream activities and the downstream activities
- In common practice, ambiguity is a combination of fuzzy information, uncertainty, missing information, conflicting information, and incorrect information
- The principal job of the architect at the interface to the upstream (the fuzzy front end) is to drive as much of this ambiguity from the system as possible
- A high-level way to do this is to start to prepare the deliverables of the architect

References

1. Crawley, Edward; Cameron, Bruce; Selva, Daniel (2016). *System Architecture: Strategy and Product Development for Complex Systems*, Pearson Higher Education Inc, Hoboken, NJ