Principal Software Architect
Discussion Syllabus

Understanding the Problem Space

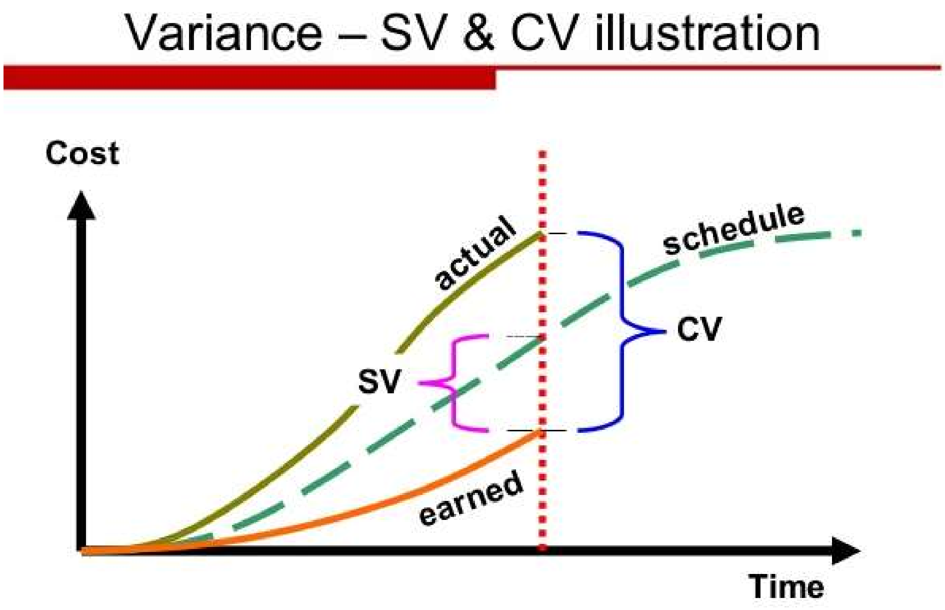
- (Read pages 1-5 of “Use Cases” white paper [1])
- Use cases to iteratively model the problem space
  - Actors (aka Roles)
  - Enumerate usage scenarios
- Define glossary of terms
- Model business information – key business concepts, relationships, and attributes
- Elicit project parameters
  - Quality
  - Constraints
  - Risks
- Elicit priorities
  - Must have, need, nice to have
  - Importance of use, frequency of use
  - Risk

Project Estimation

- Determine project’s required quality constraint ranging from prototype to mission or life-critical system
- Cost-Schedule-Features triangle
- (Read “The Triple Constraint: The Project Management Triangle of Scope, Time, and Cost” [2])
- Estimation “Q”
  - “T Shirt Size” estimates
  - Optimistic-Nominal-Pessimistic estimates
  - Detailed estimates
- Basis of Estimate – Work Breakdown Structure (WBS)
  - Project task granularity
- Feature Points verses Code Size (SLOC)
  - (Read “It’s Not About Lines of Code” [3])
  - Correlating feature points to effort-hours

BCWS verses BCWP verses ACWP

(Read "ACWP (Earned Value Analysis) [4]")



Variance – SV & CV illustration

Communication

Goal is to achieve common understanding of problem and solution space

One of an Architect's important responsibilities is to facilitate understanding among project stakeholders as well as project team members

Misunderstanding among team members is a key project risk

English is an ambiguous language

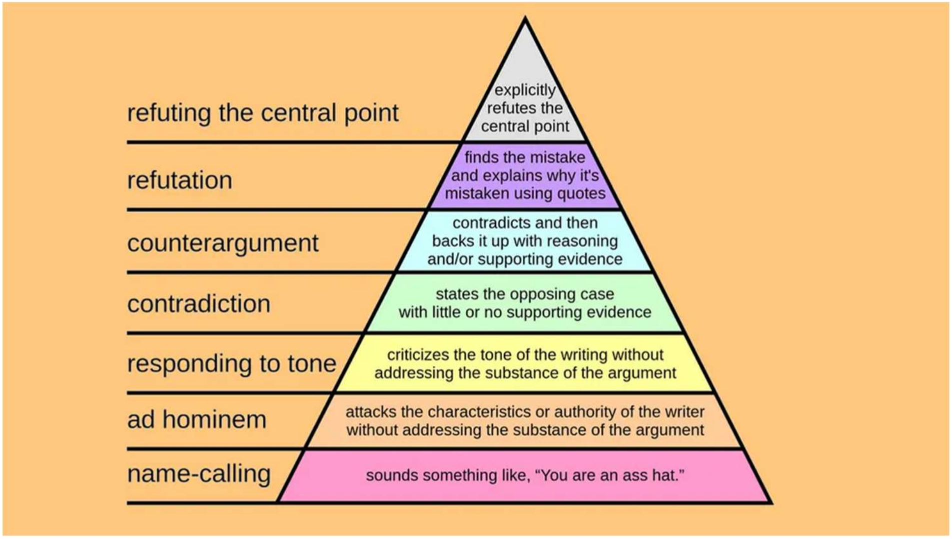
Avoid pronouns ("user", "it", "he", "they", "that")

Define business and technical terms; document in project glossary

Strive for concise, clear, and accurate speech

Resolving conflict

Disagree well (Read "How to Disagree" [5])


refuting the central point
explicitly refutes the central point
refutation
finds the mistake and explains why it's mistaken using quotes
counterargument
contradicts and then backs it up with reasoning and/or supporting evidence
contradiction
states the opposing case with little or no supporting evidence
responding to tone
criticizes the tone of the writing without addressing the substance of the argument
ad hominem
attacks the characteristics or authority of the writer without addressing the substance of the argument
name-calling
sounds something like, "You are an ass hat."

Structured disagreement (Read "Reaching Consensus from Conflicting Opinions" [6])

Designing the Solution Space

Model proposed solution from User-Centered Design perspective

(Read "User Centered Design for Different Project Types" [7])

Goal is to propose solution to client to gain approval and funding

Elaborate use cases (Read pages 6-13 of "Use Cases" white paper [1])

Define Logical UI Model

Elaborate Logical UI Model into Physical UI Model

Scope project into phases, estimate cost and schedule of each

Identify deliverables

Determine Build verses Buy

Agile Model-Driven Development

(Read "Agile Model-Driven Development (AMDD): The Key to Scaling Agile Software Development [8])

Use cases to understand user workflows and elicit needed features

UML

Class diagrams to model information concepts and interrelationships

Collaboration diagrams to identify key components, interfaces, and time flows

State diagrams to model system states and state transitions

Define system architecture

Define components; allocate responsibilities to each

Using Java as a design specification language

(Read "Java theory and practice: I have to document THAT?" [9])

Define interfaces between components, information passed over interfaces

- Identity information to be persisted
- Identify risks
    - (Read “Early Warning Signs of IT Project Failure: The Dominant Dozen” [10])
    - (Read “Risks with 100% Probability” [11])
- Design for extensibility (aka “future-proof designing”)
    - (Read “The Open-Closed Principle” [12])

Project Management

- (Read “Agile Model-Driven Development [13])
- Elephant in Room – Requirements verses Design verses Code
    - Cost of fixing bugs (Read “Cost of Fixing Software Bugs” [14])
- Allocating effort between requirements-design-implementation
    - (Review “Agile Iterations” [15])
- Iterative phases
    - Build system “skeleton” first to rapidly get something working
    - Incrementally add features “flesh” onto system “skeleton”
        - Focus first on must-have and frequently-used features
        - Defer nice-to-have features to later in project
    - Prototype risks early to “fail fast”
- Sprints
    - From scrum manifesto: *“The heart of Scrum is a Sprint, a time-box of one month or less during which a “Done”, useable, and **potentially** releasable product Increment is created*. “
    - Potentially Shipping product: see definition at https://innolution.com/resources/glossary/potentially-shippable-product-increment
- Scrum
    - *“The Daily Scrum is a 15-minute time-boxed event for the Development Team… At it, the Development Team plans work for the next 24 hours….The Development Team or team members often meet immediately after the Daily Scrum for detailed discussions, or to adapt, or replan, the rest of the Sprint’s work.”*
    - (Read “How to Lead Meetings People Actually Want To Go To” [16])

Architecture Patterns

- Enterprise architecture patterns
- Design patterns (e.g., Gang of Four)

References

1. Steve Nies, 2000, "Use Cases – Methodology for Capturing Functional Requirements" White Paper, Modus Operandi, December 28, http://www.stevenies.com/wp-content/uploads/2019/09/UseCaseOverview.pdf.
2. Kate Eby, 2017, “The Triple Constraint: The Project Management Triangle of Scope, Time, and Cost”, September 20, https://www.smartsheet.com/triple-constraint-triangle-theory
3. Charles Connell, 2013, “It’s Not About Lines of Code”, March 15, http://www.stevenies.com/wp-content/uploads/2019/09/Its-Not-About-Lines-of-Code.pdf
4. Bernie Roseke, 2016, “ACWP (Earned Value Analysis)”, May 20, https://www.projectengineer.net/acwp-earned-value-analysis/
5. Paul Graham, 2008, “How to Disagree”, March, http://www.paulgraham.com/disagree.html.
6. Steve Nies, 2015, “Reaching Consensus from Conflicting Opinions”, February 9, http://www.stevenies.com/team-development-reaching-consensus-from-conflicting-opinions/
7. Jack Scanlon and Lynn Percival, 2002, “User-Centered Design for Different Project Types”, developerWorks, March, http://www.stevenies.com/wp-content/uploads/2019/09/userCenteredDesign.pdf
8. Scott Ambler, 2012, “Agile Model Driven Development (AMDD): The Key to Scaling Agile Software Development”, http://www.stevenies.com/wp-content/uploads/2019/09/Agile-Model-Driven-Development-AMDD_-The-Key-to-Scaling-Agile-Software-Development.pdf
9. Brian Goetz, 2002, “Java Theory and Practice: I Have to Document THAT?”, developerWorks, August, http://www.stevenies.com/wp-content/uploads/2019/09/goodJavaDoc.pdf
10. Leon Kappelman, Robert McKeeman, Lixuan Zhang; 2006, “Early Warning Signs of IT Project Failure: The Dominant Dozen”, http://www.stevenies.com/wp-content/uploads/2019/09/projectFailure.pdf
11. Dr. David Hillson, 2014, “Risks with 100% Probability”, Executive Brief, http://www.stevenies.com/wp-content/uploads/2019/09/Risks-with-100-Probability.pdf
12. *Author Unknown*, “The Open-Closed Principle”, http://www.stevenies.com/wp-content/uploads/2019/09/Open-closed-principle.pdf
13. Steve Nies, 2002, “Agile Model-Driven Development”, http://www.stevenies.com/wp-content/uploads/2019/09/amdd.pdf
14. *Author Unknown*, 2009, “Cost of Fixing Software Bugs”, November 25, http://www.stevenies.com/wp-content/uploads/2019/09/Cost-of-Fixing-Software-Bugs.pdf
15. Steve Nies, 2018, “Agile Iterations”, http://www.stevenies.com/wp-content/uploads/2019/09/agileIterations.pdf
16. H.V. MacArthur, 2018, “How to Lead Meetings People Actually Want To Go To”, December 13, http://www.stevenies.com/wp-content/uploads/2019/09/Good-Meetings.pdf

Further Reading

1. *Various Authors*, 2009, "97 Things Every Software Architect Should Know", O’Reilly, http://www.stevenies.com/wp-content/uploads/2019/09/97-Things-Every-Software-Architect-Should-Know.pdf
2. Amy Mortensen, 2015, “The Hidden Cost of Context Switching”, July 10, http://www.stevenies.com/wp-content/uploads/2019/09/The-Hidden-Cost-of-Context-Switching.pdf