Workshop #7
What Does Technical Debt Mean at a System Level?

Bob Epps/Garry Roedler
Lockheed Martin Corporation
PSM User's Group Conference
July 14, 2011

Technical Debt Workshop for Systems Engineering

Agenda

- Management of Technical Debt- Steve McConnell
- Technical Debt Observations- Jim Highsmith
- Types of Debt- Chris Sterling
- Break
- Workshop Exercise # 1-Identifying Technical Debt
- Management of Architectural Debt- Ipek Ozkapa
- Workshop Exercise # 2- Architecture & Technical Debt
- Workshop Exercise # 3- System Design & Technical Debt
- Workshop Summary/Action Plan

Technical Debt

"Management of Technical Debt", Steve McConnell

– " 'Technical Debt' refers to the delayed technical work that is incurred when technical short cuts are taken, usually in pursuit of calendar driven software schedules. Just like financial debt, some technical debt can serve valuable business purposes. Other technical debts are simply counter productive. The ability to take on debt safely, track their debt, manage their debt and pay down their debt varies among organizations. Explicit decision making before taking on debt and more explicit tracking of debt are advised"

Technical Debt

"Management of Technical Debt". Steve McConnell

– " 'Technical Debt' refers to the delayed technical work that is incurred when technical short cuts are taken, usually in pursuit of calendar driven software schedules. Just like financial debt, some technical debt can serve valuable business purposes. Other technical debts are simply counter productive. The ability to take on debt safely, track their debt, manage their debt and pay down their debt varies among organizations. Explicit decision making before taking on debt and more explicit tracking of debt are advised"

Technical Debt

"Management of Technical Debt" Steve McConnell

- " The term 'technical debt' was coined by Ward Cunningham to describe the obligation that a software organization incurs when it chooses a design or construction approach that's expedient in the short term but that increases complexity and is most costly in the long term"
- There are two kinds of technical debt
  - Type I, Debt incurred unintentionally
    - Inexperienced individuals produces error prone results
  - Type II, Debt incurred intentionally
    - Conscious decision to optimize for the "present" rather than the "future"

Technical Debt

"Management of Technical Debt", Steve McConnell

- **Type II, Debt incurred intentionally**
  - "Short-Term" Debt (Type II.A)
    - » A company takes on a short term debt when it has the money; it just does not have it now.
    - » Short term debt is expected to be paid off frequently
    - » Focused Short-Term Debt(Type II.A.1)
    - » Unfocused Short-Term Debt (Type II.A.2)
      - Should be avoided
  - "Long-Term" Debt(Type II.B)
    - » A company takes on strategically and proactively
    - » Primary rationale is that the development work "today" is seen as more expensive than the cost in the future.
    - » Example:
      - Responding to "Time to Market" pressures
      - Preservation of Startup capital
      - Delaying Development expense
  - Debt Service
    - » The "interest" charged for incurring the debt

Technical Debt

"Management of Technical Debt", Steve McConnell

Summary of Kinds of Debt

Non Debt

Features backlog, deferred features, cut features, and so on. Not all incomplete work is debt. These are not debt because they do not require interest payments

Debt

I**. Unintentional Debt**. Debt incurred unintentionally due to low quality

II. **Intentional Debt**. Debt incurred intentionally

II.A Short-Term Debt. Short Term Debt, usually incurred reactively, for tactical reasons

II.A.1 Focused Short Term Debt. Individually identifiable shortcuts(like a car loan)

II.A.2 Unfocused Short-Term Debt. Numerous tiny shortcuts(like a credit card)

II.B Long-Term Debt. Long-term debt, usually incurred proactively, for strategic reasons

Technical Debt

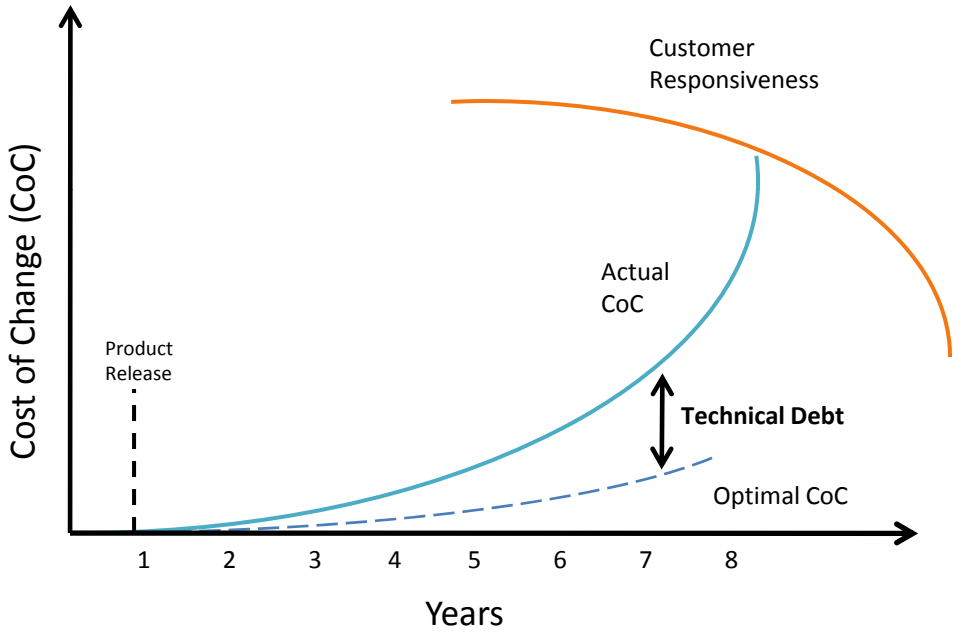
"Management of Technical Debt", Steve McConnell

Communicating about Technical Debt

- Shift from Technical vocabulary to a Financial vocabulary
- Use a projects Maintenance budget as a rough proxy for its technical debt service load
- Discuss Debt in terms of "money" instead of "features"
- Be sure you're taking the right kind of debt
- Treat the discussion of Debt as an ongoing dialog rather than a single discussion

Technical Debt Observations

"Agile Project Management", Jim Highsmith, second edition



Technical Debt Observations

"Agile Project Management", Jim Highsmith, second edition

- "When product development teams give lip service to technical excellence, when project and product managers push teams beyond quickness into hurrying, technical debt is incurred"(pp 216)
- "Technical debt can arise during initial development, ongoing maintenance (keeping a product at its original state), or enhancements (adding functionality)."(pp 216)

When does the insertion of Technical Debt have its greatest impact?

Technical Debt Observations

"Agile Project Management", Jim Highsmith, second edition

- "Without a firm dedication to long-term technical debt management, development groups are pressured into increasing technical debt trap. As the debt gets worst, the delays become greater. As the delays lengthen, the pressure increases, usually leading to another hurried implementation, which increases the technical debt yet again."(pp 217)

Technical Debt Observations

"Agile Project Management", Jim Highsmith, second edition

- "It must be noted that managing technical debt does not keep products from becoming obsolete. A technical debt strategy does not attempt to stave off eventual obsolescence, but keeps the cost of change low so that customer responsiveness remains as high as possible during a product life."(pp 217)

Types of Debt

"Managing Software Debt: Building for Inevitable Change", Chris Sterling

- Software Debt
  - Composed of the following forms of "Debt"
    - Technical Debt, Quality Debt, Configuration Management Debt, Design Debt & Platform Debt
  - Indicators of Software debt are the following:
    - Do you have "like-to-like" migration?
    - Do you have "limited expertise" available?
    - Do you have "expensive release stabilization" phases?
    - Do you have "increased cost for regression testing" your software assets?

Types of Debt

"Managing Software Debt: Building for Inevitable Change", Chris Sterling

- Technical Debt
  - These are activities that a team or team members choose not to do well now and will impede future development if left undone
- Quality Debt
  - There is a diminishing ability to verify the functional and technical quality of software
- Configuration Management Debt
  - Integration and release management becomes more risky, complex and error-prone
- Design Debt
  - The cost of adding features is increasing toward the point where it is more than the cost of writing from scratch.
- Platform Debt
  - The availability of people to work on software changes is becoming limited or cost-prohibitive.

Types of Debt

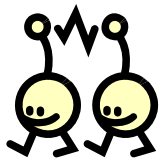
"Managing Software Debt: Building for Inevitable Change", Chris Sterling

- Technical Debt
  - "..the decay of component and inter-component behavior when the application functionality meets a minimum standard of satisfaction for its users"
  - Produced by work patterns:
    - Schedule pressure
      - "Excessive pressure causes team to take short cuts to meet expectation of management and customer"
    - Duplication
      - "..because of cut-and-paste tactics results in teams making even simple changes in more than one place."
    - Mentality of getting it right the first time
      - "..incorrect assumptions about what we can know about the future."
  - "..payoff Technical Debt immediately, insert strategically placed runtime exceptions, and add technical debt to the Product Backlog."

Relationship to SW Technical Debt?

Examples of Technical Debt

- Design
  - Inconsistent approaches
  - Poor choice of components/frameworks
  - Hindsight design
- Code
  - Duplicate code
  - Overly complex modules
  - Standards violations
  - Lack of documentation
  - Style drift or clash
- Test
  - Incomplete coverage
  - Hard to maintain tests
  - Excessive tests



Today's tools make it easy to identify and measure many of these forms of technical debt.

- Do these examples apply to Systems?
  - All design items here are valid to systems
  - Although code does not apply, the sub-bullets apply if it were for requirements
  - Test items apply, but could be augmented to V&V
- Are there other sources of Technical Debt?
  - Could go through the system life cycle and determine where shortcuts are often taken

Types of Debt

"Managing Software Debt: Building for Inevitable Change", Chris Sterling

- Quality Debt
  - "To sustain the internal quality of software, teams must approach development in a disciplined way. Common approaches that teams use to sustain internal quality are the following:
    - Sustainable pace
    - Early identification of internal quality problems
    - Close collaboration
    - Refactoring
    - Small batches of work
    - Defining technically done
    - Potentially shippable product increments
    - Single work queue"

Types of Debt

- Thoughts about:
  - Configuration Management Debt
    - Were shortcuts taken in Configuration Control?
    - Lack of version control
    -
  - Design Debt
    - Adapting the existing design to meet new requirements costs more than a new design
    -
    -
  - Platform Debt
    - Look for SW solution to compensate for platform inadequacies
    -
    -

Relationship to SW Requirements Debt?

Examples of Requirements Debt

TBDs and TBRs late in lifecycle

Missing TBDs and TBRs

Overly detailed requirements

Obsolete requirements

- Do these examples apply to Systems?
  - Each apply
  - TBDs/TBRs late drive unintended change
  - Missing TBDs/TBRs drive potential rework due to missed analysis
  - Detailed requirements cross the boundary into design
- Are there other examples of Requirements Debt?
  - Operational Concepts?
    - Incomplete
    -
  - Stakeholder/mission requirements?
    - Same as the requirements debt here, but different perspective
    - Can come to contractor already with technical debt

Relationship to SW Decision Debt?

Decision Debt

Decisions made too late

Decisions made too early

Decisions based on inadequate information

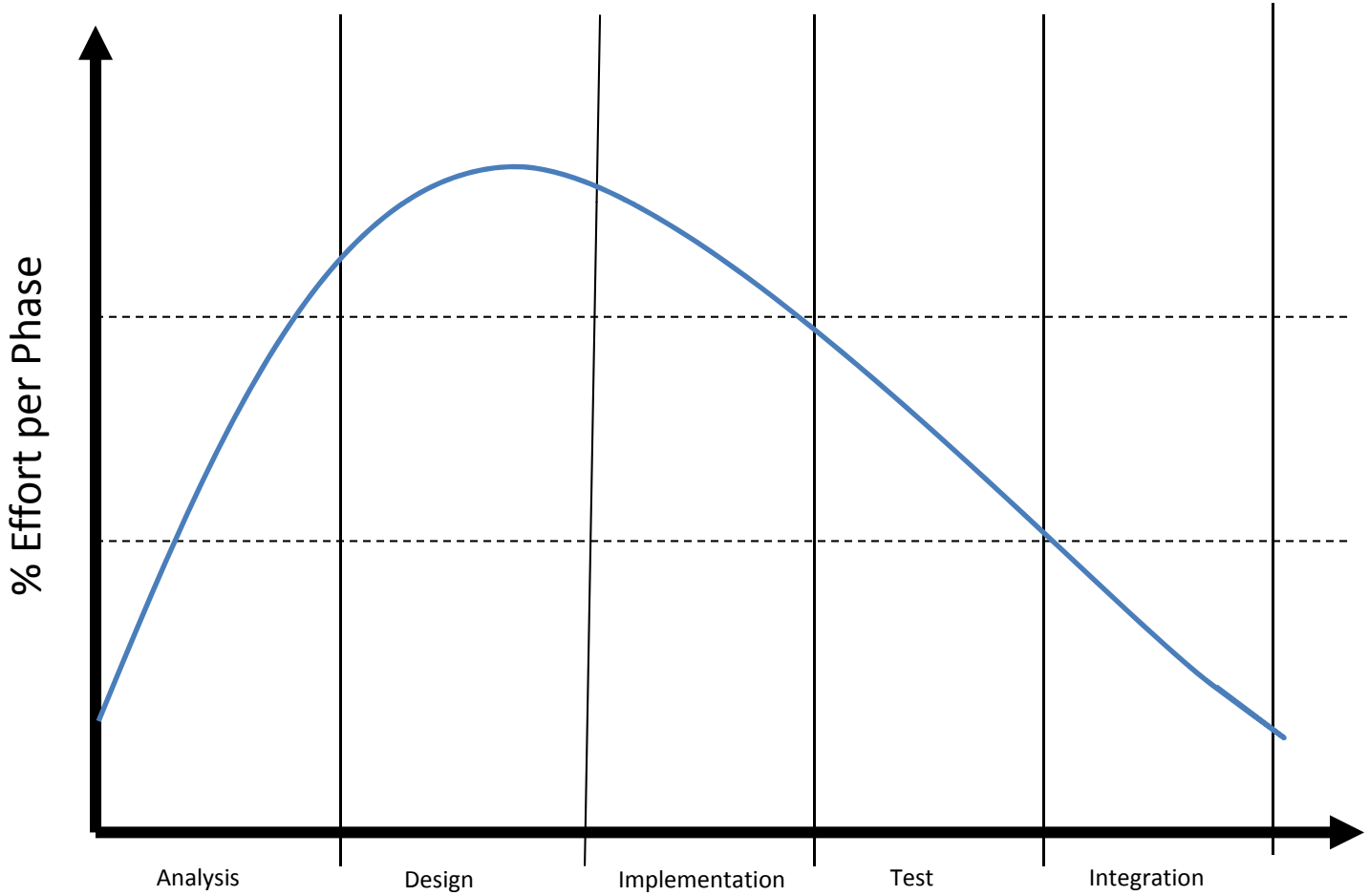
What is the level of confidence that you have in your decisions?

- Do these examples apply to Systems?
  - All apply
- What are impacts of decisions for:
  - Technology selection?
    - Too early – can provide obsolete/non-supportable solution
    - Too late – miss schedule
    - Inadequate info – choose wrong or immature technology
  - Manufacturability?
    -
    -
  - Sustainability?
    -
    -

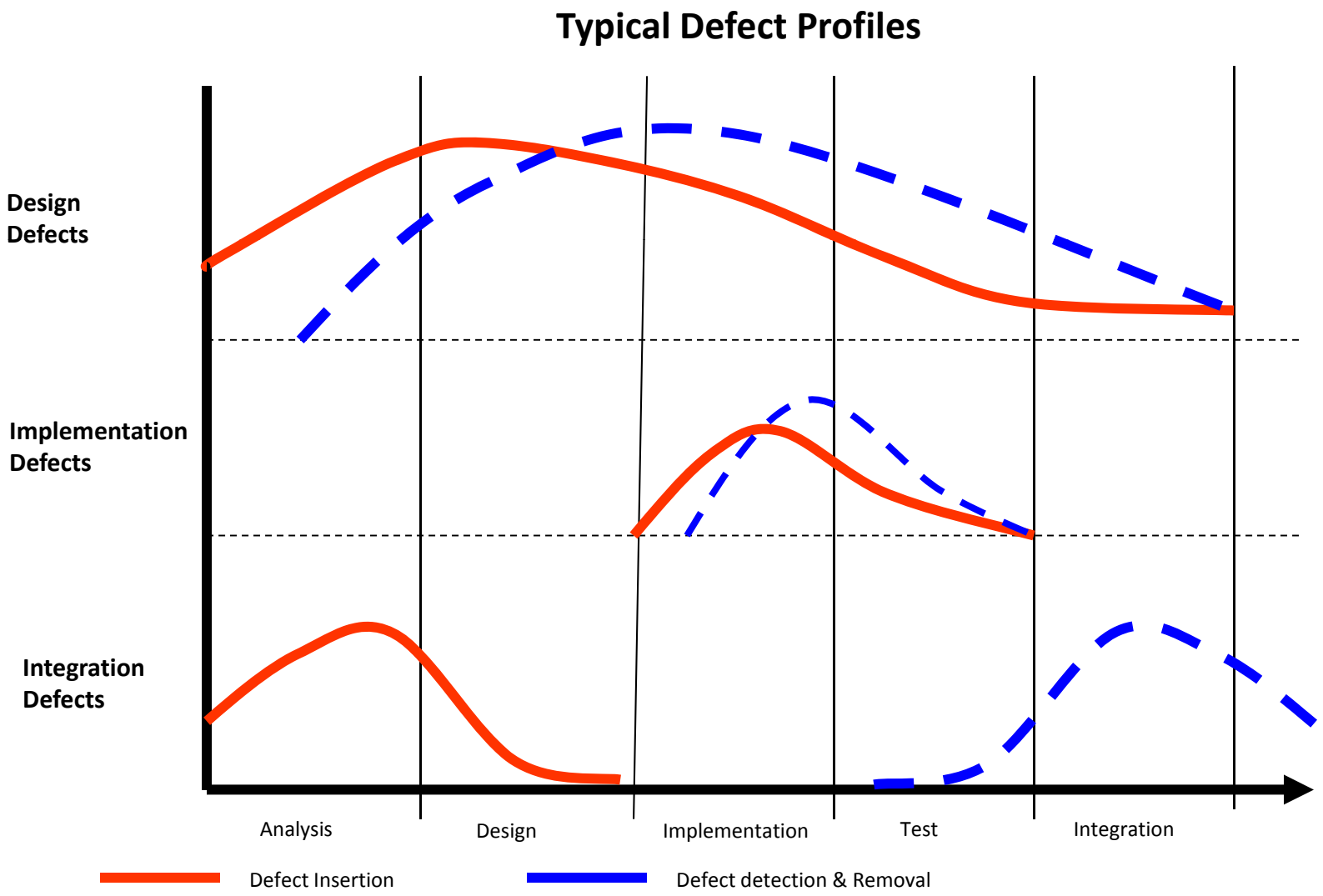
Workshop Exercise # 1
Identifying Technical Debt

- Development profile (Perfect World)
- Defect Density profile
- Development profile(Real World)
- Mapping Development Profiles

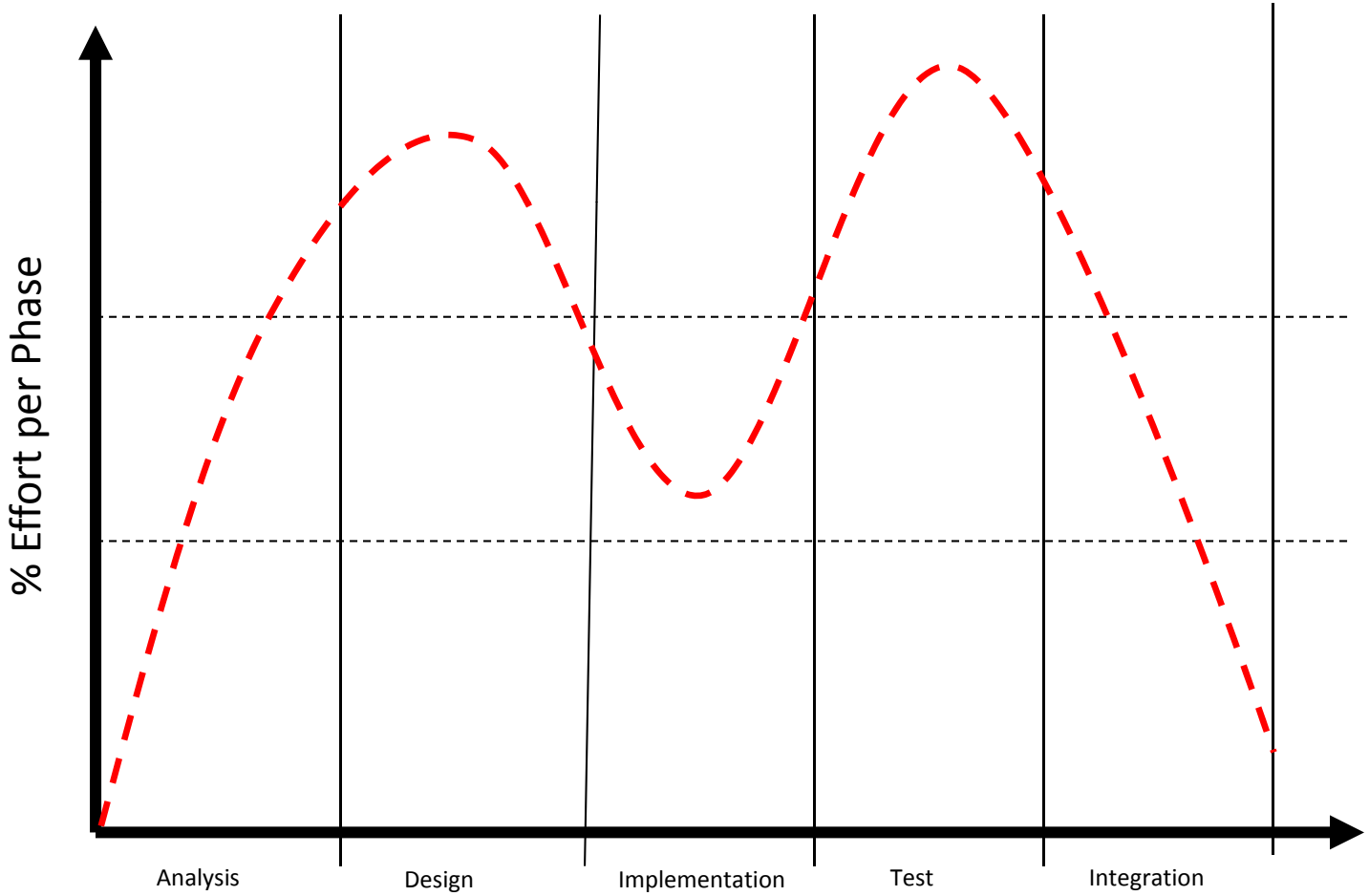
Development Cost(Perfect World)



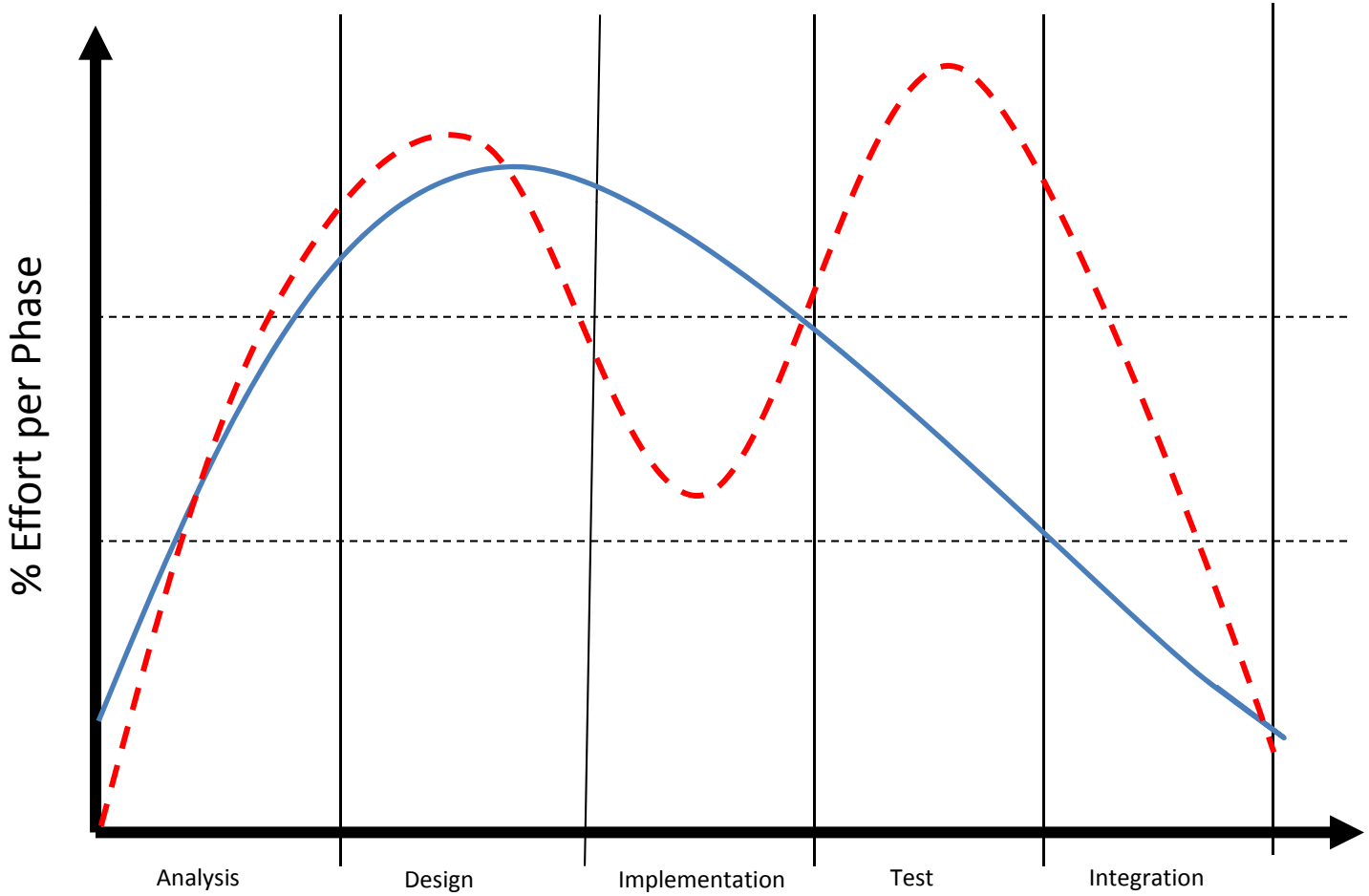
Classification of Defects



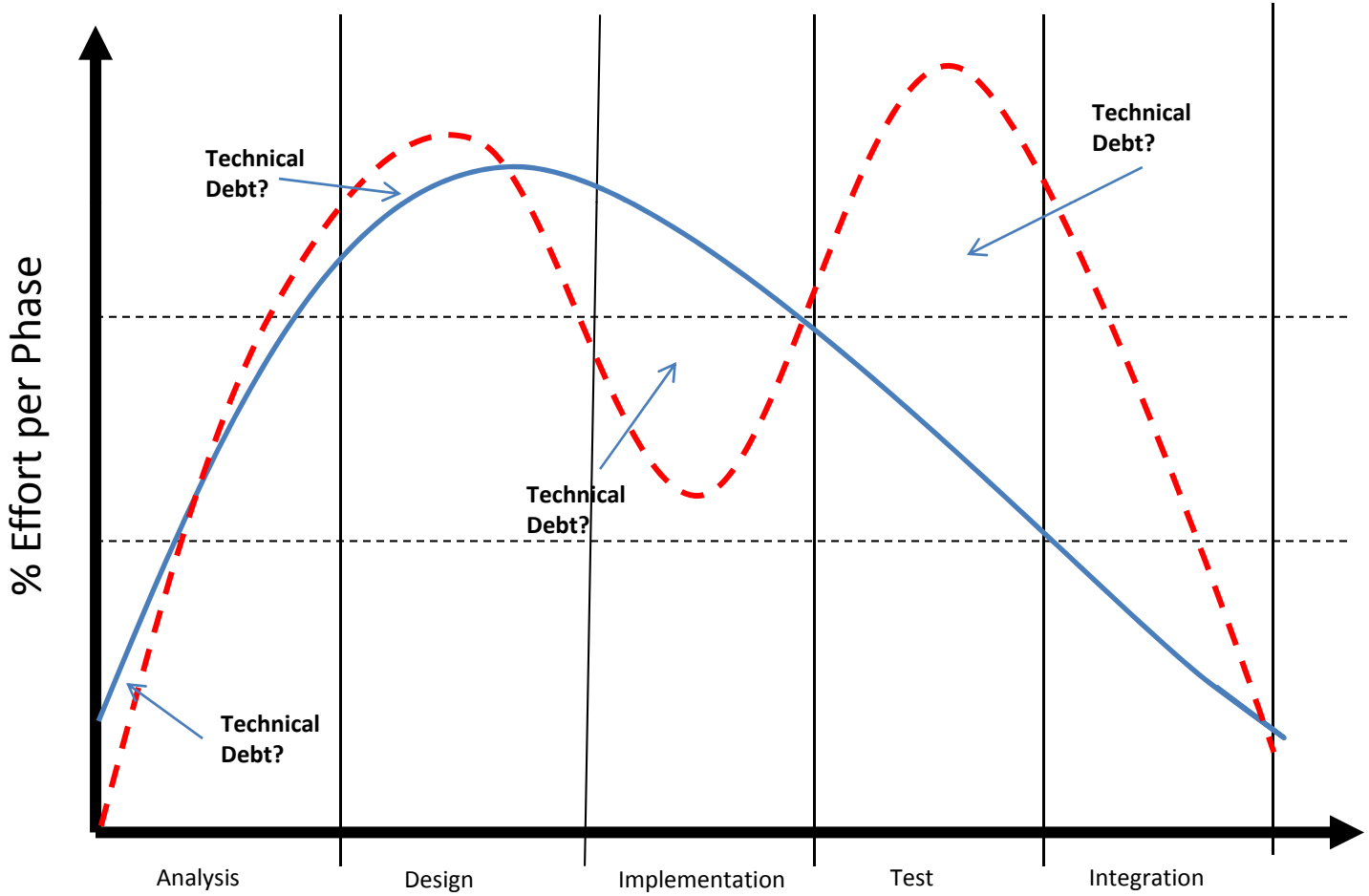
Development Cost(Real World)



Development Cost

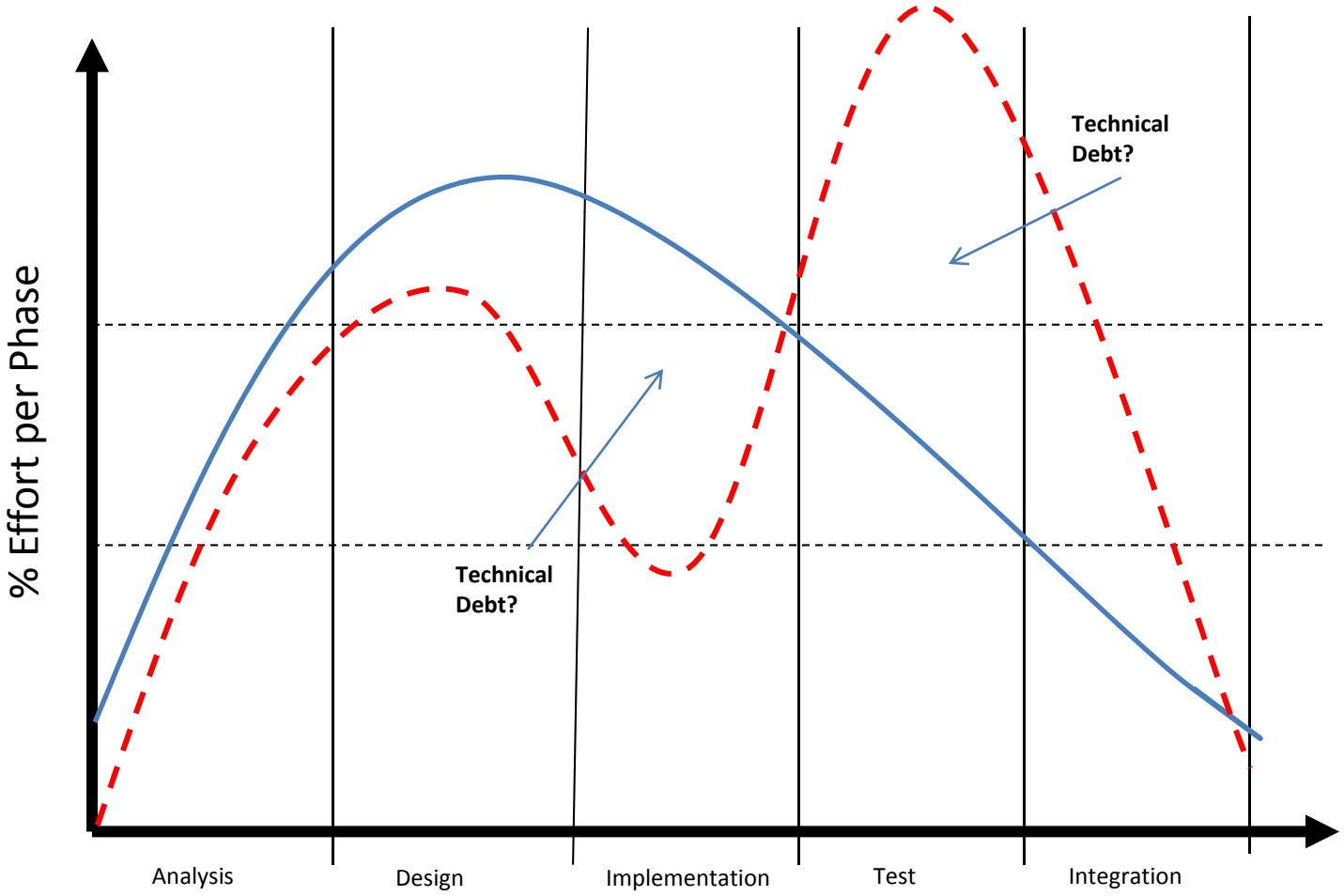


Development Cost



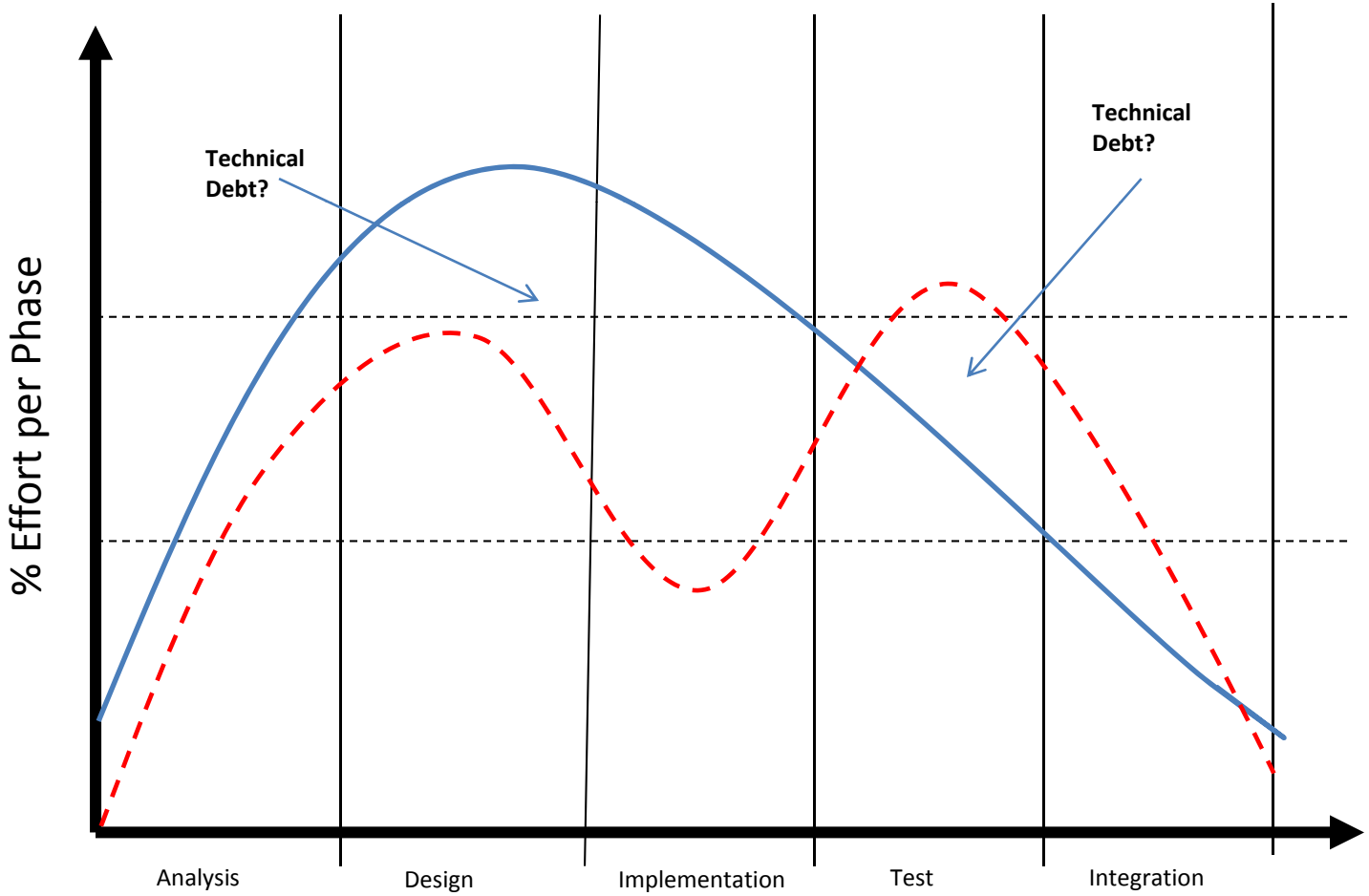
Development Cost

Better or Worse?

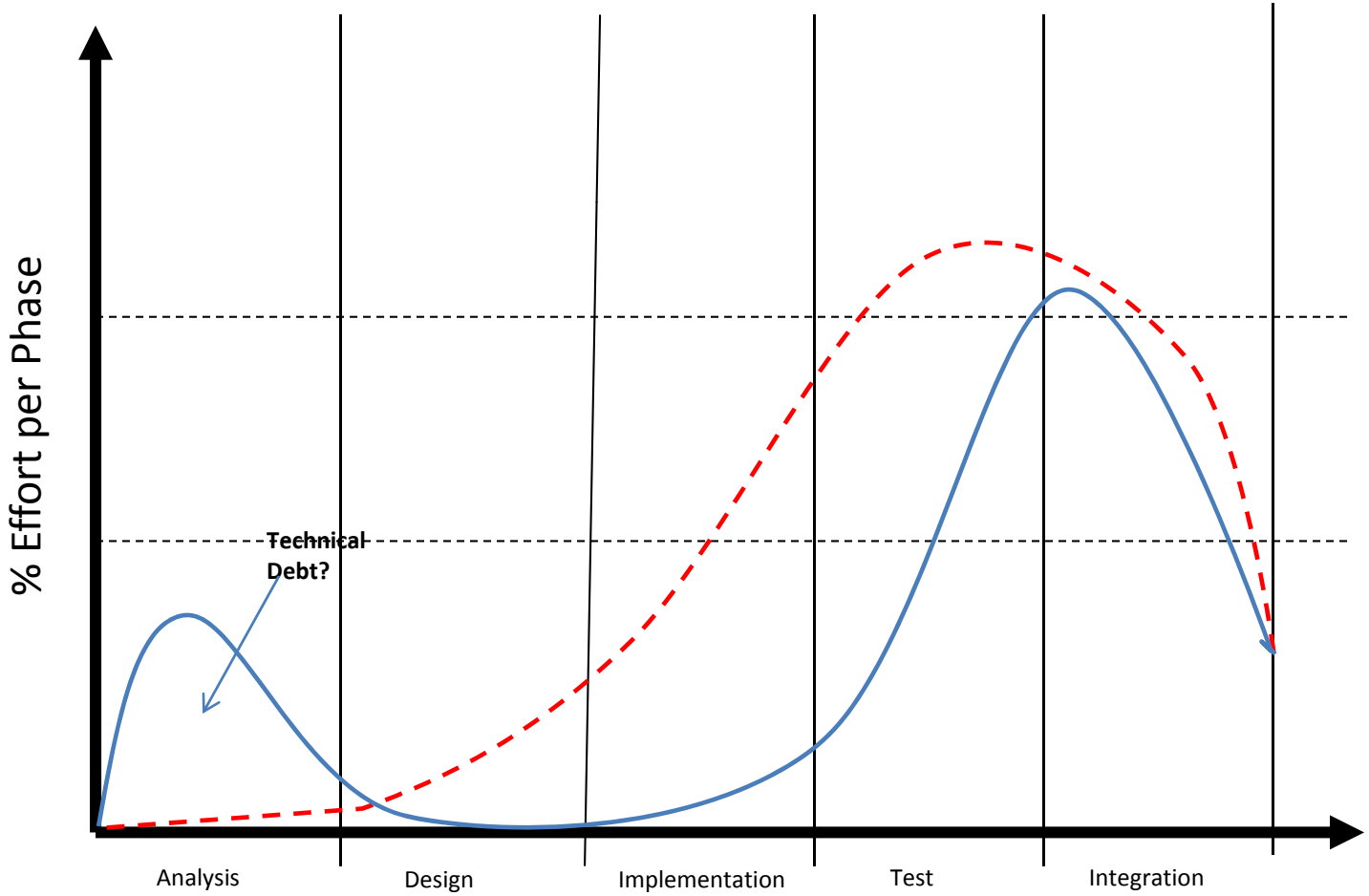


Development Cost

Better or Worse?



COTS Integration



Technical Debt

"Enabling Agility by Strategically Managing Architectural Technical Debt", Ipek Ozkaya

- "Practices intended to speed up the delivery of value to users, however, often result in high rework costs that ultimately offset the benefits of faster delivery, especially when good engineering practices are forgotten along the way. The rework and degrading quality often is referred to as technical debt"
- "For example, through our work on architecture-centric engineering, we often encounter projects that defer modifiability requirements, specifically portability."

Technical Debt

"Enabling Agility by Strategically Managing Architectural Technical Debt", Ipek Ozkaya

- "Our current work focuses on architectural technical debt, which involves architectural decisions made to defer necessary work during the planning or execution of software projects, such as short-cuts taken in designing the structure of the system that may require rework."
- "We are particularly interested in identifying the measureable aspects of architectural technical debt by exploring dependency analysis"

Technical Debt

"Enabling Agility by Strategically Managing Architectural Technical Debt", Ipek Ozkaya

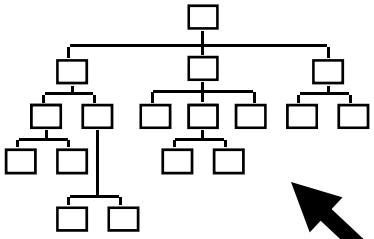
- "By the end of this project, we will produce a model for managing technical debt that will allow the incurrence of some debt to increase delivery tempo when needed, but prevent too much accumulation, which would impede the ability to deliver."

Reference: "Enabling Agility through Architecture",
Nanette Brown, Robert Nord, Ipek Ozkaya; CrossTalk-Nov/Dec 2010

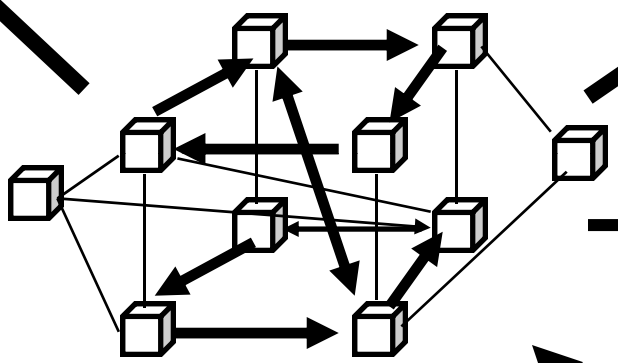
Workshop Exercise #2
Architecture & Technical Debt

Notional Architecture Process

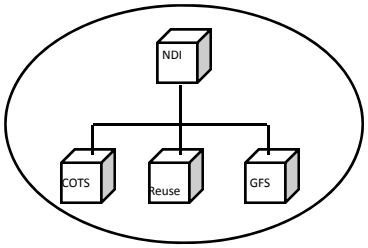
WBS



Architecture



NDI Products



Cost Estimate

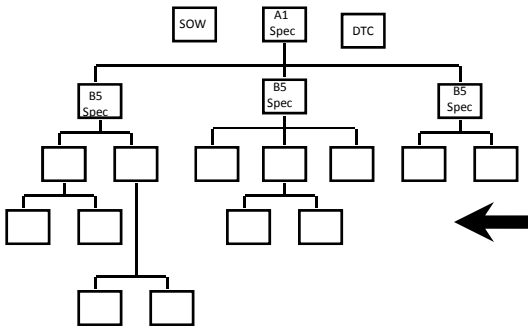
	Low	Prob	High
Segment A			
Segment B			
Segment D			
Segment H			
Segment I			

Risks

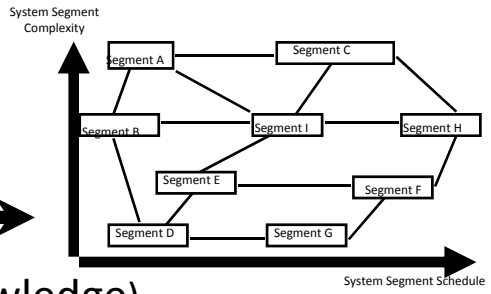
	Schd	Cost	Tech
Risk A	L		
Risk B			M
Risk C	M		
Risk D			H
Risk E		H	

Where can Technical Debt be incurred in this process?

System Requirements

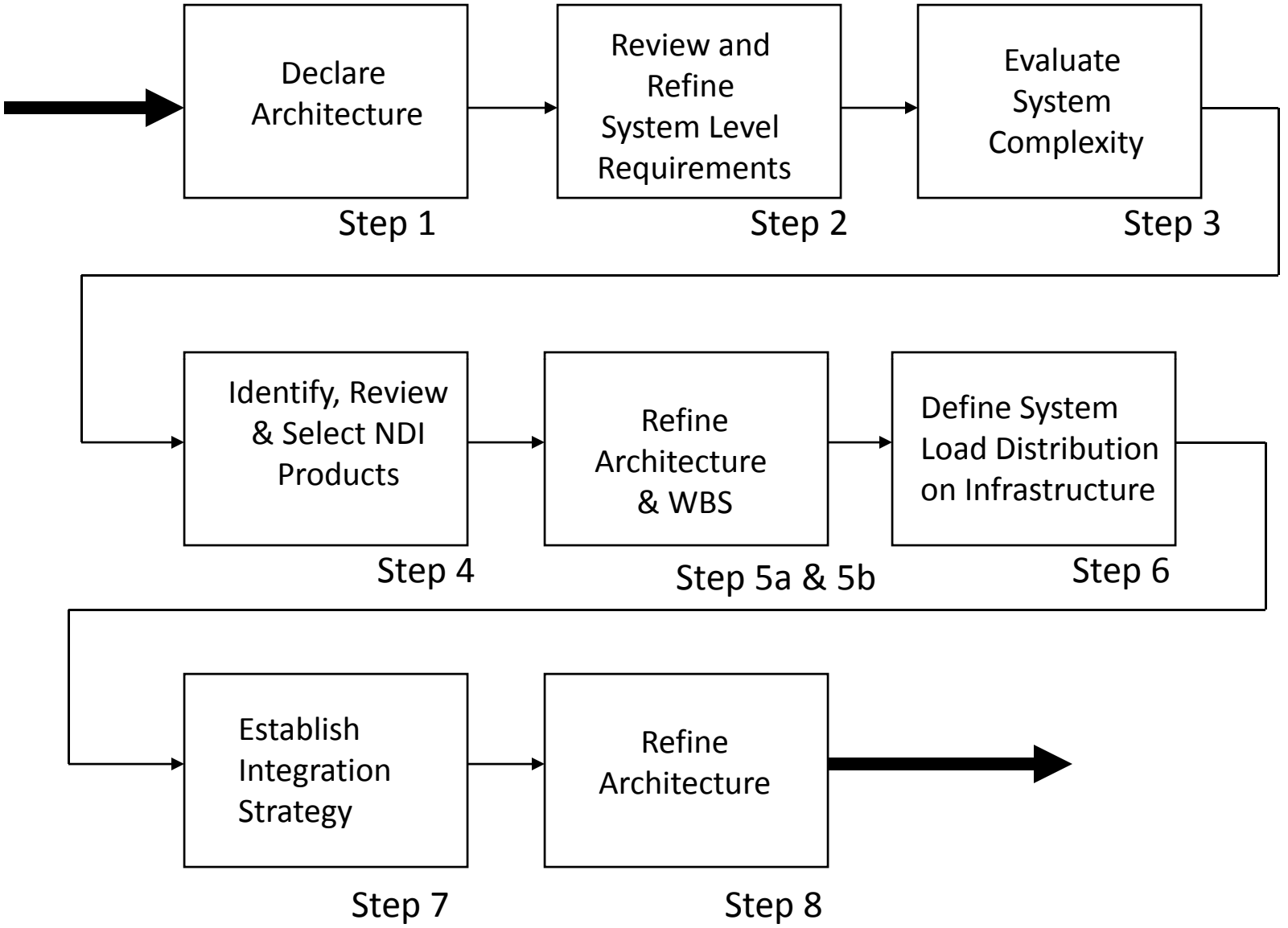


System Integration Plan

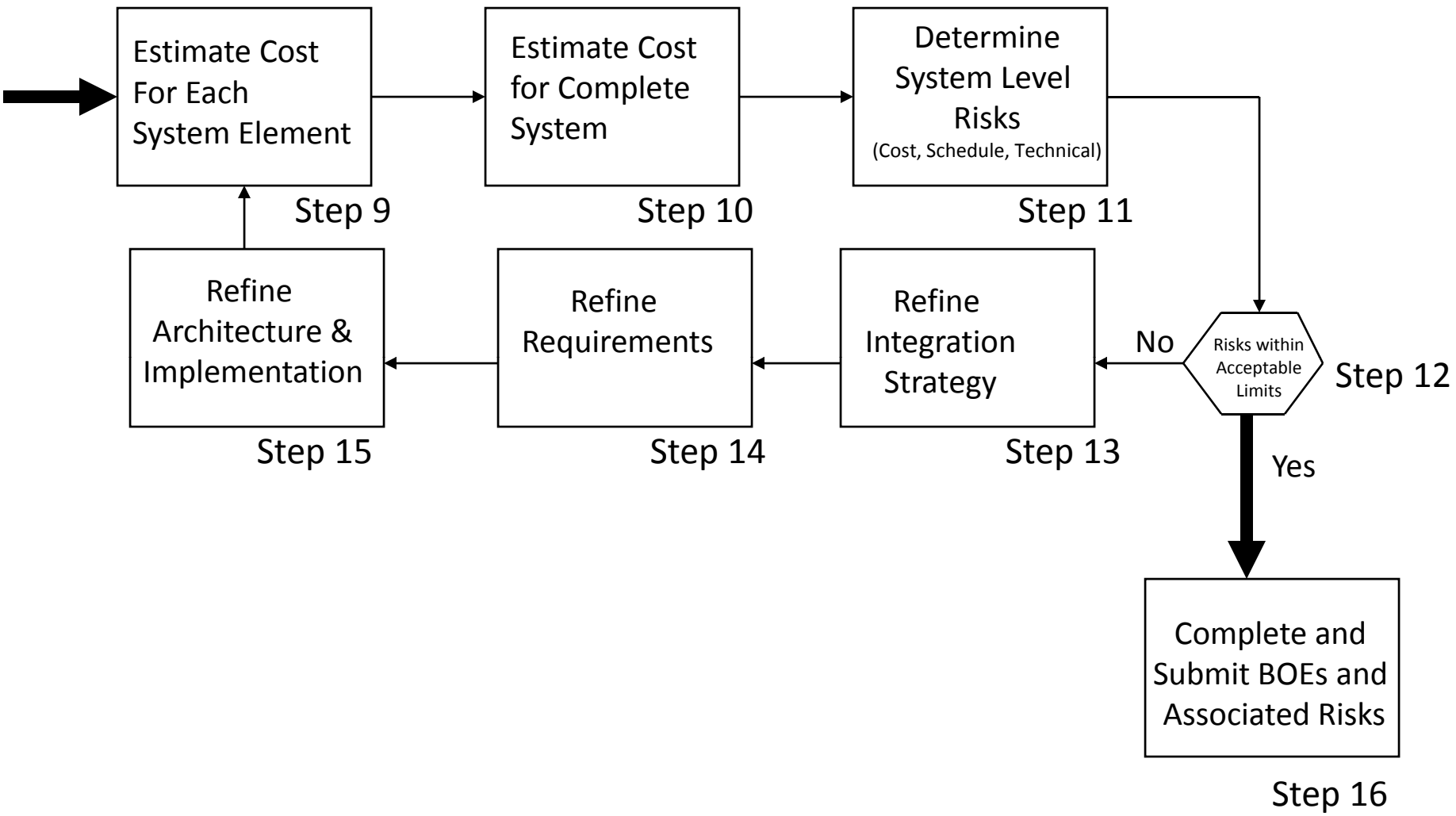


(Domain Knowledge)

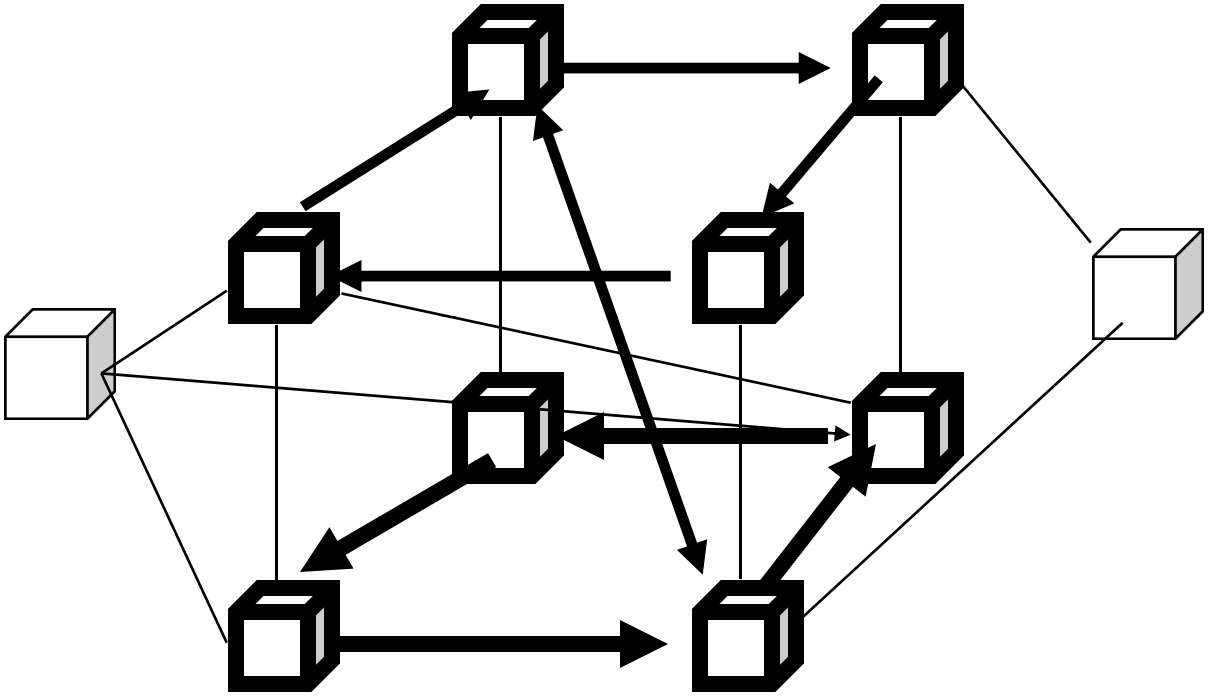
Notional Architecture Process



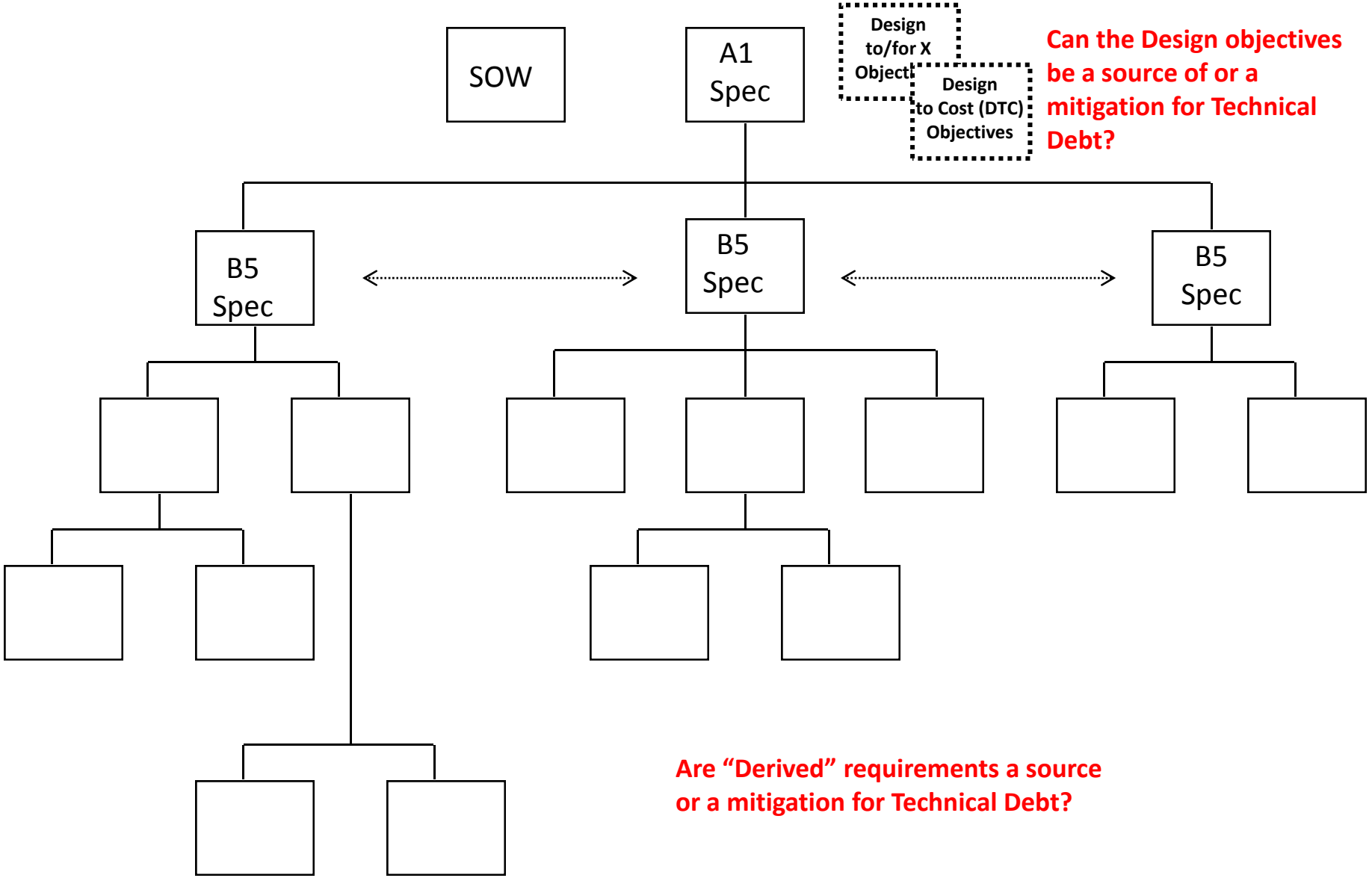
Notional Architecture Process (continuation)



Step 1 Declare Architecture



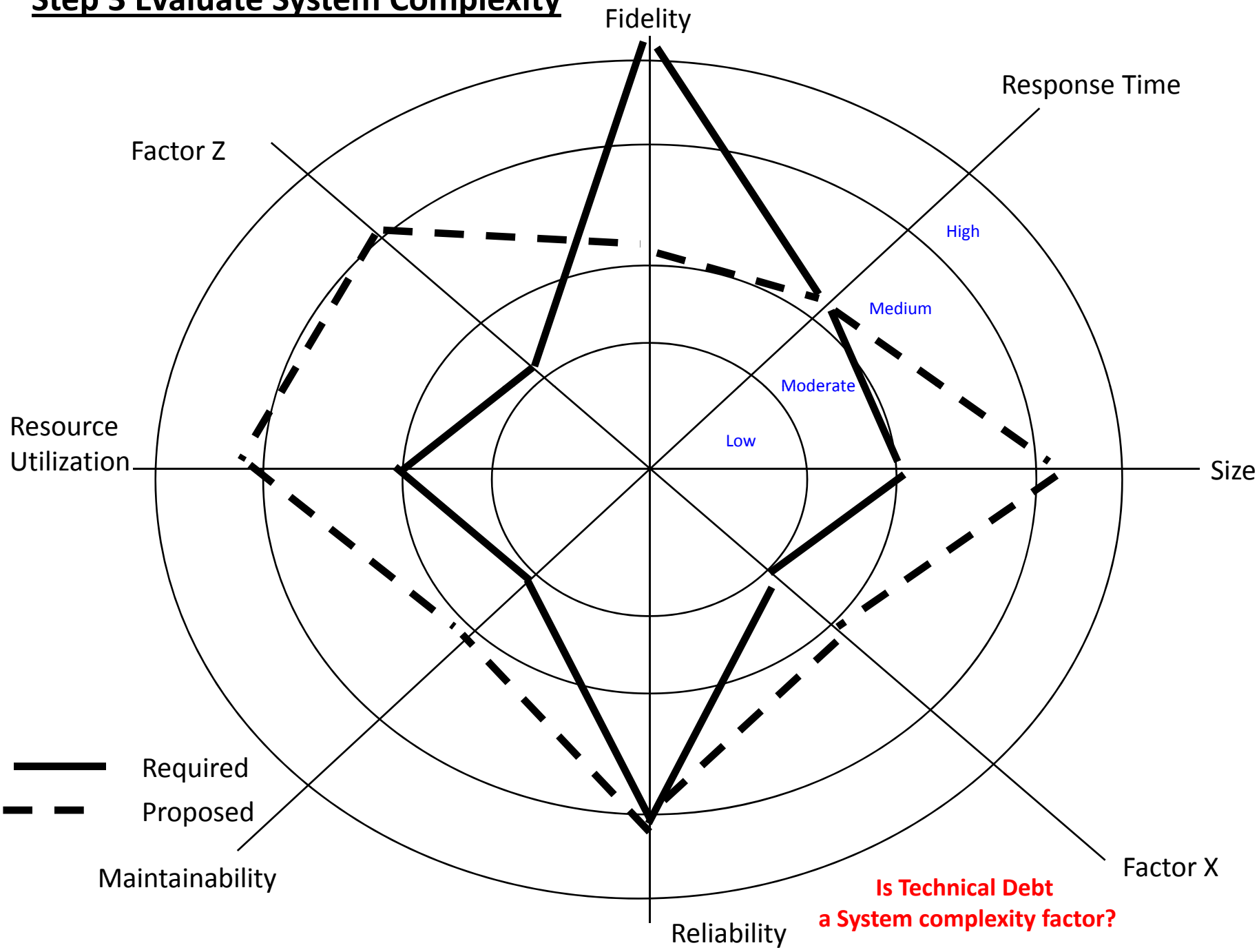
Steps 2 & 14 Review & Refine System Requirements



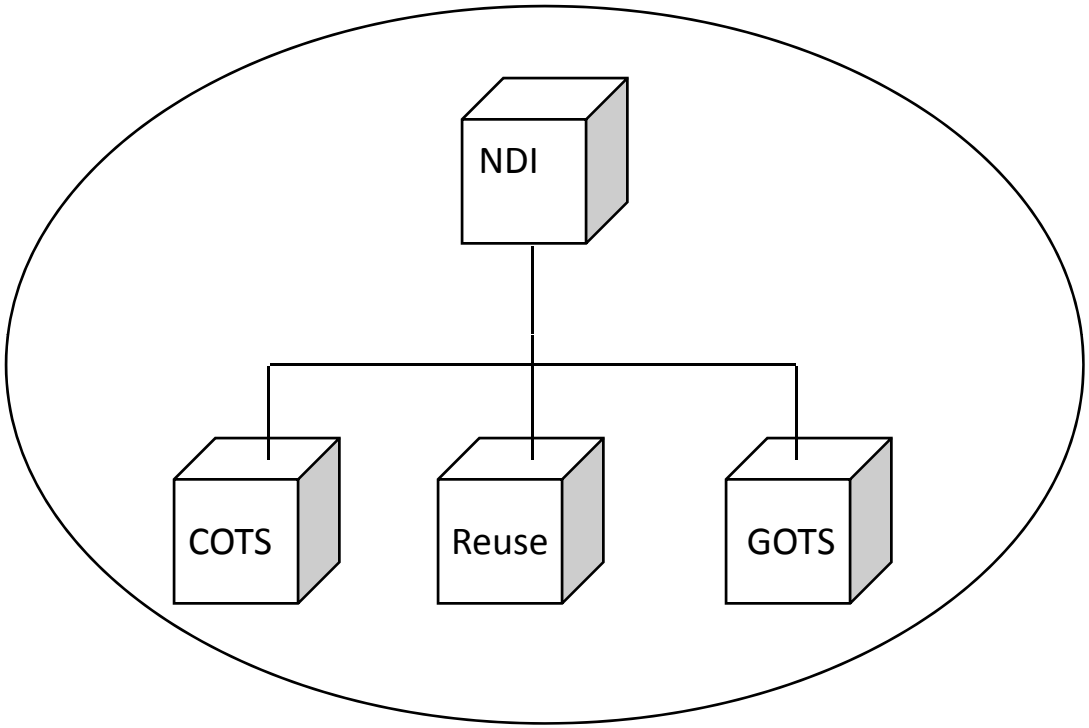
Can the Design objectives be a source of or a mitigation for Technical Debt?

Are "Derived" requirements a source or a mitigation for Technical Debt?

Step 3 Evaluate System Complexity

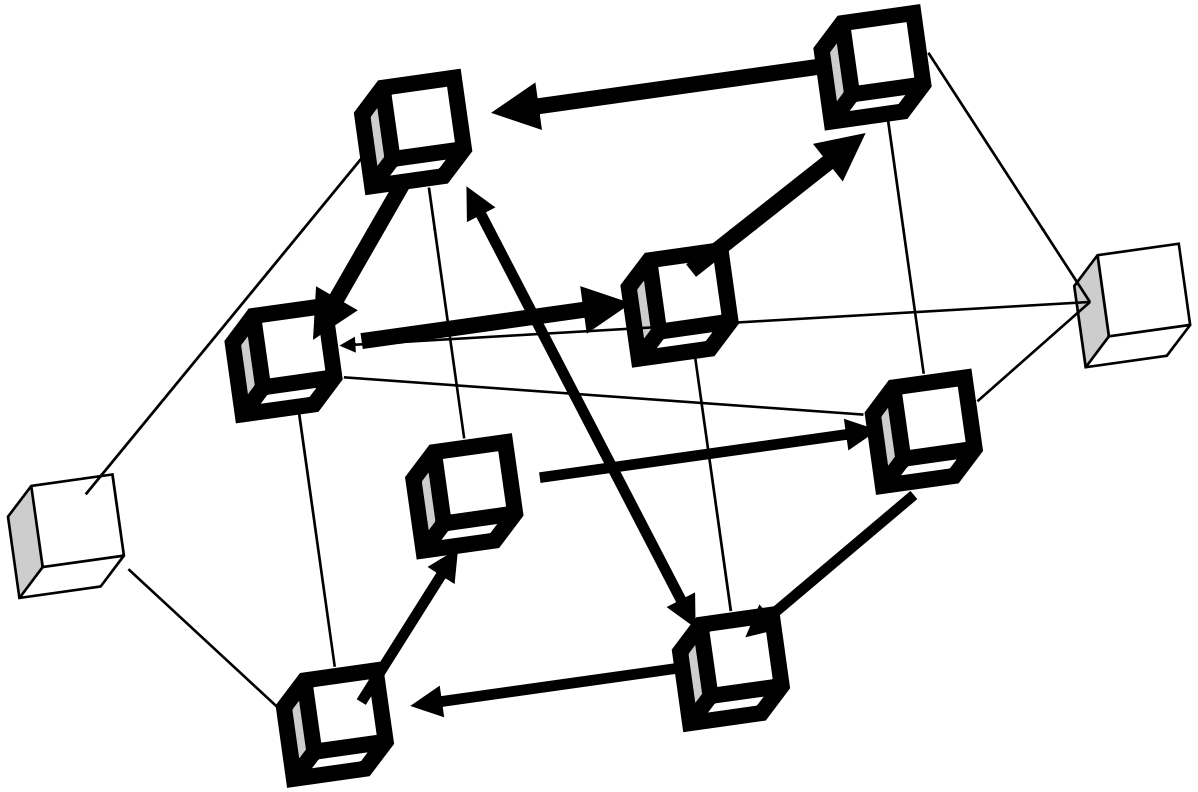


Step 4 Identification, Review & Selection of NDI Products

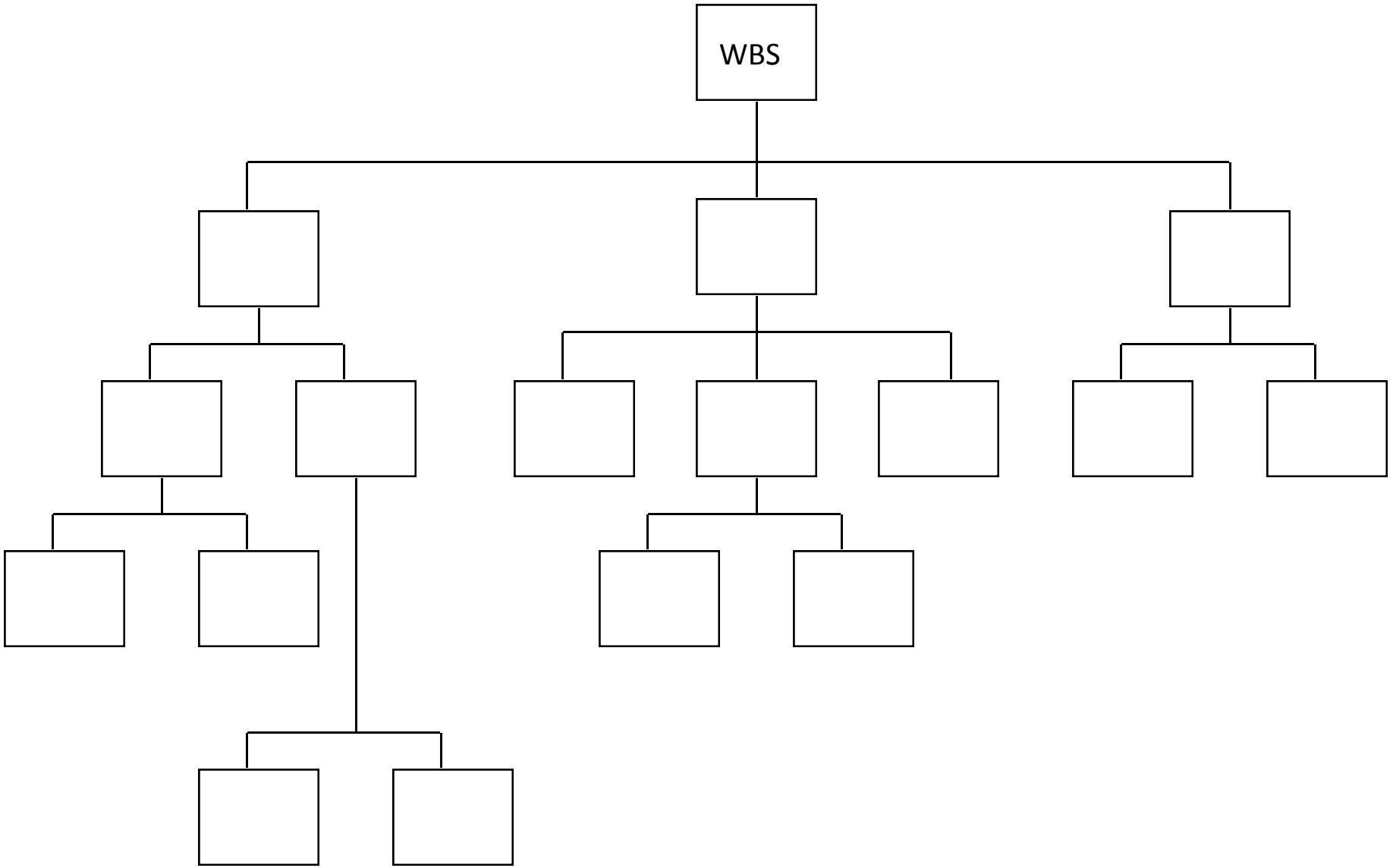


Does selection of NDI incur Technical Debt?

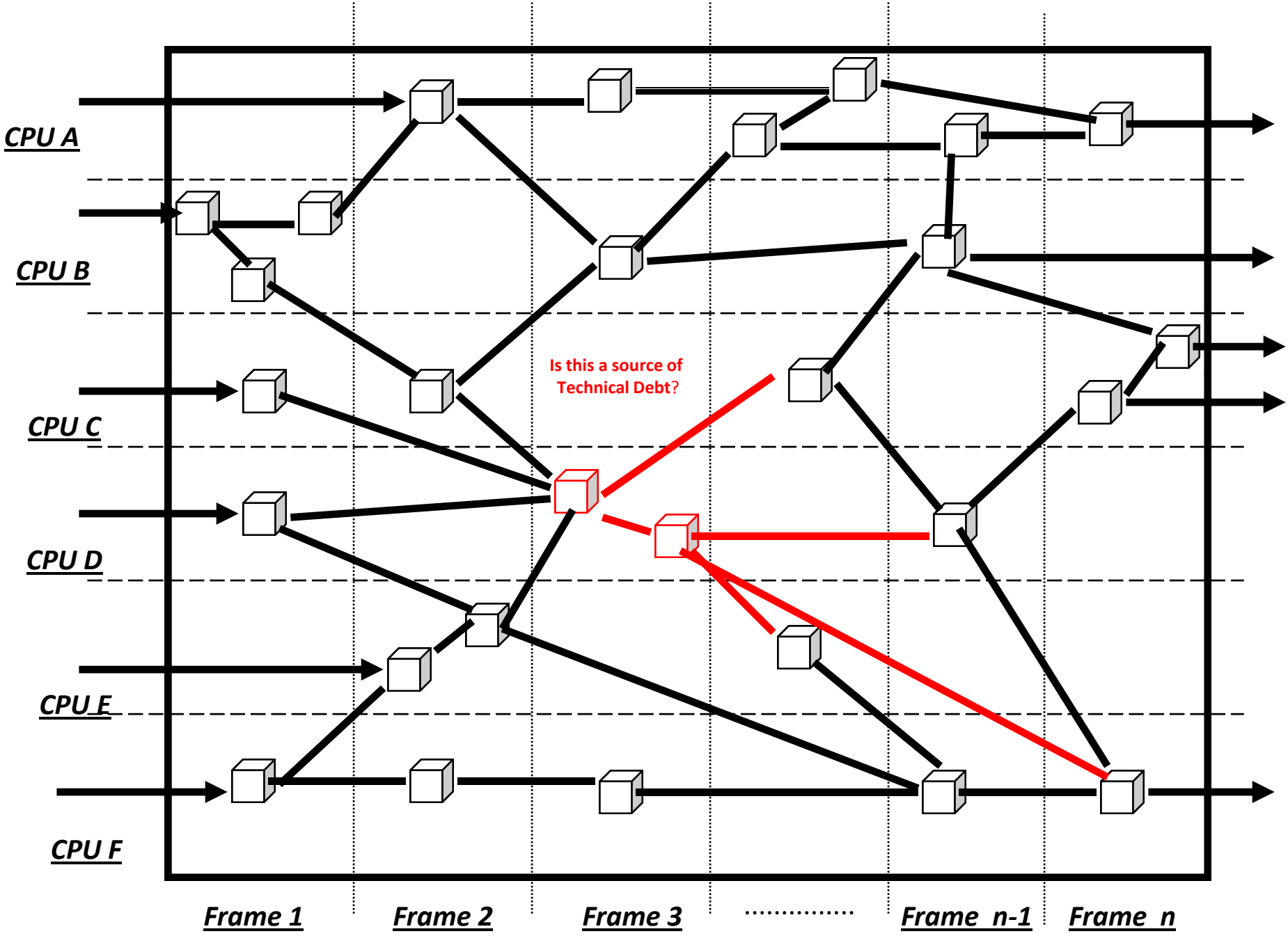
Steps 5a, 8 & 15 Refine Architecture



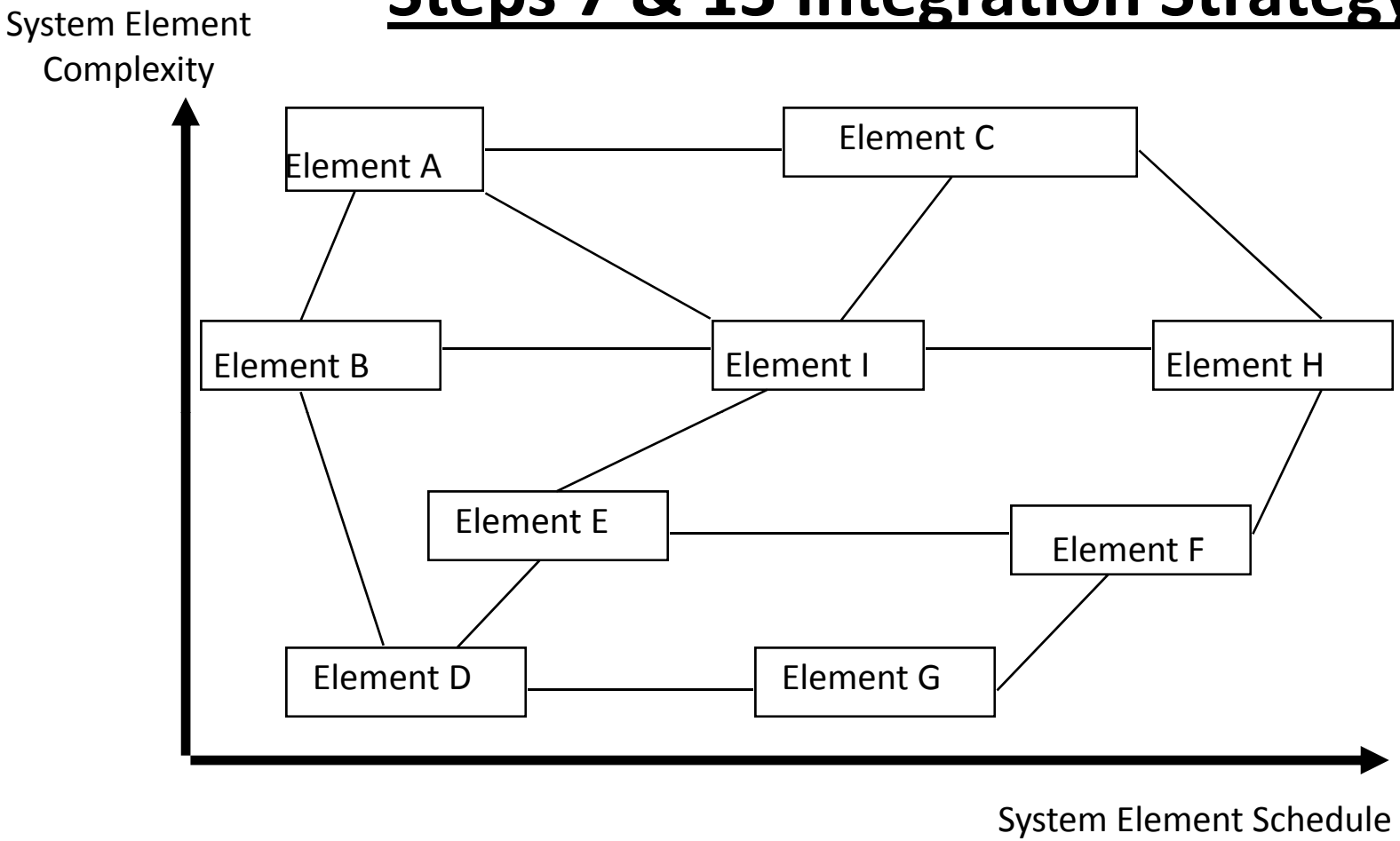
Step 5b Refine WBS



Step 6 Load Build Plan



Steps 7 & 13 Integration Strategy/Plan



What type of Technical Debt is incurred if the Integration Strategy is ill-defined?

Step 9 System Element Level Cost Estimate

Cost Estimate				
	Low	Prob	High	Estimate
Analysis				
Design				
⋮				
Integration				
Total				

$$\text{Estimate} = \frac{\text{High} - \text{Low} + 4(\text{Prob})}{6}$$

Step 10 System Level Cost Estimate

Cost Estimate				
	Low	Prob	High	Estimate
Element A				
Element B				
⋮				
Element E				
Integration				

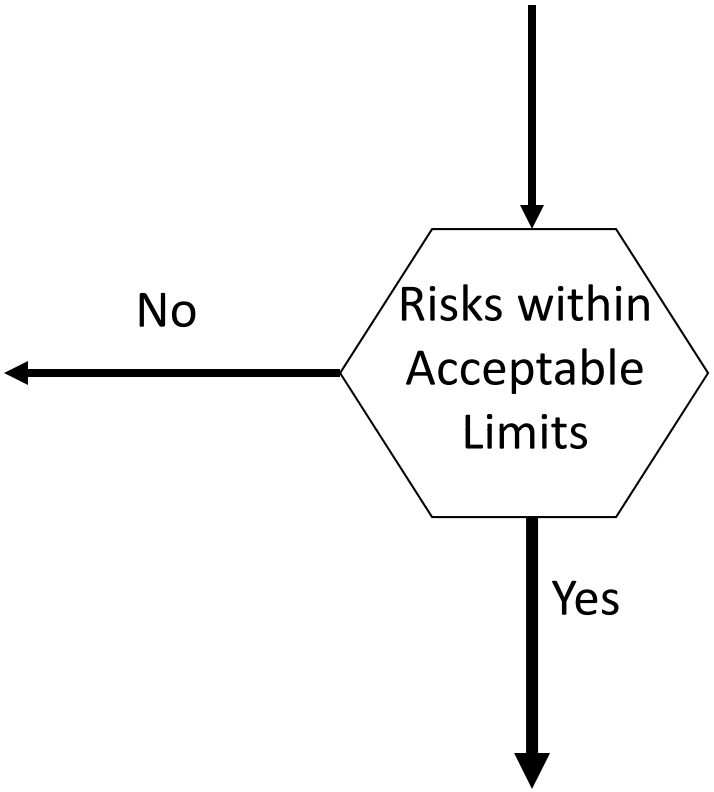
$$\text{Estimate} = \frac{\text{High} - \text{Low} + 4(\text{Prob})}{6}$$

Step 11 Risk Identification/ Mitigation

Risks			
	Sch	Cost	Tech
Risk A	L		
Risk B			H
Risk C	H		
Risk D			L
Risk E		H	

How is Technical Debt related to Schedule, Cost and Technical Risks ?

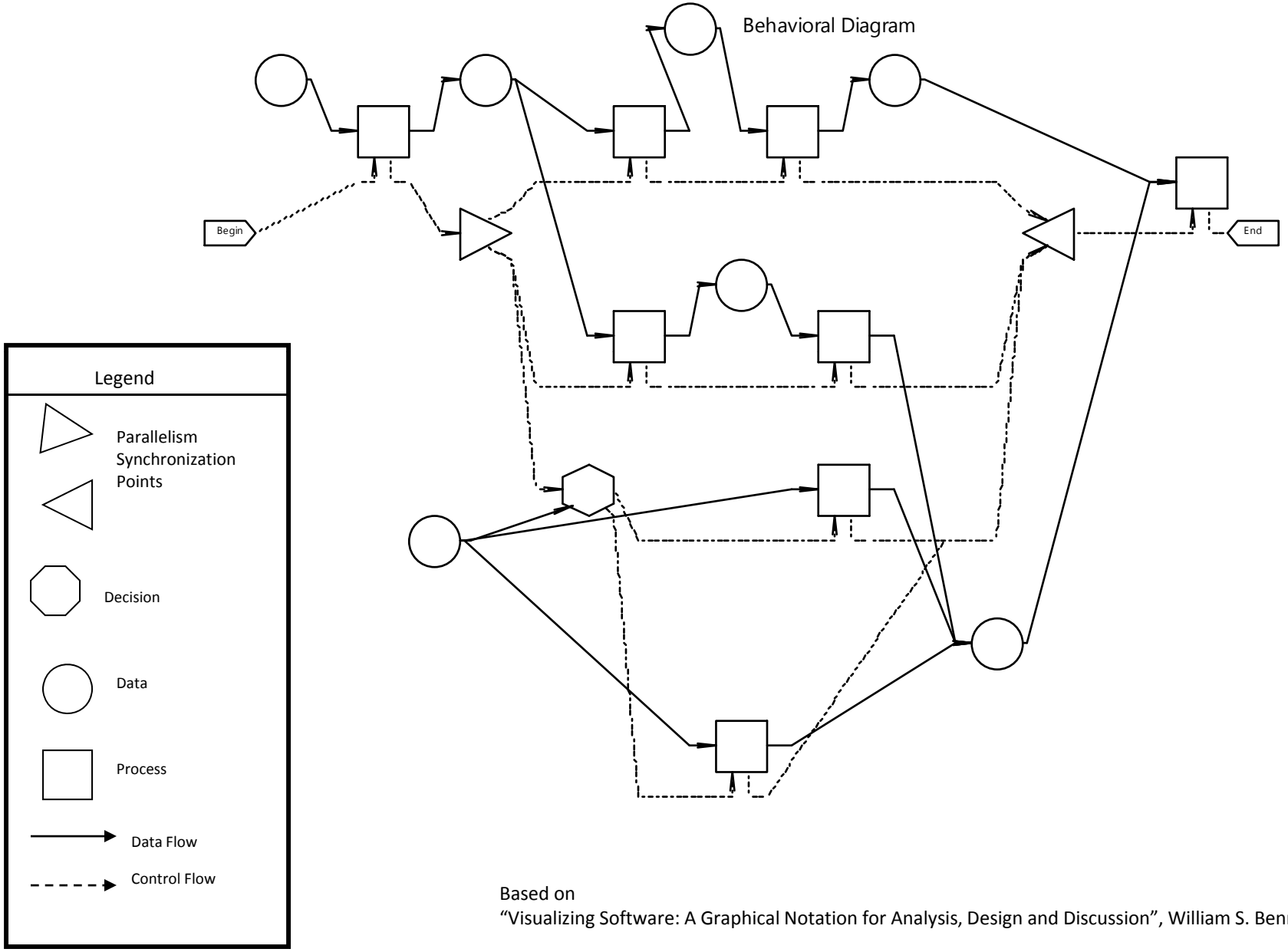
Step 12: Assessment of Risks against Acceptance Criteria



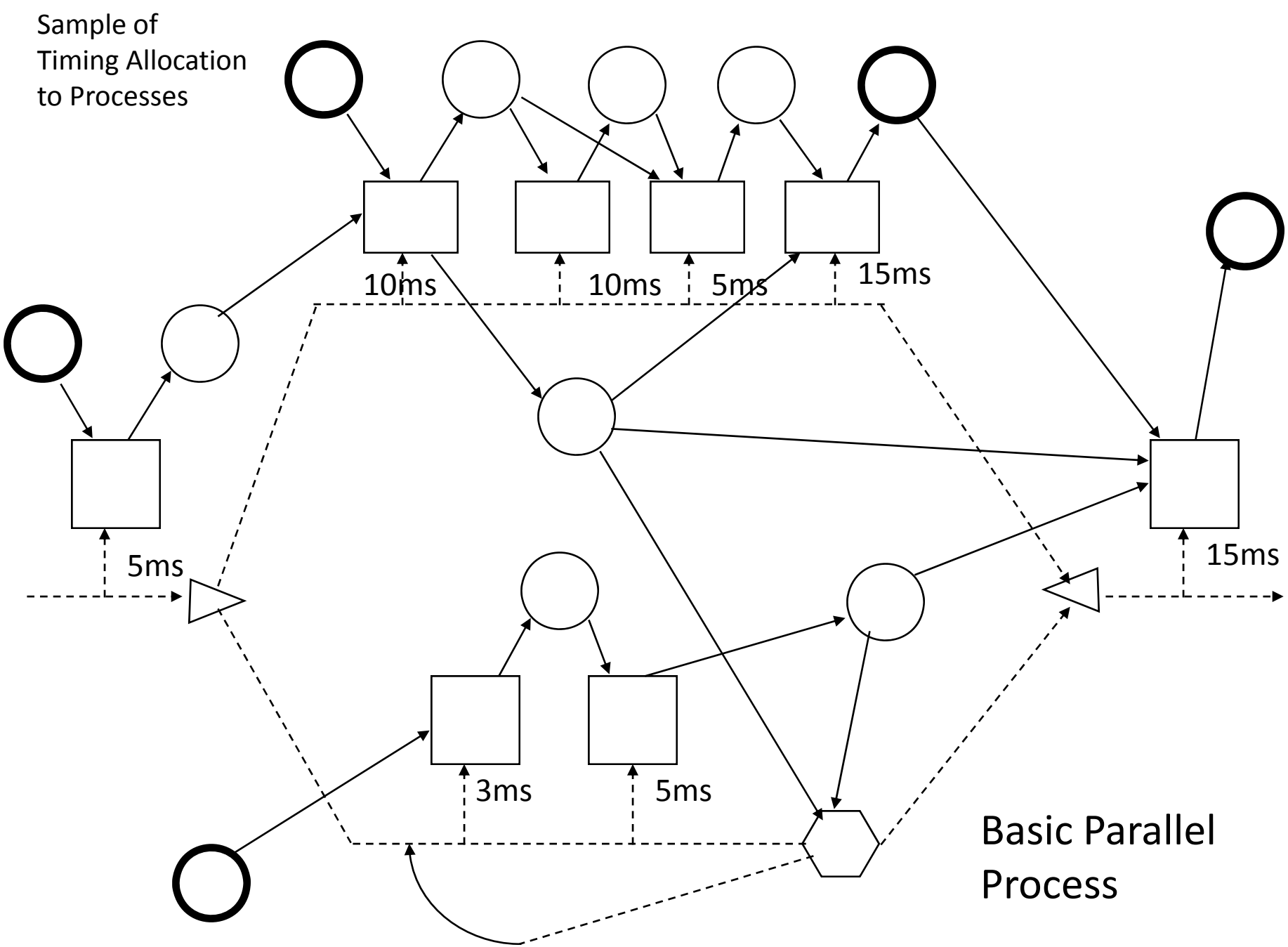
Is this where “intentional” or “unintentional” level of Technical Debt is determined?

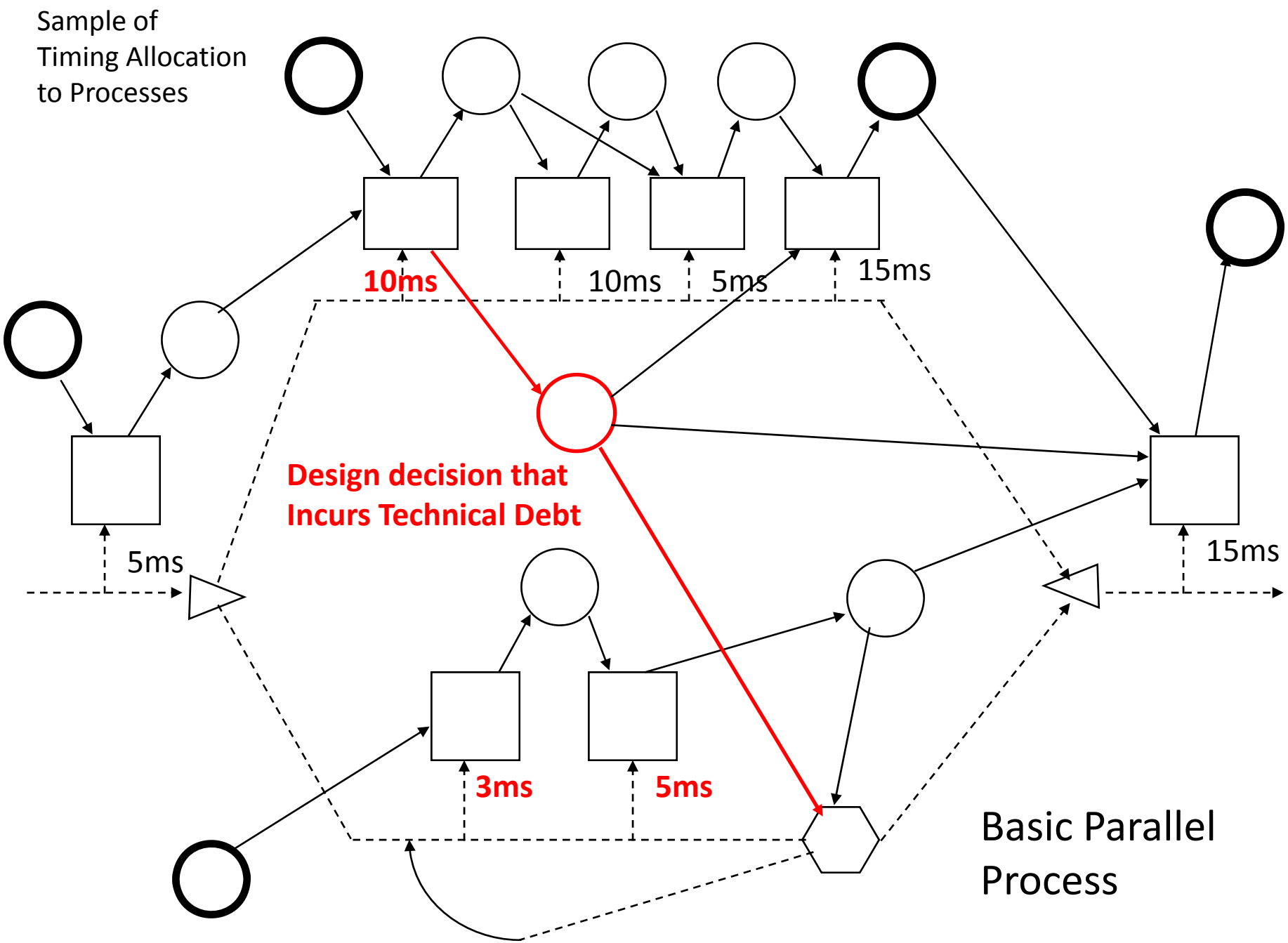
Workshop Exercise # 3
System Design & Technical Debt

Bennett Notation



Based on
"Visualizing Software: A Graphical Notation for Analysis, Design and Discussion", William S. Bennett

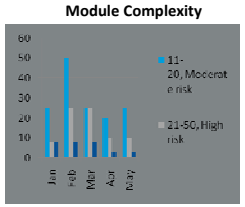

Sample of
Timing Allocation
to Processes
10ms
10ms
5ms
15ms
5ms
15ms
3ms
5ms
Basic Parallel
Process


Sample of
Timing Allocation
to Processes
10ms
10ms
5ms
15ms
Design decision that
Incurs Technical Debt
5ms
15ms
3ms
5ms
Basic Parallel
Process

How to Measure and Manage System Debt

Measure and Manage– Technical Debt

- Technical debt from static analysis tools
  - Measure trends
  - Manage by setting thresholds for action and by identifying areas for developer attention



- Test coverage debt
  - Measure test coverage throughout the lifecycle
  - Manage by adding tests where needed to meet coverage goals or reduce risk

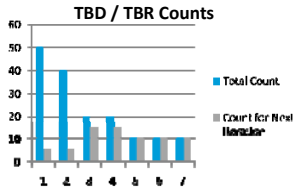
Test Coverage for Iteration N

Package Name	# Classes	Line Coverage	Branch Coverage
abcd	55	75%	64%
xyz	55	52%	43%
abqrs	3	0%	0%
vxxy	13	90%	75%
abcdefg	10	86%	88%

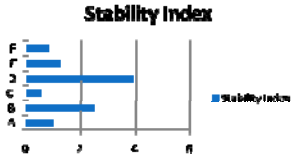
Controlling technical debt is essential for long-term success on Agile programs.

Measure and Manage– Requirements Debt

- TBDs and TBRs
  - Measure total count and count for requirements scheduled for implementation in the next couple of iterations
  - Manage to reduce open TBDs and TBRs in upcoming iterations



- Requirements Stability Index
  - Measure average complexity by component
  - Manage by using isolation patterns in areas with most instability



Respect software's limitations while leveraging its power.

- Do these measures apply for System Debt?
  - Fixes would be based on prioritization/severity
  - Technical reviews for architecture instead of Static Anl
  - Test coverage may apply
  - Requirements debt measures apply
- Other thoughts on measures
  - Confidence in architectural elements – could drive prototyping
  - Early validation of architecture
  - % of architecture integrated
  - Requirements validation debt (% requirements validated)
  - Requirements simulation debt (% requirements simulated)
  - Requirements verification debt (% requirements verified)

Technical Debt Workshop Summary

- Technical Debt
  - Management of Technical Debt- Steve McConnell
  - Technical Debt Observations- Jim Highsmith
  - Types of Debt- Chris Sterling
  - Management of Architectural Debt- Ipek Ozkapa
- Technical Debt applicability to Systems
  - Workshop Exercise # 1-Identifying Technical Debt
  - Workshop Exercise # 2- Architecture & Technical Debt
  - Workshop Exercise # 3- System Design & Technical Debt

Technical Debt References

- Managing Technical Debt
  - Steve McConnell
  - http://www.construx.com/Page.aspx?cid=2801
- The Financial Implications of Technical Debt
  - Jim Highsmith
  - http://www.jimhighsmith.com/2010/10/19/the-financial-implications-of-technical-debt/
- Second International Workshop on Managing Technical Debt
  - http://www.sei.cmu.edu/community/td2011/
- Technical Debt or Strategic Opportunity
  - https://files.me.com/philippe.kruchten/rgw078
- Enabling Agility by Strategically Managing Architectural Technical Debt
  - Ipek Ozkaya
  - http://blog.sei.cmu.edu/post.cfm/enabling-agility-by-strategically-managing-architectural-technical-debt

Technical Debt Workshop Action Plan

- White paper on Technical Debt applicability to Systems
  - Outline
  - Describe Technical Debt in Systems Engineering Vernacular
  - Identify sources and methods of measurement of Technical Debt within the Systems Engineering Life Cycle
    - System Requirements Analysis
    - System Architectural Design (all levels)
    - System Implementation
    - System Integration, Verification and Validation
    - System Transition (Deployment)
    - System Operations and Maintenance
  - Identify of implication of System Level Technical Debt to Software and Hardware Elements