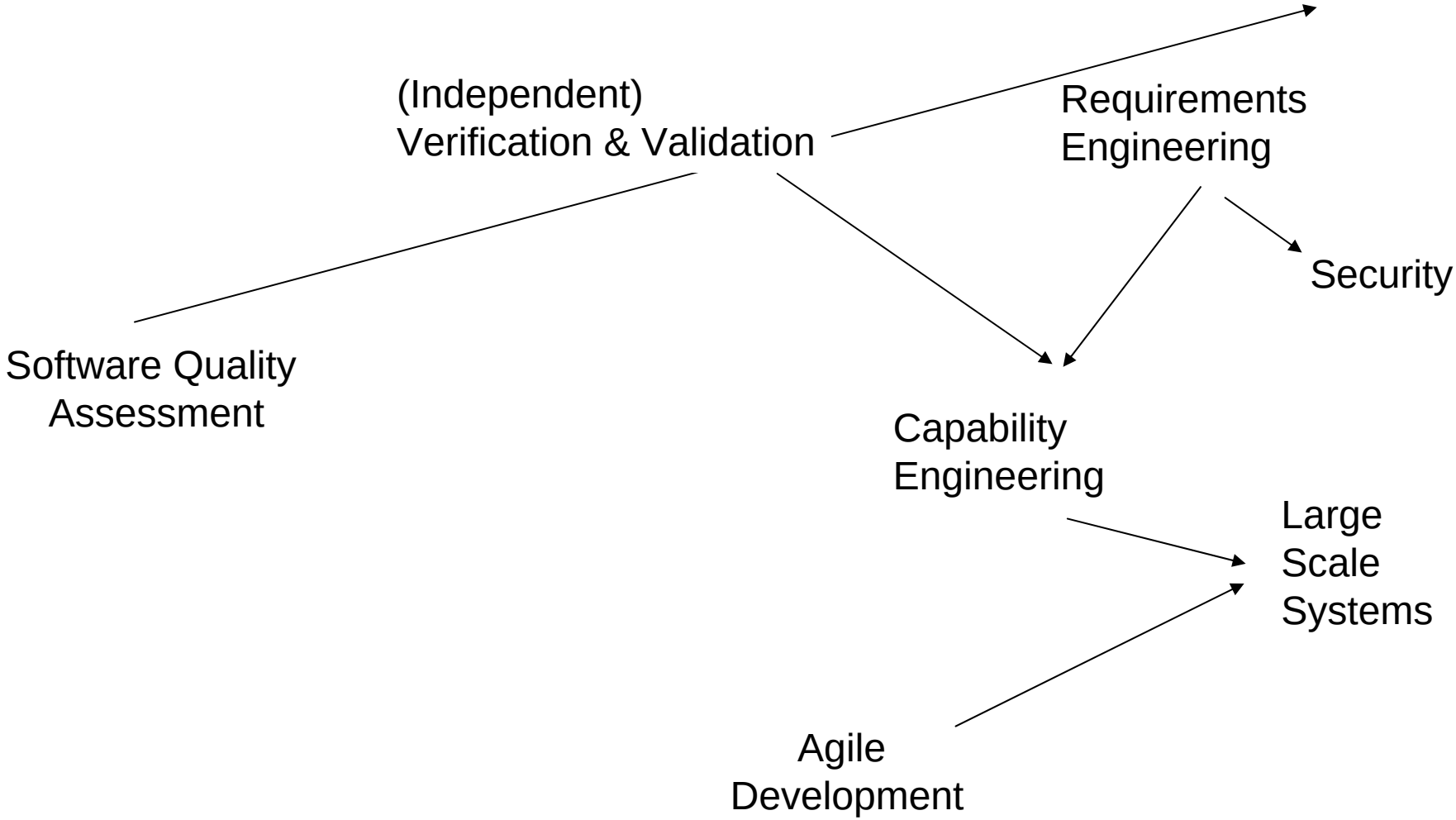
The Pervasiveness of Software Engineering: A Snapshot of Research Initiatives

James D. Arthur

Path of Progression



Software Quality Assessment

Tasking: US Navy

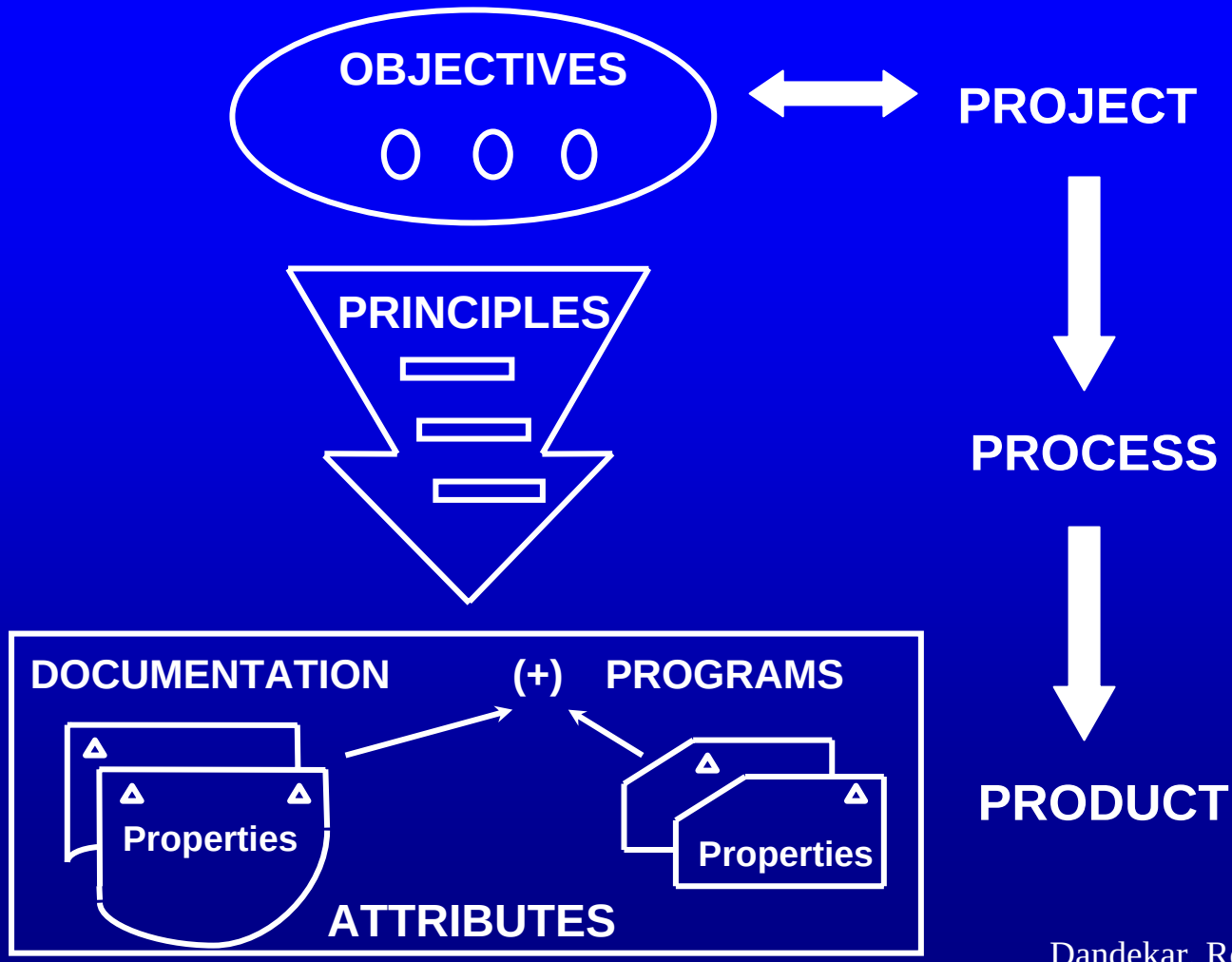
Two Distinct Software Development Efforts Using Different Methodological Approaches

Maintaining Two Functionally Equivalent Systems

Which Methodology is best?

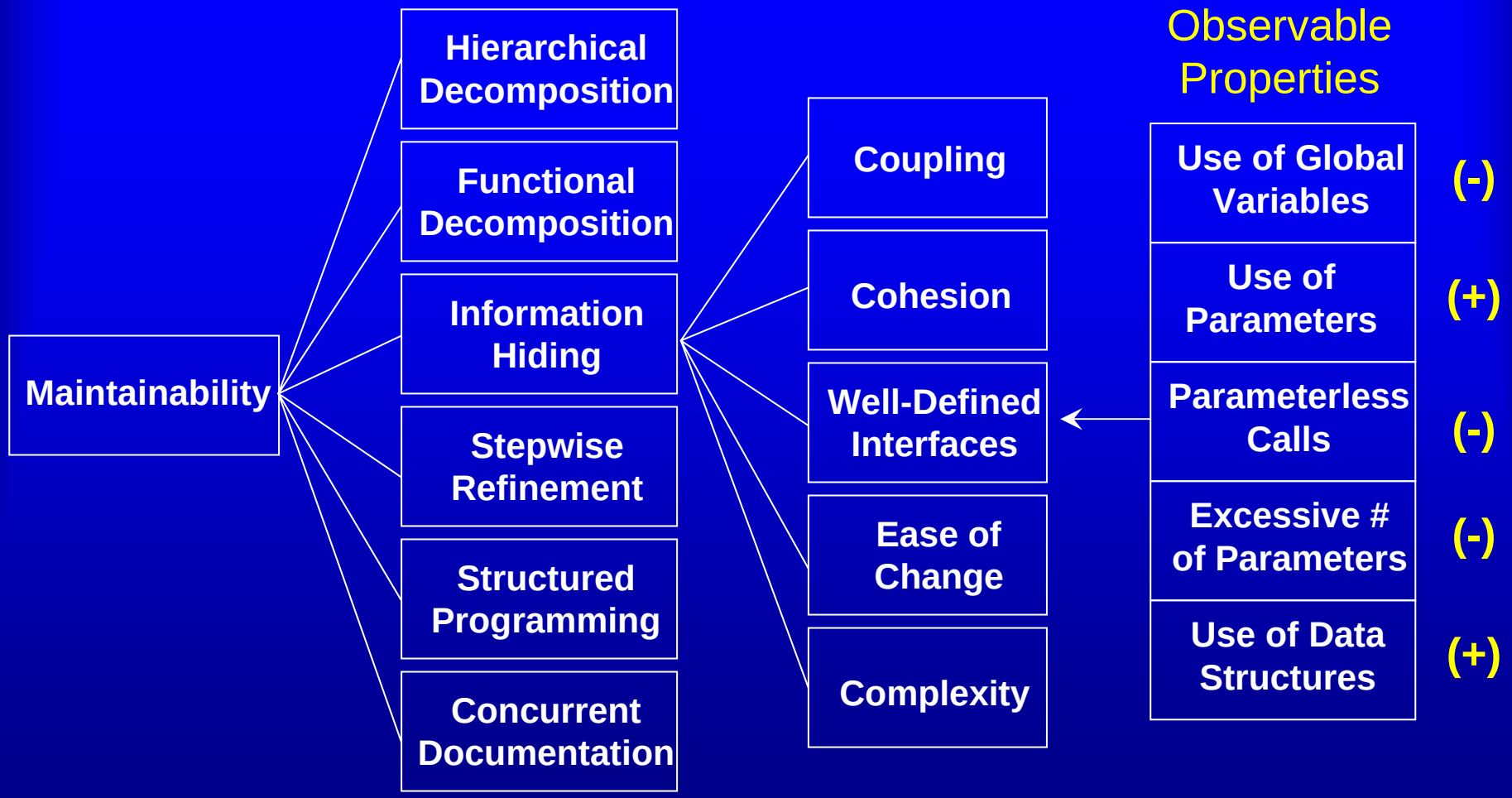
- How to Assess the *Adequacy* of a Development Methodology
- How to assess the *Effectiveness* of a Development Methodology

OPA Framework for Software Quality Assessment



Dandekar, Rosson, Bundy

Illustration of the Framework Linkages

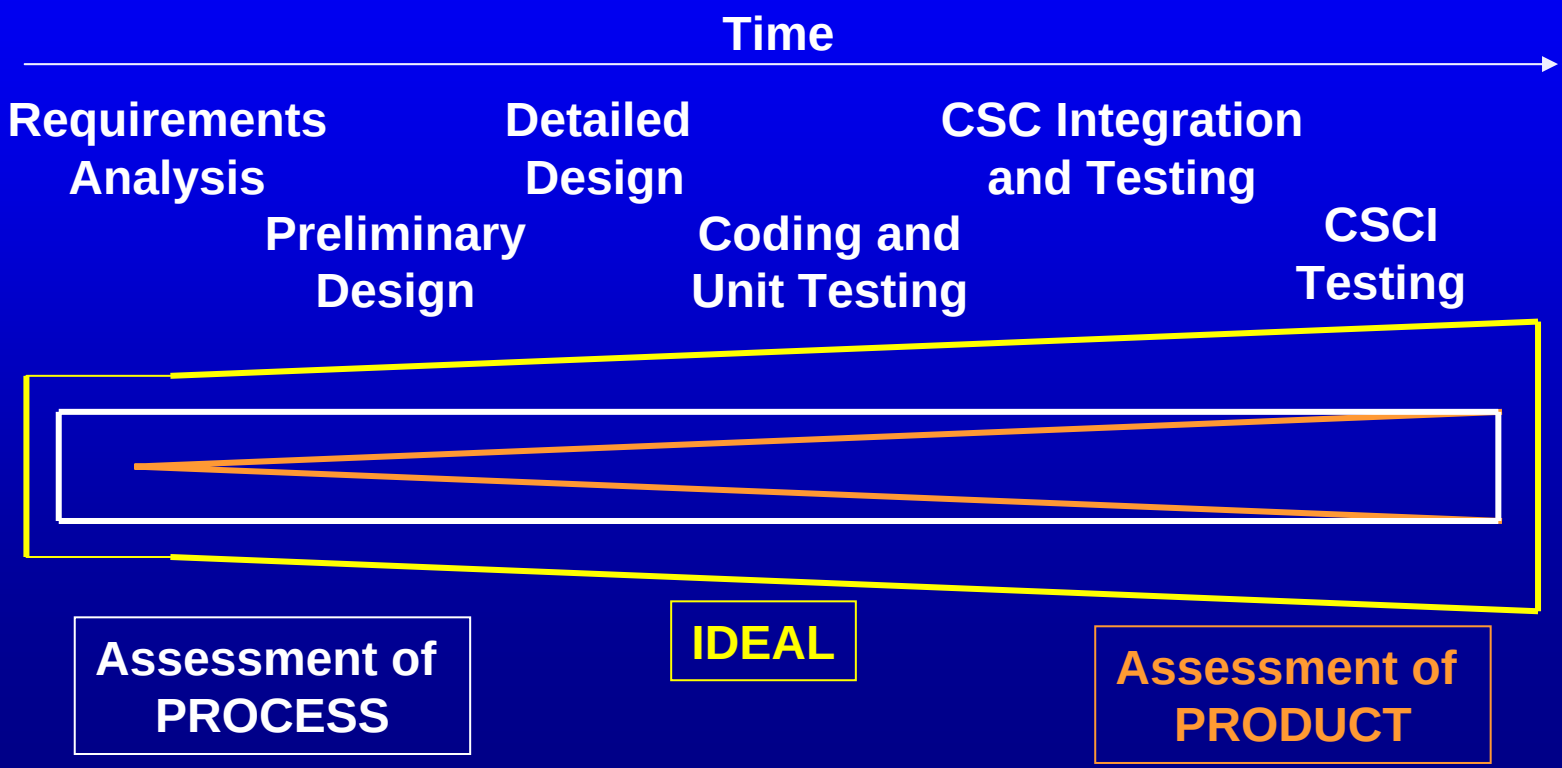


Dandekar, Rosson, Bundy

Software Measurement

Life-Cycle Basis for Software Quality Assessment

The OPA Framework exploits both *Process* and *Product Indicators*



Dandekar, Rosson, Bundy

OPA study indicated that
a *substantial* contributor to
poor product quality
was the lack of good
Verification and Validation

(Independent) Verification & Validation

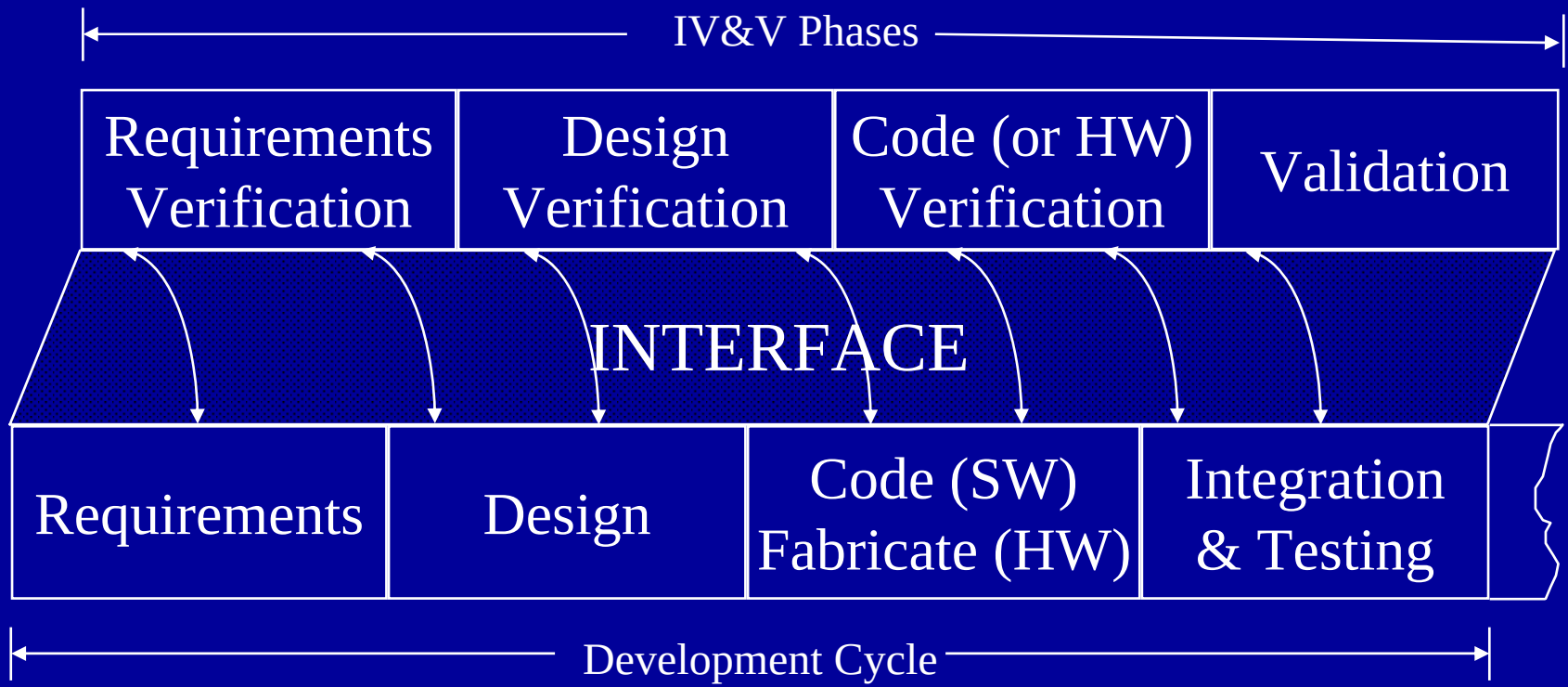
Tasking: NASA Langley

The Software Engineering Evaluation System (SEES) was a Lifecycle process that emphasized the use of V&V activities throughout the development process

To what extent can the Software Engineering Evaluation System (SEES) support an *Independent* V&V process?

Applied SEES' to Aircraft Sizing Problem

IV&V Interface to Development Cycle



- IV&V should complement the SD process
- IV&V should always be an adaptive , overlay process

Aircraft Sizing Study

Group 1

IV&V	Development
Requirements Analysis	HL Design
HLD Analysis	LL Design
LLD Analysis	STD
Code Analysis	Coding
Validation	Unit Test
	I&T
	Acceptance Test

Group 2

Development
HL Design
LL Design
STD
Coding
Unit Test
I&T
Acceptance Test

Data Analyzed*

Group 1
Total DTR/FRs: 223

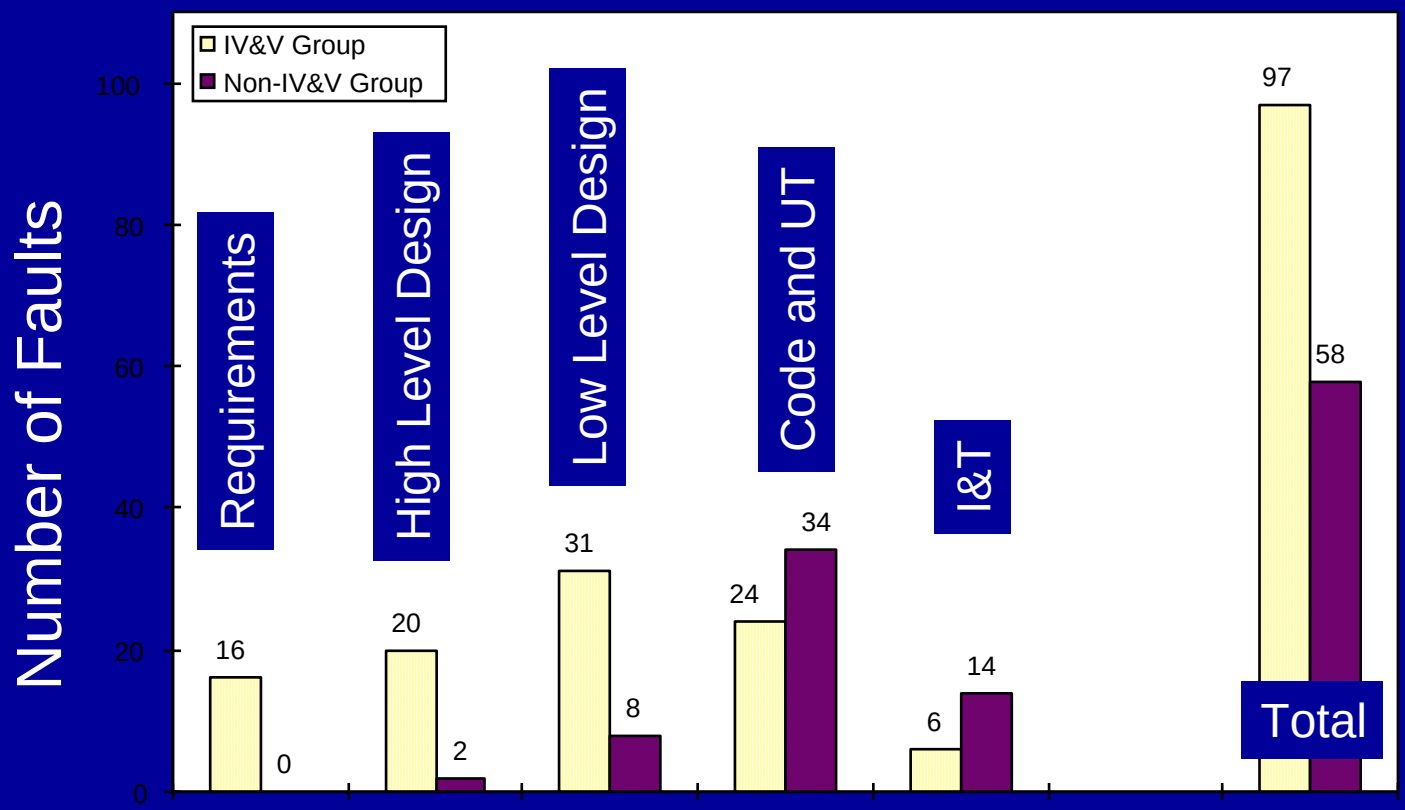
Critical: 97
Non-Critical: 126

Group 2
Total FRs: 62

Critical: 58
Non-Critical: 4

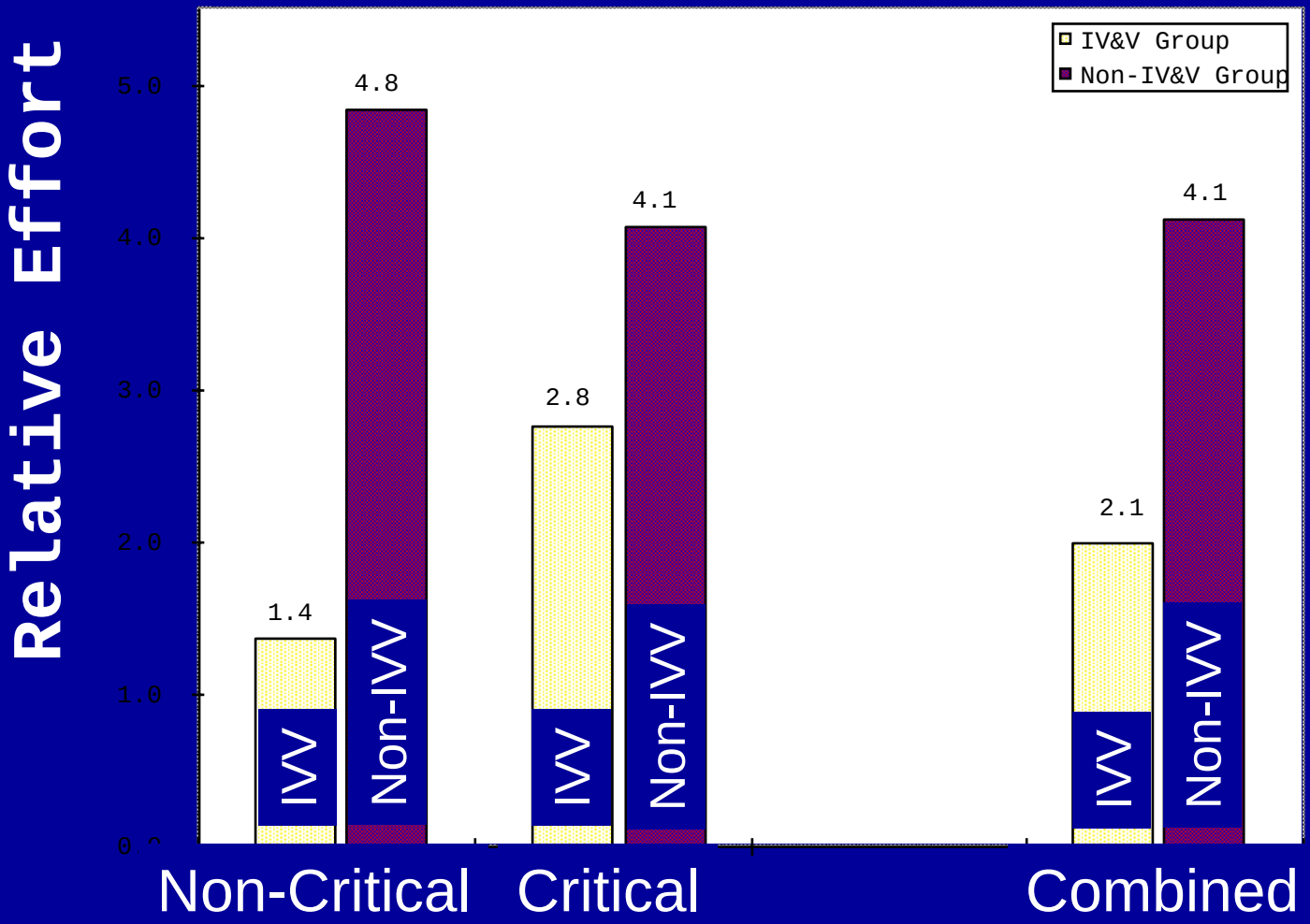
*** Includes only data with recorded effort >= 1 minute**

Comparison Between IV&V and Non-IV&V Groups



Phase in which Critical Faults Found

Mean Effort to Remove Faults
Boehm's Measure



IV&V study indicated that the more critical errors were due to poor Requirement Specs

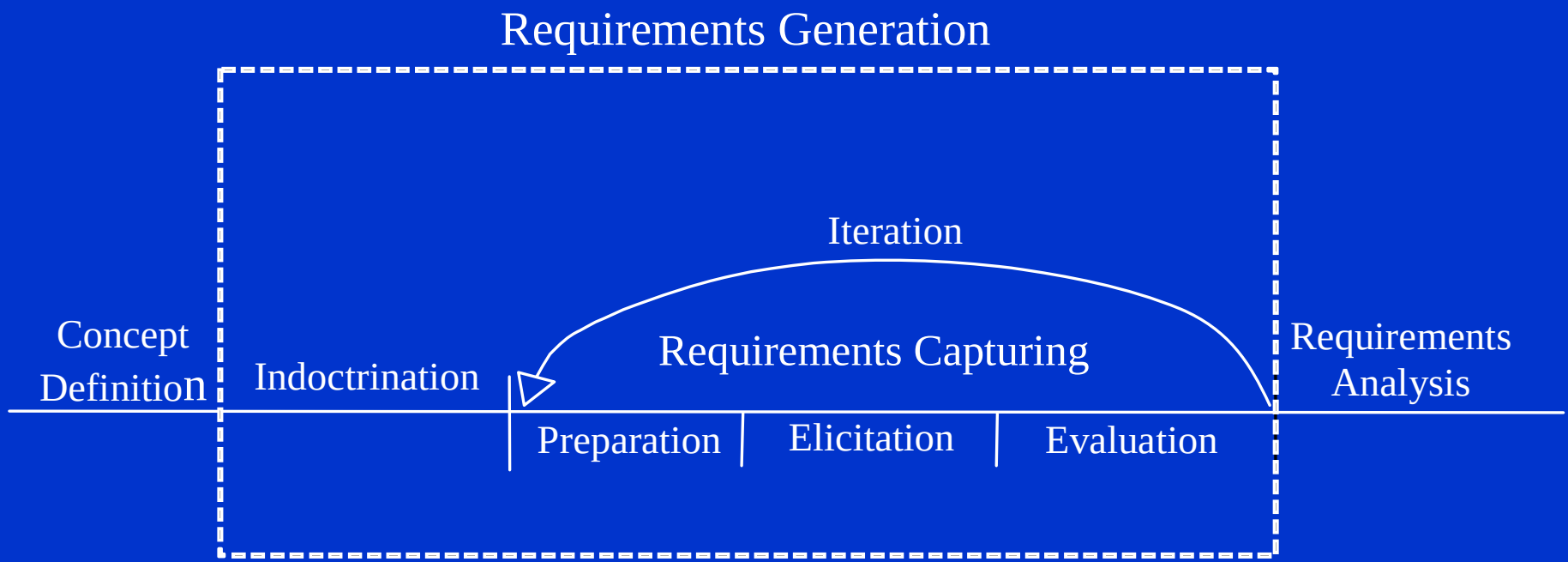
Problem Statement

How does one evolve requirements that meet the customers' Needs and Intent?

What was needed:

An RE framework that guided rather than dictated the use of RE methods

A Framework for Requirements Generation



Bring into focus the distinct components and their role within requirements generation

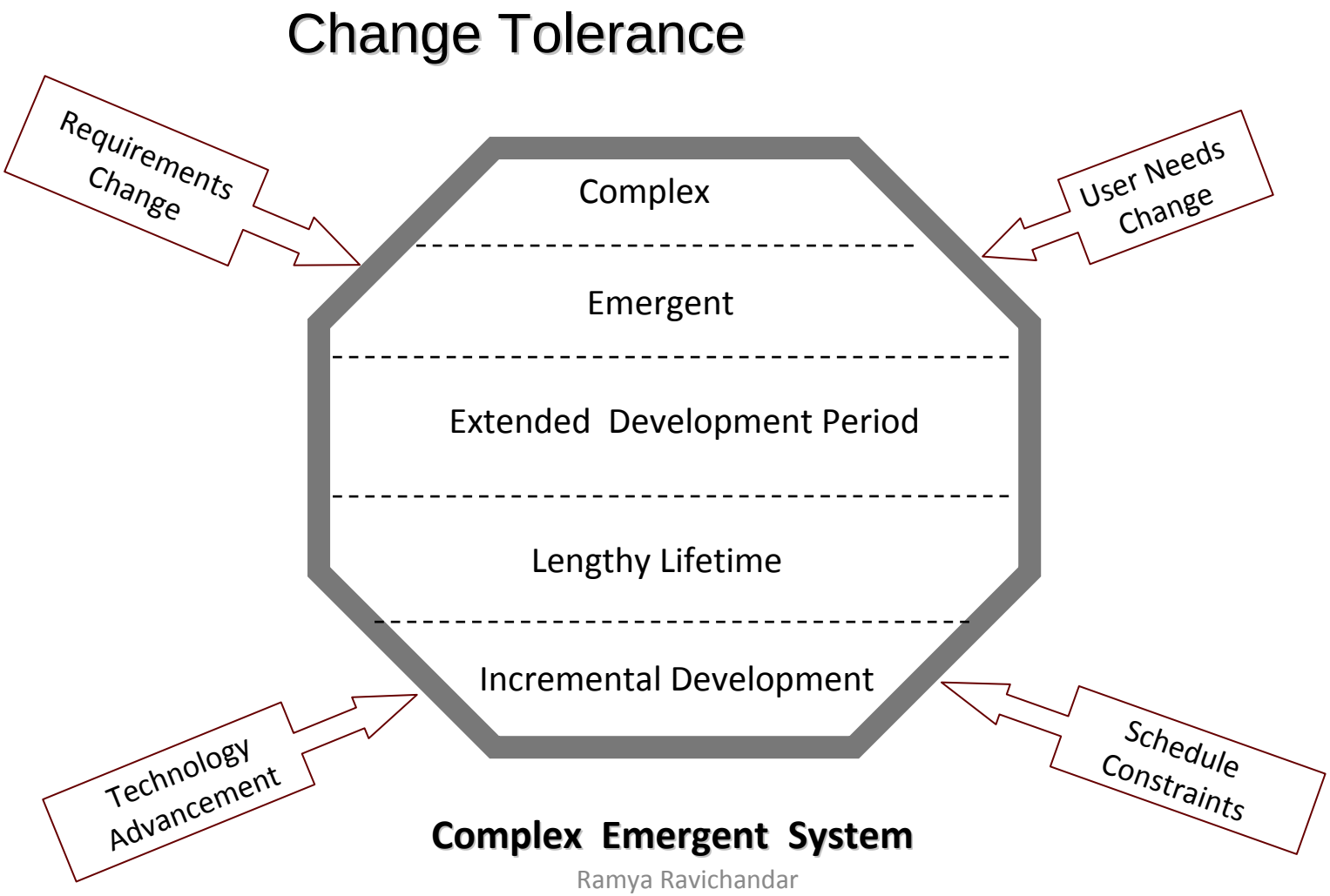
Process Components

- **Guidelines** – suggestions or recommendations that offer support by serving in an *advisory* capacity
- **Protocols** – are rules that
  - establish boundaries through pre-defined constraints, and
  - impose operational, goal oriented actions through mandates
- **Monitoring Methodology** – a methodology where procedures are
  a) continuously applied to monitor activities within the requirements elicitation process to detect irregularities and
  b) indicate methods to correct those irregularities

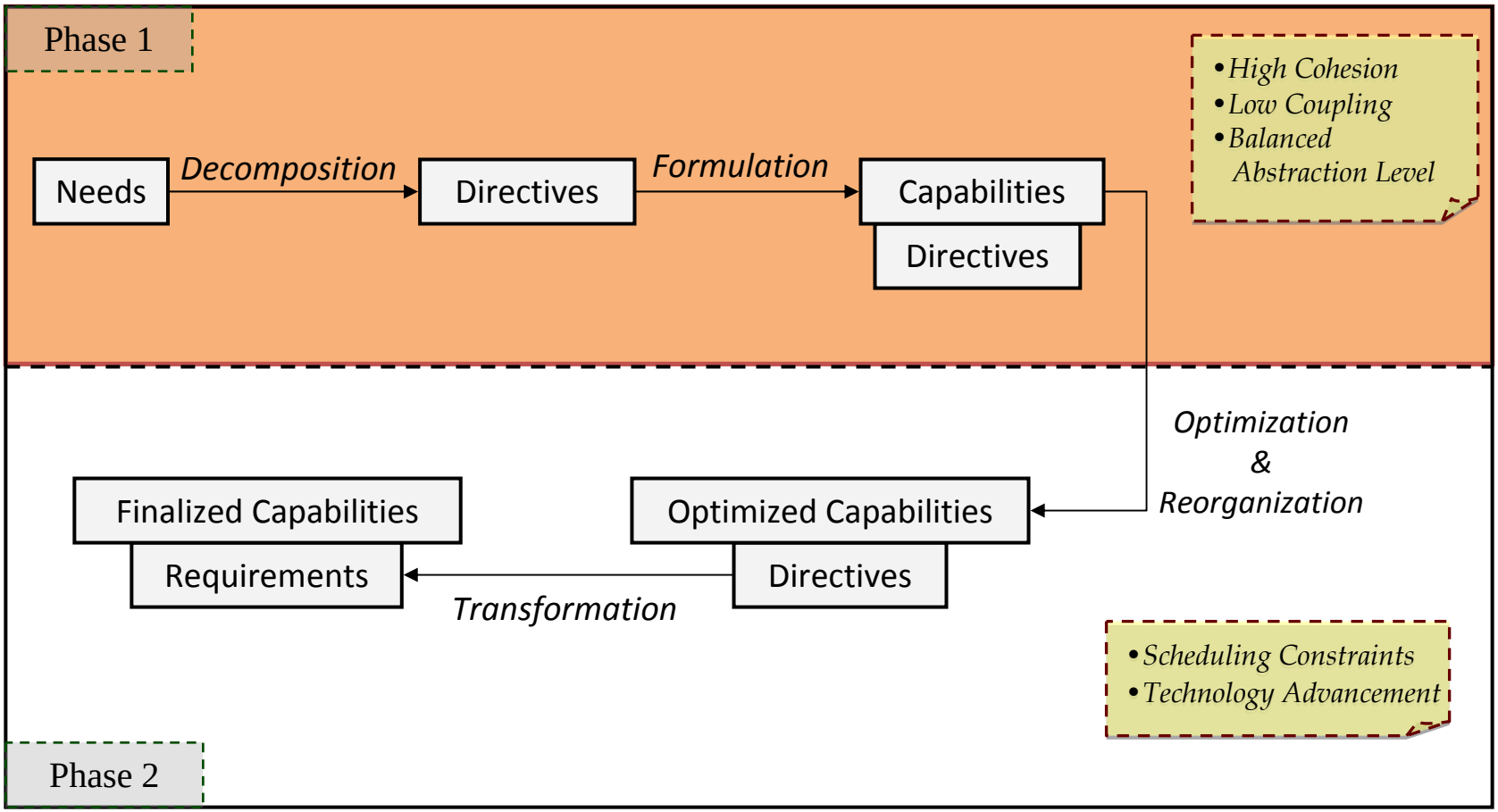
How might we apply our understanding Software Engineering to the development of Large Scale Systems?

Producing
Change Tolerant Systems

Problem


Change Tolerance
Requirements Change
User Needs Change
Complex
Emergent
Extended Development Period
Lengthy Lifetime
Incremental Development
Technology Advancement
Schedule Constraints
Complex Emergent System

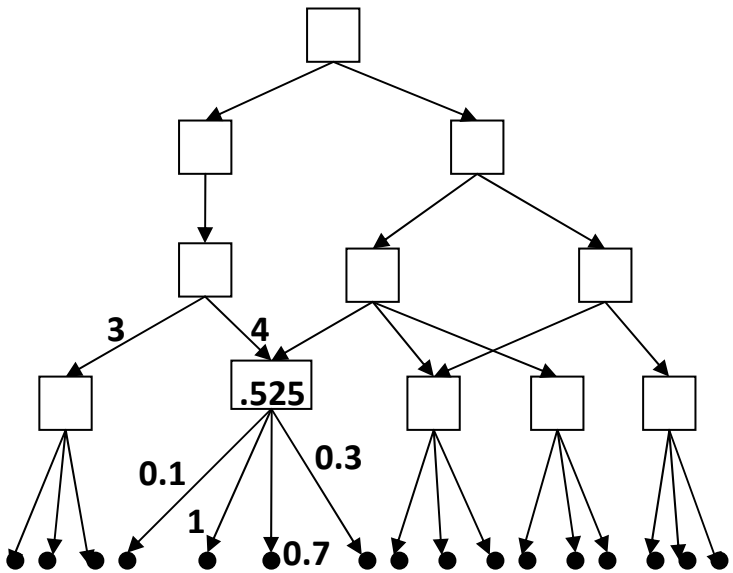
Capabilities Engineering Process



Cohesion

Relevance Values

Category	Scale	Failure to implement directive
Catastrophic	1.0	Task Failure
Critical	0.7	Task success questionable
Marginal	0.3	Reduction in technical performance
Negligible	0.1	Non-operational impact



- Functional Cohesion
  - **Relevance Values :** Immediate Parent
  - **Size:** Higher level nodes

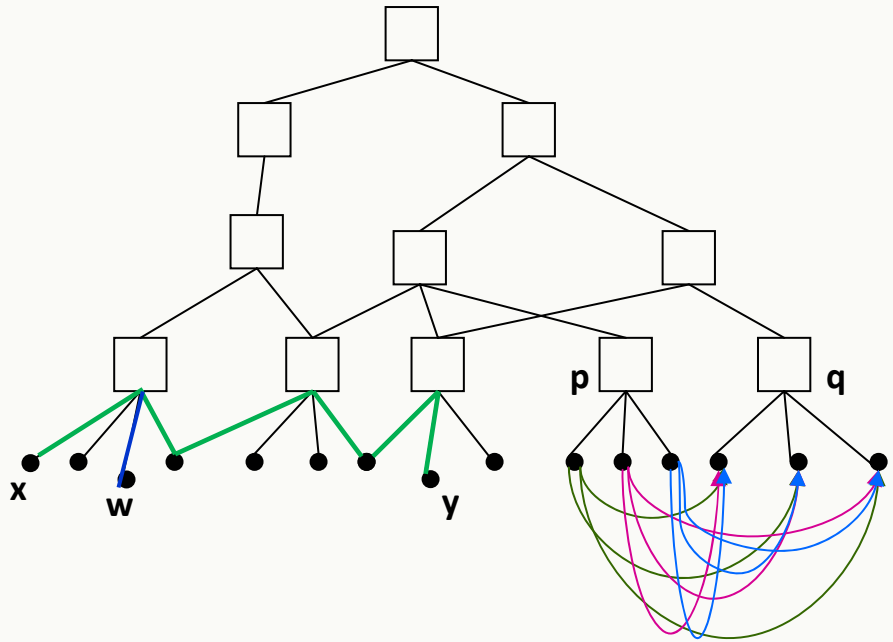
$$Cohesion(n) = \begin{cases} \dfrac{\sum_{i=1}^{t}(wt(n,v_i).Cohesion(v_i))}{\sum_{i=1}^{t} wt(n,v_i)} & (n,v_i) \in E_n, outdegree(v_i) > 0 \\ \dfrac{\sum_{i=1}^{t} wt(n,v_i)}{size(n)} & (n,v_i) \in E_n, outdegree(v_i) = 0 \end{cases}$$

$$wt(n,v) = \begin{cases} size(v) & outdegree(v) > 0 \\ Rel(v,n) & outdegree(v) = 0 \end{cases}$$

$$size(n) = \begin{cases} \sum_{i=1}^{t} size(v_i) & (n,v_i) \in E_n; outdegree(v_i) > 0; \\ 1 & outdegree(n) = 0 \end{cases}$$

Coupling

- Among Capabilities
  - Between constituent directives
- Distance
  - *Coupling* α $\frac{1}{distance}$
  - `dist( x, y)` = 6
  - `dist( x, w)` = 2
- Probability Of Change
  - $P(y) = \frac{1}{\#\ directives}$
  - $P(y) = 1/3$

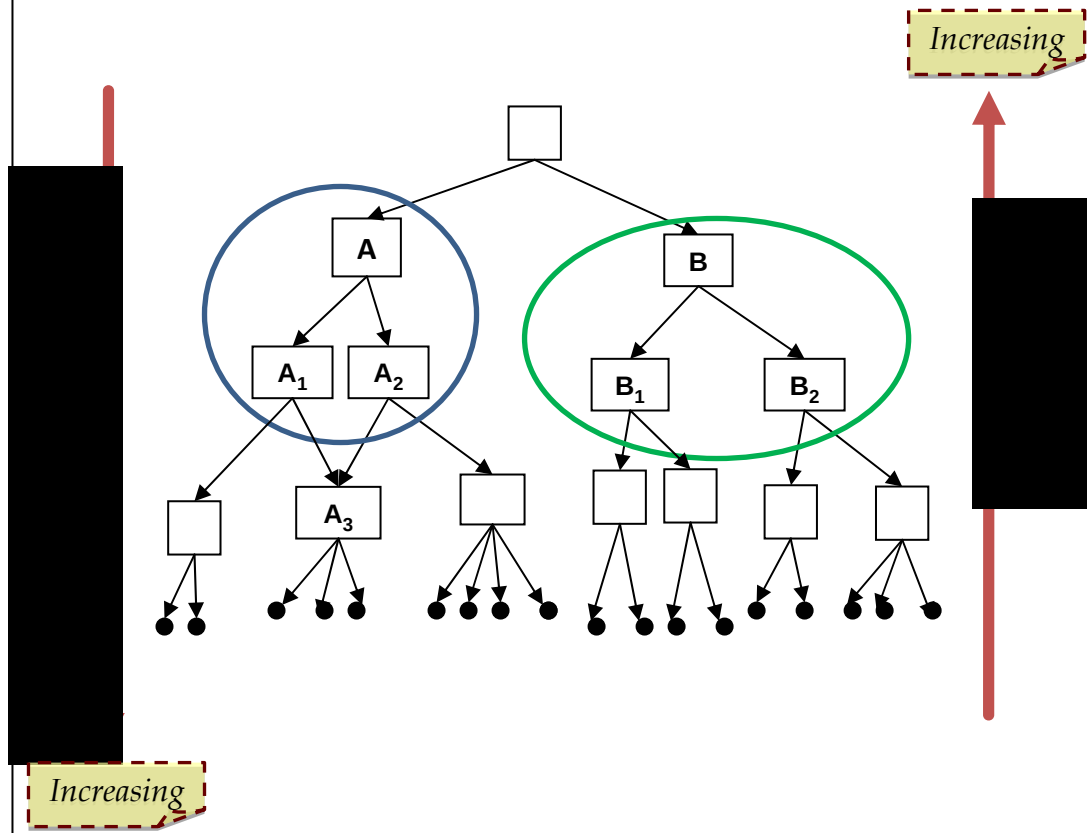


$$Cp(p,q) = \frac{\sum_{d_i \in D_p} \sum_{d_j \in D_q} Cp(d_i, d_j)}{|D_p|.|D_q|}$$

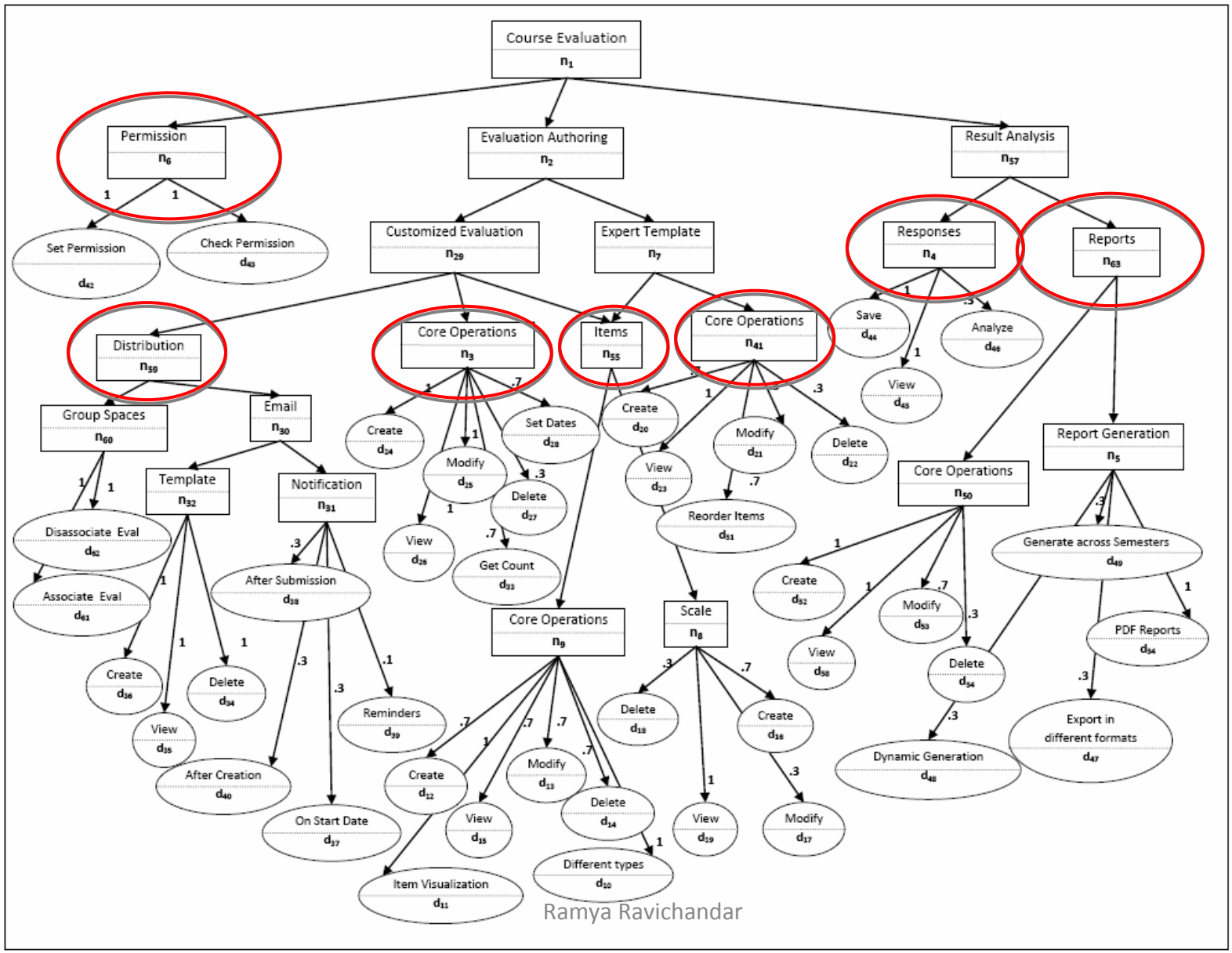
$$\text{where } Cp(d_i, d_j) = \frac{P(d_j)}{dist(d_i, d_j)} \text{ and } P(d_j) = \frac{1}{|D_q|}$$

Balanced Abstraction Level

- Right Level of Abstraction ?
- Trade-off Analysis
  - Increased Coupling
  - Decreased Size
- Scenarios
  - Common Functionality
  - No Common Functionality



Function Decomposition Graph



Ramya Ravichandar

Validation: SAKAI System

Change-Tolerance

H_0: The change-tolerance of a system is independent of an RE or a CE-based design

H_1: The change-tolerance of a system improves with the use of a CE-based design

Wilcoxon Signed Rank test: P-value 0.018

Change Reduction

H_0: The number of change-requests generated during system development is independent of an RE or a CE process

H_1: The number of change-requests generated during system development is reduced with the use of a CE process

Wilcoxon Signed Rank test: P-value 0.0002

How might we apply our understanding Software Engineering to the development of Smaller Scale Systems?

Introducing Agile Practices into an Organizations Development Process

Motivation & Problem Statement

- Growth of Agile Adoption
- People asking *how* to adopt agile practices

Problem Statement

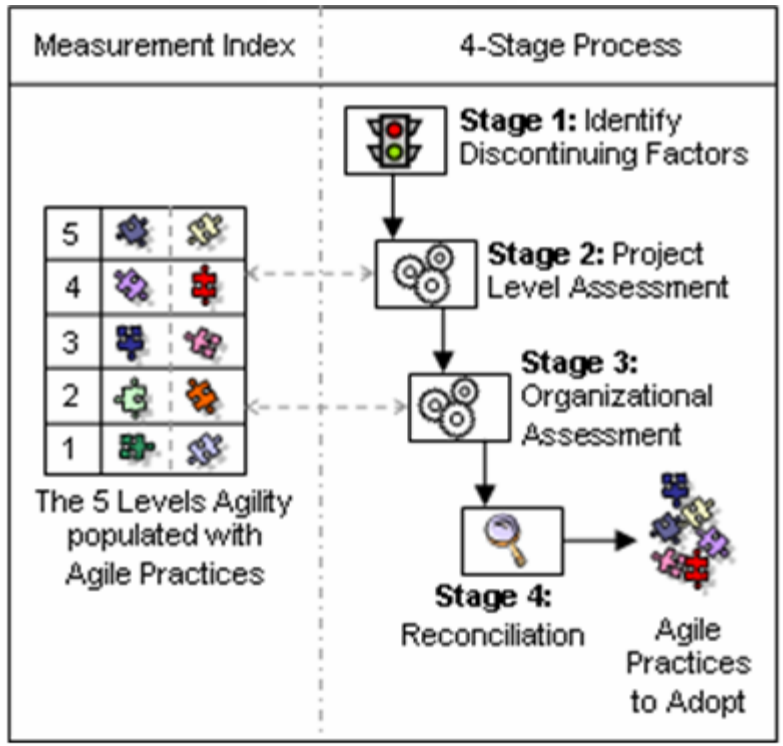
Absence of a structured approach to guide agile adoption efforts

The Solution Approach

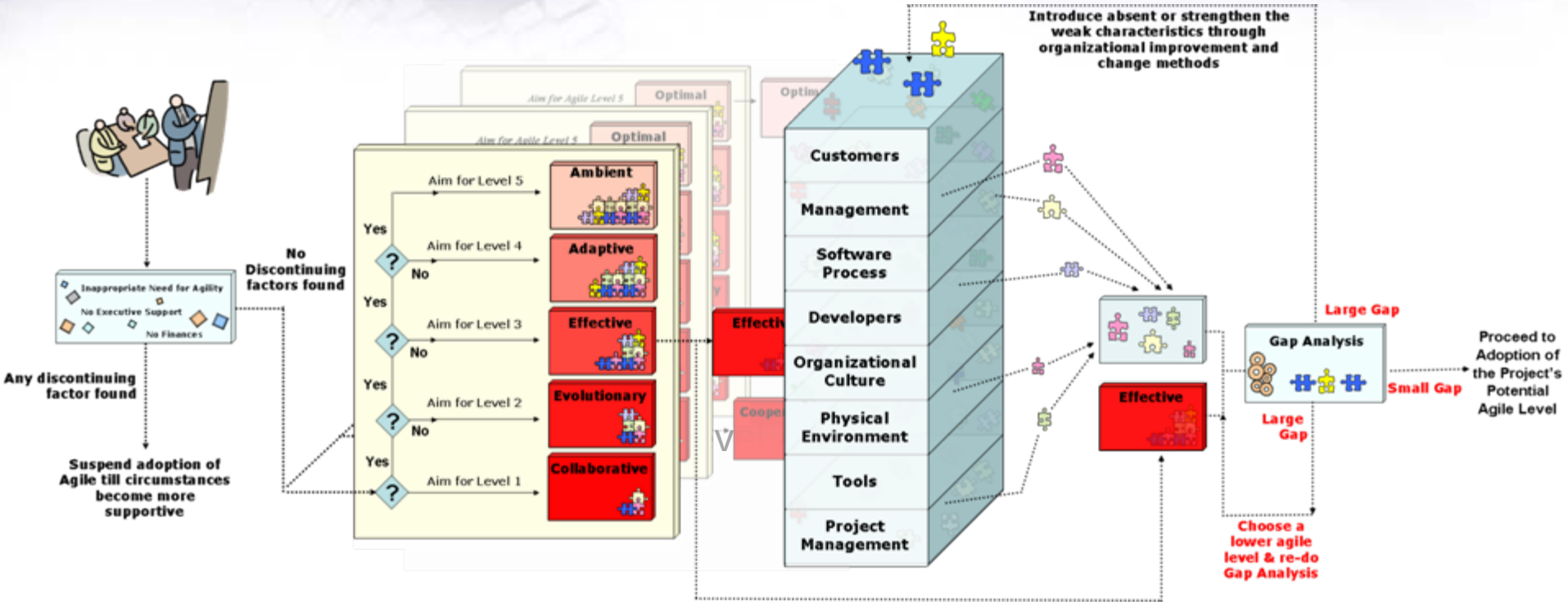
The Agile Adoption Framework

To guide and assist organizations in adopting agile practices in their projects

- **4-Stage Process**
- **Sidky Agile Measurement Index**



Overview of the 4-Stage Process



Stage 1
Discontinuing Factors

Stage 2
Project Level Assessment

Stage 3
Organizational Assessment

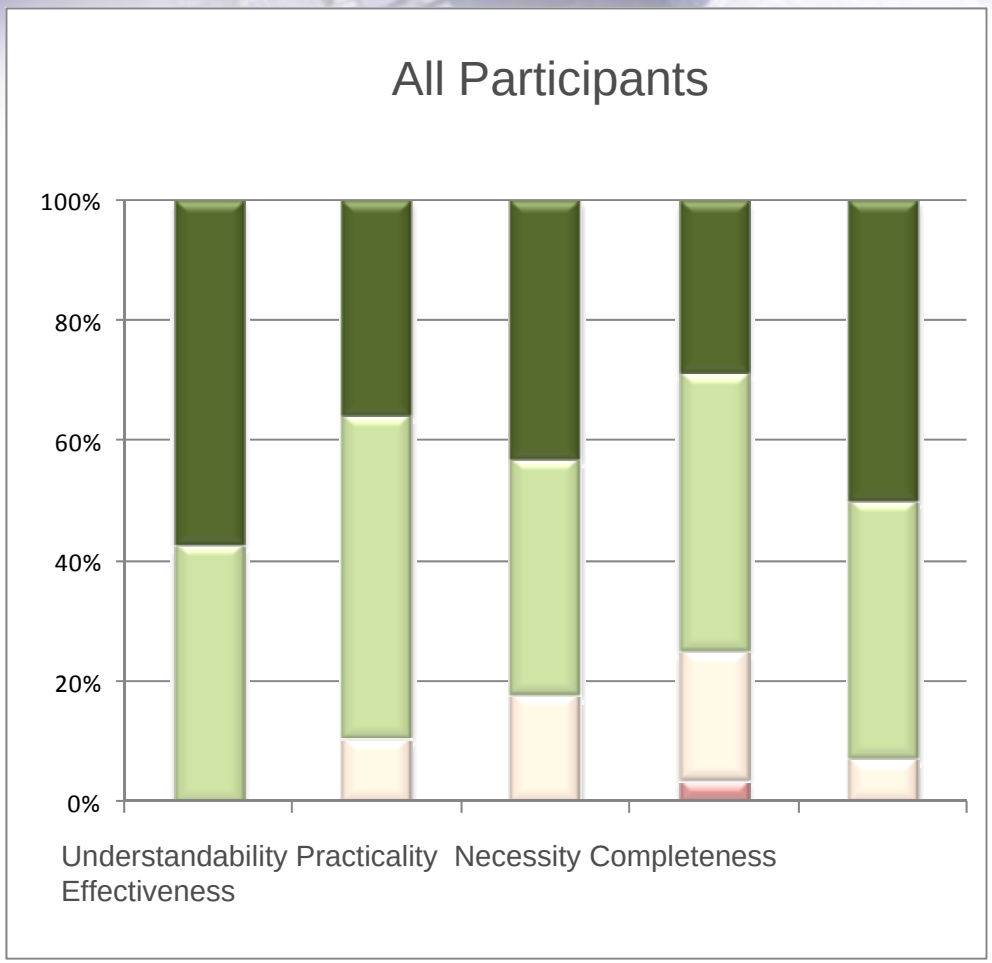
Stage 4
Reconciliation

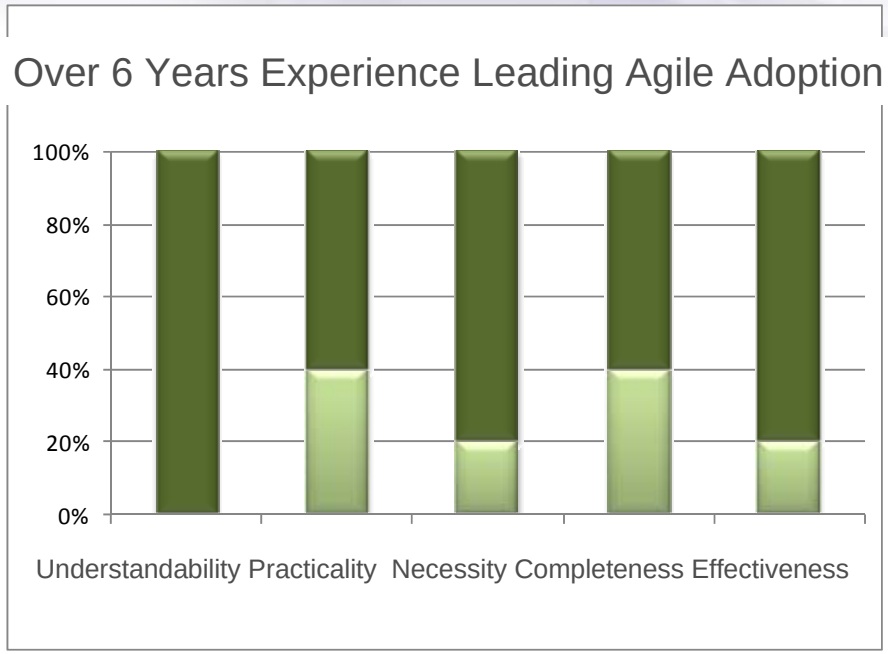
A Sidky

	Embrace Change to Deliver Customer Value	Plan and Deliver Software Frequently	Human Centric	Technical Excellence	Customer Collaboration
Level 5 **Encompassing**	Low Process Ceremony	Agile Project Estimation	Ideal Agile Physical Setup	Test Driven Development Paired Programming No/minimal number of Cockburn Level -1 or 1b people on team	Frequent Face-to-face interaction between developers & Users (Collocated)
Level 4 **Adaptive**	Client Driven Iterations Customer Satisfaction Feedback	Smaller and More Frequent Releases (*4-8 Weeks*) Adaptive Planning		Daily Progress Tracking Meetings Agile Documentation (from Agile Modeling) User Stories	Collaborative, Representative, Authorized, Committed and Knowledgeable (CRACK) Customer Immediately Accessible Customer contract revolves around commitment of collaboration, not features
Level 3: **Effective**		Risk Driven Iterations Maintain a list of all remaining features (Backlog)	Self Organizing Teams Frequent face-to-face communication between the team	Continuous Integration Continuous Improvement (i.e. Refactoring) Have around 30% of Cockburn Level 2 and Level 3 people on team Unit Tests	
Level 2: **Evolutionary**	Evolutionary Requirements	Continuous Delivery (Incremental-Iterative development) Planning at different levels		Software Configuration Management Tracking Iteration through Working Software No Big Design Up Front (BDUF)	Customer Contract reflective of Evolutionary Development
Level 1: **Collaborative**	Reflect and tune Process	Collaborative Planning	Collaborative teams Empowered and Motivated Teams	Coding Standards Knowledge Sharing Tools *(Wikis, Blogs)* Task Volunteering not Task Assignment	Customer Commitment to work with Developing Team

A Sidky

4-Stage Process

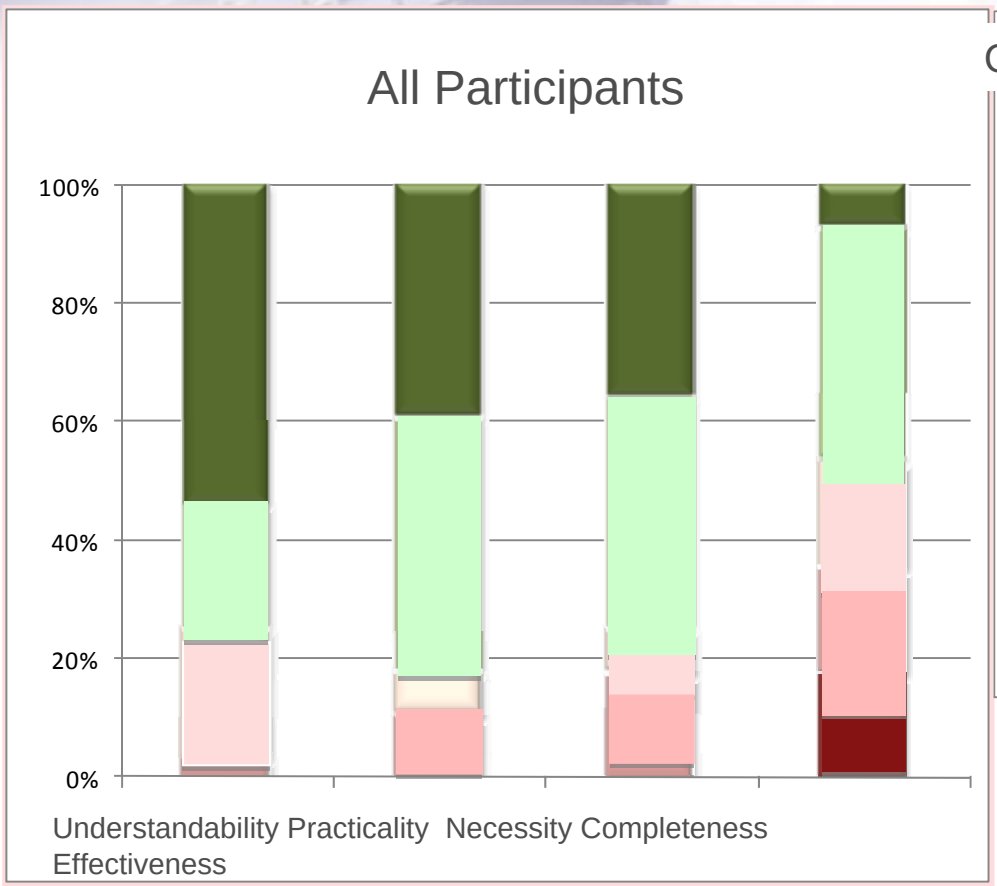


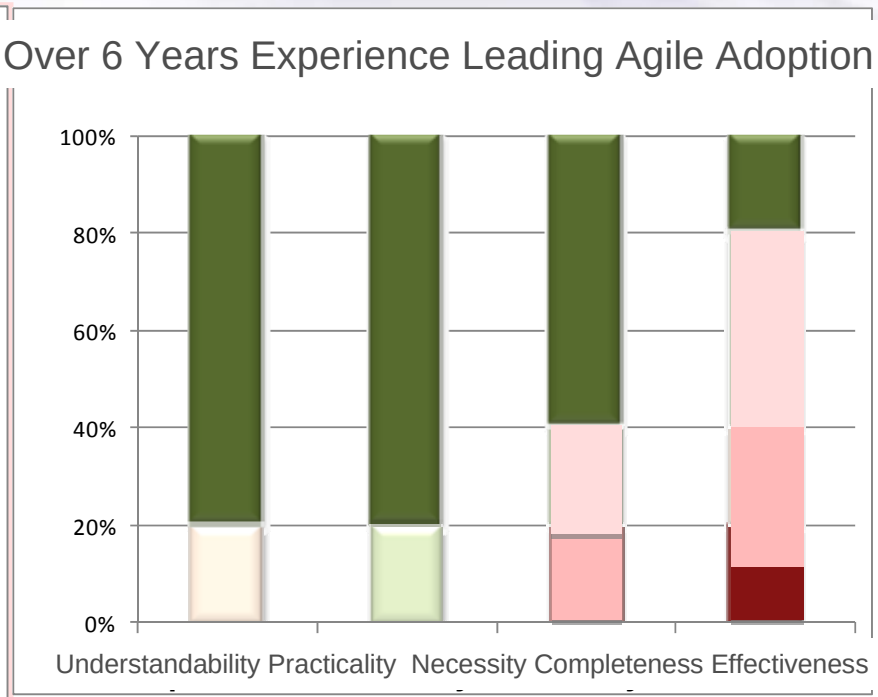


- Strongly Agree
- Slightly Agree
- Neither Agree nor Disagree
- Slightly Disagree
- Strongly Disagree

A Sidky

Sidky Agile Measurement Index (SAMI)





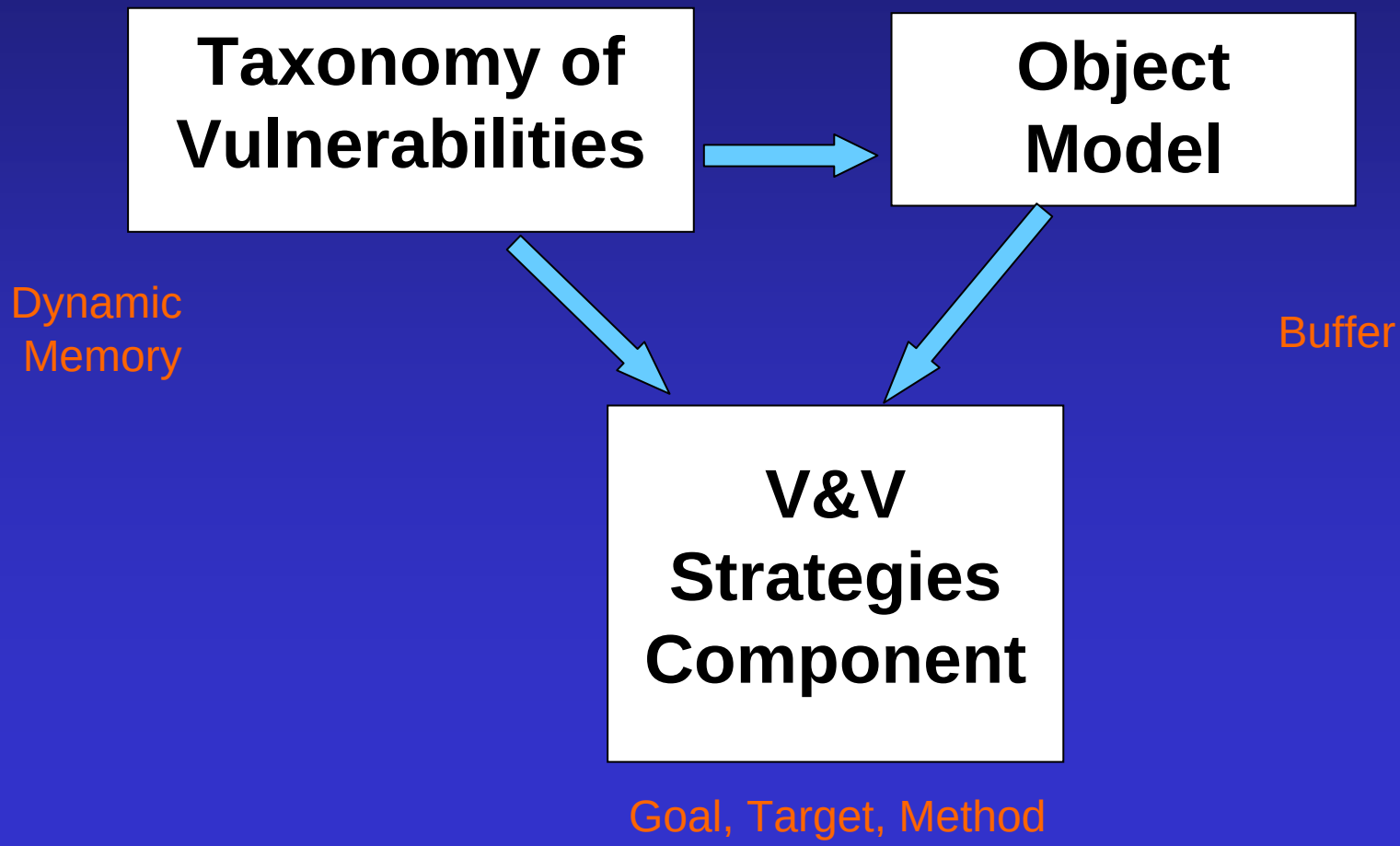
- Strongly Agree
- Slightly Agree
- Neither Agree nor Disagree
- Slightly Disagree
- Strongly Disagree

A Sidky

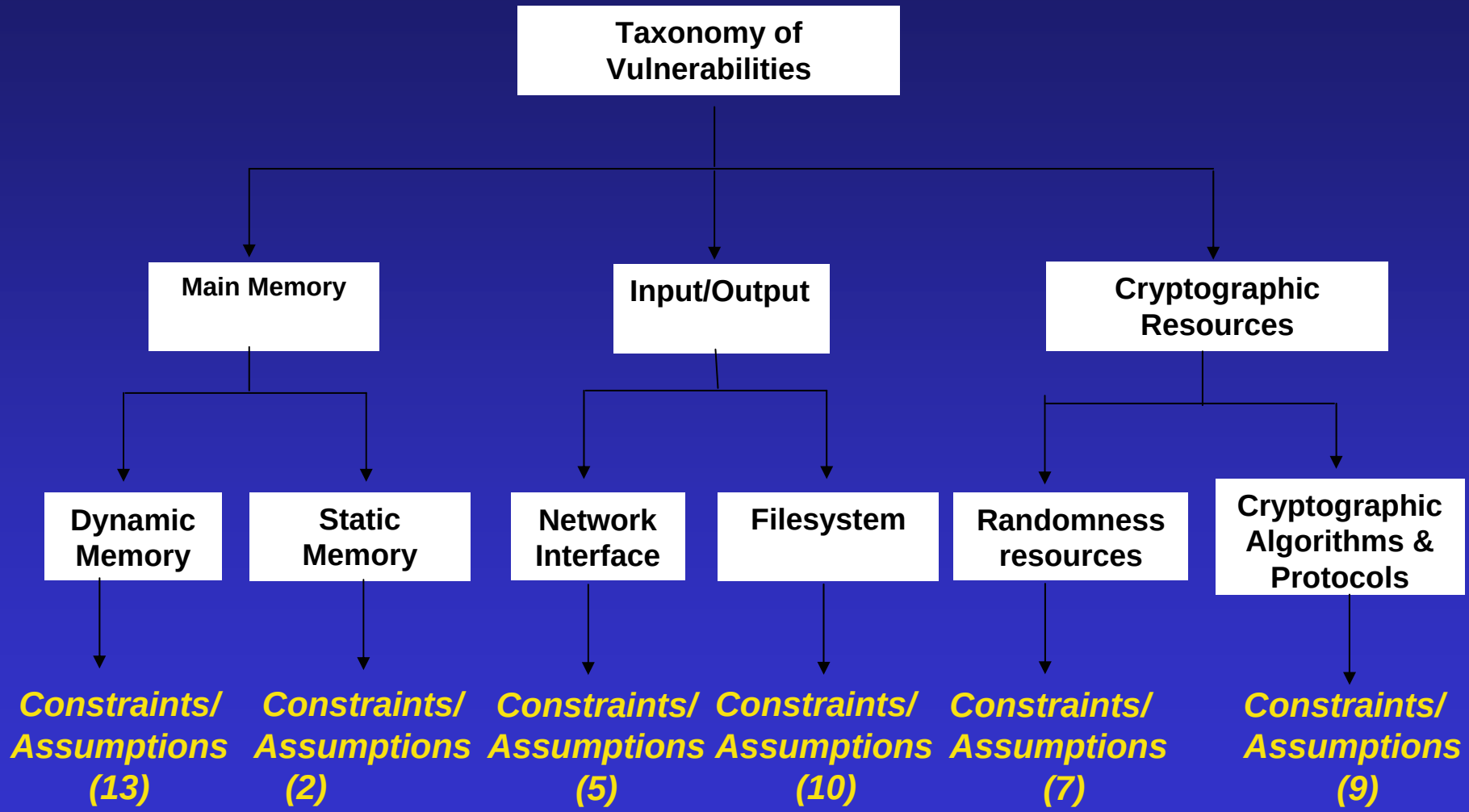
How might we apply our understanding Software Engineering to develop Secure Systems?

Specification,
Verification and Validation

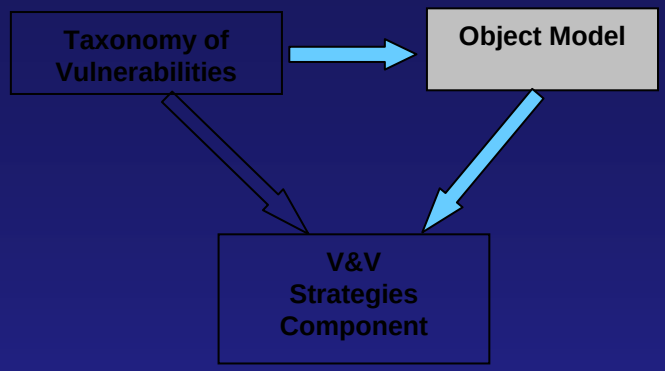
The Framework

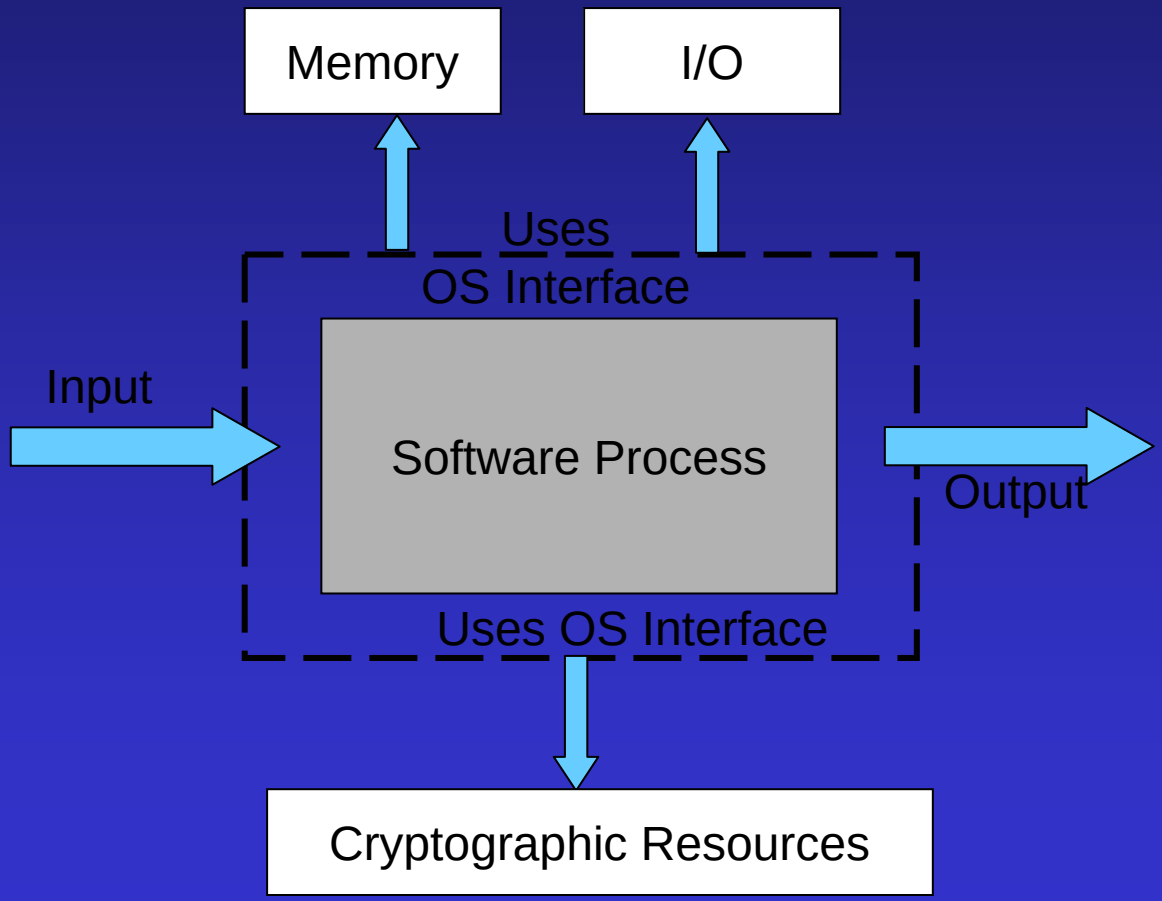


Taxonomy of Vulnerabilities: Structure



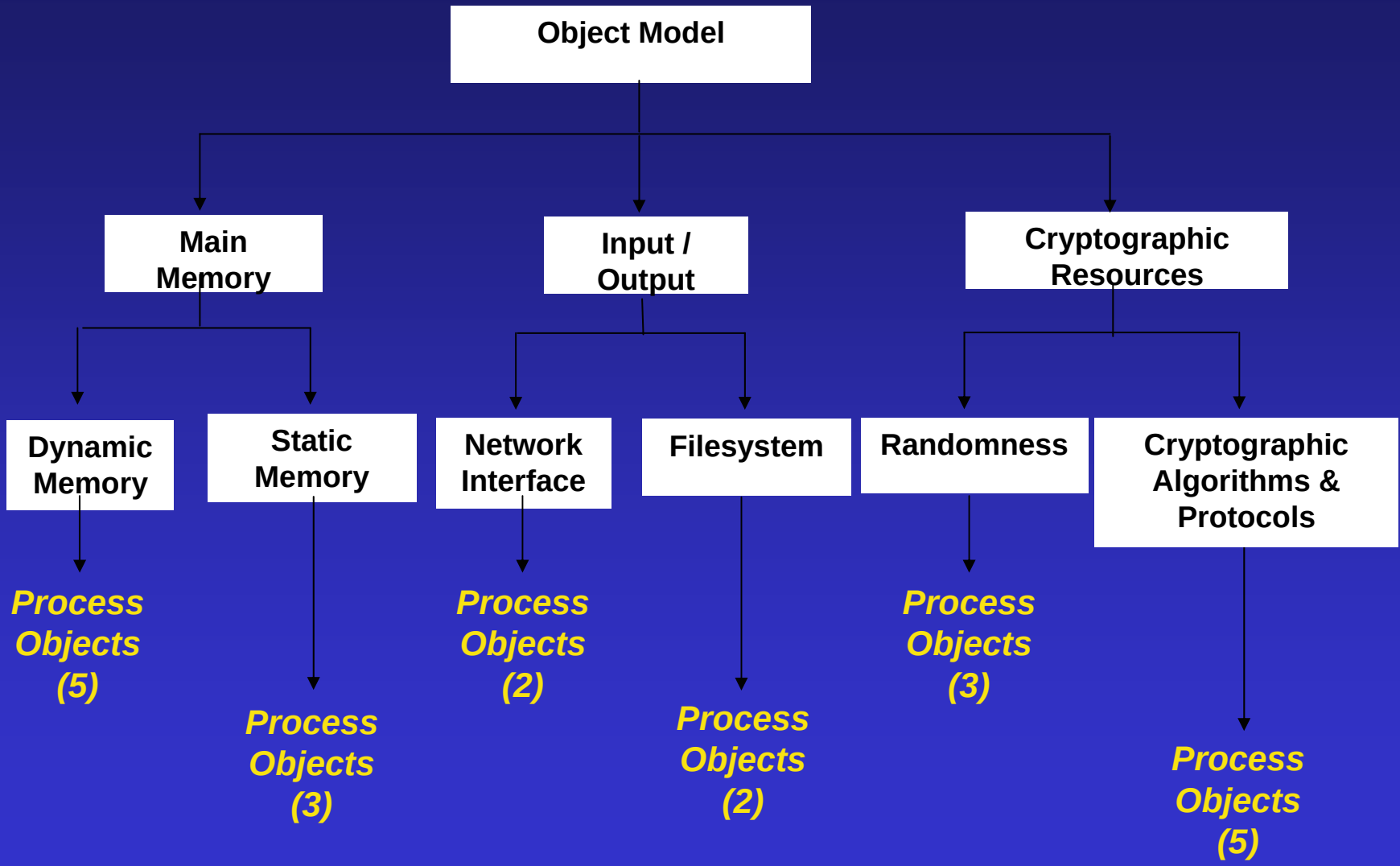
Process / Object Model Of Computing



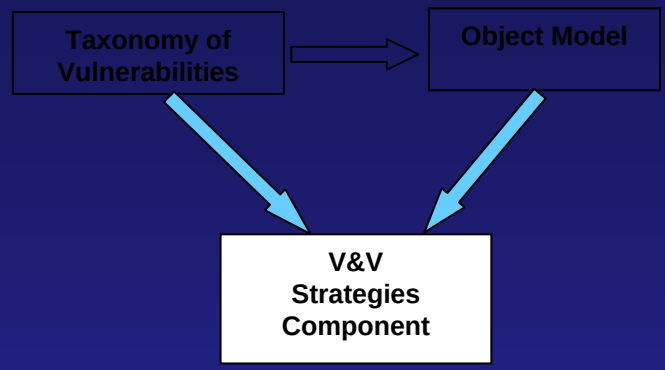


- Process objects
  - Holds data
  - Set of operations, e.g., create, remove, read etc.
  - Examples: files, directories, buffers, pointer variables, etc.

Object Model: Structure



V&V strategies component



- Guided by individual object / vulnerability relationships as defined by the taxonomy
  - Develop base strategies to test for presence of vulnerabilities
    - Target, Goal, and Method
  - Base Strategies → Test Strategies → Test Cases
- 46 base-strategies currently defined

Current Initiatives

Shvetha Soundararajan

A Software Structured Agile Approach to Software Development (MS)

Assessing Product and Process Quality: An AGILE Perspective (PhD)

Lee Clagett & Beau Frazier

Software Security: Access-Driven VV&T

The Proud, The Few….
The Software Engineers






But, when the truth is told….

We Are ALL Software Engineers