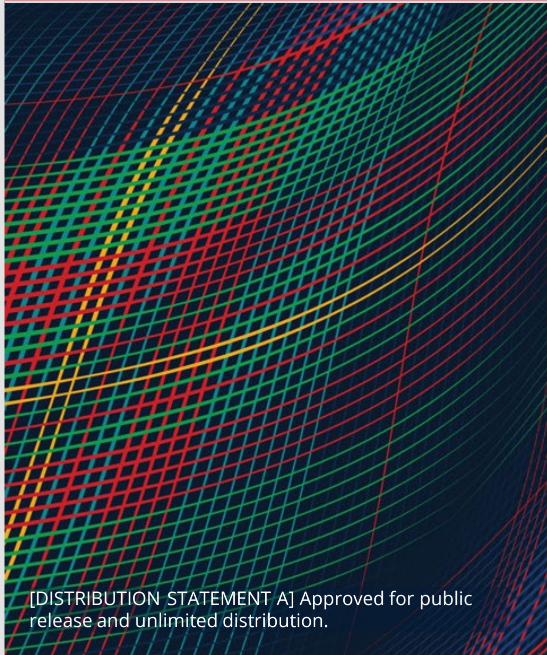
ADVANCING SOFTWARE for NATIONAL SECURITY
40 years
SOFTWARE ENGINEERING INSTITUTE

Carnegie Mellon University
Software Engineering Institute

Prioritizing and Testing Non-Functional Requirements: A Practical Guide

AUGUST 19, 2025

Lyndsi Hughes
Senior Systems Engineer



Agenda

- Background
- Technique #1 – Requirements Analysis
- Technique #2 – Creative Testing
- Conclusions

Software Engineering is Hard

But we adapt to improve software quality

Carnegie Mellon University Software Engineering Institute

Software Quality is "the degree to which software possesses a desired combination of attributes." [1]

Technique #1

Well-Reasoned Requirements Analysis

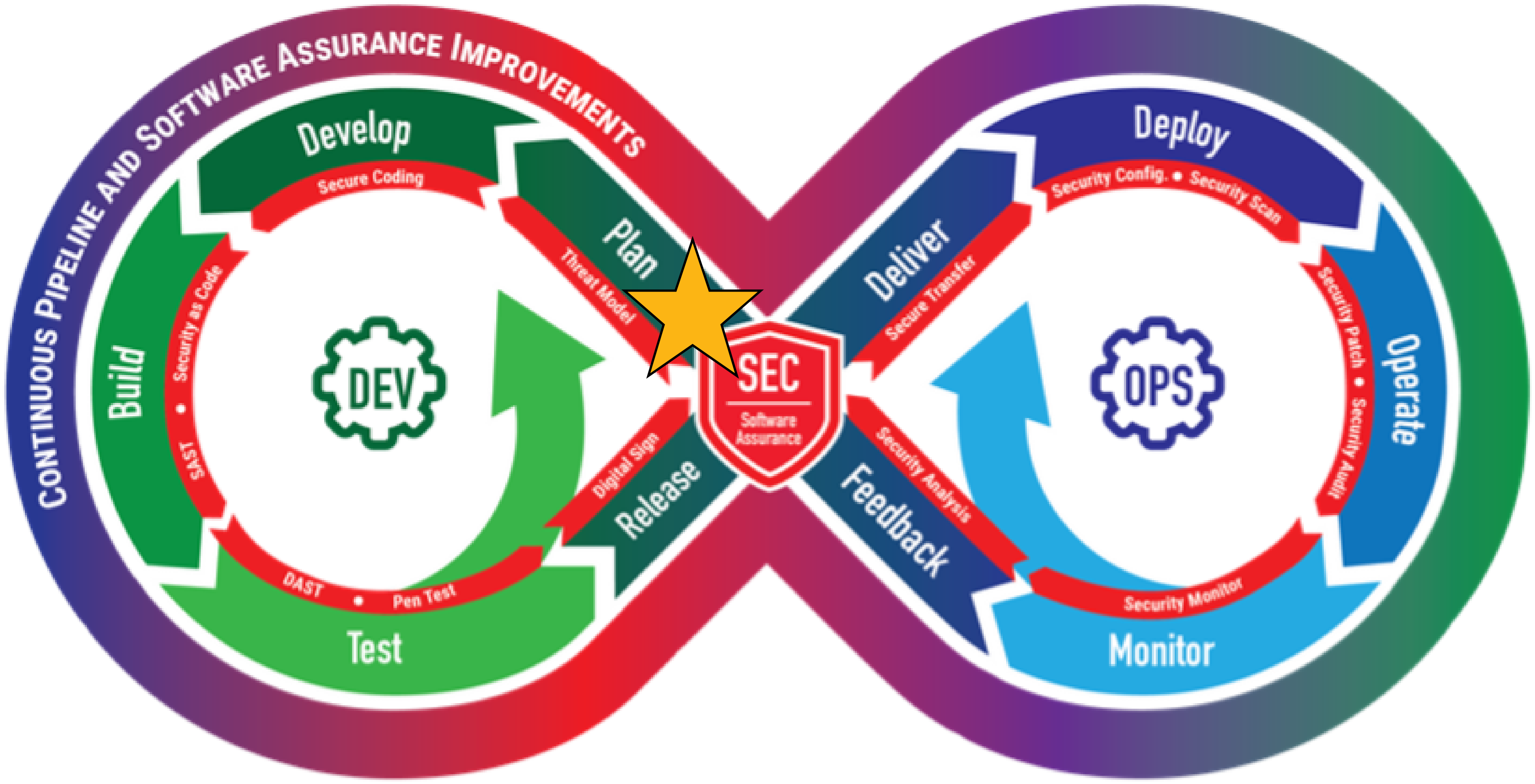
Technique #2

Modern Software Engineering Practices

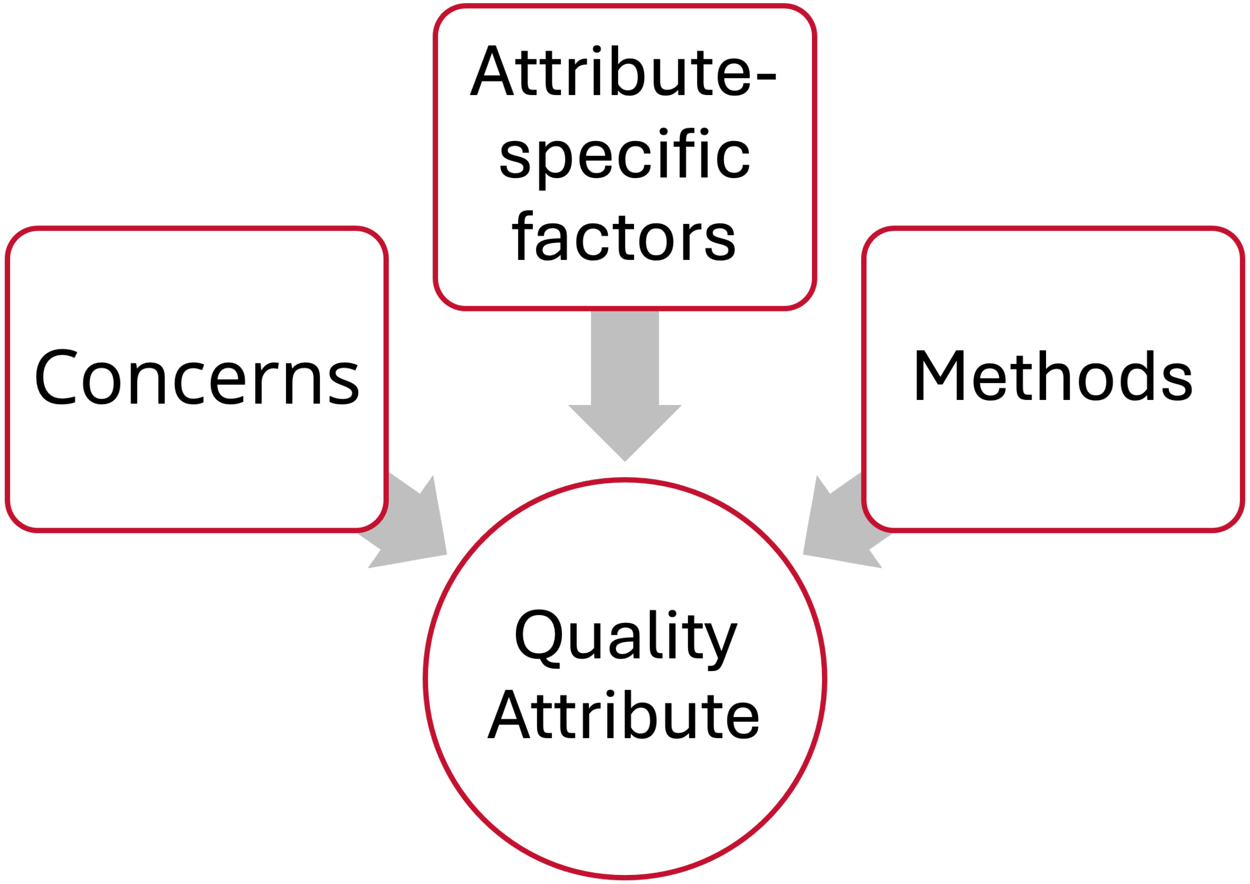
1: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=237006

Where do we start reasoning about requirements?

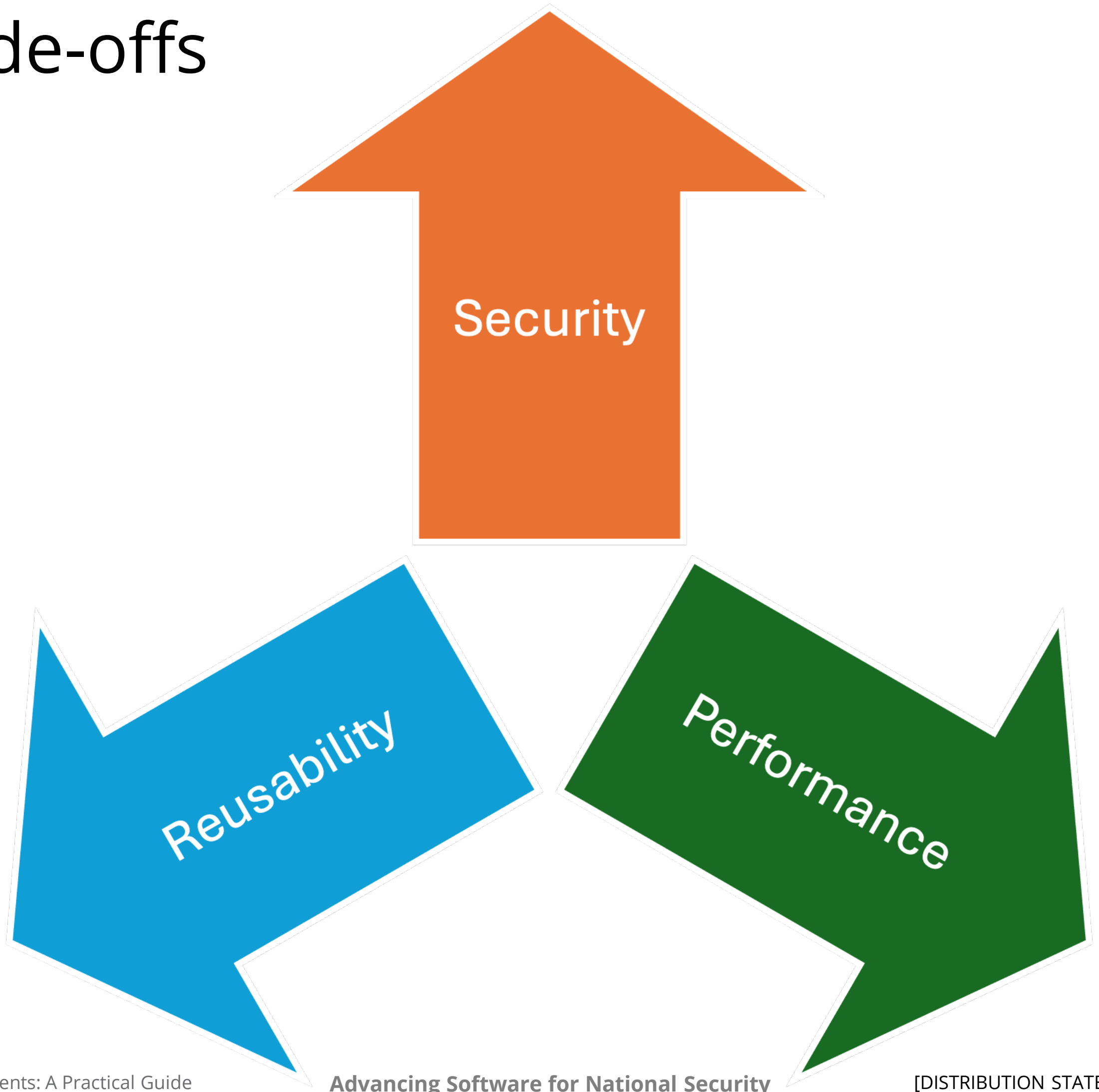
Carnegie Mellon University Software Engineering Institute


CONTINUOUS PIPELINE AND SOFTWARE ASSURANCE IMPROVEMENTS
Develop
Secure Coding
Plan
Threat Model
Build
Security as Code • SAST
DEV
Test
DAST • Pen Test
Release
Digital Sign
SEC
Software Assurance
Deliver
Secure Transfer
Deploy
Security Config. • Security Scan
OPS
Operate
Security Patch • Security Audit
Monitor
Security Monitor
Feedback
Security Analysis

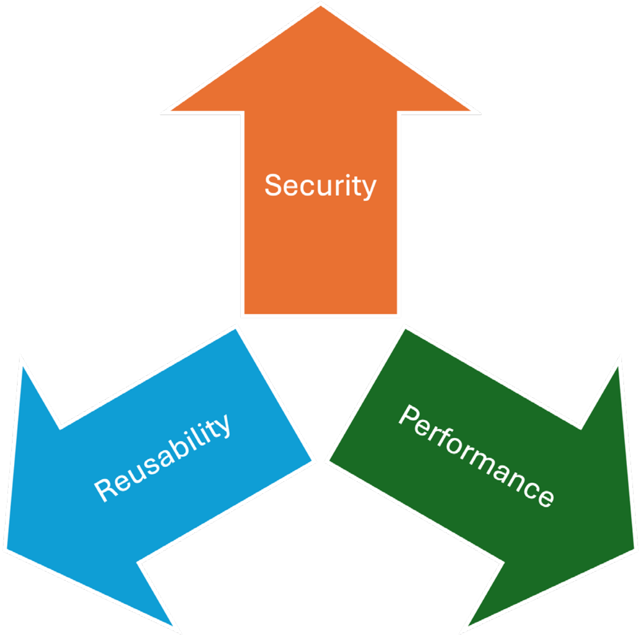
Building Blocks of Software Quality Attributes



Attribute Trade-offs



Prioritization Process



Methodology: Architecture Tradeoff Analysis Method (ATAM) [2]

Purpose: “To assess the consequences of architectural decisions in light of quality attribute requirements.”

Goals: To elicit, concretize, and prioritize the driving quality attribute requirements.

In Practice: Provides structure for decision making about trade-offs.

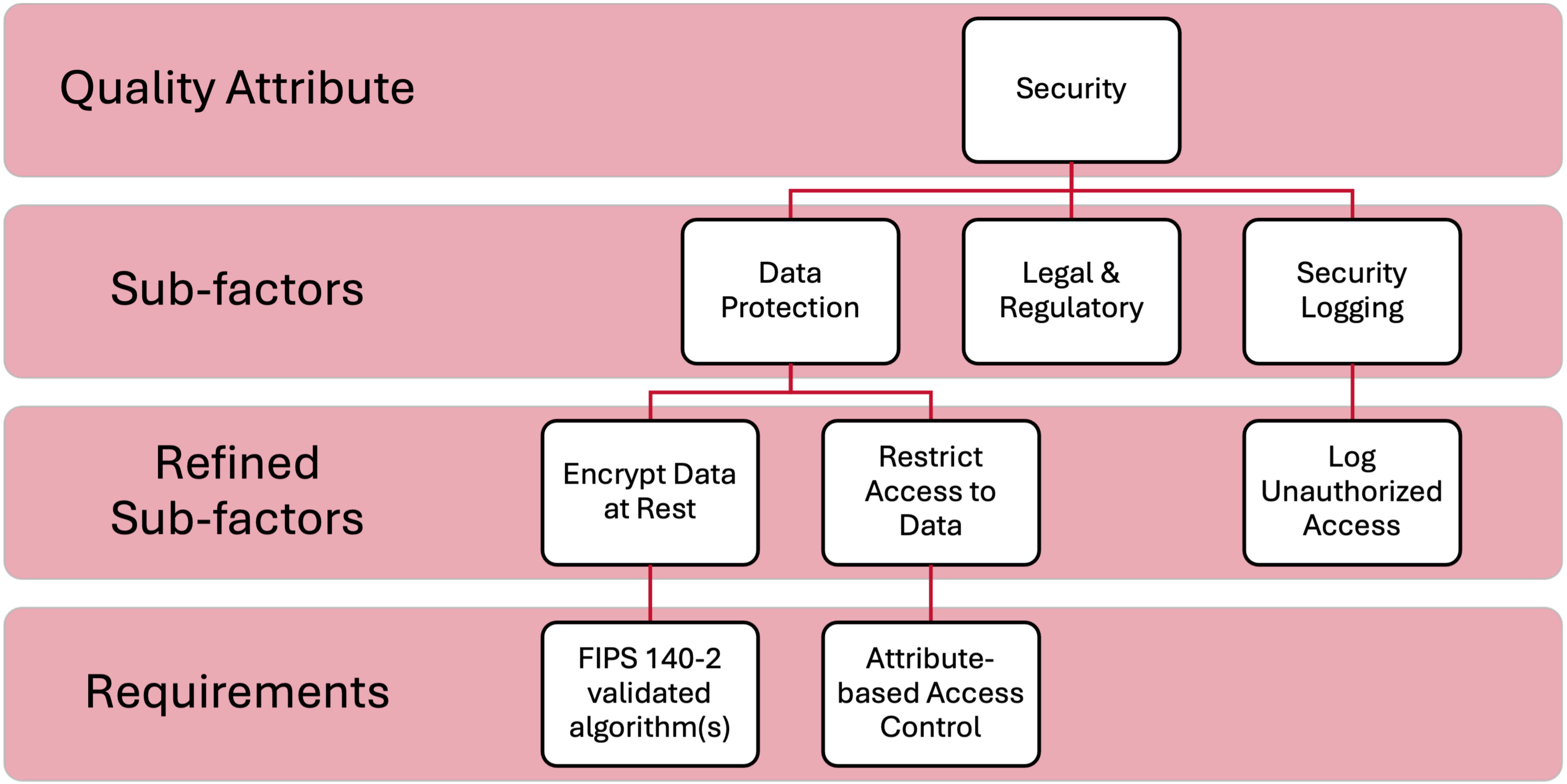
2: https://insights.sei.cmu.edu/library/atam-method-for-architecture-evaluation/

Security Quality Attribute Sub-factors

Carnegie Mellon University Software Engineering Institute

- Authentication & Authorization
- Disaster Recovery
- Third-party Dependencies
- Security Logging
- Legal & Regulatory
- Data Protection
- Encryption

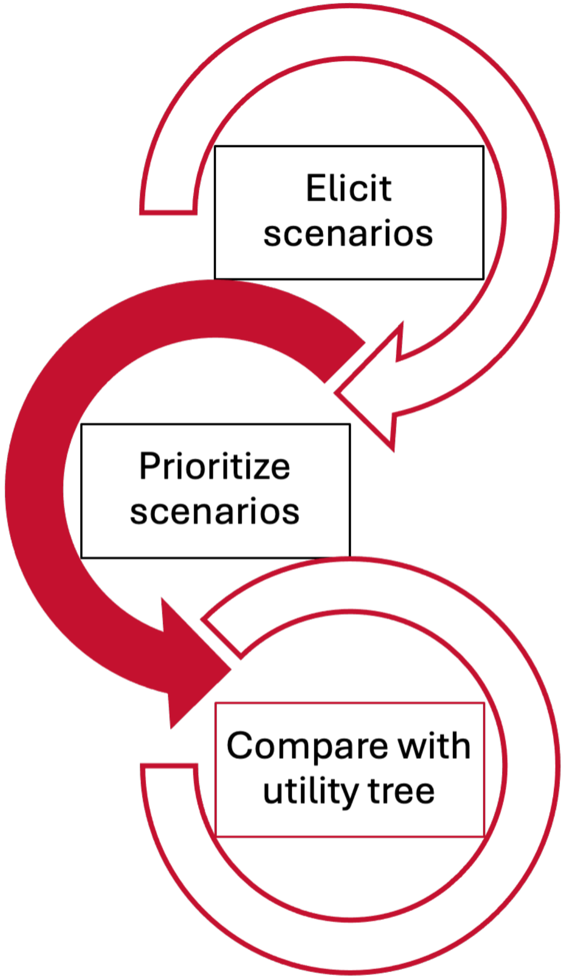
Building a Utility Tree



Brainstorming Scenarios

Goals:

- Represent stakeholders' interests
- Understand quality attribute requirements



Planning Process Accomplishments

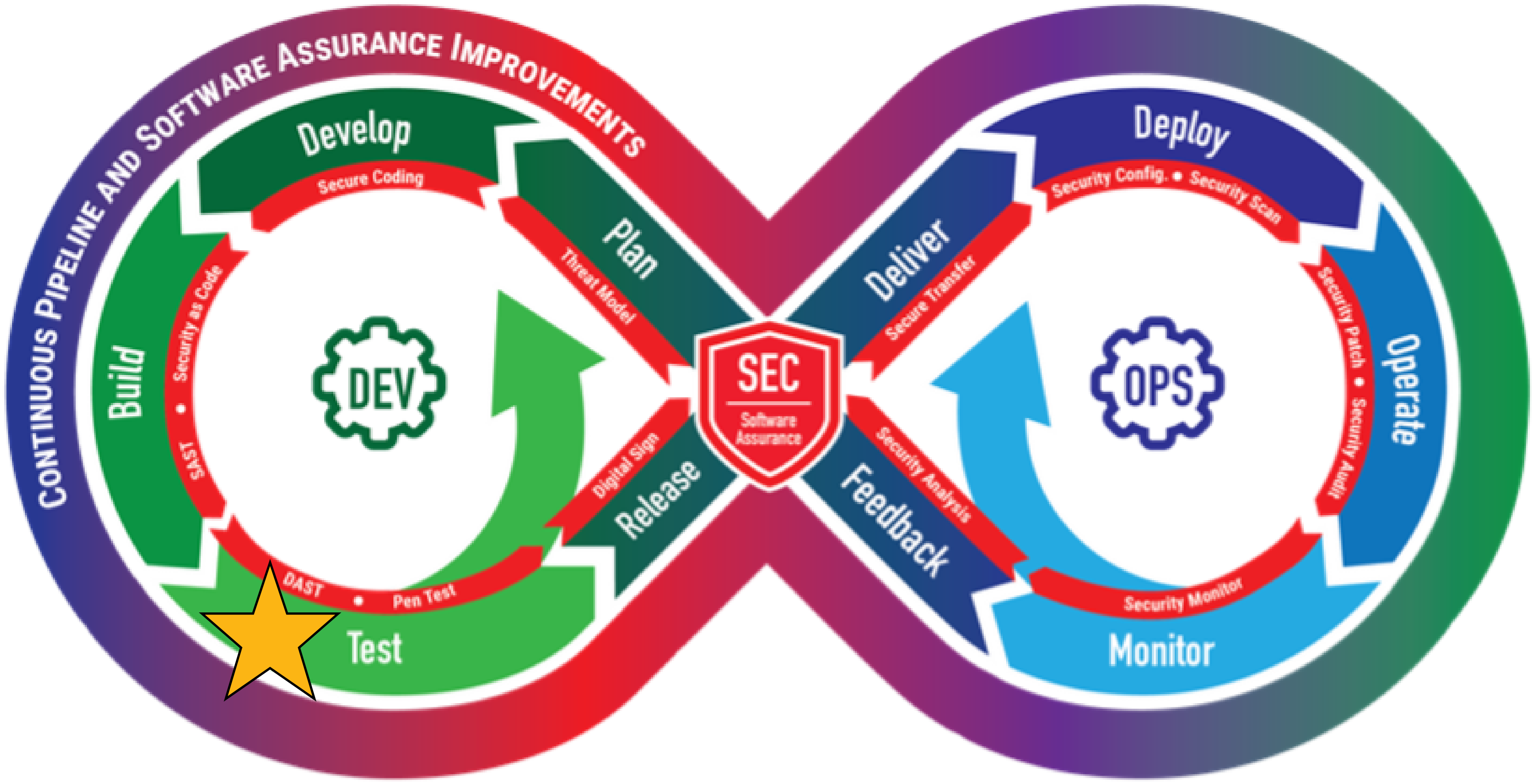
Requirements are:

- Precisely defined
- Measurable
- Prioritized

Software design is:

- Well understood
- Aligned with requirements

How do we measure success?


CONTINUOUS PIPELINE AND SOFTWARE ASSURANCE IMPROVEMENTS
Develop
Secure Coding
Plan
Threat Model
Build
Security as Code
SAST
DEV
Test
DAST
Pen Test
Release
Digital Sign
SEC
Software Assurance
Deliver
Secure Transfer
Deploy
Security Config.
Security Scan
Operate
Security Patch
Security Audit
OPS
Monitor
Security Monitor
Feedback
Security Analysis

Traditional Goals of Software Testing

- Unit testing verifies that individual functions in the code produce the expected output
- Regression testing validates that recent code changes won't break existing functionality
- Performance testing ensures that the application can handle higher loads and stress
- Static Analysis Security Testing (SAST) scans software for known security vulnerabilities

Carnegie Mellon University Software Engineering Institute

Threats to Testing Success

Unaddressed challenges:

- Inadequate testing tools
- Manual testing procedures
- Incomplete testing procedures

Negative Outcomes:

- Inconsistent test results can negatively impact software quality
- Poor software quality can negatively impact business success

Designing Your Test Plan

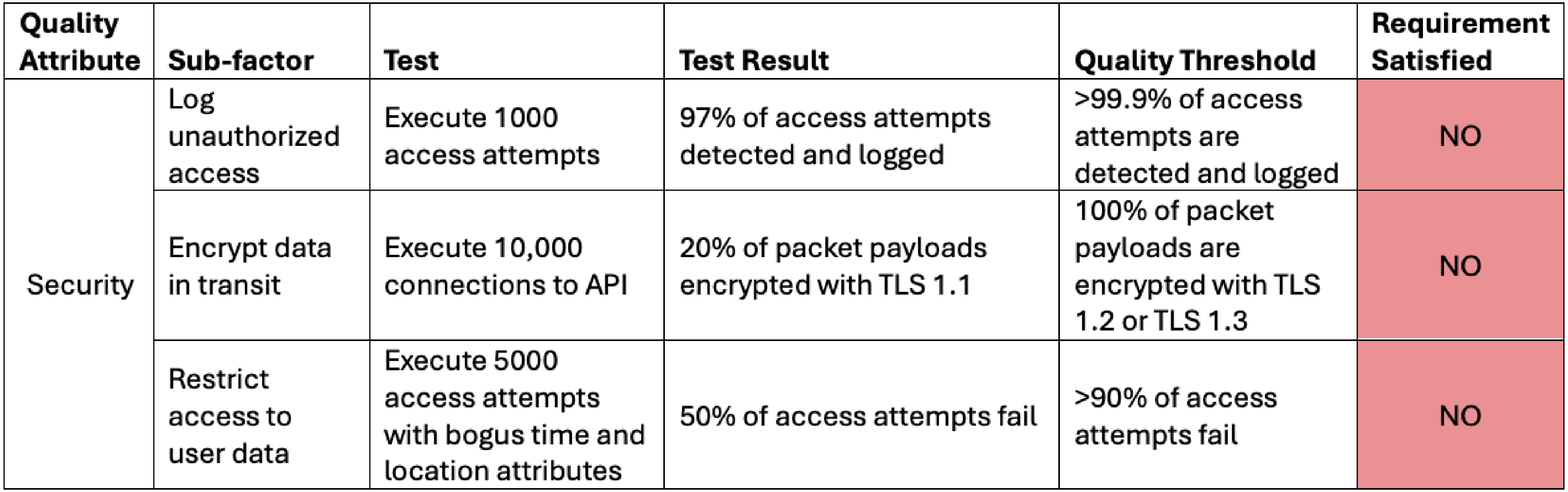
Test what you care about: **Your Requirements!**

When designing your tests:

- Get automated
- Get objective
- Get creative

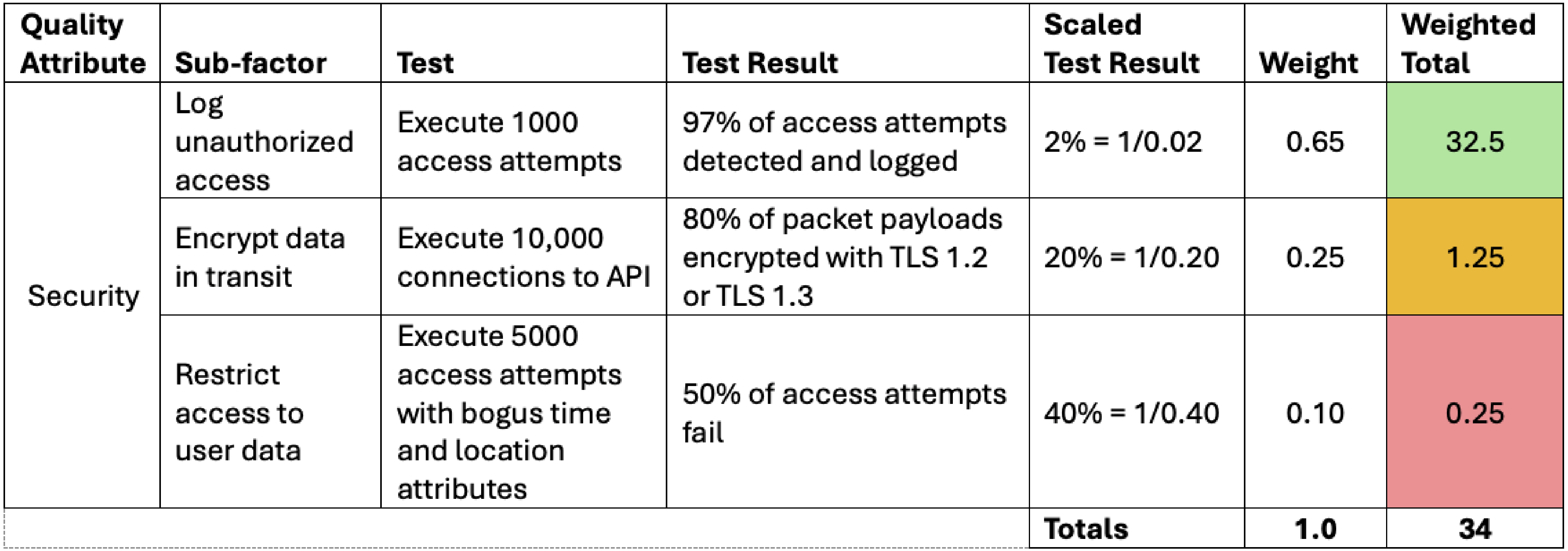
The metrics you collect will provide decision support

Decision Matrix



Quality Attribute	Sub-factor	Test	Test Result	Quality Threshold	Requirement Satisfied
Security	Log unauthorized access	Execute 1000 access attempts	97% of access attempts detected and logged	>99.9% of access attempts are detected and logged	NO
	Encrypt data in transit	Execute 10,000 connections to API	20% of packet payloads encrypted with TLS 1.1	100% of packet payloads are encrypted with TLS 1.2 or TLS 1.3	NO
	Restrict access to user data	Execute 5000 access attempts with bogus time and location attributes	50% of access attempts fail	>90% of access attempts fail	NO

Weighted Decision Matrix



Quality Attribute	Sub-factor	Test	Test Result	Scaled Test Result	Weight	Weighted Total
Security	Log unauthorized access	Execute 1000 access attempts	97% of access attempts detected and logged	2% = 1/0.02	0.65	32.5
	Encrypt data in transit	Execute 10,000 connections to API	80% of packet payloads encrypted with TLS 1.2 or TLS 1.3	20% = 1/0.20	0.25	1.25
	Restrict access to user data	Execute 5000 access attempts with bogus time and location attributes	50% of access attempts fail	40% = 1/0.40	0.10	0.25
				Totals	**1.0**	**34**

Prioritizing and Testing Non-Functional Requirements: A Practical Guide

Conclusions

Carnegie Mellon University Software Engineering Institute

Conclusions

There will be tradeoffs

- Understand all your requirements
- ATAM provides scaffolding for reasoning about the tradeoffs between requirements
- Scaffolding materials can be prepared in advance

There must be measurements

- Metrics relate directly to requirements
- Objective metrics support sound design decisions

Questions

Carnegie Mellon University Software Engineering Institute

Team Acknowledgements

Lyndsi Hughes
Senior Systems Engineer

Lori Flynn
Senior Software Security Researcher

Vanessa Jackson
Senior Engineer

Joseph Sible
Software Engineer

Telephone:
+1 412.268.5800

Email:
info@sei.cmu.edu