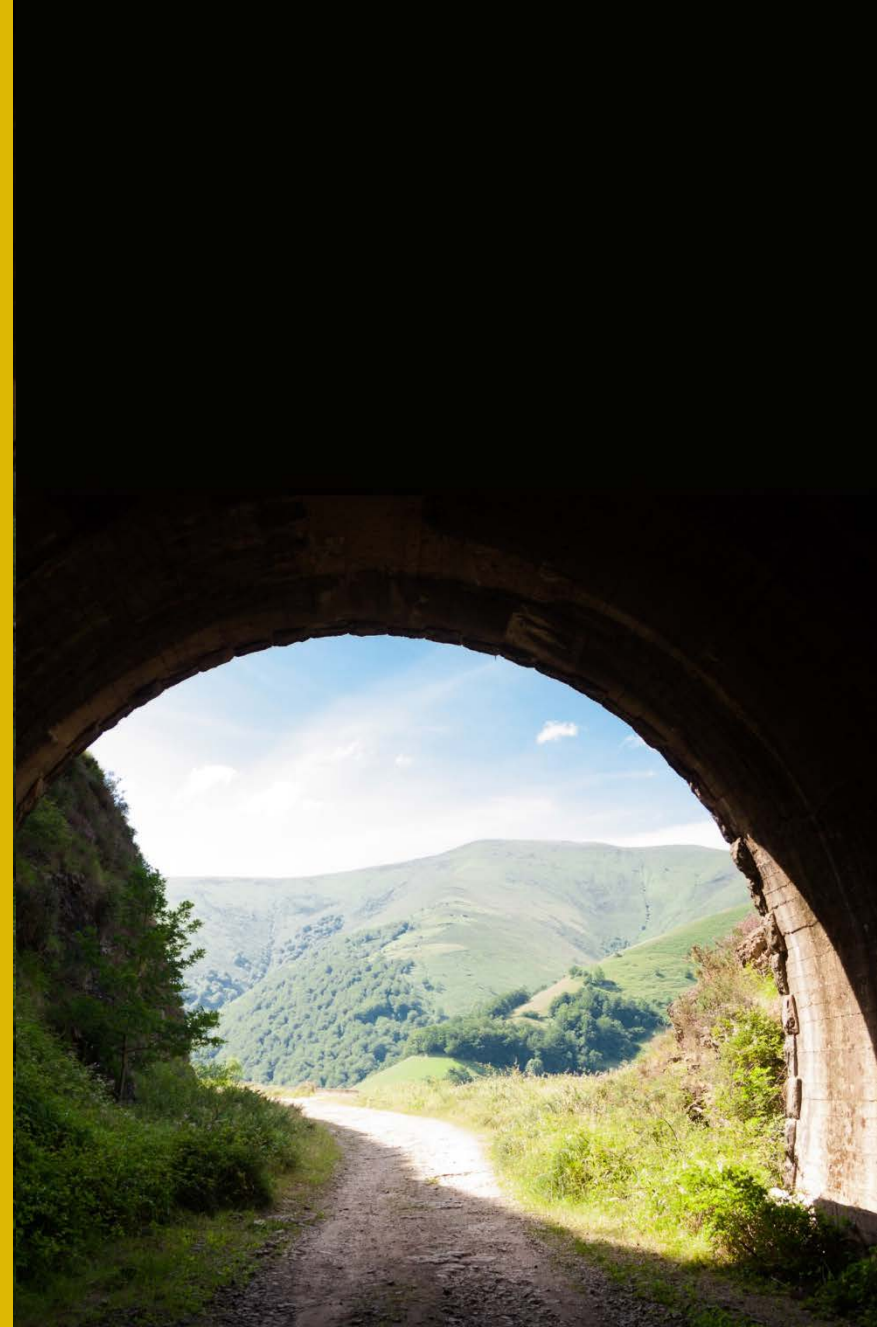


SQUARE Workshop

SQUARE Overview

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213



Software Engineering Institute

Carnegie Mellon University

© 2017 Carnegie Mellon University

[Distribution Statement A] This material has been approved for public release and unlimited distribution.

Copyright 2017 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material is distributed by the Software Engineering Institute (SEI) only to course attendees for their own individual study.

Except for any U.S. government purposes described herein, this material SHALL NOT be reproduced or used in any other manner without requesting formal permission from the Software Engineering Institute at permission@sei.cmu.edu.

Although the rights granted by contract do not require course attendance to use this material for U.S. Government purposes, the SEI recommends attendance to ensure proper understanding.

Carnegie Mellon® and CERT® are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM17-0350

SQUARE Overview





Understand the challenges of security requirements engineering

Learn the importance of developing security requirements in the same timeframe as functional requirements.

Learn why the methods used to identify functional requirements may not work directly for security requirements.

Be exposed to methods for security risk analysis, security requirements elicitation, and security requirements prioritization.

Learn how to apply the SQUARE method for security requirements engineering.

I. Requirements Engineering

II. Introduction to SQUARE

Requirements Engineering



- RE defects cost up to 200 times more once fielded than if caught in requirements engineering
- Reworking defects consumes >50% of project effort
- >50% of defects are introduced in requirements engineering
- Takeaway: Errors during requirements engineering are costly!

- Requirements identification may not include relevant stakeholders
- Requirements analysis may or may not be performed
- Requirements specification are typically haphazard

Bad requirements cause projects to

- exceed schedule
- exceed budget
- have significantly reduced scope
- deliver poor-quality applications
- deliver products that are not significantly used
- be cancelled

Think About the Following

Have you seen examples of requirements problems that caused projects to

- exceed schedule
- exceed budget
- have significantly reduced scope
- deliver poor-quality applications
- deliver products that are not significantly used
- be cancelled

- Address security in a particular application
- Are often ignored in the requirements elicitation process
- Incur high costs when incorporated later
- Must be addressed early

Security Requirements Engineering Issues – Example

Cost of Fixing Vulnerabilities Later

Stage	Critical Bugs Identified	Cost of Fixing One Bug	Cost of Fixing All Bugs
Requirements		\$139	
Design		\$455	
Coding		\$977	
Testing	50	\$7,136	\$356,800
Maintenance	150	\$14,102	\$2,115,300
Total	200		\$2,472,100

Cost of Fixing Vulnerabilities Early

Stage	Critical Bugs Identified	Cost of Fixing One Bug	Cost of Fixing All Bugs
Requirements		\$139	
Design		\$455	
Coding	150	\$977	\$146,550
Testing	50	\$7,136	\$356,800
Maintenance		\$14,102	
Total	200		\$503,350

As can be seen, identifying defects early in the lifecycle reduced costs by nearly \$2 million.

Microsoft Windows: 45% Fewer Vulnerabilities in Windows Vista

Windows Vista was the first Microsoft operating system to benefit from the SDL. After the first year, Windows Vista had 45% fewer vulnerabilities than Windows XP. In a comparison of security vulnerabilities, Windows Vista also fares better than competing operating systems

Microsoft SQL Server: 91% Fewer Vulnerabilities in SQL Server 2005

SQL Server serves as an excellent example for security improvements resulting from incorporating the SDL. Within the three years after release, Microsoft has issued three security bulletins for the SQL Server 2005 database engine

Reference:

<http://www.microsoft.com/security/sdl/learn/measurable.aspx>

- SQUARE
- CLASP
- Core Security Requirements Artifacts
- SREP
- Security Patterns
- TROPOS
- Others

SQUARE Methodology

What is It? Who is Involved?



SQUARE

- Security Quality Requirements Engineering
- Nine-step process
- SQUARE-Lite
- SQUARE for Privacy
- SQUARE for Acquisition
- Can be used with existing requirements engineering process

- Developed by the CERT Division at the SEI, Carnegie Mellon University
- Stepwise methodology for eliciting, categorizing, and prioritizing security requirements for information technology systems and applications
- Security requirements are quality attributes

Who is involved?

- stakeholders of the project
- requirement engineers with security expertise

In SQUARE, security requirements are

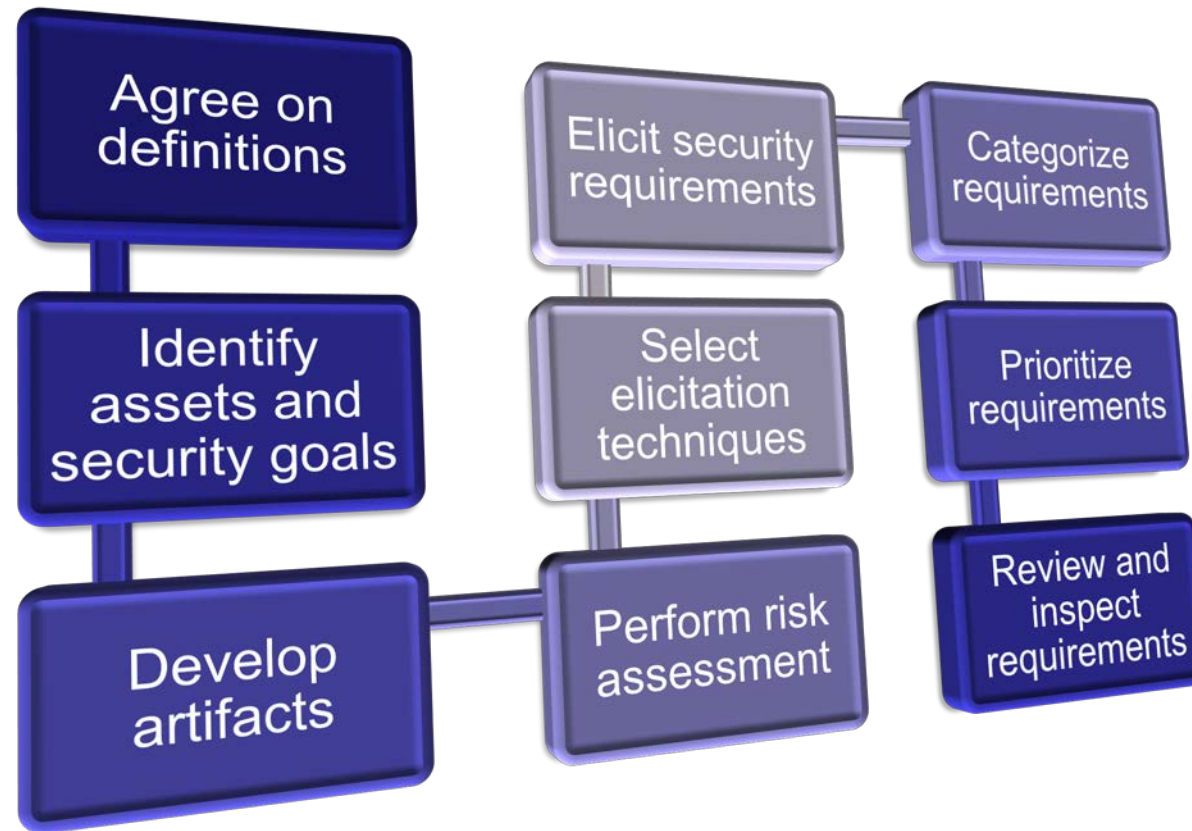
- treated at the same time as the system's functional requirements, AND
- specified in the early stages of the SDLC
- specified in similar ways as software requirements engineering and practices
- determined through a process of nine discrete steps

SQUARE Steps

The Nine Steps



The SQUARE Process



Materials and tool at <http://www.cert.org/sse/square/>

1. Agree on definitions.
2. Identify assets and security goals.
3. Develop artifacts to support security requirements definition.
4. Assess risks.
5. Select elicitation technique(s).
6. Elicit security requirements.
7. Categorize requirements.
8. Prioritize requirements.
9. Inspect requirements.

1	2	3	4	5	6	7	8	9
Def.	Goals	Artifacts	Risk	Technique	Elicit	Categorize	Prioritize	Inspect

Agree on Definitions

- Requirements engineers and stakeholders agree on a set of definitions.
- Process is carried out through interviews.
- Exit criteria: documented set of definitions
- Examples: non-repudiation, denial-of-service (DoS), intrusion, malware

1	2	3	4	5	6	7	8	9
Def	Goals	Artifacts	TechniqueRisk		Ellicit	Categorize	Prioritize	Inspect

Identify Assets and Security Goals

- Identify assets to be protected in the system.
- Goals are required to identify the priority and relevance of security requirements.
- Security goals must support the business goal.
- Goals are reviewed, prioritized, and documented.
- Exit criteria: one business goal, several security goals

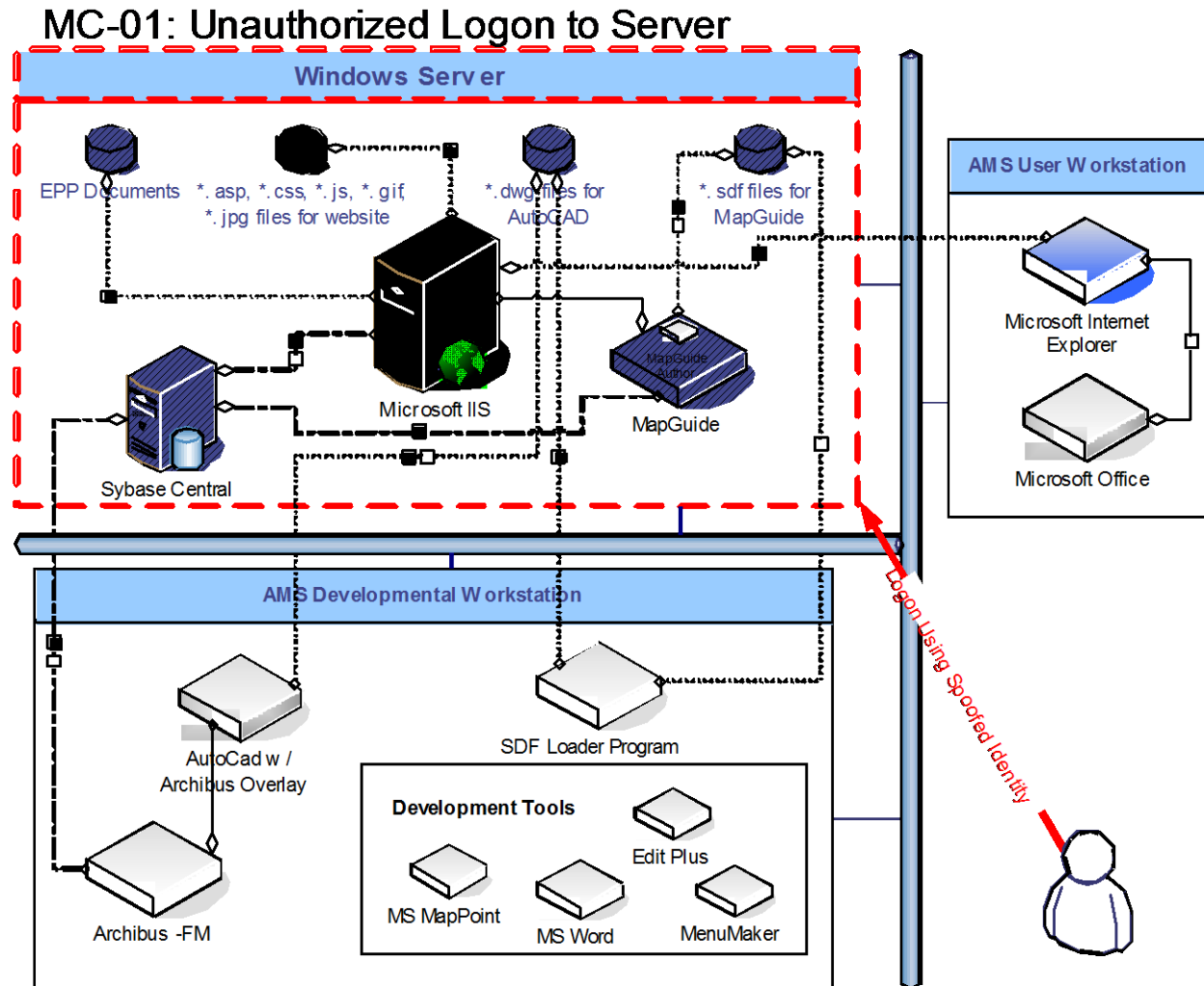
9	Inspect
8	Prioritize
7	Categorize
6	Elicit
5	Risk
4	Technique
3	Artifacts
2	Goals
1	Def.

Develop Artifacts

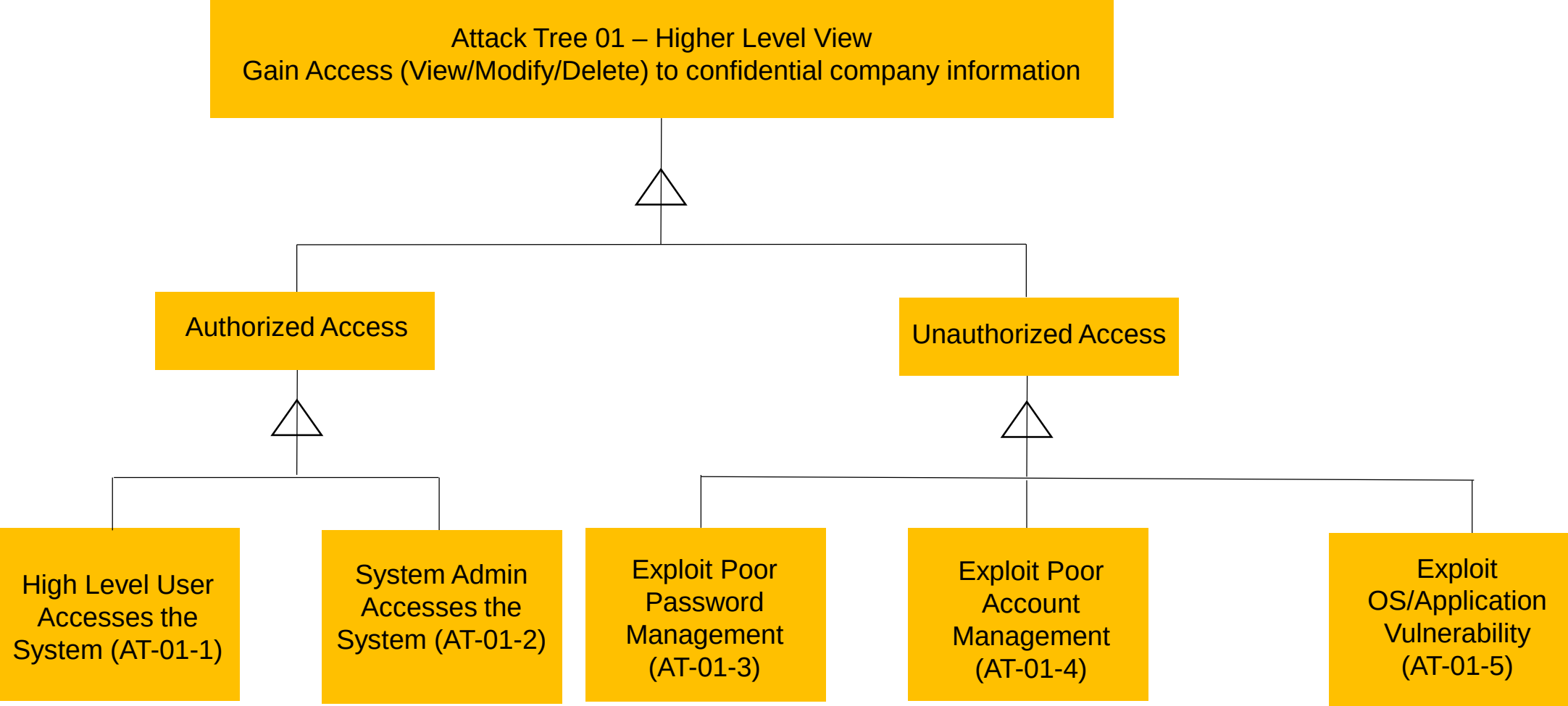
- Collect or create artifacts that will facilitate generation of security requirements.
- Jointly verify their accuracy and completeness.
- Examples: system architecture diagrams, use/misuse case scenarios/diagrams, attack trees, templates and forms

Examples of Artifacts

Misuse Case Diagram



Attack Tree



1	2	3	4	5	6	7	8	9
Def	Goals	Artifacts	Risk	Technique	Elicit	Categorize	Prioritize	Inspect

Perform Risk Assessment

- Identify threats to the system and its vulnerabilities.
- Calculate likelihood of their occurrence. Classify them.
This will also help in prioritizing requirements later.
- Risk expert might be required.
- Exit criteria: documentation of all threats, their likelihood and classifications

9	8	7	6	5	4	3	2	1
Inspect	Prioritize	Categorize	Elicit	Technique	Risk	Artifacts	Goals	Def.

Select Elicitation Technique

- Select appropriate technique for the number and expertise of stakeholders, requirements engineers, and size and scope of the project.
- Techniques: structured/unstructured interviews, **accelerated requirements method (ARM)**, soft systems methodology, issue based information systems (IBIS), Quality Function Deployment

1	2	3	4	5	6	7	8	9
Def.	Goals	Artifacts	Risk	Technique	Elicit	Categorize	Prioritize	Inspect

Elicit Security Requirements

- Execute the elicitation technique.
- Avoid non-verifiable, vague, ambiguous requirements.
- Concentrate on what, not how.
Avoid implementations and architectural constraints.
- Exit criteria: initial document with requirements

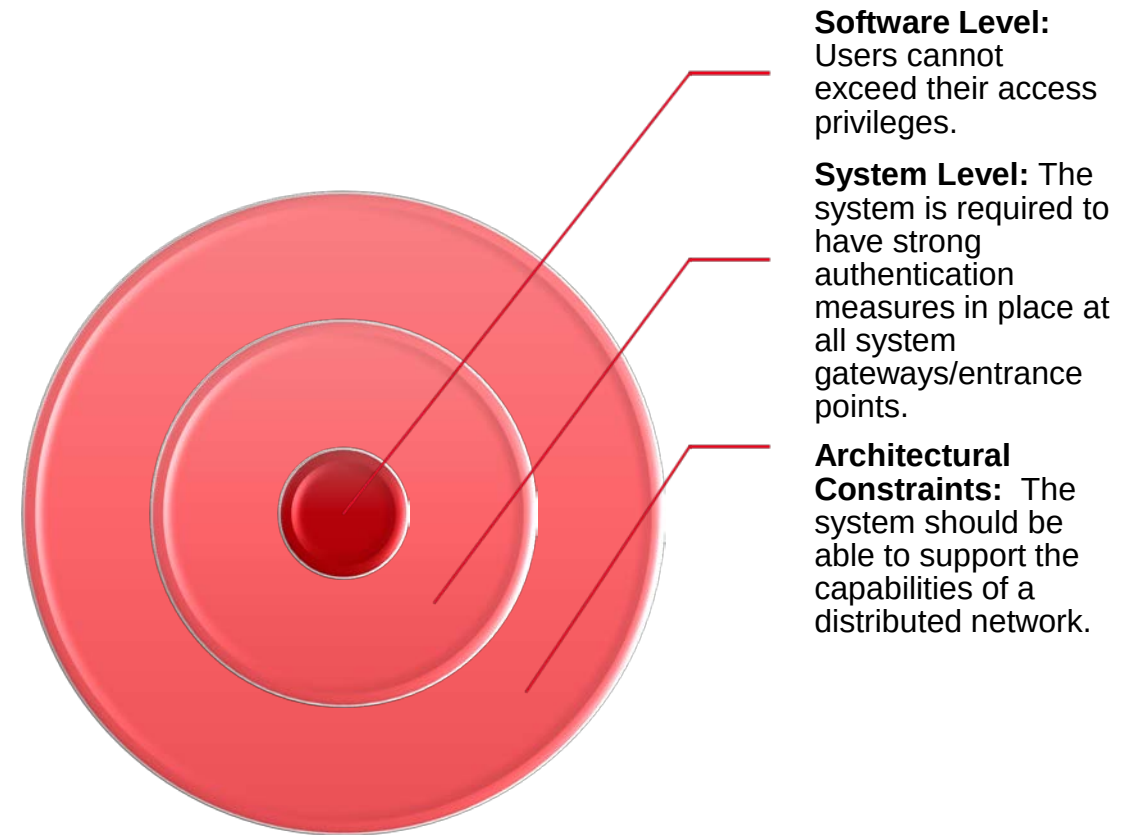
9	8	7	6	5	4	3	2	1
Inspect	Prioritize	Categorize	Elaborate	Techniques	Risk	Artifacts	Goals	Definition

Categorize Requirements

- Classify requirements into essential, non-essential, system, software, or architectural constraints.
- Sample table:

	System level	Software level	Architectural constraint
Req. 1			
Req. 2			

Step 7- Categorize Requirements Examples



1	2	3	4	5	6	7	8	9
Def.	Goals	Artifacts	Risk	Techniques	Eligibility	Categorization	Prioritize	Inspect

Prioritize Requirements

- Use risk assessment and categorization results to prioritize requirements.
- Prioritization techniques: Triage, Win-Win, Analytical Hierarchy Process
- Requirements engineering team should produce a cost-benefit analysis to aid stakeholders.

9	8	7	6	5	4	3	2	1
Inspect	Prioritize	Categorize	Elaborate	Techniques	Risk	Artifacts	Goals	Def.

Requirements Inspection

- Inspection aids in creating accurate and verifiable security requirements.
- Look for ambiguities, inconsistencies, mistaken assumptions.
- Fagan inspections / peer reviews
- Exit criteria: all requirements verified and documented

The full SQUARE process

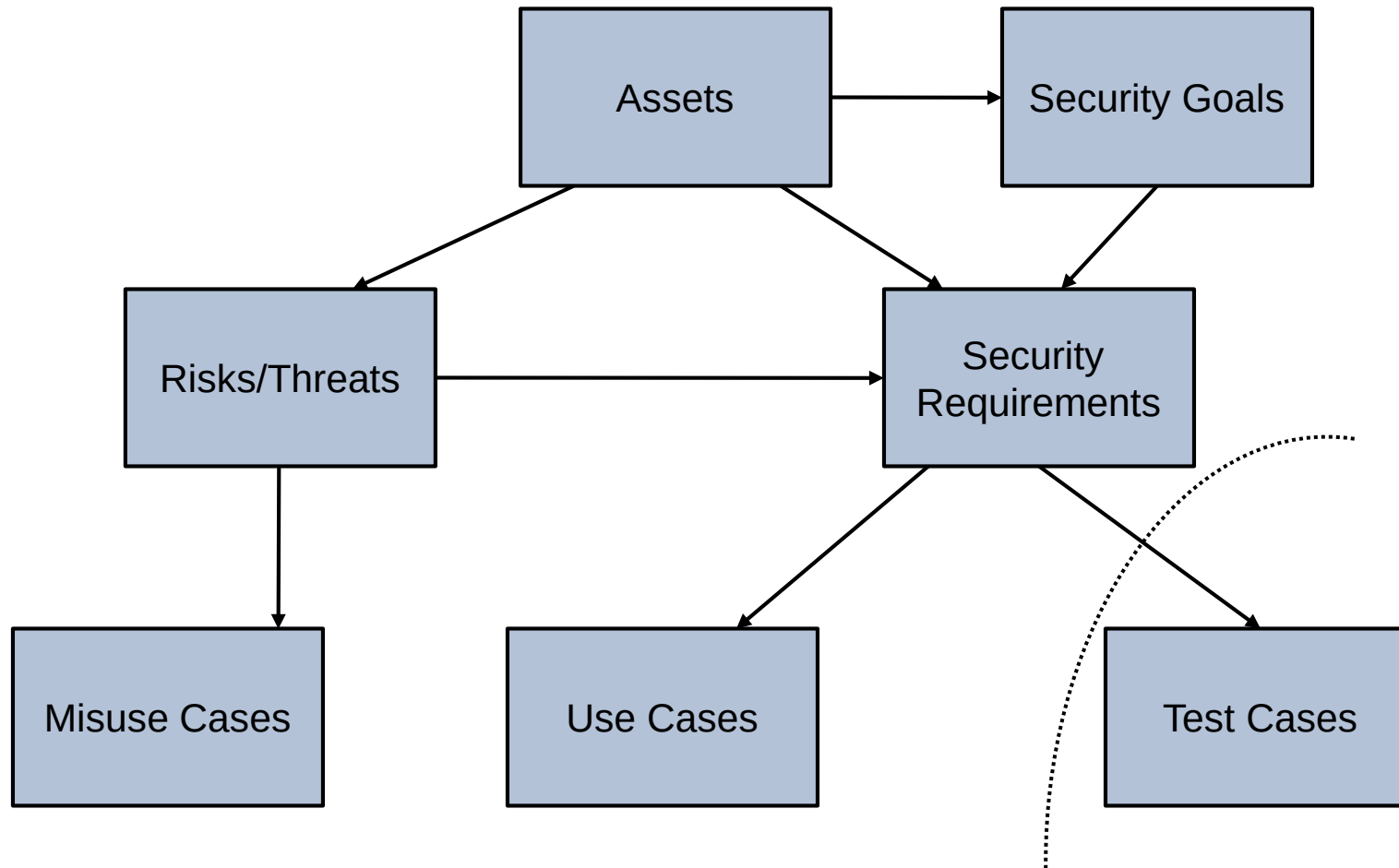
- takes about three months elapsed time to complete
- has been implemented in several case studies
- Interest has increased significantly in recent years, with focus on risk assessment and threat modeling

SQUARE-Lite

- Agree on definitions.
- Identify assets and security goals.
- Perform risk assessment.
- Elicit security requirements.
- Prioritize requirements.

SQUARE-Lite has been implemented in one case study.

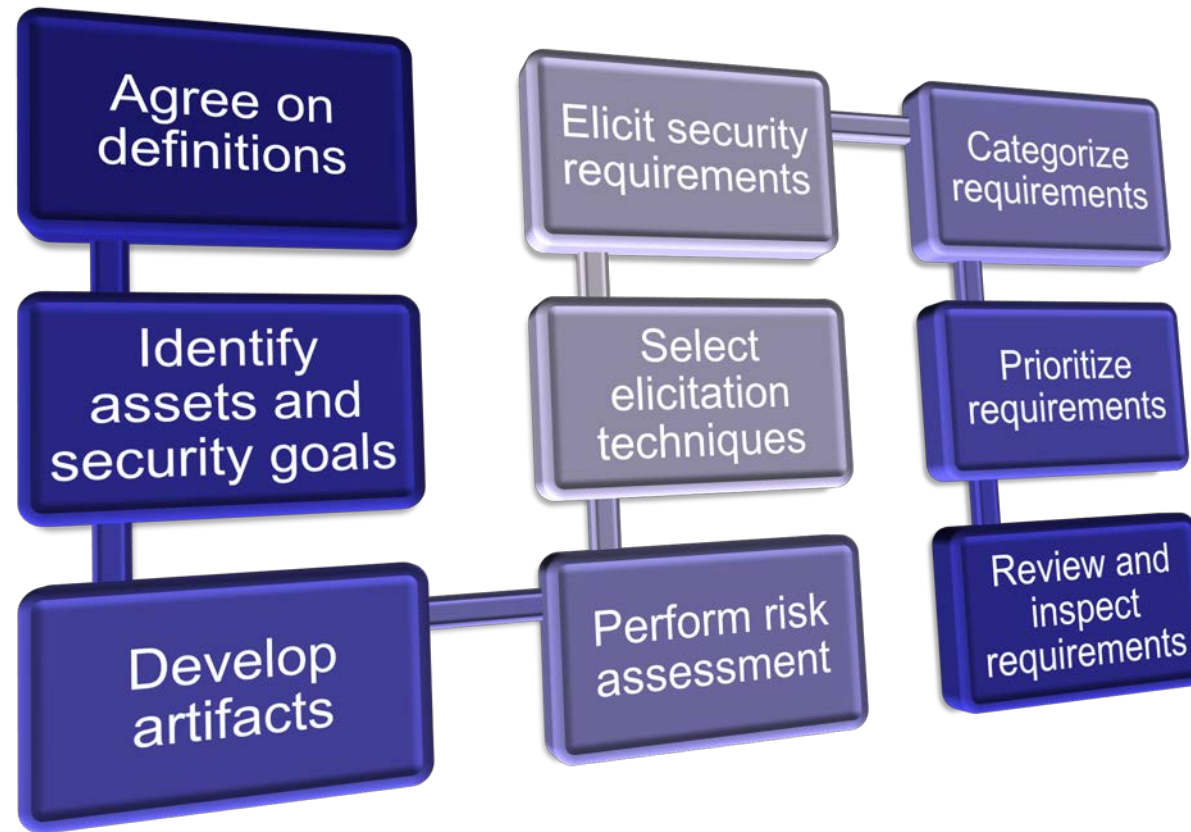
Business Goal



Summary



The SQUARE Process



Materials and tool at <http://www.cert.org/sse/square/>

- SQUARE-Lite – 5-step process
- P-SQUARE – SQUARE for both security and privacy
- A-SQUARE – SQUARE for acquisition

- SQUARE home page, including tool and demo video at <http://www.cert.org/sse/square/>
- Chapter 3 in the SQUARE course text *Software Security Engineering: A Guide for Project Managers* by Allen et al.
- BSI content on requirements engineering <https://buildsecurityin.us-cert.gov/>

- SQUARE Technical Report – SEI web site
<http://www.sei.cmu.edu/pub/documents/05.reports/pdf/05tr009.pdf>
- SQUARE Case Study Reports – SEI web site
- SQUARE Tool
<http://www.cert.org/cybersecurity-engineering/products-services/p-square-download.cfm>
- *Integrating Security and Software Engineering*
IDEA Group Publishing
<http://www.idea-group.com>

SQUARE Workshop

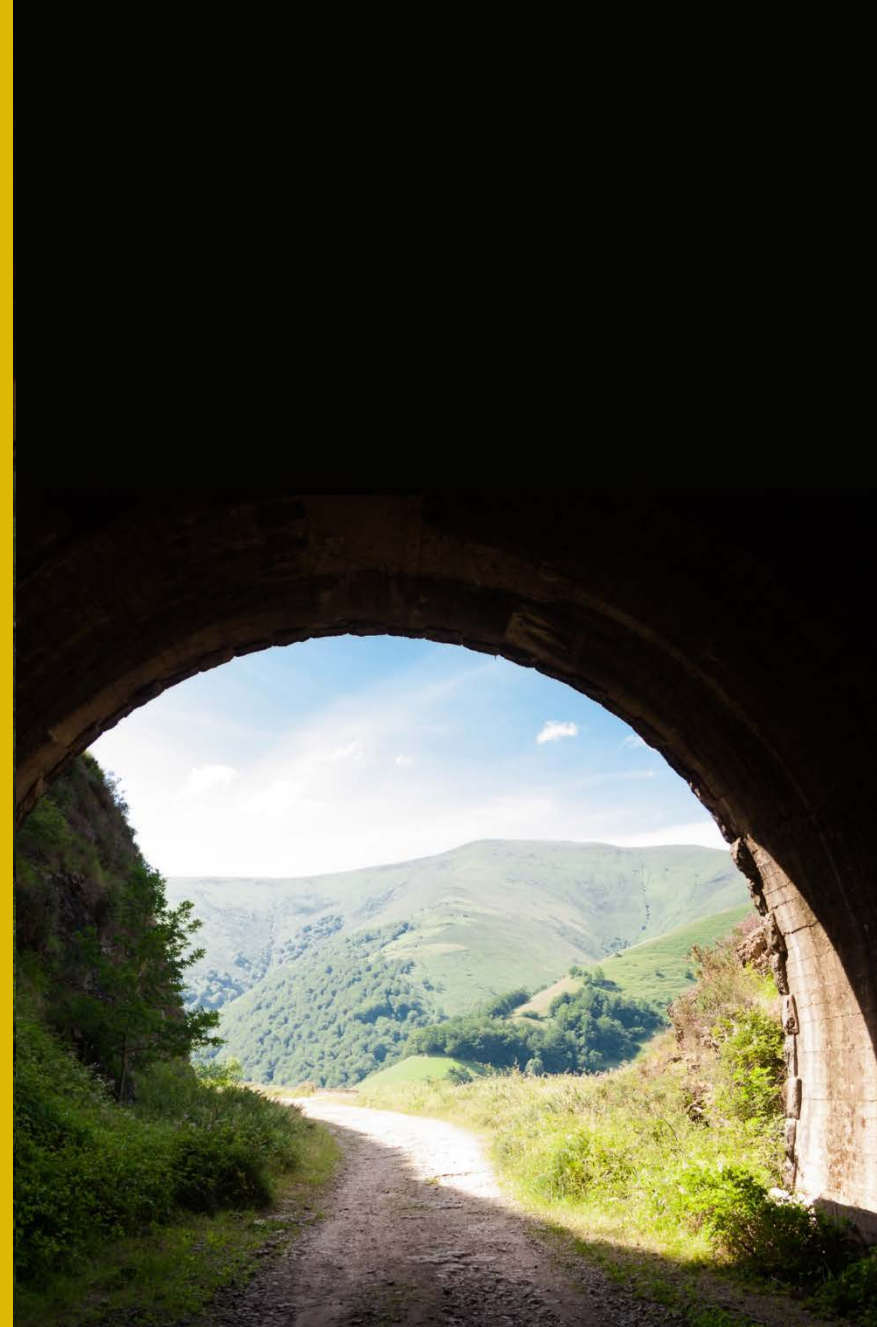
SQUARE for Acquisition

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213



Software Engineering Institute

Carnegie Mellon University



© 2017 Carnegie Mellon University

[Distribution Statement A] This material has been approved for public release and unlimited distribution.

Copyright 2017 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material is distributed by the Software Engineering Institute (SEI) only to course attendees for their own individual study.

Except for any U.S. government purposes described herein, this material SHALL NOT be reproduced or used in any other manner without requesting formal permission from the Software Engineering Institute at permission@sei.cmu.edu.

Although the rights granted by contract do not require course attendance to use this material for U.S. Government purposes, the SEI recommends attendance to ensure proper understanding.

Carnegie Mellon® and CERT® are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM17-0350

SQUARE for Acquisition





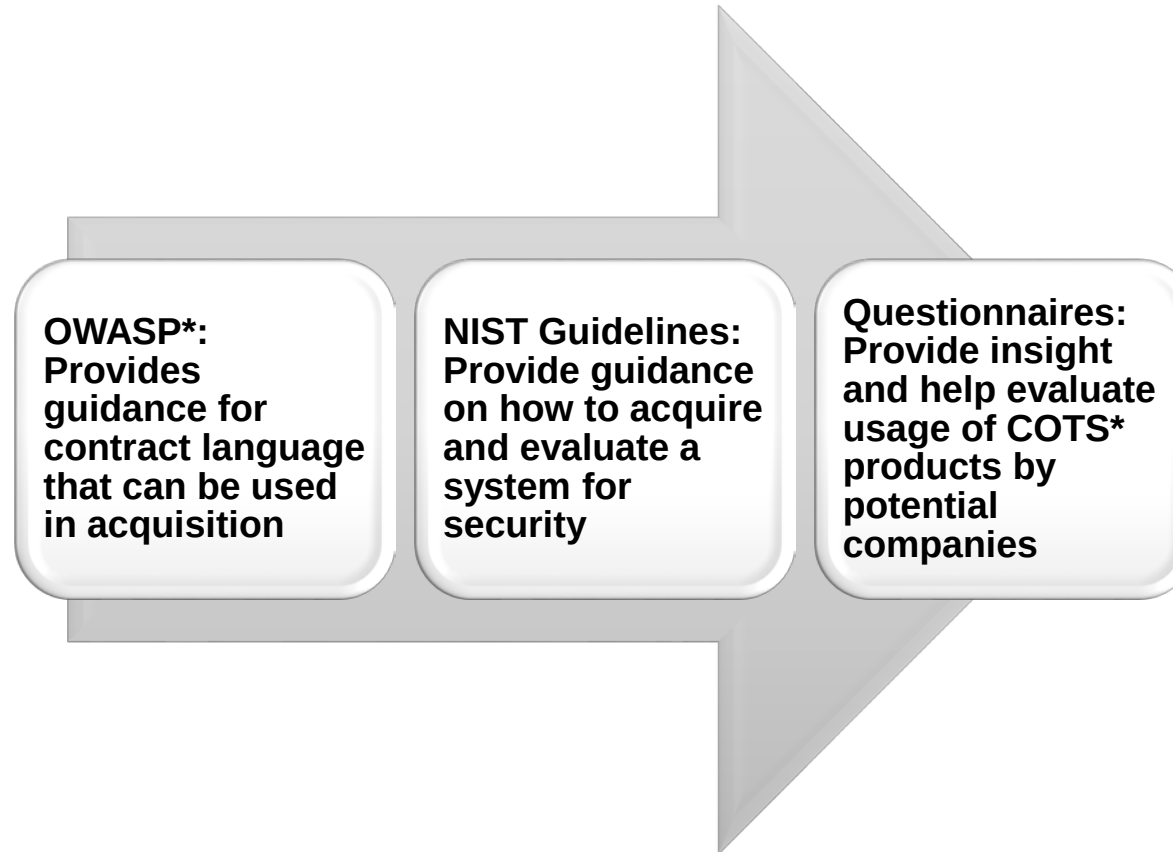
- I. Background
- II. Introduction to A-SQUARE
- III. Three Cases for SQUARE for Acquisition (A-SQUARE)
- IV. Summary and further work
- V. Questions

Background



Background

Current efforts in the field of software acquisition



OWASP – open web application security project
COTS – commercial off-the-shelf

What Is Acquisition?

Acquisition: The process of obtaining a system, software product, or software service. Software products may include commercial-off-the-shelf (COTS) products, modified-off-the-shelf (MOTS) products, open source products, or fully developed products.

The above definition was derived from these references:

Software & Systems Engineering Standards Committee, IEEE Computer Society. ISO/IEC 12207, IEEE Std. 12207-2008, *Systems and Software Engineering - Software Life Cycle Processes*, Second Edition. IEEE Computer Society, 2008.

Software & Systems Engineering Standards Committee, IEEE Computer Society. IEEE Std. 1062, *IEEE Recommended Practice for Software Acquisition*. IEEE Computer Society, 1998.

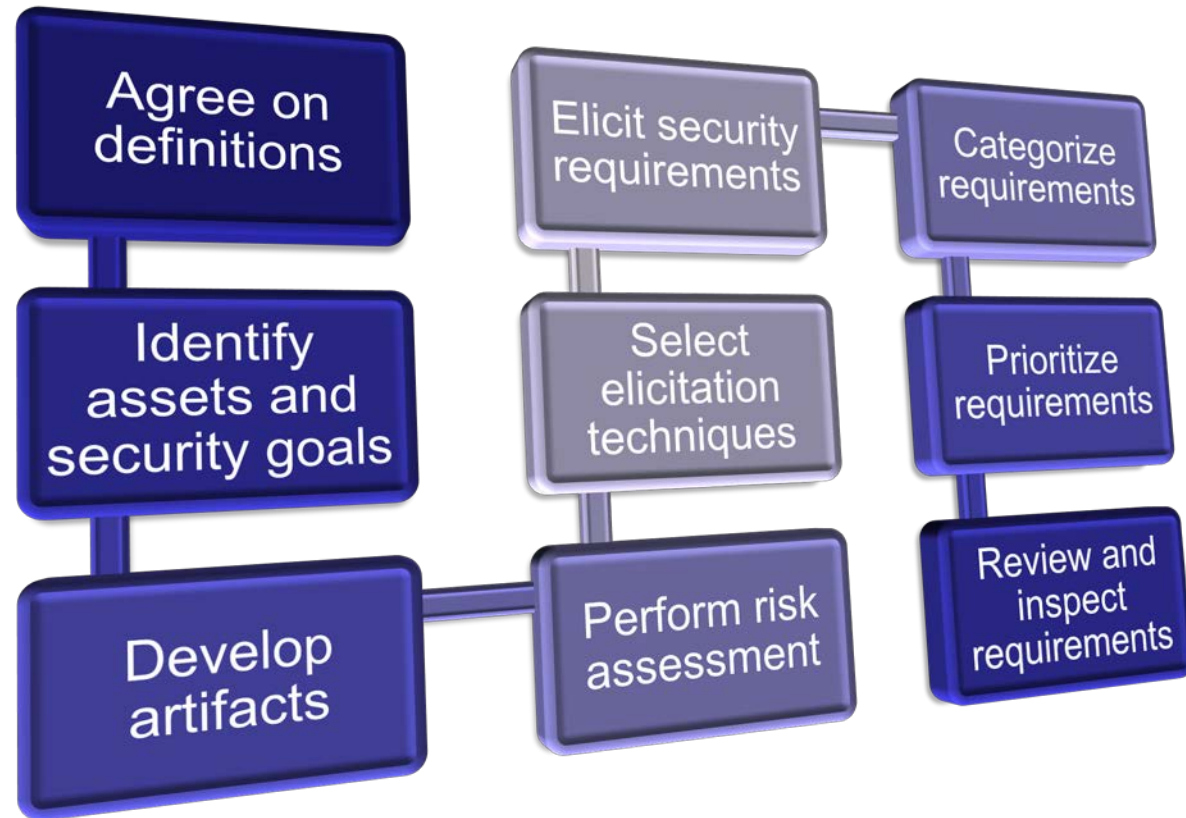
Current problems:

- insufficient security requirements in the product being acquired
- insufficient insight into security requirements when work is outsourced

SQUARE for Acquisition:

- can be easily tailored and modified for various acquisition scenarios
- well-defined process framework with clear roles and responsibilities defined for acquirers and contractors
- A-SQUARE helps address security requirements early in the project

Recall the SQUARE Process



Introduction to A-SQUARE



Case 1 – acquisition organization has typical client role for new software



Acquisition Org.



Contractor **Requirements**

Case 2 – acquisition organization does requirements specification



Acquisition Org. **Requirements**



Contractor

Case 3 – acquisition organization is purchasing COTS software



Acquisition Org.



COTS

Case 1



Nature of software acquisition:

- contractor is responsible for the requirements definition
- contractor should be on board and the contract is awarded
- acquisition organization plays a typical client role

Acquisition Org. & Contractor	Contractor
1. Agree on definitions.	
2. Identify assets and security goals.	
	3. Develop artifacts to support security requirements definition.
	4. Assess risks.
	5. Select elicitation technique(s).
	6. Elicit security requirements.
	7. Categorize requirements.
8. Prioritize requirements.	
9. Inspect requirements.	
10. Review of requirements by acquisition organization	

Case 1: Important Points

The client has the typical stakeholder role in requirements elicitation for the project.

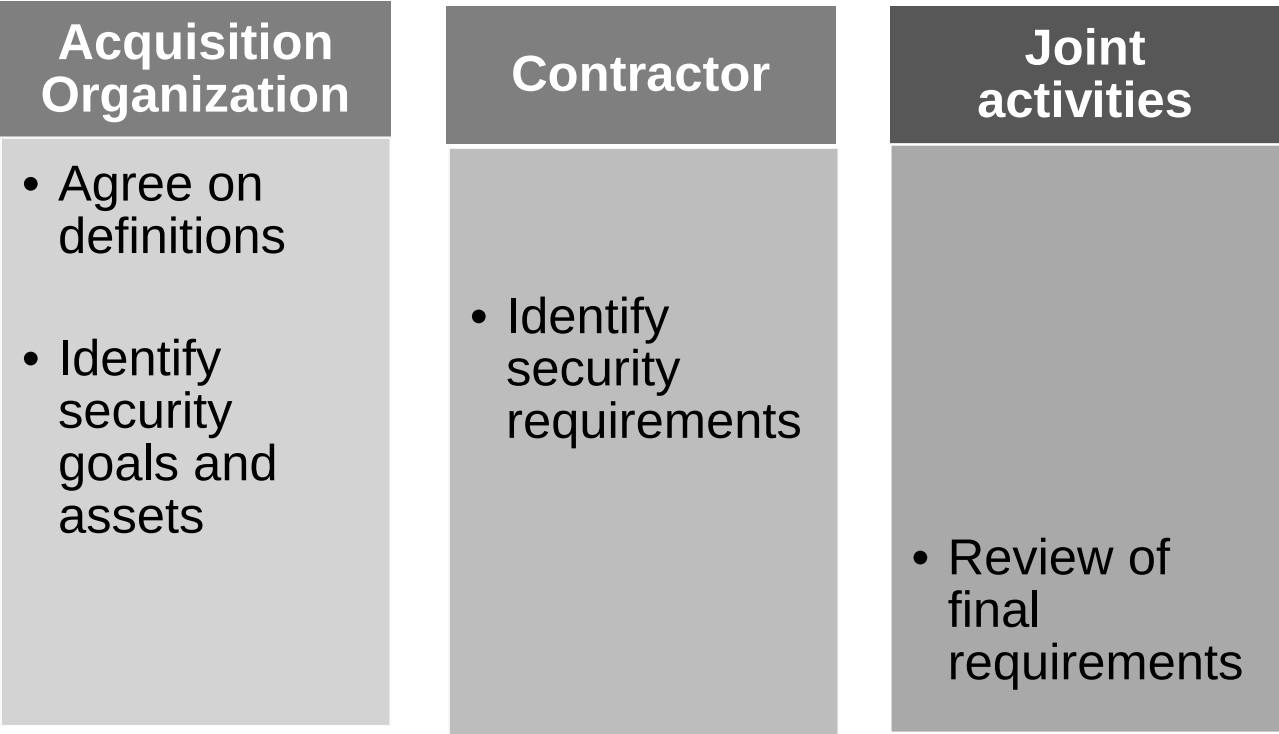
This assumes the project uses SQUARE as the process framework for identifying security requirements.

The additional review step 10 (as shown in workflow) may not be needed if both the parties work together.



Case 1: Compressed Workflow

In the event that the client has no input into the contractor's requirements engineering process, the resultant workflow is compressed as shown below



Case 2



Nature of software acquisition:

- acquisition organization specifies requirements as part of request for proposal (RFP)
- original SQUARE could be used by the acquirer
- requirements specified will have relatively high-level security requirements

Case 2: Important Points



The process workflow is similar to the nine-step SQUARE process.

Level of detail in the requirements definition is crucial.

- Too much detail can constrain the contractor.
- The contractor needs some flexibility in defining the requirements.
- The exit criteria for this process is the final review and approval of the requirements by both parties.

Case 3



Nature of software acquisition

- acquisition of COTS products

What is COTS ?

- computer software products that are ready-made and available for use
- serve as good alternatives for in-house developments

Benefits of using COTS

- applications can be built “out-of-the-box”
- improves overall productivity and reduces company costs

Examples of simple COTS applications acquired by organizations:

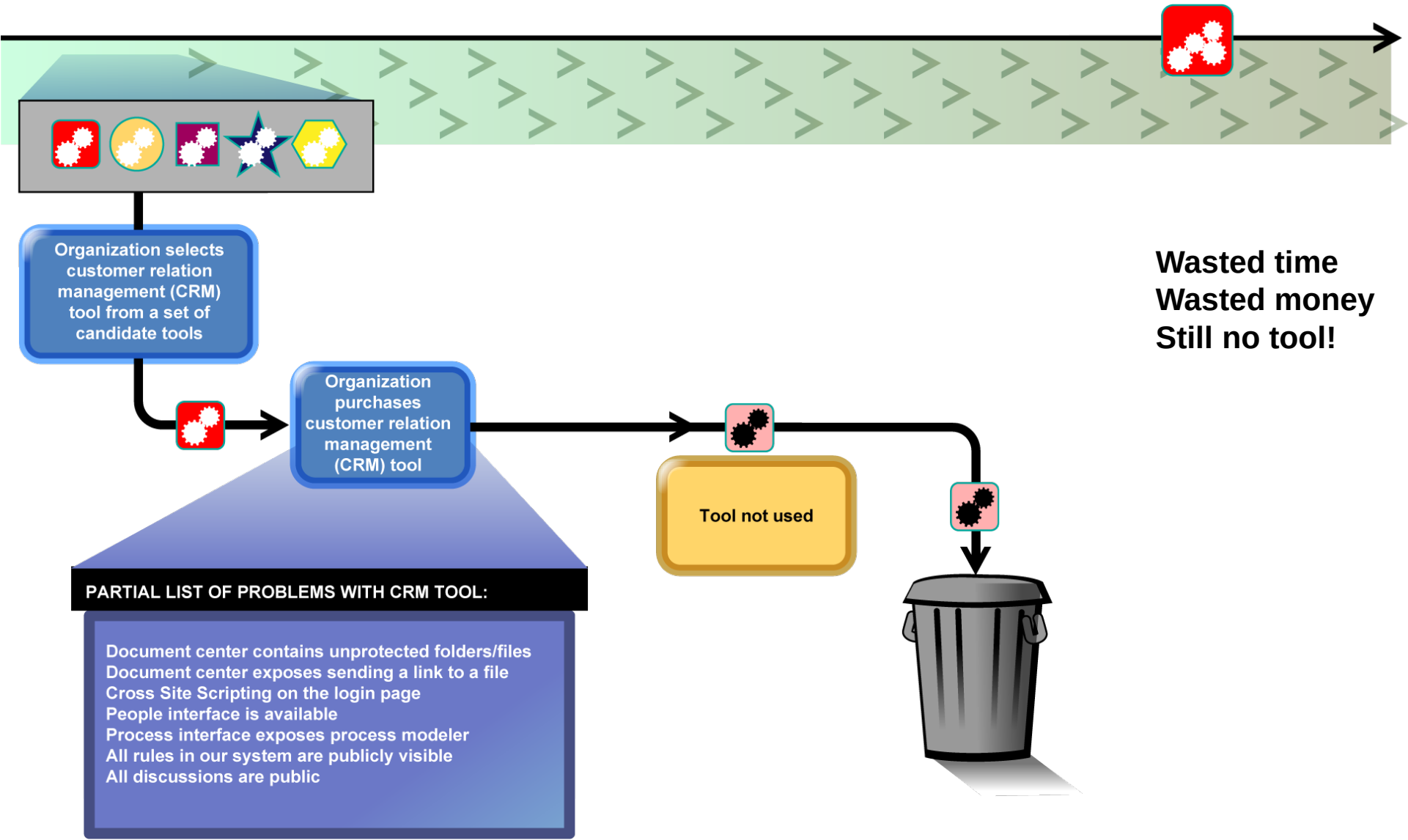
Spreadsheet

Database

Document
management
System

Email

Is There Really a COTS Security Problem?



1. **Agree on definitions.**
2. **Identify security goals.**
3. Identify preliminary security requirements.
4. Review COTS specifications.
5. Finalize security requirements.
6. Perform tradeoff analysis.
7. Write final product specification.

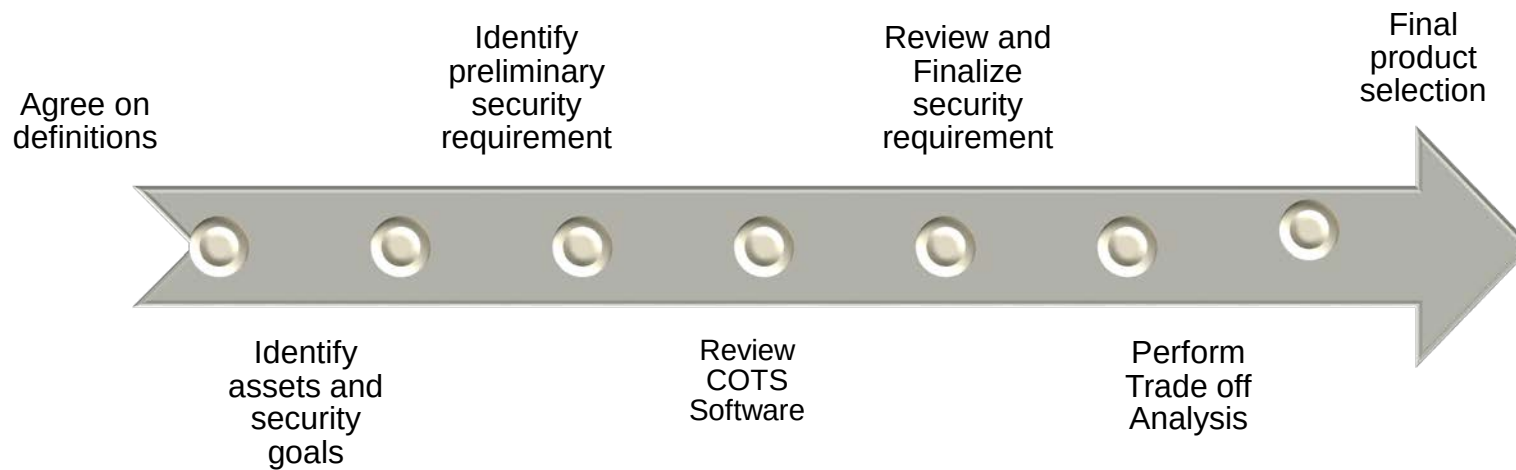
Process for acquiring COTS software

	Step	Input	Techniques	Participants	Output
1	Agree on definitions	Candidate definitions from IEEE and other standards	Structured interviews, focus group	Acquisition organization – stakeholders, security specialists	Agreed-to definitions
2	Identify assets and security goals	Definitions, candidate goals, business drivers, policies and procedures, examples	Facilitated work session, surveys, interviews	Acquisition organization – stakeholders, security specialists	Assets and goals
3	Identify preliminary security requirements	Assets and goals	Work session	Acquisition organization – security specialists	Preliminary security requirements
4	Review COTS software package information and specifications	Assets, goals, preliminary security requirements	Study security features of various packages and document them, in a spreadsheet, for example	Acquisition organization – security specialists, COTS vendors	Spreadsheet of security features of various packages

Process for acquiring COTS software

	Step	Input	Techniques	Participants	Output
5	Finalize security requirements	Preliminary security requirements, features of various packages	Work session – use the spreadsheet to refine and modify the preliminary security requirements to arrive at a final set	Acquisition organization – security specialists	Final security requirements
6	Perform tradeoff analysis	Final security requirements, spreadsheet of security features	Tradeoff analysis of COTS products relative to final security requirements	Acquisition organization – stakeholders, security specialists	Prioritized list of COTS products relative to security requirements
7	Final product selection	Prioritized list of COTS products relative to security, other important COTS product features	Tradeoff analysis	Acquisition organization – stakeholders	Final COTS product selection

Case 3: Important Points



Prioritization

- Security requirements need to be prioritized together with other requirements when acquiring COTS software.

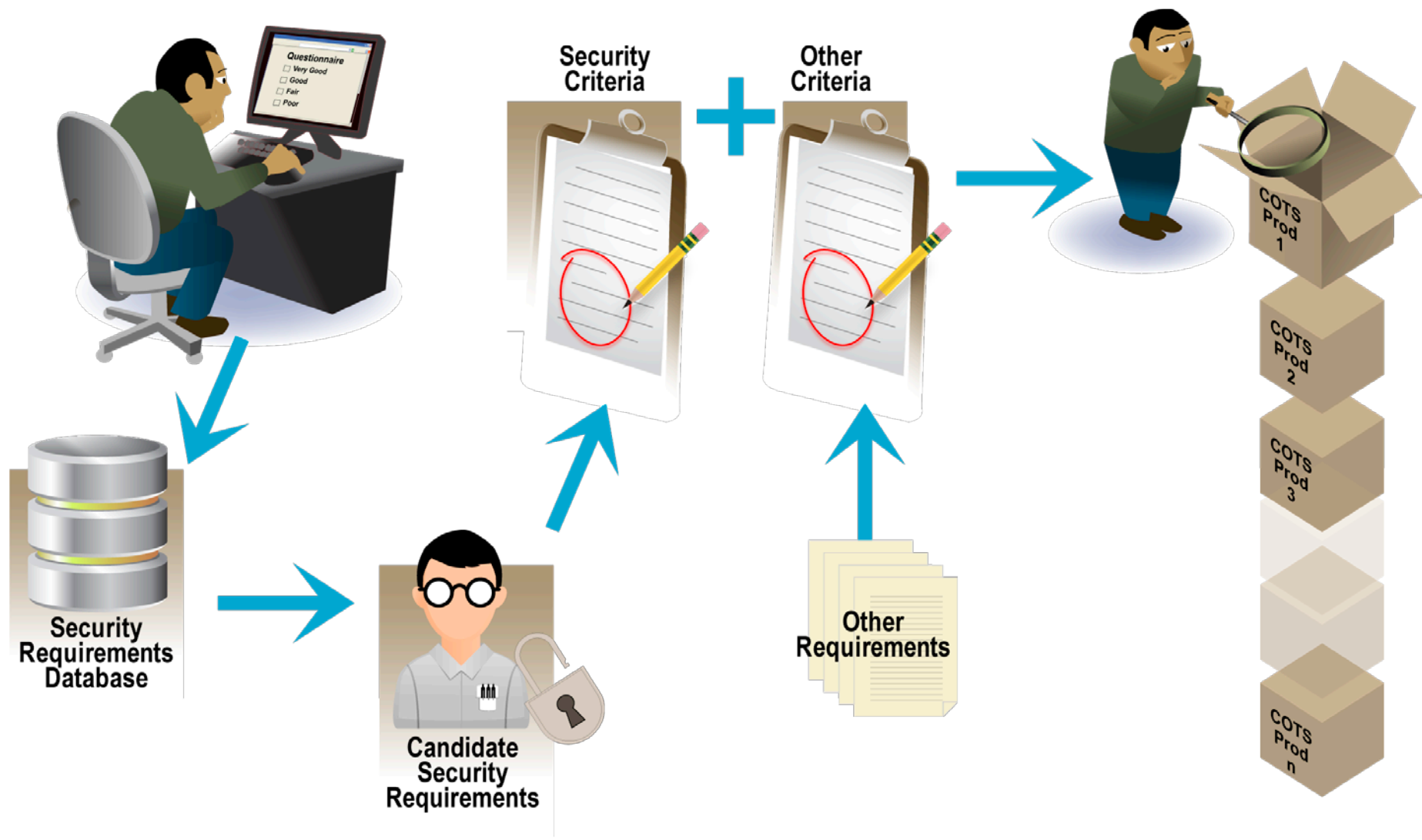
Tradeoff

- Tradeoffs and compromises might have to be made since the software might not meet all the security goals of the organization.

Review

- Reviewing the requirements may help the acquiring organization to identify important security requirements.

Future Vision – A New Scenario



Questions to Think About

Do you focus more on features or quality factors when you acquire a COTS product?

Do you consider security when you buy a COTS product? For yourself? Your organization?

Would you change anything in the future?

Summary



A-SQUARE helps identify security requirements early in the project.

It can reduce the security risk associated with software acquisition.

Prototype A-SQUARE tool developed by CMU MSIT Team

Application of A-SQUARE on projects would help:

- support acquisition organizations
- validate the practices of A-SQUARE
- understand the tailoring needed

Additional Resources

Allen, Julia H., Barnum, Sean, Ellison, Robert J., McGraw, Gary, & Mead, Nancy R. *Software Security Engineering: A Guide for Project Managers*. Addison Wesley Professional, 2008. (Available from Amazon.com.)

U.S. Department of Homeland Security. Build Security In: Requirements Engineering. <https://buildsecurityin.us-cert.gov/daisy/adm-bsi/articles/best-practices/requirements.html>

IDEA Group Publishing. <http://www.idea-group.com>

Mead, Nancy R., Hough, Eric, & Stehney II, Ed. *Security Quality Requirements Engineering* (CMU/SEI-2005-TR-009). Software Engineering Institute, Carnegie Mellon University, 2005. <http://www.sei.cmu.edu/library/abstracts/reports/05tr009.cfm>

Mead, Nancy R. "Identifying Security Requirements Using the Security Quality Requirements Engineering (SQUARE) Method" *Integrating Security and Software Engineering: Advances and Future Visions*. Edited by H. Mouratidis and P. Giorgini. Idea Group, pp. 44-69, 2006 (ISBN: 1-59904-147-2).

SQUARE case study reports:

Gayash, Ashwin, Viswanathan, Venkatesh, & Padmanabhan Deepa. Advisor: Nancy R. Mead. *SQUARE-Lite: Case Study on VADSoft Project* (CMU/SEI-2008-SR-017). Software Engineering Institute, Carnegie Mellon University, 2008.
<http://www.sei.cmu.edu/library/abstracts/reports/08sr017.cfm>

Hough, Eric, Ojoko-Adams, Don, Chung, Lydia, & Hung, Frank. Security Quality Requirements Engineering (SQUARE): Case Study Phase III (CMU/SEI-2006-SR-003). Software Engineering Institute, Carnegie Mellon University, 2006.
<http://www.sei.cmu.edu/library/abstracts/reports/06sr003.cfm>

Panusuwan, Varokas & Batlagundu Prashanth. Faculty Advisor: Nancy Mead. *Privacy Risk Assessment Case Studies in Support of SQUARE* (CMU/SEI-2009-SR-017). Software Engineering Institute, Carnegie Mellon University, 2009.
<http://www.sei.cmu.edu/library/abstracts/reports/09sr017.cfm>

SQUARE Workshop

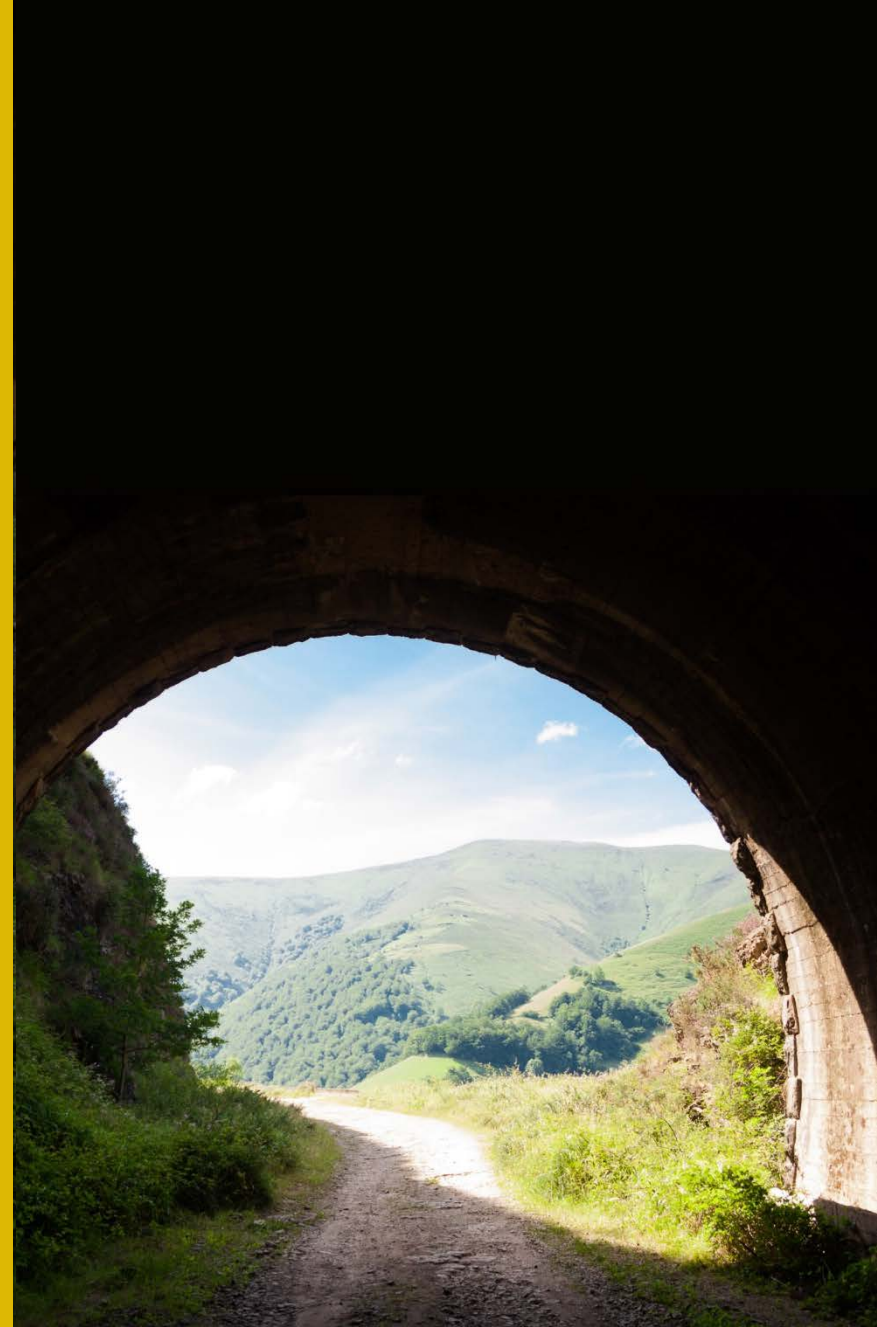
SQUARE Steps 1-2

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213



Software Engineering Institute

Carnegie Mellon University



© 2017 Carnegie Mellon University

[Distribution Statement A] This material has been approved for public release and unlimited distribution.

Copyright 2017 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material is distributed by the Software Engineering Institute (SEI) only to course attendees for their own individual study.

Except for any U.S. government purposes described herein, this material SHALL NOT be reproduced or used in any other manner without requesting formal permission from the Software Engineering Institute at permission@sei.cmu.edu.

Although the rights granted by contract do not require course attendance to use this material for U.S. Government purposes, the SEI recommends attendance to ensure proper understanding.

Carnegie Mellon® and CERT® are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM17-0350

1. **Agree on definitions.**
2. **Identify assets and security goals.**
3. Develop artifacts to support security requirements definition.
4. Assess risks.
5. Select elicitation technique(s).
6. Elicit security requirements.
7. Categorize requirements.
8. Prioritize requirements.
9. Inspect requirements.

SQUARE Step 1



Step 1: Agree on Definitions

Avoiding ambiguity

- Agreement on definitions between stakeholders and engineering team

Example

- Access controls
 - Set of policies that govern which users may be granted access to which resources?
 - or
 - The software elements in the system that actually implement this functionality?

STEP 1 | STEP 2 | STEP 3 | STEP 4 | STEP 5 | STEP 6 | STEP 7 | STEP 8 | STEP 9

Step 1: Agree on Definitions

Initial set of terms

- Build off of a list of standard terms
- Provide a list of suggested definitions to choose from

Example Term with Suggested Definitions

confidentiality	☐ The property that information is not made available or disclosed to unauthorized individuals, entities, or processes. (i.e., to any unauthorized system entity)	[SANS 03a]
	☐ Ensuring that information is available to only those with authorized access.	[ISO 04]
	☐ Restricting access to information via a hierarchy of classes of access.	[JONES 02]
	☐ Other: _____ _____	

STEP 1 | STEP 2 | STEP 3 | STEP 4 | STEP 5 | STEP 6 | STEP 7 | STEP 8 | STEP 9

Step 1: Agree on Definitions

Requirements engineering team responsibilities

- Initial set of terms (Possible definitions)
- Discuss the definitions with the client.
- Means for stakeholders to review and select definitions and to create new terms as needed
- Document and share finalized set of terms and definitions.

STEP 1 | STEP 2 | STEP 3 | STEP 4 | STEP 5 | STEP 6 | STEP 7 | STEP 8 | STEP 9

Step 1: Agree on Definitions

Stakeholder responsibilities

- Select a definition or create a new one for each term.
- Present results to the engineering team in a timely manner.

STEP 1 | STEP 2 | STEP 3 | STEP 4 | STEP 5 | STEP 6 | STEP 7 | STEP 8 | STEP 9

Step 1: Agree on Definitions

Joint responsibilities

- Agree on definitions.

STEP 1 | STEP 2 | STEP 3 | STEP 4 | STEP 5 | STEP 6 | STEP 7 | STEP 8 | STEP 9

Step 1: Agree on Definitions

Exit criteria

- Establish a well-documented, agreed-upon set of definitions.
- Establish a reference document or online database that will be used as a reference throughout the rest of the process.

STEP 1 | STEP 2 | STEP 3 | STEP 4 | STEP 5 | STEP 6 | STEP 7 | STEP 8 | STEP 9

Task: Agree on Definitions

- Exit Criteria:
 - Both teams should agree on and record the same definition

Read and follow the instructions given in your workbook for more detailed information. Use the video resources available to you.

SQUARE Step 2



Step 2: Identify Assets and Security Goals

Stakeholders must formally agree on a set of prioritized security goals for the project.

- A single business goal must be identified.
- Assets to be protected must be identified.
- Approximately five or six security goals must be agreed upon.
- Goals must be prioritized.
- Requirements engineering team must ensure that goals are agreed upon, rather than requirements or recommendations.

STEP 1 | **STEP 2** | STEP 3 | STEP 4 | STEP 5 | STEP 6 | STEP 7 | STEP 8 | STEP 9

Step 2: Identify Assets and Security Goals

A single business goal, assets, and several security goals must be identified.

- “The system allows the client to make informed decisions based on which assets are available.” (Business Goal)
- “The confidentiality, accuracy, and integrity of the system’s data shall be maintained.” (Security Goal)

Balance goals of various stakeholders against each other.

- Conflicting departments
- Conflicting security goals

STEP 1 | **STEP 2** | STEP 3 | STEP 4 | STEP 5 | STEP 6 | STEP 7 | STEP 8 | STEP 9

Step 2: Identify Assets and Security Goals

Prioritize security goals

- Goals must be prioritized in order of importance.
- It's the stakeholders' responsibility to discuss and prioritize.
- The requirements engineering team provides expertise and guidance.

STEP 1 | **STEP 2** | STEP 3 | STEP 4 | STEP 5 | STEP 6 | STEP 7 | STEP 8 | STEP 9

Step 2: Identify Assets and Security Goals

Requirements engineering team responsibilities

- Direct brainstorming session with stakeholders.
- Review and provide feedback on stakeholders' business goal, assets, and security goals.
- Document and share final business goal, assets, and corresponding security goals.

STEP 1 | **STEP 2** | STEP 3 | STEP 4 | STEP 5 | STEP 6 | STEP 7 | STEP 8 | STEP 9

Step 2: Identify Assets and Security Goals

Stakeholder responsibilities

- Identify a single business goal.
- Identify assets to be protected.
- Create security goals that support the business goal.
- Prioritize security goals.
- Provide goals to requirements engineering team for review.
- Maintain current set of goals if they change over time.

STEP 1 | **STEP 2** | STEP 3 | STEP 4 | STEP 5 | STEP 6 | STEP 7 | STEP 8 | STEP 9

Step 2: Identify Assets and Security Goals

Exit criteria

- Documented Assets
- Single business goal
- Prioritized security goals

Results of Step 2 are very important!

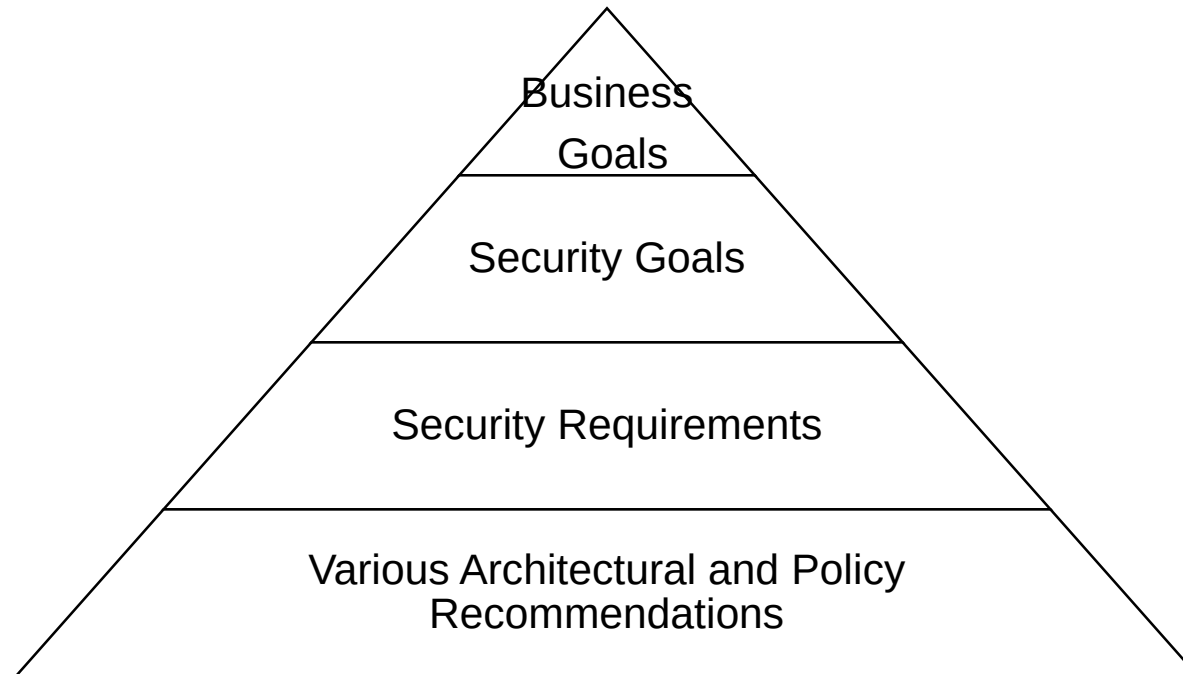
- Without overall security goals for the project, it is impossible to identify the priority and relevance of any security requirements that are generated.
- The establishment of security goals establishes the scope of the rest of the SQUARE process.

STEP 1 | **STEP 2** | STEP 3 | STEP 4 | STEP 5 | STEP 6 | STEP 7 | STEP 8 | STEP 9

Step 2: Identify Assets and Security Goals

Possible sources for guidance:

- *Core Security Requirements Artefacts* [Moffett 04]
- *Mapping Mission-Level Availability Requirements to System Architectures and Policy Abstractions* [Watro 01]



Sample Hierarchy of Goals, Requirements, and Recommendations

STEP 1 | **STEP 2** | STEP 3 | STEP 4 | STEP 5 | STEP 6 | STEP 7 | STEP 8 | STEP 9

Task: Identify, agree on, and prioritize security goals.

Exit Criteria:

- Both teams should identify, agree on, and prioritize the security goals.

Assumption: Both teams have already agreed on a common business goal.

Read and follow the instructions given in your workbook for more detailed information. Use the videos that are available to you.

SQUARE Workshop

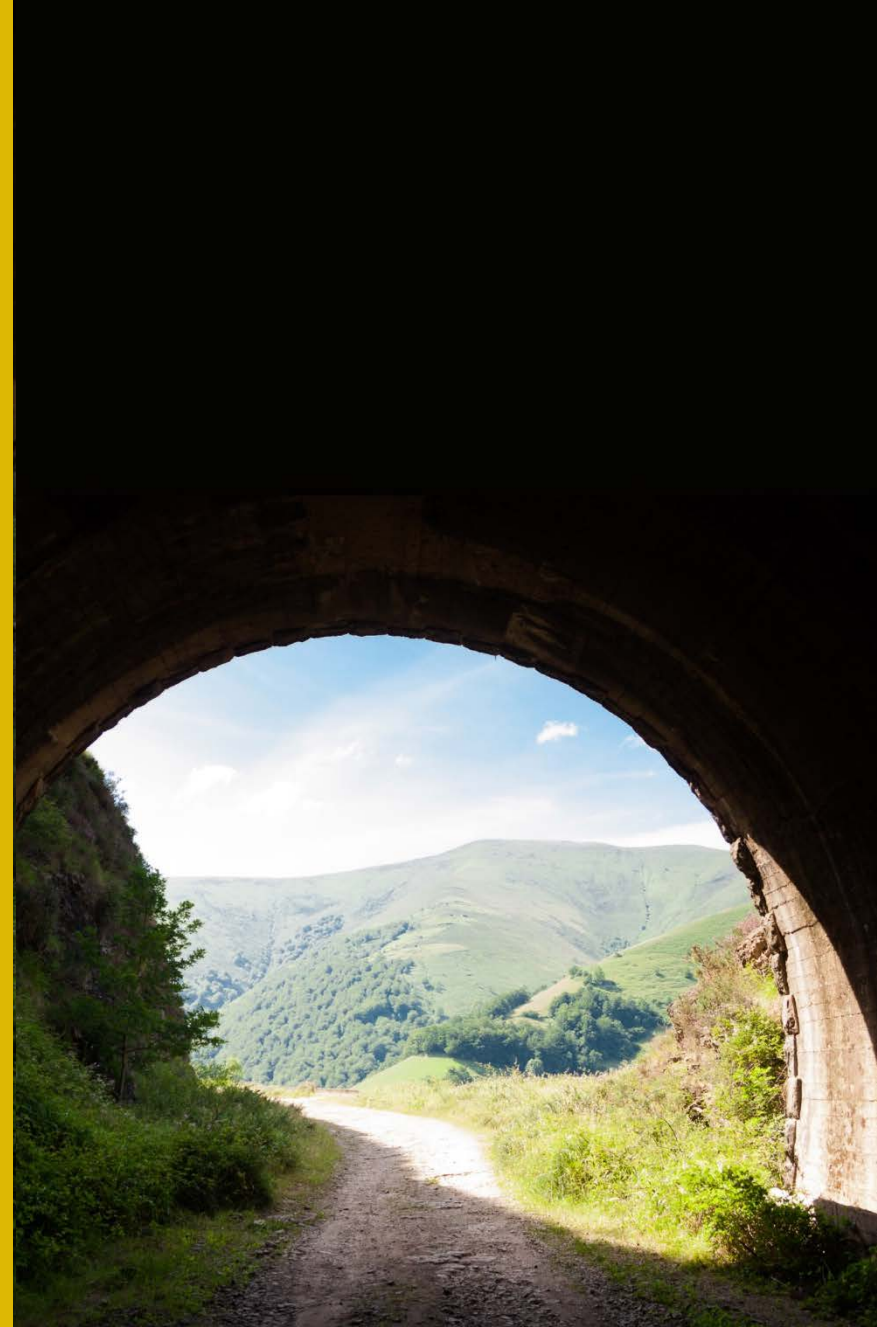
SQUARE Steps 3-4

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213



Software Engineering Institute

Carnegie Mellon University



© 2017 Carnegie Mellon University

[Distribution Statement A] This material has been approved for public release and unlimited distribution.

Copyright 2017 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material is distributed by the Software Engineering Institute (SEI) only to course attendees for their own individual study.

Except for any U.S. government purposes described herein, this material SHALL NOT be reproduced or used in any other manner without requesting formal permission from the Software Engineering Institute at permission@sei.cmu.edu.

Although the rights granted by contract do not require course attendance to use this material for U.S. Government purposes, the SEI recommends attendance to ensure proper understanding.

Carnegie Mellon® and CERT® are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

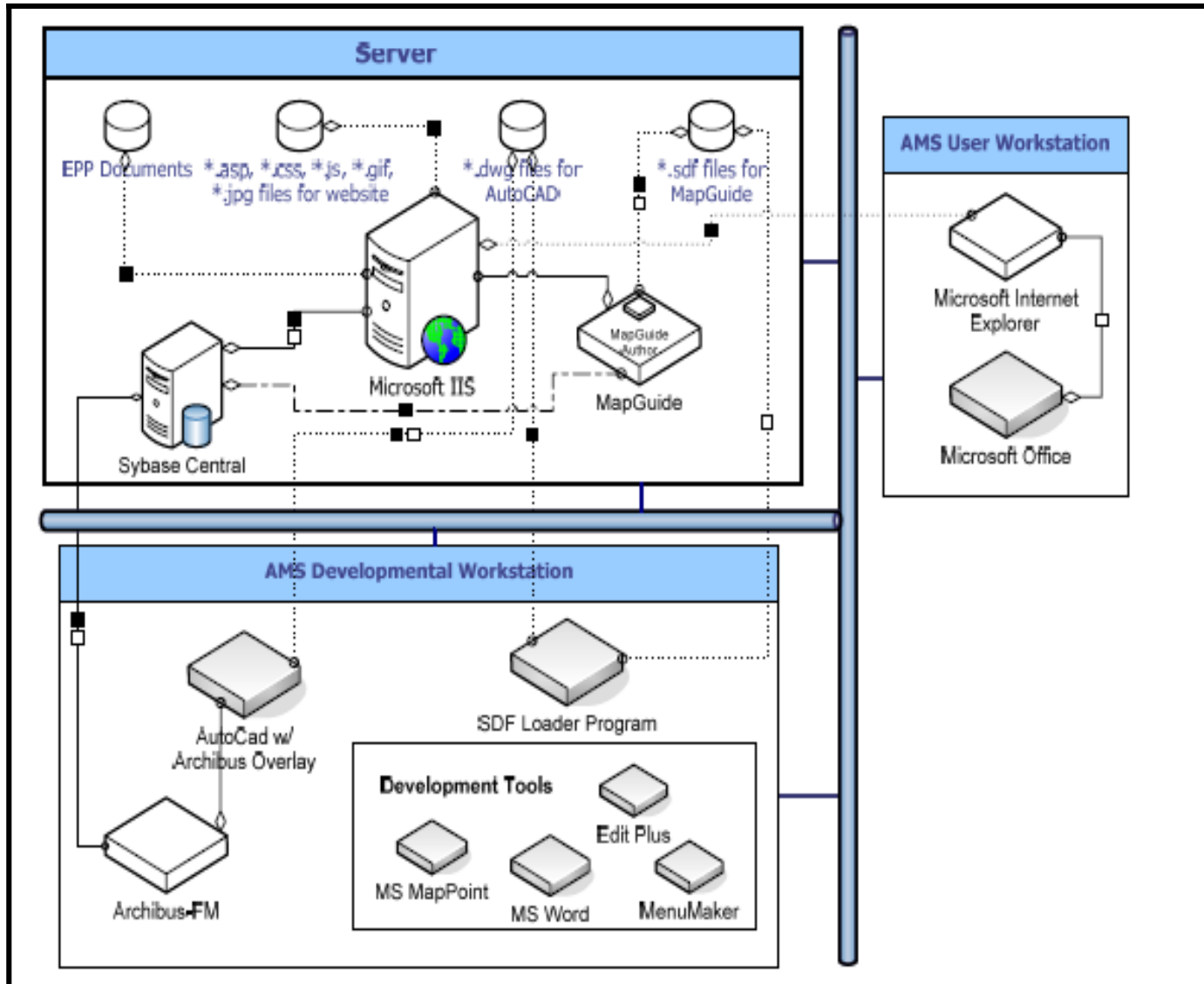
DM17-0350

1. Agree on definitions.
2. Identify assets and security goals.
3. **Develop artifacts to support security requirements definition.**
4. **Assess risks.**
5. Select elicitation technique(s).
6. Elicit security requirements.
7. Categorize requirements.
8. Prioritize requirements.
9. Inspect requirements.

SQUARE Step 3



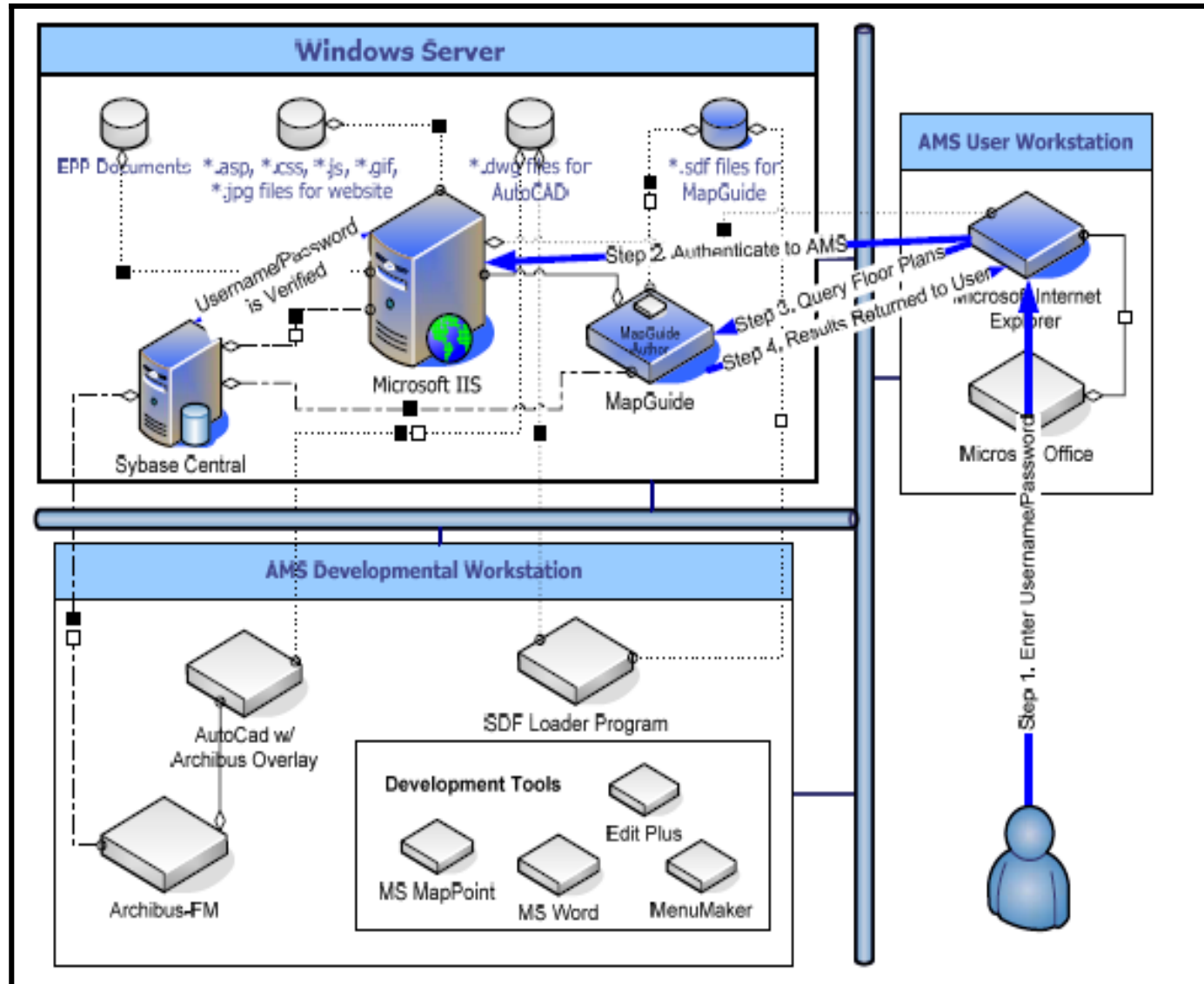
Step 3: Develop Artifacts



Types of artifacts to collect

- System architecture diagrams (should exist for the project)

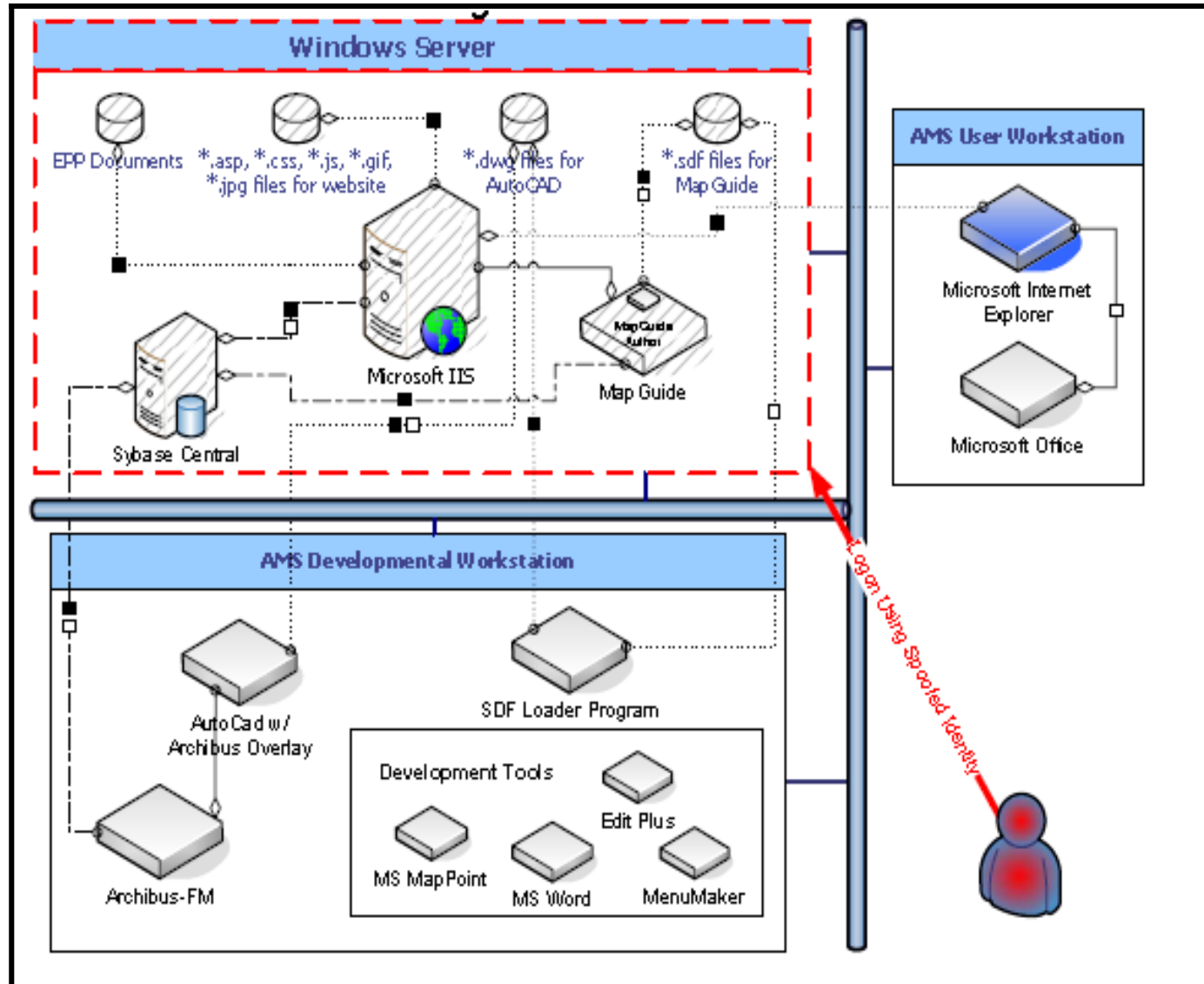
Step 3: Develop Artifacts



Types of artifacts to collect

- Use case scenarios/ diagrams (should exist for the project)

Step 3: Develop Artifacts



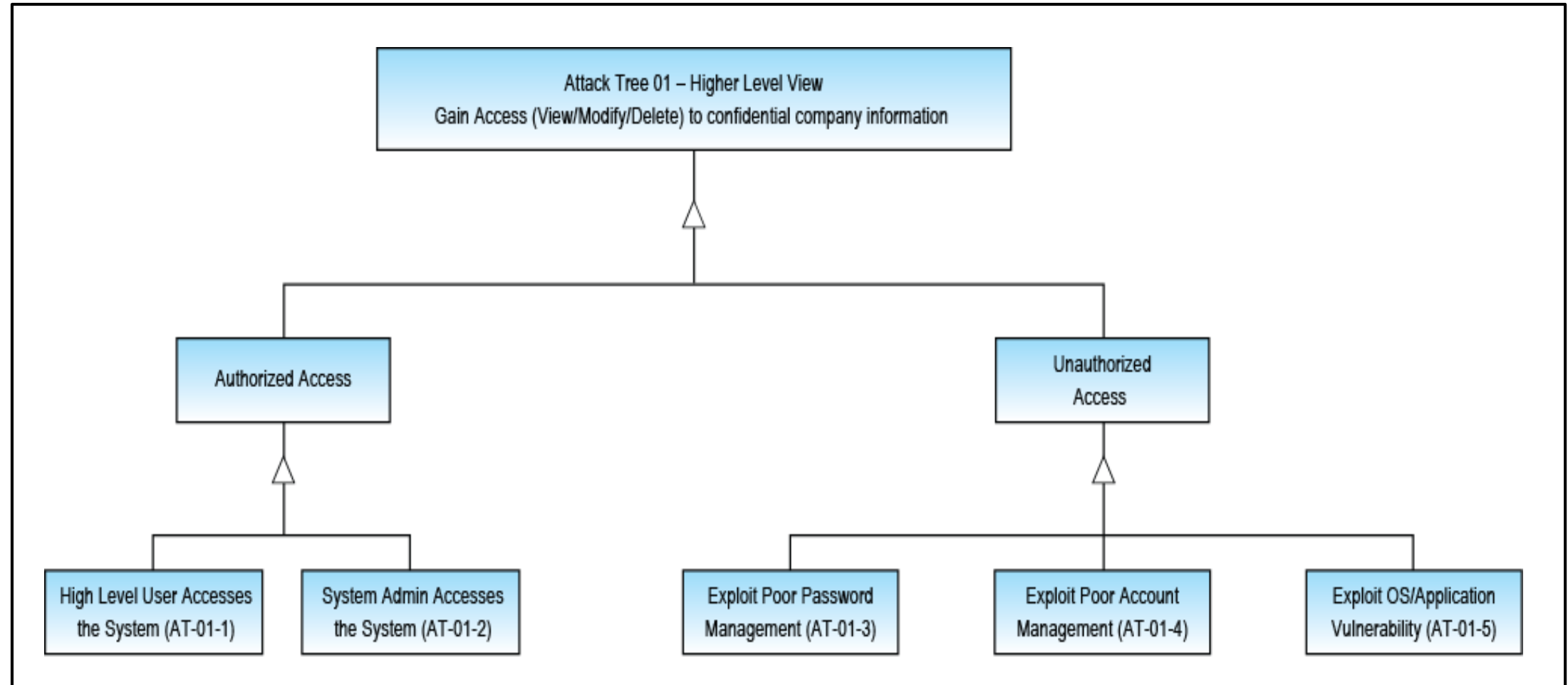
Types of artifacts to collect

- Misuse case scenarios/ diagrams (exemplar misuse cases)

Step 3: Develop Artifacts

Types of artifacts to collect

- Attack trees



Step 3: Develop Artifacts

Enlist the aid of knowledgeable engineers from the organization.

Encourage stakeholders to preserve artifacts for future use.

Some artifacts should already exist.

Step 3: Develop Artifacts

Requirements engineering team responsibilities

- Work with stakeholders and client organization to identify and collect as many artifacts as possible.

Step 3: Develop Artifacts

Stakeholder responsibilities

- Generate or collect system artifacts.
- Present artifacts to the requirements engineering team.

Step 3: Develop Artifacts

Joint responsibilities

- Generate and review some of the artifacts.
- Verify the accuracy and completeness of all artifacts.

Step 3: Develop Artifacts

Exit criteria

- A set of artifacts for the system generated by a collaborative effort between the requirements engineering team and the stakeholders

Task: Generate artifacts

Exit Criteria:

- Both teams should have discussed, documented, and verified all the artifacts.

For the purpose of this workshop, two use cases and four misuse cases are provided.

Read and follow the instructions given in your workbook for more detailed information. Use the videos that are available to you.

SQUARE Step 4



Step 4: Perform Risk Assessment

The stakeholders and the requirements engineering team must identify

- Vulnerabilities and threats that face the system
- Likelihood that threats will materialize as real attacks
- Potential consequences of an attack

By doing so, they can

- Prevent implementing unnecessary countermeasures
- Aid in eliciting and prioritizing security requirements

Step 4: Perform Risk Assessment

Risk assessment methods

- General Accounting Office model [GAO 99]
- National Institute of Standards and Technology model [Stoneburner 02]
- Security Engineering Risk Analysis (SERA)
- NSA's INFOSEC Assessment Technology [NSA 04]
- Shawn Butler's Security Attribute Evaluation Method [Butler 02]
- CMU's "V-RATE" method [Lipson 01]
- Yacov Haimes's RFRM model (recommended) [Haimes 04]
- CMU's Survivable Systems Analysis method [Mead 02]
- Martin Feather's DDP model [Cornford 04]

Risk assessment example (NIST model)

- Six relevant steps
 - Threat identification
 - Vulnerability identification
 - Control analysis
 - Likelihood determination
 - Impact analysis
 - Risk determination

Risk assessment example (NIST model)

	NIST
Tier 1	- Insider or terrorist alters or disables key architecture components. - Insider or terrorist discloses proprietary information. - Terrorist gains unauthorized use of system resources.
Tier 2	- Insider installs malicious software (viruses, Trojans, key loggers, etc.). - Insider or natural forces physically destroy system components. - Insider steals system components.
Tier 3	- Terrorist steals system components. - Terrorist installs malicious software (viruses, Trojan's, key loggers, etc.). - Terrorists physically destroys system components. - Insider or terrorist alters or corrupts data.

Step 4: Security Engineering Risk Analysis (SERA)

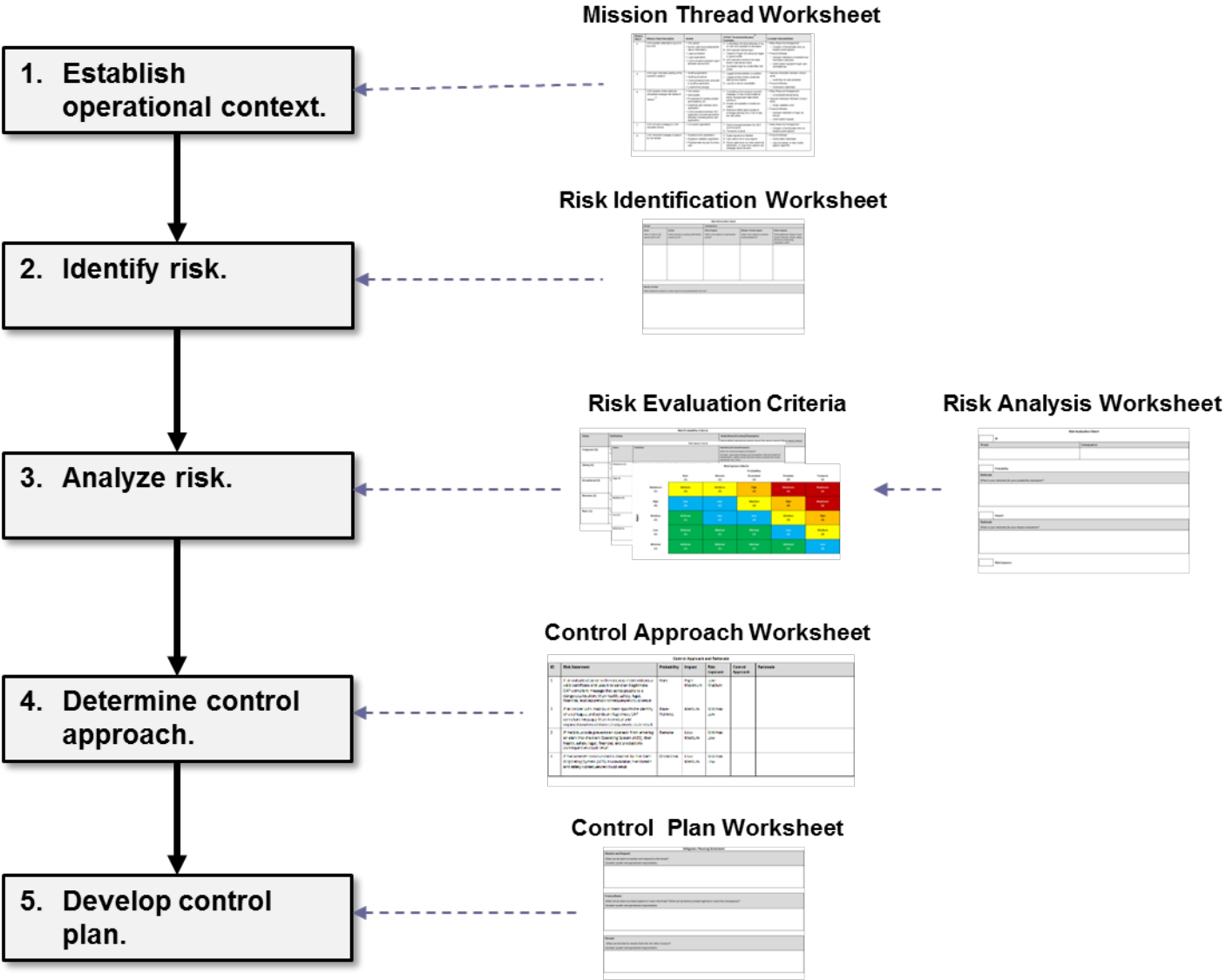
Assesses operational security risks early in the software lifecycle.

- Requirements
- Architecture
- Design

Employs structured, systematic risk analysis to handle the complex nature of security risk.

Goal:

- To identify and address design vulnerabilities early in the lifecycle (i.e., build security in)



Step 4: Perform Risk Assessment

Classification

- Each risk classified according to likelihood
- Classifications help identify a quantifiable, verifiable response

Step 4: Perform Risk Assessment

Requirements engineering team responsibilities

- Facilitate the completion of a structured risk assessment
- Review the results of the risk assessment and share them with stakeholders

Step 4: Perform Risk Assessment

Stakeholder responsibilities

- Present material to the RE team
- Discuss the different artifacts

Step 4: Perform Risk Assessment

Exit criteria

- All vulnerabilities and threats are identified according to likelihood.
- Results are well documented and shared with the stakeholders.

Task: Identify risks, evaluate, and prioritize

Exit Criteria:

- Risks should be identified, evaluated, and prioritized.

For the purpose of this workshop, a risk assessment method will be given to you. Risks should be evaluated based on severity and probability as well as the cost and likelihood of success of the mitigation.

Read and follow the instructions given in your workbook for more detailed information. Use the videos that are available to you.

Alberts, Christopher & Dorofee, Audrey. *Mission Risk Diagnostic (MRD) Method Description* (CMU/SEI-2012-TN-005). Software Engineering Institute, Carnegie Mellon University, 2012.

<http://www.sei.cmu.edu/reports/12tn005.pdf>

Alberts, Christopher; Allen, Julia; & Stoddard, Robert. *Risk-Based Measurement and Analysis: Application to Software Security* (CMU/SEI-2012-TN-004), Software Engineering Institute, Carnegie Mellon University, 2012.

<http://www.sei.cmu.edu/reports/12tn004.pdf>

Alberts, Christopher & Dorofee, Audrey. *A Framework for Categorizing Key Drivers of Risk* (CMU/SEI-2009-TR-007). Software Engineering Institute, Carnegie Mellon University, 2009.

<http://www.sei.cmu.edu/library/abstracts/reports/09tr007.cfm>

SQUARE Workshop

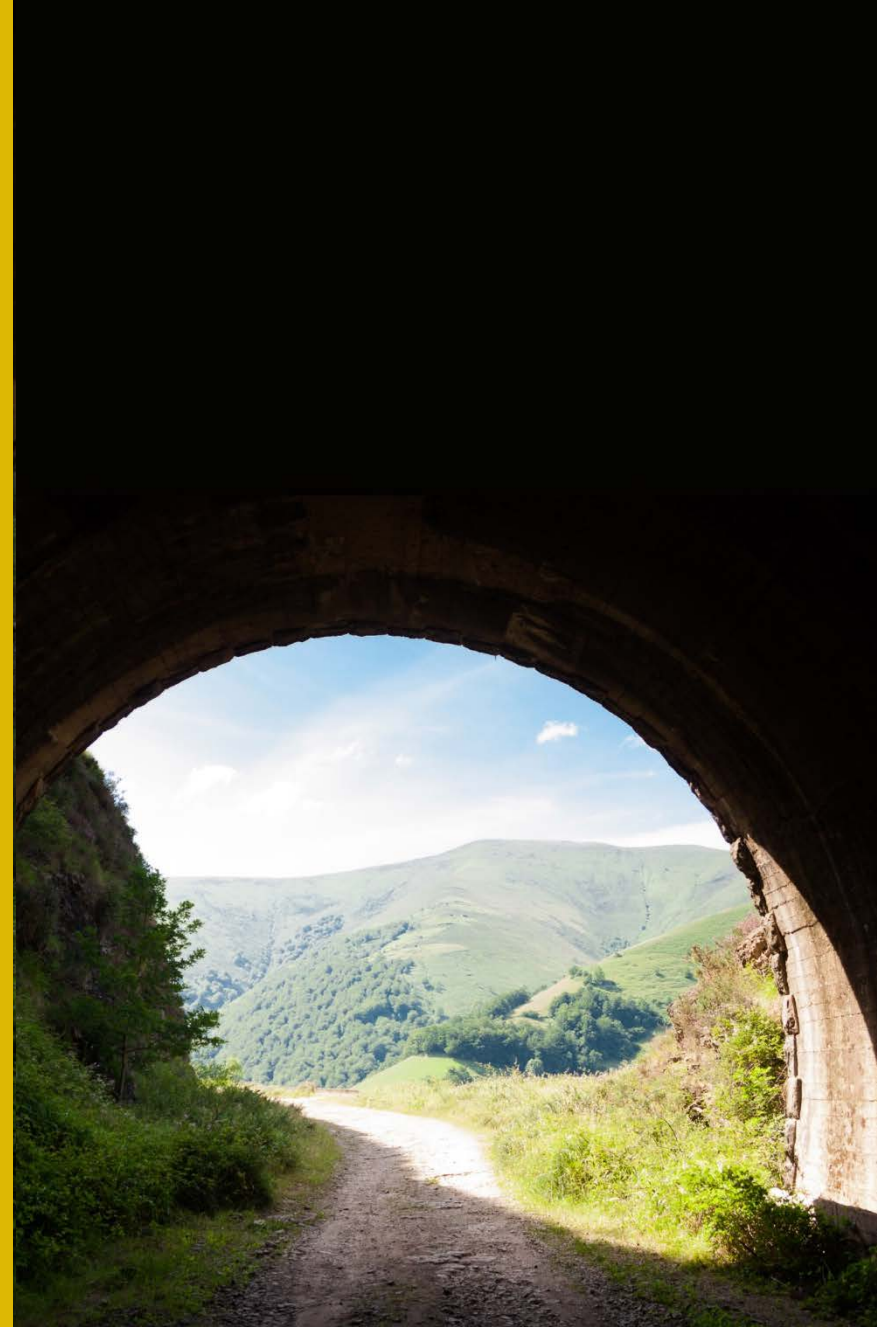
Threat Modeling with STRIDE

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213



Software Engineering Institute

Carnegie Mellon University



© 2017 Carnegie Mellon University

[Distribution Statement A] This material has been approved for public release and unlimited distribution.

SQUARE Workshop

Threat Modeling
(Based on Threat Modeling Material from David Ladd at
Microsoft)



Copyright 2017 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material is distributed by the Software Engineering Institute (SEI) only to course attendees for their own individual study.

Except for any U.S. government purposes described herein, this material SHALL NOT be reproduced or used in any other manner without requesting formal permission from the Software Engineering Institute at permission@sei.cmu.edu.

Although the rights granted by contract do not require course attendance to use this material for U.S. Government purposes, the SEI recommends attendance to ensure proper understanding.

Carnegie Mellon® and CERT® are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM17-0350

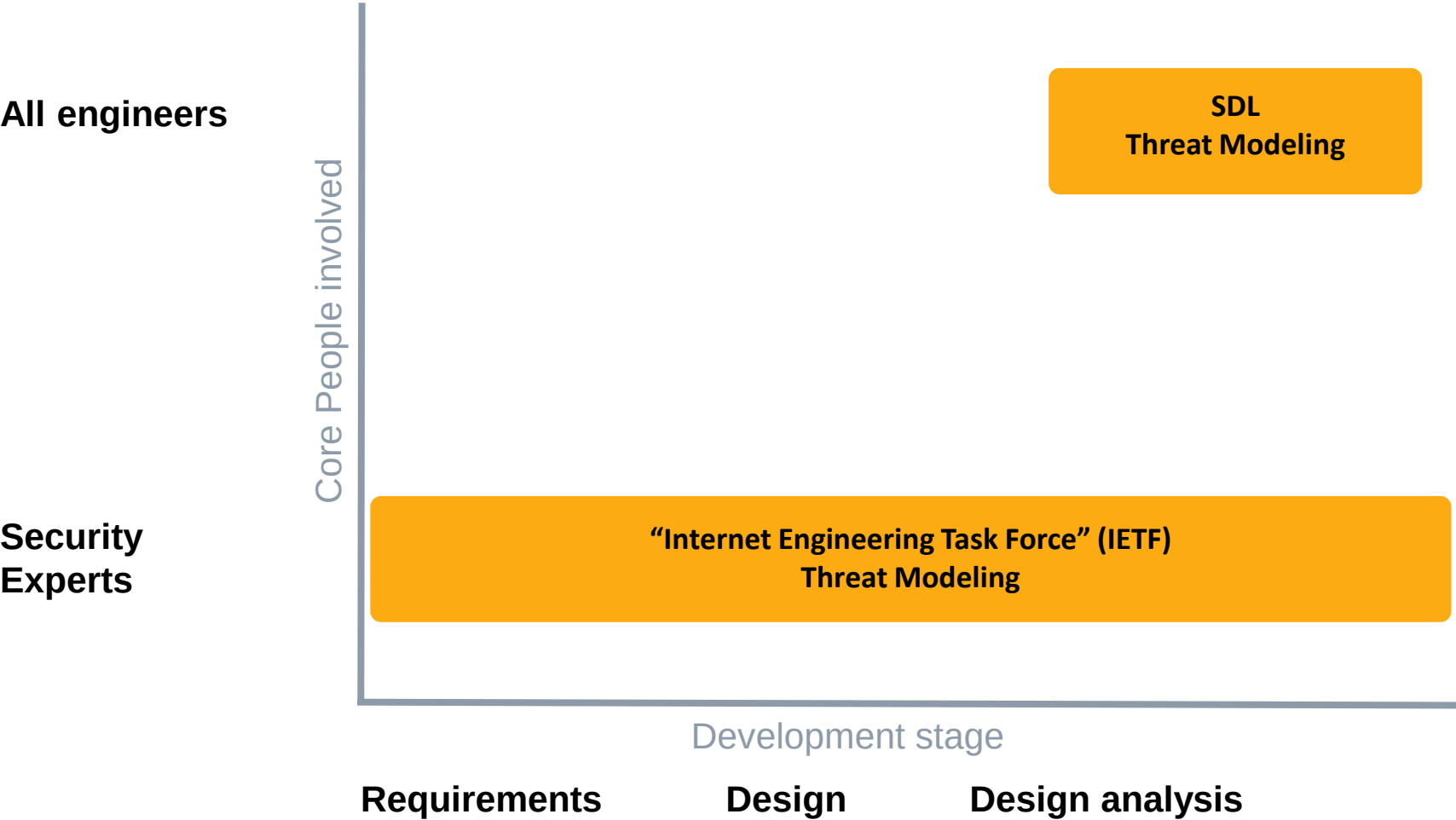


- Introduction and Goals
- How to Threat Model
- The STRIDE Per Element Approach to Threat Modeling
- Diagram Validation Rules of Thumb
- Exercise

SQUARE Workshop

Introduction and Goals





Who?

- Attackers will do it.
- You can also do it.

What?

- A repeatable process to find and address all threats to your product

When?

- The earlier you start, the more time to plan and fix.
- Worst case is when you're trying to ship/deliver:
Find problems, make scope and schedule choices, revisit those features soon.

Why?

- Find problems when there's time to fix them.
- Deliver more secure products.
- At Microsoft, it's a Security Development Lifecycle (SDL) requirement.

How?

Building a threat model (at Microsoft)

- Program Manager (PM) owns overall process.
- Testers
 - Identify threats in analyze phase.
 - Use threat models to drive test plans.
- Developers create diagrams.

Customers for threat models

- Your team
- Other feature teams, other product teams
- Customers, via user education
- Quality assurance staff external to your team, such as penetration testers/ethical hackers

You'll need to decide what fits.

Consider, document, and discuss security in a structured way.

Threat model and document

- The product as a whole
- The security-relevant features
- The attack surfaces

Assurance that threat modeling has been done well

Produce software that's secure by design.

- Improve designs the same way code has improved by good practices.

Because attackers think differently

- Creator blindness/new perspective

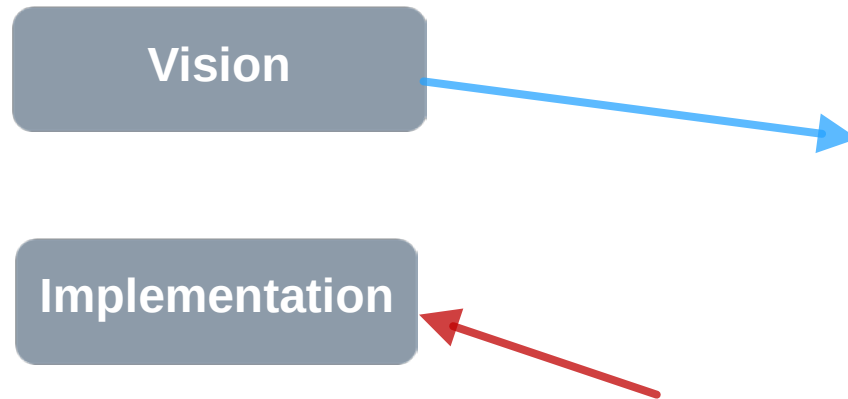
Allow you to predictably and effectively find security problems early in the process

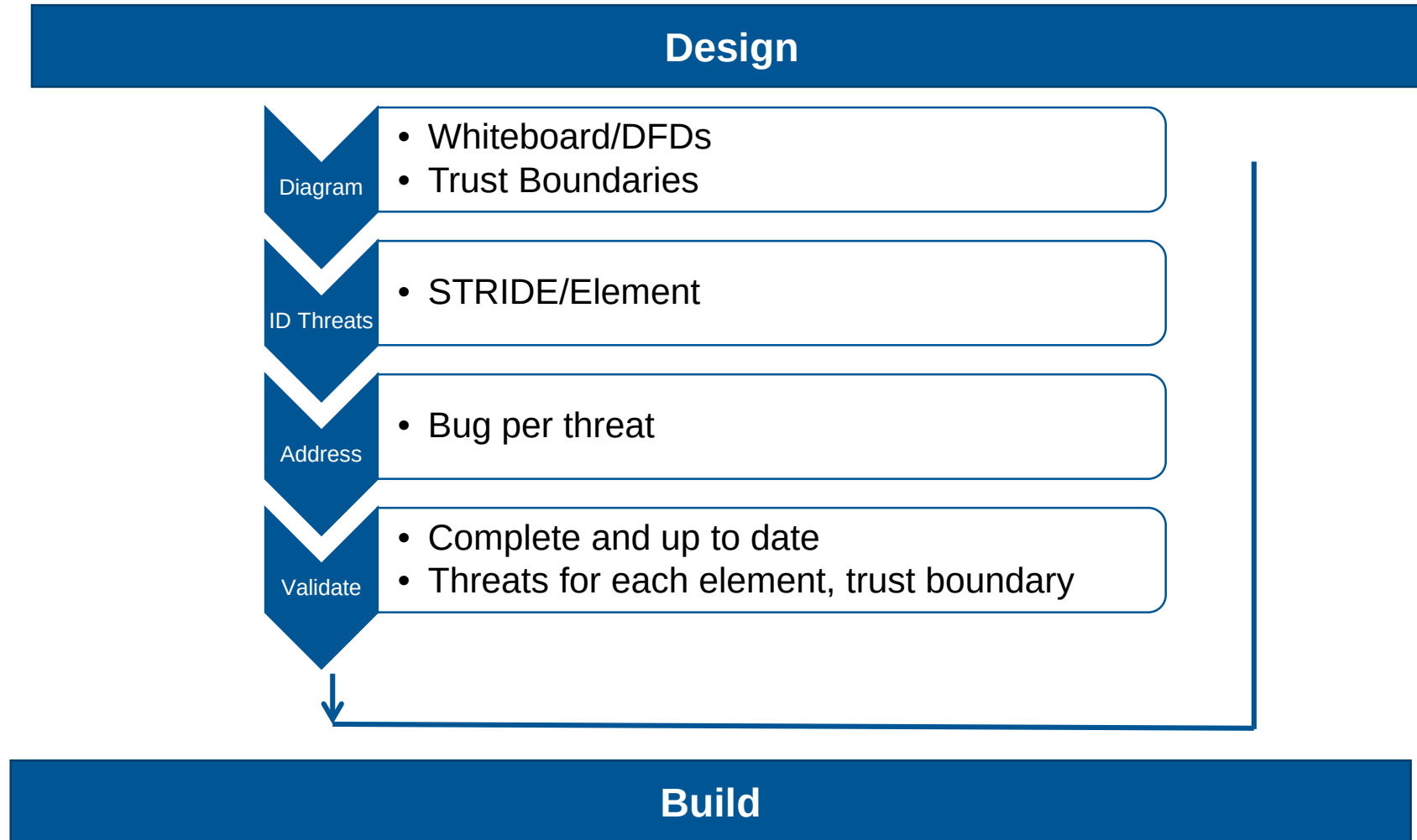
SQUARE Workshop

How Microsoft Threat Models

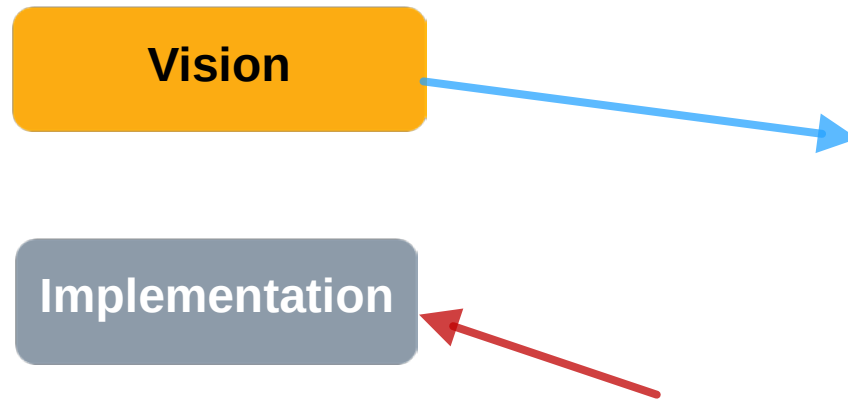


The Process in a Nutshell





The Process in a Nutshell



Scenarios

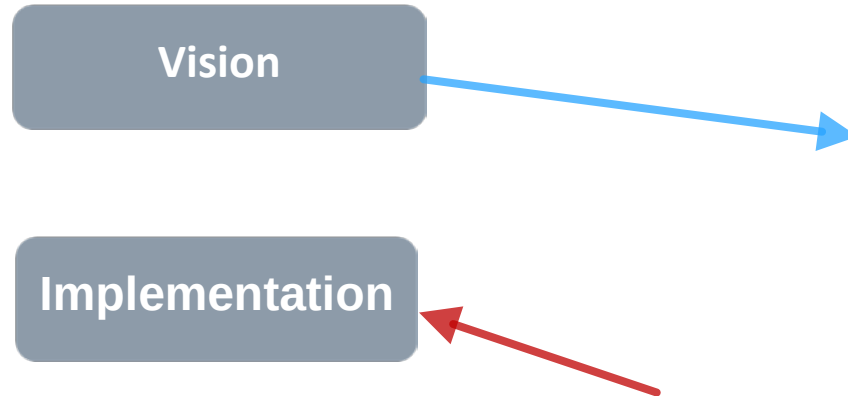
- Where do you expect the product to be used?
- Products like Excel are different from Windows.

Use cases/personas

Add security to scenarios, use cases.

- Think about what you are telling stakeholders about the product's security....

The Process: Diagramming



Go to the whiteboard.

Start with an overview which has:

- A few external interactors
- One or two processes
- One or two data stores (maybe)
- Data flows to connect them

Check your work.

- Can you tell a story without edits?
- Does it match reality?

Use DFDs (Data Flow Diagrams)

- Include processes, data stores, data flows.
- Include trust boundaries.
- Diagrams for each scenario may be helpful.

Update diagrams as product changes.

Enumerate assumptions, dependencies

Number everything (if manual).

Diagram Elements: Examples

External Entity

- People
- Other systems
- YourOrg.com

Process

- DLLs
- EXEs
- COM object
- Components
- Services
- Web Services
- Assemblies

Data Flow

- Function call
- Network traffic
- Remote Procedure Call (RPC)

Data Store

- Database
- File
- Registry
- Shared Memory
- Queue / Stack

Trust Boundary

- Process boundary
- File system

Add trust boundaries that intersect data flows.

Points/surfaces where an attacker can interject

- Machine boundaries, privilege boundaries, integrity boundaries are examples of trust boundaries

Processes talking across a network always have a trust boundary.

- They may create a secure channel, but they're still distinct entities.
- Encrypting network traffic is an 'instinctive' mitigation.
 - But doesn't address tampering or spoofing

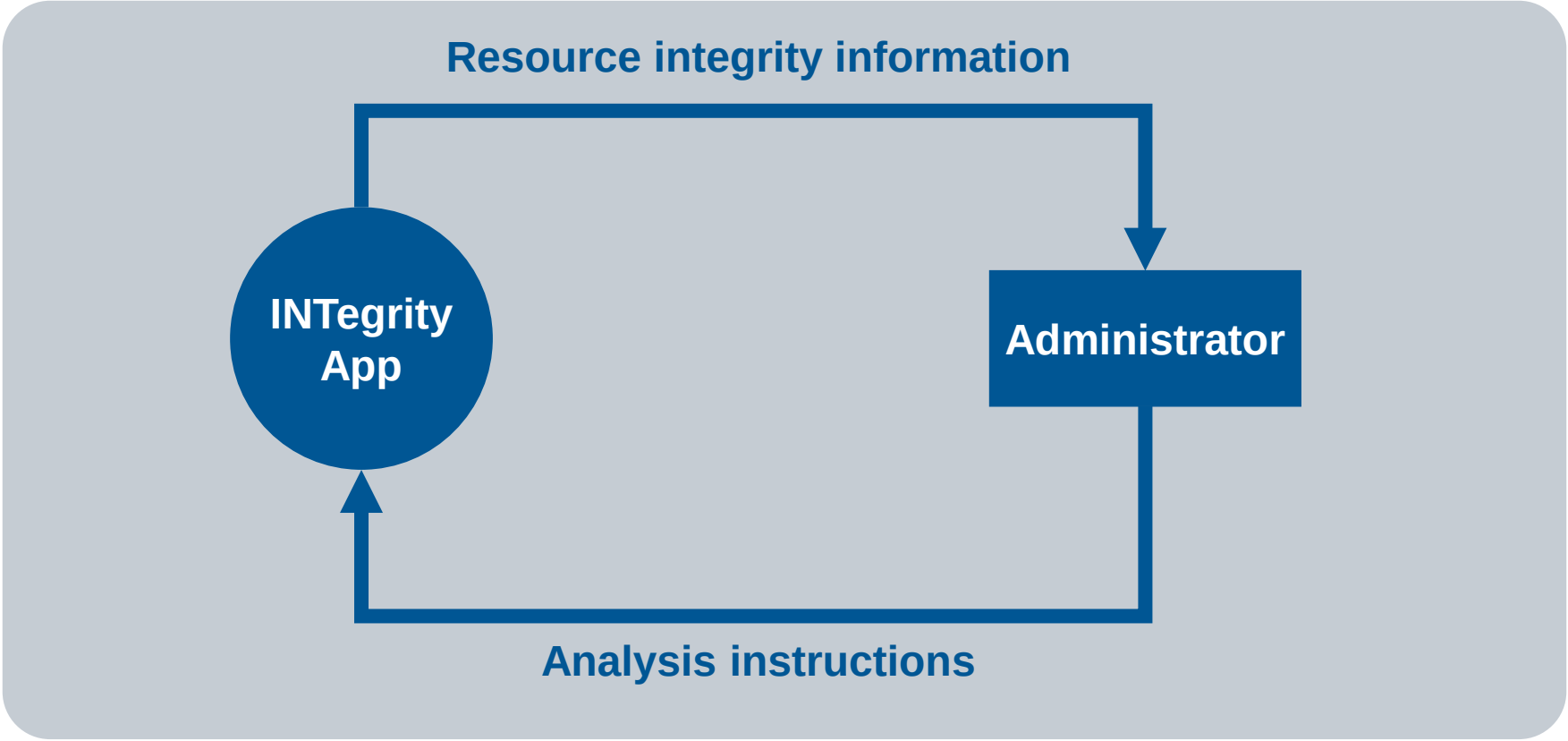
Building a threat model (at Microsoft)

- Program Manager (PM) owns overall process.
- Testers
 - Identify threats in analyze phase.
 - Use threat models to drive test plans.
- Developers create diagrams.

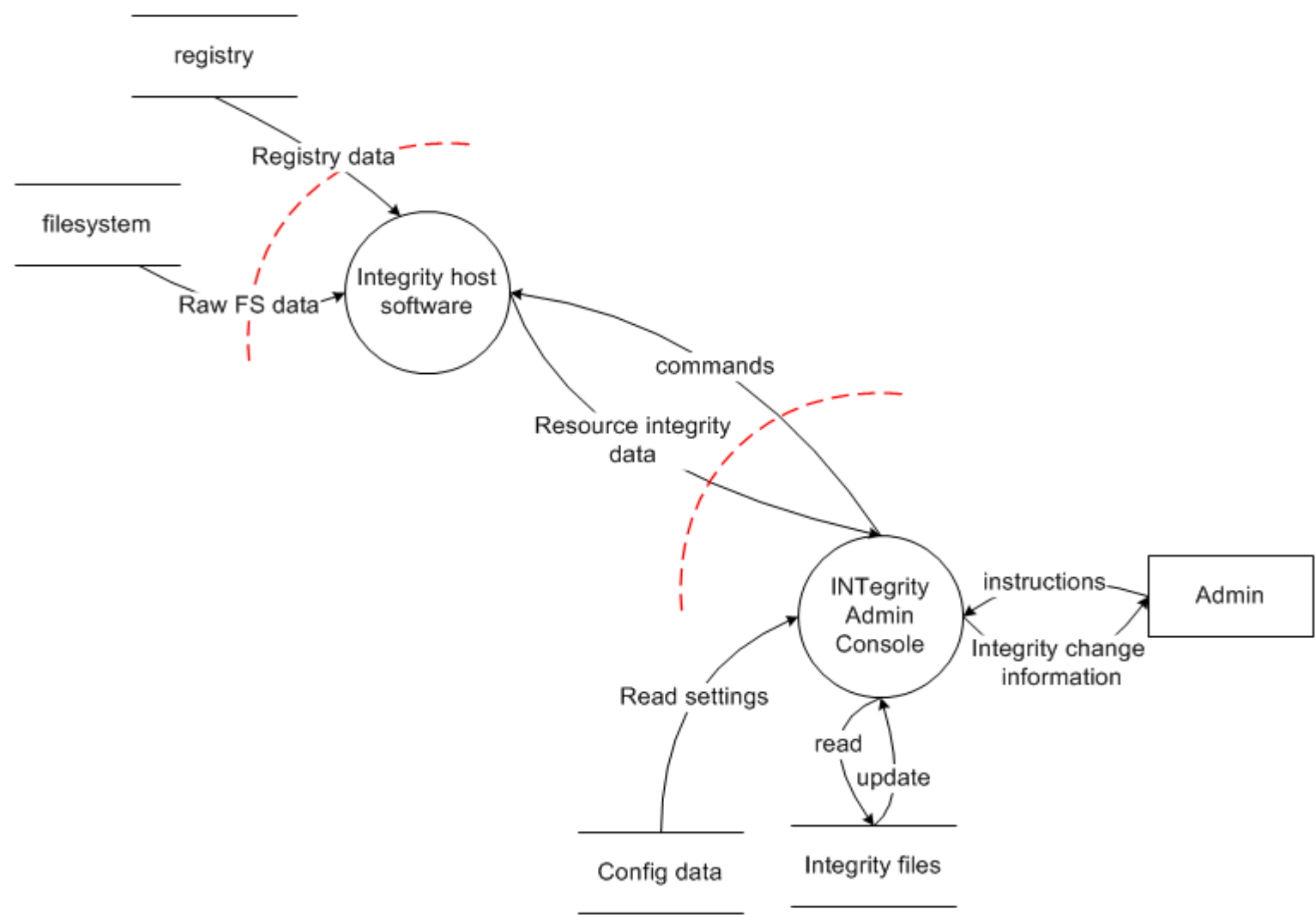
Customers for threat models

- Your team
- Other feature teams, other product teams
- Customers, via user education
- Quality assurance staff external to your team, such as penetration testers/ethical hackers

You'll need to decide what fits.



Level 1 Diagram



Context Diagram

- Very high-level; entire component / product / system

Level 1 Diagram

- High level; single feature / scenario

Level 2 Diagram

- Low level; detailed sub-components of features

Level 3 Diagram

- More detailed
- Rare to need more layers, except in huge projects or when you're drawing more trust boundaries

Top down

- Gives you the “context” in context diagram
- Focuses on the system as a whole
- More work at the start

Bottom up

- Developers know their product features.
- Approach not designed for synthesis
- More work overall

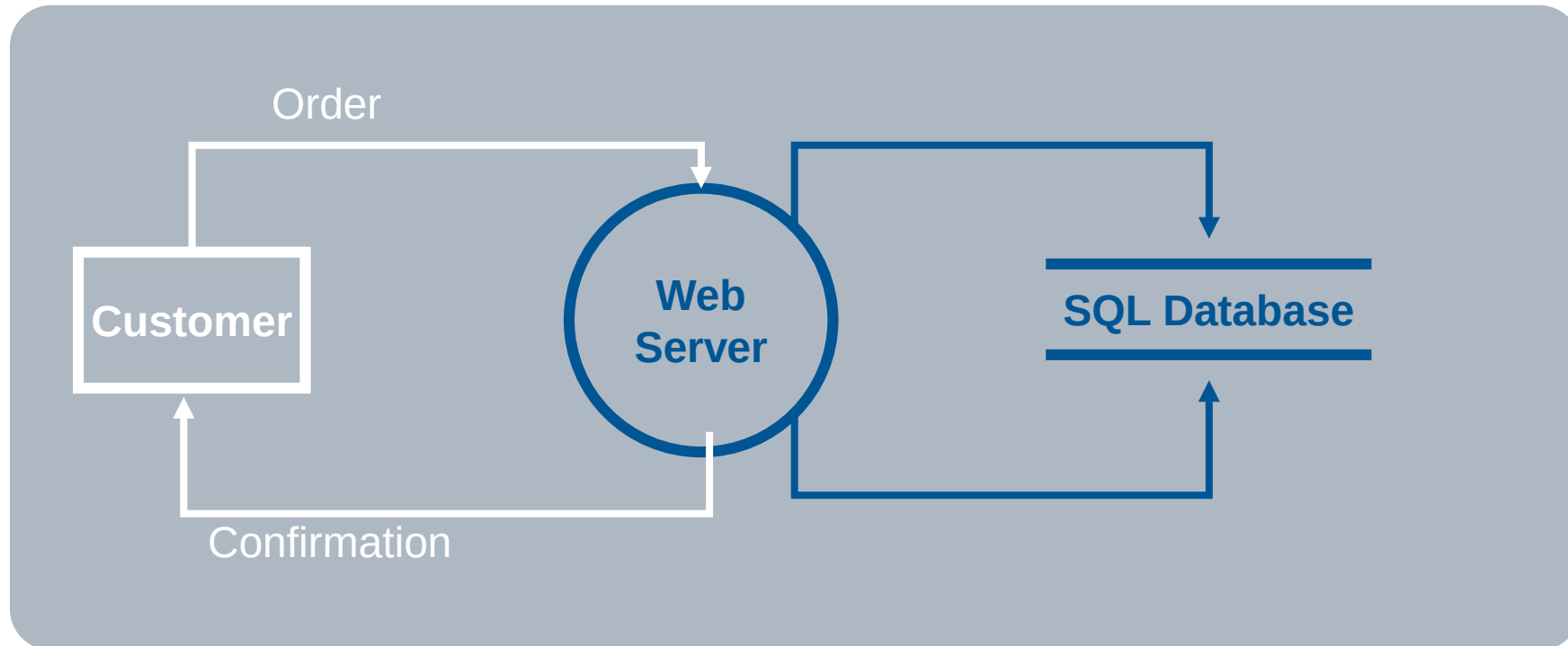
SQUARE Workshop

Diagram Validation Rules of Thumb



Diagram Validation Rules of Thumb

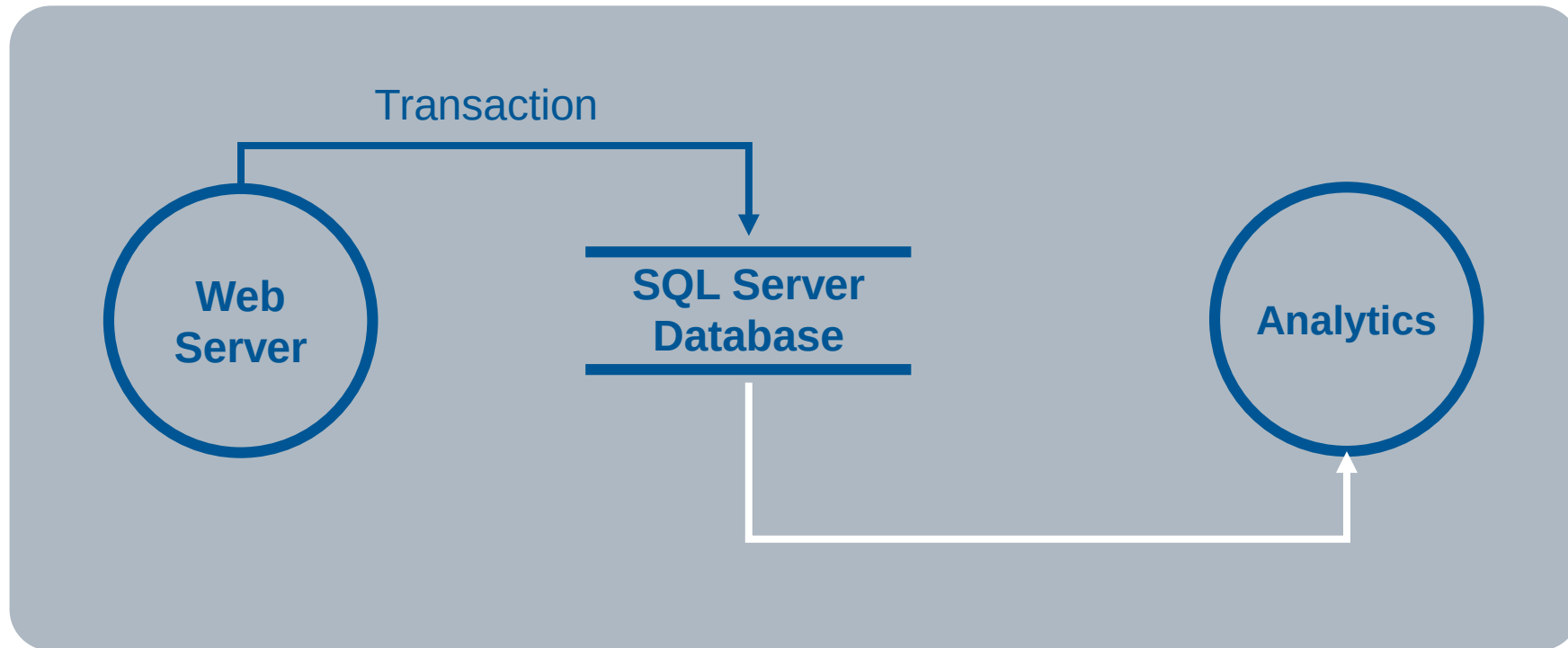
Does data magically appear?



Data must come from external entities or data stores

Diagram Validation Rules of Thumb

Are there data sinks?



You write to a store for a reason: Someone should be using it.

Diagram Validation Rules of Thumb

Data doesn't flow magically

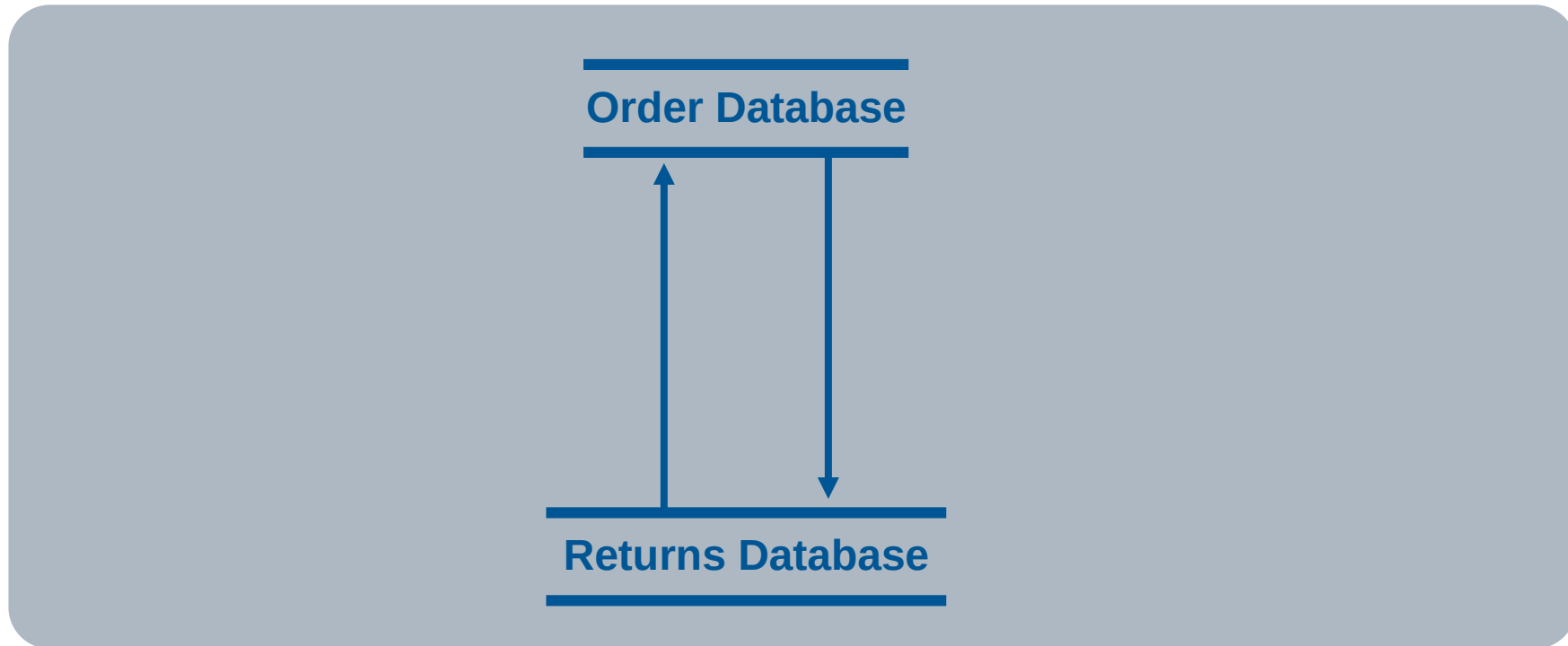
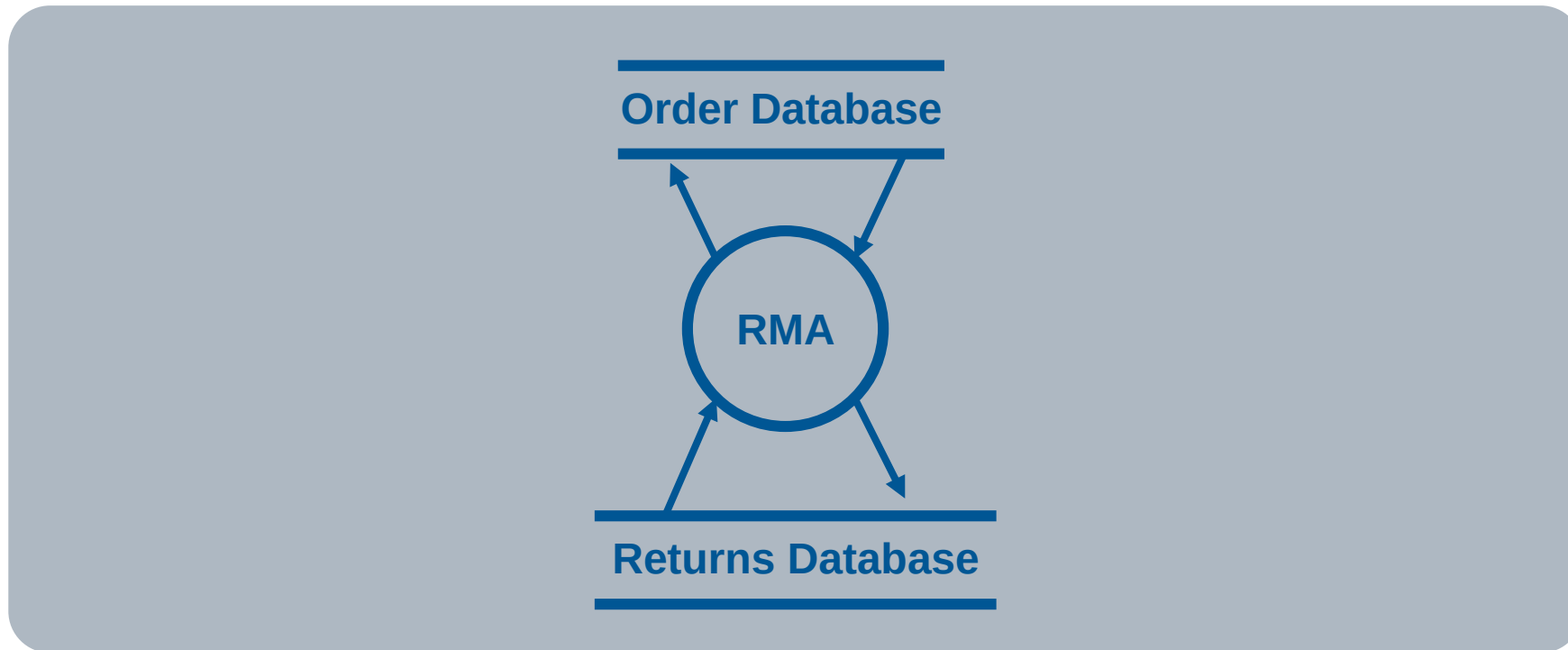


Diagram Validation Rules of Thumb

Data doesn't flow magically

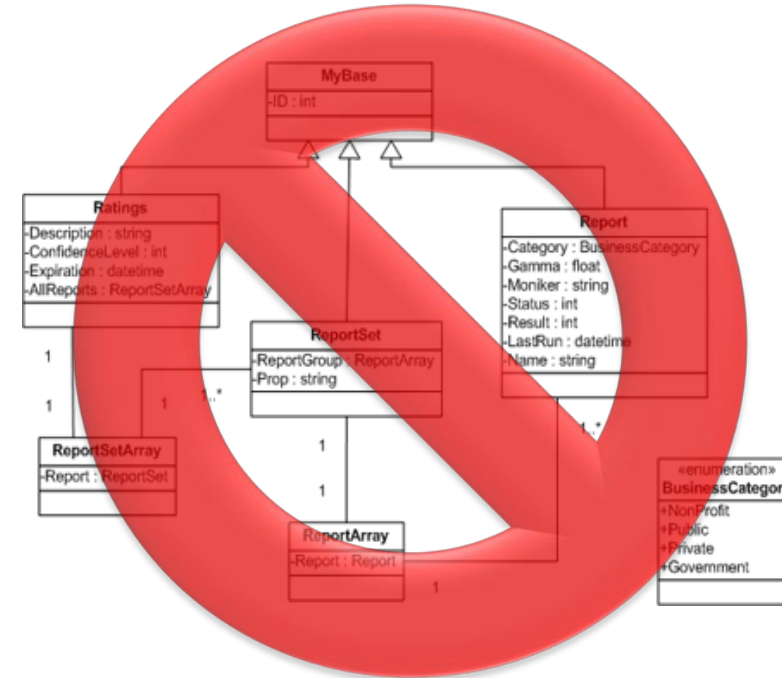
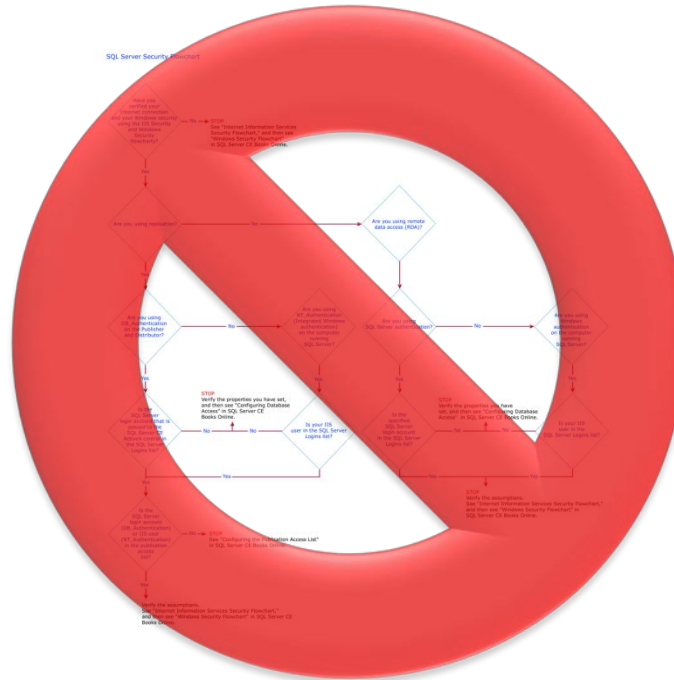


Diagrams Should Not Resemble

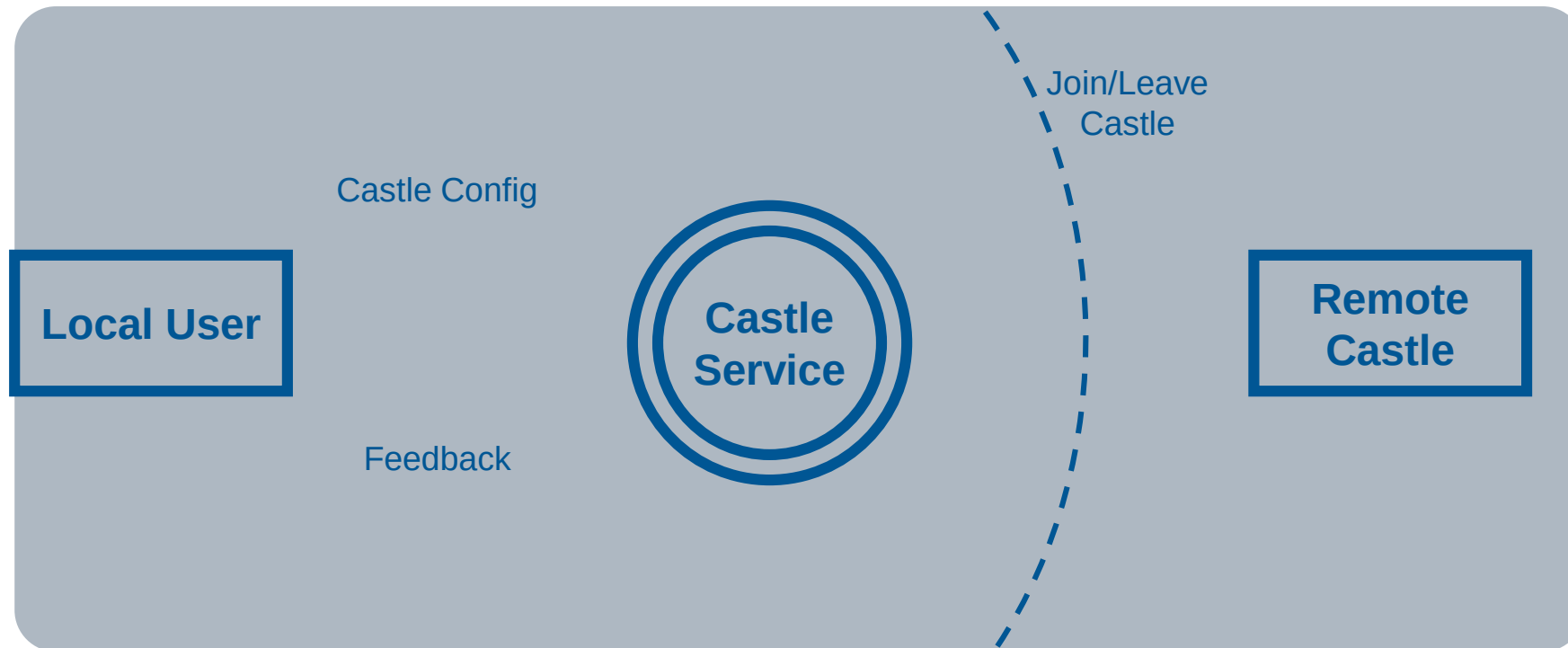
Flow charts

Class diagrams

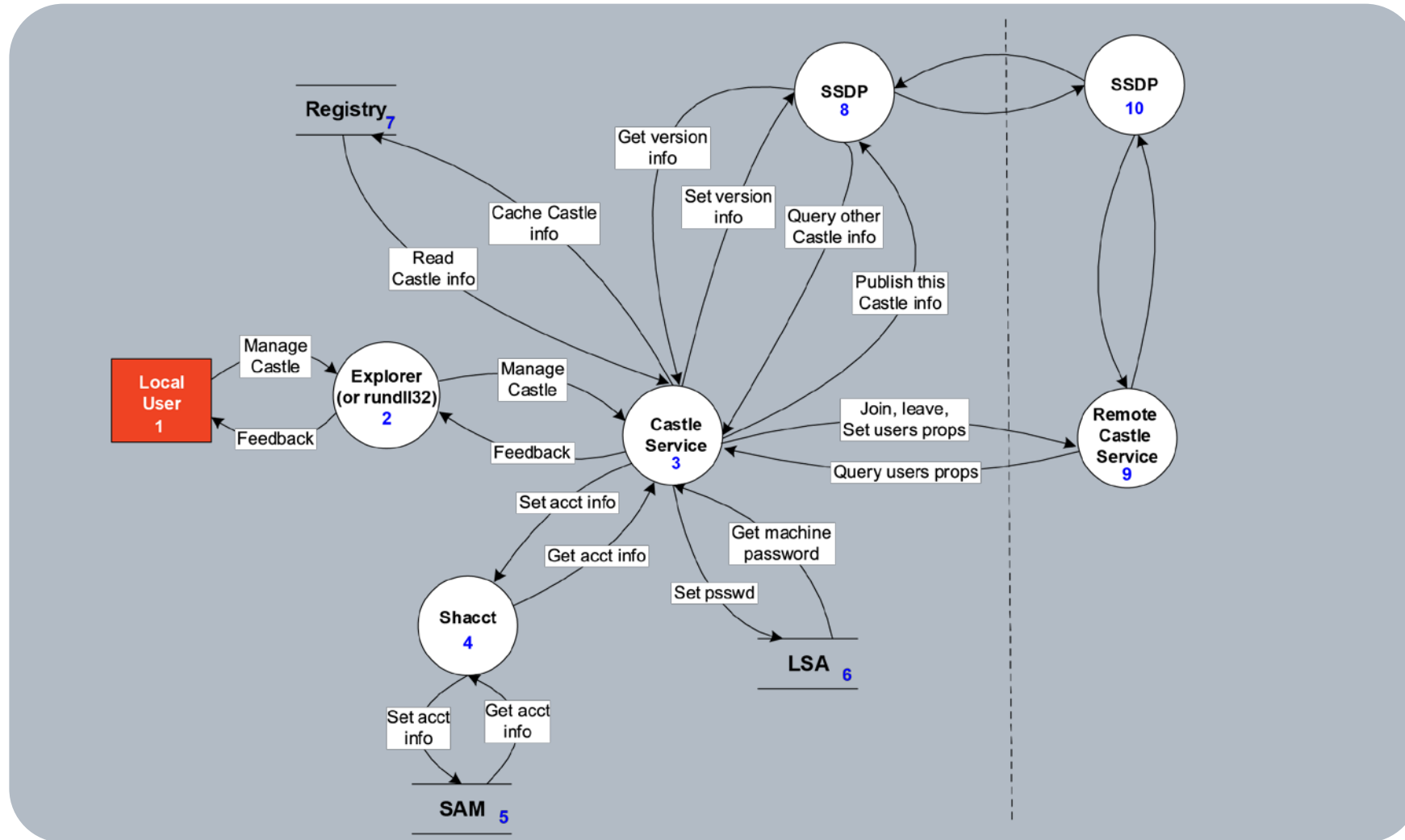
Call graphs



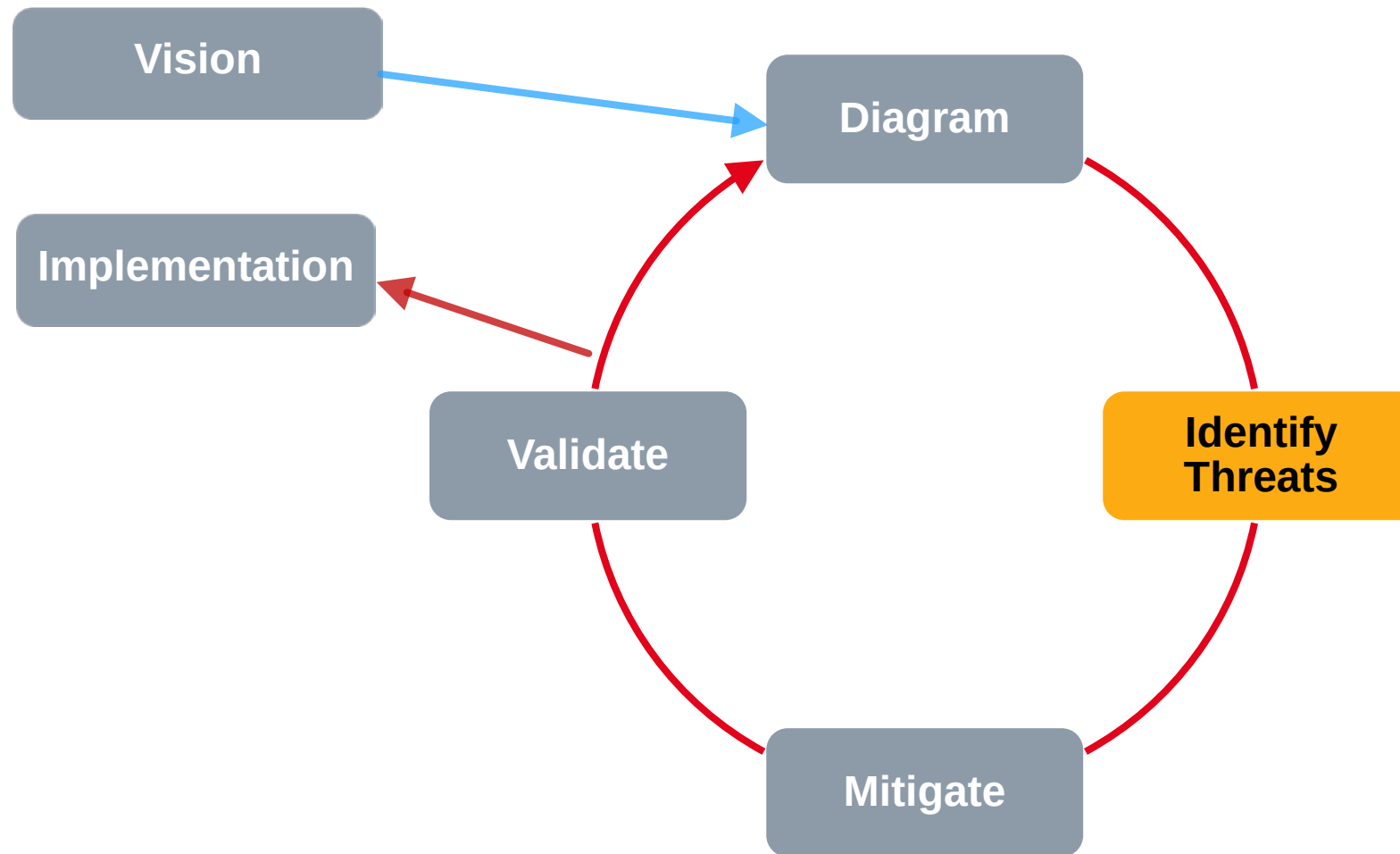
Real Context Diagram (“Castle”)



Castle Level 1 Diagram



The Process: Identifying Threats



Identifying Threats

Experts can brainstorm

How to do this without being an expert?

- Use STRIDE to step through the diagram elements
- Get specific about threat manifestation

Threat

Spoofing

Tampering

Repudiation

Information Disclosure

Denial of Service

Elevation of Privilege

Property we want

Authentication

Integrity

Nonrepudiation

Confidentiality

Availability

Authorization

Threat	S poofing
Property	Authentication
Definition	Impersonating something or someone else
Example	Pretending to be CEO or YourOrg.com

Threat	T ampering
Property	Integrity
Definition	Modifying data or code
Example	Modifying a packet as it traverses the LAN

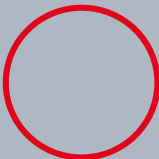
Threat	Repudiation
Property	Non-Repudiation
Definition	Claiming to have not performed an action
Example	“I didn’t send that email,” “I didn’t modify that file,” “I certainly didn’t visit that Web site!”

Threat	Information Disclosure
Property	Confidentiality
Definition	Exposing information to someone not authorized to see it
Example	Allowing someone to read source code without authorization; publishing a list of customers to a Web site

Threat	Denial of Service
Property	Availability
Definition	Deny or degrade service to users
Example	Crashing Windows or a Web site, sending a packet and absorbing seconds of CPU time, or routing packets into a black hole

Threat	E levation of Privilege (EoP)
Property	Authorization
Definition	Gain capabilities without proper authorization
Example	Allowing a remote Internet user to run commands is the classic example, but going from a “Limited User” to “Admin” is also Elevation of Privilege.

Different Threats Affect Each Element Type

ELEMENT	S	T	R	I	D	E
 External Entity	✖		✖			
 Process	✖	✖	✖	✖	✖	✖
 Data Store		✖	?	✖	✖	
 Data Flow		✖		✖	✖	

Apply STRIDE Threats to Each Element

For each item on the diagram:

- Apply relevant parts of STRIDE
- Process: STRIDE
- Data store, data flow: TID
 - Data stores that are logs: TID+R
- External entity: SR
- Data flow inside a process:
 - Don't worry about T, I, or D.

This is why you number things.

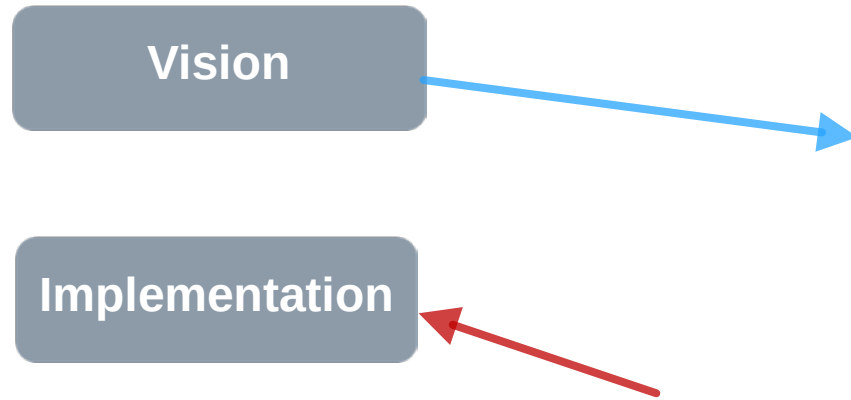
Trusted/high code reading from untrusted/low

- Validate everything for specific and defined uses.

High code writing to low

- Make sure your errors don't give away too much.

The Process: Mitigation



Mitigation

- To address or alleviate a problem

Protect customers/users

Design secure software

Why bother if you:

- Create a great model
- Identify lots of threats
- Stop

So, find problems and fix them.

Address each threat

Four ways to address threats

- Redesign to eliminate
- Apply standard mitigations
 - What have similar software packages done and how has that worked out for them?
- Invent new mitigations (riskier)
- Accept vulnerability in design.
 - At Microsoft, there are SDL rules about what you can accept.

Address each threat.

Standard Mitigations

Spoofing	Authentication	To authenticate principals: • Cookie authentication • Kerberos authentication • PKI systems such as SSL/TLS and certificates To authenticate code or data: • Digital signatures
Tampering	Integrity	• Windows Mandatory Integrity Controls • ACLs • Digital signatures
Repudiation	Non Repudiation	• Secure logging and auditing • Digital Signatures
Information Disclosure	Confidentiality	• Encryption • ACLs
Denial of Service	Availability	• ACLs • Filtering • Quotas
Elevation of Privilege	Authorization	• ACLs • Group or role membership • Privilege ownership • Input validation

- Mitigation #3: The Publish License is created by Rights Management System (RMS), and we've been advised that it's only OK to include an unencrypted email address if it's required for the service to work. Even if it is required, it seems like a bad idea due to easy email harvesting.
- Primary Mitigation: Bug #123456 has been filed against the RMS team to investigate removing the email address from this element. If that's possible, this would be the best solution to our threat.
- Backup Mitigation: It's acceptable to mitigate this by warning the document author that their email address may be included in the document. If we have to ship it, the user interface will be updated to give clear disclosure to the author when they are protecting a document.

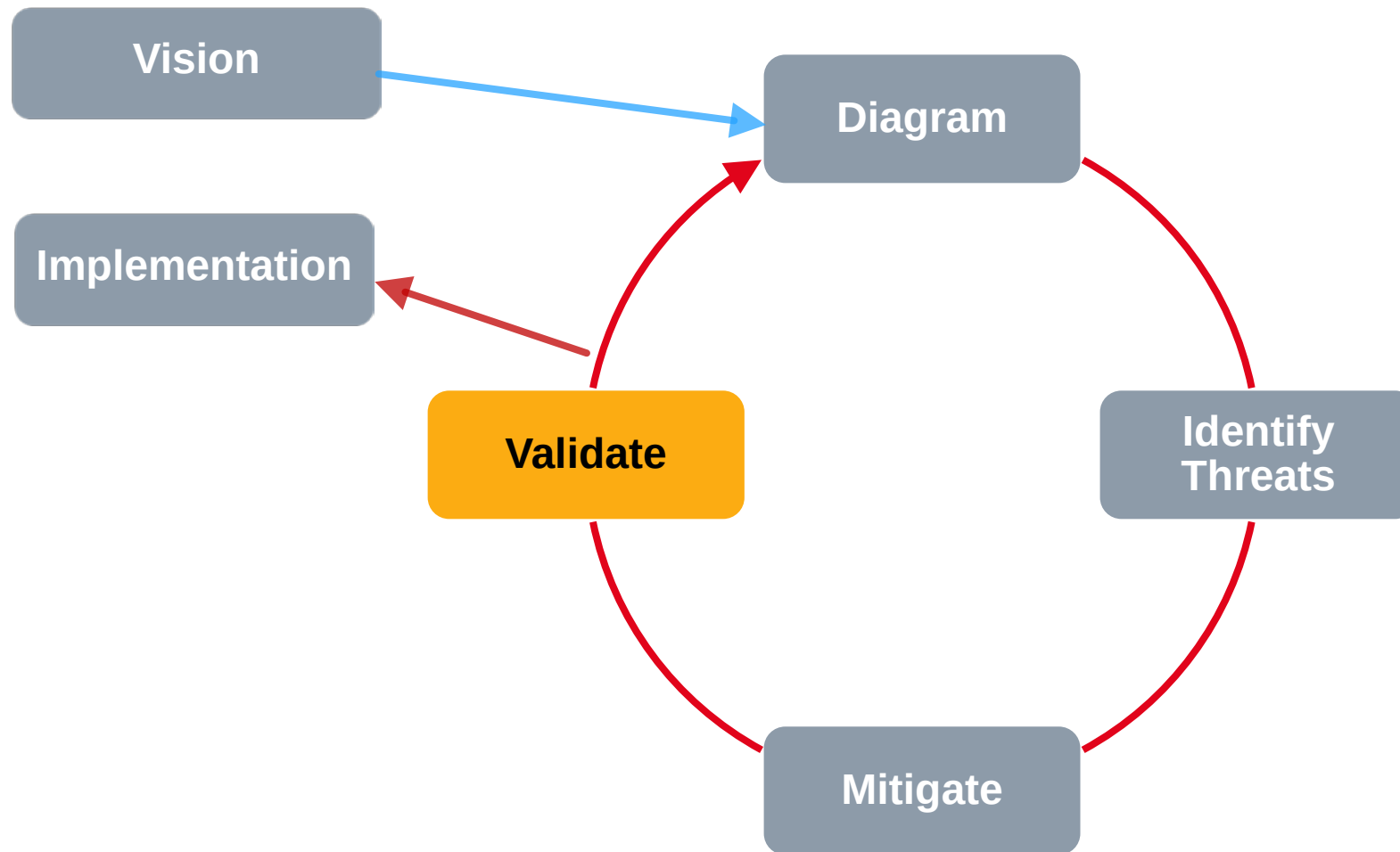
Mitigations are an area of expertise, such as networking, databases, or cryptography.

Amateurs make mistakes, but so do pros.

Mitigation failures will appear to work, to a novice, but may not.

When you need to invent mitigations, get expert help.

The Process: Validation



Validate the whole threat model.

- Does diagram match final code?
- Are threats enumerated?
- Minimum: STRIDE per element that touches a trust boundary
- Has Test / QA reviewed the model?
 - Tester often finds issues with threat model or details.
- Is each threat mitigated?
- Are mitigations done right?

Did you check these before Final Security Review?

- Shipping/delivery will be more predictable

Threats: Do they

- Describe the attack?
- Describe the context?
- Describe the impact?

Mitigations

- Associate with a threat
- Describe the mitigations.
- File a bug report to correct the design.

Dependencies

- What other software are you using?
- What security functions are in that other software?
- Are you sure?

Assumptions

- Note these as you build the threat model.
- Validate your assumptions.

Develop draft threat model before the meeting.

- Use the meeting to discuss

Start with a DFD walkthrough.

Identify most interesting elements

- Assets (if you identify any)
- Entry points/trust boundaries

Walk through STRIDE against those elements.

For threats that cross elements/recur

- Consider redesign

Books

- *The Security Development Lifecycle: SDL: A Process for Developing Demonstrably More Secure Software*
(Howard, Lipner) “Threat Modeling” chapter
- *Threat Modeling: Designing for Security*
(Shostack)

MSDN Magazine

<https://msdn.microsoft.com/magazine/msdn-magazine-issues>

- Reinvigorate your Threat Modeling Process. July 2008
- Threat Modeling: Uncover Security Design Flaws Using
The STRIDE Approach. November 2006



SDL Portal

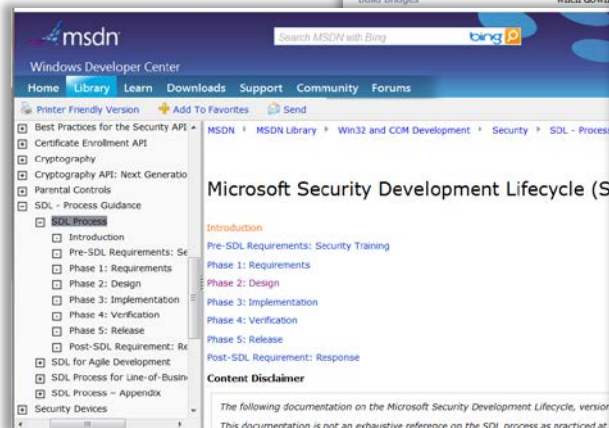
<http://www.microsoft.com/sdl>

SDL Blog

<https://blogs.microsoft.com/microsoftsecure/>

SDL Process

<http://msdn.microsoft.com/en-us/library/cc307748.aspx>

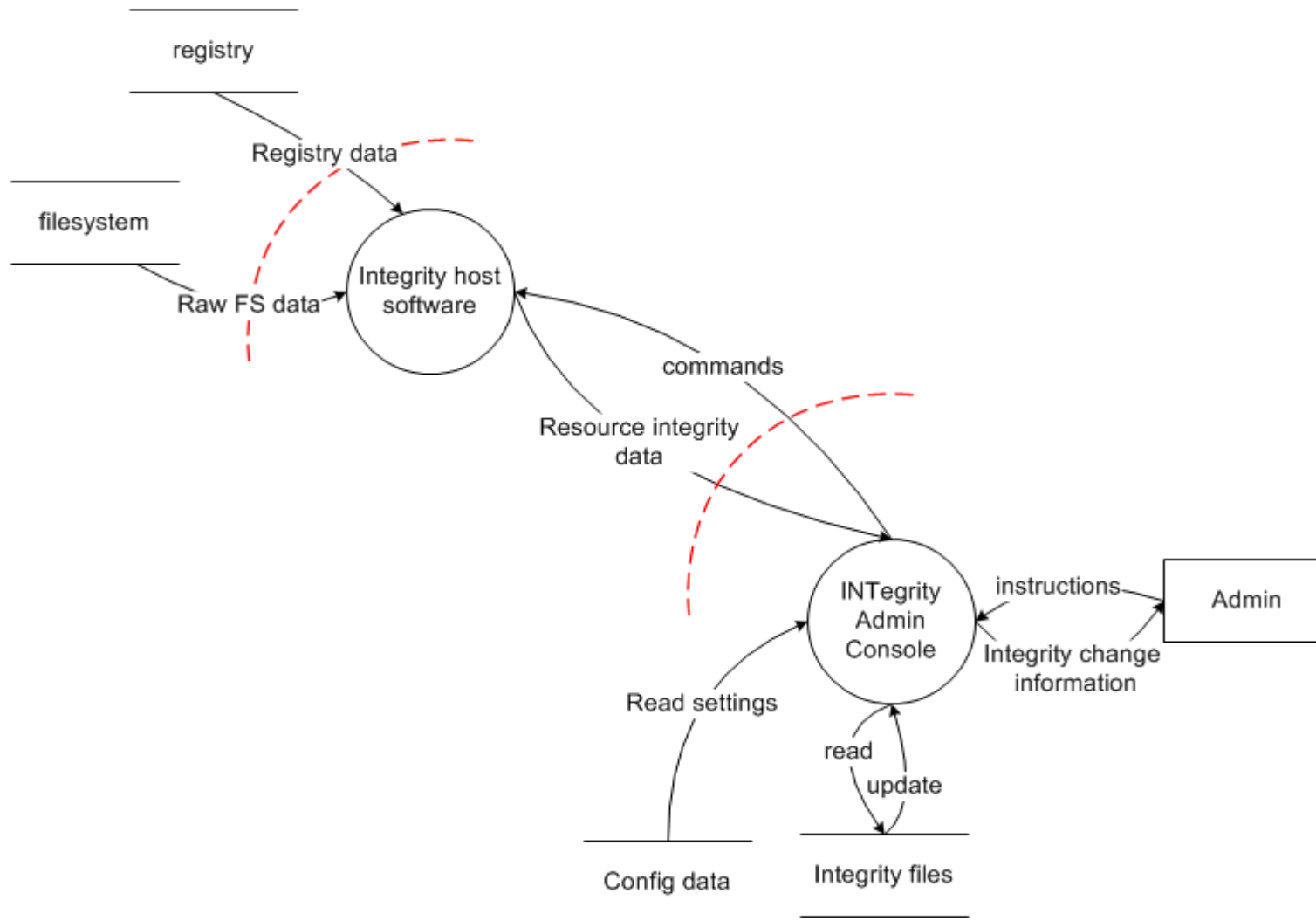


SQUARE Workshop

Exercise



Exercise – Develop a Threat Model for this Level 1 Diagram



- Identify all diagram elements
- Identify threat types to each element
- Identify at least three threats
- Identify first order mitigations
- Improve the diagram

SQUARE Workshop

Backup Slides



STRIDE

Threat	Property we want
Spoofing	Authentication
Tampering	Integrity
Repudiation	Nonrepudiation
Information Disclosure	Confidentiality
Denial of Service	Availability
Elevation of Privilege	Authorization

STRIDE

Threat	Property	
Spoofing	Authentication	To authenticate principals: • Basic authentication • Digest authentication • Cookie authentication • Windows authentication (NTLM) • Kerberos authentication • PKI systems, such as SSL or TLS and certificates • IPSec • Digitally signed packets To authenticate code or data: • Digital signatures • Message authentication codes • Hashes

STRIDE

Threat	Property
--------	----------

Tampering	Integrity
-----------	-----------

- Windows Vista mandatory integrity controls
- ACLs
- Digital signatures
- Message authentication codes

STRIDE

Threat

Property

Repudiation

Nonrepudiation

- Strong authentication
- Secure logging and auditing
- Digital signatures
- Secure time stamps
- Trusted third parties

STRIDE

Threat

Information
Disclosure

Property

Confidentiality

- Encryption
- ACLs

STRIDE

Threat

Denial of Service

Property

Availability

- ACLs
- Filtering
- Quotas
- Authorization
- High-availability designs

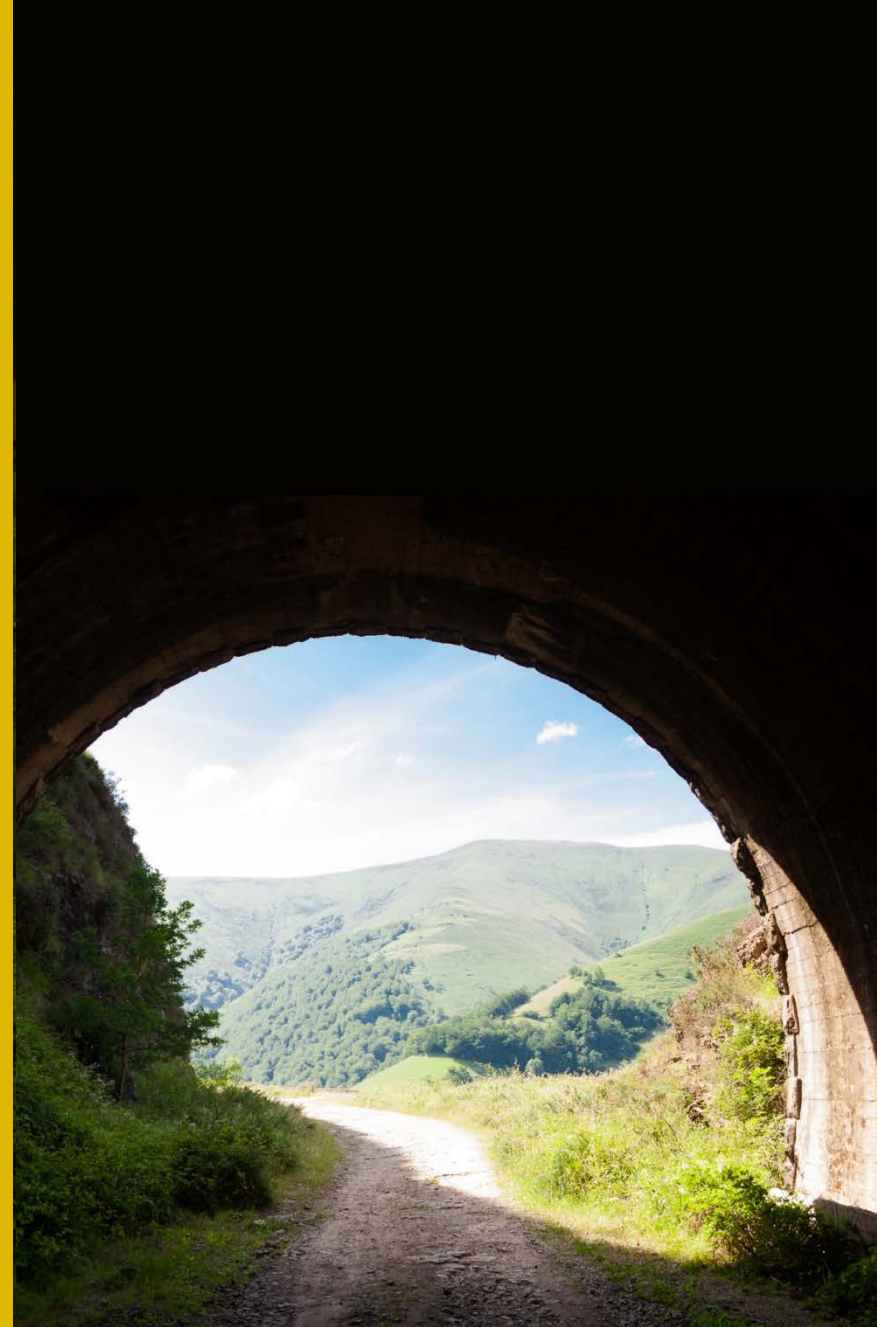
STRIDE

Threat	Property
Elevation of Privilege (EoP)	Authorization

SQUARE Workshop

SQUARE Steps 5-7

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213



Software Engineering Institute

Carnegie Mellon University

© 2017 Carnegie Mellon University

[Distribution Statement A] This material has been approved for public release and unlimited distribution.

Copyright 2017 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material is distributed by the Software Engineering Institute (SEI) only to course attendees for their own individual study.

Except for any U.S. government purposes described herein, this material SHALL NOT be reproduced or used in any other manner without requesting formal permission from the Software Engineering Institute at permission@sei.cmu.edu.

Although the rights granted by contract do not require course attendance to use this material for U.S. Government purposes, the SEI recommends attendance to ensure proper understanding.

Carnegie Mellon® and CERT® are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM17-0350

1. Agree on definitions.
2. Identify assets and security goals.
3. Develop artifacts to support security requirements definition.
4. Assess risks.
5. **Select elicitation technique(s).**
6. **Elicit security requirements.**
7. **Categorize requirements.**
8. Prioritize requirements.
9. Inspect requirements.

SQUARE Step 5



Step 5: Select Elicitation Technique

Potentially many different options available in any given setting

Can use existing requirements elicitation technique

Must choose a technique that can adapt to the number and expertise of stakeholders

Step 5: Select Elicitation Technique

Potential techniques and their references:

- Misuse Cases [Jacobson 92]
- Soft Systems Methodology (SSM) [Checkland 90]
- Quality Function Deployment (QFD) [QFD 05]
- Controlled Requirements Expression (CORE) [SDS 86]
- Issue Based Information Systems (IBIS) [Kunz 70]
- Joint Application Development (JAD) [Wood 89]
- Feature-Oriented Domain Analysis (FODA) [Kang 90]
- Critical Discourse Analysis (CDA) [Schiffrin 94]
- Accelerated Requirements Method (ARM) [Hubbard 00]
- Structured/Unstructured Interviews

Techniques may be graded on the following example criteria. Criteria can be weighted if some are more important than others:

- Adaptability
- CASE tool
- Client acceptance
- Complexity
- Graphical output
- Implementation duration
- Learning curve
- Maturity
- Scalability

Step 5: Select Elicitation Technique

Subjective comparison of elicitation techniques prior to pilot studies. IBIS, JAD, and ARM used in pilots.

	ARM	GDA	FODA	JAD	IBIS	CONC	QFD	SSM	Cases	Misuse
	2	1	2	3	2	2	3	1	3	
	1	1	1	2	3	1	1	2	1	
	3	3	1	2	3	2	2	2	2	
	2	1	1	2	3	2	1	2	2	
	3	2	1	1	2	1	1	2	2	
	3	2	2	1	2	1	1	2	2	
	1	1	1	2	3	1	2	1	3	
	1	2	3	3	2	3	3	3	2	
	2	1	2	2	2	2	2	2	1	

3 = very good, 2 = fair, 1 = poor

Step 5: Select Elicitation Technique

ARM performed best in pilot studies

Issue Cases	SSM	QFD	QFD	QFD	QFD	QFD	QFD	QFD	QFD
	3	1	3	2	2	3	2	1	2
	1	2	1	1	3	2	1	1	1
	2	2	2	2	3	2	1	3	3
	2	2	1	2	3	2	1	1	2
	2	2	1	1	2	1	2	2	3
	2	2	1	1	2	1	2	2	3
	3	1	2	1	3	2	1	1	1
	2	3	3	3	2	3	2	2	1
	1	2	2	2	2	2	2	1	2

3 = very good, 2 = fair, 1 = poor



Most preferred option

Step 5: Select Elicitation Technique

Accelerated Requirements Method (ARM)

- Performed best in eliciting security requirements during pilot case studies
- Participants asked to contribute seven security requirements in seven minutes
- Participants asked to pick three of those requirements as the most important in three minutes
- Results posted on a wall and discussed

Step 5: Select Elicitation Technique

Requirements engineering team responsibilities

- Select an elicitation technique that is appropriate for the number and expertise of stakeholders, size and scope of the project, and expertise of the requirements engineering team.
- Document the rationale for the choice.
- Make necessary preparations to execute the technique.

Step 5: Select Elicitation Technique

Stakeholder responsibility

- If the organization wants to use an existing requirements elicitation technique, the stakeholders must let the RE team know about it.

Step 5: Select Elicitation Technique

Exit criteria

- Selection of an appropriate elicitation technique
- Documented rationale for the choice

Task: Agree on the elicitation technique

Now you will have the opportunity to determine an elicitation technique.

Read and follow the instructions given in your workbook for more detailed information. Use the videos that are available to you.

SQUARE Step 6



Step 6: Elicit Security Requirements

The requirements engineering team must work with the stakeholders to elicit security requirements for the project (face to face).

Requirements must

- be verifiable
- not be vague or ambiguous
- be quantifiable whenever possible

Example: “The system must have authentication measures in place at all gateways/entrance points.”

Focus on what, not how.

Step 6: Elicit Security Requirements

Requirements engineering team responsibilities

- Execute elicitation technique chosen in Step 5
- Logistic preparation and orientation for stakeholders
- Document requirements as they are collected

Step 6: Elicit Security Requirements

Stakeholder responsibilities

- Follow the instructions given by the requirements engineering team during the elicitation process.

Joint responsibilities

- Encourage generation of verifiable, preferably quantifiable security requirements.
- Ensure that only requirements are generated and not implementations or architectural constraints.

Task: Agree on the requirements set

Exit Criteria:

- Both teams have to discuss and finalize the requirements
- Requirements should be quantifiable and verifiable

Read and follow the instructions given in your workbook for more detailed information. Use the videos that are available to you.

SQUARE Step 7



Step 7: Categorize Requirements

Allows requirements engineers and stakeholders to classify requirements, for example, as

- System level
- Software level
- Architectural constraint

Step 7: Categorize Requirements

Minimal set of categories for requirements categorization

	System Level	Software Level	Architectural Constraint
Req. 1			
Req. 2			

Step 7: Categorize Requirements

Avoid producing **architectural constraints** instead of requirements.

- The table mentions architectural constraints only because some “requirements” may be deemed architectural constraints after categorization.

Requirements may become architectural constraints but one should not produce them and call them requirements! When architectural constraints become requirements, the software solution is constrained in ways that can result in increased cost, schedule, or other undesirable side effects.

Step 7: Categorize Requirements

Example method

- ARM (Accelerated Requirements Method)

Group A : Confidentiality 1. Information must be kept private from outside world 2. Selectively secure communication with outside entities	Group B : Access Control 1. Role-based access/views 2. Need for an access control system
Group C : Data Integrity 1. Partitioned data store(public read only and private read/write) 2. Indelibility	Group D : Authentication 1. Strong authentication

Step 7: Categorize Requirements

Requirements engineering team responsibilities

- Provide baseline set of categories
 - May have to suggest alternative categories
- Facilitate the stakeholders' categorization process

Step 7: Categorize Requirements

Stakeholder responsibilities

- Come to a consensus on the categorization for each requirement.

Step 7: Categorize Requirements

Exit criteria

- Initial set of requirements organized into stakeholder-defined categories
- Architectural constraints identified

Task: Categorize the security requirements

Exit Criteria:

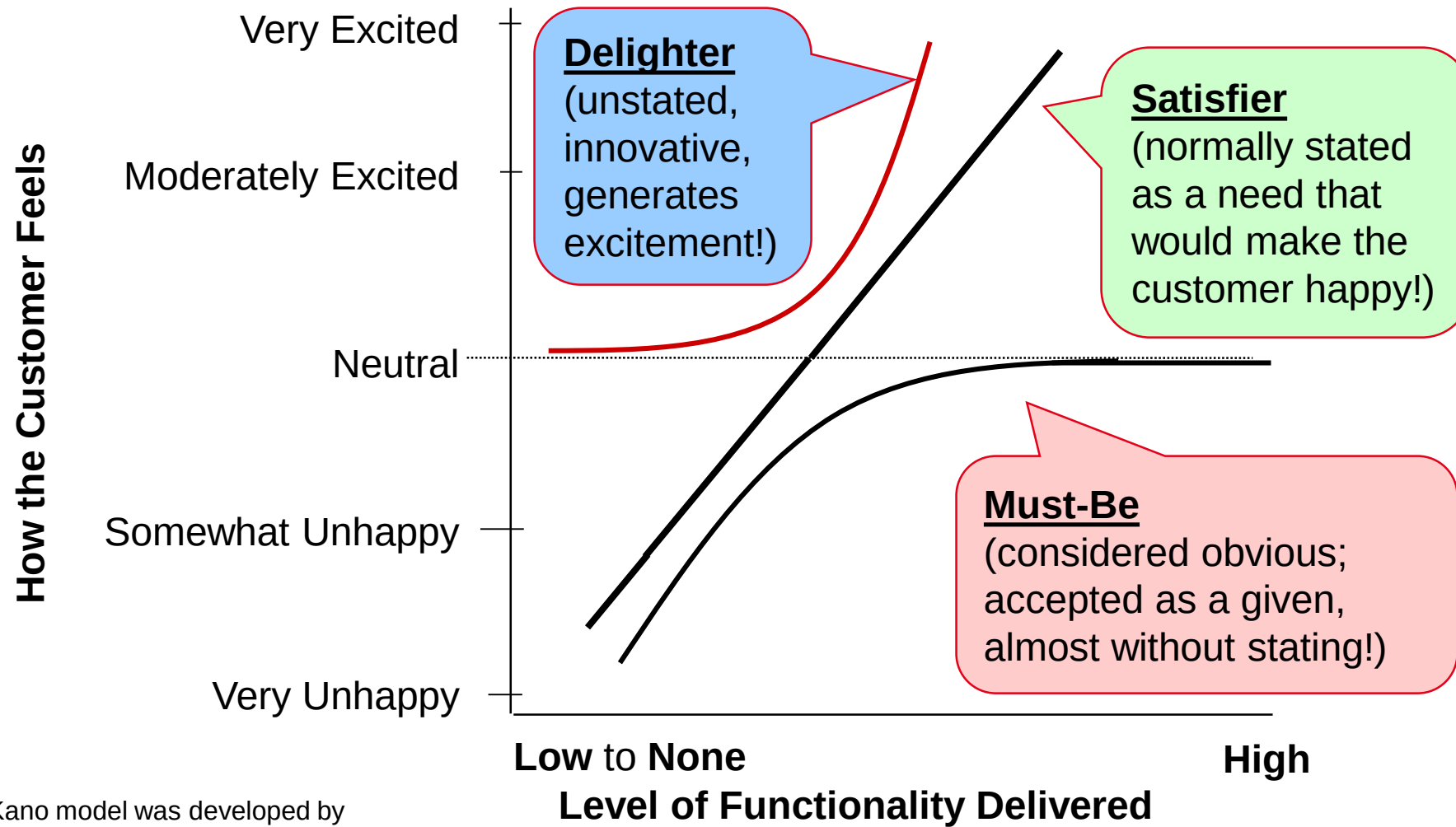
- Both teams have to discuss and categorize the security requirements.

Categorization can be essential, non-essential, or software-level and system-level requirements.

If a particular requirement is actually an architectural constraint, it should be eliminated.

Read and follow the instructions given in your workbook for more detailed information. Use the videos that are available to you.

Another Categorization Method - Kano Model (for a given requirement)



The Kano model was developed by Professor Noriaki Kano in the 1980s.

How Kano May Be Used Within Requirements Elicitation Process

Kano Analysis is an approach to categorize requirements into one of three categories (“Must-Be” vs. “Satisfier” vs. “Delighter”).

All requirements, whether stated or unstated, may be categorized using Kano analysis.

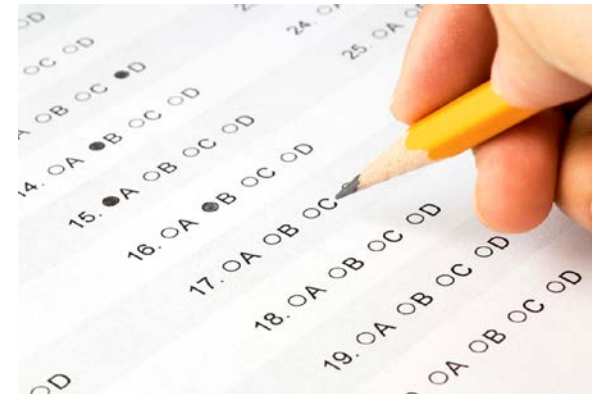
Kano analysis is conducted using a survey in which a pair of questions are asked for each requirement.

New/refreshed products or services are more compelling if they have some “Delighters.”

Programs/projects must not forget to cover the “Must-Be’s” automatically as they “go without saying.”

“Satisfiers” are good to have but the “Delighters” are much more compelling.

“Must-Be’s” and “Satisfiers” are more easily identified.

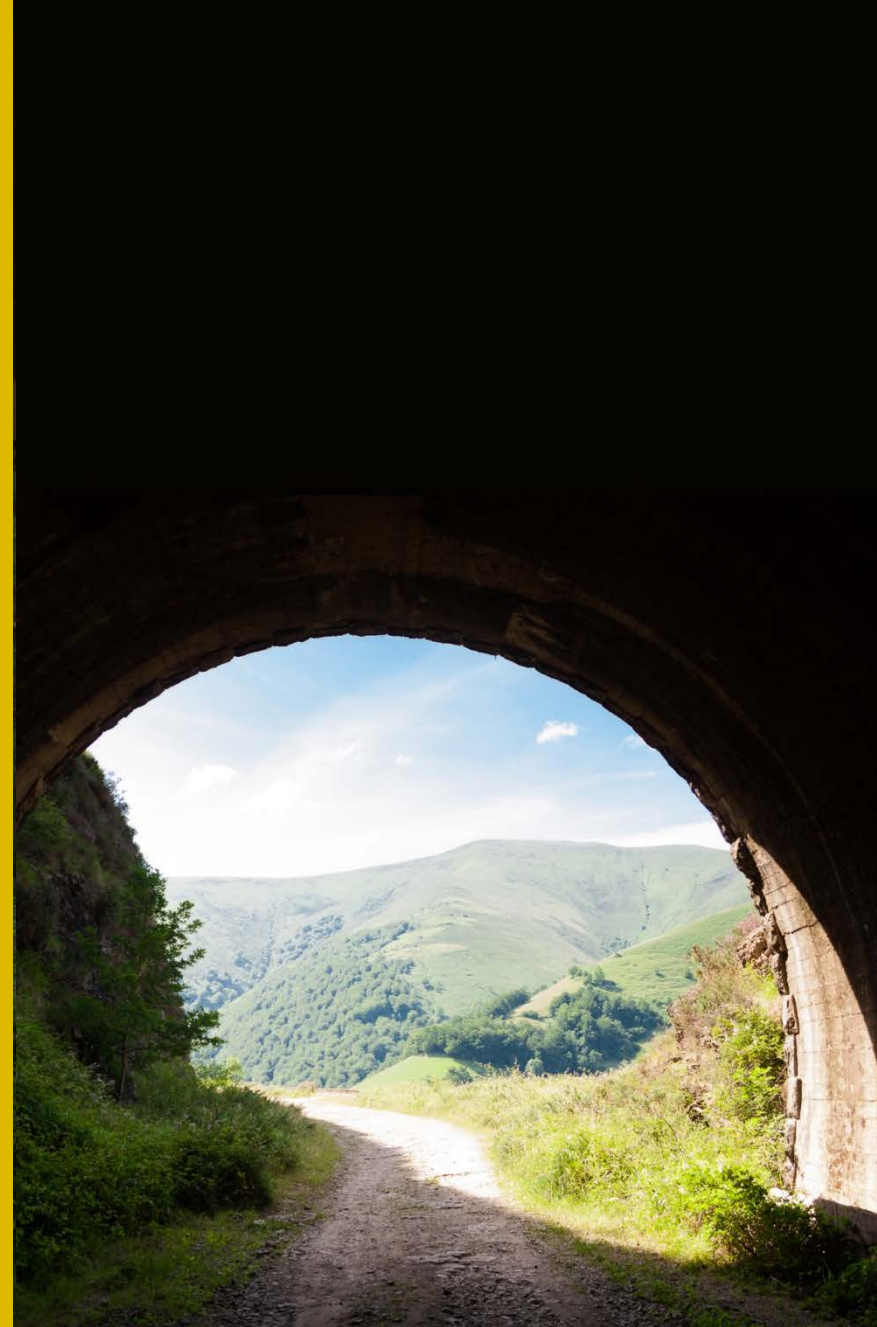


We have described a few categorization methods.
Which are likely to be best for security requirements?
Can methods such as Kano be used successfully for
security requirements? Why/why not?

SQUARE Workshop

SQUARE Steps 8-9, Wrap-up

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213



Software Engineering Institute

Carnegie Mellon University

© 2017 Carnegie Mellon University

[Distribution Statement A] This material has been approved for public release and unlimited distribution.

Copyright 2017 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material is distributed by the Software Engineering Institute (SEI) only to course attendees for their own individual study.

Except for any U.S. government purposes described herein, this material SHALL NOT be reproduced or used in any other manner without requesting formal permission from the Software Engineering Institute at permission@sei.cmu.edu.

Although the rights granted by contract do not require course attendance to use this material for U.S. Government purposes, the SEI recommends attendance to ensure proper understanding.

Carnegie Mellon® and CERT® are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM17-0350

1. Agree on definitions.
2. Identify assets and security goals.
3. Develop artifacts to support security requirements definition.
4. Assess risks.
5. Select elicitation technique(s).
6. Elicit security requirements.
7. Categorize requirements.
- 8. Prioritize requirements.**
- 9. Inspect requirements.**

SQUARE Step 8



Step 8: Prioritize Requirements

Client organization unlikely to implement all of the security requirements

- Lack of time
- Lack of resources
- Changes in the goals of the project

Step 8: Prioritize Requirements

Crucial input

- Step 4: Risk Assessment
- Step 7: Categorization

Step 8: Prioritize Requirements

Prioritization techniques

- Binary search trees
- Numeral assignment techniques
- Planning game
- 100-Point method
- Theory-W
- Triage
- Wiegers' Method
- Requirements Prioritization Framework
- Analytical Hierarchy Process

Step 8: Prioritize Requirements

Prioritization example

- AHP – Analytic Hierarchy Process
 - Handles requirements well
 - Calculates relative value and cost among security requirements

Step 8: Prioritize Requirements

Prioritization example: Pairwise Comparison Matrix

	A	B	C	D	E	F	G	H	I	J
1		SR-1	SR-2	SR-3	SR-4	SR-5	SR-6	SR-7	SR-8	SR-9
2	SR-1									
3	SR-2	1/8								
4	SR-3	5	5							
5	SR-4	1/3	7	1						
6	SR-5	1	7	1/2	2					
7	SR-6	1/2	7	1	2	1/3				
8	SR-7	1/2	7	1/3	1/3	1/3	3			
9	SR-8	1/3	9	1	2	1	1	1/3		
10	SR-9	1	9	1	1	3	1	1/2	6	

i, j shows how much higher (or lower) the value/cost for requirement i is than that of requirement j

1 - Requirements i and j are of equal cost.

3 - Requirement i has slightly higher cost than j.

5 - Requirement i has strongly higher cost than j.

7 - Requirement i has very strongly higher cost than j.

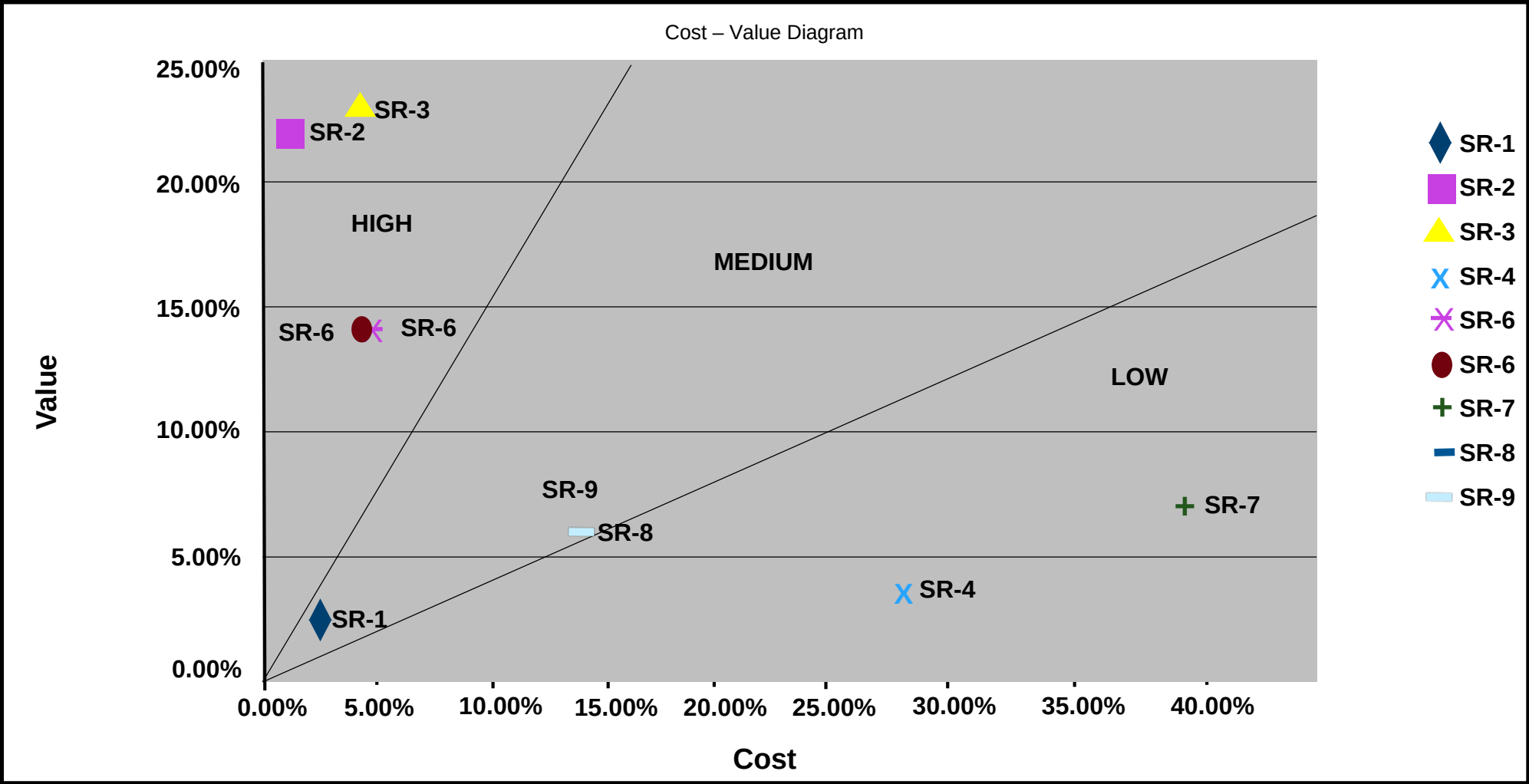
9 - Requirement i has absolutely higher cost than j.

2, 4, 6, 8 - These are intermediate scales between two adjacent judgments.

Reciprocals - If requirement i has lower cost than j.

Step 8: Prioritize Requirements

Prioritization example



Cost/Value Diagram of Requirements

Step 8: Prioritize Requirements

Some requirements may be deemed infeasible

- Dismiss requirement from further consideration
- Document the requirement as “future work”

Decision should be made after consulting with stakeholders

Step 8: Prioritize Requirements

Requirements engineering team responsibilities

- Facilitate prioritization process with stakeholders
- Teach stakeholders how to perform a structured prioritization process, if they have not already been trained in the selected method

Step 8: Prioritize Requirements

Stakeholder responsibilities

- Prioritize the security requirements using the risk assessment and categorization results.

Step 8: Prioritize Requirements

Exit criteria

- All security requirements have been prioritized.

Task: Prioritize requirements

Exit Criteria:

- Prioritization could be an unstructured, informal discussion or a structured one.

Structured prioritization techniques might take a lot of time and require a cost/benefit analysis.

In this exercise, try to use AHP for prioritization. Even if the results are not perfect, you will learn about the method.

Read and follow the instructions given in your workbook for more detailed information. Use the videos that are available to you.

SQUARE Step 9



Step 9: Requirements Inspection

Goal is to find defects in the requirements

- Ambiguities
- Inconsistencies
- Incorrect assumptions

Varying levels of formality

- Fagan Inspection
- Peer reviews

Peer review log format

SNo	DATE	ORIGIN	DEFECT TYPE	DESCRIPTION	SEVERITY	OWNER	REVIEWER	STATUS

Assigns each team member inspection responsibility

Ranks problems according to severity

Requirements engineering team responsibilities

- Facilitate the inspection process
 - Provide orientation to the structured inspection
 - Informal inspection guides

Stakeholder responsibilities

- Come to a consensus on the validity of each security requirement.
- Verify that each requirement is verifiable and feasible to implement.
- Remove requirements, if needed, from the working set. (Last chance!)
- Produce final requirements document.

Joint responsibilities

- Verify that each requirement is directly applicable to one or more of the security goals of the project or supports a higher level requirement.

Exit criteria

- All requirements have been verified both by the requirements engineering team and the stakeholders.

Task: Review and document review comments

Exit Criteria:

- All requirements should be carefully reviewed and the review comments should be documented.

If there is any drastic change in the requirements, the categorization and prioritization might change.

Read and follow the instructions given in your workbook for more detailed information. Use the videos that are available to you.

- Did you have occasion to do formal inspections on projects?
- What about peer reviews?
- What about pair programming in Agile? Any experience with it?

Things to think about

SQUARE Recap



SQUARE – Security Quality Requirements Engineering

Nine steps:

- | | |
|--|----------------------------------|
| (1) agree on definitions | (6) elicit security requirements |
| (2) identify assets and security goals | (7) categorize requirements |
| (3) develop artifacts | (8) prioritize requirements |
| (4) assess risks | (9) inspect requirements |
| (5) select elicitation technique(s) | |

SQUARE-Lite, P-SQUARE, A-SQUARE

- BSI content on requirements engineering
<https://buildsecurityin.us-cert.gov/>
- SQUARE Technical Report – SEI web site
www.sei.cmu.edu/pub/documents/05.reports/pdf/05tr009.pdf
- SQUARE Case Study Reports – SEI web site
- SQUARE Tool
<http://www.cert.org/cybersecurity-engineering/products-services/p-square-download.cfm>
- *Integrating Security and Software Engineering*
IDEA Group Publishing
www.idea-group.com

<http://www.cert.org/sse/square/square-tool.html>

Additional Background on Inspections

(Developed by Mel Rosso-Llopart and Anthony Lattanze)



- Software work product inspection
- Benefits of work product inspections
- The cost of inspections
- The formal work product inspection process

A process where a software work product is reviewed by a group of peers

- documents, designs, code

There are two general types of inspections:

- more formal inspections (formal)
- less formal inspections (informal)

In either case, the work product is always reviewed by a group of peers.

There are formal and informal inspection methods

Formal inspections feature

- highly structured process for preparation, meeting protocol, data collection, and post meeting activities
- table-top meeting oriented
- excellent for ensuring final revision product quality and standards

Informal inspections feature

- less structure, formality, and rigidity
- more open discussion (peer review)
- table-top or presentation oriented
- excellent for initial artifact presentation, artifact discussion, or selection of alternatives

Both formal and informal inspections have their place in a project.

The benefit of the inspection process is threefold:

- ensure that work products are of the highest possible quality
- ensure that software work products meets organizational and legal standards
- standard products are more maintainable than those that do not follow conventions for coding, commenting, and documentation

Provides visibility into the products and services that each engineer provides individually

Facilitates detailed technical communication which helps:

- foster reuse
- acts as a training vehicle
- provides fresh insight into technical problems

- The earlier a defect is found in the software development process, the less it costs to repair.
- The longer a defect goes undetected the more it will cost to repair. WHY?

Watts Humphrey, *A Discipline for Software Engineering*, Addison Wesley, 1995.

The dollar cost of finding a defect during system test vs. finding the defect in design is 100:1

- the time cost is 200:1

Jet Propulsion Lab (JPL) reported the cost of finding a defect in an inspection is \$100, in test \$10,000

On average, design and code reviews **reduce the cost of testing** by 50-80% including the cost of the reviews

Watts Humphrey, *A Discipline for Software Engineering*, Addison Wesley, 1995.

Exact cost will vary with organization

Rule of thumb for code inspections:

- ≈ 3 man hours per 100 lines of source code written

Cost for design or requirements inspections varies widely

Formal Inspection Process Roles

Producer: creator of the artifact to be inspected

Reviewers: will review the document; there should be at least three reviewers for formal inspections

Moderator: keeps the inspection meeting focused, can also be a reviewer

Timekeeper: can also be a reviewer, producer, or moderator

Pre-Inspection

- set expectations
- schedule inspection and provide agenda 48 hours prior to the meeting
 - recommendation: limit meeting to two hours
 - plan on at least three reviewers (including producer)
 - provide supporting documents
 - reviewers **must** review artifact prior the inspection meeting

Inspection Meeting for Documents

- moderator will lead inspection
- paragraphs or major sections should be called out

Inspection Meeting for Code

- moderator will lead inspection
 - line numbers can be called out
 - code can be read directly or paraphrased

Agree on reading style before inspection meeting

Reviewers make comments in turn

Producer will record each issue

- may formally address each issue after the inspection meeting
 - not all issues will be defects

Hints for success

- keep meeting within time constraints
 - various studies have shown that yield declines after two hours or too many Line of code reviewed per hour
- moderator will keep discussion on track
 - don't solve problems during the inspection meeting
 - raise issues
 - record issues
 - address them later

Stay focused and maintain time constraints

- avoid degenerating into a freeform discussion
- set expectations
- record relevant discussion such as
 - decisions
 - issues
 - points of contention
 - action items

Understand what software work product inspections are

Understand the cost of software work product inspections

Understand the benefits of software work product inspections

Know basically how to conduct software work product inspections

Hoover, C., Rosso-Llopart, M., Taran, G. *Evaluating Project Decisions: Case Studies in Software Engineering*. Boston, MA: Addison Wesley, 2009.

Pressman, R. *Software Engineering: A Practitioner's Approach*. McGraw-Hill Science/Engineering/Math; 7 edition (January 20, 2009).



Understand the challenges of security requirements engineering

Learn the importance of developing security requirements in the same timeframe as functional requirements.

Learn why the methods used to identify functional requirements may not work directly for security requirements.

Be exposed to methods for security risk analysis, security requirements elicitation, and security requirements prioritization.

Learn how to apply the SQUARE method for security requirements engineering.