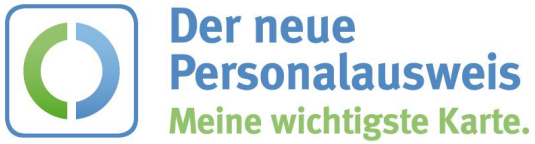
Secure Integration of the PersoApp-Open-Source-Library



Konstituierende Sitzung des Beirates
BMI, September 4, 2013

Dr. Thorsten Henkel

Fraunhofer SIT

Agenda

I. Security- and quality management measures of the PersoApp-Open-Source-Library

II. Integration of the PersoApp-Open-Source-Library

III. Secure Software Engineering – Recommondations for the development of secure software

QM PersoApp

- Detailed description of QM-measures and processes in first project phase:
  - ***Quality criterea***
  - ***Programming guidelines***
  - ***Development process***
  - ***Architecture concept***
  - Types of review
  - Review processes
  - Release process
  - …
- Integration of secure engineering measures:
  - Threat modeling
  - Code review/tests
  - Code analysis
  - Security architecture documentation
  - Risk analysis
  - …

Quality criteria

- Goal: Establish an Open-Source-Community und maintain software quality over whole project life cycle:

I. Compile set of quality criteria (based on ISO/IEC 9126) as basis for review processes
II. Compile set of metrics for measurement of software quality
III. Methods to validate criteria

Criteria	Description	Metric
Functionality	e.g. plattform independence, interoperability, …	e.g. compatibility to common windows platforms
Reliability	e.g. Number of Error/Bugs…	e.g. Number of new defects per month
Maintainability	e.g. efficiency of bug fixing	e.g. cyclomatic complexity per method
…	…	…

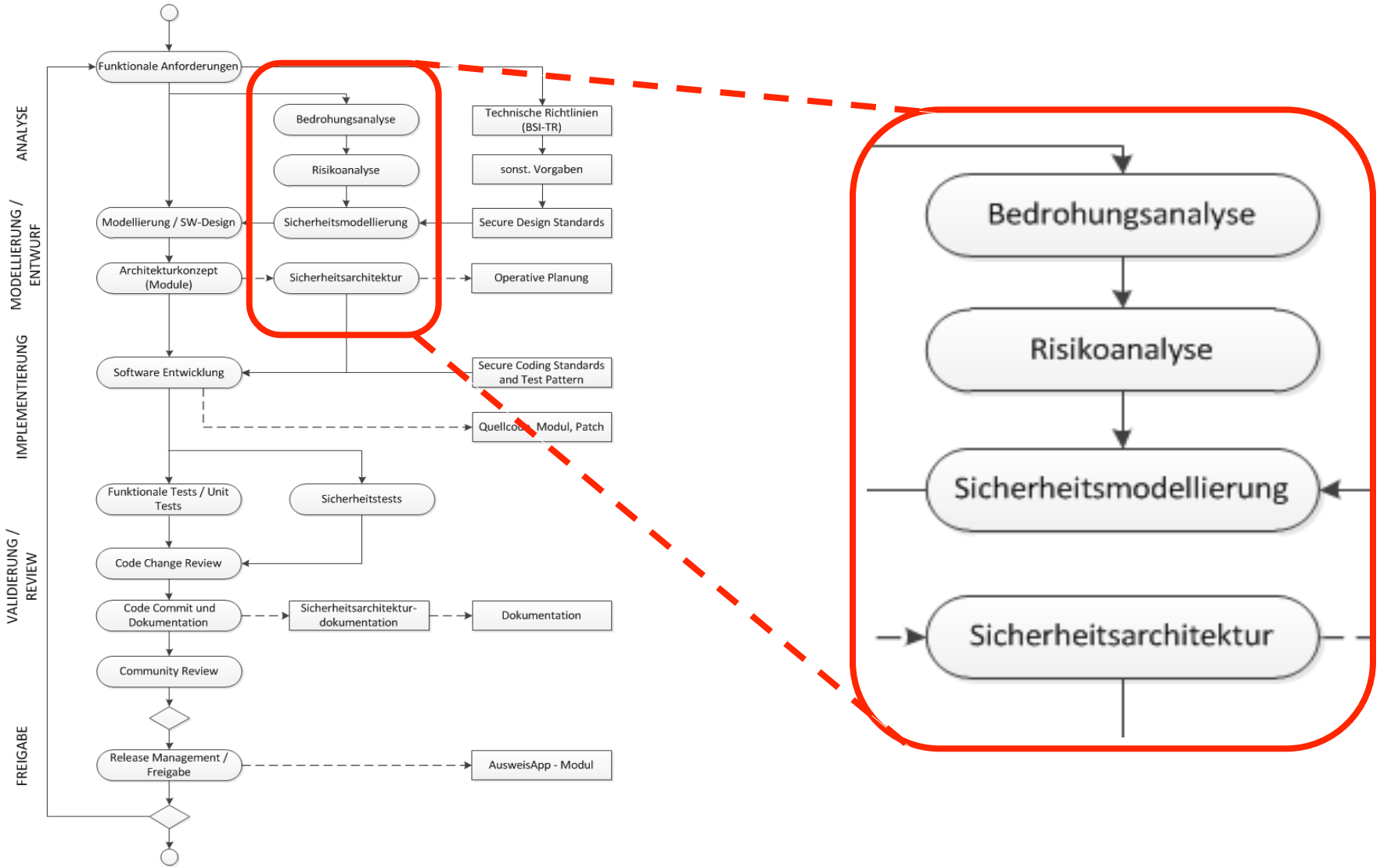
- Validation of criteria can mostly be automated by using analysis frameworks, e.g. SonarQube

Programming guidelines

- Common guidelines for the impolementation of secure and high-value software:
  - Syntax, naming guidelines, structure,
  - Used technologies and algorithms,
  - Usage of external libraries, e.g. cryptographic libraries

Guidline	Description	Example
Technische Richtlinien	Conformance to BSI Technisce Richtlinie	BSI-TR-03112 (eCard API Framework), BSI-TR-03130 (eID-Server), ..
Source code format	Maintain readability of code even with many contributors	Copyright comment at top of each Class
Comment conventions	Comprehensibility of code and desicions by other contributors	Comments shall supply information not directly readable from sole code
Usage of external libraries	Quality assessment of external libraries	Assessment checklist, e.g. • Existing vulnerabilites in public databases? • Is the security architecture well documented? • …
…	…	…

Development process (1)



Development process (2)

- Threat and risk analysis:

Activity	Description	Process task
Description of system users	Description of used funcionality and access control	Threat and risk analysis
Description of stakeholders	Listing of authorized stakeholders for identification of assets in later step	Threat and risk analysis
Description of assets	Listing of relevant assets	risk analysis
Security goals for assets	Which security goals must be reached for each asset?	risk analysis
…	…	…

- Security modeling
  - Description of system architecture, target platform(s) and technologies
  - Listing of relevant vulnerabilities from vulnerability databases
  - Attacker model
  - Data flow diagrams
  - …

Architecture concept (1)

1. Architecture of the software (modules and interfaces)

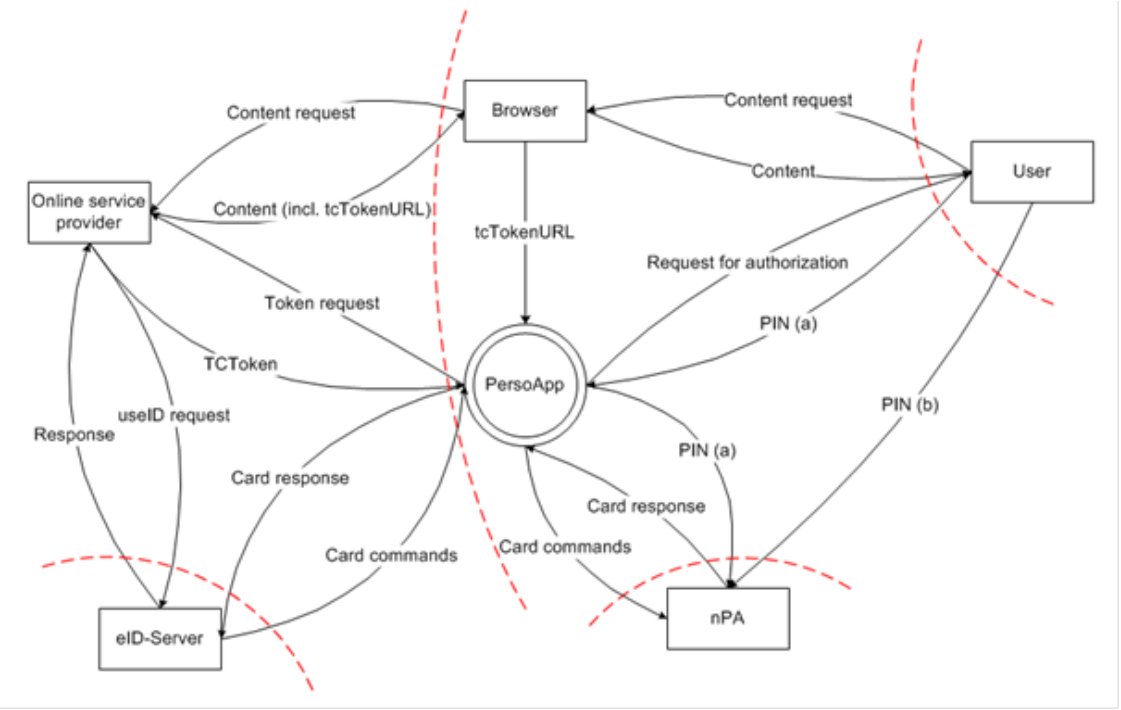
- Platform independent by using Java

- Portability of the core by modularization:
  - Protocol core
  - GUI
  - Hardware abstraction layer (HAL)

Architecture concept (2)

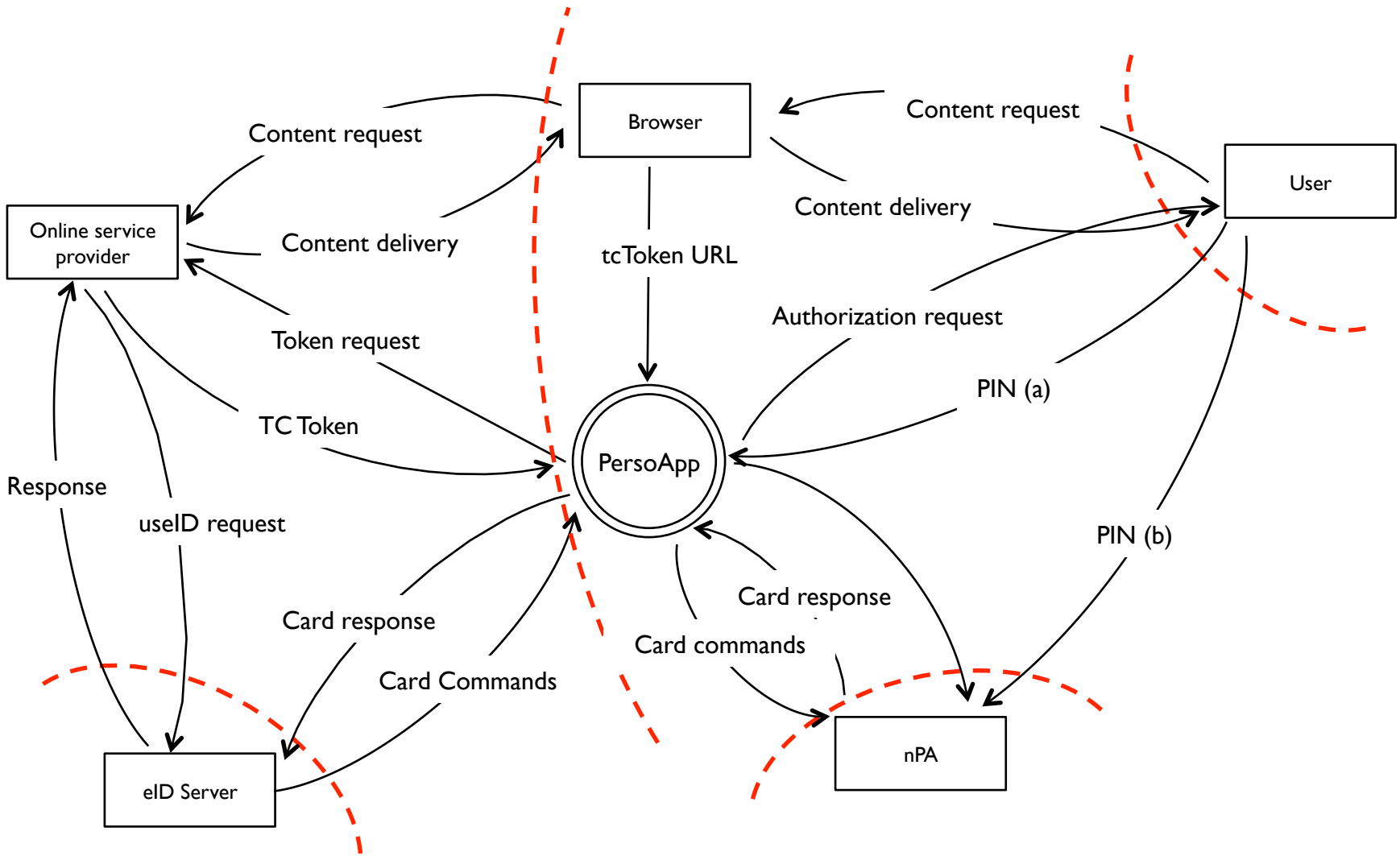
2. Security requirements

- Application context in form of DFD



- Definition of security goals, e.g.:
  - Confidentiality of PIN
  - Confidentiality and integrity of personal data

Architecture concept (2)



Integration of PersoApp-Lib (I)

- PersoApp-OS-Library is more than source code and binaries:
  - Extensive documentation on QM-measures
  - Security architecture documentation

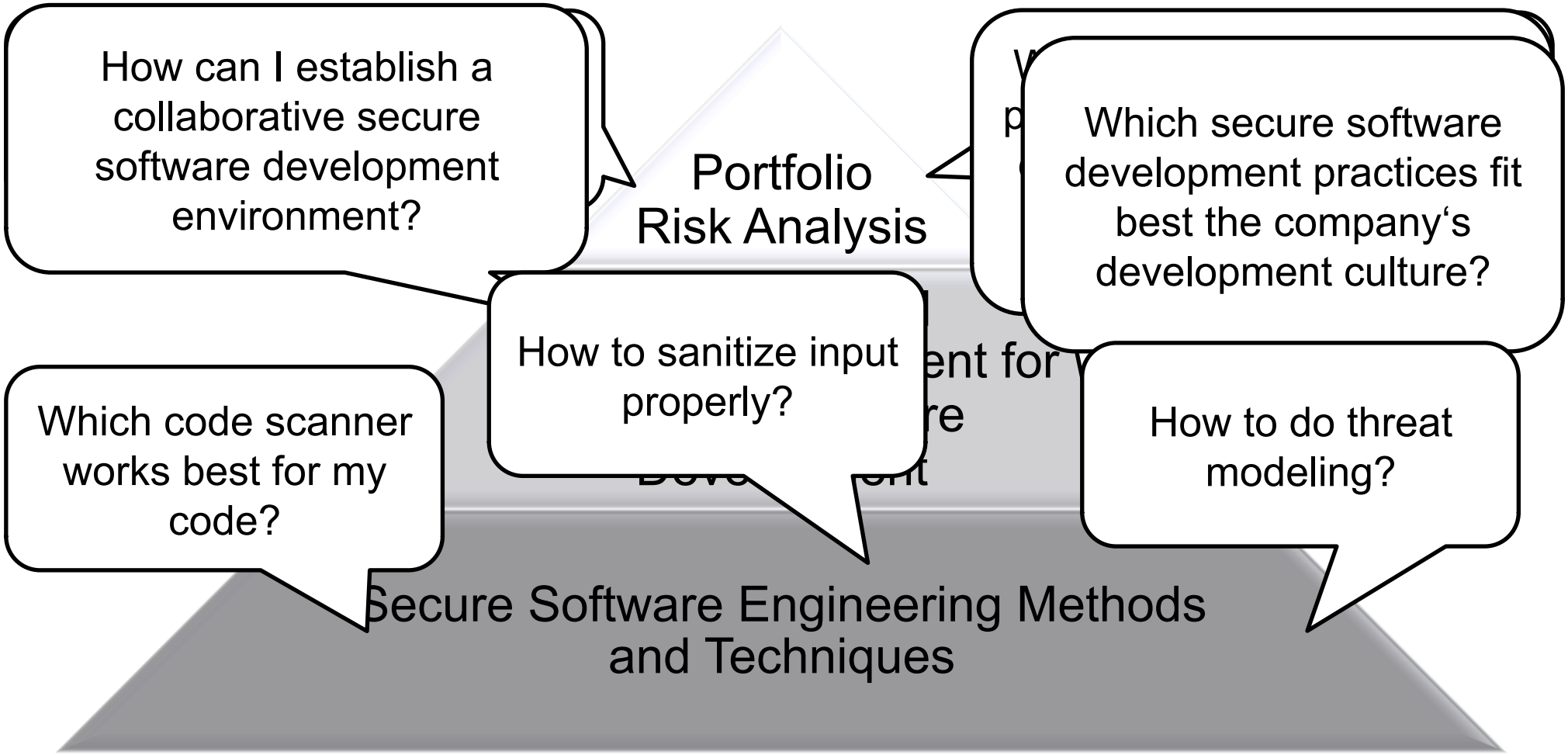
- QM-measures allow for transparent view on design decision made by PersoApp-OS-Library team

Integration of PersoApp-Lib (II)

- Security of an application using PersoApp-OS-lib is not depending very much on the PersoApp-OS-lib
- Decisions on security and quality need to be taken for the integrating application
  - Considering the application‘s usage scenario, stakeholders and assets
- PersoApp-OS-library is just one of potentially many 3rd-party-components of an application, just as e.g. logging-, GUI- und framework-libraries
- Secure Software Engineering processes must be established for the application as well

Secure Software Engineering

Three areas of action for secure software development



Overview

1. Adopting FAIR for software portfolio risk asssement

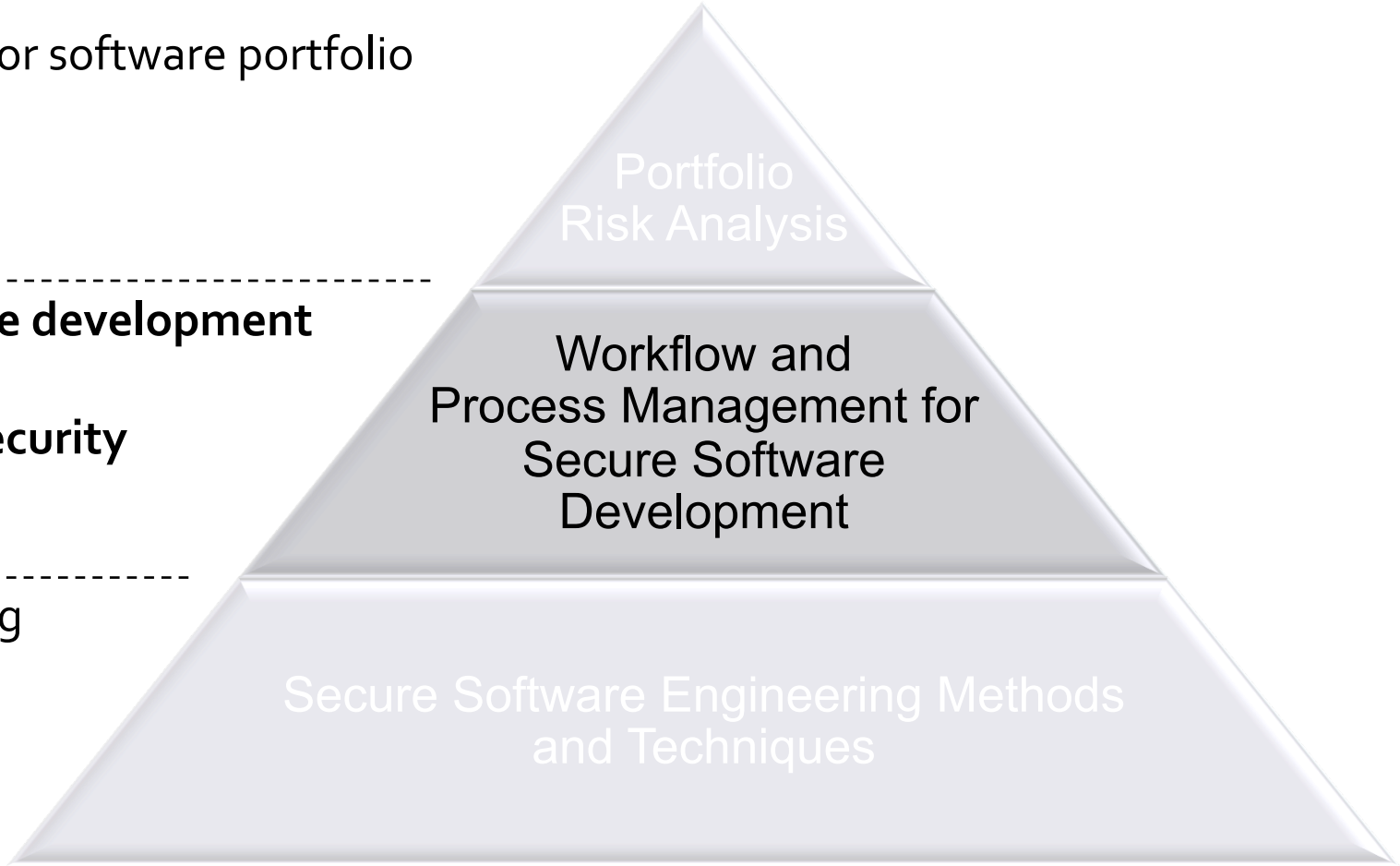
---

2. Secure software development landscape

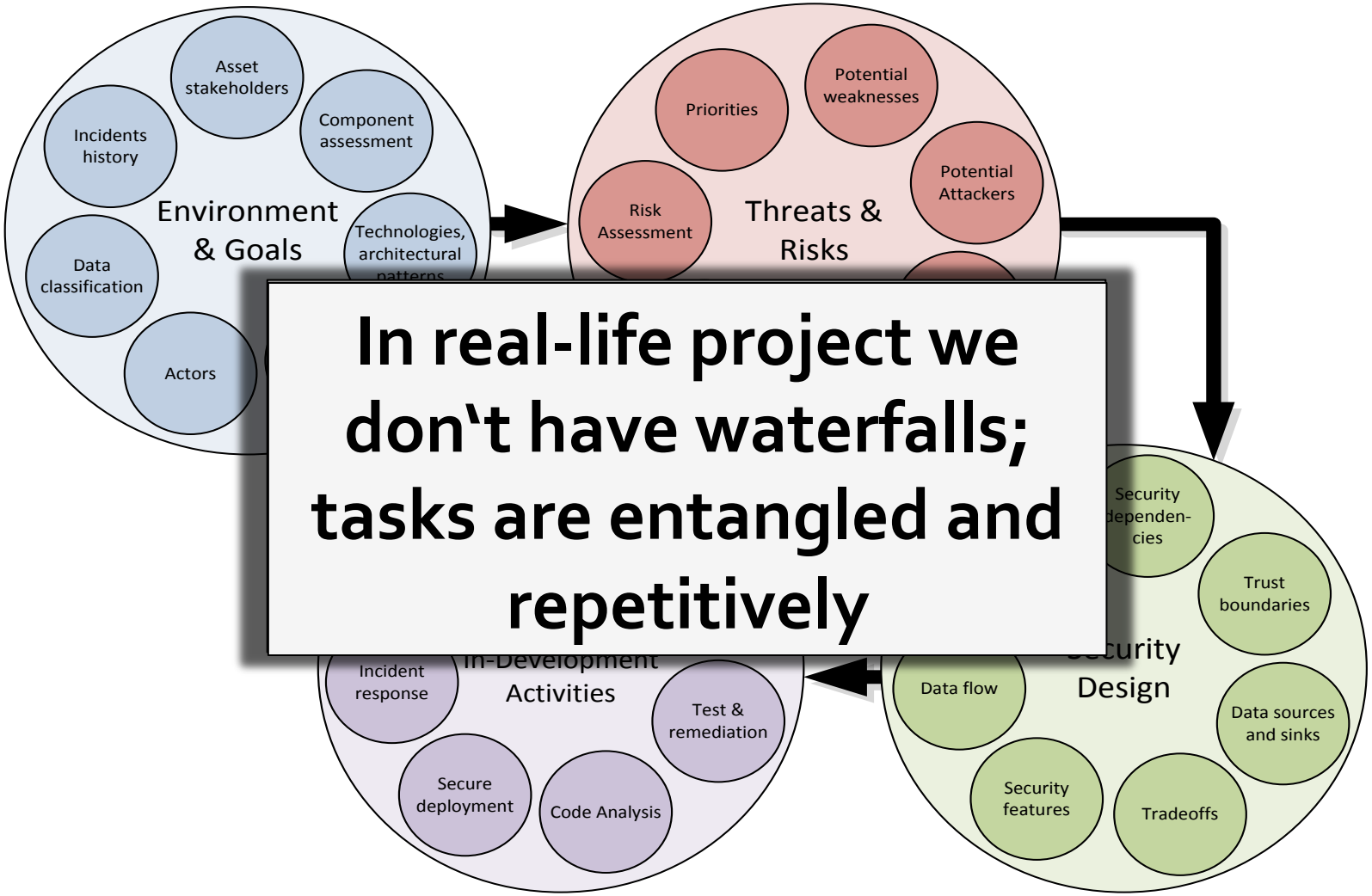
3. Supply chain security

---

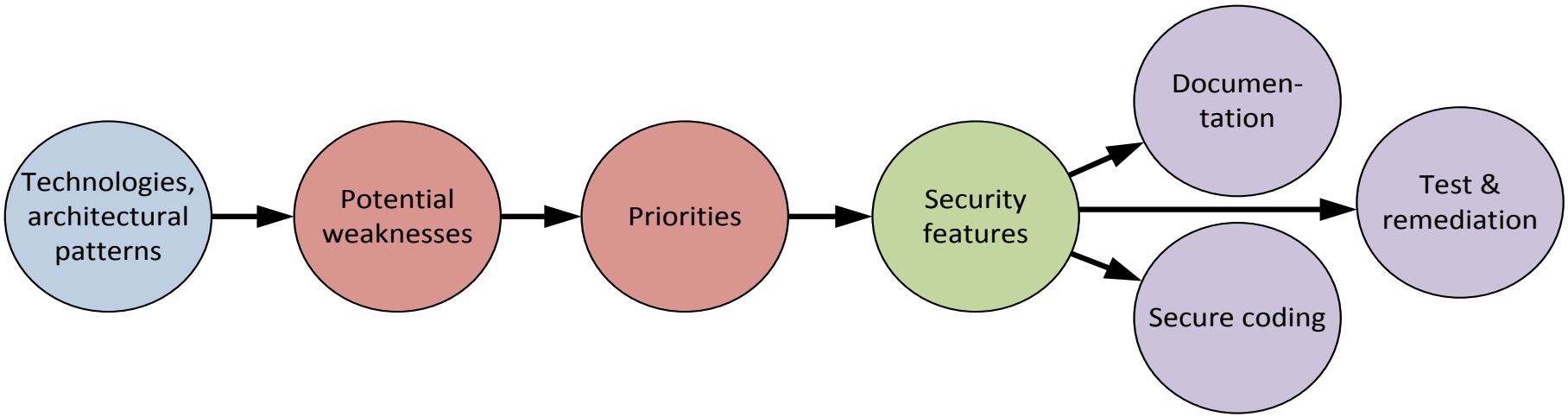
4. Threat modeling techniques
5. Code analysis



Secure Software Development Landscape



Developing a Lightweight Process for Preventing Common Weaknesses



Represents one appropriate composition of tasks that fits specific vendor requirements but can be extended later on if required.

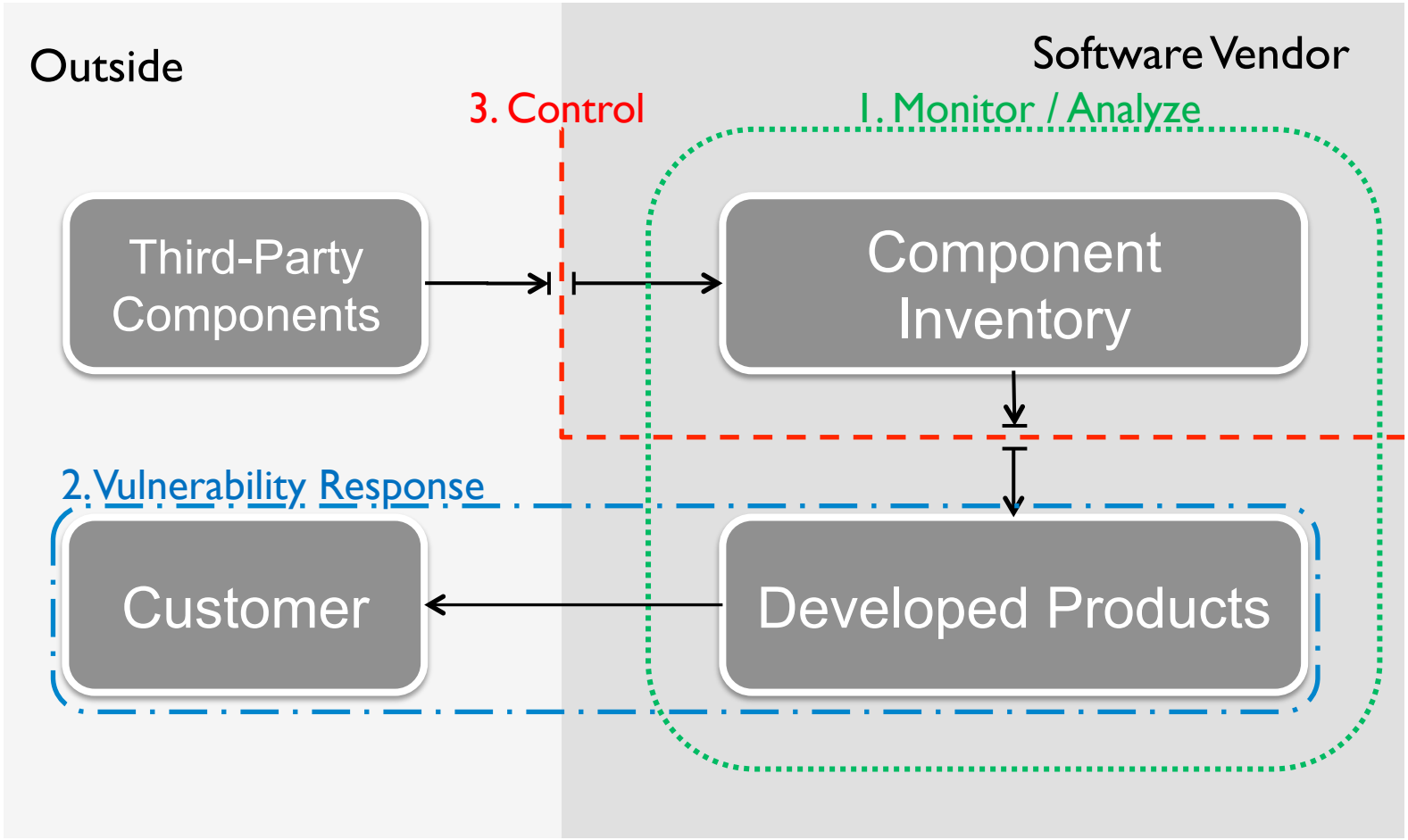
Managing Supply Chain Security

Enhancing Information Flows in Software-Developing Organizations

- **Problem:** Security issues result from information gaps along the software supply chain
  - Developers require **security-related information about the components** they integrate, e.g., safe function handling and configuration, security and trust assumptions that needs to be fulfilled
  - **How trustworthy are integrated third-party components** , e.g., open source components?
  - On the other hand, **component developers require security requirements from the integration scenarios**; e.g., middleware developer require information in which solutions the middleware is used
- **Goal:** Foster upstream and downstream information flow between developers of components, products and solutions.

Approaches to Supply Chain Security

Employing available non-security clearinghouse databases by enhancing internal workflows



Overview

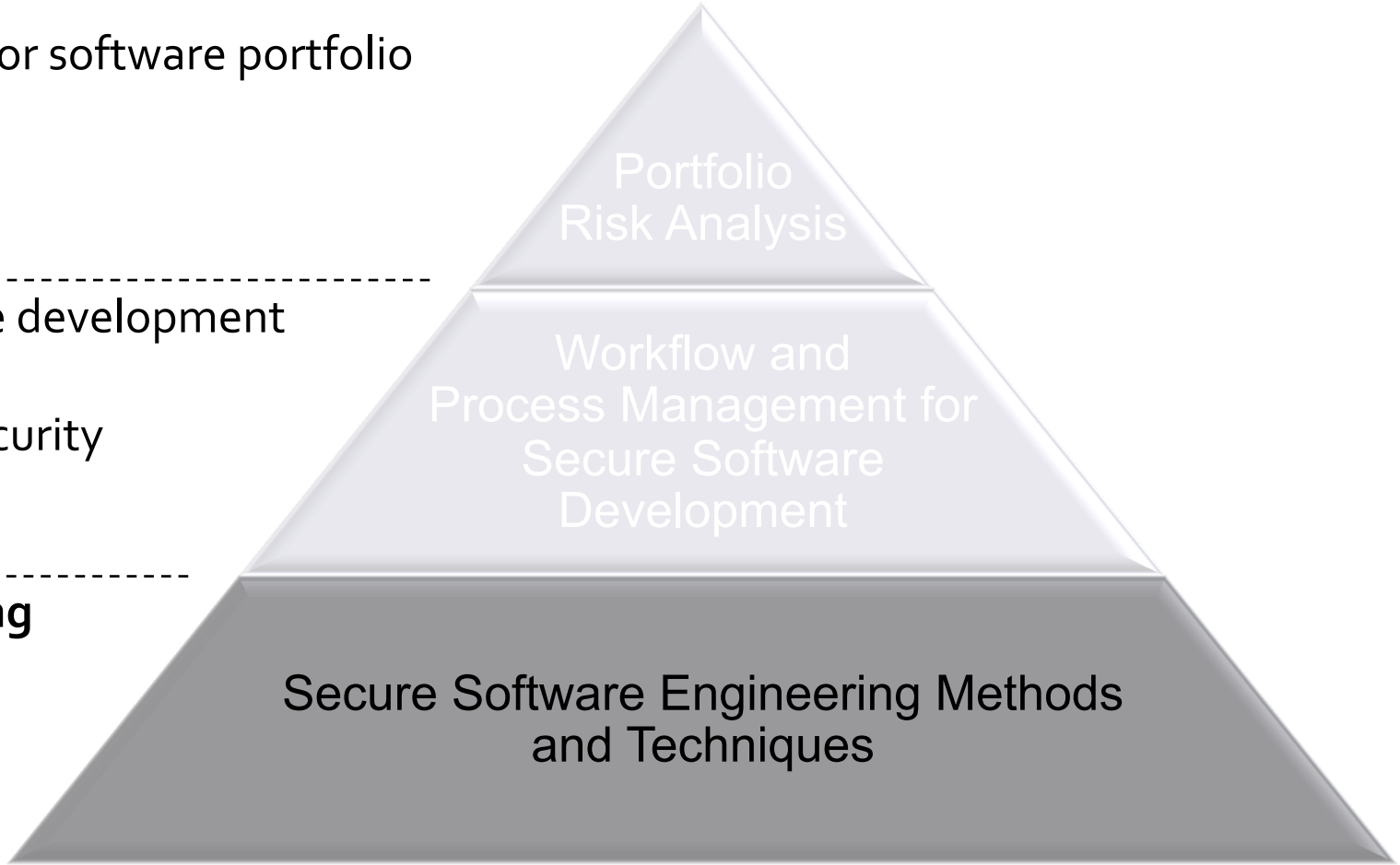
1. Adopting FAIR for software portfolio risk asssement

---

2. Secure software development landscape
3. Supply chain security

---

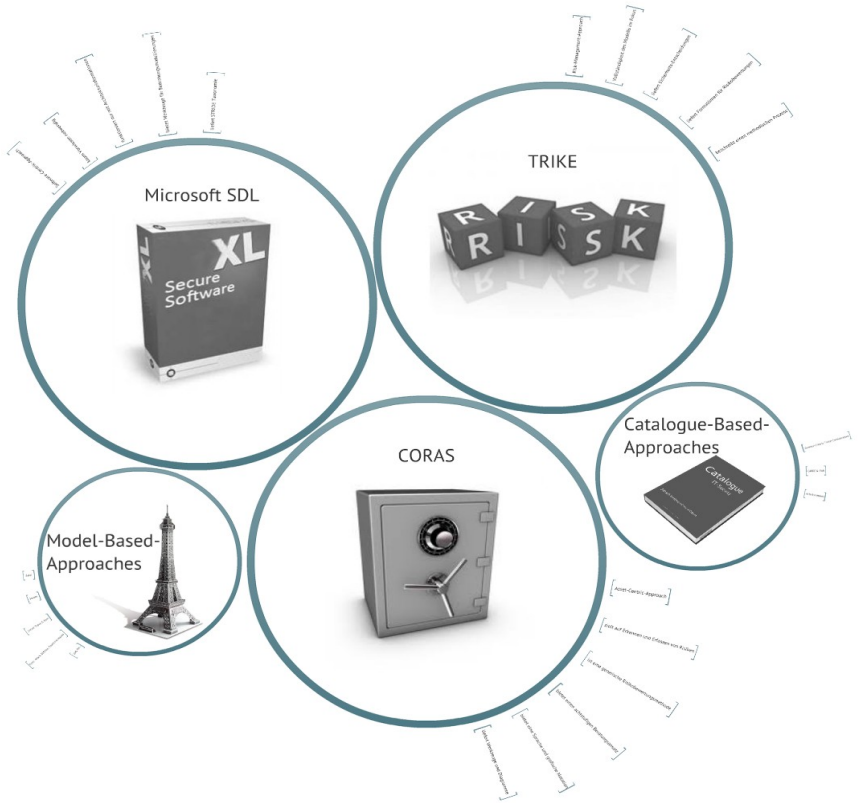
4. Threat modeling techniques
5. Code analysis



Threat Modeling Techniques

State of the Art

Approaches & Methodologies analyzed, evaluated and empirically tested by SIT SSE Department



MS- SDL Security Analysis
- software centric approach
- provides tools & taxonomies (STRIDE)

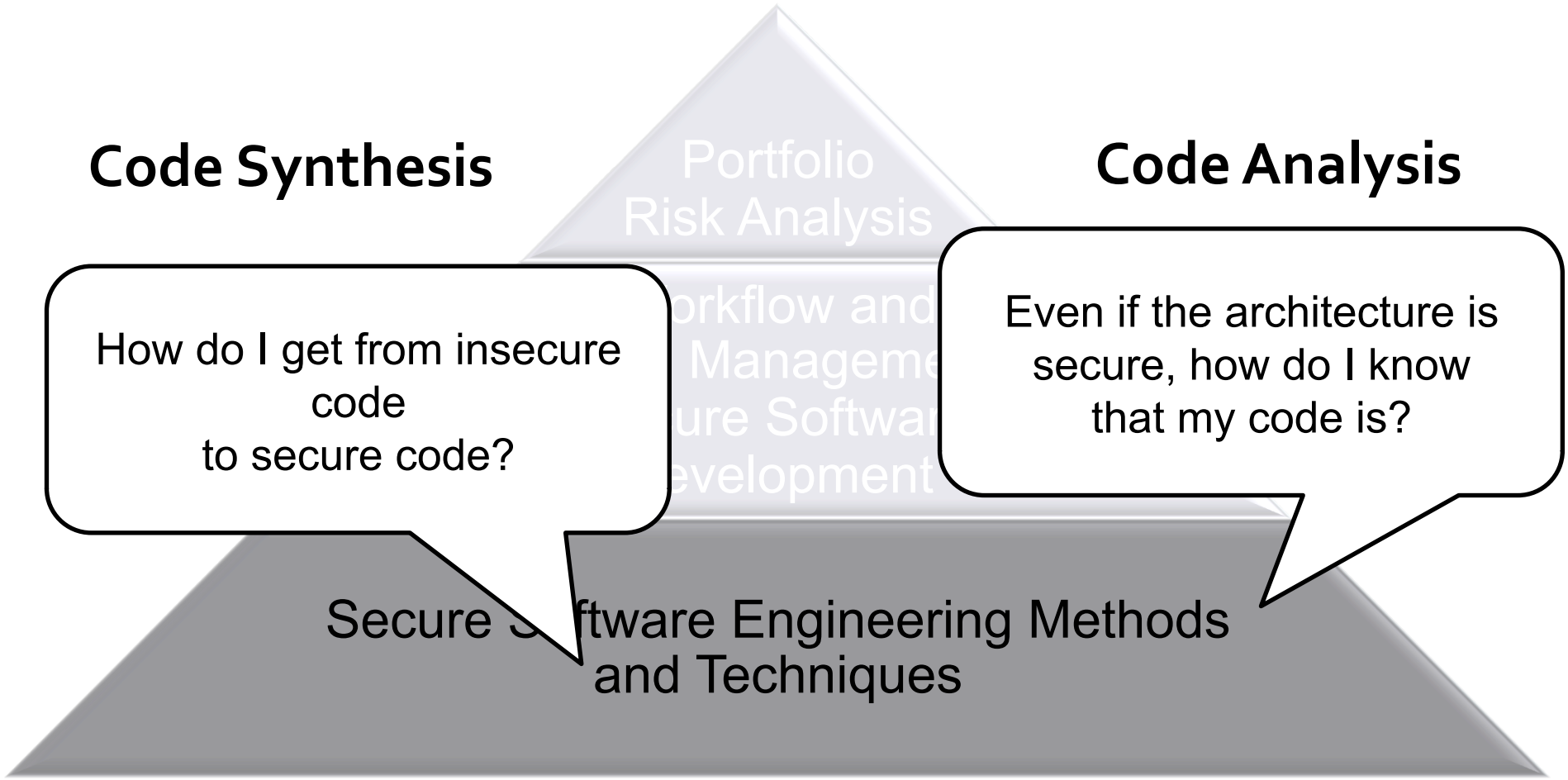
TRIKE
- risk management approach
- provides formalized risk-evaluations

CORAS
- asset centric approach
- provides a own language and graphical notation

UML-Sec, Attack-Trees, etc.
- Model-based approaches

CC-Threat Considerations, CWE, etc.
- Catalogue-based approaches

Secure Software Engineering through Code Analysis & Synthesis



General Methodology

- Analyze as much as possible ahead of runtime

- Use results to:
  - Find implementation faults
  - Generate installation/configuration requirements for secure deployment
  - Optimize runtime checks to assure secure execution
  - Report general security metrics

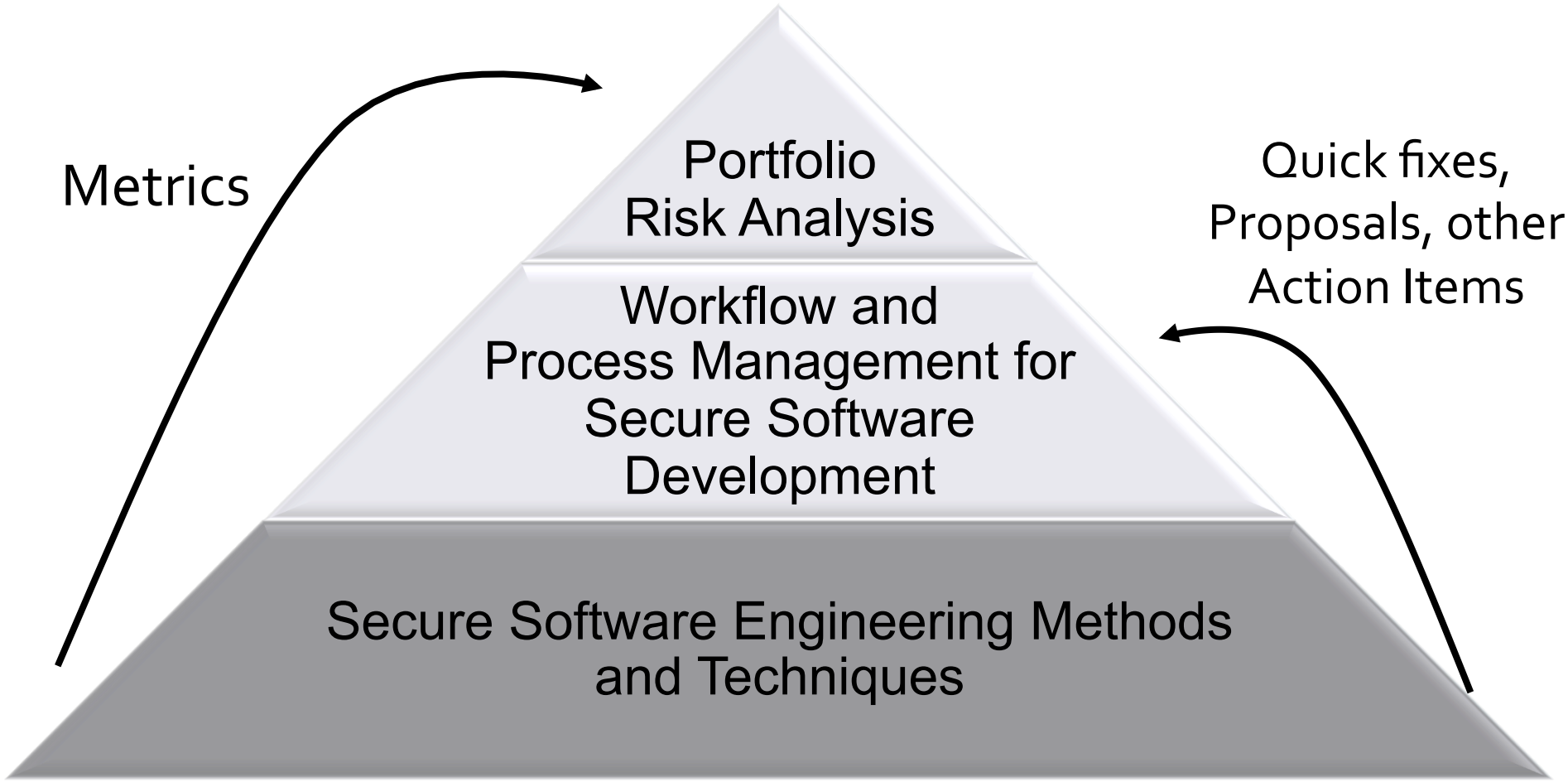
Code Analysis

- Constantly analyze library or software product to find errors, acquire metrics and deduce software's attack surface
  - "Manual" Code Reviews
  - Static Code Analysis (Veracode, IBM Appscan, findbugs, etc.)
  - Traffic analysis (wireshark, etc.)

- Document analysis results and measures/decisions taken
  - Bug Tracking
  - Security architecture documentation

- Derive installation/configuration manual to make sure attack surface is closed

Secure Software Engineering through Code Analysis & Synthesis



Conclusion

- PersoApp-OS-Library is extensively tested to ensure software quality
- Software integrating persoapp-os-library is not necessarily secure
- PersoApp-os-lib is just one of many 3rd-party-components for an integrating application
- SSE-measures can enhance software security throughout the whole development lifecycle