THE UNIVERSITY OF BRITISH COLUMBIA

# Towards Agile Security Assurance

Konstantin Beznosov & Philippe Kruchten
University of British Columbia

# Outline

- Problem
- Contribution
- Conventional assurance & agile methods
- Solution
- Summary

# Problem

**Mismatch** between

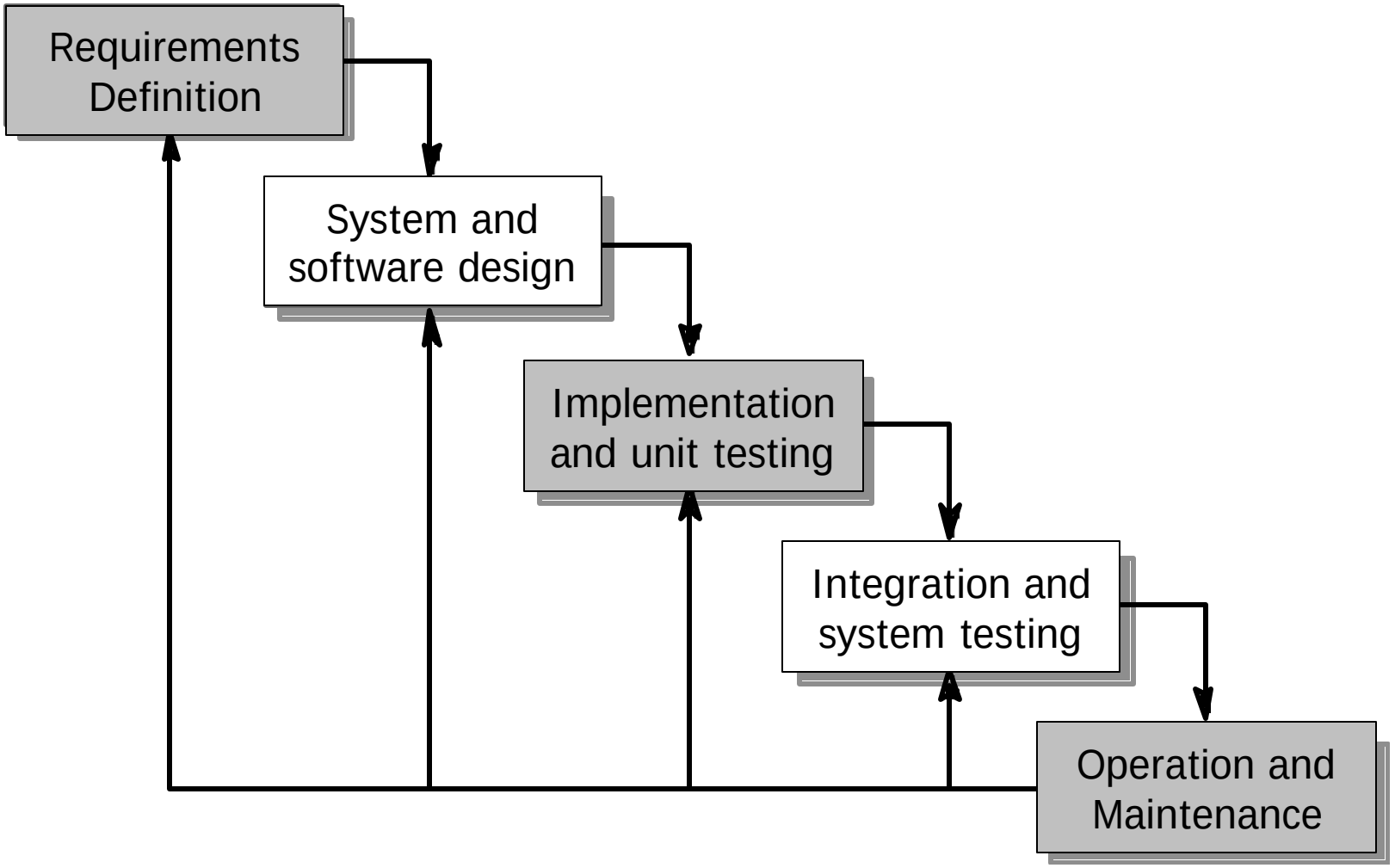
- **agile methodologies** for software development
- **conventional methods for security assurance**

**Hard to assure** with agile development

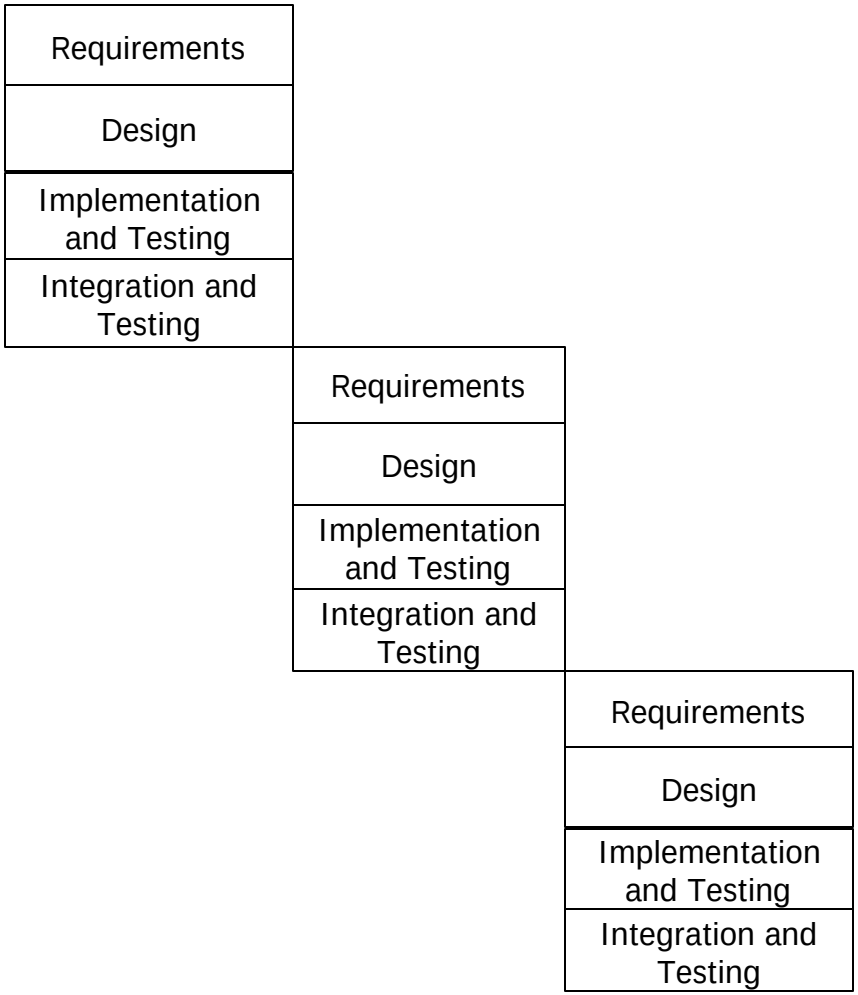
# Contribution

1. **examine** the **mismatch** between agile and security assurance methods
2. **classify** conventional security **assurance** depending on the **degree of clash**
3. suggest **ways** of **alleviating** the **conflict**

# What's Waterfall Development?

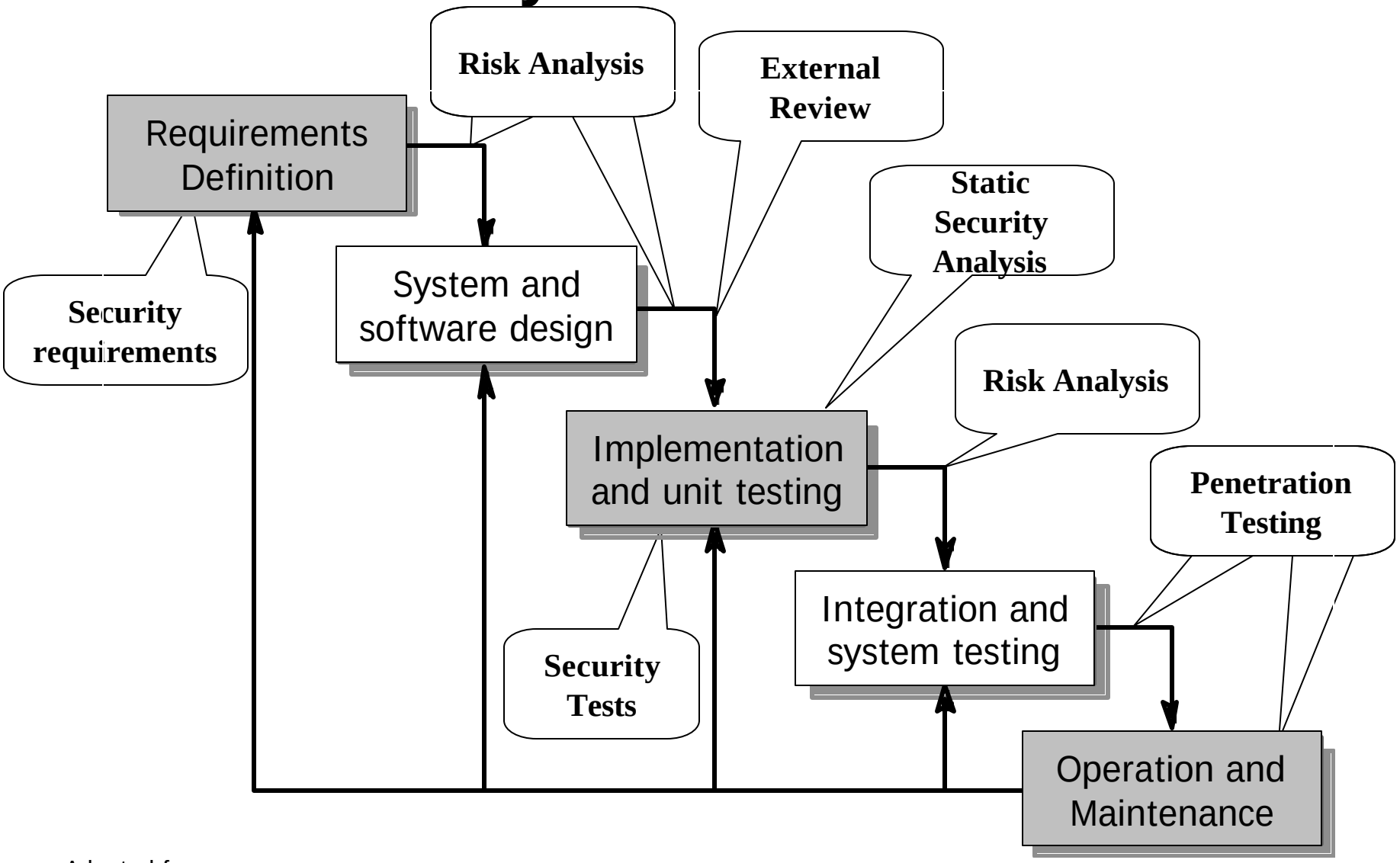


# What's Agile Development?




- Characteristics
  - Iterative lifecycle
  - Requirements and design emergence
  - Direct communication
  - Tacit knowledge
- Sample methodologies
  - Crystal
  - Adaptive Development
  - Feature-driven Development
  - Scrum
  - Lean Software Development
  - XP

# What's Conventional Security Assurance About?




Adapted from

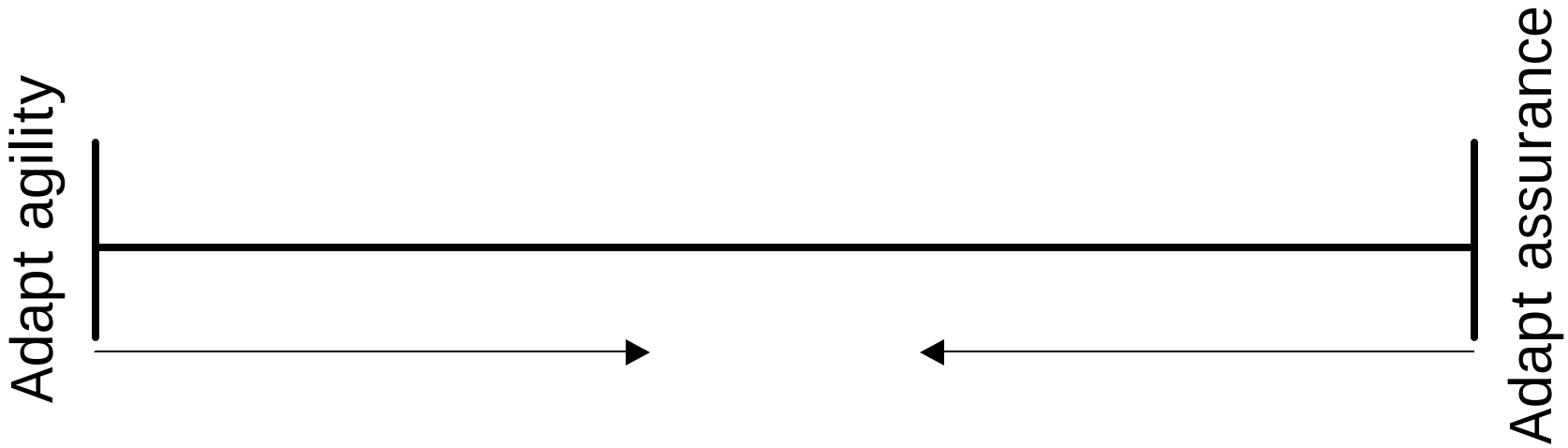
D. Verdon and G. McGraw, "Risk analysis in software design," *IEEE Security & Privacy*, vol. 2, no. 4, 2004, pp. 79-84.

# **Why** is addressing the mismatch **important?**

- More security-critical software
- Agile methods are there to stay

# Solution(s)?

If the mountain will not go to Mahomet,
let Mahomet go to the mountain. (proverb)



# Examination Results

Assurance relies on **third party**

- reviews
- evaluation
- testing

Points of clash

1. **direct** communication and **tacit** knowledge
2. **iterative** lifecycle
3. design **refactoring**
4. **testing** “philosophy”

# (Mis)match Classification

1. Natural **Match**

   e.g., pair programming ♥ internal review & coding standards

2. Methodology-**neutral**

   e.g., language (e.g., Java, C# vs. C, C++),
   version control and change tracking

3. Can be (semi-)**automated**

   e.g., code static analysis,
   security testing/scanning

4. **Mismatch** (˜ 50%)

   e.g., external review, analysis,
   testing, validation change authorization

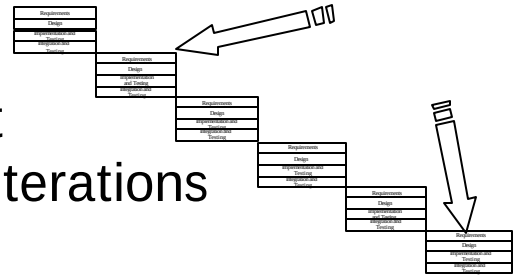
# Alleviating the Mismatch

For (semi)-automatable

- **Increase** acceptance through **tools**
- **Codify** security **knowledge** in tools
  - automated fault injection, test generation

For mismatching

- Search for new **agile-friendly assurance** methods
  - **direct** communication and **tacit** knowledge
  - **iterative** lifecycle
  - design **refactoring**
  - **testing** “philosophy”
- **Intermittent** assurance
  - apply at the first and last **iterations**
  - use the results to “**align**” the development
  - Have a **security engineer** involved in all iterations (Wäyrynen et al. 2004)



# Summary

## Problem

mismatch between agile development & security assurance

## Contributions

1. **Examine** (pain points)
2. **Classify** assurance methods
3. **Alleviate** (tools, knowledge codification, new methods research, intermittent assurance)