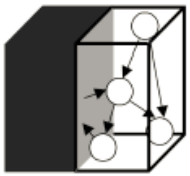
EXIGEN services

Return on Intelligence company

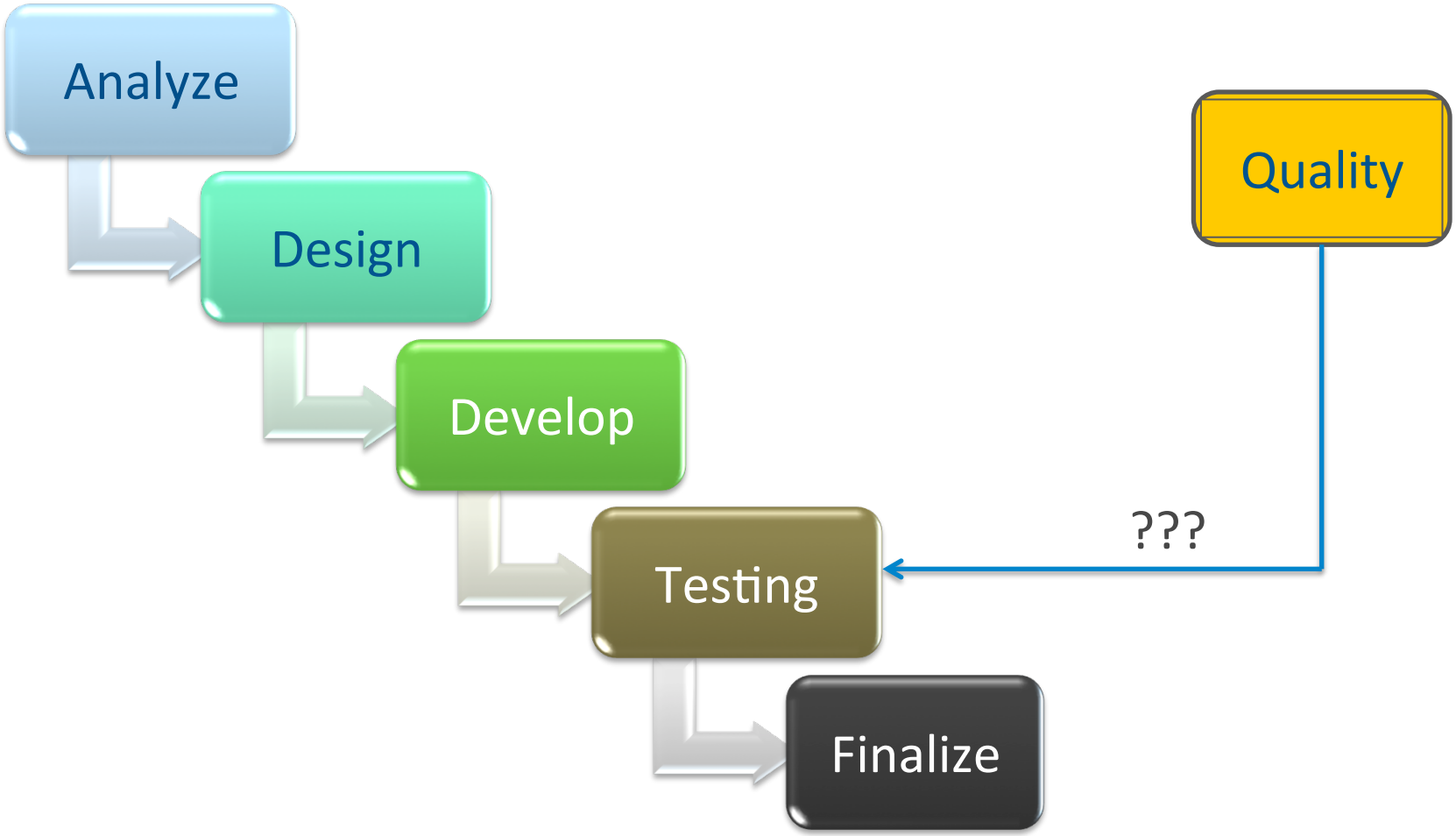
Application Threat Modelling

Ainārs Galvāns
Security Tester
Exigen Services Latvia

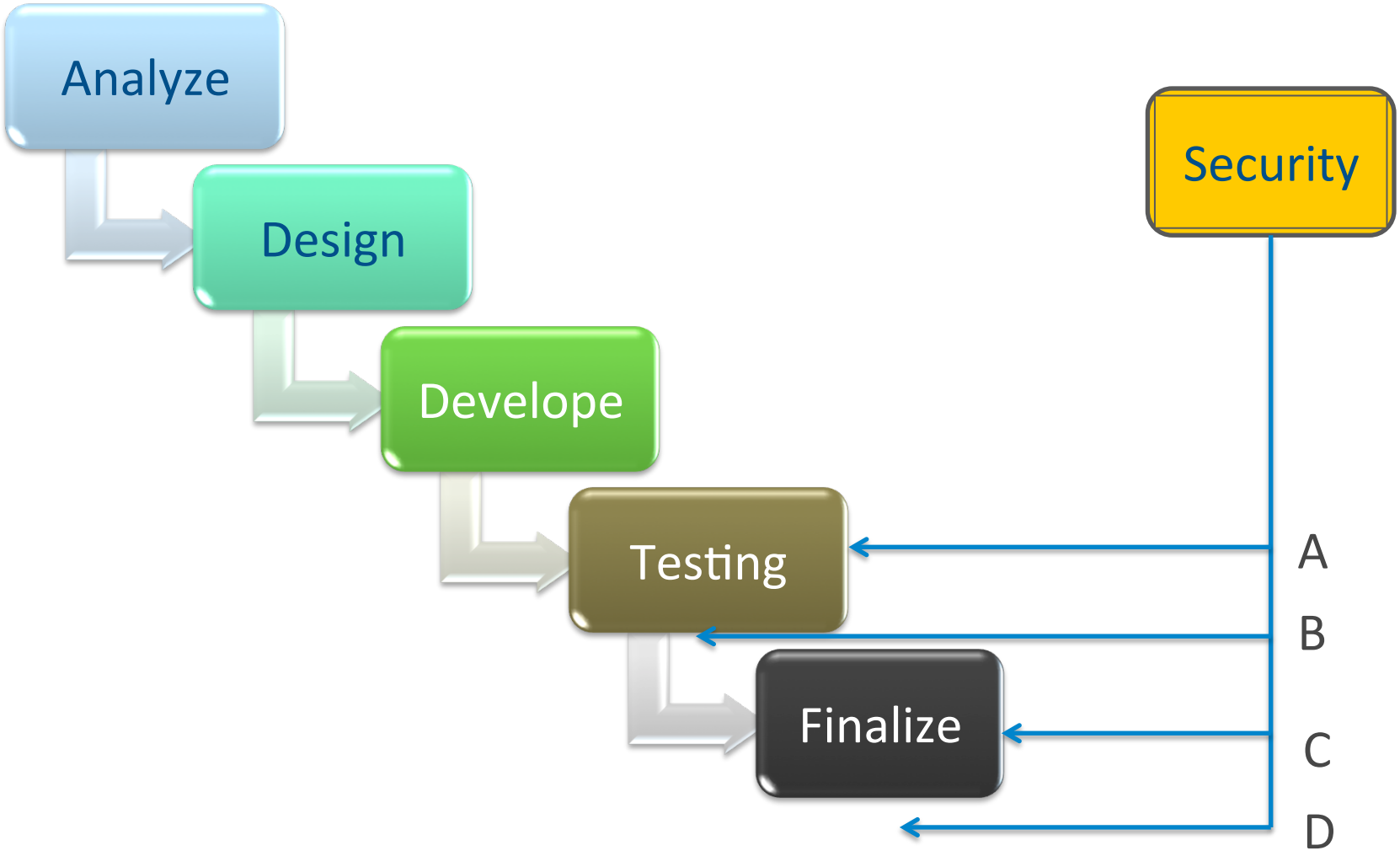


September 25, 2014

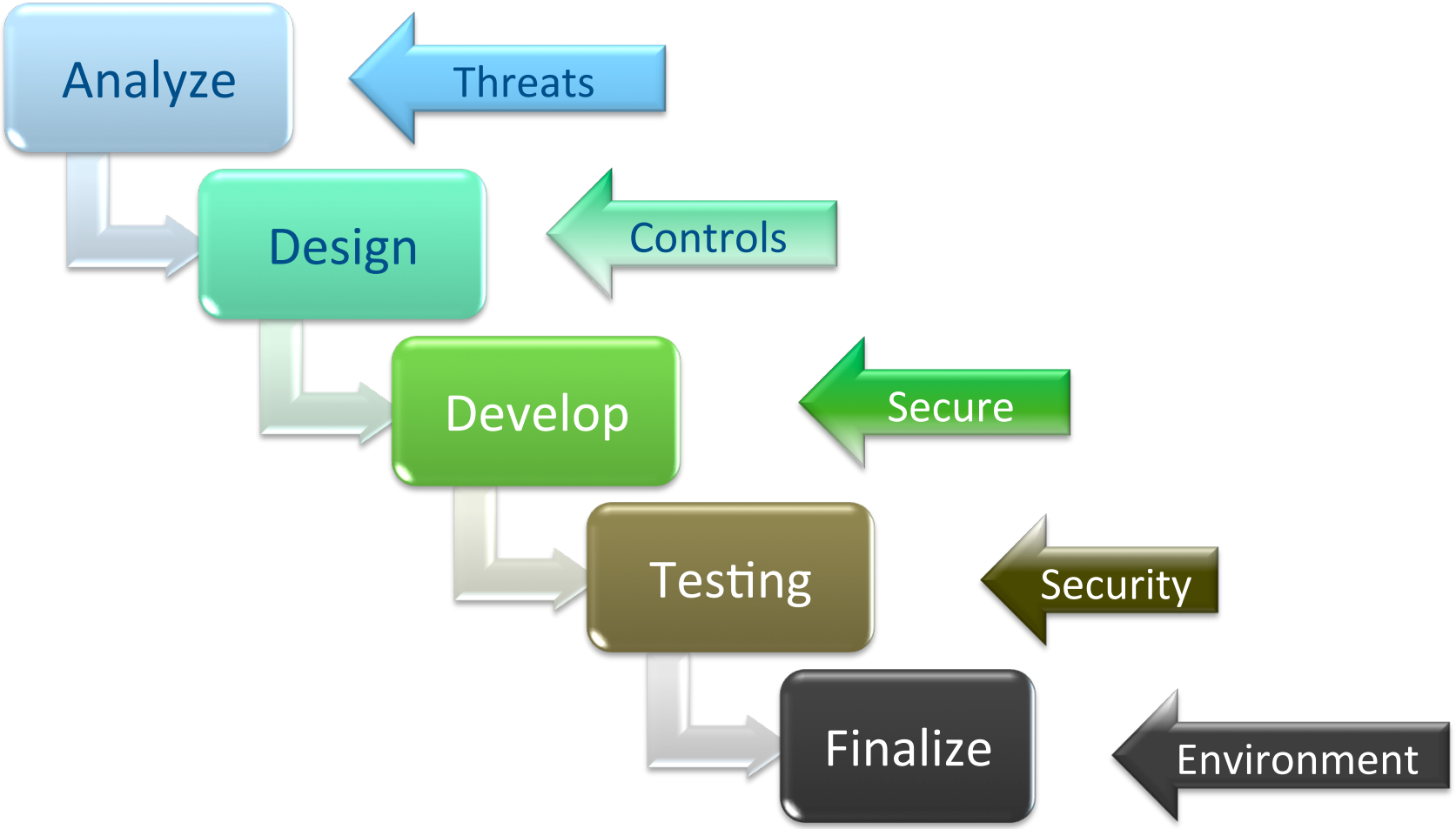
Can we test quality in?



Can we test security in?



When should we test scurity



References

- OWASP application threat modelling
  - Available under a Creative Commons 3.0 License
- Microsoft Secure Development Live Cycle
  - Threat modelling: visio based tool
- Other alternatives
  - cigital: Mobile Application Threat Modeling
  - Various: Threat Modeling as service provided

OWASP Application *Threat Modeling*

Step 1

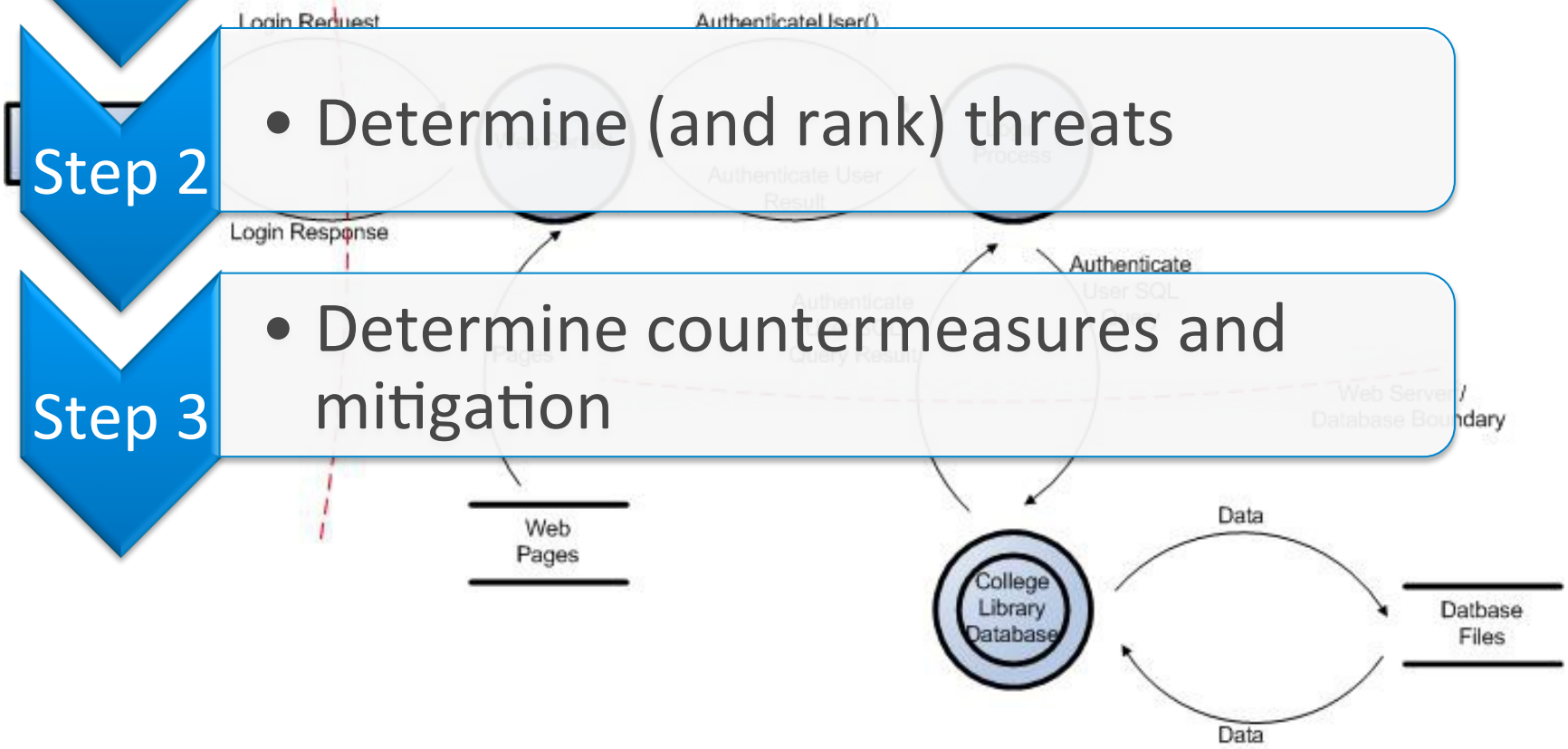
- Decompose the application

Step 2

- Determine (and rank) threats

Step 3

- Determine countermeasures and mitigation



Read more at

https://www.**owasp.org**/index.php/Application_Threat_Modeling

QUESTIONS?

How I extended OWASP threat modelling recommendations?

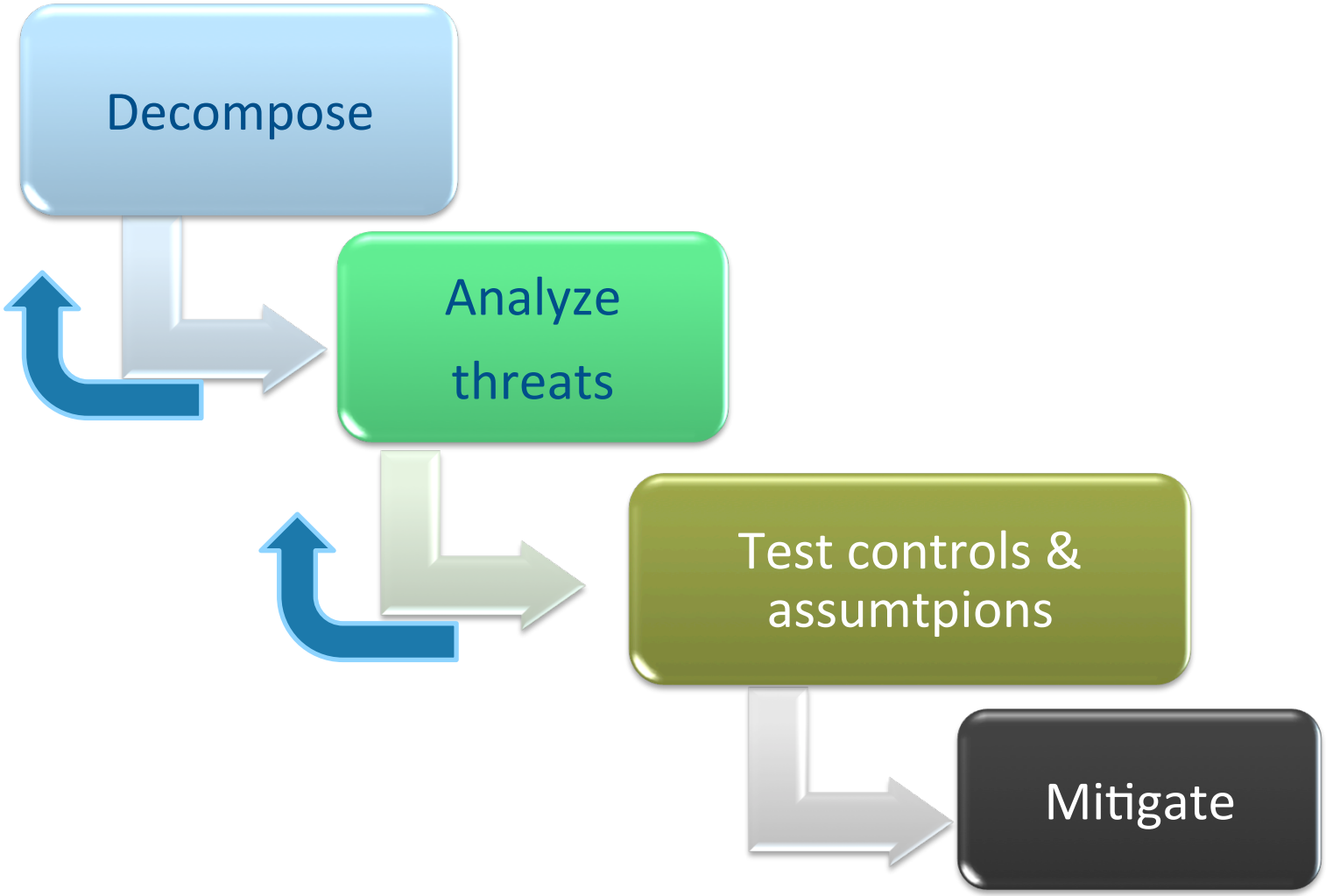
ADAPTATION NOTES

Content

- 4 step process
  - Decomposition
  - Determine threats
  - Test
  - Analyze results
- Lessons learned

- Appendix

4 step process



Applicaiton Decomposition is an art

- OWASP gives an example
  - Data entry/exit points
  - Assets (all types of)
  - Trust levels
- I tailor decomposition for each project individually
  - Read funct. requirements
  - Scan interfaces
  - Talk with architect
  - etc.

Determine threats: STRIDE checklist

Type	Security Control
Spoofing	Authentication
Tampering	Data validation and encoding
Repudiation	Event Logging
Information disclosure	Encryption and authorization
Denial of service	Authorization, filtering, etc.
Elevation of privilege	Conequent authorization (every request)

My STRIDE countermeasures

STRIDE type	My countermeasure
Spoofing	Black box testing
Tampering	• Educate developers • Test for injections and XSS
Repudiation	Implement proper persistent logging and auditing
Information disclosure	• HTTPS solve most of the problems • CAPTCHA and lockouts solves the rest
Denial of service	Blackbox test any long running function
Elevation of privilege	Implement privilege check on every request/URL

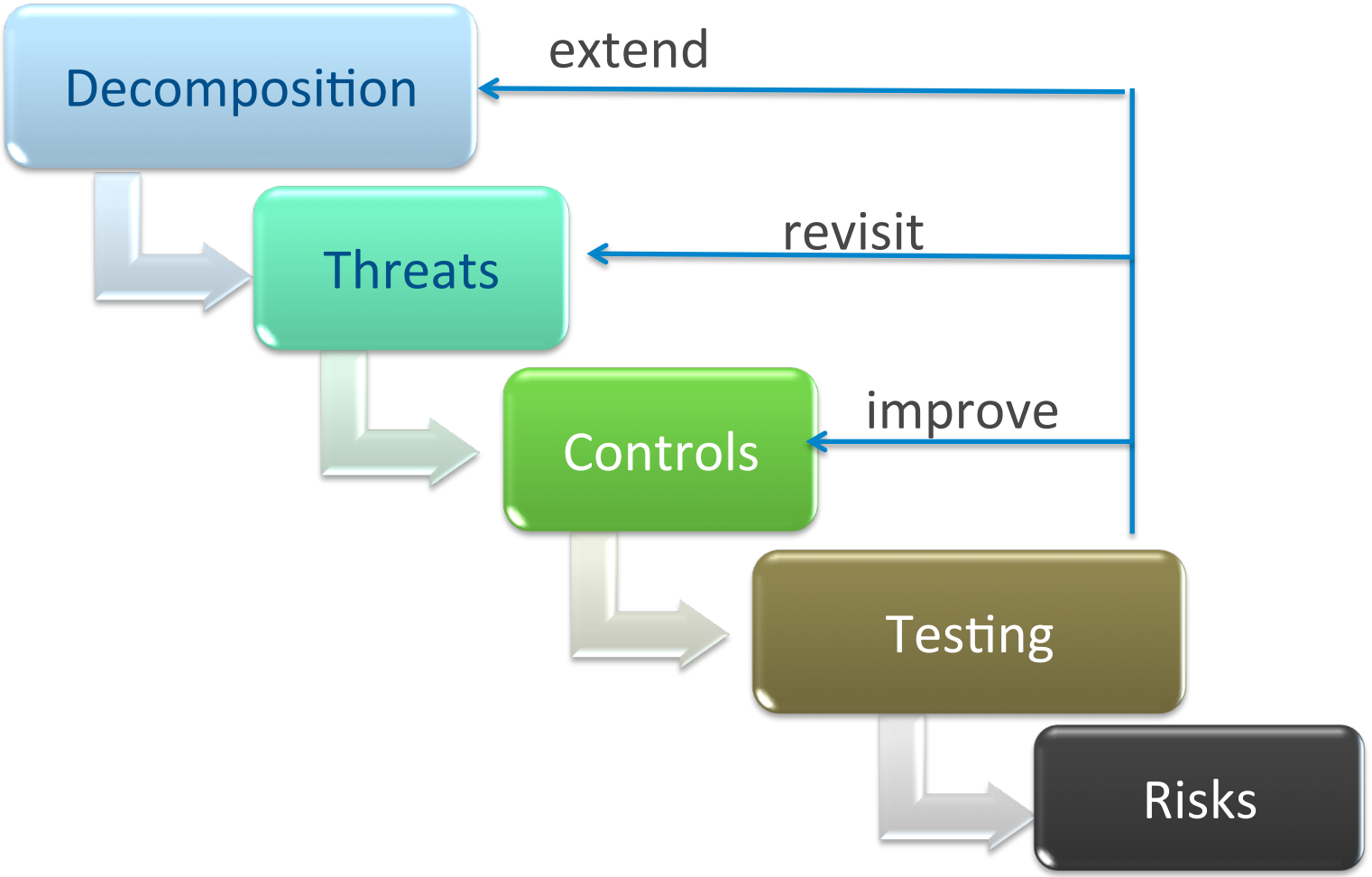
Test application to validate your model

- Do a black box testing on all interfaces
  - Add threats missed in modelling
  - Add interfaces missed
- Test each security control
  - Test may be simpler than code review
  - Testing may discover default controls

Analyze and rank outstanding risks

- Testing provides a list of «bugs», including:
  - How easy it is to discover and use the security hole?
  - What are the business consequences possible?
- Unfixed issue mitigation :
  - **Inform about the risk:** for example educate end users
  - **Plan a fix**: agree to postpone
  - **Accept the risk:** the risk is too low
    (i.e. Security hole could only be used by “insider”, i.e. back-office user)

Summary: security testing cycle



LESSONS LEARNED

Generic Observations

Method extensions and results

Testing improves threat understanding

- Validates control effectivity and rank risks
- Discovers additional (unforeseen) threats

Meetings with development team improves model

- Team makes good threat brainstorming
- Developer's input could reduce test scope

Black Box tests may show team's wrong assumptions

- Wrong usage of a prebuilt security controls

More than penetration testing

	Penetration testing	Threat modelling
When	After code freeze	Through the SDLC
Goal	Discover *technical* security "holes"	Prevent *business* threats
Who	Security expert alone	Whole team, led by security expert
+	Less effort from team	Find, prevent issues early Find only important issues
-	Late discovered bugs Unimportant issues Miss complicated issues	Requires team education Effort through the project Miss unimportant issues

QUESTIONS?

Contact:

Ainārs Galvāns

Security Tester, Exigen Services Latvia

ainars.galvans@exigenservices.com

Eizensteina iela 29a | Riga, LV-1079, Latvia

phone +371 6707 2976 | mobile +371 2943 2698

www.exigenservices.lv

Threate model example based on real application model

APPENDIX

Threat Analysis: after a meeting with developers

Exit points	Spoofing	Data Tampering	Repudiation	Information disclosure	Denial of service	Elevation of privilege
WEBSERVISS	A&A	T?	Audits	HTTPS	L?	L!
WebApp	A&A	T?	Audits	HTTPS	L?	A&A
(public) Portal	A&A	N/A	L!	HTTPS	H?	N/A
WebApp: admin			L!			
WebApp: user						A&A
WebApp: legacy pages		A?				
WebApp: Ajax Calls		L!				L!
WS: attachments		H?				

Controls

Code	Description of a control or it's absence implications
H?	No know controls, high risk
T?	Generic contorls exist. Must be Tested carefully
L?	No know controls, but risk is Low
L!	There is a know vulnerability, but risks ir Low
HTTPS	Control: only HTTPS allowed
A&A	Control: Authorization and Authentication. To be tested
UUID	Control: temporal (30 sec) uuid generation