Secure Software Development Levels and Costs

Barry Boehm

Elaine Venson

October 13th, 2020

Outline

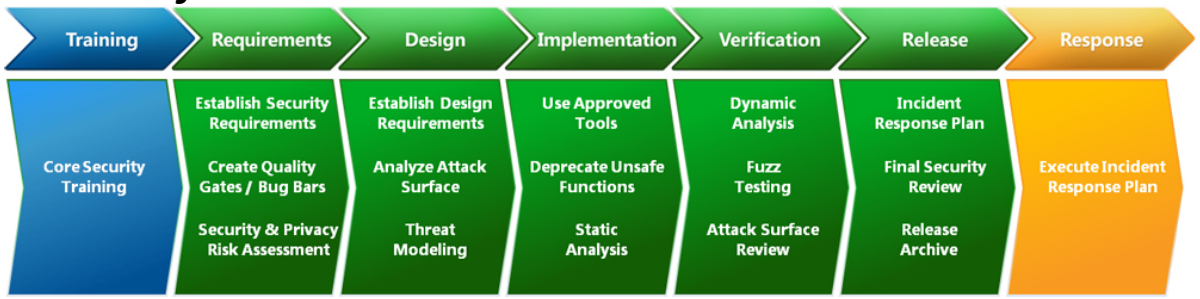
- Secure Software Development Costs
- Scale Development
- Resulting Estimates from Security Experts
- Next Steps

Outline

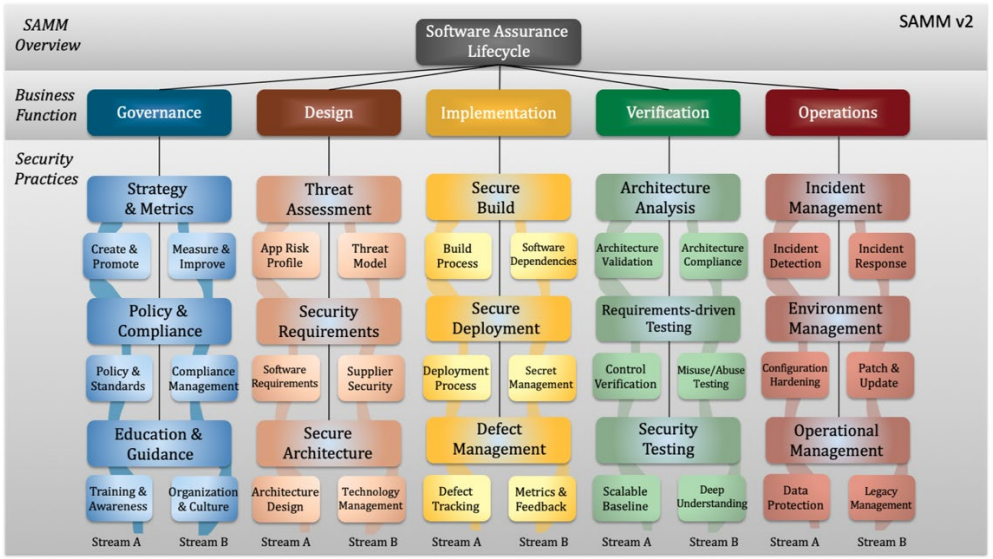
- Secure Software Development Costs
- Scale Development
- Resulting Estimates from Security Experts
- Next Steps

Secure Software Development

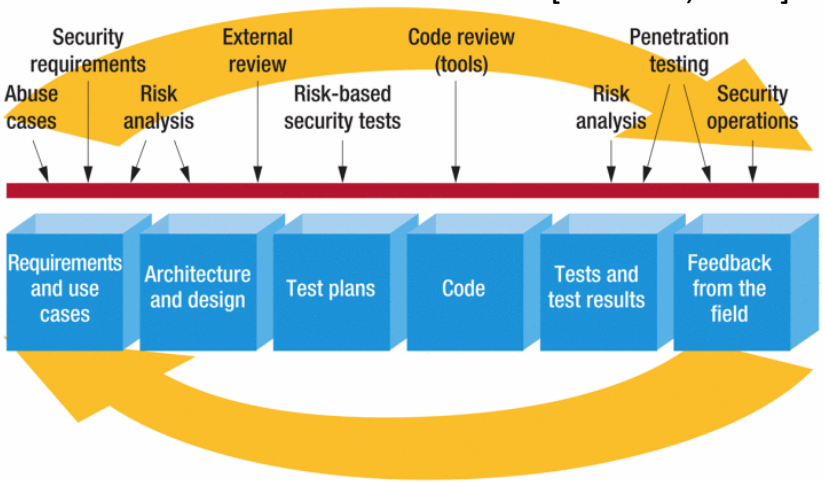
Microsoft SDL https://www.microsoft.com/



OWASP SAMM https://owaspsamm.org/



Touchpoints
[McGraw, 2011]



Software Security as a Trade-off

Costs

- Higher fixing costs
- Patching
- Down-time
- Recovery costs
- Reputation loss

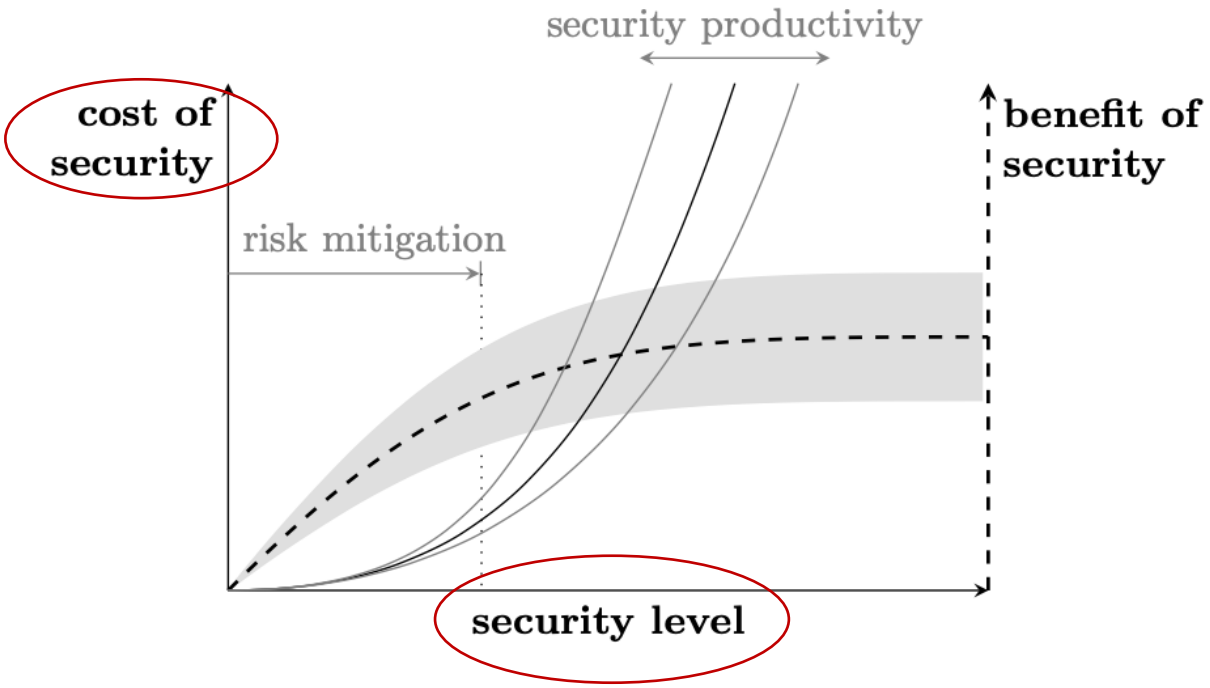
- Expertise
- Tools
- Training
- Improving processes
- Investment in early phases

Benefits

- Priority to features
- Better time to market

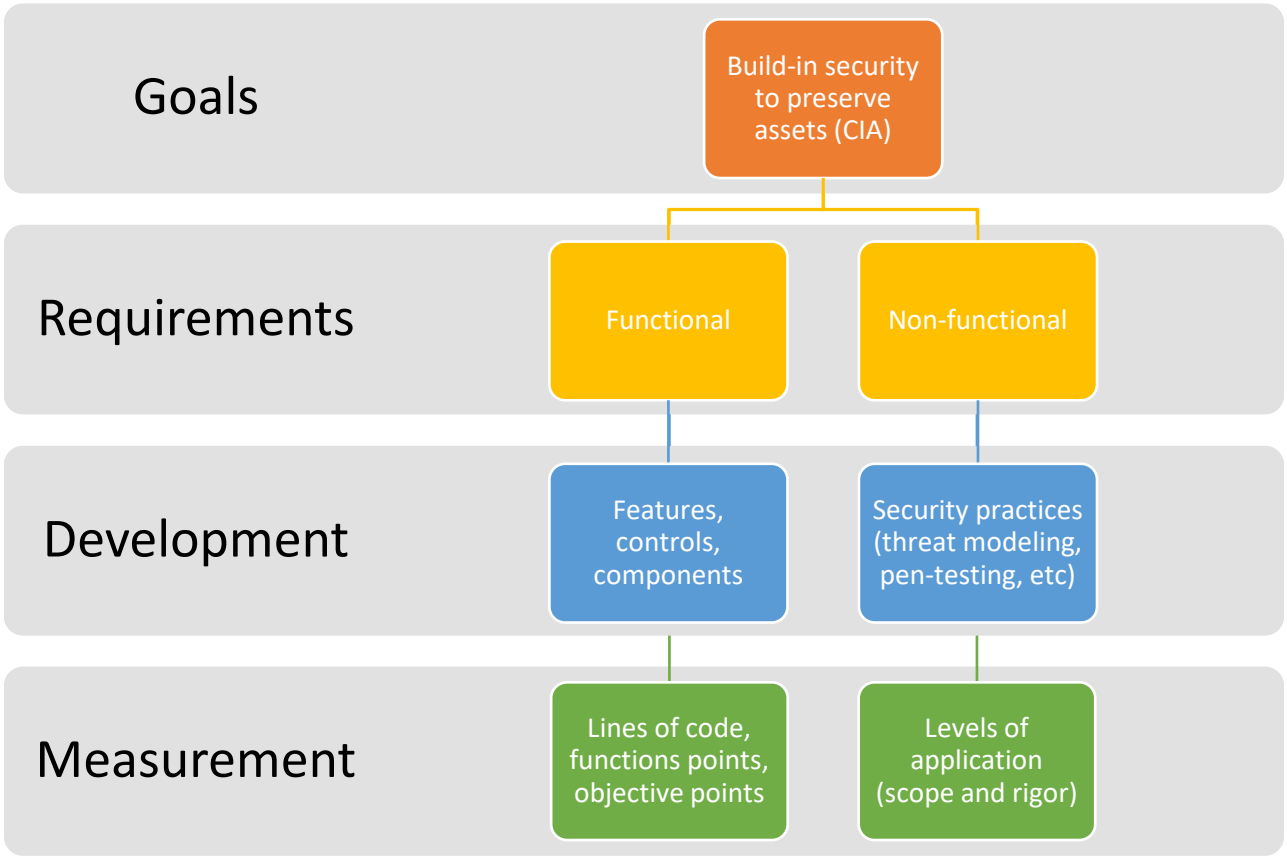
- Vulnerabilities prevention/detection
- Avoided risks
- Reduced total cost

The right amount of security



Böhme, R., 2010. Security Metrics and Security Investment Models, in: Echizen, I., Kunihiro, N., Sasaki, R. (Eds.), Advances in Information and Computer Security.

Costs of SecSw Development



Measuring SecSw Development

Measurement	Lines of code, functions points, objective points	Levels of application (scope and rigor)
	Security Features Size: • Directly estimated using sw sizing methods, or • Estimated using a **Security Sizing Factor**	***Secure Sw Dev Level:*** • Development of an *ordinal scale* based on application of software security practices – **Secure Software Development Scale**

Outline

- Secure Software Development Costs
- Scale Development
- Resulting Estimates from Security Experts
- Next Steps

Secure Software Development Scale

- Ordinal scale defining degrees of application of security practices
- Scale items development based on:
  - Literature
  - BSIMM (Building Security in Maturity Model)
  - OWASP SAMM (Software Assurance Maturity Model)
  - COCOMO descriptors of attribute levels

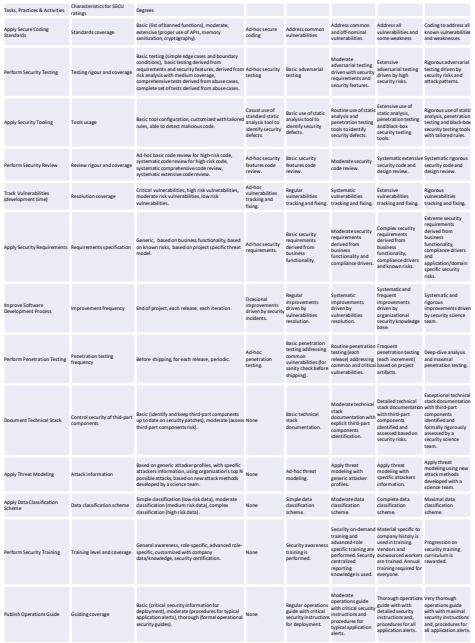
Software Security Practices

Apply Security Requirements	Consider and document security concerns prior to implementation of software features.
Apply Data Classification Scheme	Maintain and apply a Data Classification Scheme. Identify and document security-sensitive data, personal information, financial information, system credentials.
Apply Threat Modeling	Anticipate, analyze, and document how and why attackers may attempt to misuse the software.
Document Technical Stack	Document the components used to build, test, deploy, and operate the software. Keep components up to date on security patches.
Apply Secure Coding Standards	Apply (and define, if necessary) security-focused coding standards for each language and component used in building the software.
Apply Security Tooling	Use security-focused verification tool support (e.g. static analysis, dynamic analysis, coverage analysis) during development and testing.
Perform Security Testing	Consider security requirements, threat models, and all other available security-related information and tooling when designing and executing the software's test plan.
Perform Penetration Testing	Arrange for security-focused stress testing of the project's software in its production environment. Engage testers from outside the software's project team.
Perform Security Review	Perform security-focused review of all deliverables, including, for example, design, source code, software release, and documentation. Include reviewers who did not produce the deliverable being reviewed.
Publish Operations Guide	Document security concerns applicable to administrators and users, supporting how they configure and operate the software.
Track Vulnerabilities	Track software vulnerabilities detected in the software and prioritize their resolution.
Improve Development Process	Incorporate "lessons learned" from security vulnerabilities and their resolutions into the project's software development process.
Perform Security Training	Ensure project staff are trained in security concepts, and in role-specific security techniques.

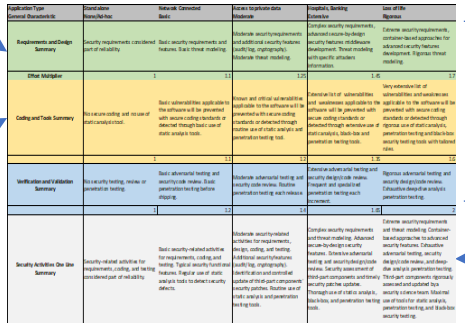
Morrison, P., Smith, B.H., Williams, L., 2017. Surveying Security Practice Adherence in Software Development, in: Proceedings of the Hot Topics in Science of Security: Symposium and Bootcamp, HoTSoS. ACM, New York, NY, USA, pp. 85–94.

Scale Development

Practices Levels' Description → Practices Grouping → Practices Summarization



Task	Practices	Characteristics for SECU ratings	Low	Nominal	High	Very High	Extra High
Requirements and Design	Apply Security Requirements	Requirements specification	Ad-hoc security requirements.	Basic security requirements derived from business functionality.	Moderate security requirements derived from business functionality and compliance drivers.	Complex security requirements derived from business functionality, compliance drivers and known risks.	Extreme security requirements derived from business functionality, compliance drivers and application/domain specific security risks.
	Security Features	Scope and rigour	None.	Build basic security features (authentication, role management, key management, audit/log, cryptography, protocols).	Build additional security features (authentication, role management, key management, audit/log, cryptography, protocols).	Build secure-by-design middleware for security features (authentication, role management, key management, audit/log, cryptography, protocols).	Build container-based approaches for security features (authentication, role management, key management, audit/log, cryptography, protocols).
	Apply Threat Modeling	Attack information	None.	Ad-hoc threat modeling.	Apply threat modeling with generic attacker profiles.	Apply threat modeling with specific attackers information.	Apply threat modeling using new attack methods developed with a science team.
Coding	Apply Secure Coding Standards	Standards coverage	Ad-hoc secure coding	Address common vulnerabilities	Address common and off-nominal vulnerabilities	Address all vulnerabilities and some weakness	Coding to address all known vulnerabilities and weaknesses
	Apply Security Tooling	Tools usage	Casual use of standard static analysis tool to identify security defects.	Basic use of static analysis tool to identify security defects.	Routine use of static analysis and penetration testing tools to identify security defects.	Extensive use of static analysis, penetration testing and black-box security testing tools.	Rigorous use of static analysis, penetration testing and black-box security testing tools with tailored rules.
Verification and Validation	Perform Security Testing	Testing rigour and coverage	Ad-hoc security testing	Basic adversarial testing	Moderate adversarial testing driven with security requirements and security features.	Extensive adversarial testing driven by high security risks.	Rigorous adversarial testing driven by security risks and attack patterns.
	Perform Security Review	Review rigour and coverage	Ad-hoc security features code review.	Basic security features code review.	Moderate security code review.	Systematic extensive security code and design review.	Systematic rigorous security code and design review.
	Perform Penetration Testing	Penetration testing frequency	Ad-hoc penetration testing.	Basic penetration testing addressing common vulnerabilities (for sanity check before shipping).	Routine penetration testing (each release) addressing common and critical vulnerabilities.	Frequent penetration testing (each increment) based on project artifacts.	Deep-dive analysis and maximal penetration testing.



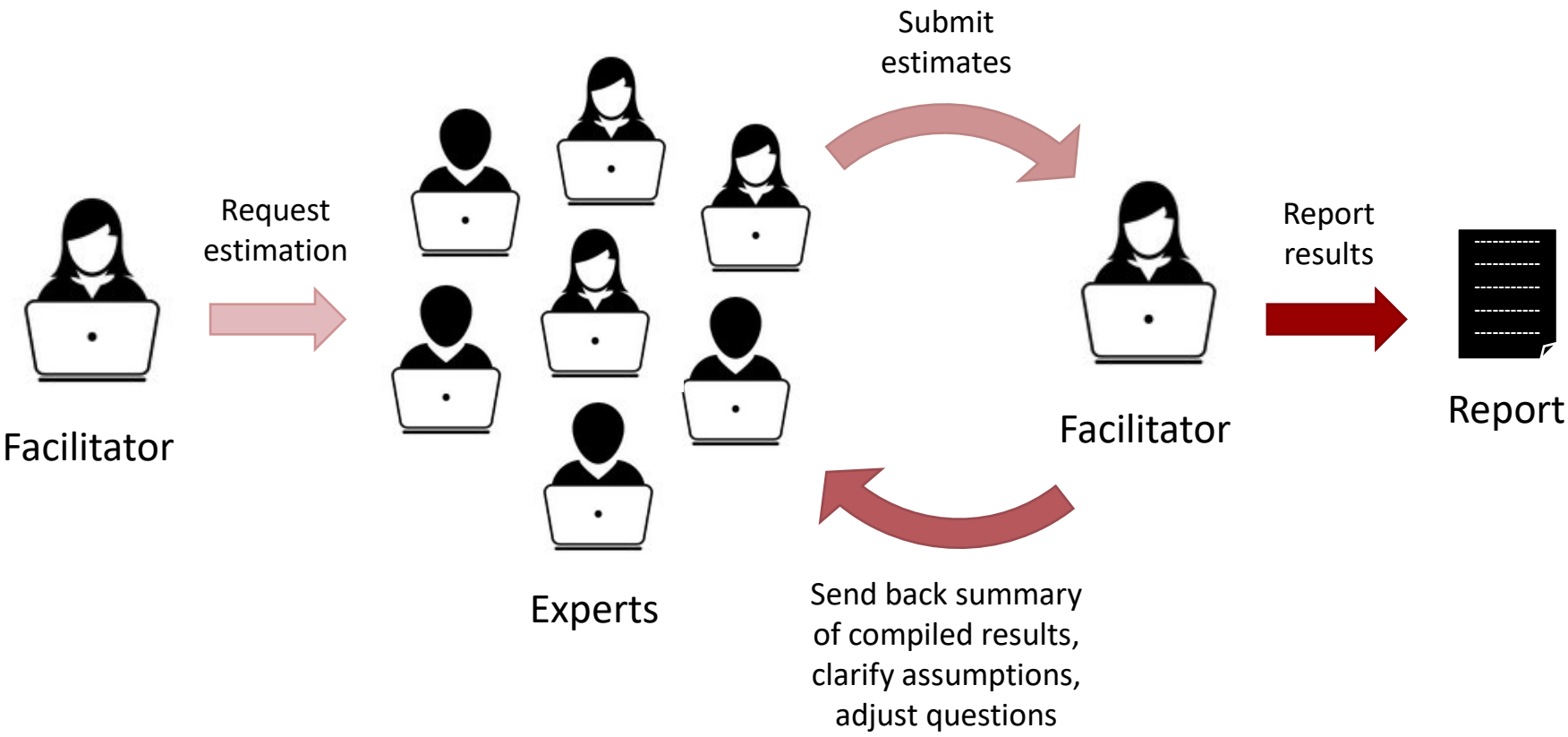
Resulting Rating Scale

Security Requirements and Security Design	Secure Coding and Security Tools	Security Verification and Validation (V&V)
LEVEL 0 None	**LEVEL 0** No secure coding and no use of static analysis tool.	**LEVEL 0** None
LEVEL 1 **Basic analysis to identify security requirements.** Basic threat modeling.	**LEVEL 1** Basic vulnerabilities applicable to the software will be prevented with secure coding standards and/or detected through basic use of static analysis tools.	**LEVEL 1** Basic adversarial testing and security code review. Basic penetration testing. Security V&V activities conducted within the project.
LEVEL 2 **Additional analysis to identify security requirements such as audit/log, cryptography, etc.** Moderate threat modeling.	**LEVEL 2** Known and critical vulnerabilities applicable to the software will be prevented with secure coding standards and/or detected through routine use of static analysis tools.	**LEVEL 2** Moderate adversarial testing and security code review. Routine penetration testing. Security V&V activities conducted by an independent group.
LEVEL 3 **Thorough analysis to identify security requirements, advanced secure-by-design needs**. Threat modeling with specific attack strategies.	**LEVEL 3** Extensive list of vulnerabilities and weaknesses applicable to the software will be prevented with secure coding standards and/or detected through extensive use of static analysis and black-box tools.	**LEVEL 3** Extensive adversarial testing and security design/code review. Frequent and specialized penetration testing. Security V&V activities conducted by an independent group at the organizational level.
LEVEL 4 **Extensive analysis to identify security requirements, including off-nominal cases,** container-based approaches for advanced security features development. Rigorous threat modeling.	**LEVEL 4** Very extensive list of vulnerabilities and weaknesses applicable to the software will be prevented with secure coding standards and/or detected through rigorous use of static analysis and black-box security testing tools with tailored rules. Employ formal methods in coding.	**LEVEL 4** Rigorous adversarial testing and security design/code review. Exhaustive deep-dive analysis penetration testing. Use of formal verification and custom developed V&V tools. Security V&V activities conducted by an outside certified company.

Outline

- Secure Software Development Costs
- Scale Development
- Resulting Estimates from Security Experts
- Next Steps

Online Delphi

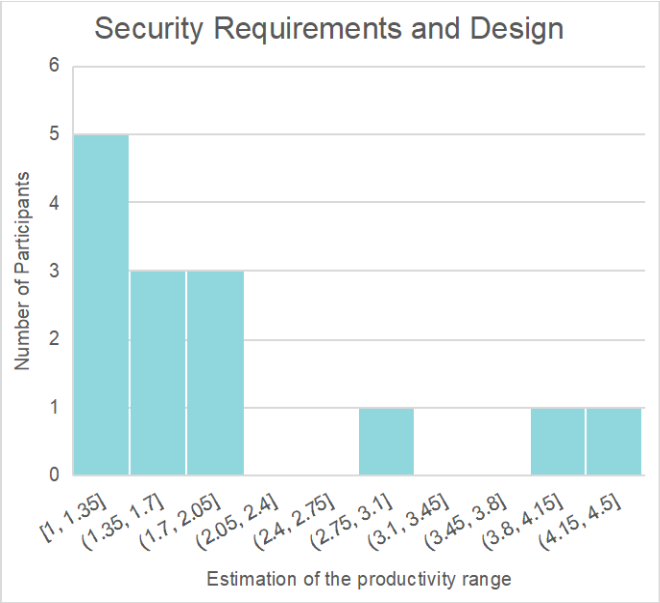


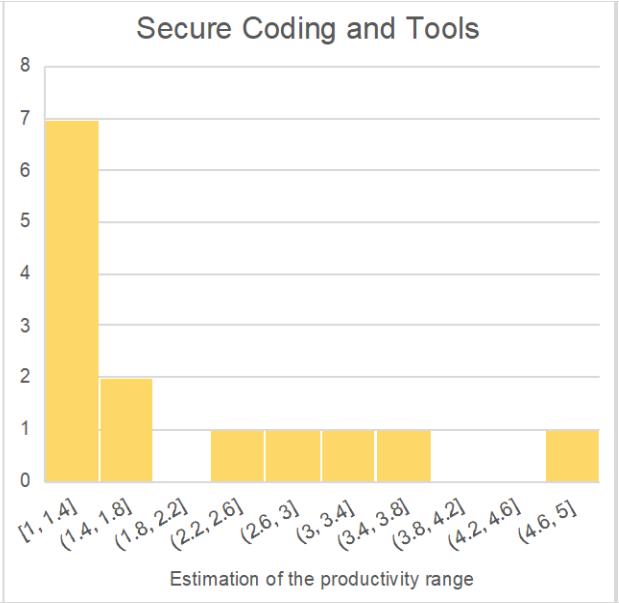
Results from online Delphi

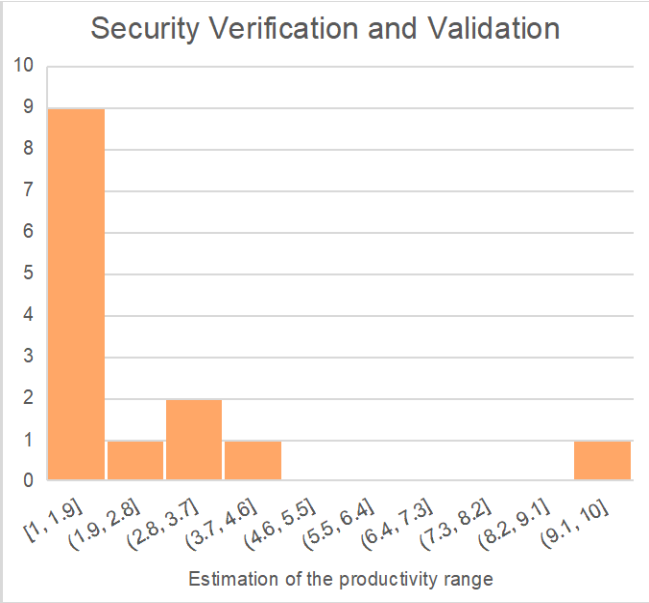
- September 2020
- Participants invited from the Software Security Group on LinkedIn
- 2 rounds
  - 17 participants
  - 14 participants
- 10.88 years average experience with Secure Software Development
- 11.06 years average experience with Software Estimation

Productivity Range*

Histograms for each group of security practices







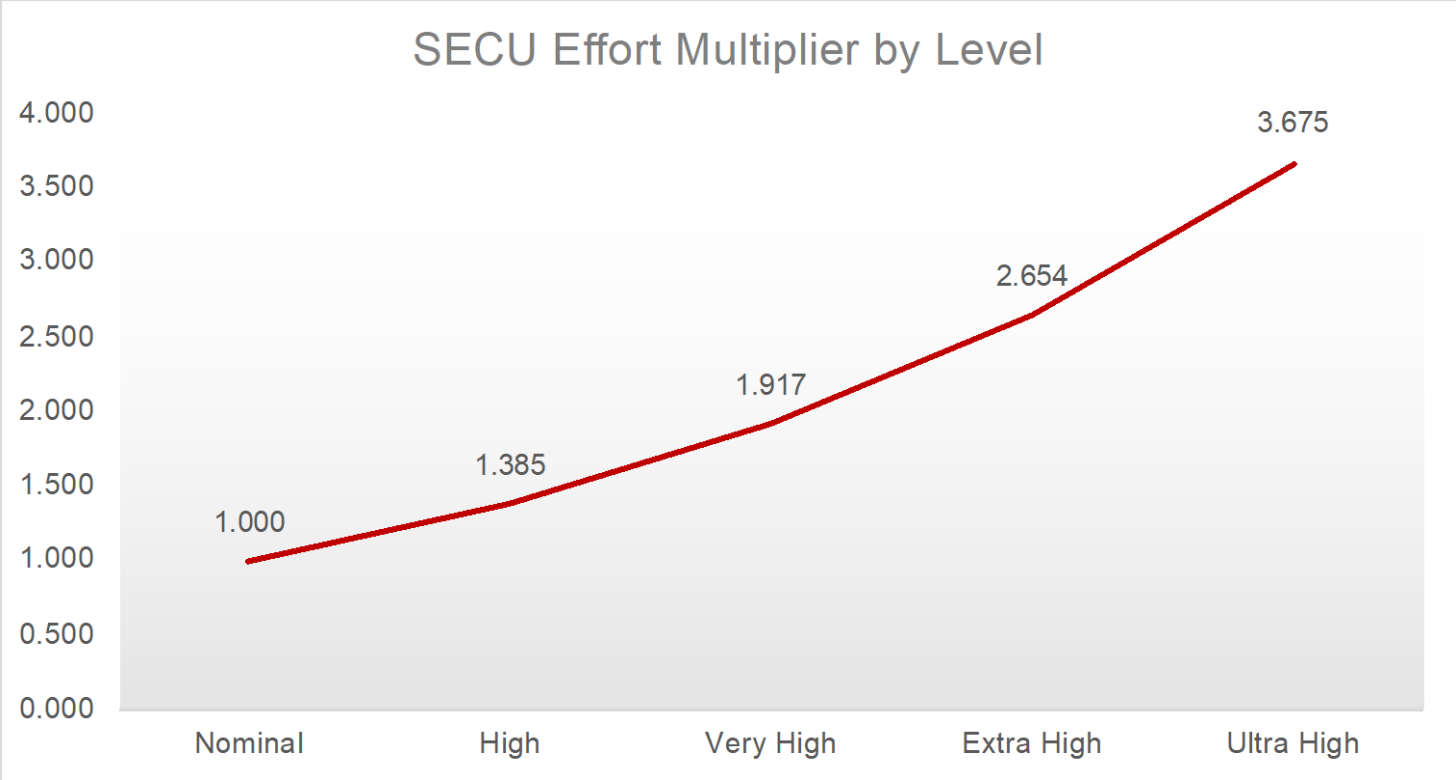
* Productivity range is the ratio between the highest level (Level 4) and the lowest level of the scale (Level 0).

Productivity Range

Group	Average	Median	Standard Deviation	Coefficient of Variation
Requirements and Design	1.957	1.5	1.093	56%
Coding and Tools	2.046	1.4	1.193	58%
Verification and Validation	2.561	1.75	2.335	91%
Productivity Range	**10.256**	**3.675**		

Added Effort by Security Level

Based on median productivity range



Increase in Application Size

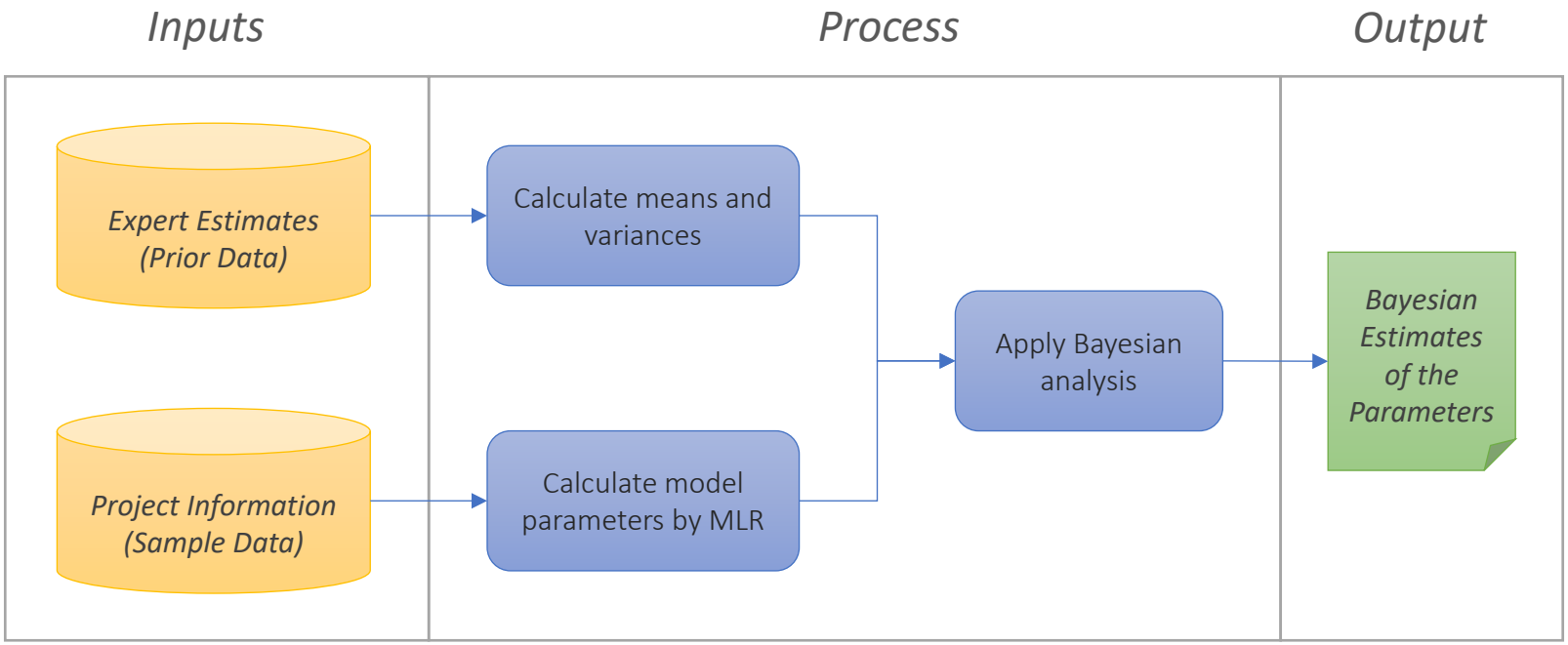
Estimates from 14 participants (only in 2nd round)

	L1 High	L2 Very High	L3 Extra High	L4 Ultra High
Average	1.170	1.393	1.668	1.914
Median	1.100	1.250	1.500	1.675
Std Deviation	0.125	0.366	0.590	0.839
Coefficient of Variation	11%	26%	35%	44%

Outline

- Secure Software Development Costs
- Scale Development
- Resulting Estimates from Security Experts
- Next Steps

Cost Estimation Model Building



Proposed Cost Model Form

- Original COCOMO II equation

$$Effort = A \cdot Size^{E} \cdot \prod_{i-1}^{n} EM_i$$

- Addition of the parameter for secure software development level, and adjusted size:

$$Effort = A \cdot Size^{E} \cdot SECU \cdot \prod_{i-1}^{n} EM_i$$

Includes Security Functional Features

Effort multiplier for secure software development level

Data Collection

Security experts' estimates for the security parameter

Estimation experts' estimates for the security parameter

} Wideband Delphi

Projects' Data → Manual Data Collection Form

Projects' Data → Automated Data Collection

Projects' Data → Survey OSS developers

Poll - Get involved!

1) Participate in an online Delphi study
   - Share your estimates and assumptions anonymously
   - Compare your your estimates with other participants
2) Participate in data collection
   - Provide sanitized data
   - Receive a version of the model calibrated for your organization

Contact: Elaine Venson
venson@usc.edu

Contact: Brad Clark (COCOMO III Project Coordinator)
clarkbk@usc.edu

Thank you!

Barry Boehm
boehm@usc.edu

Elaine Venson
venson@usc.edu

References

- R. Böhme, “Security Metrics and Security Investment Models,” in *Advances in Information and Computer Security*, vol. 6434, I. Echizen, N. Kunihiro, and R. Sasaki, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 10–24.
- G. McGraw, “Technology Transfer: A Software Security Marketplace Case Study,” IEEE Software, vol. 28, no. 5, pp. 9–11, Sep. 2011, doi: 10.1109/MS.2011.110.
- Morrison, P., Smith, B.H., Williams, L., 2017. Surveying Security Practice Adherence in Software Development, in: Proceedings of the Hot Topics in Science of Security: Symposium and Bootcamp, HoTSoS. ACM, New York, NY, USA, pp. 85–94.
- E. Venson, R. Alfayez, G. Marília M. F., F. Rejane M. C., and B. Boehm, “The Impact of Software Security Practices on Development Effort: An Initial Survey,” in 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Sep. 2019, pp. 1–12, doi: 10.1109/ESEM.2019.8870153.
- E. Venson, X. Guo, Z. Yan, and B. Boehm, “Costing Secure Software Development: A Systematic Mapping Study,” in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, New York, NY, USA, 2019, p. 9:1–9:11, doi: 10.1145/3339252.3339263.