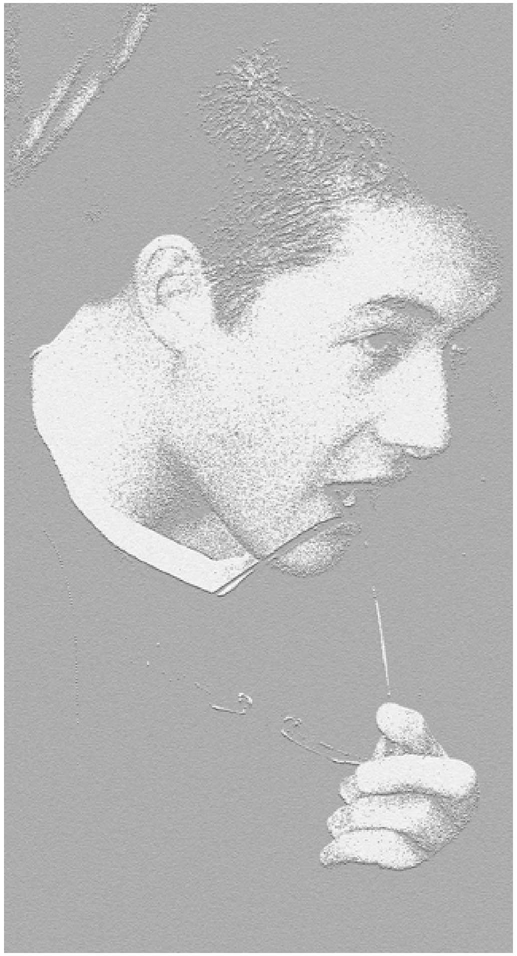
Alternative Assurance Criteria

Dr. David Brewer

Gamma Secure Systems Limited

www.gammassl.co.uk

Agenda



- Motivation
- Meta Criteria
- Common Criteria Interpretation
- Alternative Assurance Criteria Interpretation
- Conclusions

Common Criteria

Motivation

Reported at recent ICCCs

- "Marketing is finding out what the customer wants", *Marketing panel (Tokyo)*
- "After we have developed a product we employ a consultant to create the Common Criteria design documentation", *Microsoft (Tokyo and Berlin)*
- There has to be a better way

... and then I remembered

a long, long time ago...

- Before the first CESG evaluation facility (1986)
- Successful evaluations of UK government and banking IT systems
- High assurance gained through formal code analysis techniques (SPADE/MALPAS)
- Some work published but long since forgotten
- Approach abandoned in favour of quasi-harmonisation with Orange Book

Meta Criteria (pre 1986)

A piece of ancient history

Goals

- “Correct operation confirmed or weaknesses and countermeasures identified”
- In those days we spoke of software integrity (not security):
  - *Software does what it is supposed to do and <u>not what it is not supposed to do</u>”*
- Use precluded functions/properties (e.g. covert channels) as well as required functions/properties

Assurance

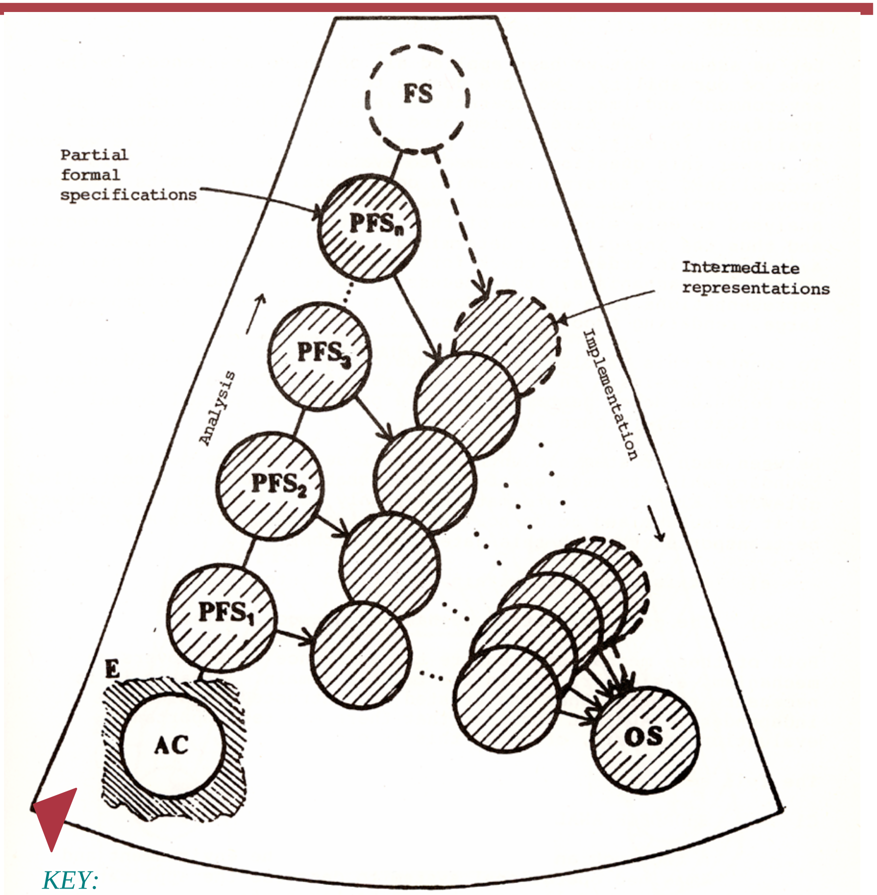
- Based on the extent of evaluator’s knowledge about the TOE
- Always used a model of the implementation as a means to reason about software integrity
- Always included testing (machine code verification regarded as impractical)
- Always checked quality controls (to ensure evaluating the right thing)
- Principle of escalation
  - *Note: Orange Book also has some examples of this*

Meta Criteria

- Level of abstraction of the model
  - *How far removed from the implementation*
- Form of the abstraction
  - *What does it allow the analyst to reason about*
  - *How does it allow the analyst to reason*
- Means of model creation
  - *Design intent or reverse engineering of the implementation*

Creation of the model

- Two principal routes:
  - *Implementation route (i.e. use the design documentation)*
  - *Analysis route (i.e. reverse engineer the models from the final implementation)*
- Can be a mix



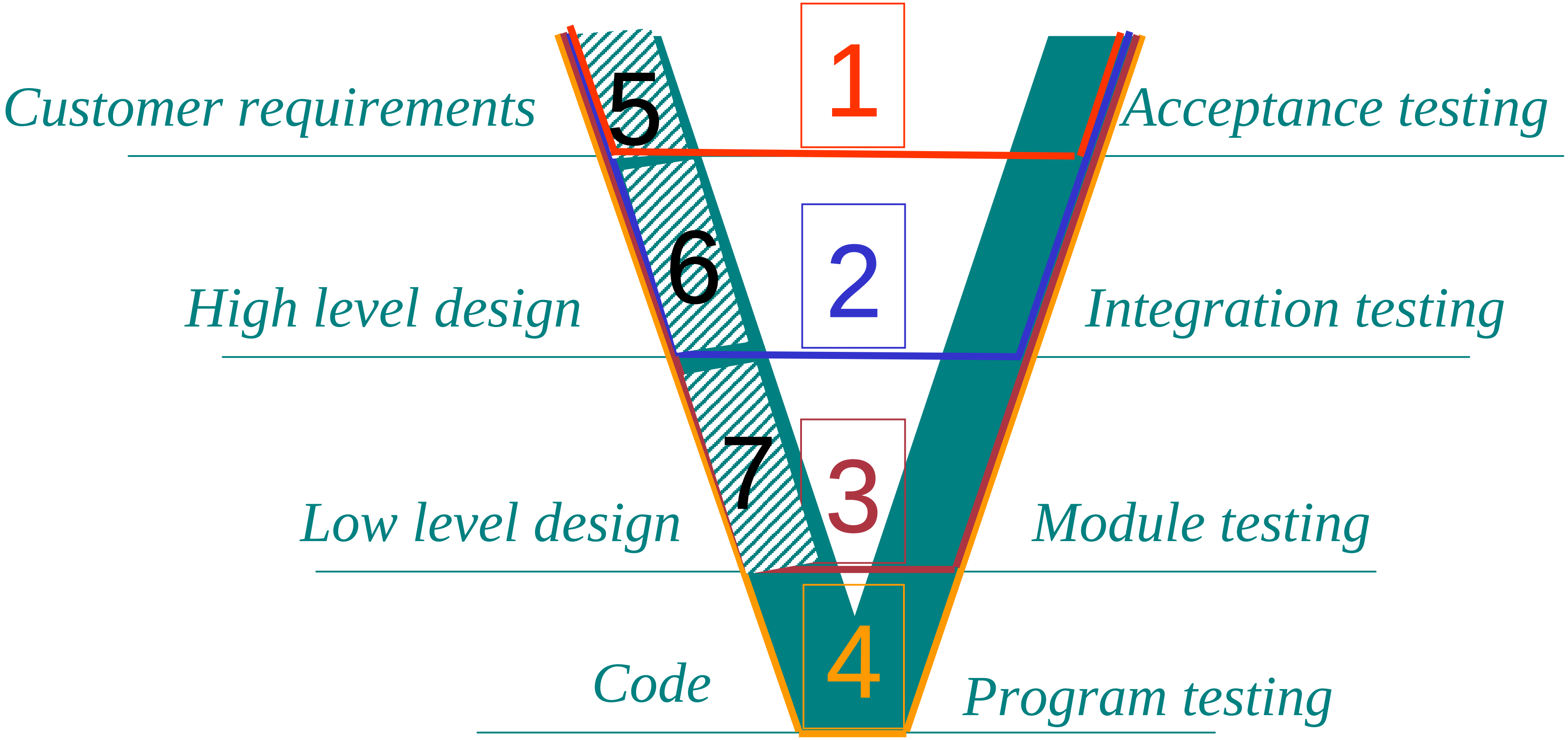
KEY:
AC: Application concept (e.g. user requirement)
FS: Formal specification (e.g. top level design)
PFS: Partial formal specifications (i.e. intermediate representations)
OS: Operational system (the actual resultant implementation)
E: Environment

Common Criteria Interpretation

Basic thesis

- Use effectiveness criteria to reason about security
- Use design documentation for the models
- Use correctness criteria to argue that the models are a sufficiently correct representation of the implementation

EALs (V-model view)



Customer requirements

High level design

Low level design

Code

Acceptance testing

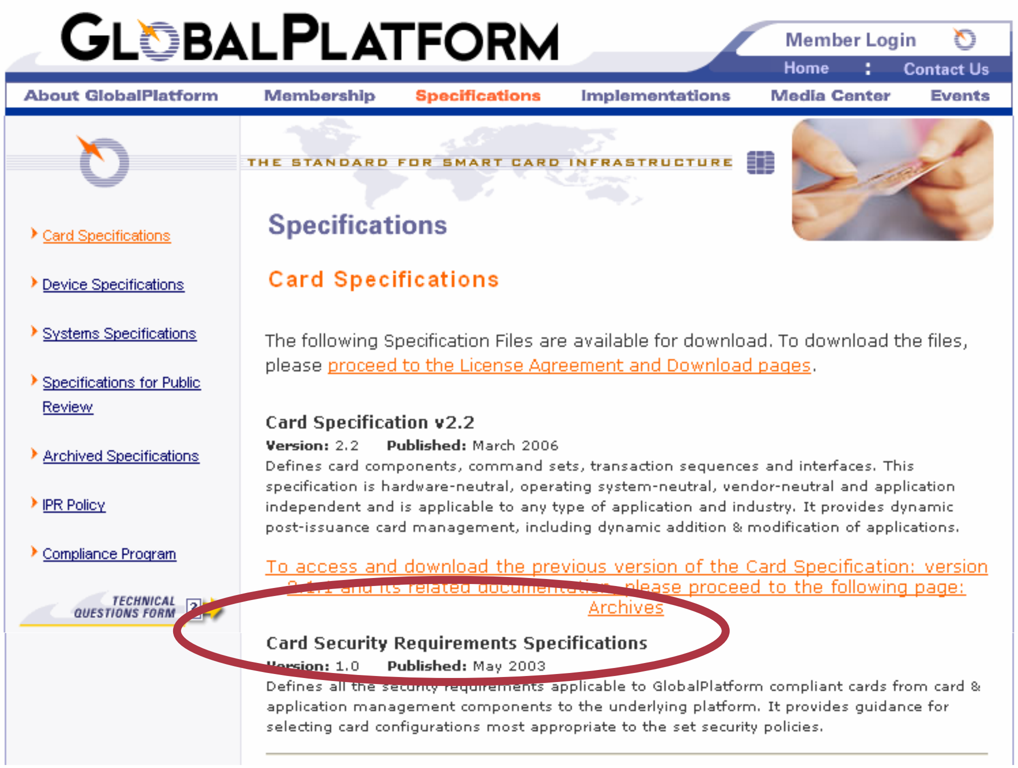
Integration testing

Module testing

Program testing

Example 1: GlobalPlatform

- Could have produced PP based on Visa Open Platform PP (OP3) – see ICCC2 (Brighton)
- But too many PPs (OP3, JavaCard, SCSUG, SSVG)
- Wanted something more intelligible
- Way ahead shown by ITRI – see ICCC3 (Ottawa)
- Result was the Card Security Requirements Specification – see www.globalplatform.org



Specifications

Card Specifications

The following Specification Files are available for download. To download the files, please proceed to the License Agreement and Download pages.

Card Specification v2.2

Version: 2.2 **Published:** March 2006

Defines card components, command sets, transaction sequences and interfaces. This specification is hardware-neutral, operating system-neutral, vendor-neutral and application independent and is applicable to any type of application and industry. It provides dynamic post-issuance card management, including dynamic addition & modification of applications.

To access and download the previous version of the Card Specification: version 2.1.1 and its related documentation, please proceed to the following page: Archives

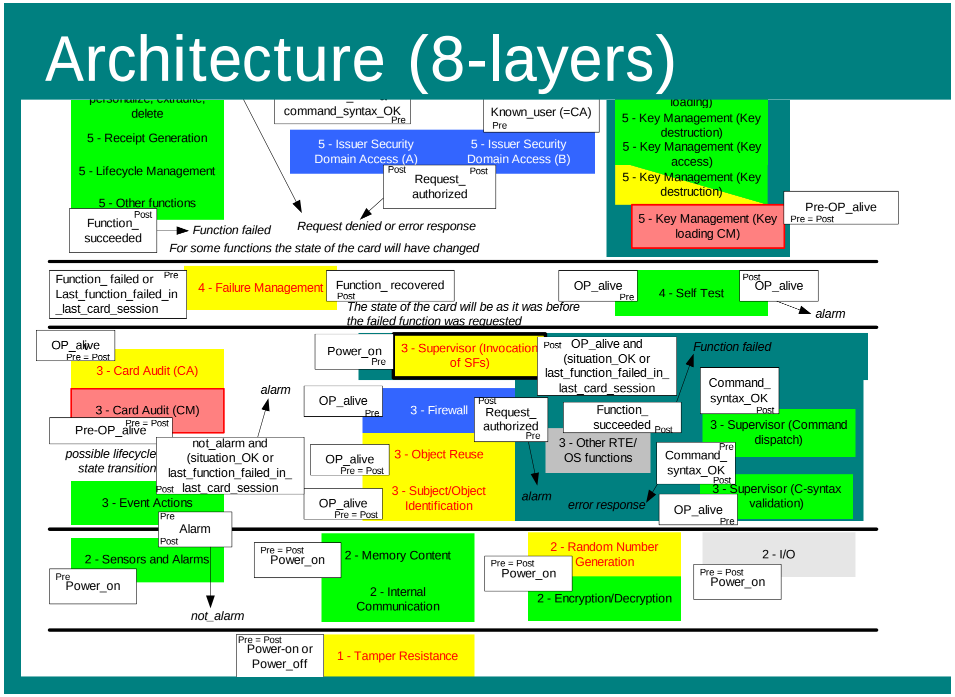
Card Security Requirements Specifications

Version: 1.0 **Published:** May 2003

Defines all the security requirements applicable to GlobalPlatform compliant cards from card & application management components to the underlying platform. It provides guidance for selecting card configurations most appropriate to the set security policies.

Example 1: GlobalPlatform

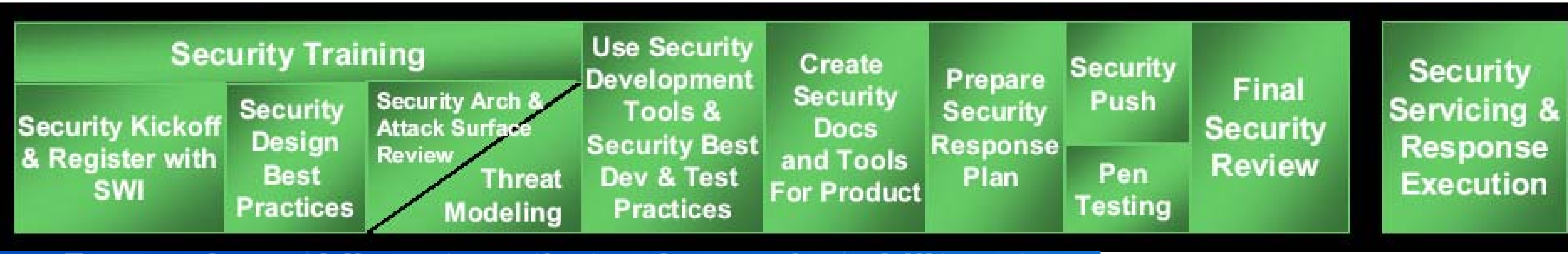
- Semi-formal specification covering card content management down to and including the IC (physical)
- Addressed chip card “composition problem”
- Facilitated EAL6 evaluation (at least)
- But card vendors just go for EAL4+



Architecture (8-layers)

personalize, extradite, delete
5 - Receipt Generation
5 - Lifecycle Management
5 - Other functions
Function_ succeeded
Post
Function failed
command_syntax_OK
Pre
5 - Issuer Security Domain Access (A)
5 - Issuer Security Domain Access (B)
Known_user (=CA)
Pre
Request_ authorized
Post
Post
Request denied or error response
For some functions the state of the card will have changed
loading)
5 - Key Management (Key destruction)
5 - Key Management (Key access)
5 - Key Management (Key destruction)
5 - Key Management (Key loading CM)
Pre-OP_alive
Pre = Post
Function_ failed or Last_function_failed_in _last_card_session
Pre
4 - Failure Management
Function_ recovered
Post
The state of the card will be as it was before the failed function was requested
OP_alive
Pre
4 - Self Test
Post
OP_alive
alarm
OP_alive
Pre = Post
3 - Card Audit (CA)
3 - Card Audit (CM)
Pre-OP_alive
Pre = Post
possible lifecycle state transition
not_alarm and (situation_OK or last_function_failed_in_ last_card_session
Post
alarm
3 - Event Actions
Pre
Alarm
Post
Power_on
Pre
3 - Supervisor (Invocation of SFs)
Post
OP_alive and (situation_OK or last_function_failed_in_ last_card_session
Function failed
OP_alive
Pre
3 - Firewall
Post
Request_ authorized
Pre
Function_ succeeded
Post
Command_ syntax_OK
Post
3 - Supervisor (Command dispatch)
OP_alive
Pre = Post
3 - Object Reuse
3 - Other RTE/ OS functions
Command_ syntax_OK
Pre
Post
OP_alive
Pre = Post
3 - Subject/Object Identification
alarm
error response
OP_alive
Pre
3 - Supervisor (C-syntax validation)
2 - Sensors and Alarms
Pre
Power_on
not_alarm
Pre = Post
Power_on
2 - Memory Content
2 - Internal Communication
Pre = Post
Power_on
2 - Random Number Generation
2 - Encryption/Decryption
2 - I/O
Pre = Post
Power_on
Pre = Post
Power-on or Power_off
1 - Tamper Resistance

Example 2: Microsoft's SDL



Security Training

Security Kickoff & Register with SWI

Security Design Best Practices

Security Arch & Attack Surface Review

Threat Modeling

Use Security Development Tools & Security Best Dev & Test Practices

Create Security Docs and Tools For Product

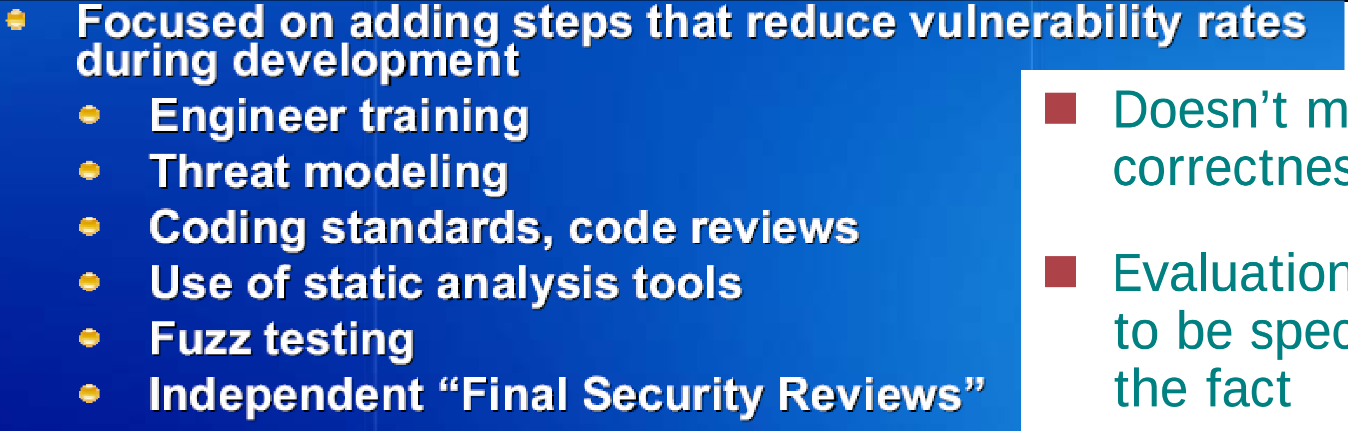
Prepare Security Response Plan

Security Push

Pen Testing

Final Security Review

Security Servicing & Response Execution



- Focused on adding steps that reduce vulnerability rates during development
  - Engineer training
  - Threat modeling
  - Coding standards, code reviews
  - Use of static analysis tools
  - Fuzz testing
  - Independent "Final Security Reviews"

Steven B. Lipner, Microsoft, ICCC6 (Tokyo)

- Doesn't map at all well to correctness requirements
- Evaluation documentation has to be specially produced after the fact
- Unnecessary expense

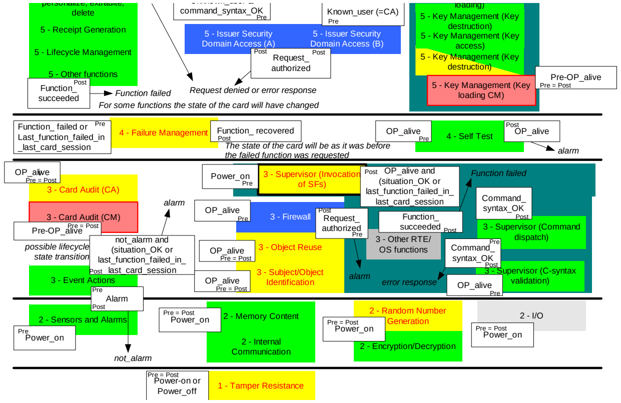
Alternative Assurance Interpretation

The alternative

- CC assurance uses models derived from design intent
- Alternative: use models reverse engineered from the actual implementation (or a mix)
- Note:
  - *less expensive (no need for correctness criteria)*
  - *more reliable (based on the actual implementation)*
  - *Higher assurance (analysis will use formal code (logic) analysis methods)*

Example 1: GlobalPlatform

- Proposition 1:
  - *Animate the Card Security Requirements Specification and verify effectiveness*
  - *Use the results to generate tests*
  - *Use these tests to test the TOE*


delete
5 - Receipt Generation
5 - Lifecycle Management
5 - Other functions
Function_ succeeded
Post
Function failed
command_syntax_OK
Pre
Known_user (=CA)
Pre
5 - Issuer Security Domain Access (A)
5 - Issuer Security Domain Access (B)
Post
Request_ authorized
Post
Request denied or error response
For some functions the state of the card will have changed
5 - Key Management (Key destruction)
5 - Key Management (Key access)
5 - Key Management (Key destruction)
5 - Key Management (Key loading CM)
Pre-OP_alive
Pre = Post
Function_ failed or Last_function_failed_in _last_card_session
Pre
4 - Failure Management
Function_ recovered
Post
The state of the card will be as it was before the failed function was requested
OP_alive
Pre
4 - Self Test
Post
OP_alive
alarm
OP_alive
Pre = Post
3 - Card Audit (CA)
3 - Card Audit (CM)
Pre-OP_alive
Pre = Post
possible lifecycle state transition
not_alarm and (situation_OK or last_function_failed_in_ last_card_session
Post
alarm
3 - Event Actions
Pre
Alarm
Post
Power_on
Pre
3 - Supervisor (Invocation of SFs)
OP_alive
Pre
3 - Firewall
Post
Request_ authorized
Pre
OP_alive
Pre = Post
3 - Object Reuse
OP_alive
Pre = Post
3 - Subject/Object Identification
alarm
Post
OP_alive and (situation_OK or last_function_failed_in_ last_card_session
Function failed
Function_ succeeded
Post
3 - Other RTE/ OS functions
Command_ syntax_OK
Post
3 - Supervisor (Command dispatch)
Pre
Command_ syntax_OK
Post
3 - Supervisor (C-syntax validation)
error response
OP_alive
Pre
2 - Sensors and Alarms
Pre
Power_on
not_alarm
Pre = Post
Power_on
2 - Memory Content
2 - Internal Communication
Pre = Post
Power_on
2 - Random Number Generation
2 - Encryption/Decryption
2 - I/O
Pre = Post
Power_on
Pre = Post
Power-on or Power_off
1 - Tamper Resistance

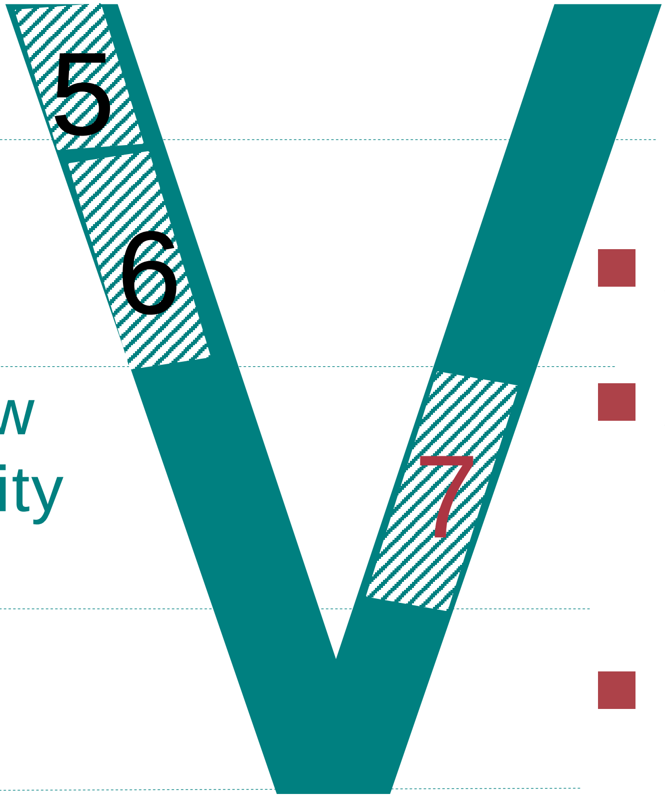
- Proposition 2:
  - *As above, plus*
  - *Analyse reversed-engineered logic modules and confirm results with TOE-specific testing*

- Note the GlobalPlatform Card Security Requirements Specification, animation and associated tests would be a **reusable** evaluation resource

Example 1: GlobalPlatform

Proposition 1:

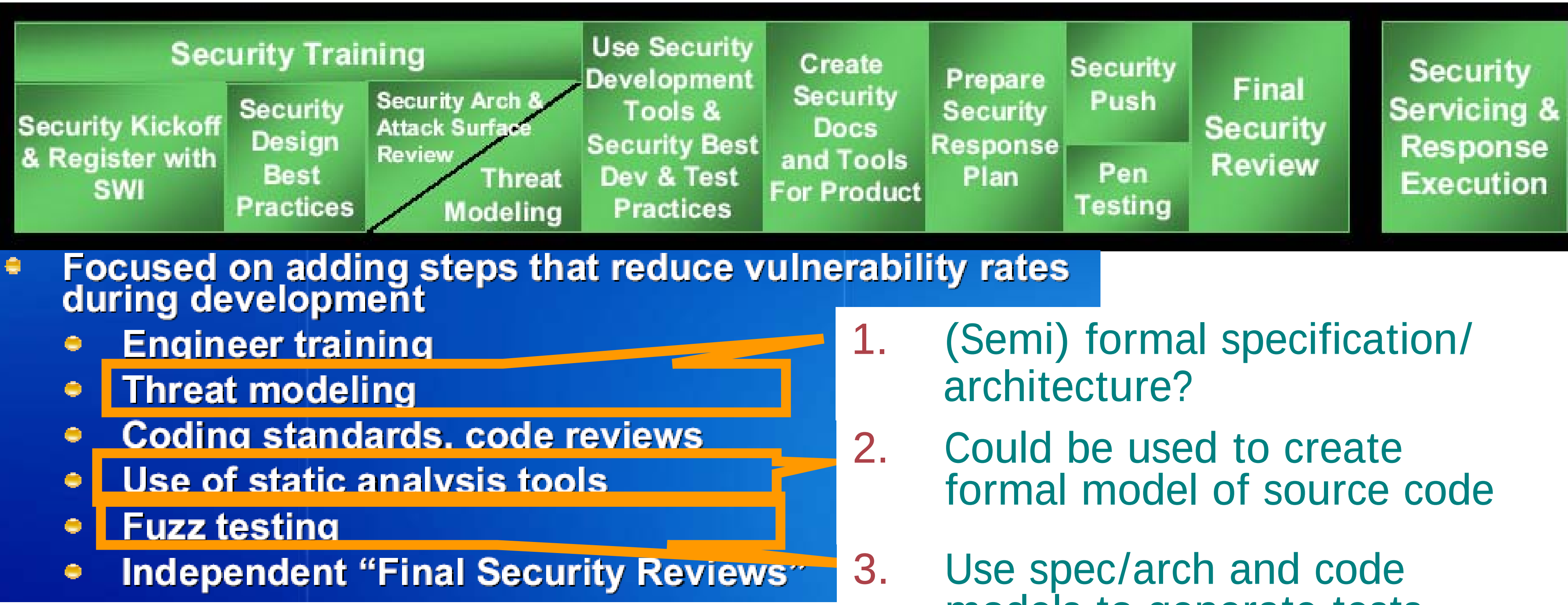
- Animation and verification
- Derive test case
- Use these to show TOE meets security specification



Proposition 2:

- As 1 plus
- Analyse reverse-engineered logic modules
- Conform results with TOE-specific testing

Example 2: Microsoft's SDL



Security Training

Security Kickoff & Register with SWI

Security Design Best Practices

Security Arch & Attack Surface Review

Threat Modeling

Use Security Development Tools & Security Best Dev & Test Practices

Create Security Docs and Tools For Product

Prepare Security Response Plan

Security Push

Pen Testing

Final Security Review

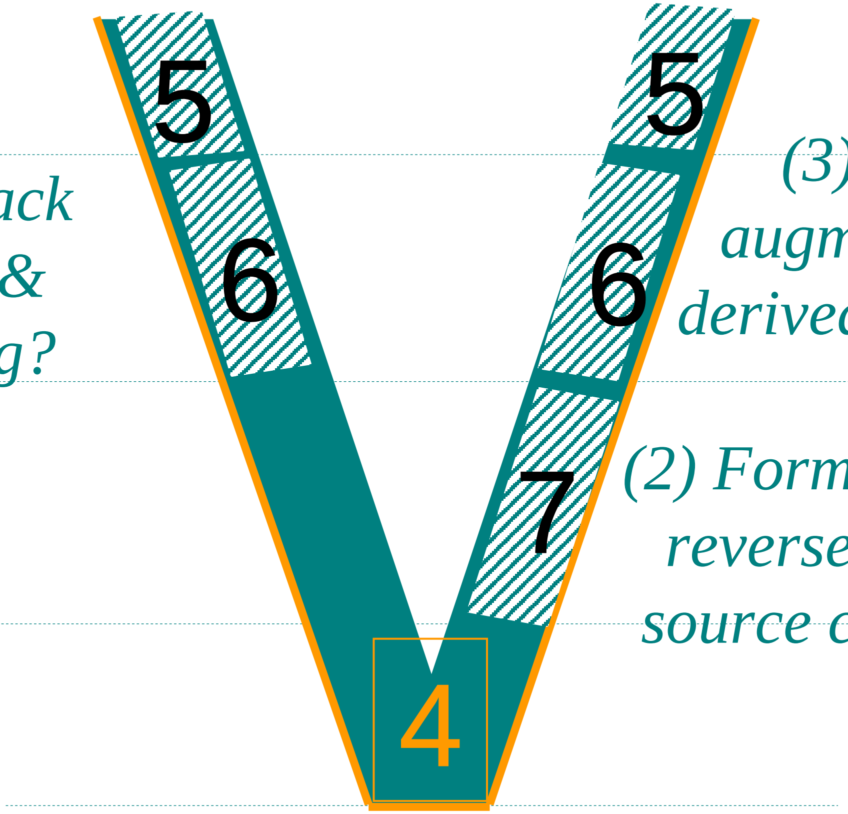
Security Servicing & Response Execution

- Focused on adding steps that reduce vulnerability rates during development
  - Engineer training
  - Threat modeling
  - Coding standards, code reviews
  - Use of static analysis tools
  - Fuzz testing
  - Independent "Final Security Reviews"

1. (Semi) formal specification/ architecture?
2. Could be used to create formal model of source code
3. Use spec/arch and code models to generate tests

Steven B. Lipner, Microsoft, ICCC6 (Tokyo)

Example 2: Microsoft's SDL



(1) Security Architecture, Attack Surface Review & Threat Modelling?

(3) Fuzz testing, augmented with tests derived from (1) and (2)

(2) Formal analysis of reverse engineered source code modules

Conclusions

Conclusions

- Our goal is to remove vulnerabilities
- Could create an alternative criteria based on meta-criteria
- Could be defined to yield an equivalence in terms of EAL
- Could never be created by incremental CC development
- More suited to actual development methodologies?
- Higher assurance for less cost?
- Would need to be driven by vendors
- Go for it?

Alternative Assurance Criteria

Any Questions?

Dr. David Brewer

Gamma Secure Systems Limited

www.gammassl.co.uk