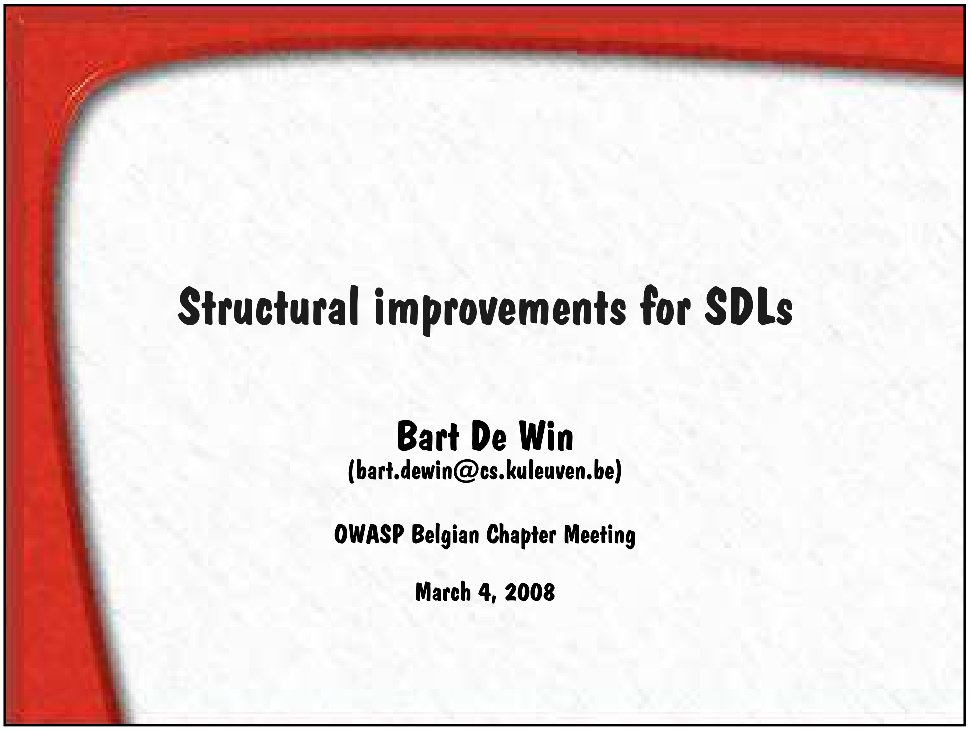
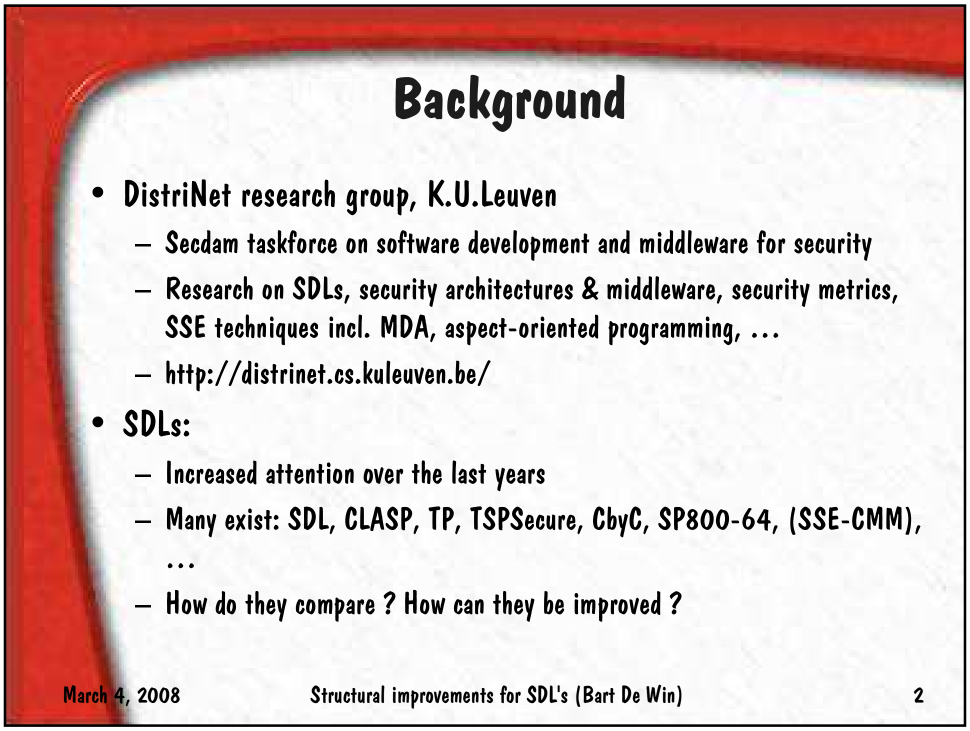


Structural improvements for SDLs

Bart De Win
(bart.dewin@cs.kuleuven.be)

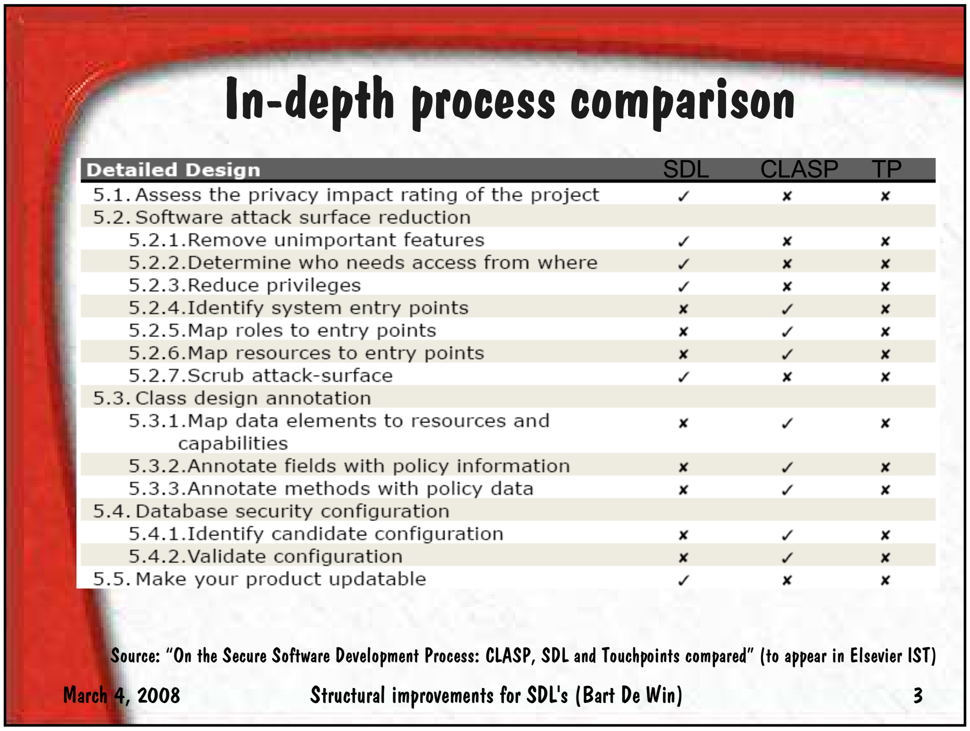
OWASP Belgian Chapter Meeting

March 4, 2008



Background

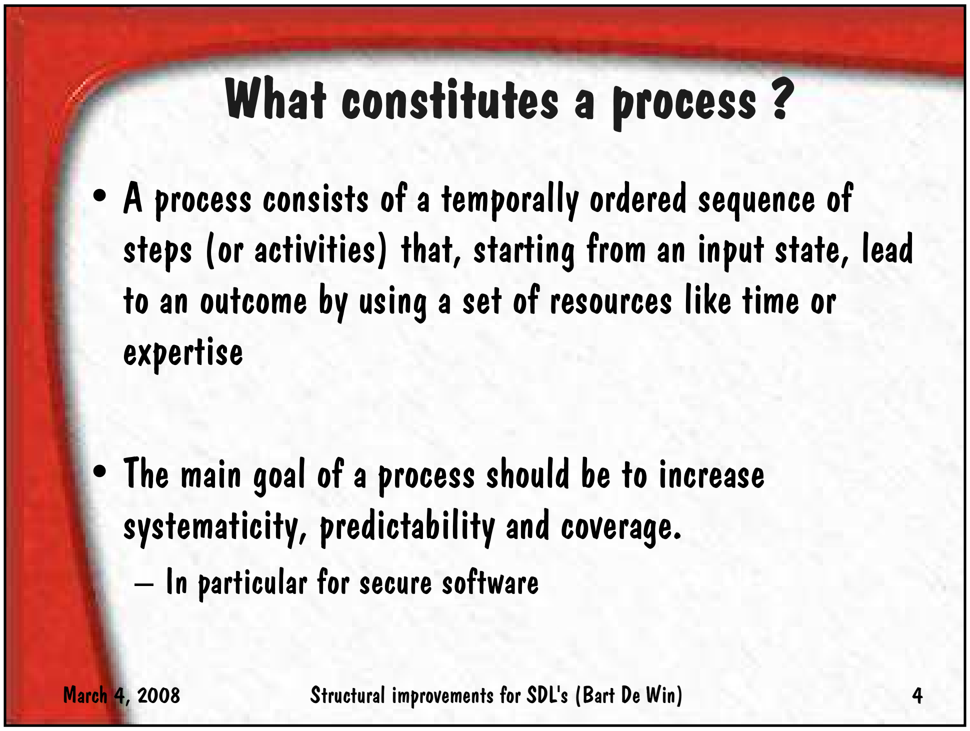
- DistriNet research group, K.U.Leuven
  - Secdam taskforce on software development and middleware for security
  - Research on SDLs, security architectures & middleware, security metrics, SSE techniques incl. MDA, aspect-oriented programming, …
  - http://distrinet.cs.kuleuven.be/
- SDLs:
  - Increased attention over the last years
  - Many exist: SDL, CLASP, TP, TSPSecure, CbyC, SP800-64, (SSE-CMM), …
  - How do they compare ? How can they be improved ?



In-depth process comparison

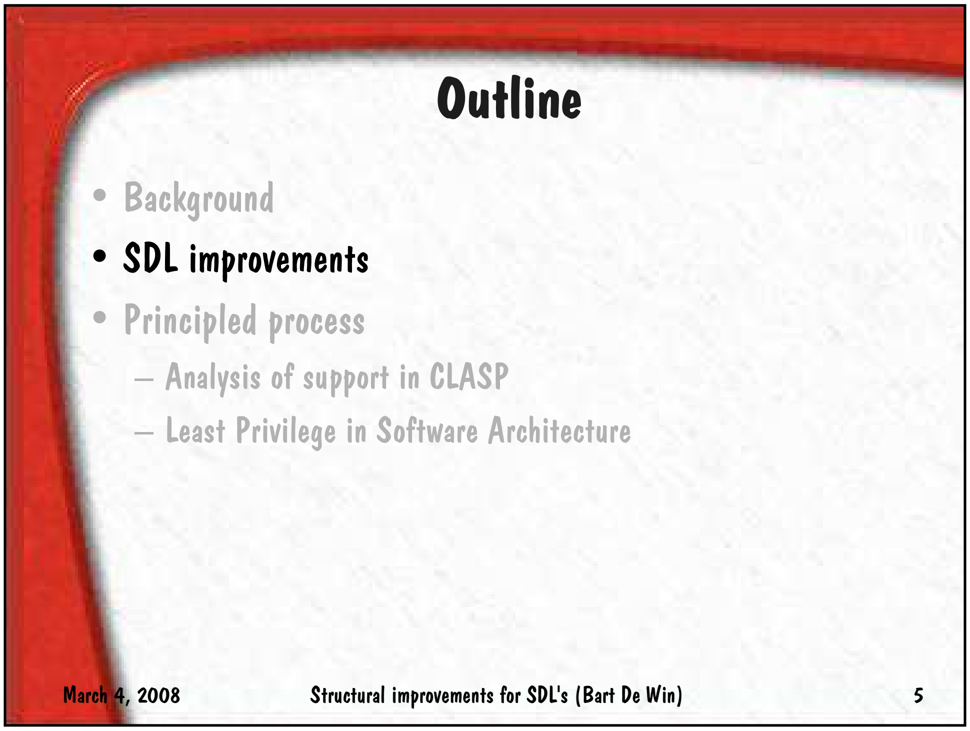
Detailed Design	SDL	CLASP	TP
5.1. Assess the privacy impact rating of the project	✓	✘	✘
5.2. Software attack surface reduction			
5.2.1. Remove unimportant features	✓	✘	✘
5.2.2. Determine who needs access from where	✓	✘	✘
5.2.3. Reduce privileges	✓	✘	✘
5.2.4. Identify system entry points	✘	✓	✘
5.2.5. Map roles to entry points	✘	✓	✘
5.2.6. Map resources to entry points	✘	✓	✘
5.2.7. Scrub attack-surface	✓	✘	✘
5.3. Class design annotation			
5.3.1. Map data elements to resources and capabilities	✘	✓	✘
5.3.2. Annotate fields with policy information	✘	✓	✘
5.3.3. Annotate methods with policy data	✘	✓	✘
5.4. Database security configuration			
5.4.1. Identify candidate configuration	✘	✓	✘
5.4.2. Validate configuration	✘	✓	✘
5.5. Make your product updatable	✓	✘	✘

Source: "On the Secure Software Development Process: CLASP, SDL and Touchpoints compared" (to appear in Elsevier IST)



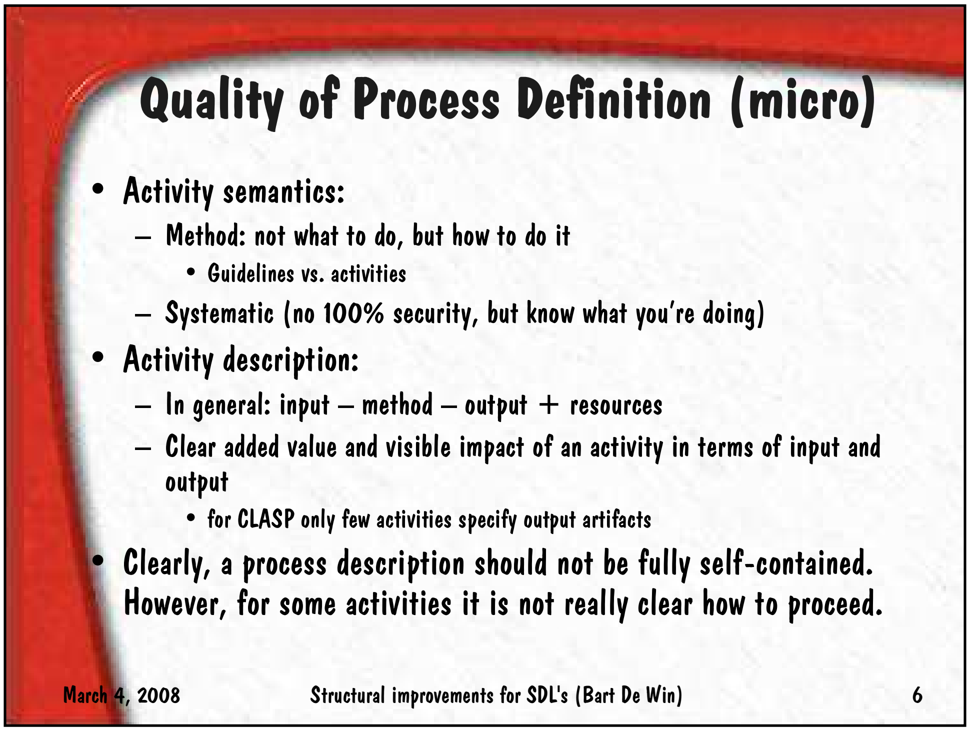
What constitutes a process ?

- A process consists of a temporally ordered sequence of steps (or activities) that, starting from an input state, lead to an outcome by using a set of resources like time or expertise
- The main goal of a process should be to increase systematicity, predictability and coverage.
  - In particular for secure software



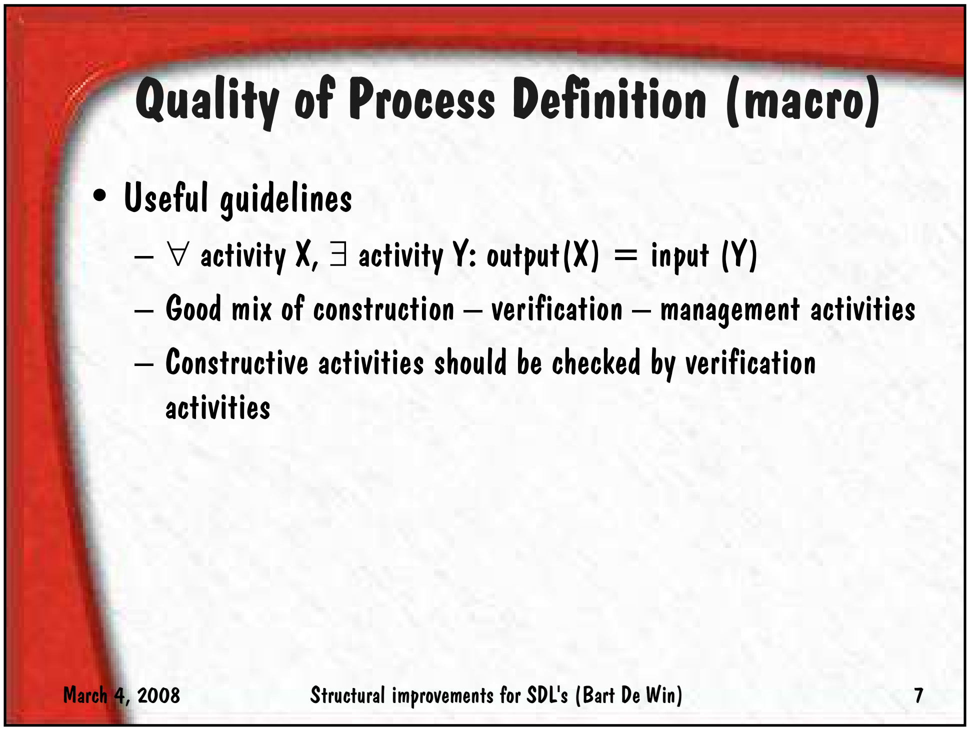
Outline

- Background
- SDL improvements
- Principled process
  - Analysis of support in CLASP
  - Least Privilege in Software Architecture



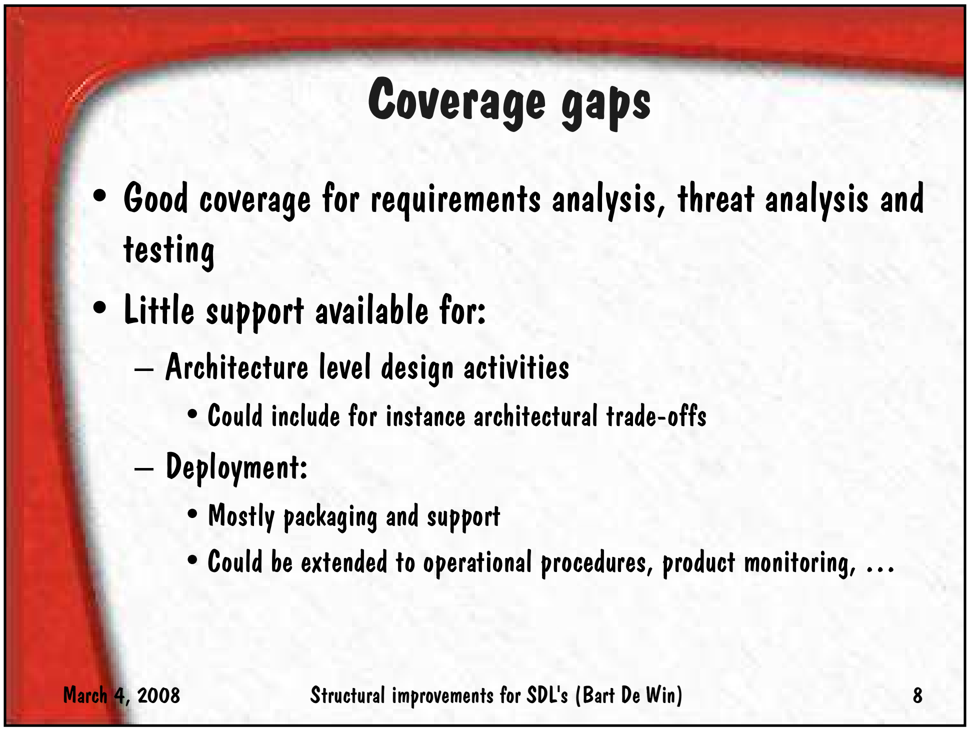
Quality of Process Definition (micro)

- Activity semantics:
  - Method: not what to do, but how to do it
    - Guidelines vs. activities
  - Systematic (no 100% security, but know what you're doing)
- Activity description:
  - In general: input – method – output + resources
  - Clear added value and visible impact of an activity in terms of input and output
    - for CLASP only few activities specify output artifacts
- Clearly, a process description should not be fully self-contained. However, for some activities it is not really clear how to proceed.



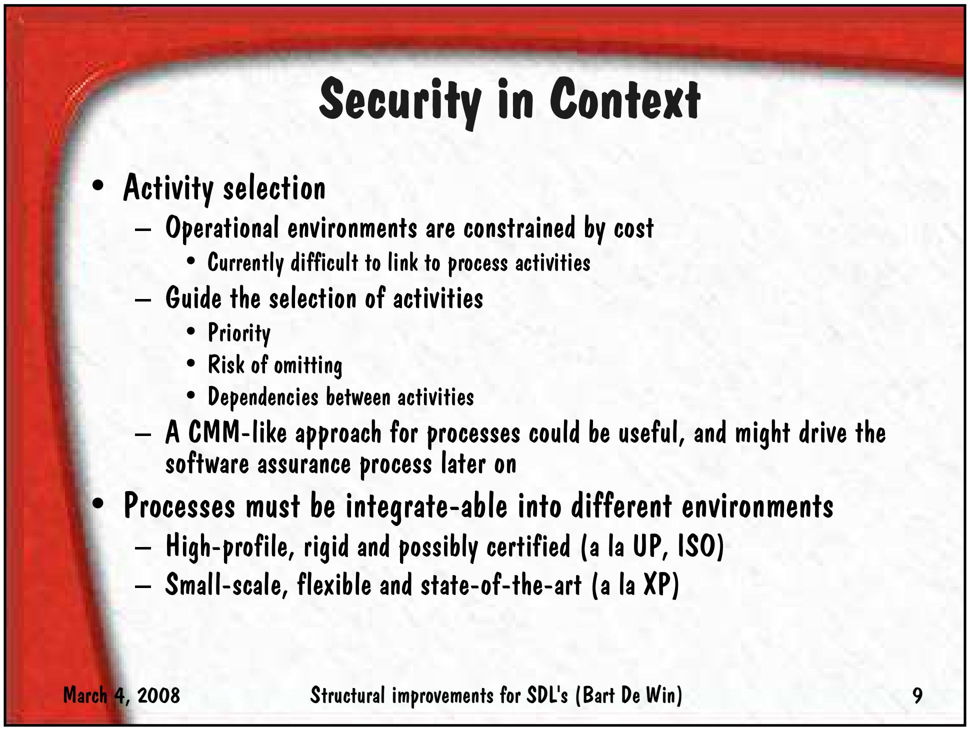
Quality of Process Definition (macro)

- Useful guidelines
  - $\forall$ activity X, $\exists$ activity Y: output(X) = input (Y)
  - Good mix of construction – verification – management activities
  - Constructive activities should be checked by verification activities



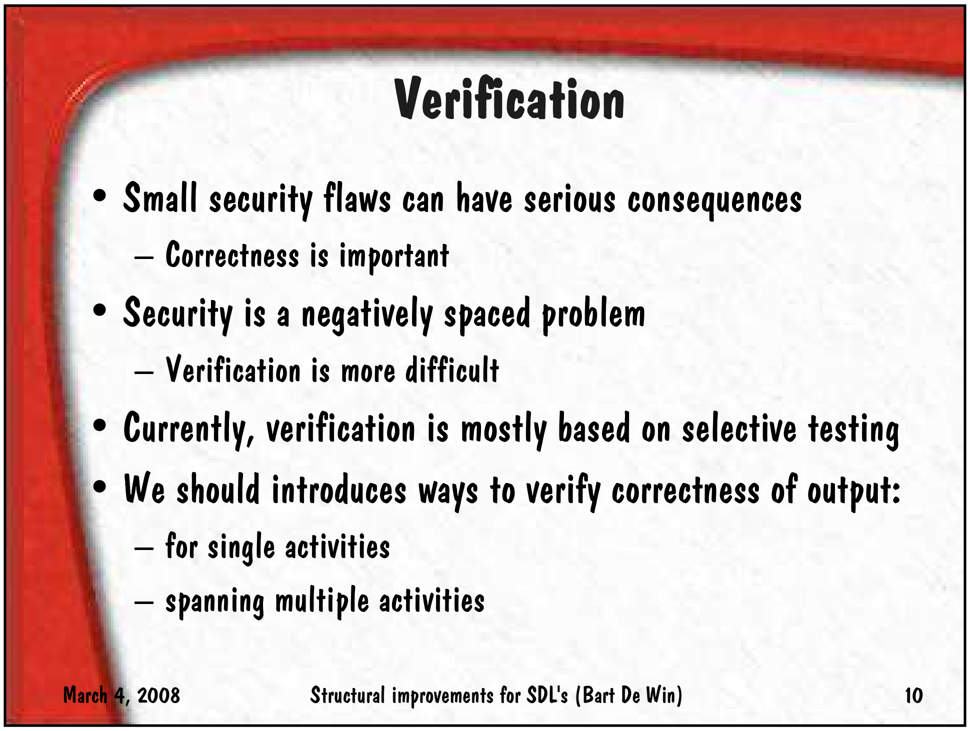
Coverage gaps

- Good coverage for requirements analysis, threat analysis and testing
- Little support available for:
  - Architecture level design activities
    - Could include for instance architectural trade-offs
  - Deployment:
    - Mostly packaging and support
    - Could be extended to operational procedures, product monitoring, …



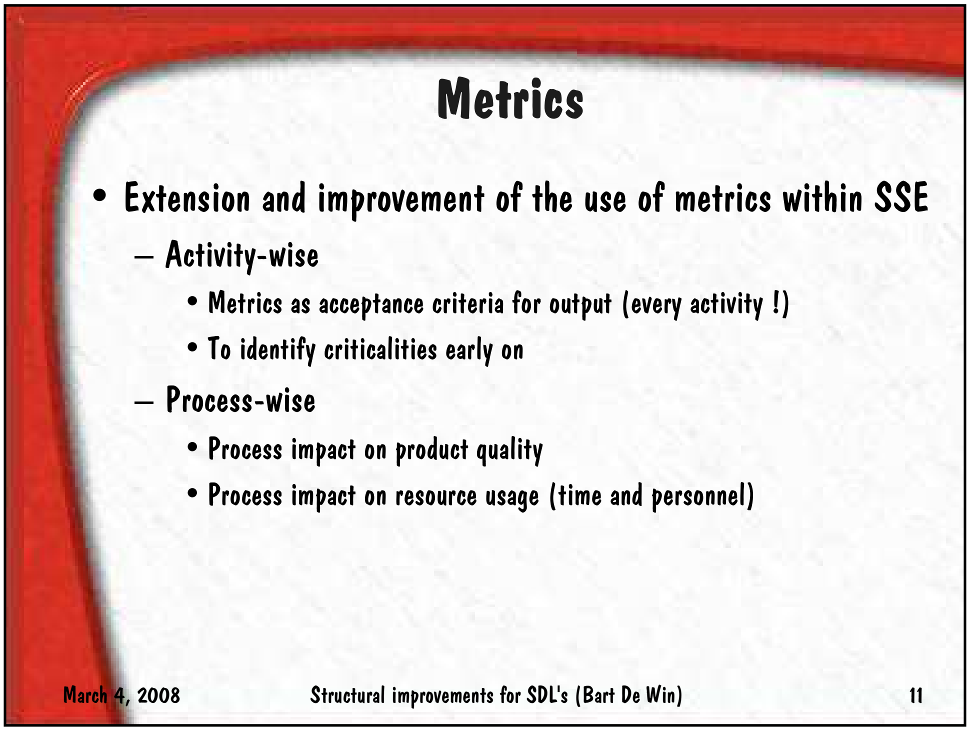
Security in Context

- Activity selection
  - Operational environments are constrained by cost
    - Currently difficult to link to process activities
  - Guide the selection of activities
    - Priority
    - Risk of omitting
    - Dependencies between activities
  - A CMM-like approach for processes could be useful, and might drive the software assurance process later on
- Processes must be integrate-able into different environments
  - High-profile, rigid and possibly certified (a la UP, ISO)
  - Small-scale, flexible and state-of-the-art (a la XP)



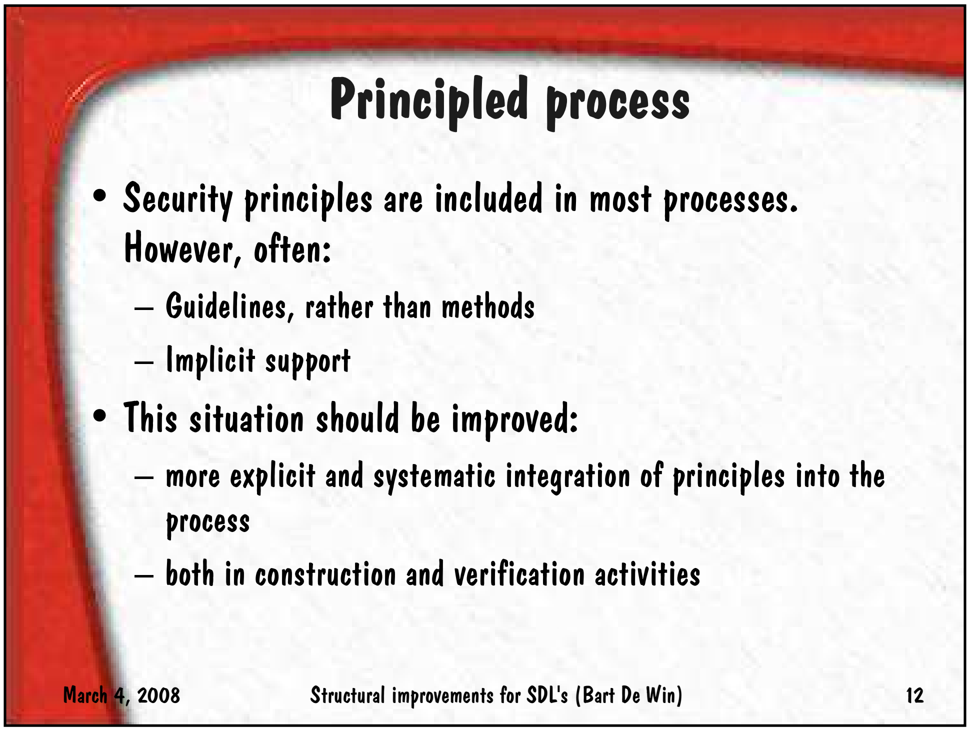
Verification

- Small security flaws can have serious consequences
  - Correctness is important
- Security is a negatively spaced problem
  - Verification is more difficult
- Currently, verification is mostly based on selective testing
- We should introduces ways to verify correctness of output:
  - for single activities
  - spanning multiple activities



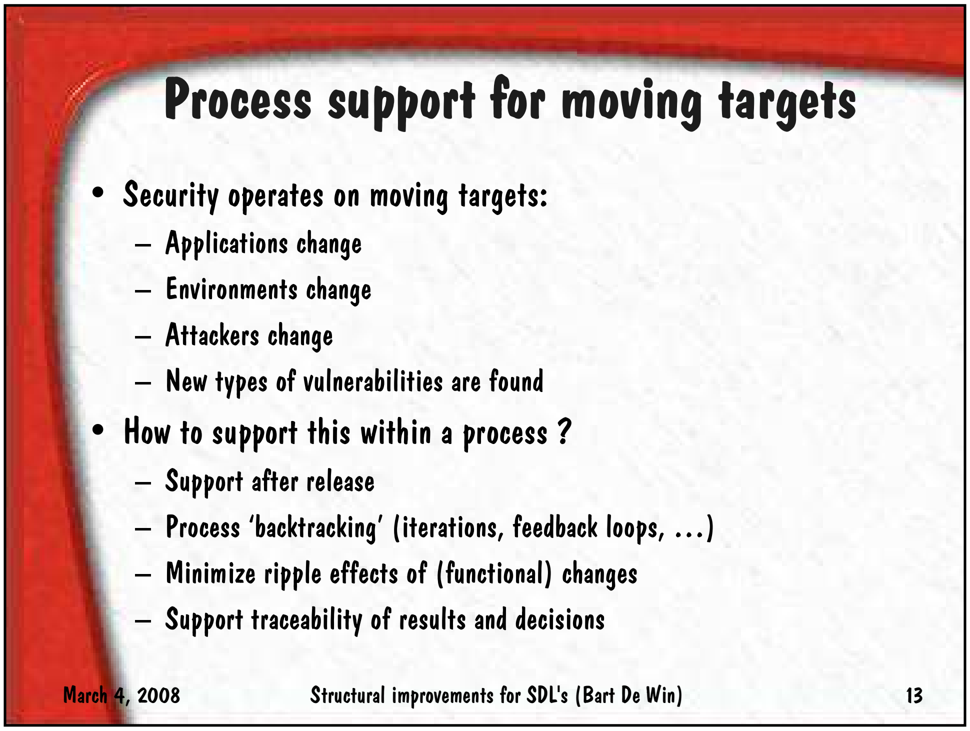
Metrics

- Extension and improvement of the use of metrics within SSE
  - Activity-wise
    - Metrics as acceptance criteria for output (every activity !)
    - To identify criticalities early on
  - Process-wise
    - Process impact on product quality
    - Process impact on resource usage (time and personnel)



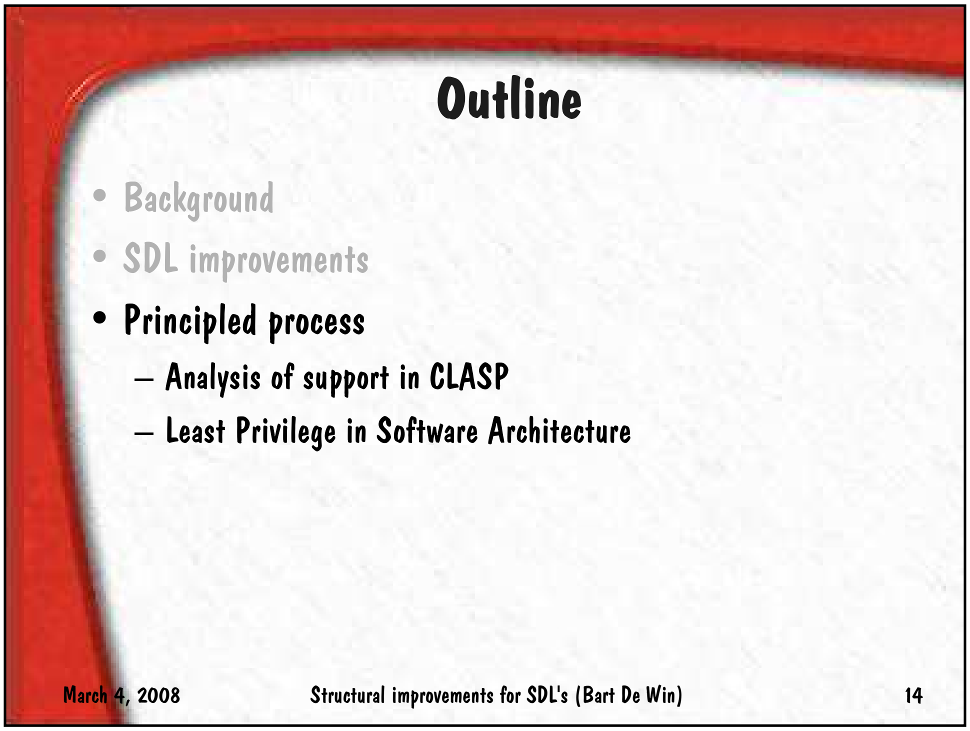
Principled process

- Security principles are included in most processes. However, often:
  - Guidelines, rather than methods
  - Implicit support
- This situation should be improved:
  - more explicit and systematic integration of principles into the process
  - both in construction and verification activities



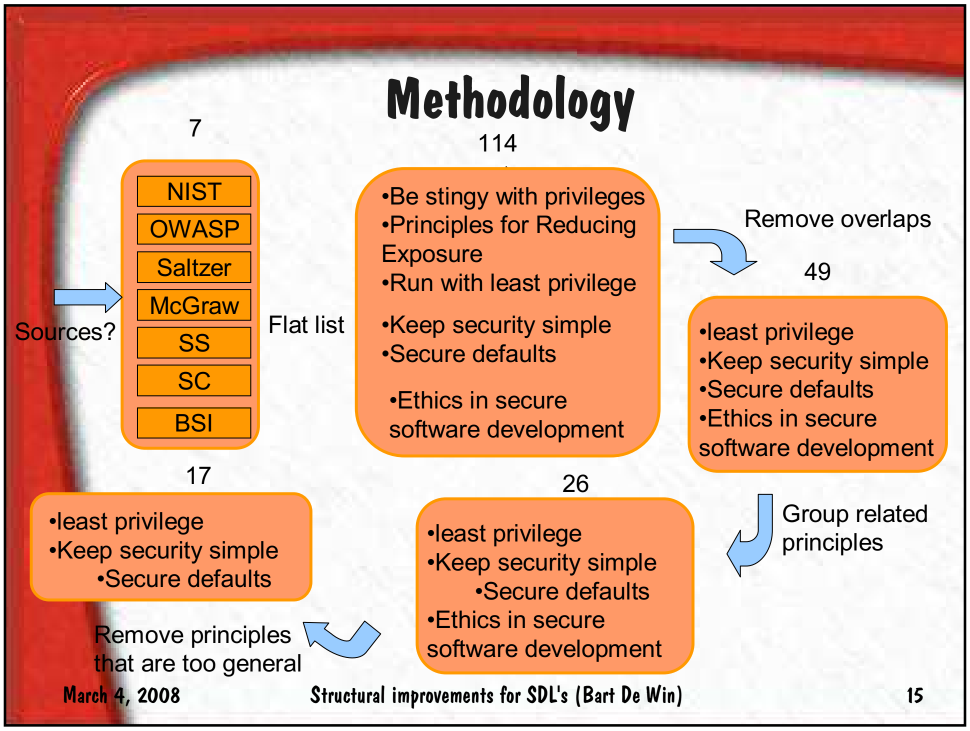
Process support for moving targets

- Security operates on moving targets:
  - Applications change
  - Environments change
  - Attackers change
  - New types of vulnerabilities are found
- How to support this within a process ?
  - Support after release
  - Process 'backtracking' (iterations, feedback loops, ...)
  - Minimize ripple effects of (functional) changes
  - Support traceability of results and decisions



Outline

- Background
- SDL improvements
- Principled process
  - Analysis of support in CLASP
  - Least Privilege in Software Architecture



Methodology

Sources?

7

- NIST
- OWASP
- Saltzer
- McGraw
- SS
- SC
- BSI

Flat list

114

- •Be stingy with privileges
- •Principles for Reducing Exposure
- •Run with least privilege
- •Keep security simple
- •Secure defaults
- •Ethics in secure software development

Remove overlaps

49

- •least privilege
- •Keep security simple
- •Secure defaults
- •Ethics in secure software development

Group related principles

26

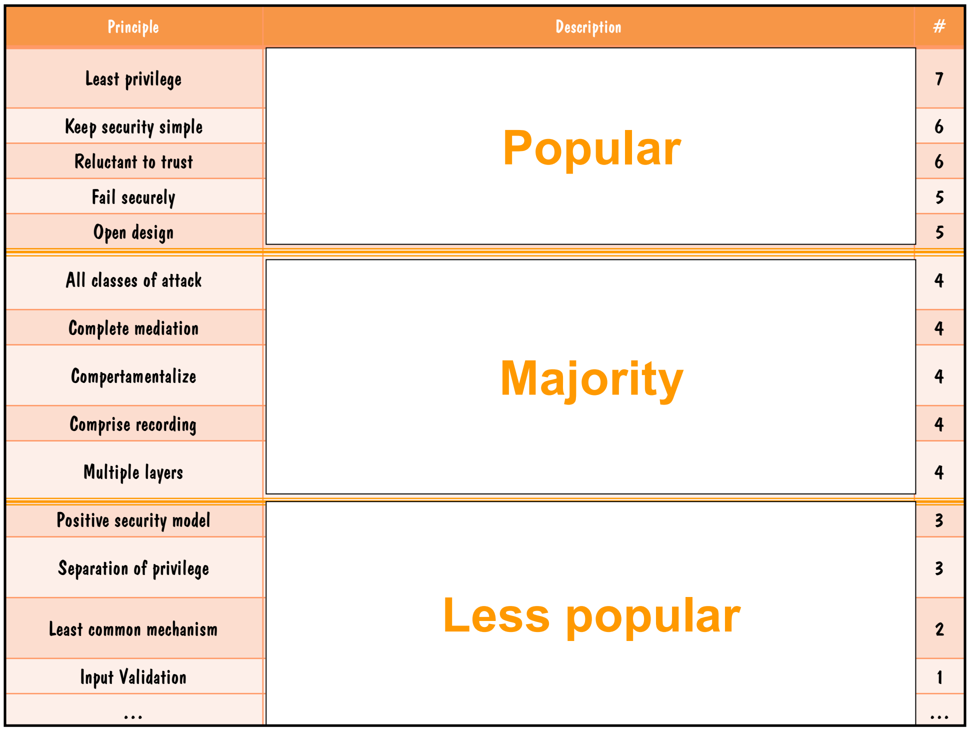
- •least privilege
- •Keep security simple
- •Secure defaults
- •Ethics in secure software development

Remove principles that are too general

17

- •least privilege
- •Keep security simple
- •Secure defaults

March 4, 2008 Structural improvements for SDL's (Bart De Win) 15



Principle	Description	#
Least privilege	Popular	7
Keep security simple		6
Reluctant to trust		6
Fail securely		5
Open design		5
All classes of attack	Majority	4
Complete mediation		4
Compertamentalize		4
Comprise recording		4
Multiple layers		4
Positive security model	Less popular	3
Separation of privilege		3
Least common mechanism		2
Input Validation		1
…		…

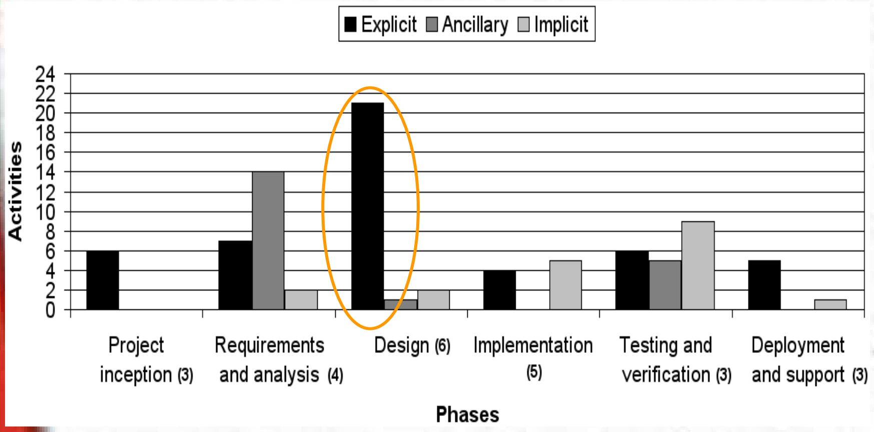
Relationships

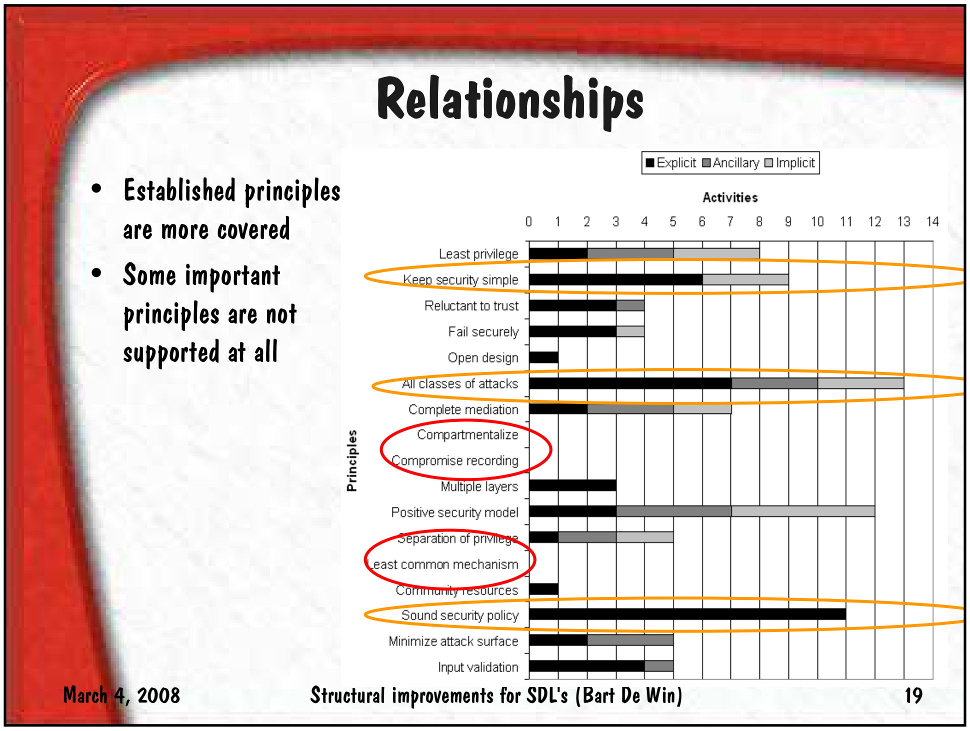
- Support for principles is substantial
  - 85 elementary activities in CLASP
  - 30 mention a principle explicit
  - 9 are ancillary
  - 9 are implicit
  - => 35% of the activities is explicitly connected to principles

Work by Koen Buyens, Riccardo Scandariato and Wouter Joosen

Relationships

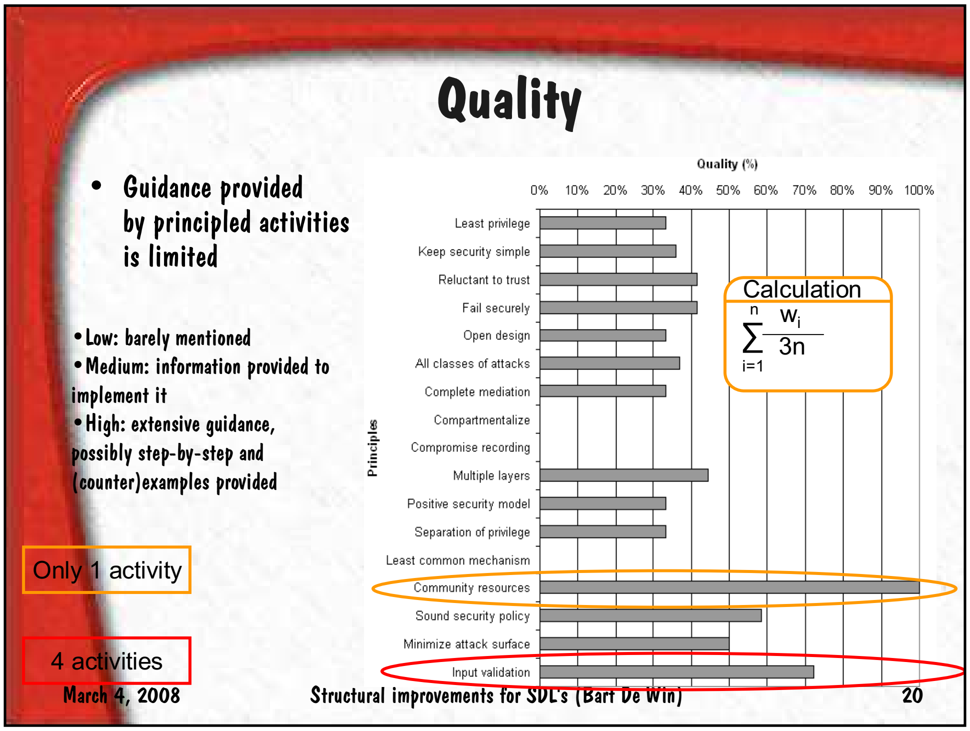
- Early phases are best connected to security principles





Relationships

- Established principles are more covered
- Some important principles are not supported at all



Quality

- Guidance provided by principled activities is limited

- Low: barely mentioned
- Medium: information provided to implement it
- High: extensive guidance, possibly step-by-step and (counter)examples provided

Only 1 activity

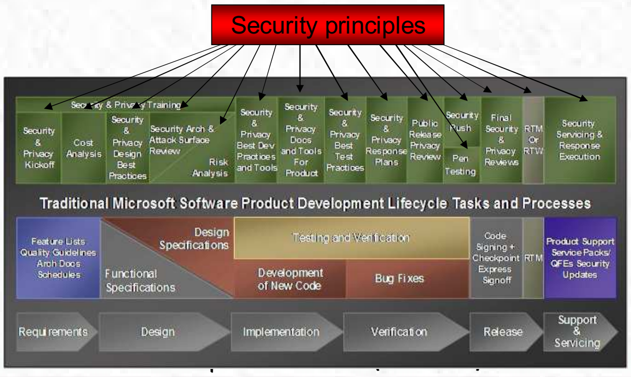
4 activities

Calculation

$$\sum_{i=1}^{n} \frac{w_i}{3n}$$

Our longer term goals

- Materialize security principles within SDL: make them concrete in different activities
  - Different activities, different viewpoints
  - Security optimization vs. cost optimization

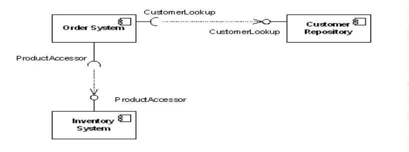

Security principles
Security & Privacy Training
Security & Privacy Kickoff
Cost Analysis
Security & Privacy Design Best Practices
Security Arch & Attack Surface Review
Risk Analysis
Security & Privacy Best Dev Practices and Tools
Security & Privacy Docs and Tools For Product
Security & Privacy Best Test Practices
Security & Privacy Response Plans
Public Release Privacy Review
Security Push
Pen Testing
Final Security & Privacy Reviews
RTM Or RTW
Security Servicing & Response Execution
Traditional Microsoft Software Product Development Lifecycle Tasks and Processes
Feature Lists Quality Guidelines Arch Docs Schedules
Design Specifications
Functional Specifications
Testing and Verification
Development of New Code
Bug Fixes
Code Signing + Checkpoint Express Signoff
RTM
Product Support Service Packs/ QFEs Security Updates
Requirements
Design
Implementation
Verification
Release
Support & Servicing

Least Privilege as an example

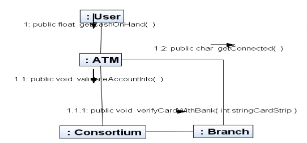
Global security policy	determine the project-wide goals wrt. LP
Map roles to capabilities	analyze for 'over-assignment' ; introduce new roles if necessary
Threat modeling	spot LP threats in (M)UCs ; categorize UCs according to sensitivity
Requirements specification	include specific LP constraints
Architectural design	massage architecture (e.g., by splitting components)
Arch. threat modeling	spot LP threats in architecture
Attack surface reduction	reduce privileges in entry points (explicit in SDL)
…	…

LP @ architecture level

- Available information:
  - Component and Connector diagram



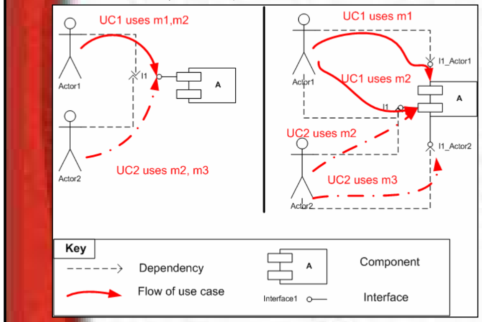
  - Collaboration diagram (UC)


: User
1: public float getCashOnHand( )
1.2: public char getConnected( )
: ATM
1.1: public void validateAccountInfo( )
1.1.1: public void verifyCardWithBank(int stringCardStrip)
: Consortium
: Branch

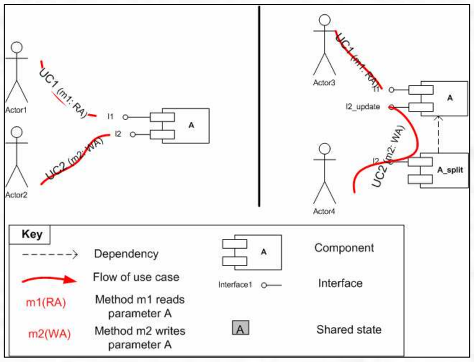
- Idea: transform SW architecture into artifact that is 'better geared towards' enforcing least privilege
  - Architectural structure vs. architectural policy
- Different strategies: <u>splitting</u>, rewiring, introduce established components, …

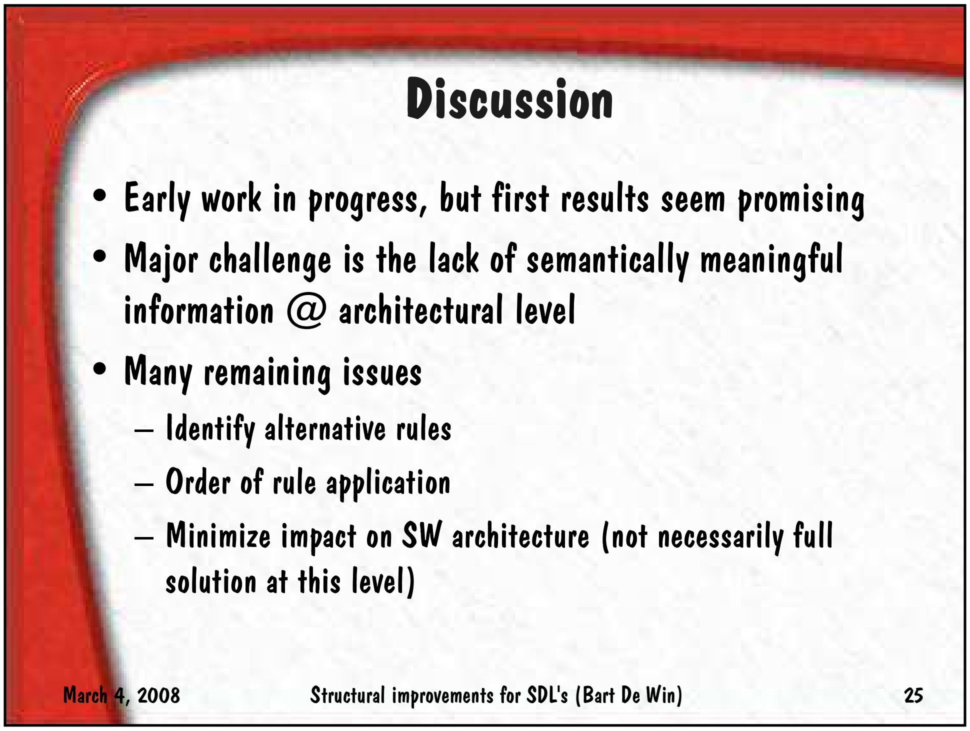
Basic rules

Splitting interfaces



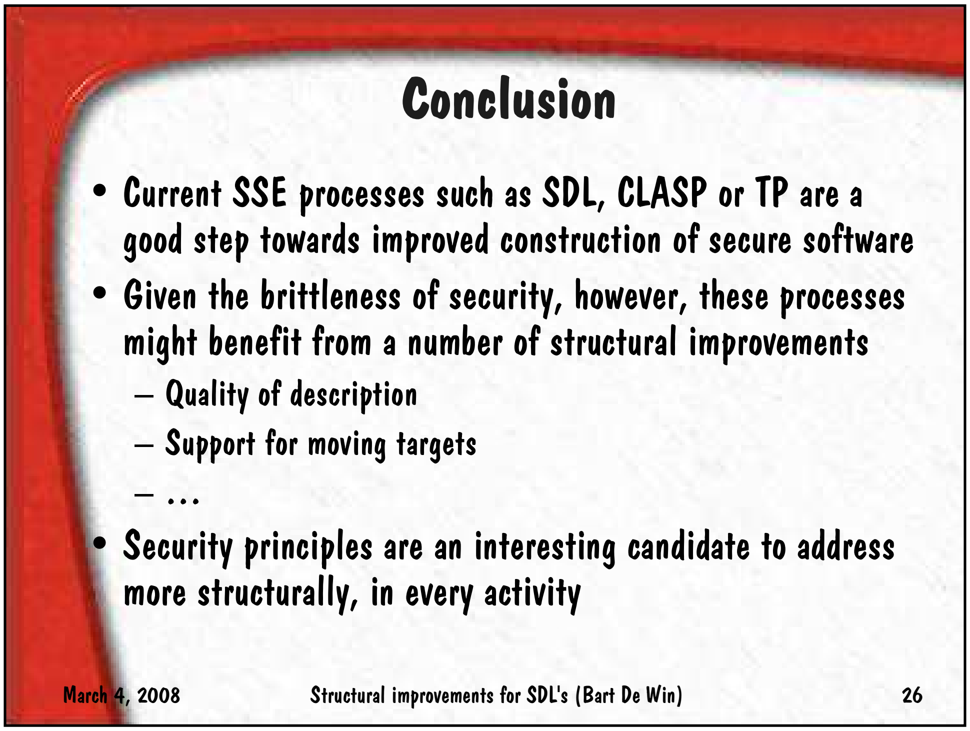
Splitting components





Discussion

- Early work in progress, but first results seem promising
- Major challenge is the lack of semantically meaningful information @ architectural level
- Many remaining issues
  - Identify alternative rules
  - Order of rule application
  - Minimize impact on SW architecture (not necessarily full solution at this level)



Conclusion

- Current SSE processes such as SDL, CLASP or TP are a good step towards improved construction of secure software
- Given the brittleness of security, however, these processes might benefit from a number of structural improvements
  - Quality of description
  - Support for moving targets
  - …
- Security principles are an interesting candidate to address more structurally, in every activity