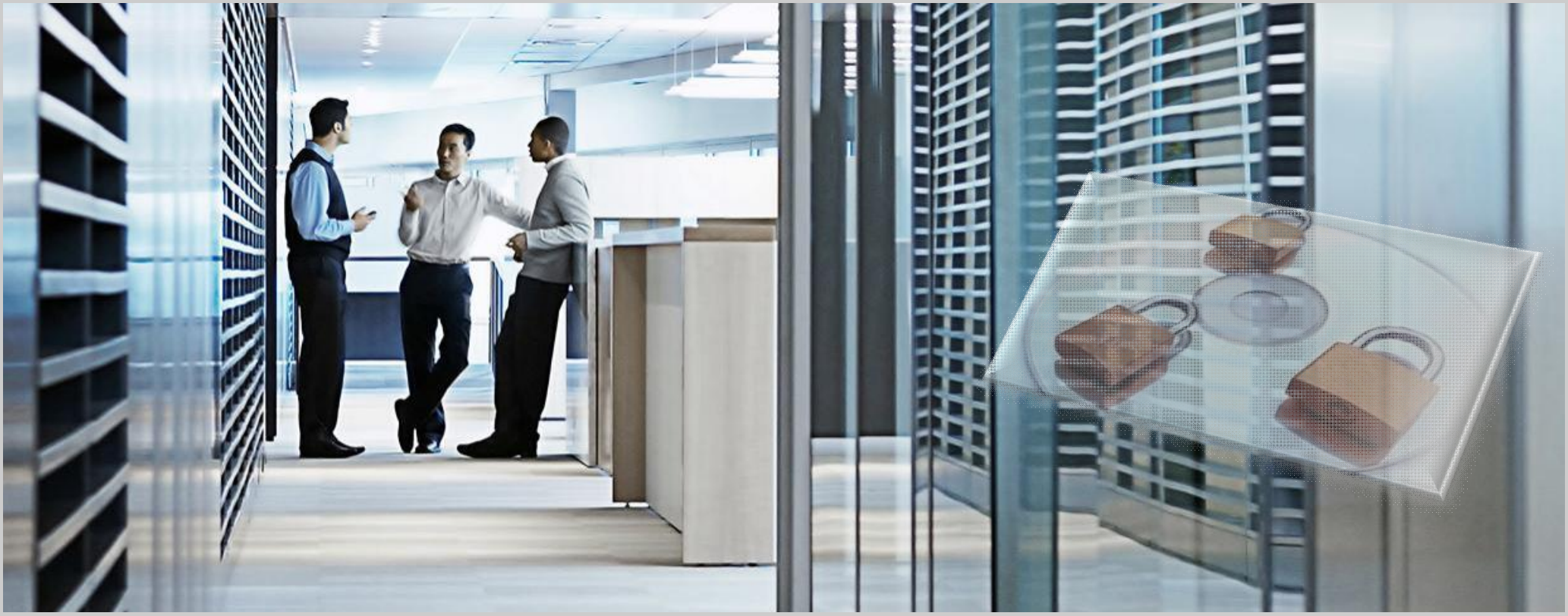


Towards Timeless Software Security

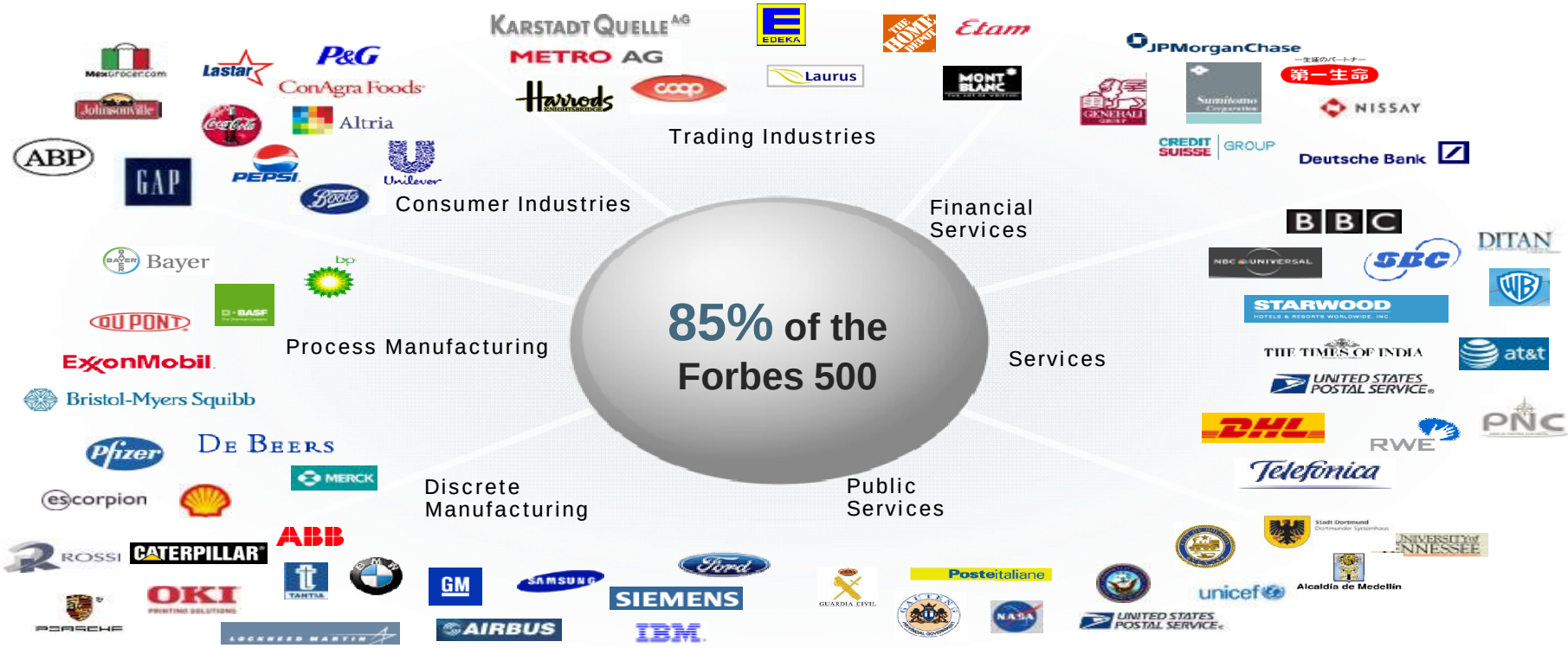
Kaj van de Loo
SVP, Technology Strategy
Office of the CTO

Yuecel Karabulut, Ph.D.
Chief Security Advisor and Head of Security Strategy, Technology Strategy
Office of the CTO

OWASP Summit Keynote
Feb 25, 2010

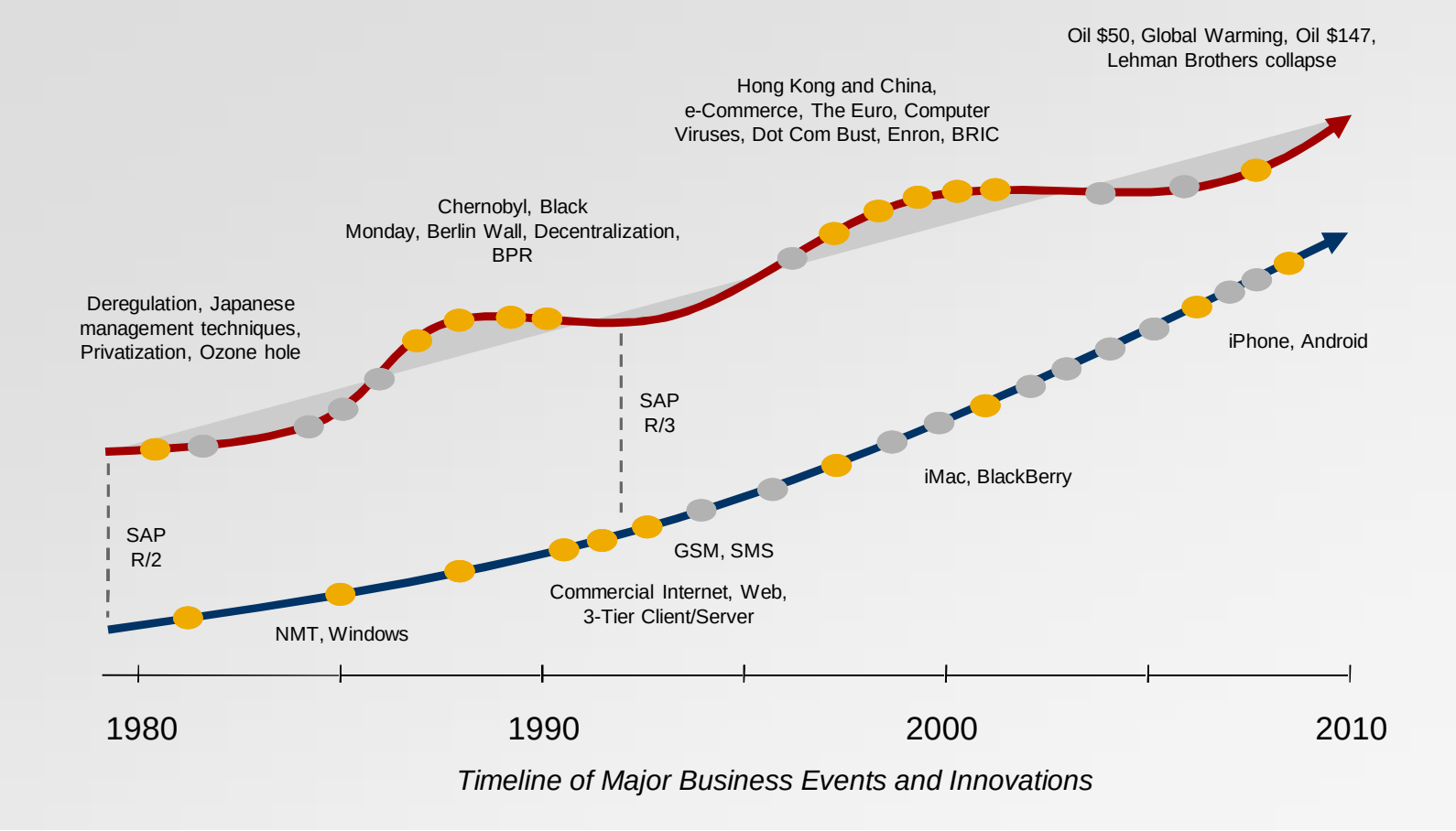
SAP

Decades-Long Relationships With the World's Largest Enterprises



SAP is in a unique position to transform change into an opportunity for our customers

Change Is Disruptive



Timeline of Major Business Events and Innovations

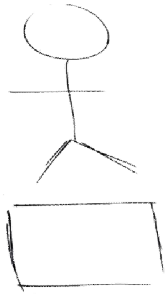
The Governing Dynamic
Preparing for Change

Is there an architecture that assumes continuous change?

Can our systems non-disruptively consume innovation?

Even fundamental innovation?

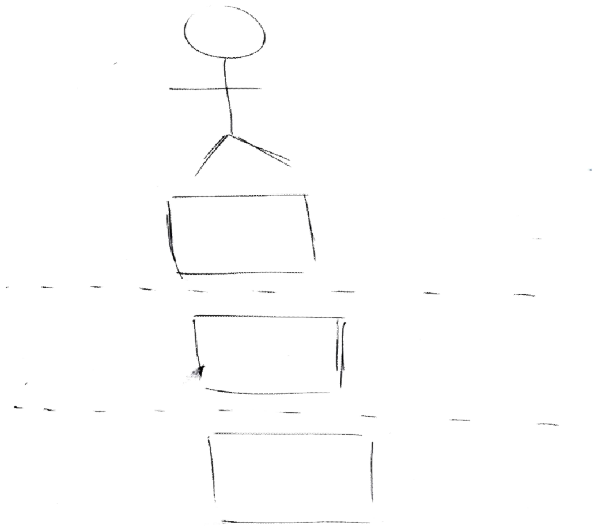
Timeless Software
Design-thinking



Focus on the needs and expectations of the user

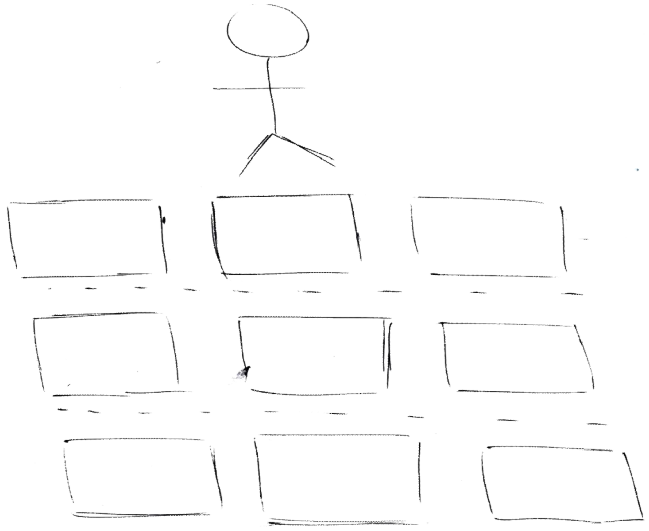
Timeless Software

Separation of concerns between layers



Introduce a limited number of layers with clear contract to allow different speed of innovation

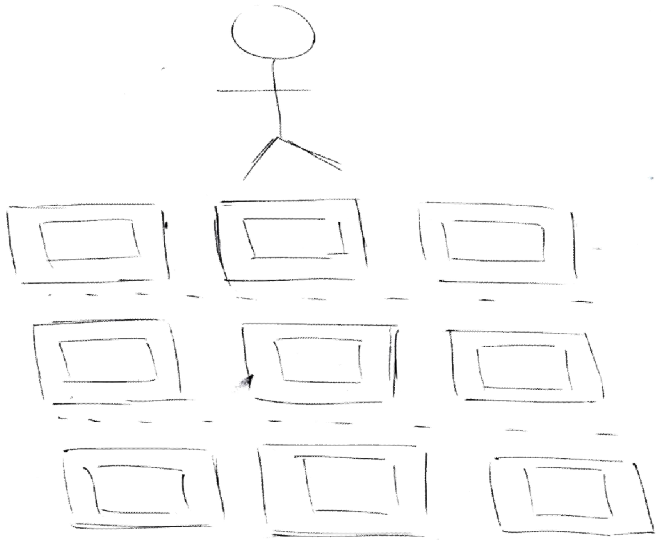
Timeless Software
Componentization



Identify a reasonable set of components to reduce dependencies, increase flexibility and allow for reuse

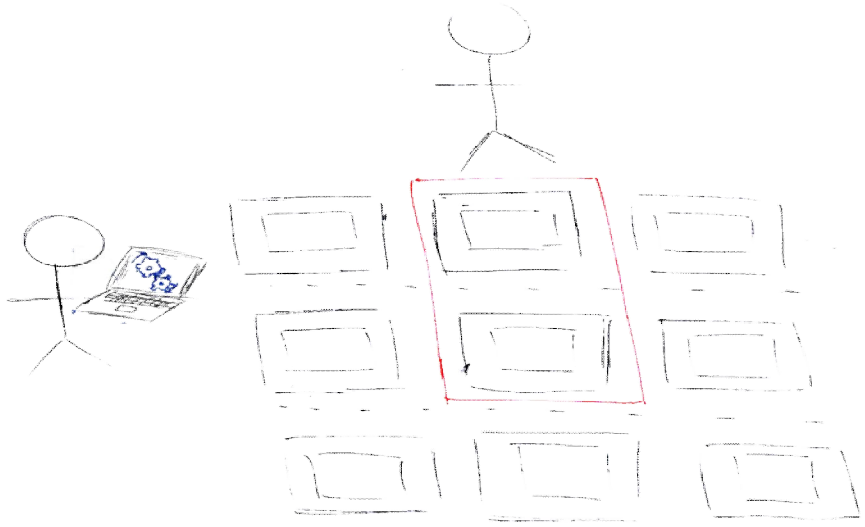
Timeless Software

Decoupling of “content” and “container”



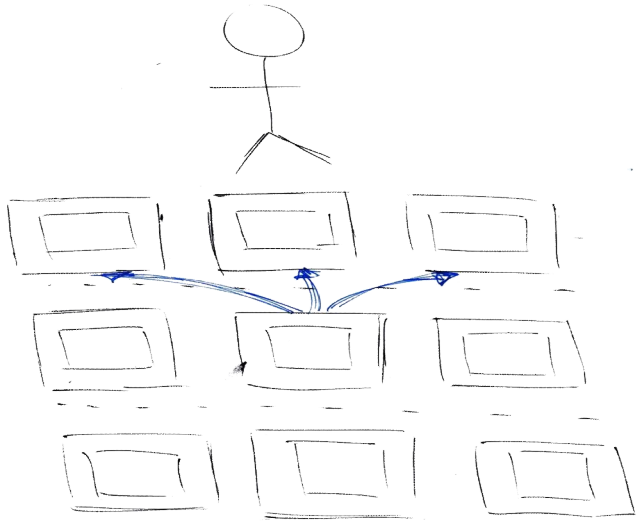
Separate content from container to allow new content without the need for a container upgrade. Allow partners and customers to create their own content.

Timeless Software
Design locality



Optimize design experience for the user. LOB developer should stay in one environment, SAP expert maybe in another.

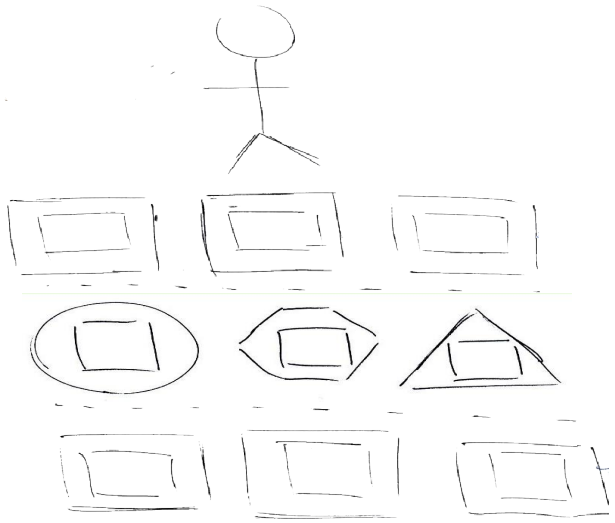
Timeless Software
Adaptable provisioning



Reduce assumptions about usage of content of a component in upper layers

Timeless Software

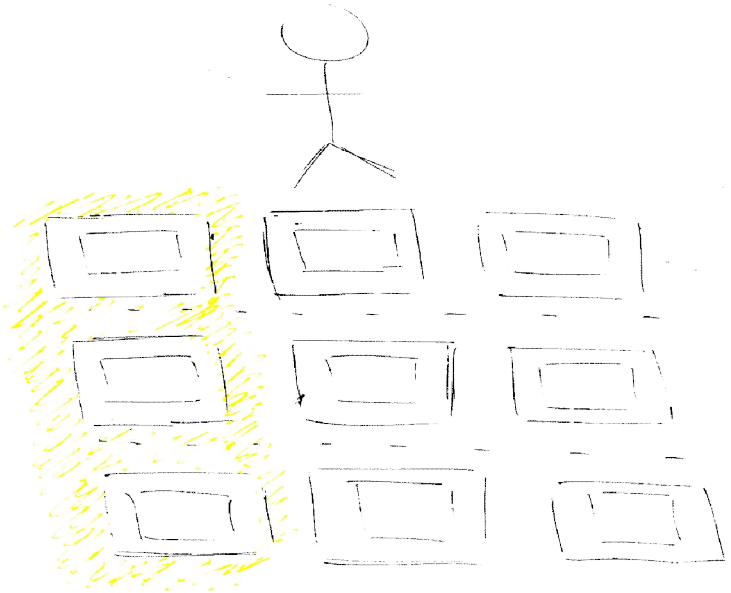
Separating intent from optimization



Ensure that optimizations in a container can be done without changes to the content

Timeless Software

Optimizing across layers of abstraction



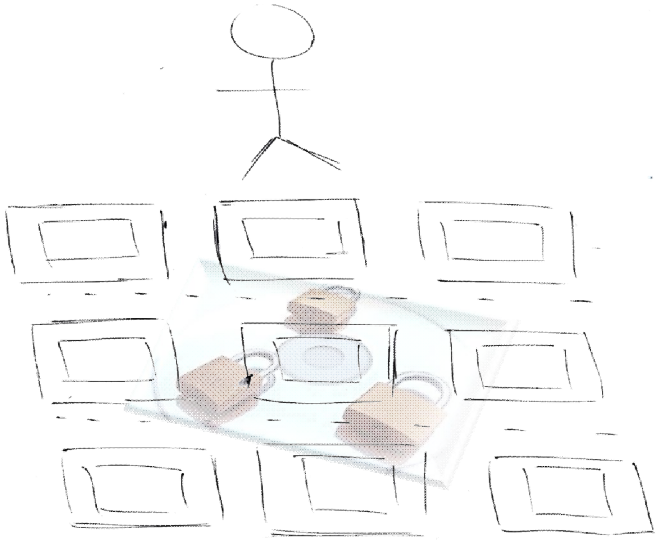
Optimize across layers without violating the overall contract.

What are the security aspects of timeless software?

Does the endeavour to be “timeless” destroy or support security properties?

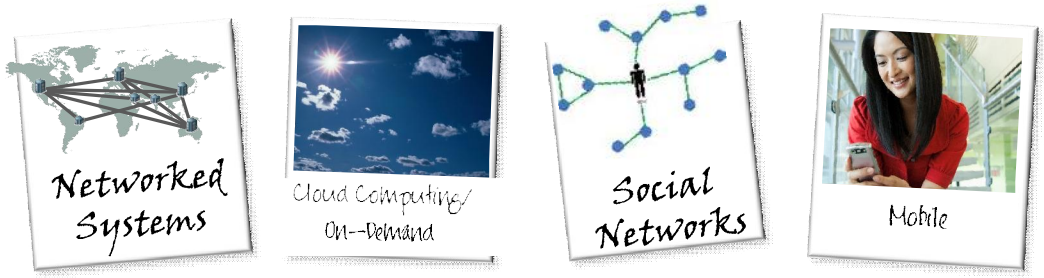
Can security properties be enhanced and integrated with timeless software principles?

Can security built in a way that it assumes and support continues change?



Application Security is a Challenging Area

- Applications are complex
- Continuously evolving architectural paradigms and new software delivery models
- Attackers are focusing on the application layer, getting smarter and using sophisticated tools



Why is traditional industry approach not enough?

Reason #0

Security is Not Properly Embedded into CS/Engineering Curricula

Reason #1

Majority of Security Architectures and Governance Processes Do Not Assume Continuous Change

Reason #2

Shipping Vulnerability Free Software is Hard

What should we do?


Evolve From a “Today” Centric Security
Thinking to a Timeless Software Security
Thinking Which Assumes Continuous Change

How can we do this?

Example Approaches For Timeless Software Security

- Decoupling of content and container
  - Self-Defending Data
  - Attack Surface Measurement & Reduction
- Separating intent from optimization
  - Model Driven Security
  - Attack Surface Measurement & Reduction

History of Attack Surface Measurement and Reduction

- 1975 Design Principles by Saltzer and Schroeder
- 2003 Relative Attack Surface Quotient by Michael Howard of Microsoft
- 2003 Generalized Attack Surface Method by Howard, Pincus and Wing
- 2007 Formalized Attack Surface Measurement Method by Manadhata and Wing of CMU
- 2009 Refined Attack Surface Measurement Method for SAP Software Systems and The MASUBA Tool by Karabulut (SAP) and Manadhata (CMU)

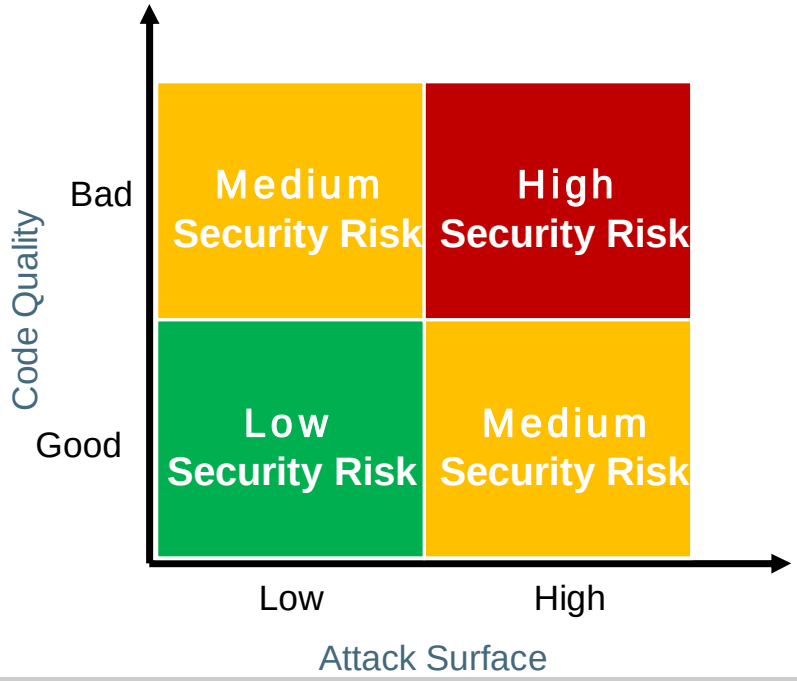
Why to Minimize the Security Risk with Future Vulnerabilities

A system's attack surface is defined in terms of the system's resources <**Methods, Channels, Data Items**>

Smaller attack surface
→ less security risk

Makes the exploitation harder

Lowers the damage potential



Abstract Attack Surface Measurement Method

Manadhata and Wing, 2007

1. Determine
   - M: set of entry points and exit points
   - C: set of channels
   - I: set of untrusted data items

2. Estimate
   - Damage potential-effort ratio (***der***) for each individual m, c, and i

3. Compute Attack Surface (AS)

The higher the damage potential, the higher the contribution

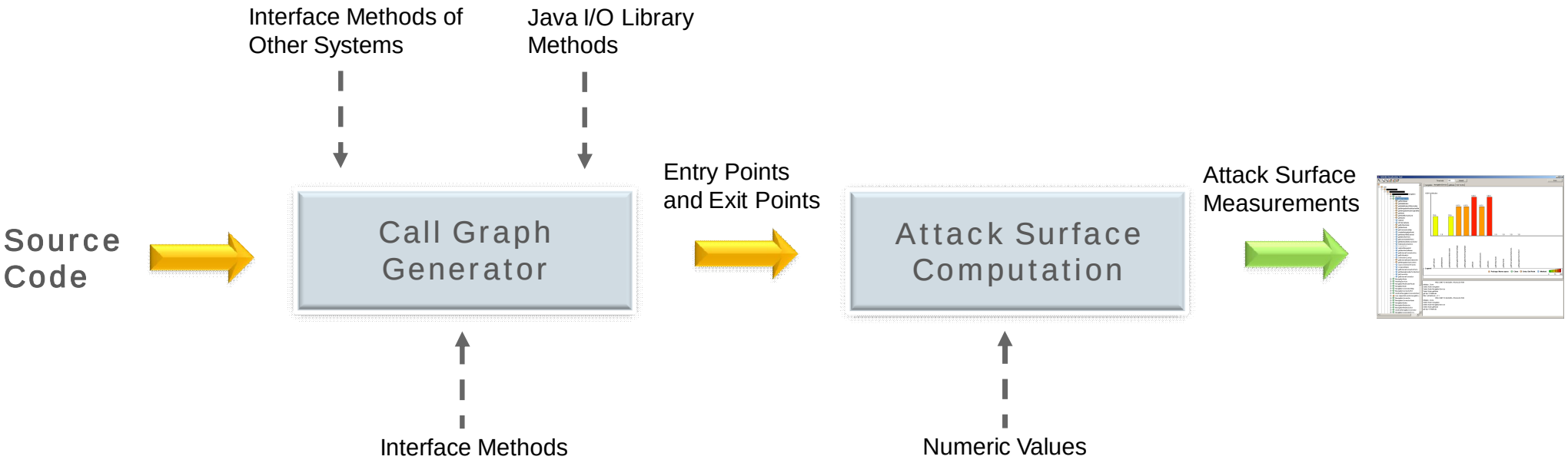
The higher the effort, the lower the contribution

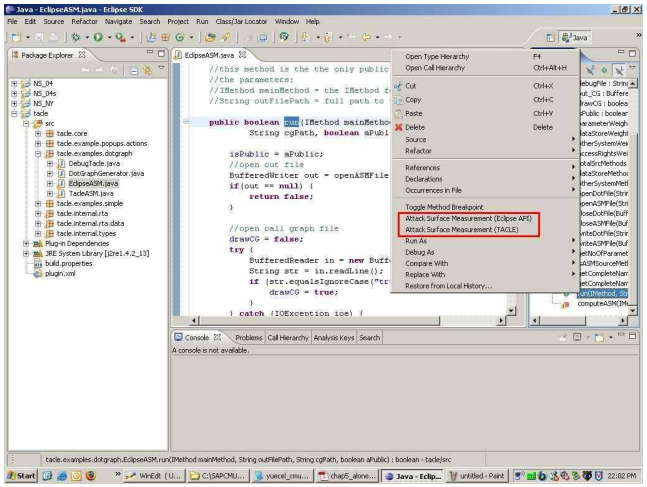
$$AS = < \sum_{m \in M} der(\mathrm{m}) \ , \ \sum_{c \in C} der(\mathrm{c}) \ , \ \sum_{d \in I} der(\mathrm{i}) >$$

Attack Surface Measurement Method for SAP Software Systems: A Case Study

- Chose a core SAP NetWeaver component implemented in Java
- The component doesn't use any persistent data items and open only one TCP socket
- Hence we only considered the method dimension of the attack surface in our measurement
- Two Key Steps
  1. Identification of Entry Points and Exit Points
  2. Estimation of the Damage Potential-Effort Ratio

Attack Surface Measurement Steps




Java - EclipseASM.java - Eclipse SDK
Open Type Hierarchy
Open Call Hierarchy
Cut
Copy
Paste
Delete
Source
Refactor
References
Declarations
Occurrences in File
Toggle Method Breakpoint
Attach Surface Measurement (Eclipse API)
Attach Surface Measurement (TACLE)
Run As
Debug As
Compare With
Replace With
Restore from Local History...

Demo


MASUBA Visualization Tool
METHOD INFORMATION
Data Storage
Parameter
Other

Numeric Value Assignment*

- Assign numeric values to sources of input and access rights levels to compute damage potential-effort ratio
- Internal threat modeling process
  - Identified possible attacks on the component
  - Assigned severity ratings to the attacks
- We correlated the sources of input with possible attacks on the component
- Total ordering among the access rights level: internal > public

Attribute	Avg. Rating	Value
Parameter	5	35
Data Store	3	18
Other System	1	1

Access Rights	Value
Public	1
Internal	18

* P. Manadhata, Y. Karabulut, J. Wing: Measuring the Attack Surfaces of Enterprise Software, ESSOS 2009, Belgium

Open Questions in Timeless Software Security

- **Data-centric security** How can we associate usage policies with content in a container-independent way, such that these policies can be enforced or checked as the content is migrated between containers?

- **Verifiable Secure Composition:** How can we prove that different compositions of security policies, protocols and mechanisms are overall secure?

- **Updating features without breaking security** As technologies change what happens to the security properties? How can we update features without breaking security? How can the impact of technological change on existing architectures be systematically assessed?

- **Updating security without breaking features** How can security controls be designed and integrated into software in such a way that they can be updated without breaking the functional properties and qualities of software?

What Does This Mean for Cloud Computing?

- Cloud Computing is not necessarily more or less secure than current on-premise environments
- Old security problems in new setting... Some new security problems...
- Two main areas of innovation needed for Cloud security
  - Security controls
  - Secure software development processes

Concluding Remarks...

- We live in a networked world... Threats have changed
- We are under no illusion that we're done with security
- But we need to change our outlook...

LAST BUT NOT LEAST...

- Balancing security and usability is hard but extremely important
- Balancing security and performance is hard but extremely important

Thank you!