Model-based Security Engineering for Compliance with Regulatory and Business Requirements

Jan Jürjens

Department of Computing
The Open University, GB

J.Jurjens@open.ac.uk

http://www.umlsec.org

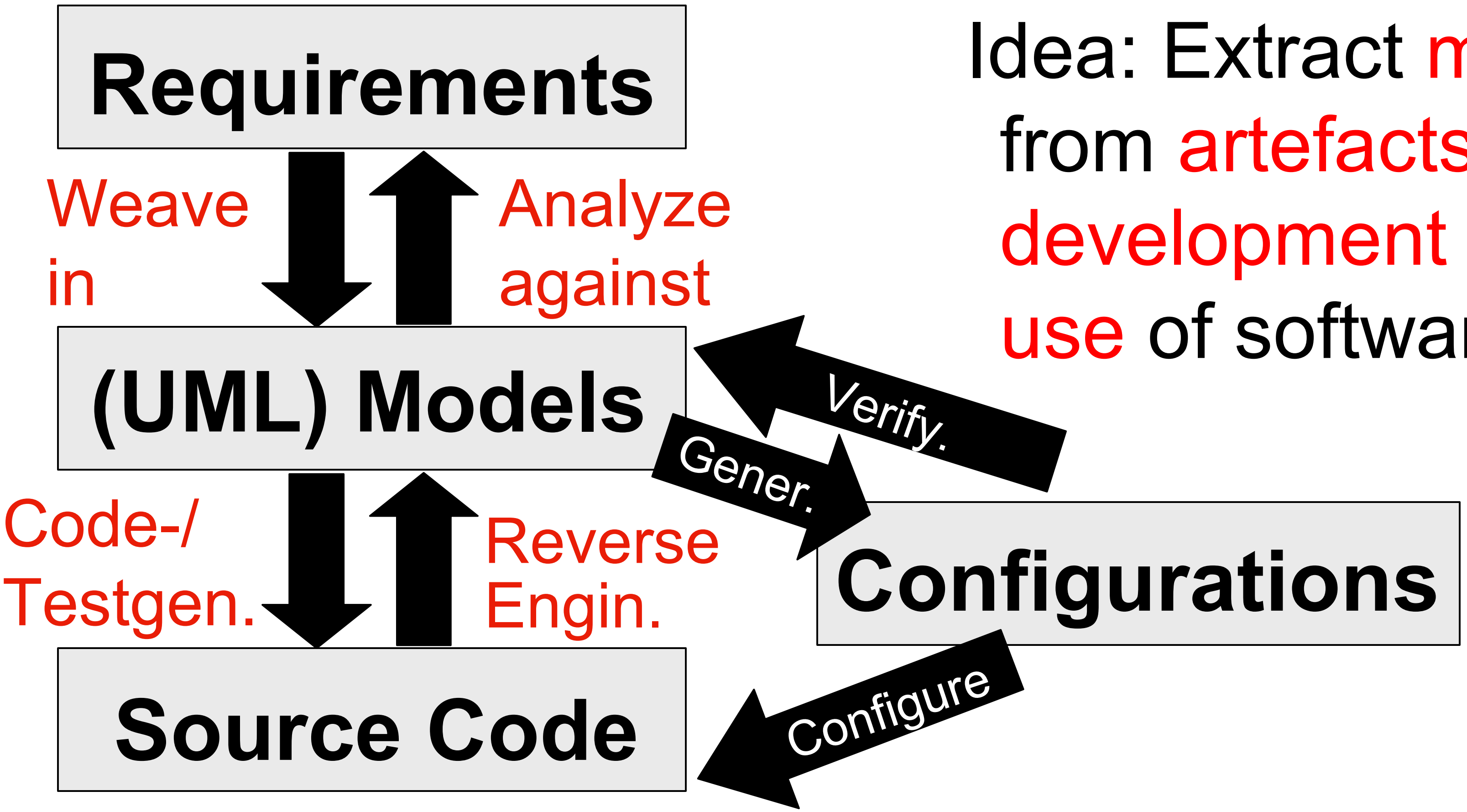
Challenge: Security

Security is holistic property:

- Attackers often circumvent (not: break) mechanisms.
- Transform (in)secure components to secure systems ?

„Those who think that their problem can be solved by simply applying cryptography don`t understand cryptography and don`t understand their problem" (B. Lampson / R. Needham).

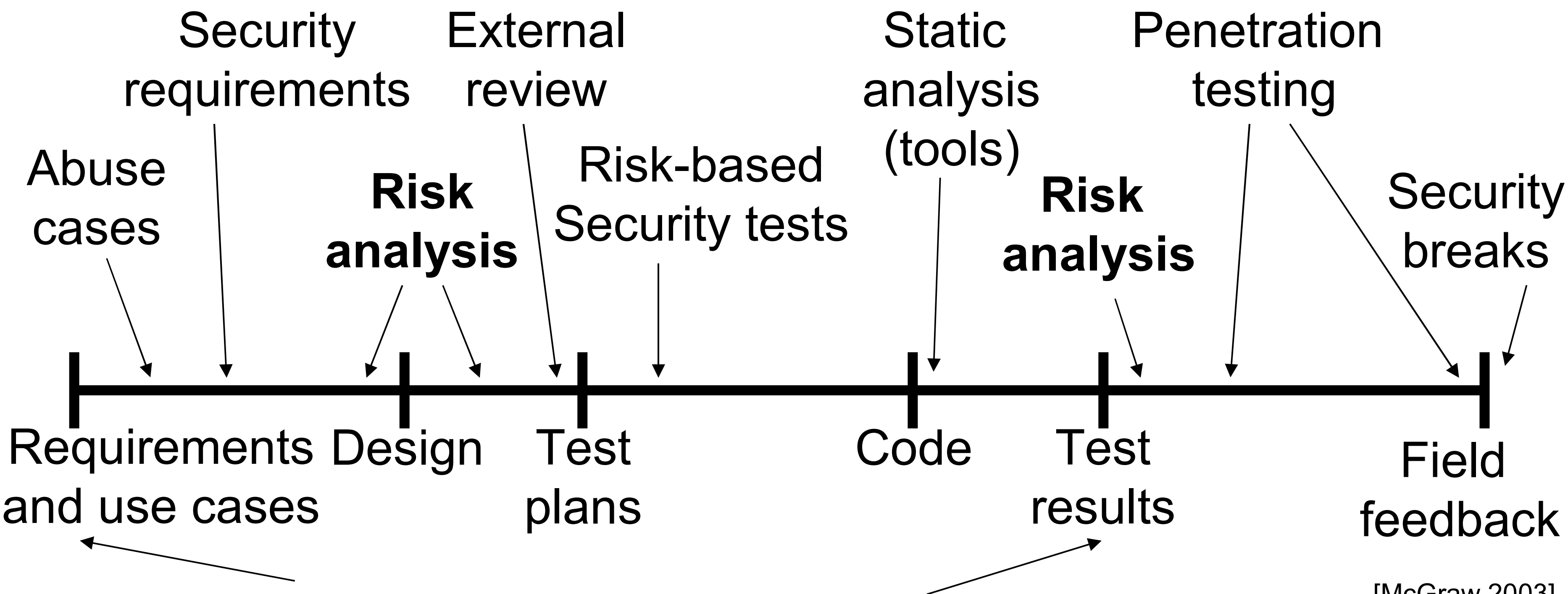
Model-based Security Engineering



Idea: Extract models from artefacts in development and use of software.

➔ Tool-supported, theoretically sound, efficient automated security design & analysis.

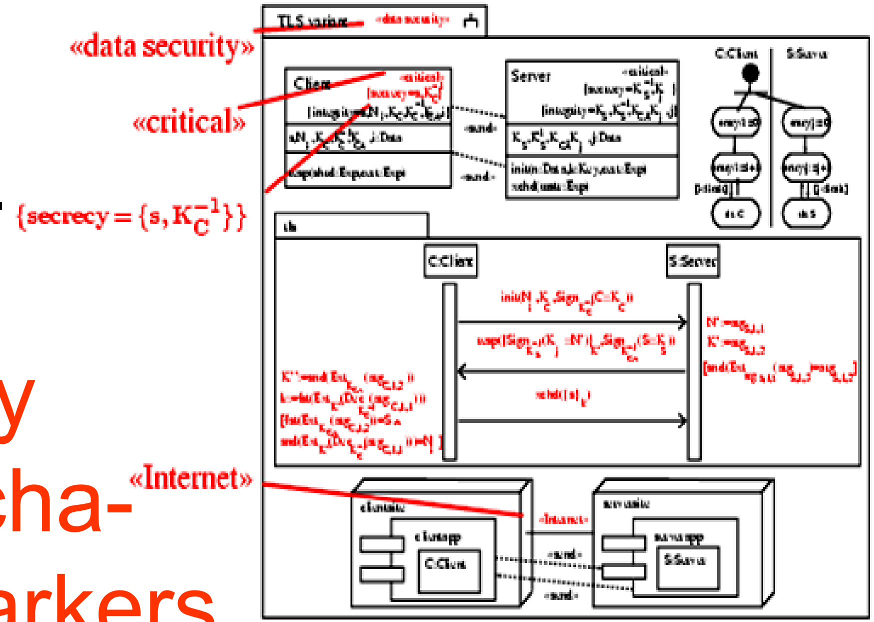
Secure System Lifecycle



Model-based Security Engineering

Note: emphasis on high-level requirements.

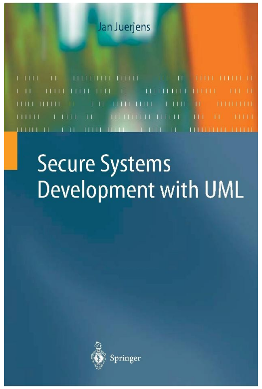
UMLsec



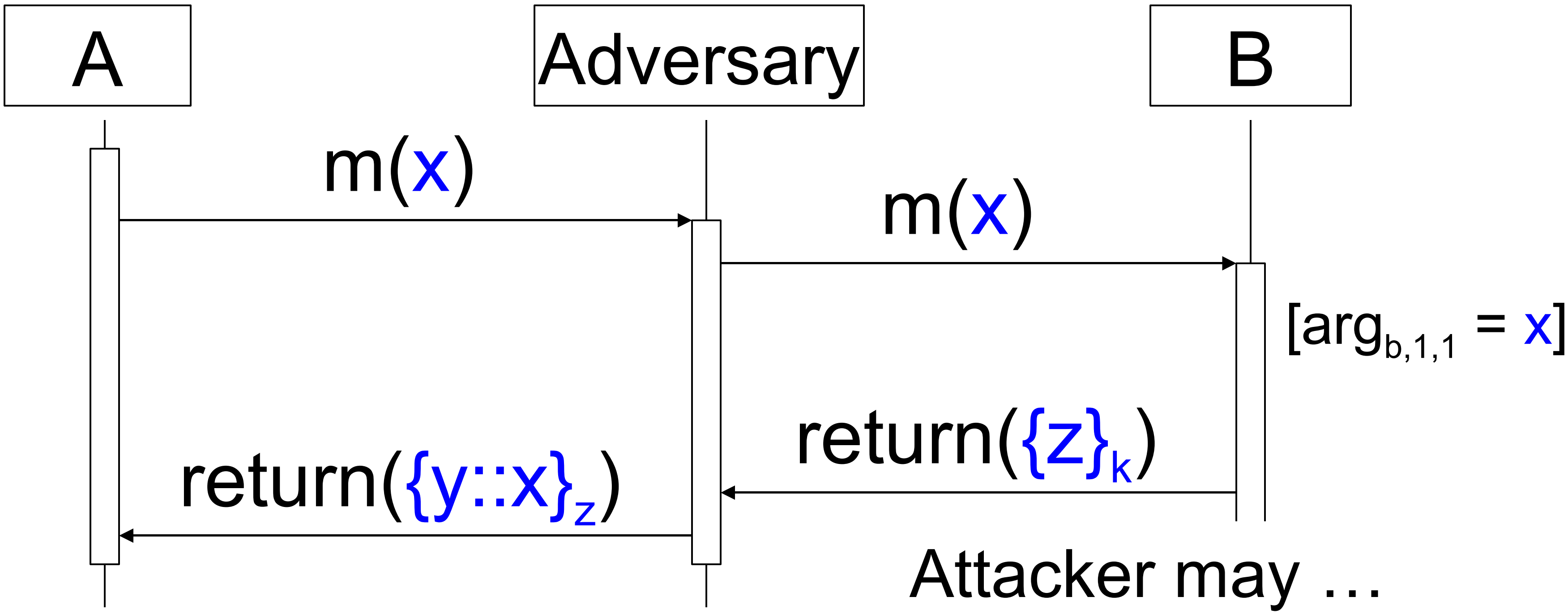
Insert recurring security requirements, adversary scenarios, security mecha-nisms as predefined markers.

Verify associated logical constraints using model checkers and ATPs (based on formal semantics).

Ensures that UML specification enforces relevant security requirements wrt Dolev-Yao type adversaries. [FASE01,UML02,FOSAD05,ICSE05]



Example: Crypto-based Distributed System



Attacker may …

- control system parts,
- know data in advance,
- intercept messages,
- delete messages,
- inject messages.

Adversary knowledge: k^{-1}, y, x, $\{z\}_k$, z

(cf. [Dolev, Yao 1982])

Security Analysis in First-order Logic

Approximate adversary knowledge set from above:

Predicate *knows(E)* meaning that adversary may get to know *E* during the execution of the system.

E.g. secrecy requirement:
For any secret *s*, check whether can derive *knows(s)* from model-generated formulas using automatic theorem prover. [ICSE05]

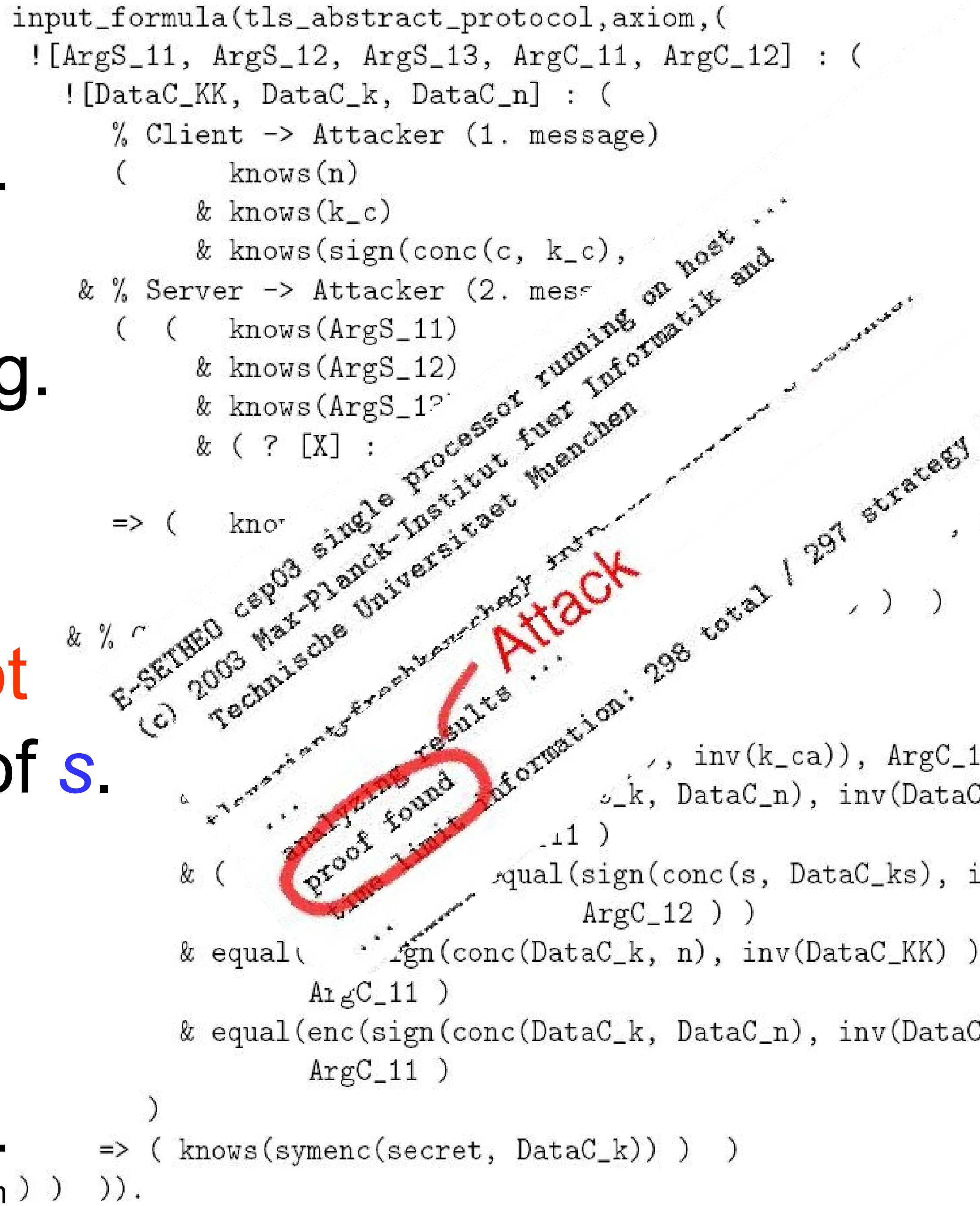
Analysis

Check whether can derive *knows(s)* e.g. using e-Setheo.

Surprise: Yes !

➔ Protocol does not preserve secrecy of *s*.

Why ? Use Prolog-based attack generator.



Security Analysis: Model or Code ?

Model:

+ earlier (less expensive to fix flaws)

+ more abstract → more efficient

- more abstract → may miss attacks

- programmers may introduce security flaws

- even code generators, if not formally verified

Code:

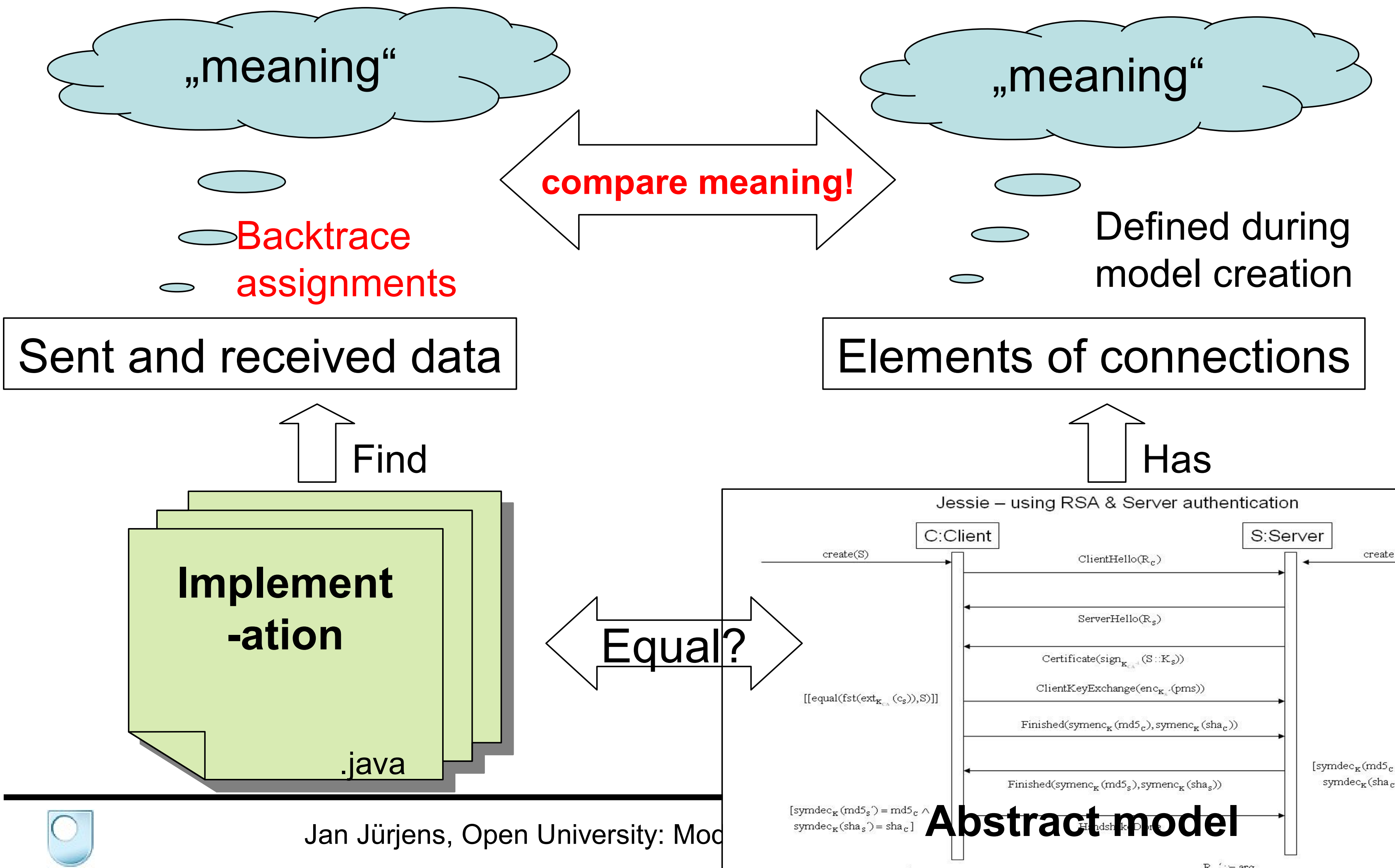
+ „the real thing“ (which is executed)

→ Do both !

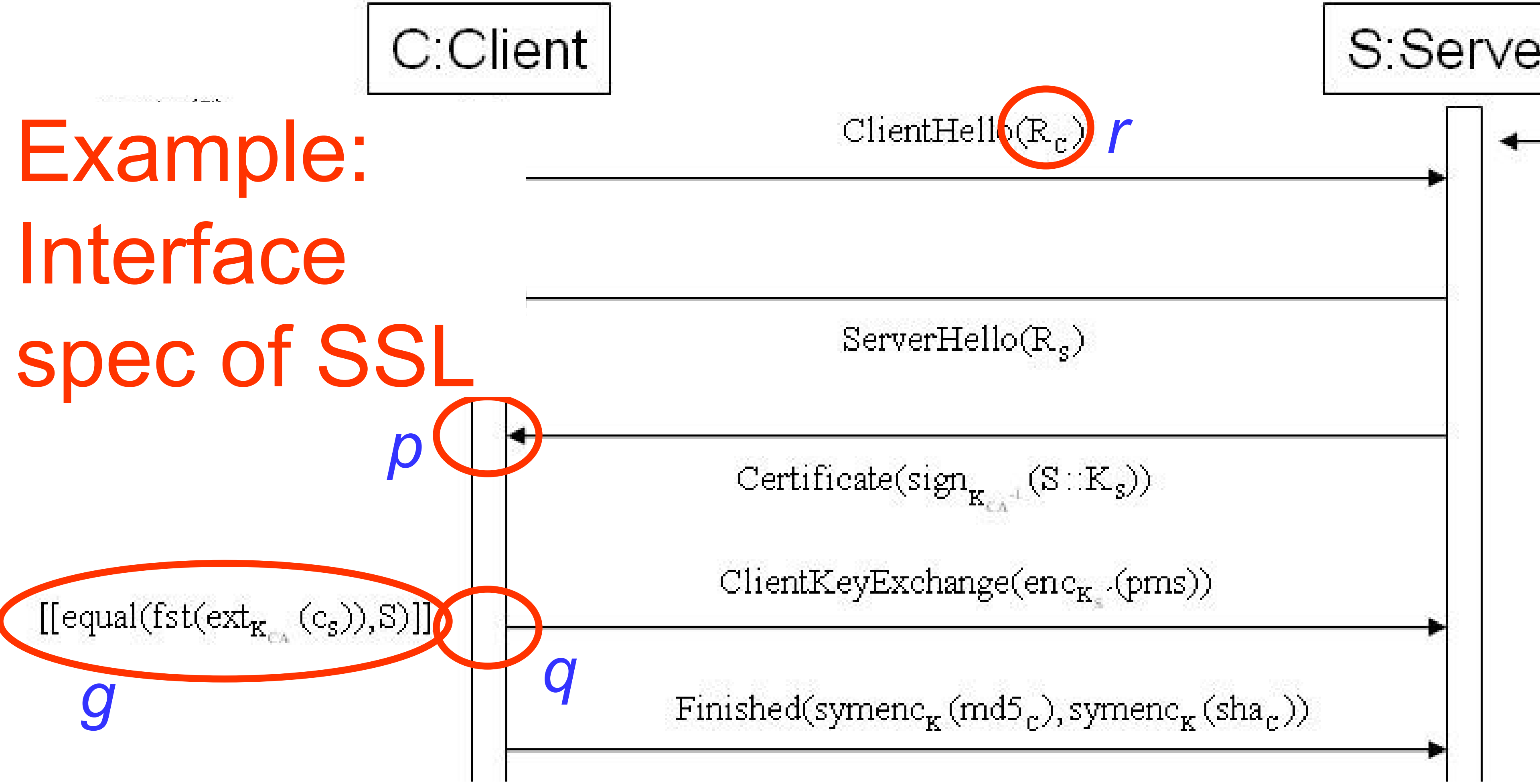
Surprise: Essentially no existing work (eg for crypto prots) !

Model vs. Implementation

[with David Kirscheneder]



Example: Interface spec of SSL

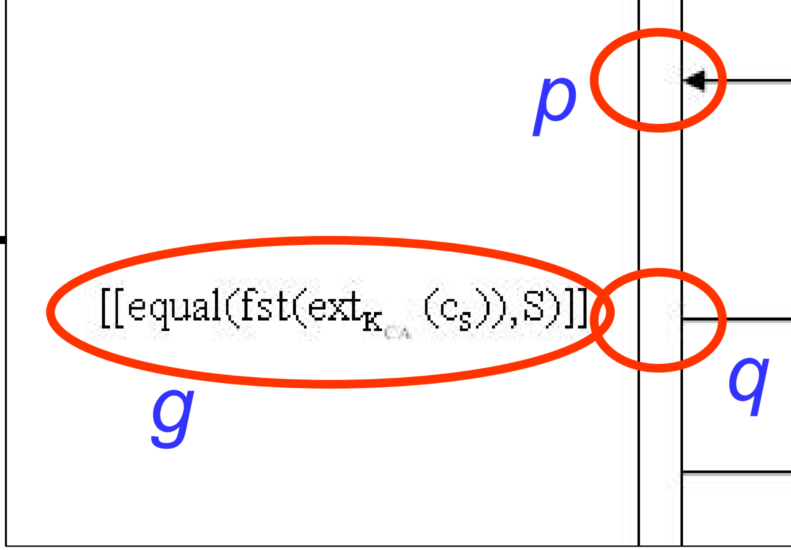


I) Identify program points:

value (*r*), receive (*p*), guard (*g*), send (*q*)

II) Check guards enforced

Checking Guards

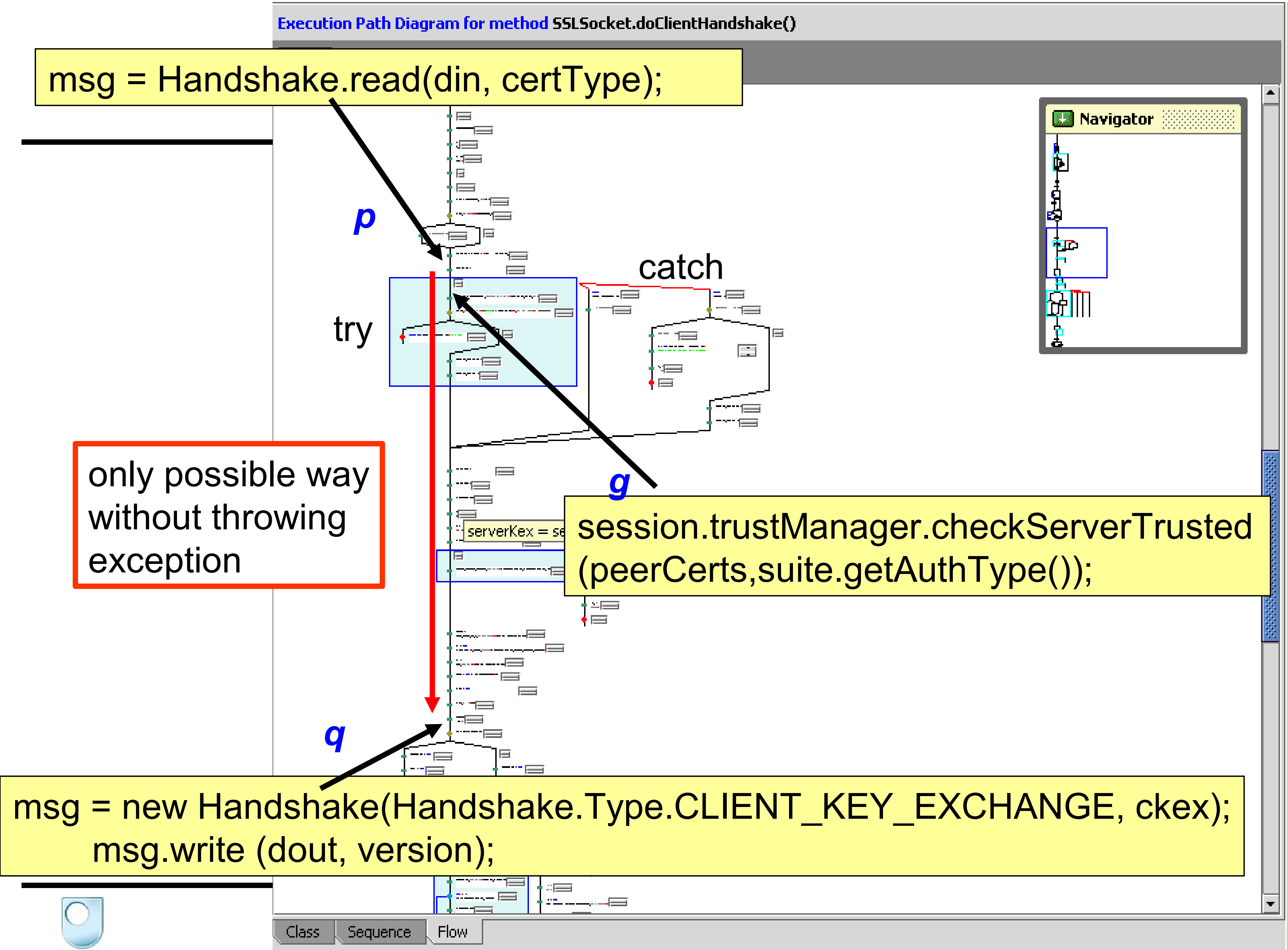


Guard *g* enforced by code?

b) Generate runtime check for *g* at *q* from diagram: simple + effective, but performance penalty.

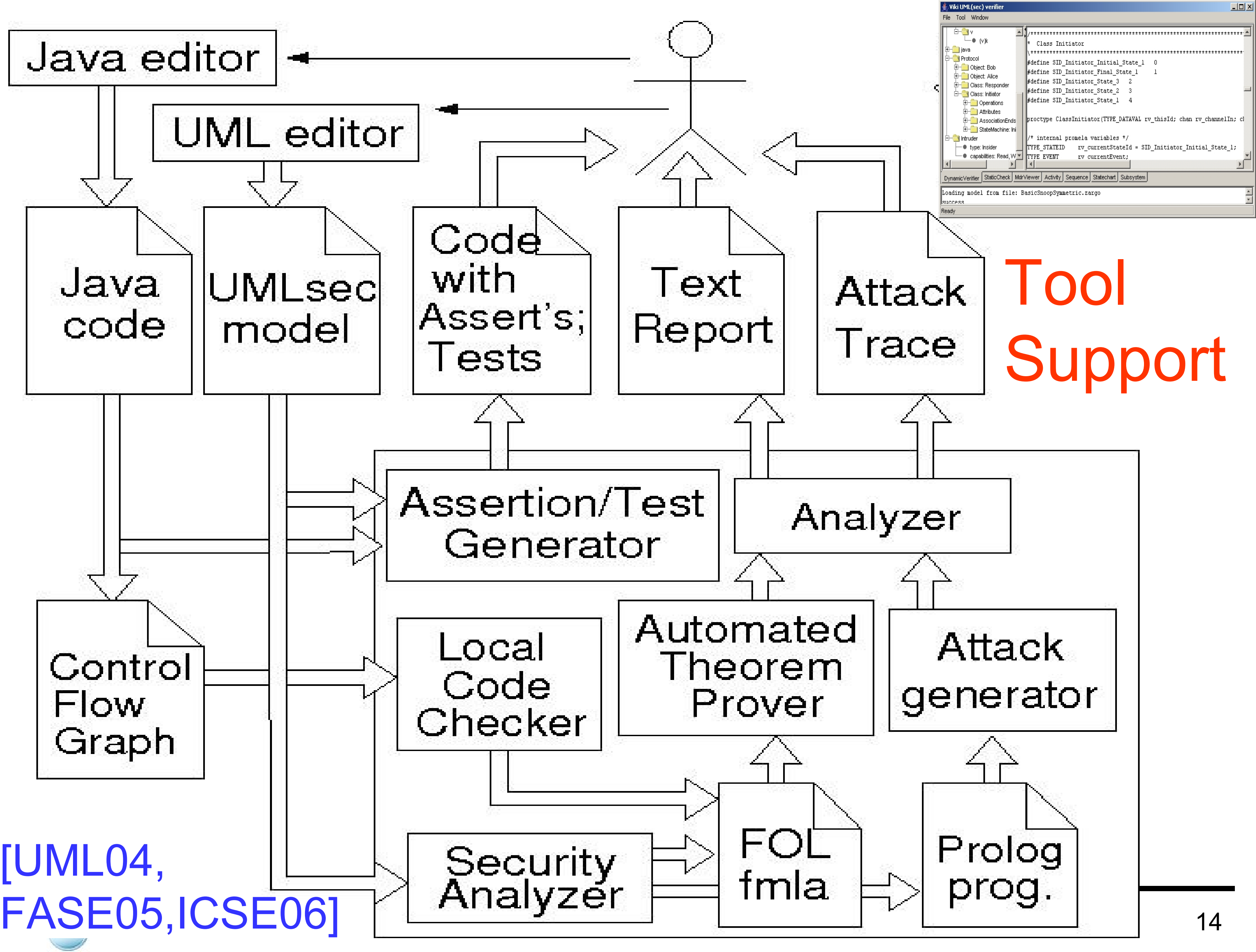
c) Testing against checks (symbolic crypto for inequalities). [ICFEM02]

d) Automated formal local verification: conditionals between *p* and *q* logically imply *g* (using ATP for FOL). [ASE06]



Execution Path Diagram for method SSLSocket.doClientHandshake()
msg = Handshake.read(din, certType);
p
try
catch
Navigator
only possible way without throwing exception
g
serverKex = se
session.trustManager.checkServerTrusted (peerCerts,suite.getAuthType());
q
msg = new Handshake(Handshake.Type.CLIENT_KEY_EXCHANGE, ckex);
msg.write (dout, version);
Class
Sequence
Flow

Tool Support



[UML04, FASE05,ICSE06]

Some Applications

[ACSAC05,ICSE07]

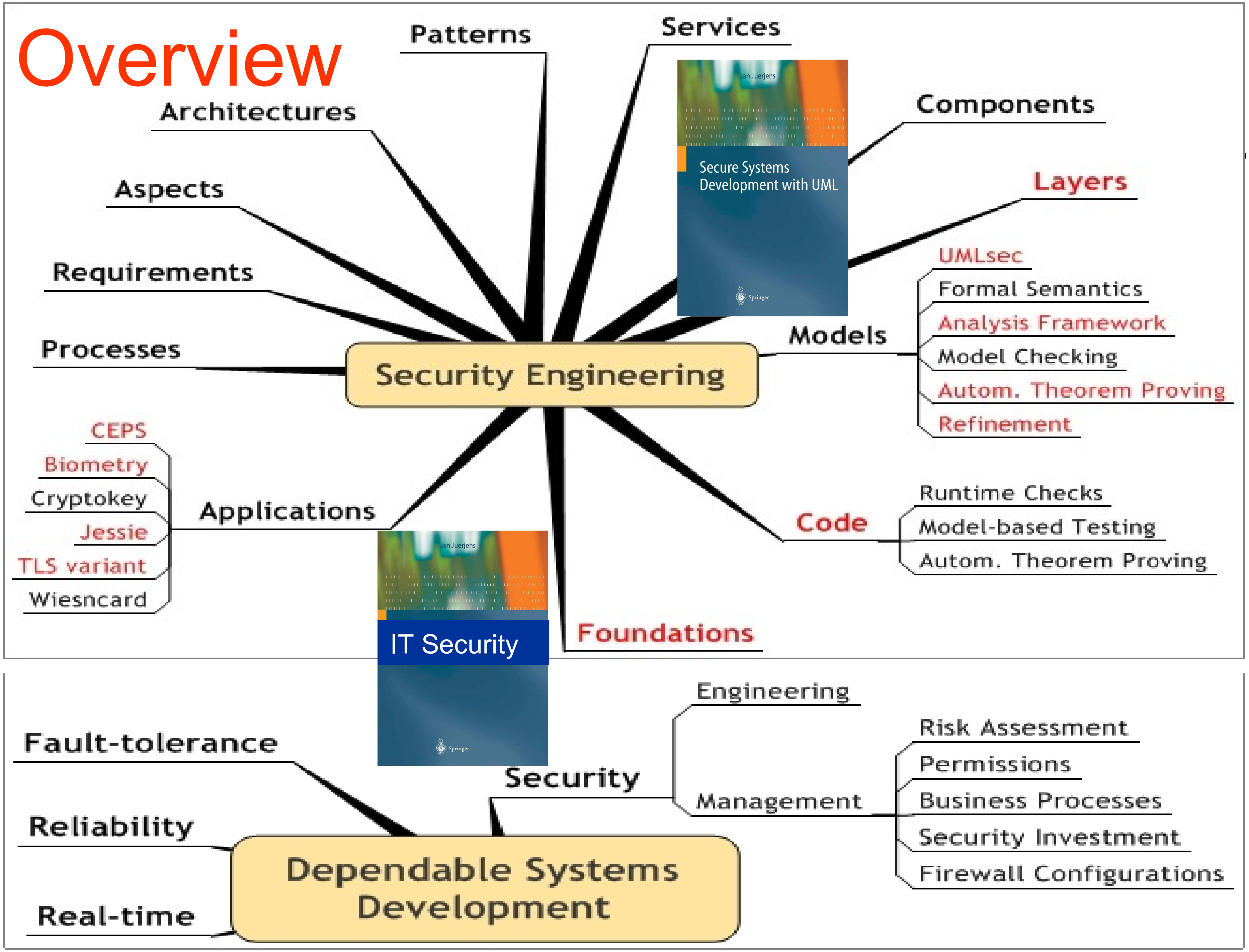
Analyzed designs / implementations / configurations for

- biometry, smart-card or RFID based identification
- authentication (crypto protocols)
- authorization (user permissions, e.g. SAP systems)

Analyzed security policies, e.g. for privacy regulations.

Bundesministerium für Bildung und Forschung - Bundesministerium für Wirtschaft und Technologie

Overview


Patterns
Services
Architectures
Components
Aspects
Layers
Requirements
UMLsec
Formal Semantics
Analysis Framework
Processes
Models
Model Checking
Security Engineering
Autom. Theorem Proving
Refinement
CEPS
Biometry
Cryptokey
Jessie
TLS variant
Wiesncard
Applications
Runtime Checks
Code
Model-based Testing
Autom. Theorem Proving
Foundations
Jan Juerjens
Secure Systems Development with UML
Springer
IT Security
Engineering
Risk Assessment
Fault-tolerance
Permissions
Security
Management
Business Processes
Security Investment
Reliability
Firewall Configurations
Dependable Systems Development
Real-time