Modellbasierte Compliance und Sicherheit für Cloud-basierte Services und Prozesse

Jan Jürjens

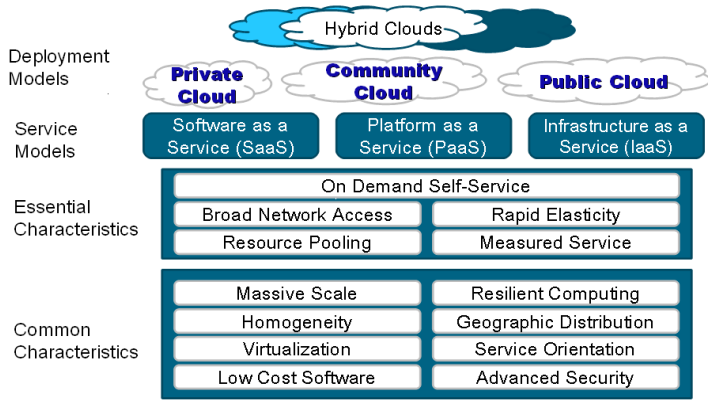
TU Dortmund und Fraunhofer ISST

Clouds

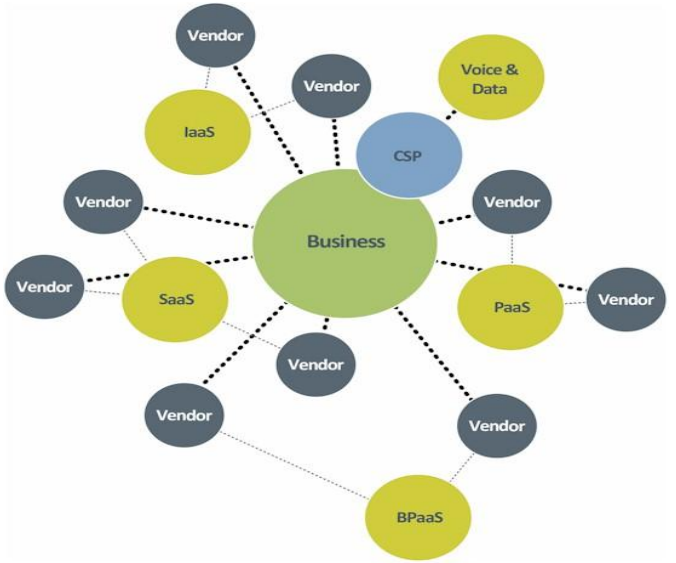
Es gibt ein großes Angebot für Cloud Services, die kostengünstig, einfach, flexibel und skalierbar erworben werden können (besonders interessant für KMUs).

Das große Angebot erzeugt eine erhebliche Komplexität:

- Einsatz redundanter Services.
- Services verschiedener Anbieter sind möglicherweise nicht interoperabel.
- Die Einhaltung von Compliance-Anforderungen muss für jeden SaaS-Anbieter jeweils separat überprüft werden.
- Administration, Support und Abrechnung der Services verschiedener Anbieter verursacht zusätzlichen Aufwand.



NIST Definition of Cloud Computing [NIST Special Publication 800-145, 2011].



[http://www.smallcloudbuilder.com]

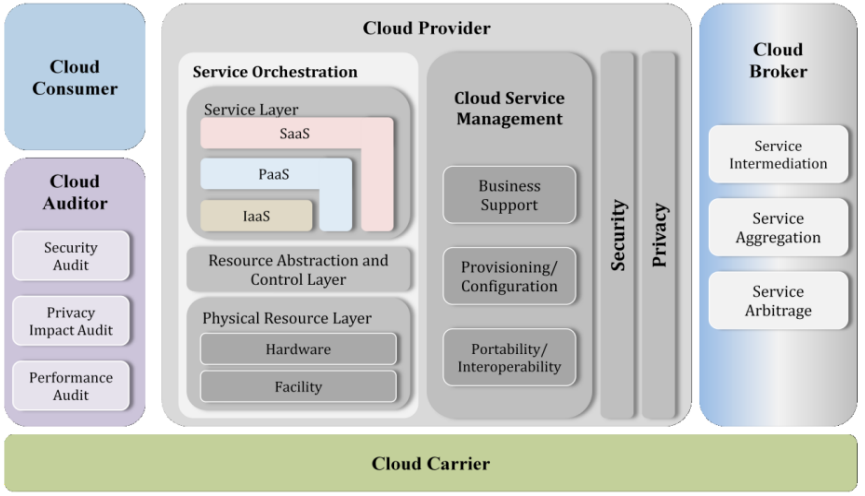
Cloud-Service-Marktplätze

Cloud-Service-Marktplätze (“Cloud Broker”) bieten Hilfe bei der Bewältigung dieser Komplexität: Unterstützung für Administration, Support, Monitoring und Abrechnung der erworbenen Services.

Verringern Risiko eines Vendor-Lock-In, da leicht zu anderen Anbietern am Marktplatz gewechselt werden kann.

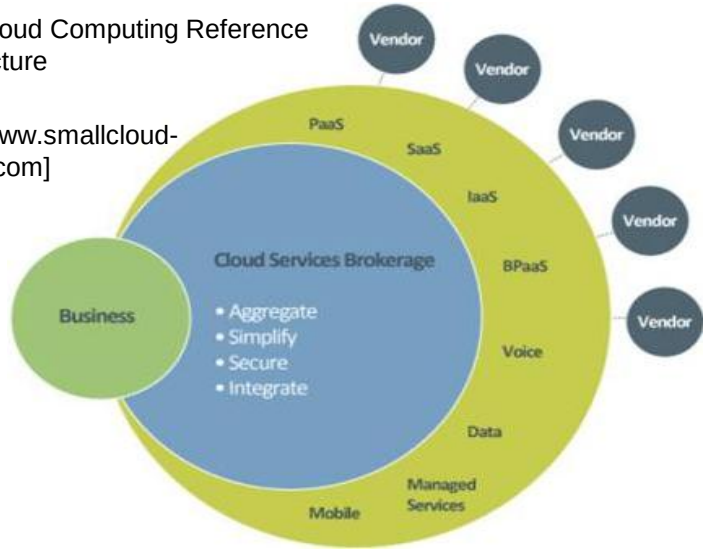
Gleichzeitig sind Cloud-Service-Marktplätze eine attraktive neue Vertriebsoption gerade für kleinere Software-Hersteller (analog zu Apple’s App-Store in einem nicht-cloud-basierten Kontext).

Beispiele: AWS Marketplace (Amazon Web Services), Microsoft Azure Marketplace, VERECLOUD’s cloudwrangler.



NIST Cloud Computing Reference Architecture

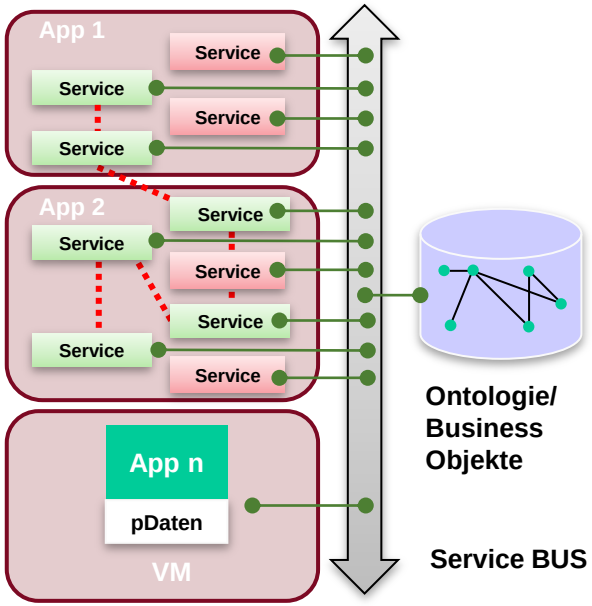
[http://www.smallcloud-builder.com]



Logistics Mall: Cloud Computing in der Logistik

Besonders vielversprechend: Bereitsstellung **domänenspezifischer Cloud-Marktplätze**, die sich auf die vertikale Integration von Services in einer Domäne spezialisieren. Beispiel: Logistics Mall von Fraunhofer IML und Fraunhofer ISST.

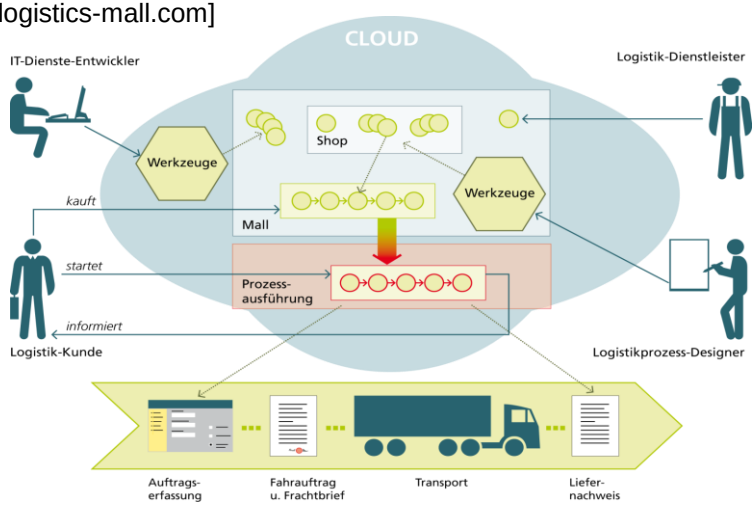
Anwendungen werden aus interoperablen Servicebausteinen zusammengestellt und sind über die Business-Ontologie standardisiert.

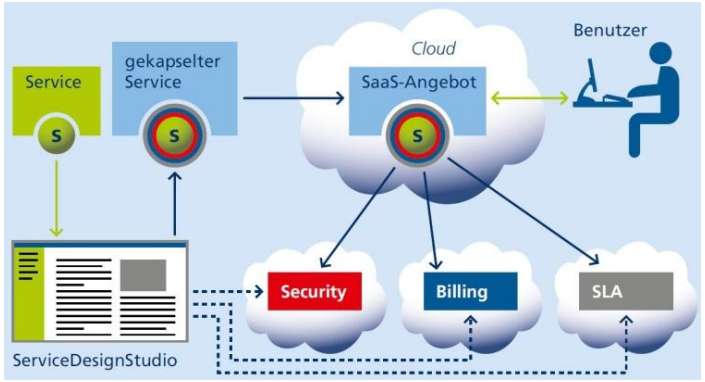


[http://logistics-mall.com]

EffizienzCluster LogistikRuhr

logistics mall





Clouds: Herausforderungen

Trotz der Vorteile des Einsatzes von Clouds zögern viele Unternehmen bislang, Clouds zu nutzen.

Zwei hauptsächliche Gründe sind Bedenken hinsichtlich **Sicherheit** sowie die Einhaltung regulatorischer Anforderungen („**Compliance**“).

Zwei Beispiel-Umfragen:

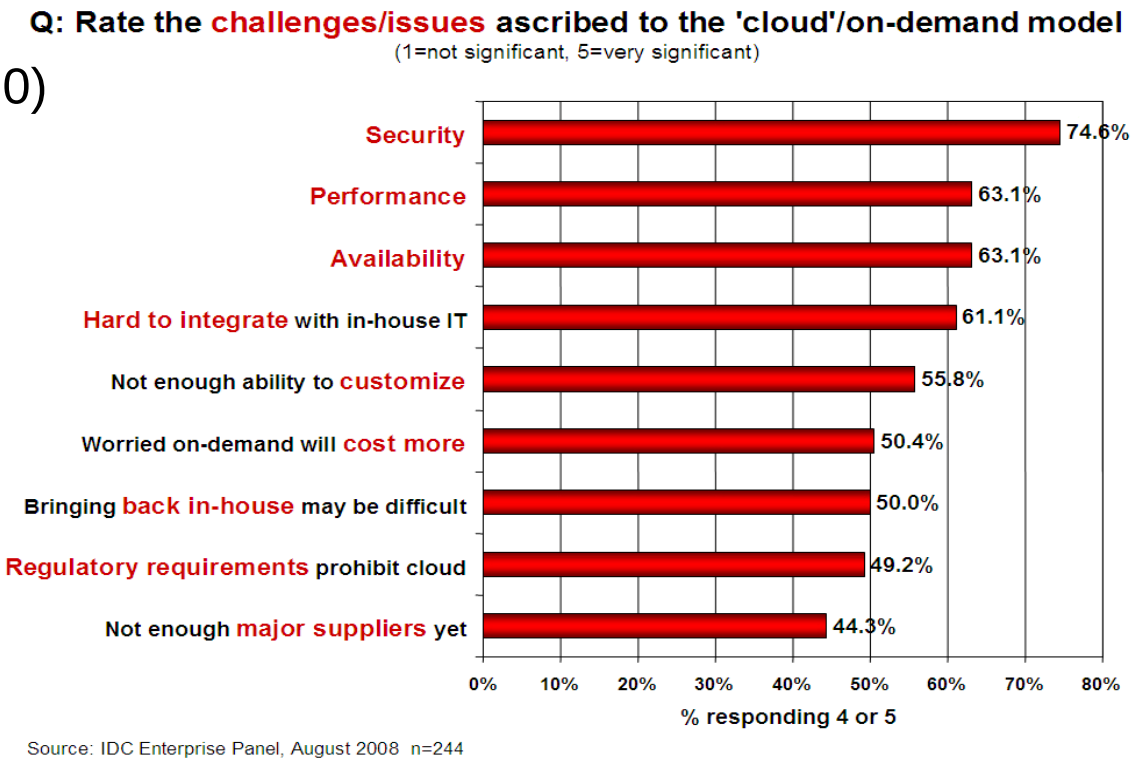
Umfrage von Novell (Okt. 2010)

90%: “concerned about cloud security”

81%: “worried about regulatory compliance”

76%: “private data more secure when stored on the premises”

50%: “security concerns primary barrier to cloud adoption”



Source: IDC Enterprise Panel, August 2008 n=244

Herausforderung: Cloud-Sicherheit

Vertraulichkeit	Verarbeitete Daten in Clouds sind unverschlüsselt (Verarbeitung verschlüsselter Daten bislang unpraktikabel).
Verfügbarkeit	Cloud nur über Internetverbindung erreichbar. Fehler bei Zuteilung und Management von Cloud-Ressourcen. Angriffe auf Verfügbarkeit (DOS). Risiko Vendor-Lock-In.
Integrität	Fehler/Angriffe von Mitarbeitern des Providers oder Angriffe von anderen Kunden.
Authentizität	Angriffe auf Authentizität von Cloud gegenüber Nutzer und umgekehrt (z.B. Missbrauch der Verwaltungsplattform).
Nicht-Abstreitbarkeit	Revisionsfähigkeit der Transaktionen gewährleistet ?
Datenschutz	Einhaltung gesetzlicher Regelungen (Standort beachten). Erstellung von Benutzerprofilen ?

[BSI: IT-Grundschutz und Cloud Computing, 2009]

Herausforderung: Cloud-Compliance

Unternehmen müssen **Konformanz** mit steigender Zahl **Regulierungen** demonstrieren.

Ziel ist zum Einen Mindestanforderungen an Risikomanagement (z.B. Versicherungen: Solvency-II; Banken: Basel III; branchenunabhängig: KontraG; US: Sarbanes-Oxley).

Weiteres Ziel: Schutz personenbezogener Daten durch Bundesdatenschutzgesetz.

Manueller Nachweis der Einhaltung aufwendig und kostenintensiv.

Komplexe Compliance-Beziehungen besonders im Bereich von **Cloud-Marktplätzen**:

- Service-Anbieter ⇔ Cloud-Marktplatz
- Cloud-Kunde ⇔ Cloud-Marktplatz
- Cloud-Kunde ⇔ Service-Anbieter

(z.B. Einhaltung von Regularien bzgl. Datenschutz, Risikomanagement; Einhaltung von Service Level Agreements etc.).

Die Überprüfung der Einhaltung der gesetzlichen Regularien bei der Durchführung der eigenen Geschäftsprozesse (auf der Prozess- und der IT-Ebene) ist auch beim Einsatz von Cloud-Services weiter **Verantwortung des Kunden**.

Der **Cloud-Marktplatz** muss die eigene Einhaltung der Regularien gegenüber dem Kunden belegen (insbes. die Angemessenheit des eigenen Sicherheitsmanagements). Weiter kann er die Überwachung der Regularien auf Seiten der Service-Anbieter als Added-Value anbieten („**Compliance-as-a-Service**“).

Modellbasierte Compliance und Sicherheit für Cloud-basierte Services und Prozesse

- Einführung: Herausforderung Compliance und Sicherheit in Cloud-Marktplätzen 
- Modellierung und Analyse von Software-basierten Services auf Compliance-Anforderungen
  - Modell-basierte Spezifikation (Schritt 1)
  - Von Modell zur Implementierung (Schritt 2)
  - Konformanz der Implementierung zu Modell (Schritt 3)
- Anwendungen

Lösungsansatz:
Modell-basiertes Compliance-Management

Ziele:

- Bessere Überprüfbarkeit und Nachvollziehbarkeit des Nachweises der Einhaltung von Compliance-Anforderungen.
- Kostenersparnis durch Werkzeugunterstützung und Integration mit vorhandenen Aktivitäten im Bereich Qualitätssicherung und Risikomanagement.

Idee:

- Entwicklung automatischer Werkzeuge, die Management von Compliance-Anforderungen auf Basis von vorhandenen Artefakten unterstützen.
- Insbesondere automatisierte Compliance- und Risiko-Analysen auf Basis von Textdokumenten, Schnittstellen-Spezifikationen, Geschäftsprozess-Modellen, Log-Daten, Quellcode und anderen Datenquellen.
- Insbesondere auch Anwendung auf den Einsatz von Cloud-Computing (http://secureclouds.de).
- Zwei Ebenen:
  - Geschäftsprozesse
  - Software / Services

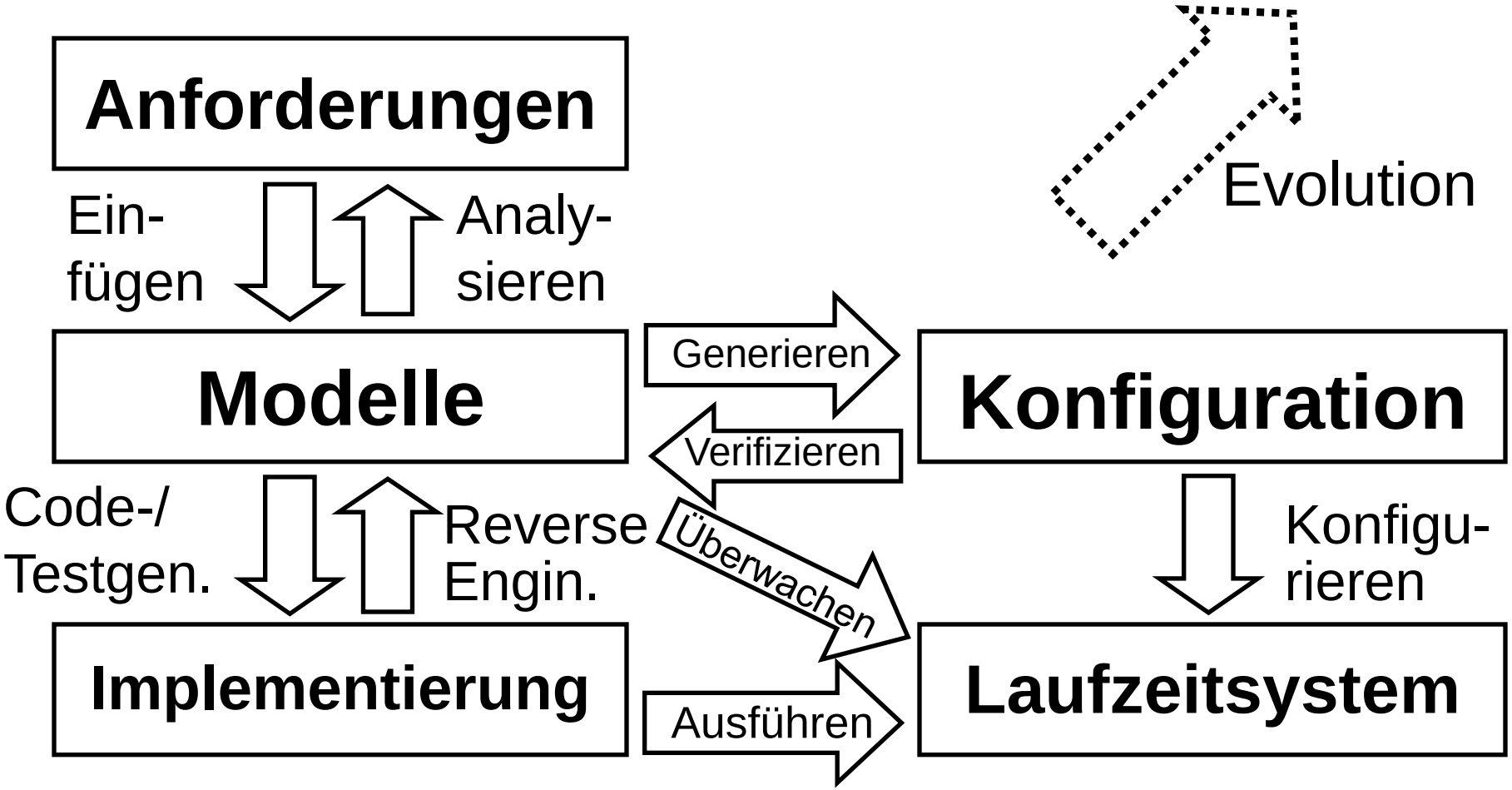
Compliance-Report

Compliant: NEIN
Verstöße:
- MaRISK VA 7.2: Einhaltung von BSI G3.1 nicht erfüllt

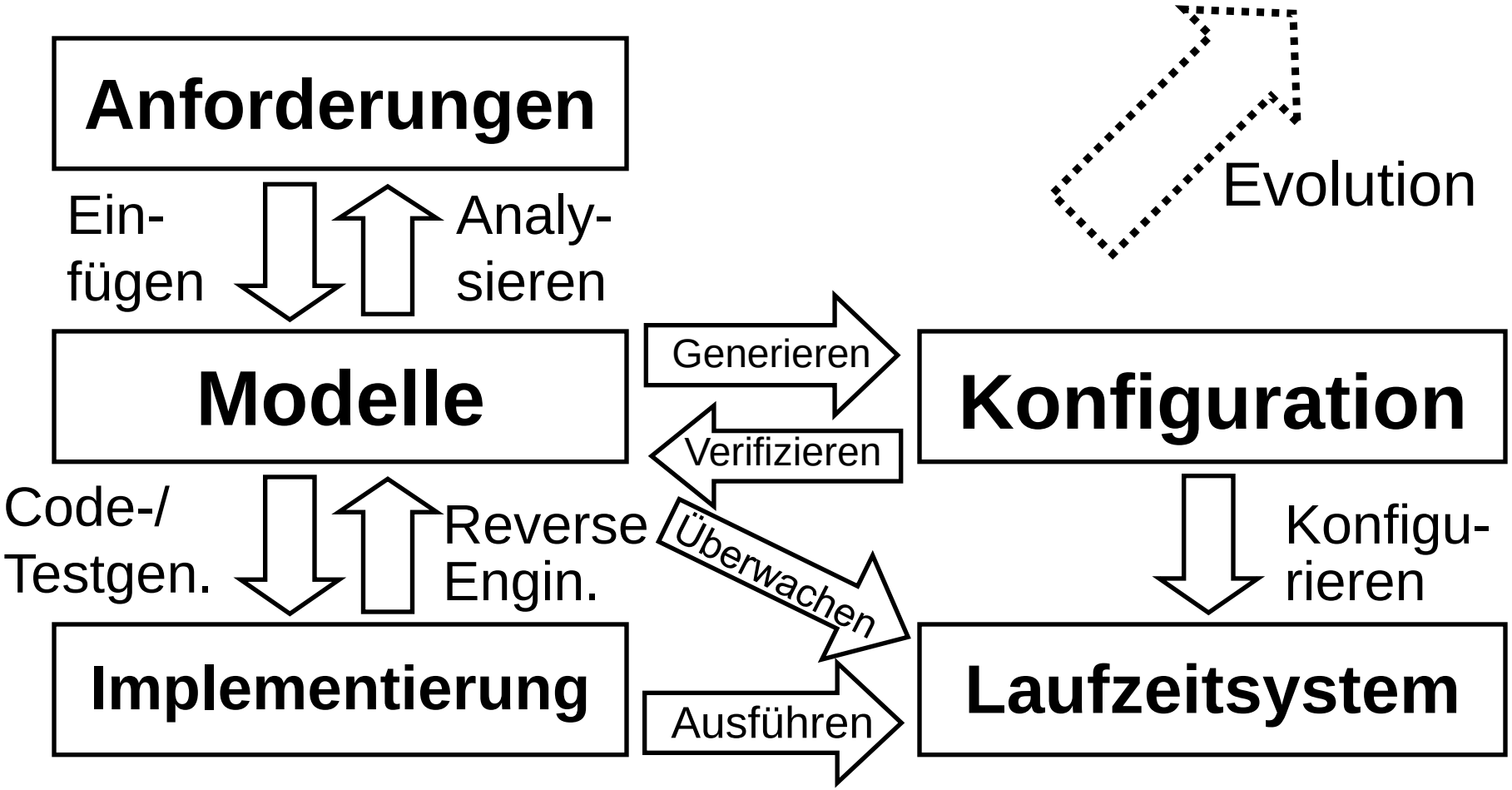
Maßnahmen:
- BSI Maßnahmen-katalog M 2.62

[S. Wenzel, C. Wessel, T. Humberg, J. Jürjens, Securing Processes for Outsourcing into the Cloud, 2nd Int. Conf. on Cloud Computing and Services Science (Closer 2012)]

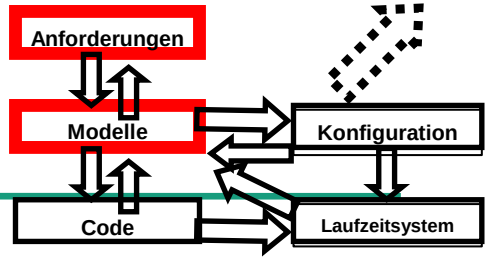
Modell-basiertes Compliance-Management



Modell-basiertes Compliance-Management



Compliance-Modellierung mit UMLsec



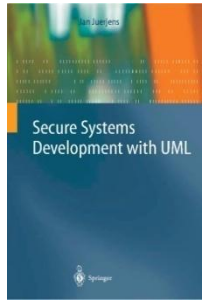
Ziel:

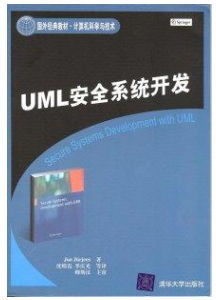
- Dokumentation und automatische Analyse von compliance-relevanten Informationen (z.B. Sicherheits-Eigenschaften und Compliance-Anforderungen) als Teil der Systemspezifikation.

Idee:

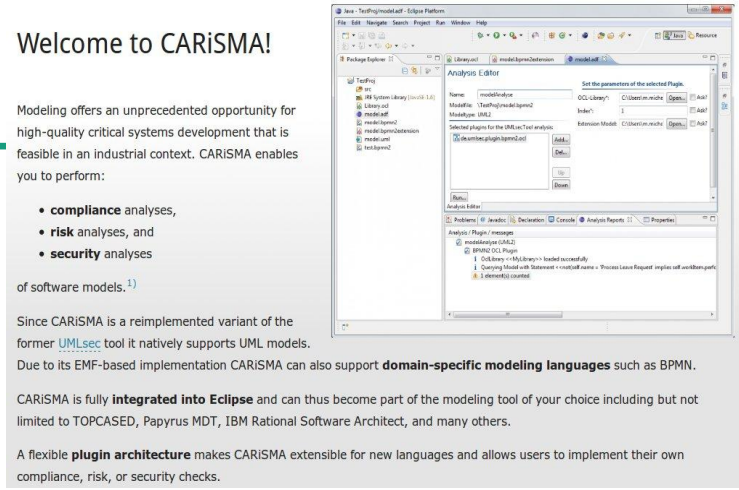
- UML für System-Modellierung.
- Sicherheitsinformationen als Markierungen (Stereotypen) einfügen, mithilfe der UML-Erweiterung UMLsec.
- Automatische Verifikation der Modelle gegen die Sicherheitsanforderungen auf Basis von formaler Semantik.

[Jan Jürjens: Secure systems development with UML. Springer 2005. Chines. Übers. 2009]

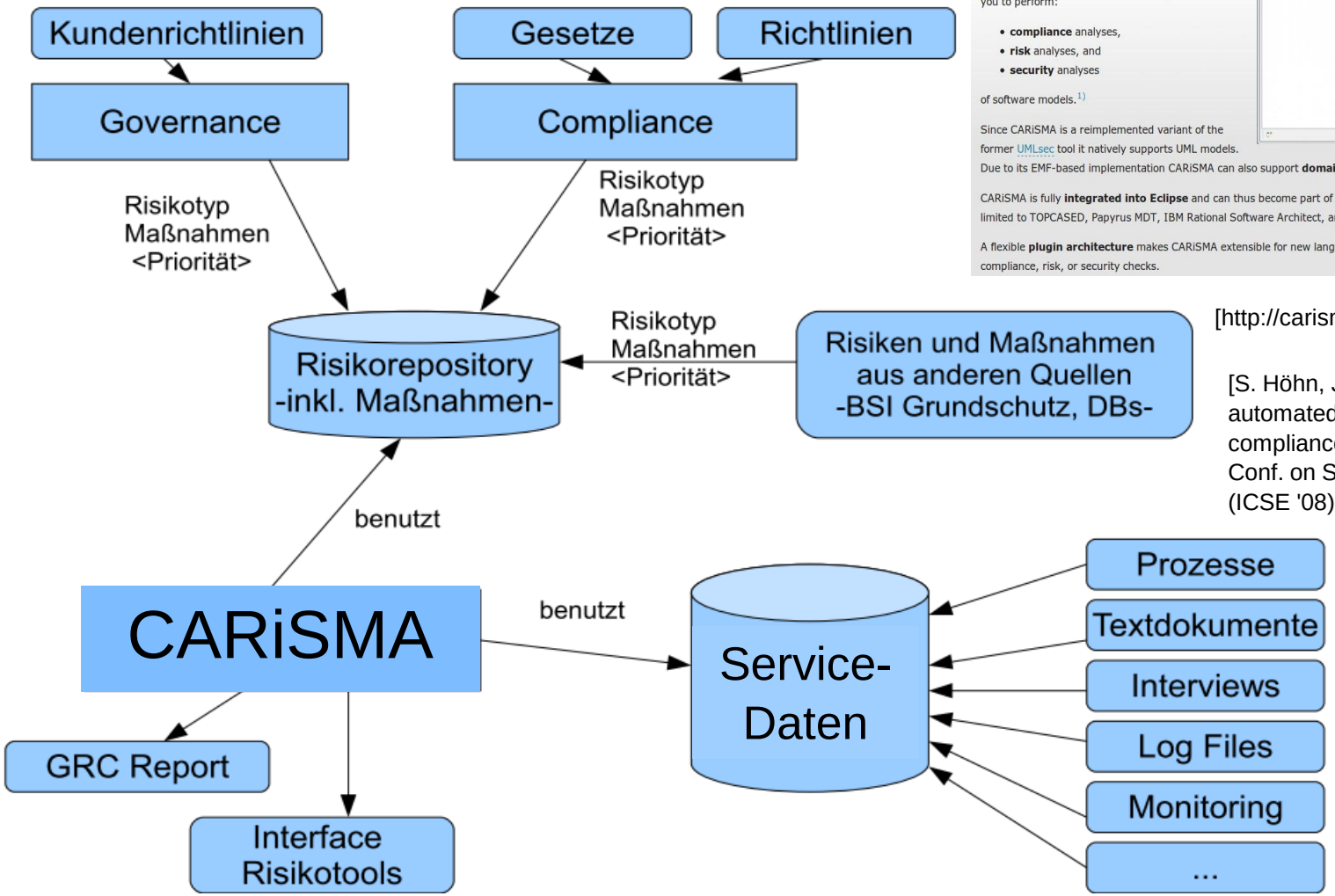




Werkzeugunterstützung: Workflow

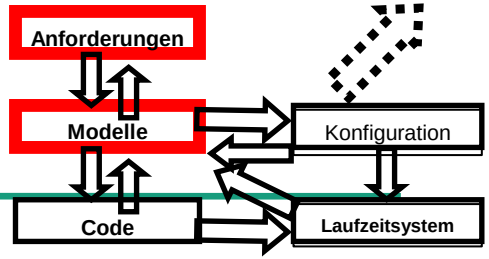


[http://carisma.umlsec.de]



[S. Höhn, J. Jürjens: Rubacon: automated support for model-based compliance engineering. 30th Int. Conf. on Software engineering (ICSE '08). ACM]

Modellbasierte Compliance und Sicherheit für Cloud-basierte Services und Prozesse



- Einführung: Herausforderung Compliance und Sicherheit in Cloud-Marktplätzen
- Modellierung und Analyse von Software-basierten Services auf Compliance-Anforderungen
  - Modell-basierte Verifikation (Schritt 1)
  - Von Modell zur Implementierung (Schritt 2)
  - Konformanz der Implementierung zu Modell (Schritt 3)
- Anwendungen

Beispiel-Anwendung: Mobile Services bei O_2

UMLsec-basierte Sicherheitsanalyse der Regulierungen für den Einsatz mobiler Endgeräte bei O_2 (Germany)

62 Sicherheitsanforderungen aus Security Policy extrahiert.

21 Geschäftsprozess-relevante Anforderungen in 8 Aktivitätsdiagrammen modelliert mithilfe der UMLsec-Stereotypen <<fair exchange>> and <<provable>>

10 Datensicherheits-Anforderungen (Vertraulichkeit, Integrität) in Deployment-Diagramm modelliert.

3 Anforderungen bzgl. Rollenbasierter Zugangskontrolle (RBAC) modelliert

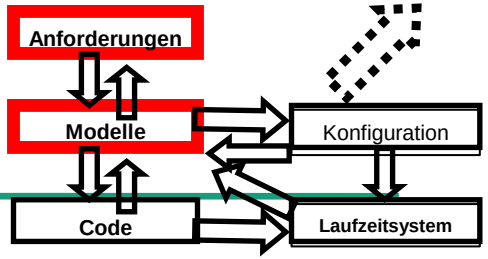
15 Anforderungen bzgl. Sicherheit der Netzwerkdienste, und Einsatz von Firewalls und Antivirensoftware modelliert (mithilfe weiterer Erweiterung von UMLsec)

13 Anforderungen konnten nicht direkt in UMLsec modelliert werden

[J. Jürjens, J. Schreck, P. Bartmann. 2008. Model-based security analysis for mobile communications. 30th Int. Conf. on Software engineering (ICSE '08). ACM]

Nr.	Sicherheitsanforderungen	<<Secure Links>>, Secure Links with XML	<<Fair Exchange>>	<<Provable>>	SecrecyIntegrity	TPTP-Datei z. Analyse v. Netzwerk-architekturen
1.9	Authentifizierung des Benutzers (Mitarbeiters) gegenüber dem Endgerät durch Chipkarten			X		
1.10	Verschlüsselung der auf den mobilen Endgeräten befindlichen Daten	X				
1.14	Keine zum Fernzugang parallele Verbindungen in andere Netze - Vermeidung der Kopplung mit unsicheren Netzen durch Umgehung der Firewall					X
1.26	Starke Verschlüsselung der Verbindung zwischen Endgerät und Fernzugangs-Server	X				
1.37	Bei O_2 übliche Virenschutzprogramme auf den Endgeräten					X
1.38	Aktualisierung des Virenschutzes über Fernzugang					

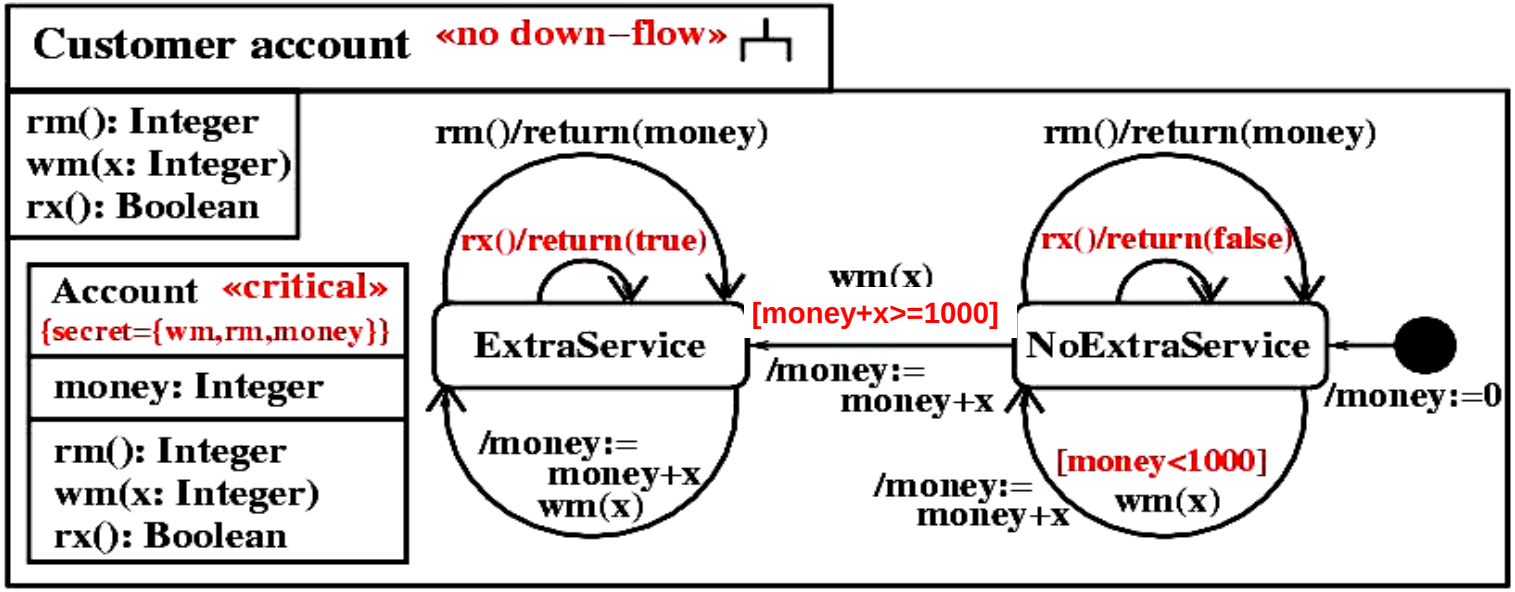
Schritt 1: Modell-basierte Compliance (Spezifikation)



Spezifikation der Compliance-Anforderungen erfolgt durch einfache Annotation des Modells.

Beispiel: Sicherer Informationsfluss in einem cloud-basierten e-commerce-Service.

Unsicher, weil der Rückgabe-Wert der öffentlichen Methode *rx()* von dem vertraulichen Attribut *money* abhängt.



Modell-basierte Compliance: Automatische Verifikation (1)

Herausforderung: Effiziente Verifikation für Modelle realistischer Größe.

1) **Formalisierung der Modellausführung.** Gegeben: Statechart-Transition *t=(source,msg,cond[msg],action[msg],target)* und erhaltene Nachricht *m*. Folgende Formel in Logik erster Stufe formalisiert Ausführung der Transition:

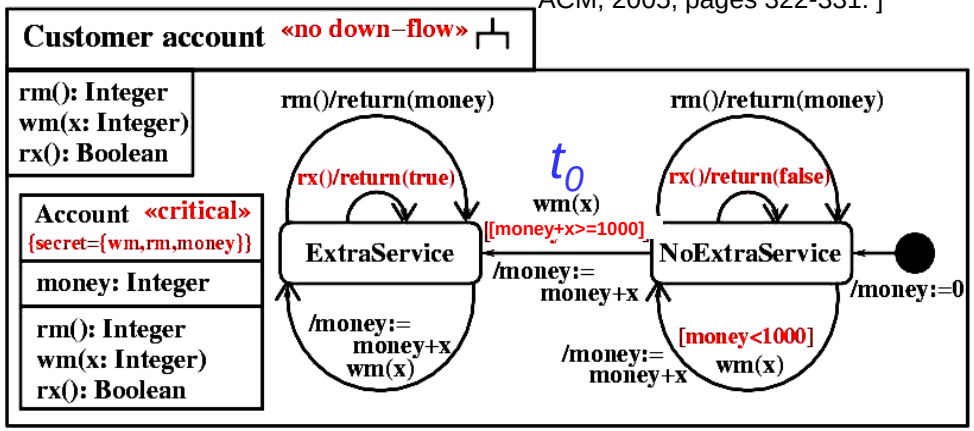
$$Exec(t,m) = [state_{current}=source \wedge m=msg \wedge cond[m]=true \Rightarrow action[m] \wedge state_{current;(t,m)}=target\ ].$$

(wobei $state_{current;(t,m)}$ der aktuelle Zustand und $state_{current;(t,m)}$ der nach Ausführung von *t* mit Nachricht *m* folgende Zustand).

[J. Jürjens, Sound Methods and Effective Tools for Model-based Security Engineering with UML, 27th Int. Conf. on Software Engineering, ACM, 2005, pages 322-331.]

Beispiel: Transition t_0:

$$Exec(t_0,m)=$$
$$[\ state_{current}=NoExtraService \wedge m=wm(x) \wedge money_{current}+x>=1000 \Rightarrow money_{current;(t_0,m)}=money_{current}+x \wedge state_{current;(t_0,m)}=ExtraService\ ].$$



Modell-basierte Compliance: Automatische Verifikation (2)

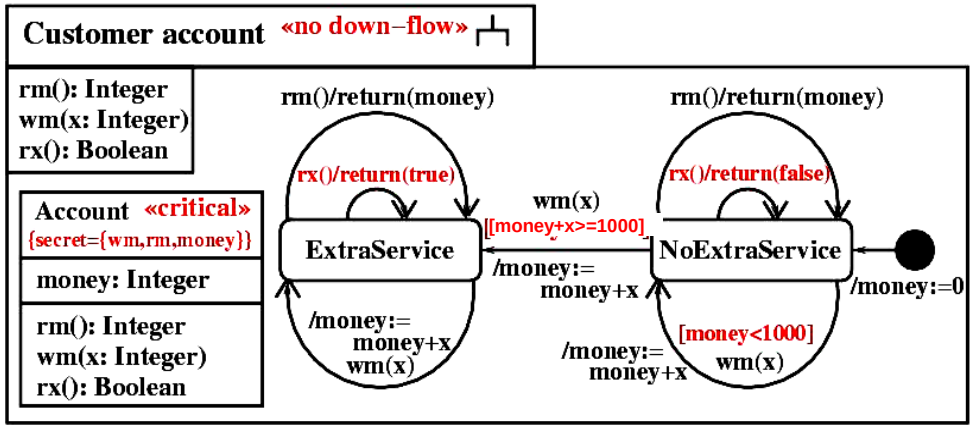
2) Formalisierung der Compliance-Anforderung „sicherer Informationsfluss“: Wenn zwei Systemzustände *t*, *t‘* sich nur in den Werten der vertraulichen Attributen unterscheiden, darf ihr öffentlich beobachtbares Verhalten sich nicht unterscheiden:

$t \approx_{pub} t' \Rightarrow t.t_1(inp) \approx_{pub} t'.t_1'(inp)$

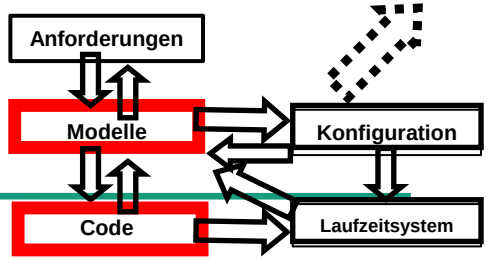
(wobei $t.t_1 \approx_{pub} t'.t_1'$ falls $Exec(t,t_1,inp) \Rightarrow att=wert$ genau dann wenn $Exec(t',t_1',inp) \Rightarrow att=wert$ für jedes öffentliches Attribut oder Rückgabewert *att*).

Beispiel: $wm(0) \approx_{pub} wm(1000)$, aber nicht:

$wm(0).rx() \approx_{pub} wm(1000).rx()$

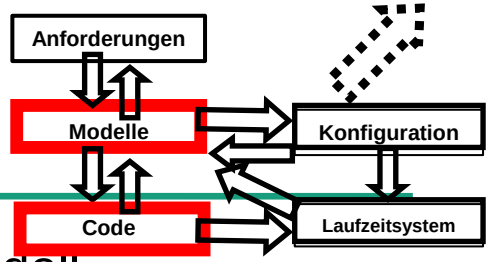


Modellbasierte Compliance und Sicherheit für Cloud-basierte Services und Prozesse



- Einführung: Herausforderung Compliance und Sicherheit in Cloud-Marktplätzen
- Modellierung und Analyse von Software-basierten Services auf Compliance-Anforderungen
  - Modell-basierte Verifikation (Schritt 1)
  - Von Modell zur Implementierung (Schritt 2)
  - Konformanz der Implementierung zu Modell (Schritt 3)
- Anwendungen

Schritt 2: Modell vs. Implementierung



Nach Spezifikation der Compliance-Anforderungen im Modell muss zunächst die Verbindung zur Implementierung hergestellt werden, um anschließend die Einhaltung der Compliance überwachen zu können.

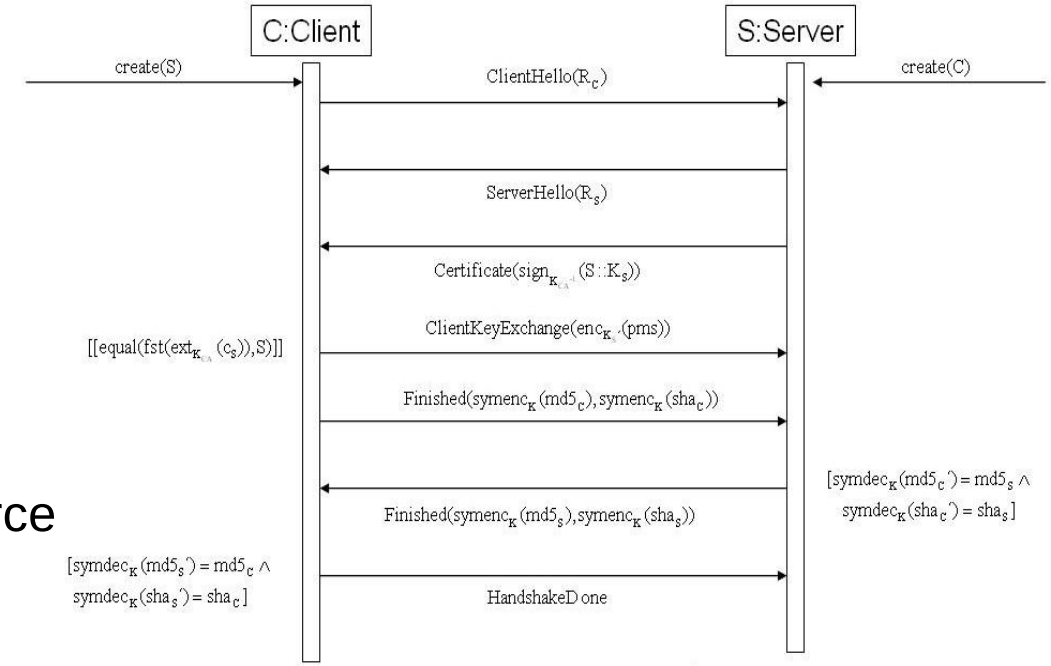
Beispielanwendung:

Cloud Security Alliance: “Security as a Service – Defined Categories of Service 2011”

Kategorie 8:
Verschlüsselungs-Services.

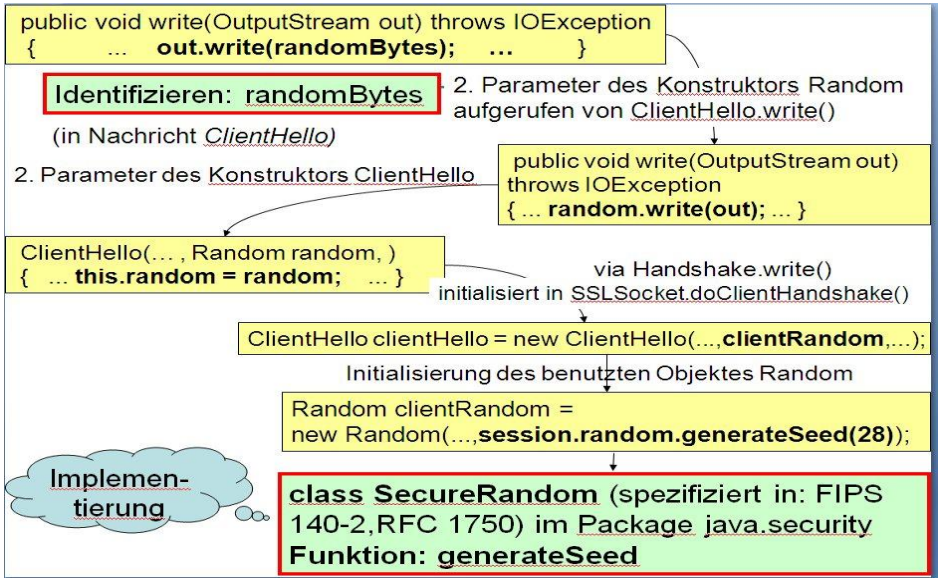
Verwendete Technologie:
Secure Sockets Layer (SSL).

Java-Implementierung: Java Secure Sockets Extension (JSSE). Alternative open-source Implementierung: Jessie.



Von Modell zu Implementierung: Nachrichten

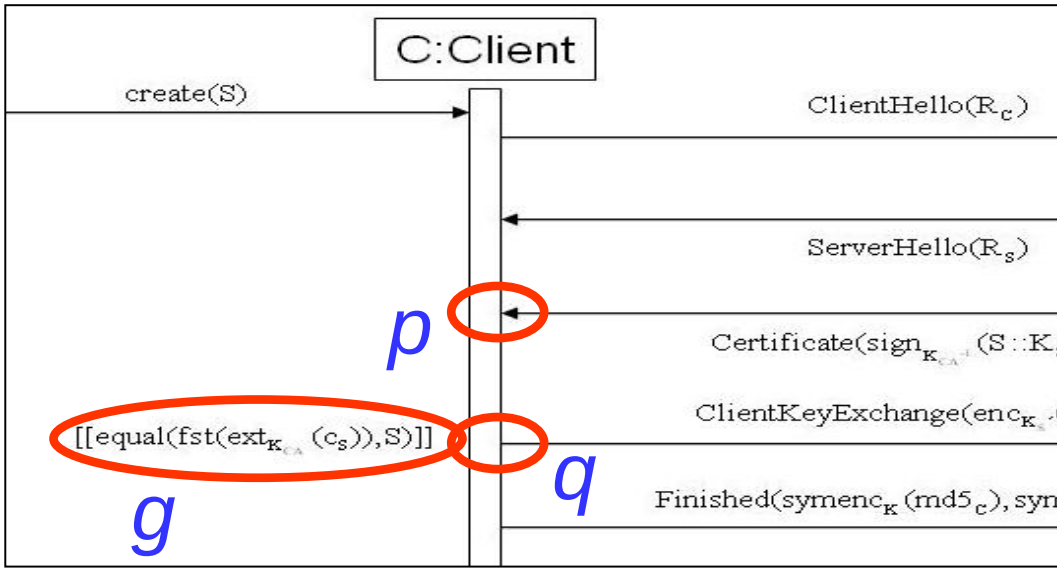
Im Modell verwendete symbolische Werte müssen auf die während Ausführung verwendeten konkreten Werte abgebildet werden (durch Datenflussanalyse des Source-Code falls vorhanden; sonst aufgrund der Dokumentation oder Analyse des Laufzeitverhaltens).



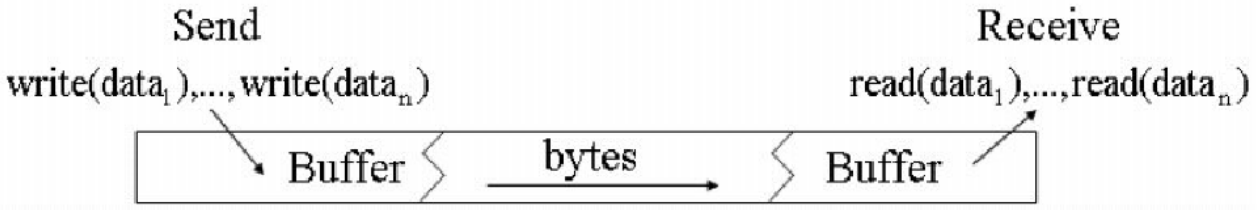
C	type.getValue()
Pver →	major
→	minor
	((gmtUnixTime >>> 24) & 0xFF)
	((gmtUnixTime >>> 16) & 0xFF)
	((gmtUnixTime >>> 8) & 0xFF)
	(gmtUnixTime & 0xFF)
r_c →	randomBytes
	sessionId.length
Sid →	sessionId

Von Modell zu Implementierung: Kommunikation

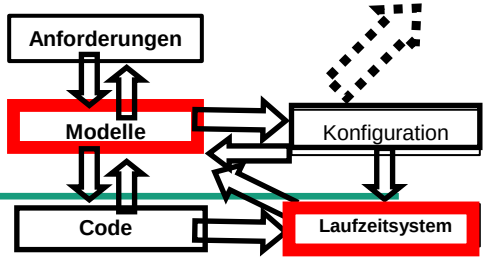
Relevant für Compliance-Verifikation von Services ist insbesondere die Schnittstellen-Spezifikation (Nachrichtenempfang (p), Bedingung (g), Antwortnachricht (q)).



Realisierung der Kommunikation erfolgt typischerweise über Eingabe-/Ausgabe-Puffer; können leicht in Implementierung identifiziert werden.



Modellbasierte Compliance und Sicherheit für Cloud-basierte Services und Prozesse



- Einführung: Herausforderung Compliance und Sicherheit in Cloud-Marktplätzen
- Modellierung und Analyse von Software-basierten Services auf Compliance-Anforderungen
  - Modell-basierte Verifikation (Schritt 1)
  - Von Modell zur Implementierung (Schritt 2)
  - Konformanz der Implementierung zu Modell (Schritt 3)
- Anwendungen

[J. Jürjens. Security Analysis of Crypto-based Java Programs using Automated Theorem Provers. 21st Int. Conf. on Automated Software Engineering (ASE 2006), IEEE/ACM, 2006, pp. 167-176]

Schritt 3: Konformanz der Implementierung zum Modell

Verbleibender Teil der Schnittstellen-Spezifikation: Bedingung (g).

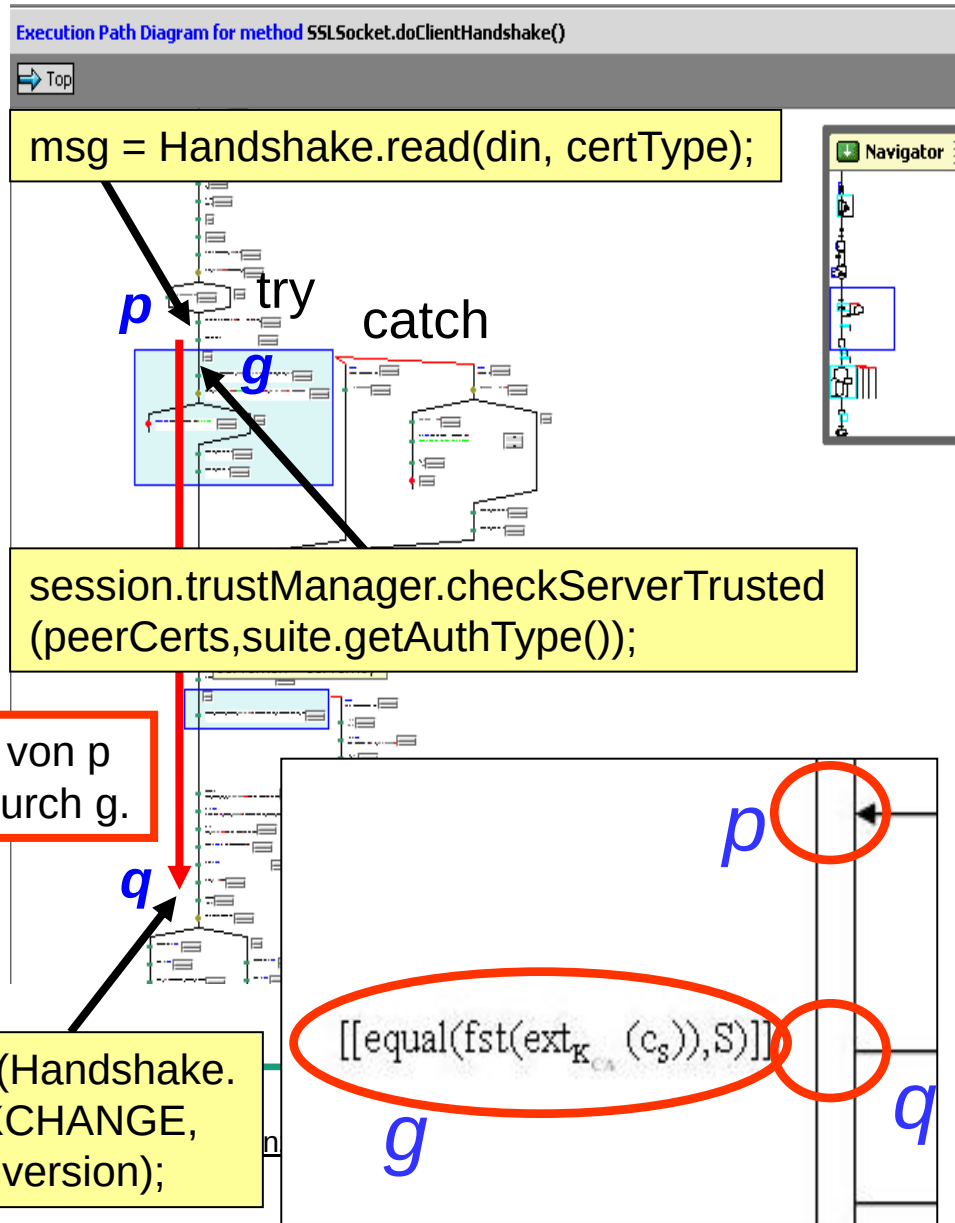
Falls Source-Code zugänglich: Statische Analyse auf Kontrollflussgraph.

Sonst: Laufzeit-Monitoring.

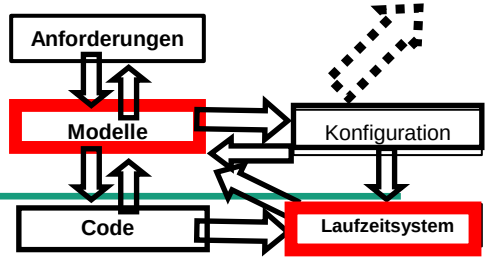
Möglicher Ansatz: Fred Schneider's Security Automata[1]. Betrachtet nur Safety-Eigenschaften (= Eigenschaften einzelner Systemläufe). Zu restriktiv für manche Compliance-Eigenschaften (z.B. Informationsfluss, s. früheres Beispiel).

Daher: neuer Ansatz auf Basis von Run-time-Verification.

[1] Fred B. Schneider: Enforceable security policies. ACM Trans. Inf. Syst. Secur. 3(1): 30-50 (2000)



Monitoring Cloud-Services mittels Runtime-Verification



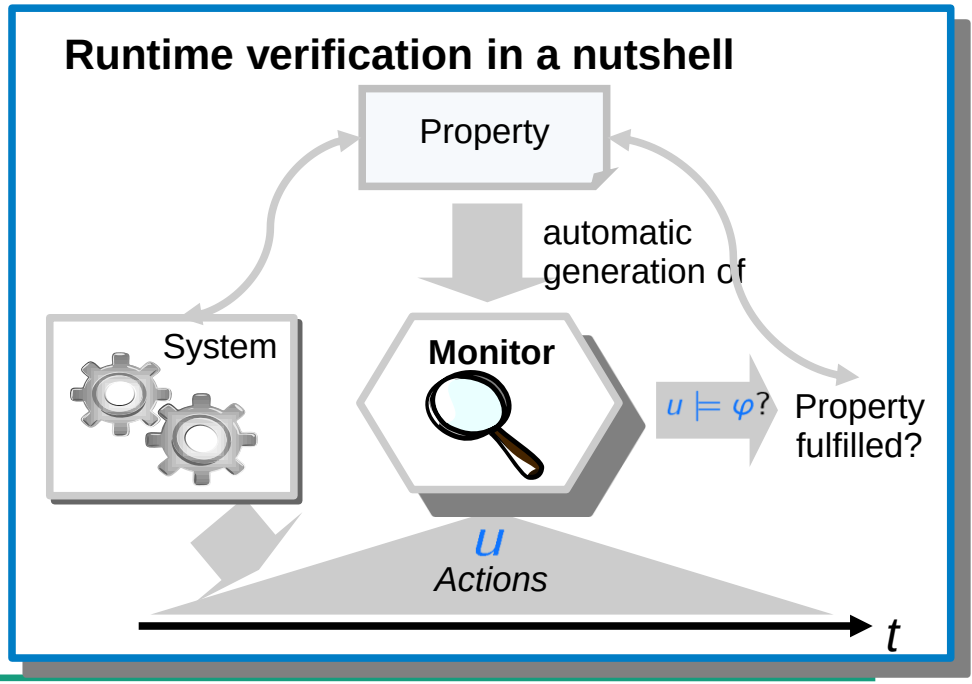
Runtime-Verification: Laufzeit-basierte Verifikation, abgeleitet aus Model-Checking und Testen (“Lazy model-checking”: nur die jeweils gegenwärtig ausgeführte Systemausführung wird verifiziert).

Compliance-Anforderung φ in Linear-Zeit-Logik LTL.

Kontinuierliche Überprüfung von φ über Folgen u von System-Ereignissen.

Automatische Monitor-Erzeugung: Inspiriert durch Übersetzung von LTL-Formel φ nach Büchi-Automat BA_φ sodass die erzeugten Sprachen übereinstimmen:

$$\varphi \rightarrow BA_\varphi \;\; s.t. \;\; L(BA_\varphi) = L(\varphi)$$



Zur Erinnerung: Syntax und Semantik für LTL

LTL-Syntax: $\varphi ::= true \mid p \mid \neg p \mid \varphi \; op \; \varphi \mid \varphi \mathbf{U} \varphi \mid \mathbf{X}\varphi \quad (p \in AP)$

Semantik: Sei *w* eine Folge von Ereignissen, *i* der Folgenzähler und *w(i)* das *i*-te Ereignis. Dann definieren wir:

$$
\begin{array}{lcl}
w, i \models true & & \\
w, i \models \neg\varphi & \Leftrightarrow & w, i \not\models \varphi \\
w, i \models p \in AP & \Leftrightarrow & p \in w(i) \\
w, i \models \varphi_1 \vee \varphi_2 & \Leftrightarrow & w, i \models \varphi_1 \vee w, i \models \varphi_2 \\
w, i \models \varphi_1 \boldsymbol{U} \varphi_2 & \Leftrightarrow & \exists k \geq i.\; w, k \models \varphi_2 \wedge \\
 & & \forall i \leq l < k.\; w, l \models \varphi_1 \\
w, i \models \boldsymbol{X}\varphi & \Leftrightarrow & w, i+1 \models \varphi
\end{array}
$$

Wir schreiben $w \models \varphi$ (d.h. die Folge w erfüllt die LTL-Formel φ) genau dann, wenn $w, 0 \models \varphi$.

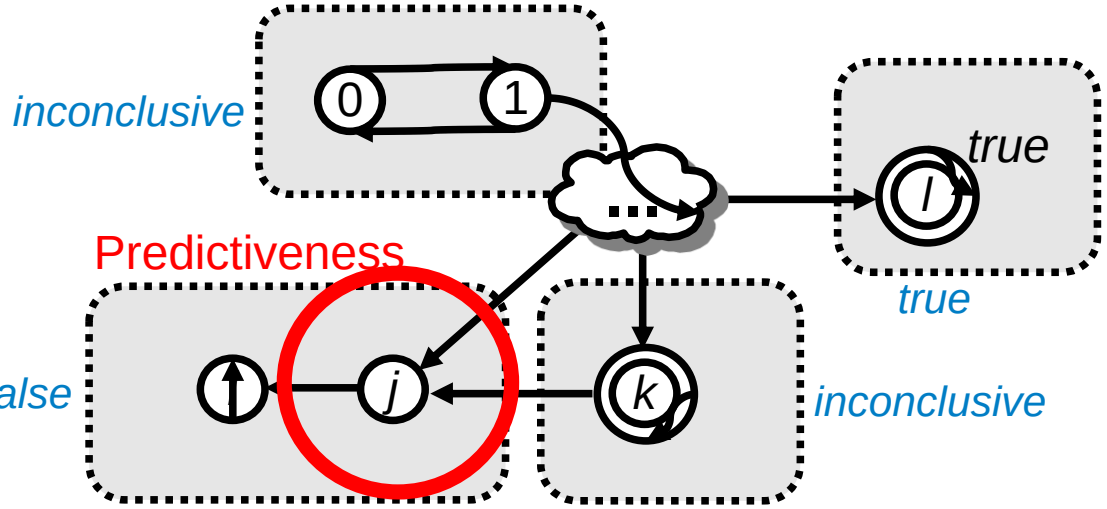
Für Monitoring: 3-wertige Variante der LTL-Semantik

Ziel beim Monitoring: So früh wie möglich feststellen, dass Compliance-Anforderung verletzt werden wird.

Verwende dafür 3-wertige Semantik:

$$[u \models \varphi] = \begin{cases} \top & \text{if } \forall \sigma \in \Sigma^{\omega} : u\sigma \models \varphi \\ \bot & \text{if } \forall \sigma \in \Sigma^{\omega} : u\sigma \not\models \varphi \\ ? & \text{otherwise} \end{cases}$$

Gibt endlichen Zustands-automat, um minimale Präfixe für unsicheren Zustand zu finden:



Beispiel-Anforderung: Server Finished

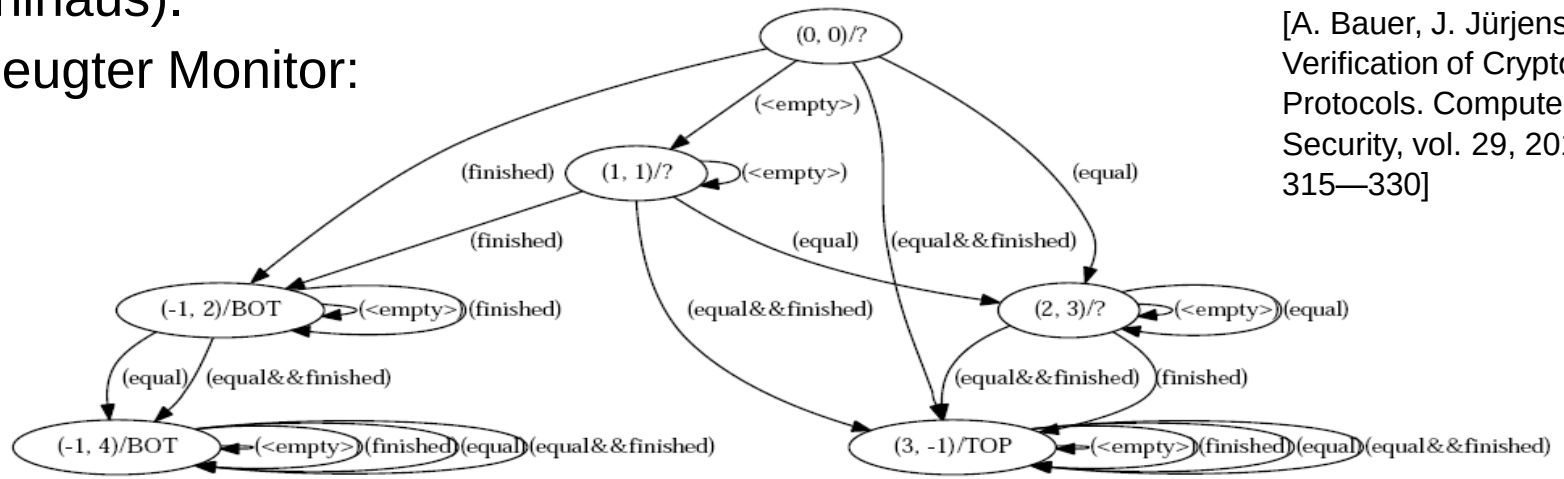
Server wird Finished Nachricht nicht schicken (Ereignis “finished”), bevor MD5 innerhalb Finished Nachricht vom Client erhalten wurde und gleich dem MD5 auf Seiten des Servers ist (Ereignis “equal”). Dann wird sie herausgeschickt. Formalisiert in LTL:

$$\varphi_2 = (\neg \text{finished } \mathbf{W} \text{ equal} \wedge (\mathbf{F} \text{ equal} \Rightarrow \mathbf{F} \text{ finished}))$$

(Schreibe F phi für true U phi (“eventually phi”), G phi für not F not phi (“globally phi”) und phi1 W phi2 für G phi1 or (phi1 U phi2) (“weak-until”).)

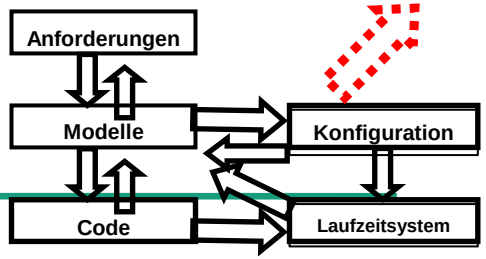
Keine Safety-Eigenschaft (d.h. geht über Schneider’s Security Automata hinaus).

Erzeugter Monitor:



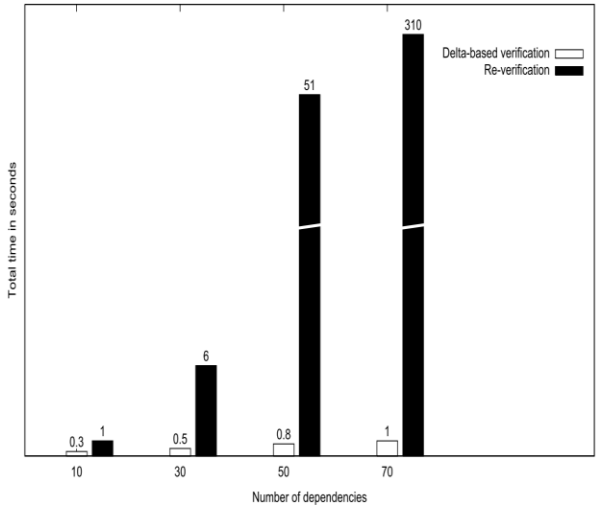
[A. Bauer, J. Jürjens. Runtime Verification of Cryptographic Protocols. Computers and Security, vol. 29, 2010, pp. 315—330]

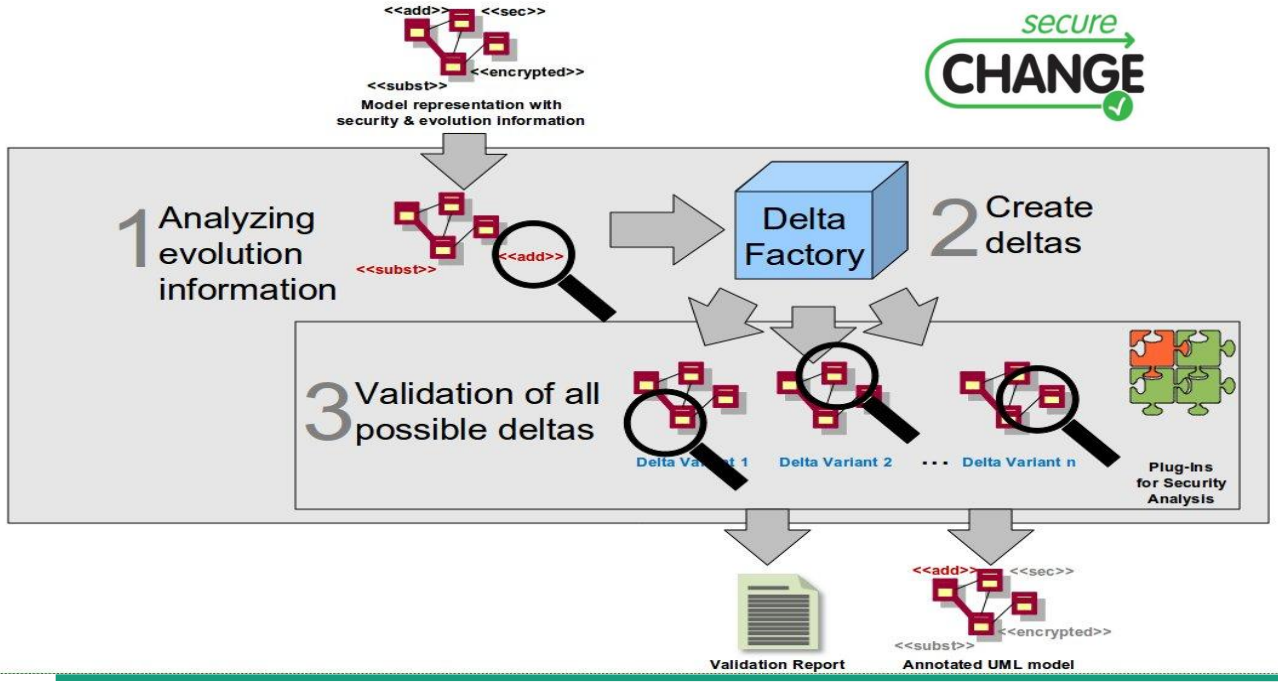
Werkzeugunterstützung: System-Evolution:



Bei Software-Evolution muss Verbindung zwischen Modell und Implementierung bewahrt (z.B. für Monitoring), sowie geändertes Modell re-verifizert werden.

Dafür Evolutions-basierte Analyse im Werkzeug; erheblicher Performanzgewinn gegenüber einfacher Re-verifikation.





[Jan Jürjens. 2011. Automated security hardening for evolving UML models. 33rd Int. Conf. on Software Engineering (ICSE '11). ACM.]

[A. Bauer, J. Jürjens, Y. Yu: Run-Time Security Traceability for Evolving Systems. Computer Journal 54(1): 58-87 (2011)]

Modellbasierte Compliance und Sicherheit für Cloud-basierte Services und Prozesse

- Einführung: Herausforderung Compliance und Sicherheit in Cloud-Marktplätzen 
- Modellierung und Analyse von Software-basierten Services auf Compliance-Anforderungen 
  - Modell-basierte Verifikation (Schritt 1) 
  - Von Modell zur Implementierung (Schritt 2) 
  - Konformanz der Implementierung zu Modell (Schritt 3) 
- Anwendungen

Weitere Anwendungen

- Gesundheitskarte: Architektur mit UMLsec untersucht, Schwachstellen aufgedeckt [Jour. Meth. Inform. Medicine 08]
- Internes Informationssystem [ICSE 07]
- Digitaler Formularschrank [SAFECOMP 03]
- Common Electronic Purse Specifications (Globaler Standard für elektr. Geldbörsen): mehrere Schwachstellen aufgedeckt [IFIPSEC 01, ASE 01]
- Biometrische Authentisierungssysteme: mehrere Schwachstellen aufgedeckt [ACSAC 05, Models 09]
- Gesundheitsinformationssysteme [Caise 09]
- Return-on-Security Investment Abschätzung
- Analyse Digitale-Signatur-Architektur
- IT-Sicherheits-Risikomodellierung
- Smart-card Software-Update Plattform

Aktuell:

- Cloud-Anwender Sicherheitsanalyse
- Sicherheitsökonomische Analysen
- Cloud-Anbieter Sicherheitsanalyse

Modell-basierte Compliance und Sicherheit: Überblick über das Forschungsfeld (Auswahl)

2001: UMLsec: UML profile for security modelling (Jürjens)
Model-based security testing with AutoFocus (Wimmel, Jürjens)
2002: Secure UML: Modelling RBAC with UML (Basin et al.)
Hypermedia security modeling with Ariadne (Aedo, Diaz et al.)
Aspect-oriented Security Modelling (France et al.)
Model-based IT security risk assessment (Stølen et al.)
Interactive theorem proving of UML models for security (Haneberg, Reif et al.)
2003: Formal verification for UML models of access control (Koch, Parisi-Presicce)
2004: Automated verification tools for UMLsec (Shabalin et al.)
Actor-centric modeling of user rights with UML (Breu et al.)
Extending OCL for secure database development (Fernández-Medina et al.)
2005: UMLsec Buch (Jürjens)
2007: Security monitors for UML policy models (Massacci et al.)
2008: Executable misuse cases for security concerns (Whittle et al.)
Business Process Compliance Checking (El Kharbili, de Medeiros, Stein, van der Aalst)
2009: Model-based security vs performance evaluation (Woodside et al.)
UMLsec Buch: Übersetzung ins Chinesische (Jürjens)
2010: Supporting Compliance through Enhancing Internal Control Systems by Conceptual Business Process Security Modeling (Riesner, Pernul)
From requirements to UMLsec models (Houmb et al.; Islam et al.; Mouratidis et al.)
Security monitoring for UMLsec models (Bauer et al.; Pironti et al.)
…

Einsatz in Unternehmen

Firmen setzen zunehmend modell-basierte Ansätze für Sicherheit und Compliance ein (z.B. Interactive Objects, ObjectSecurity, Corisecio, Thales, Foundstone (McAfee), …)

2005: Model-based Security ist Teil des Sammelwerkes “Build Security In” des US Dep. For Homeland Security.

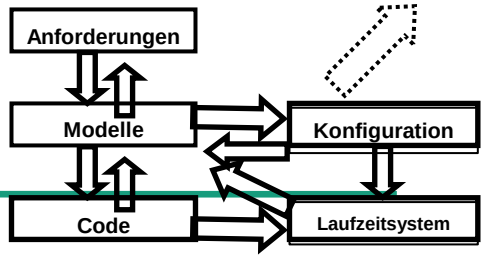
2007: “Model-Driven Security: Enabling a Real-Time, Adaptive Security Infrastructure” (Gartner, 21 September 2007): “Model-driven security is embryonic, but it will have a significant impact as information security infrastructures become increasingly real time, automated, and adaptive to changes in organizations and their environments.” [http://www.gartner.com/DisplayDocument?id=525109]

Lohnen sich automatische / modellbasierte QS-Ansätze im Vergleich mit klassischen Techniken (z.B. Testen) ?

Statische Analyse vs. Code Review: industrielle Software bei O2 (Germany) auf Fehler untersucht. Ergebnisse: Statische Analyse-Werkzeuge finden nur bestimmte Fehlerklassen, aber besonders verlässlich. Wichtigstes Ziel: “false positive”-Rate verringern. [Testcom 05]

Model-checking vs. Simulation / Testen: Türsteuergerät (in Koop. M. BMW). Typische Fehlerklassen: Simulation / Testen findet viele “einfache” Fehler schnell und effizient (z.B. falsche Transitionspriorität: einige Min.). Model-checking findet auch obskure Fehler (z.B. race conditions), aber Zusatz-Aufwand (1-2 Tage für LTL Formel). [Models 08]

Modellbasierte Compliance und Sicherheit für Cloud-basierte Services und Prozesse



Compliance-Report

Compliant: NEIN
Verstöße:
- MaRISK VA 7.2: Einhaltung von BSI G3.1 nicht erfüllt

Maßnahmen:
- BSI Maßnahmen-katalog M 2.62

Problem: Sicherheit und Compliance in Cloud-basierten Umgebungen sind komplexe und vielfältige Probleme (vielfältig wie Clouds selbst).

Ziel: Analyse / Überwachung der von Cloud-Service-Anbietern zugesicherten Sicherheit / Compliance

Idee: Entwicklung von automatischen Werkzeugen, die das Monitoring von Compliance-Anforderungen auf Basis von relevanten Artefakten unterstützen (z.B. Textdokumente, Schnittstellen-Spezifikationen, Geschäftsprozess-Modelle, Log-Daten, Quellcode).

Ergebnisse: Erfolgreiche Validierung der eingesetzten Techniken in industriellen Anwendungsprojekten.

Weitere Arbeiten: Analyse der eigenen Geschäftsprozesse auf Eignung zur Auslagerung in eine Cloud (bzgl. Sicherheit / Compliance) [vgl. früherer Vortrag]

Aktuelle Arbeiten:

- Kombination von Services unter Bewahrung von Compliance
- Modell-basierter Service-Entwicklungsansatz für „Compliance-by-Design“.