



Access Control Management

Dr. Andreas Wolf



NorCom Information Technology AG
Stefan-George-Ring 23
81929 München



Inhalt dieser Präsentation

- NorCom Information Technology AG
- Grundlagen
 - Trends in der IT-Sicherheit
 - Schlagworte und wie sie zusammenhängen
 - Ziele von Access-Control-Lösungen
- Die Theorie: Access-Control-Modelle
 - Role-Based Access Control (RBAC) und andere Modelle
 - Butterfly-Modell
- Der Anwendungsfall: EJB-Security
 - Abläufe in Application-Servern, Probleme und Lösungsansätze
 - Der J2EE Standard
 - Wünschenswerte Erweiterungen
- Die Realisierung: NGS BeanGuard
 - Wettbewerber
 - Features and Functions, interessante Details
 - Ausblick



NorCom Historie

- 1989
von Viggo Nordbakk gegründet
Software- und Beratungshaus für
große Rechnernetze
- Von 1989 bis 1999
solides Wachstum und stetiger
Aufbau der Kundenbasis im
Bereich System-integration, E-
Business/Internet Banking
- 1999 IPO
Exponentielles Wachstum
Effiziente Produktentwicklung
Diversifikation
- Seit 2000
Fokussierung
auf Secure E-Business, Virtual
Private Networks, Portale,
Marktplätze

NorCom

Erfahrung

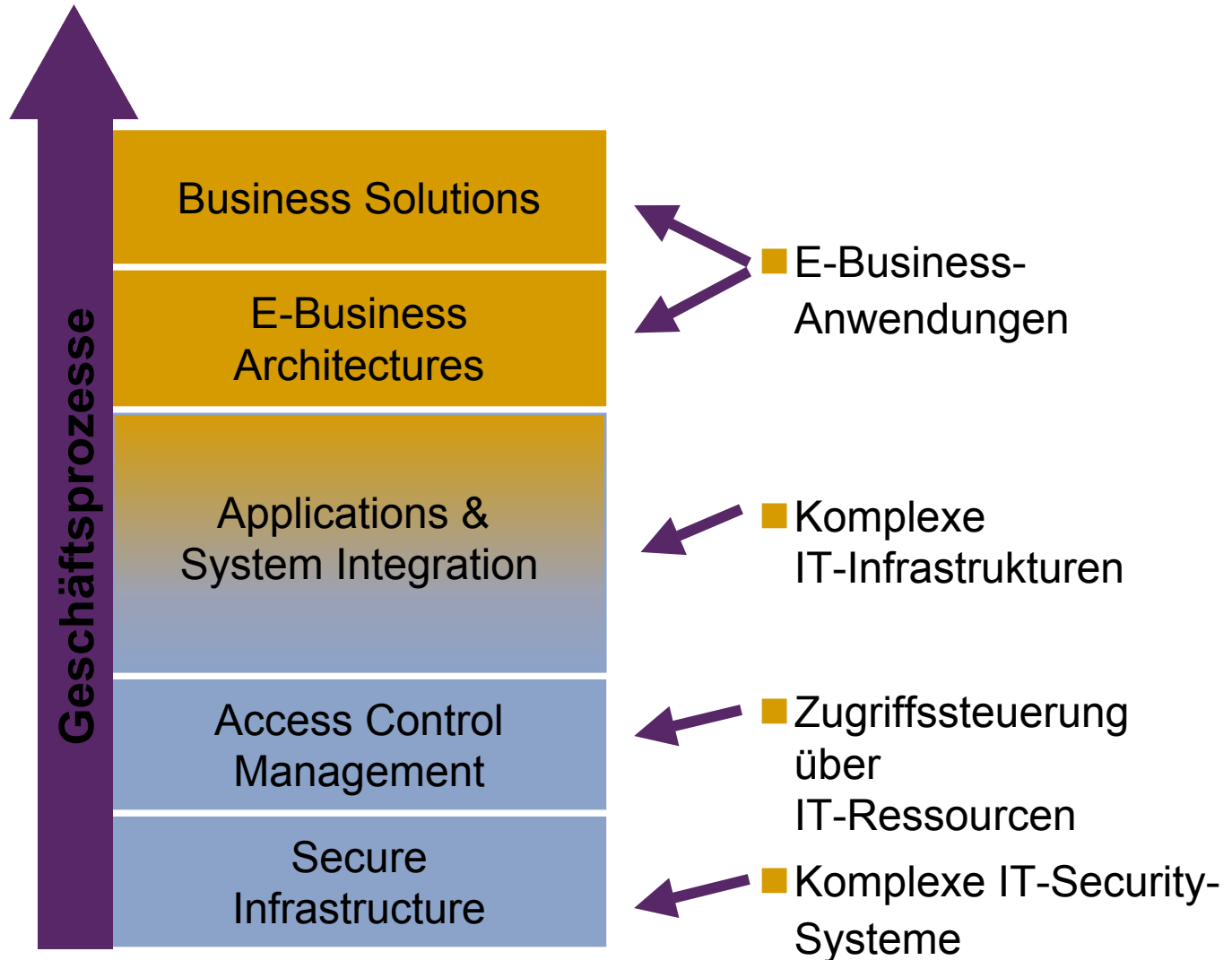
Kompetenz

Innovationskraft

Effektivität



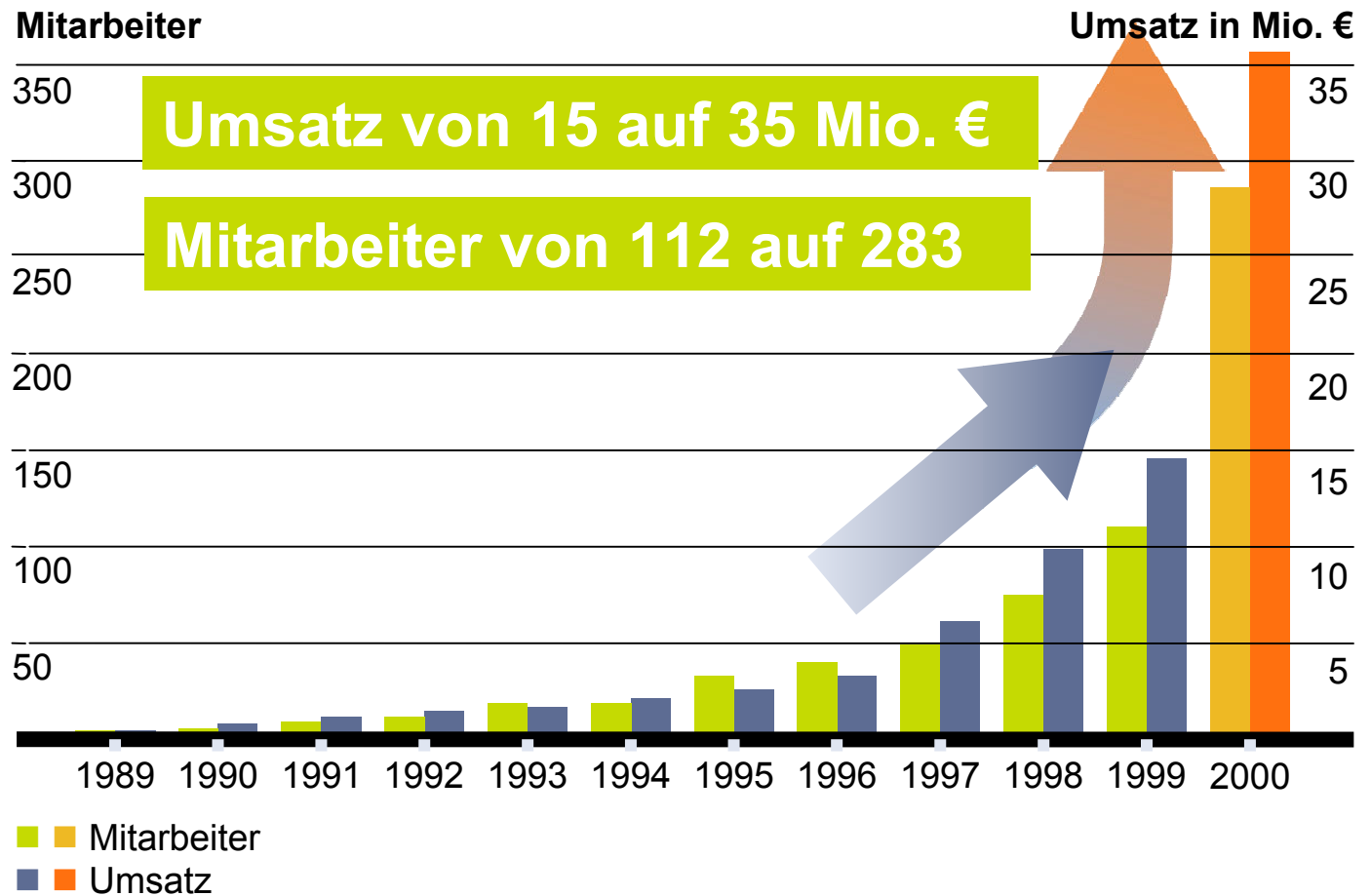
NorCom Solutions und Services



NorCom



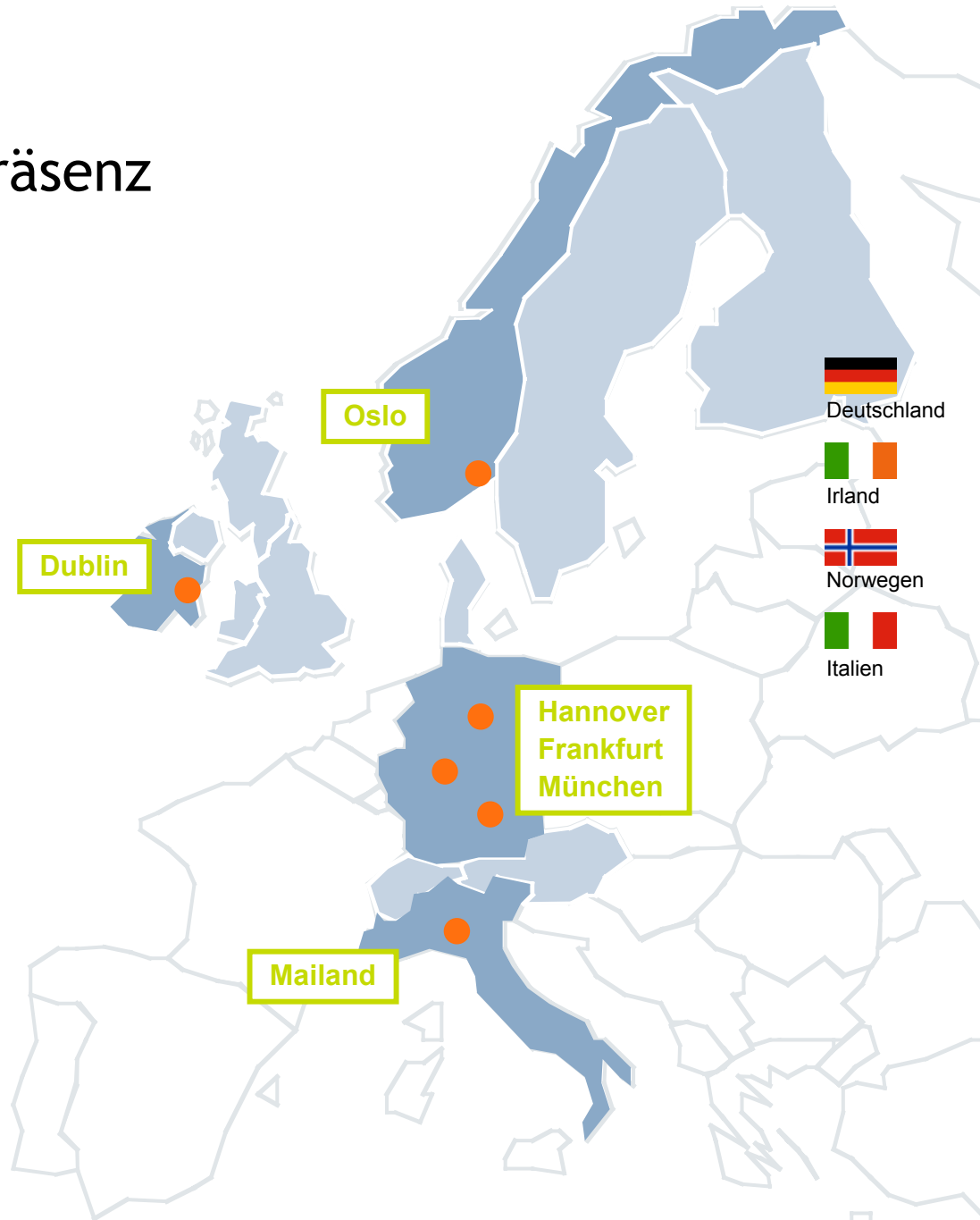
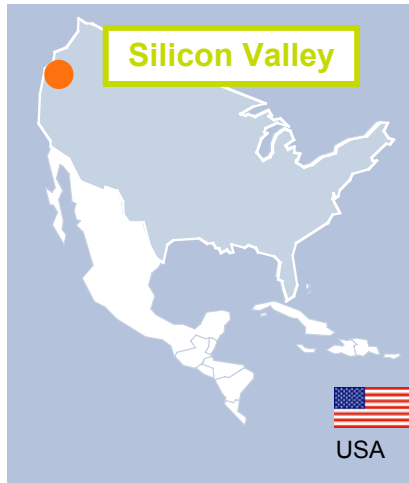
Konzernentwicklung



NorCom



Internationale Präsenz



NorCom



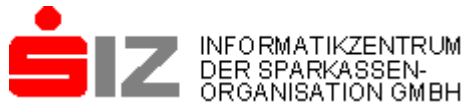
Auszug aus der aktuellen Kundenliste



Dresdner Bank



Bundesamt
für
Finanzen



SIEMENS



NorCom





Grundlagen

Die vier „A“ der IT-Sicherheit

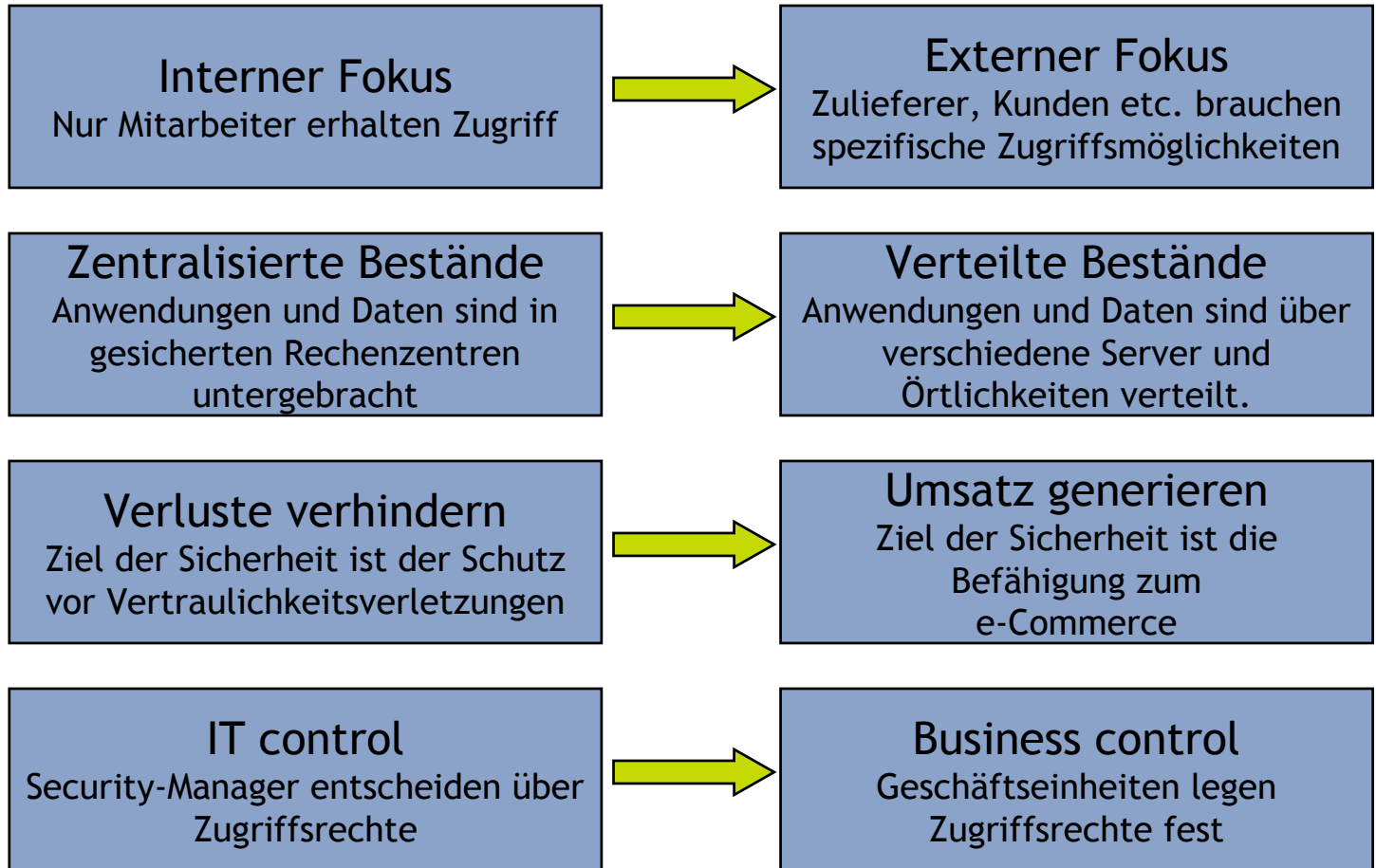


Trends der IT-Sicherheit

	Gestern	Morgen
Ziel	Bedrohungen abwehren	Zugriff ermöglichen
Hauptprinzip	Alles verbieten, was nicht erlaubt ist	Alles erlauben, was nicht verboten ist
Methode	Vorbeugung	Verantwortlichkeit
Durchsetzung	Zentralisierte Sicherheitsdienste	Einbettung in Anwendungen/Netze
Ansatz	Reaktiv	Proaktiv
Kontrolle	Zentral in IT-Abteilung	Unternehmensweit verteilt
Skalierbarkeit	Schlecht	Gut



Trends der IT-Sicherheit





Angriffe: Die Realität

- 80% aller bekanntgewordenen Angriffe auf IT-Systeme werden von Insidern begangen.
- Der durchschnittliche Schaden pro Angriff betrug dabei 2.500.000 US\$.
- Zum Vergleich: Nur 20% aller bekanntgewordenen Angriffe kommen von Personen außerhalb der betroffenen Unternehmen.
- Der durchschnittliche Schaden pro Outsider-Angriff beträgt „nur“ 57.000 US\$, das sind 2% des Durchschnittes pro Insider-Angriff.
- Insider sitzen meist hinter der Firewall und innerhalb des VPN.



Neun verbreitete Security-Fehleinschätzungen

- Sicherheit ist ein wichtiges Thema, aber nicht für uns.
- Das Problem wird sich von selbst lösen, wenn wir es nur lange genug ignorieren.
- Wir haben jetzt Technik gekauft. Damit ist das Problem gelöst. Auch wenn wir die Technik noch nicht beherrschen.
- Unsere Daten und unser Image sind nichts wert.
- Was hat Sicherheit mit unserem Geschäft zu tun?
- Wir haben doch eine Firewall.
- Sicherheitsaspekte können wir immer noch später hinzufügen.
- Sicherheit ist ein lästiger Kostenfaktor, keine Investition.
- Wir brauchen keine unternehmensweite Sicherheitsarchitektur.

VERTRAUEN ist Grundvoraussetzung!



Grundsätzliche Sicherheits-Eigenschaften

- **Confidentiality** → Sind Daten und Transaktionen vor unberechtigtem Zugriff geschützt?
- **Integrity** → Sind Daten und Transaktionen vor Verfälschungen geschützt?
- **Authentication** → Kann die Identität von Benutzern, und technischen Systemen überprüft werden?
- **Auditing** → Können Aktivitäten überwacht und protokolliert werden?
- **Authorization** → Erhalten alle Benutzer nur die ihnen zugewiesenen Rechte?
- **Encryption** → Sind Kommunikation und Datenhaltung geschützt?
- **Availability/
Reliability** → Sind die Dienste immer verfügbar?



Layered Approach

■ Gateway Layer

- Antwort auf die Frage: „Darf ich hereinkommen?“
- Festlegung des Zugangs zum Netzwerk
- Beispiele: Firewalls, Intrusion-Detection-Systeme, VPNs, Authentifizierungssysteme

■ Control Layer

- Antwort auf die Frage „Wohin kann ich gehen?“
- Ermöglichen der Administration von Zugriffserlaubnissen
- AC-Management-Systeme, Single Sign-On Lösungen
- Einheitliche Administration, unmittelbare Wirksamkeit, Ersatz für papiernes Antragswesen

■ Data Layer

- Antwort auf die Frage “Was kann ich tun?“
- Festlegung der Zugriffsrechte, Datenhaltung
- Datenbanken, LDAP-Verzeichnisse



A-1: Authentifikation

- Zugriffssteuerung in Abhängigkeit von Authentifikations-Qualität und Zugang
 - HTTP basic, forms, Zertifikate, SSL, Smart Cards, Biometrie
- Authentifikationsqualität
 - What I know: Passwörter, TAN-Systeme
 - What I have: Smart Card, Token
 - What I am: Iris-Scan, Fingerprint-Scan, Spracherkennung
- Zugang
 - Intranet
 - Offenes Internet
 - SSL
 - VPN
 - ...



A-1: Authentifikation

- Sitzungsmanagement, Sitzungsverfolgung
 - Session time-out: Festlegung einer maximalen Sitzungsdauer
 - Idle time-out: Inaktive Sitzungen werden verworfen
 - Sitzungsbasierte Aktivitätsverfolgung
- Anonyme Benutzer
 - Anonyme Benutzer können ohne explizite Authentifikation zugreifen
 - Die Sitzungen anonymer Benutzer sollten unterschieden werden können
- Cross-domain single sign-on
 - Benutzer sollen auf verschiedene Ressourcen, selbst in unterschiedlichen Domains, ohne neue Anmeldung zugreifen können
 - Lösungen: Session-Cookies, URL-Rewriting
- Integration mit existierender Infrastruktur
 - LDAP-Zugriff
 - API für Integration mit bestehenden Anwendungen



A-2: Autorisierung

- Deklaratives Access-Control Management oder programmatischer Zugriffsschutz?
 - Aufrufe aus der Programmlogik (z.B. `isCallerInRole()`)
 - Deklaratives AC-Management (Deployment-Deskriptoren)
 - Trennung von Business-Logik und Security-Logik empfehlenswert
- Access-Control-Lists (ACL) oder komplexe Strukturen?
 - Die Auswirkungen von ACLs sind (meist) klar und überschaubar
 - Die Bewertung der Wirksamkeit einer Gesamtpolicy ist schwer
 - Graphbasierte/Rollenbasierte Verfahren erlauben die kompakte Modellierung komplexer Rechtesysteme, haben aber eine flachere Lernkurve
 - Die Beurteilung der Wirksamkeit einer Gesamtpolicy ist einfacher
- Schutz heterogener Umgebungen
 - Zentrale/verteilte bzw. standardisierte/proprietäre Zugriffskontrolle
 - API-Zugriffsmöglichkeiten aus diversen Plattformen (Betriebssysteme, Application-Server, Agenten etc.)
- Personalisierung
 - Ermittlung von Rechteprofilen



A-3: Auditing

- Ereignisbezogene bzw. sitzungsbasierte Verfolgung von Aktivitäten
 - Logging aller Authentifizierungs-, Autorisierungs- und Administrations-Ereignisse (Anfragen und Ergebnisse), eventuell auch technischer Ereignisse
 - Unterscheidung verschiedener anonymer Sitzungen
- Personalisierung von Anwendungen mittels Benutzerprofilen und Zugriffsprotokollen
 - Protokollierung aller Zugriffe, nicht nur der auf geschützte Ressourcen
 - Ereignisdatenbank für die Auswertung des Benutzerverhaltens
- Integration mit existierender Infrastruktur
 - Schnittstellen z.B. für Intrusion-Detection-Tools
 - API für Integration mit bestehenden Anwendungen (Auditing- und Reporting-Tools)



A-4: Administration

- Leichte Bedienung und Abdeckung der benötigten Funktionalitäten
 - Diese beiden Anforderungen stehen oft etwas im Widerspruch
 - Werkzeuge mit unterschiedlichen Zielgruppen
 - Einfache GUI mit 80% der Funktionalität
 - CLI mit direktem API-Durchgriff und 100% Funktionalität
- Standardisierte Verwaltung oder proprietäre Einzellösungen
 - Vertrautheit und Bedienungssicherheit oder
 - Perfekte Eignung für speziellen Zweck
- Integration mit existierender Infrastruktur
 - API für Integration mit bestehenden Administrations-Anwendungen
- Verteilte und/oder delegierte Administration
 - Unterteilung der Privilegien in Administrationsgruppen für Ressourcen, Rechte, Benutzer etc.
 - Eventuell werden feingranular vergebene Rechte benötigt
 - Rechtedelegation auf Geschäftseinheiten
- Ziel: möglichst geringe Administrationskosten



Ziele des Access-Control-Managements

- Kontrolle/Steuerung des Zugriffs auf Ressourcen sichern
 - Verhindern von unautorisierten Zugriffen auf Ressourcen
 - Protokollierung von sicherheitsrelevanten Ereignissen
 - Datengewinnung für Personalisierungen
 - Integration mit existierenden Systemen, ganzheitlicher Ansatz
- Möglichst „einfache“ Implementierung und Durchsetzung von Sicherheitsrichtlinien von Unternehmen
 - Definierte Wege von einer verbalen Festlegung zur Implementierung
 - Handhabbare Kontrollmöglichkeit der Realisierung
 - Einfache Administration
 - Geringe Anfälligkeit gegen Fehlbedienungen
 - Reduktion von Sicherheitsrisiken
- Reduktion von Entwicklungskosten und Entwicklungszeit
 - Trennung von Business- und Securitylogik, Modularität
- Unsichtbarkeit für den Benutzer
 - Der Surfer am Browser soll/will von der AC-Maschinerie nichts wissen
- Keine oder zumindest keine wahrnehmbaren Performanzverluste
 - Teilweise nichttriviale Algorithmen erforderlich
 - Balance zwischen Zeitbedarf und Funktionalität

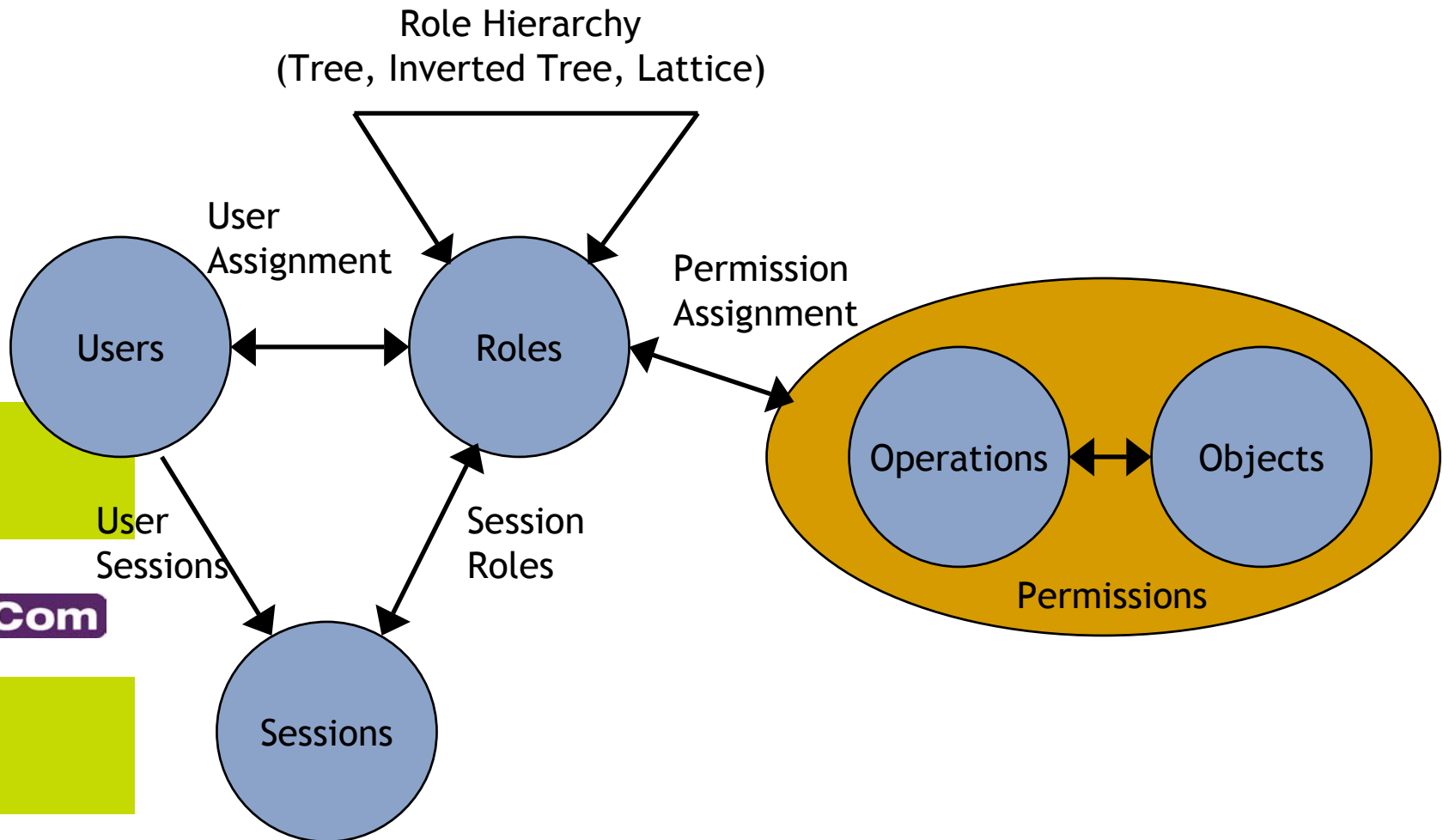


Access-Control-Modelle

RBAC und Co.



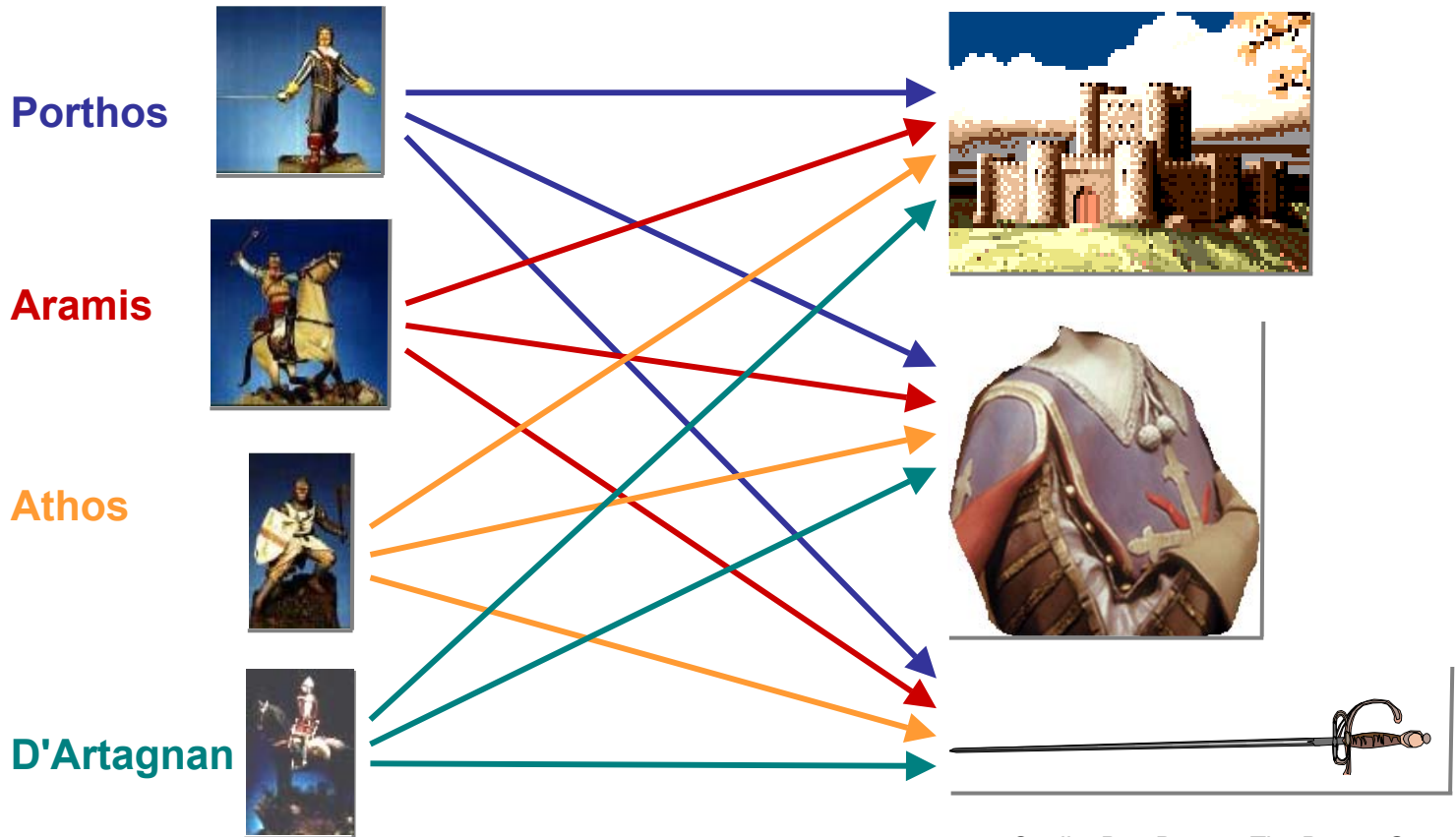
Role-Based Access Control



NorCom



Vier Musketiere - ohne RBAC



#User x #Resources = 12 Zuordnungen

Quelle: Don Bowen, The Burton Group



Vier Musketiere - mit RBAC

Porthos



Aramis



Athos



D'Artagnan



Rolle: Musketier



#User + #Permissions = 7 Zuordnungen

Quelle: Don Bowen, The Burton Group

NorCom



Related work

- Role-Based Access Control Models (Sandhu et. al., IEEE Computer 29(2), 1996)
- Security Models for Web-Based Applications (Joshi et. al., ACM Communications 44(2), 2001)
- A Proposed Standard for Role-Based Access Control (Ferraiolo et. al., NIST, 2000)
- Injecting RBAC to Secure a Web-based Workflow System (Ahn et. al., RBAC, 2000)
- Andere Veröffentlichungen (speziell von Sandhu und Ferraiolo)



Vergleich RBAC-ACL

	Role-Based Access Control	Access Control Lists
Objekte	User/Group Role Session Object Operation Permission	User/Group -/- -/- Datei/Verzeichnis Berechtigung -/-
Struktur	hierarchisch graph-basiert	flach punktuell
Ausstrahlung	Vererbung	Duplikation
Sichtbarkeit	ganzer Graph	einzelne Datei einzelnes Verzeichnis

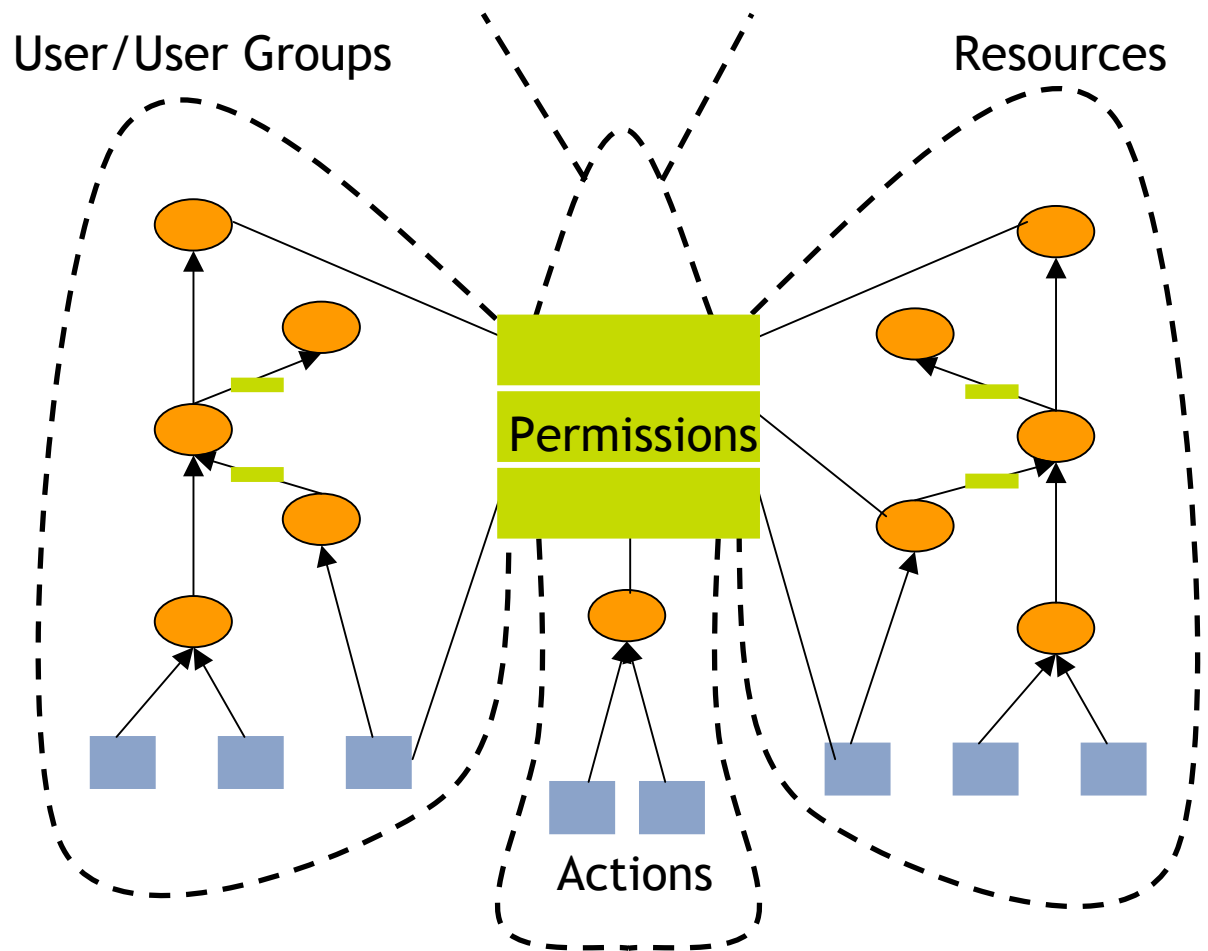


Argumente für RBAC

- Im RBAC ist eine ganzheitliche Modellierung möglich, mit ACL nicht.
- RBAC ermöglicht die Schaffung von Hierarchien, ACLs bieten nur punktuelle Wirksamkeit.
- RBAC ermöglicht Vererbung der Wirksamkeit von Rechten, ACLs werden physisch kopiert und müssen später einzeln administriert werden.
- RBAC ermöglicht die Antwort auf Fragen, die Kenntnis vom Gesamtzusammenhang erfordern, wie „Auf welche Ressourcen darf ein Benutzer A zugreifen?“
- RBAC ist kompakter.
- RBAC ist plattformübergreifend implementierbar.
- Die Wirksamkeit einer in einem RBAC-Modell implementierten Policy ist leichter zu beurteilen.
- RBAC-Modelle sind leichter und schneller zu verändern.



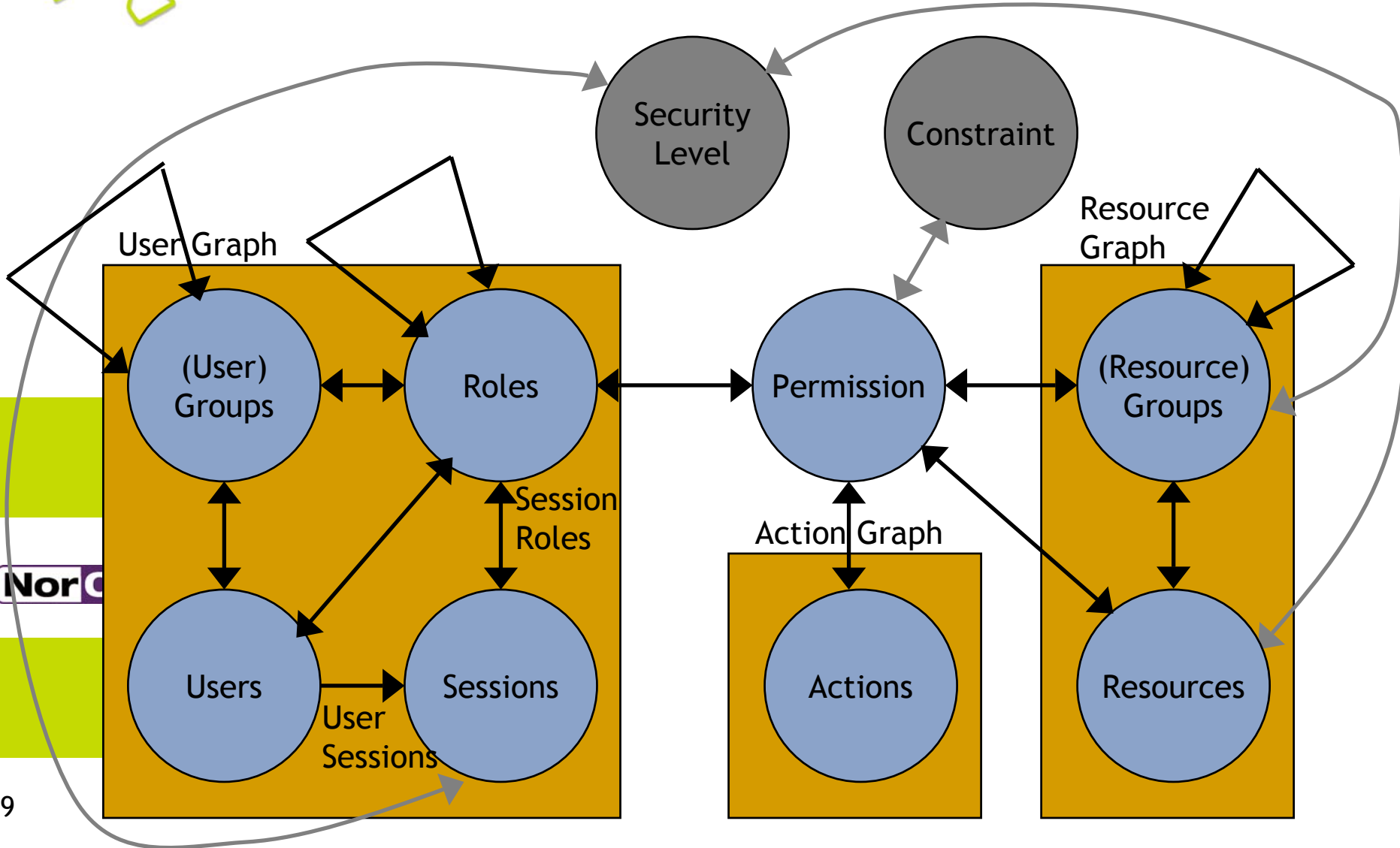
Das Butterfly-Modell



NorCom



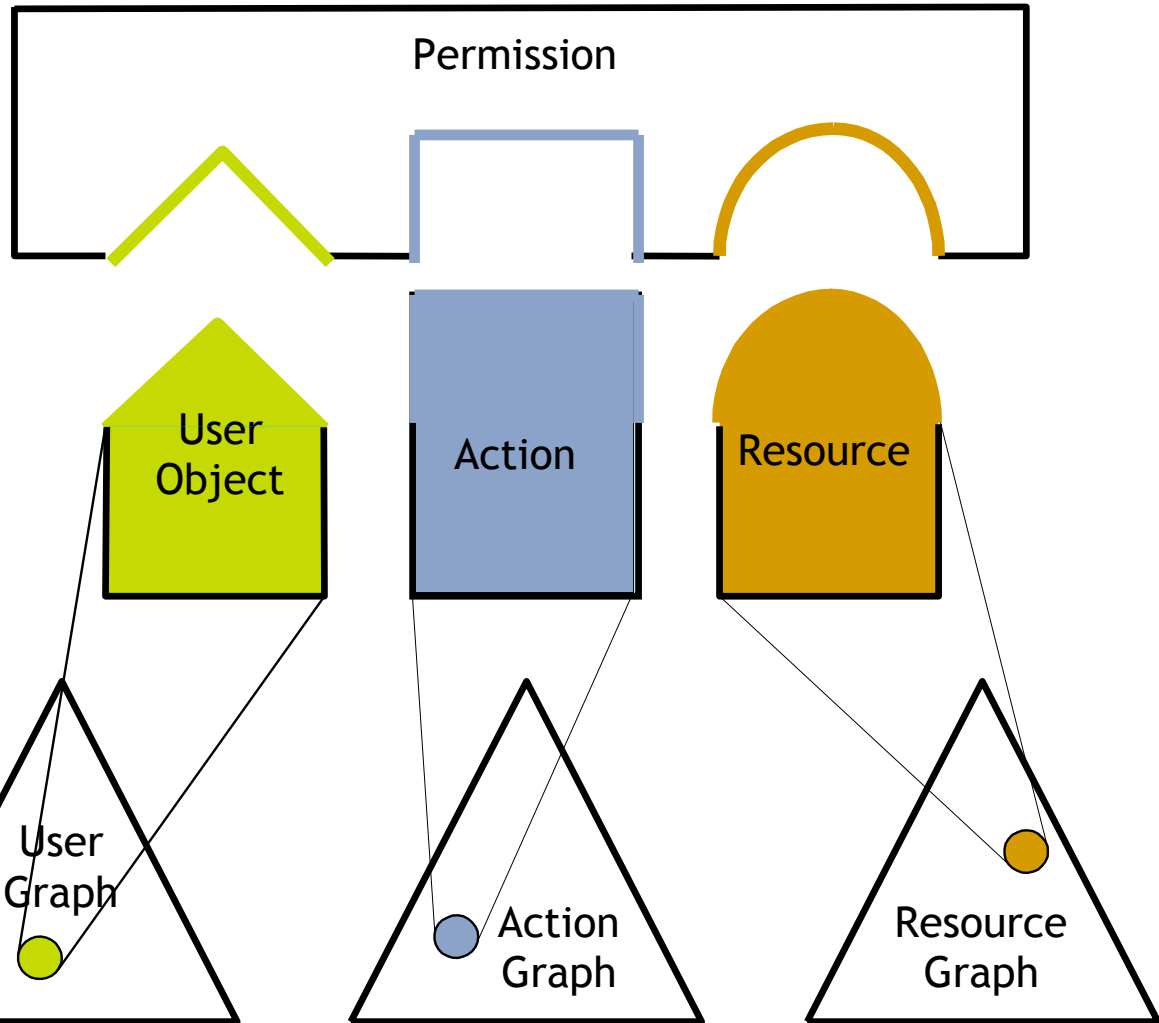
Objektrelationen im Butterfly-Modell



NorC



Rechte



NorCom



Vergleich RBAC-Butterfly

	Role-Based Access Control	Butterfly Model
Objekte	User/Role Object Operation Permission	User/User Group/Role Resource/ <u>Resource Group</u> Action (Permission)
Zuordnungsobjekte	Role Session	Permission Session
Vergleich	<ul style="list-style-type: none">■ Hierarchisierung von Rechten■ Einführung von Sessions■ Flache Ressourcen	<ul style="list-style-type: none">■ <u>Strukturierung von Benutzern und Ressourcen</u>■ Trennung von Strukturierung und Rechtezuordnung■ Butterfly ist RBAC-Variante, die Ressourcenmodellierung erweitert

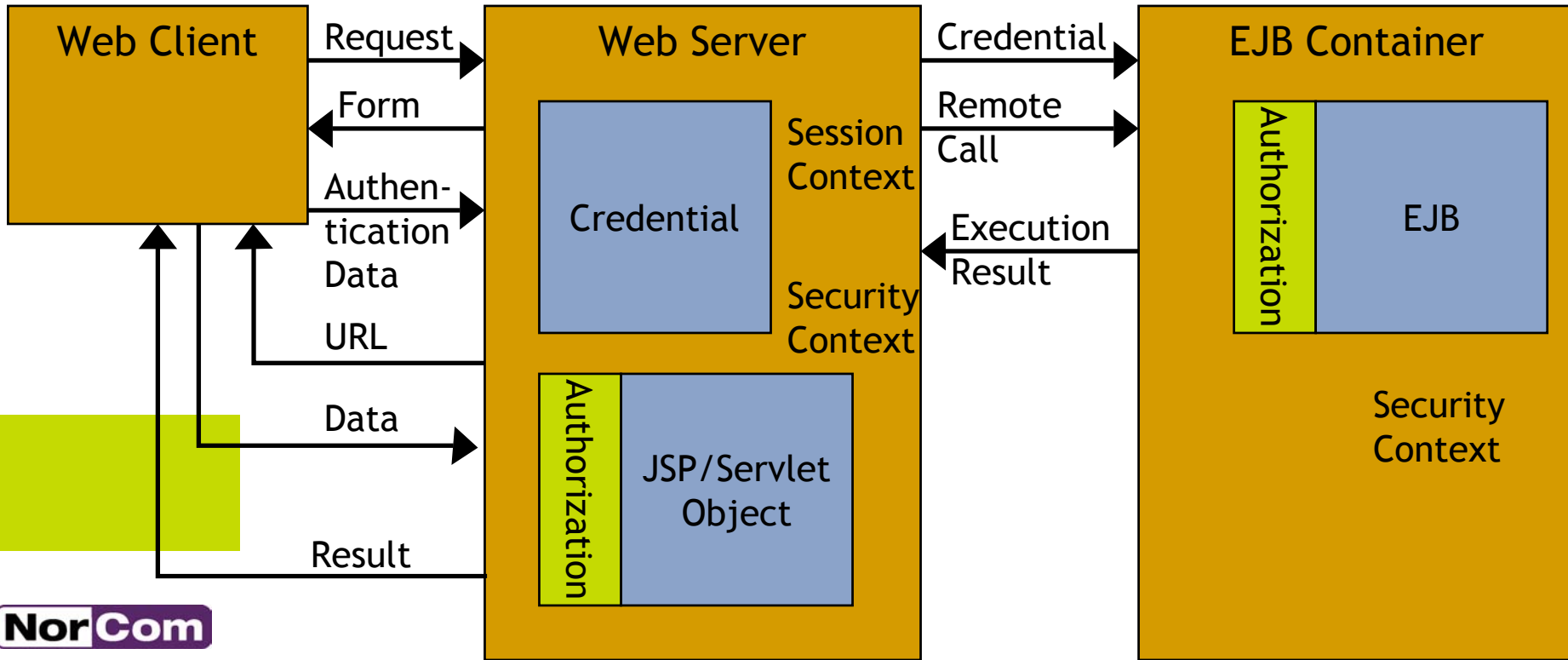


J2EE (1.3)/EJB (2.0)-Security

Verteilt = Verstreut ?



Abläufe im Application Server



1. Initial Request
2. Initial Authentication
3. URL Authorization
4. Fulfilling Request
5. Invoking Bean Methods



Ziele der J2EE Security-Architektur

- Portabilität
- Transparenz
 - Anwendungsprogrammierer brauchen keine Security-Kenntnisse
- Isolation
 - Trennung der Security-Aufgaben von Programmierer, Deployer und Systemadministrator
- Erweiterbarkeit
 - Komponentenmodell, standardisierte Schnittstellen
- Flexibilität
- Abstraktion
 - Alle sicherheitsrelevanten Eigenschaften einer Anwendung sind in den Deploymentdeskriptoren beschrieben
- Unabhängigkeit
- Kompatibilitäts-Testbarkeit
- Sicheres Zusammenwirken



J2EE-Security-Terminologie

- Principal
 - Authentifizierbare Einheit, wird über Namen und Authentifizierungsdaten authentifiziert
- Security Policy Domain
 - Auch Security Domain, Schutzbereich, in dem eine einheitliche Security Policy wirkt
- Security Technology Domain
 - Schutzbereich, in dem ein einheitlicher Schutzmechanismus (z.B. Kerberos) wirkt
- Security Attributes
 - Einem Principal zugeordnete Eigenschaften für diverse Verwendungen, z.B. für Authentifizierungsprotokolle
- Credential
 - Security-Attribute (oder Verweis auf solche), die zur Authentifizierung eines Principals benutzt werden können
 - Ein Principal bekommt ein Credential durch Authentifizierung oder Delegation



J2EE-Security-Verantwortlichkeiten

■ Bean Provider

- Programmatischer Zugriffsschutz
 - Test auf Rollenzugehörigkeit (codierte Rollen)
 - Prüfung des Aufrufers

■ Application Assembler

- Deklarativer Zugriffsschutz
 - Definition von (logischen) Rollen
 - Zugriffserlaubnisse für logische Rollen
 - Zuordnung von codierten zu logischen Rollen
 - Impersonierungen: run-as Identitäten

■ Deployer

- Zuordnung von Security Domain und Principal Realm
- Zuordnung von logischen Gruppen zu Principals bzw. Gruppen von Principals
- Principal-Delegation: run-as Identitäten

■ System Administrator

- Benutzer/Benutzergruppen-Administration
- Principal-Zuordnung

■ (EJB Client und EJB Container Provider)



Schutzmöglichkeiten nach J2EE

- Sämtliche deklarativen Schutzvereinbarungen werden in den XML-Deploymentdescriptoren getroffen
- Programmatische Security
 - Boolean `isCallerInRole (string roleName)`
 - `java.security.Principal` `getCallerPrincipal ()`
 - Servlet-Kontext: `isUserInRole ()`, `getUserPrincipal ()`



Schutzmöglichkeiten nach J2EE

■ Deklarative Security

- **<assembly-descriptor>**: Enthält Security-Definitionen
- **<role-name>**: Namensdefinition
- **<security-role-ref>**: Definition der aus isCallerInRole vergleichbaren Rollen, Namensraum beanweit
- **<role-link>**: Verbindung der Namen von security-role und security-role-ref
- **<security-role>**: Definition der für Zugriffsrechte und Impersonierungen verwendbaren Rollen, Namensraum ejb-jar fileweit
- **<method-permission>/<method-name>/<method-param>**: Zuordnung von Zugriffsrechten auf Beans/Methoden/Methoden+Signatur
- **<exclude-list>**: Liste der nicht zugreifbaren Methoden
- **<security-identity>/<run-as>**: Impersonierung zu angegebener Rolle, Eindeutige Auflösung der Rolle zum Principal notwendig



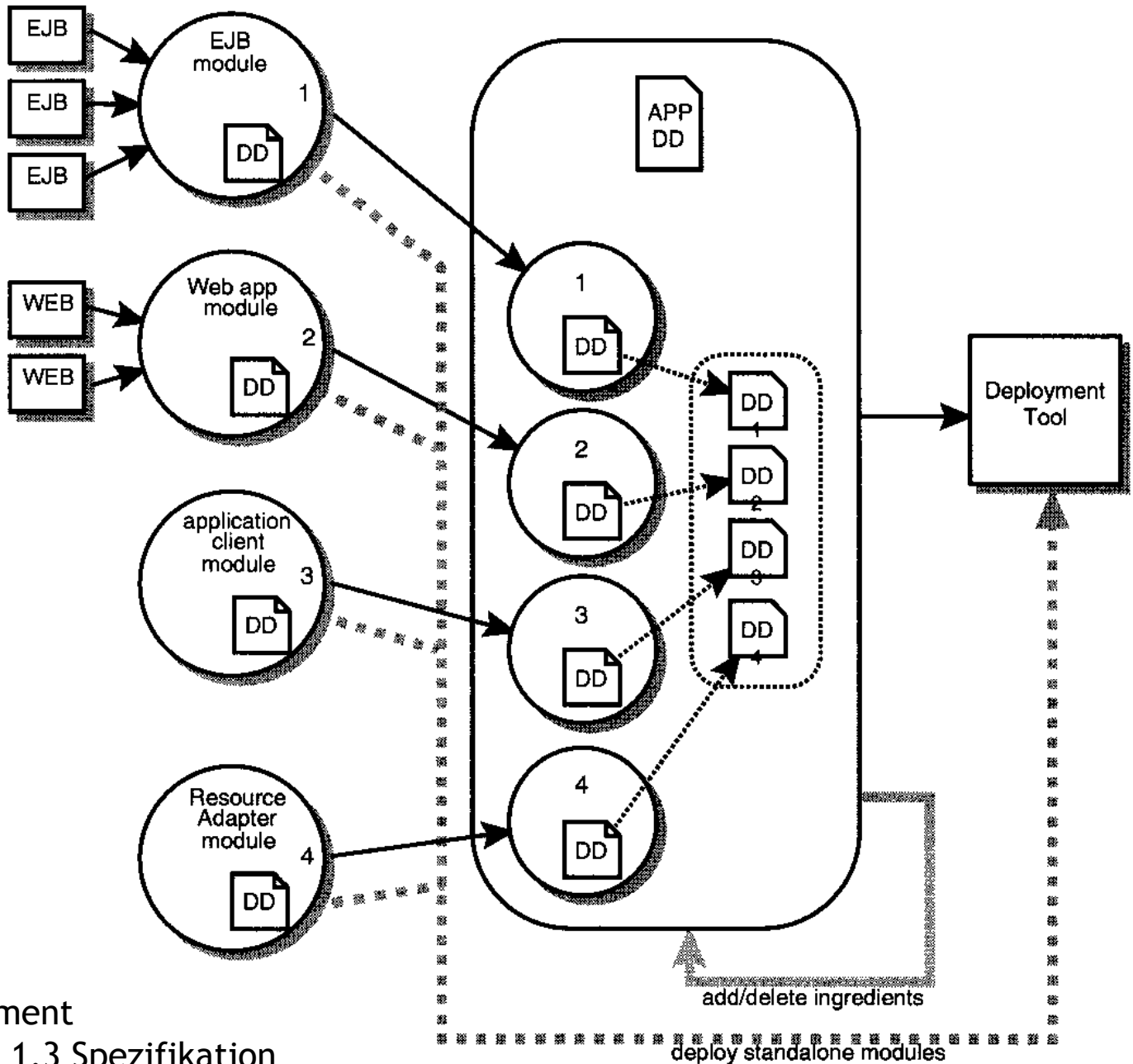
Schutzmöglichkeiten nach J2EE

- Schützbare Einheiten
 - „ganze Bohnen“
 - Methoden unabhängig von ihrer Signatur
 - Methoden mit Angabe ihrer Signatur
 - Impersonierungen für ganze Beans
- Zugriffs-Subjekte
 - Codierte Rollen
 - Logische Rollen

Components

J2EE
Modules

J2EE Application



NorCom



Lokalisierung der Security-Definitionen

- Bean-Code:
 - Definition der codierten Rollen
- Deployment-Deskriptoren der Bean(s): ejb-jar (.jar-File)
 - Definition der logischen Rollen
 - Zuordnung der logischen zu den codierten Rollen
 - Zugriffserlaubnisse der Rollen auf Methoden
 - Impersonierung
- Deployment-Deskriptoren der Applikationen (.ear-File)
- Deployment-Deskriptoren der Web-Komponenten (.war-File)
- Deployment-Descriptor des Containers
- Betriebssystem/Directory



Wünschenswerte Erweiterungen

- Schutz weiterer „Objekte“ wie
 - Interfaces
 - Instanzen: nicht nur Schutz auf Klassenebene (J2EE: Instance-based Access Control)
- Administrationsvereinfachung
 - Zusammenfassung der Security-Administration der diversen Deployment-Deskriptoren
 - Standardisierung der Security-Administration für diverse Application Server
 - Zusammenfassung mit der Security-Administration z.B. des Webservers
- Hierarchisierung
 - Zusammenfassung der diversen EJB-Ressourcen in Hierarchien
 - Bezeichnung von Ressourcen mit regulären Ausdrücken



Lösungsansätze für Access-Contol-Mechanismen

- Instrumentierung vorhandener Beans/Applikationen und Einführung einer AC-Dispatcher-Bean
 - Veränderung bei Applikationen
 - Supportverlust beim Applikationshersteller wahrscheinlich
 - Containerspezifisches Rewriting der Applikation
 - Keine Containerveränderung
- Patch des Containers zur Umlenkung der Abfragen auf den eigenen Access-Control-Mechanismus
 - Veränderung des Containers
 - Supportverlust des Containerherstellers wahrscheinlich
 - Keine Applikationsveränderung



Lösungsansätze für Access-Contol-Mechanismen

- API-Zugriff aus dem Applikationscode mittels Java-Connector-Architektur
 - Generisches Verfahren
 - Eingriff in Applikationscode
 - Noch nicht von allen Containern unterstützt
- Generierung der Deployment-Deskriptoren aus dem eigenen Access-Control-Mechanismus
 - Keine unmittelbare Wirksamkeit
 - Kein Eingriff in Container



Realisierung

NorCom NGS BeanGuard





Marktübersicht (Auswahl)

- Mitspieler im Access Control Management Markt



Webthority



SiteMinder



NGS BeanGuard



Assure Access



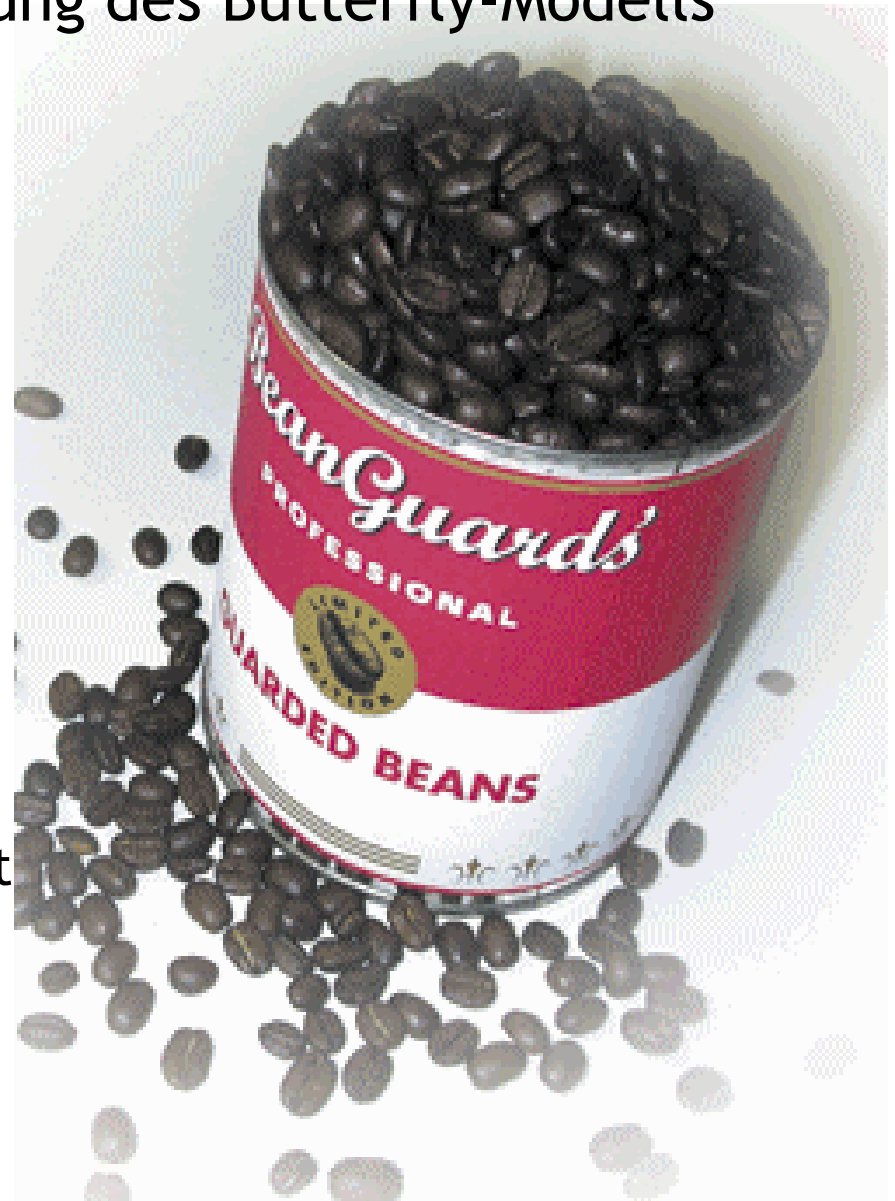
Policy Director



Technische Realisierung des Butterfly-Modells

NGS BeanGuard versetzt Anwender in die Lage, eine sichere und vertrauenswürdige E-Commerce-Infrastruktur zu betreiben, auf der weltweite Geschäftsprozesse mit Partnern, Mitarbeitern und Kunden durchgeführt werden können.

NorCom





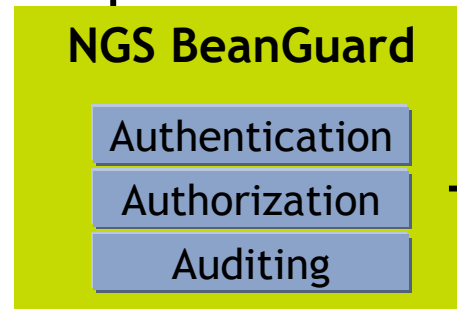
Datenfluss



Supplier
Employee
Reseller
Customer



Content

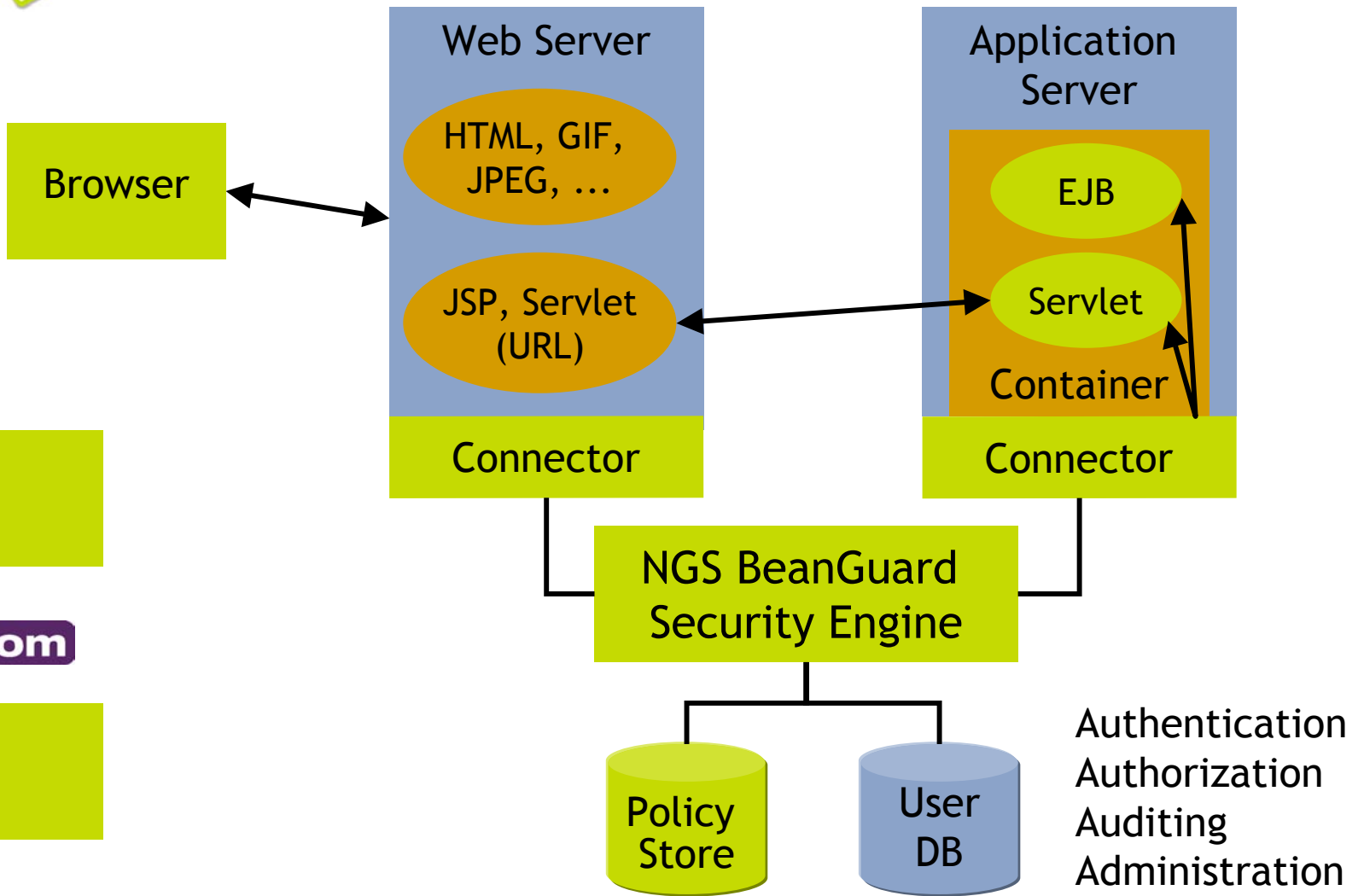


Applications



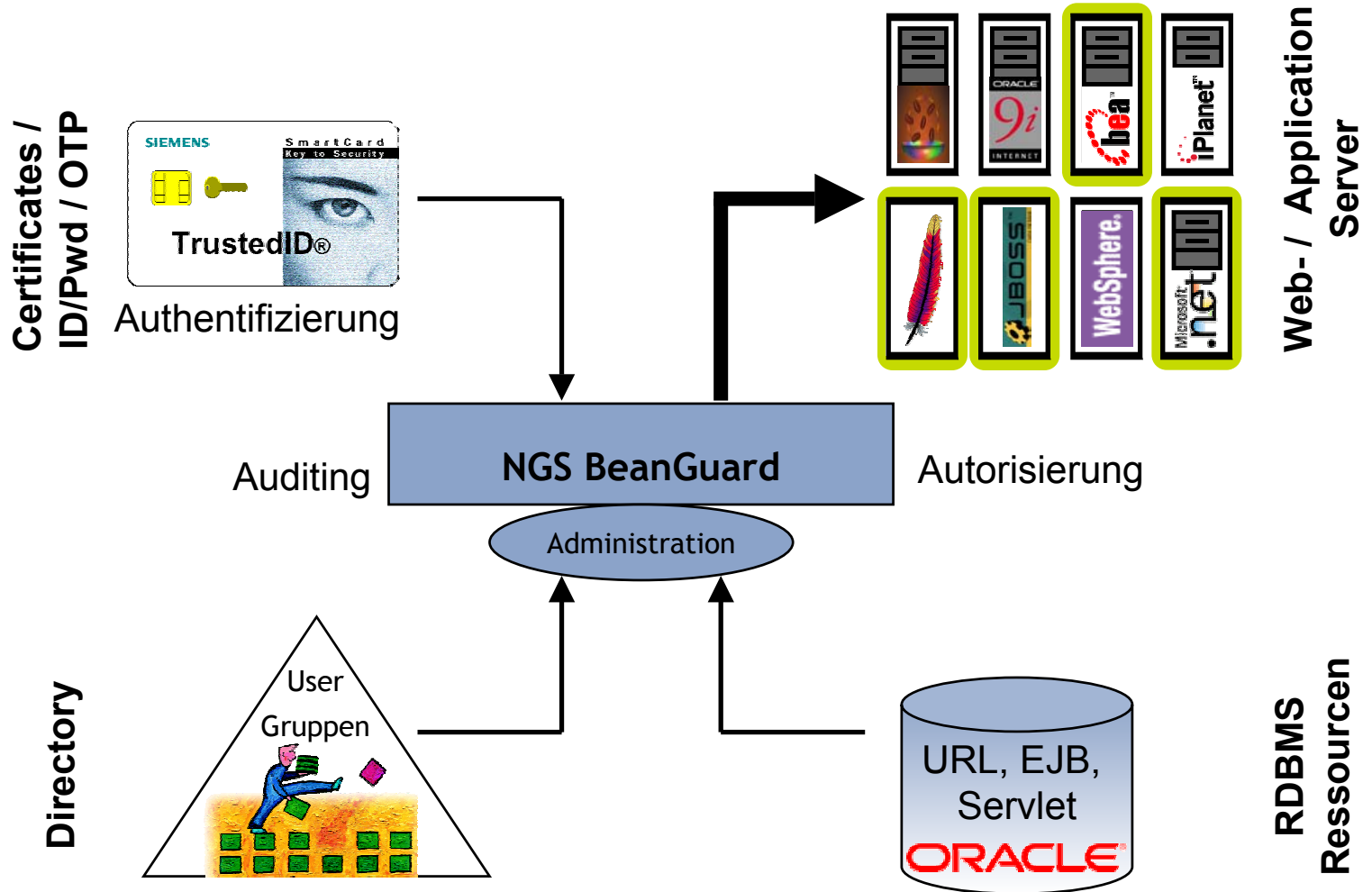


Konfigurationsbeispiel





Umgebung mit NGS BeanGuard



NorCom



Schützbare EJB-Ressourcen in BeanGuard

- URL-Ressourcen
 - Files,
 - Scripts,
 - Servlets,
 - etc.
- URL-Muster
 - Directories,
 - einfache reguläre Ausdrücke geplant
- EJB-Ressourcen
 - Beans als Gesamtheit
 - EJB-Komponenten nach J2EE: Methoden, Methoden mit Signatur
- Zusätzliche EJB-Komponenten
 - Interfaces,
 - Instanzen,
 - Instanzen-Interfaces, Instanzen-Methoden, Instanzen-Methoden mit Signatur



Schützbare EJB-Ressourcen in BeanGuard

- **Codierte Rollen**
 - <security-role-ref> XML-Terme
 - Als Attribute der Beans und Verweise auf Benutzergruppen
- **Logische Rollen**
 - <security-role> XML-Terme
 - Als Benutzergruppen
- **Impersonierungen**
 - <security-identity> und <run-as> XML-Terme
 - Als Attribute der Beans und Verweise auf Benutzergruppen
- **Ressourcengruppen**
 - Beliebige Hierarchien von Ressourcen
 - An die Bean-Struktur angelehnte Hierarchien



Bean-Komponenten-Modellierung

- EJB-Komponenten: Eindeutige Identifikation durch 6-Tupel:
 - **Server:** Application Server, auf dem sich die Bean befindet
 - **JNDI name:** Klassen-Name der Bean
 - **Primary key:** Primärschlüssel, mit dem die Bean-Instanz beim Zugriff in der Persistenzschicht (Datenbank) identifiziert werden kann
 - **Interface:** Home- oder Remote-Interface
 - **Method name:** Name der Methode
 - **Signature:** Liste der Parametertypen der Methodensignatur

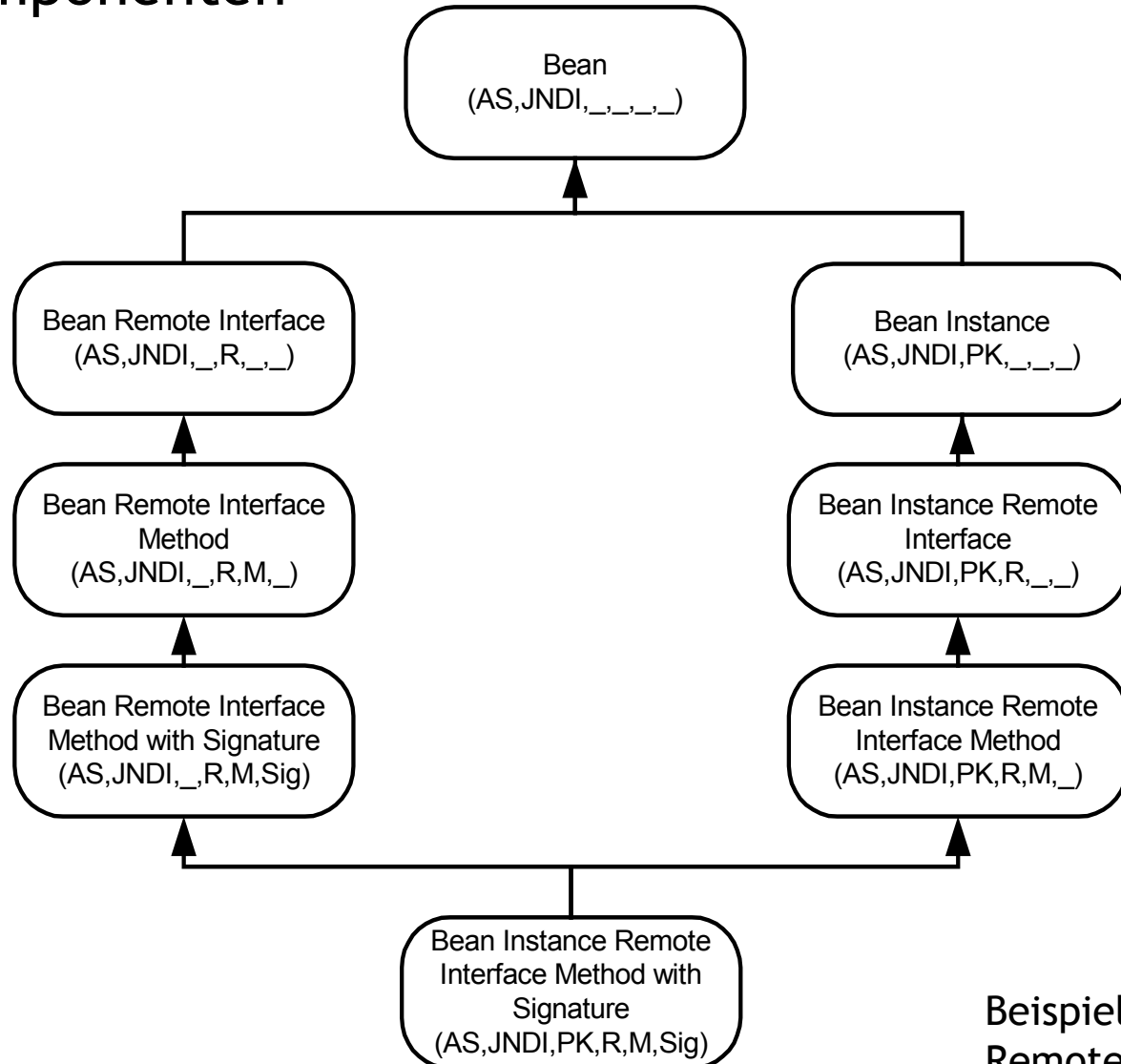


Autorisierung: Zu lösende Probleme

- Problem: Darf ein Benutzer U eine Aktion A auf einer Ressource R jetzt durchführen?
- Benutzer- und Aktions-Identifikation - vergleichsweise trivial
- Identifikation der modellierten Ressource
 - URLs: initiale Segmente des Pfadstrings können modelliert sein
 - URL-Pattern: Suche nach matchenden regulären Ausdrücken notwendig
 - EJB-Komponenten: Identifikation anhand des definierenden 6-Tupels, mehrfache Treffer bei Modellierung als Klassen- und Instanzenressource möglich
- Bestimmung der gültigen Rechte
 - Alle zutreffenden modellierten Ressourcen müssen bestimmt werden
 - Vererbungswege sind zu beachten
 - Constraints müssen ausgewertet werden
- Auflösung von Inkonsistenzen
 - Ordnung auf Rechten erforderlich, derzeit: Verbote überlagern Erlaubnisse (das ist aber nicht wirklich gut)
- Entscheidung der Anfrage



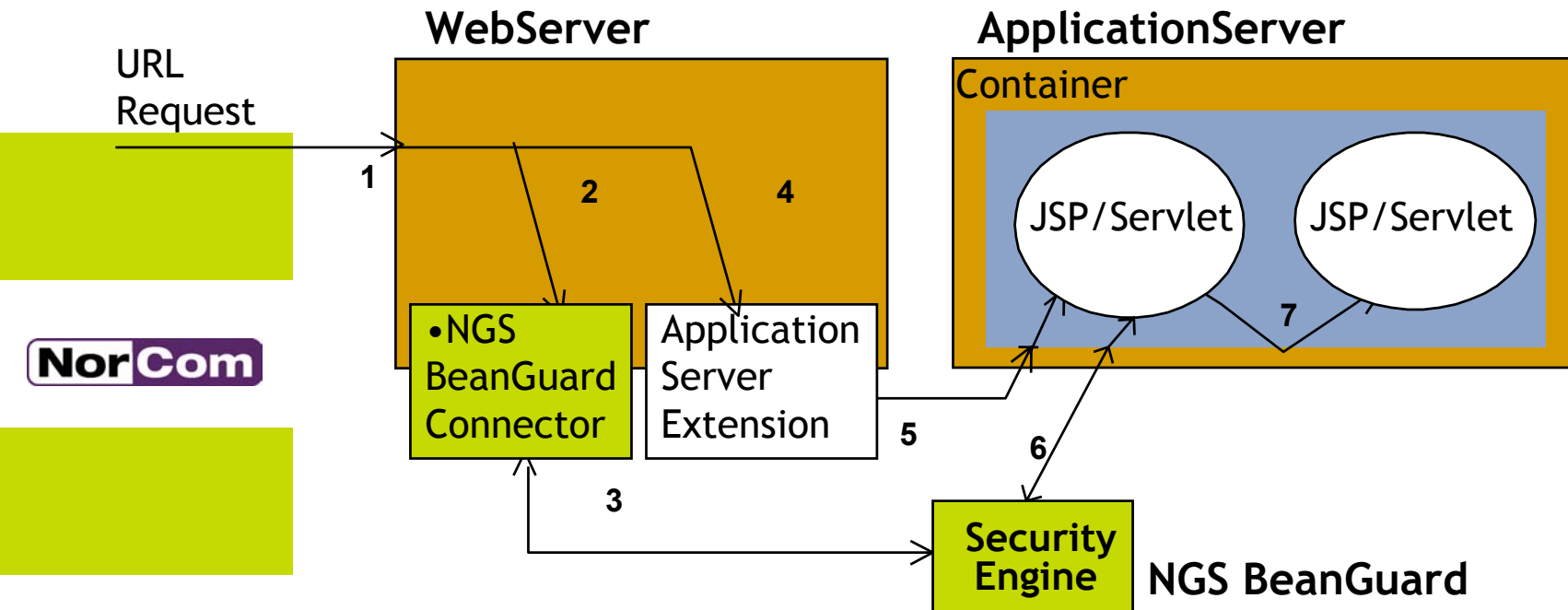
Suche zur Identifikation modellierter Bean-Komponenten





Zusammenwirken mit dem Application Server

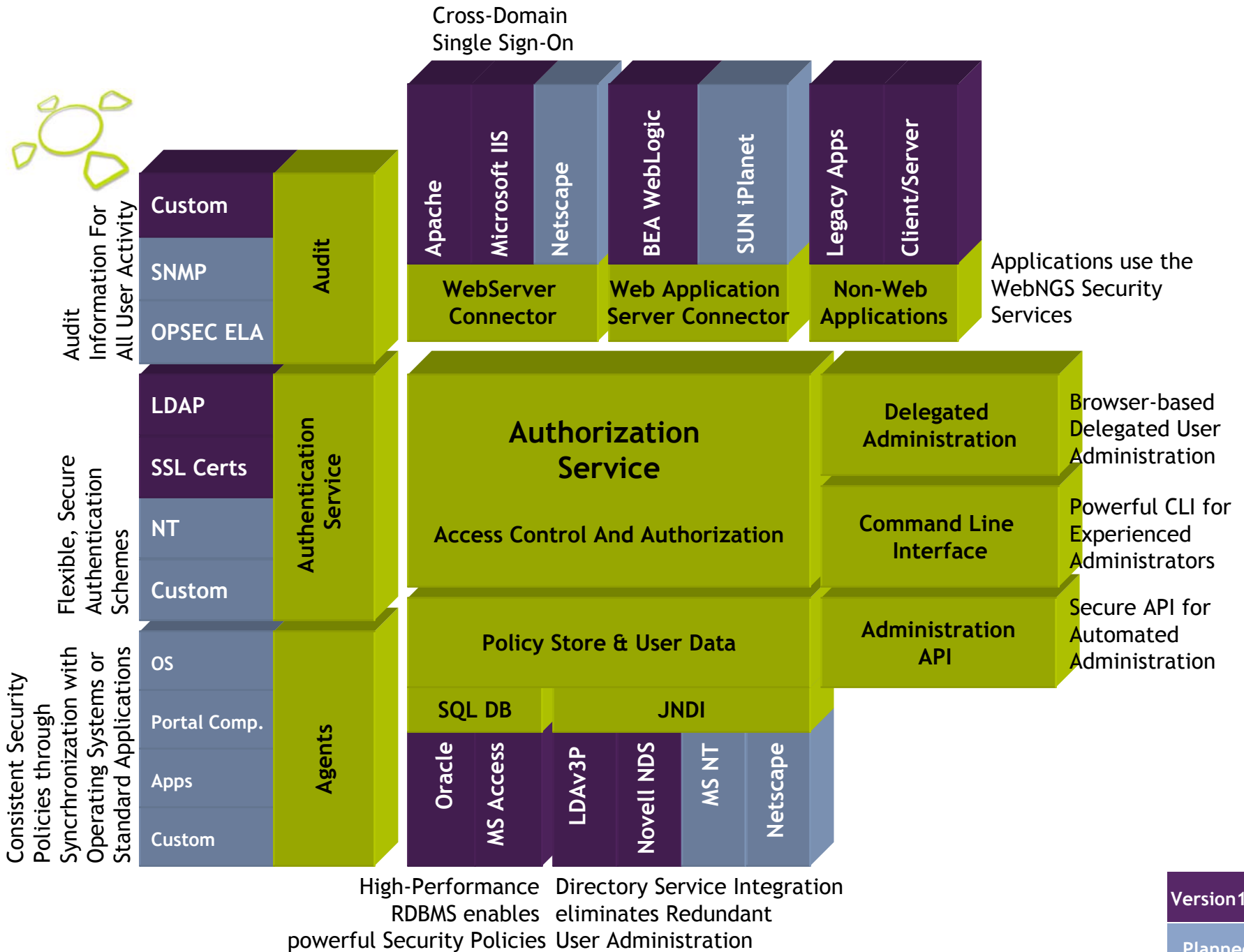
- Im Web Server: Benutzung des Standard-Plug-In-Mechanismus
- Im Application Server:
 - Container-Modifikation (für BEA WebLogic)
 - API-Zugriff über Dispatcher-Bean (JDK 1.3 compatible AS)
 - Java-Connector + Deployment-Descriptor-Generierung (geplant)





Geplante Entwicklungen

Die nächsten Releases



Version 1.0

Planned



Stichworte zur weiteren Produktentwicklung

- Werkzeuge zum XML-Import und -Export
- Reguläre Ausdrücke zur Ressourcenmodellierung
- Feingranulare run-as Möglichkeiten
- Uneingeschränkte DAGs (gerichtete nichtzyklische Graphen) für alle Teile des Butterfly-Modells
- Überlagerung verschiedener Hierarchien im Ressourcengraph (gefärbte Kanten) zur Unterscheidung etwa von Namens- und Vererbungshierarchien
- Programmiermöglichkeiten für
 - Identifizierung modellierter Ressourcen
 - Bestimmung aktuell geltender Rechte
 - Auflösung von Inkonsistenzen
- Neue Authentifikations-Schemata
- Vollständige Realisierung schaltbarer Rollen



Danke für Ihre Aufmerksamkeit

andreas.wolf@norcom.de



NorCom Information Technology AG
Stefan-George-Ring 23
81929 München