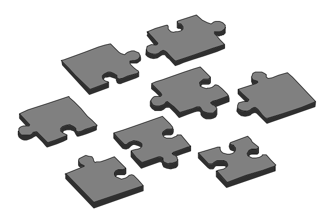
Ein Generisches Management-Modell fuer CORBA, CMIP und SNMP



Dissertationspruefung, 19.11.1997

Struktur des Vortrags

- Motivation
- Ziel der Dissertation
- Ueberblick Forschung
- Vorgehen / Architektur
- Beitrag

Motivation

- Heterogener 'Modell-Mix' im Management
  - Standards: CMIP (TMN), SNMP(v2)
  - Proprietaere APIs
- Neue objektorientierte verteilte Modelle im Netz- und System-Management
  - CORBA, Java, DCOM

➡ Nutzung der neuen OO-Modelle unter Beruecksichtigung der alten

Ziel

- Einheitliches integriertes Management-Modell basierend auf objektorientiertem verteilten Modell
  - CORBA als gemeinsames Modell auf Client-Seite
  - Schwerpunkt auf CORBA, CMIP und SNMP auf Server-Seite

Interesse an Interoperabilitaet

- Integration CORBA, CMIP, TMN, SNMP, Java, DCOM, ODP
- NOMS'98 (Feb. 1998)
  - 4 tutorials (16)
  - 12 technical presentations (84)
  - 2 panels (3)

Ueberblick Forschung (XoJIDM)

1. Statische Transformation von Quellmodell X nach Zielmodell Y (Kompilierzeit)
   - Beispiel: GDMO/ASN.1 (CMIP) --> CORBA IDL
2. Generierter Code sorgt fuer Kommunikation mit Modell X (Laufzeit)
   - Beispiel: Proxymethode A::foo() in Y ruft A::foo() in X auf

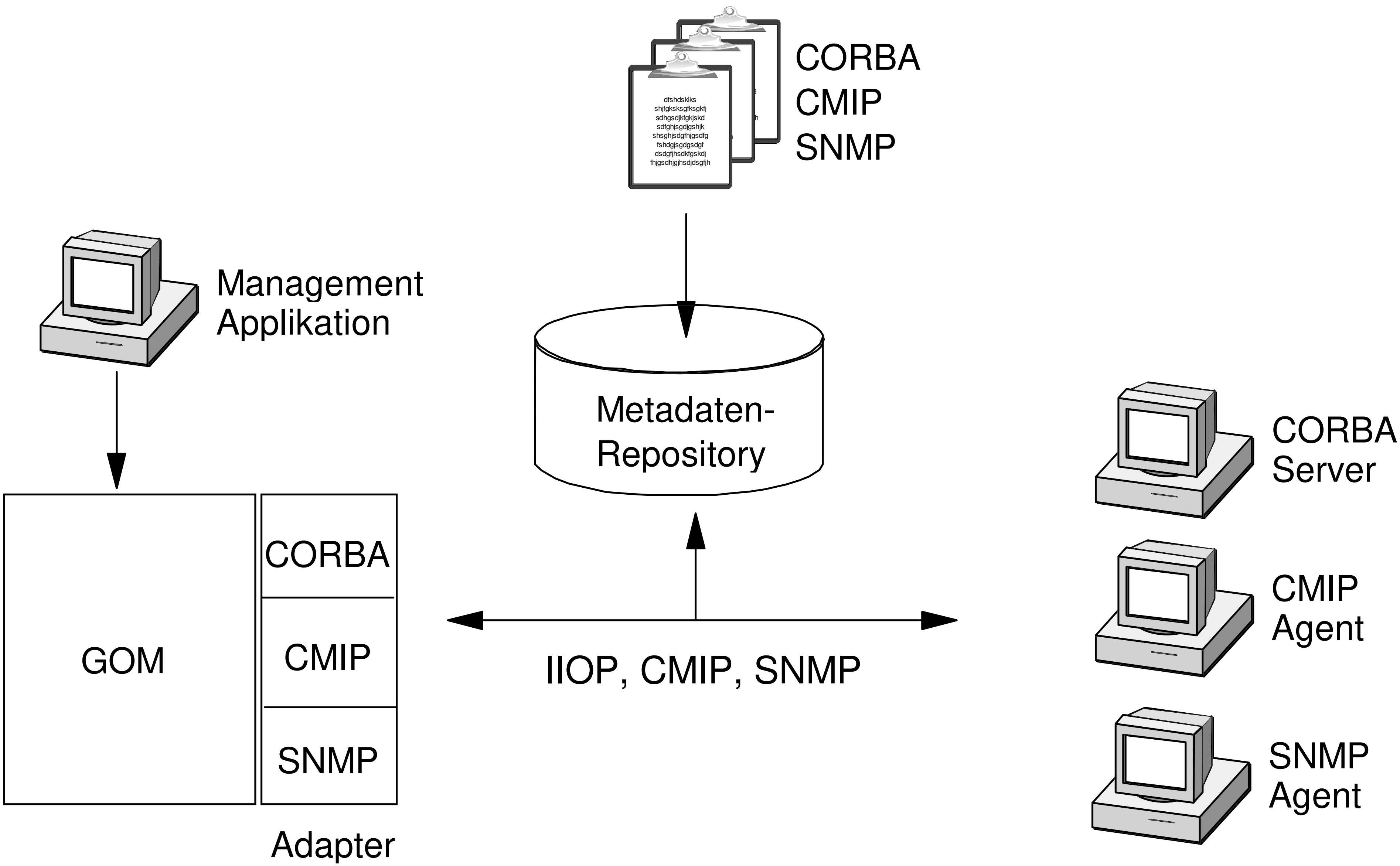
Nachteile

- Grosse Anzahl generierter Klassen und Methoden
- Grosser Speicherverbrauch
- Neugenerierung bei (selbst kleineren) Aenderungen am Modell
- Potentieller Informationsverlust
- Keine Unterstuetzung 'dynamischer' Konstrukte
  - Conditional Packages (CMIP)
  - Extensible Attribute Groups (CMIP)

Eigener Ansatz

1. Transformation eines Modells X in Metadaten X' (Metadaten-Repository)
2. Generisches Objektmodell (GOM) als einheitliche Management-Platform
   - Z.B.: Abbildung aller Klassen auf 1 GOM Klasse
3. Laufzeit-Konvertierung zwischen GOM und X mittels Metadaten X'

Architektur

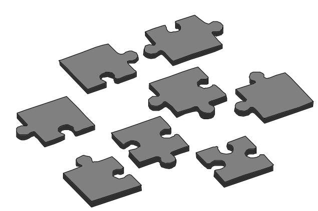


Vorteile

- Einheitliches Management-Modell (wie XoJIDM), aber anderer Ansatz:
- Wahl von dynamischer Programmierung basierend auf Metadaten
  - Geringe Bindung zw. Clients und Servern
  - Flexibilitaet bei Integration anderer Modelle
  - Unterstuetzung dyn. Konstrukte (CMIP)
  - Unterstuetzung von dynamischen Applikationen: Interpreter, Mobile Agenten
  - Kleine, finite Anzahl von Klassen
  - Geringer Speicherverbrauch

Beitrag

- Modell f. ein generisches und einheitliches Management-Modell basierend auf CORBA
- Dynamischer Ansatz: wichtig im Managementbereich (Kap. 5,6)
- Prototyp zur Validierung des Ansatzes



Warum CORBA und nicht Java ?

- Vorteile von Java:
  - Core Reflection API: dynamischer Methodenaufruf
  - Laden von Klassen zur Laufzeit
    - Laden von Adaptern anhand des Namens
    - Nachladen von Klassen ueber das Netz
- Nachteile von Java
  - Proprietaer vs. De-Facto Standard
  - Effizienz (RMI vs. IIOP)
  - Reifegrad (CORBA seit 1992)