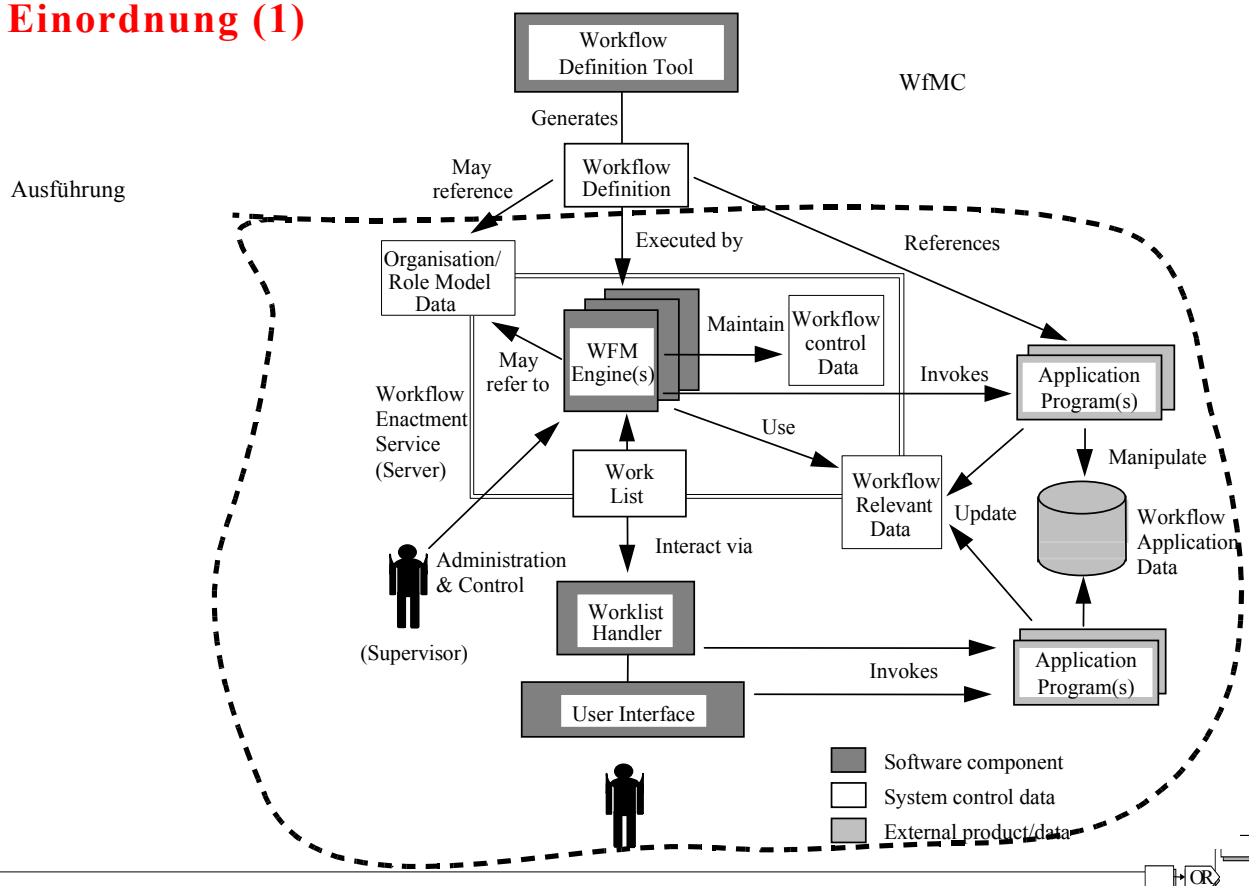
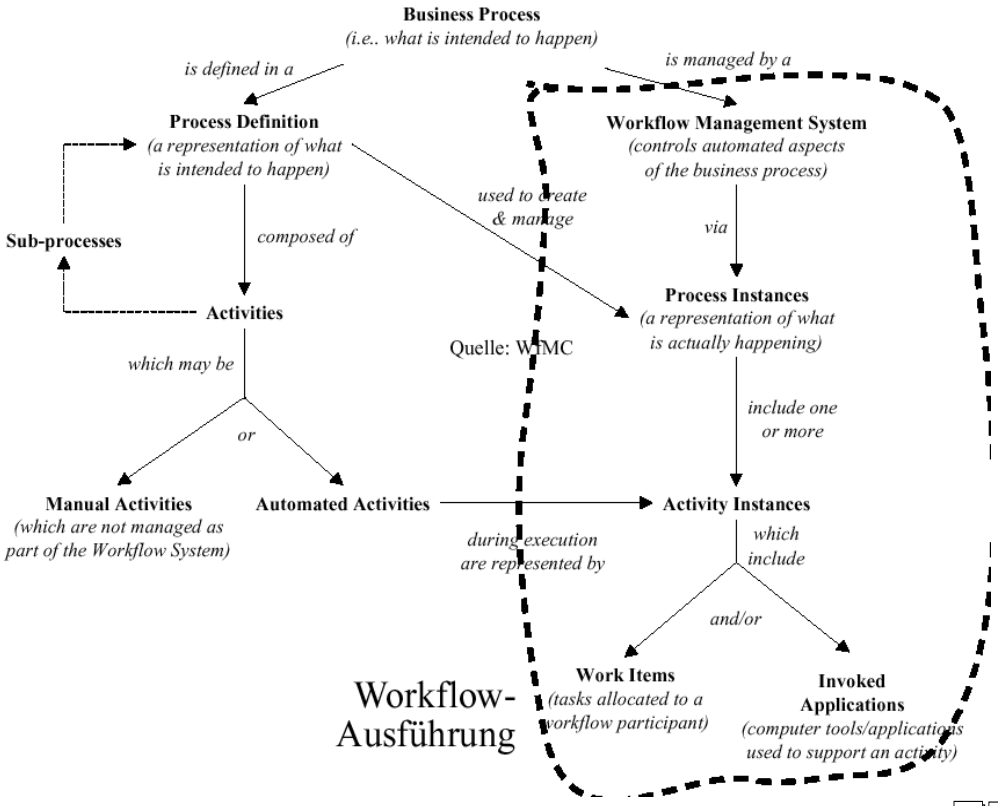
Kapitel 3: Workflow-Ausführung, Workflow-Architekturen und Workflow-Interoperabilität

- Grundaspekte der Workflow-Ausführung
- Architekturtypen für Workflow-Management-Systeme
- Das Ausführungs-Referenzmodell der Workflow Management Coalition
- Die Workflow Facility der Object Management Group
- Architekturmodelle für prozessorientierte E-Service-Systeme
- Bemerkungen
  - Fehlerbehandlung (transaktionale Workflows) → Kap. 4
  - Adaptive Workflows → Kap. 5
  - Dieses Kapitel verwendet Material von P. Dadam, M. Reichert und Th. Bauer (Uni Ulm)

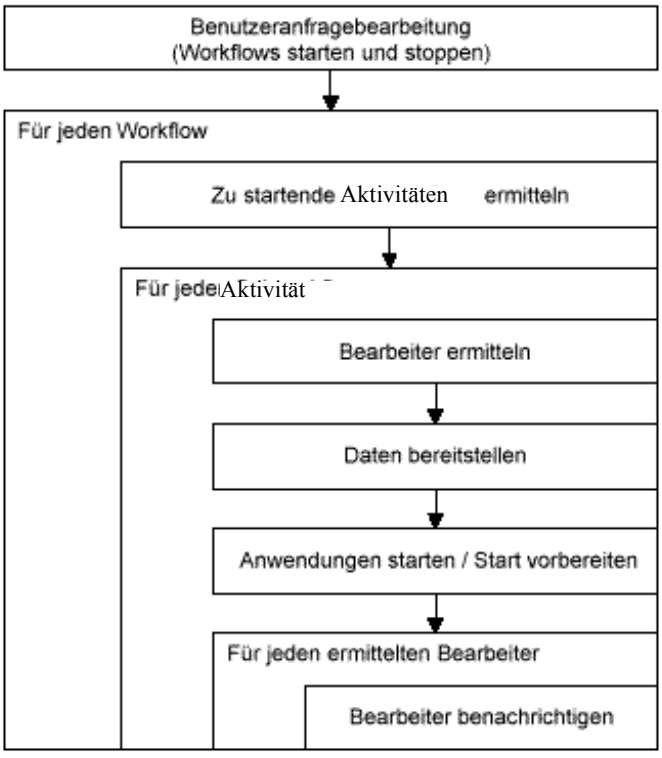
Einordnung (1)



Einordnung (2)

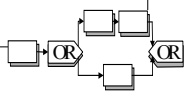


Prinzipieller Ablauf zur Laufzeit



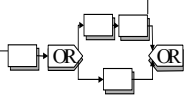
Grundaspekte der Workflow-Ausführung (1)

- Instanziieren und Ausführen von Workflow-Definitionen
  - Wie wird Workflow-Graph intern repräsentiert?
    - als Graph?
    - in Form kompilierter Prozessfragemente
    - In Form von Regeln?
    - als „Objekt“?
- Arbeitet Workflow-Instanz auf Originaldefinition oder auf Kopie?
  - Implikationen?
- Abarbeitung der Kontrollflusses
  - Bestimmung der als nächstes auszuführenden Aktivitäten
  - Evaluierung von Eintritts-, Austritts- und Verzweigungsbedingungen bzgl. der Aktivitäten
  - Aufruf der an die Aktivität gekoppelten Applikationen
  - Eintrag in Arbeitslisten der für die Aktivität zuständigen Mitarbeiter



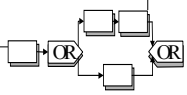
Grundaspekte der Workflow-Ausführung (2)

- Benutzerzuordnung
  - Einmalig bei Instanziierung?
  - Erst bei konkretem Arbeitsschritt?
  - Findet eine „Nachevaluierung“ bei aktivierten Schritten bei Änderungen in der Organisation statt?
    - Neueinstellung
    - Bearbeiter wird krank
    - Vertretungsregel ändert sich
    - ...
- Datenfluss
  - Initialisierung globaler Workflow-Variablen
  - Versorgung der Aktivitäten mit Eingabeparametern
  - Übernahme der Rückgabewerte
  - Intra-Workflow-Datenfluß
  - Datenfluß zu Applikationen und externen Datenquellen
  - Inter-Workflow-Datenfluß



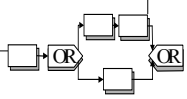
Grundaspekte der Workflow-Ausführung (3)

- Monitoring und Protokollierung
  - Selbstüberwachung (alle angeschlossenen Teilsysteme erreichbar?)
  - Prozessüberwachung (z.B. Deadline-Management)
  - Führen einer Protokolldatei (Wer macht was wann?)
  - Datenschutzproblematik
- Robustheits- und Verfügbarkeitsaspekte
  - Sauberer Wiederanlauf nach System-Crash?
  - Fehlerbehandlung
    - in Anwendungsprogrammierung „hart verdrahtet“?
    - generisch
  - Synchronisierung von Workflow-Instanz mit Anwendungsprogrammen (Legacy-Problematik)
  - Arbeitsfähigkeit von Teilsystemen



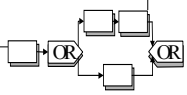
Grundaspekte der Workflow-Ausführung (4)

- Performanzaspekte
  - Performante Server-Implementierung
  - Vermeidung unnötiger Kommunikation
  - Multi-Server-Ausführung
  - Dynamische Lastbalancierung versus „hart-verdrahtete“ Server-Zuordnung



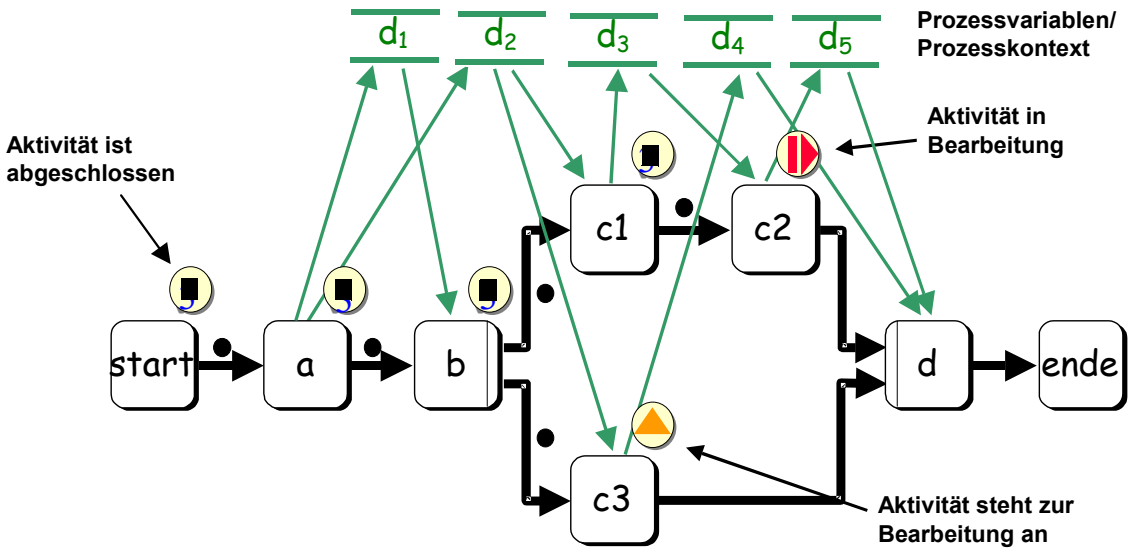
Workflow als interpretierte Graph-Struktur

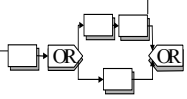
- Übertragung der „externen“ Graph-Repräsentation in äquivalente interne Graph-Darstellung
- „Anreicherung“ des Graphen um Markierungsinformation
- Interpretation des Workflow-Graphen durch entsprechende Ausführungskomponente
- Varianten:
  - Jede Workflow-Instanz referenziert die globale Vorlage **[V1]**
  - Bei der Instanziierung wird eine Kopie der globalen Vorlage erzeugt **[V2]**
  - Mischformen



WF als interpretierte Graph-Struktur (2)

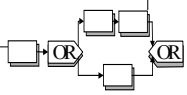
- "Anreicherung" des Graphen um Markierungsinformation und (ggf. um Versionierungsinformation für Prozessvariable)





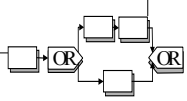
Workflow als interpretierte Graph-Struktur: Bewertung

- Positiv
  - Direkte Analogie (externes – internes Ausführungsmodell)
  - Unterstützung von Schema-Evolution und Ad-hoc-Änderungen (bei V2)
  - Variante V1: Geringer Speicherbedarf
- Negativ
  - Potentiell aufwendigere Ausführung
  - Geringere Effizienz durch Interpreter-Modus?
  - Variante V1: Schema-Evolution und Propagierung auf laufende Workflow-Instanzen kritisch

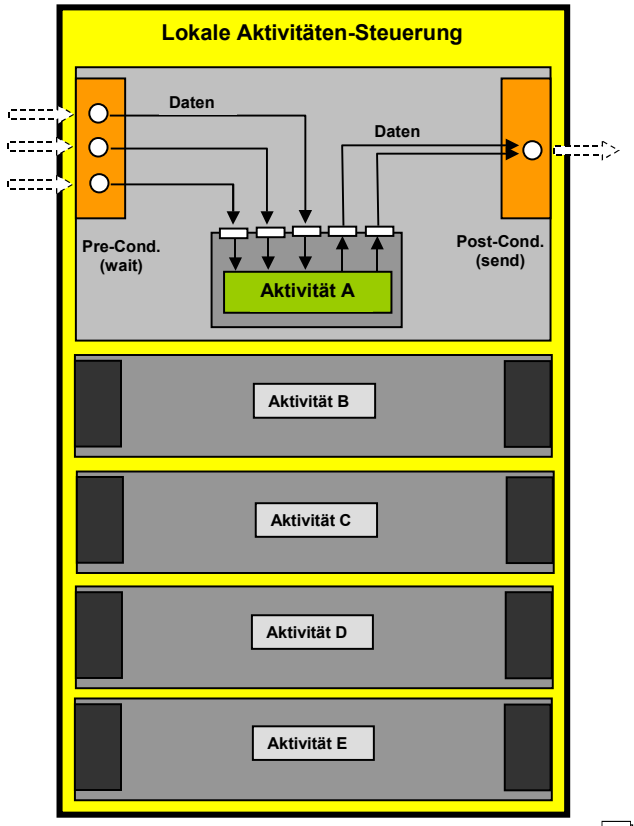


Workflow als kompilierte Prozeßfragmente

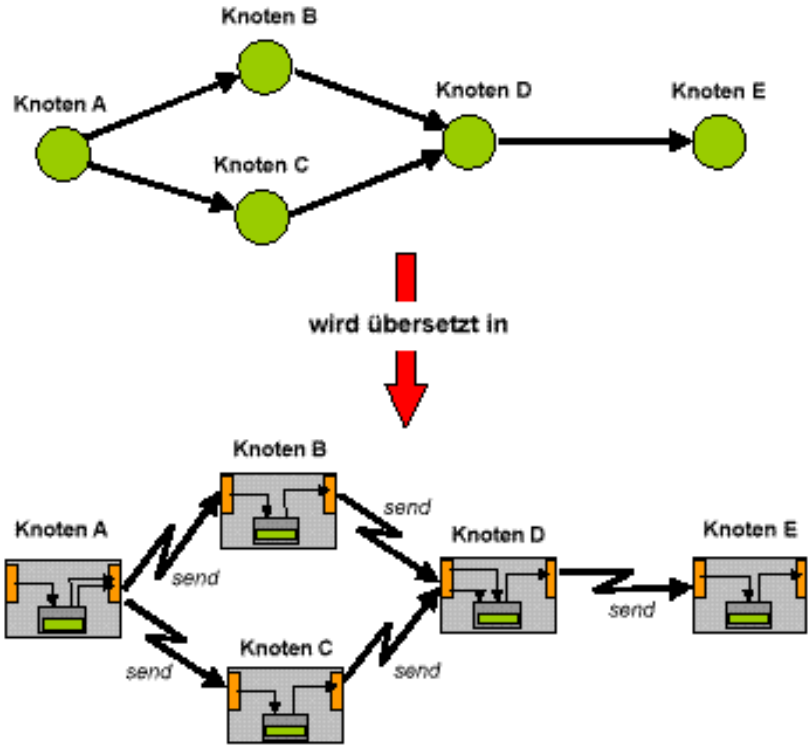
- Grundansatz
  - Übersetzung des Workflow-Graphen in kommunizierende Programme
  - Im Prinzip je Aktivität ein Program
  - Falls mehrere Aktivitäten an einem Netzwerk-Knoten auszuführen, Zusammenfassungen möglich
  - Verteilung der Programme einer Workflow-Vorlage auf die beteiligten Knoten
  - Nachrichtenbasierte, dezentrale Workflow-Steuerung
- Realisierung
  - Jedes Programm ist eingekapselt in Pre- und Post-Conditions
  - Pre-Condition: Nachrichten von anderen Programmen, auf die es warten muss
  - Post-Condition: Die Nachrichten, die es nach Beedingung an andere Programme senden muss
- Vertreter z.B. METEOR, MENTOR



Einkapselung einer Aktivität (Prinzip)

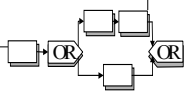


Workflow als kompilierte Prozeßfragmente (2)



Workflow als kompilierte Prozeßfragmente: Bewertung

- Positiv
  - Kommt ohne zentrale Workflow-Server aus
  - hoher Grad an lokaler Autonomie
- Negativ
  - Kein globaler Status (bzw. nur mit Zusatzaufwand ermittelbar)
  - Unterstützung von Ad-hoc-Abweichungen nicht realisierbar



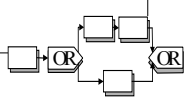
Workflow als Regelmenge (1)

- Darstellung von Kontrollflussbedingungen als (evtl. erweiterte) ECA-Regeln

ON	*Event*
IF	*Condition*
DO	*Action*

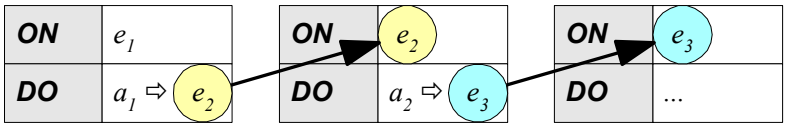
ON	*Event*
IF	*Condition*
Then DO	*Action*
Else DO	*Alternative Action*

- (Erweitertes) DBMS als WfMS
- Kontrollfluss: Aktives DBMS / Triggermechanismen
- Datenfluss: Über Datenbankobjekte
- Vertreter: ATM, INCAs, MOKASSIN

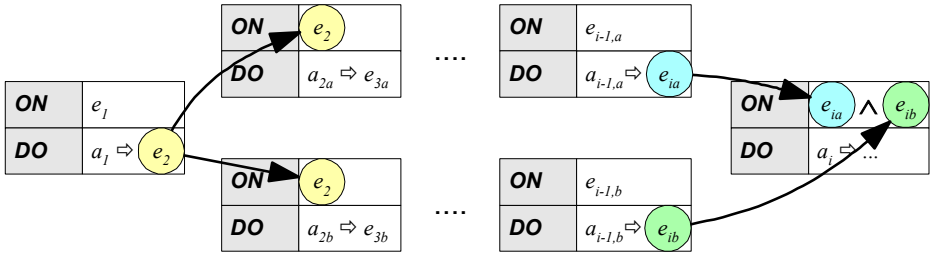


Workflow als Regelmenge (2)

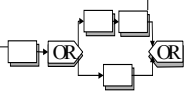
- Sequenz



- Parallel Aktivitäten

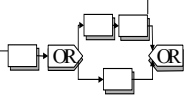


- Verzweigung
  - Erweiterte ECA-Regeln



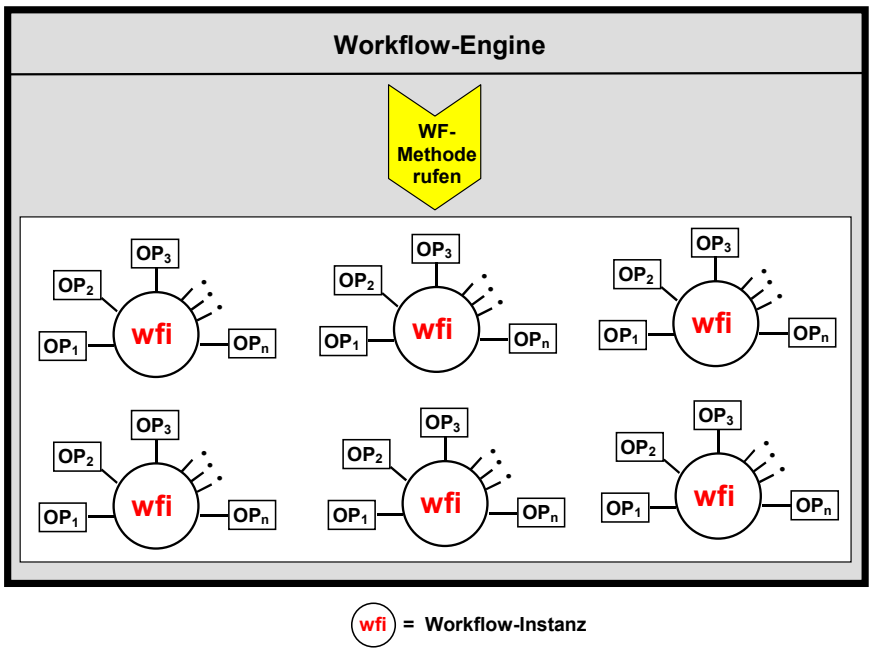
Workflow als Regelmenge: Bewertung

- Positiv
  - Voll datenbankgestütztes Konzept
  - Bei gemeinsamer Datenbank für Anwendung und WfMS relativ einfache Synchronisation von WfMS + Anwendungsdaten
  - Geringer Implementierungsaufwand für WfMS
- Negativ
  - Regelmenge kann sehr groß werden, damit potentielles Performanzproblem (sehr viele Triggeraktivierungen)
  - Globaler Status schwierig zu ermitteln
  - Unterstützung von Ad-hoc-Abweichungen schwer realisierbar



Workflow als migrierende Objekte

- Workflow-Instanz = Objektinstanz (z.B. “intelligentes” Dokument)
- Operationen auf Workflow-Instanz = Objekt-Methodenaufrufe
- Kontrollfluß im Objekt selbst (main-Routine) oder extern
- Objekt „wandert“ von Netz-Knoten zu Netz-Knoten
- Vertreter: ProMInanD

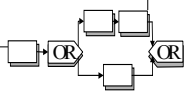


Workflow als migrierende Objekte: Bewertung

- Positiv
  - Übertragung objektorientierter Konzepte auf Workflow-Implementierung
    - Information Hiding
    - Spezialisierung
    - Assoziationen (z.B. zwischen Workflows)
    - Globaler Workflow-Status einfach abrufbar
- Negativ
  - Objekt-Laufzeitsystem beschränkt max. Anzahl aktiver Workflow-Instanzen
  - Unterstützung paralleler Verzweigungen schwierig

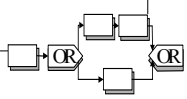
Workflow-Architekturen

- Anforderungen
- Typen

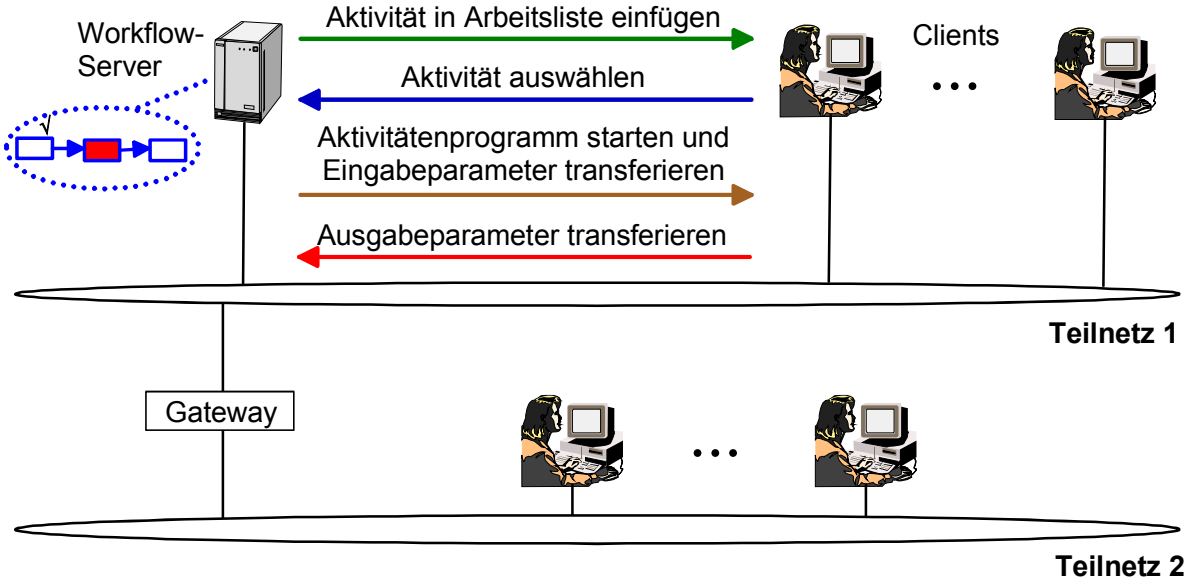


Anforderungen an Workflow-Architekturen

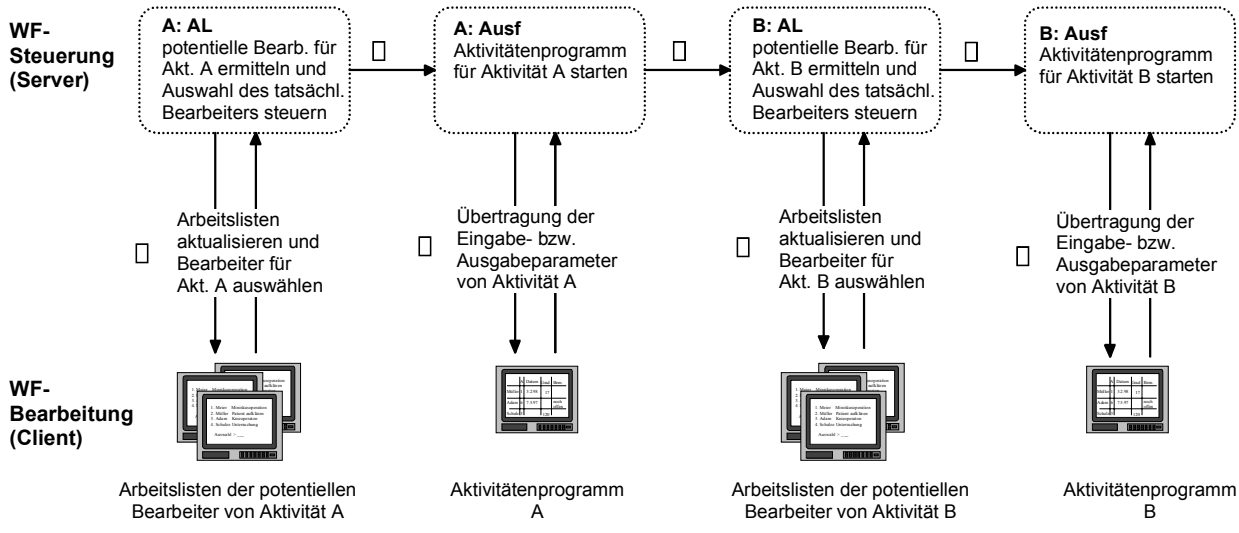
- Korrekte Ausführung der Workflow-Definition
- Reduktion von Kommunikationskosten zwischen Workflow-Server und -Clients
- Beispiel bzgl. Kommunikationslast (Schadensbearbeitung in einer Versicherung):
  - Einscannen von eingehenden Versicherungsdokumenten (z.B. Schadensmeldungen, Rechnungen, Gutachten)
  - Zuleitung der gescannten Dokumente zu 150 Sachbearbeitern (1 MB pro Dokument und Arbeitsschritt)
  - ca. 5 min (= 300 Sekunden) pro Arbeitsschritt
  - Im Mittel 150 / 300 = 0.5 Arbeitsschritte pro Sekunde (d.h. jede Sekunde müssen 0,5 MB bewegt werden)
- Skalierbarkeit
- Minimierung der Folgen bei Ausfall eines Servers
- Ausführungszuverlässigkeit bzgl. technischer und logischer Fehler → Kapitel 4 und 5



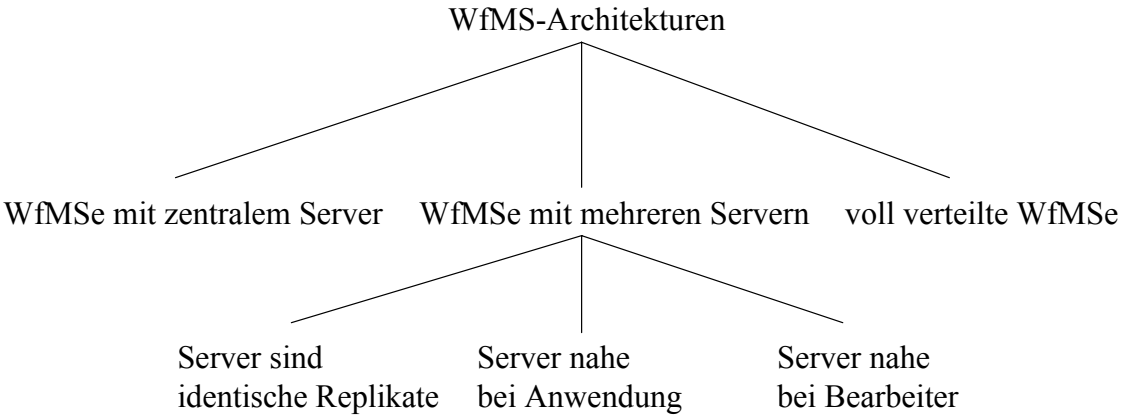
Minimal-Kommunikation beim Ausführen einer Aktivität



Steuerung und Bearbeitung eines Workflows, der aus Aktivitäten A und B besteht

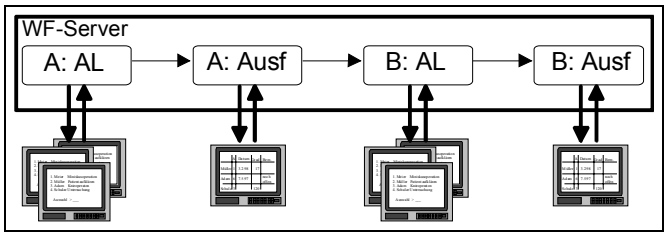


Architekturklassifikation (nach Bauer & Dadam)

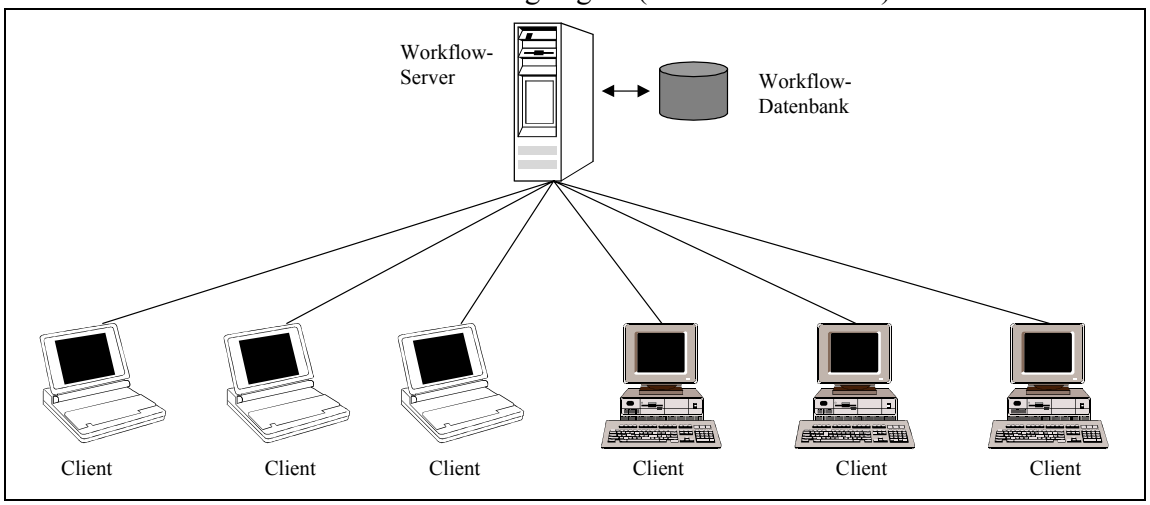


Workflow-Management-Systeme mit zentralem Server

- Ein Workflow-Server (mit einer Workflow-Datenbank)
- „Variante“: Mehrere Workflow-Server mit *einer* Datenbank (DB-Server zentraler Engpaß); konkreter Server wird bei Instanziierung festgelegt

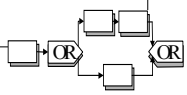


- Nur für relativ kleine Anzahl von Benutzern geeignet (ca. 10-20 Benutzer)



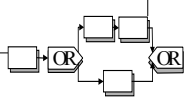
Workflow-Management-Systeme mit zentralem Server: Vertreter

- Großteil der kommerziellen Systeme
- COSA, PROMINAND, IBM FLOWMARK (bis Version 2.1)
- Erweiterung der zentralen Server-Architektur durch verschiedene Hersteller:
  - Mehrere 1-Server Cluster, die miteinander "kooperieren"
  - Kooperation heißt: Gegenseitiger Aufruf von Workflows und Subworkflows
  - *Keine* Abarbeitung einer Workflow-Instanz durch *mehrere* Server
  - Vertreter z.B. FLOWMARK ab Version 2.2



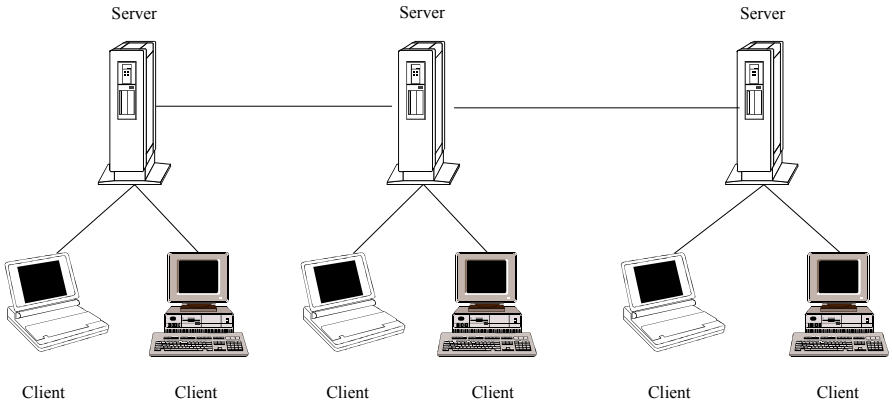
Bewertung

- Positiv
  - Am einfachsten zu implementieren (insb. 1. Variante)
  - Einfacher Überblick über alle laufenden Prozesse
- Negativ
  - Workflow-Server und/oder DB-Server können zum Flaschenhals werden
  - Hoher Kommunikationsaufwand im Teilnetz der Servers



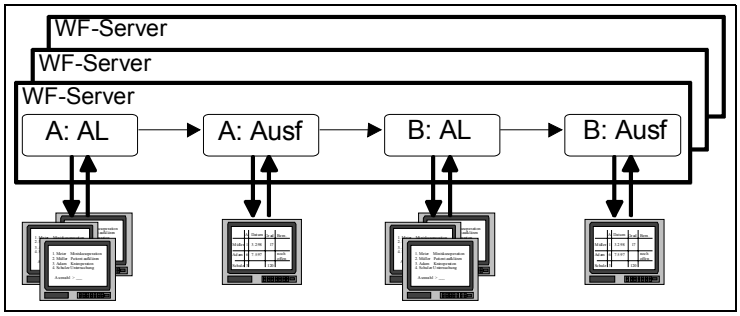
Workflow-Management-Systeme mit mehreren Servern

- Mehrere Workflow-Server auf unterschiedlichen Maschinen (Mehr-Server-Architektur)
- Allokations-Problematik: Welcher Server bearbeitet welche Workflow-Instanz und welche Aktivitäten?
- 3 grundsätzliche Varianten:
  - Server-Replikation (jede Server-Instanz hat gleiche „Kompetenz“)
  - Server nahe bei Anwendung: Server-Lokalisation nahe bestimmter Applikationen
  - Server nahe bei Bearbeiter: Server-Lokalisation nahe bestimmter Benutzergruppen

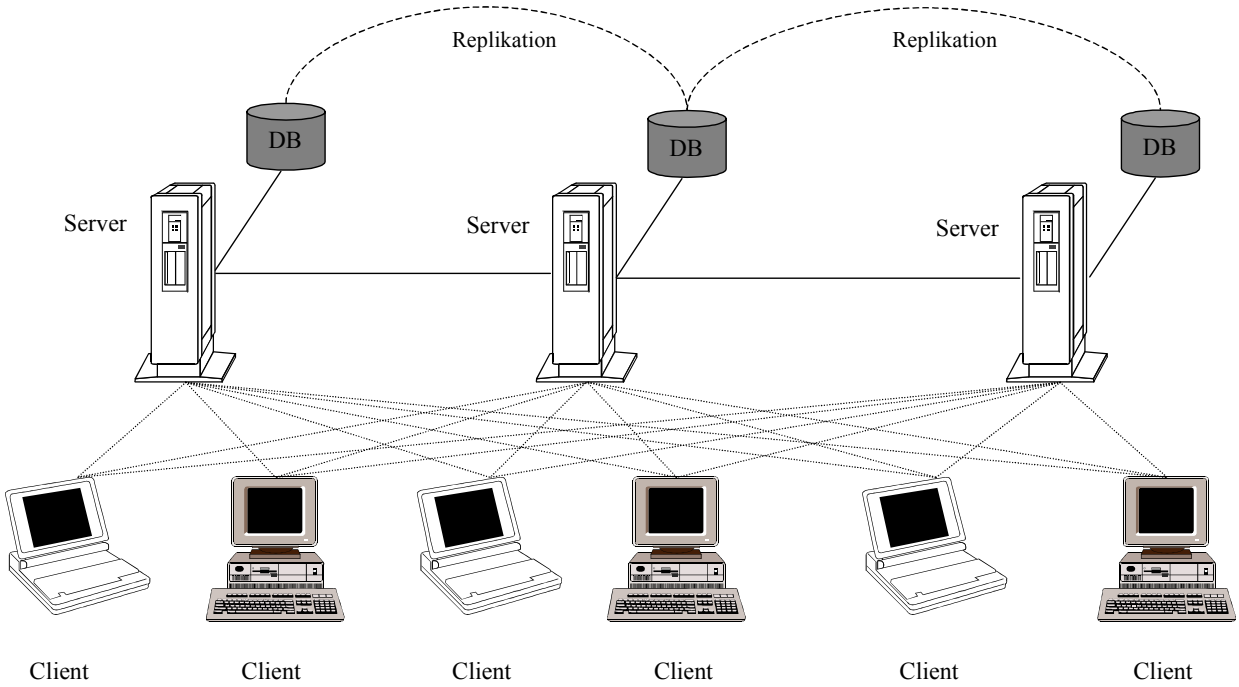


Mehr-Server-Architektur mit identischen Replikaten (1)

- *n* Workflow-Server mit jeweils eigener Workflow-Datenbank auf *n* Rechnern
- Jede Workflow-Datenbank enthält Replikate aller Workflow-Definitionen (→ Konsistenzproblem)
- Jeder Workflow-Server kann Workflows bzgl. *aller* Workflow-Definitionen abarbeiten (Reduzierung der Last pro Server auf $Last_{Gesamt}$ / # *Server)*
- Jeder Client muss zu jedem Workflow-Server eine Verbindung aufbauen können
- Im allg. bearbeitet jeder Client Wf-Instanzen verschiedener Workflow-Server
- Wahl des zuständigen Servers bei Start einer Workflow-Instanz
  - zufällig
  - zyklisch
  - lastabhängig
- Speicherung der Workflow-Instanz-Daten nur einmal bei betroffenem Server
- Bei Ausfall eines Workflow-Servers:
  - Blockierung aller von diesem Workflow-Server verwalteten Workflow-Instanzen
  - Benutzer können eingeschränkt weiterarbeiten (bzgl. der Workflow-Instanzen der nicht ausgefallenen Server)

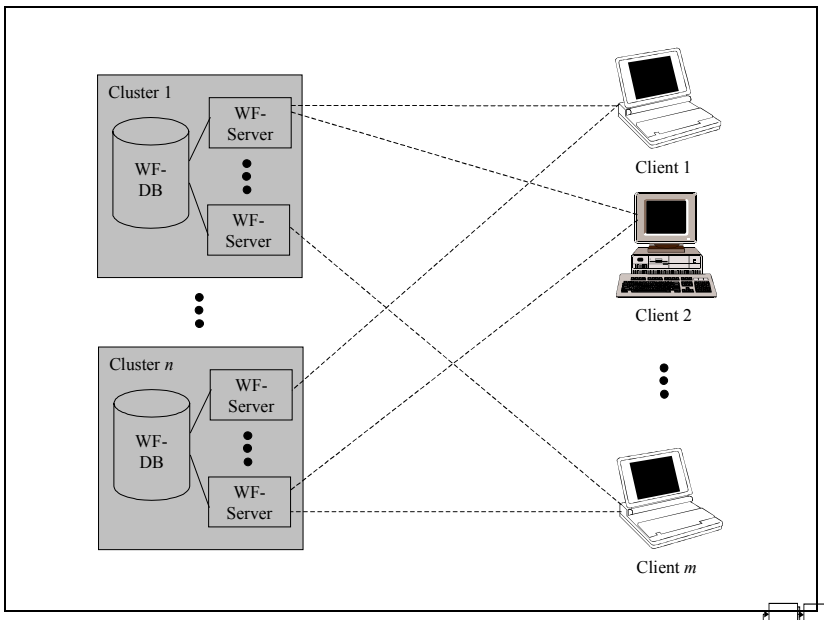


Mehr-Server-Architektur mit identischen Replikaten (2)



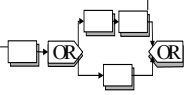
Mehr-Server-Architektur mit identischen Replikaten (3)

- Vertreter u.a.: EXOTICA (IBM ALMADEN RESEARCH CENTER)
- Hochverfügbarkeits-Lösung durch *n* Server-Cluster mit identischer WF-Datenbank
- Jedes Cluster enthält mehrere Workflow-Server
- Jeder Client mit genau einem Server jedes Clusters verbunden
- Bei Ausfall eines Servers Wechsel zu anderem Server des Clusters
- Workflow-Datenbank zentraler Flaschenhals



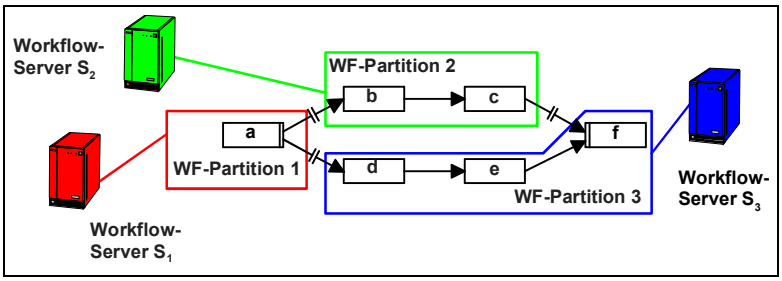
Bewertung

- Positiv
  - Verteilung der Last auf mehrere Workflow- und DB-Server
  - WfMS – in Teilen – auch nach Ausfall eines Workflow-Servers oder einer Workflow-Datenbank noch arbeitsfähig (bzgl. der Workflow-Instanzen der nicht ausgefallenen Server)
  - Bei längerem Ausfall eines Servers Ausführung der neuen Instanzen auf anderen Servern
- Negativ
  - Konsistenzproblem durch Replizierung aller Workflow-Definitionen
  - Jeder Cluster muß potentiell Teile der Arbeitslisten-Einträge aller WfMS-Benutzer verwalten
  - Grund: Verschiedene Workflow-Instanzen eines Bearbeiters können durch verschiedene Cluster kontrolliert werden.
  - Anzahl der zu verwaltenden Arbeitslisten kann also durch die Verwendung mehrer Cluster nicht verringert werden
  - Keine Struktur-Ausnutzung bzgl. Applikationslokalisation oder Organisationseinheiten, um Kommunikationskosten zwischen Server und Clients zu reduzieren
  - Bei ungünstiger Serverwahl aufwendige „Long-Distance“-Kommunikation

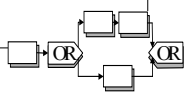


Mehr-Server-Architektur mit Server nahe bei Anwendung

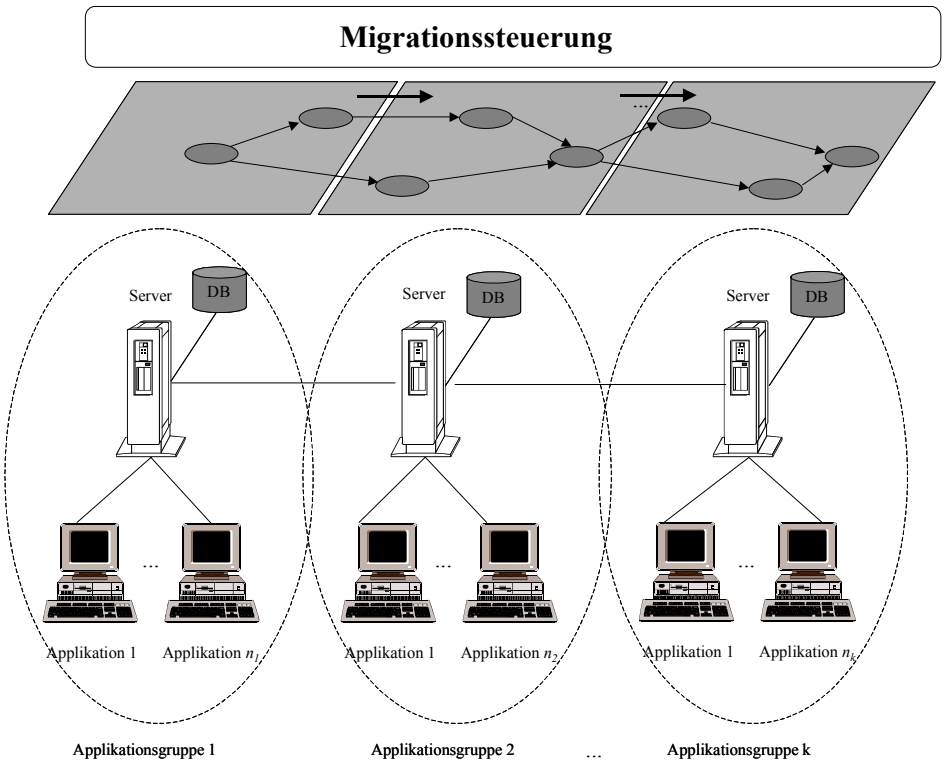
- Problematik bei Server-Replikation:
  - Jeder Server muß Arbeitslisten *aller* Clients bedienen
  - Keine Struktur-Ausnutzung bzgl. Applikationslokalisation oder Organisationseinheiten, um Kommunikationskosten zwischen Server und Clients zu reduzieren
- Platzierung eines Servers in Nähe best. Anwendungen (bzgl. Netz-Topologie)
  - Zuordnung eines Servers zu einer Gruppe von Applikationen mit fester Lokalisation
  - Beispiel: Applikationsgruppe zur Berechnung von Schadenszahlungen bei KFZ-Schäden
  - Server nur zuständig für Aktivitäten im Kontrollfluß, die dieser Applikationsgruppe zugeordnet sind
- Notwendigkeit der *Migration* von Workflow-Instanzen
  - Eine Workflow-Definition referenziert i. allg. Applikationen unterschiedlicher Gruppen
  - Bei Wechsel der Applikationsgruppe muß Workflow-Instanz zu dem für die neue Applikationsgruppe zuständigen Server *migrieren*



  - Migration bedeutet: Übertragen aller Workflow-Instanz-Daten zum neuen Server, Fortführung der Workflow-Instanz durch neuen Server (Zusätzliches Kontrollmodul für Migrationssteuerung nötig)



Mehr-Server-Architektur mit Server nahe bei Anwendung (2)

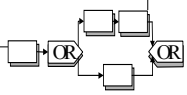


Mehr-Server-Architektur mit Server nahe bei Anwendung (3)

- Problematik: Anwendung muß „feste“ Lokalisation bzgl. ihrer Ausführung haben
  - Oft nur der Fall bei Applikationen mit vielen DB-Zugriffen (→ feste Lokalisation in der Nähe des DB-Servers)
  - Standard-Anwendungen werden oft von Applikations-Server geladen / laufen lokal auf Rechner des Bearbeiters
  - Falls feste Lokalisation wird diese vom Applikations-Server verborgen
  - Ab bestimmter Anzahl von Anwendungen mit nicht-fester Lokalisation zu hohe Kommunikationskosten
- Vorteile
  - Reduktion der Kommunikationskosten zwischen Server und Applikationen
  - Server bedient nur Arbeitslisten derjenigen Benutzer, die an assoziierte Applikationsgruppe gekoppelt sind
- Nachteile
  - Zusätzliche Migrationskosten
  - Bei Server-Ausfall alle Work-flows blockiert, die assoziierte Applikationsgruppe brauchen
  - Kein Zuordnungskriterium für rein manuelle Aktivitäten
- Ob sinnvoll, hängt ab von Möglichkeit der Gruppenbildung bei Applikationen mit fester Lokalisation
  - Anteil von Applikationen mit nicht-fester Lokalisation
  - Partionierungsmöglichkeiten der Workflow-Definitionen bzgl. der Applikationsgruppen
  - Größe der bei Migration zu übertragenden Workflow-Instanz-Daten

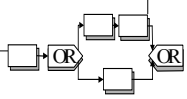
Mehr-Server-Architektur mit Server nahe bei Anwendung (4)

- Vertreter u.a. CODALF, BPAFRAME
- Basieren auf OSF DCE (CODALF) bzw. CORBA (BPAFRAME)
- Kapselung jeder Anwendung in einem DCE- bzw. CORBA-Objekt
- Jedes Objekt hat feste Lokalisation
- Zuordnung der Server zu den Anwendungs-Objekten
- Kombination von Mehr-Server-Architektur mit Server nahe bei Anwendung mit Workflow-Repräsentierung als migrierendes Objekt
  - Jede Workflow-Definition entspricht einem Objekt-Typ, eine Workflow-Instanz ist somit ein Objekt zu dem entsprechenden Objekt-Typ
  - Workflow-Objekt migriert während der Ausführung zu den Lokalisationen der Anwendungsobjekte
  - Verwendung des Traders (“Gelbe Seiten” der Middleware-Schicht)
- Zwecks Lastverteilung Replikation der Anwendungs-Objekte

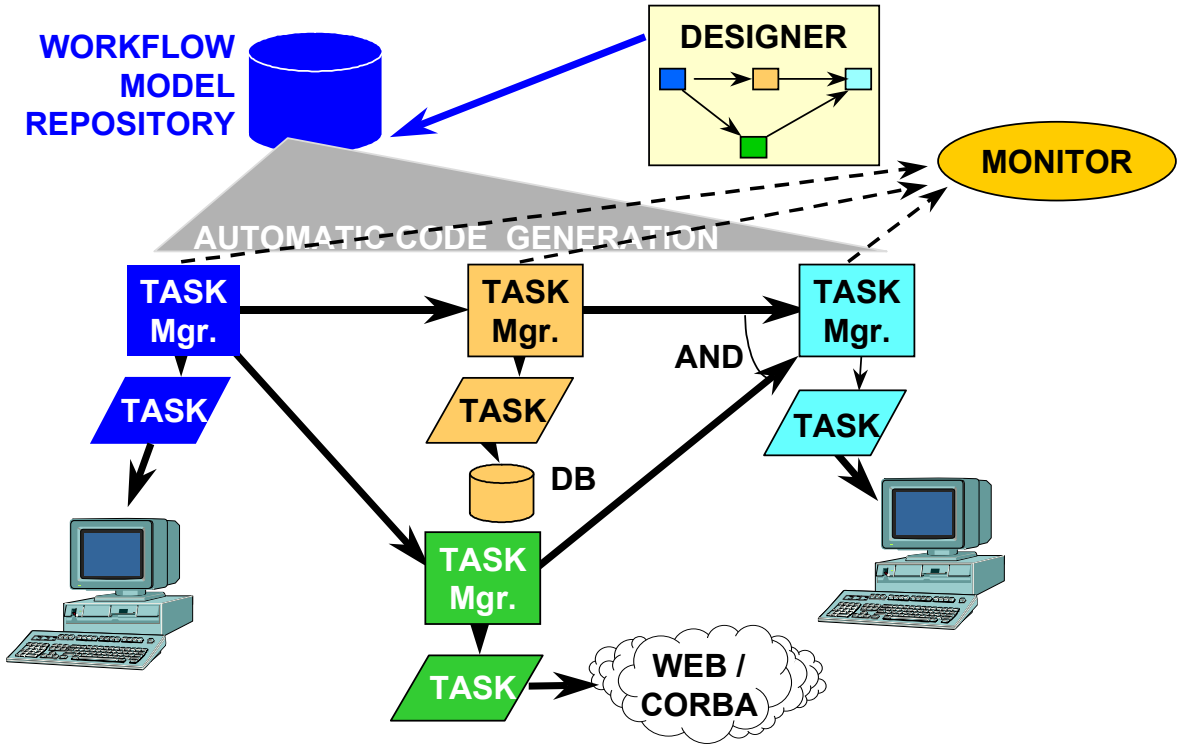


Mehr-Server-Architektur mit Server nahe bei Anwendung (5)

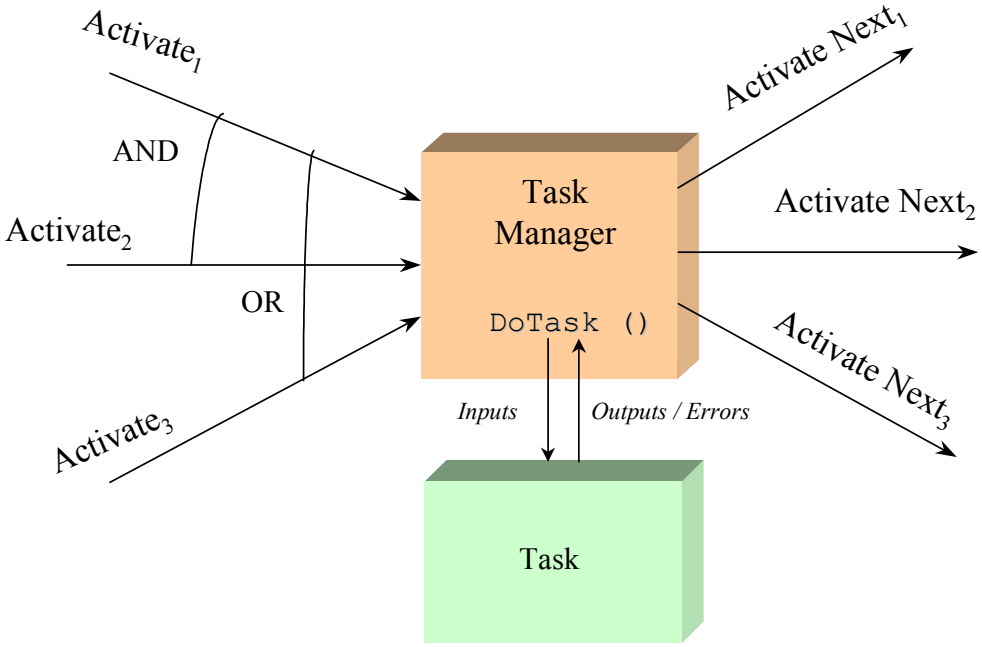
- Weiterer Vertreter: $Meteor_2$ (University of Georgia, Sheth et al.)
- Generierung eines sogenannten Task-Managers (= Servers) für jeden Aktivitäts-Typ bei Compilierung der Workflow-Definitionen
- Ein Task-Manager steuert alle Schritte die, die mit dem assoziierten Aktivitäts-Typ verbunden sind
- Allokation des Task-Managers am Ort der assoziierten Anwendung
- Keine Migration der Workflow-Instanzen, sondern gegenseitiges Aufrufen der Task-Manager (durch vom Compiler erzeugte Aufrufe)
- Also Erzeugung von workflow-spezifischen “light” Application (= Task) Servern
- Keine dynamische Lastbalancierung



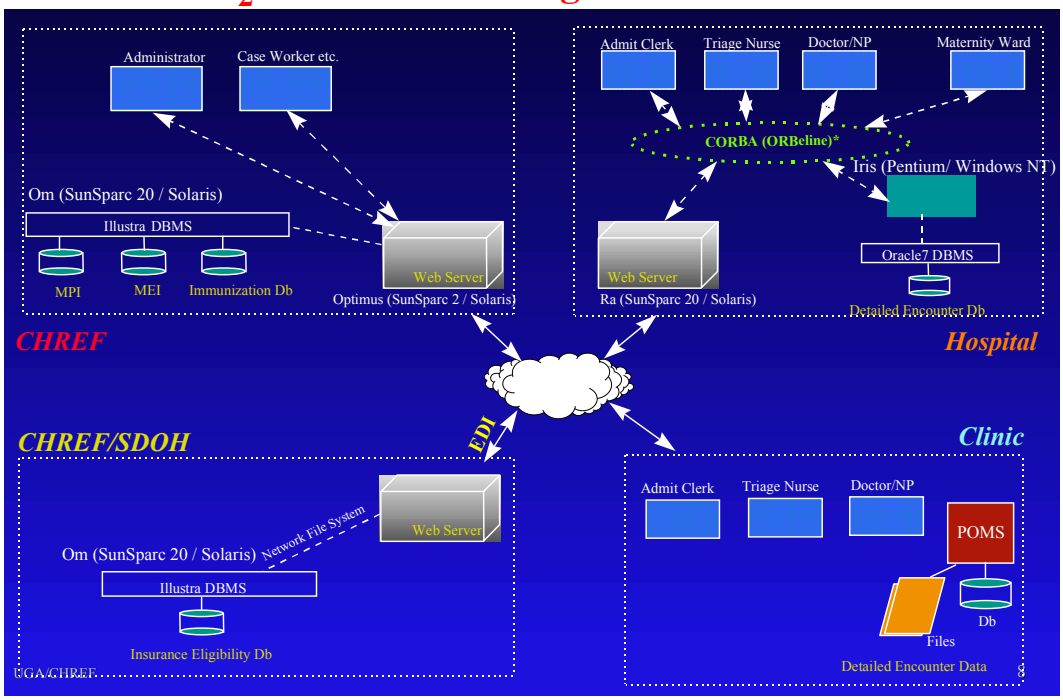
METEOR$_2$ (University of Georgia, Sheth et al.)



METEOR$_2$: Task Manager



METEOR$_2$: Immunisierung in Connecticut



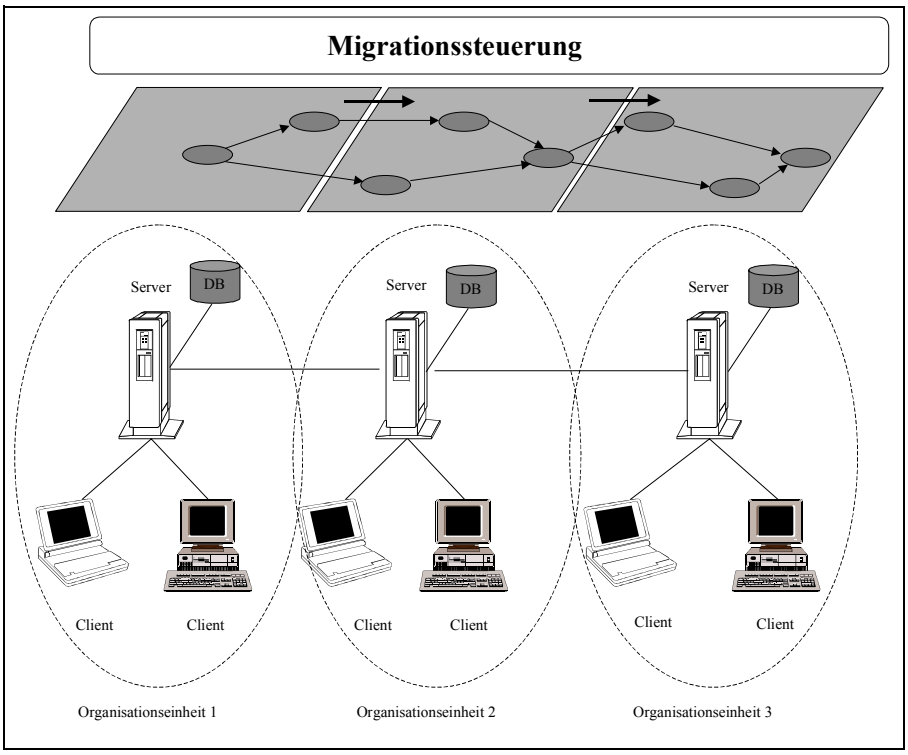
CHREF: Connecticut Healthcare Research and Education Foundation
SDOH: State Department of Health
POMS: Practice Office Management System

Mehr-Server-Architektur mit Server nahe bei Bearbeiter (1)

- Problematik bei Server nahe bei Anwendung: Voraussetzung, daß Anwendungen „feste“ Lokalisation haben, ist oft nicht ausreichend gegeben
- Beobachtung:
  - Oft werden bestimmte Aktivitäten ausschließlich von *einer* bestimmten Organisationseinheit durchgeführt
  - Beispiel *Bearbeitung eines Kundenauftrages*: Auftragsannahme durch Verkaufsabteilung, Verschicken des Produkts durch Versandabteilung, Rechnungsstellung durch Rechnungsabteilung
  - Eine Workflow-Definition kann somit oft einer bestimmten Organisationseinheit zugeordnet werden oder disjunkt bzgl. mehrerer Organisationseinheiten logisch partitioniert werden.
- Konsequenz für Architektur:
  - Verteilung der Server auf die verschiedenen Organisationseinheiten
  - ein Server steuert nur die für seine Organisationseinheit relevanten Workflow-Partitionen und die damit verbundene Aktivitäten

Mehr-Server-Architektur mit Server nahe bei Bearbeiter (2)

- Notwendigkeit der Migration von Workflow-Instanzen (wie bei Server nahe bei Anwendung)
  - Falls Workflow-Definition bzgl. *mehrerer* Organisationseinheiten partitioniert ist: Beim Verlassen einer Partition muß Workflow-Instanz zum für die neue Partition zuständigen Server migrieren
- Falls 1:1 - Abbildung zwischen Organisationseinheiten und Applikationsgruppen: Identität der Architekturtypen Server nahe bei Anwendung und Server nahe bei Bearbeiter



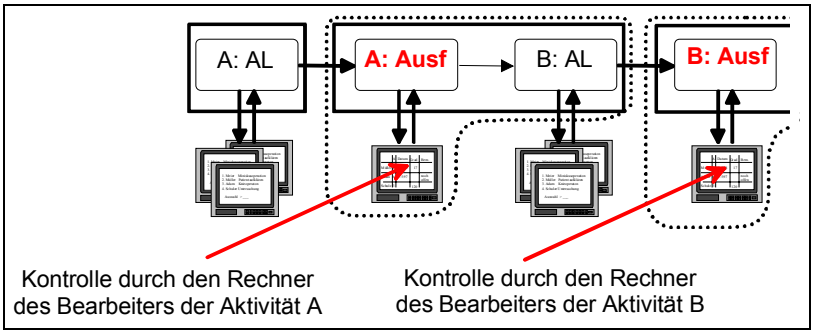
Mehr-Server-Architektur mit Server nahe bei Bearbeiter (3)

- Vorteile
  - Server muß Arbeitslisten nur für Clients der eigenen Organisationseinheit verwalten
  - Reduktion der Datenübertragung, da Übertragung der Daten von Server ↔ Client sich weitgehend auf Netz der Organisationseinheit beschränkt
- Nachteile
  - Zusätzliche Migrationskosten
  - Bei Ausfall eines Server alle Workflows blockiert, die Aktivitäten in betroffener Organisationseinheit durchführen
- Ob Architektur nahe bei Bearbeiter sinnvoll, hängt ab von
  - Partitionierungsstruktur der Workflows
  - Größe der bei Migration zu übertragenden Workflow-Instanz-Daten
  - Anzahl und Größe der Organisationseinheiten
- Vertreter: MOBILE, MENTOR, WIDE

Voll verteilte Workflow-Management-Systeme

- Problematik bei Server-basierten Architekturen: Server und Datenverbindungen zwischen Server und Client stellen Engpässe des Systems dar

- Voll verteilte Workflow-Management-Systeme:
  - Verzicht auf Server
  - „Compiler“ generiert auf der Basis einer Workflow-Definition eine Anzahl von Client-Programmen inkl. der assoziierten Kommunikationsprozeduren zwischen den Clients



  - Engine-Logik im „Compiler“ implementiert
  - Clients rufen sich zur Laufzeit gegenseitig auf und führen den Workflow aus (... *Client 1 führt Aktivität 1 aus, ruft Client 2 auf, der Aktivität 2 ausführt, übergibt an diesen die Kontrolle* ... usw.)
  - Jeder Arbeitsplatzrechner fungiert als "Mini-WF-Server" für seine lokalen Anwendungen

- Systeme mit und ohne Rollenauflösung ("hart verdrahtete" versus dynamische Bearbeiterzuordnung)

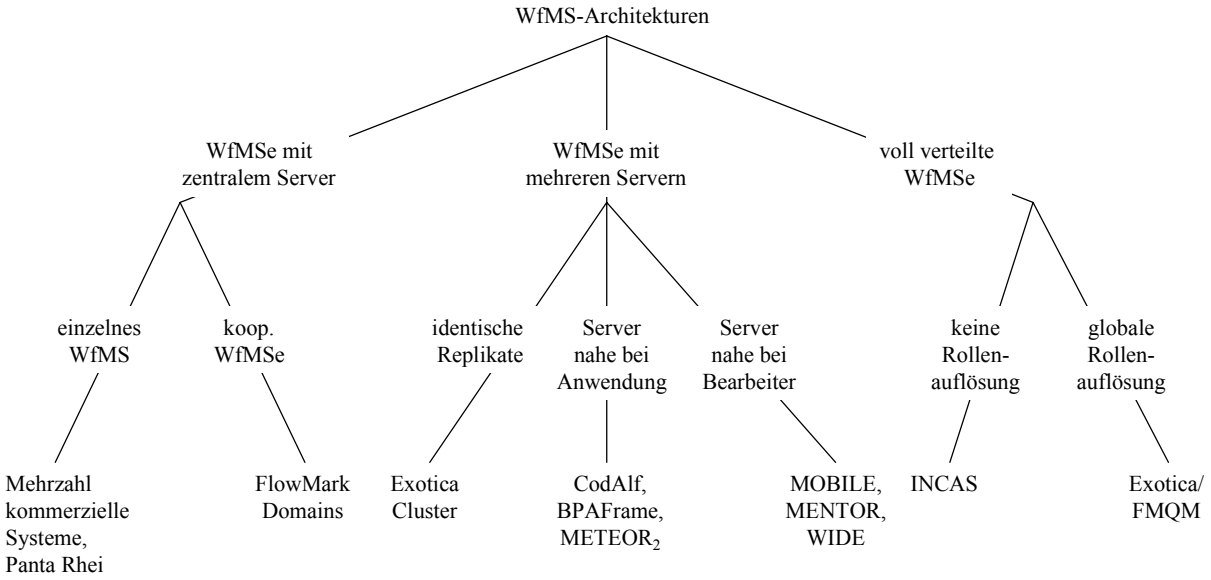
- Vorteil: Vermeidung von Engpässen, Last verteilt sich auf viele (kleine) Rechner

- Nachteile
  - Zersplitterung einer Workflow-Instanz in einzelne Programme
  - Gesamt-Zustand einer Workflow-Instanz nur mühsam zu bestimmen
  - Transaktionsschutz einer Workflow-Instanz nur sehr eingeschränkt möglich (im Prinzip nur auf Aktivitäts-Ebene) → Kap. 4
  - Dynamische Adaptionen einer Workflow-Instanz nur sehr eingeschränkt möglich → Kap. 5

- Vermeidung der Nachteile durch zusätzliche Monitoring-Komponente

- Systeme mit und ohne Rollenauflösung ("hart verdrahtete" versus dynamische Bearbeiterzuordnung)

Architekturklassifikation (Systemübersicht)



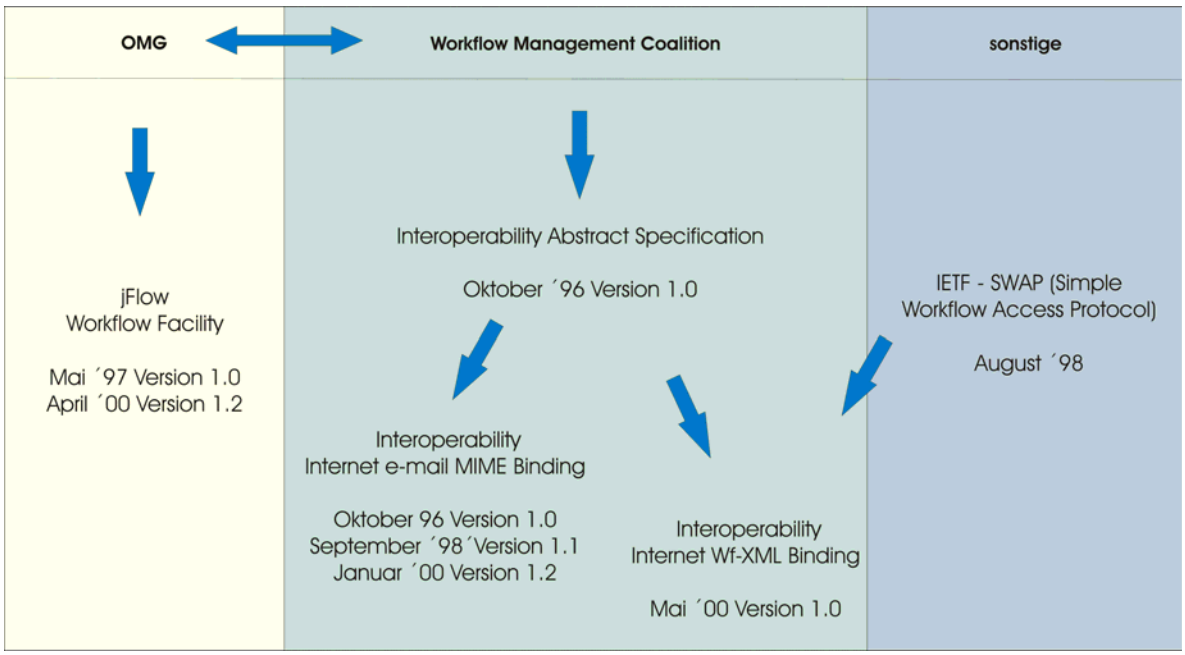
Arbeitslistenverwaltung

- **Poll-Implementierung**
  - **Vorgehen:**
    Server ist passiv, Client muss sich selbst um Aktualisierung kümmern
  - **Positiv:**
    - sehr niedrige Kommunikationslast erreichbar
    - Client entscheidet selbst, wann aktualisiert werden soll
  - **Probleme:**
    - Client fragt selten beim Server nach:
      - \+ geringe Kommunikationslast
      - – veraltete Arbeitslisten
    - Client fragt sehr häufig beim Server nach:
      - \+ relativ aktuelle Arbeitslisten
      - – u.U. hoher Kommunikationsaufwand ohne Nutzen (da keine Aktualisierung erforderlich)
  - **Verbesserung: Zeitgesteuertes Poll**
    - Client fragt in bestimmten Zeitintervallen an
    - Zeitintervalle werden nach jeweiligem Bedarf individuell gesetzt
  - **aber:**
    - Problem der unnützen Kommunikation bleibt
    - Wichtige Aktivitäten und Ereignisse (z.B. Terminüberschreitung) werden u.U. zu spät angezeigt

- **Push-Implementierung**
  - **Vorgehen:**
    Server ist aktiv, bei jeder Statusänderung informiert er alle betroffenen Clients
  - **Positiv:**
    - alle Arbeitslisten stets aktuell (wenn Server immer sofort aktualisiert)
    - es fällt nur Kommunikation an, wenn auch tatsächlich etwas zu aktualisieren ist
  - **Problem:**
    - es wird (potentiell) sehr viel kommuniziert
  - **Abhilfe: zeitgesteuertes Push**
    - Client kann Aktualisierungs-Intervall vorgeben
    - Kommunikation wird nur durchgeführt, wenn eine Veränderung eingetreten ist
    - Prioritäts-Nachrichten werden sofort weitergeleitet

Interoperabilität zwischen WfMS

- Einfache Interoperabilitätsstandards mit unterschiedlichen Nachrichtenformaten und/oder Übertragungsprotokollen



Workflow Management Coalition (WfMC)

- http://www.aiim.org/wfmc/mainframe.htm
- 1993 gegründetes Konsortium von Herstellern und Anwendern mit ca. 130 Mitglieder, u.a. HP, IBM, MS, SAP, University of Georgia, Universität Stuttgart
- Forschungsgruppen relativ schwach vertreten
- Spezifikation von Referenzmodellen u.a. für Prozeßdefinitionen und Workflow-Ausführung
- Pragmatischer, auf standardisierter API-Beschreibung basierender Ansatz zur technischen Interoperabilität zwischen Komponenten eines WfMS sowie zwischen verschiedenen WfMS
- Ziel: Beteiligte Hersteller bieten ihre Systeme mit WfMC-APIs an
  - Wiederverwendbarkeit von Prozeßdefinitionen
  - verbesserte Integrationsfähigkeit von Workflow-Komponenten
  - verbesserte Kommunikation mit Applikationen

WfMC: Interoperabilitäts-Interfaces

Schnittstelle 1 (Interoperabilität zwischen WfMS und Prozess-Modellierungswerkzeugen):

- Workflow-Metamodell
- Standardformat für Austausch von Prozessmodellen zwischen Modellierungswerkzeugen und WfMS

Schnittstelle 2+3 (Interoperabilität zwischen WfMS und Client-Programmen bzw. Aktivitätenprogrammen)

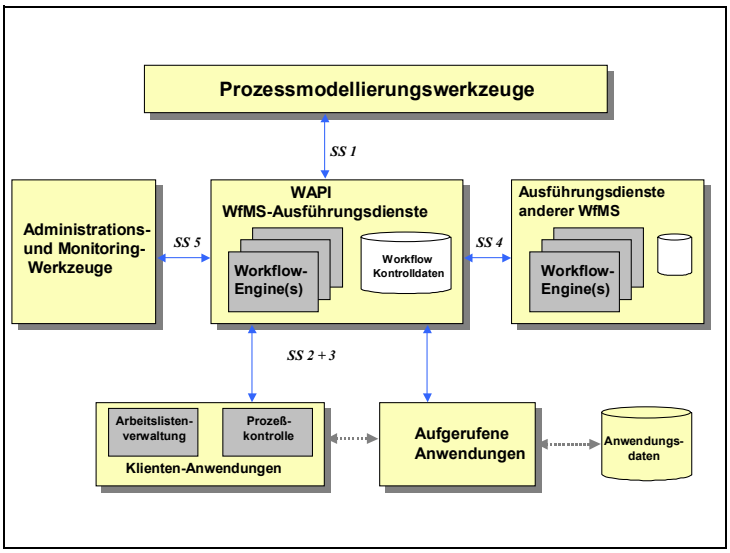
- Standard-API für die Implementierung von Klientenprogrammen (z.B. zur Arbeitslistenanzeige und -verwaltung)
- API für die Parameterversorgung und -übernahme bei Aufruf von Aktivitätenprogrammen

Schnittstelle 4 (Interoperabilität zwischen WfMS verschiedener Hersteller)

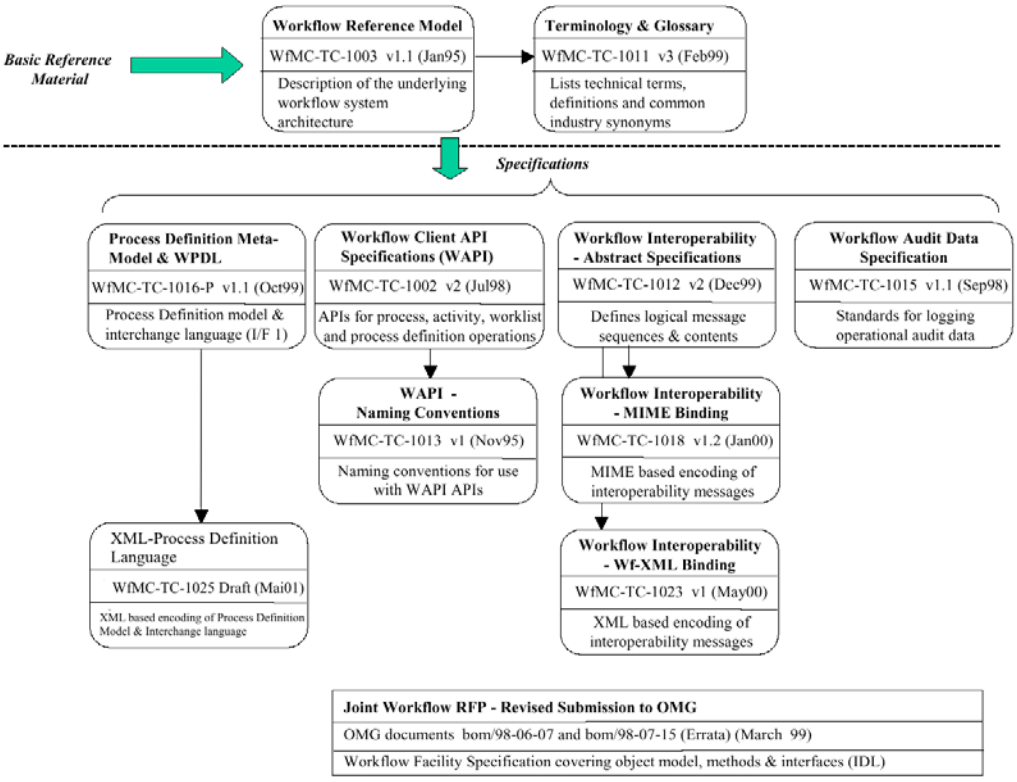
- standardisiertes Nachrichtenformat
- verschiedene Bindings (E-Mail-MIME, WF-XML)

Schnittstelle 5: (Interoperabilität zwischen WfMS und Monitoring-Werkzeugen)

- Standard-Format für Audit-Daten



WfMC: Bisher veröffentlichte Standarddokumente



WfMC: Auszug aus Interface 4: *Engine* ↔ *Engine*-Interoperabilität

Specification WMAReturnCode WMRequestGetProcessInstanceState (
in engine_identifier WMAEngineID,
in process WMAObjectID,
out stateWMAObjectState
);

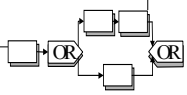
Description Get the current status of a given process instance which is being enacted by another workflow engine

Parameters engine_identifier: identifies the target workflow engine
process: the id of the process instance that is being enacted
state: the returned state - a string prefixed by one of:

open.not-running // created but not yet started
open.running
closed.completed
closed.aborted

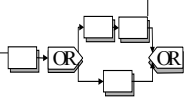
Return Values Success | Failure | Operation_not_performed | Operation_not_implemented

Rationale For long term sub-processes with a life beyond that of the session during which they were created, it is important that the invoking workflow engine be able to check as necessary that the invoked sub-process is alive and well or has completed.



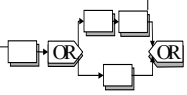
Workflow Management Coalition (WfMC): Diskussion

- Als Kompromißlösung und kleinster gemeinsamer Nenner bzgl. einer Vielzahl von Ansätzen nur sehr grob-granuläre Spezifikation
- Sehr einfaches Meta-Modell bzgl. Definitions- und Ausführungszeit
- Unzureichende Berücksichtigung von Mehr-Server-Architekturen (mit Prozeß-Migration)
- Keine Unterstützung bzgl. Fehlerbehandlung
- Unklarheit bzgl. tatsächlicher WfMC-Unterstützung der WfMS- *und* Applikations-Hersteller



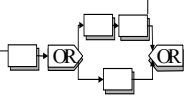
Die OMG Workflow Facility: Globale Zielsetzungen

- Bereitstellung eines Corba-orientierten Ausführungs- und Architekturmodells
- Definition eines IDL-Schemas mit Interfaces zur „Repräsentierung“ von Workflow-Definitionen, Workflow- und Aktivitäts-Instanzen, Workflow Engines etc.
- Zielvorstellung: Hersteller von Workflow-Management-Systemen „exportieren“ ihre Workflow-Definitionen und Ausführungsobjekte als Corba-Objekte mit den entsprechenden Workflow-Facility-IDL-Interfaces
- Nutzen: Erleichterte Integration existierender Workflow-Systeme in Corba-Netzwerke
- Orientierung am Referenzmodell der Workflow Management Coalition
- Weitgehende Verwendung bereits existierender Corba Services and Corba Facilities
- *Keine* Spezifikation einer eigenen Workflow-Definitionssprache oder eines eigenen Workflow Engine-Algorithmus, sondern Schwerpunkt auf Integrationsunterstützung



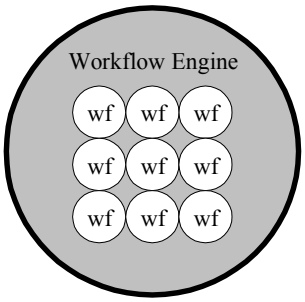
OMG Workflow Management Facility: Historie

- Januar 1997: Einrichtung einer Workflow-Arbeitsgruppe durch die *Business Object Domain Task Force* der OMG
- 1997 Request for Proposals: Aufforderung zu Vorschlägen zur Workflow Facility
- Anforderung von Spezifikations-Vorschlägen bzgl. Workflow-relevanten Diensten, die Objekten in einer Corba-Umgebung zur Verfügung gestellt werden können, um Workflow-Funktionalität zu erbringen
  - Mandatory features (Prozeßmodellierung und -ausführung etc.)
  - Optional features (Flexibilität, Kompensation)
- Juli 1998: Zwei konkurrierende Entwürfe
  - jFlow: IDL-Spezifikation von traditionell (prozedural) entwickelten WfMS, Defizite in der Objektmodellierung und in der Verwendung der CORBA COS, alle führenden WfMS-Anbieter beteiligt
  - Nortel (Northern Telecom): OO-Ansatz, Einbettung in CORBA-Kontext nicht befriedigend
  - Tendenz zu jFlow
- Derzeit noch keine Verabschiedung des Workflow Management Facility-Standards
- Im Folgenden Skizzierung des jFlow-Ansatzes

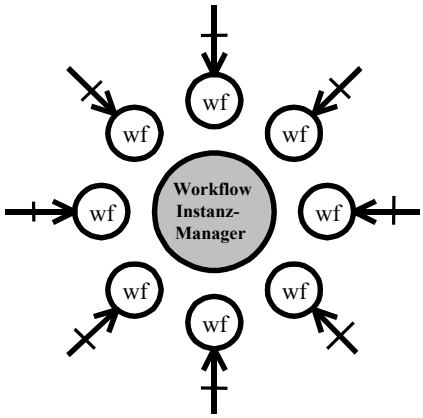


Die OMG Workflow Facility: Corba-Sichtweise von Workflows

„Klassische“ Sichtweise

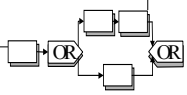


Corba-Sichtweise

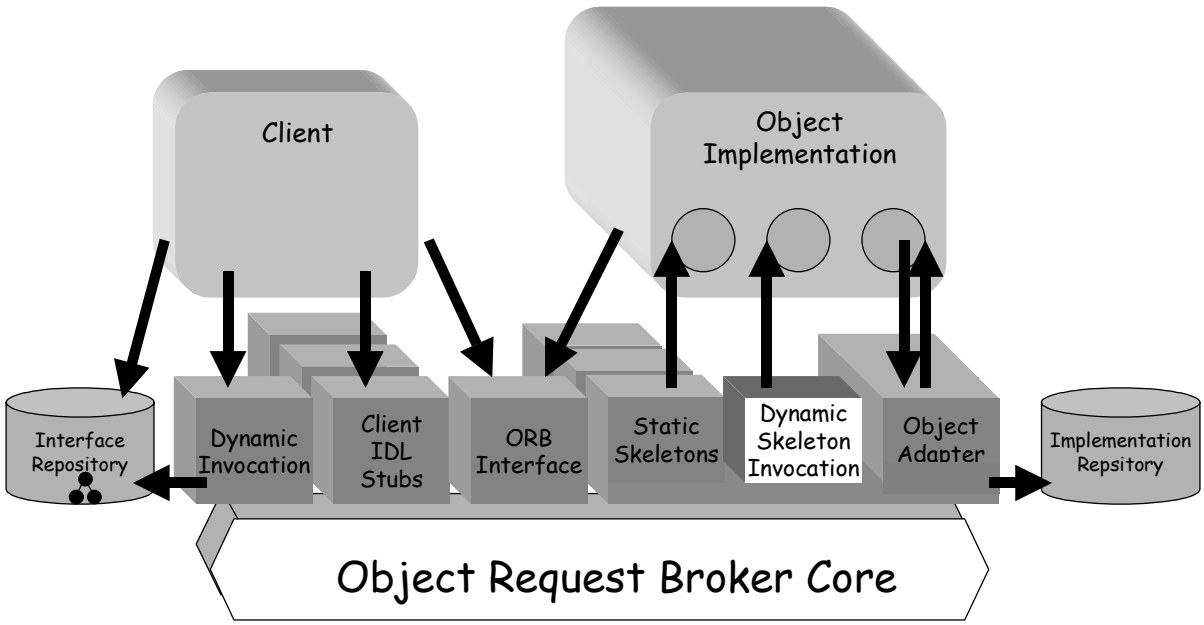


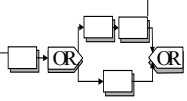
Workflow-Instanz

Workflow-Instanz als „stand-alone“ Objekt mit Interface



Corba-Hintergrund: Object Request Broker

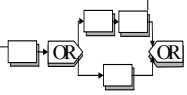




Corba-Hintergrund: Corba Services

- Globale Zielsetzung: Bereitstellung elementarer, betriebssystemorientierter Basisfunktionalitäten für verteilte Objektsysteme
- Kapselung als Corba-Objekte mit standardisiertem IDL-Interface
- Generische High-Level-Schnittstelle zu Betriebssystemen
- Unterstützung der betriebssystem- und hardwareunabhängigen Entwicklung verteilter Applikationen
- Problem: Implementierung vieler Services steht bzgl. kommerzieller Corba-Produkte noch aus

Service	Funktionalität
Naming	Auffinden von Objekten im Corba-Netz
Trading	Beschreibung der Kompetenz und Lokalisation von Corba-Objekten („Gelbe Seiten“)
Event	Management asynchroner Ereignisse
LifeCycle	Generierung, Migration und Löschen von Corba-Objekten
Persistent Object	Persistenz von Corba-Objekten
Concurrency	Nebenläufigkeit von Objekt-Zugriffen
Transaction	Bereitstellung eines Transaktionsschutzes
...	

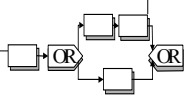


Corba Services und Workflow-Management

- Beobachtung: Teil-Funktionalitäten zur Ausführung von Workflows in vielen Corba Services bereits enthalten
- Folgerung: Nutzung der Services für hochgradig verteilte und interoperable Workflow-Ausführung

→ Workflow Management Facility

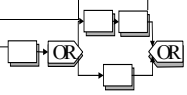
Service	Verwendung im Workflow-Kontext
Event	• Synchronisation von Arbeitslisten • Statusberichte bzgl. Workflow-Ausführung
LifeCycle	• Migration und Replikation von Workflows
Naming	• Lokalisation von Applikationen und Datenquellen
Persistent Object	• Persistente Speicherung und Retrieval von Workflows und Datenobjekten
Collection	• Implementierung von Arbeitslisten
Trading	• Registrierung von Applikationen und ihrer Funktionalitäten
Transaction	• Unterstützung transaktionaler Workflows (→ Kap. 4)
...	



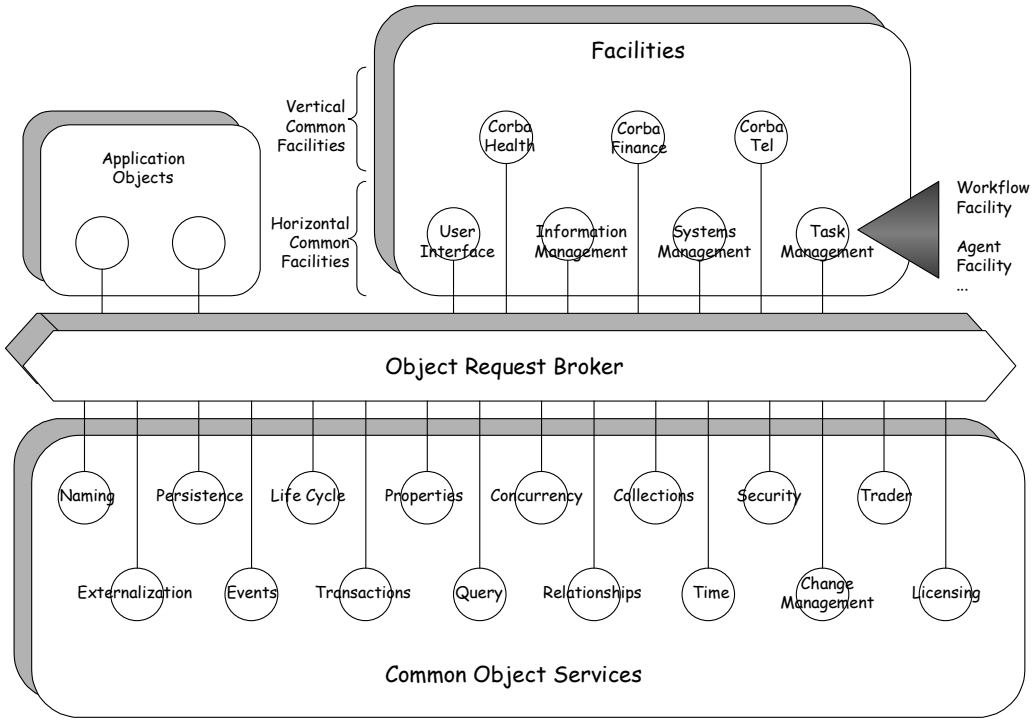
Corba-Hintergrund: Corba Facilities

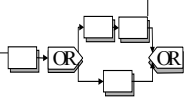
- Bereitstellung spezialisierter Funktionalitäten, die von vielen Applikationsklassen benötigt werden (stärker spezialisiert als Corba Services)
- Applikations-spezifische Konfigurierbarkeit
- Kapselung als Corba-Objekte mit standardisiertem IDL-Interface
- Zwei Hauptgruppen:
  - *Common Facilities* (auch *Horizontal* Facilities): Funktionseinheiten, die unabhängig von einem bestimmten Anwendungsgebiet von einer großen Klasse von Applikationen benötigt werden
  - *Vertical Market Facilities*: Spezifisch für ein Anwendungsgebiet (Medizin, Finanz-Dientleistungen etc.)

Typ	Facility (Auswahl)	Funktionalität
Common	User Interface	Unterstützung der Präsentation von Objekten (graphische Benutzungsoberflächen, Druckersteuerung etc.)
	Data Interchange (Domain *Information Management*)	Austausch von formatierten Daten (TIFF, GIF, EPS, NITF etc); Bulk Datenübertragung; Transfer von OMG IDL-Datentypen Daten-Austausch zwischen Legacy-Systemen
	Time Operations (Domain *Information Management*)	Repräsentierung und Operatoren bzgl. Zeitstempeln, Intervallen etc.
	Rule Management (Domain *Task Management*)	Repräsentierung von ECA-Regeln; Forward- und Backward-Chaining von Regeln
Vertical	Corba Finance	Finanzwesen
	Corba Health	Gesundheitswesen

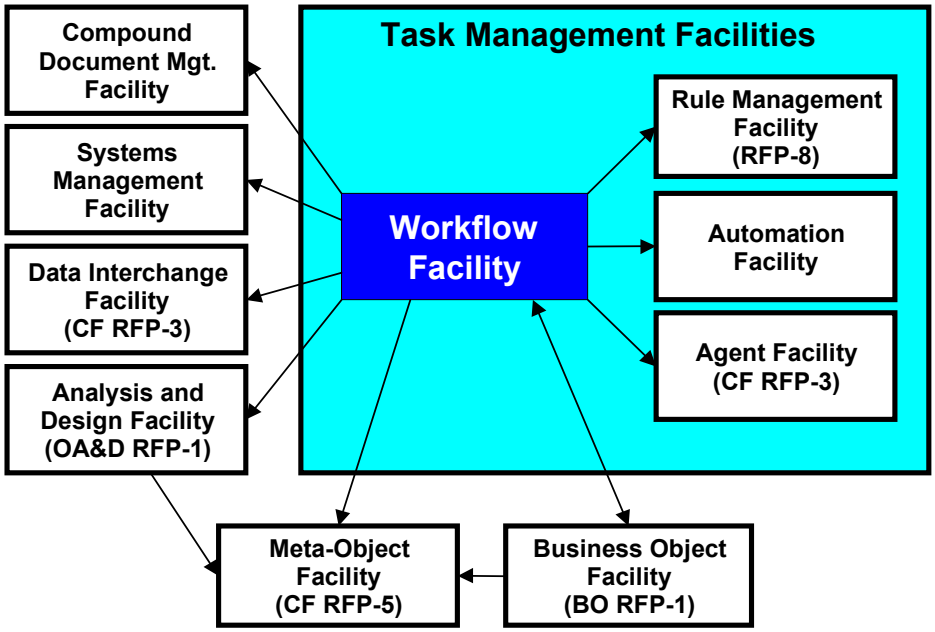


Corba: Services and Facilities

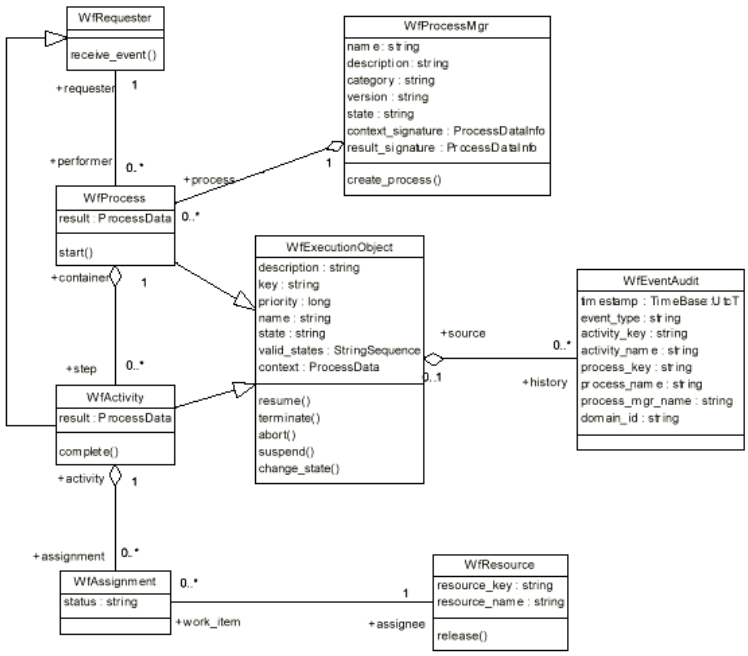




OMG Workflow Management Facility: Einbettung



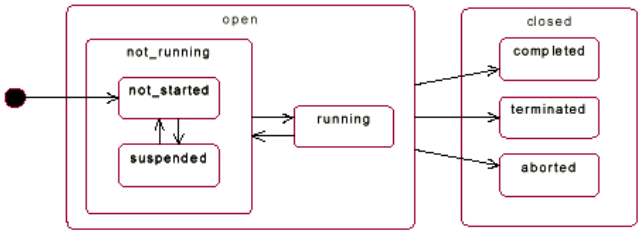
OMG Workflow Management Facility: UML-Schema der IDL-Interfaces



OMG Workflow Management Facility: Beispiele für IDL-Interfaces (1)

Interface	Beschreibung	IDL-Methoden (Auszug)
WfExecution-Object	*A*bstraktes Interface bzgl. Basis-Funktionalitäten für *WfProcess*- und *WfActivity*-Instanzen. Verantwortlich für das Management interner Ausführungs-Stati von *WfProcess*- and *WfActivity*-Instanzen	void suspend() raises (WfBase::BaseException, CannotSuspend, AlreadySuspended); void resume() raises (WfBase::BaseException, CannotResume); void abort() raises (WfBase::BaseException, CannotStop, NotRunning);

- Stati einer *WfExecutionObject*-Instanz:

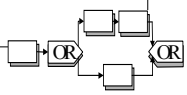


OMG Workflow Management Facility: Beispiele für IDL-Interfaces (2)

Interface	Funktion	IDL-Methoden (Auszug)
WfProcess	Spezialisierung von WfExecutionObject Repräsentiert exakt eine in Ausführung befindliche Workflow-Instanz. Enthält mehrere *WfActivity*-Instanzen für die einzelnen Aktivitäten der Workflow-Instanz	WfActivityIterator get_iterator_step() raises (WfBase::BaseException); WfActivitySequence get_sequence_step(in long max_number) raises (WfBase::BaseException);
WfActivity	Spezialisierung von WfExecutionObject Repräsentiert exakt eine Aktivitäts-Instanz.	WfProcess container() raises(WfBase::BaseException); void complete() raises (WfBase::BaseException, CannotComplete);
WfProcessMgr	Repräsentiert exakt ein Template (= eine Workflow-Definition) erzeugt und verwaltet Workflow-Instanzen (d.h. *WfProcess*-Objekte) zu diesem Template (→ Factory-Objekt)	WfProcess create_process(in WfRequester requester) raises (WfBase::BaseException, NotEnabled);

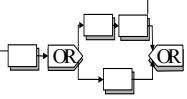
Workflow Management Facility (OMG): Diskussion

- Objekt-orientierter Ansatz mit Schwerpunkt auf Integration von Workflow-Systemen in Corba-Umbungen
- *Keine* eigene Workflow-Definitionssprache sowie *kein* eigener Ausführungs-Algorithmus, sondern rein Interface-basierter Ansatz (mit IDL)
- Unklarheit bzgl. tatsächlicher Unterstützung der WfMS-Hersteller

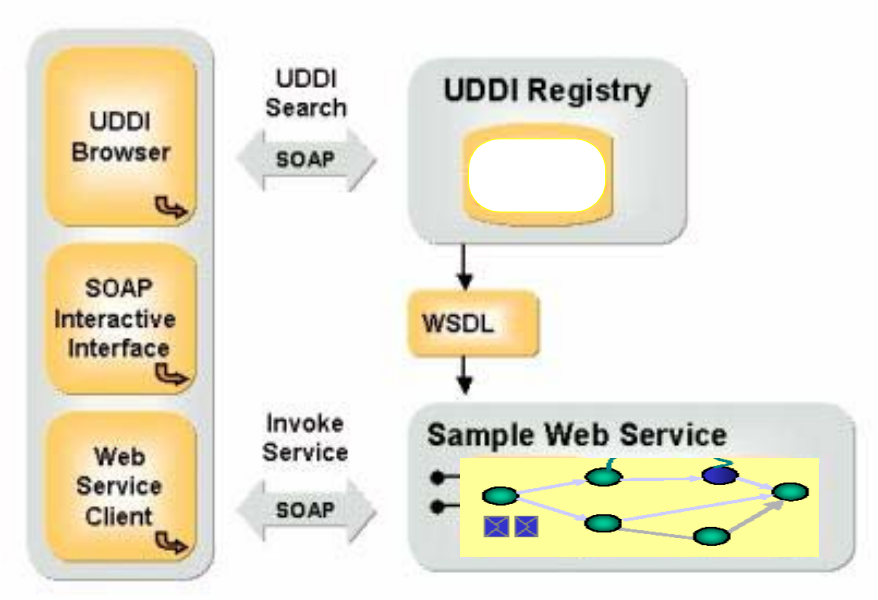


Prozessorientierte E-Service Architekturen

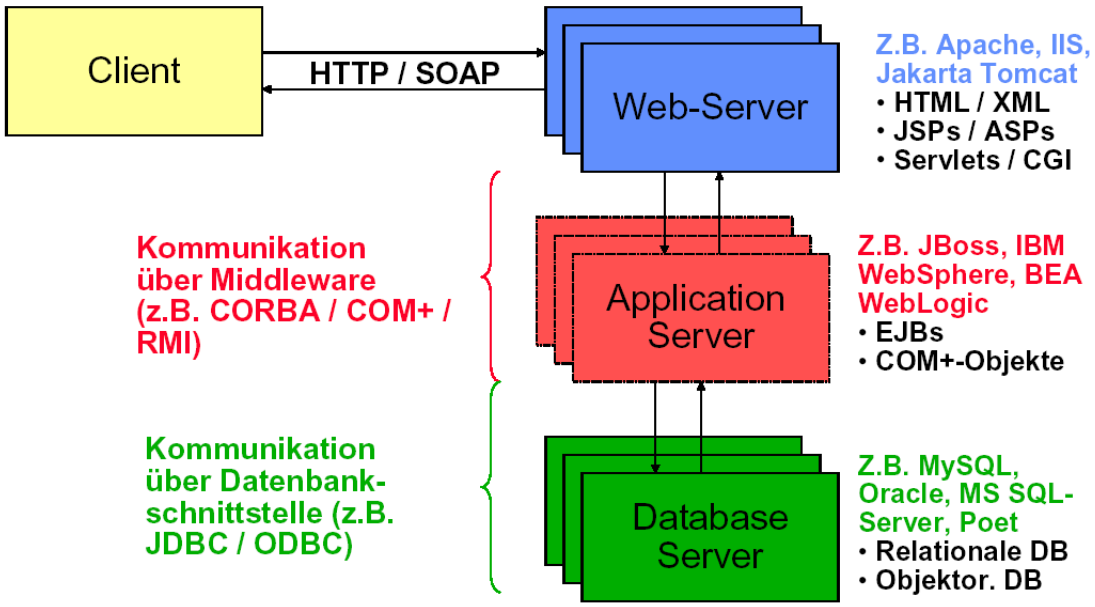
- Ausgangssituation
- Spezifische Anforderungen
- Bespielrealisierungen



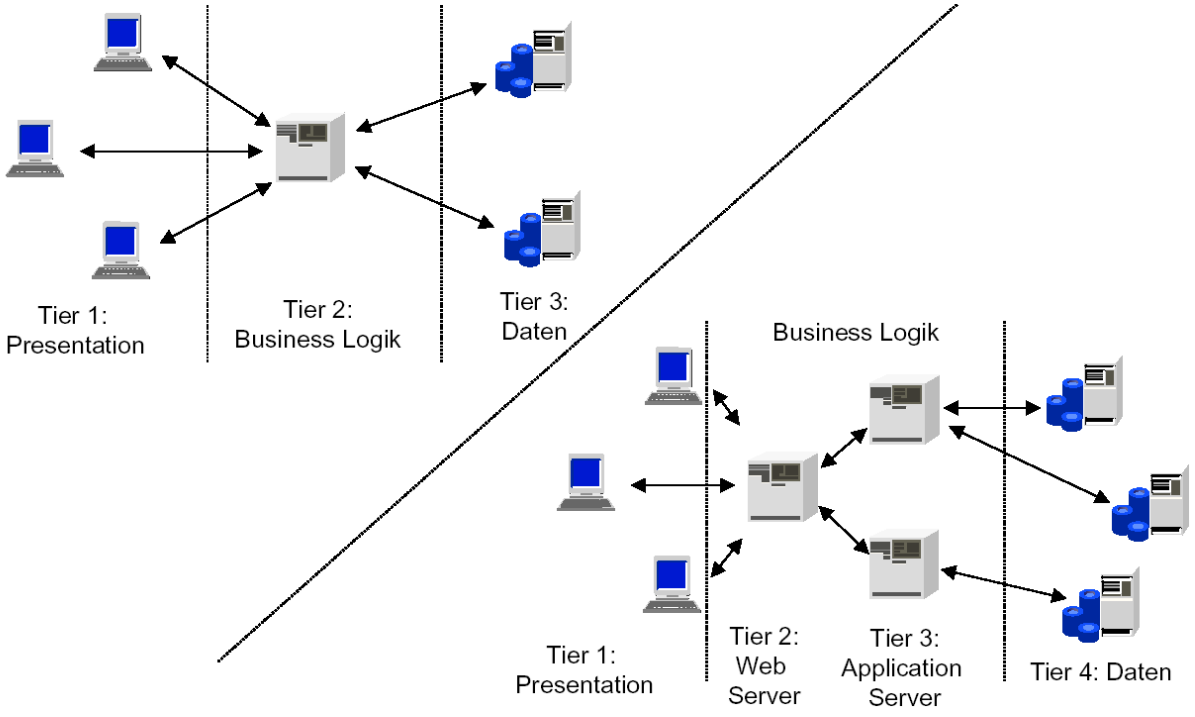
Ausgangssitutation



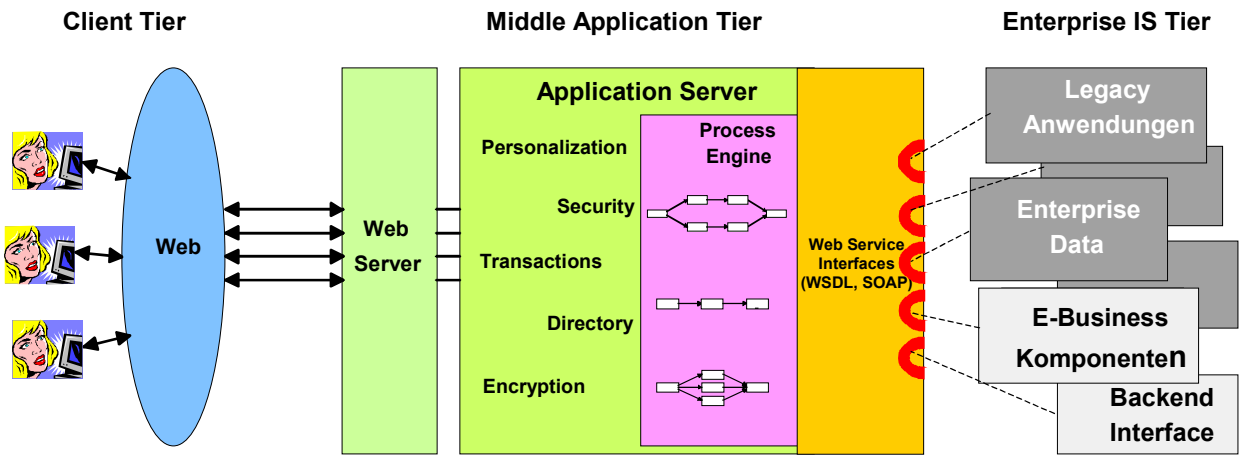
Application Server-Architekturen



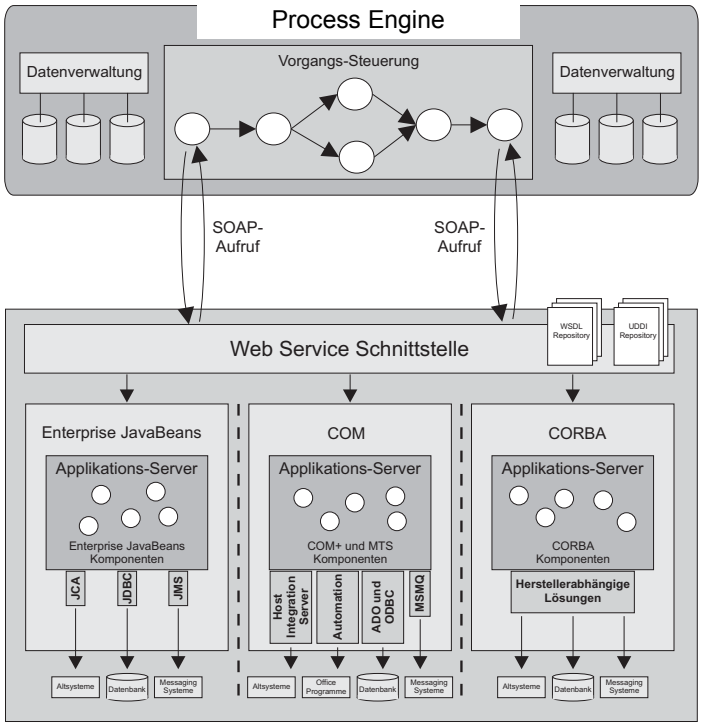
3- und 4-Schichten-Architekturen



Erweiterte Application Server-Architekturen für E-Services



Einheitl. Zugriff auf Komp.-Modelle über Web Service Schnittstelle

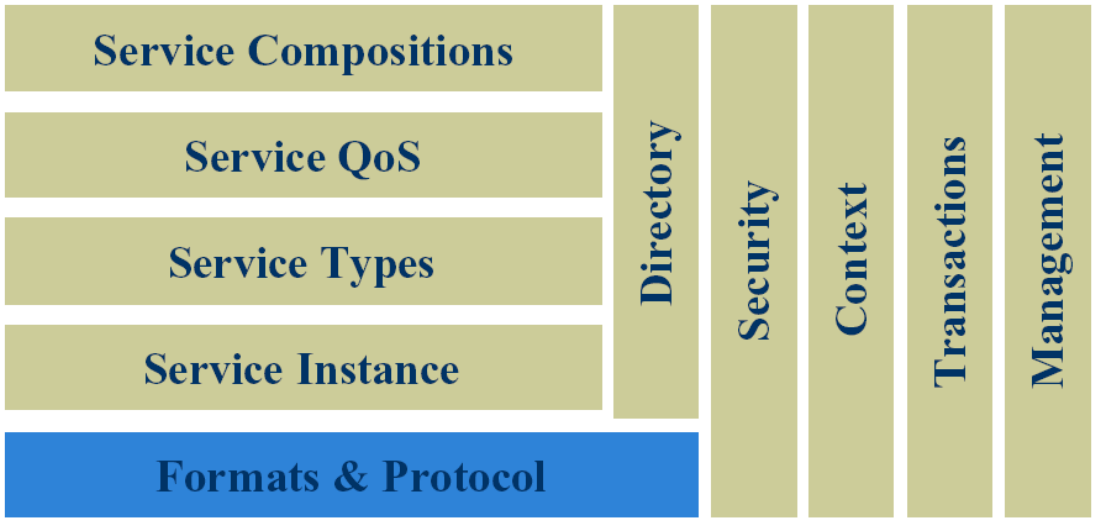


E-Service Architekturen: Aspekte

Ausführungsplattformen (Service Bus)

- Zahlreiche Entwicklungs- und Ausführungsplattformen für Web-Services (nicht standardisiert, herstellerabhängig)
  - Dienste und Schnittstellen für
    - Präsentation
    - Sicherheit
    - Transaktionen
    - Kontextverwaltung
    - Prozess-Definition und -Management
    - Integration von Middleware und Altsystemen
    - ...
  - Beispiele:
    - .NET (Microsoft)
    - Sun ONE
    - Bea Integration Server
    - IBM WebSphere

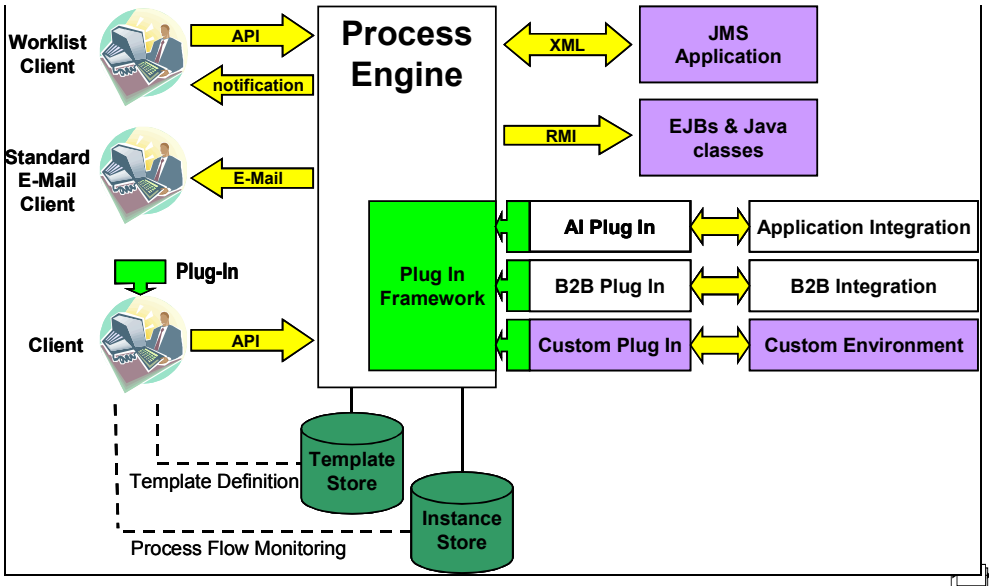
Web Services Framework



Beispiel-Architektur: BEA

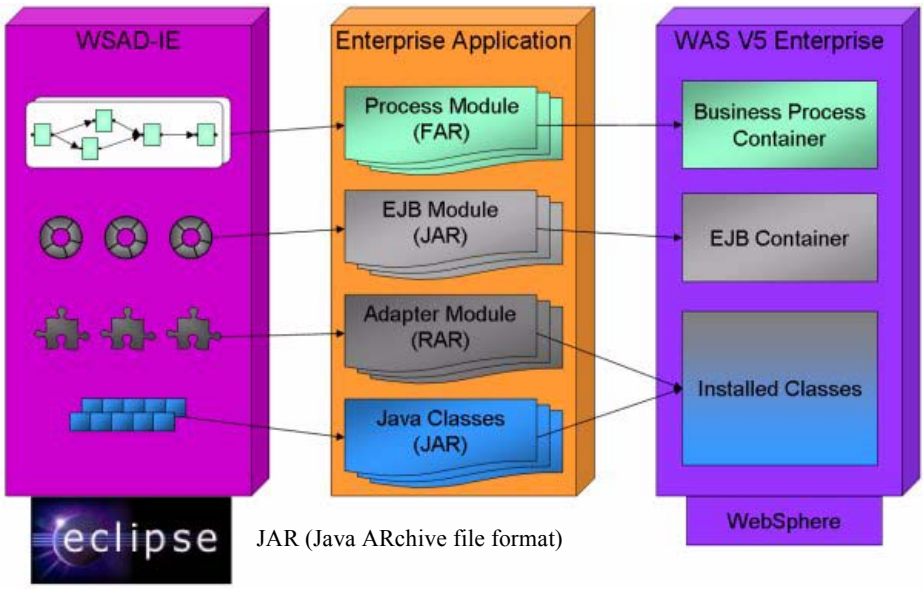
- Process Definition
  - Modellierung von Geschäftsprozessen; Graphischer Entwurf
- Process Execution
  - Ausführung von Geschäftsprozessen; Eingebunden in J2EE Architektur
- Process Monitoring
  - Monitoring von Geschäftsprozessen; Überwachung des Ablaufs

JMS: Java Message Service



Beispiel-Architektur: WebSphere

- WebSphere Studio Application Developer - Integration Edition (WSAD-IE) allows you to create, deploy, install, test, and debug process applications
  - integrates a base development environment for Java classes, servlets, JSPs, EJBs, and static HTML pages with the enterprise functionality of creating Java connectors and Web services
  - offers choreography for dynamic artifacts through Process Choreographer processes
- Process Choreographer
  - covers defining the process, its activities, and the conditions that determine which path of the process is executed and when
  - includes specifying which services implement which activities (such as CICS or IMS via Java Connectors, or access services in an ERP system using Web services)



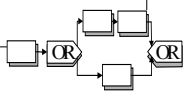
JAR (Java ARchive file format)

WISE-Projekt

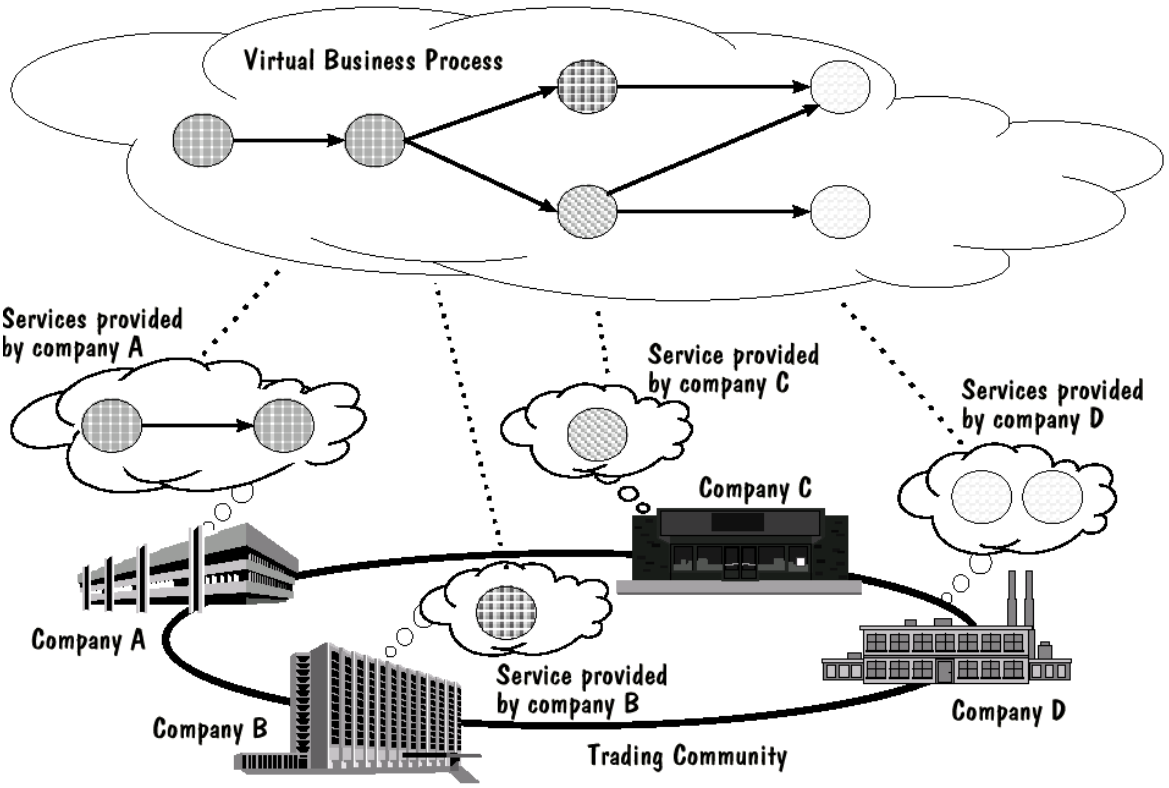
- WISE (Workflow-based Internet SErvices)
- ETH Zürich (Alonso et al.)
- Zielsetzung: Prozessorientierte Unterstützung von B2B E-Commerce
- Quellen
  - http://www.inf.ethz.ch/department/IS/iks/research/wise.html
  - A. Lazcano, G. Alonso, H. Schuldt, C. Schuler: *The WISE approach to Electronic Commerce*. International Journal of Computer Systems Science & Engineering, special issue on Flexible Workflow Technology Driving the Networked Economy, Vol. 15, No. 5, September 2000.

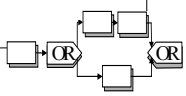
WISE: Modellansatz

- Virtueller Geschäftsprozess
  - Beschreibt konkrete Geschäftsziele und korrespondierende Aktivitäten
  - Nicht beschränkt auf einzelnes Unternehmen oder einzelne Organisationseinheit
  - Kopplung von Aktivitäten und Subprozessen der am Geschäftsprozess Beteiligten
  - Können als Meta-Prozess zu Unternehmensprozessen aufgefaßt werden
  - Hierarchisches Information Hiding zwischen virtuellem Prozess und Subprozessen (die wiederum virtuelle Prozesse sein können)
- Virtuelles Unternehmen
  - Vereinigung der den virtuellen Prozess ausführenden Geschäftspartner
  - Menge der Ziele, Regeln, Anforderungen, Constraints und Ressourcen zur Ausführung virtueller Prozesse
  - Muß *keiner* betriebswirtschaftlichen Rechtsform entsprechen (also z.B. keine GmbH, Aktiengesellschaft oder Firmenverbund sein)
  - Temporärer Charakter



Virtueller Prozess

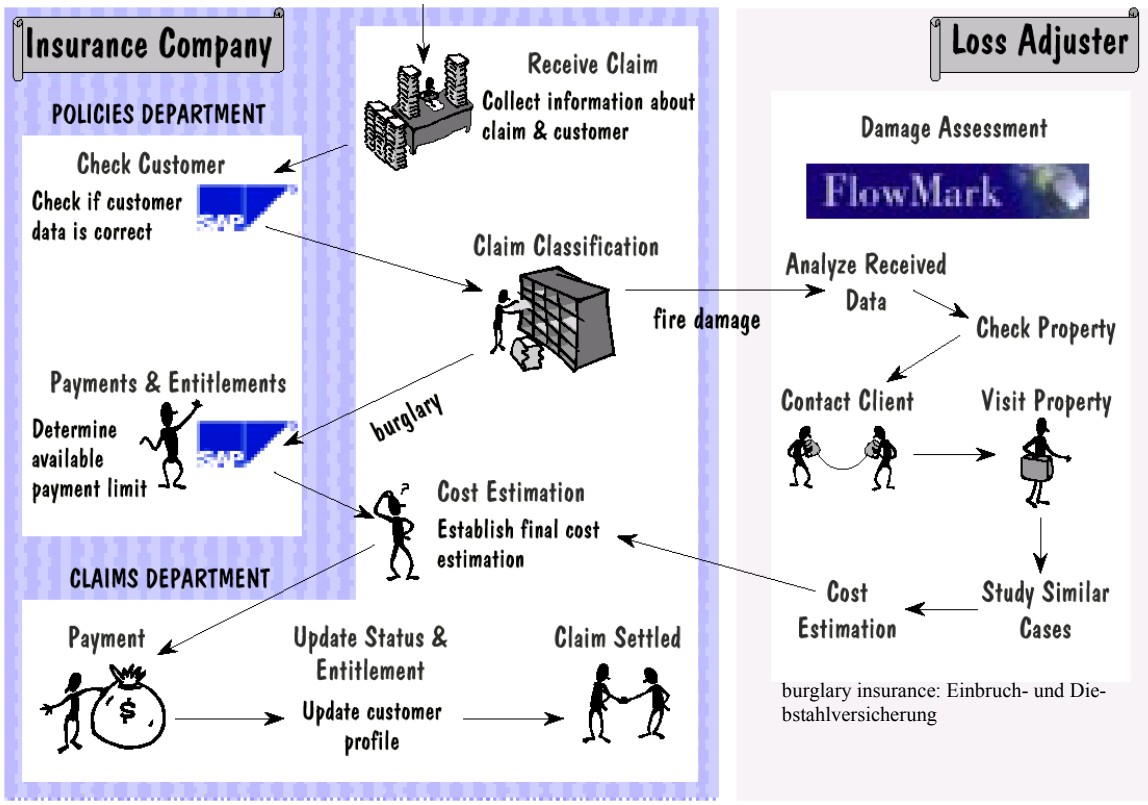




Virtuelle Prozesse/Unternehmen

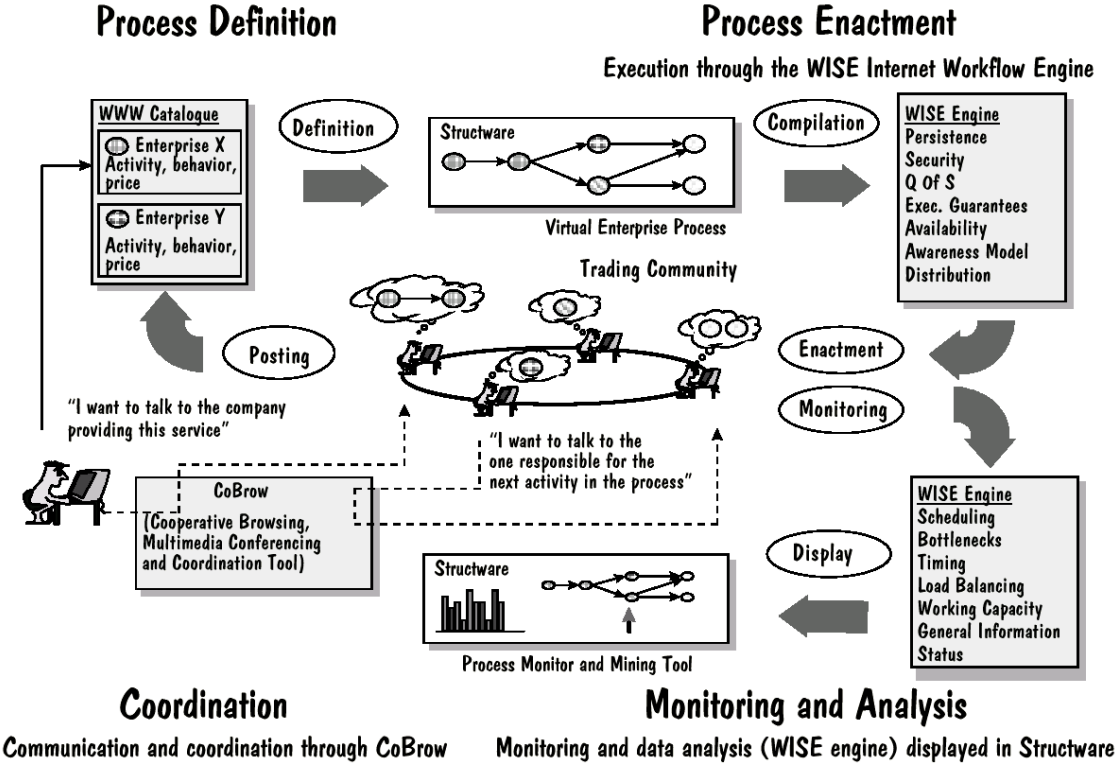
- Vorteile
  - Modelltechnische Organisation (Modularisierung, hierarchische Modellierung)
  - Übergreifende Instanz für Fehlerbehandlung
- Technologische Anforderungen
  - Globale “Internet“-Engine
  - Koordination der Subsysteme
  - Interoperabilität mit den lokalen Workflow-Servern
  - Transaktionsmodell (Kaskadierende Kompensation)

Virtuelle Prozesse/Unternehmen: Beispiel



burglary insurance: Einbruch- und Diebstahlversicherung

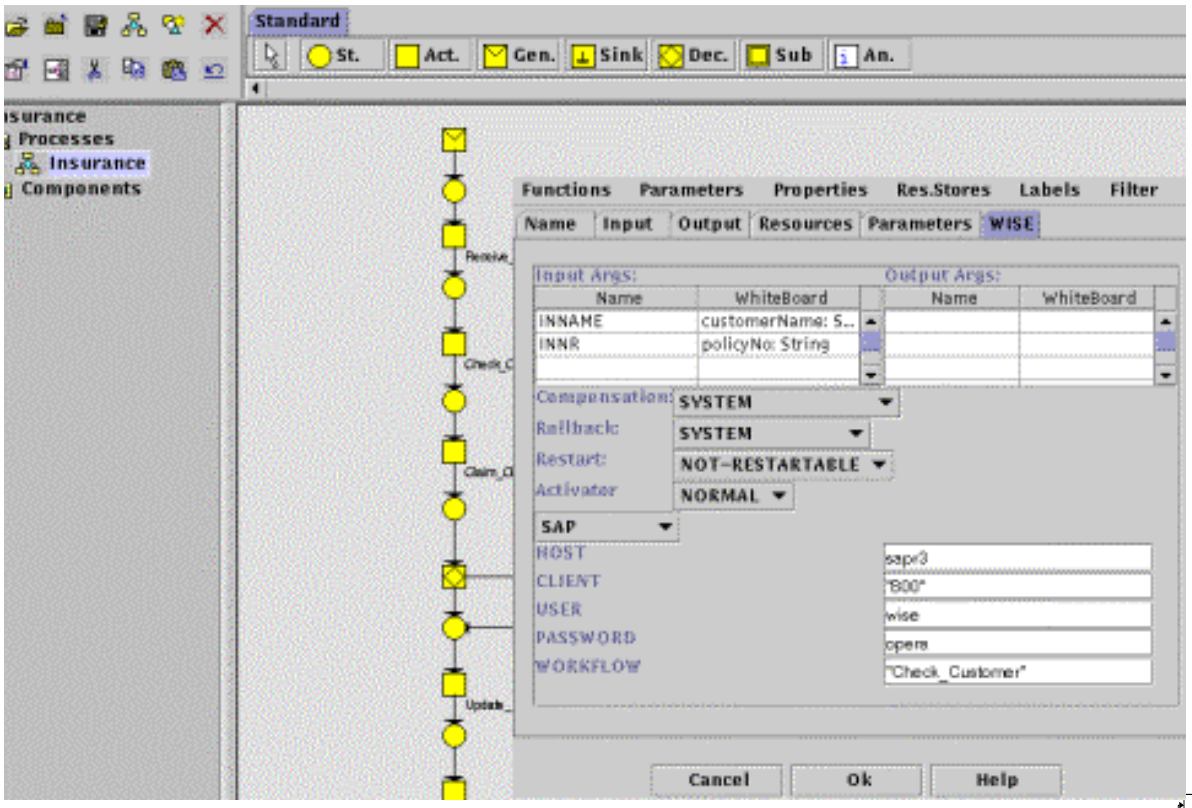
WISE: Infrastruktur

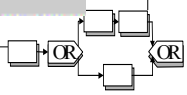


WISE: Prozess-Definition

- Grundbausteine: Die von den Geschäftspartnern bereitgestellten Dienstleistungen (Services)
  - Service-Publikation durch Geschäftspartner mit WWW-Katalog-Tool
  - Java applet/servlet Technology
  - Service-Meta-Spezifikationen: Kosten, Dauer, Garantien, Anforderungen, Seiteneffekte etc.
- Kontroll- und Datenfluss-Definition bzgl. Services
  - Petrinetz-basiert
  - Kommerzielles Prozess-Definitions-Tool STRUCTWARE (IvyTeam)
- Übersetzung nach OCR (Opera Canonical Representation)

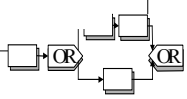
IvyFrame



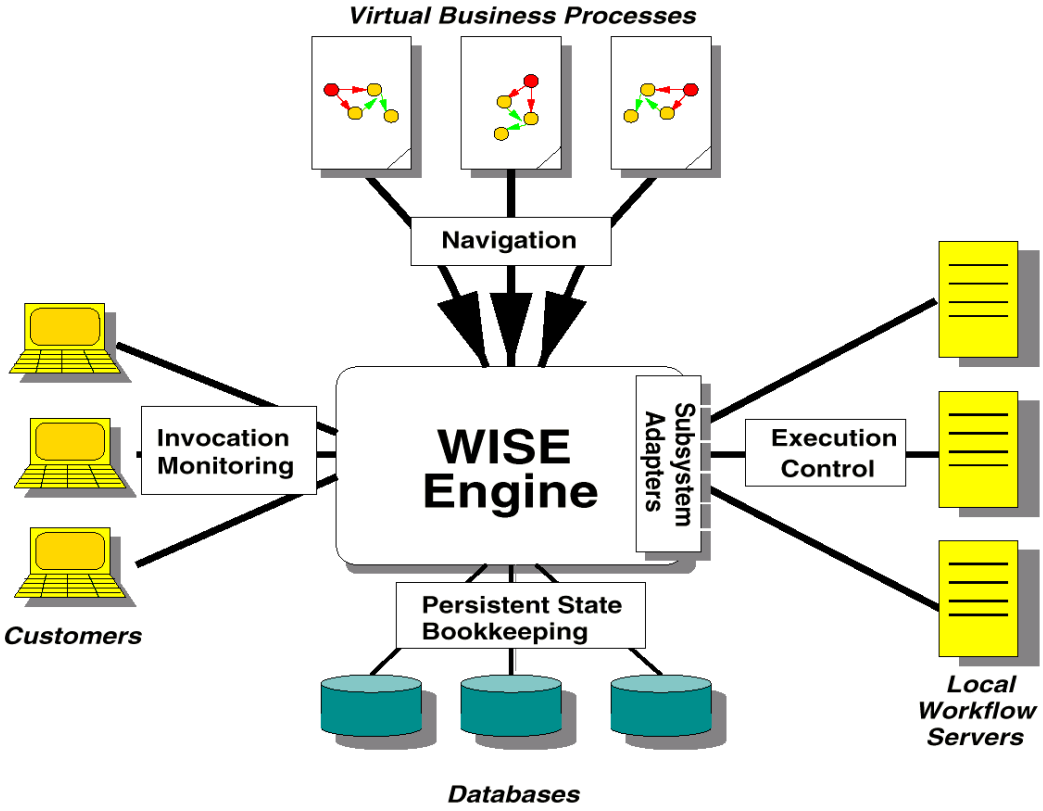


OCR-Beispiel

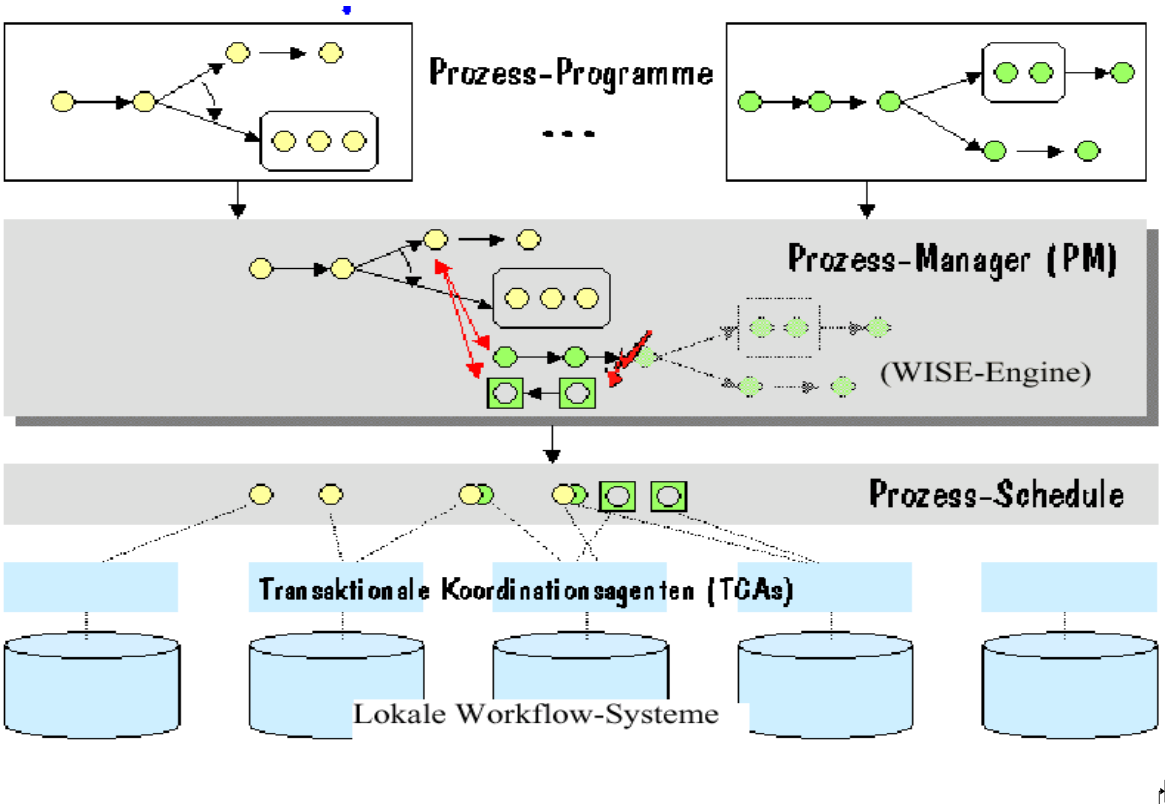
```
PROCESS Insurance ()
        RETURNS ()
        WHITEBOARD (policyNo: String, clientOk: Integer,
                    customerName: String, claimType: String,
                    paymentLimit: Integer, amount : Integer),
        COMPENSATION      SYSTEM,
        ROLLBACK          SYSTEM,
        RESTART           not-restartable

TASKS
        PROGRAM  T_Receive_Claim: Receive_Claim ()
                 STORE (amount > amount) (
                 ACT initial (STARTUP),
                 COND true
        );

        PROGRAM T_Check_Customer: Check_Customer
                  (INNAME = T_Receive_Claim.customerName ,
                   INNR = T_Receive_Claim.policyNo)
                 STORE ( ) (
                 ACT finished (T_Receive_Claim),
                 COND true
        );

        PROGRAM T_Claim_Classification: Claim_Classification
                  (customerName = T_Receive_Claim.customerName,
                   policyNo = T_Receive_Claim.policyNo,
                   clientOk = "1")
                 STORE (customerName > customerName,
                        policyNo > policyNo, clientOk > clientOk,
                        claimType > claimType ) (
                 ACT finished (T_Check_Customer),
                 COND true
        );
```



WISE: Architektur

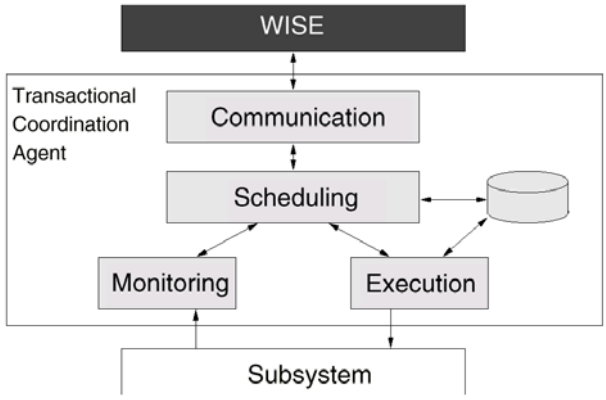


Wise: Prozessbearbeitung



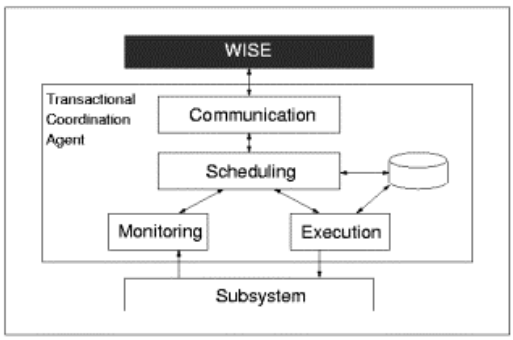
Transactional Coordination Agents (TCAs)

- Anforderungen an Subsysteme u.a.
  - Atomarität der Aktivitätsausführung
  - Falls logische Kompensation angeboten wird: Garantie der erfolgreichen Ausführung (notfalls manuell)
  - Akzeptierung der von der WISE-Engine bestimmten Serialisierungs-Reihenfolge bei Zugriffen auf gemeinsame Datenquellen
  - I.A. nicht von allen Subsystemen per se erfüllt
- Daher: Wrapping jedes Susbsystems in spezifischen TCA
  - Low Level Scheduler bzgl. Subsystem
  - Zwischenschicht zwischen WISE-Engine und lokalen Workflow-Engines
  - 4 Kernaufgaben: Kommunikation, Scheduling, Monitoring und Ausführung

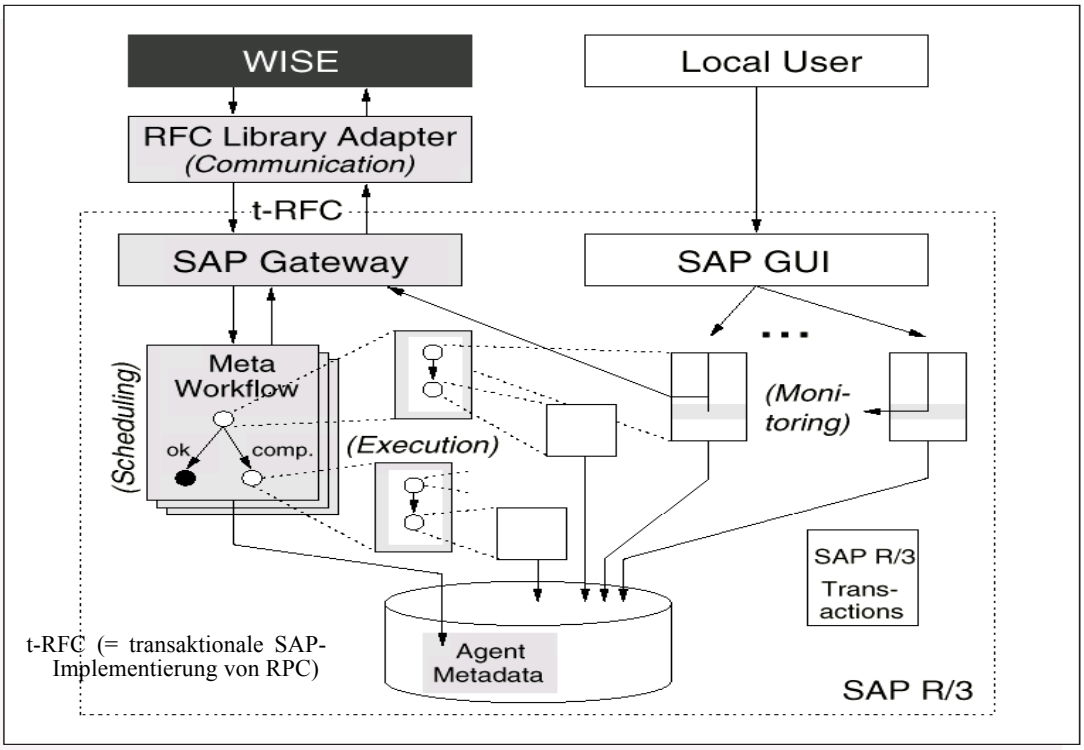


Struktur eines TCA

- Besteht aus vier Modulen
  - Kommunikation
    - Datenaustausch zu WISE
  - Scheduling
    - Lokale Gerantien gegenüber WISE weiterreichen oder selber implementieren
    - Ausführ-Ordnung
    - Kompensationsaktivität
    - Persistenter Log
  - Monitoring
    - Lokale Statusinformationen
  - Ausführung
    - Die Aktivitäten mit den gewünschten Eigenschaften und Garantien ausführen.



WISE SAP R/3 Agent



Zusammenfassung: WISE

- Modellansatz zum Management unternehmensübergreifender Prozesse im B2B-Kontext
  - Virtuelle Prozesse
  - Virtuelle Unternehmen
- Möglichkeiten der übergeordneten Koordination und Fehlerbehandlung
- Technische Herausforderung: „Wrapper“ der lokalen Workflow-Systeme (TCAs)
- Unklar: OrganisatorischeVerantwortlichkeiten auf oberster Ebene
  - Virtuelles Unternehmen ist konzeptionell motivierte Definition, keine Organisationsform
  - Wer ist verantwortlich bei manuellen Eingriffen auf oberster Ebene?
  - Wer kommt finanziell auf, wenn Fehler bei der Prozessausführung witschaflichen Schaden verursachen?

Zusammenfassung

- Grundaspekte der Workflow-Ausführung
- Basis-Architekturen für Workflow-Management-Systeme
  - Ein-Server-Architektur
  - Mehr-Server-Architektur mit Server-Replikation
  - Mehr-Server-Architektur mit Server nahe bei Anwendung
  - Mehr-Server-Architektur mit Server nahe bei Bearbeiter

  Geeignete Architektur hochgradig von Organisationsstruktur des Anwenders abhängig
- Standardisierungsbemühungen (WfMC, OMG) derzeit nur eingeschränkt nutzbar
- Architekturen für prozessorientierte E-Services
- Weiterführendes Material findet sich bei (Auswahl):

Gruppe	Adresse
Universität Ulm (Dadam et al.)	http://www.informatik.uni-ulm.de/dbis/index.html
Universität Saarbrücken (Weikum et al.)	http://www-dbs.cs.uni-sb.de/projekte/workflow_de.htm
Universität Nürnberg (Jablonski et al.)	http://www6.informatik.uni-erlangen.de/research/workflow
TU Dresden	http://wwwdb.inf.tu-dresden.de/