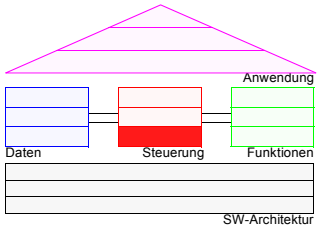
9. Workflow-Management

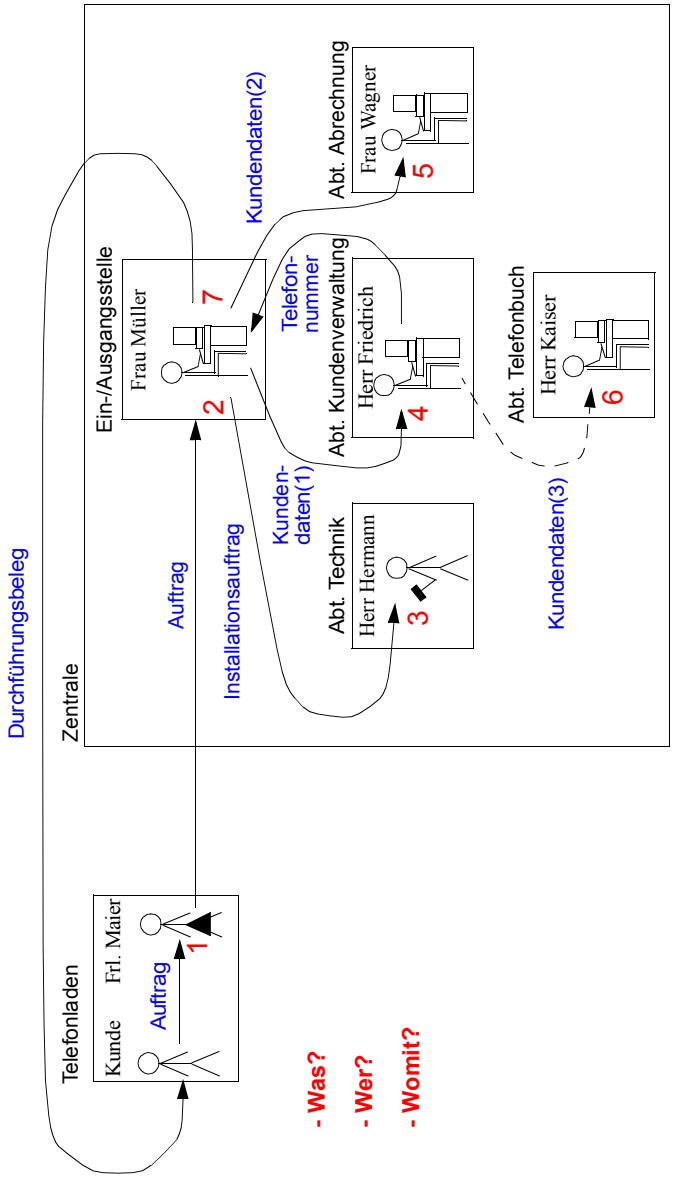
- **GBIS-Rahmen: Einordnung**



- **Aspekte von Workflows**
  - Was, Wer, Womit, Wann
- **Zusammenspiel der Komponenten**
  - Workflow-Referenzmodell der WfMC
  - Ausführung von Workflows
- **Transaktionen und Workflows**
  - Einsatz persistenter Warteschlangen
  - Transaktionsgestützte Wf-Ausführung
- **Funktionale Architektur eines WfMS**
  - Anforderungen an ein WfMS
  - Was sind TP-Monitore?
  - Komponenten der WfMS-Ausführungskomponente
- **Integration von Anwendungen**
  - Enterprise Application Integration
  - Adapter, Connector, Wrapper, Channel
  - Neuer Lösungsansatz: Message Broker und Web Services
- **Beziehungen zwischen den grundlegenden Begriffen**

Workflows

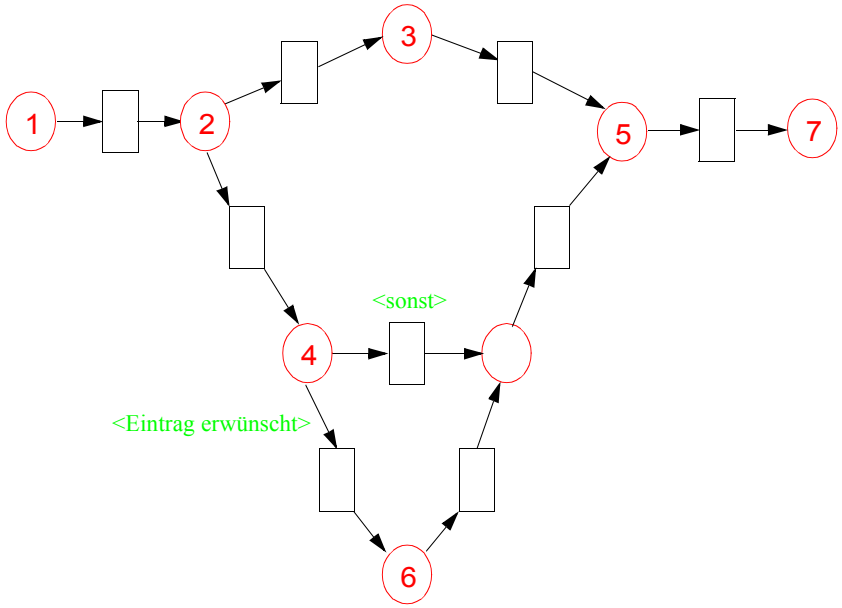
- **Beispiel: Telefonanschluss**



- **Was?**
- **Wer?**
- **Womit?**

Workflows (2)

- **WANN?**

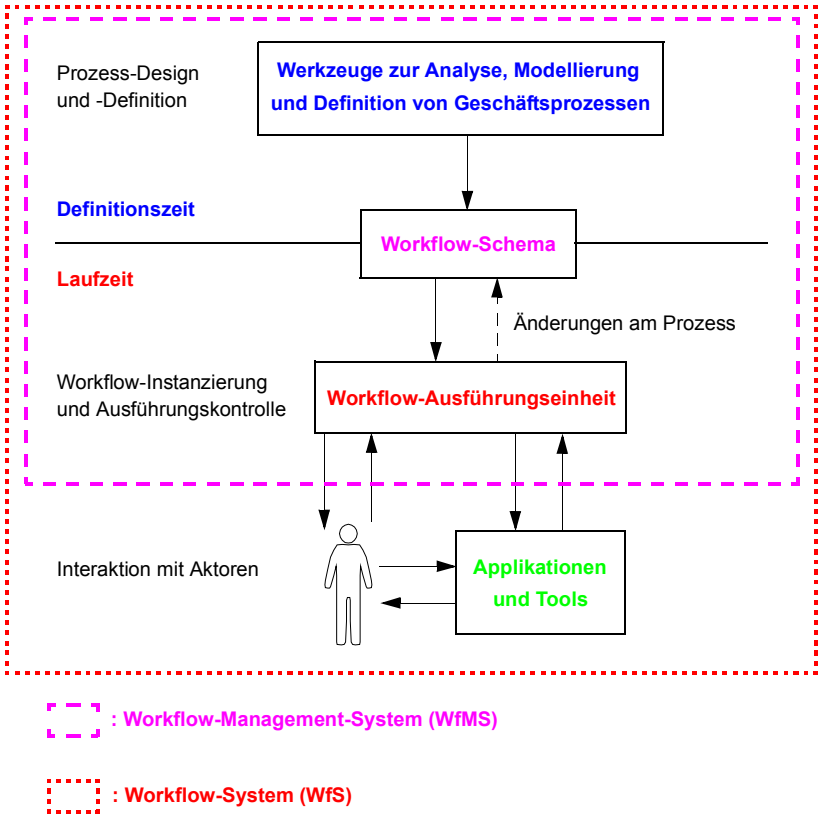


- Graph beschreibt zeitliche Abhängigkeiten zwischen einzelnen Tätigkeiten
- Parallelausführung verschiedener Tätigkeiten möglich
- optionale Ausführung von Tätigkeiten möglich
- Datenfluss i. Allg. verschieden von Kontrollfluss

Workflow-System als Gesamtsystem

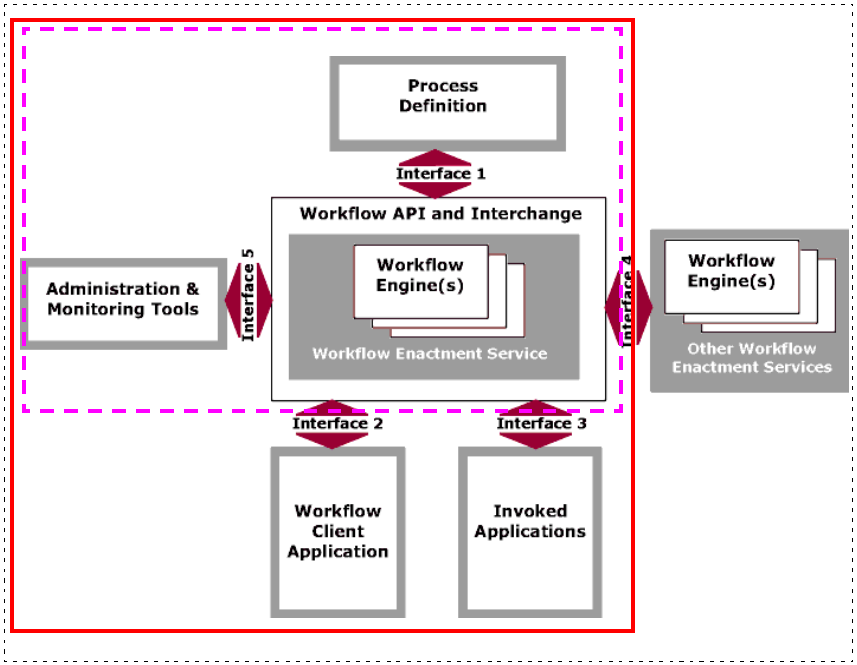
- **Workflow-System (WfS)**
  - besteht aus WfMS, Aktoren und allen zur Aktorenanbindung benötigten Komponenten
  - WfMS besitzt Komponenten für die **Definitionszeit** und für die **Laufzeit**
  - Wf-Ausführungseinheit wird auch als **Workflow-Engine** bezeichnet; sie kann durch mehrere Workflow-Engines realisiert sein

- **Funktionsbereiche eines WfS**



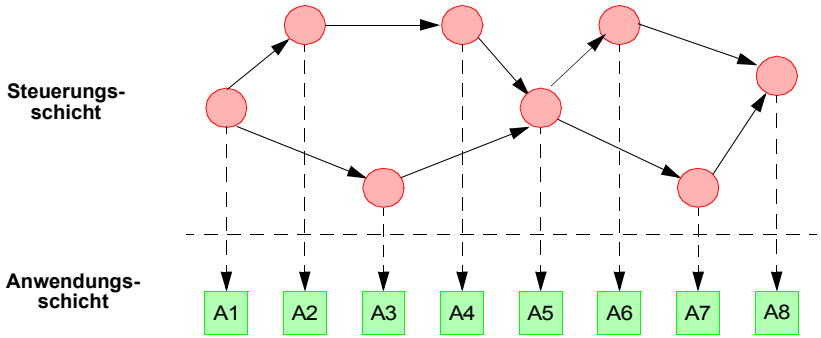
Workflow-Referenzmodell der WfMC

- **Workflow Management Coalition (WfMC)**
  - beschreibt eine generische WfS-Architektur
  - definiert die Schnittstellen und Komponenten
  - Kern des Systems: Wf-Ausführungseinheit (Wf Enactment Service)
  - Schnittstellen 1 und 5 trennen unterschiedliche Funktionschichten im WfMS
  - Schnittstellen 2, 3 und 4 beschreiben die Grenzen des WfMS-Kontrollbereichs



Ausführung von Workflows

- **Eigenschaften von Workflows**
  - verteilt, parallel, heterogen, hierarchisch organisiert, langlebig
  - Aktionen auf getrennten und gemeinsamen Datenbereichen (Wf- und AW-spezifisch)
  - Kooperation zwischen unabhängigen Komponenten (z. B. als Ressourcen-Mgr (RM) realisiert)

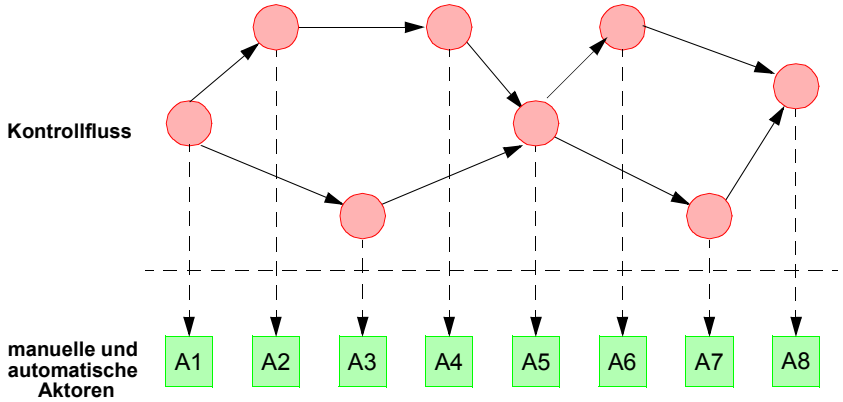


- **Sicherung durch TA-Konzept?**
  - Workflows als globale Transaktionen? - nicht erreichbar, aber auch nicht wünschenswert
  - ACID zumindest selektiv erforderlich
  - kritische Kooperationen erfordern Transaktionsschutz durch spezielle RM-Protokolle

- **Konsequenzen für die WF-Ausführung**
  - frühzeitige, aber trotzdem kontrollierte Freigabe von Änderungen auf gemeinsamen Daten
  - Verwaltung der Ausführungshistorie und der Kontextdaten erlauben Unterbrechbarkeit sowie oft eine Wiederholbarkeit von Aktivitäten
  - persistente Zwischenergebnisse und Nachrichten

Ausführung von Workflows (2)

- **Zwei Arten von Datenhaltung erforderlich**
  - Kontrolldaten (durch WfMS verwaltet)
  - Produktionsdaten (durch DBMS oder Anwendung verwaltet)



- **Anforderungen an die Wf-Ausführung**
  - Kosteneffektivität, Verlustminimierung im Fehlerfall
  - einfache ACID-Transaktionssemantik ist nicht anwendbar

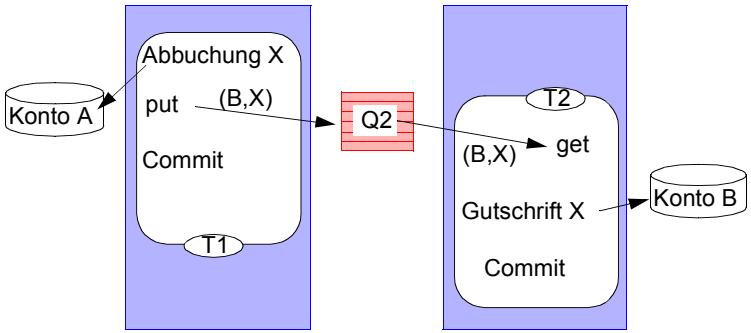
    ➥ semantisch reichhaltigere Fehlerbehandlungsmodelle zwingend erforderlich
  - frühzeitige Freigabe von Betriebsmitteln (v.a. Daten), Recovery im Fehlerfall

    ➥ Semantik der Wf-Ausführung?

Transaktionen und Workflows

- **Workflow als eine globale Transaktion T**
  - Einfluss auf Durchsatz und Leistungsverhalten des Systems
  - ACID-Eigenschaft, dass alle Operationen zur gleichen Zeit erfolgreich beendet werden, aus AW-Sicht oft nicht erforderlich

    ➥ Garantie ausreichend, dass nach erfolgreicher Beendigung einiger Operationen die restlichen „irgendwann“ erfolgreich ausgeführt werden
- **Einsatz von „Persistenten Warteschlangen“ (recoverable messaging)**



Zerlegung der globalen Transaktion
“Buchung von Konto zu Konto”

- **Semantisch reichhaltigere Fehlermodelle**
  - Kombination von ungeschützten Aktionen und Transaktionen
  - Wiederanlauf: persistente Warteschlangen erlauben Wiederholung von TA und Vorwärts-Recovery
  - Scheitern des WF:
    Rücksetzen und Kompensation von TA-Ergebnissen

Transaktionen und Workflows (2)

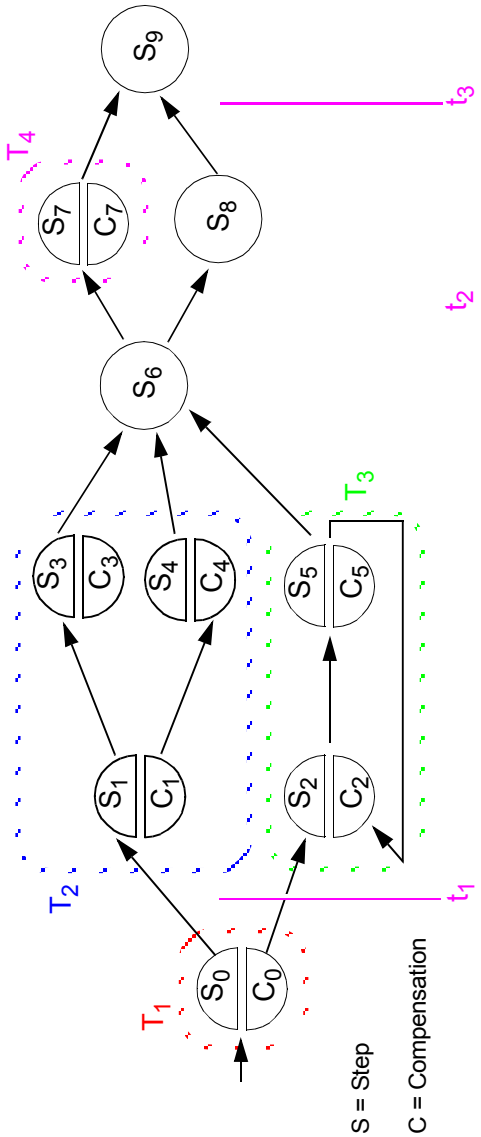
- **AW-orientierte Zerlegung von Transaktion T**
  - in $T_1, ..., T_n$
  - mit persistenten Warteschlangen $Q_1, ..., Q_n$
  - lineare Reihenfolge nicht zwingend
- **Einsatz von persistenten Warteschlangen**
  - zeitliche Unabhängigkeit der Funktionsausführung

    ➥ höhere Unabhängigkeit der Komponenten, höhere Verfügbarkeit des Gesamtsystems
- **Ausführung der Transaktion T**
  - T_i schreibt in Q_{i+1}, T_{i+1} liest aus Q_{i+1}
  - Verkettung der T_i durch spezielles Protokoll (XA-Protokoll von X/Open)
  - jede T_i macht unabhängig Commit
  - es soll sichergestellt werden, dass alle Transaktionen T_i in T erfolgreich beendet werden

    ➥ Voraussetzung: alle aktualisierten Ressourcen (auch Nachrichten) sind wiederherstellbar
- **Scheitern der Transaktion T**
  - zunächst Vorwärts-Recovery bei Crash
  - Weiterführung scheitert auch bei wiederholten Versuchen
  - Rücksetzen offener T_i
  - Kompensation erfolgreich beendeter T_i

Transaktionen und Workflows (3)

- **Transaktionsgestütztes Workflow-Szenario – allgemeine Problemstellung**



- **Recovery-Aktionen bei**
  - Rücksetzen von T_2
  - Crash bei t_1, t_2, t_3
- **Was passiert mit den ungeschützten Aktionen?**

Transaktionen und Workflows (4)

- **Crash in Workflows**
  - offene Transaktionen werden bei Crash zurückgesetzt
  - abgeschlossene Transaktionen bleiben vom Crash unbeeinflußt; bei Rollback des Workflow müssen sie, wenn sie betroffen sind, vollständig kompensiert werden
  - Aktionen, die nicht dem Transaktionsschutz unterliegen, gehen verloren
- **Voraussetzungen für Crash-Recovery**
  - alle persistenten Statusinformationen aller aktiven Workflows können für Vorwärts-Recovery (Fortführung des Wf) benutzt werden
  - persistente Kontexte können einer Anwendungsfunktion wiederholt zur Verfügung gestellt werden
  - persistente Ausführungshistorie gestattet ein Rollback mit zeitlich gestaffelten Aufsetzmöglichkeiten
- **Semantisch reichere Fehlermodelle erforderlich**
  - Es geht immer nach vorne! Ein **„Zurück“ ist eigentlich** nicht vorgesehen
  - frühzeitige Freigabe von Änderungen auf gemeinsamen Daten
  - Verwaltung der Ausführungsgeschichte und der Kontextdaten erlauben Unterbrechbarkeit sowie eine gewisse Art von Vorwärts-Recovery
  - **persistente** Zwischenergebnisse und Nachrichten

➥ Kombination von TA, persistenten Warteschlangen und Kompensation bei Blockierung, Wiederanlauf sowie Rücksetzen von TA-Ergebnissen

Anforderungen an ein WfMS

- **AW-bezogene Anforderungen**
  - Einsatz als Middleware-Komponente
  - Unterstützung verschiedenster Anwendungsgebiete (Flexibilität)
  - Erweiterbarkeit des Wf-Modells
  - Offenheit des Wf-Modells (neue Einsatzbereiche, Integration von Altsystemen)
  - dynamische Anpassbarkeit von Wfs an neue AW-Situationen (Versions- und Konfigurationsverwaltung)
- **Systembezogene Anforderungen**
  - allgemeine Systemeigenschaften: Zuverlässigkeit, Wartbarkeit, Korrektheit, . . .
  - Integration verteilter und heterogener Plattformen
  - Portabilität der WfMS-Implementierung
  - . . .
- **Systemdienste für die TA-orientierten Verarbeitung**

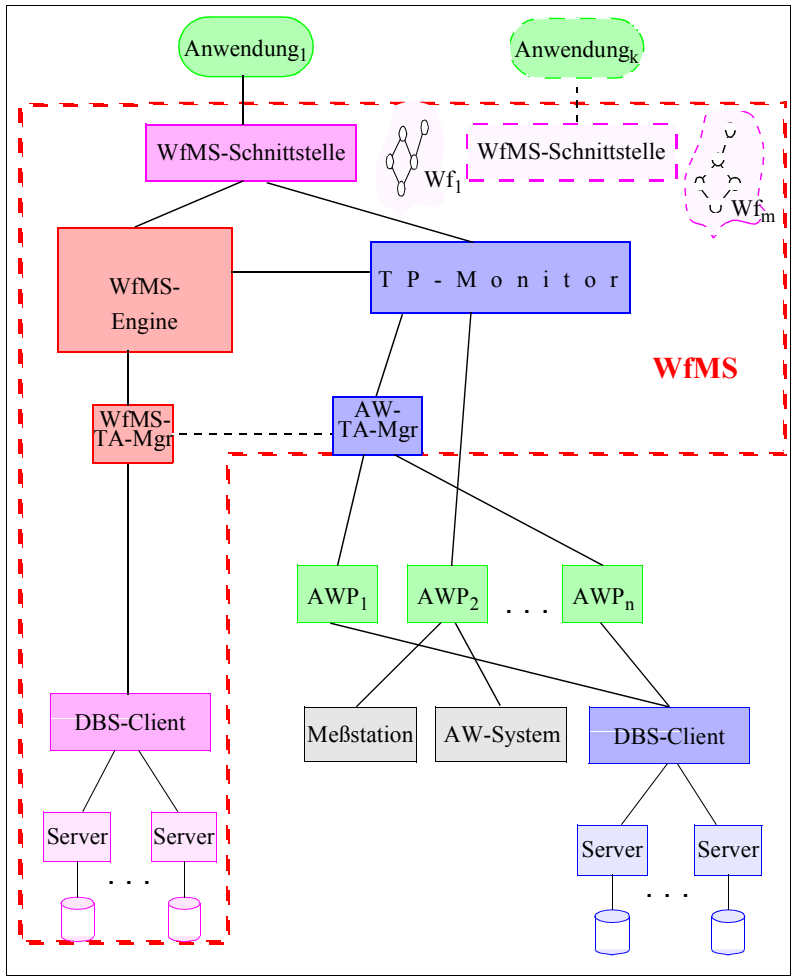
  Gewährleistung der ACID-Eigenschaften für DB-Daten und auch für andere Betriebsmittel (BM wie persistente Warteschlangen, Nachrichten)

  ➥ Weitere Funktionalität erforderlich neben den beiden Arten der Datenverwaltung, geschützt durch TA-Mgr
- **Einsatz eines TP-Monitors**
  - Scheduling in Wfs - Ausführung von AW-Funktionen: **TA-orientiertes BM-Scheduling** (kurzfristige Aspekte, Allokation von Ressourcen für Anforderungen)
  - Verantwortung für Korrektheit der verteilten TA-Verarbeitung und für einheitliche Fehlersemantik (zusammen mit TA-Mgr)

  ➥ alle TP-Monitor-Aufgaben fallen bei jeder Anforderung an: Authentifikation, Autorisierung, Prozess-, Speicher-Allokation

Funktionale Architektur von WfMS

- **Funktionale Architektur im Überblick**



Integration von Anwendungen

- **Stufen der Integration**
  - Die meisten Funktionen eines Unternehmens werden durch SW unterstützt

  ➥ Geschäftsprozess-Integration,
  um weitere Automatisierung und Optimierung der Abläufe zu erreichen

  - Interaktion zwischen mehreren AW-Systemen ist zunehmend auch **unternehmensübergreifend** erforderlich

  ➥ Enterprise Application Integration (EAI),
  z. B. zwischen CRM-System (Customer Relationship Management) des Lieferanten und e-Procurement-System (Beschaffung) des Kunden

  - Schwierigere Integrationsanforderungen, wenn **Daten aus verschiedenen SW-Systemen** verknüpft werden müssen

  ➥ Enterprise Information Integration (EII)
  verlangt Behandlung semantischer Aspekte

- **Typische Probleme der Integration**
  - Plattformabhängigkeit
    - Systeme sind oft auf Betriebssystemplattform zugeschnitten
    - keine Interaktionsmöglichkeiten mit anderen Systemen (-> Adapter)
  - Datenformatabhängigkeit
    - unterschiedliche Datenformate und Datenmodelle
  - Prozessabhängigkeit
    - Ablauf, der Interaktion mehrerer Komponenten erfordert, kann oft nur ganz oder gar nicht verändert werden
    - Fehlerbehandlung ist eine Herausforderung
  - Management und Optimierung nach Integration
    - Beurteilung der Sicherheit, Verfügbarkeit und Leistungsfähigkeit des Gesamtsystems ist schwierig: keine Garantie für Ausfall- und Antwortzeiten?
    - Optimierung (Lastbalancierung, Caching) ist schwierig

  ➥ Neue Techniken: Autonomic Computing, Grid Computing usw.

Integration von Anwendungen (2)

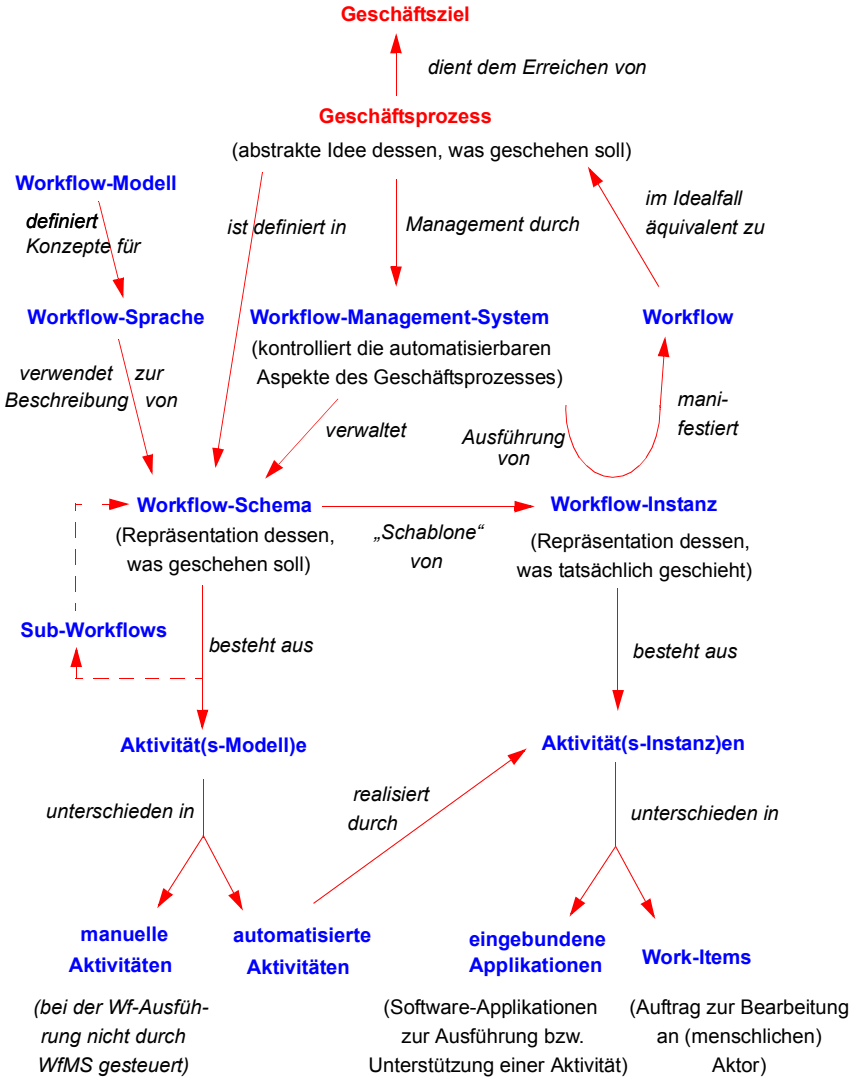
- **Traditionalle EAI-Technologie ist sehr verwirrend**
  - Adapter, Connector, Wrapper, Channel, ...
- **Neuer Lösungsansatz nach folgenden vier Prinzipien**
  - Lose Kopplung
    - Systeme bleiben autonom und kommunizieren über Nachrichten
    - gemeinsames Kompilieren und Binden findet nicht statt: unabhängige Weiterentwicklung
  - Virtualisierung
    - Austauschbarkeit von Komponenten
    - Kommunikationspartner werden häufig dynamisch bestimmt
  - Einheitliche Konventionen
    - Dienste unterstützen viele unterschiedliche Datenformate, protokolle und Mechanismen
    - verwendete Technologien werden nicht eingeschränkt
  - Standards
    - Erfolgsrezept ist Festlegung von Standards
    - alle großen Plattform-Anbieter (wie BEA, IBM, Microsoft, ...), Anbieter von Softwareanwendungen (wie SAP, Siebel, ...) und Anwender halten sich daran!

➥ **Sprachansatz: BPEL4WS**
(Business Process Execution Language for Web Services)

- **Neue Technologie: Web Services**
  - Werden durch einen URI (Uniform Resource Identifier) identifiziert
  - Schnittstelle ist maschinenlesbar und wird durch WSDL (Web Service Definition Language) beschrieben
  - Dienstnutzung durch XML-Nachrichten und Internet-Protokollen (HTTP, ...)
  - Qualitätseigenschaften durch zusätzliche Vereinbarungen geregelt

Zusammenfassung – Workflow-System

- **Beziehungen zwischen den grundlegenden Begriffen (nach WfMC)**



Zusammenfassung

- **Eigenschaften von Workflows**
  - verteilt, langlebig, parallel, heterogen, hierarchisch organisiert
  - TA-geschützte und ungeschützte Aktivitäten
  - Workflow als globale TA? **–**
    nicht erreichbar, aber auch nicht wünschenswert

- **Anforderungen an die Wf-Ausführung**
  - Kosteneffektivität, Verlustminimierung im Fehlerfall
  - semantisch reichhaltigere Fehlerbehandlungsmodelle
    zwingend erforderlich
  - frühzeitige Freigabe von Betriebsmitteln (v.a. Daten),
  - Kompensation / Recovery oder manuelle Behebung im Fehlerfall

  ➥ **kein globales ACID, aber zumindest selektiv erforderlich**

- **Prozeß- und Anwendungsintegration**
  - Geschäftsmodelle und -Prozesse im Web: Beschreibung der Abläufe
    (Workflows): BPEL4WS
  - Schutz durch Geschäftstransaktionen (BTs), die ACID-TA als „Bausteine“
    benutzen (zur Sicherung unternehmenskritischer Aktivitäten)

- **TAs in einer WS-Umgebung (BTs)**
  - sind komplex, mehrere (nicht-vertrauenswürdige) Teilnehmer
  - überspannen Organisationsgrenzen und sind langlebig
  - benutzen Web Services zur Realisierung (XML, WSDL, SOAP, UDDI)
  - neue Formen der Atomarität erforderlich:
    Verbindlichkeits-, Konversations-, Vertrags-, Zahlungs-, Warenatomarität
    sowie als komplexeste Form Atomarität bei zertifizierter Lieferung

  ➥ **Alle Konzepte beruhen auf Kompensationen, meist erforderlich wegen vorzeitiger Freigabe von Daten**