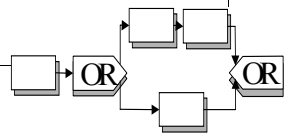
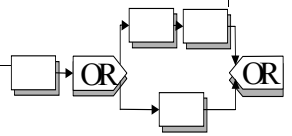
Weitere Workflow-relevante Transaktionsmodelle

- Kontrollsphären
- Kompensationssphären
- Atomaritäts- und Isolationssphären
- Zusammenfassung und Diskussion



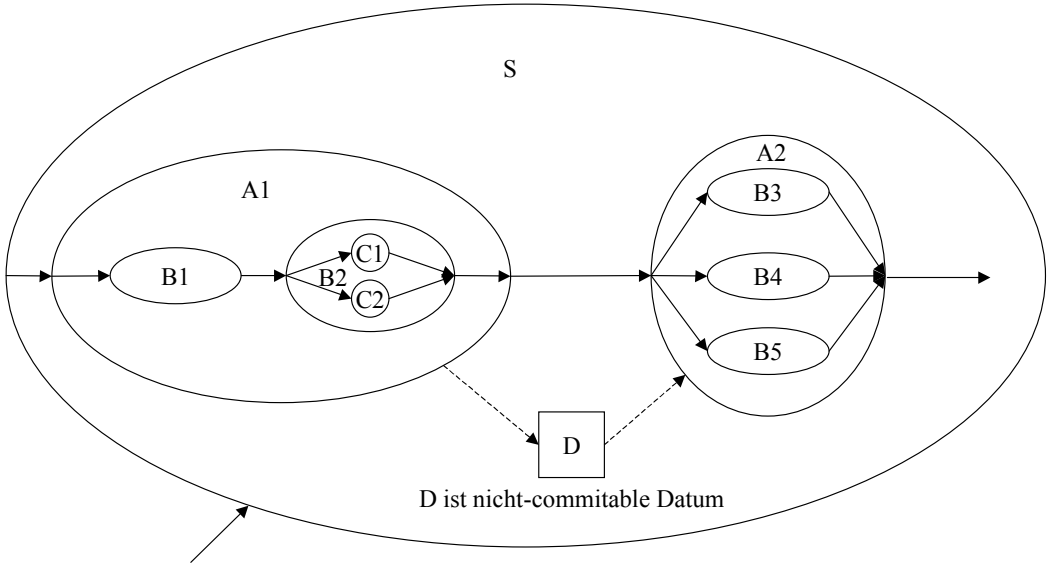
Kontrollsphären

- Davies 1978, Gray & Reuter 1993
- Modell zur Beschreibung des Verhaltens komplexer Applikationen unter Berücksichtigung von Konfliktsituationen und Fehlerbedingungen
- Eine Kontrollsphäre (Sphere of Control; SoC)
  - enthält aus Sicht der Außenwelt eine atomare Aktion *A*
  - kontrolliert Commitment der Datenoperationen von *A*
  - führt Monitoring und Aufzeichnung der Inter-Prozeß-Abhängigkeiten durch, um im Fehlerfall Ausführungshistorie analysieren zu können
  - kann aus Untersphären aufgebaut sein
- Geeignet zur Beschreibung und Behandlung von Daten-Abhängigkeiten zwischen Workflows
- Formen: Kompensationsphären, Atomaritätssphären, Isolationsphären



Kontrollsphären: Kernidee

- Statische Strukturierung von Prozessen in hierarchische, logische Einheiten (Atomarität einer Sphäre)
- Daten-Commitment nicht streng an Prozeßgrenzen gebunden
- Kontrolle über Ergebnis-Daten eines Prozesses *nach* dessen Beendigung
- Anwendungsbeispiel *Hausverkauf*:
  - Andreas (*A1*) will Jim (*A2*) Haus verkaufen
  - Dazu nötig: Aushändigung der Besitzurkunde an Jim; Übergabe der vereinbarten Geldsumme an Andreas
  - Gegenseitiges Mißtrauen:
    - Andreas: keine Urkunde an Jim, bevor nicht Geld überwiesen ist
    - Jim: Keine Geldüberweisung, bevor nicht Urkunde erhalten



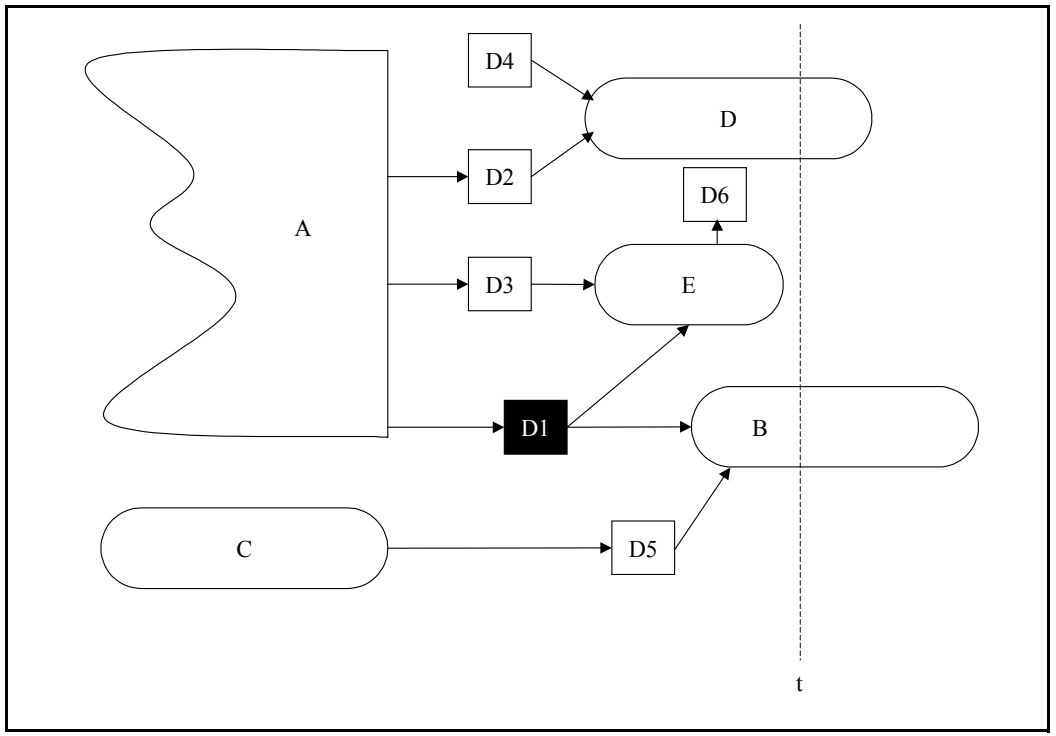
Dynamisch erzeugt für Kontrolle bzgl. Commitment von A1

- Lösungsmöglichkeit:
  - Dynamische Erzeugung einer Kontrollsphäre *S*, die *A1* und *A2* umfaßt
  - Ausgabe von A1 als non-commitable Datum unter Kontrolle von S
  - Im Fehlerfall (z.B. während Ausführung von *A2*) unilaterales Undo durch *S*

Kontrollsphären: Dynamisches Verhalten (1)

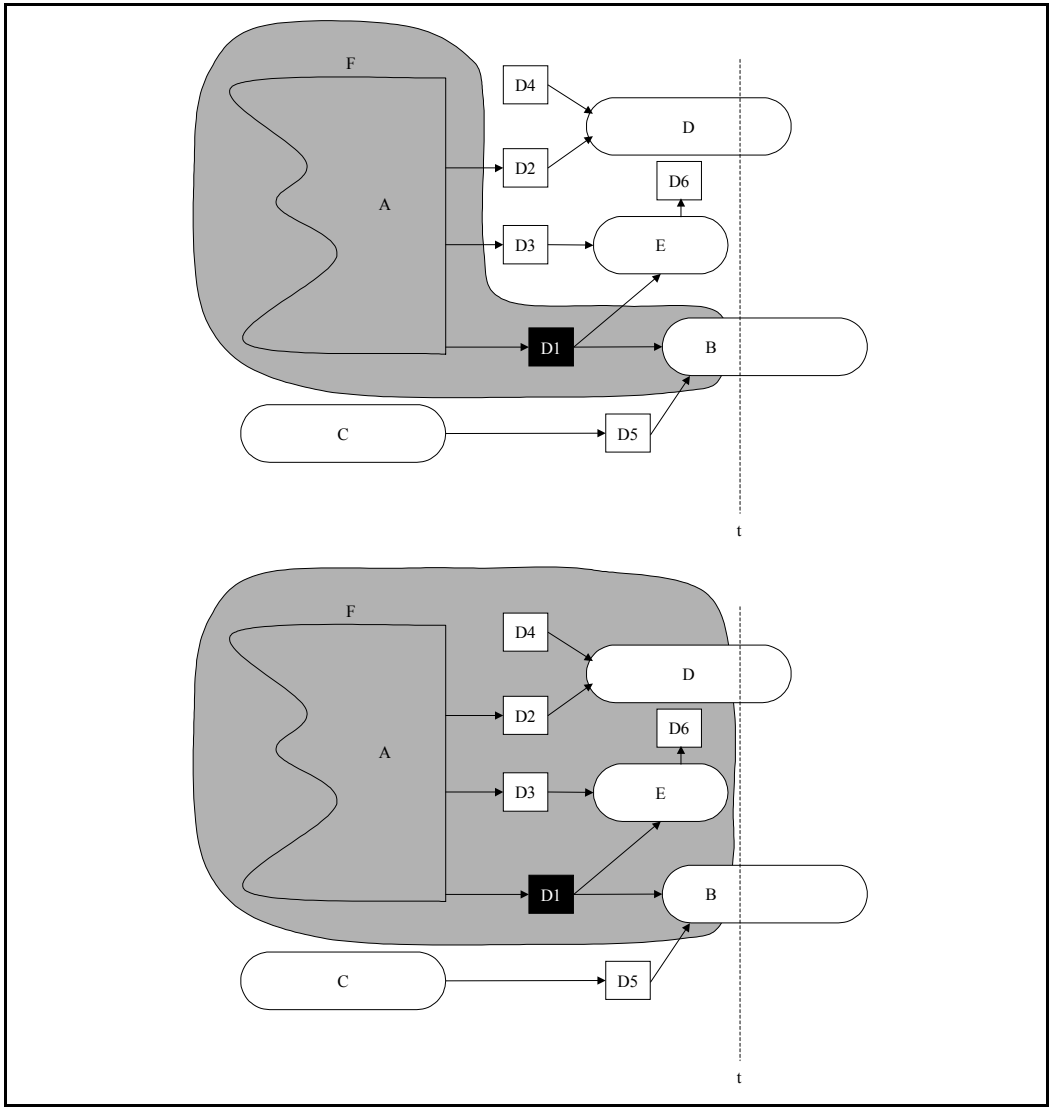
- Kontrollsphären erlauben auch die Behandlung von Fehlern, die
  - auf invaliden Ausgabe-Daten eines Prozesses *P* beruhen und
  - erst lange nach Beendigung von *P* erkannt werden (z.B. durch Verletzung von Constraints bzgl. eines Folgeprozesses)

- Beispiel:
  - Kontrollsphären *A*, *C* und *E* sind beendet
  - Fehler zum Zeitpunkt *t* während Durchführung von *B* (aufgrund von *D1*)
  - Undo problematisch, da *D1*-abhängige Prozesse (hier: *E*) bereits ihre Resultate (hier: *D6*) freigegeben haben können



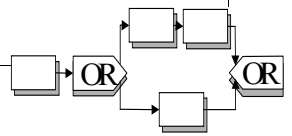
Kontrollsphären: Dynamisches Verhalten (2)

- Schritt 1: Identifikation des für die invaliden Daten verantwortlichen Prozesses (hier: *A*)
- Schritt 2: Erzeugung einer dynamischen SoC (hier: *F*), die verantwortlichen SoC (hier: *A*) enthält
- Schritt 3: Erweiterung der dynamischen SoC *F* bzgl. aller Prozesse, die eine Abhängigkeit zu den invaliden Daten aufweisen (hier: *E*)
- Schritt 4: Recovery innerhalb der dynamischen SoC *F*



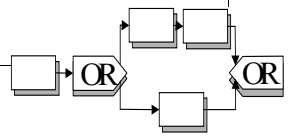
Kontrollsphären: Diskussion

- Konzept zur Fehler-Behandlung unter Berücksichtigung von Inter-Prozeß-Abhängigkeiten
- Dynamische Kontrollsphären entsprechen strukturell zur Laufzeit generierten offen geschachtelten Transaktionen
- Realisierung der Recovery innerhalb der dynamischen generierten SoC ist
  - im Ansatz nicht spezifiziert
  - hochgradig von den beteiligten Applikationen abhängig
- In typischen Workflow-Szenarien werden Ergebnisse, die auf (nachträglich) als invalide indentifizierten Daten basieren, oft durch *manuelle* Aktivitäten weiterbearbeitet
- Beispiel
  - Auf Girokonto (mit 2000 DM) einer Person *P* wird per Scheck hoher Betrag gutgeschrieben (10.000 DM)
  - Person hebt 13.000 DM ab (keine Überschreitung des Dispokredits von 1000 DM)
  - Scheck wird als ungedeckt identifiziert
  - Dynamisch erzeugte SoC für Recovery enthält Prozeß *„P hebt Geld ab“*
  - Kompensation dieses Prozesses: Benachrichtigung von *P*, mit Aufforderung (ausgegebenes?) Geld zurückzubringen
- Bisher keine vollständige Formalisierung und Implementierung erfolgt



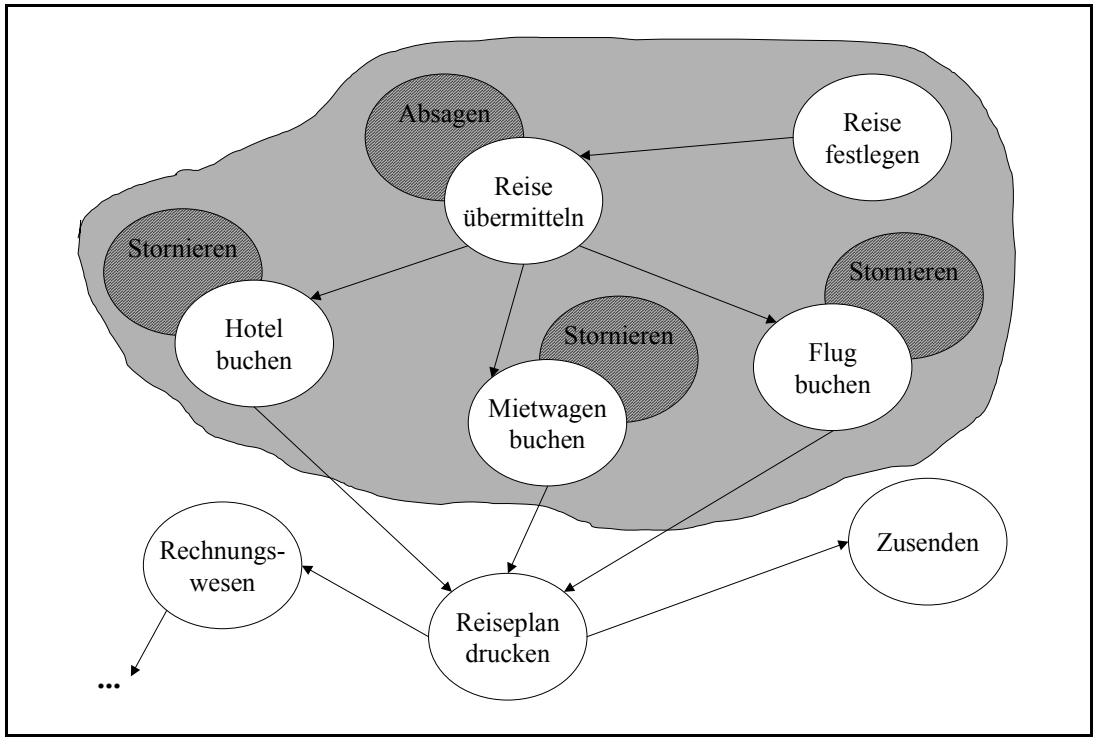
Kompensation

- Bei langlebigen Transaktionen impliziert Lockerung der Isolation, daß im Fehlerfall Transaktion nicht einfach auf Ausgangszustand zurückgesetzt werden kann
- Ausführung der semantischen Inverse
- Problem: In vielen (Workflow-)Anwendungsszenarien signifikanter Anteil an Operationen, deren Effekte sich nicht einfach rückgängig machen lassen
  - Verabreichung eines Medikaments
  - Verschicken eines Mahnbriefes
  - Benachrichtigung über Lottogewinn
- Kompensationskonzept: Im Fehlerfall Anwendung von Operationen, die Effekte einer Transaktion möglichst minimieren (kompensieren)
  - Verabreichung eines Medikaments: Absetzen und Verabreichung eines Gegen-Medikamentes (falls vorhanden); Laborwertkontrolle
  - Verschicken eines Mahnbriefes: Verschicken eines Entschuldigungsschreibens
  - Benachrichtigung über Lottogewinn: Verschicken eines Widerrufs und Trostzahlung
- Kompensationssphären
  - Formalisierung der Kompensation
  - Berücksichtigung der Inter-Sphären- und Inter-Transaktion-Abhängigkeiten



Kompensationssphären

- nach F. Leymann, IBM
- Sei $P = (N,E,...)$ Workflow-Modell. $\phi \neq S \subset N$ heißt *Kompensationssphäre* genau dann, wenn gilt
  - alle Aktivitäten $A \in S$ müssen "graph-syntaktisch erfolgreich" beendet worden sein, oder
  - alle Aktivitäten $A \in S$ müssen kompensiert werden
  - A ist graph-syntaktisch erfolgreich, falls A aktiviert und ohne Fehler beendet wurde, oder aber vom Kontrollfluß innerhalb S nicht mehr erreicht werden kann
- Sei $Spheres_P$ Menge aller Kompensationspären zu P. Dann heißt $P = (N,E,...,Spheres_P)$ transaktionaler Workflow
- Kompensationsspären können
  - sich überschneiden
  - hierarchisch aufgebaut sein
  - müssen nicht zusammenhängend sein
- Zuordnung von Kompensationsaktivitäten zu Aktivitäten und Kompensationssphären
  - $K : N \cup Spheres_P \rightarrow$ *Menge aller Aktivitäten* (inkl. NOP = no operation)
  - Kompensationsaktivität kann Programm, manuelle Aktivität oder ganzer Workflow sein

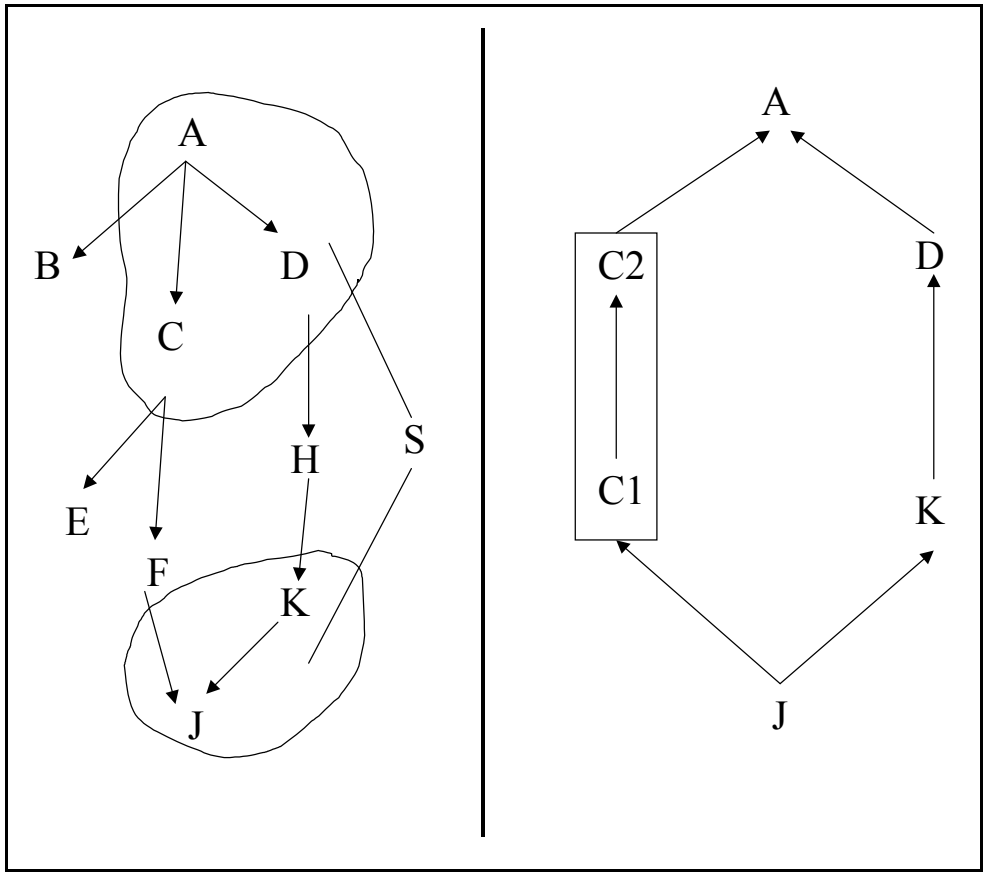


- Kompensationssphären spezifizieren semantische Einheiten des Rücksetzens im Fehlerfall

Kompensation einer Sphäre

- Beim Scheitern einer Aktivität $A_{Fehler} \in S$ Durchführung der Kompensationsaktivitäten in Rückrichtung des Kontrollflusses

- Bei Schleifen-Aktivitäten ("do A until exit-condition holds")
  - einmalige oder iterative Kompensation
  - Spezifikation durch Attribut *iterate_compensation[yes | <u>no</u>]*

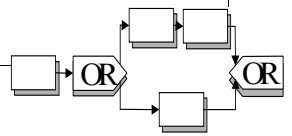


- Mitarbeiterzuordnung
  - Mitarbeiter der *A* durchgeführt hat führt auch *K(A)* durch
  - *K(A)* wird von anderem, speziellem Mitarbeiter durchgeführt (z.B. durch Abteilungsleiter bei Entschuldigungsschreiben an Kunden)

- Alternative zu Kompensation in Rückrichtung: Parallel-Kompensation (alle Aktivitäten werden parallel kompensiert)

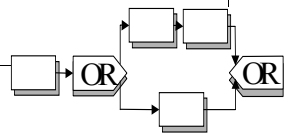
Granularität einer Kompensation

- Falls Sphäre *S* Aktivität *A* enthält, die durch Subworkflow ausgeführt wird: Flache oder tiefe Kompensation
  - *A* beschrieben durch $P_A = (N_A, E_A, ...)$
  - Flache Kompensation: Nur *K(A)* wird ausgeführt
  - Tiefe Kompensation: Ausführung der $K(A_i)$ für $A_i \in N_A$
  - Spezifikation durch Attribut *nesting[deep | <u>shallow</u>]*
- Integrale oder diskrete Kompensation einer Sphäre *S*
  - Integrale Kompensation: Nur *K(S)* wird ausgeführt
  - Diskrete Kompensation: Ausführung der *K(A)* für $A \in S$
  - Spezifikation durch Attribut *integral[yes | <u>no</u>]*
- Beispiel
  - Kunde bestellt Produkt
  - Bestätigungsschreiben an Kunden (mit Lieferdatum)
  - Produktionsauftrag geht an Produktseinrichtung, Lieferauftrag an Spediteur
  - Fehlersituation: Spediteur kann zum Lieferdatum nicht liefern
  - Diskrete Kompensation: Informationsbrief an Kunden, Meldung an Produktionseinrichtung über größere Zeitspanne
  - Integrale Kompensation: Nur Brief an Kunden



Proliferation einer Kompensation

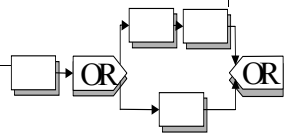
- Zuordnung eines Proliferationsattributs zu Aktivitäten oder Sphären zur Beschreibung des Einflusses einer Kompensation auf die Nachbarschaft
- Proliferation bzgl. einer Aktivität:
  - *proliferation[local | <u>global</u>]*
  - *local*: Tritt ein Fehler bei Durchführung bei einer Aktivität *A* mit *proliferation = local* auf, so wird nur *A* kompensiert
  - *global*: Bei Fehler bzgl. *A* wird gesamte Sphäre, die *A* umfaßt, kompensiert
- Proliferation bzgl. einer Sphäre:
  - *proliferation[sphere | <u>cascading</u>]*
  - *sphere*: Tritt ein Fehler bei Durchführung innerhalb einer Sphäre *S* mit *proliferation = sphere* auf, so wird nur *S* kompensiert (nicht aber von *S* abhängige Sphären)
  - *cascading*: auch von *S* abhängige Sphären werden kompensiert
- Festlegung der Proliferation zur Definitionszeit oder dynamisch



Kompensationssphären: Backout

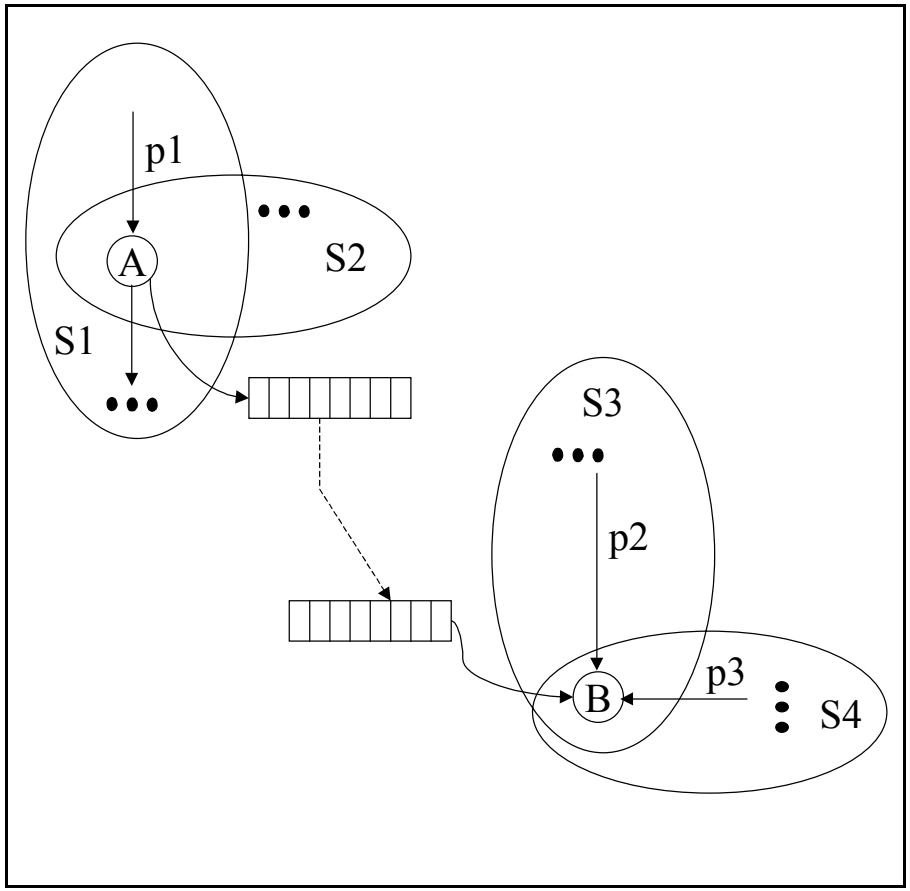
- Sei *S* Sphäre bzgl. *P = (N,E,...)*:
  - $S_{formal\text{-}initial} = \{A \in S \mid \{ e \in E \mid \pi_2(e) = A \} = \phi \vee \{ e \in E \mid \pi_2(e) = A \wedge \pi_1(e) \notin S\} \neq \phi \}$
  - $S_{formal\text{-}initial}$ heißt Menge der formalen Initial-Knoten von *S*
  - $S_{de\text{-}facto\text{-}initial}$: Menge der zum Zeitpunkt des Fehlers aktuell durchlaufenden formalen Initial-Knoten
- Backout[2]-Modi
  - *Retry*: Nach Kompensation von *S*: Erneute Ausführung bzgl. $S_{de\text{-}facto\text{-}initial}$ (Default)
  - *Undo*: Keine erneute Ausführung nach Kompensation
  - *Rerun*: Keine Kompensation, sondern direkt erneute Ausführung
  - Festlegung zur Definitionszeit oder dynamisch

2 to back out: zurücksetzen, aussteigen

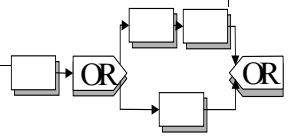


Kompensationssphären: Kaskadierung (1)

- Sphären nicht unabhängig voneinander
- Zwei prinzipielle Abhängigkeiten bzgl. Sphären
  - Gemeinsame Aktivitäten
  - Datenabhängigkeit
- Gemeinsame Aktivitäten: $S1 \cap S2 \neq \phi$
  - Notwendige Bedingung für Kaskadierung *backout(S1)* $\rightarrow$ *backout(S2)*: $\exists A \in S1 \cap S2 : A$ wurde durchgeführt
  - Hinreichende Bedingung: *S2* hat zur Aktivierung von *A* beigetragen
  - Beispiel: Bedingung für B sei $p2 \vee p3$. Falls *backout(S3)* und *p2 = TRUE, p3=FALSE,* so folgt nicht *backout(S4)*. Falls *backout(S3)* und *p2 = TRUE, p3=TRUE,* so folgt *backout(S4)*

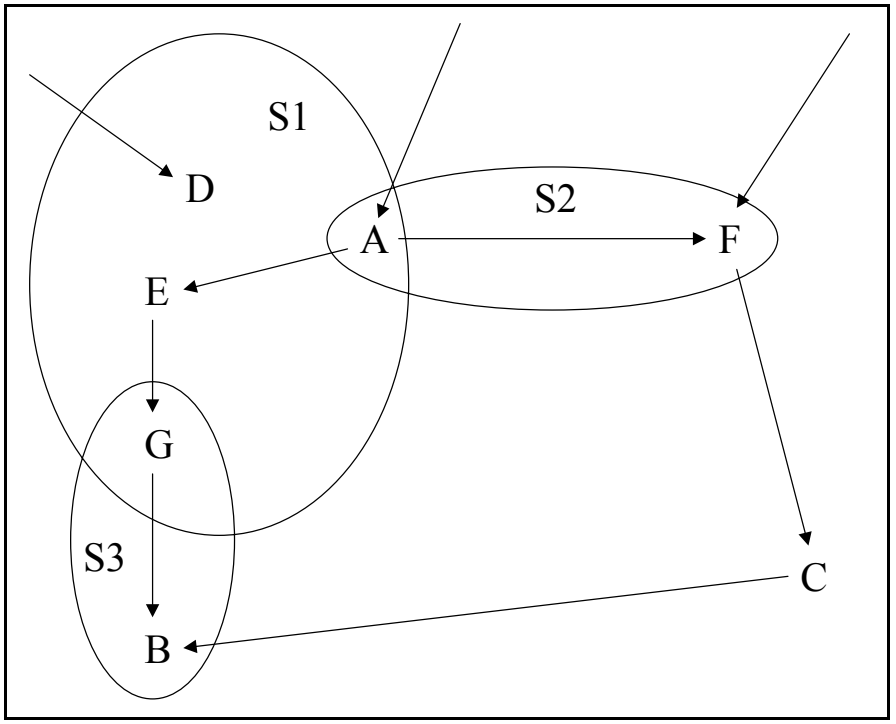


- Datenabhängigkeiten
  - Aktivität einer Sphäre produziert Daten, die von Aktivität einer anderen Sphäre konsumiert werden
  - Beispiel: *A* in *S1* erfolgreich, *B* in S3 hat Daten von *A* konsumiert, *backout(S1)* erforderlich $\rightarrow$ *backout(S3)* erforderlich
  - Reihenfolge: *backout(S3)* $\rightarrow$ *backout(S1)*

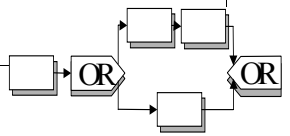


Kompensationssphären: Kaskadierung (2)

- Falls mehrere Sphären S_1 ,..., S_n gemeinsame Aktivitäten besitzen: Bei welchen Initial-Knoten wird Kontrollfluß nach kaskadierendem Backout wieder aufgenommen?
- Problem: Initial-Knoten der einen Sphäre sind von Initial-Knoten einer anderen Sphäre aus erreichbar
- Wiederaufnahme bei allen de-facto Initial-Knoten kann zu unerwünschtem wiederholten Ausführen bestimmter Aktivitäten führen
- Daher nur Berücksichtigung von Knoten, die *nicht* von Initial-Knoten anderer Sphären aus erreichbar sind

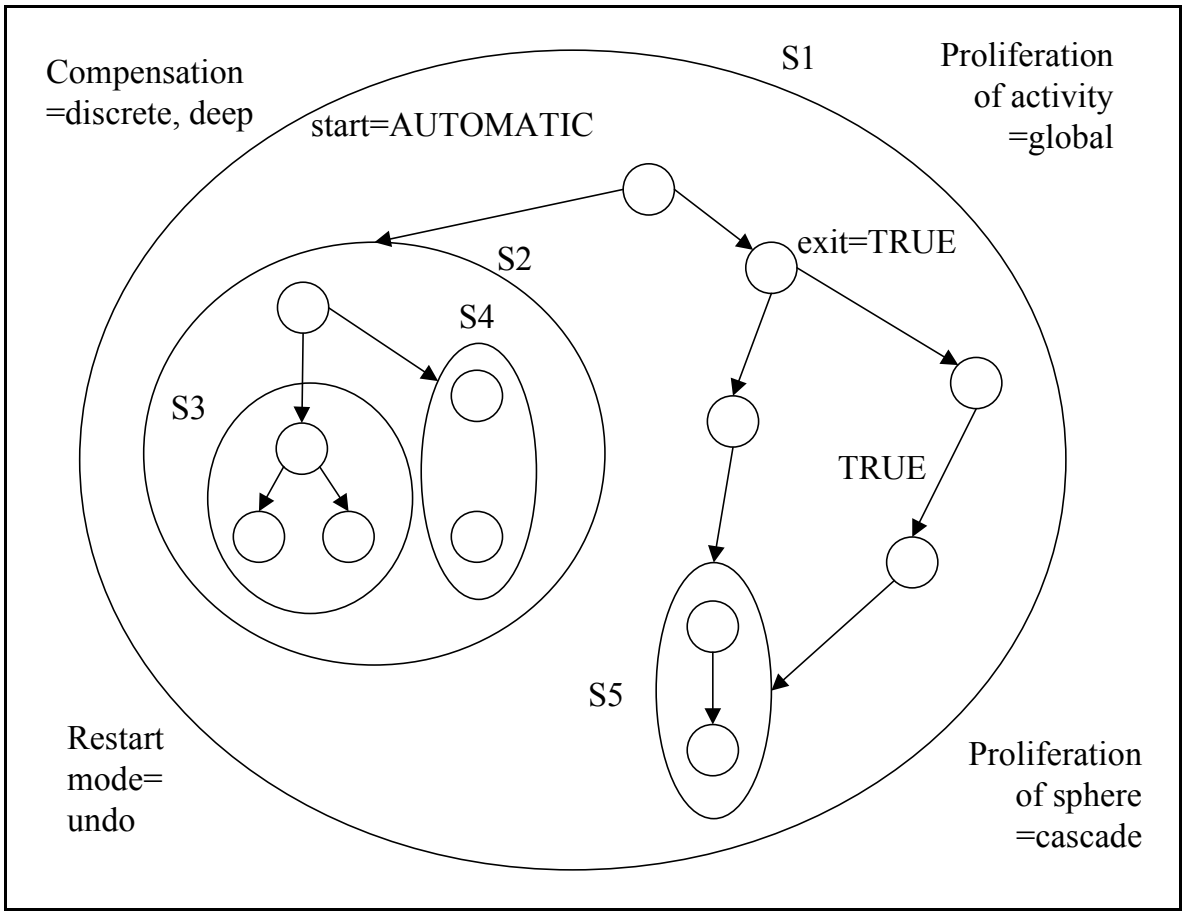


- Beispiel:
  - $(S_1)_{formal\text{-}initial} \cup (S_2)_{formal\text{-}initial} \cup (S_3)_{formal\text{-}initial} = \{A,D\} \cup \{A,F\} \cup \{G,B\} = \{A,B,D,F,G\}$
  - Falls alle aktiviert: Wiederaufnahme nur bei $\{A,D\}$



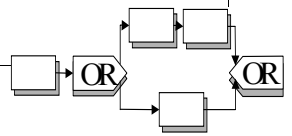
Kompensationssphären und Sagas

- Kompensationssphären inkorporieren Sagas und geschachtelte Sagas
- Beispiel: Geschachtelte Sagas
  - Root Saga = S1
  - S2 (S3, S4)
  - Proliferastion der Aktivitäten: global
  - Proliferation der Sphären: cascading
  - Backout-Modus: Undo
  - Tiefe und diskrete Kompensation



Kompensationssphären: Zusammenfassung und Diskussion

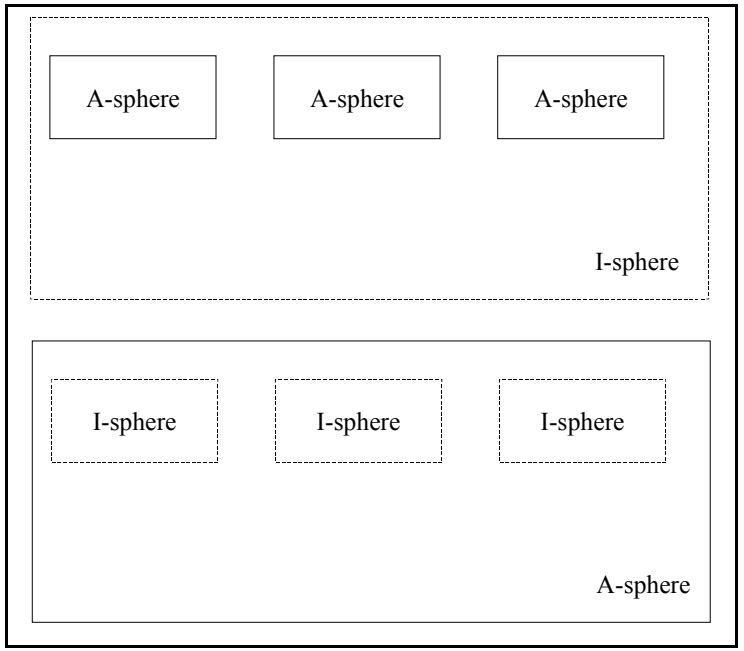
- Mächtiges Modell zur semantischen Partitionierung von Workflows aus Sicht der Fehlerbehandlung
- Hoher Spezifikationsaufwand zur Definitionszeit
- Bei Verwendung der vollen Modellkomplexität und großer Anzahl von Workflows sind Sphärenabhängigkeiten nur noch schwer zu überschauen
- Derzeit keine Implementierung vorhanden

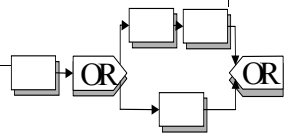


Atomaritäts- und Isolationssphären

(nach Schuldt & Schek, ETH Zürich)

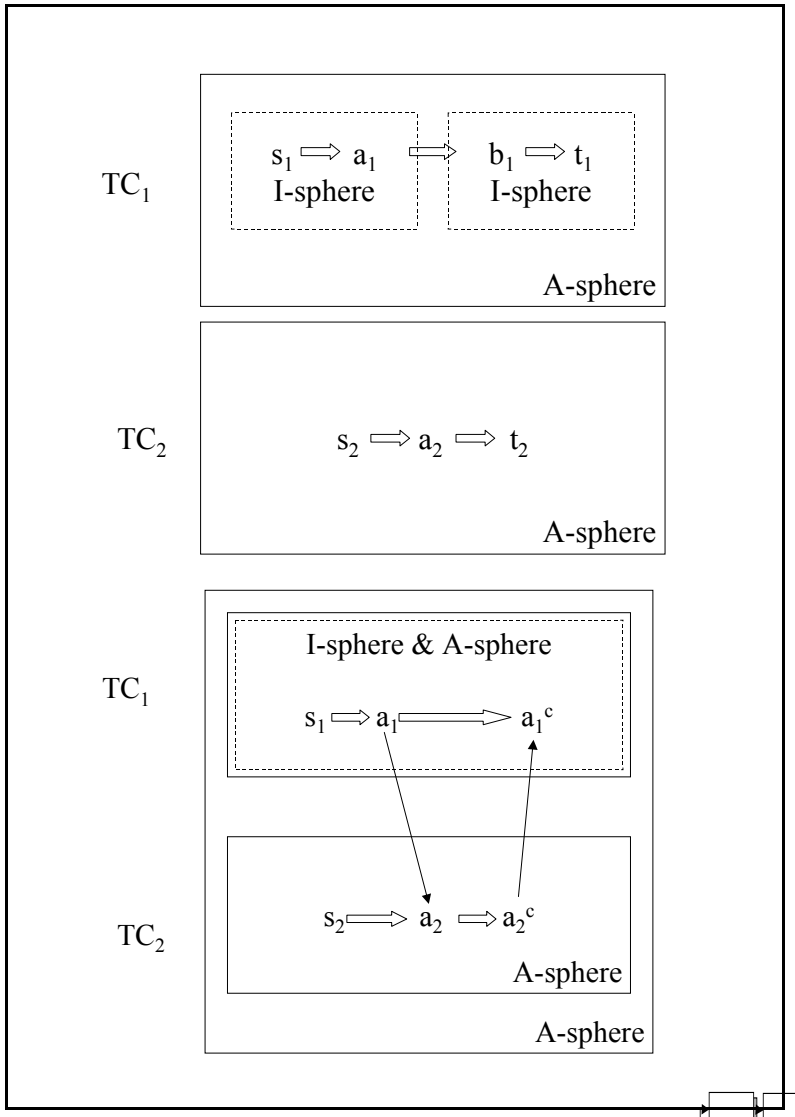
- Atomaritätssphäre
  - Gerichteter azyklischer Untergraph SA von TC
  - Bei Ausführung wird entweder ein Terminal-Knoten von SA erreicht, oder SA wird per Kompensation zurückgefahren
  - Entspricht Kompensationssphäre nach Leymann
- Isolationssphäre
  - Gerichteter azyklischer Untergraph SI von TC
  - Konfliktfunktion Con_{SI}: $Activities_{SI} \rightarrow P(Activities)$ liefert zu Aktivität A von SI die Menge der potentiellen Konflikt-Aktivitäten (bzgl. Zugriffen auf gemeinsame Datenobjekte)
- Konfliktbehandlung
  - $a_1 \in SI$ soll ausgeführt werden; $Con_{SI}(a_1) = \{a_2\}$
  - Kontextabhängige Überprüfung, ob Ausführung von a_2 in Konflikt zu a_1 steht
  - Falls ja: Serialisierung $a_1 \rightarrow a_2$
- Orthogonalität von A- und I-Sphären (Vollständige Entkopplung von Atomarität und Isolation)





A- und I-Sphären: Kritik

- Je nach Konstellation gleiche Problematik wie bei offen geschachtelten Transaktionen
- Beispiel
  - $a_1 \in TC_1$ und $a_2 \in TC_2$ greifen auf dasselbe Datenobjekt zu; $Con_{SI}(a_1) = \{a_2\}$
  - Nach Verlassen der I-Sphäre von a_1 kann a_2 ausgeführt werden
  - a_2 verarbeitet Ergebnis-Daten von a_1
  - Kurz vor Erreichen von t_1 Fehler, deswegen Kompensation u.a. von $a_1 \in TC_1$
  - Unklare Behandlung bzgl. der ausgeführten Aktivität a_2
  - Möglicher Ausweg: Ganz TC_1 ist I-Sphäre; damit aber Zusammenfall von A- und I-Sphäre



Workflow-Transaktionsmodelle: Zusammenfassung

Workflows

- sind langlebige Prozesse (Tage, Wochen, Monate)
- konkurrieren untereinander bzgl. Daten und Resourcen
- integrieren eine Vielzahl von manuellen und automatischen Arbeitsschritten

Beiträge der Workflow-orientierten Transaktionsmodelle

- Forward Recovery (dadurch möglichst geringer Verlust bzgl. bereits durchgeführter Arbeitsschritte)
- Lockerung der Isolation (Isolationsprädikate, Isolationssphären)
- Semantische Kompensation

Kritik

- Lockerung der Isolation bedingt im Fehlerfall u.U. schwer zu überschauende Workflow-Abhängigkeiten (z.B kaskadierende Kompensationen)
- Oft keine Evaluierung der Konzepte durch Implementierungen