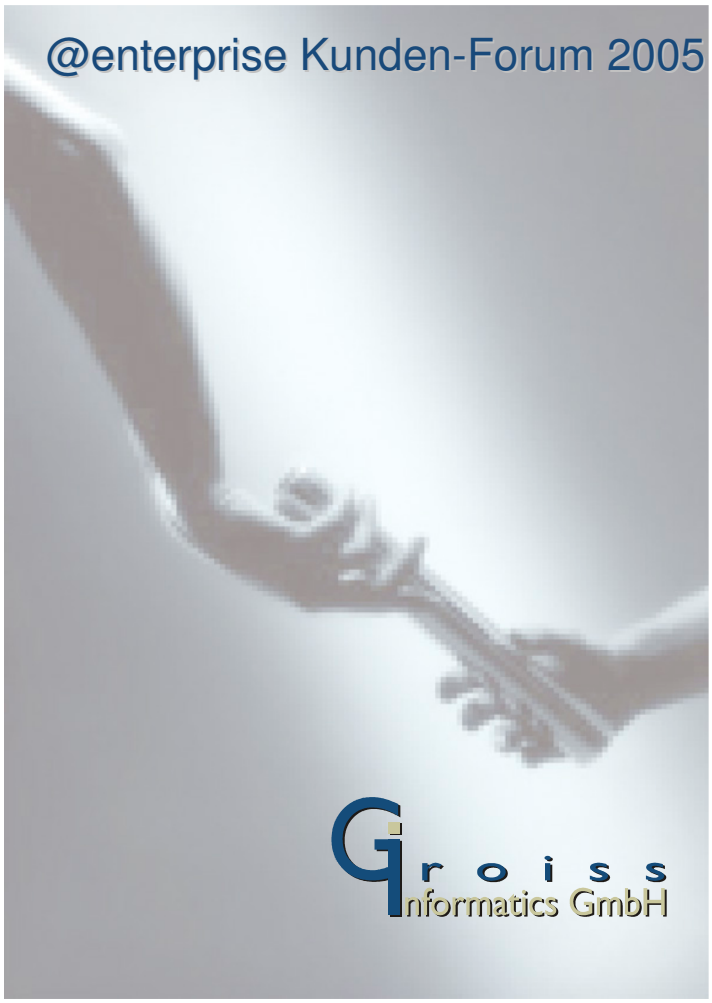
@enterprise Kunden-Forum 2005

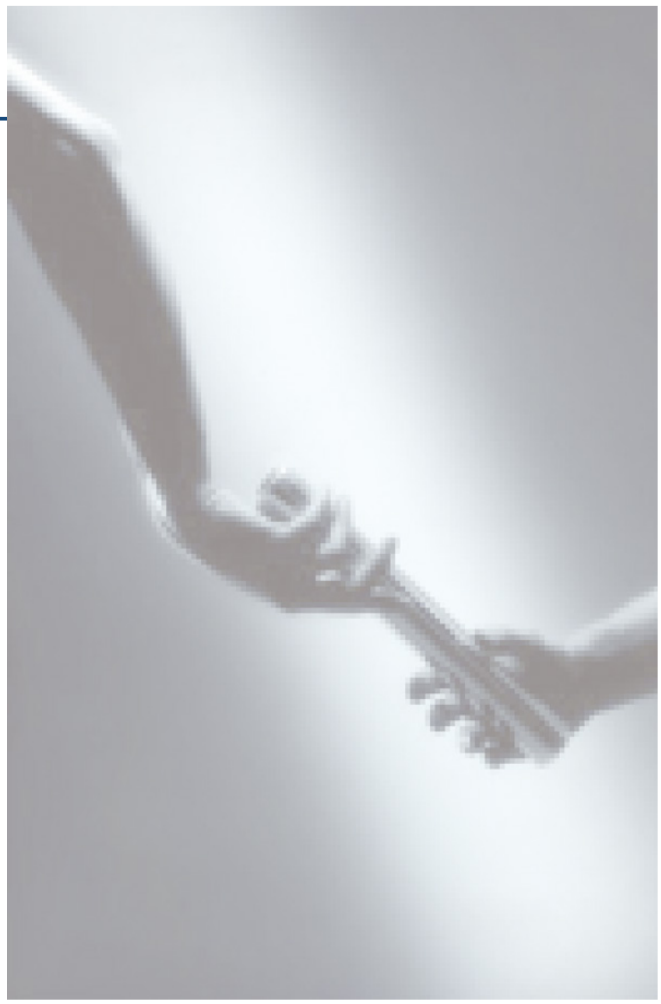
Workflow-Patterns

Michael Dobrovnik, Herbert Groiss



Inhalt

- Motivation und Überblick
- Control Patterns
- Resource Patterns
- Data Patterns



Motivation und Überblick

- Software-Patterns: Muster von wiederkehrenden Situationen
  - Meist komplexere Strukturen und Interaktionszusamenhänge zwischen mehreren beteiligten Artefakten
  - Popularisierung in der Informatik durch die Gang of Four [GOF: Gamma, Helm, Johnsson, Vlissides: Design Patterns – Elements of Reusable Object-Oriented Software, 1995]
  - vorgefertigte Abstraktionen

- Hilfreich bei:
  - Analyse:
    - Erkennen, Einordnen, Klassifizieren von Anforderungssituationen
  - Entwurf:
    - Schablone zur Problemlösung
    - Alternative Lösungsmöglichkeiten
  - Implementierung:
    - Schablone zur Problemlösung
  - Wartung:
    - bessere Qualität bewährter Artefakte
    - wieder Erkennen des größeren Zusammenhangs
  - Systembewertung
    - prinzipielle Mächtigkeit
    - Adäquatheit für die Anwendungssituation
  - Systemkonstruktion
    - "Komplettheit"

Genese der Patterns

- van der Aalst, et. al.:
  - Postulierung "typischer" Anwendungsmuster
  - Beschreibung, Untersuchung der (unmittelbaren) Umsetzbarkeit mit kommerziellen Systemen

- 2000: Workflow (Control) Patterns
  - W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Distributed and Parallel Databases, 14(3), pages 5-51, July 2003.
  - (lag bereits seit 2000 als Technical Report vor)

- 2004: Workflow Data Patterns
  - N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. QUT Technical report, FIT-TR-2004-01, Queensland University of Technology, Brisbane, 2004.

- 2004: Workflow Resource Patterns
  - N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven, 2004.

www.workflowpatterns.com

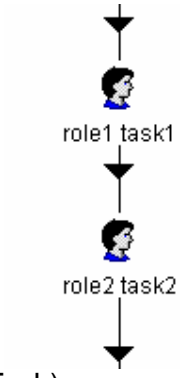
Control Patterns

- Kontrollflusskonstrukte (20 Stück)
- Behandeln Verhaltensaspekt des Workflows
  - Ablaufreihenfolge
  - Bedingungen für Ausführung
  - Wiederholung
  - Parallelitäten
  - Alternativen
  - Entscheidungen
- Einteilung in:
  - Basic Control Flow Patterns
  - Advanced Branching and Synchronization Patterns
  - Structural Patterns
  - Patterns involving Multiple Instances
  - State-based Patterns
  - Cancellation Patterns

Basic Control Flow Patterns (I):

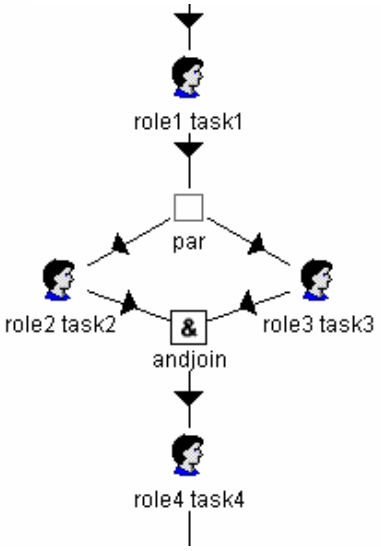
- CP-1: Sequence
  - Einfache Abfolge

```
role1 task1();
role2 task2();
```



- CP-2: Parallel Split (AND-split, Fork)
  - Splittung in mehrere Threads

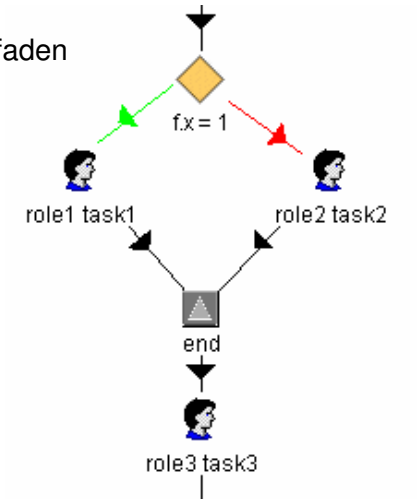
```
role1 task1();
andpar
  role2 task2();
  |
  role3 task3();
end;
role4 task4();
```



Basic Control Flow Patterns (II):

- CP-3: Synchronization (AND-join)
  - Zusammenführen mehrerer Threads
  - siehe CP-2
- CP-4: Exclusive Choice (XOR-split)
  - Auswahl einer von mehreren alternativen Pfaden

```
if (f.x = 1) then
  role1 task1();
else
  role2 task2();
end;
role3 task3();
```

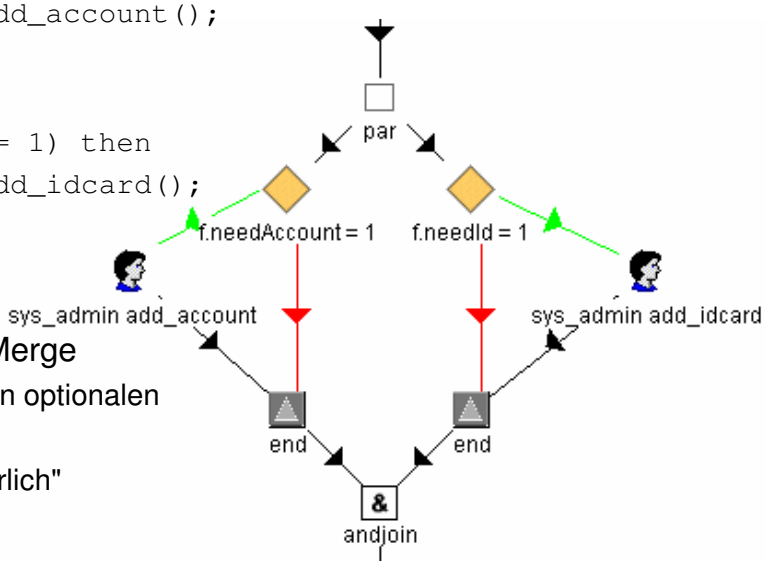


- CP-5: Simple Merge (XOR-Join)
  - end alternativer Konstrukte

Advanced Branching & Synchronization Patterns (I):

- CP-6: Multi-Choice (OR-split)
  - ein oder mehrere Nachfolger

```
andpar
  if (f.needAccount = 1) then
    sys_admin add_account();
  end
|
  if (f.needId = 1) then
    sys_admin add_idcard();
  end;
end;
```

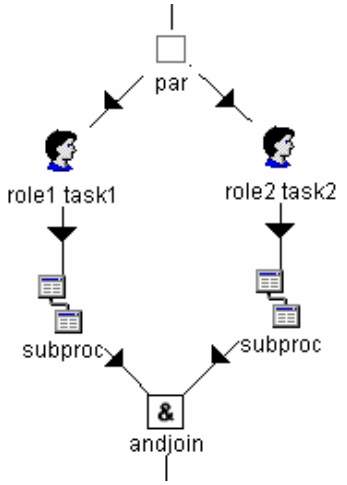


- CP-7: Synchronizing Merge
  - Zusammenführen von optionalen Alternativen
  - in @enterprise "natürlich"

Advanced Branching & Synchronization Patterns (II):

- CP-8: Multi-Merge
  - Instanzierung der Nachfolgeraktivität **pro Thread**

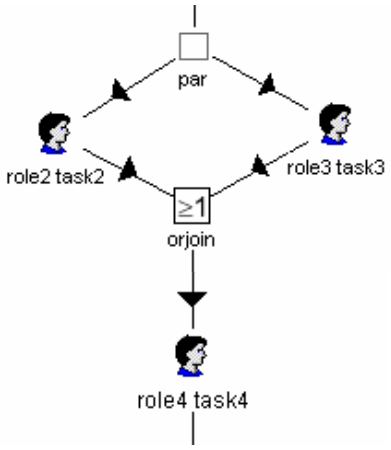
```
andpar
  role1 task1();
  call subproc();
|
  role2 task2();
  call subproc();
end;
```



Advanced Branching & Synchronization Patterns (III):

- CP-9: Discriminator
  - Ausführung der Nachfolgeaktivität exakt einmal (auch bei mehreren parallelen Vorgängeraktivitäten)

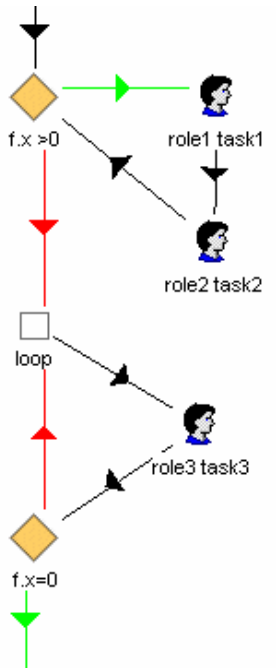
```
orpar
  role2 task2();
|
  role3 task3();
end;
role4 task4();
```



Structural Patterns:

- CP-10: Arbitrary Cycles (Loop, Iteration)
  - wiederholte Ausführung von Aktivitäten

```
while (f.x > 0) do
  role1 task1();
  role2 task2();
end;
repeat
  role3 task3();
until (f.x = 0);
```

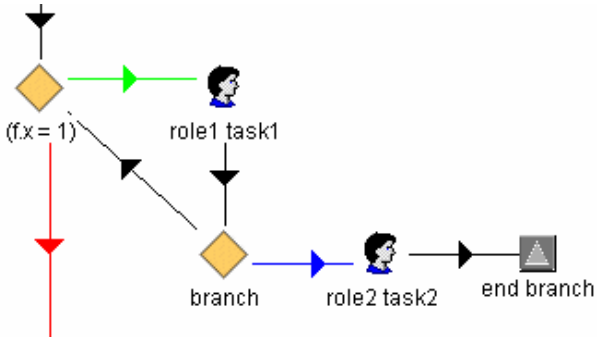


- CP-11: Implicit Termination
  - Workflow terminiert, wenn alle Aktivitäten abgeschlossen sind.
  - in @enterprise "natürlich"

Patterns involving Multiple Instances (I):

- CP-12: Multiple Instances without Synchronization (spawn)
  - Generierung von unabhängigen Nachfolger(threads)

```
while (f.x = 1) do
  role1 task1()
  branch
    role2 task2();
  end;
end;
```



Patterns involving Multiple Instances (II):

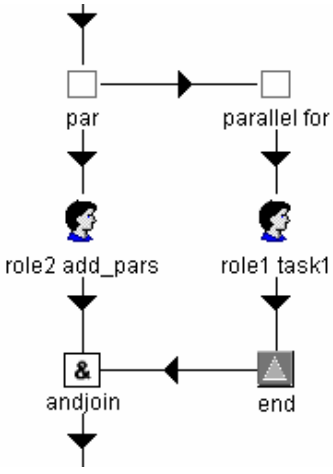
- CP-13: Multiple Instances with a Priori Design Time Knowledge
  - n-fache Parallelität & Synchronisation
  - in @enterprise direkt als ANDPAR realisierbar
- CP-14: Multiple Instances with a Priori Runtime Knowledge
  - n-fache Parallelität, zur Verzweigungszeit bekannt

```
parallel for review in mainform.1 do
  review.reviewer task1(review);
end;
role2 task2();
```

Patterns involving Multiple Instances (III):

- CP-15: Multiple Instances without a Priori Runtime Knowledge
  - Anzahl der parallelen Zweige zur Verzeigungszeit nicht bekannt
  - Kombination aus p*arfor* und *andpar*
    - im (and)parallelen Zweig kann man über API oder Taskfunktion neue Threads ins parfor-Konstrukt einfügen

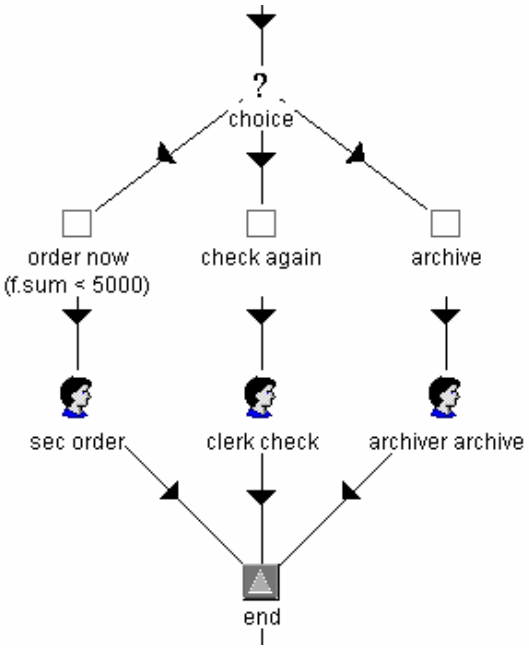
```
andpar
  role2 add_pars(f);
|
  parallel for x in f.1 do
    role1 task1(x);
  end;
end;
```



State-Based Patterns (I):

- CP-16: Deferred Choice
  - Angebot mehrerer alternativer Nachfolgemöglichkeiten, von denen aber nur eine ausgeführt wird.
  - Realisierung direkt mit *choice*

```
choice
"order now", f.sum < 5000:
  sec order(f);
"check again":
  clerk check(f);
"archive":
  archiver archive(f);
end;
```

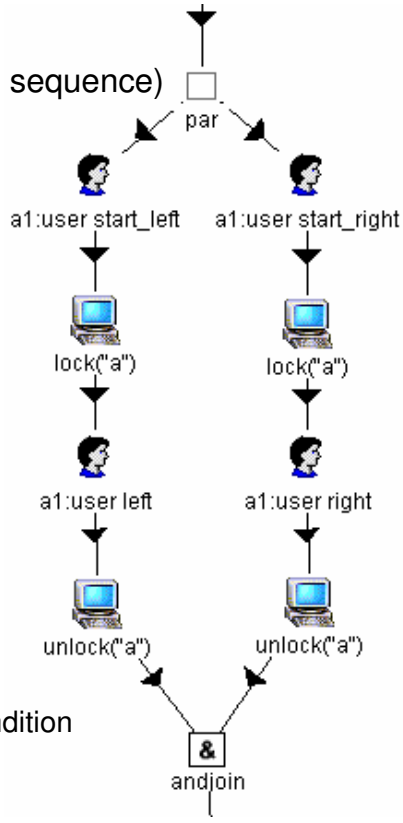


State-Based Patterns (II):

- CP-17: Interleaved parallel Routing (unordered sequence)
  - durchlaufen von mehreren Nachfolgeaktivitäten in einer beliebigen, aber sequenziellen Folge

```
andpar
  a1:user start_left(f);
  system SystemAction.lock("a");
  a1:user left(f);
  system SystemAction.unlock("a");
|
  a1:user start_right(f);
  system SystemAction.lock("a");
  a1:user right(f);
  system SystemAction.unlock("a");
end;
```

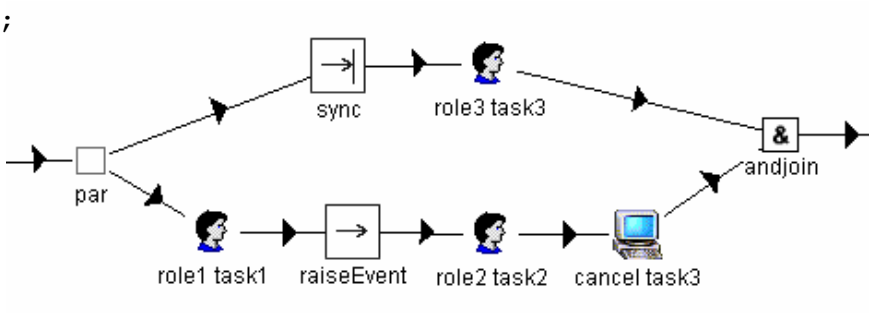
  - lock/unlock alternativ in Preprocessing / Postcondition



State-Based Patterns (III):

- CP-18: Milestone
  - Aktivität erst bei einem bestimmten Prozessfortschritt ausführbar
  - Aktivität ab einem bestimmten Prozessfortschritt nicht mehr ausführbar

```
andpar
  role1 task1();
  raiseEvent(e1, current_tx);
  role2 task2();
  system SystemAction.cancelActivity("task3");
|
  sync(e1,com.groiss.event.EventHandler);
  role3 task3();
end;
```



Cancellation Patterns (I):

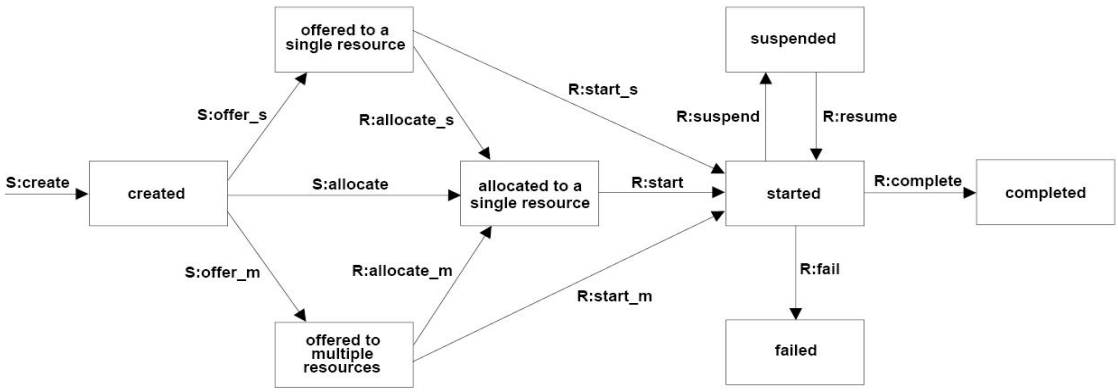
- CP-19: Cancel Activity
  - Abbruch einer Aktivität, Fortsetzen bei Nachfolgeraktivität
  - Über Funktion ***cancel_task*** (über API bzw. als Task-Funktion)
- CP-20: Cancel Case
  - Abbruch einer ganzen Prozessinstanz
  - Über Administrative Funktion bzw. über API

Resource Patterns

- behandeln Organisationsaspekte des Workflow(-Systems) (43 Stück)
  - Zuordnung von Activities zu Resources
  - Resource ist im wesentlichen ein User
- Einteilung in:
  - Creation Patterns
  - Push Patterns
  - Pull Patterns
  - Detour Patterns
  - Auto-Start Patterns
  - Visibility Patterns
  - Multiple Resource Patterns

Resource Patterns

- Orientiert sich am Lebenszyklus eines Work Items



  - R: Resource
  - S: System

RP: Creation Patterns (I)

- RP-1: Direct Allocation
  - Zuordnung zu einem Akteur zu Design-Time
- RP-2: Role-Based Allocation
  - Zuordnung zu einer Rolle
- RP-3: Deferred Allocation
  - Entscheidung über Resource erst zur Laufzeit
  - Akteur aus Formularfeld
- RP-4: Authorization
  - Einschränkung über RechteSystem, auch wirksam bei Delegation
  - keine direkte Entsprechung in @enterprise
  - Setzen von Berechtigungen auf Formulare, die in Tasks verwendet werden
- RP-5: Separation of Duties
  - Zwei Tasks müssen von verschiedenen Akteuren bearbeitet werden (Vieraugenprinzip)
  - Realisierung über Prüfung in Preprocessing oder onTake Hook

RP: Creation Patterns (II)

- RP-6: Case Handling
  - Alle Tasks einer Prozessinstanz werden vom selben Akteur bearbeitet
  - in @enterprise über durchgängige Verwendung von Agentdescription der Form <label>:<user>
- RP-7: Retain Familiar
  - einige Tasks einer Prozessinstanz werden vom selben Akteur bearbeitet
  - in @enterprise über Agentdescription der Form <label>":user"
- RP-8: Capability Based Allocation
  - Welche Resource ist fachlich in der Lage, den Task zu übernehmen
  - Akteur setzen in Preprocessing-Methode auf der Basis der Falldaten und eines (erweiterten) Resourcenmodells
- RP-9: History-Based Allocation
  - Zuordnung auf der Basis der Geschichte der der Resource; hat X schon jemals sowas gemacht?
  - Akteur setzen in Preprocessing-Methode auf der Basis der Falldaten und historischer Resourcendaten

RP: Creation Patterns (III)

- RP-10: Organizational Allocation
  - Zuordnung auf der Basis der Kombination Organisationseinheit und Resource
  - in @enterprise direkt unterstützt in der Form <orgunit>!<role>
- RP-11: Automatic Execution
  - Ohne menschlichen Akteur
  - @enterprise Systemschritt / Batch-Job

RP: Push Patterns (I)

- RP-12: Distribution by Offer – Single Resource
  - Einzelne Resource bekommt Task, hat aber Möglichkeit die Arbeit "abzulehnen"
  - "heads up" processing
  - Zurückgehen bzw. Akteur ändern
  - explizitere Form: Schleife und Allokationsschritt

```
repeat
  manager select_worker(f)
  f.user offer_work(g)
until f.taken = 1
f.user do_work(g)
```

- RP-13: Distribution by Offer – Multiple Resources
  - In den Rollenarbeitskorb stellen
- RP-14: Distribution by Allocation – Single Resource
  - direkt in den persönlichen Arbeitskorb
  - Task darf nicht abgelehnt werden
  - "heads down" processing

RP: Push Patterns (II)

- RP-15: Random Allocation
  - Zufällige Zuordnung zu Akteuren (beispielsweise einer der Inhaber einer Rolle)
  - Akteur setzen im Preprocessing
- RP-16: Round Robin Allocation
  - Zuordnung Reihum
  - wie RP-15
- RP-17: Shortest Queue
  - Zuordnung nach niedrigster Arbeitslast
  - wie RP-15 aber: Faire Bewertung durch das System??

RP: Push Patterns (III)

- RP-18: Early Allocation
  - Vorabinformation über zu erwartende Arbeit
  - Realisierung über Branch
  - *Personal Scheduling*
- RP-19: Distribution on Enablement
  - Zuordnung der Tasks im Moment Ihrer Erzeugung
  - Standard in @enterprise
- RP-20: Late Distribution
  - Zuordnung zu konkreter Resource wird verzögert (Vermeidung exzessiv langer Worklists)
  - Realisierung über eigenen Allokationsschritt

```
dispatcher select_worker(f)
f.user do_work();
```

RP: Pull Patterns (I)

- RP-21: Resource-Initiated Allocation
  - Resource entscheidet über konkrete Tasks die bearbeitet werden (evtl. auch erst später)
  - Take von Rollenarbeitskorb
- RP-22: Resource-Initiated Excecution - Allocated Work Item
  - Resource entscheidet über konkreten Arbeitsbeginn für einen Task
  - Beginn des Formularausfüllens
  - Evtl. erweiterte Stati mitschreiben
  - Unterbrechungen sind möglich
- RP-23: Resource-Initiated Execution – Offered Work Item
  - Unmittelbarer Beginn der Arbeit nach Auswahl aus Rollenarbeitskorb ohne Zwischenschritt
  - Realisierung über eigene Funktion die im Rollen-AK zur Verfügung steht

RP: Pull Patterns (II)

- RP-24: System Determined Work Queue Content
  - System bestimmt Ordnung und Umfang der angezeigten Worklist
  - Neue Worklist implementieren (ohne Sort und Filter für den Benutzer)
- RP-25: Resource Determined Work Queue Content
  - User bestimmt Reihenfolge der Items, Art der Darstellung und Filterung
  - Standard in @enterprise
- RP-26: Selection Autonomy
  - Benutzer bestimmt welche Arbeit er als nächstes erledigt
  - Ggf. Darstellung der Dringlichkeit

RP: Detour Patterns (I)

- RP-27: Delegation
  - Weitergabe an anderen Akteur initiiert durch aktuellen Akteur
  - Akteur Ändern Funktion in @enterprise
- RP-28: Escalation
  - Entzug des Items und Zuordnung zu einem anderen Akteur durch das System
  - @enterprise Eskalationsmechanismen
- RP-29: Deallocation
  - Auflösung der Zuordnung zu einem Item
  - @enterprise: Zurücklegen
- RP-30: Stateful Reallocation
  - Zuordnung zu anderer Resource ohne bereits geänderte Daten zu verlieren
  - Delegation: Standard in @enterprise
- RP-31: Stateless Reallocation
  - Zuordnung zu anderer Resource und Wiederherstellung des Zustandes vor der letzten Zuordnung (alle Änderungen des Aktuellen Akteurs werden rückgängig gemacht
  - nicht direkt unterstützt

RP: Detour Patterns (II)

- RP-32: Suspension / Resumption
  - Unterbrechung der Bearbeitung eines Tasks auf
  - Standard in @enterprise: Wiedervorlage
- RP-33: Skip
  - Resource kann weiterleiten, ohne wirklich am Item zu Arbeiten
  - ist Standard in @enterprise
  - Vermeidung mit Postconditions
  - Andere Form: Markierung eines Tasks als "Skipable"
    - wenn zur Laufzeit kein Akteur gesetzt wurde, wird der Task übersprungen
- RP-34: Redo
  - Wiederholtes Ausführen eines bereits abgeschlossenen Tasks
  - in @enterprise: Zurückgehen
- RP-35: Pre-Do
  - Vorausfüllen von zu erwartenden Tasks
  - Realisierung: Über Branches bzw. Parallelitäten
  - Achtung auf zwischenzeitliche Änderungen

RP: Auto-start Patterns

- RP-36: Commencement on Creation
  - Unmittelbare Bearbeitung nach Erzeugung
  - Beim Prozessstarten / Formularanzeige
- RP-37: Commencement on Allocation
  - Start unmittelbar nach Zuordnung
  - verwandt: Autotake in @enterprise
- RP-38: Piled Execution
  - Stapelabarbeitung ähnlicher Fälle, ohne langwierige Selektion (Bearbeitung aller zwischenzeitlich gestellten Urlaubsanträge)
  - keine direkte Unterstützung, aber Realisierung im Formular: "Speichern und nächstes"
- RP-39: Chained Execution
  - Unmittelbarer Start der Folgeaktivität mit der selben Resource
  - Realisierung mittels <label>x:user

RP: Visibility Patterns

- RP-40: Configurable Unallocated Work Item Visibility
  - Einschränkung der Sichtbarkeit von Erzeugten Tasks
  - in @enterprise nicht direkt unterstützt
- RP-41: Configurable Allocated Work Item Visibility
  - Einschränkung der Sichtbarkeit von Zugeordneten Tasks
  - in @enterprise nicht direkt unterstützt

RP: Multiple Resource Patterns

- RP-42: Simultaneous Execution
  - Mehrere Work-Items gleichzeitig bearbeitbar
  - nicht Standard in @enterprise
  - Realisierbar über Formularanzeige in eigenen Window
  - Worklist mit integrierten bearbeitbaren Formularfeldern
- RP-43: Additional Resources
  - Teamarbeit an einem Item
  - Groupware-Funktionalität
  - keine Unterstützung in @enterprise

Data Patterns

- behandeln Informationsaspekte des Workflow(-Systems) (39 Stück)
- Einteilung in:
  - Data visibility Patterns
    - Ausmaß und Art der Sichtbarkeit von Daten in Prozesskomponenten
  - Data interaction Patterns
    - Art der Datenkommunikation zwischen aktiven Elementen in einem Workflow
  - Data Transfer Patterns
    - Konkrete Art der Transfermechanismen der Daten
  - Data-based routing Patterns
    - Zusammenspiel zwischen Daten und Kontrollstrukturen

Data visibility Patterns (I):

- DP-1: Task Data
  - nur innerhalb einer Task Instanz
  - Realisierung durch entsprechende Angabe von Formularen im "Taskaufruf" eines Workflows

```
role1 task1(f,g,h)
```

Allerdings:

- nicht verhinderbar, dass die Forms in anderen Tasks verwendet werden.
- Zugriff über API nicht eingeschränkt

- DP-2: Block Data
  - nur innerhalb eines Subprozesses
  - Realisierung über lokales Formular im Subprozess
- DP-3: Scope Data
  - Scope ist beliebiges Subset der Tasks eines Workflows
  - Siehe DP-1

Data visibility Patterns (II):

- DP-4: Multiple Instance Data
  - Datenstrukturen können auf parallele Zweige aufgeteilt werden
  - Realisierung mittels parfor (vgl. CP-14 und CP-15)
- DP-5: Case Data
  - Fallspezifische Daten sichtbar auch über Subworkflows hinweg
  - in @enterprise müssen Formulare an Subworkflows explizit übergeben werden
  - Realisierung ggf. über Dokumentenformulare
- DP-6: Workflow Data
  - Sichtbarkeit über alle Instanzen eines Prozesstyps
  - Verwendung der Datenbank / Allgemeine Dokumentenordner
- DP-7: Environment Data
  - Zugriff auf Daten aus der Umgebung des WfMS
  - Realisierung über Systemschritte, Task-Funktionen;
  - unterschiedlichste Kommunikationsmechanismen (HTTP, Mail, ...)

Data interaction Patterns: Internal (I)

- DP-8: Task to Task
  - Weitergabe von Daten an Nachfolgetasks
  - Realisierung siehe CP-1
- DP-9: Block Task to Sub-Workflow Decomposition
  - Weitergabe an Subworkflow
  - Realisierung über Formular-Passing
- DP-10: Sub-Workflow Decomposition to Block Task
  - Rückgabe aus Subworkflow
  - Realisierung über Formular-Passing
- DP-11: to Multiple Instance Task
  - aufgeteilte Weitergabe an mehrere Nachfolgetasks
  - mitels parfor vgl. DP-4

Data interaction Patterns: Internal (II)

- DP-12: from Multiple Instance Task
  - Zusammenfassung der Daten aus mehreren Vorgängertasks
  - implizit bei der Realisierung mit parfor, da jede der multiplen Taskinstanzen eine Referenz auf eines der Datenelemente hat
- DP-13: Case to Case
  - Weitergabe an anderen laufenden Workflow, Bezug auf (beendeten) Workflow
  - (gegenseitige) Referenzierung nötig, oder Besitz von Datenschlüsselwerten
  - Zugriff auf Formulare anderer Prozesse,
  - Verwendung von Dokumentenformularen in gemeinsamen Ordnern
  - Verwendung der DB

Data interaction Patterns: External (I)

- DP-14: Task to Environment – Push-Oriented
  - Systemschritt, Task-Funktion, Seiteneffekt
  - Schreiben auf File, in DB, senden von Mail, HTTP Request, XML-RPC, Wf-XML Notify und ProcessStateChanged
- DP-15: Environment to Task – Pull Oriented
  - Systemschritt, Task-Funktion
  - HTTP Request, Zugriff über API, DB, XML-RPC
- DP-16: Environment to Task – Push Oriented
  - Finden der geeigneten Instanz und deren Daten
  - Event-Handling Mechanismen (asynchron zur Prozessausführung)
- DP-17: Task to Environment – Pull-Oriented
  - Finden der geeigneten Instanz
  - API, DB, Wf-XML GetProcessInstanceData

Data interaction Patterns: External (II)

- Auf Ebene einer Prozessinstanz
  - DP-18: Case to Environment – Push Oriented
  - DP-19: Environment to Case – Pull Oriented
  - DP-20: Environment to Case – Push Oriented
  - DP-21: Case to Environment – Pull Oriented
- Auf Ebene aller Instanzen eines Prozesstyps
  - DP-22: Workflow to Environment – Push Oriented
  - DP-23: Environment to Workflow – Pull Oriented
  - DP-24: Environment to Workflow – Push Oriented
  - DP-25: Workflow to Environment – Pull Oriented

Data transfer Patterns

- DP-26: Data Transfer by Value – Incoming
  - Daten werden in @enterprise als Referenzen weitergegeben
  - Transformationen nur über explizite Systemschritte zwischen den Tasks
- DP-27: Data Transfer by Value – Outgoing
  - vgl. DP-26
- DP-28: Copy In / Copy Out
  - vgl. DP-26
- DP-29: Transfer by Reference – Unlocked
  - Objekt darf OptimisticLocking nicht implementieren
- DP-30: Transfer By Reference – With Lock
  - StandardFall in @enterprise; mittels OptimisticLocking Strategie
  - oder mittels explizitem Lock (vgl. CP-17)
- DP-31: Data Transformation- Input
  - vgl. DP-26
- DP-32: Data Transformation – Output
  - vgl. DP-26

Data-based Routing Patterns (I):

- DP-33: Task Precondition – Data Existence
  - Preprocessing Methode
  - Sync-Konstrukt, warte auf Event
- DP-34: Task Precondition – Data Value
  - Preprocessing – Methode
  - Sync mit Event-Handler (der kann Bedingung auswerten)
  - Choice (CP-16)
- DP-35: Task Postcondition – Data Existence
  - Task-Postcondition in @enterprise
- DP-36: Task Postcondition – Data Value
  - Task-Postcondition in @enterprise

Data-based Routing Patterns (II):

- DP-37: Event based Task Trigger
  - Anstoßen eines Tasks über asynchronen Event
  - Event-Mechanismus
  - Mail Senden / Prozess Starten
  - API
- DP-38: Data-based Task Trigger
  - Sync mit Event-Handler (vgl. DP-34)
- DP-39: Data-based Routing
  - if mit Expression über Daten

Resümee

- Patterns auch im Bereich von Workflow-Management hilfreich bei:
  - Analyse:
    - Erkennen, Einordnen, Klassifizieren von Anforderungssituationen
  - Entwurf:
    - Schablone zur Problemlösung
    - Alternative Lösungsmöglichkeiten
  - Implementierung:
    - Schablone zur Problemlösung
  - Wartung:
    - bessere Qualität bewährter Artefakte
    - wieder Erkennen des größeren Zusammenhangs
  - Systembewertung
    - prinzipielle Mächtigkeit
    - Adäquatheit für die Anwendungssituation
  - Systemkonstruktion
    - "Komplettheit"