Service Interaction: Patterns, Formalization, and Analysis


prof.dr.ir. Wil van der Aalst
www.vdaalst.org

Joint work with Arjan Mooij, Christian Stahl, and Karsten Wolf
BEST: Berlin - Rostock - Eindhoven
Service Technology Program

http://www2.informatik.hu-berlin.de/top/best/
Outline

• Introduction to Service Interaction
• Workflow and Service Interaction Patterns
• Challenging Analysis Questions
• A "Crash Course" in Petri Nets
• Exposing Services
• Replacing and Refining Services
• Integrating Services Using Adapters
• Service Mining
• Conclusion
Introduction to Service Interaction
Service-Oriented: Basic Idea

- Service A
  - "buy side"
  - Service requester
- Service B
  - "sell side"
  - Service provider

Actions:
- Invoke
- Receive
- Reply
Service Networks

- Service A: service requester
- Service B: service provider & requestor
- Service C: service providers
- Service D: service providers

Connections:
- Service A to Service B: "buy side" to "sell side"
- Service B to Service C: "buy side" to "sell side"
Choreography
Orchestration

service A

service B

service C

service D

orchestration
Workflow?

*workflow in the classical sense*

Some processes just work better than others
Some Terminology

Important assumption: asynchronous communication.
Interaction is a source of errors!
deadlock
restaurant is "uncontrollable"*

customer is "controllable" but will never get any food

*by any service with only dead final markings
Workflow and Service Interaction Patterns
Workflow Patterns Initiative

• Started in 1999, joint work TU/e and QUT

• Objectives:
  • Identification of workflow modeling scenarios and solutions
  • Benchmarking
    − Workflow products (MQ/Series Workflow, Staffware, etc)
    − Proposed standards for web service composition (BPML, BPEL)
    − Process modeling languages (UML, BPMN)
  • Foundation for selecting workflow solutions

• Home Page: www.workflowpatterns.com

• Primary publication:

• Evaluations of commercial offerings, research prototypes, proposed standards for web service composition, etc
## Workflow Patterns Framework

<table>
<thead>
<tr>
<th>Time</th>
<th>Control-flow P:s</th>
<th>Resource P:s</th>
<th>Data P:s</th>
<th>Revised Control-flow P:s</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>20</td>
<td>-</td>
<td>-</td>
<td>43</td>
</tr>
<tr>
<td>2003</td>
<td></td>
<td>-</td>
<td>-</td>
<td>43</td>
</tr>
<tr>
<td>Jun 2005</td>
<td></td>
<td>-</td>
<td>-</td>
<td>43</td>
</tr>
<tr>
<td>Oct 2005</td>
<td></td>
<td></td>
<td>40</td>
<td>43</td>
</tr>
<tr>
<td>Sep 2006</td>
<td></td>
<td></td>
<td></td>
<td>43</td>
</tr>
</tbody>
</table>

### Contributors

- W. van der Aalst
- A. ter Hofstede
- B. Kiepuszewski
- A. Barros
- N. Russell
- W. van der Aalst
- A. ter Hofstede
- D. Edmond
- W. van der Aalst
- N. Mulyar

### Patterns

- Control-flow P:s: 20
- Resource P:s: 43
- Data P:s: 40

### Details

- The ordering of activities in a process
- Resource definition & work distribution in a process
- Data representation and handling in a process

- 23 new patterns
- Formalised in CPN notation

### Conferences

- CoopIS’2000
- DAPD’2003
- CAiSE’2005
- ER’2005
- TR

Slides adopted from Nick Russell, et al.
## Workflow Patterns Framework

<table>
<thead>
<tr>
<th>Time</th>
<th>Control-flow P:s</th>
<th>Resource P:s</th>
<th>Data P:s</th>
<th>Evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>20</td>
<td>-</td>
<td>43</td>
<td>COSA, FLOWer, Eastman Meteor Mobile I-Flow Staffware InConcert</td>
</tr>
<tr>
<td>2003</td>
<td>20</td>
<td>43</td>
<td>-</td>
<td>Domino Workflow Visual Workflow Forte Conductor MQSeries/Workflow SAR R/3 Workflow Verve Workflow Changengine</td>
</tr>
<tr>
<td>Jun 2005</td>
<td>20</td>
<td>43</td>
<td>40</td>
<td>Staffware WebSphere MQ FLOWer COSA iPlanet</td>
</tr>
<tr>
<td>Oct 2005</td>
<td>20</td>
<td>43</td>
<td>40</td>
<td>Staffware MQSeries FLOWer COSA</td>
</tr>
</tbody>
</table>

**Languages Development:** YAWL/newYAWL
Service Interaction Patterns

- Basic Service Interaction Patterns (SIP-1, ..., SIP-15)
- Correlation Patterns (SIP-16, ..., SIP-23)
- Anti-Patterns (AP-1, AP-2, and AP-3)
Send Patterns

SIP-1 Send pattern

SIP-2 Pre-Blocking Send pattern

SIP-3 Post-Blocking Send pattern
Receive Patterns

SIP-4 Receive pattern

SIP-5 Lossy Receive pattern
Concurrent Send/Receive Patterns

SIP-6 Concurrent Send pattern

SIP-7 Concurrent Receive pattern
Choice Patterns

SIP-8 Sending Choice pattern

SIP-9 Receiving Choice pattern

SIP-10 Internal Choice pattern
Choice With a Follow-Up Patterns (1/2)

SIP-11 Sending Choice Receiving Follow-Up pattern

SIP-12 Receiving Choice Sending Follow-Up pattern
Choice With a Follow-Up Patterns (2/2)

- SIP-13 Sending Choice Sending Follow-Up
- SIP-14 Receiving Choice Receiving Follow-Up, and
- SIP-15 Internal Choice Sending Follow-Up.

See problem?
Anti-Pattern AP-1: Internal Choice Receiving Follow-Up Anti-Pattern
Another variant of AP-1

send → skip → c1 → receive → c2 → p1, p2 → c3
Two Additional Variants of AP-1
Correlation
Correlation Send Patterns

SIP-16 Leading Correlated Send pattern

SIP-17 Following Correlated Send pattern
Anti-Pattern AP-2: Uncorrelated Send Anti-Pattern
Correlation Receive Patterns (1/2)

SIP-18 Leading Correlated Receive pattern

SIP-19 Following Correlated Receive pattern
Correlation Receive Patterns (2/2)

SIP-20 Learning Correlated Receive pattern

SIP-21 Creating Correlated Receive pattern
Anti-Pattern AP-3: Uncorrelated Receive Anti-Pattern
Composite Correlation Patterns (1/2)

SIP-22 Correlation Swap pattern

[from1:=me1] (me1,*)
send1
(me1,*) (from1,*,content1)

[me1=to2] [you1:=from2]
receive2
(from2,to2,content2)
(me1,you1)

[to3:=you1]
send3
(me1,you1)

[you2:=from1] [me2:=new()]
receive1
(me2,you2)

[from2:=me2] [to2:=you2]
send2
(me2,*)

[me2=to3]
receive3
(me2,*)
Composite Correlation Patterns (1/2)

SIP-16 Leading Correlated Send pattern
[from1:=me1] send1 (me1,*) (from1,*,content1)
[SIP-17 Following Correlated Send pattern]
[to3:=you1] send3 (me1,you1)

SIP-20 Learning Correlated Receive pattern
[me1=to2] [you1:=from2] receive2 (from2,to2,content2)
[SIP-18 Leading Correlated Receive pattern]
[me2=to3] receive3 (*,to3,content3)

SIP-21 Creating Correlated Receive pattern
[you2:=from1] [me2:=new()] receive1 (me2,you2)

SIP-22 Correlation Swap pattern
[from2:=me2] [to2:=you2] send2 (me2,*)

Composite Correlation Patterns (2/2)

SIP-23 Correlation Broker pattern
Composite Correlation Patterns (2/2)

SIP-16 Leading Correlated Send pattern
[from1:=me1]

SIP-19 Following Correlated Receive pattern
[x=from1]
[to1:=y]

SIP-17 Following Correlated Send pattern
[me2=to1]

SIP-18 Leading Correlated Receive pattern
[me2=to1]

SIP-23 Correlation Broker pattern
Recommended Reading (1/2)

Recommended Reading (2/2)

Challenging Analysis Questions
Questions Addressed in this Tutorial

1. **Exposing Services**
   - How to inform others about me such that cooperation is possible?
   - Two approaches: (a) expose own behavior and (b) provide operating guideline.

2. **Replacing and Refining Services**
   - How to replace or refine a service without introducing problems?
   - Inheritance, accordance, transformation rules, etc.

3. **Integrating Services Using Adapters**
   - How to resolve behavioral incompatibilities?
   - Adapter generation.

4. **Service Mining**
   - How to analyze the run-time behavior?
Additional Questions

• **Verification** (e.g., various types of soundness)
• **Controllability** (Is there a compatible partner?)
• **Instance migration** (Can I replace a service at run-time?)
• **Querying software repositories** (Is there a service that ... ?)
• **Similarity of services** (What is the least incompatible service? How many edit steps are needed to transform one into the other?)
• **How to generate/compose services** to meet specific requirements and goals?
Design-time analysis of processes

Linear algebraic analysis techniques

Markov chain analysis techniques

State-space analysis techniques

Simulation
From BPEL to Petri Nets and Back

• Feature complete mappings from BPEL to Petri nets:
  • WofBPEL (TU/e & QUT)
  • BPEL2oWFN (Rostock & Humboldt)
• Mappings from Petri nets to BPEL:
  • WorkflowNet2BPEL4WS (TU/e & Aarhus)
  • oWFN2BPEL (Rostock & Humboldt)
• Similar results hold for the BPMN, EPCs, etc.!
• Be critical! Not all reported results exist :-(
Recommended Reading (1/3)

- www.service-technology.org
Recommended Reading (2/3)


A "Crash Course" in Petri Nets
Rules

- Connections are directed.
- No connections between two places or two transitions.
- Places may hold zero or more tokens.
- First, we consider the case of at most one arc between two nodes.
Enabled

- A transition is enabled if each of its input places contains at least one token.
Firing

- An enabled transition can fire (i.e., it occurs).
- When it fires it consumes a token from each input place and produces a token for each output place.

```
wait -> enter
before -> make_picture
after -> leave

gone

occupied

fire

free

Etc.
```
Example: Traffic Lights
Example: Producers and Consumers

```
producer
  ↓
  free
  ↓
  start_production
  ↓
  producer
  ↓
  product
  ↓
  end_production

consumer
  ↓
  wait
  ↓
  start_consumption
  ↓
  product
  ↓
  end_consumption
```
Example: Four Philosophers
Definition 1 (Petri net). A Petri net $N = [P,T,F,m_0]$ consists of

- two finite and disjoint sets $P$ and $T$ of places and transitions,
- a flow relation $F \subseteq (P \times T) \cup (T \times P)$, and
- an initial marking $m_0$, where a marking is a mapping $m : P \to \mathbb{N}$. 
For the flow relation of a Petri net $N$ we introduce the following notation to denote the pre-set and the post-set of places and transitions. Let $x \in P \cup T$ be a node of $N$. Then, $\cdot x = \{y \mid [y, x] \in F\}$ denotes the pre-set of $x$ (i.e. all nodes $y$ that have an arc to $x$) and $x^\cdot = \{y \mid [x, y] \in F\}$ denotes the post-set of $x$ (i.e. all nodes $y$ with an arc from $x$ to $y$).
The dynamics of a Petri net $N$ is defined by the firing rule. The firing rule defines enabledness of Petri net transitions and their effects. A transition $t$ is enabled at a marking $m$ if there is a token on every place in its pre-set. The firing of an enabled transition $t$ yields a new marking $m'$, which is derived from its predecessor marking $m$ by consuming (i.e. removing) a token from each place of $t$’s pre-set and producing (i.e. adding) a token on each place of $t$’s post-set. The described firing relation is denoted $m \xrightarrow{t} m'$. Thereby $m \xrightarrow{t} m'$ is a step of $N$. 
Defi nition 2 (Open net). An open net \( N = [P,T,F,I,O,m_0,\Omega] \) consists of a Petri net \([P,T,F,m_0]\) together with

- an interface \((I \cup O) \subseteq P\) defined as two disjoint sets \(I\) of input places and \(O\) of output places such that \(p^- = \emptyset\) for any \(p \in I\) and \(p^+ = \emptyset\) for any \(p \in O\), and
- a set \(\Omega\) of final markings.

We further require that in the initial and the final markings the interface places are not marked, i.e., for all \(m \in \Omega \cup \{m_0\}\) we have \(m(p) = 0\), for all \(p \in I \cup O\).
Example

\[ \Omega = \{ [p4] \} \]
Definition 3 (Closed net). An open net $N$ with an empty interface, i.e., $I_N = \emptyset$ and $O_N = \emptyset$, is a closed net.

Definition 4 (Boundedness). A closed net $N$ is $k$-bounded if there exists a $k \in \mathbb{N}$ such that for each reachable marking $m \in R_N(m_0)$, $m(p) \leq k$, for all $p \in P_N$. 
Definition 5 (Interface compatible open nets). Let $N_1$, $N_2$ be two open nets with pairwise disjoint constituents except for the interfaces. If only input places of one open net overlap with output places of the other open net, i.e., $I_1 \cap I_2 = \emptyset$ and $O_1 \cap O_2 = \emptyset$, then $N_1$ and $N_2$ are interface compatible.
Definition 6 (Composition of open nets). Let $N_1$ and $N_2$ be two interface compatible open nets. The composition $N = N_1 \oplus N_2$ is the open net with the following constituents:

- $P = P_1 \cup P_2,$
- $T = T_1 \cup T_2,$
- $F = F_1 \cup F_2,$
- $I = (I_1 \cup I_2) \setminus (O_1 \cup O_2),$ 
- $O = (O_1 \cup O_2) \setminus (I_1 \cup I_2),$ 
- $m_0 = m_{01} \oplus m_{02},$ and 
- $\Omega = \{m_1 \oplus m_2 \mid m_1 \in \Omega_1, m_2 \in \Omega_2 \}.$

For markings $m_1$ of $N_1$ and $m_2$ of $N_2$ which do not mark the interface places, their composition $m = m_1 \oplus m_2$ is defined by $m(p) = m_i(p)$ if $p \in P_i$, for $i = 1, 2.$
Compose

\[ \Omega = \{ [p4] \} \quad \Omega = \{ [p8] \} \]
Composed Net

$\Omega = \{[p4,p8]\}$
Definition 7 (Deadlock). Let \( N = [P,T,F,I,O,m_0,\Omega] \) be a closed net. A reachable marking \( m \in R_N(m_0) \) is a deadlock in \( N \) iff \( m \notin \Omega \) and no transition \( t \in T \) is enabled in \( m \). If no such \( m \) exists in \( N \), then \( N \) is deadlock-free.
Deadlock Free?

$\Omega = \{[p4,p8]\}$
Definition 8 (Strategy, controllability). Let $M, N$ be two open nets such that $I_M = O_N$ and $O_M = I_N$. Then, $M$ is a strategy for $N$ iff $M \oplus N$ is deadlock-free. With $\text{Strat}(N)$ we denote the set of all strategies for $N$. $N$ is controllable iff its set of strategies is nonempty.
Controllable ?

\[ \Omega = \{ [c4] \} \]

\[ \text{Strat}(N) = \emptyset \]

AP-1: Internal Choice Receiving Follow-Up Anti-Pattern
Controllable?
Controllable?
Controllable?

\[ \Omega = \{ [p2, p4] \} \]
Possible Additional Requirements to Rule Out Undesirable Strategies

No dead transitions / interface places

Ω states need to be dead
Recommended Reading

Exposing Services
Exposing Services

Service Broker

find -> publish

exposed

bind

Service Requester  Service Provider

Services also need to be exposed in the bilateral case!
Two main approaches

- Selecting a service means to find for a given service $R$ (whose behavior is given) a compatible service $S$ in the repository.
- One approach is to expose the behavior of $S$ (this needs to be done for all services in the repository).
- Well-behavior of the composition of $R$ and $S$ can be verified using standard state space verification techniques.
- However, organizations usually want to hide the trade secrets of their services and thus need to find a proper abstraction of $S$ which is published.
- Another approach is to not expose the behavior of $S$, but a class of services $R$ that is compatible with $S$, e.g., the set $\text{Strat}(S)$.
- Then the composition of $R$ and $S$ is compatible if $\text{Strat}(S)$ contains $R$. From the set of strategies it is in general not possible to derive the original service.
First Approach

- p1: place
- p2: eat
- p3: pay
- p4: money
- p5: accept
- p6: cook
- p7: collect
- p8: get drunk

Diagram showing the flow of actions:
- Order -> Food
- Collect
- Pay
- Cook
- Get Drunk

Repository
Second Approach

Strat(S)
Operating Guidelines

- We advocate the second approach for reasons of efficiency and hiding trade secrets.
- Problem: Strat(S) is typically infinite!
- Operating guidelines provide a finite representation of a possibly infinite set of compatible services.

- Here we do not explain how the operating guideline is computed (see recommended reading) and focus on its application.
Basic Idea

GS1

p1
place

p2
eat

p3
pay

p4
money

TS(GS1)

order

food

money

p5
accept

order

food

money

OG(RS1)

p6

p7

p8

RS1
(1) simulation relation (weak simulation)

(2) constraints of corresponding states are satisfied
Weak simulation

- **R weakly simulates** P iff R can mimic any behavior of P.
- Formally, there exists a weak simulation relation such that:

```
\[ p \rightarrow r \]
\[ p' \rightarrow r' \]
\[ \tau \]
\[ \tau^* \]
```
R does not weakly simulate P
R weakly simulates P, but ...
P does not weakly simulate R
Evaluating Expressions

expr1 = a! or b? or c! or d? or f?
expr1 = (a! or b? or c!) and d?
expr1 = (a! and final) or d!
GS1 is a Strategy for RS1

GS1: place → order → food → pay → money

RS1: order → accept → food → collect → money
Operating Guideline

- N is an open net
- $B^\Phi$ is a Boolean Annotated Service Automaton (BSA), i.e., an automaton with annotated states that aims to describe (possible infinite) sets of open nets
- Match($B^\Phi$) is the set of all open nets that match with $B^\Phi$ (all nets that are weakly simulated by $B^\Phi$ such that constraints of corresponding states are satisfied)
- OG(N) is the operating guideline for N, i.e., a BSA $B^\Phi$ such that Match($B^\Phi$) = Strat(N)

Definition 16 (Operating guidelines, OG). The operating guidelines OG(N) of an open net N is a BSA such that Match(OG(N)) = Strat(N).
Another Example
Not Matching

\[ r_5 \rightarrow \neg \text{order} \]
\[ r_6 \rightarrow \neg \text{money} \]
\[ r_7 \rightarrow \text{food} \]
\[ \tau \rightarrow \text{final} \]

\[ q_1: \neg \text{order} \lor \tau \]
\[ q_2: \neg \text{food} \lor \text{final} \lor \tau \]
\[ q_3: \neg \text{money} \lor \tau \]
\[ q_4: \text{final} \lor \tau \]
Repaired Service Behavior

\[
q_1: \neg \text{order} \lor \tau
\]
\[
q_2: (\neg \text{food} \land \text{final}) \lor \tau
\]
\[
q_3: \neg \text{money} \lor \tau
\]
\[
q_4: \text{final} \lor \tau
\]
Definition 17 (Most permissive strategy). Let $OG(N) = [Q, MP, \delta, q_0, \Phi]$ be the operating guidelines for a controllable open net $N$. Then, an open net $M$ is the most permissive strategy for $N$ iff $TS(M) = [Q, MP, \delta, q_0, \Omega]$, where $\Omega = \{q \mid \text{final occurs in } \Phi(q)\}$. 
Recommended Reading

Replacing and Refining Services
Replacing or Refining Services

replace or refine

service B

service C

service D
Definition 18 (Interface equivalent open nets). Two open nets $M$ and $N$ are interface equivalent iff $I_M = I_N$ and $O_M = O_N$.

Definition 19 (Accordance). Let $N$ and $N'$ be two interface equivalent open nets. $N'$ can replace $N$ under accordance ($N'$ accords with $N$, for short) iff $\text{Strat}(N) \subseteq \text{Strat}(N')$. 
Projection Inheritance is Too Strict

Theorem 1 (Projection inheritance implies accordance [37]). Let \( N \) and \( N' \) be two open nets. If \( N \) and \( N' \) are related by projection inheritance, then \( N' \) accords with \( N \) and \( N \) accords with \( N' \).

\[ \text{order} \rightarrow \text{food} \rightarrow \text{money} \rightarrow \text{accept} \rightarrow \text{cook} \rightarrow \text{collect} \rightarrow \text{p5} \rightarrow \text{p6} \rightarrow \text{p7} \rightarrow \text{p8} \]

accords with

(while there is no inheritance relation)
Definition 22 (Refinement of OGs). Let $N$ and $N'$ be interface equivalent open nets and let $OG(N) = [Q, MP, \delta, q_0, \Phi]$ and $OG(N') = [Q', MP', \delta', q'_0, \Phi']$ be the corresponding operating guidelines. Then, $OG(N) \subseteq OG(N')$ (i.e., $OG(N')$ refines $OG(N)$) iff there is a simulation relation $\xi \subseteq Q \times Q'$ such that for all $[q, q'] \in \xi$, the formula $\Phi(q) \Rightarrow \Phi'(q')$ is a tautology.

Theorem 2 (Checking accordance [32]). Let $N$ and $N'$ be two open nets and let $OG(N)$ and $OG(N')$ be the corresponding operating guidelines. Then, $OG(N) \subseteq OG(N')$ iff $Strat(N) \subseteq Strat(N')$.

Accordance can be checked using operating guidelines!

Details not important at this stage.
Strat(X) = Strat(Y)
Inheritance Preserving Transformation Rules (1/2)
Inheritance Preserving Transformation Rules (2/2)

Inheritance preserving transformation rules also preserve accordance!

But are too strong ...
Accordance Preserving Transformation
Rule 1
Accordance Preserving Transformation
Rule 2
Accordance Preserving Transformation
Rule 3
Accordance Preserving Transformation
Rule 4
Another Type of Anti Pattern
Strategy for one net and not the other
Strategy for one net and not the other
Accordance Preserving Transformation
Rule 5

\[
\begin{array}{c}
R \quad t1 \quad p1 \\
\quad \quad t2 \quad t3 \\
\quad \quad p2 \quad p3 \\
\quad \quad \quad t4 \quad t5 \\
S
\end{array}
\quad \subseteq
\quad
\begin{array}{c}
R \quad t1 \\
\quad \quad t6 \\
\quad \quad p4 \\
\quad \quad \quad t7 \\
\quad \quad \quad \quad t2 \quad t3 \\
\quad \quad \quad \quad p2 \quad p3 \\
\quad \quad \quad \quad \quad t4 \quad t5 \\
S
\end{array}
\]
Recommended Reading

Integrating Services Using Adapters
The Need For Adapters

service A

? 

adapter

service B

service A

service B
Example

[Diagram showing a process involving a tourist and a cook, with nodes labeled tFood, tMoney, cOrder, cFood, and cMoney]
Adapter ??????
Specification of Elementary Activities (SEA)

- To avoid creating money, deleting evidence, confusing meters with liters, etc.

<table>
<thead>
<tr>
<th>Elementary activity</th>
<th>Possible transformation rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create $a$</td>
<td>$\mapsto a$</td>
</tr>
<tr>
<td>Copy $a$</td>
<td>$a \mapsto a, a$</td>
</tr>
<tr>
<td>Delete $a$</td>
<td>$a \mapsto$</td>
</tr>
<tr>
<td>Transform $a, b, c$ into $d, e$</td>
<td>$a, b, c \mapsto d, e$ or $a, b, c \mapsto a, b, c, d, e$</td>
</tr>
<tr>
<td>Split $a$ into $b, c, d$</td>
<td>$a \mapsto b, c, d$ or $a \mapsto a, b, c, d$</td>
</tr>
<tr>
<td>Merge $a, b, c$ into $d$</td>
<td>$a, b, c \mapsto d$ or $a, b, c \mapsto a, b, c, d$</td>
</tr>
</tbody>
</table>
SEA Example

\[
\begin{align*}
&\rightarrow \text{cOrder} \\
&\text{cFood} \rightarrow \text{tFood} \\
&\text{tMoney} \rightarrow \text{cMoney}
\end{align*}
\]
Overall Idea

Controller

SEA-Based Engine

tFood

tMoney

Order

Food

Money
SEA-Based Engine

n = notify
e = enable

s = send
r = receive

c = actual transformation

\begin{align*}
&\text{tFood} \rightarrow \text{cOrder} \\
&\text{cFood} \rightarrow \text{tFood} \\
&\text{tMoney} \rightarrow \text{cMoney}
\end{align*}
Business As Usual ...

- Selecting a controller is like selecting a strategy.
- One approach is to construct "the" most permissive one.
Recommended Reading

Correctness at "model-time" is irrelevant!
Process Mining

- Process discovery: "What is really happening?"
- Conformance checking: "Do we do what was agreed upon?"
- Performance analysis: "Where are the bottlenecks?"
- Process prediction: "Will this case be late?"
- Process improvement: "How to redesign this process?"
- Etc.
• Process discovery: "What is the real curriculum?"
• Conformance checking: "Do students meet the prerequisites?"
• Performance analysis: "Where are the bottlenecks?"
• Process prediction: "Will a student complete his studies (in time)?"
• Process improvement: "How to redesign the curriculum?"
Screenshot of ProM 5.0
Example Setting for Service Mining
Recommended Reading

- www.processmining.org
Conclusion
Service Interaction Demystified

- Patterns and Anti-Patterns
- Formalization
- Analysis
Questions Addressed

1. Exposing Services
   - How to inform others about me such that cooperation is possible?
   - Two approaches: (a) expose own behavior and (b) provide operating guideline.

2. Replacing and Refining Services
   - How to replace or refine a service without introducing problems?
   - Inheritance, accordance, transformation rules, etc.

3. Integrating Services Using Adapters
   - How to resolve behavioral incompatibilities?
   - Adapter generation.

4. Service Mining
   - How to analyze the run-time behavior?
Relevant WWW sites

• http://www2.informatik.hu-berlin.de/top/best/
• http://www.service-technology.org
• http://www.workflowpatterns.com
• http://www.processmining.org
• http://promimport.sourceforge.net
• http://prom.sourceforge.net
• http://www.workflowcourse.com
• http://www.vdaalst.com