Enabling Flexibility in Process-Aware Information Systems

Challenges, Paradigms, Technologies

1

BARBARA WEBER
UNIVERSITY OF INNSBRUCK

MANFRED REICHERT
ULM UNIVERSITY

SEPTEMBER 9TH 2009

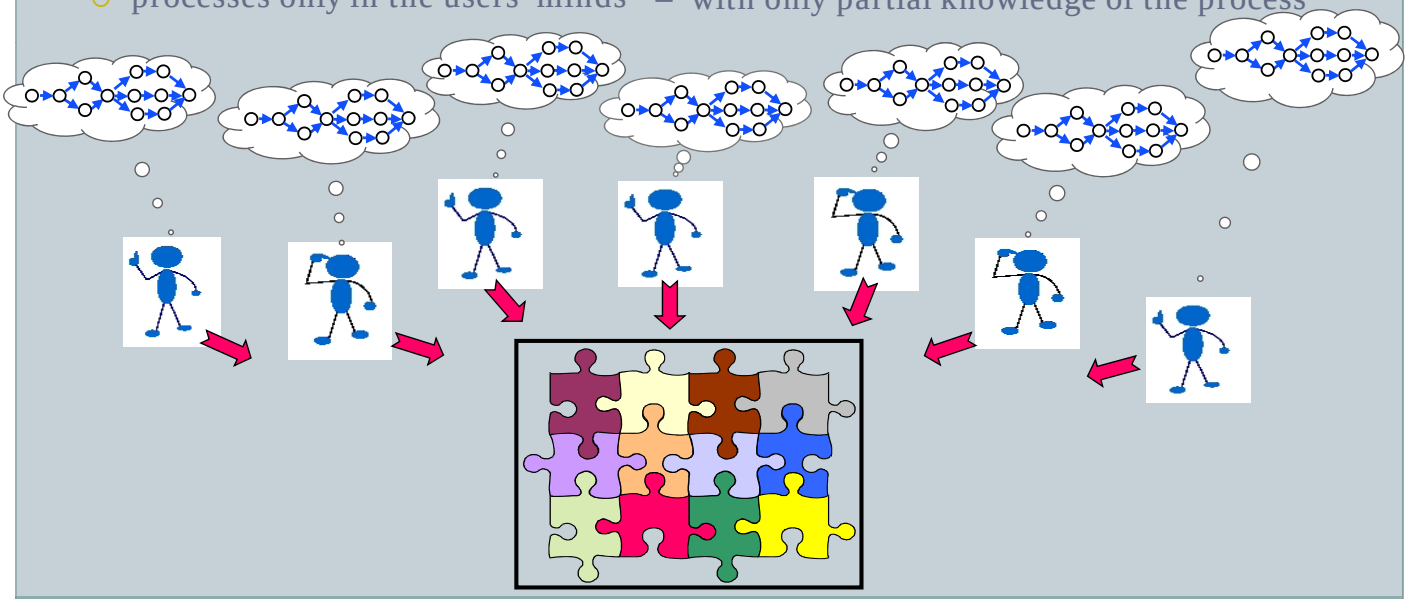
Agenda

2

1. Introduction
2. Flexibility Issues of the Imperative Approach to Business Process Management
3. Pattern-based evaluation of PAIS in respect to their ability to provide process flexibility
4. Flexibility Issues of the Declarative Approach to Business Process Management
5. Flexibility Issues in Data-driven Processes
6. Summary and Outlook

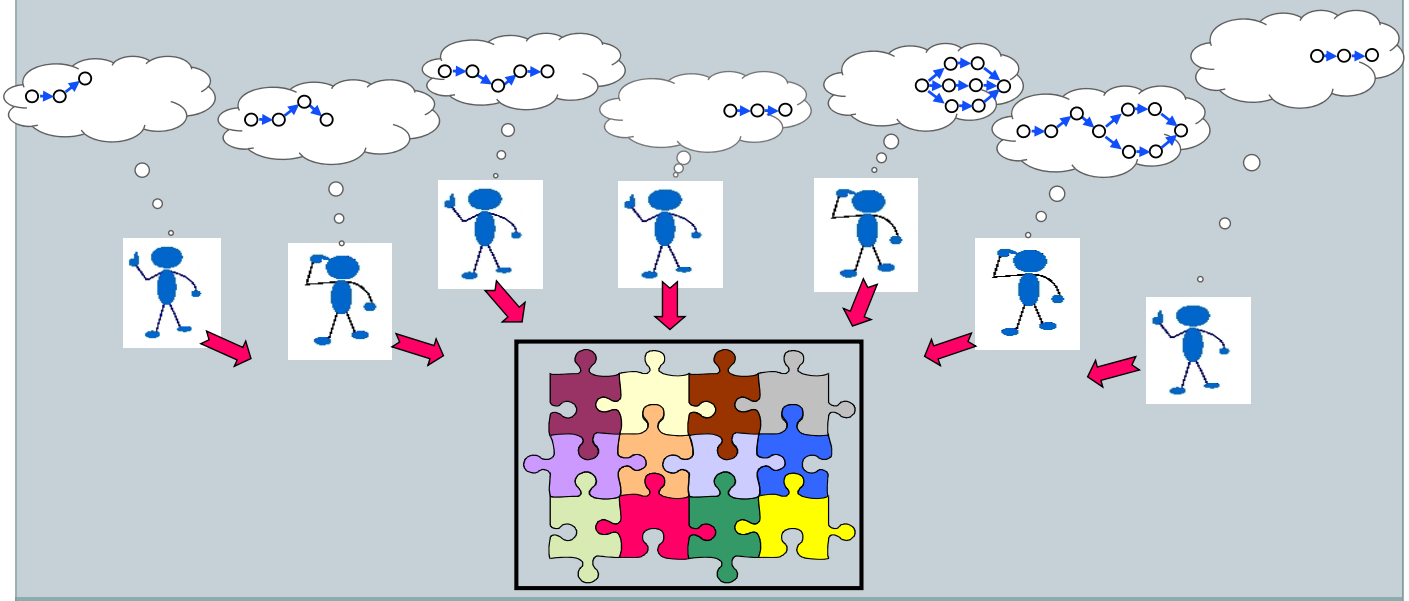
1. Introduction

- **Current Situation in Many Companies**
  - users interact with monolithic, function-oriented application systems
  - processes only in the users' minds – with only partial knowledge of the process



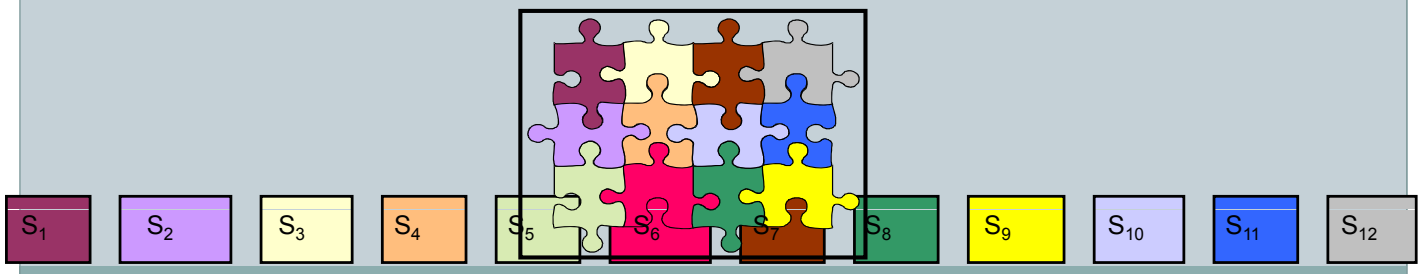
1. Introduction

- **Vision of SOA**
  - individually callable application functions ("services")



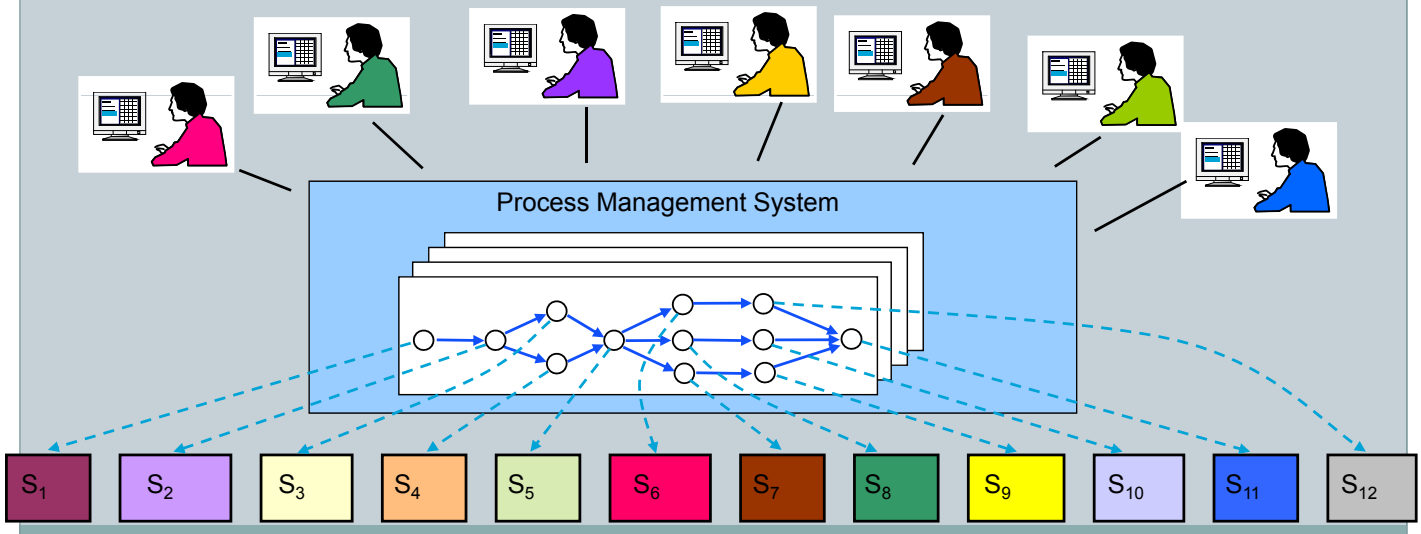
1. Introduction

- **Vision of SOA**
  - individually callable application functions (“services”)



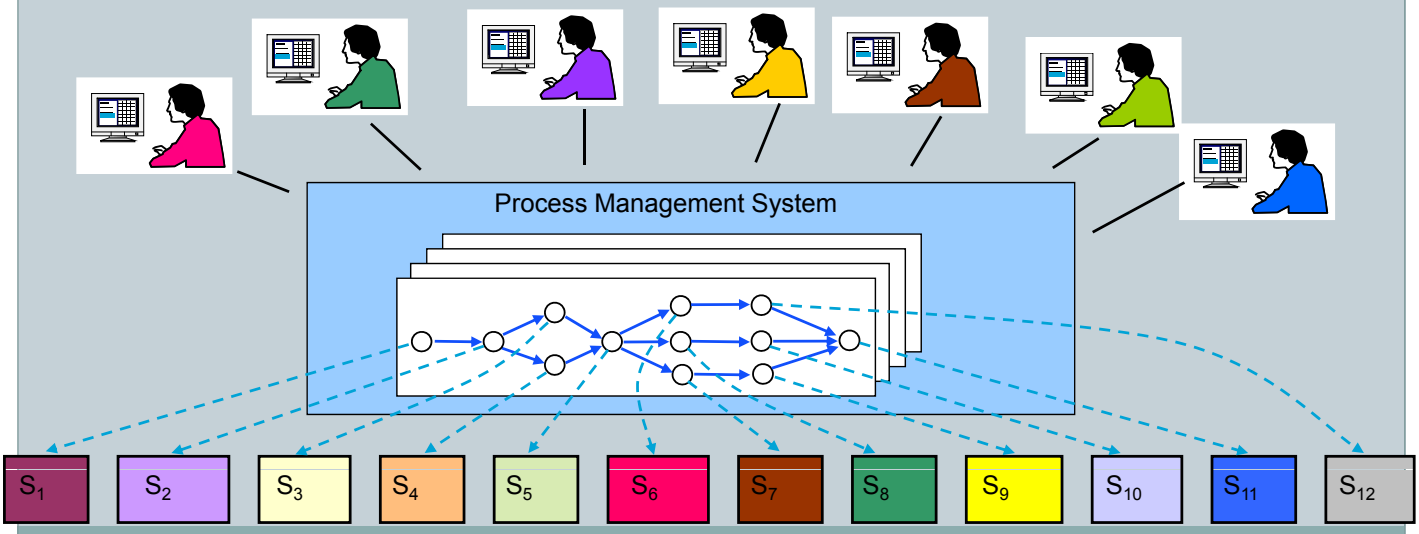
1. Introduction

- **Vision of SOA**
  - individually callable application functions (“services”)
  - combined by explicitly defined processes
  - whose execution is supported by a process management system



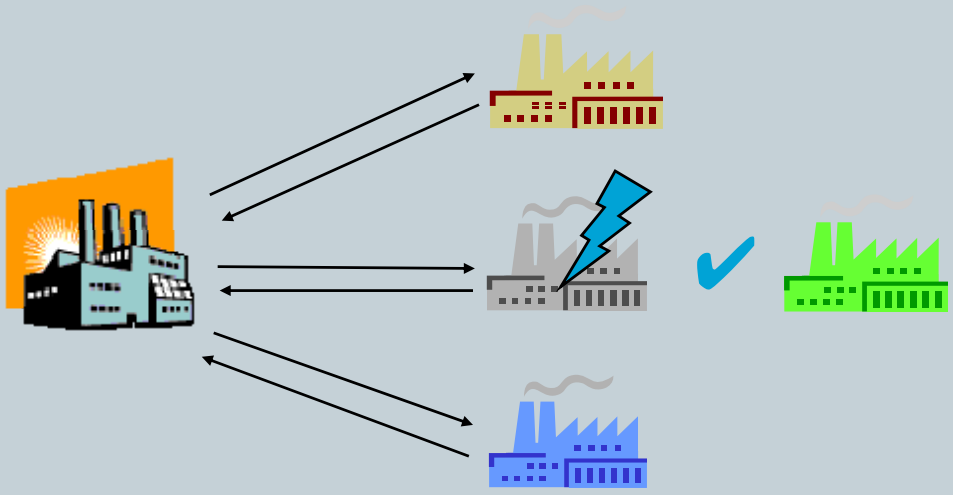
1. Introduction

- **Goals of SOA**
  - improvement of process quality (reduction of errors, waste time, ...)
  - increased **flexibility**
  - ...



1.1 Need for Process Flexibility

- **Do we believe ...**
  - ... that the on-the-fly exchange of a strategically important supplier can be reduced to the substitution of its web services by other ones?

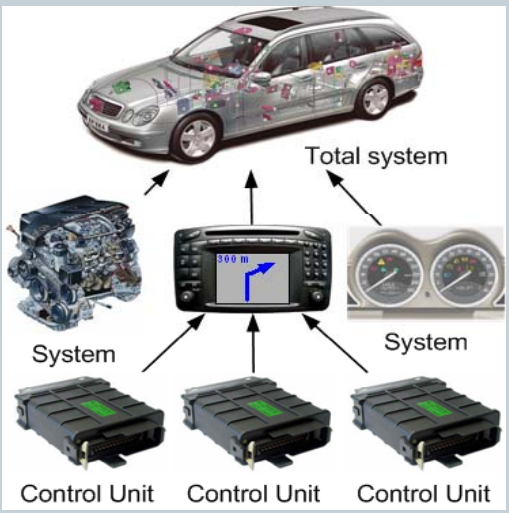


1.1 Need for Process Flexibility

- **Or do we really believe ...**
  - ... that process-aware information systems (PAIS) can prescribe to a pysician how to treat his or her patients? (see Lenz & Reichert, 2007 [LeRe07])

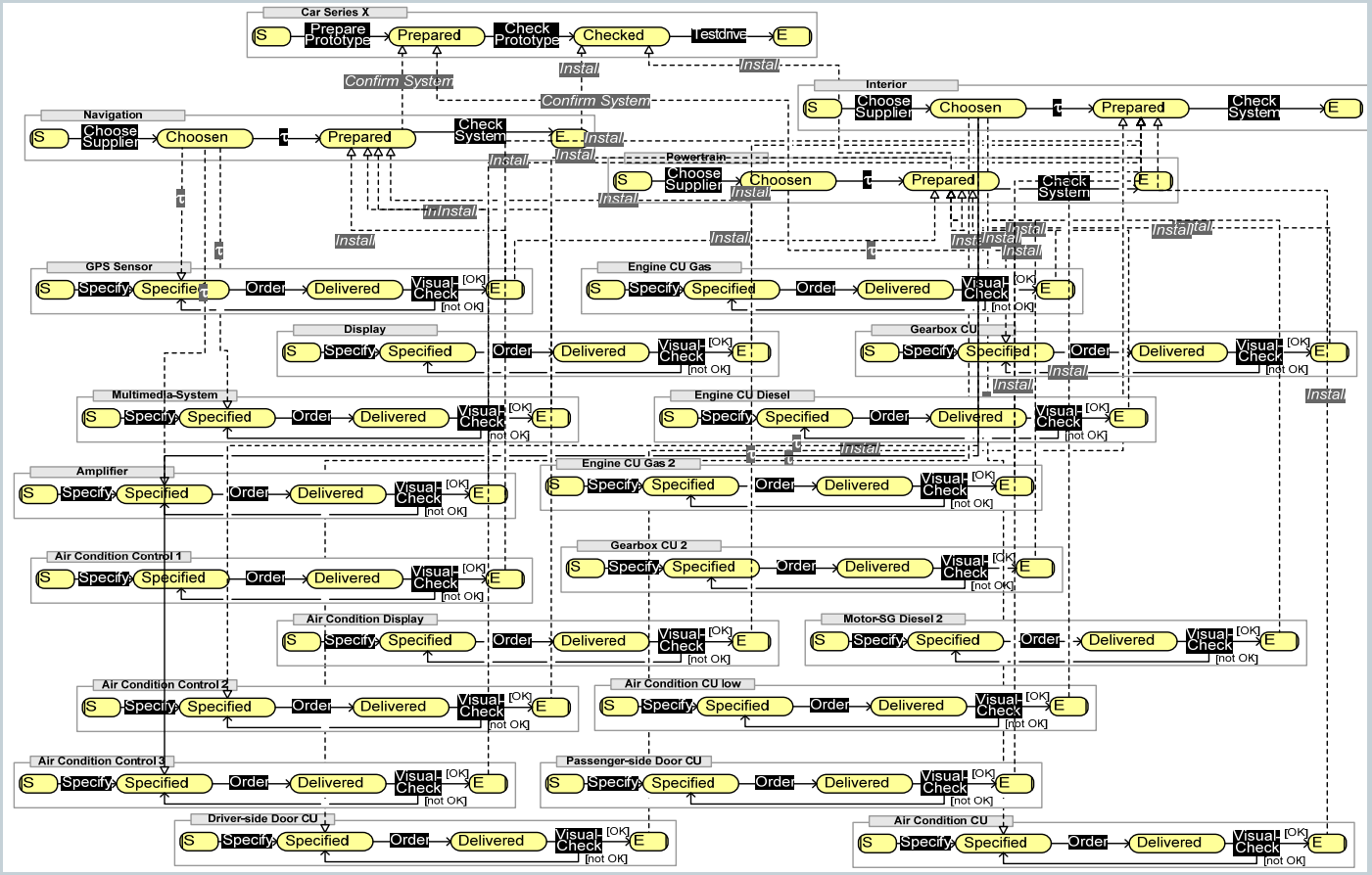
1.1 Need for Process Flexibility

- **Or do we really believe ...**
  - that long-running engineering processes can be completely pre-modeled?



- Example: Release management for E/E-systems in a car
- 200 - 300 control devices to be systematically tested and released
- Requires the execution of hundreds up to thousands of processes
- Concurrent engineering ➪ complex dependencies have to be considered

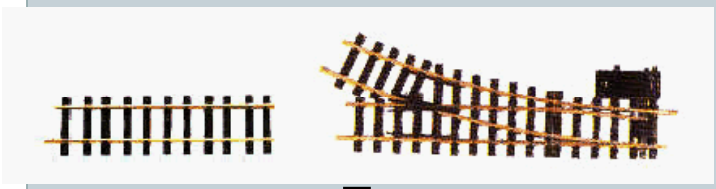
Source: [MHH+06]



1.2 Different Aspects of Process Flexibility

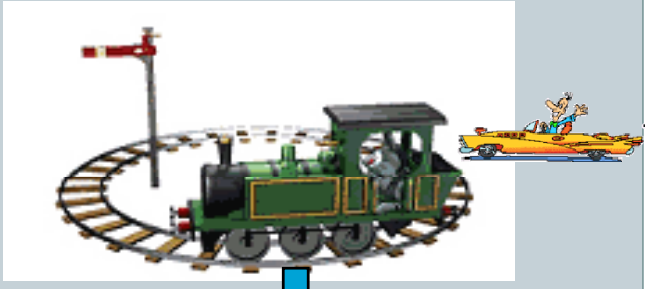
very important to discriminate

flexibility at build-time



How to quickly implement or configure new processes?

flexibility at run-time

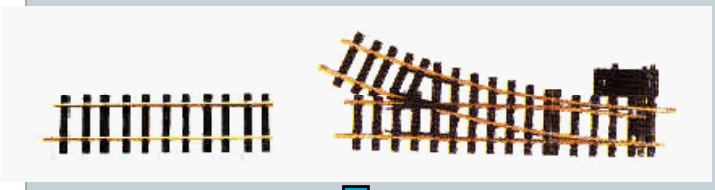


How to deal with uncertainty and exceptional cases during runtime?

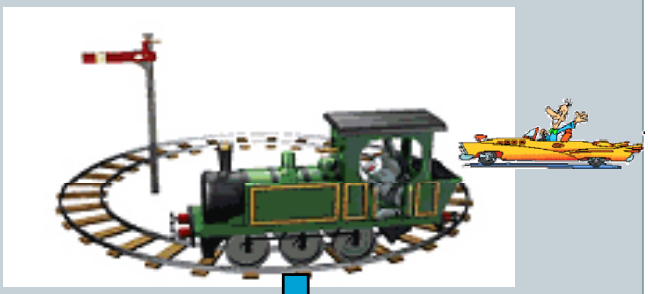
1.2 Different Aspects of Process Flexibility

very important to discriminate

flexibility at build-time



flexibility at run-time

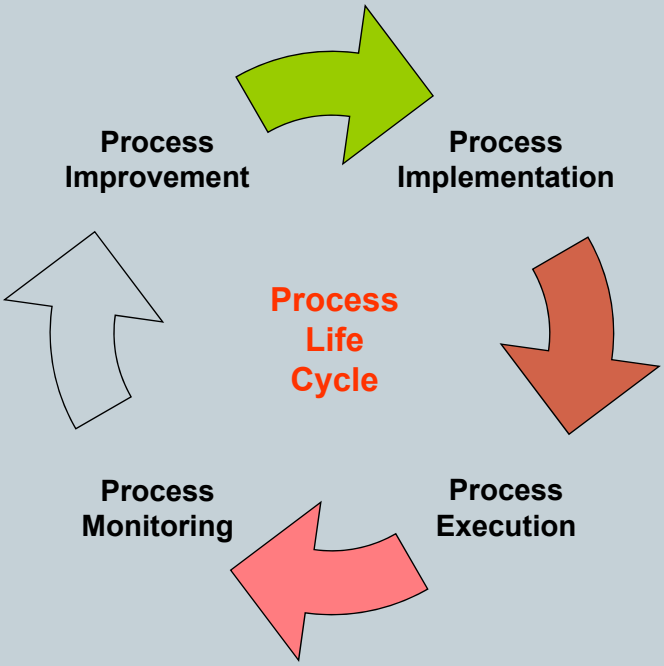


And: What can we learn in respect to process optimizations from these built-time configurations and run-time adaptations?

1.2 Different Aspects of Process Flexibility

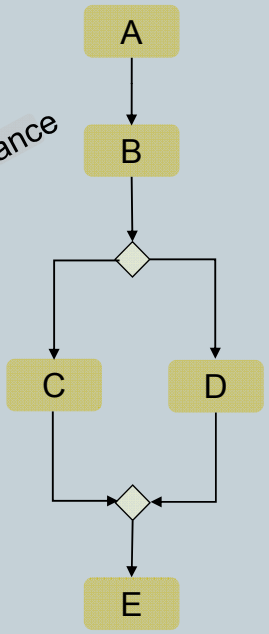
Important:

Flexibility Issues Need to be Considered for all Phases of the Process Lifecycle



1.3 Approaches for Process Modeling & Execution (1)

Imperative Approaches

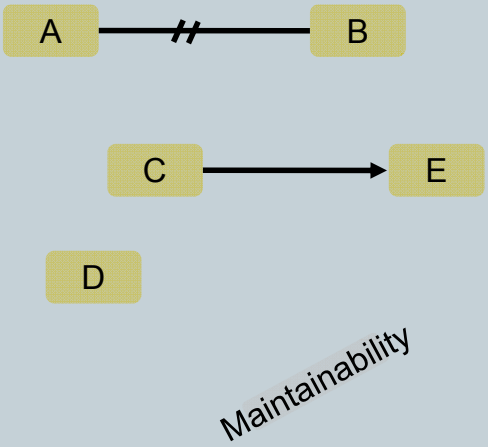


Performance

Flexibility

Declarative Approaches

Understandability

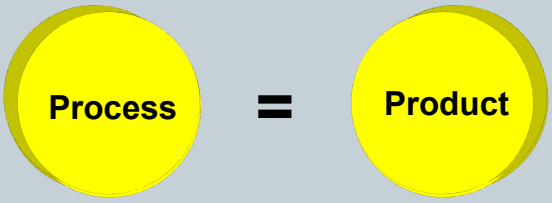


Expressiveness

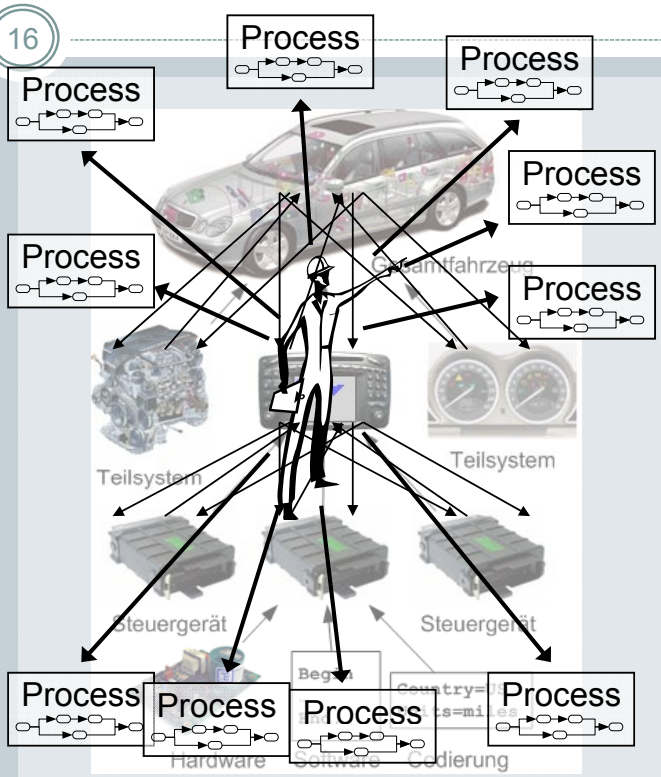
Maintainability

1.3 Approaches for Process Modeling & Execution (2)

Data-/Product-Driven Approaches



How to utilize the close relation between product data and process?



1.4 Teaching Objectives

17

- Getting a clear picture of flexibility issues in PAIS
- Obtaining an appreciation and understanding of the unique set of challenges that flexible processes are faced with
- Knowing the particular distinguishers of flexible processes in all phases of the life cycle
- Being aware of gaps in industry needs and existing solutions for realizing flexible PAIS
- Getting a better understanding of how to ideally cope with process changes in process-aware information systems

1.5 Addressed Topics

18

- Challenges and requirements for flexible PAIS.
- Paradigms for realizing flexible processes; e.g., adaptive process, declarative workflow, case handling, data-driven process, etc.
- Analysis of the diversity of these paradigms and discussion along characteristic scenarios
- State-of-the-art technology for enabling process flexibility (e.g., ad-hoc changes, late modeling)
- Implementing process changes through process schema evolution (and change propagation)
- Change patterns and change features in PAIS; pattern-based comparison of state-of-the-art technology for flexible processes

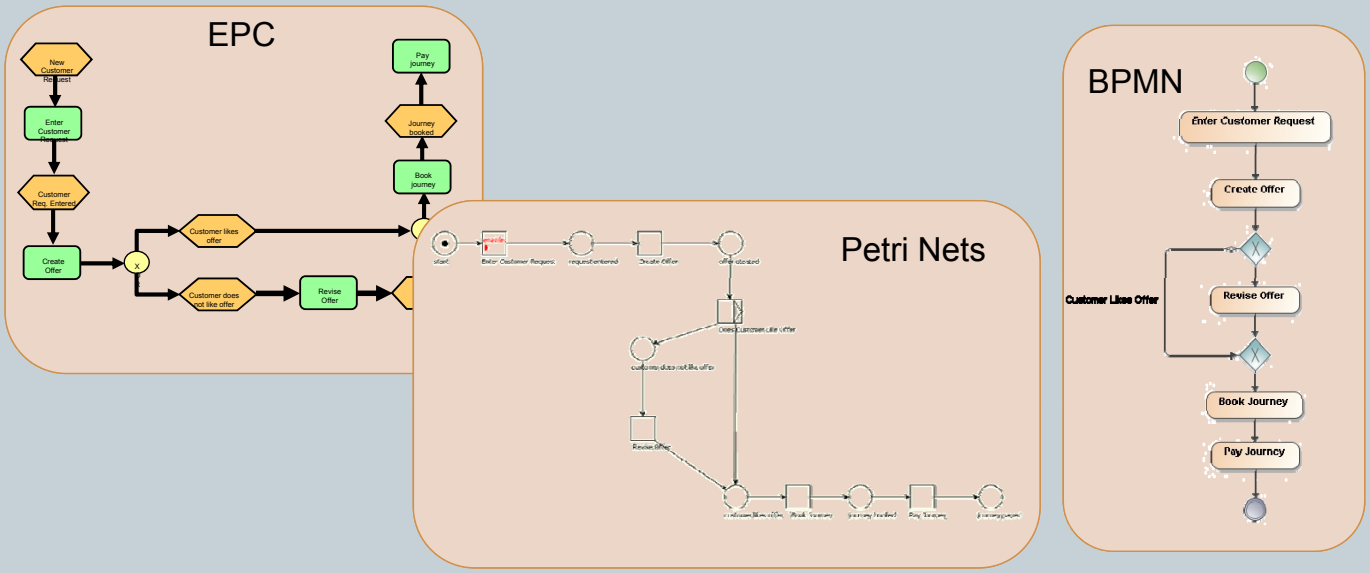
Agenda

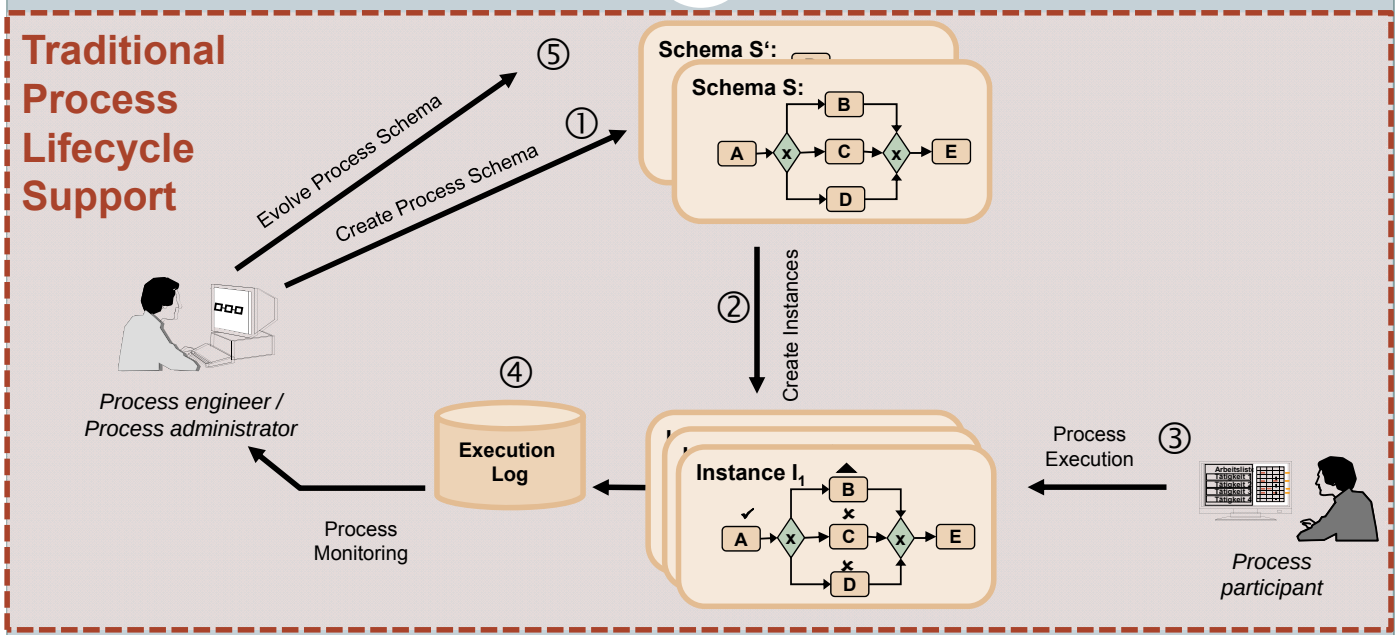
1. Introduction
2. Flexibility Issues of the Imperative Approach to Business Process Management
3. Pattern-based evaluation of PAIS in respect to their ability to provide process flexibility
4. Flexibility Issues of the Declarative Approach to Business Process Management
5. Flexibility Issues in Data-driven Processes
6. Summary and Outlook

Chapter 2: Flexibility Issues of the Imperative Approach ...

2.1 The Imperative Approach to Business Process Management
2.2 Capturing Variability in Business Process Models
2.3 Dealing with Uncertainty: Late Binding & Late Modeling
2.4 Handling Expected Process Exceptions During Runtime
2.5 Enabling Ad-hoc Process Changes During Runtime
2.6 Monitoring & Analyzing Flexible and Dynamic Processes
2.7 Evolving Process Schemes and Migrating Process Instances
2.8 All-in-one: The ADEPT2 (AristaFlow BPM Suite) Technology
2.9 Integrated Lifecycle Support for Adaptive and Dynamic Processes

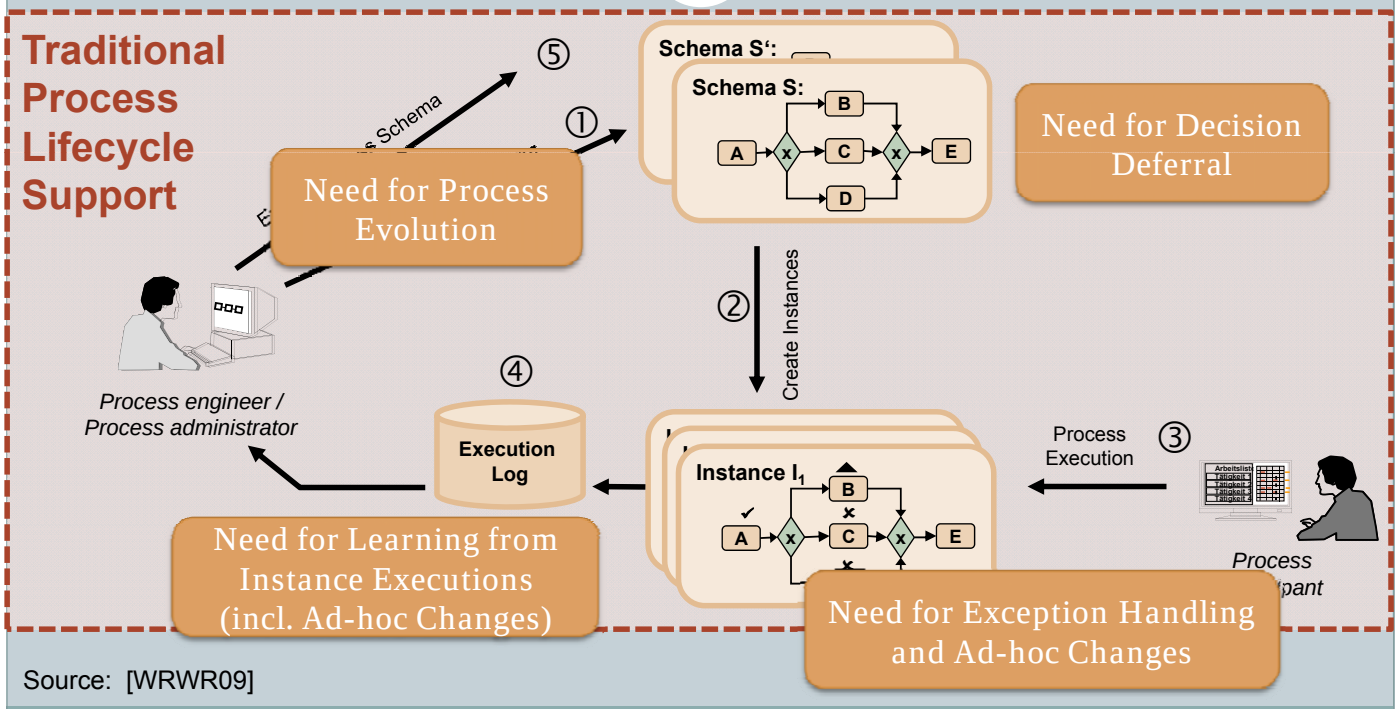
Different Notations for Imperative Process Modeling





Source: [WRWR09]

2.1.2 Some Flexibility Issues Along the Lifecycle

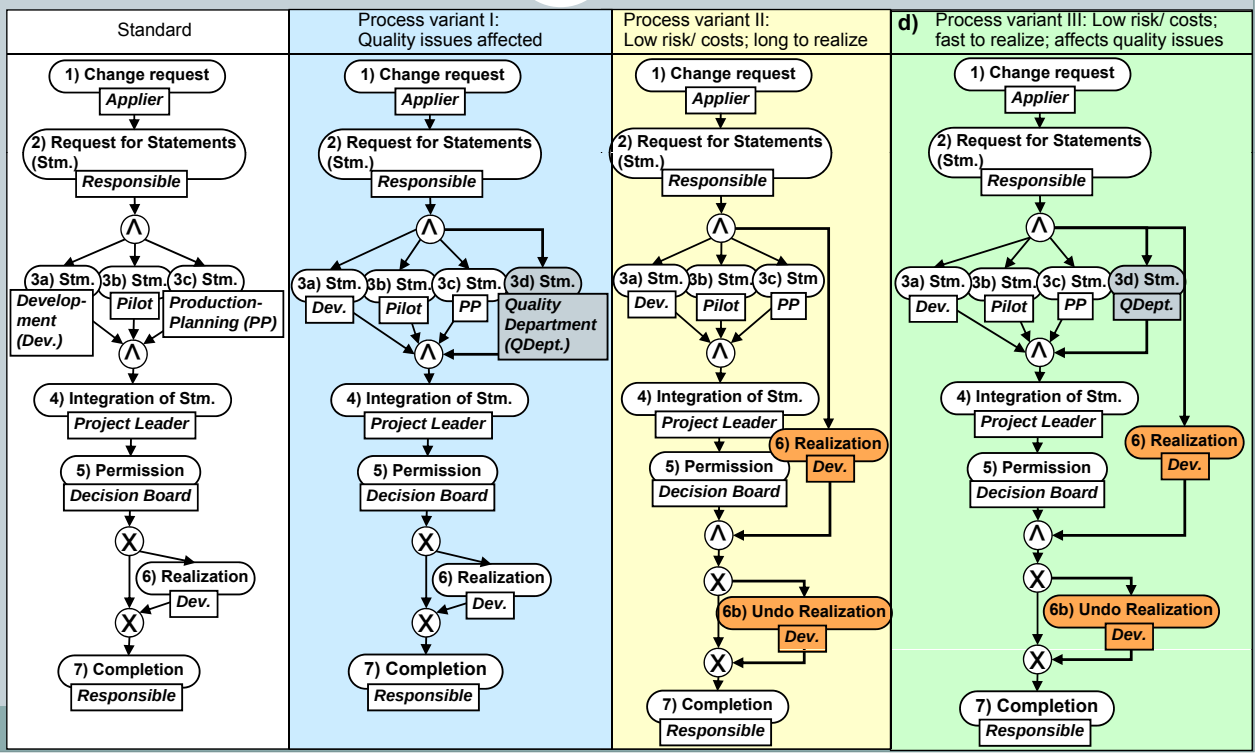


Source: [WRWR09]

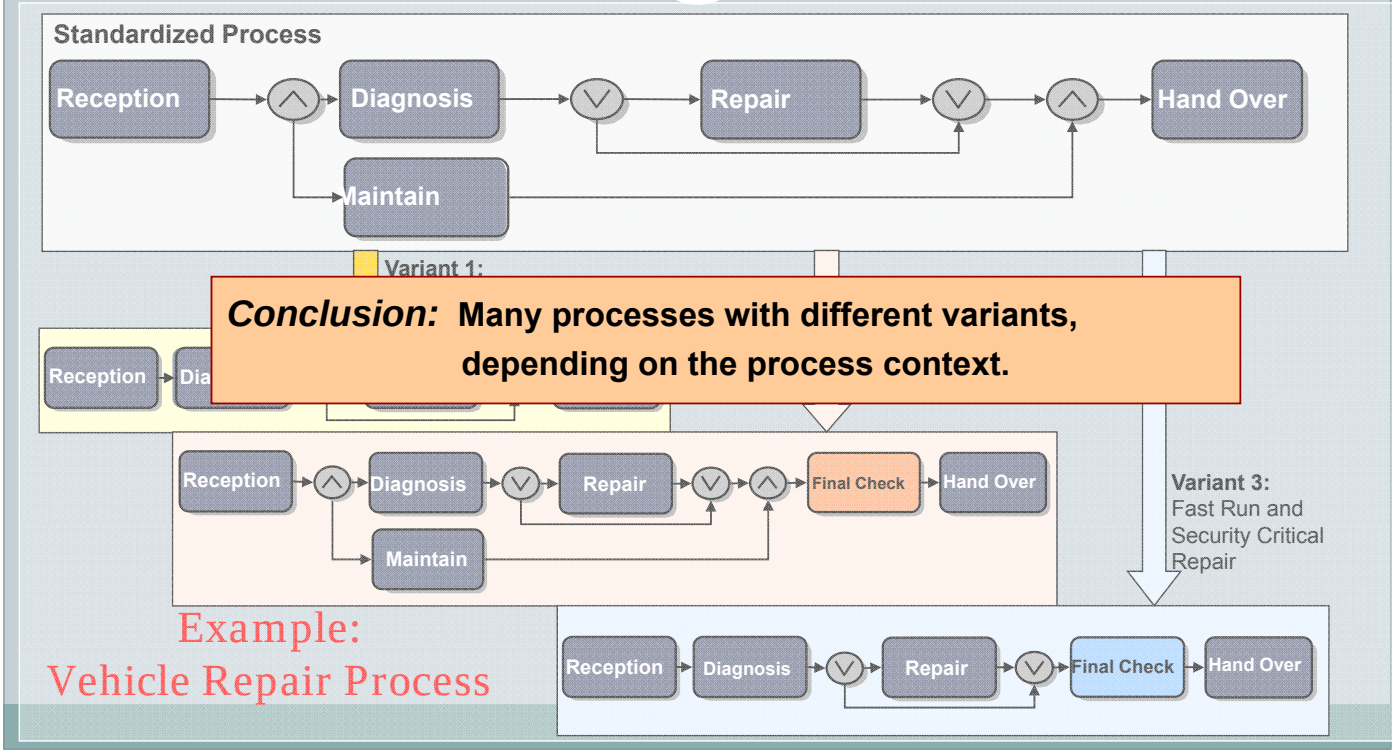
Chapter 2: The Imperative Approach ...

2.2.1 Motivation

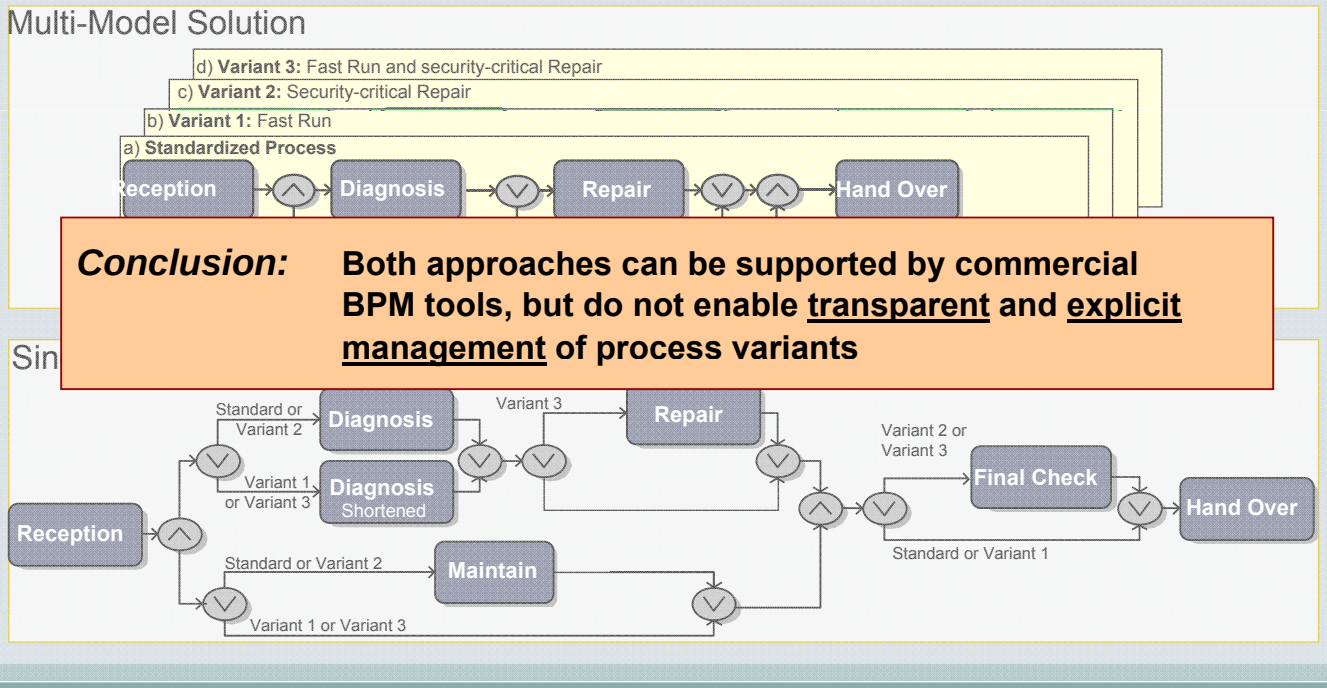
Example: Change Management



2.2.1 Motivation

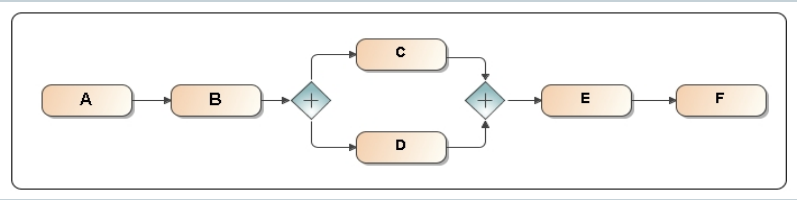


2.2.2 Configuring Process Variants in Existing BPM Tools



2.2.3 Parameter-driven Process Configuration (1)

Idea: Configuring variants by switching off paths in the master model



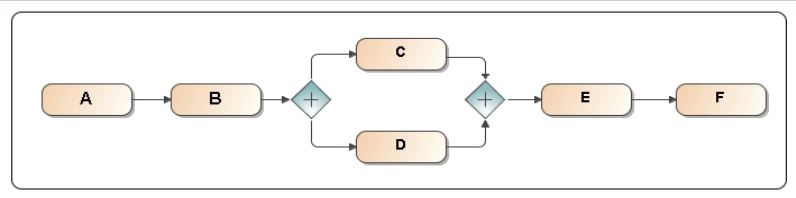
Parameter driven

- *trip-category = "adv"*
- *duration < 2 weeks*
- *total cost <5K*

Rosemann & Aalst [RoAa07]

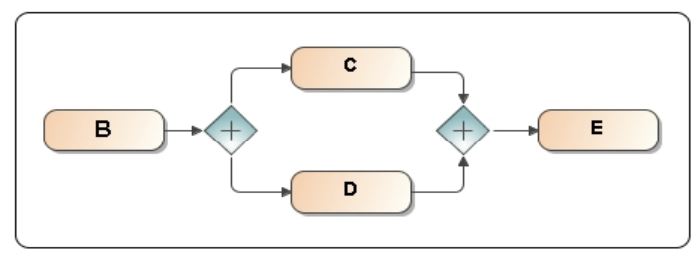
2.2.3 Parameter-driven Process Configuration (2)

Idea: Configuring variants by switching off paths in the master model



Parameter driven

- ◯ *trip-category = "adv"*
- ◉ *duration < 2 weeks*
- ◯ *total cost <5K*

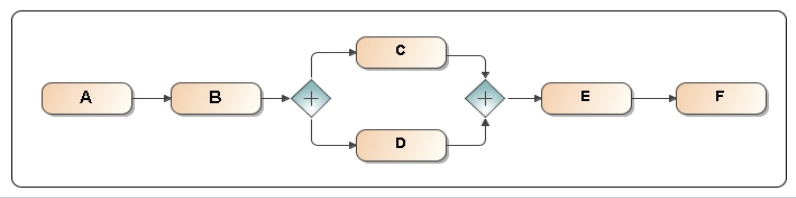


A	Off
B	On
C	On
D	On
E	On
F	Off
P	Off
Q	Off

Rosemann & Aalst [RoAa07]

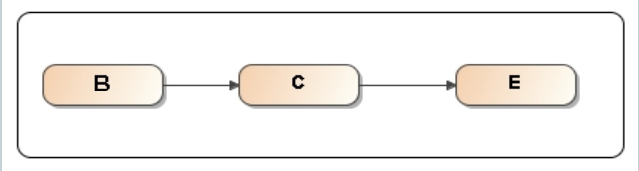
2.2.3 Parameter-driven Process Configuration (3)

Idea: Configuring variants by switching off paths in the master model



Parameter driven

- ◯ *trip-category = "adv"*
- ◉ *duration < 2 weeks*
- ◉ *total cost <5K*

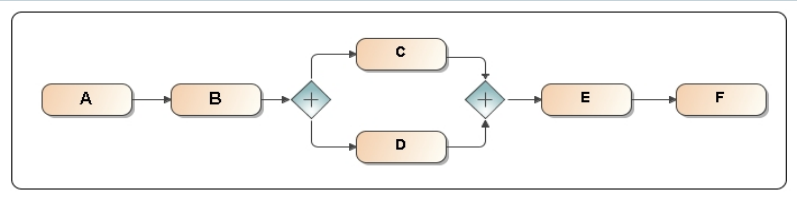


A	Off
B	On
C	On
D	Off
E	On
F	Off
P	Off
Q	Off

Rosemann & Aalst [RoAa07]

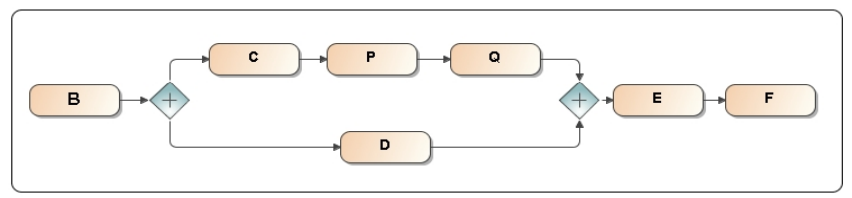
2.2.3 Parameter-driven Process Configuration (4)

Idea: Configuring variants by switching off paths in the master model



Parameter driven

- ◉ *trip-category = "adv"*
- ○ *duration < 2 weeks*
- ○ *total cost <5K*

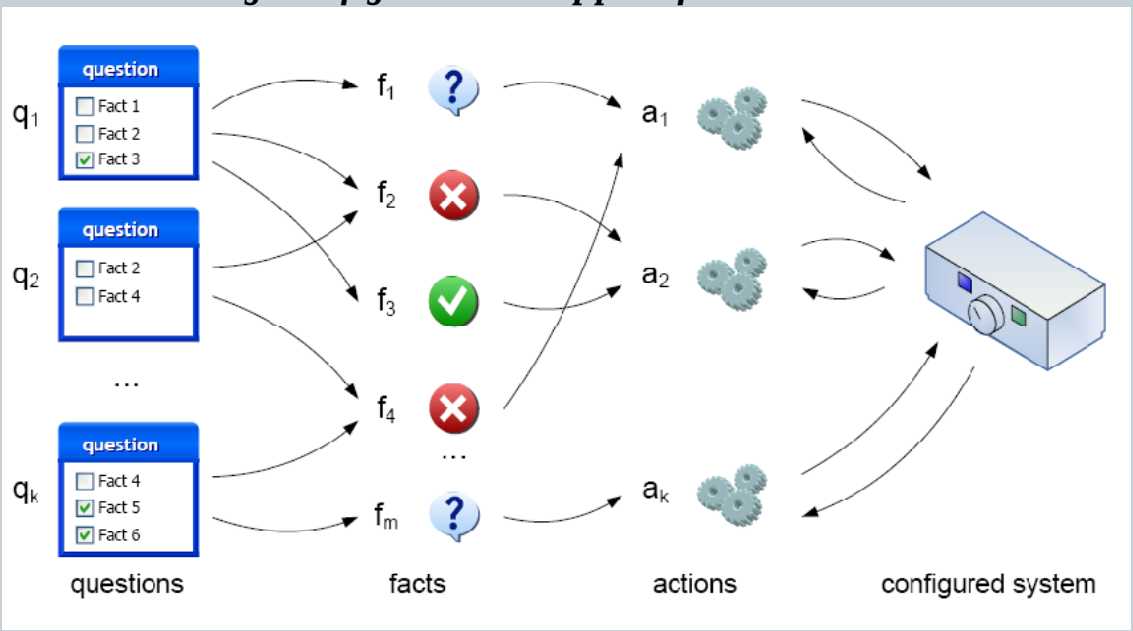


A	Off
B	On
C	On
D	On
E	On
F	On
P	On
Q	On

Rosemann & Aalst [RoAa07]

2.2.4 Questionnaire-driven Process Configuration

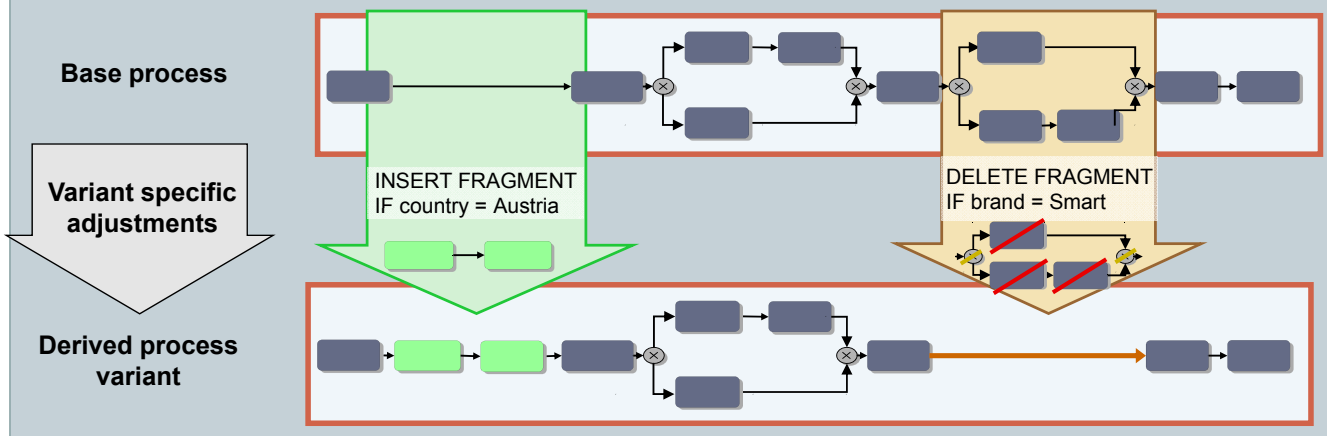
Idea: Providing Configuration Support for End Users / Domain Experts



La Rosa et al. [LaRos09]

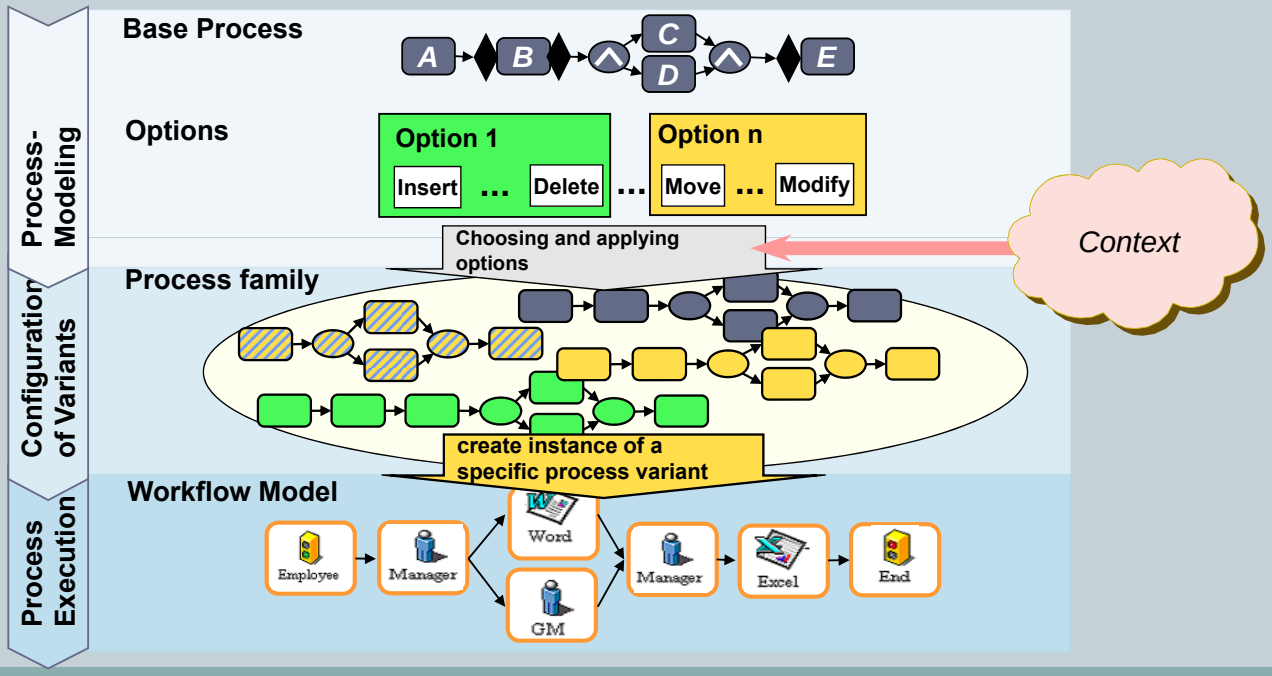
2.2.5 Context-driven Process Configuration in Provop (1)

- ***Observation: Process variants can be created by adapting a common reference model***

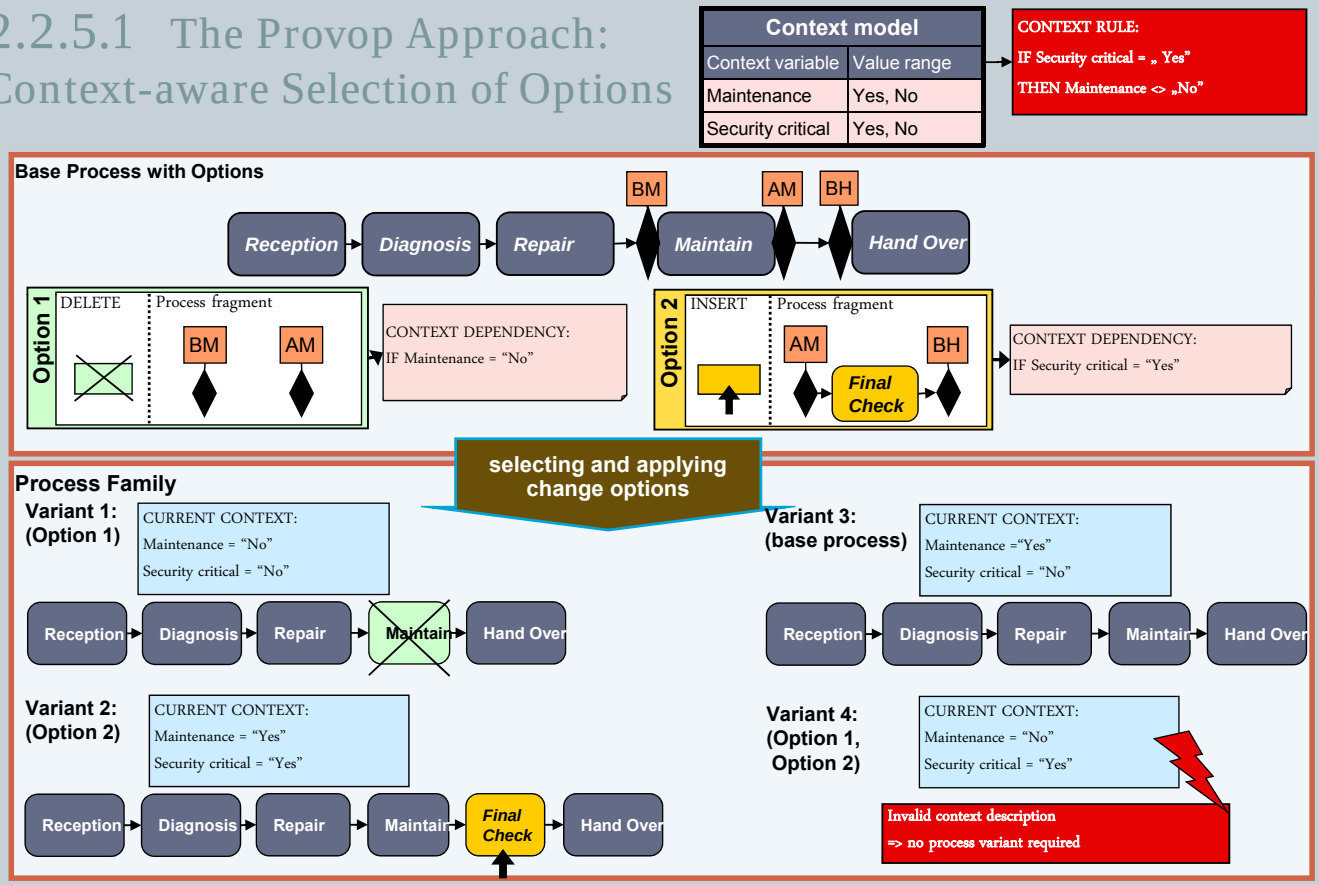


2.2.5 Context-driven Process Configuration in Provop (2)

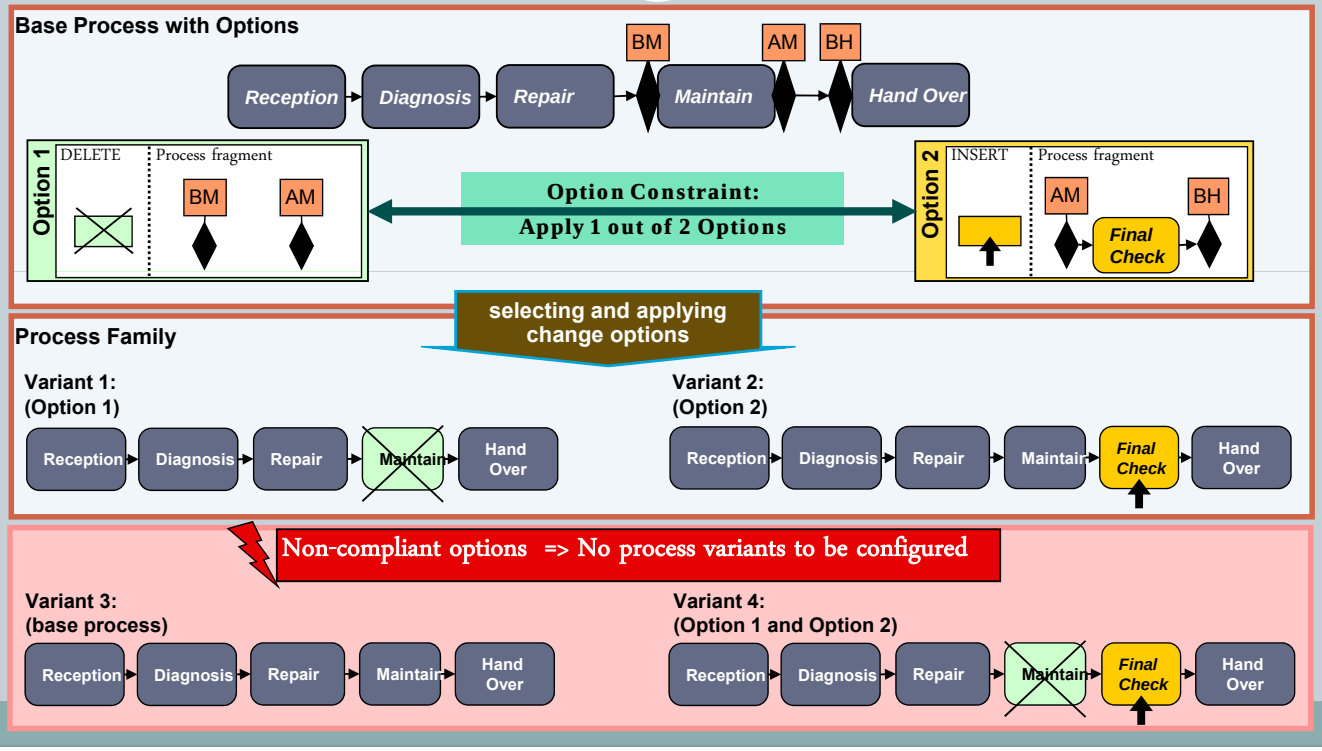
Hallerbach, Bauer & Reichert [HBR08, HBR09b, HBR09c]



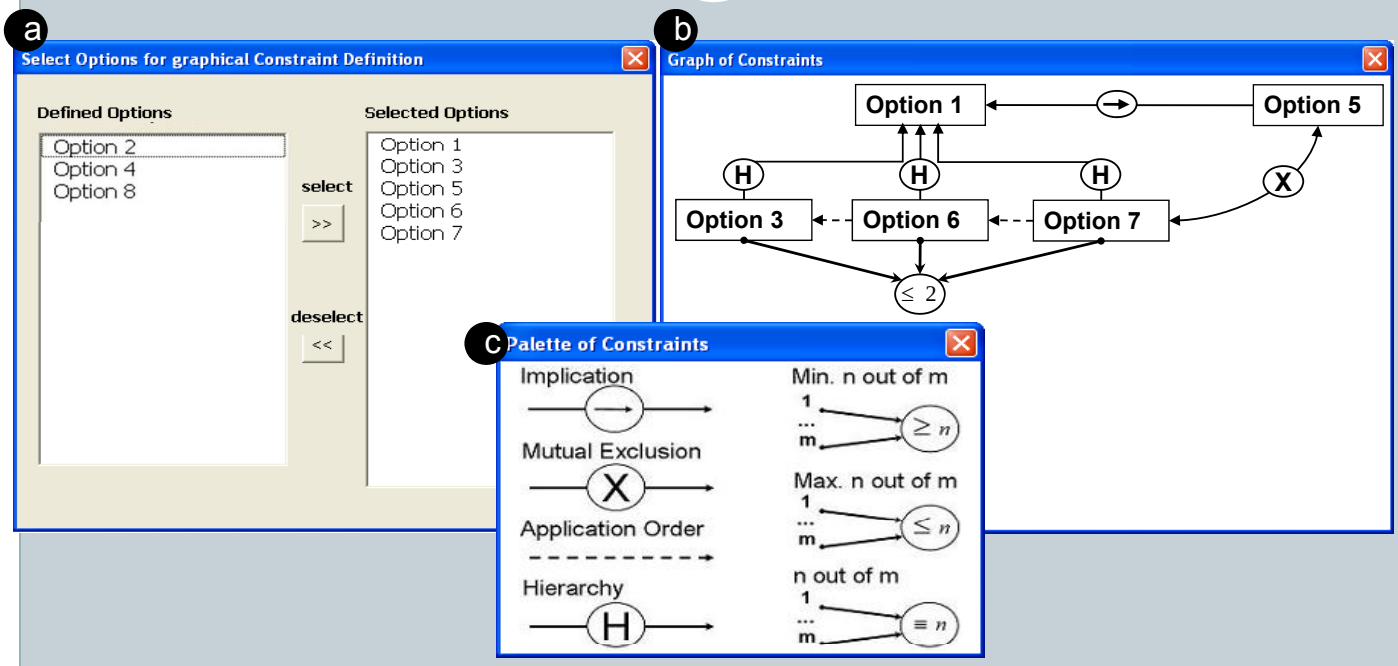
2.2.5.1 The Provop Approach: Context-aware Selection of Options



2.2.5.2 The Provop Approach: Constraining the Use of Options (1)



2.2.5.2 The Provop Approach: Constraining the Use of Options (2)



2.2.5.3 Provop: Guaranteeing Soundness of Configured Variants (1)

Problems and Premises:

- Soundness of configurable process variants has to be ensured considering
  - context dependencies
  - option constraints
  - syntactical correctness notions
- Several different process meta models with specific soundness and correctness criteria
  - generic approach needed
  - using existing verification techniques

Solution: Meta model independent framework to guarantee soundness of a process family

Step 1: Identify valid context descriptions

Step 2: Calculate corresponding sets of options

Step 3: Check whether options comply with constraints

Step 4: Apply option set to base process

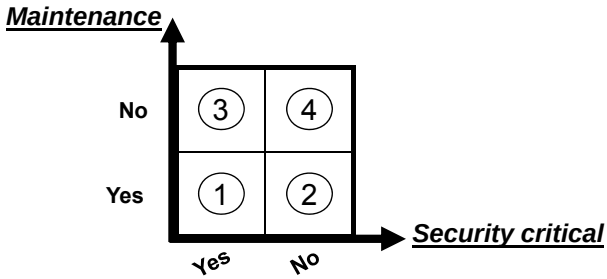
Step 5: Check soundness of variant models

2.2.5.3 Provop: Guaranteeing Soundness of Configured Variants (2)

Step 1: Identify valid context descriptions → **Step 2:** Calculate corresponding sets of options → **Step 3:** Check whether options comply with constraints → **Step 4:** Apply option set to base process → **Step 5:** Check soundness of variant models

Context model	
Context variable	Value range
Maintenance	Yes, No
Security critical	Yes, No

CONTEXT RULE:
IF Security critical = „Yes"
THEN Maintenance <> „No"



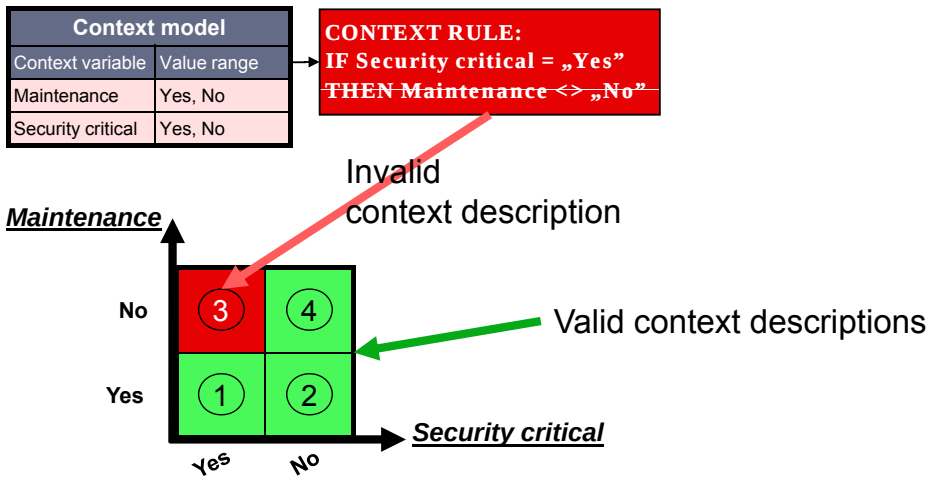
Hallerbach et al. [HBR09a]

2.2.5.3 Provop: Guaranteeing Soundness of Configured Variants (3)

Step 1: Identify valid context descriptions → **Step 2:** Calculate corresponding sets of options → **Step 3:** Check whether options comply with constraints → **Step 4:** Apply option set to base process → **Step 5:** Check soundness of variant models

Context model	
Context variable	Value range
Maintenance	Yes, No
Security critical	Yes, No

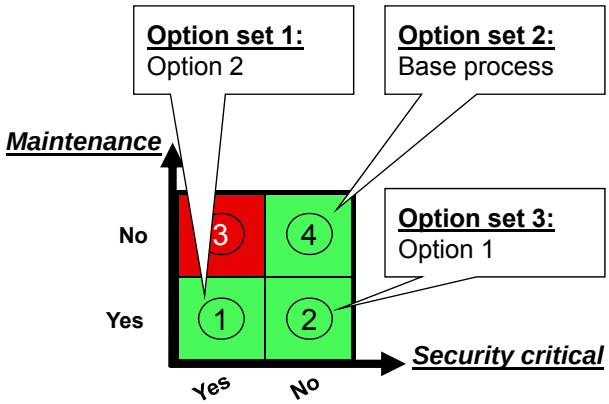
CONTEXT RULE:
IF Security critical = „Yes"
~~**THEN Maintenance <> „No"**~~



Hallerbach et al. [HBR09a]

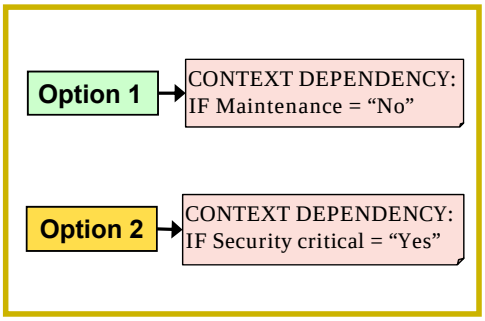
2.2.5.3 Provop: Guaranteeing Soundness of Configured Variants (4)

Step 1: Identify valid context descriptions → **Step 2:** Calculate corresponding sets of options → **Step 3:** Check whether options comply with constraints → **Step 4:** Apply option set to base process → **Step 5:** Check soundness of variant models



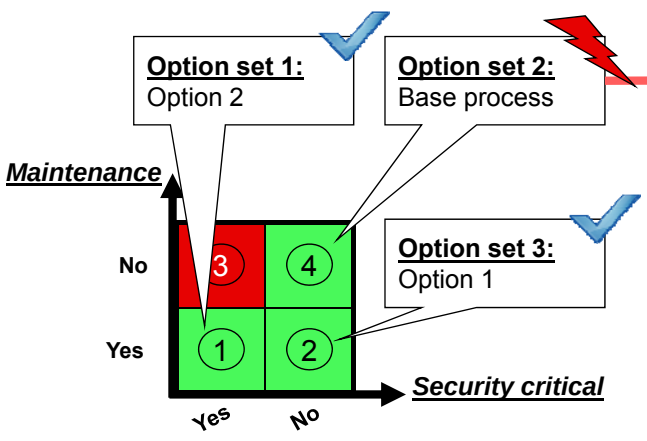
Hallerbach et al. [HBR09a]

Context dependencies:



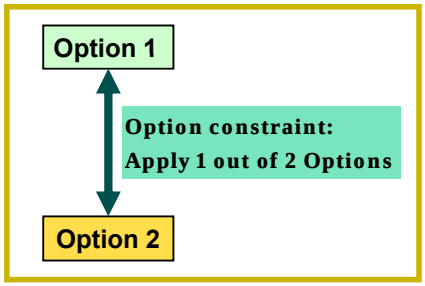
2.2.5.3 Provop: Guaranteeing Soundness of Configured Variants (5)

Step 1: Identify valid context descriptions → **Step 2:** Calculate corresponding sets of options → **Step 3:** Check whether options comply with constraint → **Step 4:** Apply option set to base process → **Step 5:** Check Soundness of variant models



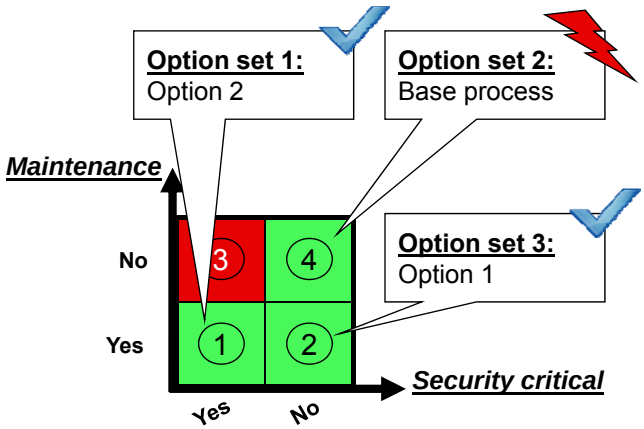
Hallerbach et al. [HBR09a]

Option constraints:



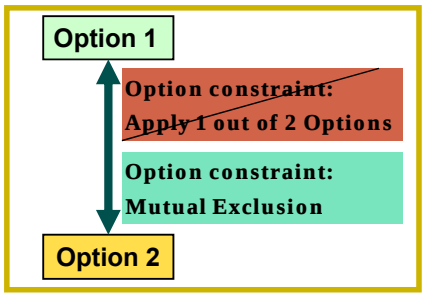
2.2.5.3 Provop: Guaranteeing Soundness of Configured Variants (6)

Step 1: Identify valid context descriptions → **Step 2:** Calculate corresponding sets of options → **Step 3:** Check whether options comply with constraint → **Step 4:** Apply option set to base process → **Step 5:** Check soundness of variant models



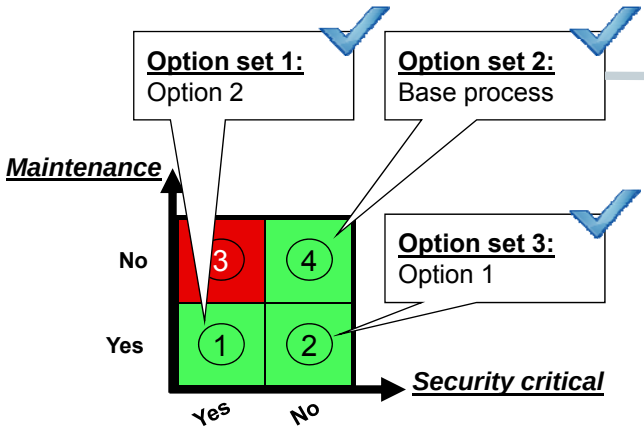
Hallerbach et al. [HBR09a]

Option constraints:



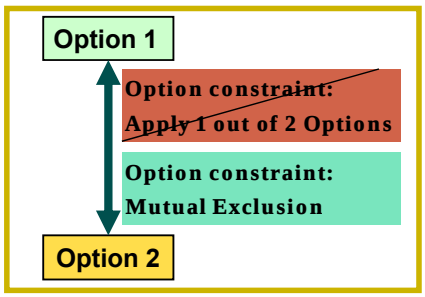
2.2.5.3 Provop: Guaranteeing Soundness of Configured Variants (7)

Step 1: Identify valid context descriptions → **Step 2:** Calculate corresponding sets of options → **Step 3:** Check whether options comply with constraint → **Step 4:** Apply option set to base process → **Step 5:** Check soundness of variant models



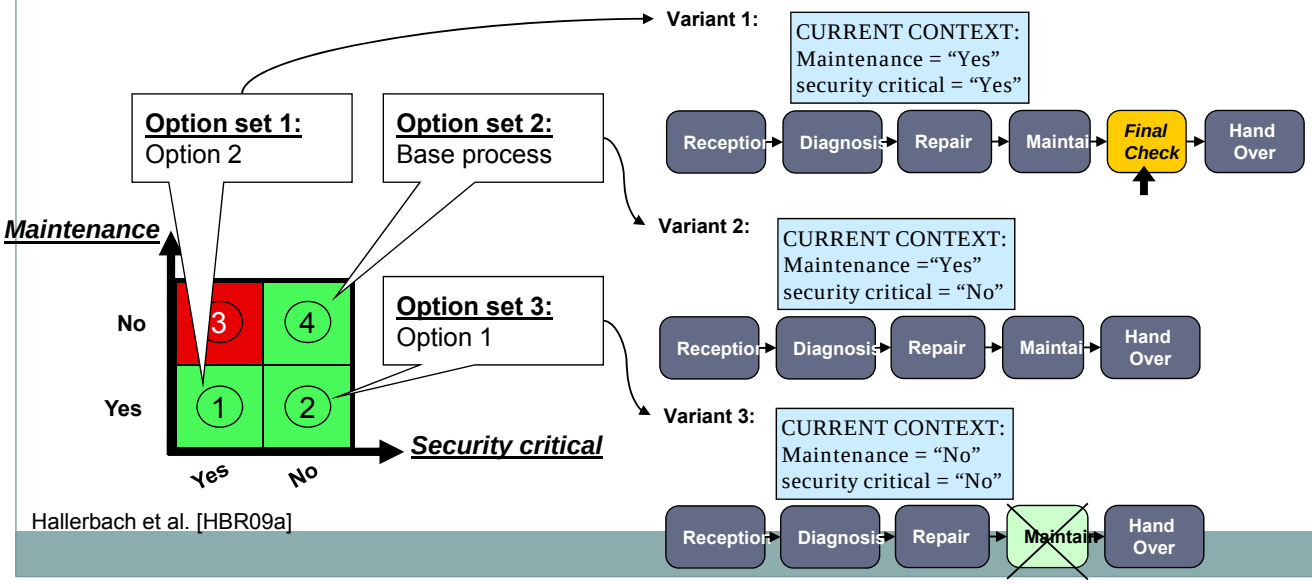
Hallerbach et al. [HBR09a]

Option constraints:



2.2.5.3 Provop: Guaranteeing Soundness of Configured Variants (8)

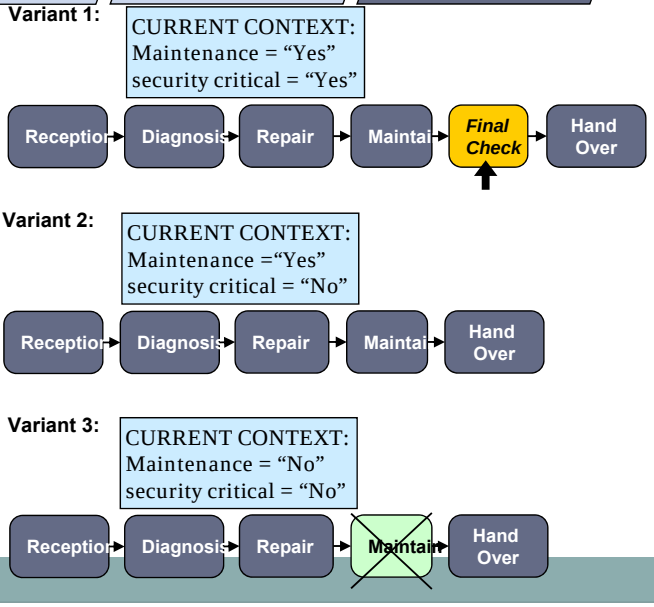
Step 1: Identify valid context descriptions → **Step 2:** Calculate corresponding sets of options → **Step 3:** Check whether options comply with constraints → **Step 4:** Apply option set to base process → **Step 5:** Check soundness of variant models



Hallerbach et al. [HBR09a]

2.2.5.3 Provop: Guaranteeing Soundness of Configured Variants (9)

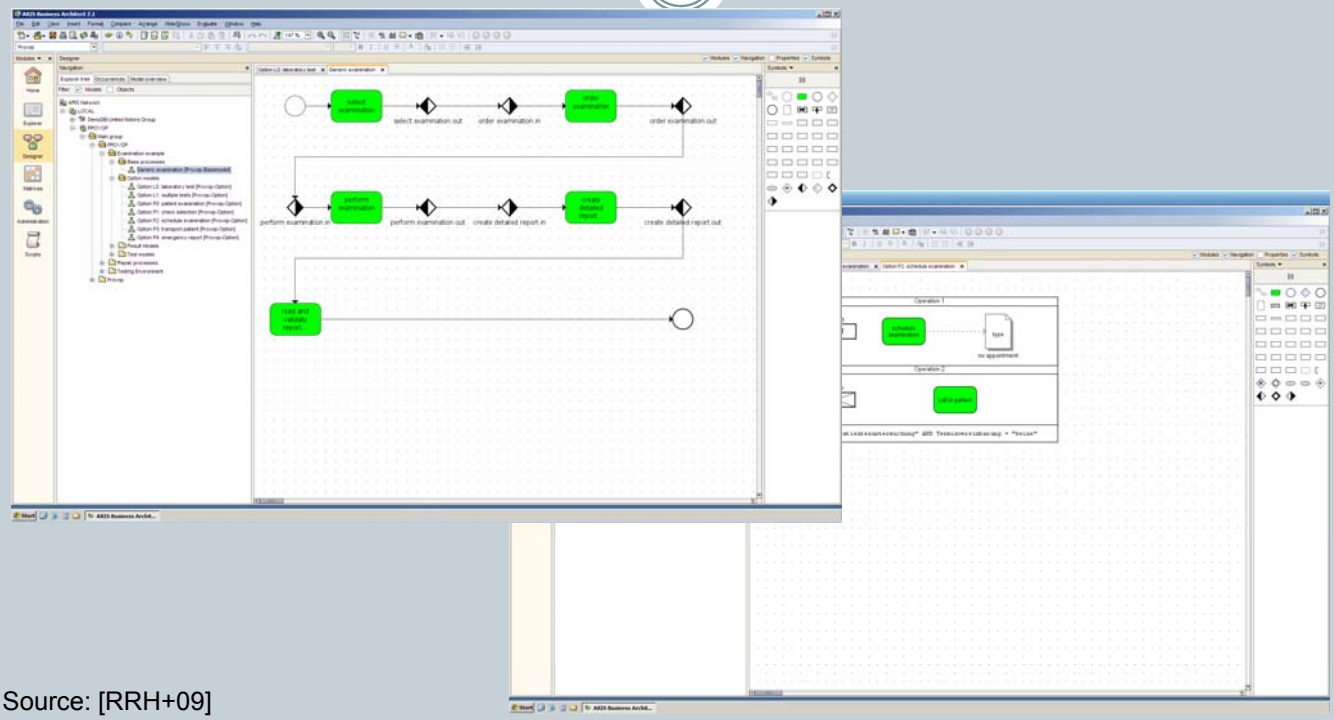
Step 1: Identify valid context descriptions → **Step 2:** Calculate corresponding sets of options → **Step 3:** Check whether options comply with constraints → **Step 4:** Apply option set to base process → **Step 5:** Check soundness of variant models



- Depends on soundness criteria of underlying process meta model
- Using meta-model specific soundness checking algorithms

Hallerbach et al. [HBR09a]

2.2.5.4 The Provop Approach: Proof-of-Concept Prototype



Source: [RRH+09]

Chapter 2: The Imperative Approach ...

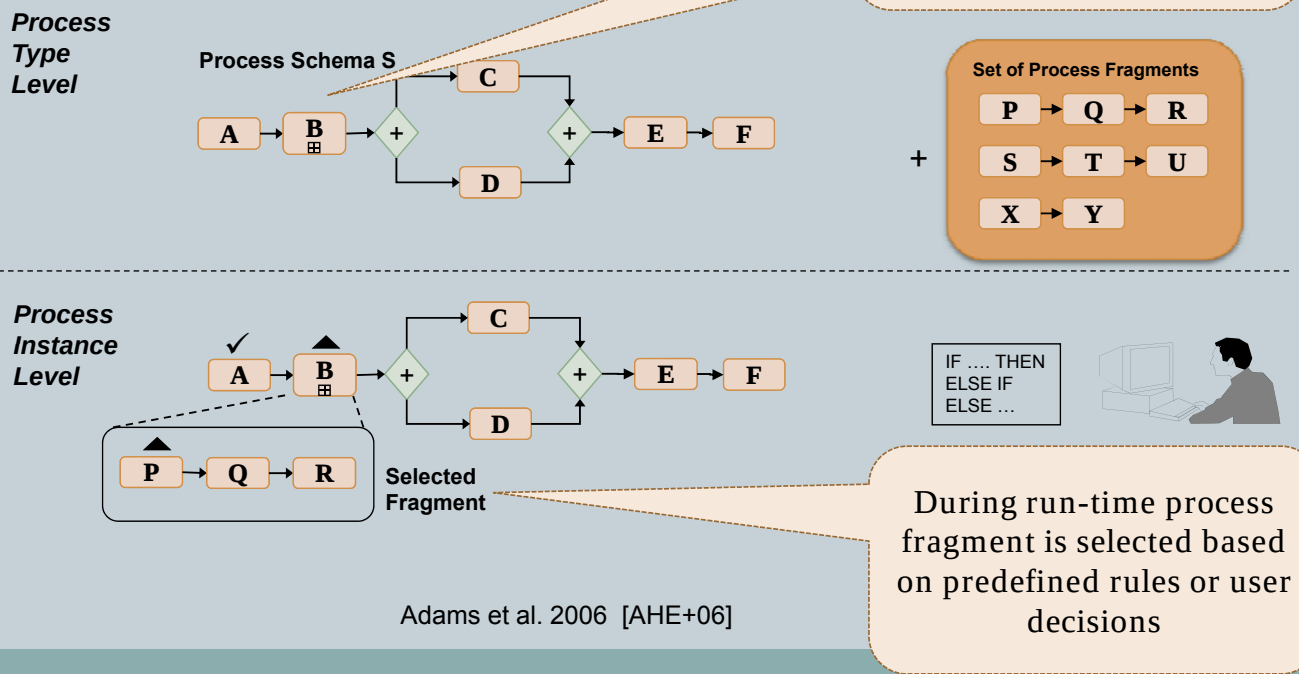
2.3 Dealing with Uncertainty

- Irreversible decisions / High stakes constraints
  - Deferring decisions to the last responsible moment
- Easy to change decisions
  - Probe and adapt

Poppendieck & Poppendieck, 2006 [PoPo06]

2.3.1 Late Binding

Process model contains placeholder activity, which needs to be refined during run-time

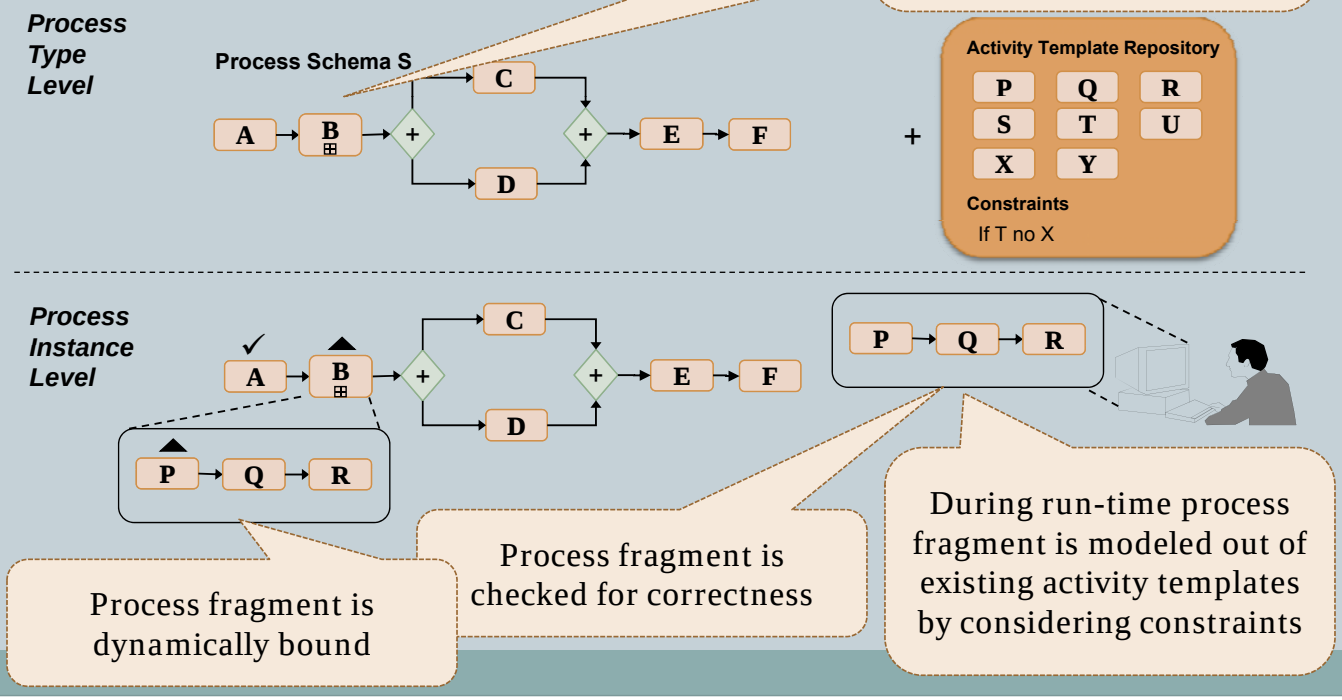


During run-time process fragment is selected based on predefined rules or user decisions

Adams et al. 2006 [AHE+06]

2.3.2 Late Modelin

Sadiq et al. 2005 [LuSa06, LuSa07a, LuSa07, SSO05]



Process model contains placeholder activity, which needs to be refined during run-time

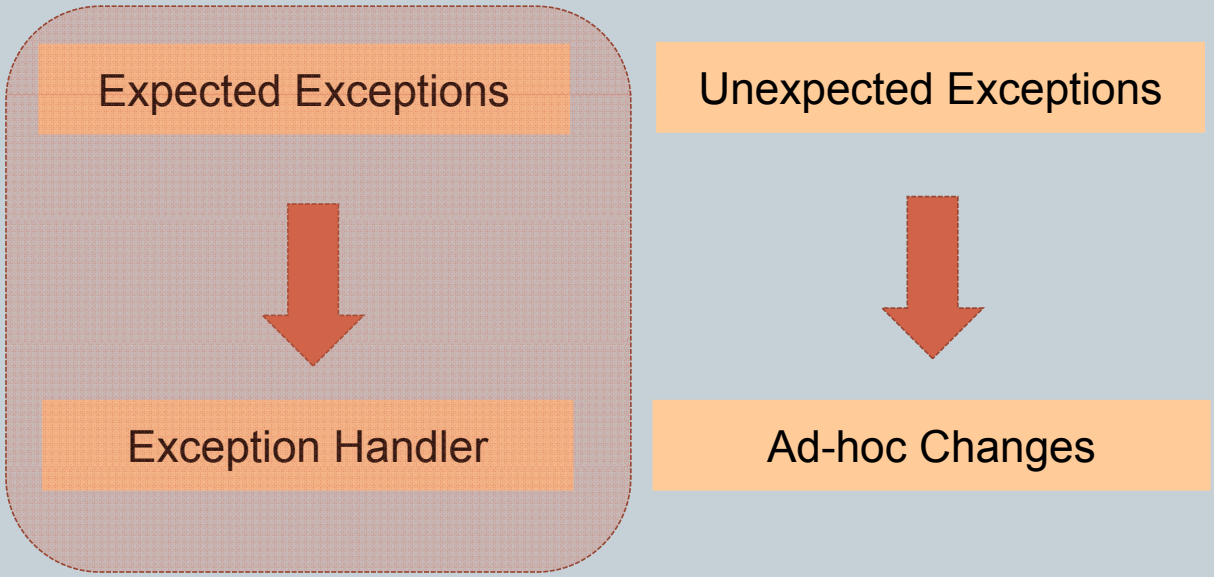
Process fragment is dynamically bound

Process fragment is checked for correctness

During run-time process fragment is modeled out of existing activity templates by considering constraints

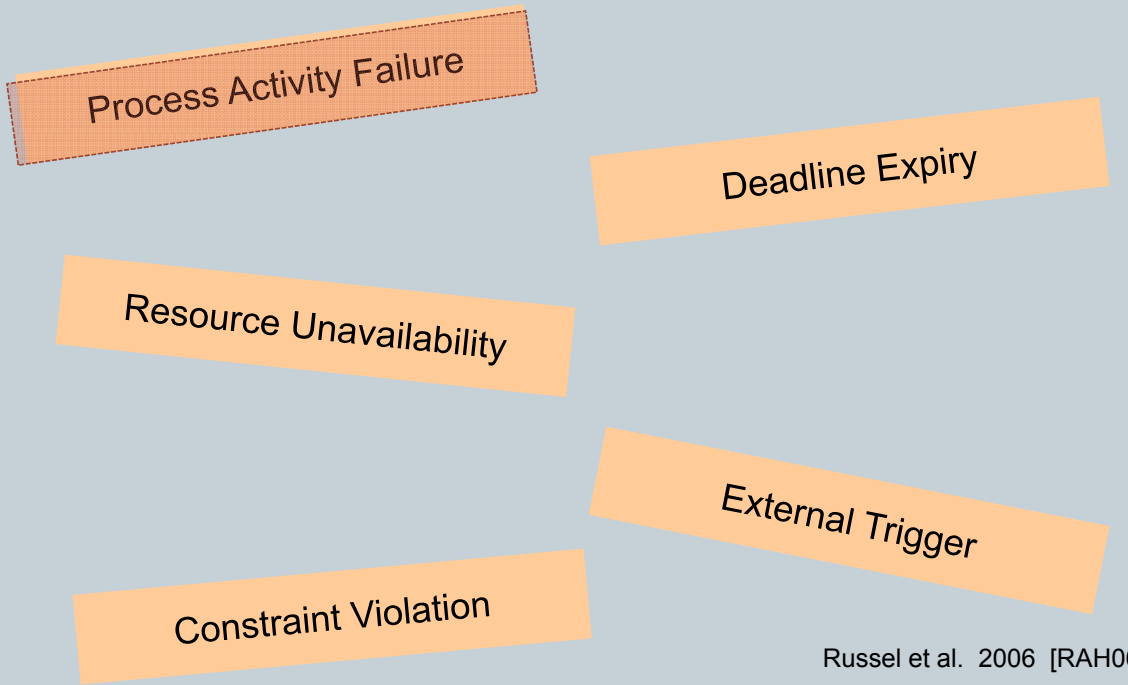
Chapter 2: The Imperative Approach ...

2.4 Handling Expected Process Exceptions During Runtime



[CaPo99, ICB+07, RDB03]

2.4.1 Some Examples of Exception Types

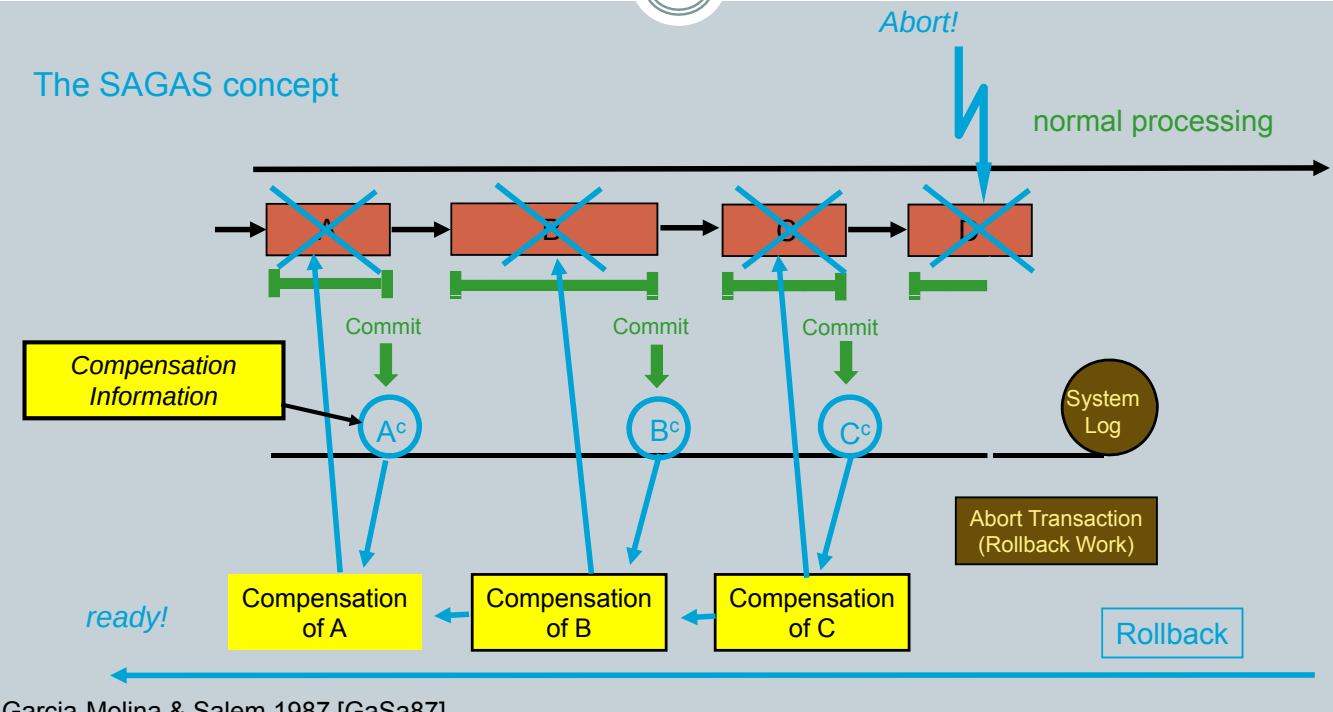


Russel et al. 2006 [RAH06]

2.4.2 Activity Failures

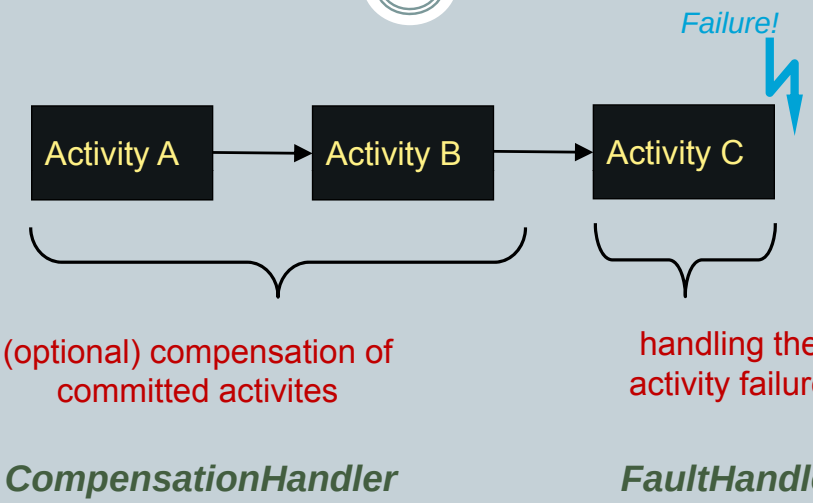
- During the execution of a process instance a corresponding activity might fail (e.g., for semantical reasons or technical problems with an application service)
- When such a failure occurs, usually, the process instance cannot be continued as planned, but an exception handling is needed – Examples:
  - Controlled abortion of the process instance
  - Backward Forward Recovery (e.g., using compensation techniques)
- In the following we exemplarily discuss exception handling techniques provided in **WS-BPEL** (Business Process Execution Languge for Web Services)

2.4.2.1 Dealing with Activity Failures through Compensation



Garcia-Molina & Salem 1987 [GaSa87]

2.4.2.2 WS-BPEL Concepts for Coping with Activity Failures (1)



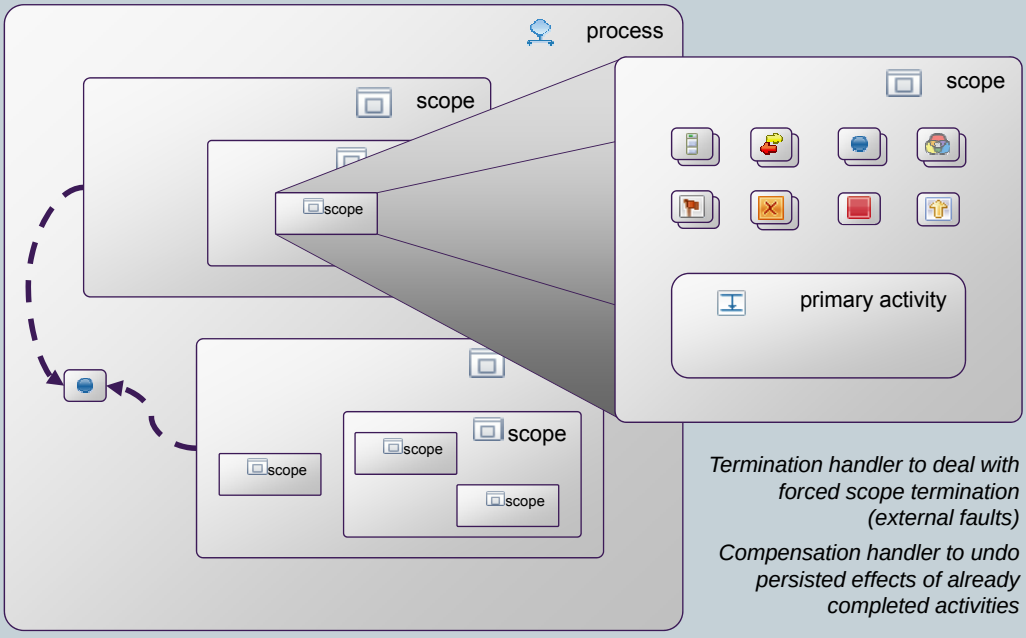
2.4.2.2 WS-BPEL Concepts for Coping with Activity Failures (2)

Scopes provide a context which influences the execution behavior of its enclosed activities

Local declarations: partner links, message exchanges, variables, correlation sets

Local handlers: event handlers, fault handlers, a termination handler, and a compensation handler

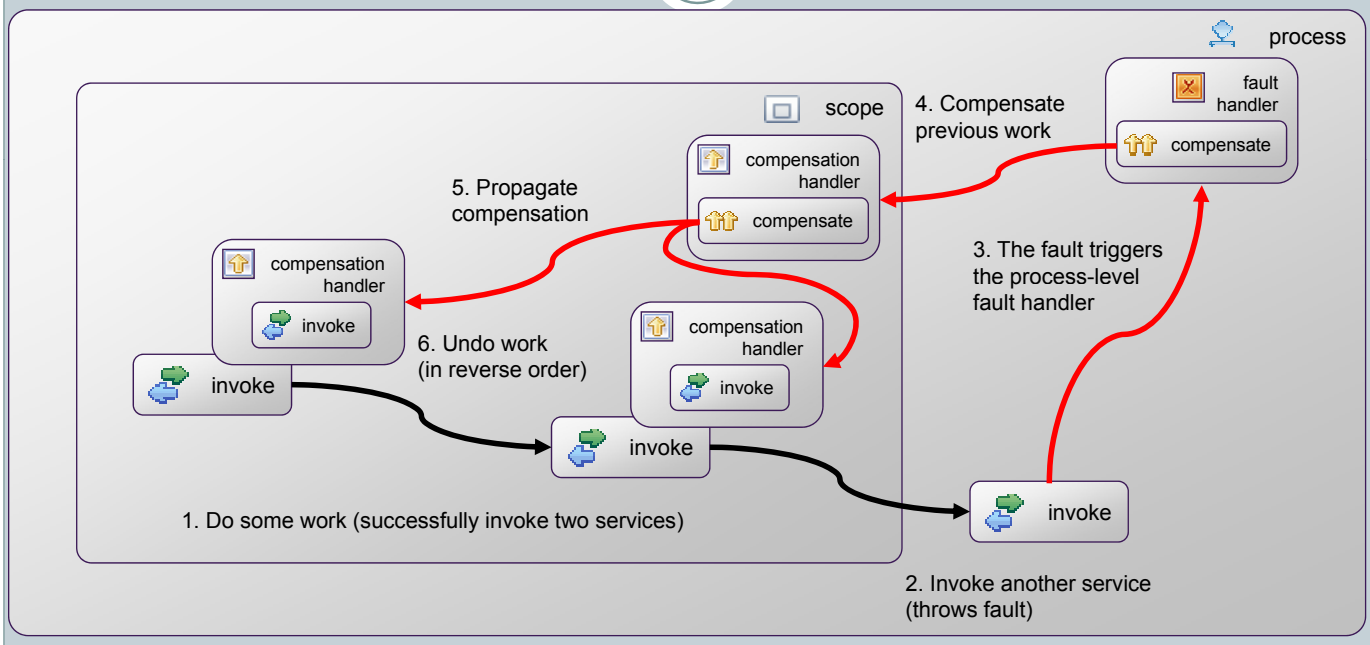
Isolated scopes provide control of concurrent access to shared resources



Termination handler to deal with forced scope termination (external faults)

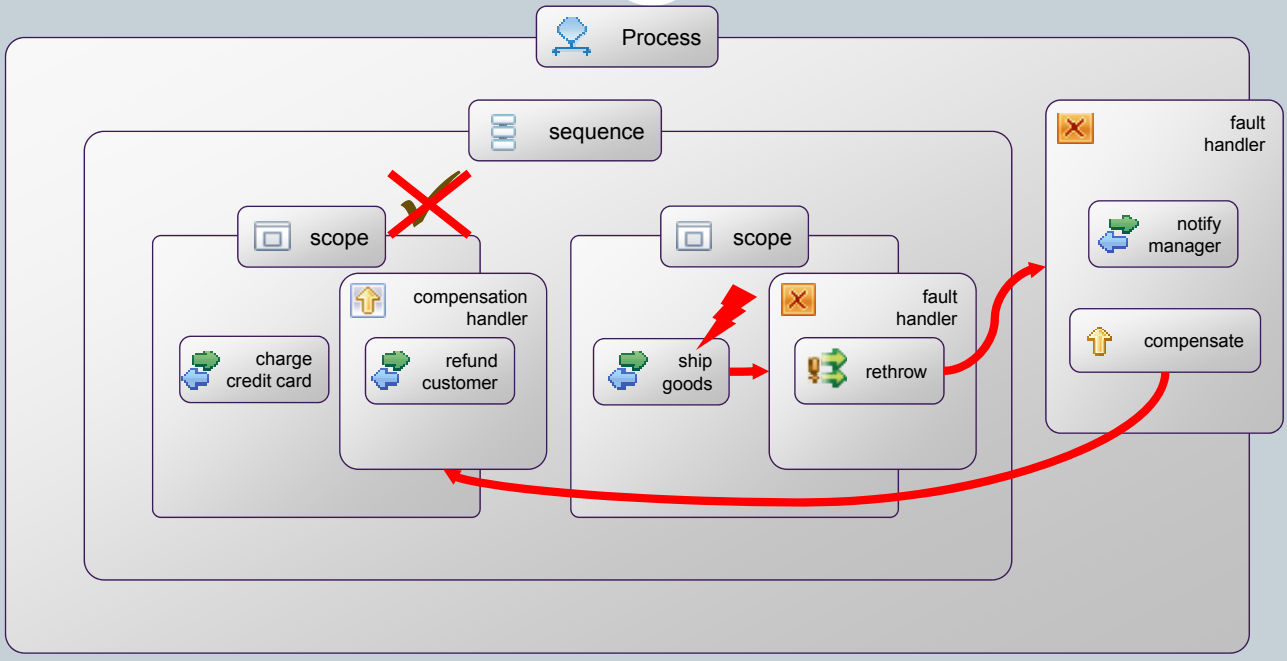
Compensation handler to undo persisted effects of already completed activities

2.4.2.3 Fault and Compensation Handling in WS-BPEL



☞ ***This example shows the default compensation behavior supported by BPEL; i.e., a completed scope is compensated by invoking the compensation handlers of its constituting activities in reverse order. However, a more specific compensation handler for a scope may be provided as well (e.g., only compensating some of the already completed activities or invoking a specific process dealing with the exception).***

2.4.2.4 Nested Fault Handling and Compensation in WS-BPEL



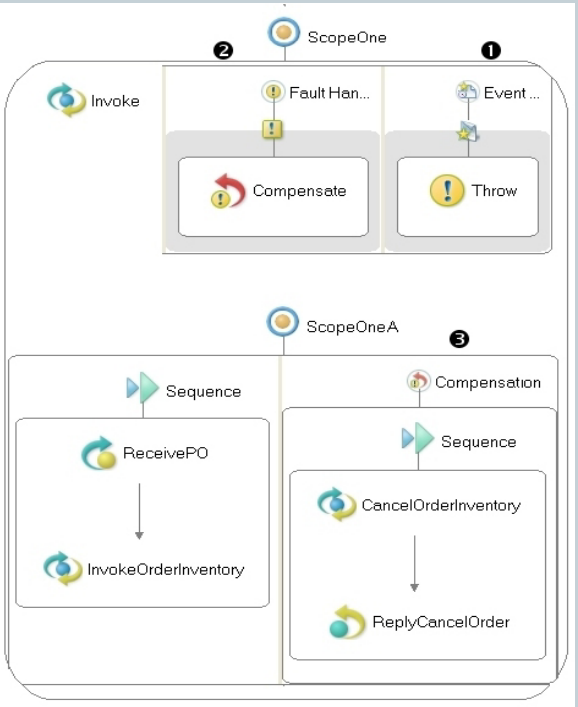
2.4.2.5 Reacting to Asynchronous Events

Example: Events in `ActiveBPEL`

With an `EventHandler` the termination of a process instance and the compensation activities required in this context can be triggered!

1. When the event handler receives an order cancellation message, it throws a fault to the fault handler.
2. The fault handler executes a compensate activity for the previously completed scope to which it is linked.
3. The compensation handler for the completed scope rolls back the work of the `InvokeOrderInventory` service.

Source: activebpel.org



Chapter 2: The Imperative Approach ...

2.5 Enabling Ad-hoc Process Changes During Runtime

Expected Exceptions

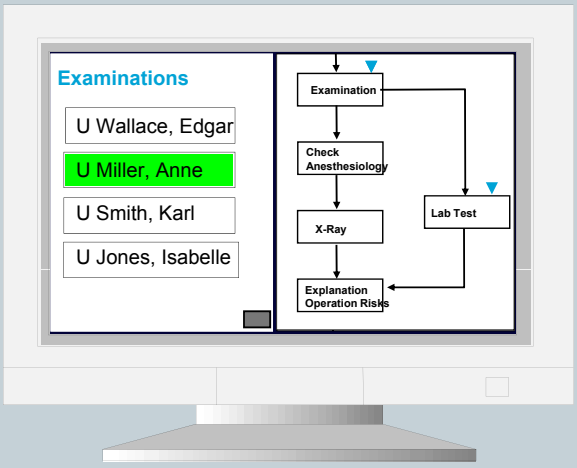
Exception Handler

Unexpected Exceptions

Ad-hoc Changes

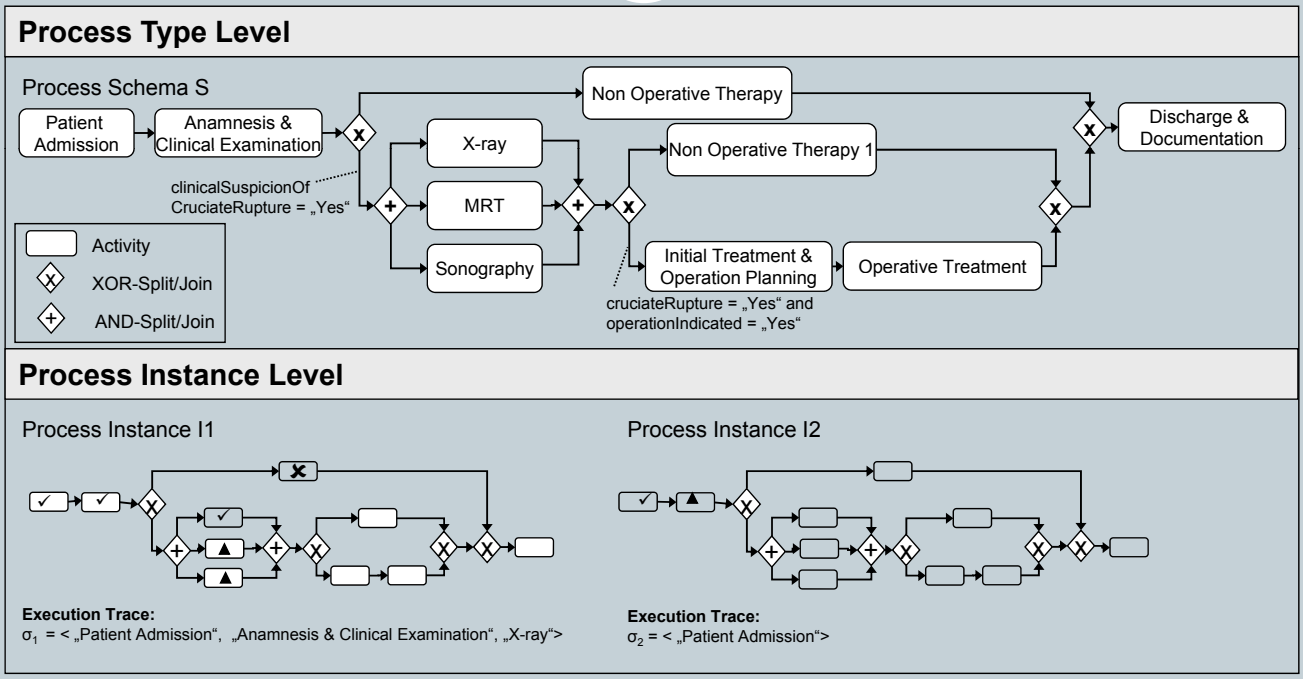
[ReDa98, ReDa09]

2.5.1 User View on an Ad-hoc Process Change

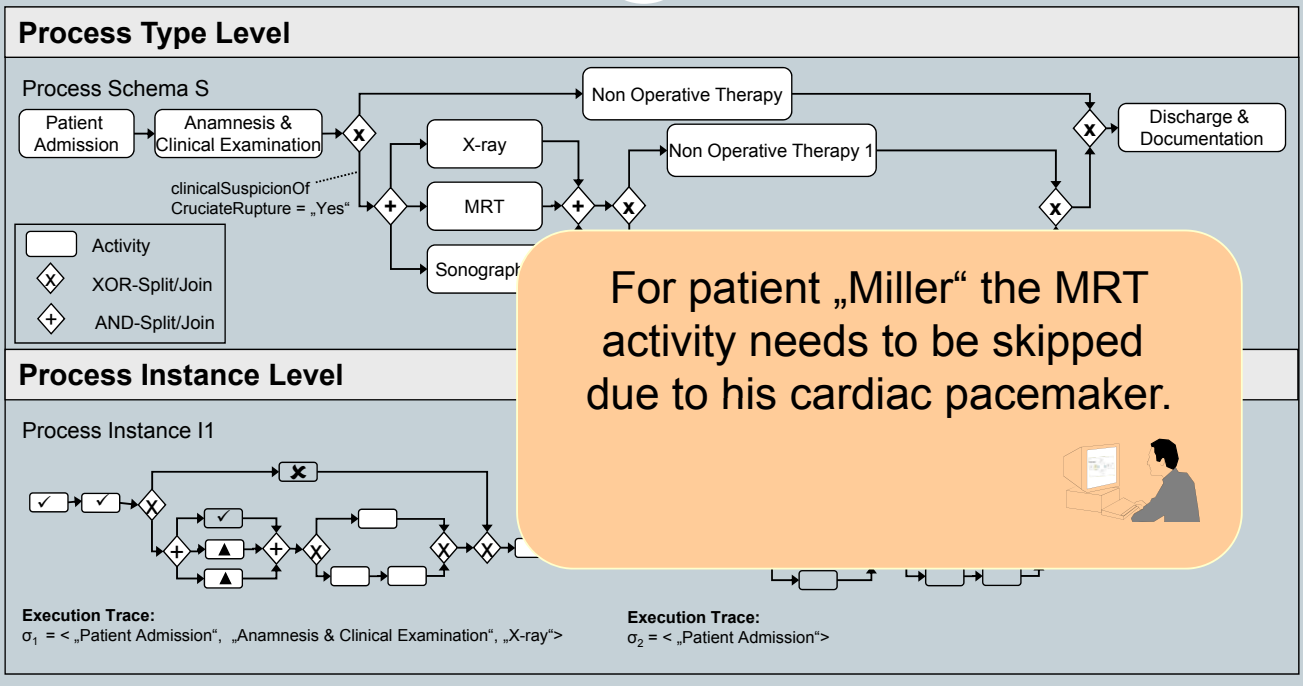


User View on an Ad-hoc Change!

2.5.2 System View on Ad-hoc Process Changes (1)

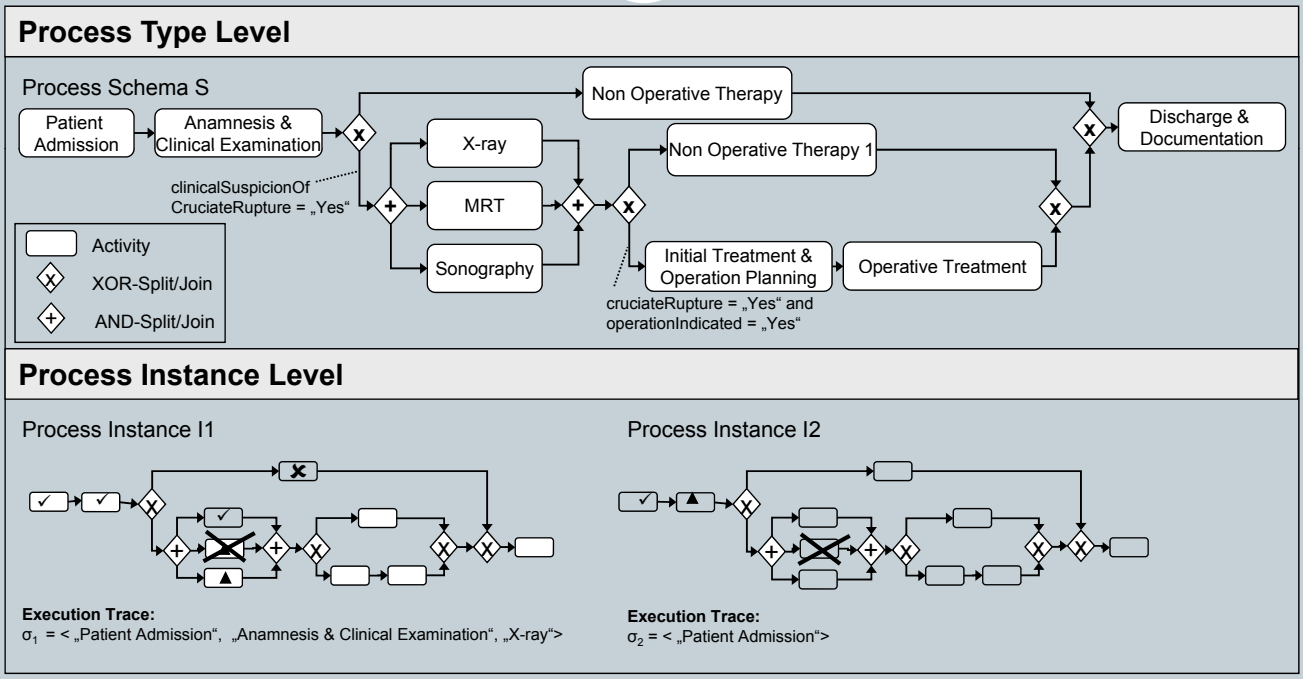


2.5.2 System View on Ad-hoc Process Changes (2)



For patient „Miller“ the MRT activity needs to be skipped due to his cardiac pacemaker.

2.5.2 System View on Ad-hoc Process Changes (3)

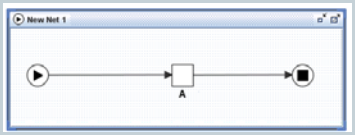


2.5.3 Fundamental Requirements for Ad-hoc Changes

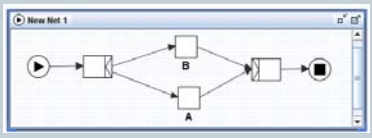
- High level of abstraction
- Correctness of instance changes
- User assistance & change reuse
- Controlling access to process change functions
- Concurrency Control
- Application Programming Interfaces

2.5.3.1 Enabling Ad-hoc Changes at High Level of Abstraction (1)

High-level vs. Low-level Change Operations



Insert B parallel to A



High-level Change Operation

1 High-Level Change Operation

```
parallelInsert (S, B, A)
```

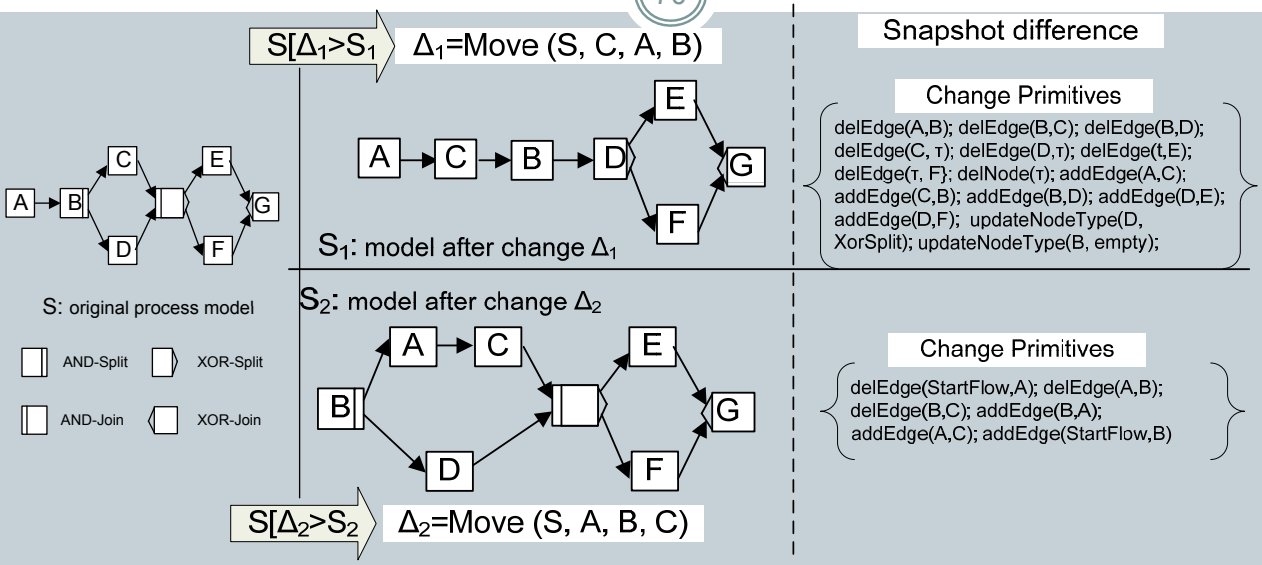
Change Primitives

9 Change Primitives

```
Add Node (3x)
Move Edge (2x)
Add Edge (4x)
```

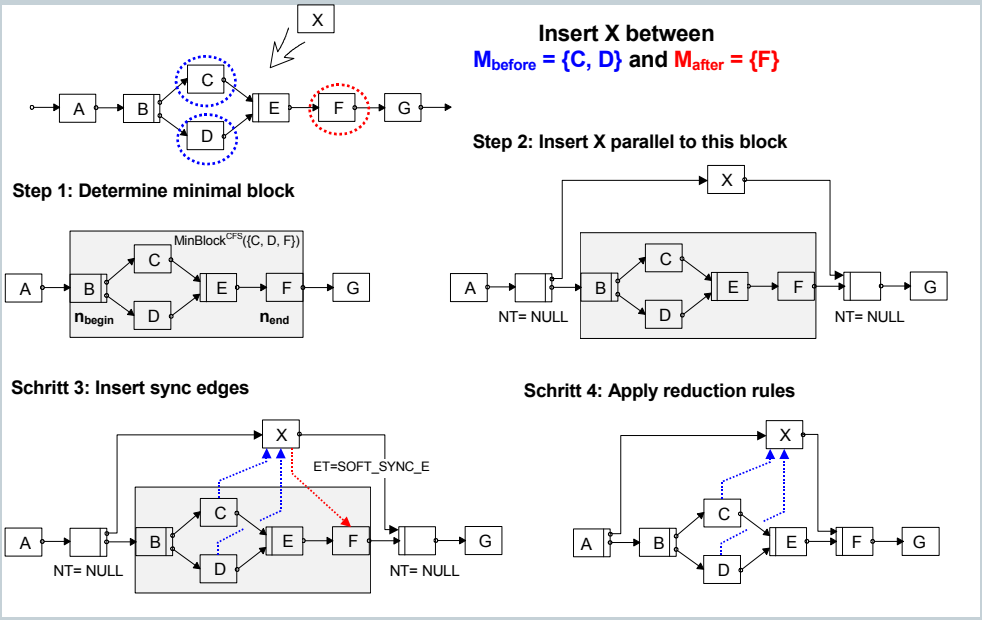
[WeRR07, WeRR08]

2.5.3.1 Enabling Ad-hoc Changes at High Level of Abstraction (2)



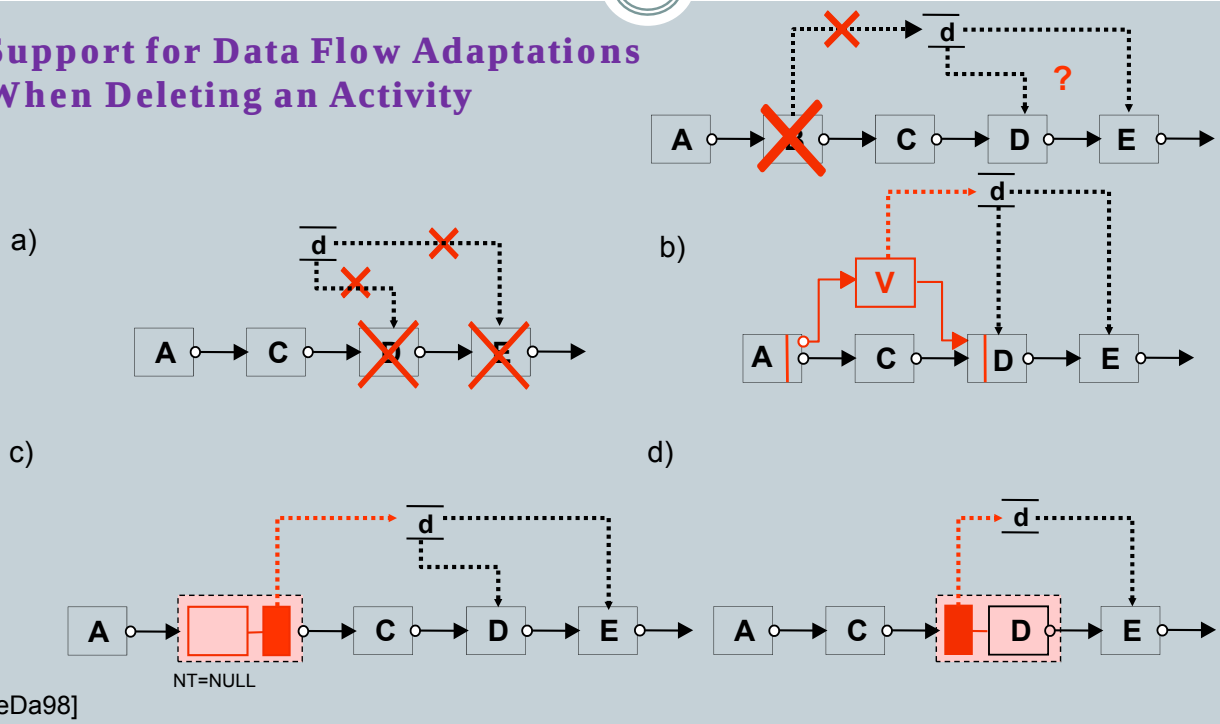
1. High-level change operations guarantee soundness
2. High-level change operations provide richer syntactical meanings
3. The number of changes are more meaningful

Example of a High-Level Change Operation in ADEPT



[ReDa98]

Support for Data Flow Adaptations When Deleting an Activity

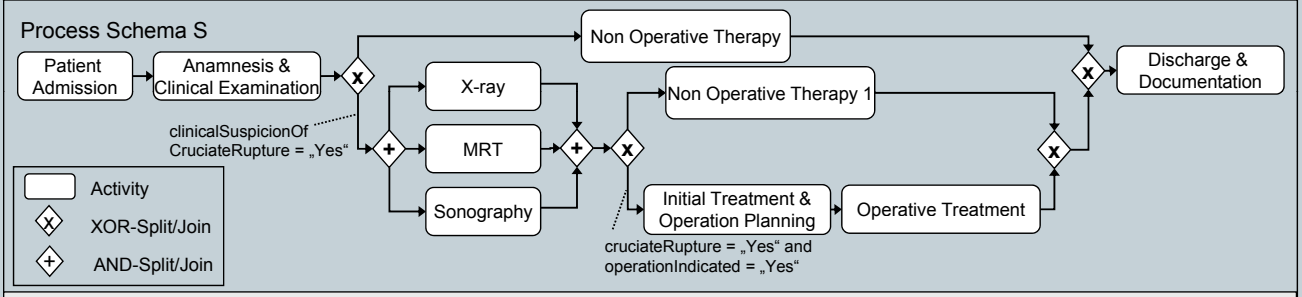


[ReDa98]

2.5.3.2 Correctness of Process Instance Changes (1)

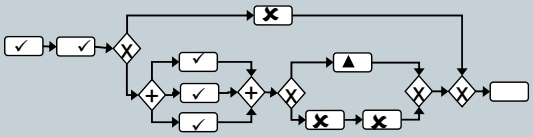
Process Type Level

Process Schema S



Process Instance Level

Process Instance I3



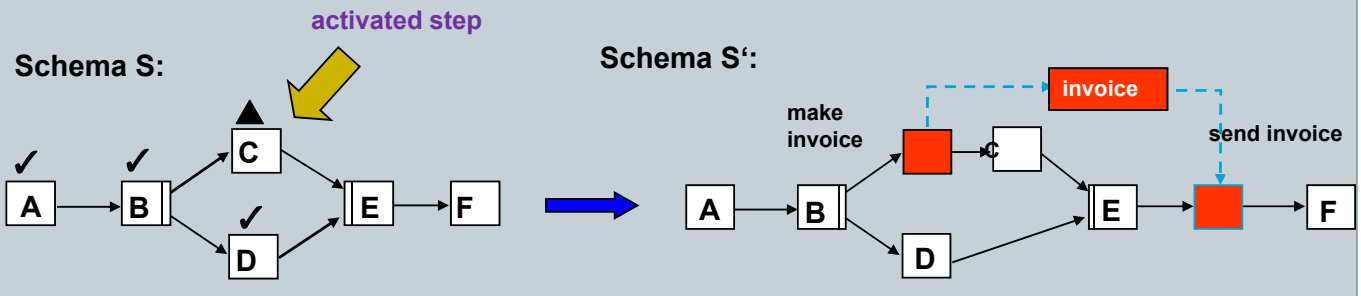
Execution Trace:
σ_3 = < „Patient Admission“, „Anamnesis & Clinical Examination“, „MRT“, „X-ray“, „Sonography“>

I3 is not compliant with change Delete (I3, MRT)

[ReDa98]

2.5.3.2 Correctness of Process Instance Changes (2)

Ensuring Dynamic Correctness



May the depicted schema change be propagated to the process instance?

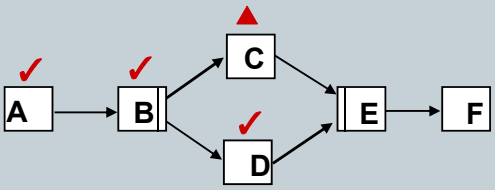
- ⇨ **Need for general correctness criterion**
- ⇨ **Trace Equivalence & Compliance**

[ReDa98, RRW08a, RRD04a, RRD04b]

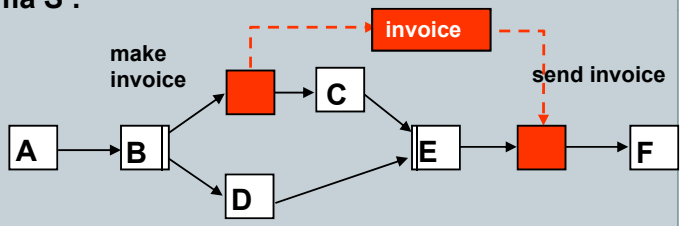
2.5.3.2 Correctness of Process Instance Changes (3)

Trace Equivalence & Compliance

Schema S:



Schema S‘:



execution log (trace) of instance I on S (simplified):

<A> , <B> , <D> ⇨ **Trace reproducible on new schema?**

More complicated: loop backs

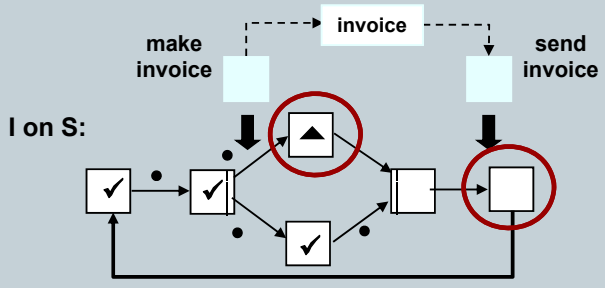
(see: S. Rinderle, M. Reichert, P. Dadam: *Flexible Support of Team Processes By Adaptive Workflow Systems.* Distributed and Parallel Databases, 16(1):91-116, 2004)

Further challenges: - How to efficiently check for compliance?
- How to efficiently migrate process instances?

[RRD04a, RRD04b]

2.5.3.2 Correctness of Process Instance Changes (4)

How to efficiently Check For Compliance? – The ADEPT Approach



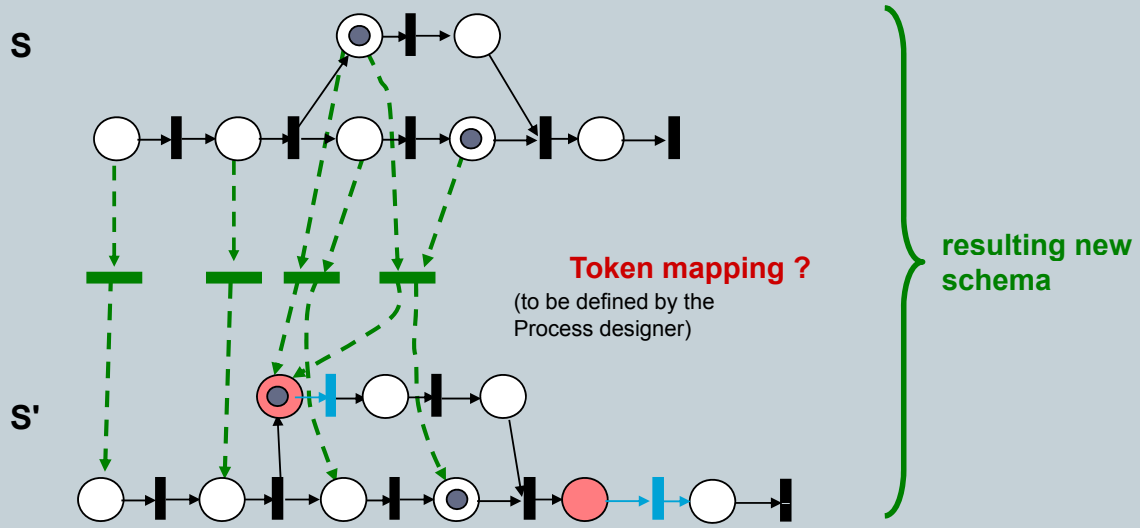
☞ **Instance I compliant with Modified Process Schema**

Change Operation Δ ...	... and related compliance conditions
addActivity(S, act, Preds, Succs)	[∀ n ∈ Preds: NS(n) = `Disabled`] ∨ [∀ n ∈ Succs: (NS(n) ∈ {`NotActivated`, `Activated`}) ∨ (NS(n) = `Disabled` ∧ ∀ m ∈ succ(S,n): NS(m) ∈ {`NotActivated`, `Activated`, `Disabled`})]

2.5.3.2 Correctness of Process Instance Changes (5)

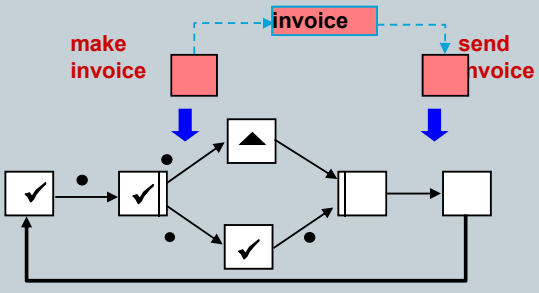
How to efficiently migrate compliant flow instances?

- ... and one "solution" from the Petri Net world (Ellis et al., 2000)



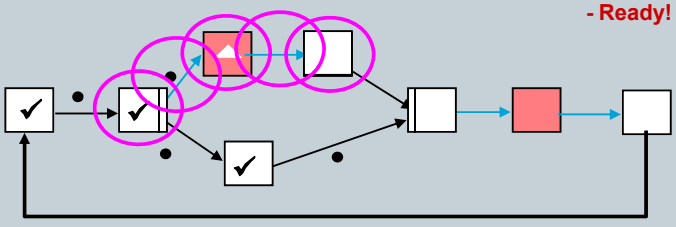
2.5.3.2 Correctness of Process Instance Changes (6)

How to efficiently migrate a process instance? - The ADEPT Approach



(running process instance)

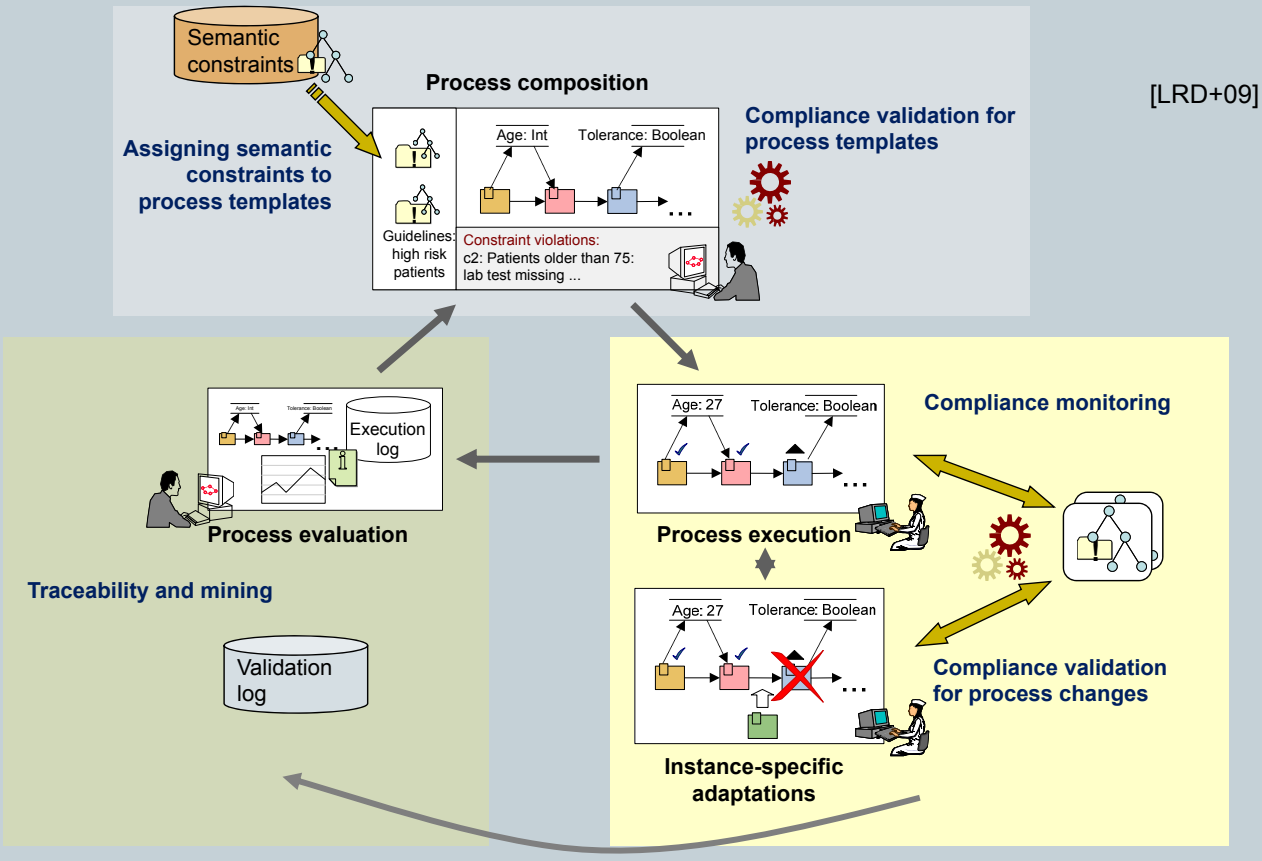
- (Logical) application of the structural change to running process instance (by system)
- Reevaluation of instance state (by system)
- Ready!



2.5.3.2 Correctness of Process Instance Changes (7)

- So far, we have only considered compliance of (modified) process instances with syntactical constraints
- Also important: Ensuring compliance of (modified) processes with semantical constraints ☞ SeaFlows [LRD+09]

SeaFlows: Supporting Semantic Constraints in Adaptive Process Management Systems



[LRD+09]

2.5.3.3 User Assistance & Change Reuse (1)

The ProCycle (= ADEPT + CBRFlow) Approach for Assisting Users in Defining and Reusing Changes:

- Annotate ad-hoc changes with information about the reasons for their introduction
- Support users in retrieving past ad-hoc changes applied in similar context
- Assist users in reusing a past ad-hoc change when coping with an exceptional situation

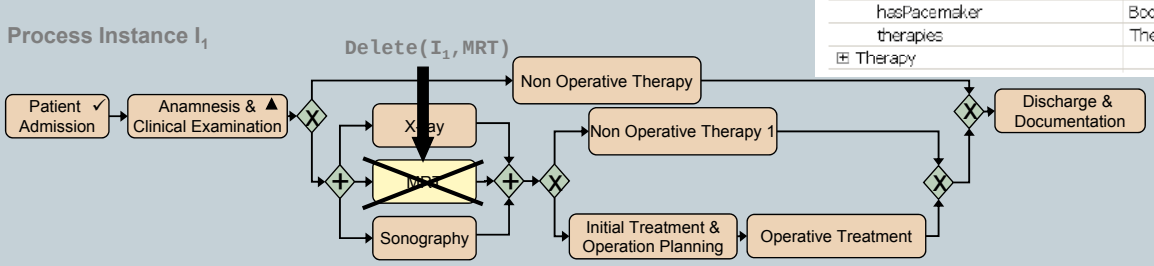
[RWR+05, WRWR09m WRR+05, WRW06, WWB04]

2.5.3.3 User Assistance & Change Reuse (2)

Memorization of instance deviations including their application context

Application Context Model

Object	Type	Min	Max
⊞ Diagnosis			
⊟ Patient			
age	Integer	1	1
problemList	ProblemList	1	1
weight	Integer	1	1
⊟ ProblemList			
diagnoses	Diagnosis	0	1000
hasPacemaker	Boolean	0	1
therapies	Therapy	0	1000
⊞ Therapy			



Case c_1

pd_{c_1} = The treatment of cruciate ruptures routinely includes a magnetic resonance tomography (MRT), an X-ray and a sonography. However, for a particular patient the MRT may have to be skipped as the respective patient has a cardiac pacemaker.

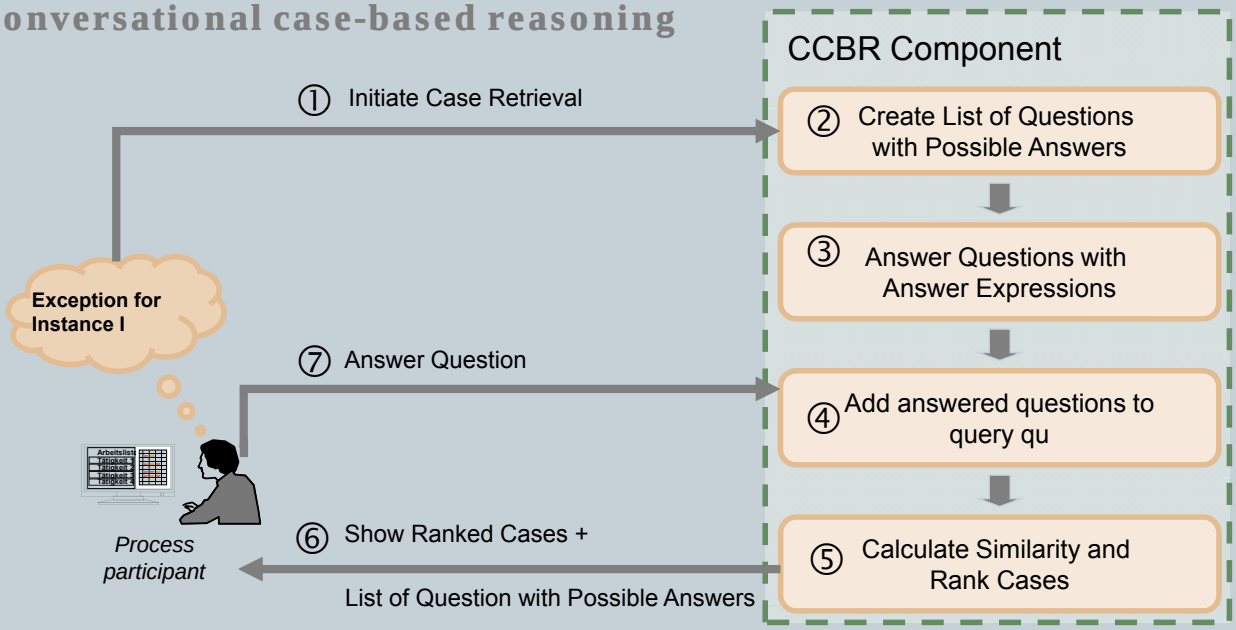
sol_{c_1} = <Delete(S_I,MRT)>

$qaSet_{c_1}$ = {(Does the patient have a cardiac pacemaker?, patient.problemList.hasPacemaker = 'Yes')}

$freq_{c_1}$ = 1

2.5.3.3 User Assistance & Change Reuse (3)

Semi-automated retrieval of similar instance deviations using conversational case-based reasoning



2.5.3.3 User Assistance & Change Reuse (4)

Retrieving similar instance deviations based on the actual context

$qaSet_{c_1}$ = {(Does the patient have a cardiac pacemaker?, Patient.problemList.hasPacemaker = 'Yes')}

Case c_1

$qaSet_{c_2}$ = {(Does the patient have fluid in the knee?, 'A significant amount'), (Does the patient have an acute effusion of the knee?, 'Yes')}

Case c_2

List of Questions with Possible Answers	
Question	**Possible Answers**
Does the patient have a cardiac pacemaker?	{Patient.problemList.hasPacemaker = ‚Yes', OTHERANSWER}
Does the patient have fluid in the knee?	{‚A significant amount', OTHERANSWER}
Does the patient have an acute effusion of the knee?	{‚Yes', OTHERANSWER}

②

Query qu'	
Question	**Given Answer**
Does the patient have a cardiac pacemaker?	OTHERANSWER
Does the patient have fluid in the knee?	‚A significant amount'

③

List of Retrieved Cases for Query qu'	
Case	**Appl. Context Similarity**
c2	50%
c1	0%

④

$$sim(qu,c) = \frac{1}{2} * \frac{same(qu, qaSet_c) - diff(qu, qaSet_c)}{|qaSet_c|} + 1$$

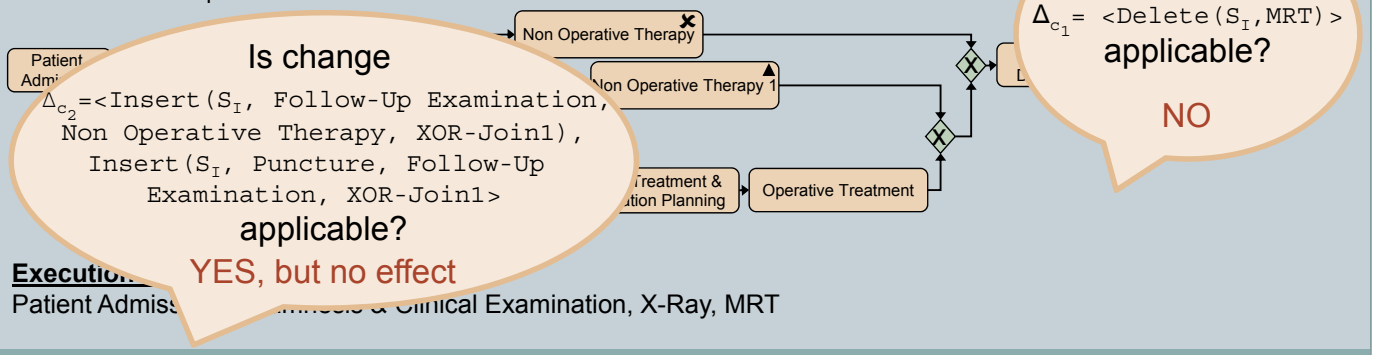
2.5.3.3 User Assistance & Change Reuse (5)

Retrieving similar instance deviations based on the actual context + status

List of Retrieved Cases		
Case	**Similarity**	
c2	75%	c_2 does not have any effect, but is adjustable
~~c1~~	~~0%~~	~~c_1 is not case compliant and not adjustable~~

Are the instance deviations of these cases compliant with the process instance to be modified?

Process Instance I_1

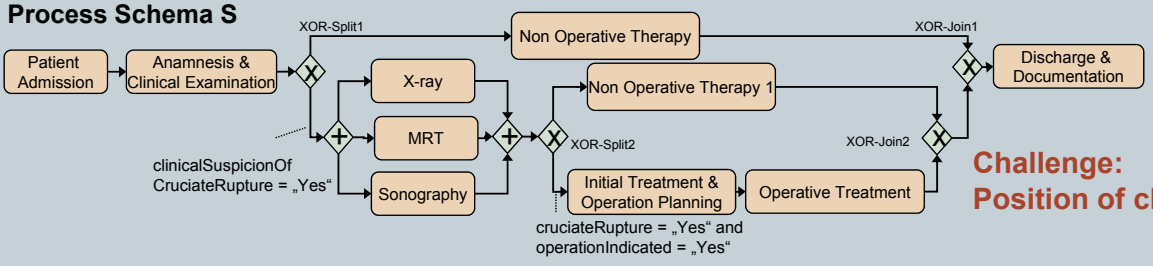


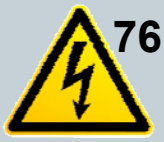
Execution:
Patient Admiss… …nical Examination, X-Ray, MRT

2.5.3.3 User Assistance & Change Reuse (6)

Deriving Type Changes from Frequently Occuring Instance Changes

Process Schema S





Challenge:
Position of change can vary

Case c_1 $freq_{c1} = 51$

```
...
Δc1  = <Delete(SI,MRT)>
```

Case c_2 $freq_{c2} = 41$

```
...
Δc2  = <Insert(SI, Follow-Up Examination,
           Non Operative Therapy, XOR-Join1),
       Insert(SI, Puncture, Follow-Up
           Examination, XOR-Join1>
```

Case c_3 $freq_{c3} = 35$

```
...
Δc3  = <Insert(SI, Follow-Up Examination,
           Non Operative Therapy1, XOR-Join2),
       Insert(SI, Puncture, Follow-Up
           Examination, XOR-Join2>
```

2.5.3.3 User Assistance & Change Reuse (7)

Deriving Type Changes from Frequently Occuring Instance Changes

Case c_2 $freq_{c2} = 41$

$qaSet_{c_2}$ = {(Does the patient have fluid in the knee?, 'A significant amount'), (Does the patient have an acute effusion of the knee?, 'Yes')}

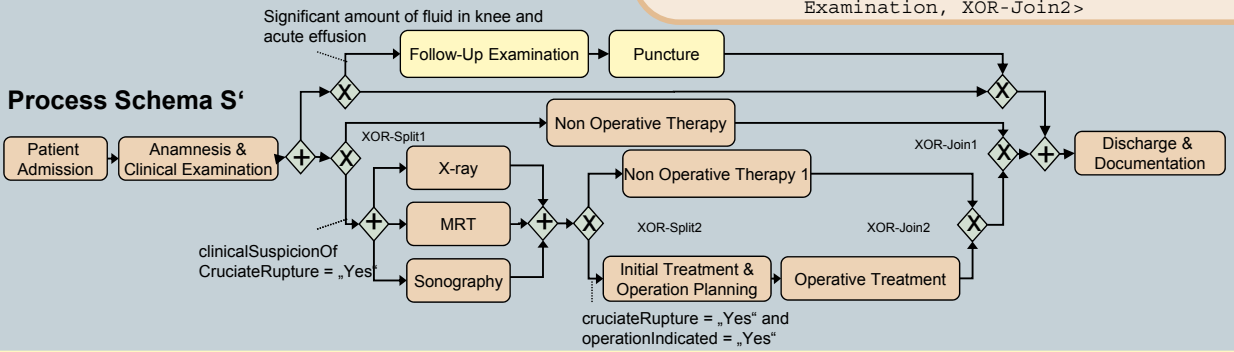
Δ_{c_2} = <Insert(S_I, Follow-Up Examination, Non Operative Therapy, XOR-Join1), Insert(S_I, Puncture, Follow-Up Examination, XOR-Join1>

Case c_3 $freq_{c3} = 35$

$qaSet_{c_3}$ = {(Does the patient have fluid in the knee?, 'A significant amount'), (Does the patient have an acute effusion of the knee?, 'Yes')}

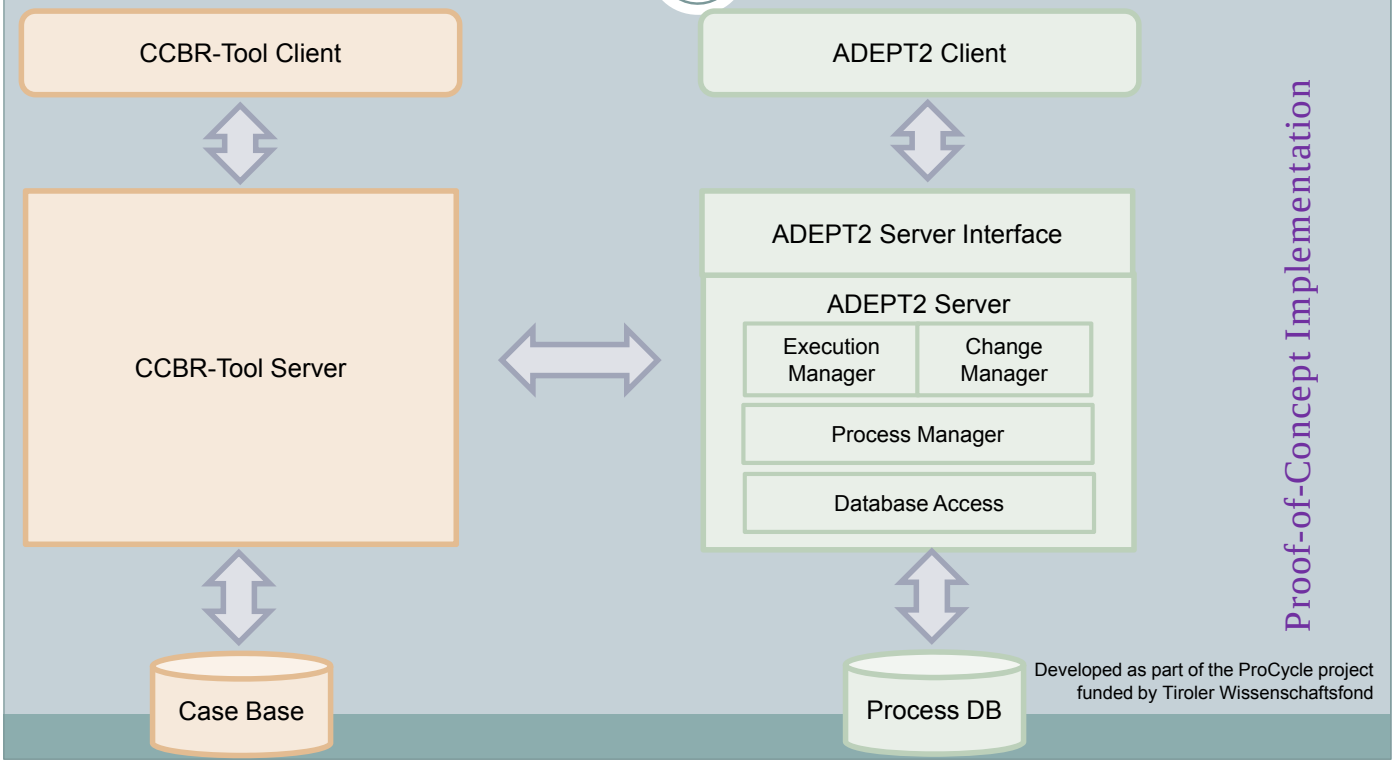
Δ_{c_3} = <Insert(S_I, Follow-Up Examination, Non Operative Therapy1, XOR-Join2), Insert(S_I, Puncture, Follow-Up Examination, XOR-Join2>

Process Schema S'



Δ_S = <CondInsert(S_I, Follow-Up Examination, Anamnesis & Clinical Examination, Discharge & Documentation), CondInsert(S_I, Puncture, Follow-Up Examination, Discharge & Documentation>

2.5.3.3 User Assistance & Change Reuse (8)

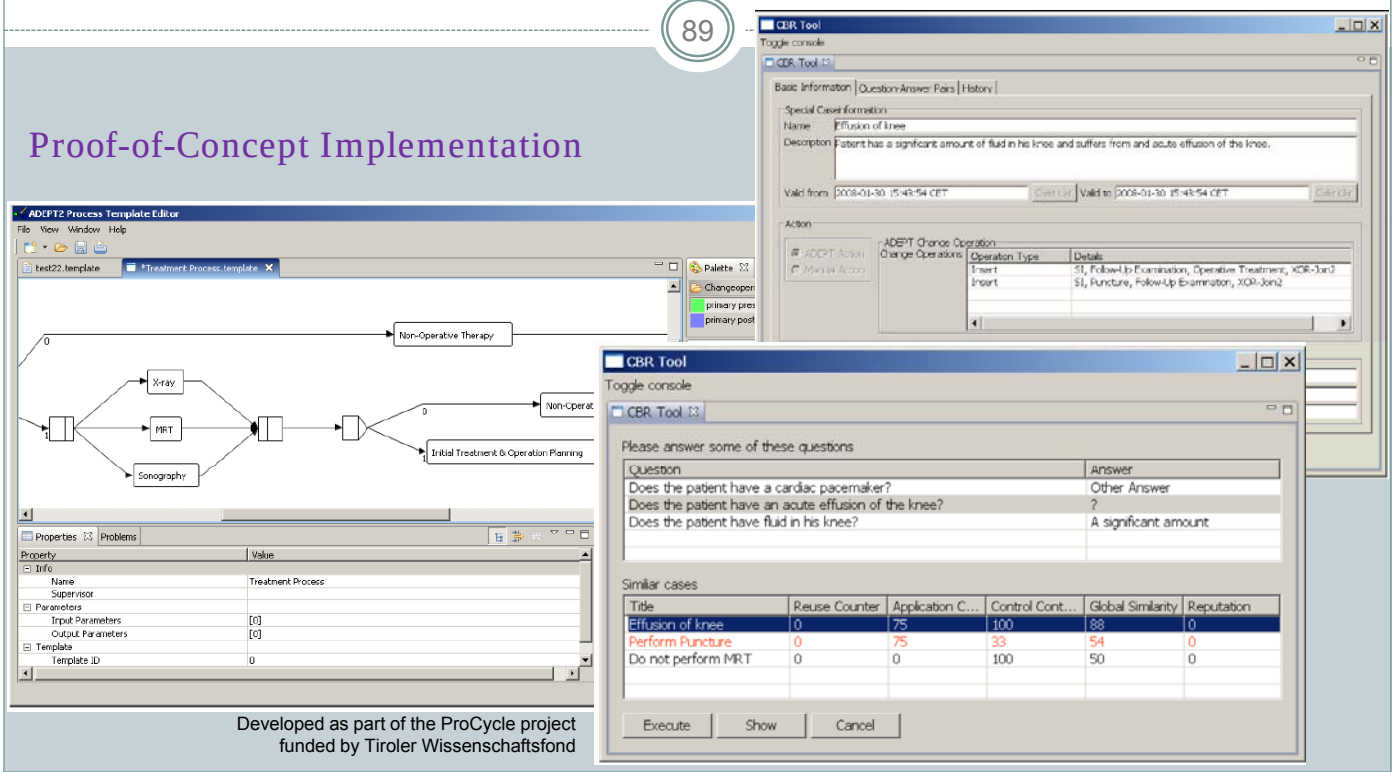


Proof-of-Concept Implementation

Developed as part of the ProCycle project funded by Tiroler Wissenschaftsfond

2.5.3.3 User Assistance & Change Reuse (9)

Proof-of-Concept Implementation



Developed as part of the ProCycle project
funded by Tiroler Wissenschaftsfond

2.5.3.4 Controlling Access to Process Change Functions

- User dependent access rights
  - E.g., Physicians are authorized to insert X-ray activities
- Process dependent access rights
  - E.g., Activity vacation request must not be inserted in medical treatment processes

[WRWR05]

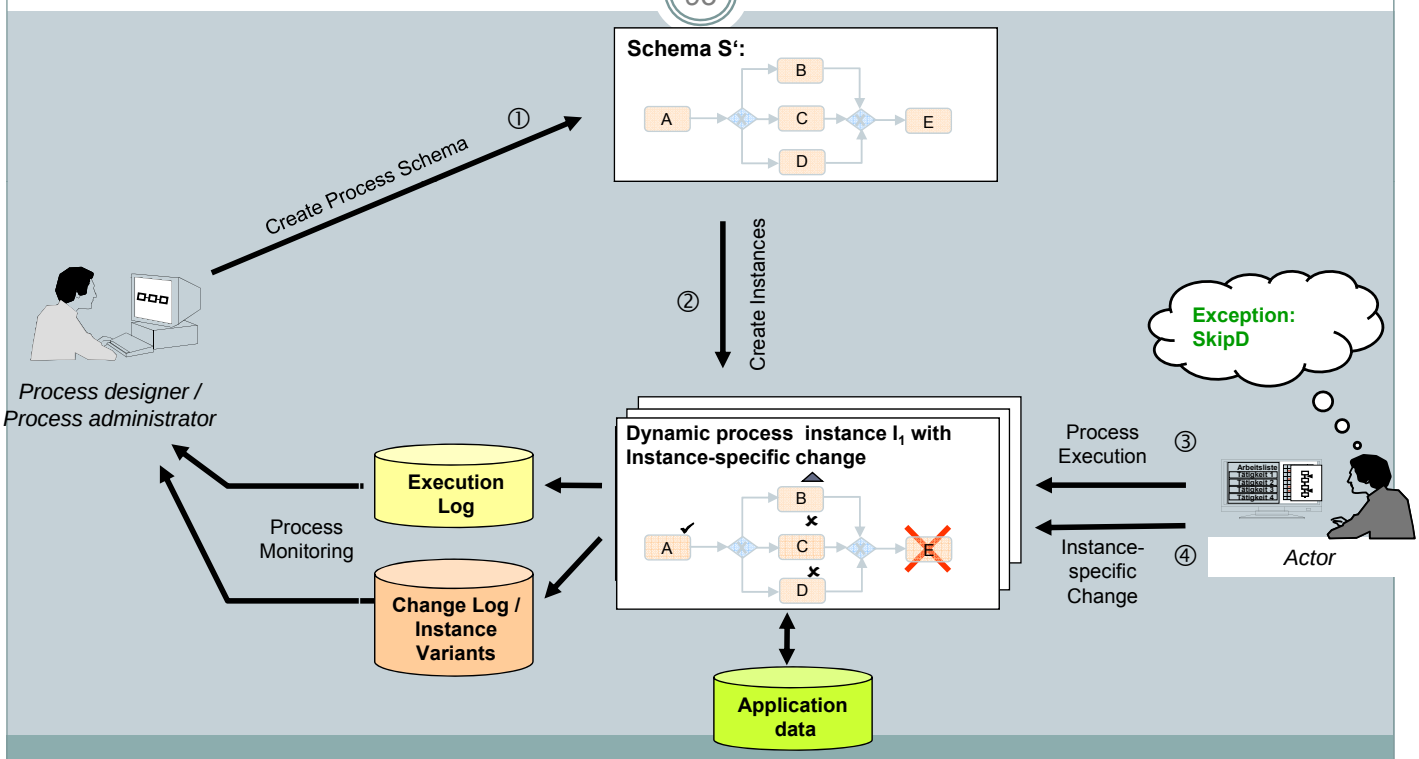
2.5.3.5 Change Concurrency Control

- Different ad-hoc changes by different users at same time
  - Prohibit concurrent changes
  - Optimistic concurrent change techniques
  - Pessimistic locking
- Concurrency of process type and process instance changes

[RJR07, RRD04c]

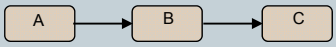
Chapter 2: The Imperative Approach ...

2.6 Monitoring and Analyzing Flexible and Dynamic Processes



2.6.1 Execution and Change Logs

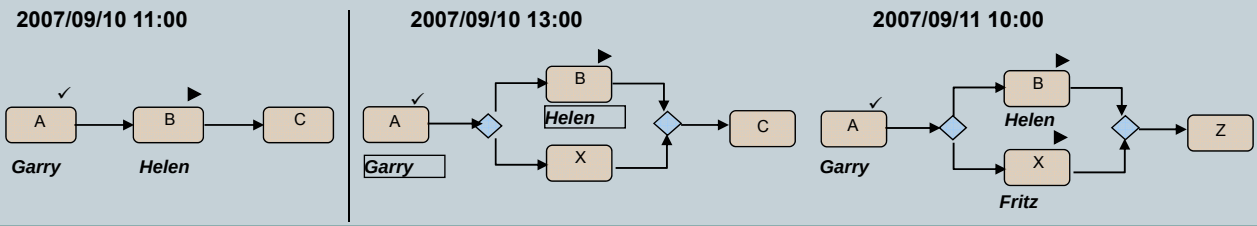
Original Schema S



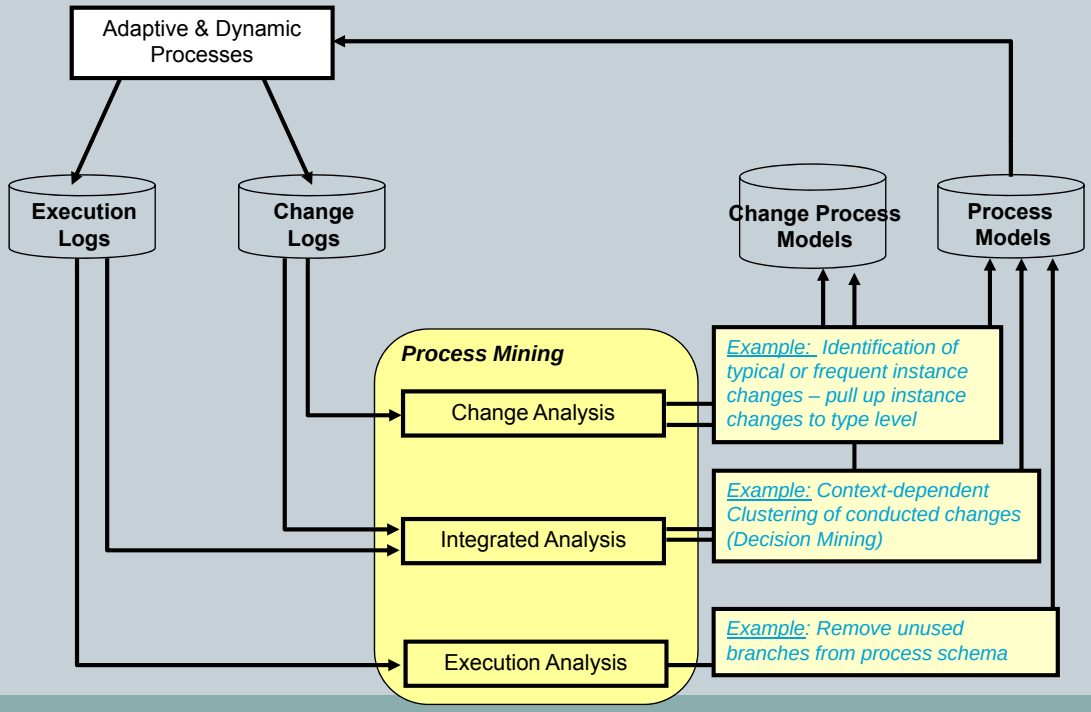

```
Instance 4711
Activity  Event            User    Timestamp
          Instance Started Garry   2007/09/08 15:00
A         Started          Garry   2007/09/08 15:30
A         Completed        Garry   2007/09/08 15:45
B         Started          Helen   2007/09/10 11:00
X         Started          Fritz   2007/09/11 09:01
```

```
Change Log Instance 4711 on Schema S

Change TX Applied Changes : User:Timestamp

001 InsertFragment[S;X,A,C]:Helen:2007/09/10 12:02
002 ReplaceFragment(S;C,Z):Jim:2007/09/11 09:31
```

Process Instance 4711



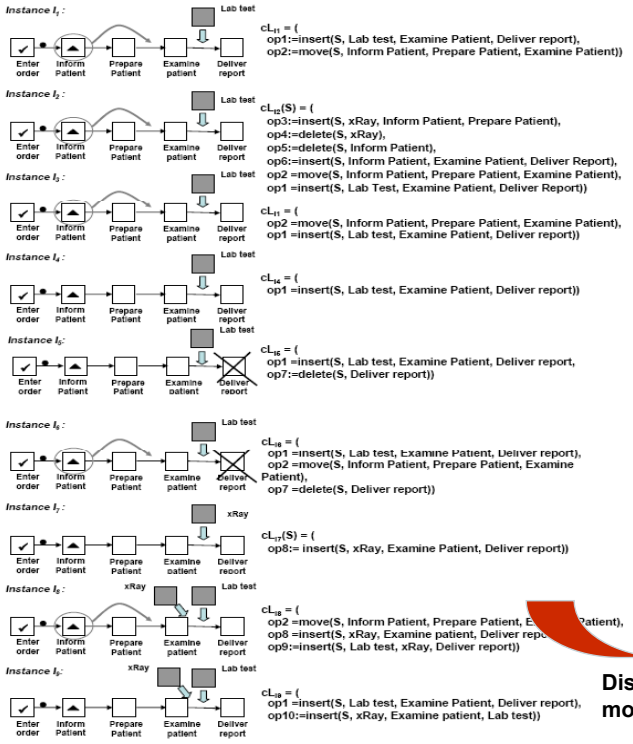
2.6.2 Execution & Change Analysis



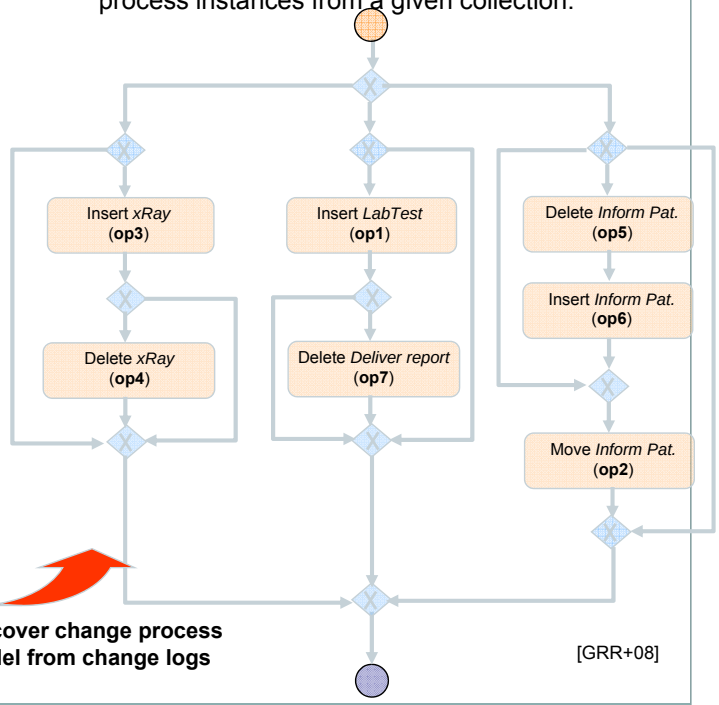
[AaWe94, ADH+03, ARW+07, AWM04, GRR+06, GRR+08]

2.6.3 Simple Change Analysis

Dynamic processes with instance-specific changes:



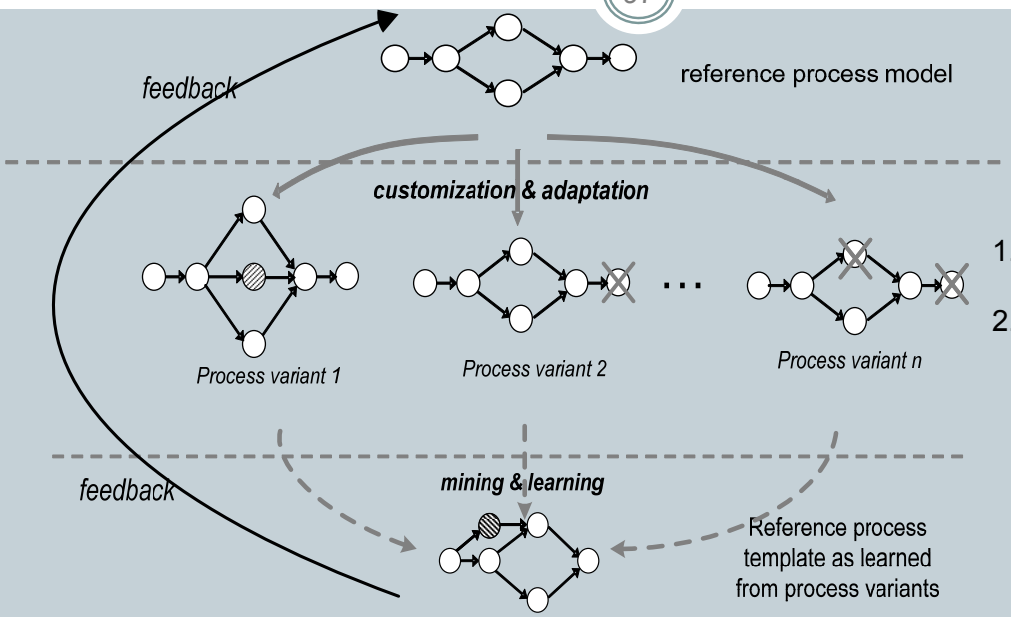
Example: Discover a *meta change process* which covers all changes applied to dynamic process instances from a given collection.



Discover change process model from change logs

[GRR+08]

2.6.4 MinAdept Project: Variant Mining



1. Configurations are very costly
2. Variants are difficult to maintain

Derive a new reference process model from the the variants such that:
Less adaptations are needed in future!

[LRW08b]

2.6.4.1 MinAdept: Basic Goal

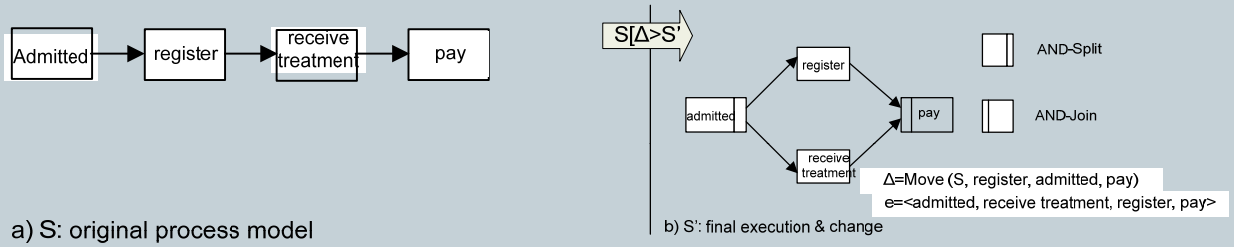
How to **discover a reference process model**

by mining a collection of **process (instance) variants**

in order to

reduce the need of future process configurations?

2.6.4.2 MinAdept: Bias and Distance



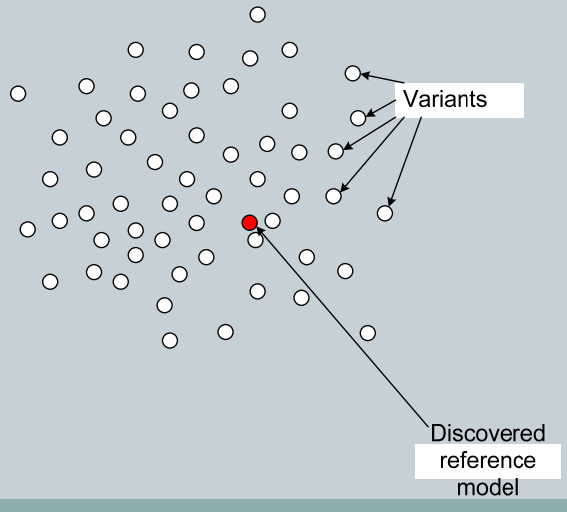
- Process Bias: minimal set of change operations needed to transform a given process model S into a model S' -- *Bias(S,S') = {move (S, register, admitted, pay)}*
- Process Distance *Distance(S,S')* :
  - # change operations of the bias between S and S' -- *Distance(S,S') = 1*
  - can measure the complexity for process change
  - computation method available for ADEPT

[LRW08c]

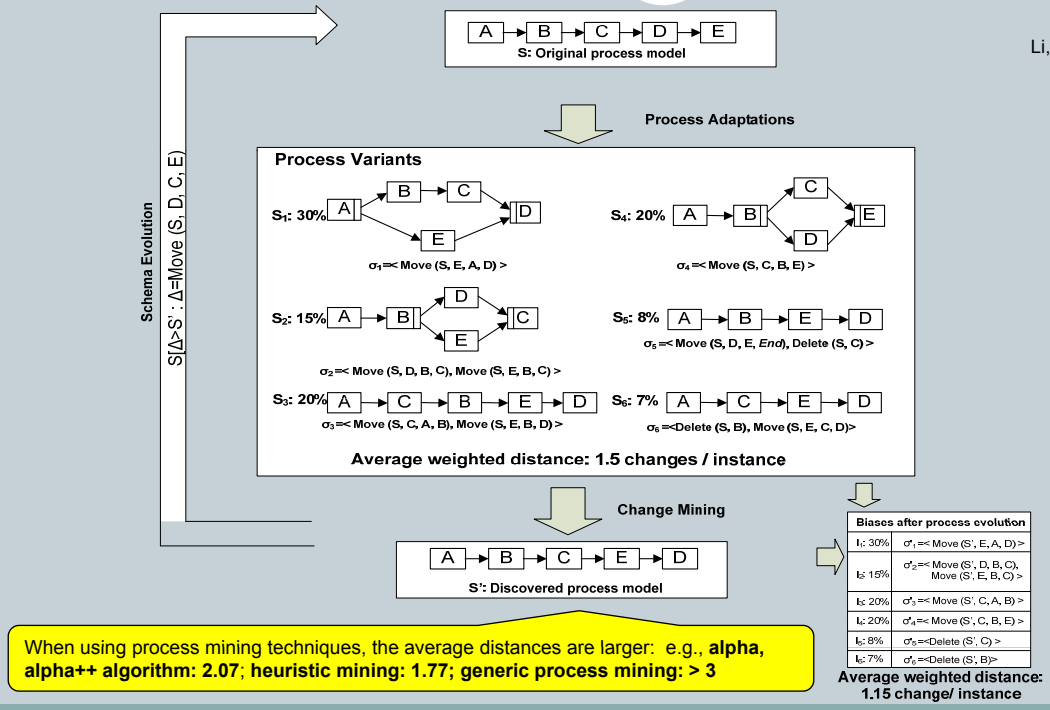
2.6.4.3 MinAdept: Reformulating Basic Goal

How to **derive a reference process model** by mining the **a collection of process (instance) variants** which has **minimal average distance to the process variants?**

When representing a process (instance) variant as a "dot" in a 2-dimensional space, discovering a reference model logically corresponds to finding the "center" (for which the average distance is minimal to all "dots").



2.6.4.4 MinAdept: Clustering Approach (1)



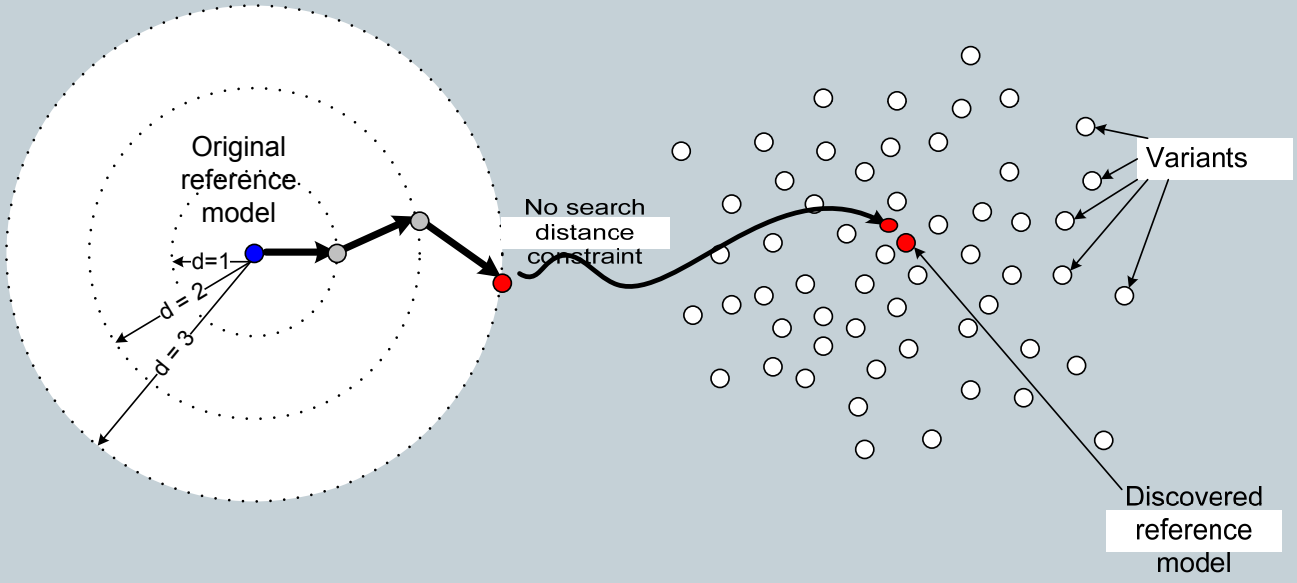
Li, Reichert & Wombacher [LRW08a]

When using process mining techniques, the average distances are larger: e.g., **alpha, alpha++ algorithm: 2.07**; **heuristic mining: 1.77; generic process mining: > 3**

2.6.4.4 MinAdept: Clustering Approach (2)

- Clustering approach does not consider the old reference model when discovering the new one:
  - We cannot control the mining result, i.e., the new reference model might be quite different from the old one.
    - Migration from the old to the new reference model becomes difficult!
    - Results can be Spagetti-like!
  - We would not know which change is more important than others.
- Idea: old reference model may act as a "counterforce" to control the result of our discovered model.

2.6.4.5 MinAdept: Heuristic Approach (1)



Li, Reichert & Wombacher [LRW09]

2.6.4.5 MinAdept: Heuristic Approach (2)

Basic Idea:

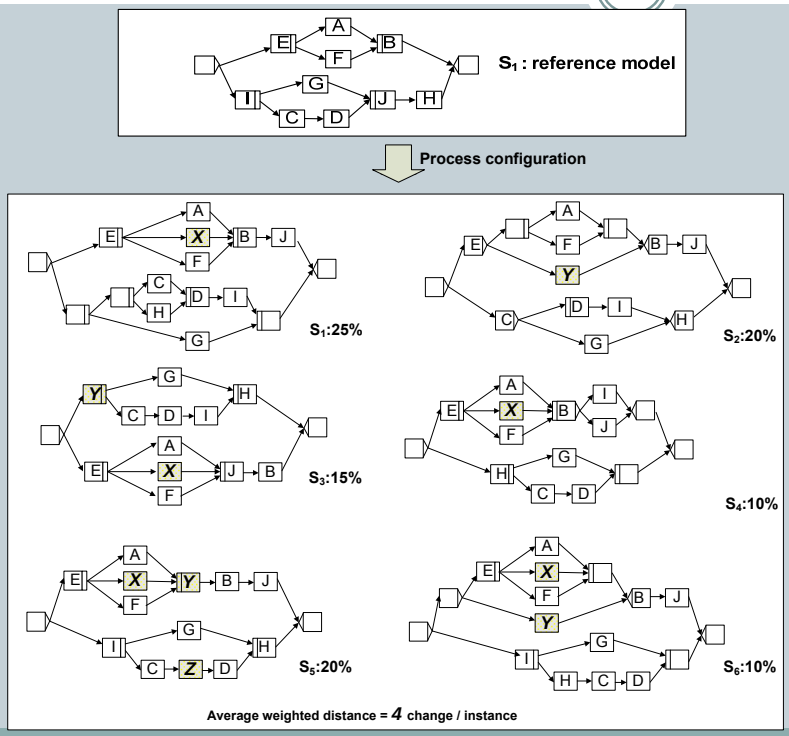
1. Use current reference model as starting point
2. For all models with distance 1 to the current reference model, search for the model fitting best to the given collection of variants!
3. If the process model found in Step 2 is "better" than the current reference model, use it as reference model in the following
4. Repeat Steps 2 and 3 until either no better model can be found or the allowed search distance is reached!

Needed:

- Fitness function (How to evaluate the result?)
- Search method (How to generate the next search step?)

For details see [LRW09]

2.6.4.5 MinAdept: Heuristic Approach (3)



Possible starting point:

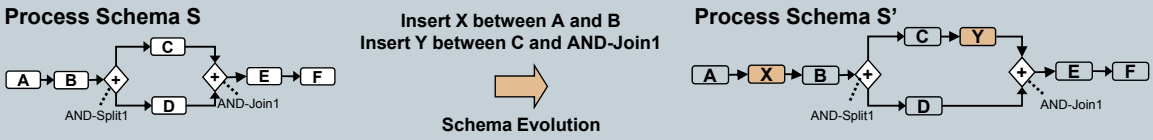
Can we find a model closer to the variants by performing at maximun two changes of the old reference model?

[LRW09]

Chapter 2: The Imperative Approach ...

2.7.1 Scenarios for Process Schema Changes (1)

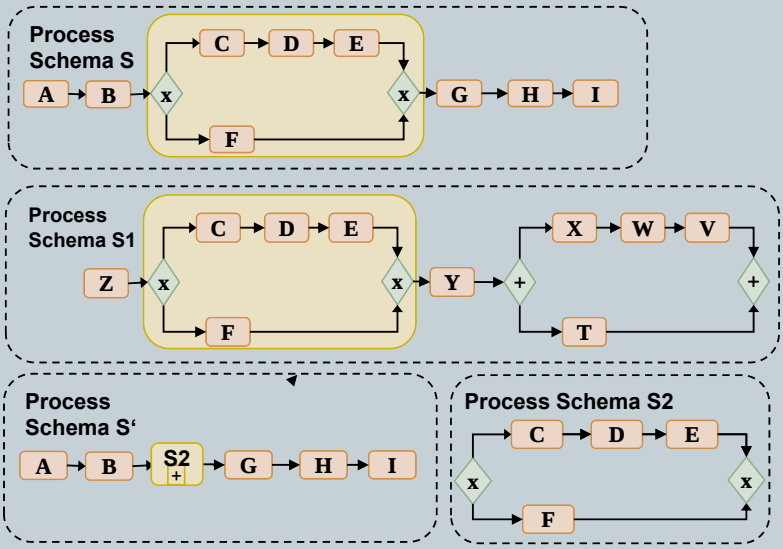
- Process schema has to be altered as response to changes in the environment
- Examples: regulatory changes, process improvement, market evolution, changes in customer behavior



[ReDa08, RRD04a]

2.7.1 Scenarios for Process Schema Changes (2)

- Schema is adapted to improve its maintainability, but without altering its behavior ☞ Process Refactoring



Process Model Smell
"Redundant Process Fragment"

Refactoring:
Extract Process Fragment

Weber & Reichert 2008 [WeRe08]

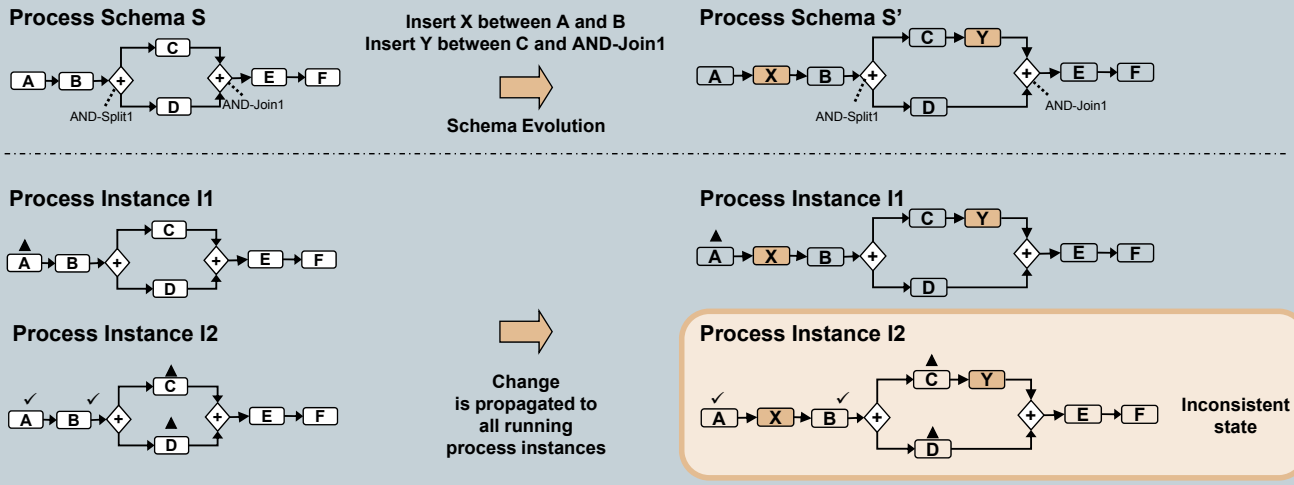
2.7.2 Change Support Features
Schema Evolution, Version Control and Instance Migration

- Schema Evolution
  - Changes at the process type level
- How to deal with running instances when adapting the original process schema?
  - Scenario 1: No version control
  - Scenario 2: Co-existence of instances of old / new schema
  - Scenario 3: Change propagation and instance migration

2.7.2.1 Scenario 1 - No Version Control

- Schema is overwritten and instances are migrated

Type change overwrites schema S

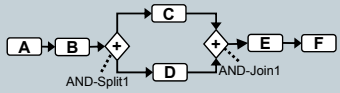


2.7.2.2 Scenario 2 - Version Control

- Co-existence of instances of different schema versions

Type change results into a new version of schema S

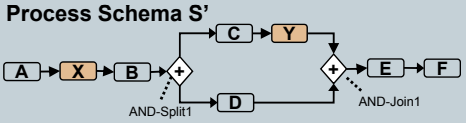
Process Schema S



Insert X between A and B
Insert Y between C and AND-Join1

Schema Evolution

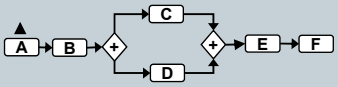
Process Schema S'



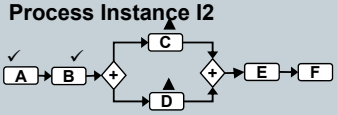
Old instances remain with schema S

Instances created from S (before schema evolution)

Process Instance I1

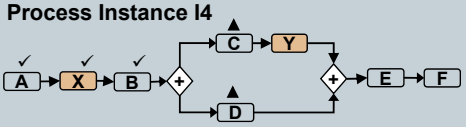


Process Instance I2

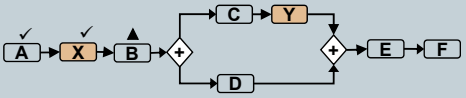


Instances created from S' (after schema evolution)

Process Instance I4



Process Instance I5

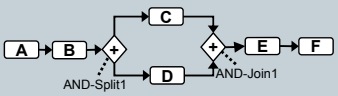


2.7.2.3 Scenario 3 – Instance Migration

- Compliant instances are migrated to the new schema

Type change results into a new version of schema S

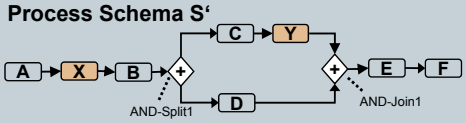
Process Schema S



Insert X between A and B
Insert Y between C and AND-Join1

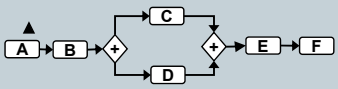
Schema Evolution

Process Schema S'

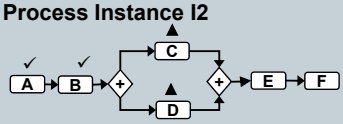


Migration of compliant process instances to S'

Process Instance I1

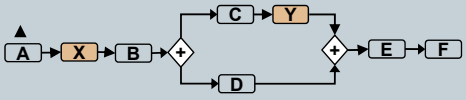


Process Instance I2



Propagation of compliant process instances to schema S' (incl. state adaptations)

Process Instance I1



Process Instance I_2 not compliant with S'

[RRD04a]

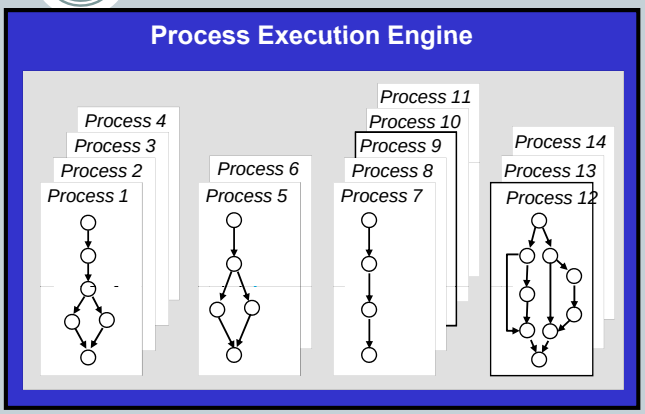
Chapter 2: The Imperative Approach ...

2.8.1 Ad-hoc Changes in ADEPT (1)

ADEPT :

Individually adaptable Process Instances

Process Instance = (individual) "Process Program"



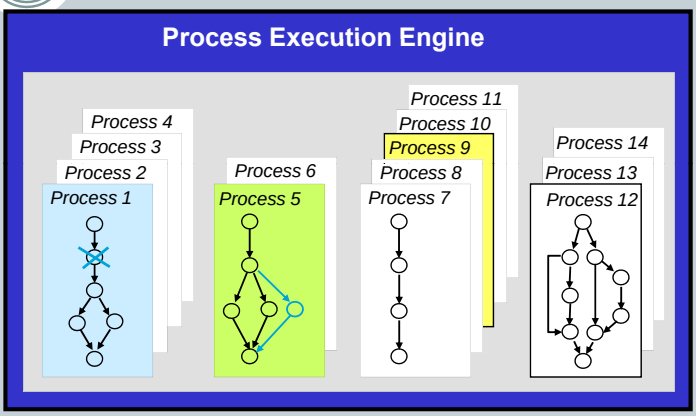
[ReDa09, DaRe09, DRR+08, RDR+09, Reic00]

2.8.1 Ad-hoc Changes in ADEPT (2)

ADEPT :

Individually adaptable Process Instances

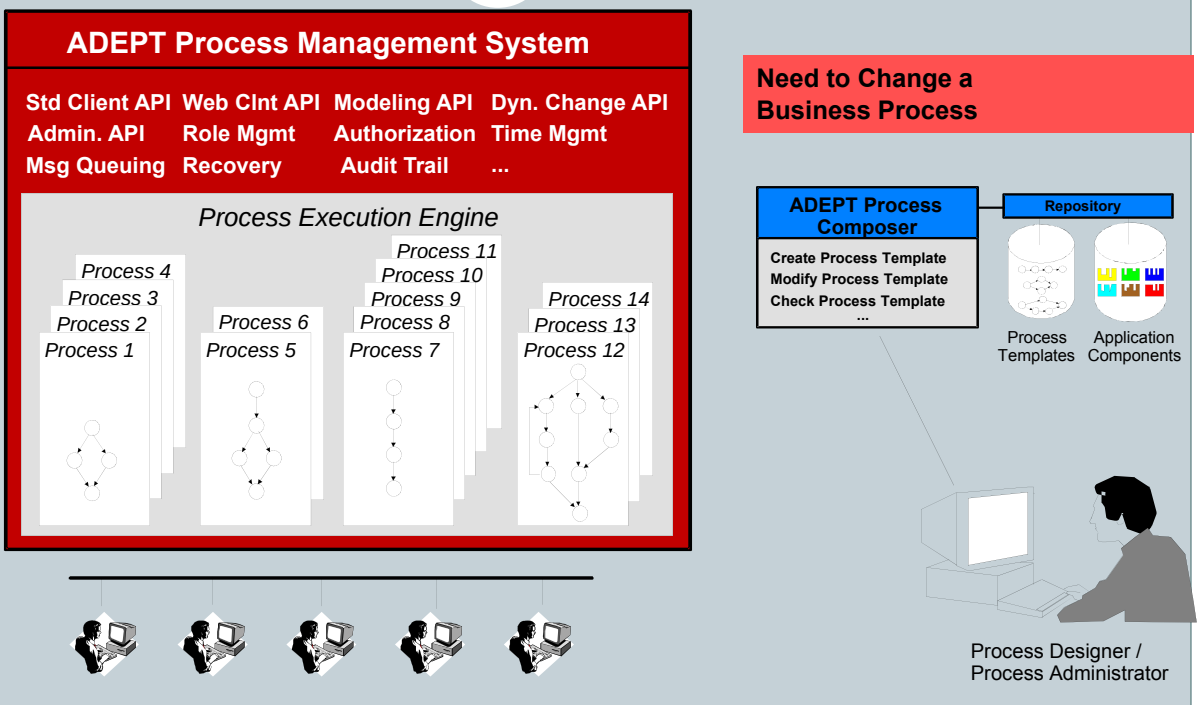
Process Instance = (individual) "Process Program"



Properties:

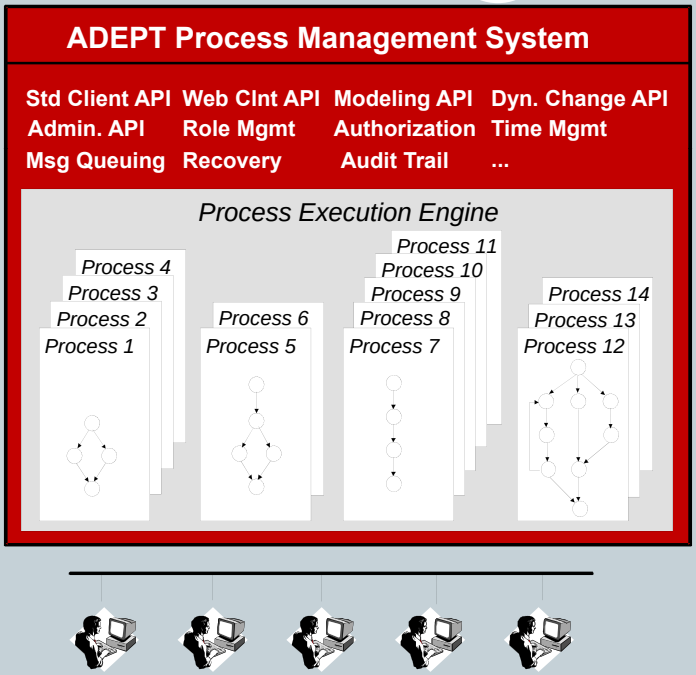
- **Formal Process Meta Model** (expressive + restricted enough)
- **Formal Criteria for Change Correctness** (incl. „Theorems & Proofs")
- **Efficient, build-in consistency checks** („no bad surprise")
- **Support of a high number of change patterns**
- **API for accomplishing ad-hoc changes**

2.8.2 Process Schema Evolution in ADEPT (1)

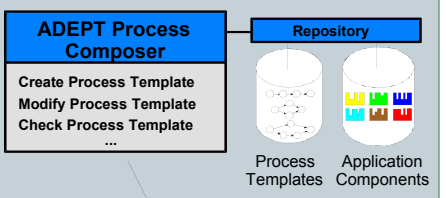


[ReDa09, DaRe09, RRD04a]

2.8.2 Process Schema Evolution in ADEPT (2)



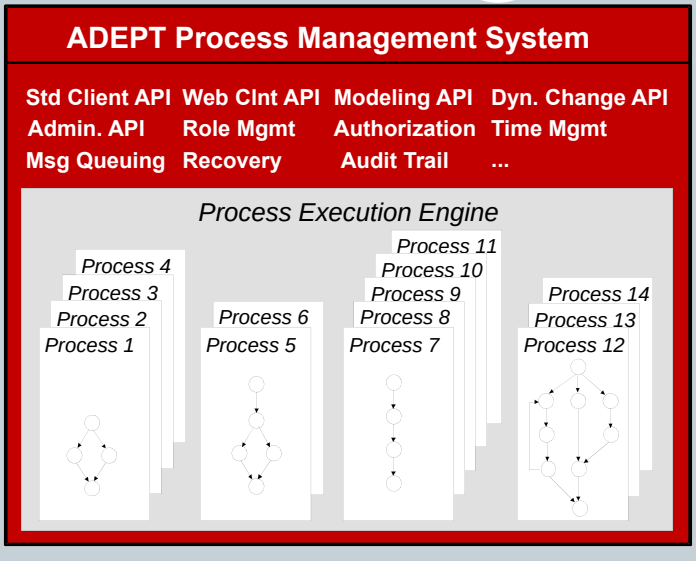
Process Designer Checks Out "Active" Process Template

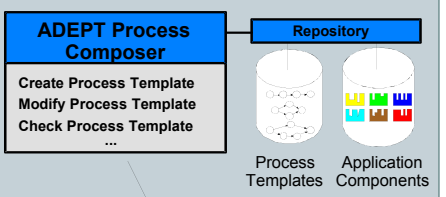


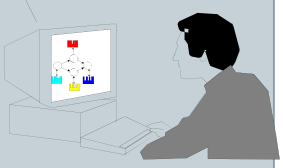
Process Designer / Process Administrator

Users

2.8.2 Process Schema Evolution in ADEPT (3)



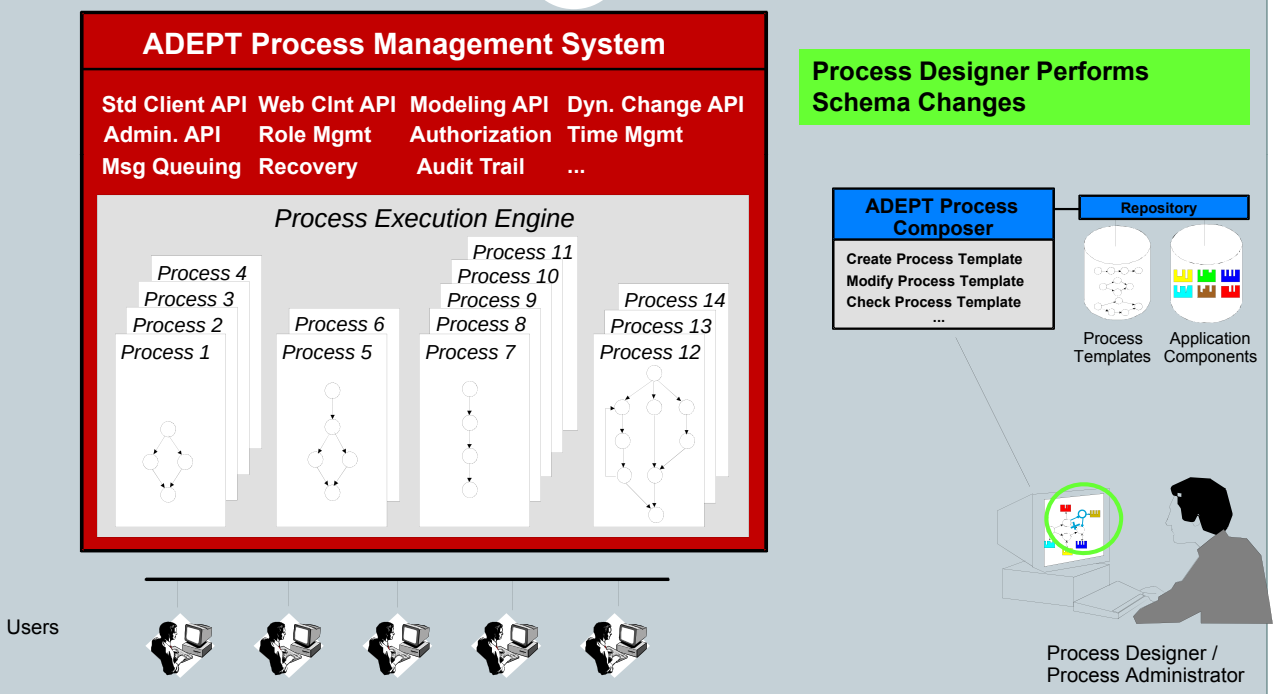




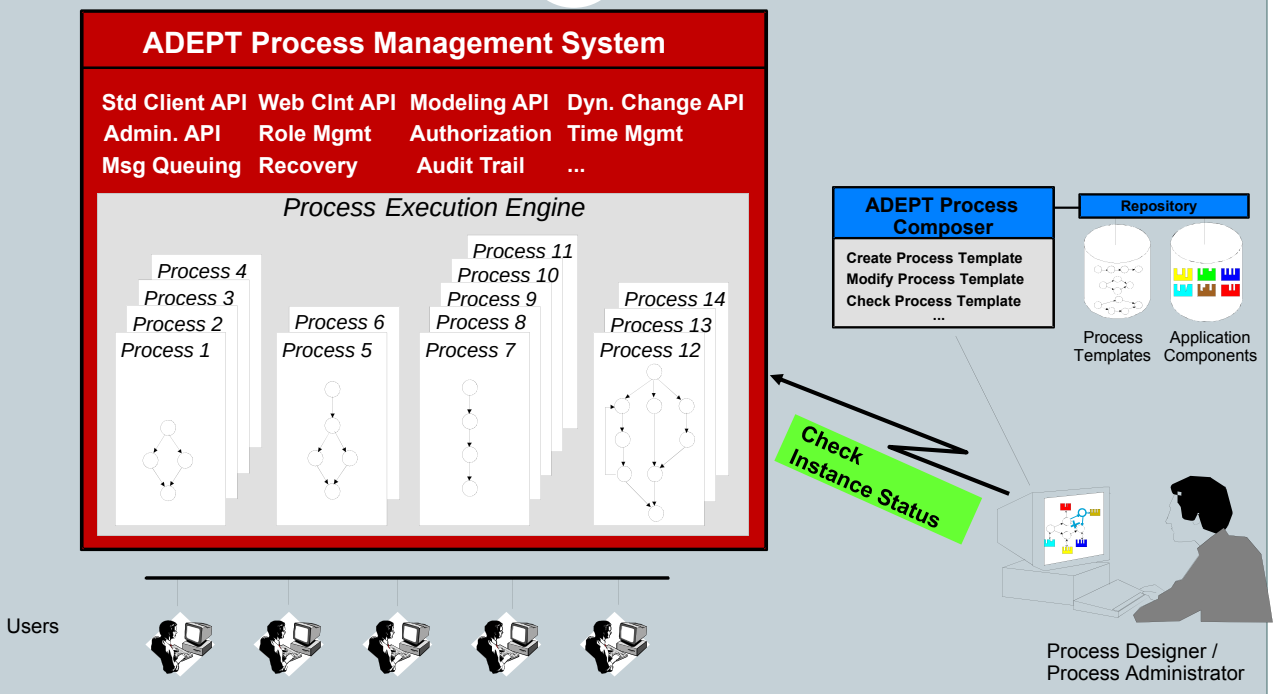
Process Designer / Process Administrator

Users

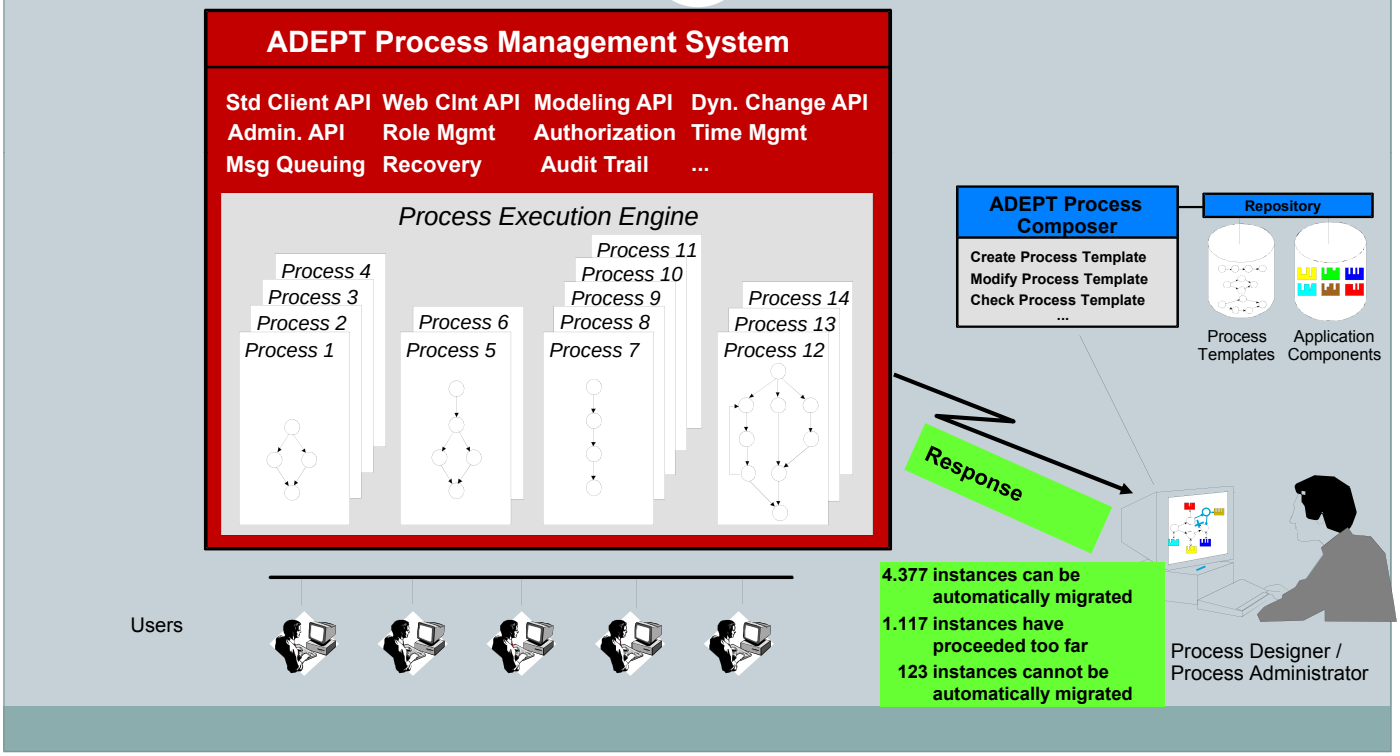
2.8.2 Process Schema Evolution in ADEPT (4)



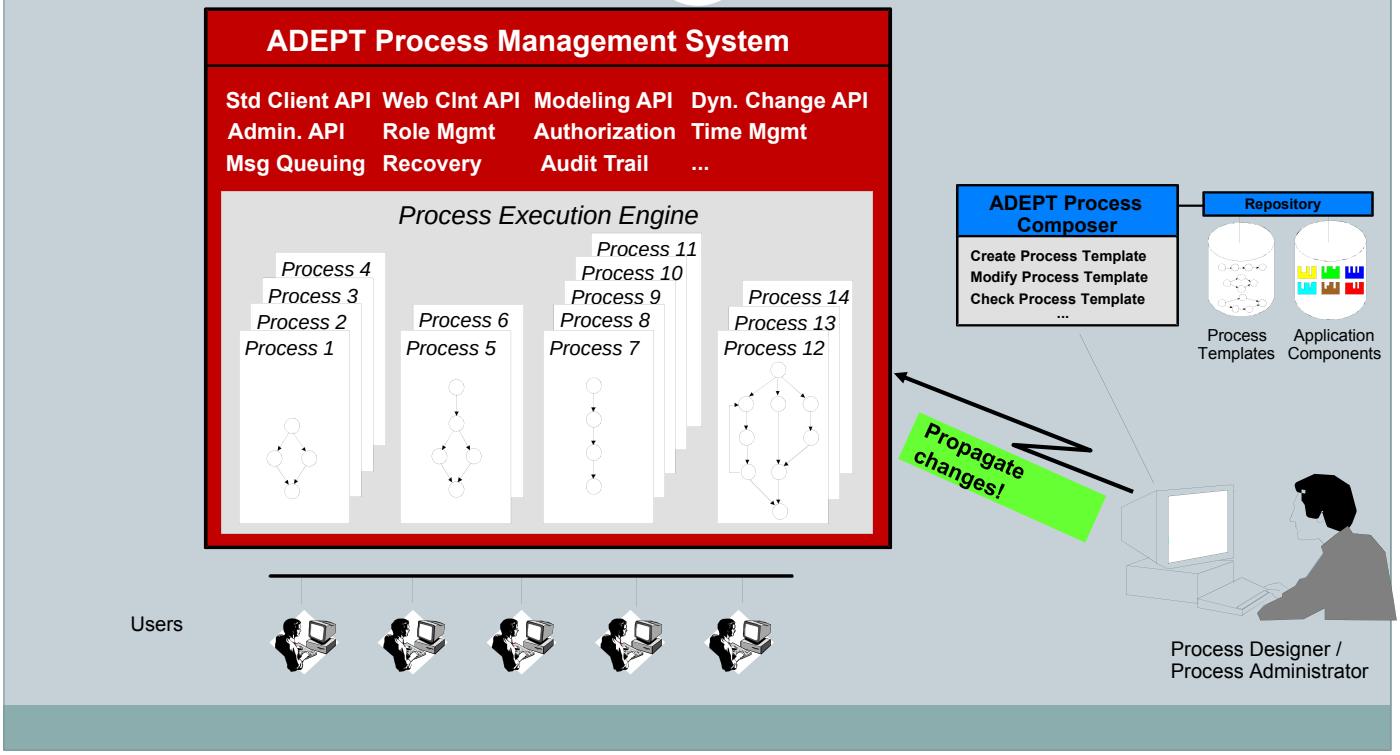
2.8.2 Process Schema Evolution in ADEPT (5)



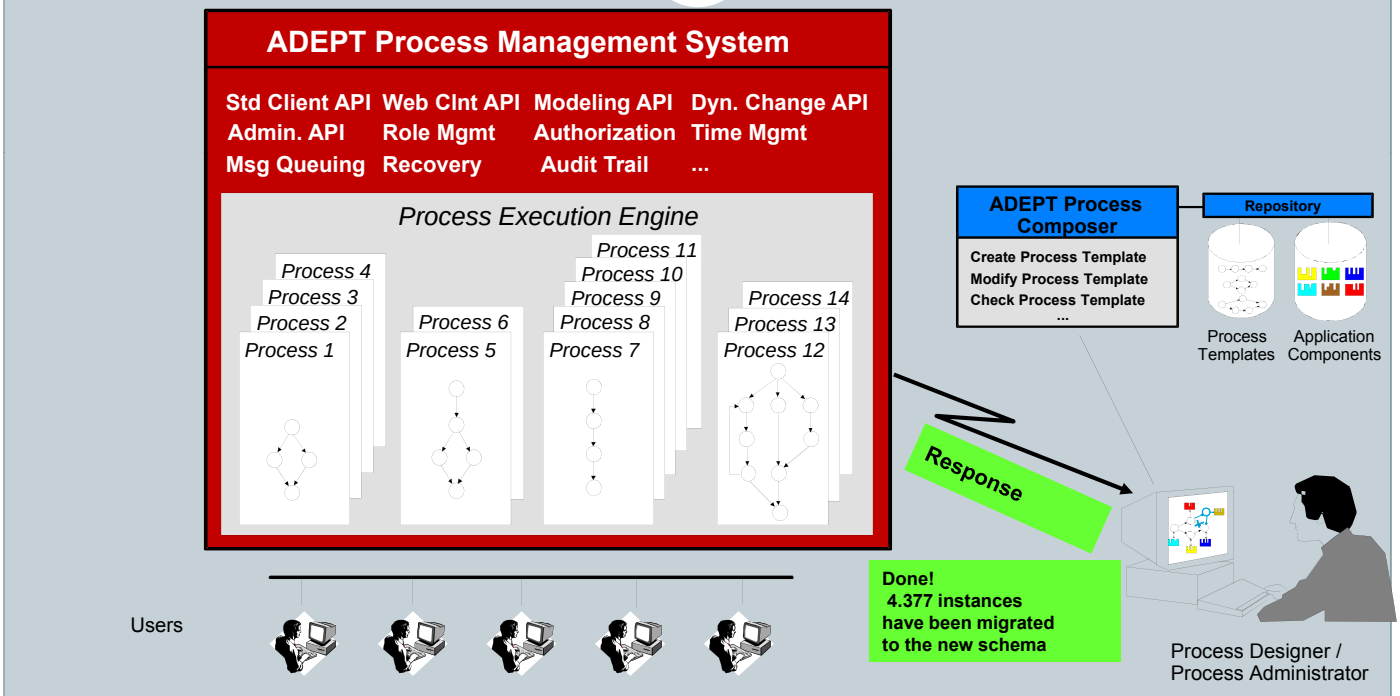
2.8.2 Process Schema Evolution in ADEPT (6)



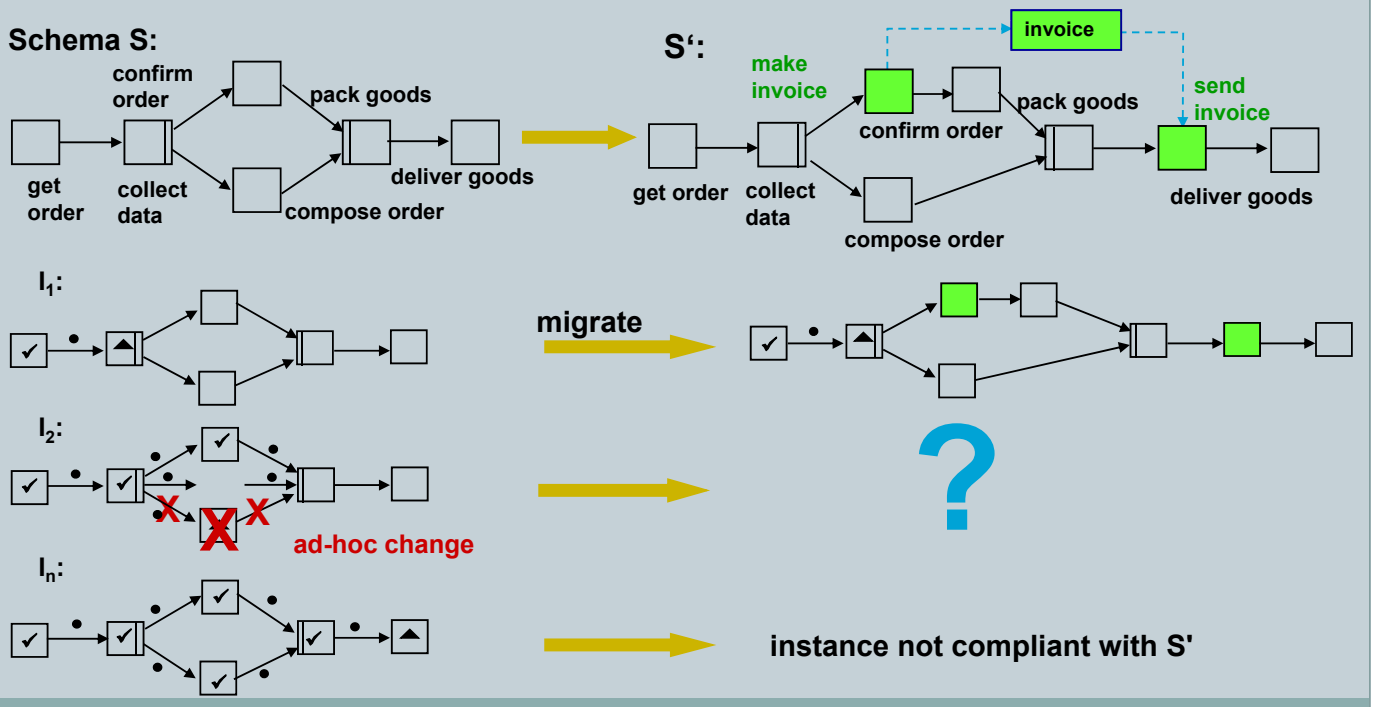
2.8.2 Process Schema Evolution in ADEPT (7)



2.8.2 Process Schema Evolution in ADEPT (8)



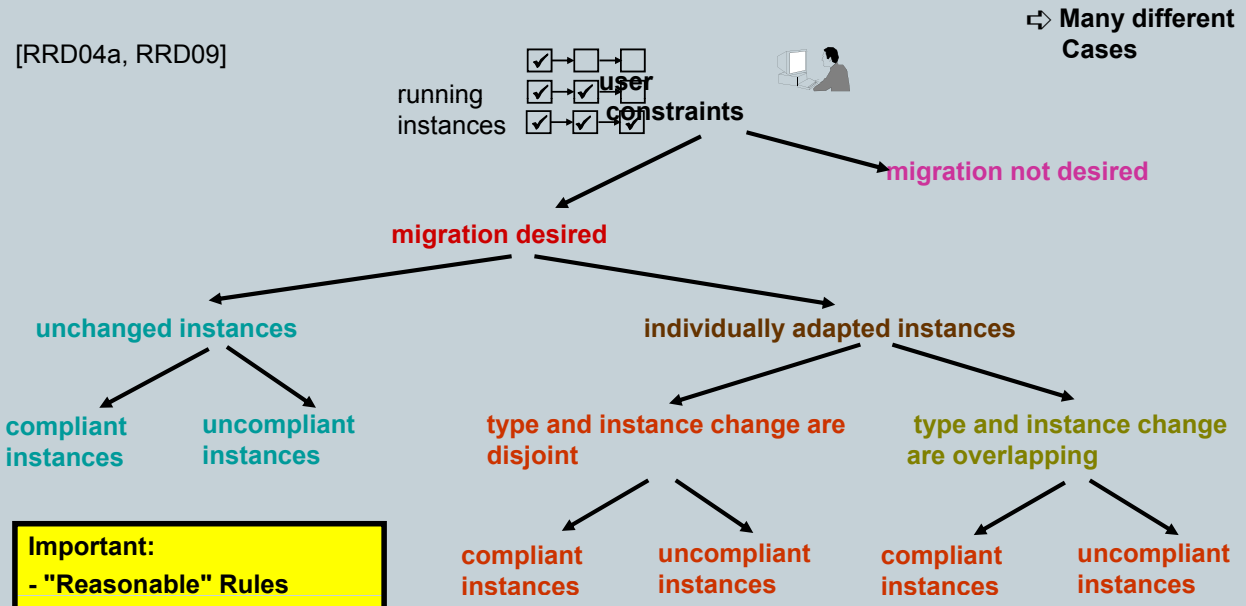
2.8.2 Process Schema Evolution in ADEPT (9)



2.8.2 Process Schema Evolution in ADEPT (10)

Efficient Migration of Compliant Flow Instances – Overall Picture ADEPT

[RRD04a, RRD09]

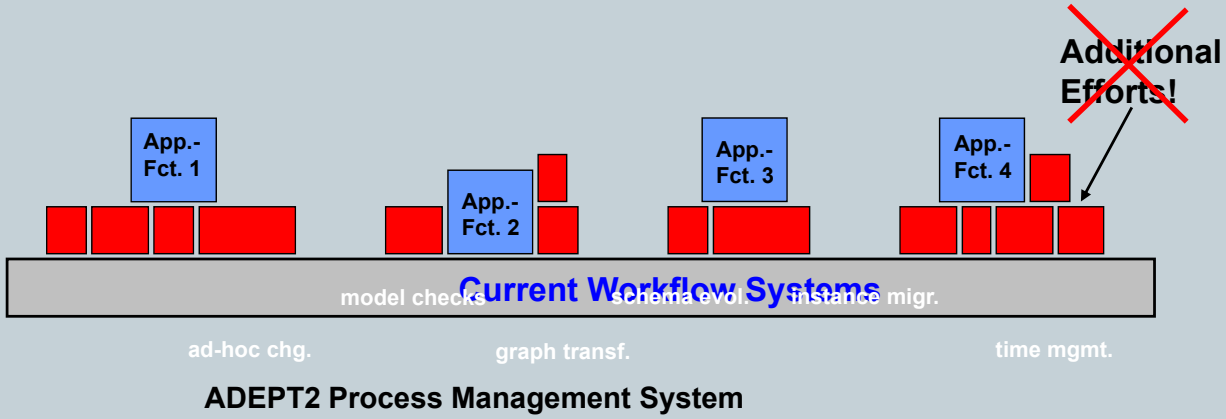


Important:
- "Reasonable" Rules
- Proper, formal Basis
- Efficient Algorithms

2.8.3 ADEPT Vision

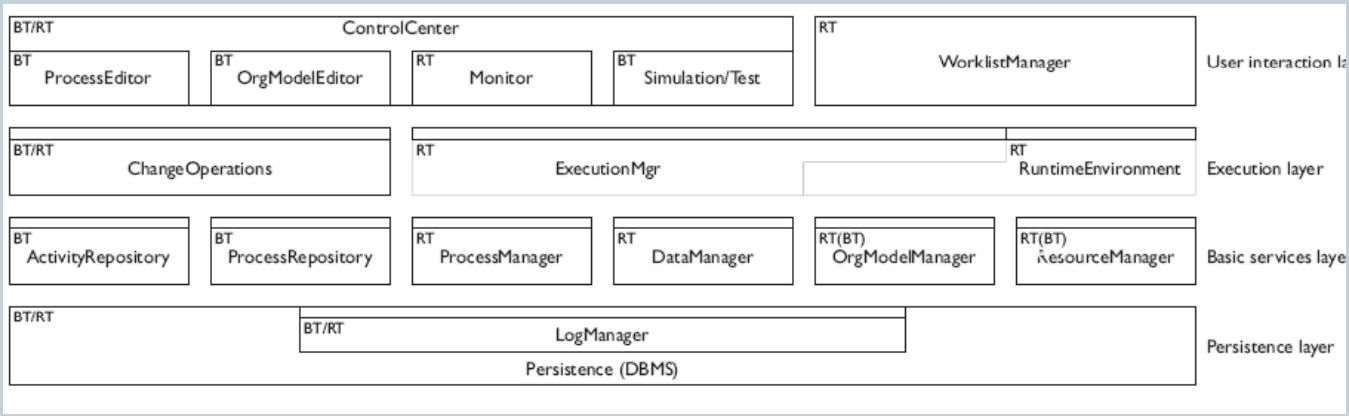
ADEPT Vision: Next Generation Process Management Technology

- Shifting complexity to the process management system
- Reducing application development costs significantly
- Allowing for dynamic process changes



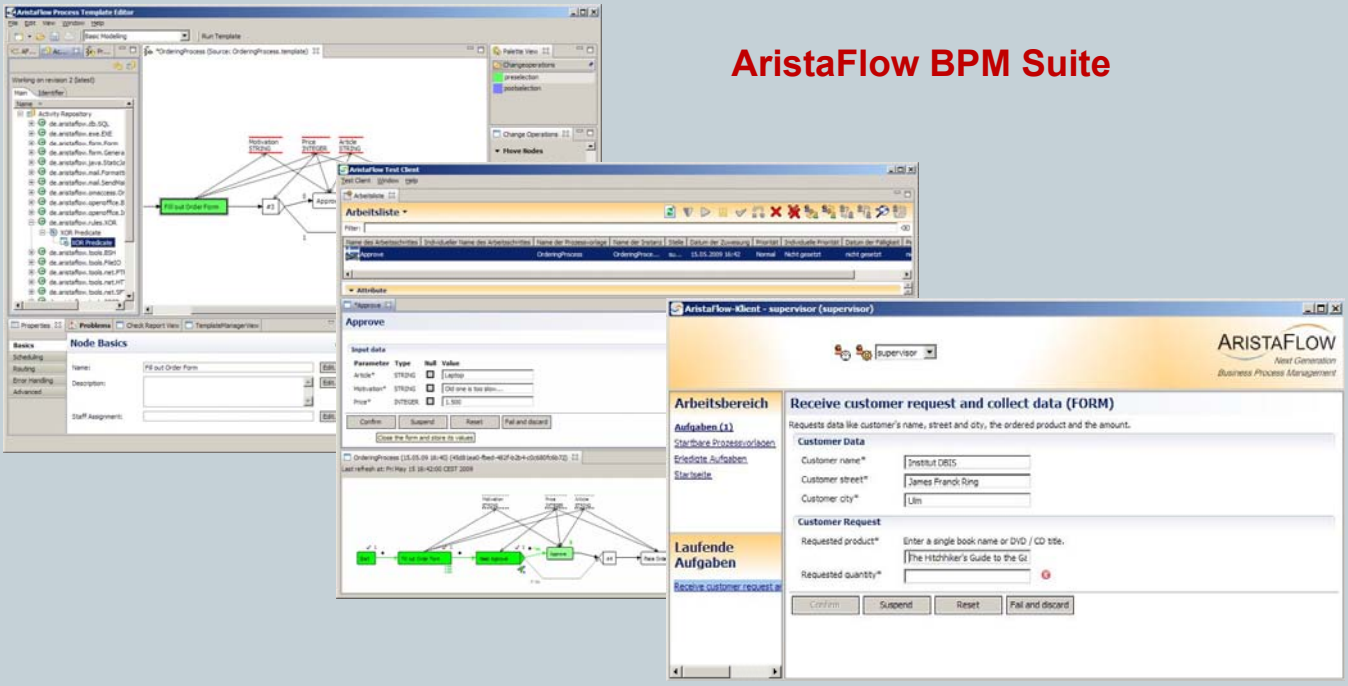
ADEPT2 Process Management System

2.8.4 ADEPT Architecture (Simplified)



[RDR+09, RRJ+06, ReBa07]

2.8.5 ADEPT: Implementation & Spin-Off

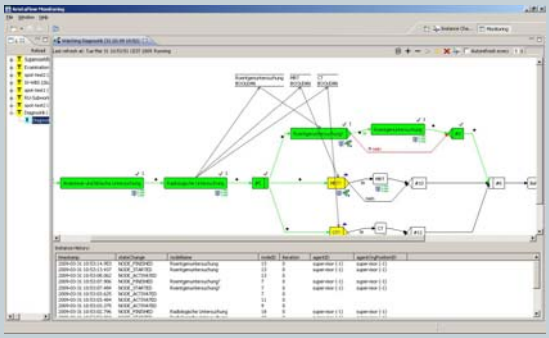


AristaFlow BPM Suite

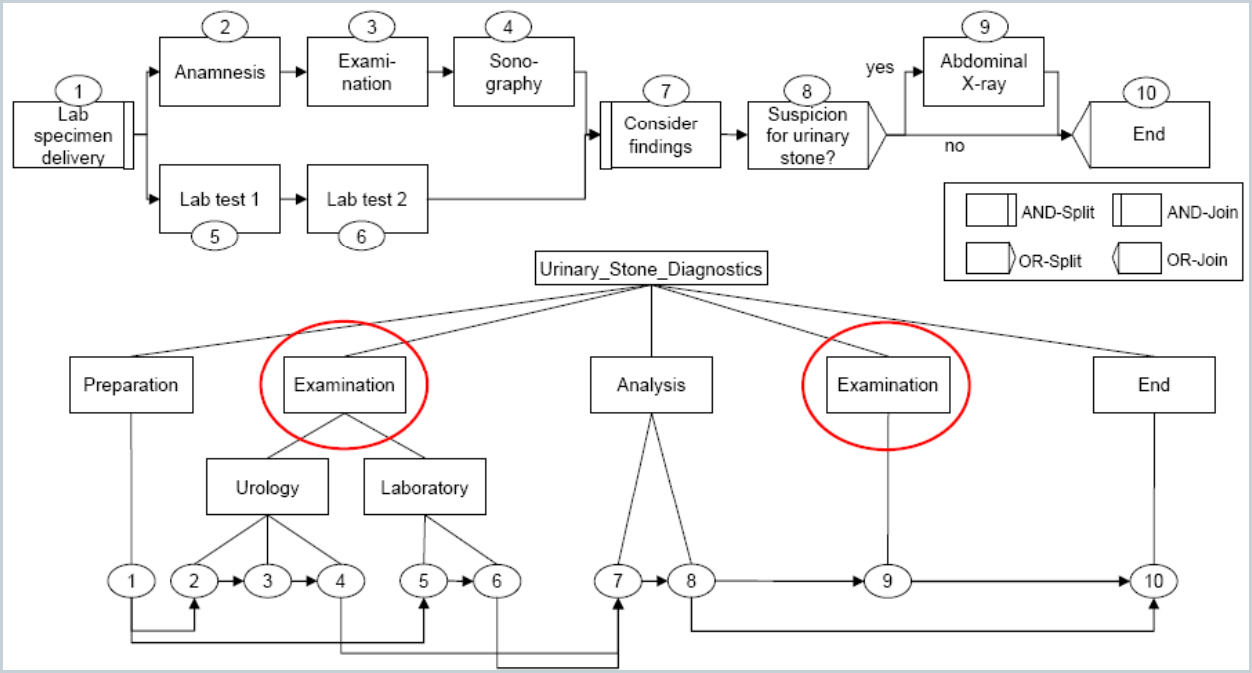
2.8.6 Partner Projects: Spot (Fraunhofer Dortmund)



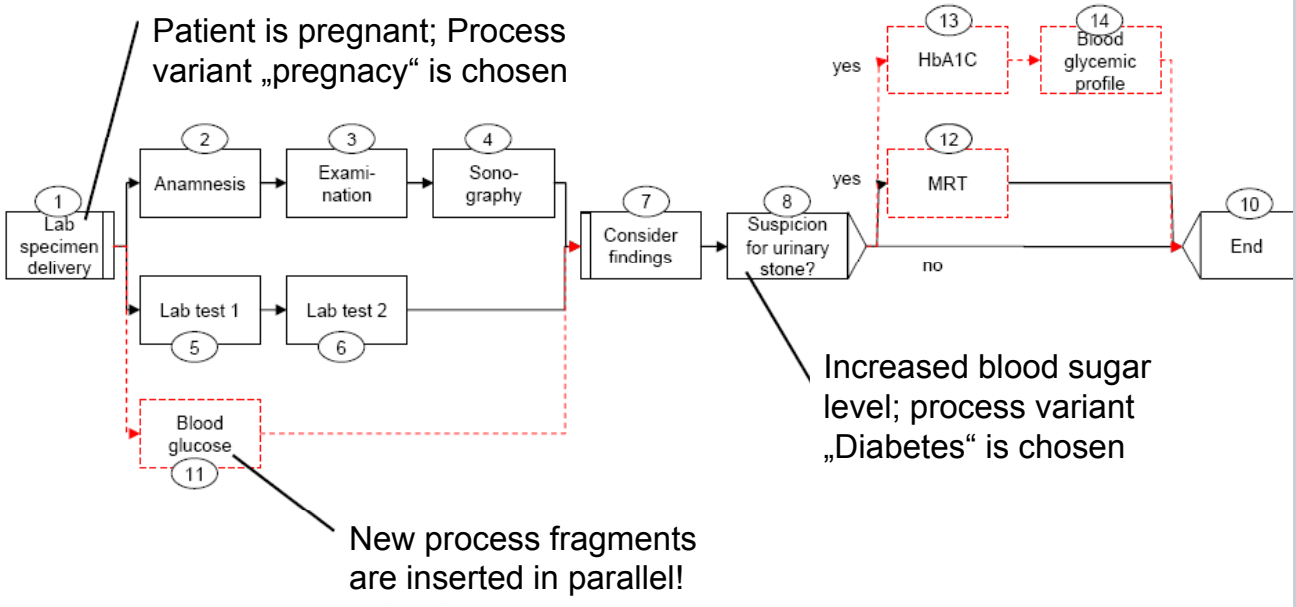
Flexible Support of Clinical Pathways!




2.8.6 Partner Projects: Spot (Fraunhofer Dortmund)

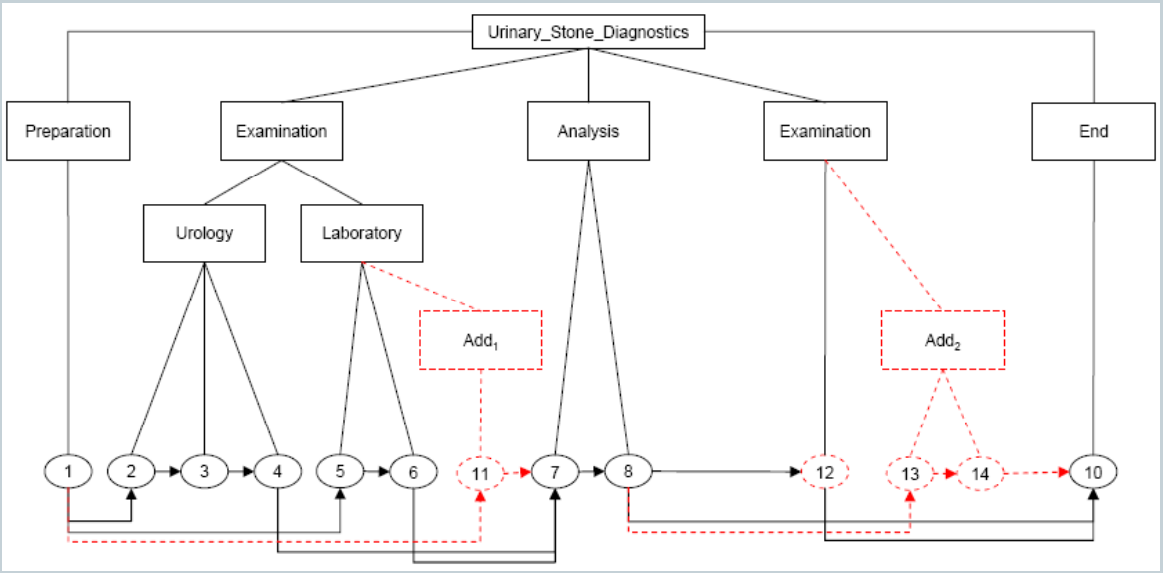


2.8.6 Partner Projects: Spot (Fraunhofer Dortmund)



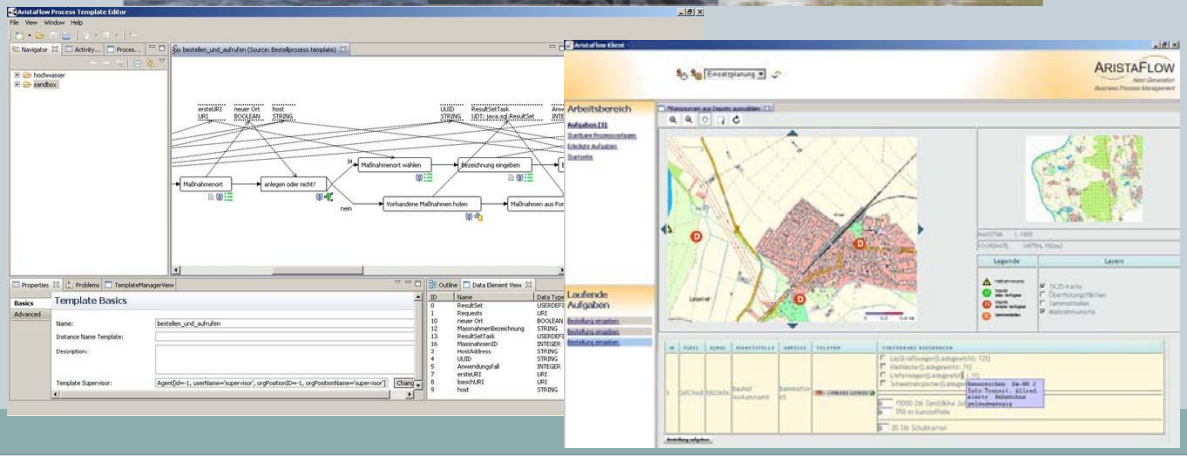
2.8.6 Partner Projects: Spot (Fraunhofer Dortmund)

… and the corresponding process tree



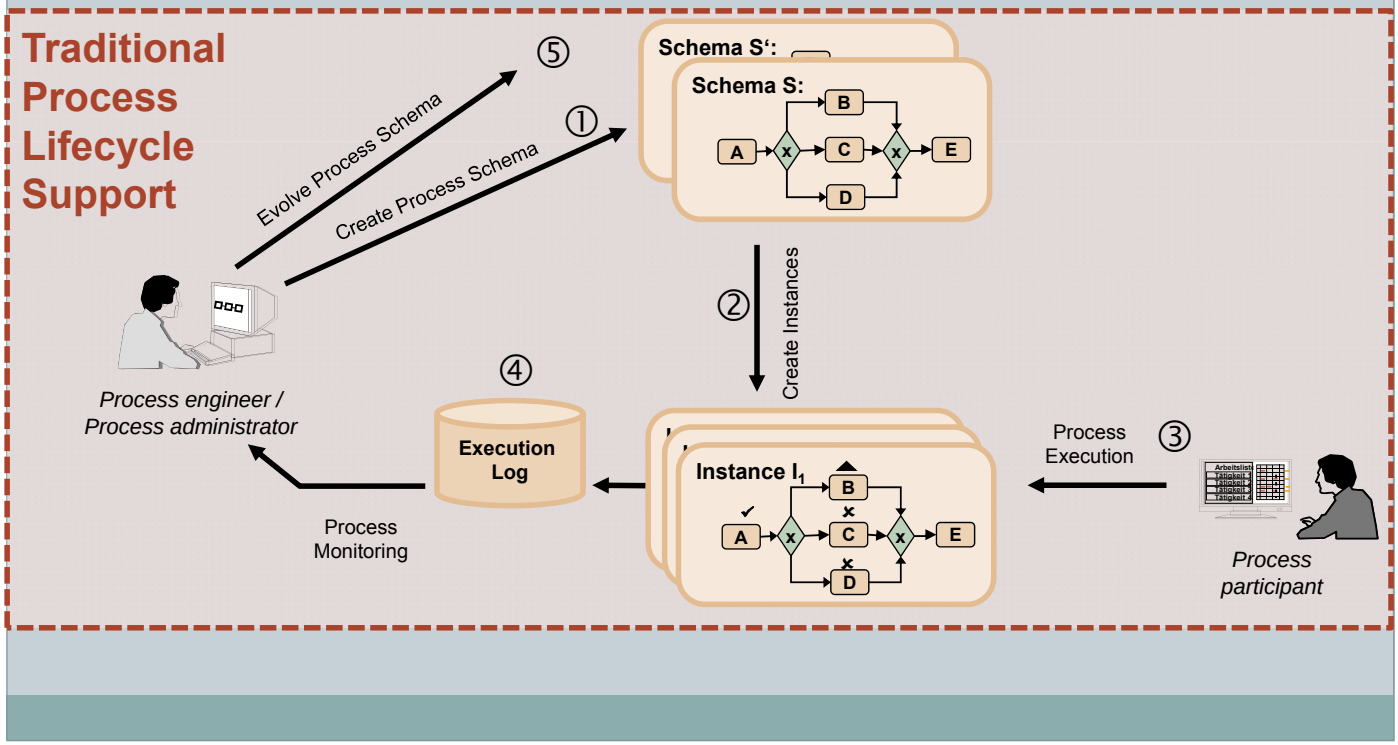
2.8.7 Partner Projects: TU Darmstadt

Process-aware, Cooperative Emergency Management for Water Infrastructures

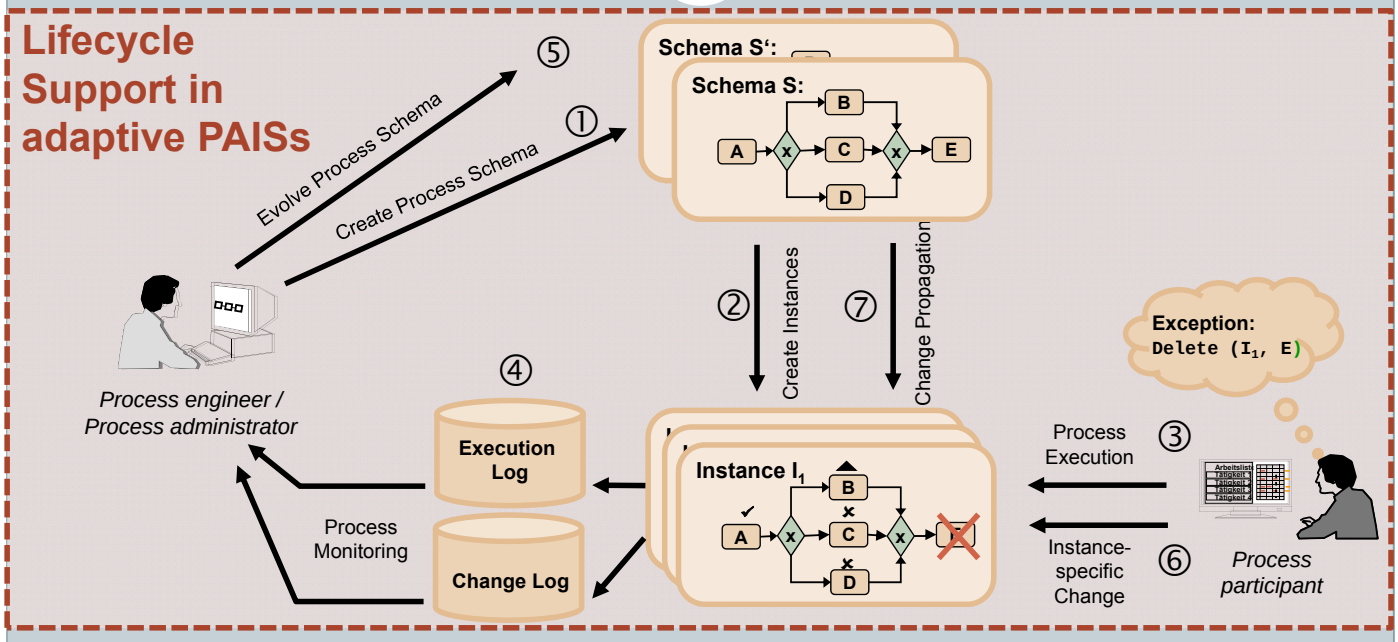


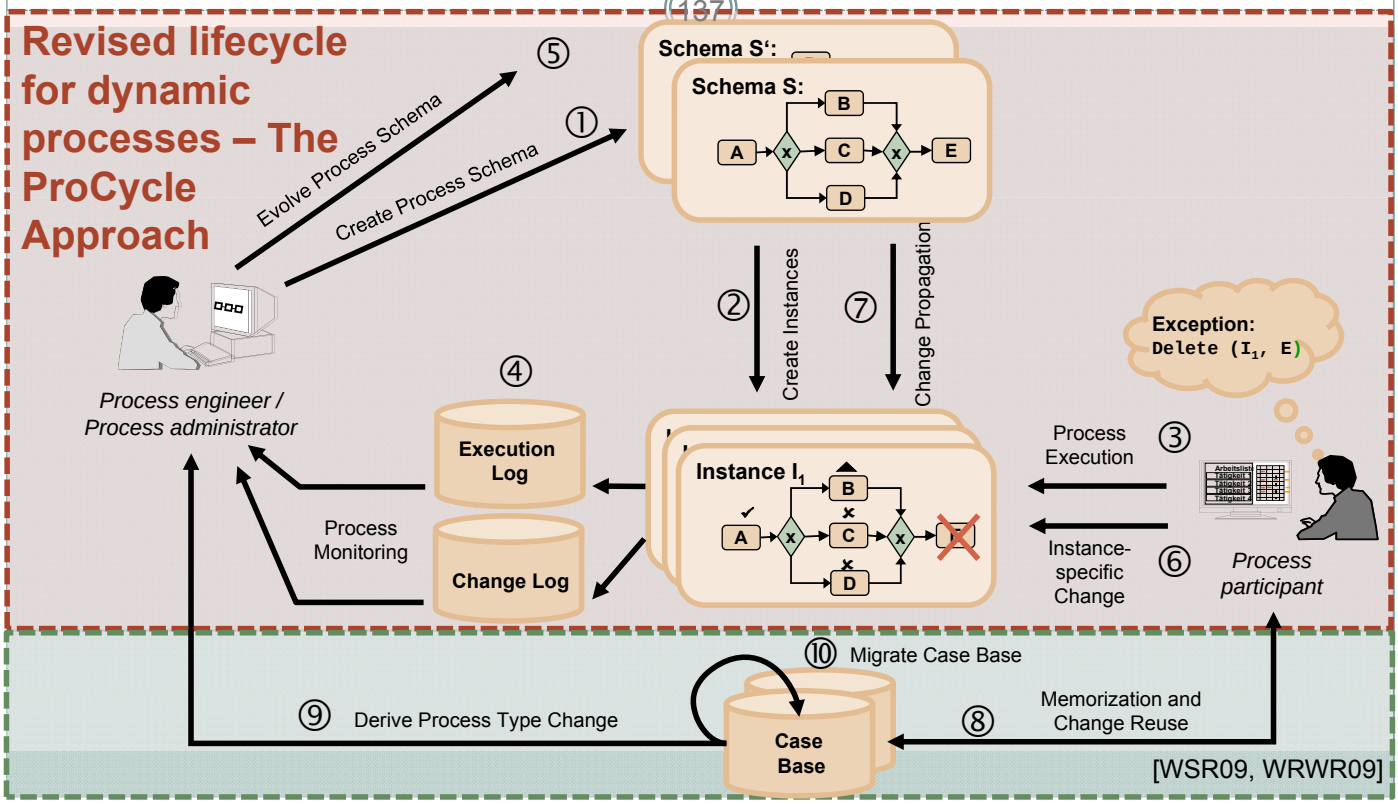
Chapter 2: The Imperative Approach ...

2.9 Integrated Lifecycle Support for Adaptive and Dynamic Processes (1)



2.9 Integrated Lifecycle Support for Adaptive and Dynamic Processes (2)





Agenda

1. Introduction
2. Flexibility Issues of the Imperative Approach to Business Process Management
3. Pattern-based evaluation of PAIS in respect to their ability to provide process flexibility
4. Flexibility Issues of the Declarative Approach to Business Process Management
5. Flexibility Issues in Data-driven Processes
6. Summary and Outlook

3.1. Comparison of Change Frameworks

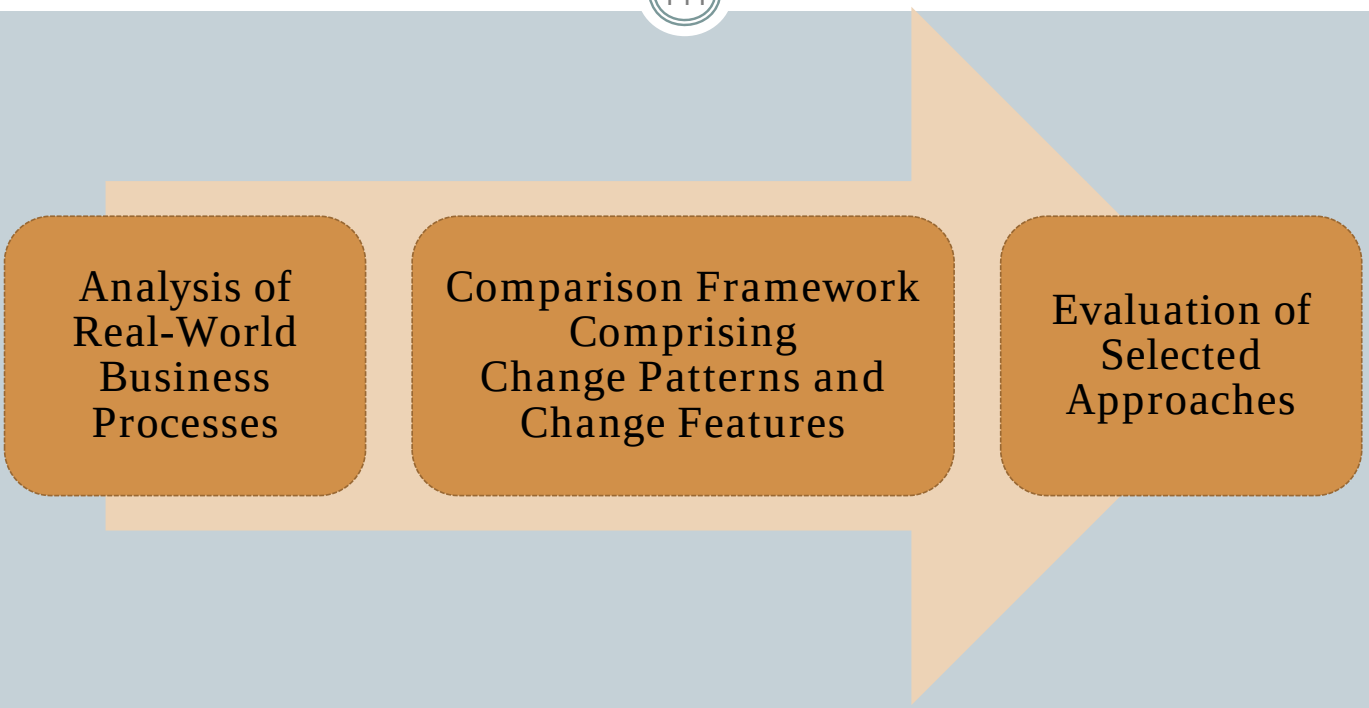
- Ability to deal with process changes is among the critical success factors for any process-aware information system (Mutschler et al. 2008 [MRB08])
- Several competing approaches to foster flexibility in process-aware information systems
  - Adaptive workflows (e.g., [ReDa98])
  - Case handling (e.g., [GRA08])
  - Declarative processes (e.g., [AaPe06, PSSA07,)
  - Late binding / Late Modeling (e.g., [SSO05])

☞Unfortunate lack of methods for a systematic comparison

3.2. Patterns-based Evaluation of PAIS Regarding Process Flexibility

- Workflow Patterns
  - Control Flow Patterns (Aalst et al. 2003 [AHK+03])
  - Data Patterns (Russell et al. 2004 [RHE+04])
  - Resource Patterns (Russell et al. 2004 [RHEA04])
- Exception Handling Patterns (Russell et al. 2006 [RAH06])

3.3. Qualitative Comparison of Change Frameworks



3.4 Research Methodology for Pattern Identification

Selection Criteria

- Patterns to effectively deal with exceptions,
- cope with the evolving nature of business processes and
- support users to better deal with uncertainty by deferring decisions to run-time

Data Sources and Data Collection

- Large case study in the healthcare domain including as-is and to-be models, documentation on frequently occurring exceptions
- Process models from the automotive engineering domain with multiple model versions

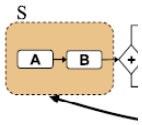
Pattern Identification Procedure

- Creation of candidate list based on literature study
- Analysis of data sources and refinement of candidate list of patterns

3.5 Systematic Description of Change Patterns

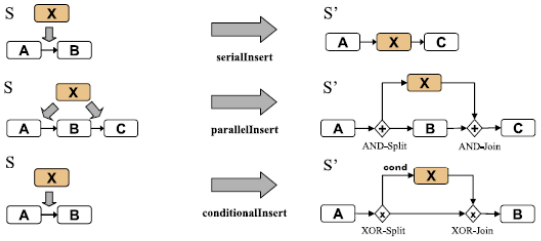
Pattern AP5: SWAP Process Fragment

Description	Two existing process fragments are swapped in process schema S.
Example	Regarding a particular delivery process the order in which requested go
Problem	
Implementation	
Related Patterns	



Pattern AP1: INSERT Process Fragment

Description	A process fragment X is added to a process schema S.
Example	For a particular patient an allergy test has to be added to his treatment process due to a drug incompatibility.
Problem	In a real world process a task has to be accomplished which has not been modeled in the process schema so far.
Design Choices (in addition to those described in Fig. 6)	C. How is the new process fragment X embedded in the process schema? 1. X is inserted between two directly succeeding activities (`serial insert`) 2. X is inserted between two activity sets (`insert between node sets`) a) without additional condition (`parallel insert`) b) with additional condition (`conditional insert`)



Implementation	This adaptation pattern can be realized by transforming the high level insertion operation into a sequence of low level change primitives (e.g., add node, add edge).

Pattern PP3: Late Composition of Process Fragments

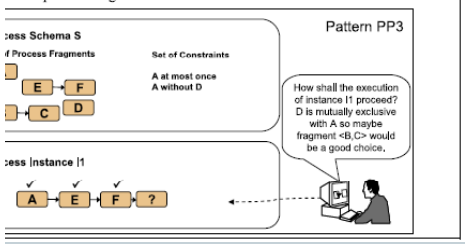
Description	At build-time a set of process fragments is defined from which the schema of a concrete process instance can be composed during run time. This can be achieved by dynamically nd by specifying the control dependencies between them on the fly.

dical examinations are accomplished in a hospital. The exact
plied to a particular patient and the order in which they are performed are
nt individually depending on his/her medical problems.

riants of how process fragments can be composed. To reduce the number
be specified by the process engineer during build time, process instances
posed from a given set of fragments.

sic building blocks for late modeling?
fragments from the repository can be chosen.
t-based subset of the process fragments from the repository can be

ies or process fragments can be defined.



Weber, Reichert & Rinderle-Ma . [WeRR08, WeRR07, RRW08b])

3.6 Change Patterns

Adaptation Patterns

Adding / Deleting Fragments

AP1: Insert Process Fragment
AP2: Delete Process Fragment

Moving / Replacing Fragments

AP3: Replace Process Fragment
AP4: Move Process Fragment
AP5: Swap Process Fragment
AP14: Copy Process Fragment

Adding / Removing Levels

AP6: Extract Sub Process
AP7: Inline Sub Process

Adapting Control Dependencies

AP8: Embed Process Fragment in Loop
AP9: Parallelize Activities
AP10: Embed Process Fragment in Conditional Branch
AP11: Add Control Dependency
AP12: Remove Control Dependency

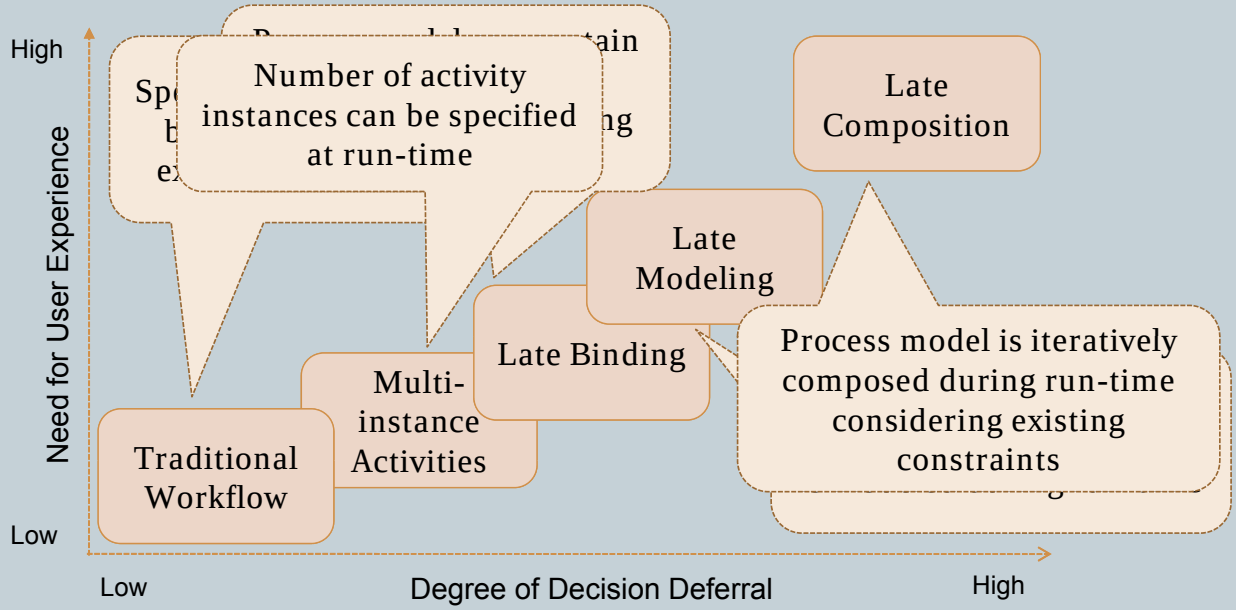
Changing Transition Conditions

AP13: Update Condition

Decision Deferral Patterns (Patterns for Changes in Predefined Regions)

PP1: Late Selection of Pr. Fragments
PP2: Late Modeling of Pr. Fragments
PP3: Late Composition of Process Fragments
PP4: Multi-Instance Activities

3.6.1 Patterns for Decision Deferral



3.7 Change Support Features

- Schema Evolution, Version Control and Instance Migration
- Support for Instance-Specific Changes
- Correctness of Changes
- Traceability and Analysis of Changes
- Access Control of Changes
- Change Reuse, User Assistance
- Change Concurrency Control
- Refactoring Support for Process Models

3.8 Change Patterns and Change Support Features in Practice – Process Type Level Changes

Primitive / Pattern	Academic								Commercial	
	ADEPT2 / CBRFlow	CAKE2	HOON	MOVE	PoF	WASA2	WIDE	YAWL + Worklets / Exlets	Flower	Staffware
Change Primitives										
PR1 – Add Node	–	+	+	+	+	+	+	+	+	+
PR2 – Remove Node	–	+	+	+	+	+	+	+	+	+
PR3 – Add Edge	–	+	+	+	+	+	+	+	+	+
PR4 – Remove Edge	–	+	+	+	+	+	+	+	+	+
PR5 – Move Edge	–	+	–	–	–	–	–	+	–	–
Adaptation Patterns										
AP1 – Insert Fragment	A[1, 2], B[1,2,3], C [1, 2]	–	–	–	–	–	A[2], B[1], C[1,2]	–	–	–
AP2 – Delete Fragment	A[1, 2], B[1,2,3]	–	–	–	–	–	A[2], B[1]	–	–	–
AP3 - Move Fragment	A[1, 2], B[1,2,3], C[1,2]	–	–	–	–	–	–	–	–	–
AP4 – Replace Fragment	–	–	–	–	–	–	A[2], B[1]	–	–	–
AP5 – Swap Fragment	–	–	–	–	–	–	–	–	–	–
AP6 – Extract Fragment	A[1,2], B[3]	–	–	–	–	–	–	–	–	–
AP7 – Inline Fragment	A[1,2], B[2]	–	–	–	–	–	–	–	–	–
AP8 – Embed Fragment in	A[1,2], B[1,2,3]	–	–	–	–	–	–	–	–	–
AP9 – Parallelize Activities	A[1,2], B[1,2,3]	–	–	–	–	–	–	–	–	–
AP10 - Embed Fragment in Conditional Branch	–	–	–	–	–	–	A[2]	–	–	–
AP11 – Add Control Dependency	A[1,2]	–	–	–	–	–	–	–	–	–
AP12 – Remove Control Dependencies	A[1,2]	–	–	–	–	–	–	–	–	–
AP13 – Update Condition	A[1,2]	–	–	–	–	–	A[2]	–	–	–
AP14 – Copy Fragment	–	–	–	–	–	–	–	–	–	–

3.8 Change Patterns and Change Support Features in Practice – Process Type Level Changes

Primitive / Pattern	Academic								Commercial	
	ADEPT2 / CBRFlow	CAKE2	HOON	MOVE	PoF	WASA2	WIDE	YAWL + Worklets / Exlets	Flower	Staffware
Change Primitives										
PR1 – Add Node	–	+	+	+	+	+	+	+	+	+
PR2 – Remove Node	–	+	+	+	+	+	+	+	+	+
PR3 – Add Edge	–	+	+	+	+	+	+	+	+	+
PR4 – Remove Edge	–	+	+	+	+	+	+	+	+	+
PR5 – Move Edge	–	+		–	–	–	–	+	–	–
Adapt atterns										
AP1 – Insert Fragment	A[1, 2], B[1,2,3], C [1, 2]	–			–	–	A[2], B[1], C[1,2]	–	–	–
AP2 – Delete Fragment	A[1, 2], B[1,2,3]	–						–	–	–
AP3 - Move Fragment	A[1, 2], B[1,2,3], C[1,2]	–						–	–	–
AP4 – Replace Fragment	–	–						–	–	–
AP5 – Swap Fragment	–	–						–	–	–
AP6 – Extract Fragment	A[1,2], B[3]	–						–	–	–
AP7 – Inline Fragment	A[1,2], B[2]	–						–	–	–
AP8 – Embed Fragment in	A[1,2], B[1,2,3]	–						–	–	–
AP9 – Parallelize Activities	A[1,2], B[1,2,3]	–						–	–	–
AP10 - Embed Fragment in Conditional Branch	–	–						–	–	–
AP11 – Add Control Dependency	A[1,2]	–	–	–	–	–	–	–	–	–
AP12 – Remove Control Dependencies	A[1,2]	–	–	–	–	–	–	–	–	–
AP13 – Update Condition	A[1,2]	–	–	–	–	–	A[2]	–	–	–
AP14 – Copy Fragment	–	–	–	–	–	–	–	–	–	–

Most evaluated approaches support process type level changes through change primitives

3.8 Change Patterns and Change Support Features in Practice – Process Type Level Changes

Primitive / Pattern	Academic								Commercial	
	ADEPT2 / CBRFlow	CAKE2	HOON	MOVE	P o F	WASA2	WIDE	YAWL + Worklets / Exlets	Flower	Staffware
Change Primitives										
PR1 – Add Node	–	+	+	+	+	+	+	+	+	+
PR2 – Remove Node	–	+	+	+	+	+	+	+	+	+
PR3 – Add Edge	–	+	+	+	+	+	+	+	+	+
PR4 – Remove Edge	–	+	+	+	+	+	+	+	+	+
PR5 – Move Edge	–	+	–	–	–	–	–	+	–	–
Adaptation Patterns										
AP1 – Insert Fragment	A[1, 2], B[1,2,3], C [1, 2]						A[2], B[1], C[1,2]	–	–	–
AP2 – Delete Fragment	A[1, 2], B[1,2,3]						A[2], B[1]	–	–	–
AP3 - Move Fragment	A[1, 2], B[1,2,3], C[1,2]						–	–	–	–
AP4 – Replace Fragment	–						A[2], B[1]	–	–	–
AP5 – Swap Fragment	–						–	–	–	–
AP6 – Extract Fragment	A[1,2], B[3]						–	–	–	–
AP7 – Inline Fragment	A[1,2], B[2]	–	–	–	–	–	–	–	–	–
AP8 – Embed Fragment in	A[1,2], B[1,2,3]	–	–	–	–	–	–	–	–	–
AP9 – Parallelize Activities	A[1,2], B[1,2,3]	–	–	–	–	–	–	–	–	–
AP10 - Embed Fragment in Conditional Branch	–	–	–	–	–	–	A[2]	–	–	–
AP11 – Add Control Dependency	A[1,2]	–	–	–	–	–	–	–	–	–
AP12 – Remove Control Dependencies	A[1,2]	–	–	–	–	–	–	–	–	–
AP13 – Update Condition	A[1,2]	–	–	–	–	–	A[2]	–	–	–
AP14 – Copy Fragment	–	–	–	–	–	–	–	–	–	–

Process type changes through change patterns are only supported in ADEPT and WIDE

3.9 Change Patterns and Change Support Features in Practice – Process Instance Level Changes

Primitive / Pattern	Academic								Commercial	
	ADEPT2 / CBRFlow	CAKE2	HOON	MOVE	P o F	WASA2	WIDE	YAWL + Worklets / Exlets	Flower	Staffware
Change Primitives										
PR1 – Add Node	–	+	–	–	–	+	–	–	–	–
PR2 – Remove Node	–	+	–	–	–	+	–	–	–	–
PR3 – Add Edge	–	+	–	–	–	+	–	–	–	–
PR4 – Remove Edge	–	+	–	–	–	+	–	–	–	–
PR5 – Move Edge	–	+	–	–	–	–	–	–	–	–
Adaptation Patterns										
AP1 – Insert Fragment	A[1, 2], B[1,2,3], C[1,2]	–	–	–	–	–	–	–	–	–
AP2 – Delete Fragment	A[1, 2], B[1,2,3]		–	–	–	–	–	–	A[2], B[1]	–
AP3 - Move Fragment	A[1, 2], B[1,2,3], C[1,2]	–		–	–	–	–	–	–	–
AP4 – Replace Fragment	–	–		–	–	–	A[1], B[1]	–	–	–
AP5 – Swap Fragment	–	–	–		–	–	–	–	–	–
AP6 – Extract Fragment	A[1,2], B[3]	–						–	–	–
AP7 – Inline Fragment	A[1,2], B[2]	–						–	–	–
AP8 – Embed Fragment in	A[1,2], B[1,2,3]	–						–	–	–
AP9 – Parallelize Activities	A[1,2], B[1,2,3]	–						–	–	–
AP10 - Embed Fragment in Conditional Branch	–	–						–	–	–
AP11 – Add Control Dependency	A[1,2]	–						–	–	–
AP12 – Remove Control Dependencies	A[1,2]	–						–	–	–
AP13 – Update Condition	A[1,2]	–						–	–	–
AP14 – Copy Fragment	–	–						–	–	–

Instance changes only supported by few approaches

3.9 Change Patterns and Change Support Features in Practice – Process Instance Level Changes

Primitive / Pattern	Academic								Commercial	
	ADEPT2 / CBRFlow	CAKE2	HOON	MOVE	PoF	WASA2	WIDE	YAWL + Worklets / Exlets	Flower	Staffware
Change Primitives										
PR1 – Add Node	–	+	–	–	–	+	–	–	–	–
PR2 – Remove Node	–	+	–	–	–	+	–	–	–	–
PR3 – Add Edge	–	+	–	–	–	+	–	–	–	–
PR4 – Remove Edge	–	+	–	–	–	+	–	–	–	–
PR5 – Move Edge	–	+	–	–	–	–	–	–	–	–
Adaptation Patterns										
AP1 – Insert Fragment	A[1, 2], B[1,2,3], C[1,2]	–	–	–	–	–	–	–	–	–
AP2 – Delete Fragment	A[1, 2], B[1,2,3]	–	–	–	–	–	–	–	A[2], B[1]	–
AP3 - Move Fragment	A[1, 2], B[1,2,3], C[1,2]	–	–	–	–	–	–	–	–	–
AP4 – Replace Fragment	–	–	–	–	–	–	A[1], B[1]	–	–	–
AP5 – Swap Fragment	–	–	–	–	–	–	–	–	–	–
AP6 – Extract Fragment	A[1,2], B[3]	–	–	–	–	–	–	–	–	–
AP7 – Inline Fragment	A[1,2], B[2]	–	–	–	–					–
AP8 – Embed Fragment in	A[1,2], B[1,2,3]	–	–	–	–					–
AP9 – Parallelize Activities	A[1,2], B[1,2,3]	–	–	–	–					–
AP10 - Embed Fragment in Conditional Branch	–	–	–	–	–					–
AP11 – Add Control Dependency	A[1,2]	–	–	–	–					–
AP12 – Remove Control Dependencies	A[1,2]	–	–	–	–					–
AP13 – Update Condition	A[1,2]	–	–	–	–					–
AP14 – Copy Fragment	–	–	–	–	–	–	–	–	–	–

CAKE and WASA2 support instance changes by means of change primitives

3.9 Change Patterns and Change Support Features in Practice – Process Instance Level Changes

Primitive / Pattern	Academic								Commercial	
	ADEPT2 / CBRFlow	CAKE2	HOON	MOVE	PoF	WASA2	WIDE	YAWL + Worklets / Exlets	Flower	Staffware
Change Primitives										
PR1 – Add Node	–	+	–	–	–	+	–	–	–	–
PR2 – Remove Node	–	+	–	–	–	+	–	–	–	–
PR3 – Add Edge	–	+	–	–	–	+	–	–	–	–
PR4 – Remove Edge	–	+	–	–	–	+	–	–	–	–
PR5 – Move Edge	–	+	–	–	–	–	–	–	–	–
Adaptation Patterns										
AP1 – Insert Fragment	A[1, 2], B[1,2,3], C[1,2]	–	–	–	–	–	–	–	–	–
AP2 – Delete Fragment	A[1, 2], B[1,2,3]	–	–	–	–	–	–	–	A[2], B[1]	–
AP3 - Move Fragment	A[1, 2], B[1,2,3], C[1,2]	–	–	–	–	–	–	–	–	–
AP4 – Replace Fragment	–	–	–	–	–	–	A[1], B[1]	–	–	–
AP5 – Swap Fragment	–	–	–	–	–	–	–	–	–	–
AP6 – Extract Fragment	A[1,2], B[3]	–	–	–	–	–	–	–	–	–
AP7 – Inline Fragment	A[1,2], B[2]	–	–	–	–					–
AP8 – Embed Fragment in	A[1,2], B[1,2,3]	–	–	–	–					–
AP9 – Parallelize Activities	A[1,2], B[1,2,3]	–	–	–	–					–
AP10 - Embed Fragment in Conditional Branch	–	–	–	–	–					–
AP11 – Add Control Dependency	A[1,2]	–	–	–	–					–
AP12 – Remove Control Dependencies	A[1,2]	–	–	–	–					–
AP13 – Update Condition	A[1,2]	–	–	–	–					–
AP14 – Copy Fragment	–	–	–	–	–	–	–	–	–	–

ADEPT allows for instance level changes through change patterns

3.10 Change Patterns and Change Support Features in Practice – Decision Deferral Patterns

Primitive / Pattern	Academic								Commercial	
	ADEPT2 / CBRFlow	CAKE2	HOON	MOVE	PoF	WASA2	WIDE	YAWL + Worklets / Exlets	Flower	Staffware
PP1 – Late Selection of Fragments	–	–	A[1,2], B[1,2], C[2]	–	–	–	–	A[1,2], B[1,2], C[2]	–	A[1,2], B[1,2], C[2]
PP2 – Late Modeling of Fragments	–	A[1], B[1], C[2,3], D[1]	–	A[1], B[1], C[3], D[1,2]	A[1,2], B[2], C[2], D[1,2]	–	–	–	–	–
PP3 – Late Composition of Fragments	–	–	–	–	–	–	–	–	–	–
PP4 – Multi…stance Activity	–	–	–		–	–	+	–	+	+

Most systems only support one of the decision deferral patterns

Declare supports Late Composition

Alaska supports Late Binding, Late Modeling and Late Composition

3.11 Change Patterns and Change Support Features in Practice – Change Support Features

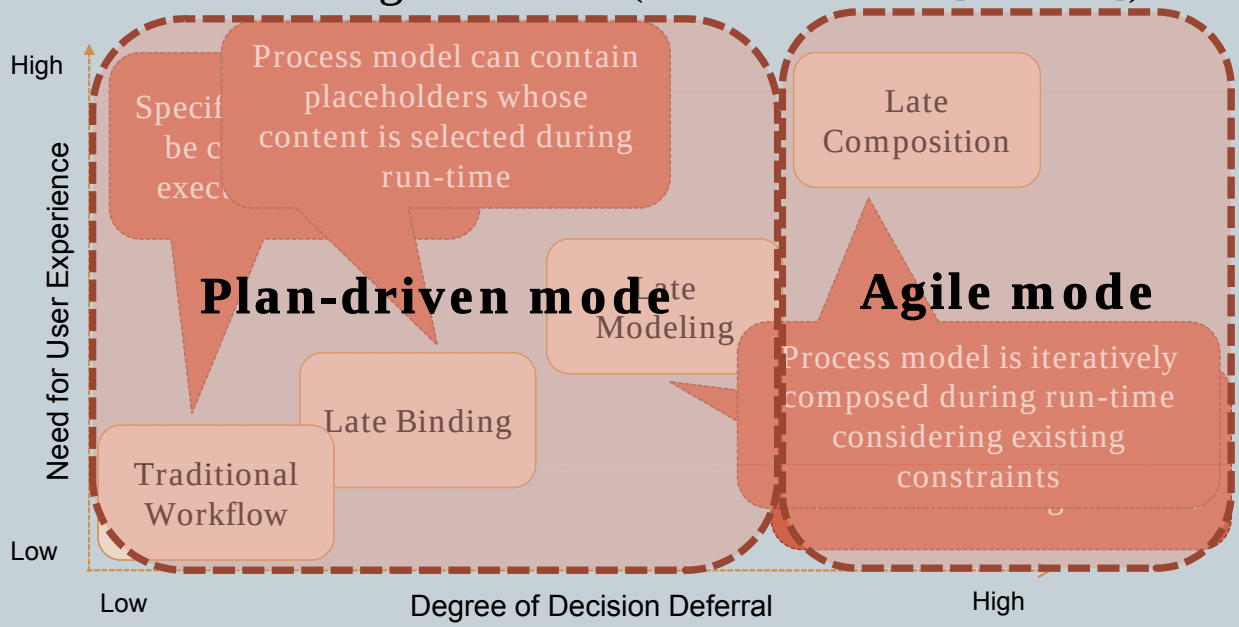
Feature	Academic								Commercial	
	ADEPT2 / CBRFlow	CAKE2	HOON	MOVE	PoF	WASA2	WIDE	YAWL + Worklets / Exlets	Flower	Staffware
F1 – Schema Evolution, Version Control and Instance Migration	3, 5	1	1	1	1	3,5	3,5	3	1,2,3	3,4
F2 – Support for Instance-Specific Changes	1a,b	1b, 2b	2a	2a	2b	1b	2b	1a,b, 2a,b	1b, 2b	2b
F3 – Correctness of Changes	+	+	+	+	+	+	+	°	–	–
F4 - Traceability & Analysis	1, 2, 3	1, 2	1	1	1	1	1	1	1	1
F5 – Access Control for Changes	1, 2, 3	–	1, 2, 3	1, 3	1, 2, 3	1	1, 3	1, 2, 3	1,2, 3*	1, 2, 3
F6 - Change Reuse	+	+	–	–	+	–	–	+	–	–
F7 - Change Concurrency Control	3, 4	3	3	3	3	2	not applicable	3	2	3

Most advanced support for change support features is provided by ADEPT2 / CBRFlow

(*) Flower supports Option 2 and 3 of feature F4 only for process instance changes, but not for process type changes

3.12 Empirical Evaluation of Process Flexibility with Alaska Simulator

- Patterns for deferring decisions (Weber et al. 2008 [WZP+09])

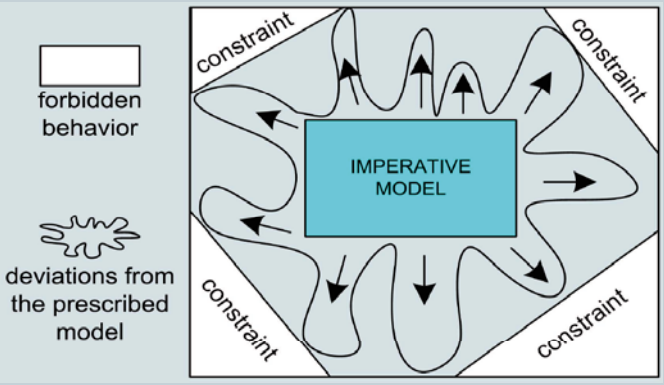


Agenda

1. Introduction
2. Flexibility Issues of the Imperative Approach to Business Process Management
3. Pattern-based evaluation of PAIS in respect to their ability to provide process flexibility
4. Flexibility Issues of the Declarative Approach to Business Process Management
5. Flexibility Issues in Data-driven Processes
6. Summary and Outlook

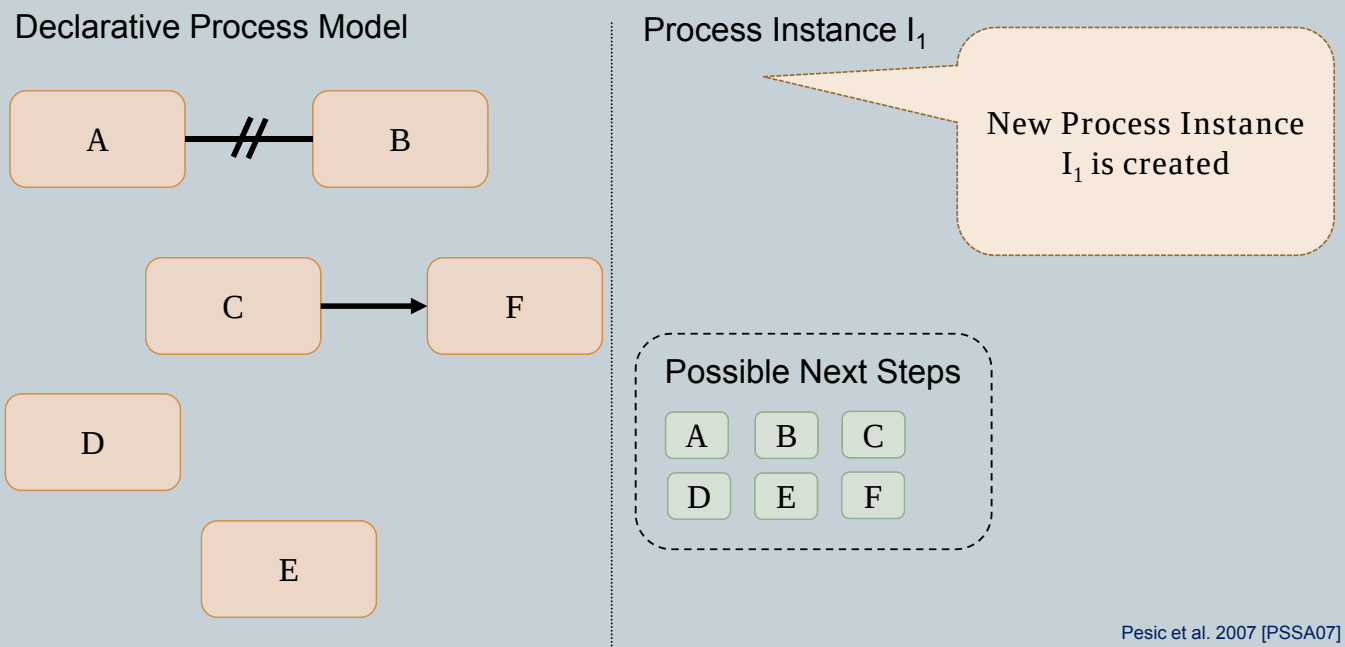
- Highest degree of flexibility offered by Late Composition as enabled by declarative approaches
- Advantages commonly attributed to declarative processes
  - Support for partial workflows (Wainer et al. 2004 [WBB04])
  - Allow users to defer decisions to run-time (Weber et al. 2008 [WeRR08])
  - Absence of over-specification (Pesic et al. 2007 [PSSA07])
  - More maneuvering room for end users (Pesic et al. 2007 [PSSA07]),

- Instead of describing exactly how a business process should be executed, declarative processes
  - describe the activities to be executed and
  - constraints prohibiting undesired behavior (e.g., selection constraints, ordering constraints, resource constraints)



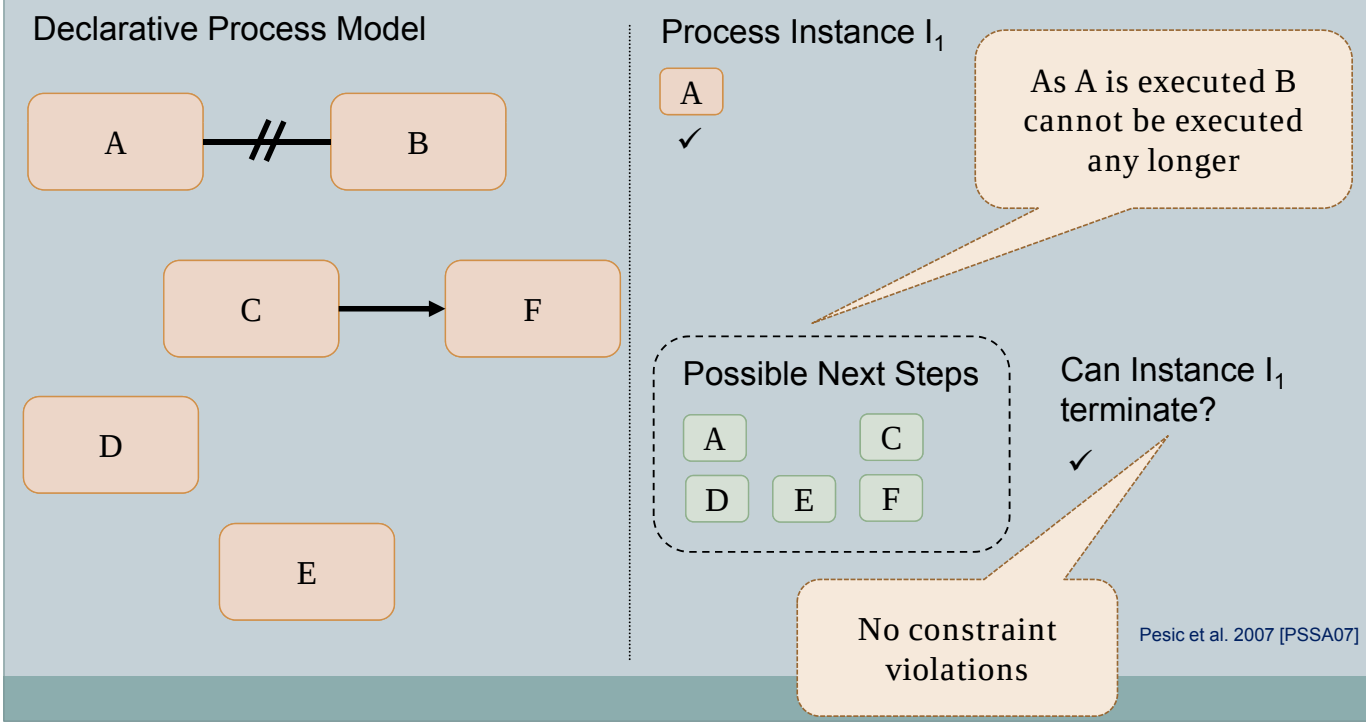
Pesic et al. 2007 [PSSA07]

4.2 Late Composition (1)



Pesic et al. 2007 [PSSA07]

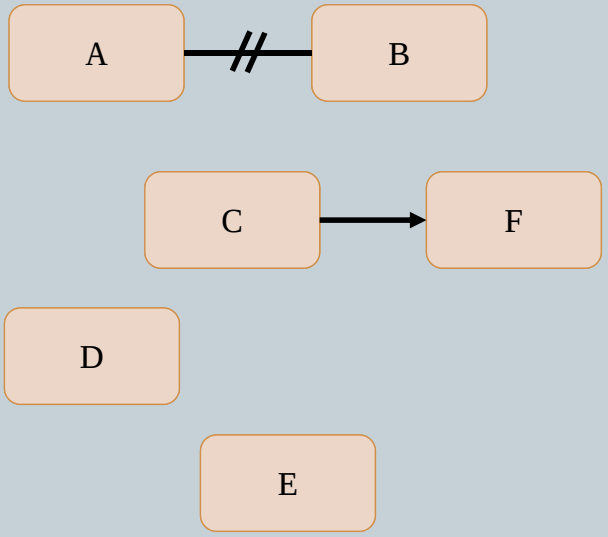
4.3 Late Composition (2)



Pesic et al. 2007 [PSSA07]

4.3 Late Composition (3)

Declarative Process Model



Process Instance I_1

A ✓ C ✓

Constraints are temporarily violated as F must eventually follow C

Possible Next Steps

A C D E F

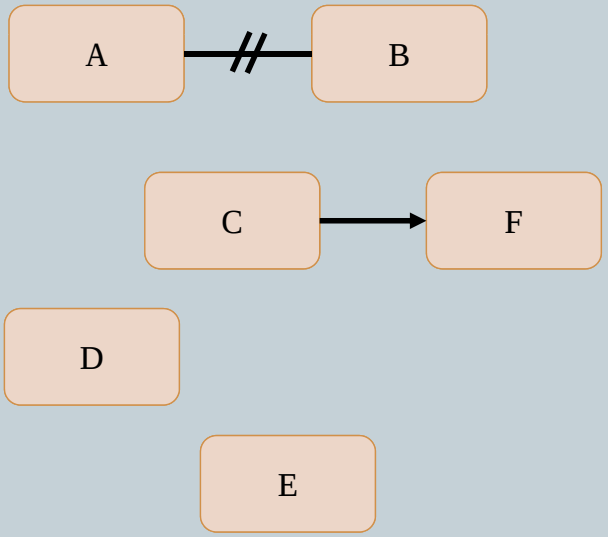
Can Instance I_1 terminate?

×

Pesic et al. 2007 [PSSA07]

4.3 Late Composition (4)

Declarative Process Model



Process Instance I_1

A ✓ C ✓ F ✓

Possible Next Steps

A C D E F

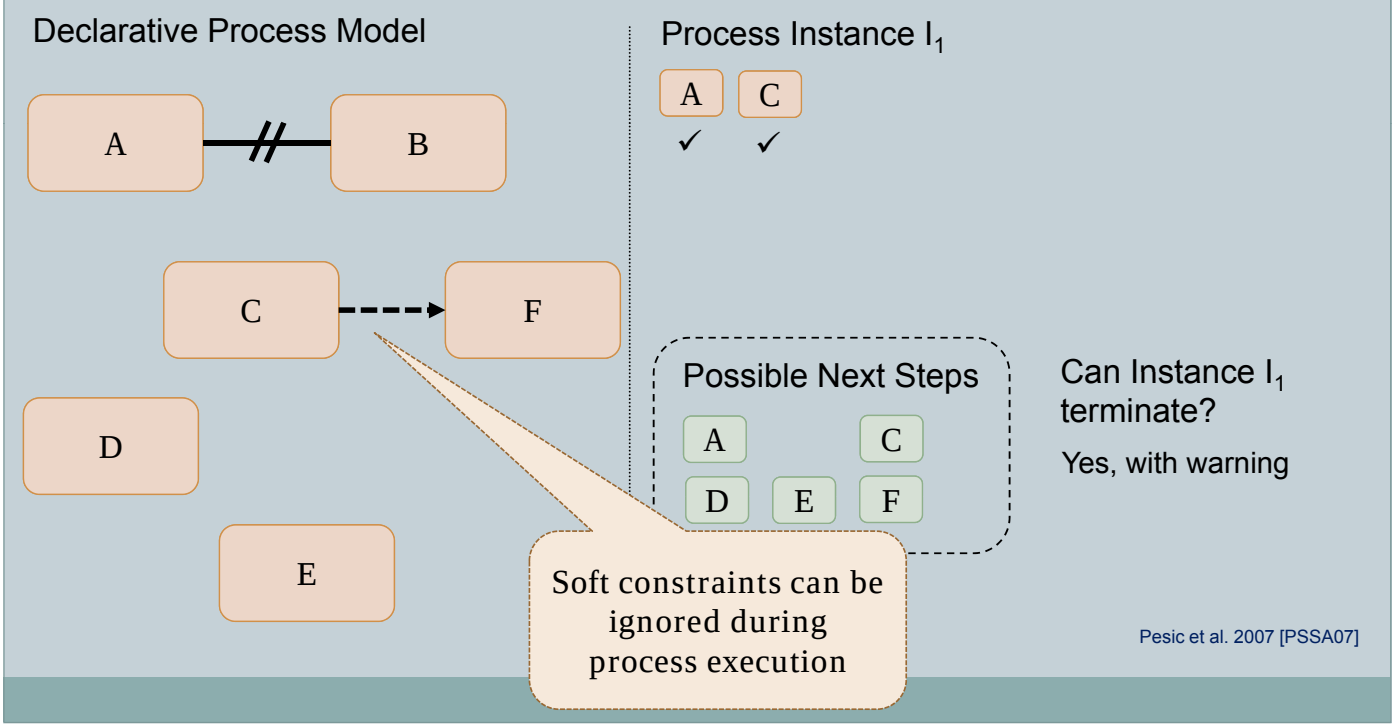
Can Instance I_1 terminate?

✓

No constraint violations

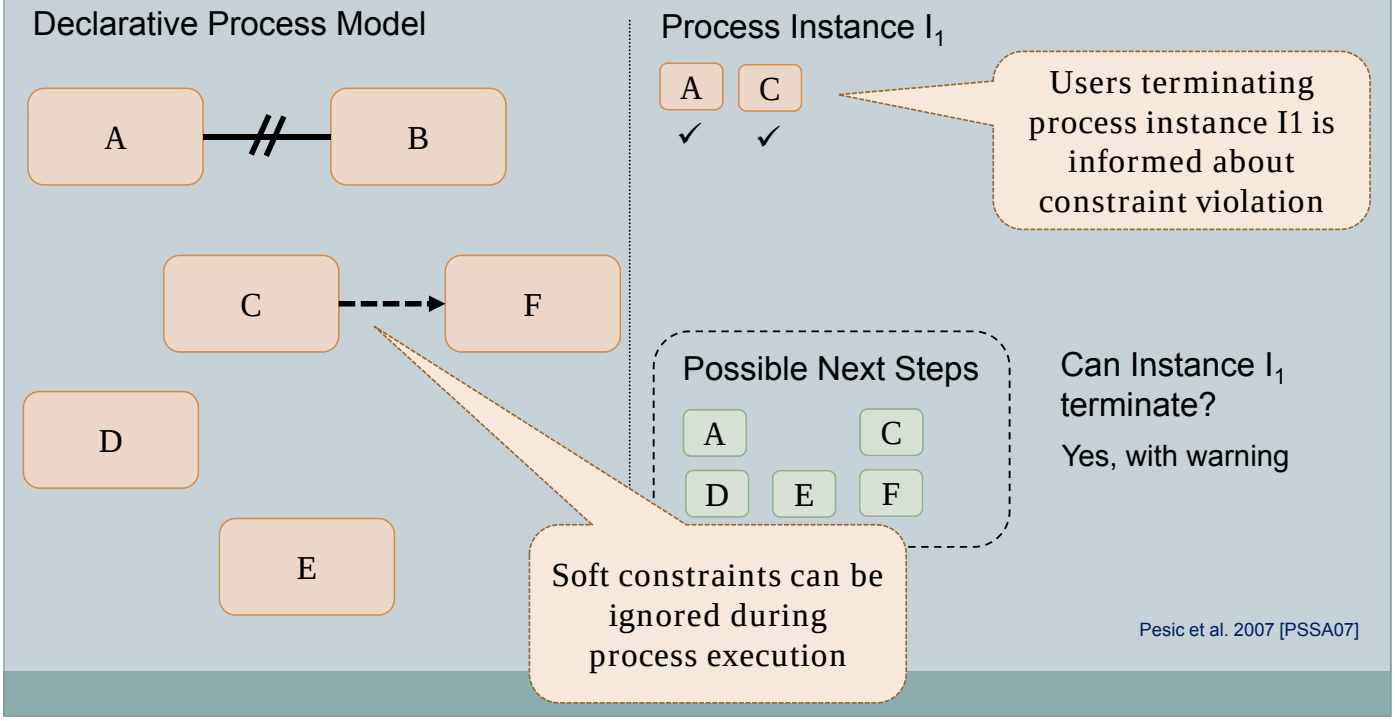
Pesic et al. 2007 [PSSA07]

4.3.1 Overriding Constraints (1)



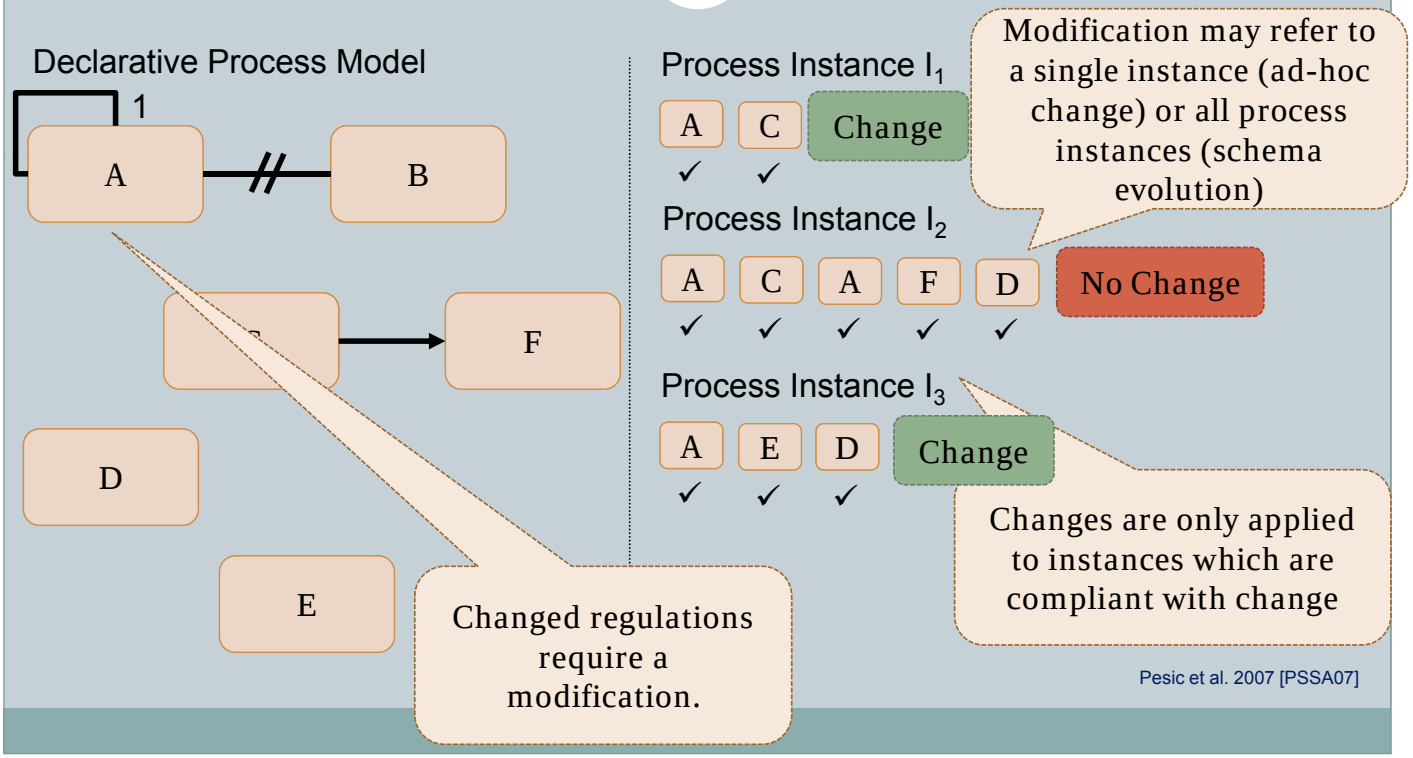
Pesic et al. 2007 [PSSA07]

4.3.1 Overriding Constraints (2)



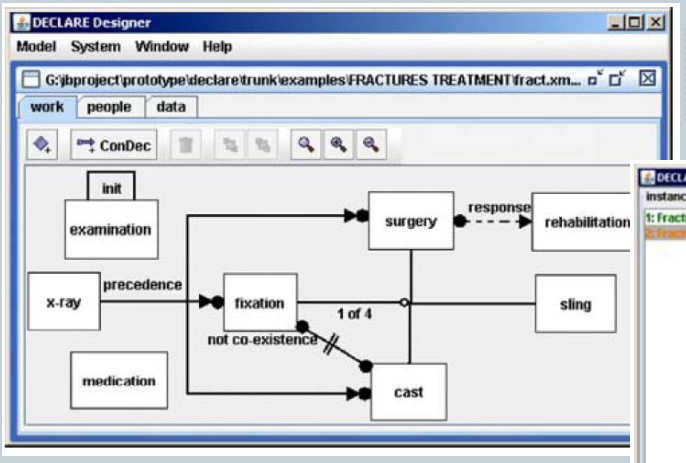
Pesic et al. 2007 [PSSA07]

4.3.2 Ad-hoc changes and Schema Evolution

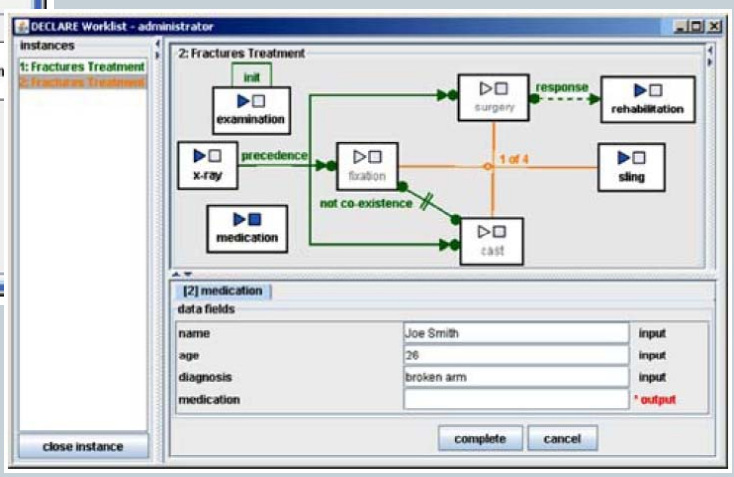


Pesic et al. 2007 [PSSA07]

4.4 The Declare System



Composing Declarative Processes with Declare

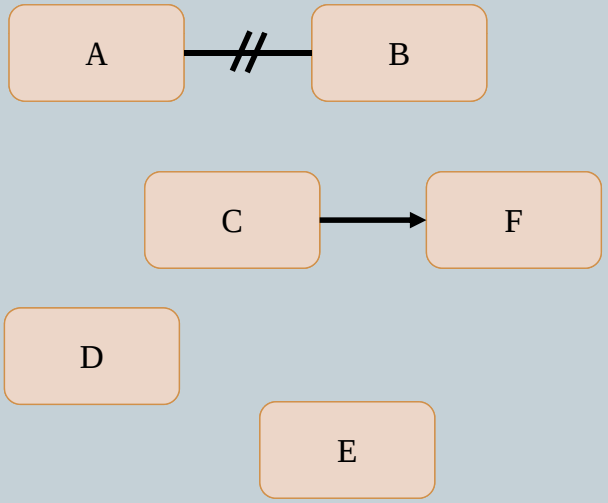


Executing Declarative Processes with Declare

van der Aalst, Pesic and Schonenberg 2009 [APS09]

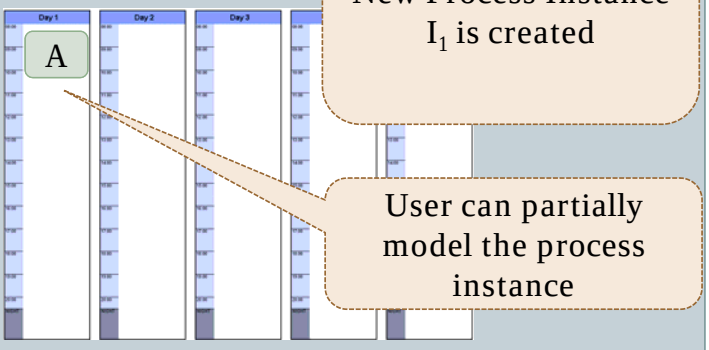
4.5 Late Composition in Alaska Simulator (1)

Declarative Process Model



Process Instance I_1

Calendar



Available Actions

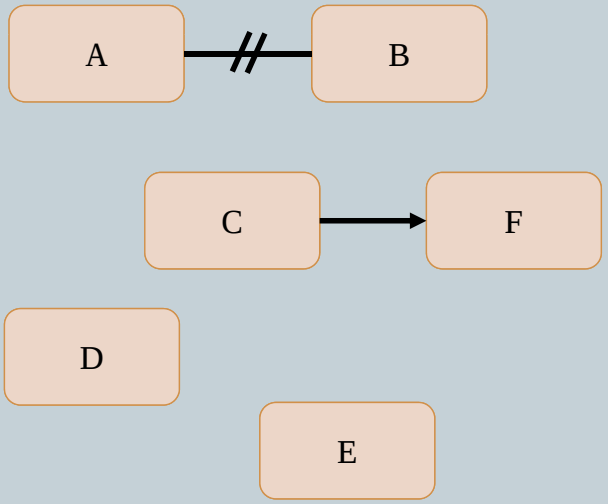
A B C

D E F

Current Problems

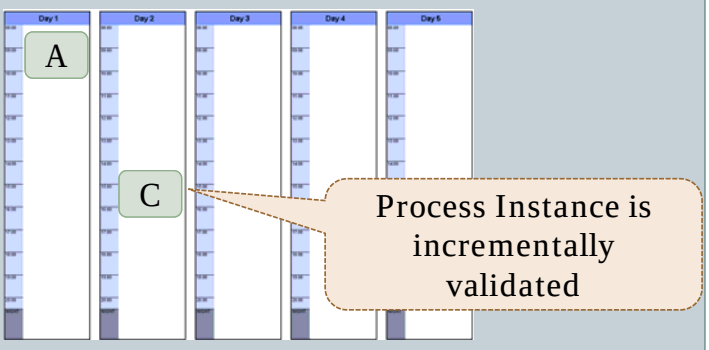
4.5 Late Composition in Alaska Simulator (2)

Declarative Process Model



Process Instance I_1

Calendar



Available Actions

A B C

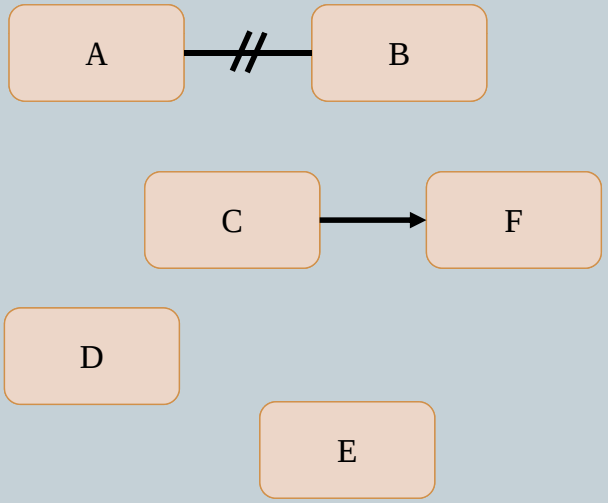
D E F

Current Problems

F must be executed

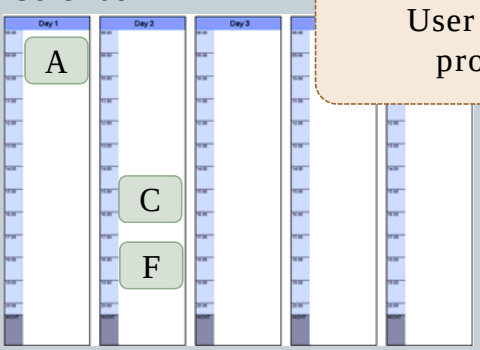
4.5 Late Composition in Alaska Simulator (3)

Declarative Process Model



Process Instance I_1

Calendar



User resolves problems

Available Actions

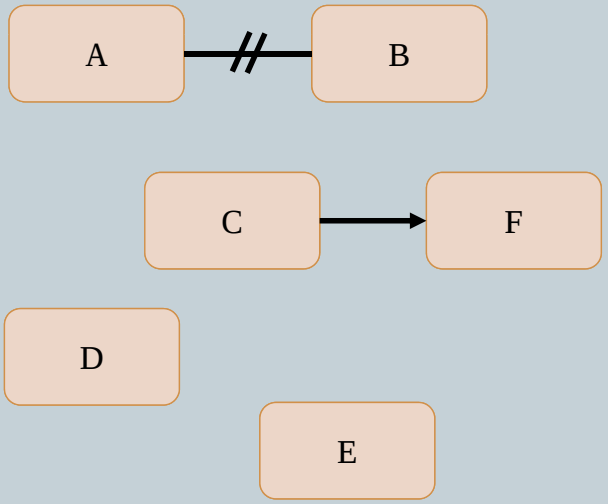
A B C

D E F

Current Problems

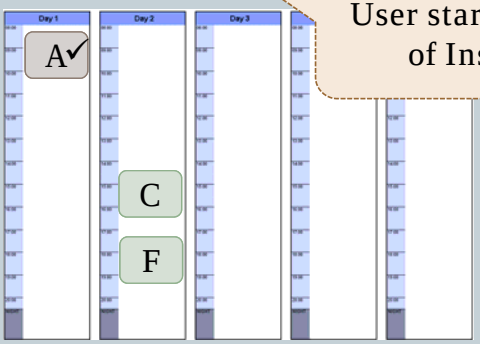
4.5 Late Composition in Alaska Simulator (4)

Declarative Process Model



Process Instance I_1

Calendar



User starts execution of Instance I_1

Available Actions

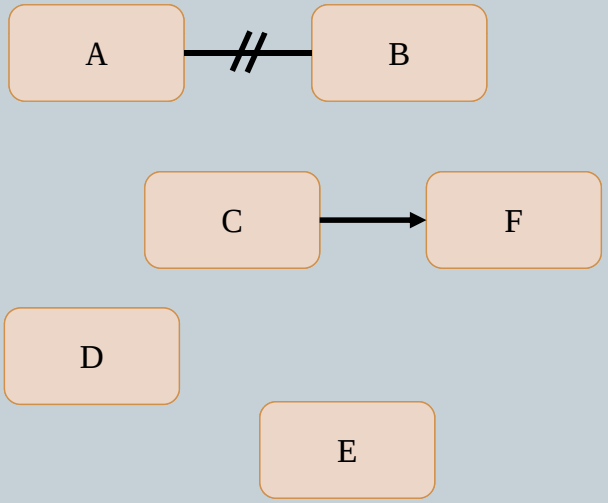
A B C

D E F

Current Problems

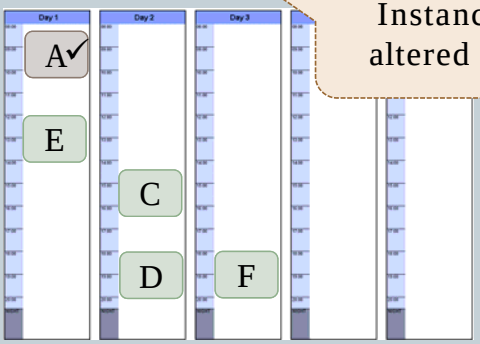
4.5 Late Composition in Alaska Simulator (5)

Declarative Process Model



Process Instance I_1

Calendar



Instance I_1 can be altered at all times

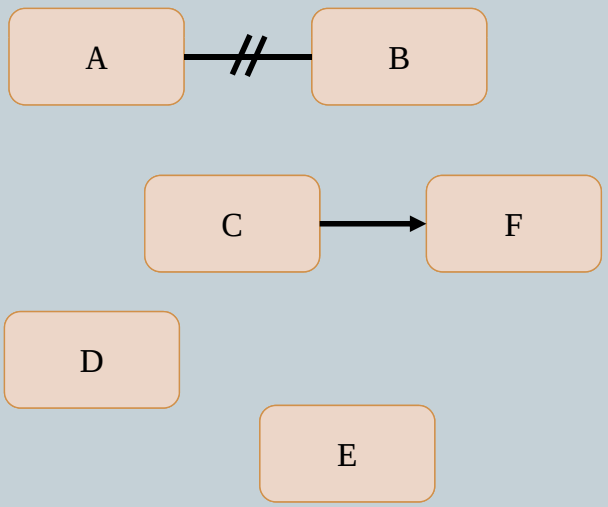
Available Actions

A B C
D E F

Current Problems

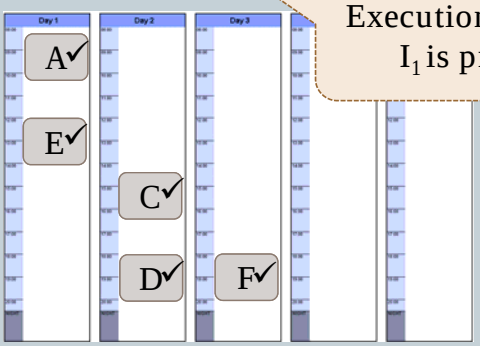
4.5 Late Composition in Alaska Simulator (5)

Declarative Process Model



Process Instance I_1

Calendar



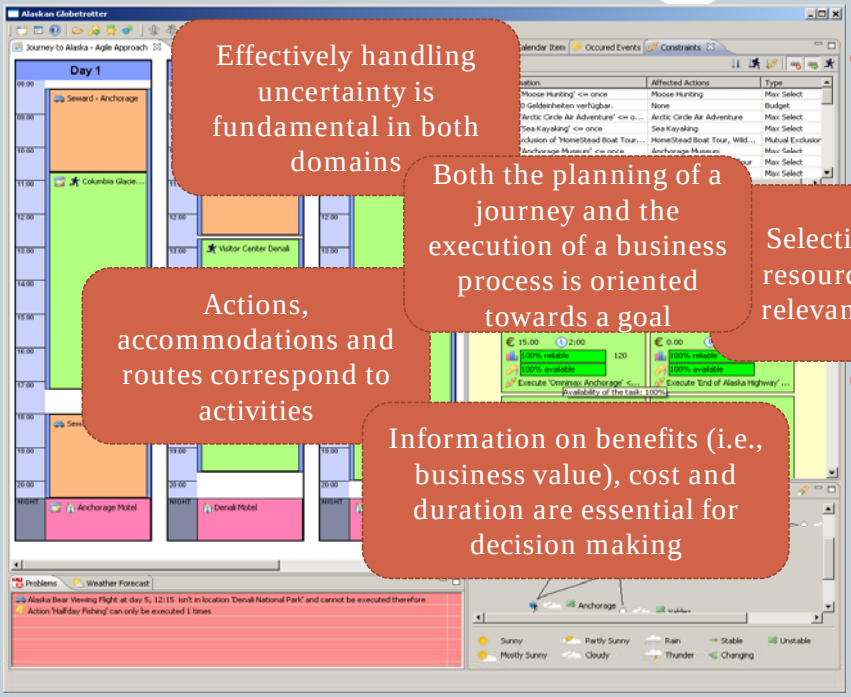
Execution of Instance I_1 is proceeded

Available Actions

A B C
D E F

Current Problems

4.6 The Alaska Simulator



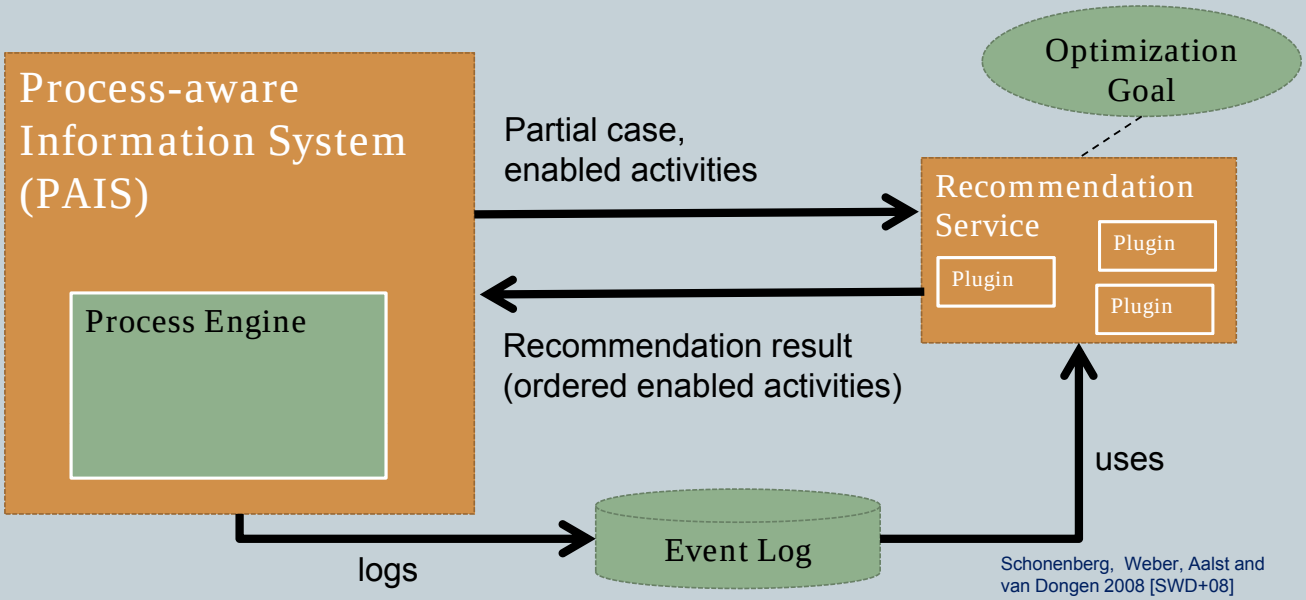
- Is an interactive planning tool providing support [illegible]nposition
- Uses journey as a metaphor for business processes

http:\\alaskasimulator.org

Weber, Zugal, Pinggera and Wild 2009 [WZP+09]

4.7 Assistance and Support for Declarative Workflows

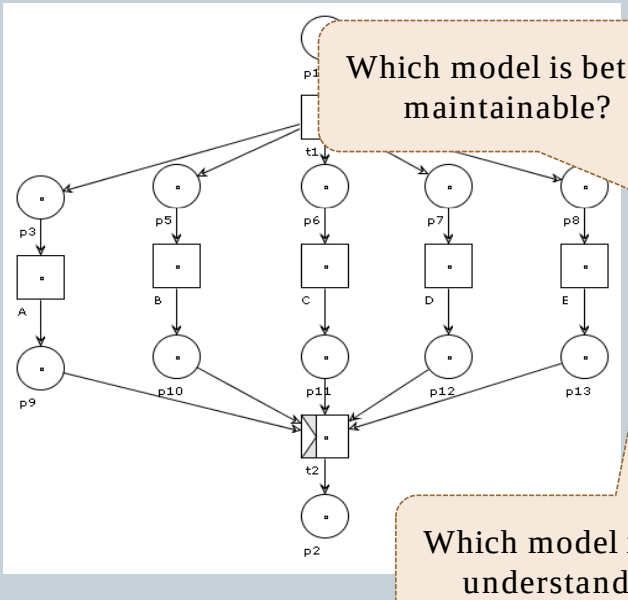
- Declarative workflows provide a lot of flexibility to the end user, but on the other hand require adequate user support



Schonenberg, Weber, Aalst and van Dongen 2008 [SWD+08]

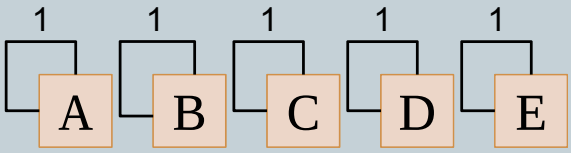
4.8 Discussion Imperative versus Declarative

Imperative Process Model (Workflow Net)



Which model is better maintainable?

Declarative Process Model (ConDec)



Lack of empirical evidence to establish superiority

Which model is better understandable?

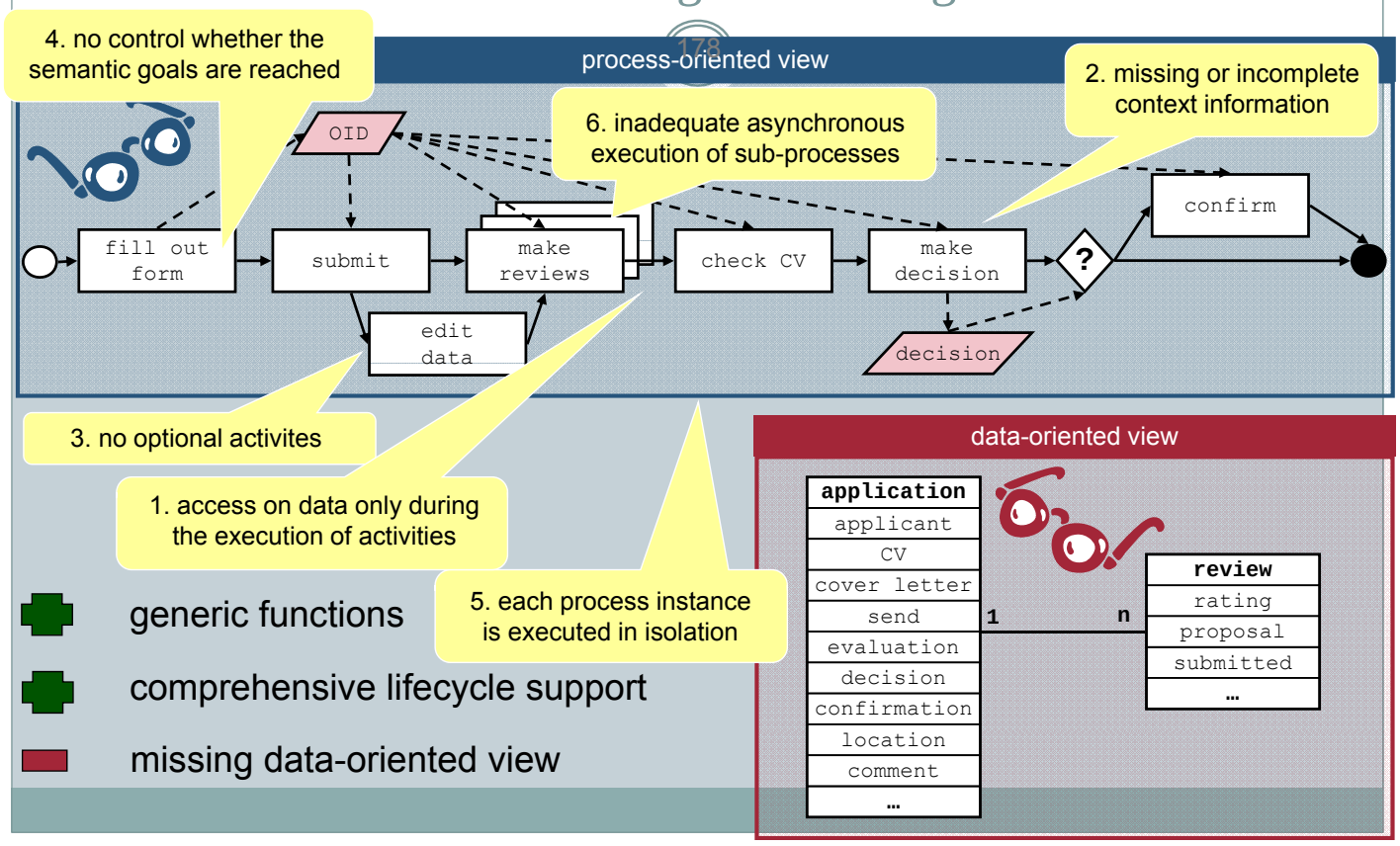
Fahland et al. 2009 [FMR+09a]
Fahland et al. 2009 [FLM+09b]

Agenda

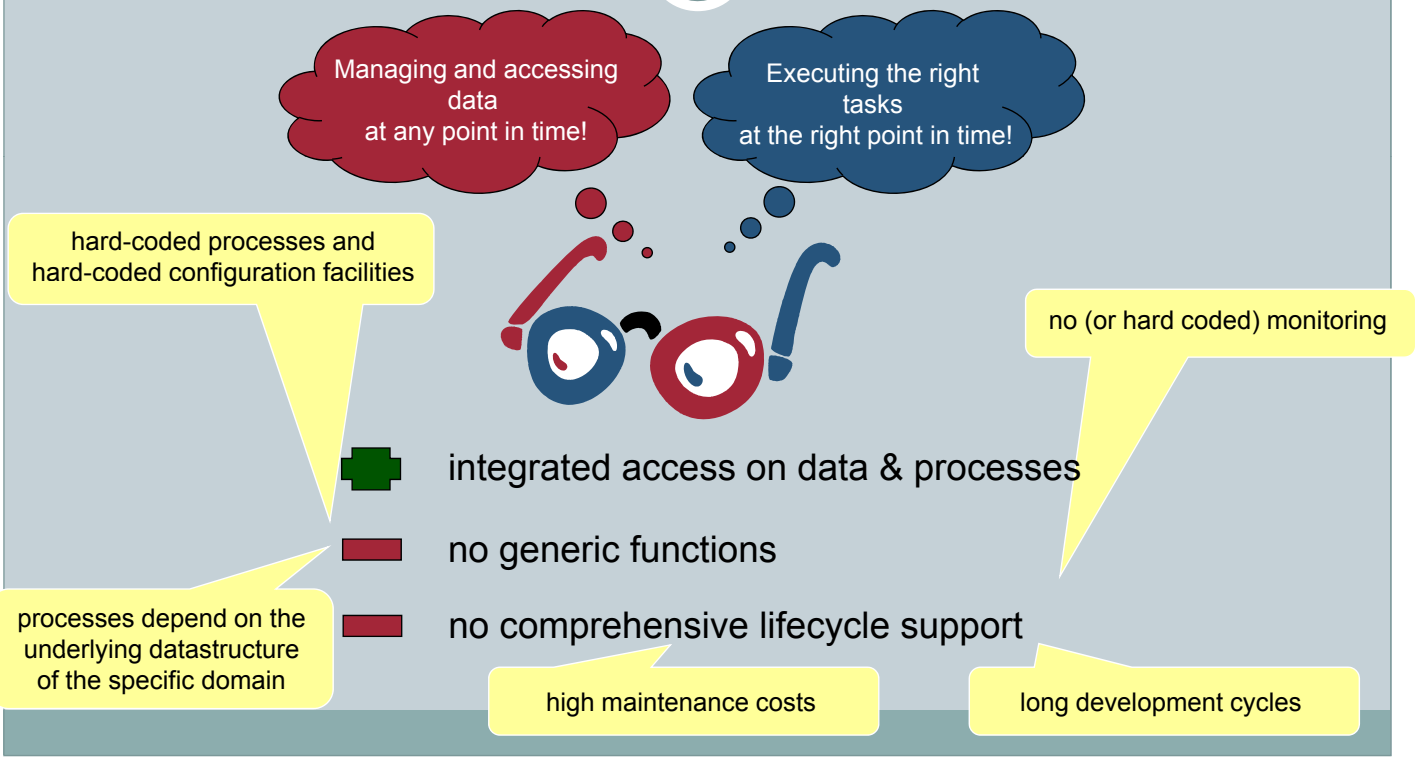
1. Introduction
2. Flexibility Issues of the Imperative Approach to Business Process Management
3. Pattern-based evaluation of PAIS in respect to their ability to provide process flexibility
4. Flexibility Issues of the Declarative Approach to Business Process Management
5. Flexibility Issues in Data-driven Processes
6. Summary and Outlook

- Contemporary WfMS & Application Systems
- Case Handling
- Object-aware Process Management Systems
- Data-driven Process Structures

5.1 Data Handling in Existing WfMS



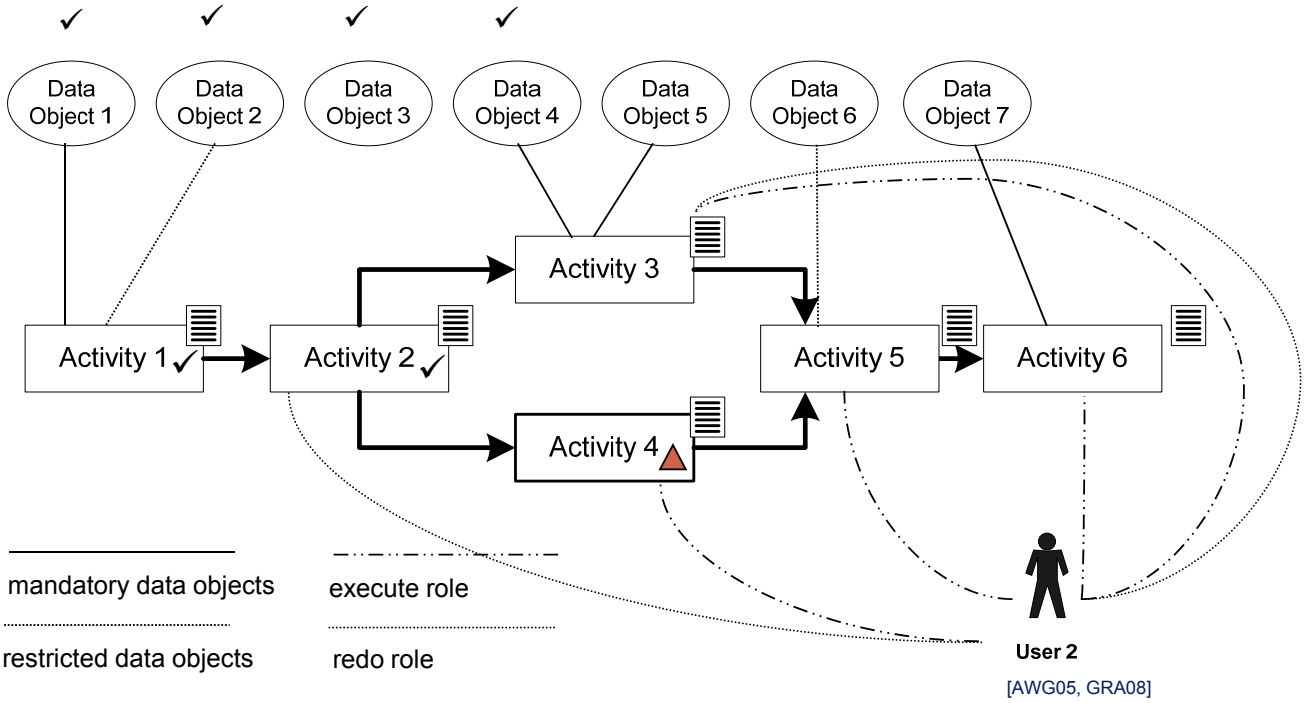
5.2 Processes and Data in Application Systems



5.3 What is needed?

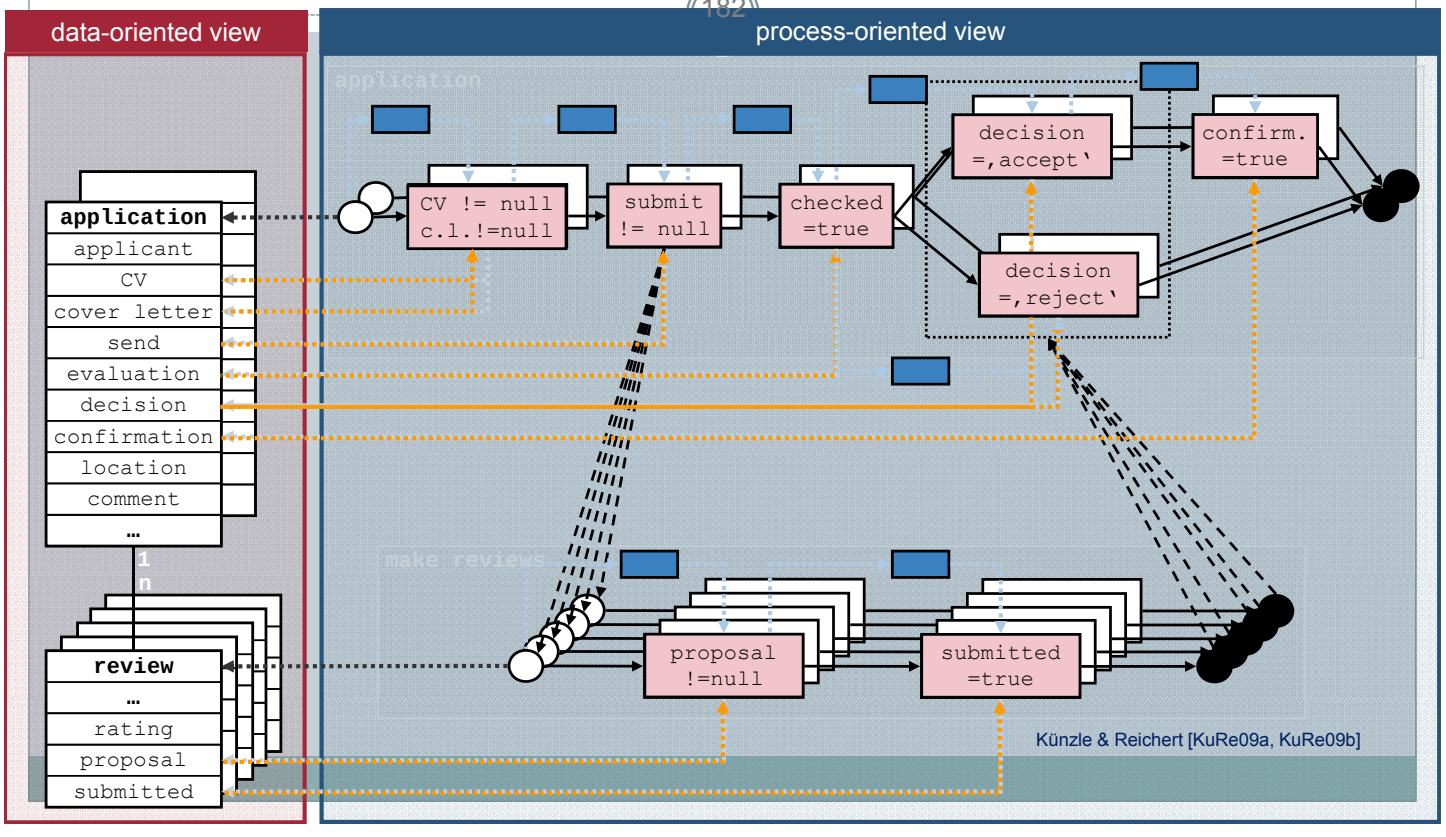
- Integrated access on business data and business processes
- Generic process support functions like in existing WfMS
- Utilization of the relation between process and data in the context of exception handling
- independency from a predefined data structure
- support of the full process lifecycle

5.4 Case Handling



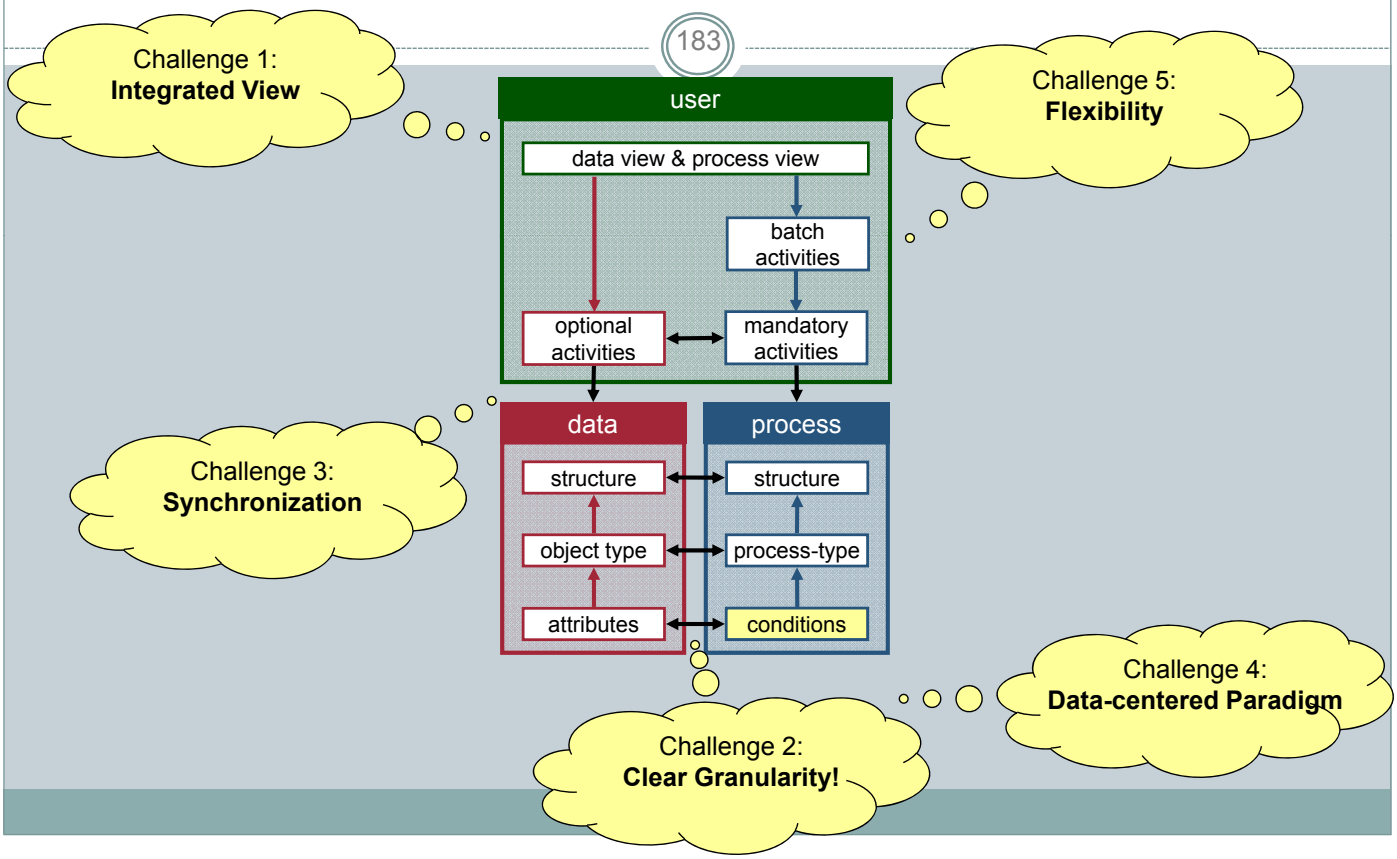
[AWG05, GRA08]

5.5 Object-aware Process Management (1)

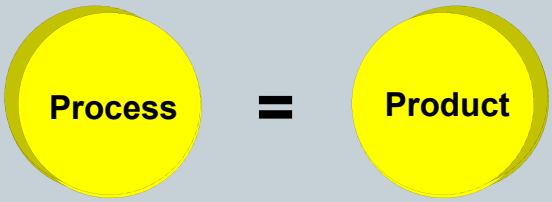


Künzle & Reichert [KuRe09a, KuRe09b]

5.5 Object-aware Process Management (2)



5.6 Data-driven Processes



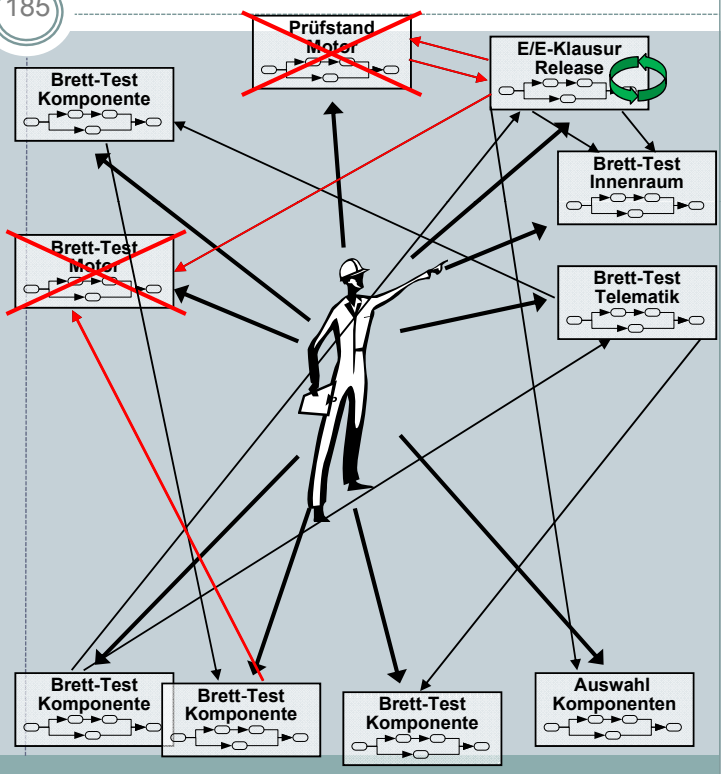
5.6.1 Data-driven Processes: Motivation (1)

Modeling

Execution

Dynamic Adaptation

Exception Handling



The Corepro Project -- [Müller, Reichert & Herbst [MRH07, MRH08]

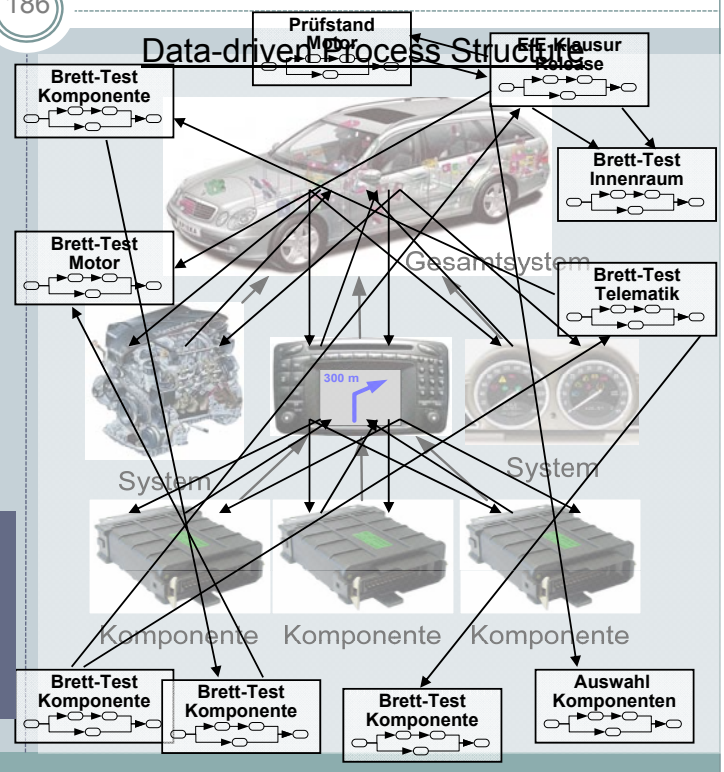
5.6.1 Data-driven Processes: Motivation (2)

Modeling

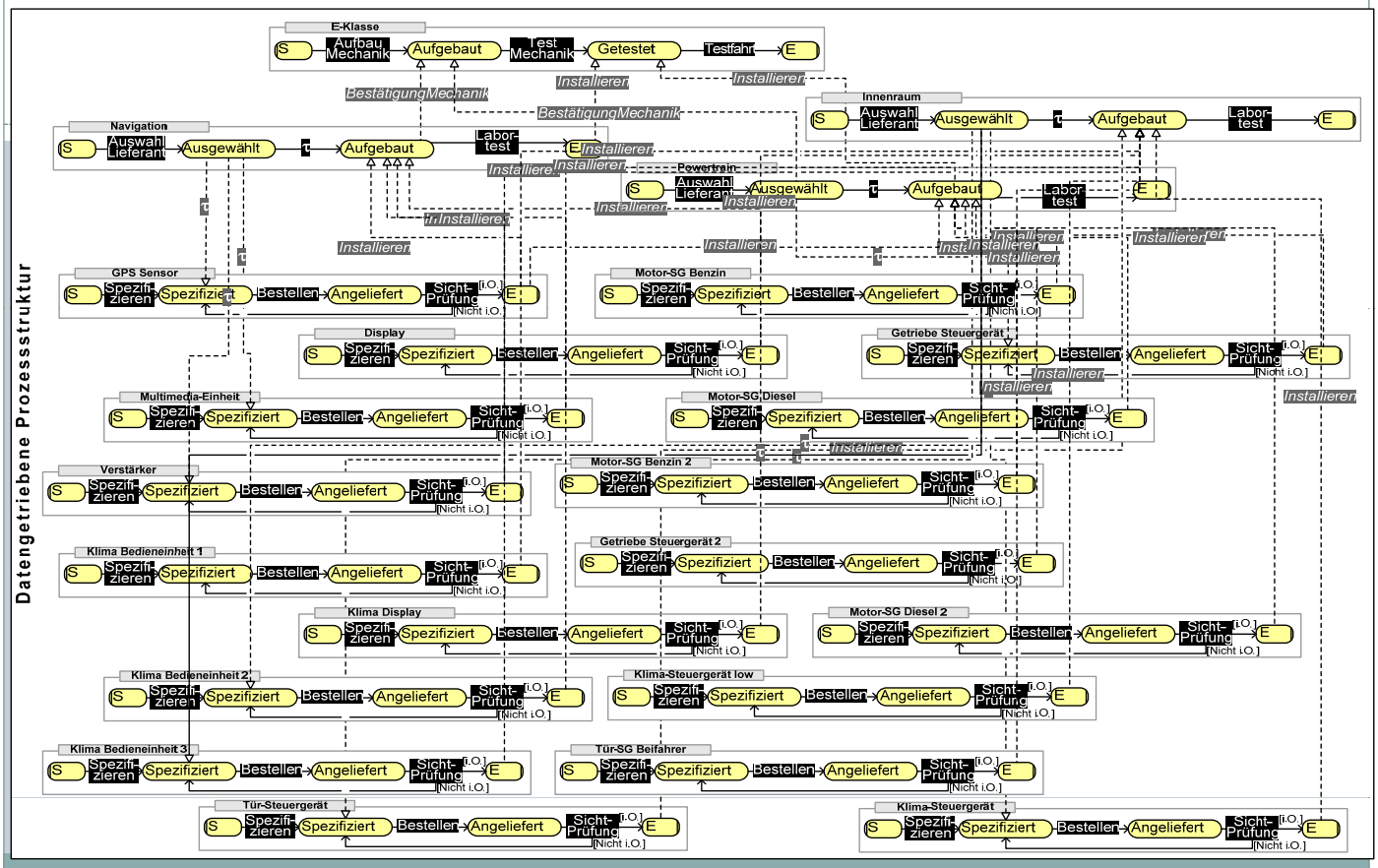
Execution

Dynamic Adaptation

Exception Handlung



Corepro: Integrated Support of Data-driven Process Structures



5.6.2 Modeling & Execution (1)

Modellebene

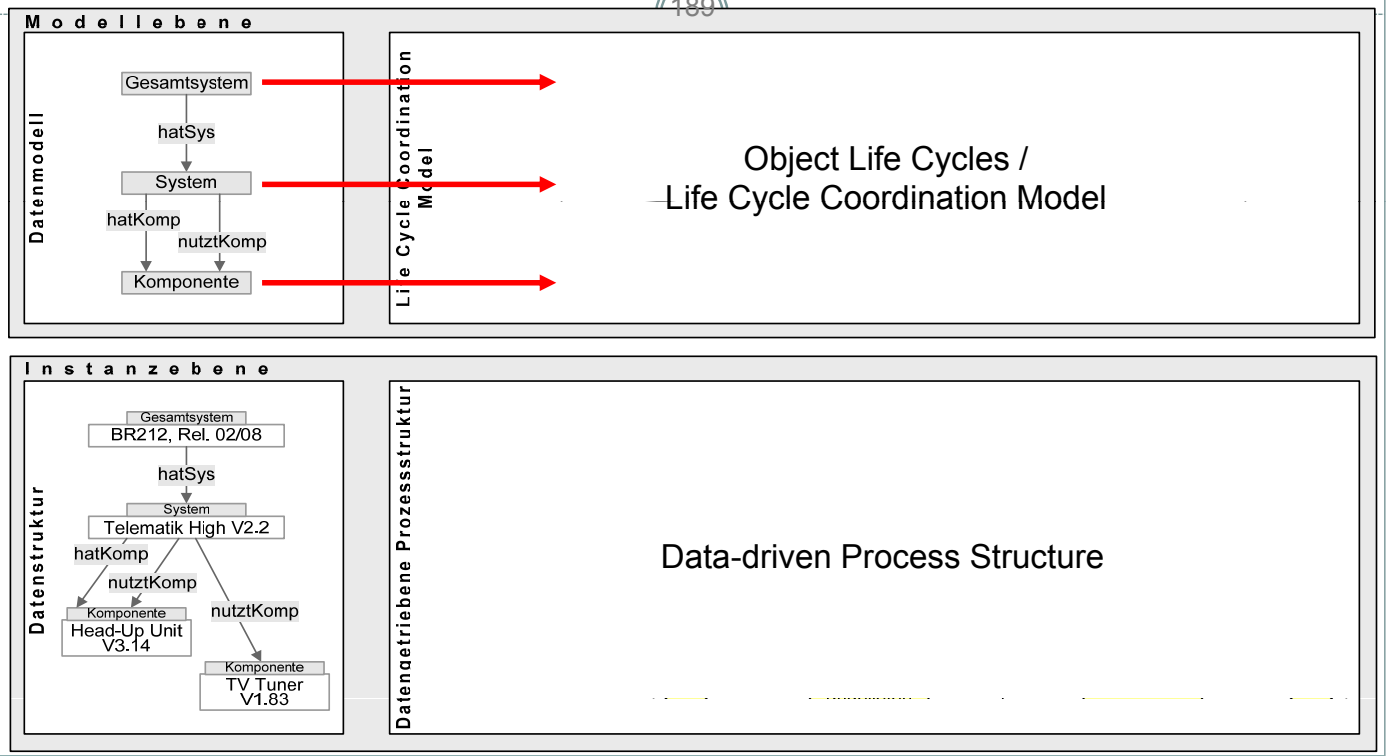
Datenmodell	Life Cycle Coordination Model
Datenmodell	Object Life Cycles / Life Cycle Coordination Model

Instanzebene

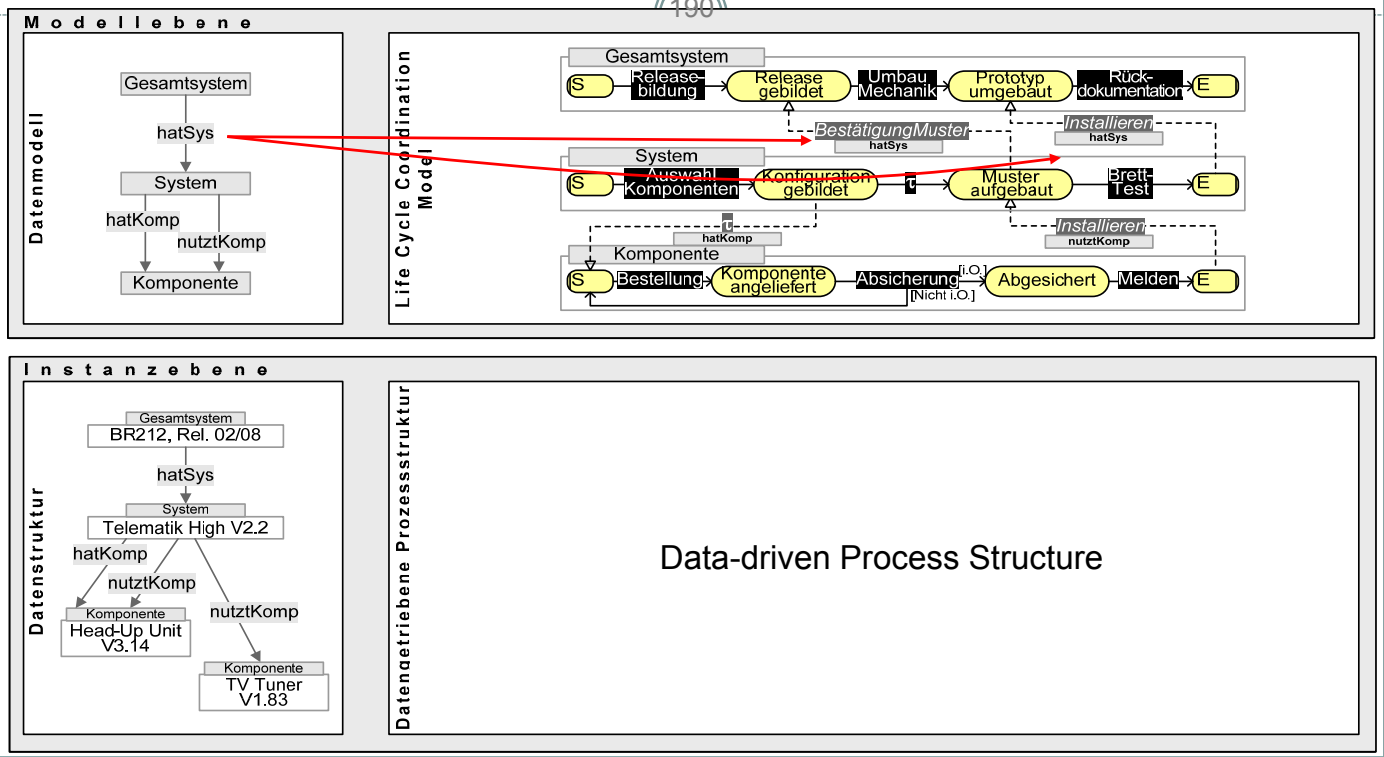
Datenstruktur	Datengetriebene Prozessstruktur
Datenstruktur	Data-driven Process Structure

[Müller, Reichert & Herbst [MRH07, MRH08]

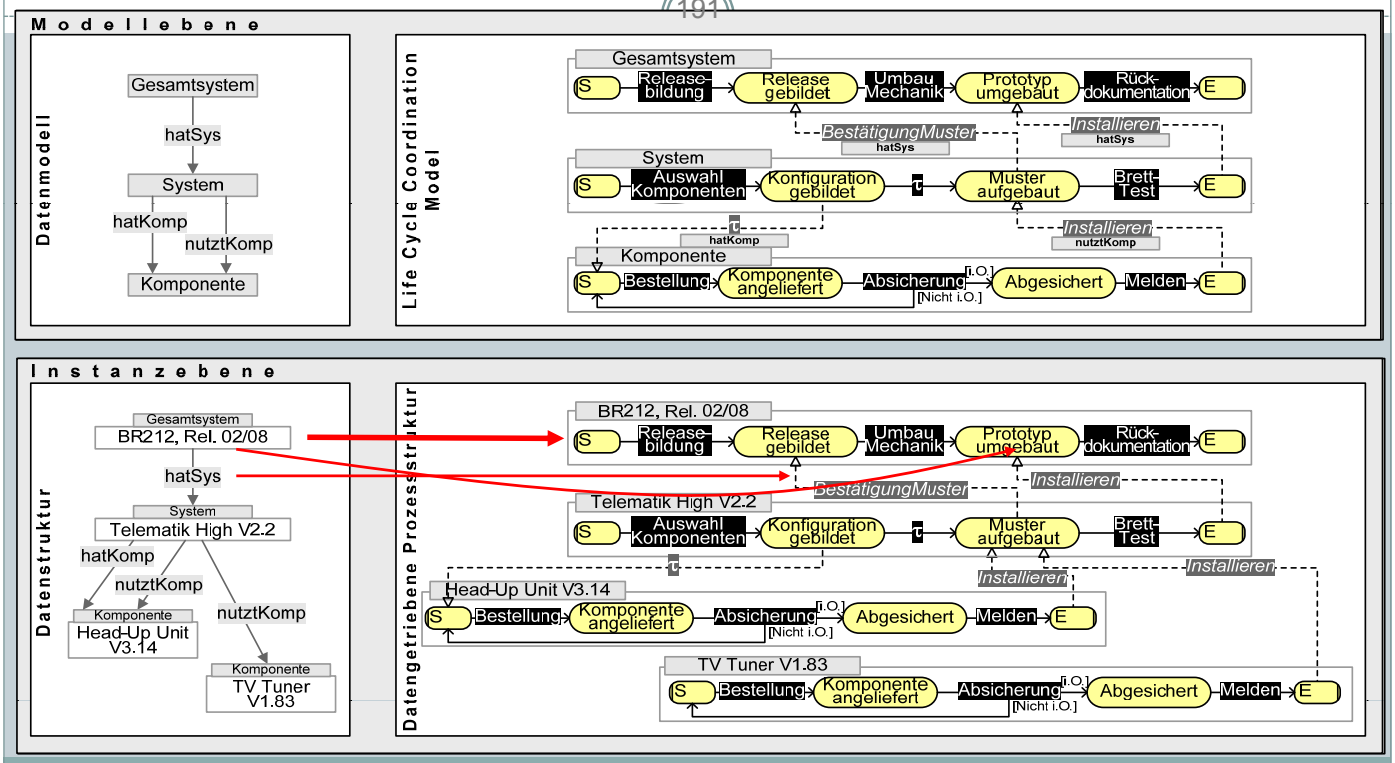
5.6.2 Modeling & Execution (2)



5.6.2 Modeling & Execution (3)



5.6.2 Modeling & Execution (4)



[Müller, Reichert & Herbst [MRH07, MRH08]

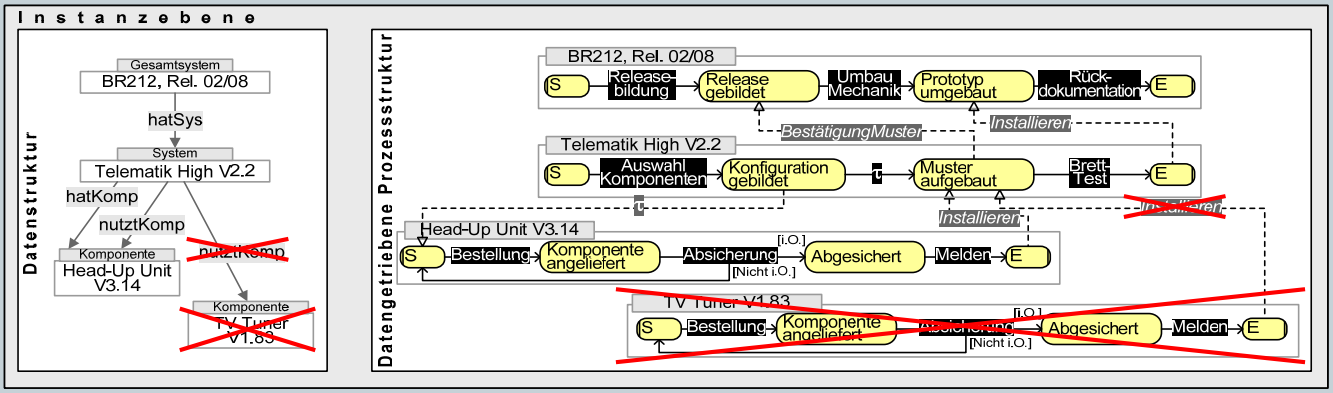
5.6.3 Dynamic Adaptation of Process Structures

Change Operation (Data Structure)

```
1) removeRelation(Telematik High
   V2.2, TV Tuner V1.83, nutztKomp);
2) removeObject(TV Tuner V1.83);
```

Change Operation (Process Structure)

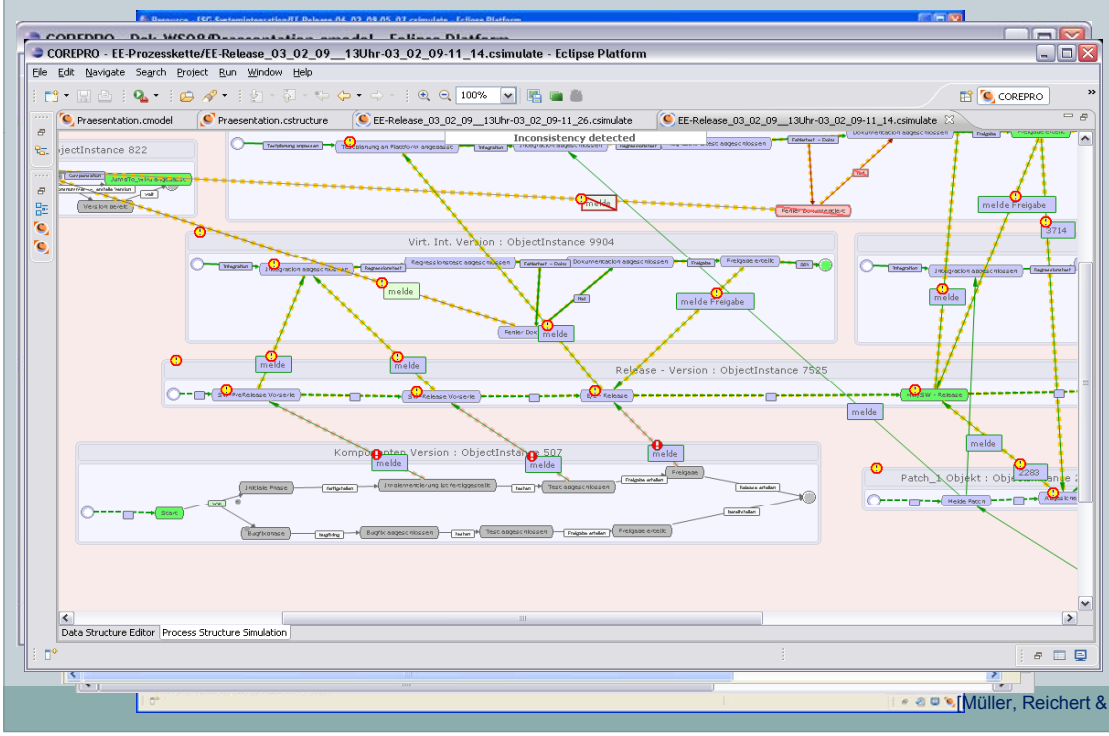
```
1) removeExtTrans(Telematik High V2.2 .
   Muster Aufgebaut, Installieren, TV Tuner
   V1.83 . E);
2) removeOLC(Tuner V1.83);
```



[Müller, Reichert & Herbst [MRH07, MRH08]

5.6.4 COREPRO Proof-of-Concept Demonstrator

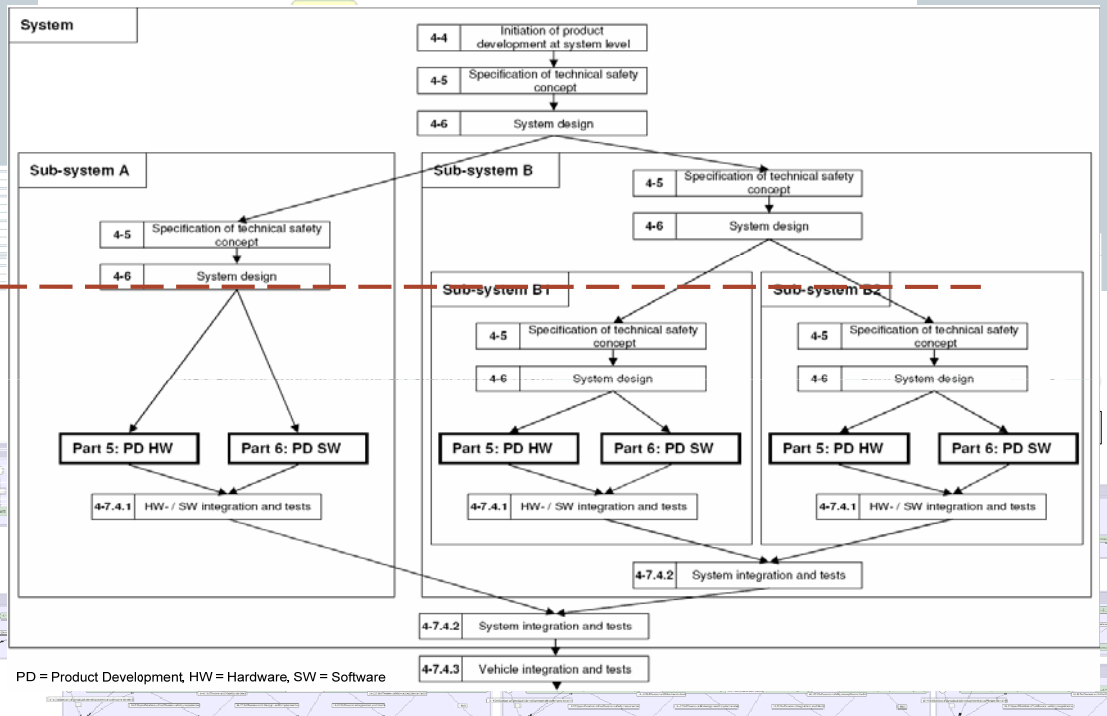
Aut[illegible] Structure



[Müller, Reichert & Herbst [MRH07, MRH08]

5.6.5 Case Study ISO 26262 „Road Vehicles – Functional Safety“

Instance Level: Data [illegible] and Process Structure



PD = Product Development, HW = Hardware, SW = Software

Agenda

1. Introduction
2. Flexibility Issues of the Imperative Approach to Business Process Management
3. Pattern-based evaluation of PAIS in respect to their ability to provide process flexibility
4. Flexibility Issues of the Declarative Approach to Business Process Management
5. Flexibility Issues in Data-driven Processes
6. Summary and Outlook

Cahpter 6: Summary & Outlook

- Different paradigms for business process support
  - Imperative Processes
  - Declarative Processes
  - Data-driven Processes
- Continuous introduction of process modeling languages and tools associated with superiority claims
- Empirical insights and theories missing

- Further Challenges
  - Changes of other aspects (e.g., org. models)
    (e.g., CEOSIS[RiRe08, RiRe09]
  - Changes of mobile / distributed processes
    (e.g., ADEPTdistribution [ReBa07])
  - Changes in process choreographies
    (e.g., DYCHOR [RWR06])

Thank you!

References

[AaPe06] van der Aalst; W.M.P.; Pesic, M. (2006) DecSerFlow: Towards a Truly Declarative ServiceFlow Language. TR, *BPMcenter.org* (2006)

[AaWe04] van der Aalst, W.M.P.; Weijters, A.J.M.M. (Eds.). Process Mining, *Computers in Industry*, Elsevier Science Publishers, Amsterdam, 2004.

[ADH+03] van der Aalst, W.M.P.; van Dongen, B.F.; Herbst, J.; Maruster, L.; Schimm, G.; Weijters, A.J. (2003): Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering* , 47(2):237-267

[AHE+06] Adams, M.J.; ter Hofstede, A.H.M.; Edmond, D.; van der Aalst, W.M.P. (2006) Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows. *Proc. 14th Int'l Conf. on Cooperative Information Systems (CoopIS'06)*, LNCS 4275, pp. 291-308

[AHK+04] Van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow Patterns. *Distributed and Parallel Databases* 14 (2003) 5-51

[APS09] van der Aalst, W.M.P.; Pesic, M.; Schonenberg, H. (2009): Declarative Workflows: Balancing Between Flexibility and Support. *Computer Science Research and Development*, 23: 99–113

[ARW+07] van der Aalst, W.M.P.; Reijers, H.A.; Weijters, A. J. M. M.; van Dongen, B.F.; de Medeiros, A.K.A.; Song, M.; Verbeek, H.M.W. (2007): Business process mining: An industrial application. *Inf. Syst.* 32(5): 713-732

[AWG05] van der Aalst, W.M.P.; Weske, M.; Grünbauer, D. (2005). Case handling: a new paradigm for business process support, *Data and Knowledge Engineering* 53(2):129–162.

[AWM04] van der Aalst, W.M.P.; Weijters, A.J.M.M.; Maruster, L. (2004): Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9): 1128-1142

[BBR06] Bobrik, R.; Bauer, T.; Reichert, M. (2006) Proviado – Personalized and Configurable Visualizations of Business Processes. *Proc. 7th Int. Conf. Electronic Commerce and Web Technologies (EC-WEB'06)*, Krakow, LNCS 4082, pp. 61-71

[CaPo99] Casati, F.; Pozzi, G. (1999) Modeling exceptional behaviors in commercial workflow management systems. *Proc. CoopIS'99*, pp.127 – 138.

[DaRe09] Dadam, P.; Reichert, M. (2009): The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support - Challenges and Achievements. Springer, *Computer Science - Research and Development*, 23(2): 81-97.

[DRR+08] Dadam, P.; Reichert, M.; Rinderle, S.; Jurisch, M.; Acker, H.; Göser, K.; Kreher, U.; Lauer, M. (2008) Towards Truly Flexible and Adaptive Process-Aware Information Systems. *Proc. UNISCON 2008*, Klagenfurt, LNBIP 5, pp. 72-83

[FMR+09a] Fahland, D.; Mendling, J.; Reijers, H.; Weber, B.; Weidlich, M.; Zugal, S. (2009): Declarative versus Imperative Process Modeling: The Issue of Maintainability. *Proc. ER-BPM'09 Workshop (in conjunction with BPM'09)*, Ulm, Germany

[FLM+09b] Fahland, D.; Lübke, D.; Mendling, J.; Reijers, H.; Weber, B.; Weidlich, M.; Zugal, S. (2009): Declarative versus Imperative Process Modeling: The Issue of Understandability. *Proc. BPMDS 2009 and EMMSAD 2009*, LNBIP 21.

[GaSa87] Garcia-Molina, H.; Salem, K. (1987): Sagas. Proc. SIGMOD'87, pp. 249-259

[GRR+06] Günther, C.W.; Rinderle, S.; Reichert, M.; van der Aalst, W.M.P (2006): Change Mining in Adaptive Process Management Systems. *Proc. 14th Int'l Conf. on Coop. Information Systems*, Montpellier, LNCS 4275, Nov. 2006, pp. 309-326.

[GRA08] Günther, C.W.; Reichert, M.; van der Aalst, W.M.P. (2008) Supporting Flexible Processes with Adaptive Workflow and Case Handling. *Proc. WETICE'08; 3rd IEEE Workshop on Agile Coop. Process-aware Information Systems (ProGility'08)*, June 2008, Rome, Italy. IEEE Computer Society Pres

[GRR+08] Günther, C.W.; Rinderle, S.; Reichert, M.; van der Aalst, W.M.P.; Recker, J. (2008): Using Process Mining to Learn from Process Changes in Evolutionary Systems. *Int'l J of Business Process Integration and Management*, 3(1):61-78

[HBR08] Hallerbach, A.; Bauer, T.; Reichert, M. (2008) Managing Process Variants in the Process Lifecycle. Proc. 10th Int'l Conf. on Enterprise Information Systems (ICEIS'08), June 2008, Barcelona, Spain. pp. 154-161

[HBR09a] Hallerbach, A.; Bauer, T.; Reichert, M. (2009): Guaranteeing Soundness of Configurable Process Variants in Provop. *Proc. 11th IEEE Conference on Commerce and Enterprise Computing (CEC'09)*, Vienna, Austria.

[HBR09b] Hallerbach, A.; Bauer, T.; Reichert, M. (2009): Capturing Variability in Business Process Models: The Provop Approach, *Software Process: Improvement and Practice*, Wiley InterScience (to appear)

[HBR09c] Hallerbach, A.; Bauer, T.; Reichert, M. (2009): Configuration and Management of Process Variants. In: *Int'l Handbook on Business Process Management.* Springer (to appear)

[ICB+07] Indulska, M.; Chong, S.; Bandara, W.; Sadiq, S.; Rosemann, M. (2007) Major Issues in Business Process Management: An Expert Perspective. *Proc. 15th European Conference on Information Systems*. St Gallen, Switzerland, June 7-9, 2007.

[KuRe09a] Künzle, V.; Reichert, M. (2009) Towards Object-aware Process Management Systems: Issues, Challenges, Benefits. *Proc. 10th Int'l Workshop on Business Process Modeling, Development, and Support (BPMDS'09)*, June 2009, Amsterdam, Springer, LNBIP 29, pp. 197-210

[KuRe09b] Künzle, V.; Reichert, M. (2009) Integrating Users in Object-aware Process Management Systems: Issues and Challenges. *Proc. BPM'09 Workshops; 5th Int'l Workshop on Business Process Design (BPD'09)*, Ulm, Germany (to appear)

[LaRos09] La Rosa, M.; van der Aalst, W.M.P.; Dumas, M.; ter Hofstede, A.H.M. (2009): Questionnaire-based variability modeling for system configuration. *Software and System Modeling* 8(2): 251-274 (2009)

[LeRe07]: Lenz, R.; Reichert, M. (2007) IT Support for Healthcare Processes - Premises, Challenges, Perspectives. *Data and Knowledge Engineering*, 61(1): 39-58

[LRD+09] Ly, L.T.; Rinderle-Ma, S.; Göser, K.; Dadam, P. (2009) On Enabling Integrated Process Compliance with Semantic Constraints in Process Management Systems. *Information Systems Frontiers* (Accepted for Publication)

[LRW08a] Li, C.; Reichert, M.; Wombacher, A. (2008) Discovering Reference Process Models by Mining Process Variants. *Proc. 6th Int'l Conference on Web Services (ICWS'08)*, September 2008, Beijing, China. IEEE Computer Society Press, pp. 45-53

[LRW08b] Li, C.; Reichert, M.; Wombacher, A. (2008): Mining Based on Learning from Process Change Logs. *Proc. BPM'08 workshops; 4th Int'l Workshop on Business Process Intelligence (BPI'08)*, September 2008, Milan, Italy. Springer, LNBIP 17, pp. 121-133

[LRW08c] Li, C.; Reichert, M.; Wombacher, A. (2008) On Measuring Process Model Similarity based on High-level Change Operations. Proc. 27th Int. Conference on Conceptual Modeling (ER'08), October 2008, Barcelona, Spain. Springer, LNCS 5231, pp. 248-264

[LRW09] Li, C.; Reichert, M.; Wombacher, A. (2009): Discovering Reference Models by Mining Process Variants Using a Heuristic Approach. *Proc. 7th Int'l Conf. on Business Process Management (BPM'09)*, September 2009, Ulm, Germany. LNCS 5701

[LuSa06] Lu, R.; Sadiq, S. (2006): Managing Process Variants as an Information Resource. *Proc. 4th Int'l Conf. on Business Process Management (BPM2006)*, Vienna

[LuSa07a] Lu, R.; Sadiq, S. (2007): On the Discovery of Preferred Work Practice through Business Process Variants. *Proc. 26th Int'l Conf. on Conceptual Modeling (ER'07)*, Nov 2007. Auckland, New Zealand.

[LuSa07b] Lu, R.; Sadiq, S. (2007): A Reference Architecture for Managing Business Process Variants. *Proc. 9th Int'l Conf. on Enterprise Information Systems (ICEIS'07)*, Funchal, Portugal, 2007

[MHH+06] Müller, D.; Herbst, J.; Hammori, M.; Reichert, M. (2006) IT Support for Release Management Processes in the Automotive Industry. Proc. 4th Int'l Conf. on Business Process Management (BPM'06), Vienna, Austria. LNCS 4102, pp. 368-377

[MRB08] Mutschler, B.; Reichert, M.; Bumiller, J. (2008) Unleashing the Effectiveness of Process-oriented Information Systems: Problem Analysis, Critical Success Factors and Implications. *IEEE Transactions on Systems, Man, and Cybernetics (Part C)*, Vol. 38, No. 3, pp. 280-291

[MRH07] Müller, D.; Reichert, M.; Herbst, J. (2007) Data-driven Modeling and Coordination of Large Process Structures. *Proc. 15th Int'l Conf. on Cooperative Information Systems (CoopIS'07)*, Vilamoura, Portugal. LNCS 4803, pp. 131-149

[MRH08] Müller, D.; Reichert, M.; Herbst, J. (2008) A New Paradigm for the Enactment and Dynamic Adaptation of Data-driven Process Structures. *Proc. 20th Int'l Conf. on Advanced Information Systems Engineering (CAiSE'08)*, Montpellier, France. Springer, LNCS 5074, pp. 48-63

[PoPo06] Poppendieck, M.; Poppendieck, T. (2006) Implementing Lean Software Development: From Concept to Cash, Addison-Wesley.

[PSSA07] Pesic, M.; Schonenberg, H.; Sidorova, N.; van der Aalst, W.M.P. (2007), *Constraint-Based Workflow Models: Change Made Easy*, Proc. CoopIS'07.

[RAH06] Russell, N.; van der Aalst, W.M.P.; ter Hofstede, A.H.M (2006): Workflow Exception Patterns *Proc. 18th Int'l Conf. on Advanced Information Systems Engineering (CAiSE 06)*, LNCS 4001, pp. 288-302.

[RoAa07] Rosemann, M.; van der Aalst, W.M.P. (2007) A Configurable Reference Modelling Language. *Information Systems*. 32(1): 1-23

[RDB03] Reichert, M.; Dadam, P.; Bauer, T. (2003): Dealing With Forward and Backward Jumps in Workflow Management Systems. *Int'l Journal Software and Systems Modeling*, 2(1):37-58

[RDR+09] Reichert, M.; Dadam, P.; Rinderle-Ma, S.; Jurisch, M.; Kreher, U.; Goeser, K. (2009): Architecural Principles and Components of Adaptive Process Management Technology. In: *PRIMIUM - Process Innovation for Enterprise Software*. LNI P-151. Koellen, pp. 81-97.

[ReBa07] Reichert, M.; Bauer, T. (2007) Supporting Ad-hoc Changes in Distributed Workflow Management Systems. *Proc. 15th Int'l Conf. on Coop. Information*

Systems (CoopIS'07), November 2007, Vilamoura, Portugal. Springer, LNCS 4803, pp. 150-168

[ReDa98] Reichert, M.; Dadam, P. (1998): ADEPT$_{flex}$-Supporting Dynamic Changes of Workflows Without Losing Control. *Journal of Intelligent Information Systems*, 10(2): 93-129

[ReDa09] Reichert, M.; Dadam, P. (2009): Enabling Adaptive Process-aware Information Systems with ADEPT2. In: Handbook of Research on Business Process Modeling. Information Science Reference, NY, pp. 173-203.

[Reic00] Reichert, M. (2000): Dynamische Ablaufänderungen in Workflow Management Systemen. *Dissertation*, Ulm University

[RHE+04] Russell, N.; ter Hofstede, A.; Edmond, D.; van der Aalst, W. (2004): Workflow data patterns. *Technical Report FIT-TR-2004-01*, QUT Brisbane

[RHEA04] Russell, N.; ter Hofstede, A.; Edmond, D.; van der Aalst, W. (2004): Workflow resource patterns. *Technical Report WP 127*, TU Eindhoven

[RiRe07] Rinderle-Ma, S.; Reichert, M. (2007) A Formal Framework for Adaptive Access Control Models. Springer, *Journal on Data Semantics IX* , LNCS 4601, pp. 82-112

[RiRe08] Rinderle-Ma, S; Reichert, M. (2008) Managing the Life Cycle of Access Rules in CEOSIS. *Proc. 12th IEEE Int'l Enterprise Computing Conf. (EDOC'08)*, September, 2008, Munich, Society Press, pp. 257-266

[RiRe09] Rinderle-Ma, S.; Reichert, M. (2009): Comprehensive Life Cycle Support for Access Rules in Information Systems: The CEOSIS Project. Enterprise *Information Systems*, 3(3): 219 --251

[RJR07] Rinderle, S.; Jurisch, M.; Reichert, M. (2007): On Deriving Net Change Information From Change Logs – The DELTALAYER-Algorithm. *Proc. 12th GI Conf. on Database Systems in Business, Technology and Web (BTW'07)*, Lecture Notes in Informatics P-103, Aachen, Germany, March 2007, pp. 364-381

[RRD04a] Rinderle, S.; Reichert, M.; Dadam, P. (2004): Flexible Support of Team Processes By Adaptive Workflow Systems. *Distributed and Parallel Databases*, 16(1):91-116

[RRD04b] Rinderle, S.; Reichert, M.; Dadam, P. (2004): Correctness Criteria For Dynamic Changes in Workflow Systems - A Survey. *Data and Knowledge Engineering*, 50(1):9-34

[RRD04c] Rinderle, S.; Reichert, M.; Dadam, P. (2004): Disjoint and Overlapping Process Changes - Challenges, Solutions, Applications. *Proc. 12th Int'l Conf. Cooperative Information Systems (CoopIS'04)*, Agia Napa, Cyprus, LNCS 3290, pp. 101-120.

[RRD09] Reichert, M.; Rinderle-Ma, S.; Dadam, P. (2009) Flexibility in Process-aware Information Systems. LNCS 5460, Springer, *LNCS Transactions on Petri Nets*

and Other Models of Concurrency (ToPNoC), Special Issue on Concurrency in Process-aware Information Systems, 2: 115-135

[RRH+09] Reichert, M.; Rechtenbach, S.; Hallerbach, A.; Bauer, T. (2009) Extending a Business Process Modeling Tool with Process Configuration Facilities: The Provop Demonstrator. In: CEUR proceedings of the BPM'09 Demonstration Track, Business Process Management Conference 2009 (BPM'09), September 2009, Ulm, Germany

[RRJ+06] Rinderle, S.; Reichert, M.; Jurisch, M.; Kreher, U. (2006): On Representing, Purging, and Utilizing Change Logs in Process Management Systems. *Proc. 4th Int'l Conf. Business Process Management (BPM'06)*, Vienna, Austria, LNCS 4102, pp. 241-256

[RRK+05] Reichert, M.; Rinderle, S.; Kreher, U.; Dadam, P. (2005): *Adaptive Process Management with ADEPT2*. Proc. 21st Int. Conf. Data Engineering (ICDE'05), Tokyo, 2005

[RRW08a] Rinderle-Ma, S.; Reichert, M.; Weber, B. (2008) Relaxed Compliance Notions in Adaptive Process Management Systems. *Proc. 27th Int'l Conf. on Conceptual Modeling (ER'08)*, October 2008, Barcelona, Spain. LNCS 5231, pp. 232-247

[RRW08b] Rinderle-Ma, S.; Reichert, M.; Weber, B. (2008) On the Formal Semantics of Change Patterns in Process-aware Information Systems. *Proc. 27th Int'l Conference on Conceptual Modeling (ER'08)*, October 2008, Barcelona, Spain. LNCS 5231, pp. 279-293

[RWR+05] Rinderle, S.; Weber, B.; Reichert, M; Wild, W. (2005): Integrating Process Learning and Process Evolution - A Semantics Based Approach. *Proc. 3rd Int'l Conf. Business Process Management (BPM'05)*, Nancy, France, Sept. 2005, LNCS 3649, pp. 252-267

[SaOr00] Sadiq, W.; Orlowska, M.E. (2000) Analysing Process Models using Graph Reduction Techniques. *Information Systems*, 25(2): 117-134.

[SSO05] Sadiq, S.; Sadiq, W.; Orlowska, M.E. (2005): A Framework for Constraint Specification and Validation in Flexible Workflows. Information Systems, 30(5).

[SWD+08] Schonenberg, H.; Weber, W.; van Dongen, B.; van der Aalst, W.M.P. (2008): Supporting Flexible Processes Through Log-Based Recommendations. *Proc. BPM'08*, Milan, Italy, pp. 51-66.

[WBB04] Wainer, J.; Bezerra, F.; Barthelmess, P. (2004): Tucupi: a flexible workflow system based on overridable constraints. In: *Proc. SAC '04*. pp. 498-502

[WeRe08] Weber, B.; Reichert, M. (2008) Refactoring Process Models in Large Process Repositories *Proc. CAiSE'08*, Montpellier, France, pp. 124-139.

[WeRR07] Weber, B., Reichert, M., Rinderle, S. (2007): Change Patterns and Change Support Features in Process-Aware Information Systems. Proc. 19th Int'l Conf. on

Advanced Inf. Systems Eng. (CAiSE'07), LNCS 4495, Trondheim, Norway, June 2007, pp. 574-588

[WeRR08] Weber, B.; Reichert, M.; Rinderle-Ma, S. (2008): Change Patterns and Change Support Features -Enhancing Flexibility in Process-Aware Information Systems. *Data and Knowledge Engineering* 66(3):. 438-466.

[WRR+05] Weber, B.; Reichert, M.; Rinderle, S.; Wild, W. (2005): Towards a Framework for the Agile Mining of Business Processes. *1st Int'l Workshop on Business Process Intelligence, Proc. BPM'05 Workshops*, LNCS 3812, Nancy, France, Sept. 2005, pp. 191-202

[WRW06] Weber, B.; Reichert, M.; Wild, W. (2006): Case-Base Maintenance for CCBR-Based Process Evolution. *Proc. ECCBR'06*, pp. 106-120

[WRWR05] Weber, B.; Reichert, M.; Wild, W.; Rinderle, S. (2005) Balancing Flexibility and Security in Adaptive Process Management Systems. In: *Proc. 13th Int'l Conf. on Cooperative Information Systems (CooplS '05)*, Agia Napa, Cyprus. Springer, LNCS 3760, pp. 59-76

[WRWR09] B. Weber, M. Reichert, W. Wild and S. Rinderle-Ma: Providing Integrated Life Cycle Support in Process-Aware Information Systems**.** *International Journal of Cooperative Information Systems* 18(1):115-165.

[WWB04] Weber, B.; Wild, W.; Breu, R. (2004): CBRFlow: Enabling Adaptive Workflow Management through Conversational Case-Based Reasoning. *Proc. ECCBR 2004*, Madrid, Spain, pp. 434-448.

[WRZW09] B. Weber and H. Reijers and S. Zugal and W. Wild (2009) The Declarative Approach to Business Process Execution: An Empirical Test**.** *Proc. CAiSE'09*, pp. 470-485.

[WSR09] Weber, B.; Sadiq, S.; Reichert, M. (2009) Beyond Rigidity - Dynamic Process Lifecycle Support: A Survey on Dynamic Changes in Process-aware Information Systems. Springer, *Computer Science - Research and Development*, 23(2), pp. 47-65

[WZP+09] Weber, B.; Zugal, S.; Pinggera, J.; Wild, W. (2009). Experiencing Process Flexibility Patterns with Alaska Simulator. *Proc. BPM'09 Demos*.