



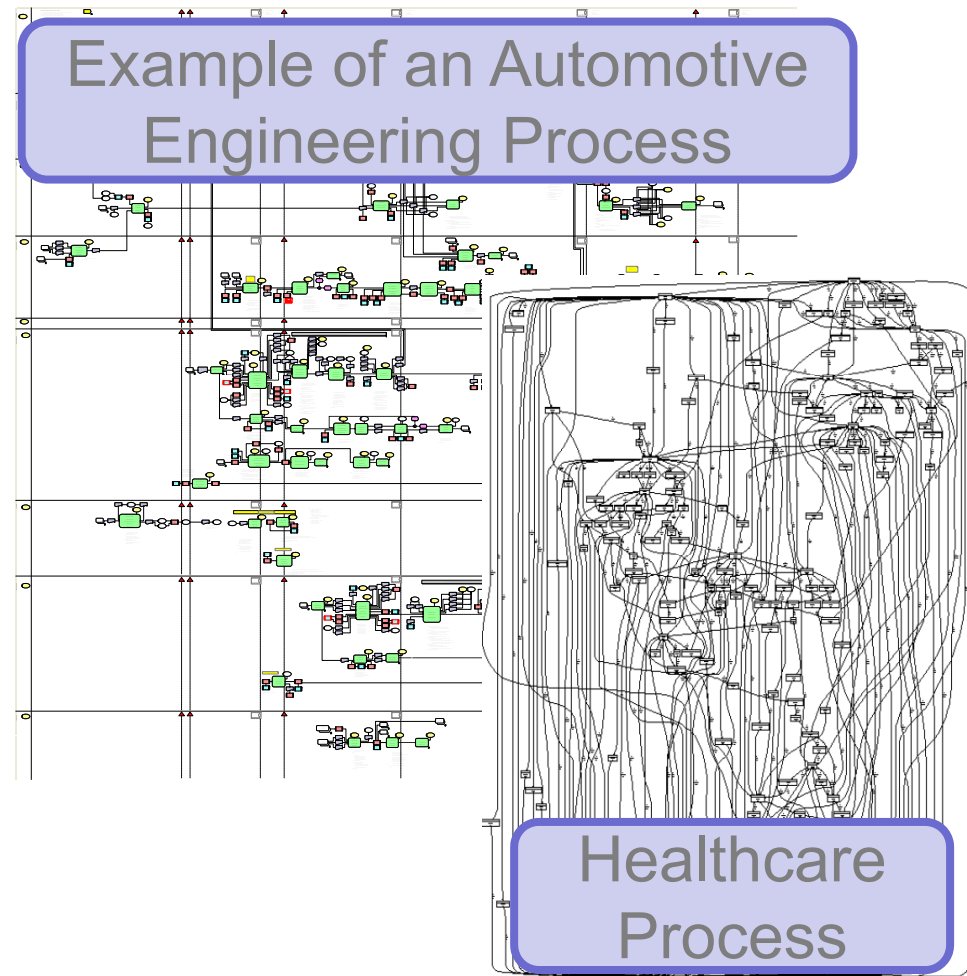
# Fostering Reuse in the Business Process Lifecycle

- Challenges, Methods, Technologies -

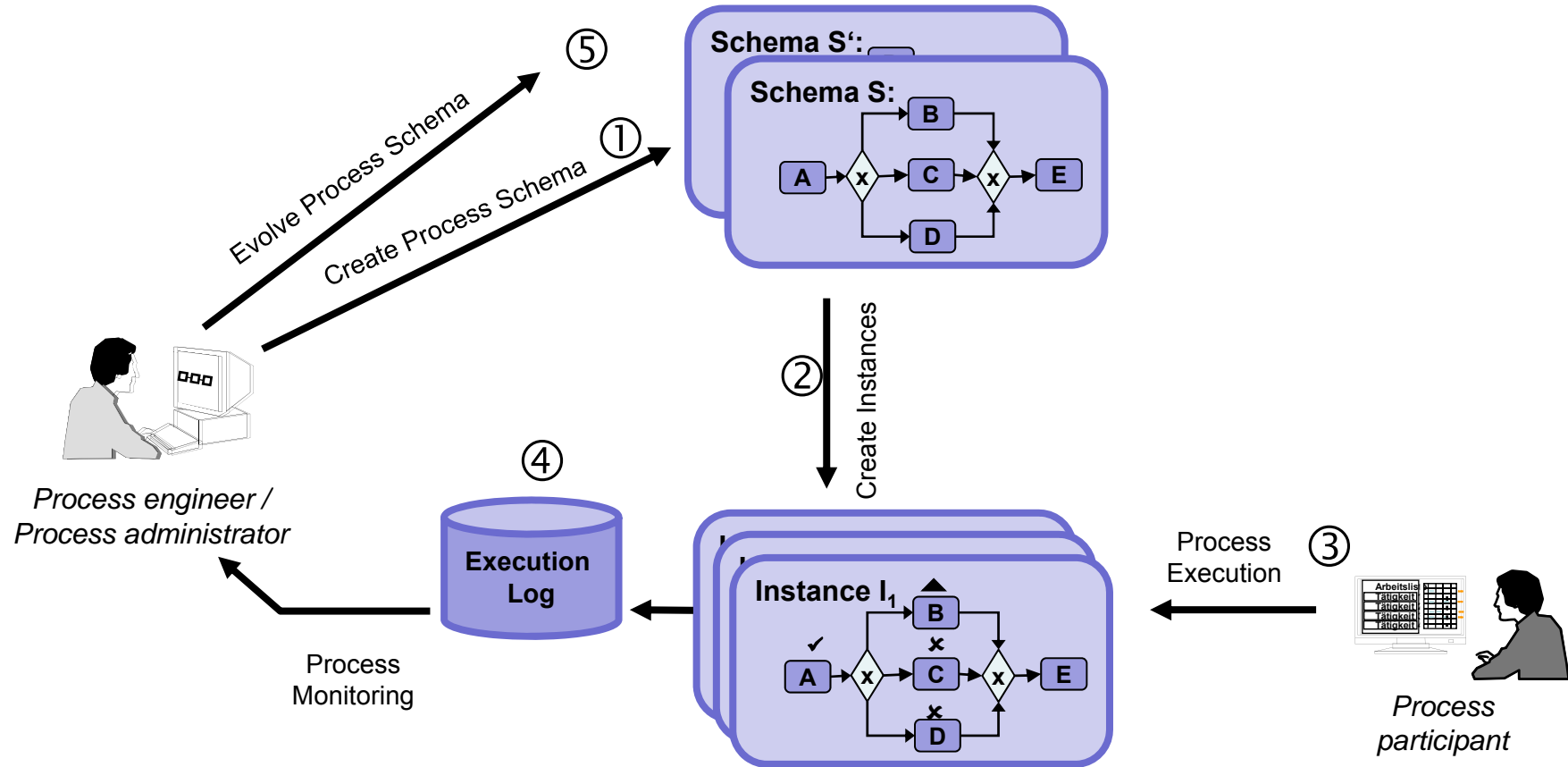
**Manfred Reichert**

## Motivation

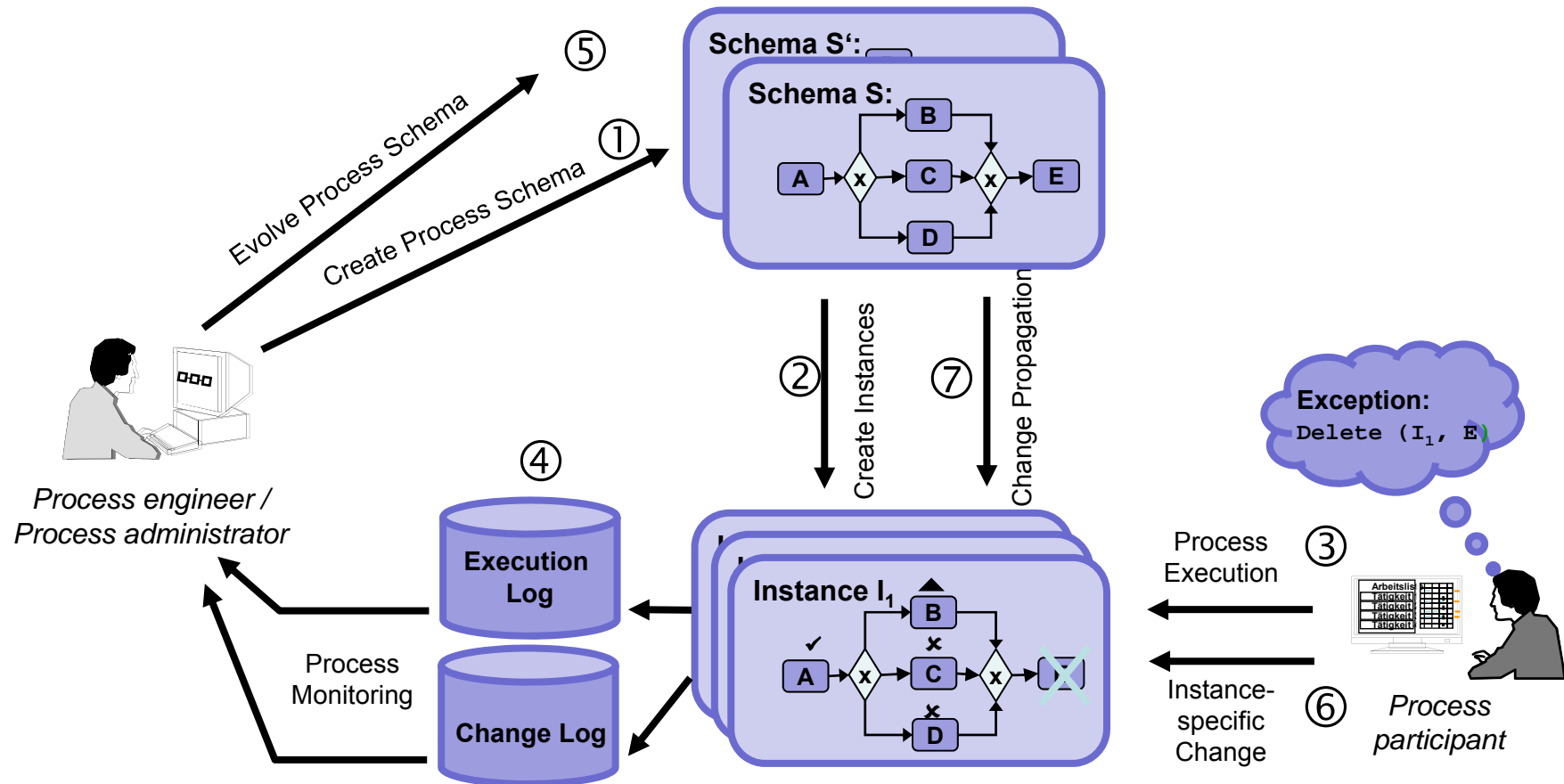
- ❑ Processes can become very large and complex
- ❑ Thousands of concurrently executed process instances
- ❑ High need for flexibility and adaptability in all phases of the process lifecycle
- ❑ PAIS correctness and PAIS robustness are fundamental
- ❑ Reuse of process artifacts is crucial along the whole process lifecycle



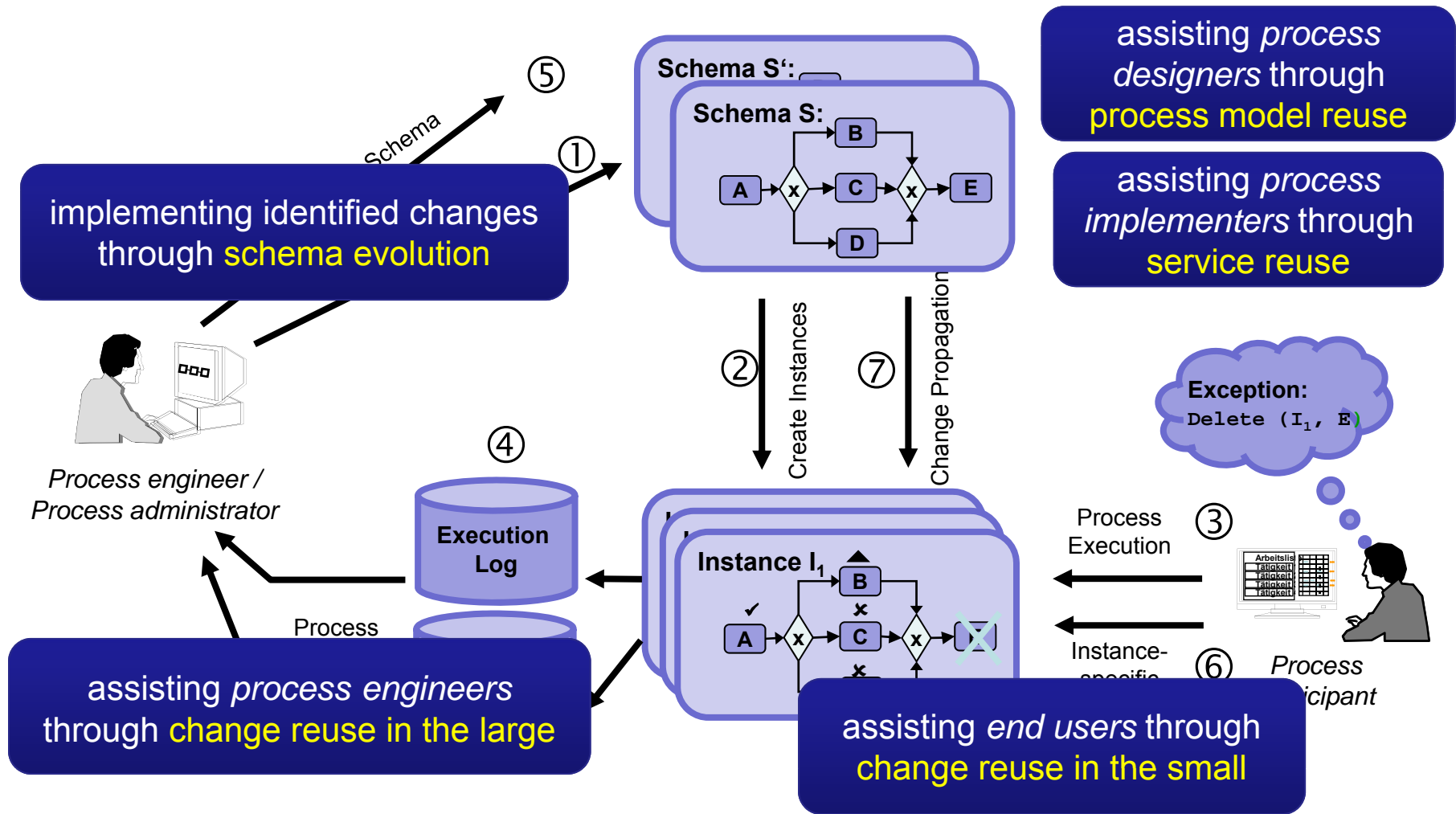
# Lifecycle Support for Dynamic Processes



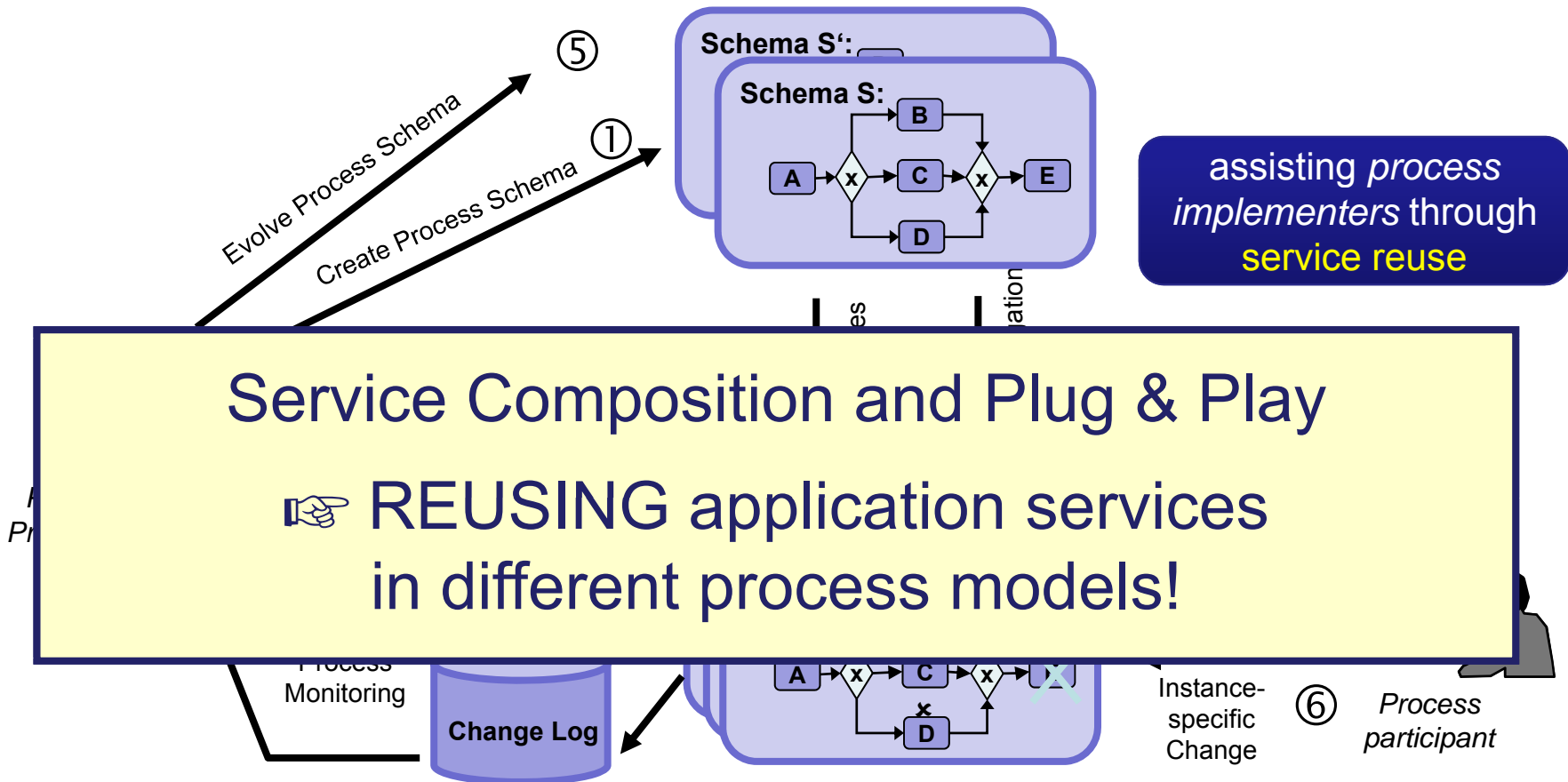
# Lifecycle Support for Dynamic Processes



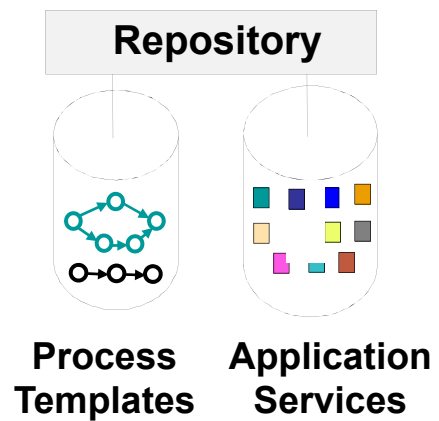
# Lifecycle Support for Dynamic Processes



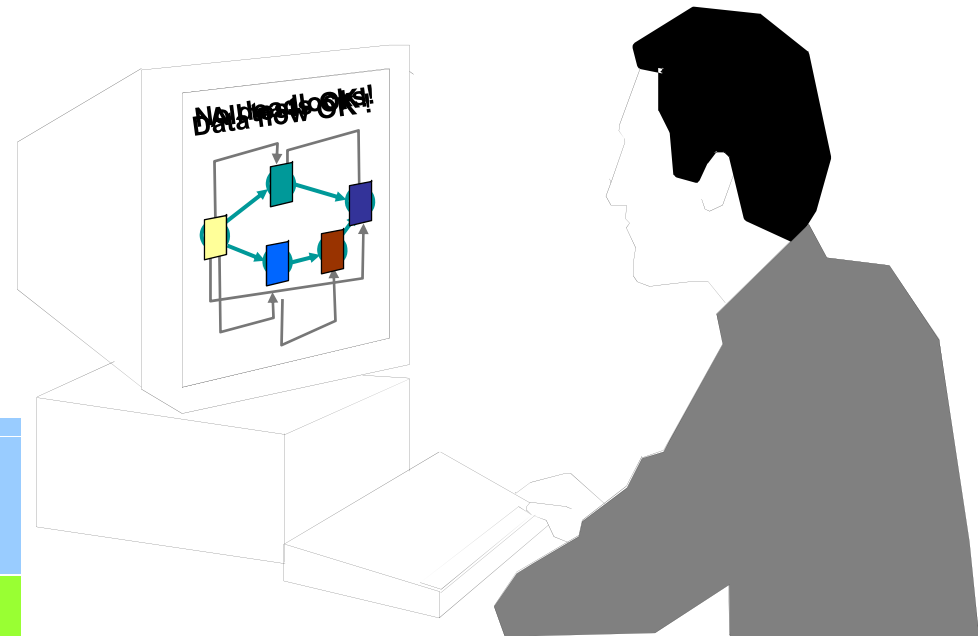
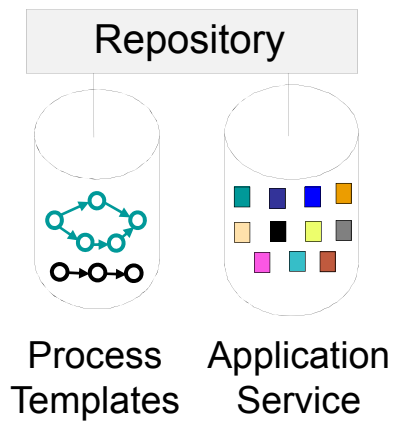
# Lifecycle Support for Dynamic Processes



# Reusing Application Services in a „Plug & Play“-like Style



## Reusing Application Services in a „Plug & Play“-like Style

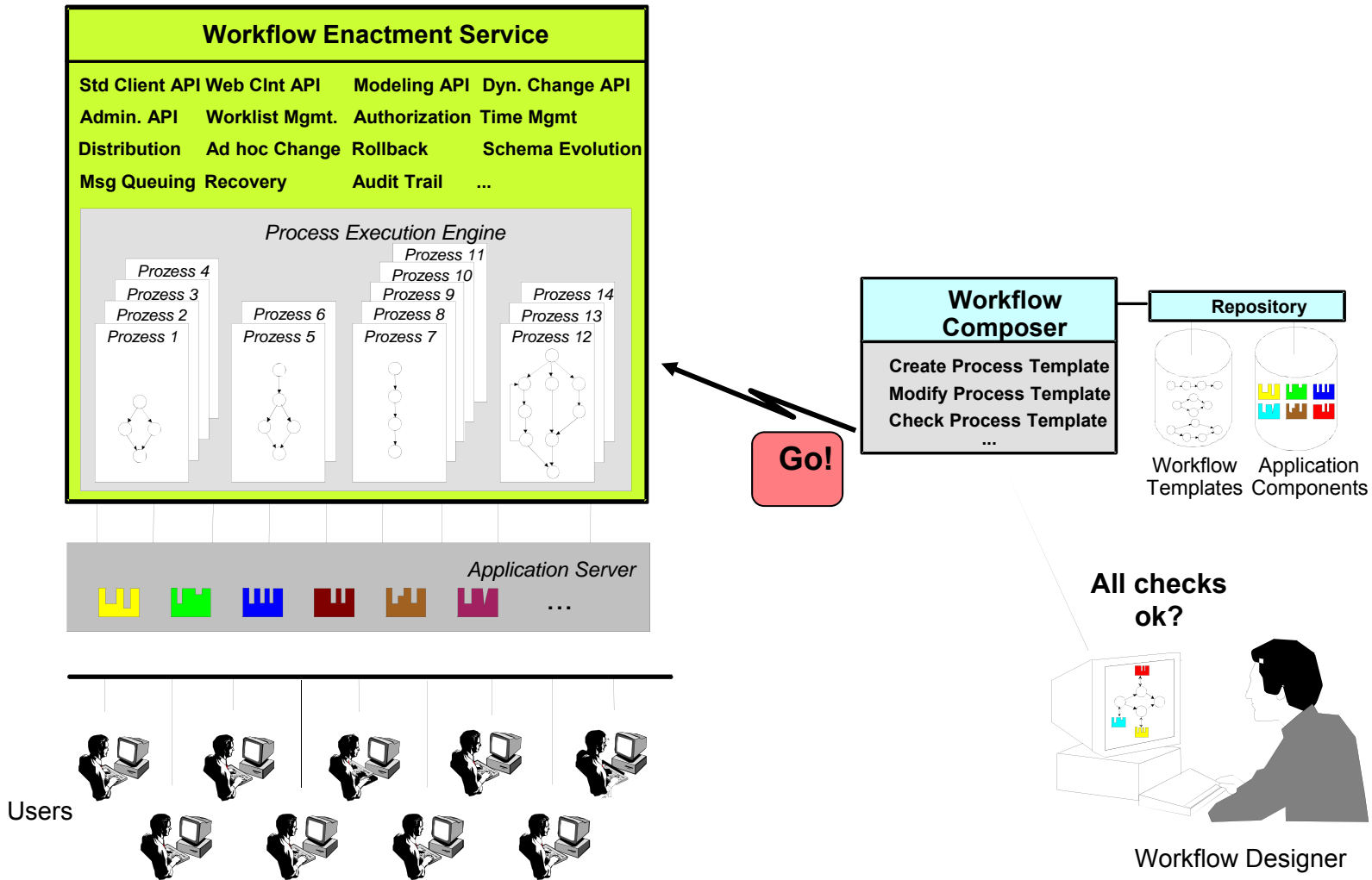


**Composing process-oriented applications out of process templates and application services**

**with system-enabled consistency checks**

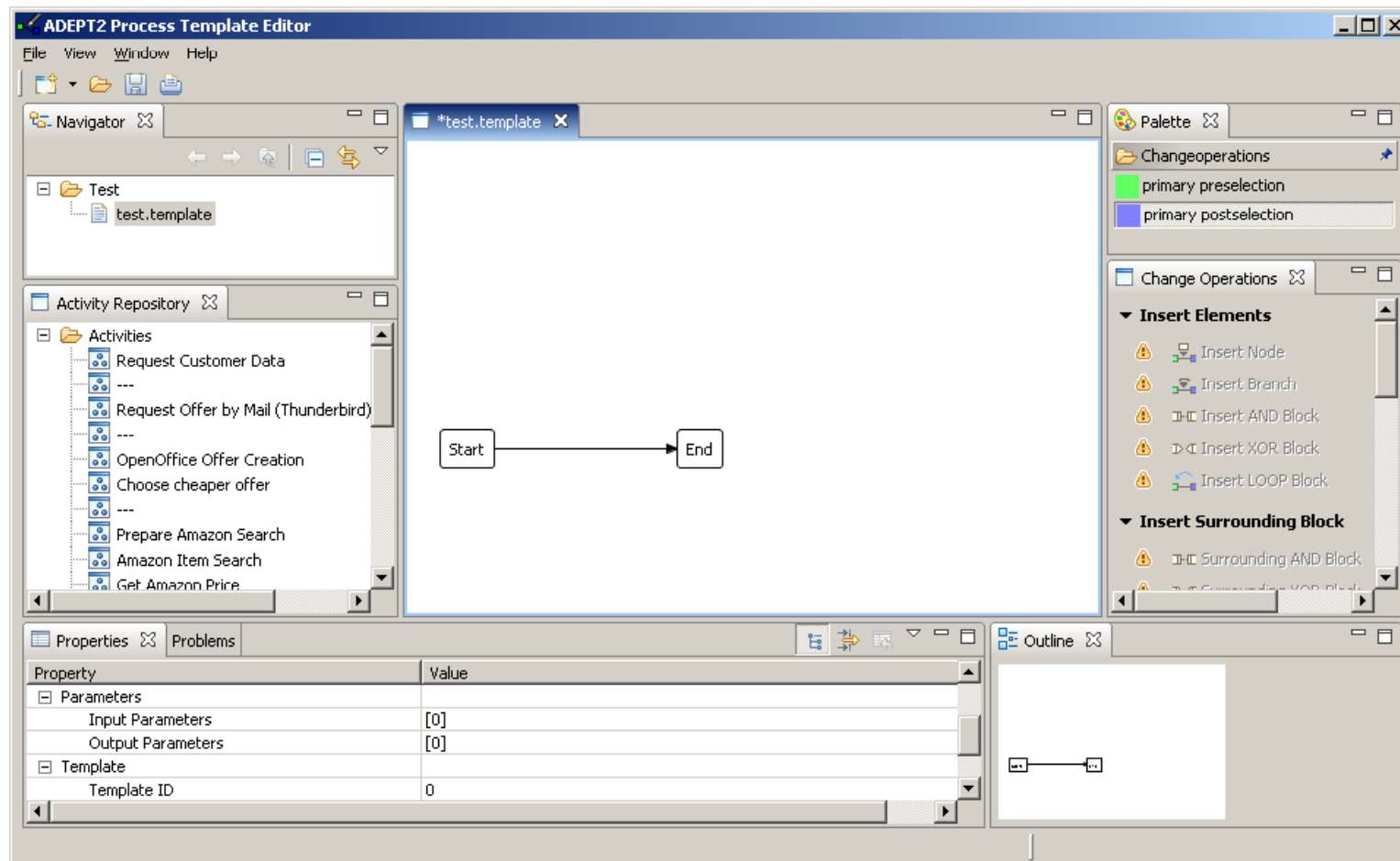


# Reusing Application Services in a „Plug & Play“-like Style



# Reusing Application Services in a „Plug & Play“-like Style: The Adept Approach

- **“Classical” construction: Context-sensitive list of legal operations**



# Reusing Application Services in a „Plug & Play“-like Style: The Adept Approach

- **“Classical” construction: Context-sensitive list of legal operations**

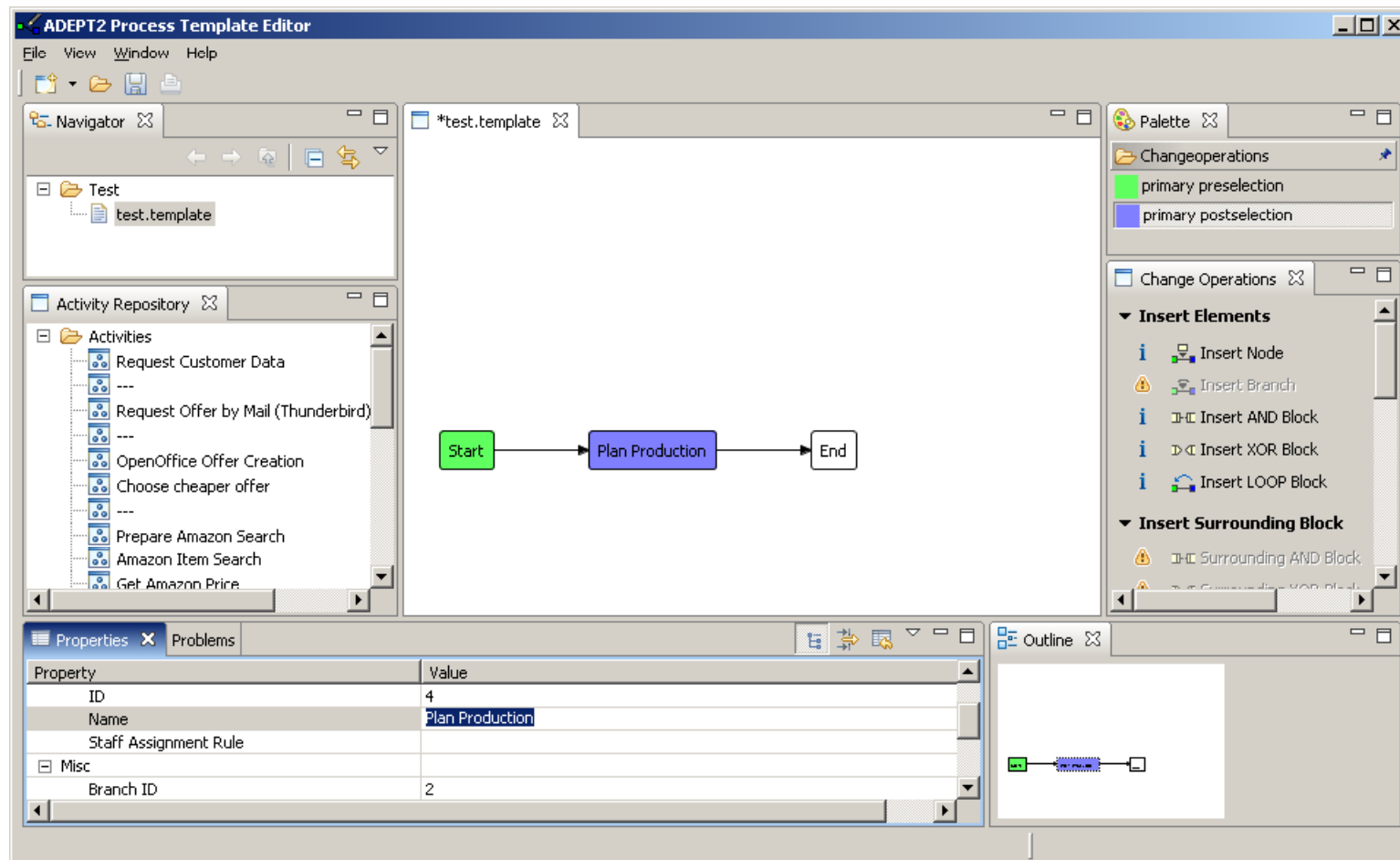
The screenshot displays the ADEPT2 Process Template Editor interface. The main workspace shows a simple workflow diagram with a green 'Start' node connected to a blue 'End' node. The interface includes several panels:

- Navigator:** Shows a folder named 'Test' containing a file 'test.template'.
- Activity Repository:** Lists various activities such as 'Request Customer Data', 'Request Offer by Mail (Thunderbird)', 'OpenOffice Offer Creation', 'Choose cheaper offer', 'Prepare Amazon Search', 'Amazon Item Search', and 'Get Amazon Price'.
- Palette:** A context-sensitive list of legal operations. It includes sections for 'Changeoperations', 'Change Operations', 'Insert Elements', and 'Insert Surrounding Block'. The 'Insert Elements' section lists options like 'Insert Node', 'Insert Branch', 'Insert AND Block', 'Insert XOR Block', and 'Insert LOOP Block'. The 'Insert Surrounding Block' section lists options like 'Surrounding AND Block' and 'Surrounding XOR Block'.
- Properties:** A table showing the properties of the selected 'End' node.
- Outline:** A small preview of the current workflow diagram.

Property	Value
ID	1
Name	End
Staff Assignment Rule	
Misc	
Branch ID	0

# Reusing Application Services in a „Plug & Play“-like Style: The Adept Approach

## □ “Classical” construction: Insertion of nodes ...



# Reusing Application Services in a „Plug & Play“-like Style: The Adept Approach

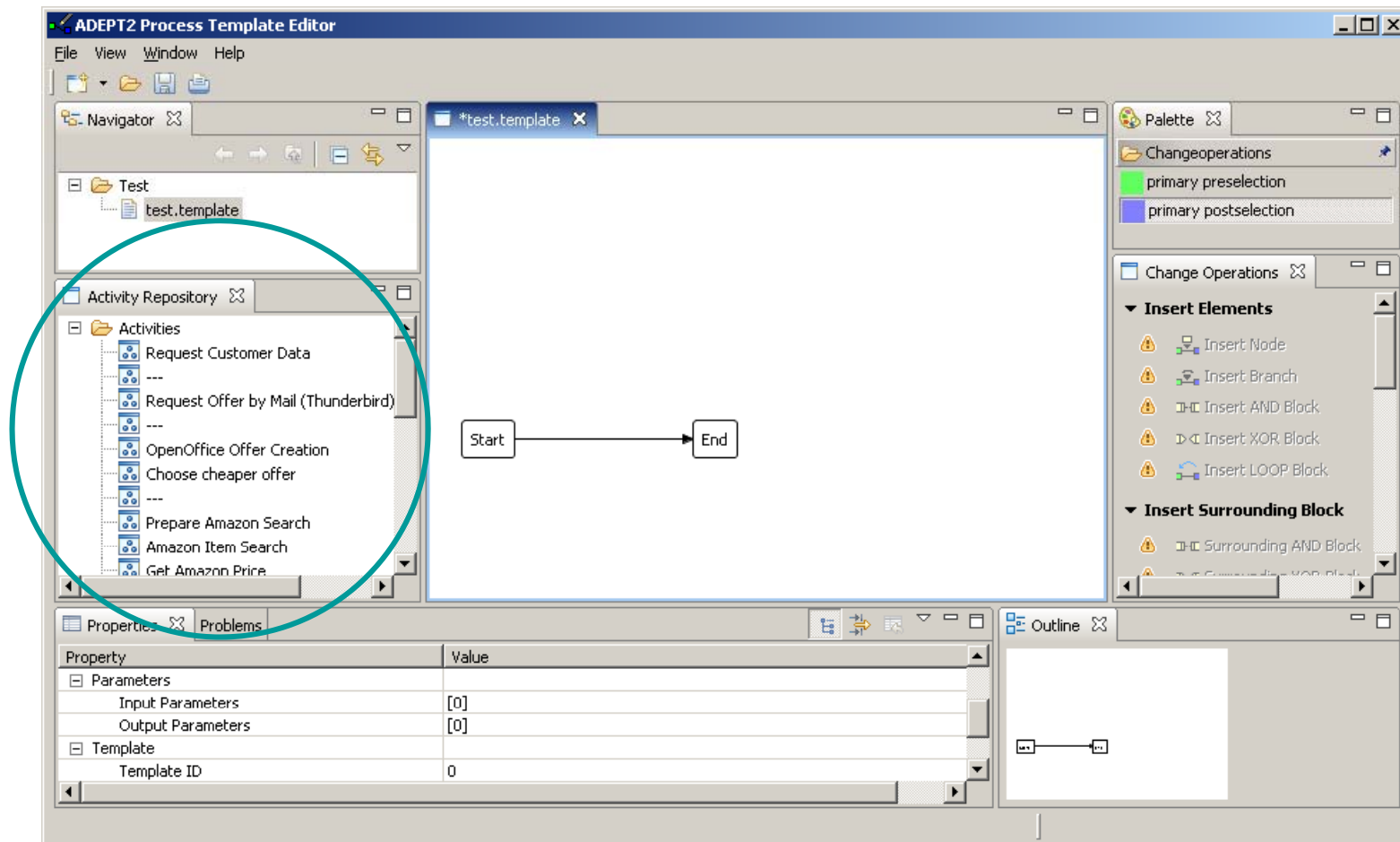
- ❑ **“Classical” construction: Insertion of data elements ...** with continuous correctness checks

The screenshot displays the ADEPT2 Process Template Editor interface. The main workspace shows a workflow diagram with a central activity node labeled 'Plan Production' (highlighted in blue) between 'Start' and 'End' nodes. A data element 'Customer OrderID STRING' is being inserted into the 'Plan Production' node, indicated by a dashed arrow. The 'Properties' window at the bottom left shows an error message: 'Data element 'Customer OrderID' is not supported by the activity 'Plan Production''. The 'Palette' on the right lists various operations, including 'Move Nodes' and 'Data Manipulation' (Insert, Read, Write, Remove).

Description	Resource	Path	Location
<b>Errors (1 item)</b>			
✘ Data element 'Customer OrderID' is not supported by the activity 'Plan Production'	test.template	Test	Node:4

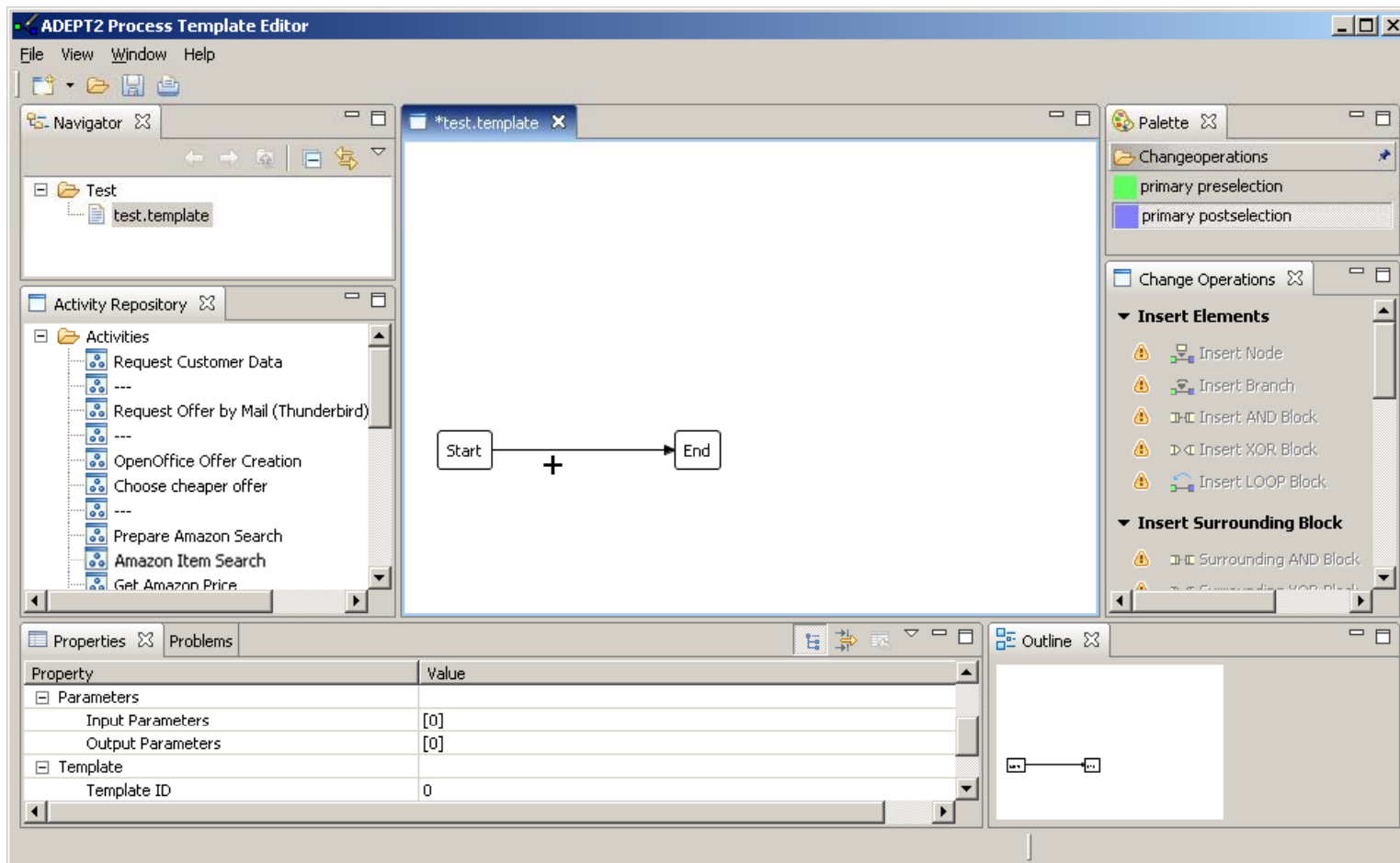
# Reusing Application Services in a „Plug & Play“-like Style: The Adept Approach

- Construction of processes in plug & play fashion



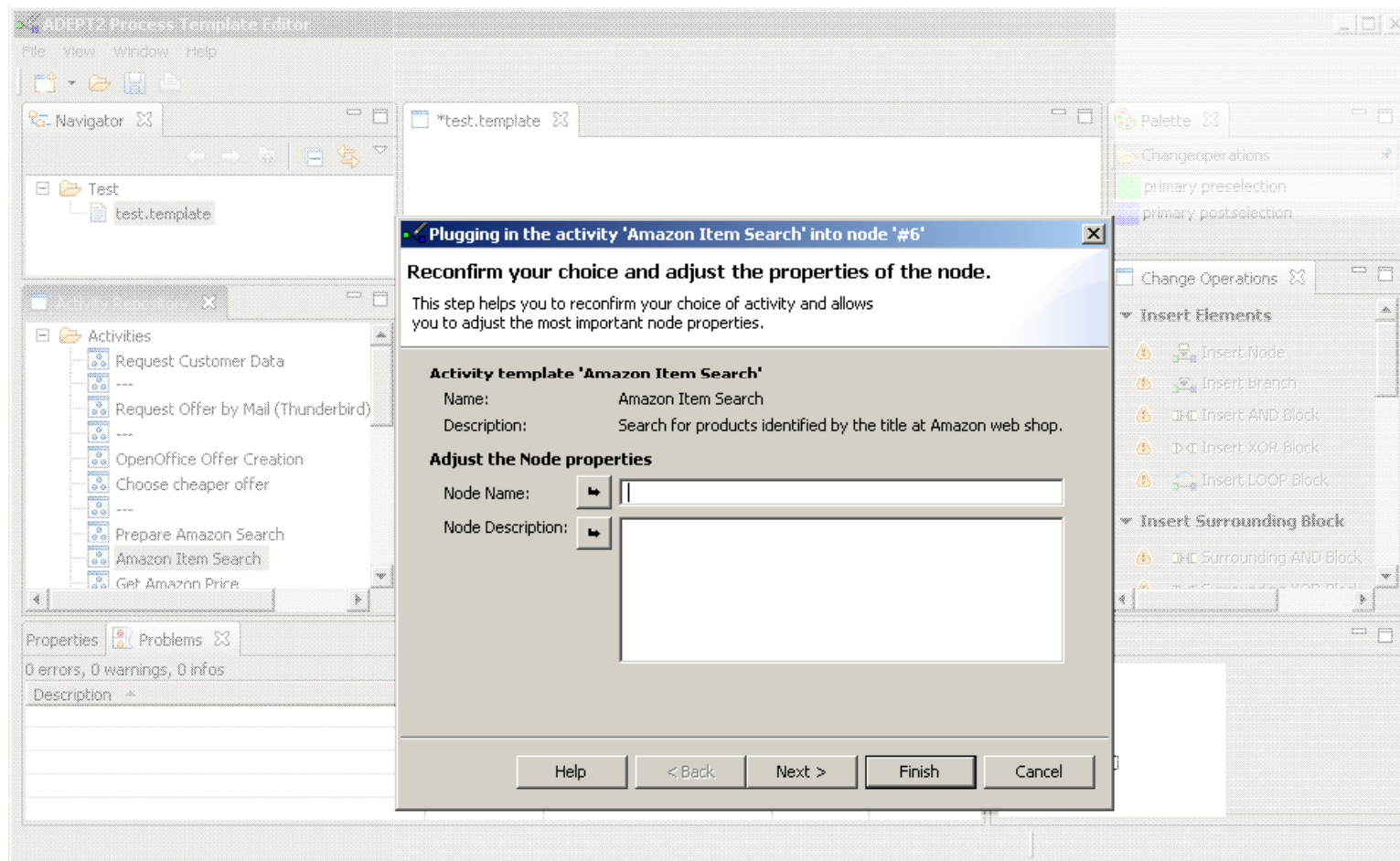
# Reusing Application Services in a „Plug & Play“-like Style: The Adept Approach

- Construction of processes in plug & play fashion



# Reusing Application Services in a „Plug & Play“-like Style: The Adept Approach

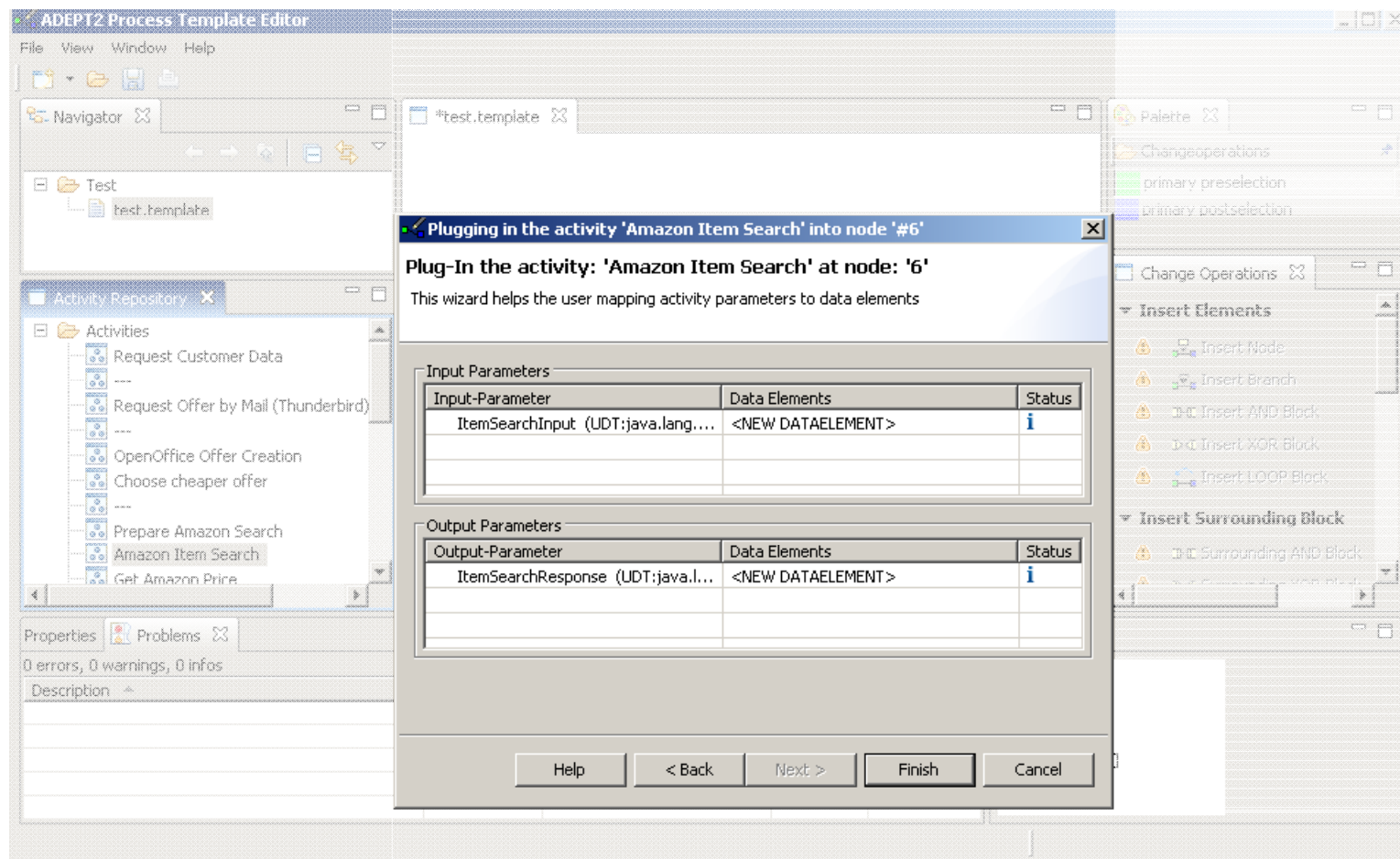
- Construction of processes by plug & play fashion: A wizard is guiding ...





# Reusing Application Services in a „Plug & Play“-like Style: The Adept Approach

- ❑ Construction of processes by plug & play fashion: A wizard is guiding ...



# Reusing Application Services in a „Plug & Play“-like Style: The Adept Approach

- Construction of processes by plug & play fashion

with continuous correctness checks

The screenshot displays the ADEPT2 Process Template Editor interface. The main workspace shows a process flow starting with 'Start', followed by an activity named 'Amazon Item Search' (marked with a red 'X'), and ending with 'End'. Above the activity, two data elements are defined: 'ItemSearchInput' and 'ItemSearchResponse', both with the user-defined type 'UDT: java.lang.Object'. The 'Amazon Item Search' activity has a red error icon next to it. The 'Properties' window at the bottom shows a table with one error item:

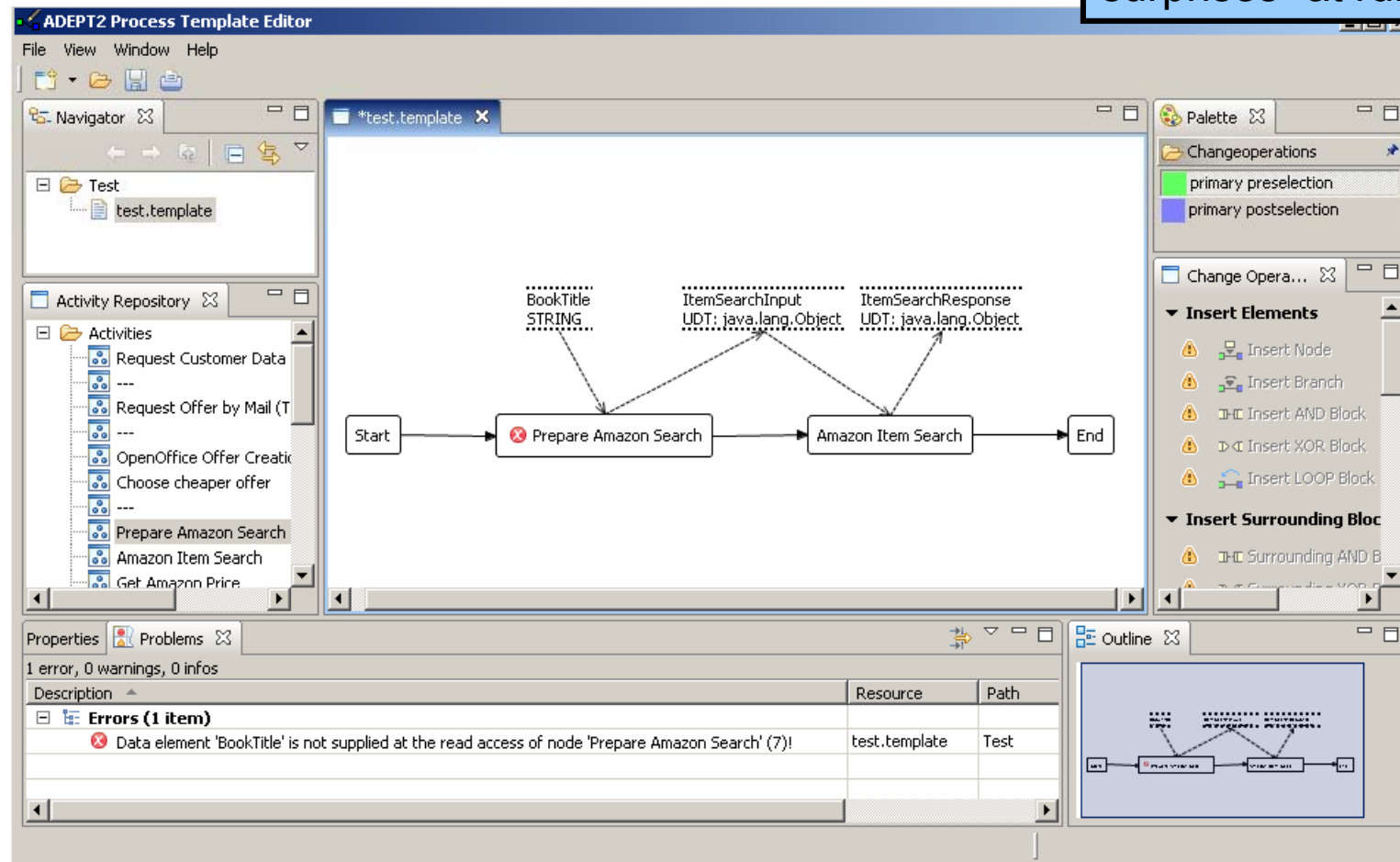
Description	Resource	Path
Errors (1 item)		
✖ Data element 'ItemSearchInput' is not supplied at the read access of node 'Amazon Item Search' (6)!	test.template	Test

The 'Activity Repository' on the left lists various activities, including 'Request Customer Data', 'Request Offer by Mail (Thunderbird)', 'OpenOffice Offer Creation', 'Choose cheaper offer', 'Prepare Amazon Search', 'Amazon Item Search', and 'Get Amazon Price'. The 'Palette' on the right contains options for 'Change Operations' and 'Insert Elements' (such as Insert Node, Insert Branch, Insert AND Block, Insert XOR Block, Insert LOOP Block) and 'Insert Surrounding Block' (such as Surrounding AND Block).

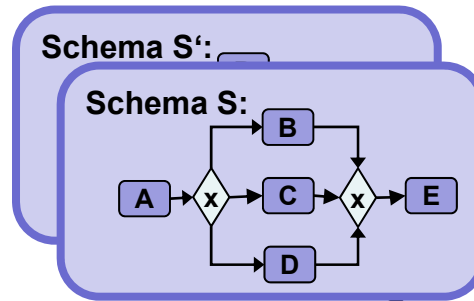
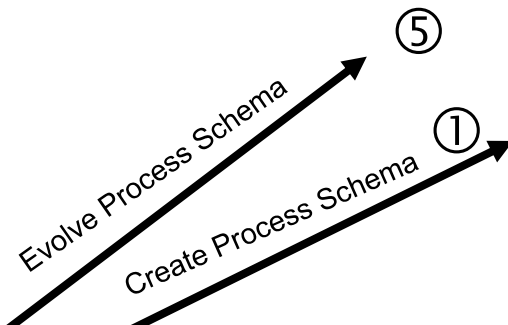
# Reusing Application Services in a „Plug & Play“-like Style: The Adept Approach

- ❑ Construction of processes by plug & play fashion

Goal: No “bad surprises” at run-time!

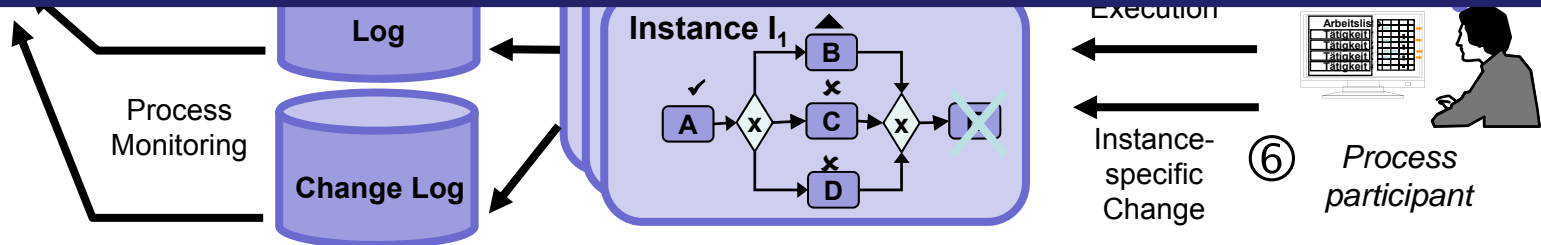


# Introduction: Lifecycle Support for Dynamic Processes



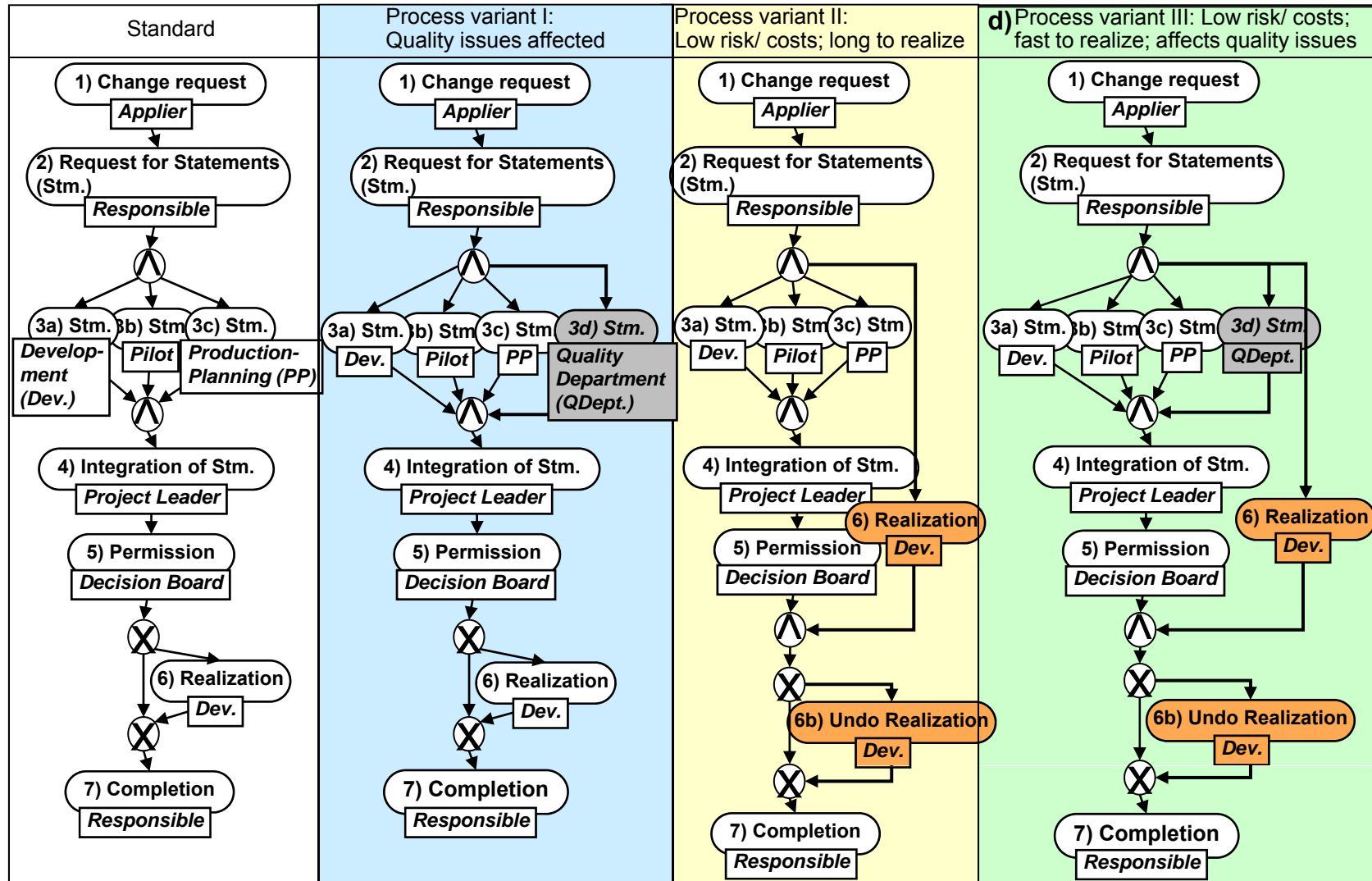
assisting *process designers* through **process model reuse**

Reference Models and Process Configuration  
 REUSING PROCESS MODELS in different context!

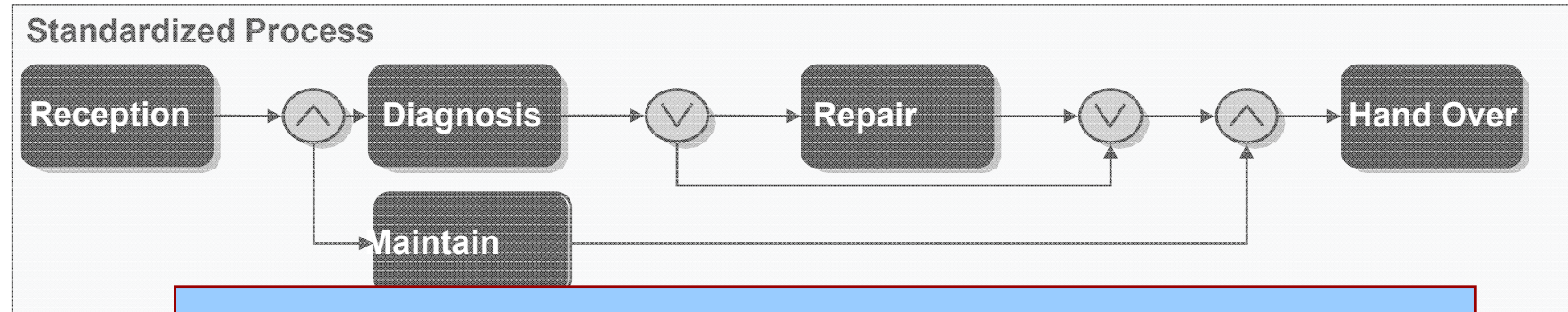


# Reuse Process Models through Configuration: Motivation

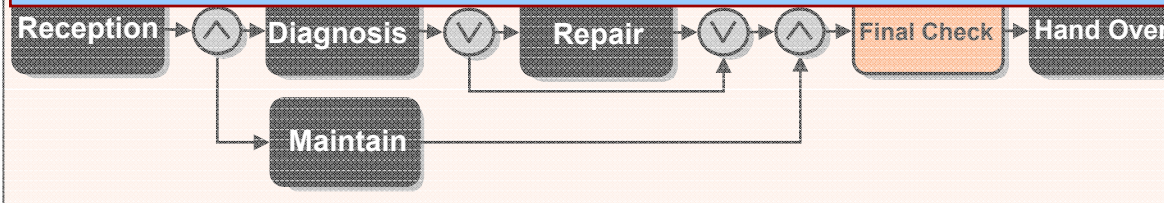
Example: Change Management



## Reuse Process Models through Configuration: Motivation



Reuse of a particular process model requires adaptation to the respective context  
 📌 Large process variant collections!



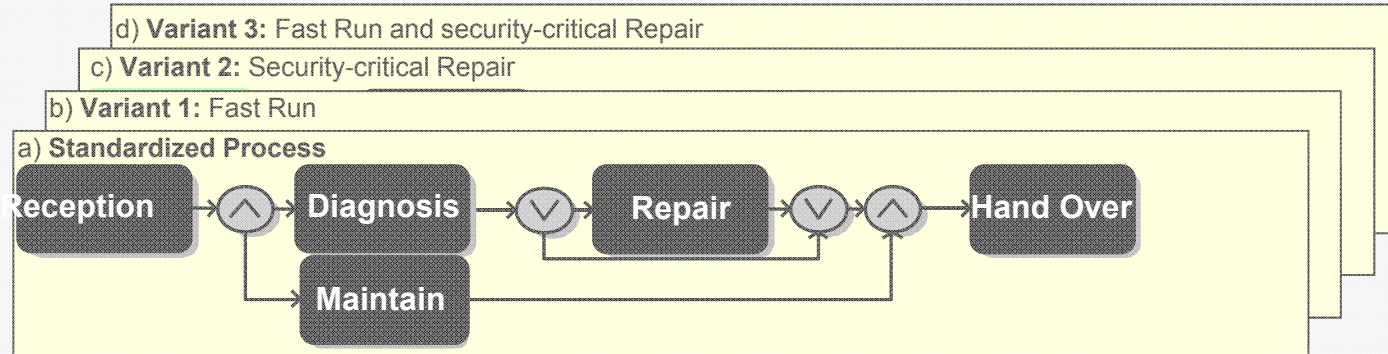
Variant 3:  
Fast Run and  
Security Critical  
Repair

Example: Vehicle Repair

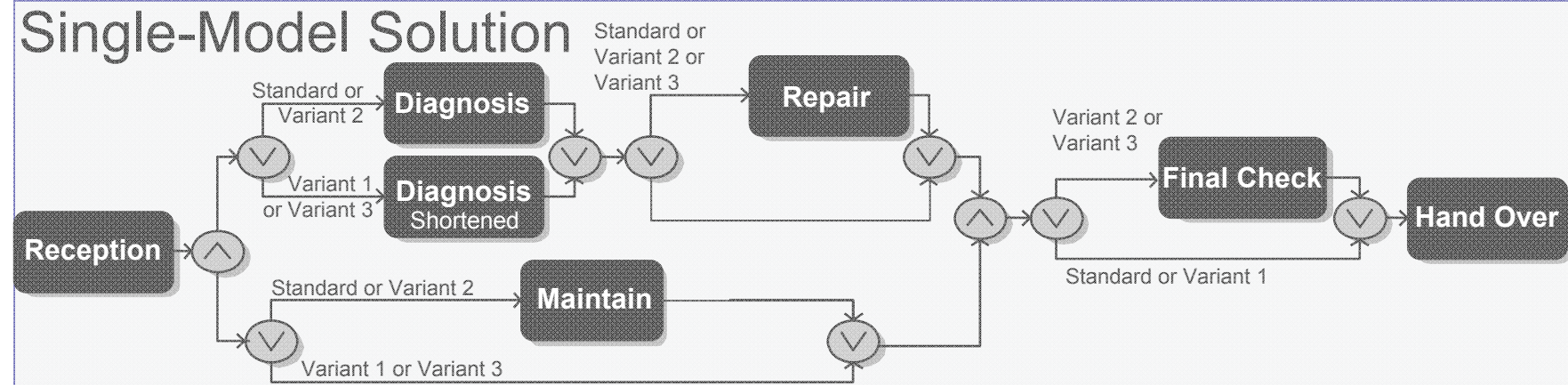


## Reuse Process Models through Configuration: Configuring Variants in Existing BPM Tools

### Multi-Model Solution



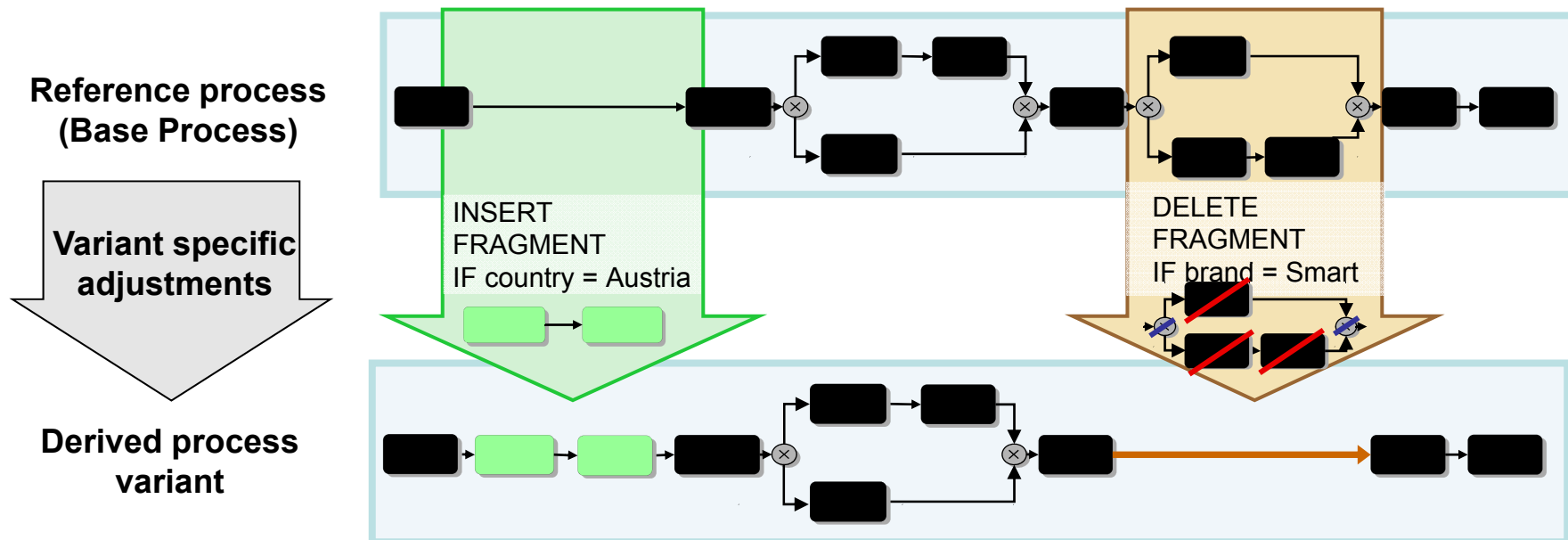
### Single-Model Solution



# Reuse Process Models through Configuration: The Provop Approach

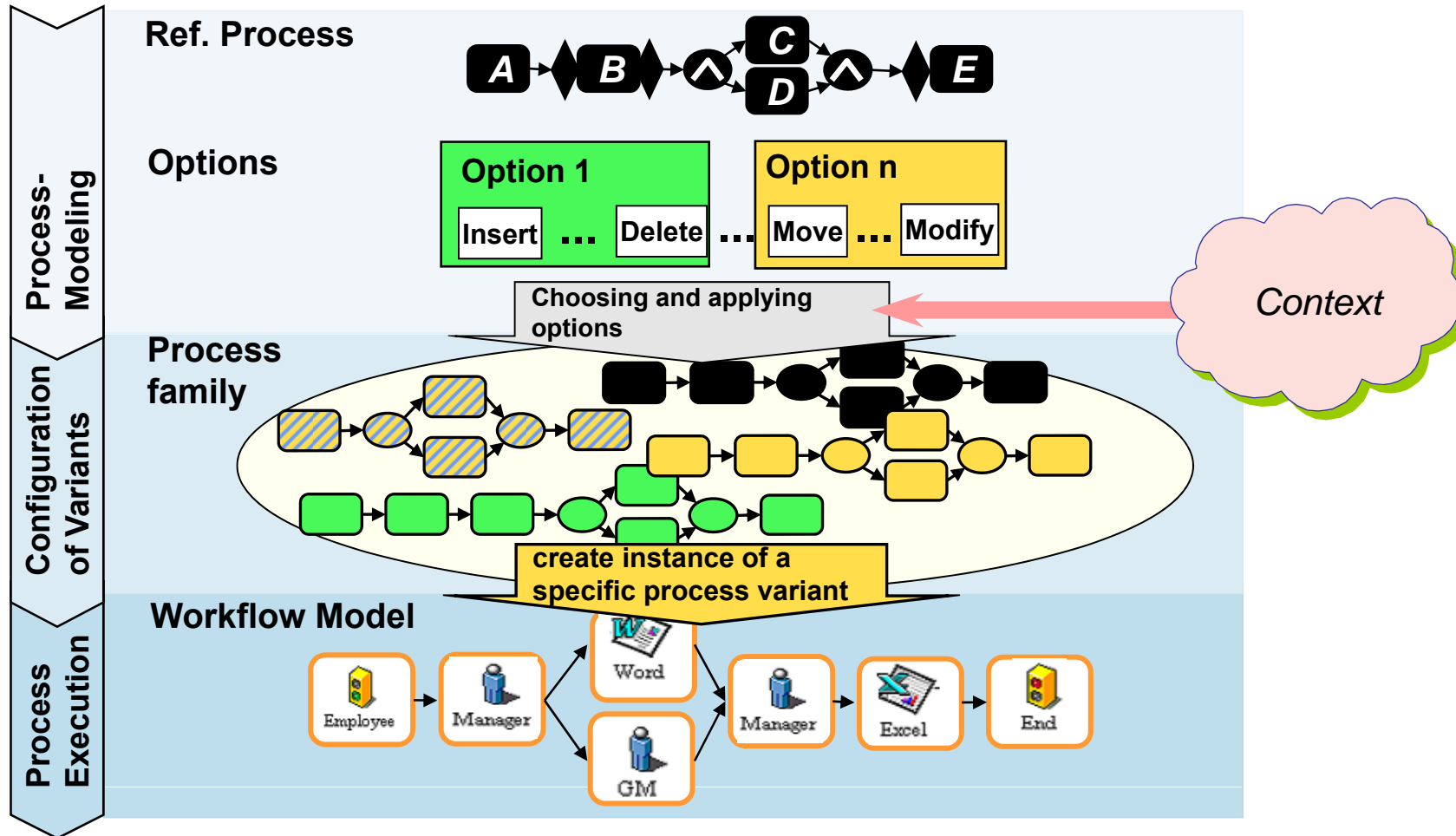
## *General observation:*

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





# Reuse Process Models through Configuration: The Provop Approach (Basic Elements)

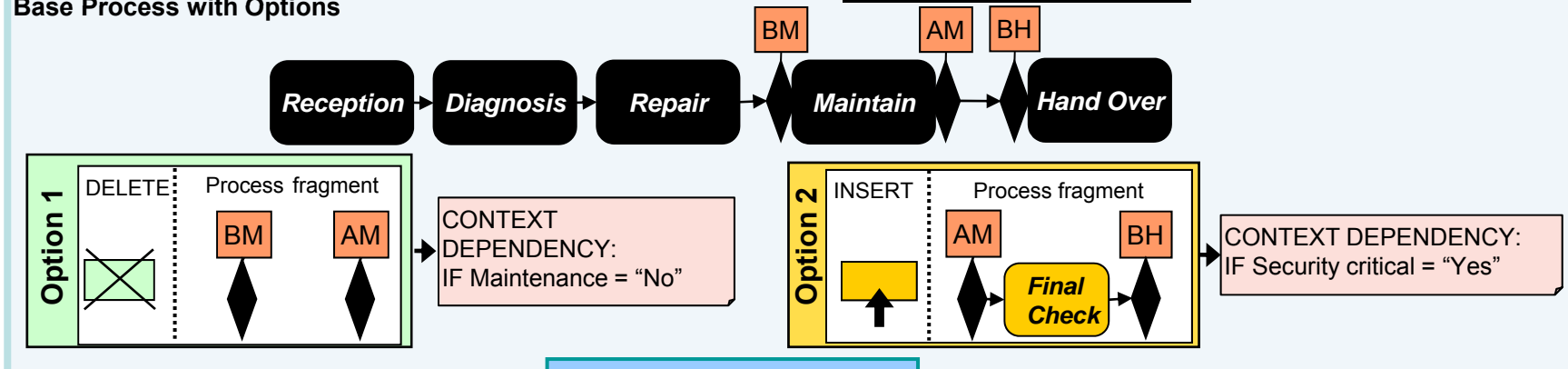


# The Provop Approach: Context-aware selection of options!

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

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

## Base Process with Options



selecting and applying change options

## Process Family

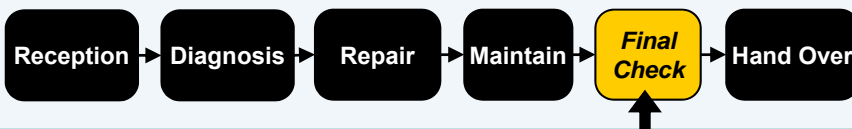
Variant 1:  
(Option 1)

CURRENT CONTEXT:  
Maintenance = „No”  
Security critical = „No”



Variant 2:  
(Option 2)

CURRENT CONTEXT:  
Maintenance = „Yes”  
Security critical = „Yes”



Variant 3:  
(base process)

CURRENT CONTEXT:  
Maintenance = „Yes”  
Security critical = „No”

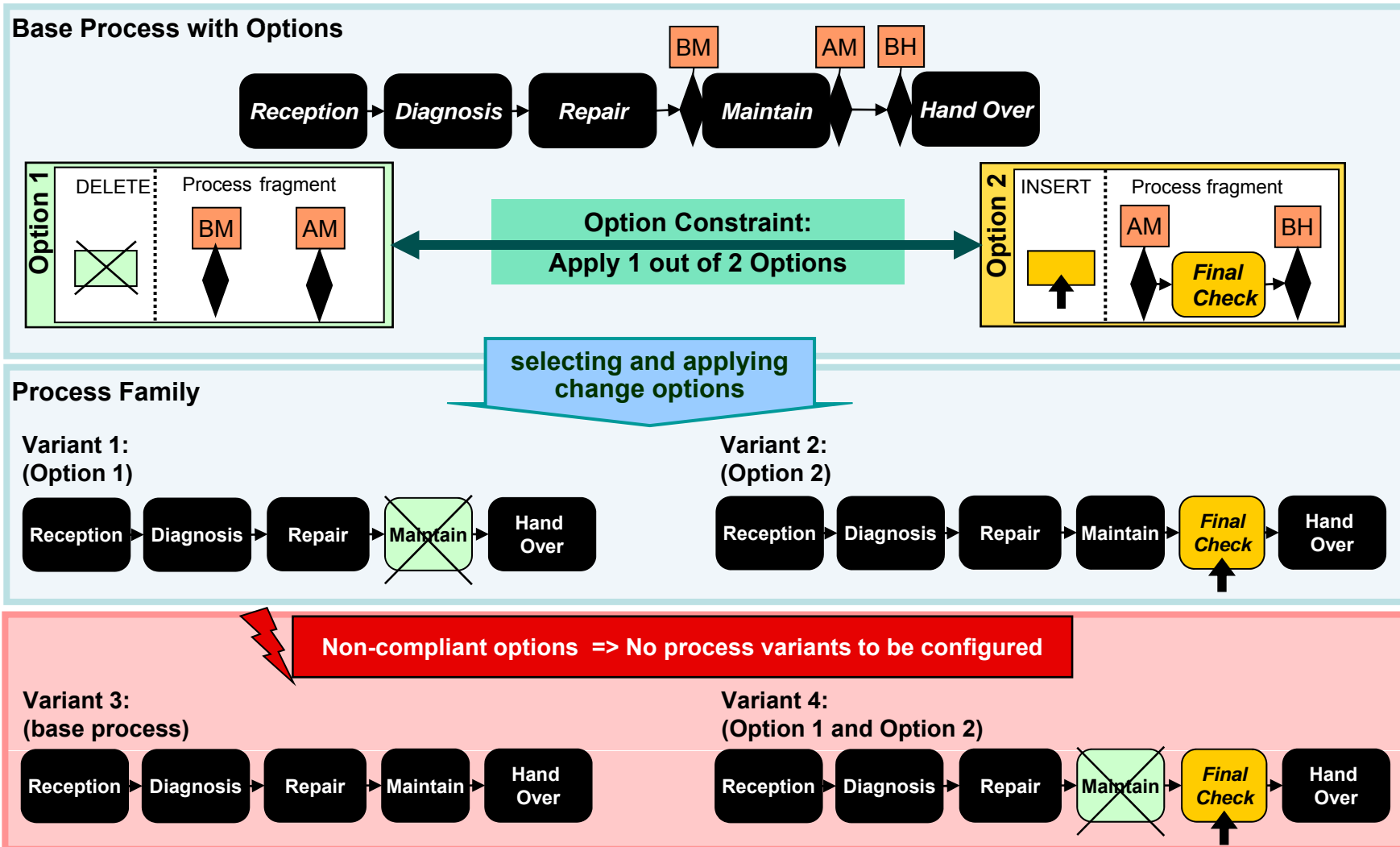


Variant 4:  
(Option 1,  
Option 2)

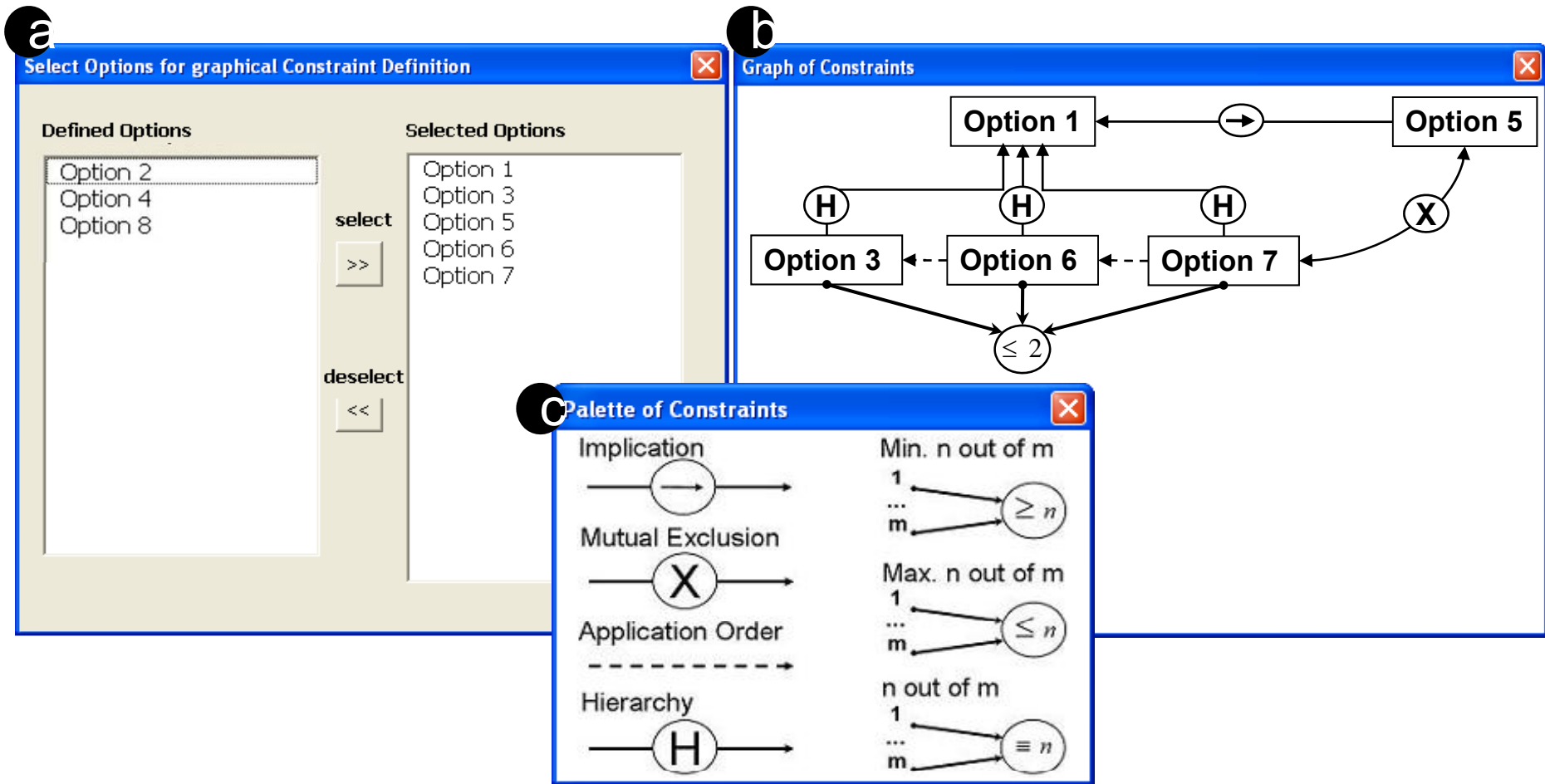
CURRENT CONTEXT:  
Maintenance = „No”  
Security critical = „Yes”

Invalid context description  
=> no process variant required

# Reuse Process Models through Configuration: The Provop Approach (Constraining the Use of Options)



# Reuse Process Models through Configuration: The Provop Approach (Constraining the Use of Options)



## Reuse Process Models through Configuration: The Provop Approach (Guaranteeing Soundness)

### **Issues:**

- 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

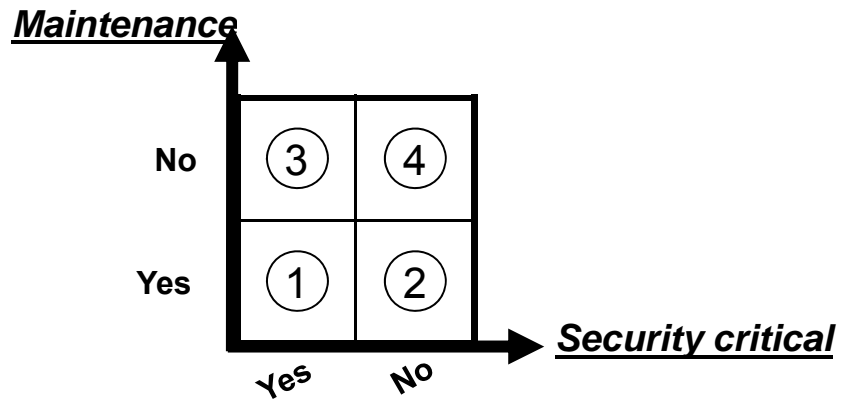
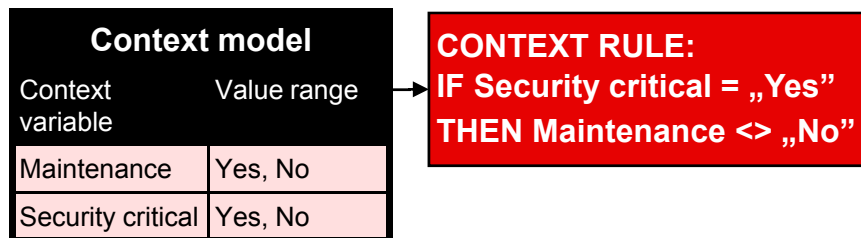
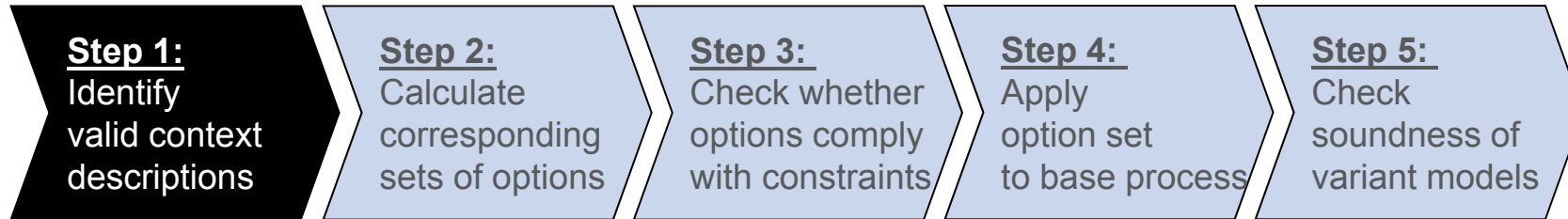
#### **Step 4:**

Apply  
option set  
to base process

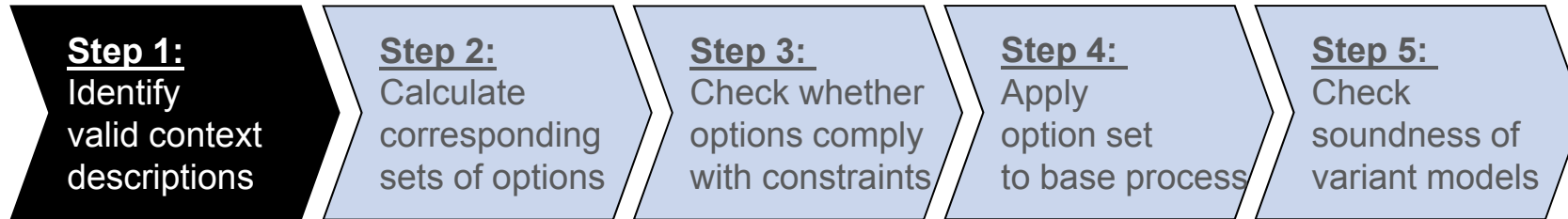
#### **Step 5:**

Check  
soundness of  
variant models

## Reuse Process Models through Configuration: The Provop Approach (Guaranteeing Soundness)

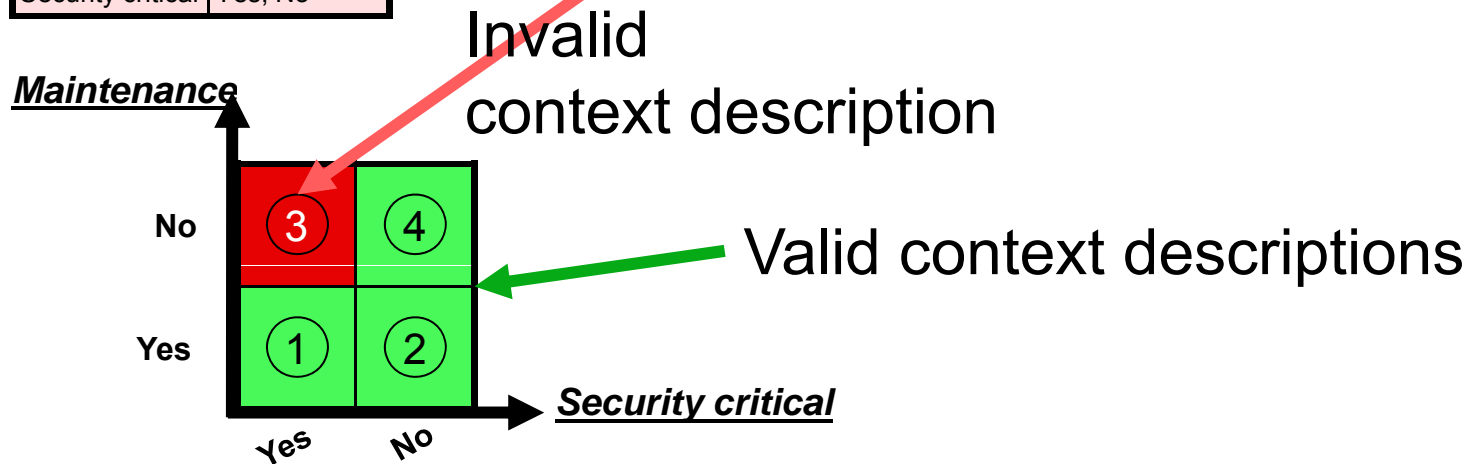


## Reuse Process Models through Configuration: The Provop Approach (Guaranteeing Soundness)

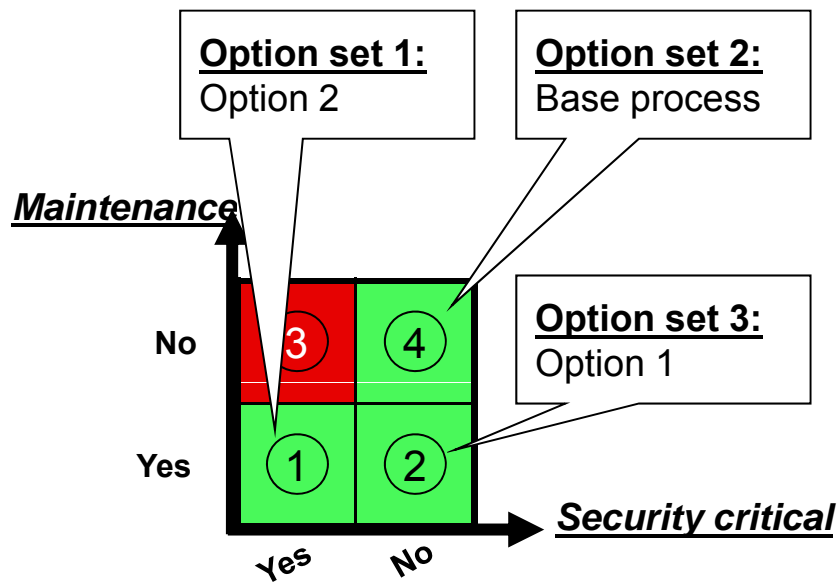
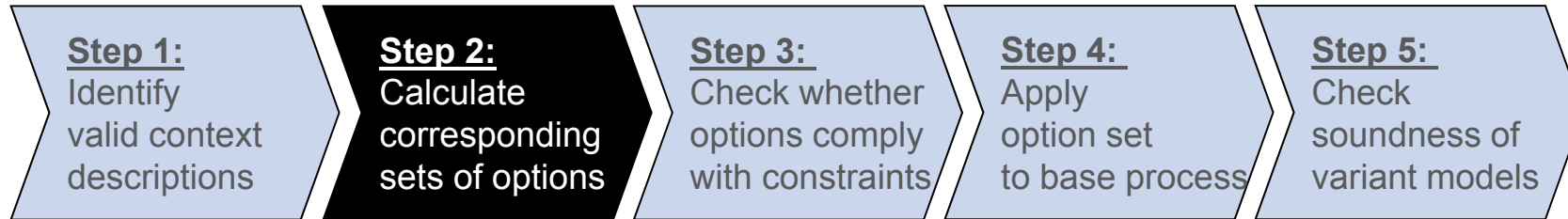


Context model	
Context variable	Value range
Maintenance	Yes, No
Security critical	Yes, No

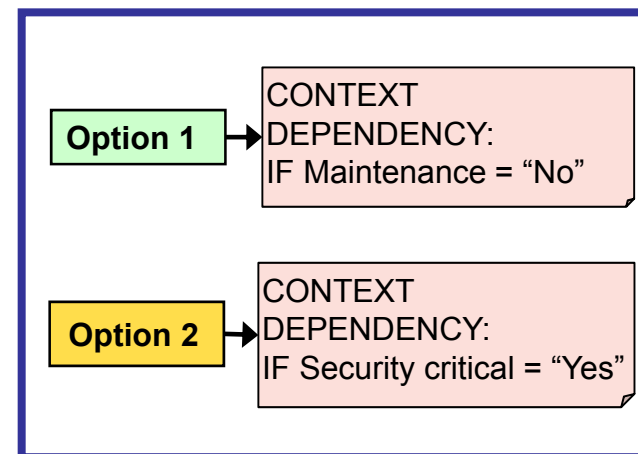
**CONTEXT RULE:**  
IF Security critical = „Yes”  
THEN Maintenance <> „No”



# Reuse Process Models through Configuration: The Provop Approach (Guaranteeing Soundness)

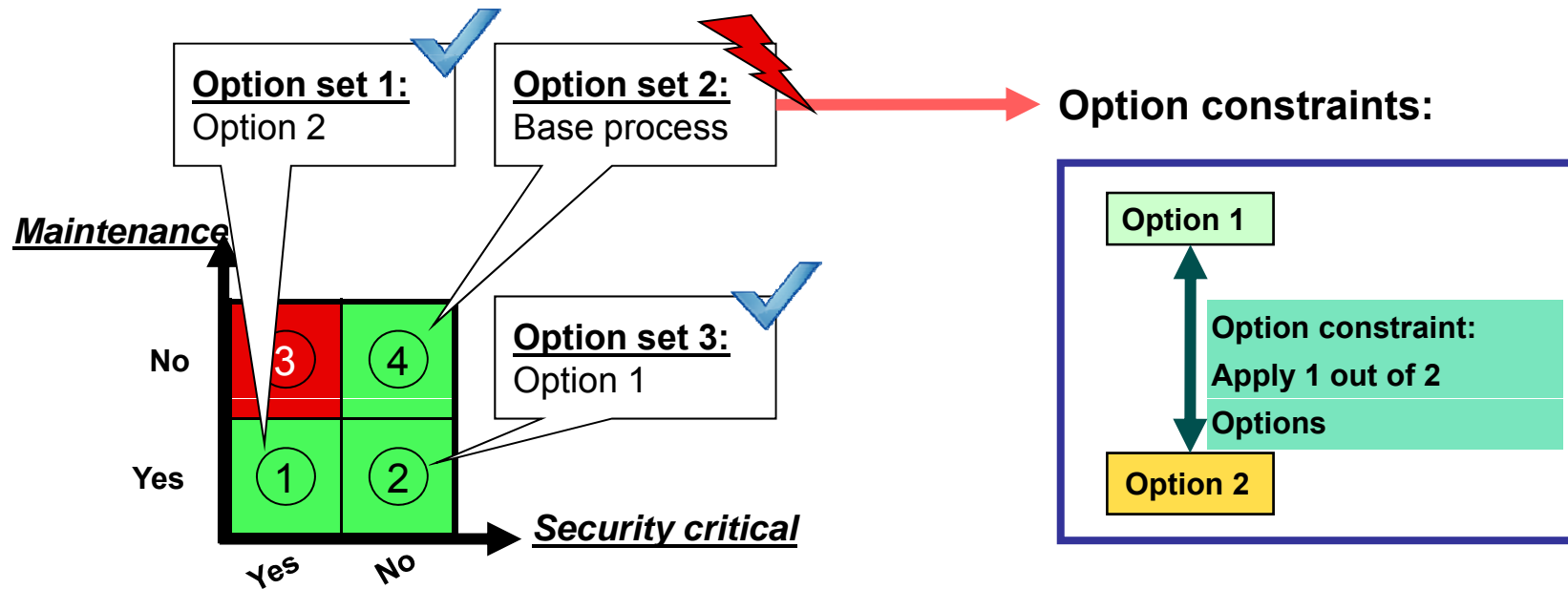
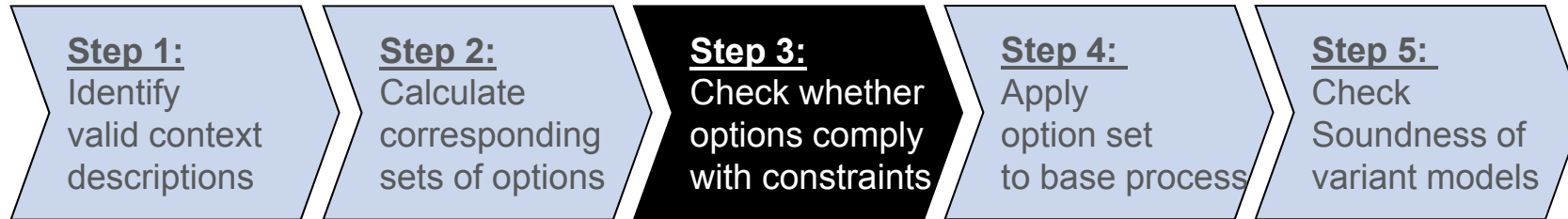


## Context dependencies:

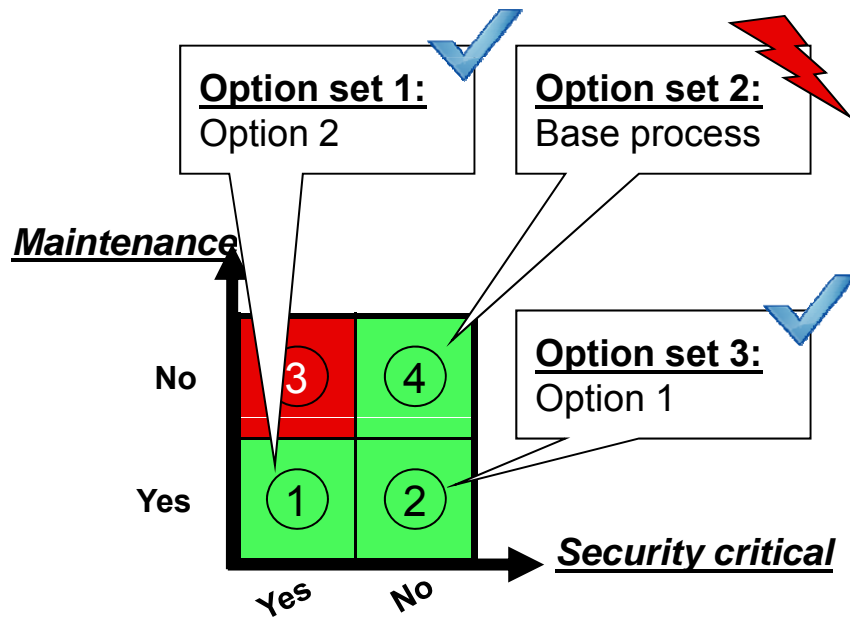
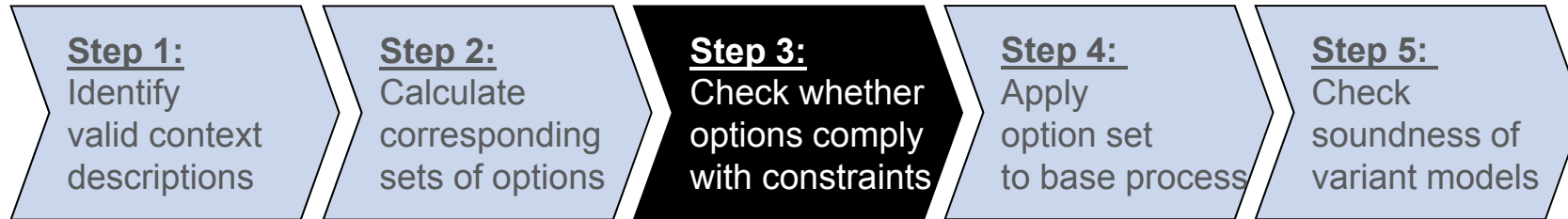




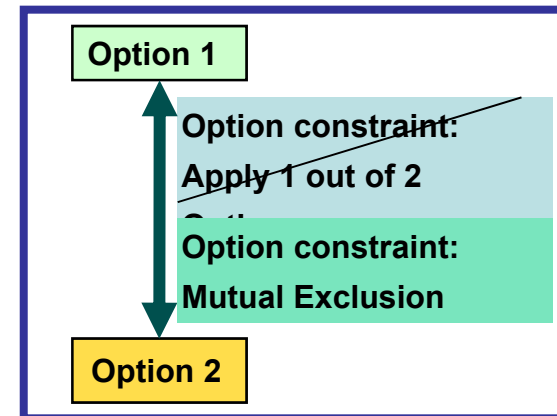
## Reuse Process Models through Configuration: The Provop Approach (Guaranteeing Soundness)



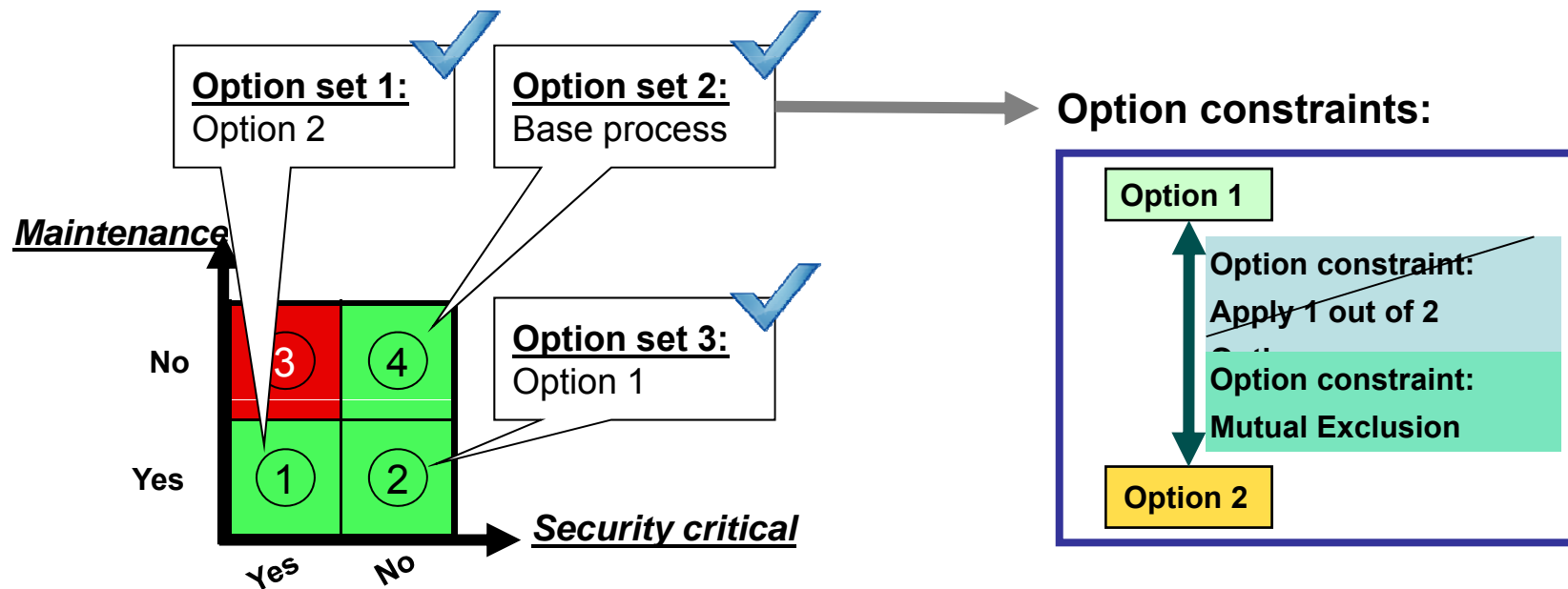
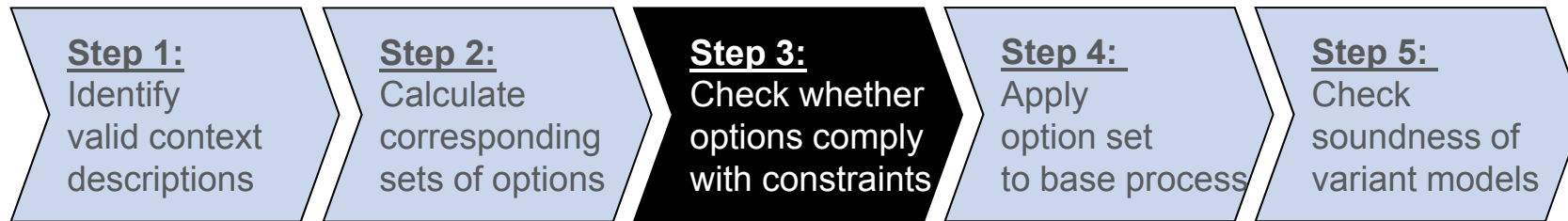
# Reuse Process Models through Configuration: The Provop Approach (Guaranteeing Soundness)



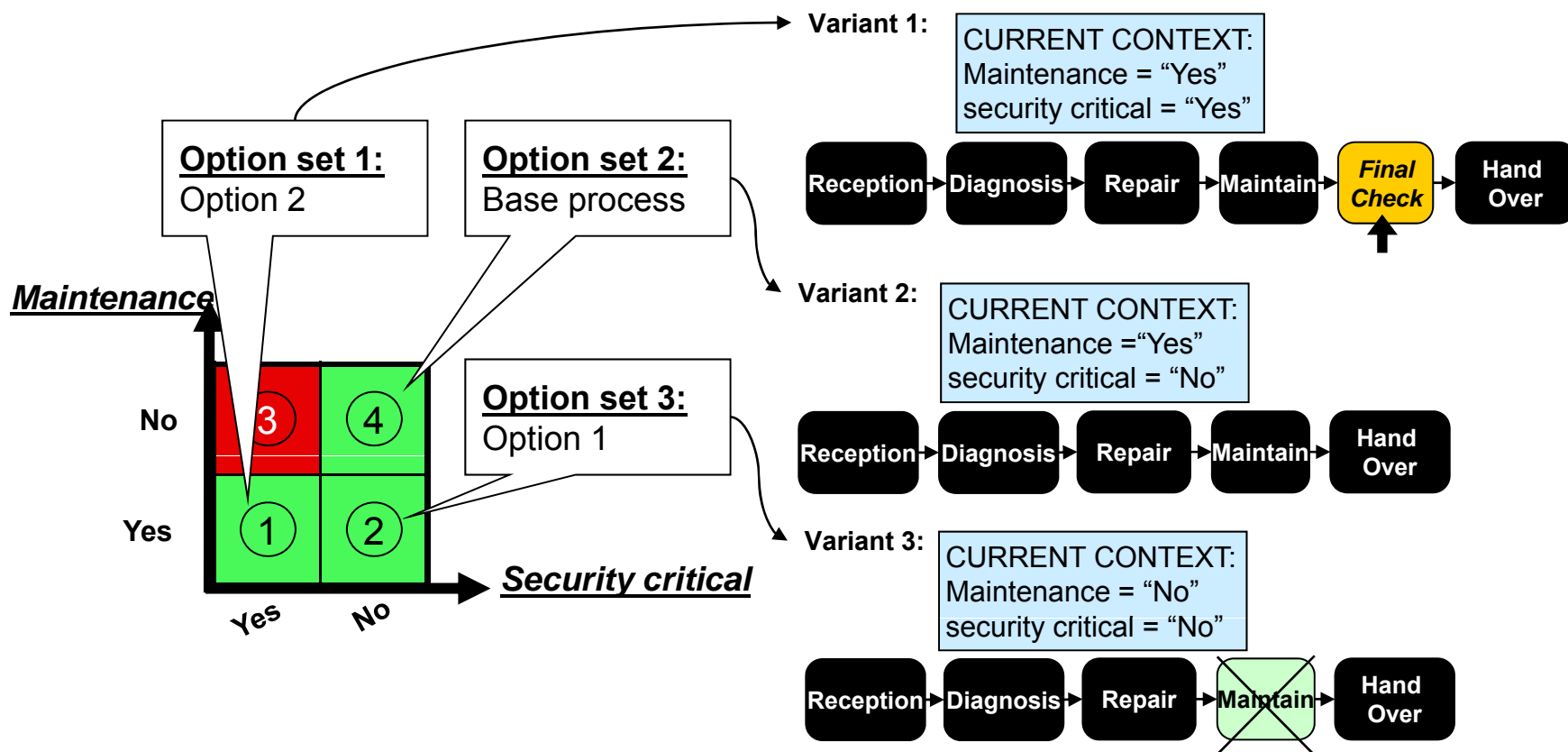
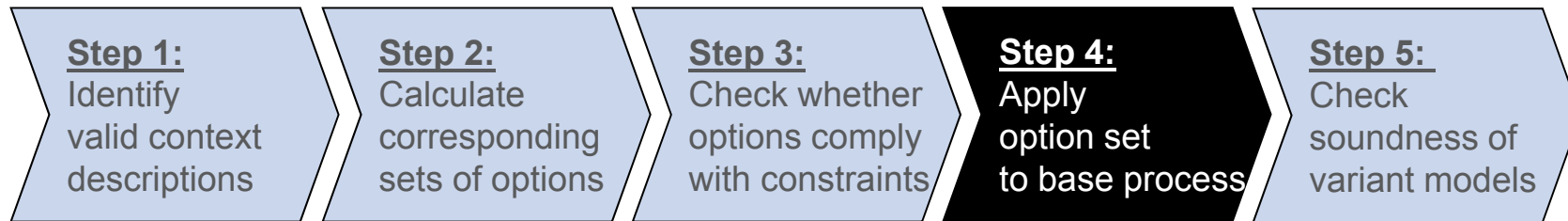
## Option constraints:



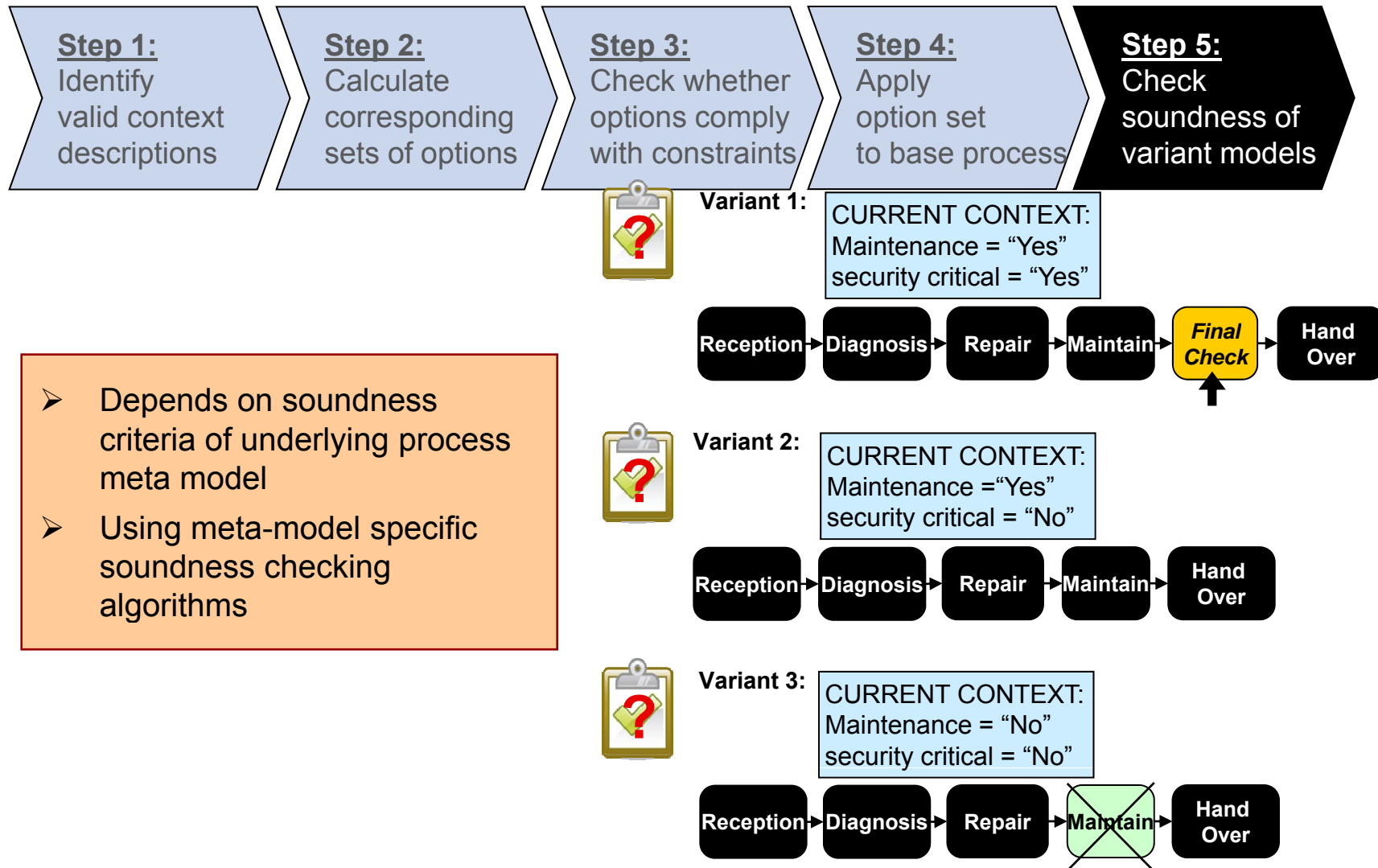
## Reuse Process Models through Configuration: The Provop Approach (Guaranteeing Soundness)



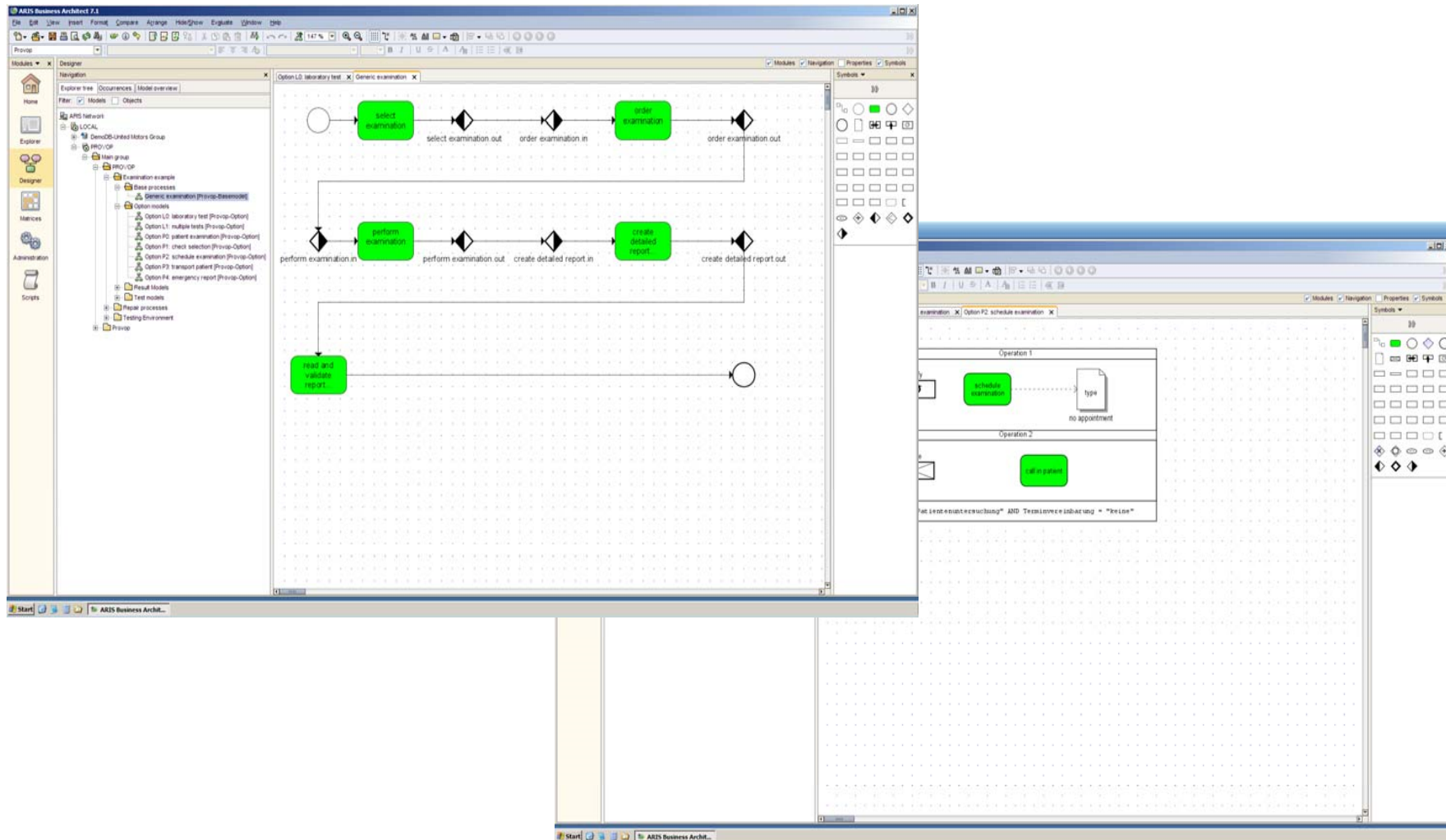
# Reuse Process Models through Configuration: The Provop Approach (Guaranteeing Soundness)



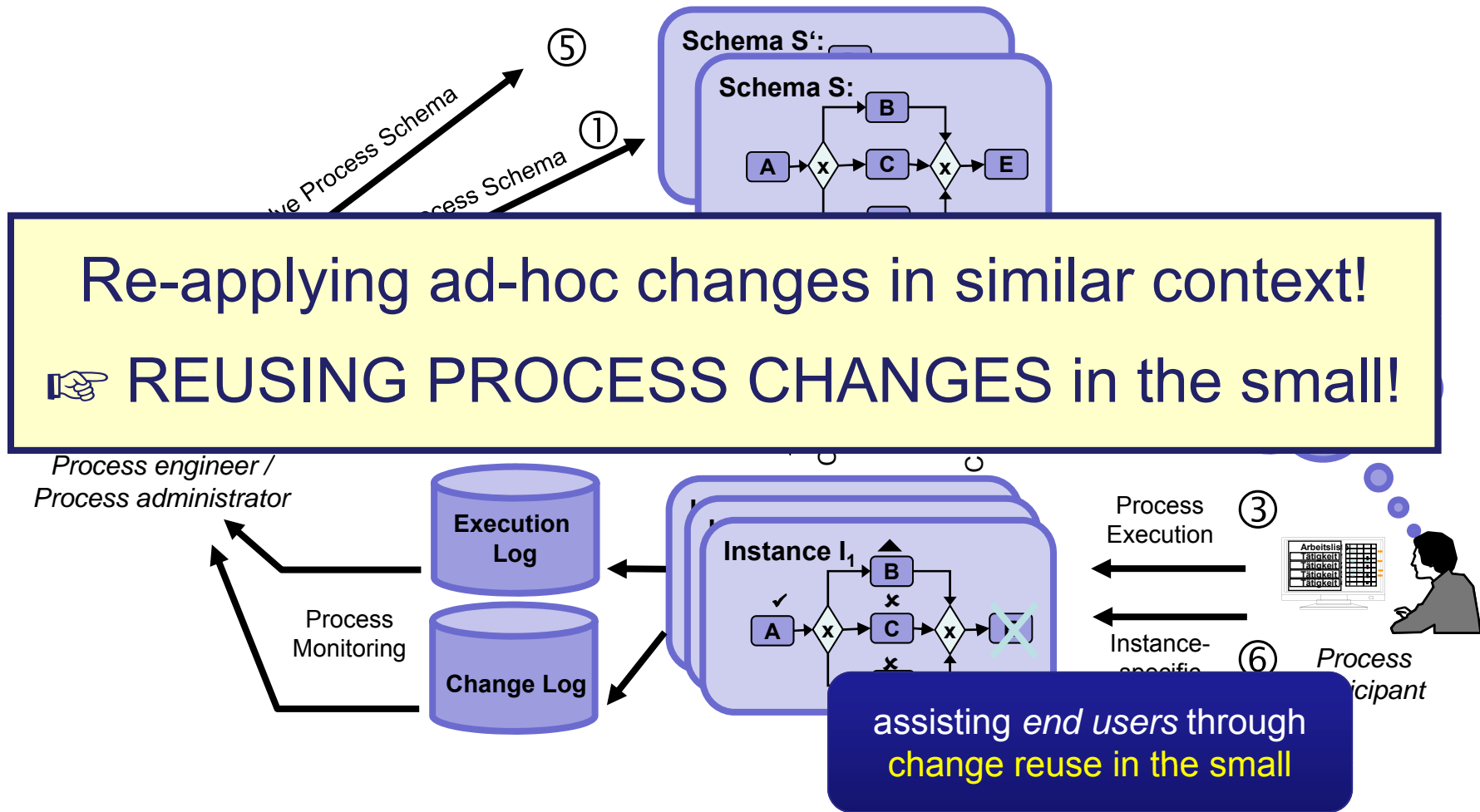
## Reuse Process Models through Configuration: The Provop Approach (Guaranteeing Soundness)



# Reuse Process Models through Configuration: The Provop Approach (Proof-of-Concept Prototype)



# Introduction: Lifecycle Support for Dynamic Processes

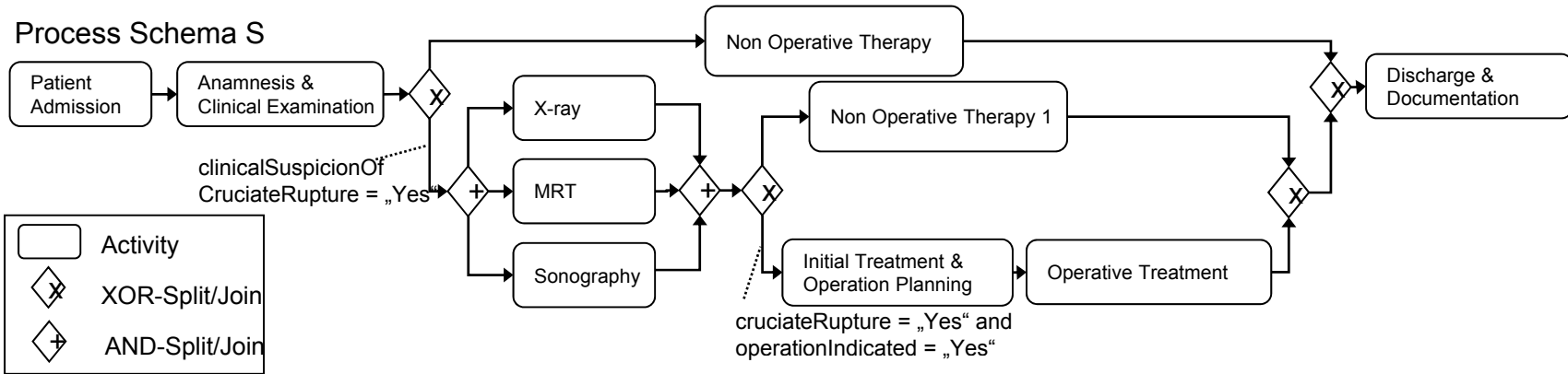


# A Short Tutorial on Ad-hoc Process Changes

## System's View

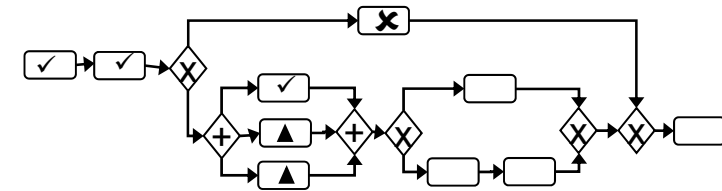
### Process Type Level

#### Process Schema S



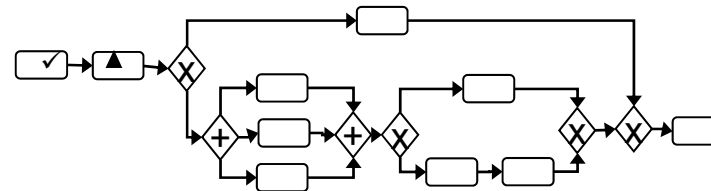
### Process Instance Level

#### Process Instance I1



Execution Trace:  
 $\sigma_1 = \langle \text{„Patient Admission“}, \text{„Anamnesis & Clinical Examination“}, \text{„X-ray“} \rangle$

#### Process Instance I2



Execution Trace:  
 $\sigma_2 = \langle \text{„Patient Admission“} \rangle$

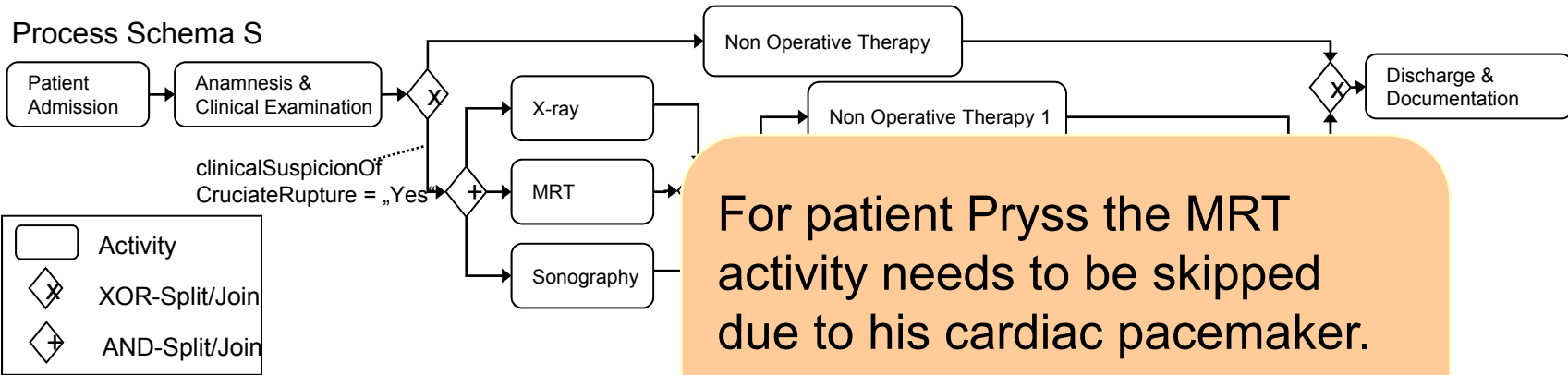


# A Short Tutorial on Ad-hoc Process Changes


## System's View

### Process Type Level

#### Process Schema S

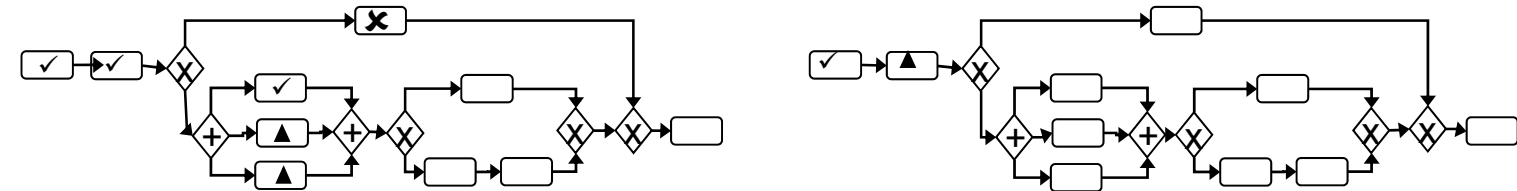


For patient Pryss the MRT activity needs to be skipped due to his cardiac pacemaker.



### Process Instance Level

#### Process Instance I1



Execution Trace:

$\sigma_1 = \langle \text{„Patient Admission“}, \text{„Anamnesis & Clinical Examination“}, \text{„X-ray“} \rangle$

Execution Trace:

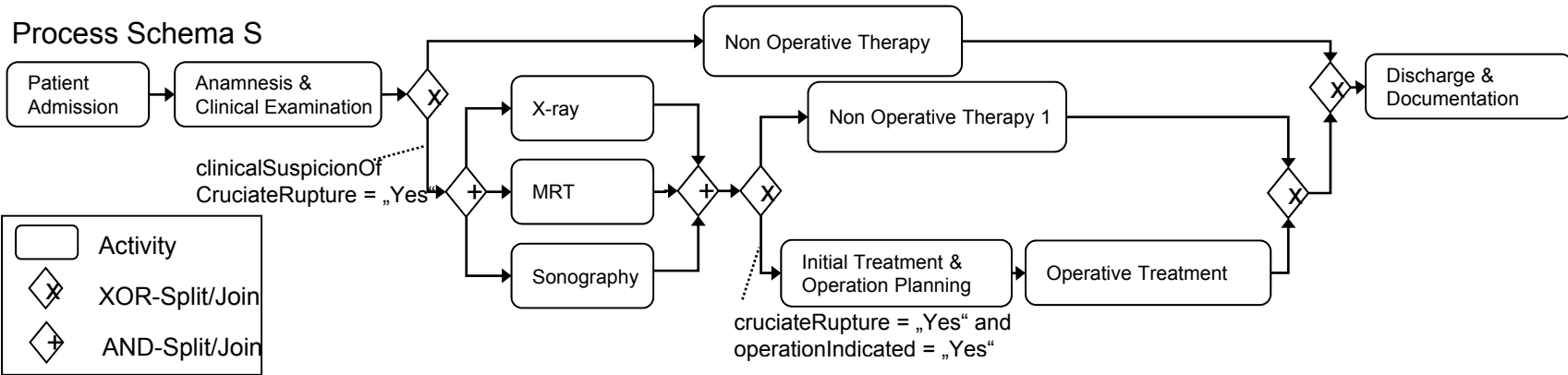
$\sigma_2 = \langle \text{„Patient Admission“} \rangle$

# A Short Tutorial on Ad-hoc Process Changes

## System's View

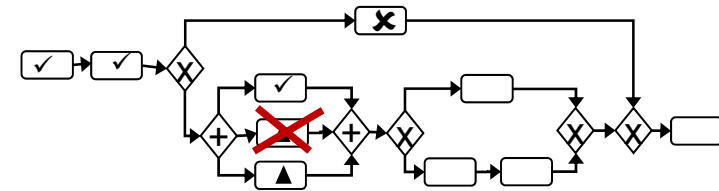
### Process Type Level

#### Process Schema S



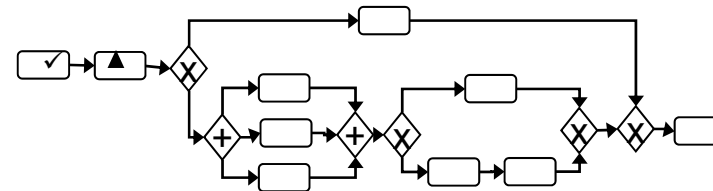
### Process Instance Level

#### Process Instance I1



Execution Trace:  
 $\sigma_1 = \langle \text{„Patient Admission“}, \text{„Anamnesis & Clinical Examination“}, \text{„X-ray“} \rangle$

#### Process Instance I2



Execution Trace:  
 $\sigma_2 = \langle \text{„Patient Admission“} \rangle$

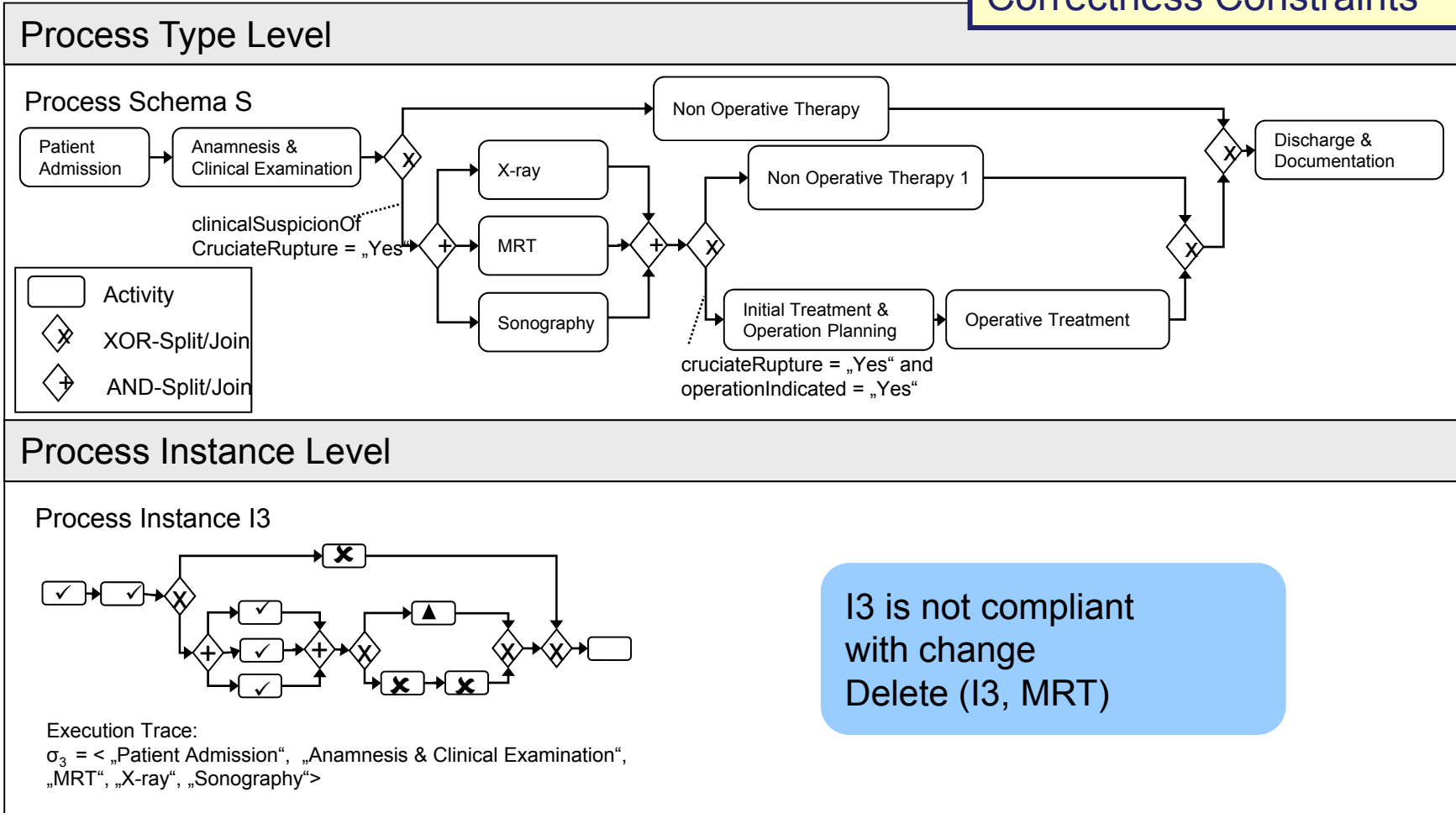
# A Short Tutorial on Ad-hoc Process Changes

## Change Patterns

<b>Pattern AP5: SWAP Process Fragment</b>		<b>Pattern PP3: Late Composition of Process Fragments</b>	
<b>Description</b> Two existing process fragments are swapped in process schema S.	<b>Description</b> 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.	
<b>Example</b> Regarding a particular delivery process the order in which requested goods delivered to two customers has to be swapped.	<b>Description</b> A process fragment X is added to a process schema S.	ical examinations are accomplished in a hospital. The exact	
<b>Problem</b>	<b>Example</b> For a particular patient an allergy test has to be added to his treatment process due to a drug incompatibility.	nt individually depending on his/her medical problems.	
	<b>Problem</b> In a real world process a task has to be accomplished which has not been modeled in the process schema so far.	ariants of how process fragments can be composed. To reduce the number	
	<b>Design Choices</b> (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 ( <i>serial insert</i> ) 2. X is inserted between two activity sets (insert between node sets) a) without additional condition ( <i>parallel insert</i> ) b) with additional condition ( <i>conditional insert</i> )	be specified by the process engineer during build time, process instances
<b>Implementation</b>		posed from a given set of fragments.	
<b>Related Patterns</b>	<b>Implementation</b> 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).	sic building blocks for late modeling? fragments from the repository can be chosen. it-based subset of the process fragments from the repository can be	
		ties or process fragments can be defined.	
		How shall the execution of instance I1 proceed? D is mutually exclusive with A so maybe fragment <B,C> would be a good choice.	

# A Short Tutorial on Ad-hoc Process Changes

## Correctness Constraints



# A Short Tutorial on Ad-hoc Process Changes

**ADEPT :**

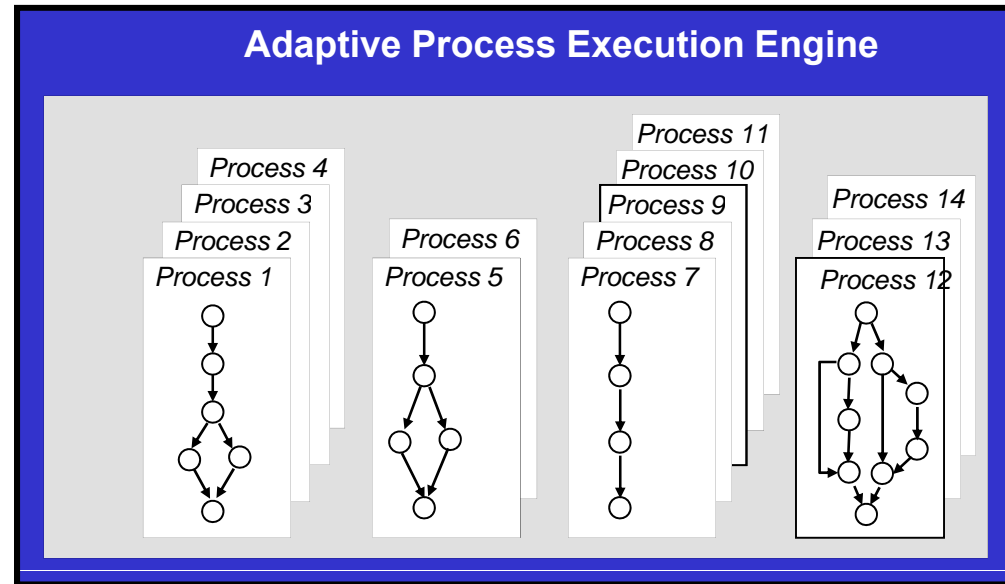
**Individually adaptable  
Process Instances**



**Process Instance**

**=**

**(individual) "Process Program"**



## A Short Tutorial on Ad-hoc Process Changes

**ADEPT :**

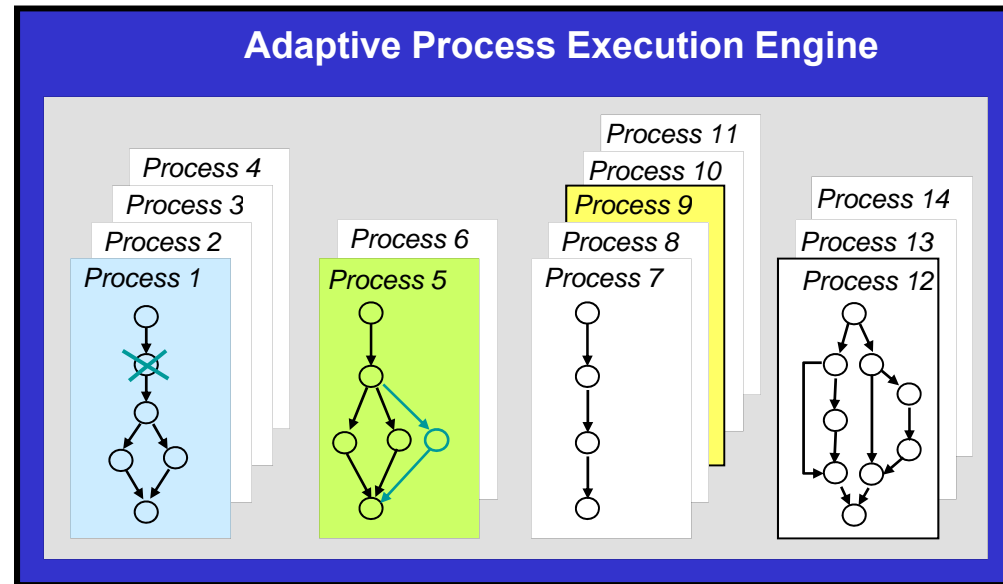
**Individually adaptable  
Process Instances**



**Process Instance**

**=**

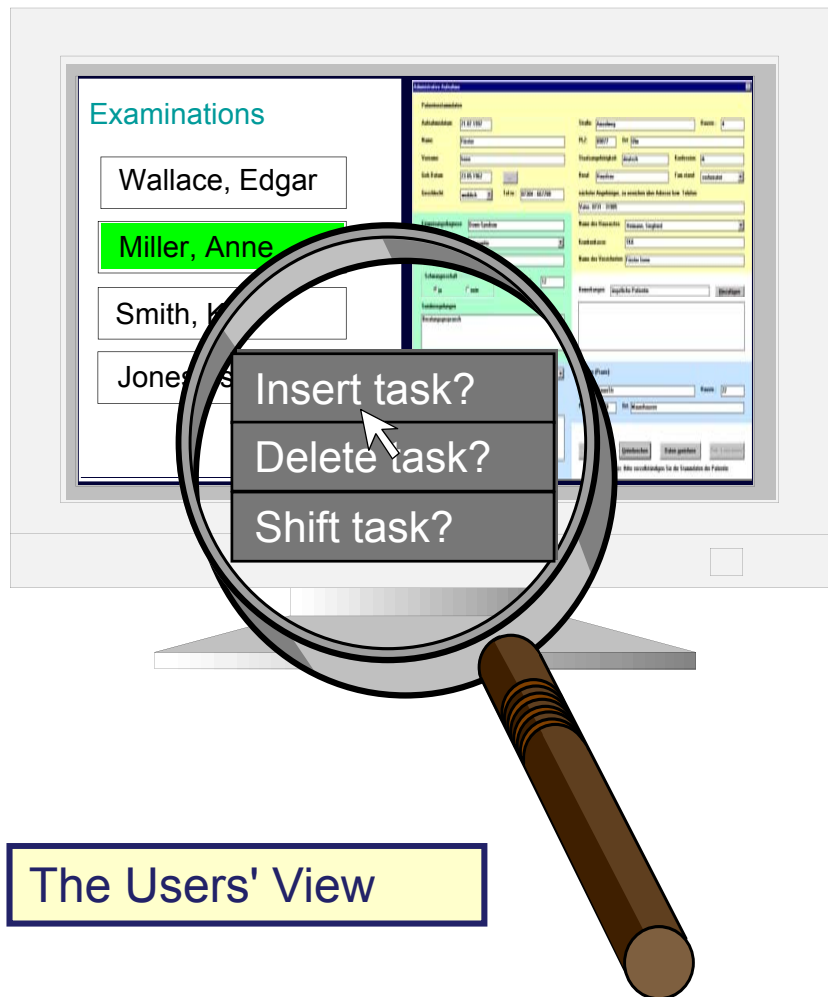
**(individual) "Process Program"**



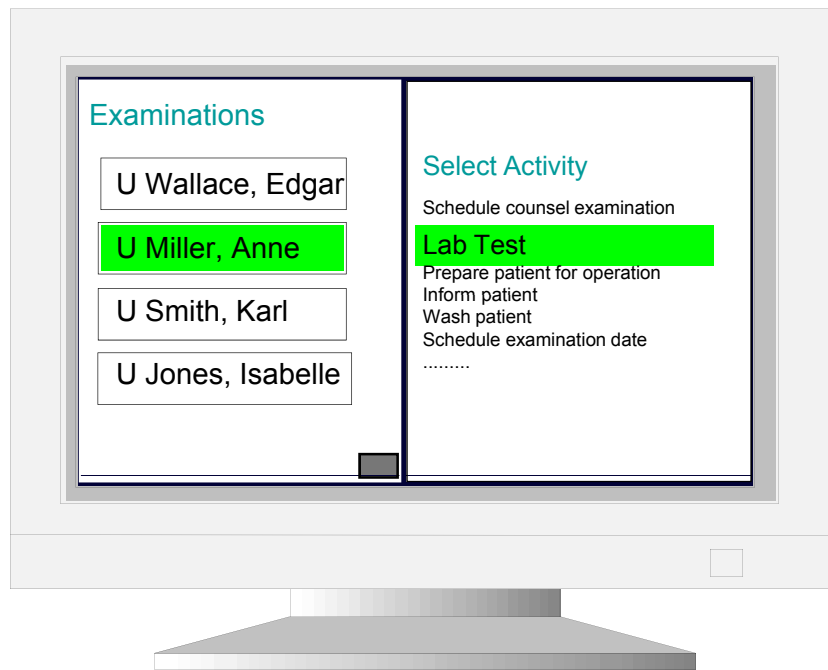
### **Achievements:**

- **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**

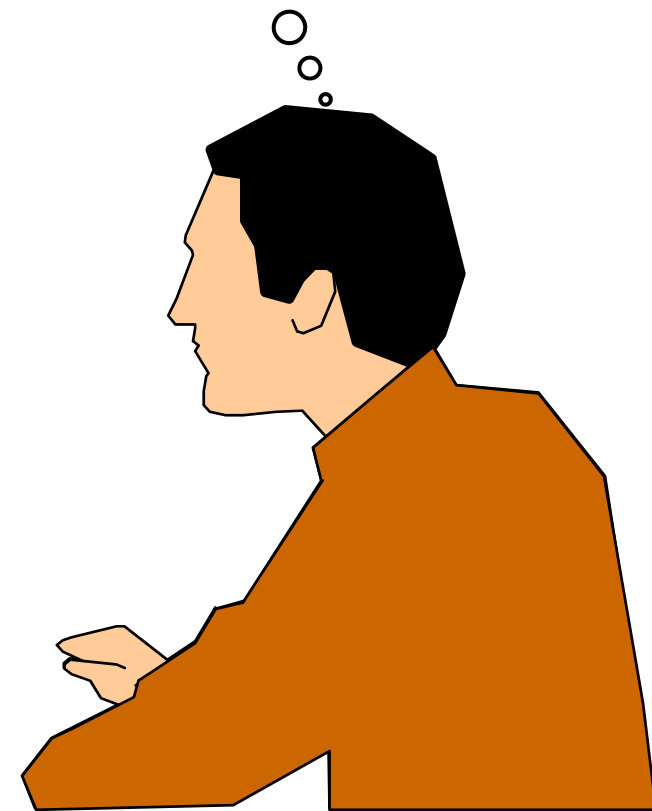
# A Short Tutorial on Ad-hoc Process Changes



# A Short Tutorial on Ad-hoc Process Changes



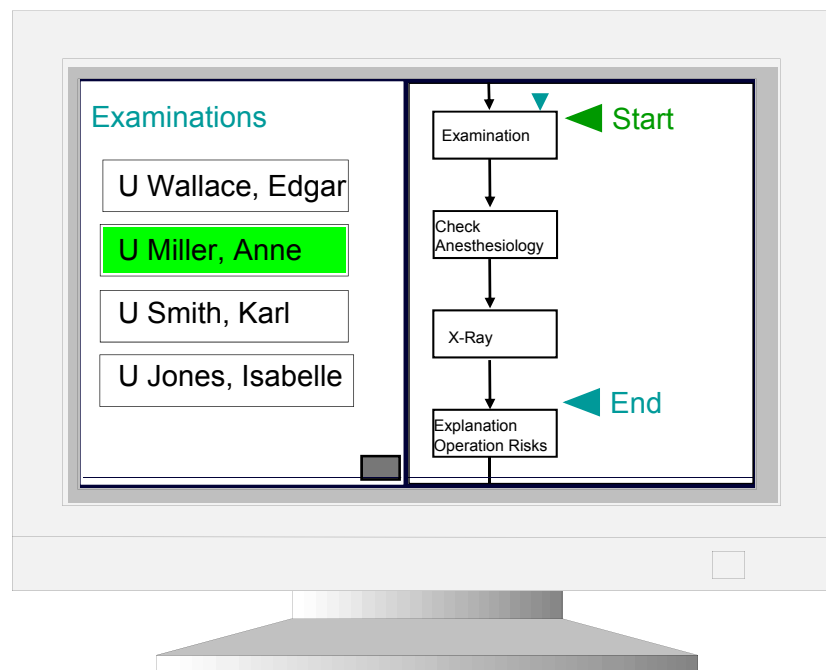
Exceptional case – we need an additional lab test !



The Users' View



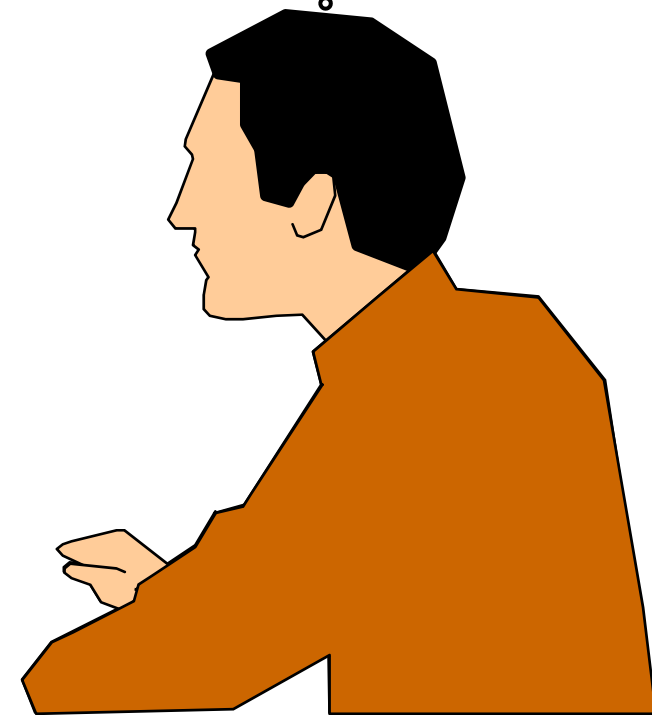
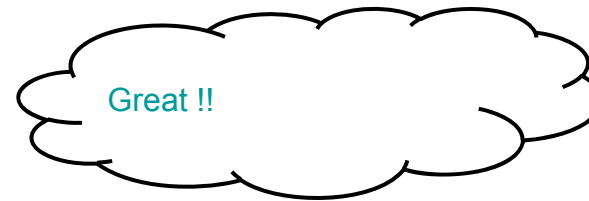
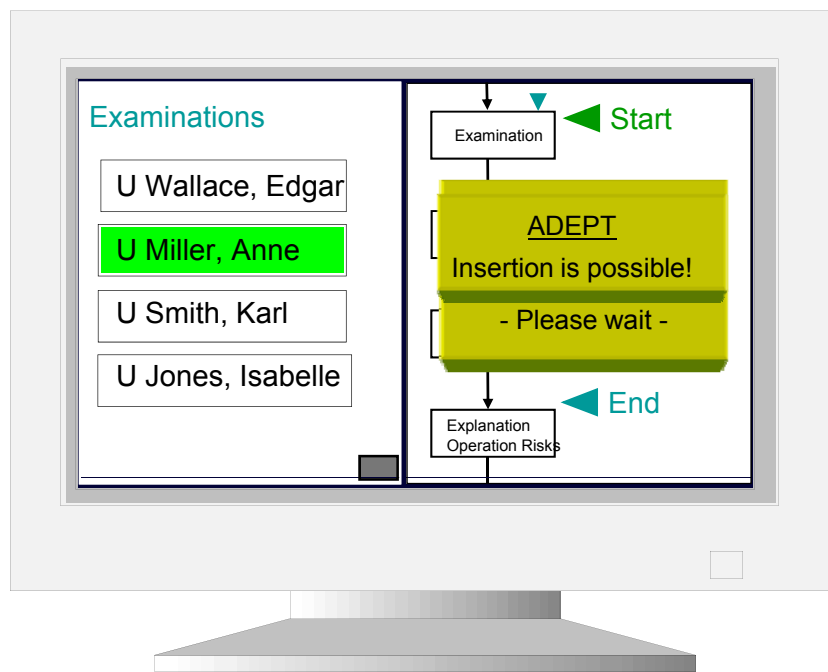
## A Short Tutorial on Ad-hoc Process Changes



The Users' View

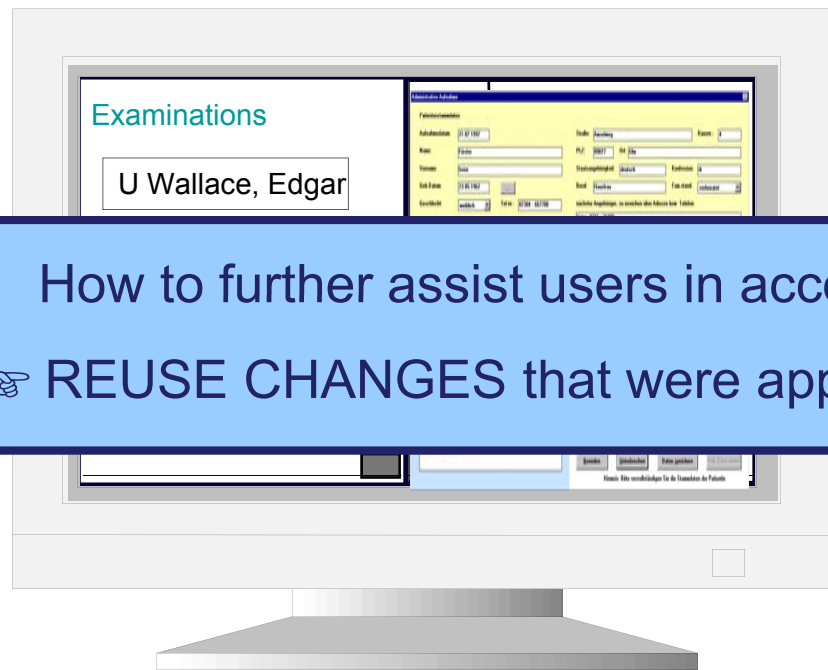


# A Short Tutorial on Ad-hoc Process Changes



The Users' View

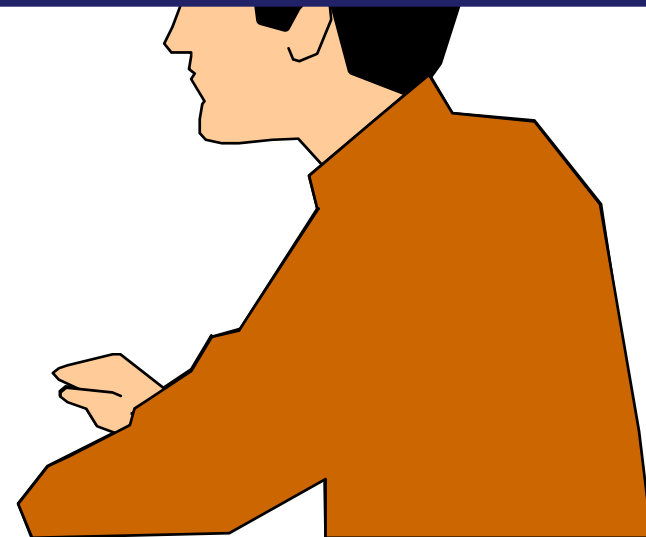
## A Short Tutorial on Ad-hoc Process Changes



OK, now let us continue with the examination !

How to further assist users in accomplishing ad-hoc changes?  
➡ REUSE CHANGES that were applied in similar problem context.

The Users' View



## Ad-hoc Changes: User Assistance and Change Reuse

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

- ❑ Annotate ad-hoc changes with information about their reasons
- ❑ Support users in retrieving past ad-hoc changes applied in similar problem context
- ❑ Assist users in reusing (i.e., re-applying) a past ad-hoc change for a particular process instance when coping with an exceptional situation

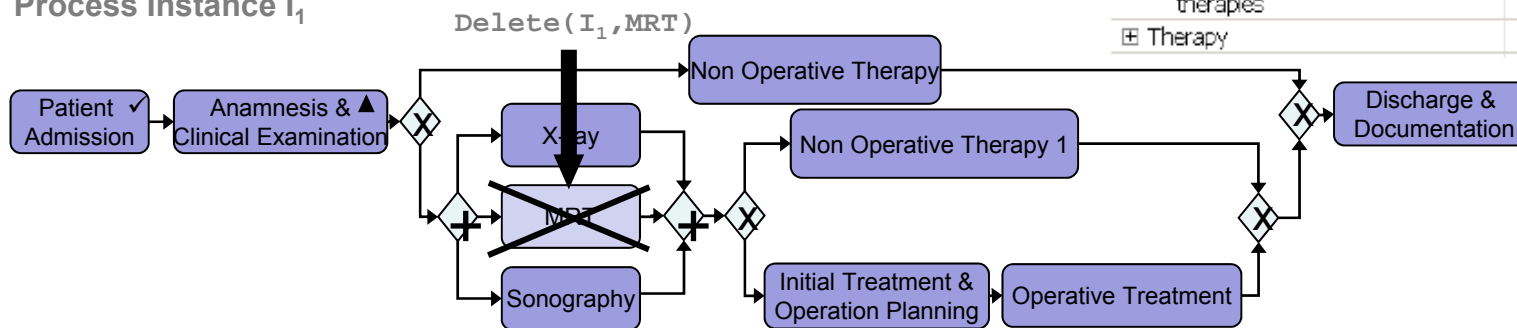
## Ad-hoc Changes: User Assistance and Change Reuse

### Memorization of ad-hoc 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			

#### Process Instance $I_1$



#### Case $c_1$

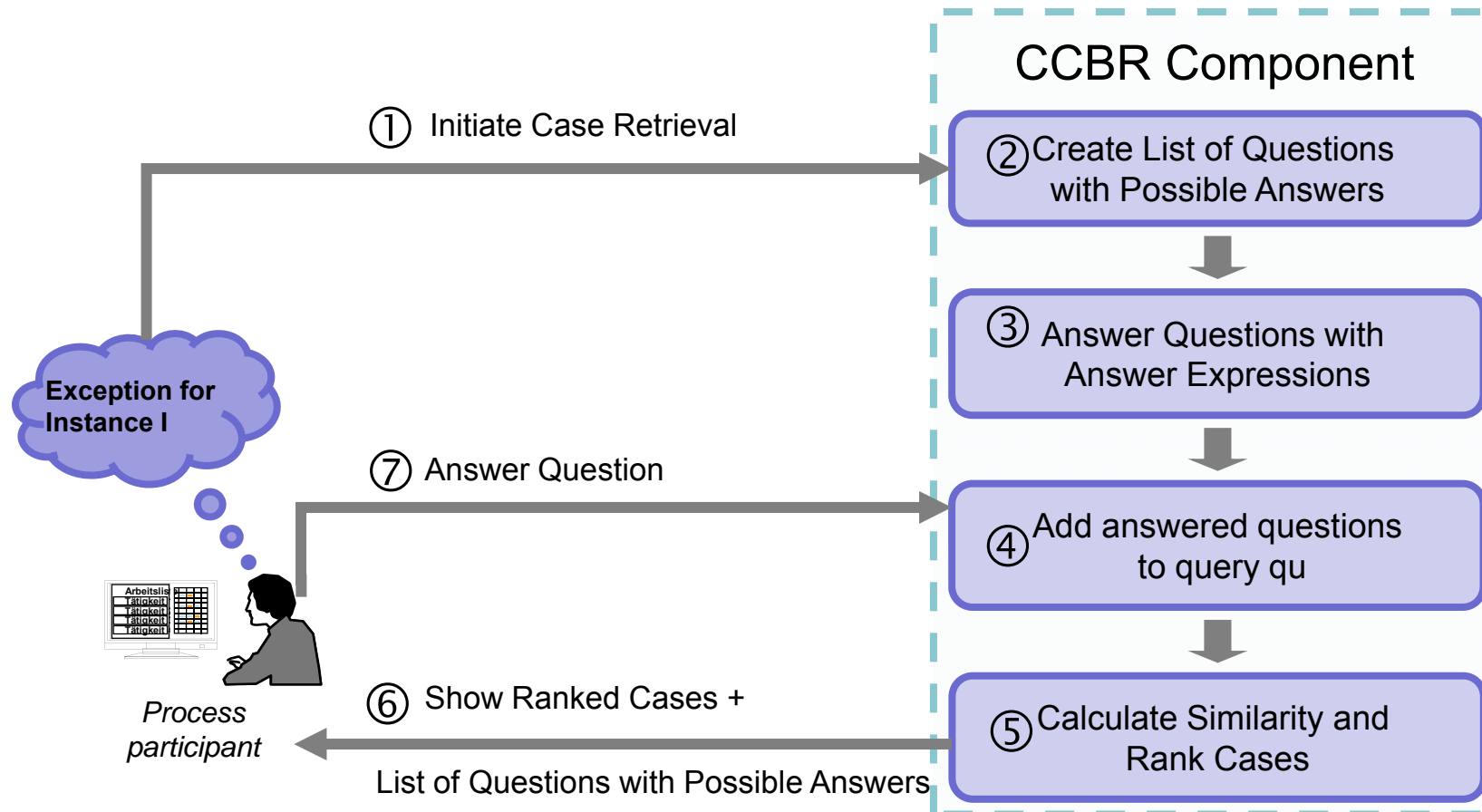
$pd_{c_1}$  = The treatment of cruciate ruptures routinely includes a magnetic resonance tomography (MRT), an X-ray and a sonography. However, for a particular patient the MRT may have to be skipped as the respective patient has a cardiac pacemaker.

$sol_{c_1}$  =  $\langle \text{Delete}(S_I, \text{MRT}) \rangle$

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

## Ad-hoc Changes: User Assistance and Change Reuse

### Semi-automated retrieval of similar *instance deviations* using conversational case-based reasoning (CCBR)

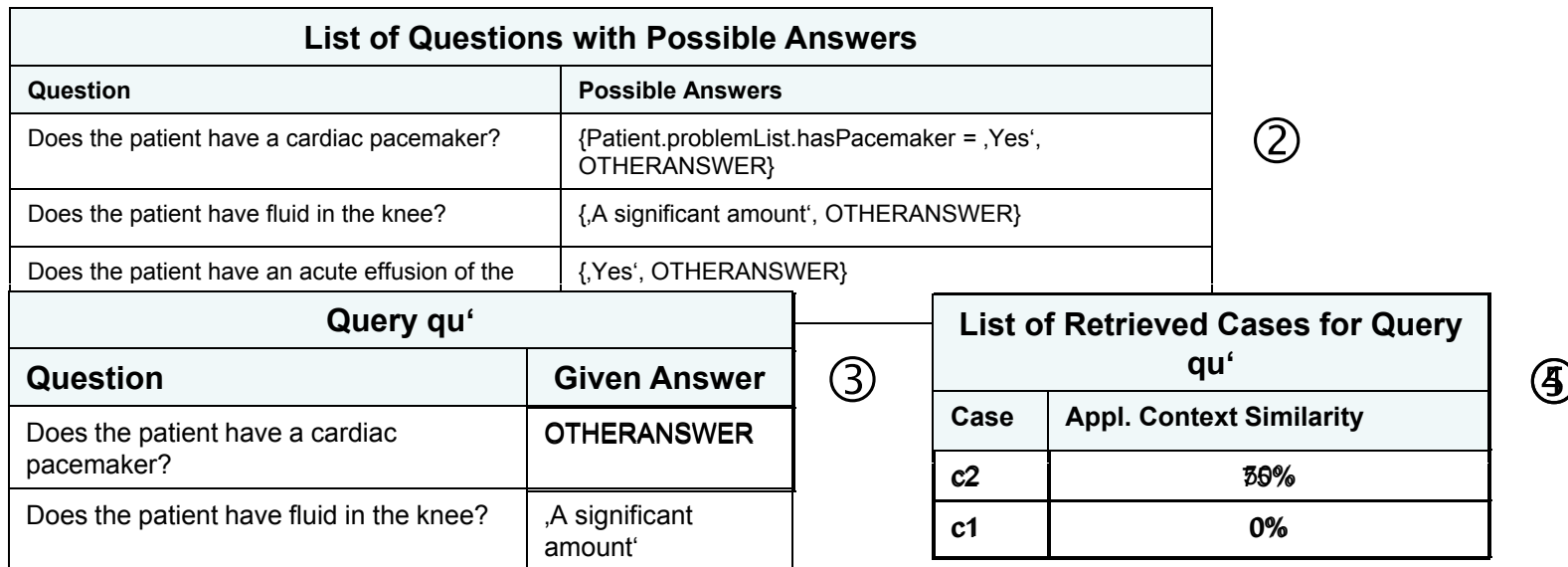


## Ad-hoc Changes: User Assistance and Change Reuse

### Retrieving similar instance deviations based on the actual context

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

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



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

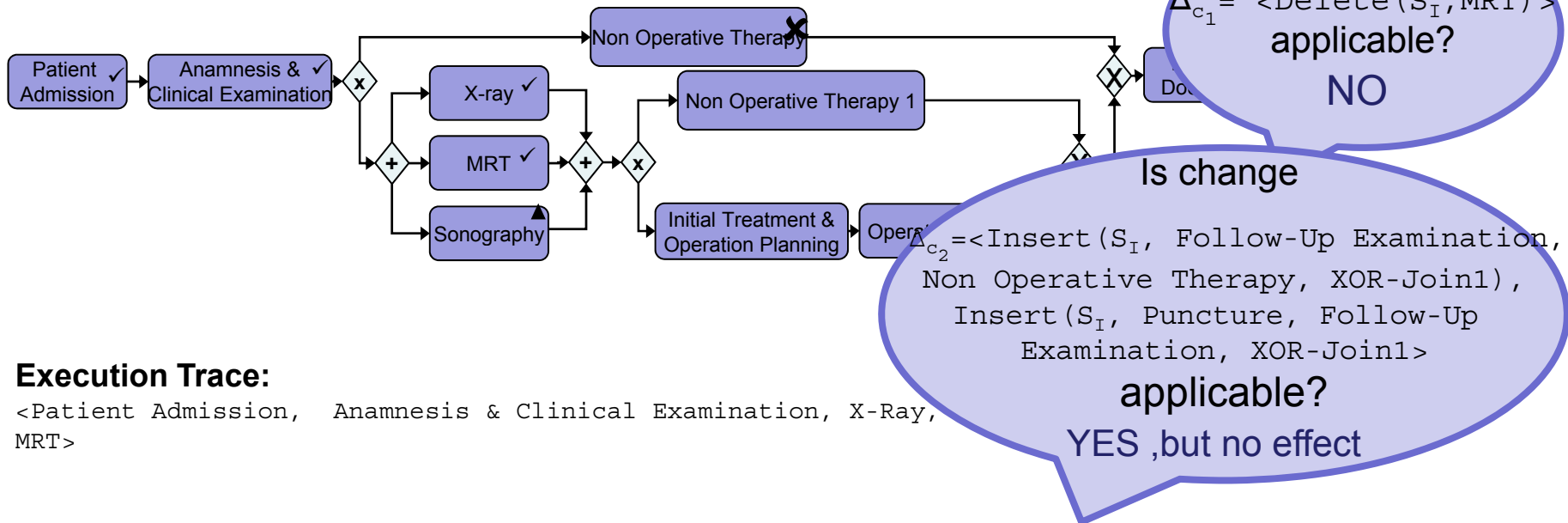
# Ad-hoc Changes: User Assistance and Change Reuse

Retrieving similar instance deviations based on actual context + status

List of Retrieved Cases		
Case	Similarity	
c2	75 %	c <sub>2</sub> does not have any effect, but is adjustable
c1	0%	c <sub>1</sub> 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<sub>1</sub>



Execution Trace:

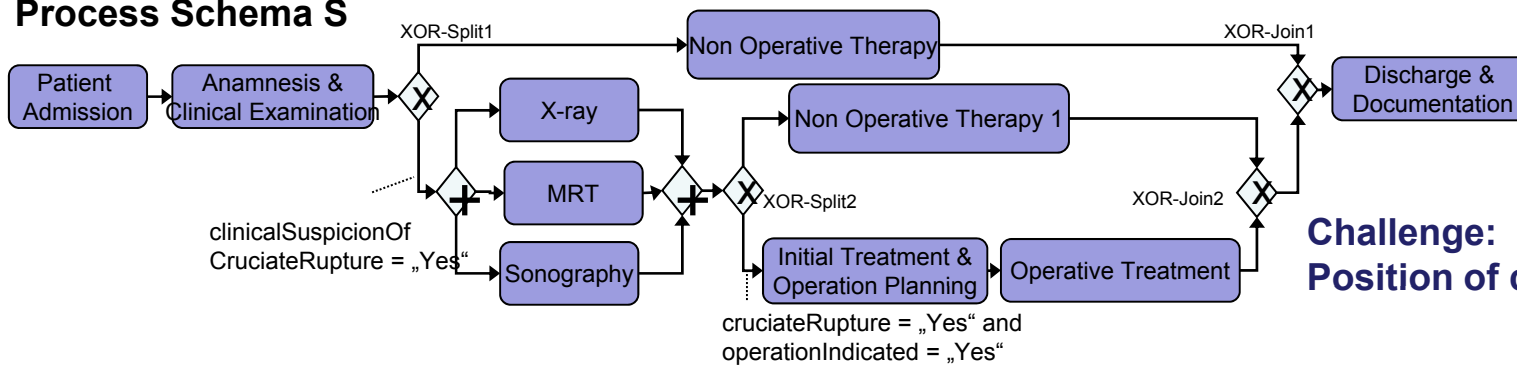
<Patient Admission, Anamnesis & Clinical Examination, X-Ray, MRT>



# Ad-hoc Changes: User Assistance and Change Reuse

## Deriving Type Changes from Frequently Occuring Instance Changes

### Process Schema S



**Challenge:**  
Position of change can vary

Case  $c_1$

$freq_{c_1} = 51$

...  
 $\Delta_{c_1} = \langle \text{Delete}(S_I, \text{MRT}) \rangle$

Case  $c_2$

$freq_{c_2} = 41$

...  
 $\Delta_{c_2} = \langle \text{Insert}(S_I, \text{Follow-Up Examination}, \text{Non Operative Therapy}, \text{XOR-Join1}), \text{Insert}(S_I, \text{Puncture}, \text{Follow-Up Examination}, \text{XOR-Join1}) \rangle$

Case  $c_3$

$freq_{c_3} = 35$

...  
 $\Delta_{c_3} = \langle \text{Insert}(S_I, \text{Follow-Up Examination}, \text{Non Operative Therapy1}, \text{XOR-Join2}), \text{Insert}(S_I, \text{Puncture}, \text{Follow-Up Examination}, \text{XOR-Join2}) \rangle$

# Ad-hoc Changes: User Assistance and Change Reuse

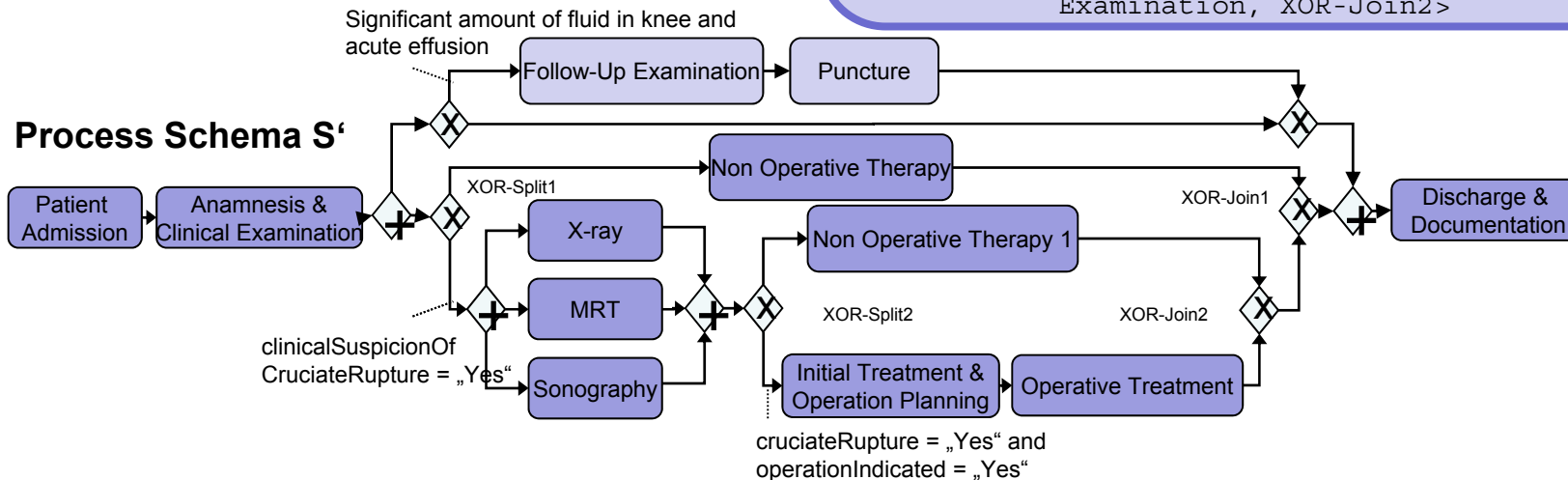
## Deriving Type Changes from Frequently Occuring Instance Changes

Case  $c_2$  $\text{freq}_{c_2} = 41$ 

$\text{qaSet}_{c_2} = \{$   
 (Does the patient have fluid in the knee?, 'A significant amount'),  
 (Does the patient have an acute effusion of the knee?, 'Yes')  
 $\}$   
 $\Delta_{c_2} =$   
 $\langle$ Insert( $S_I$ , Follow-Up Examination,  
 Non Operative Therapy, XOR-Join1),  
 Insert( $S_I$ , Puncture, Follow-Up  
 Examination, XOR-Join1) $\rangle$

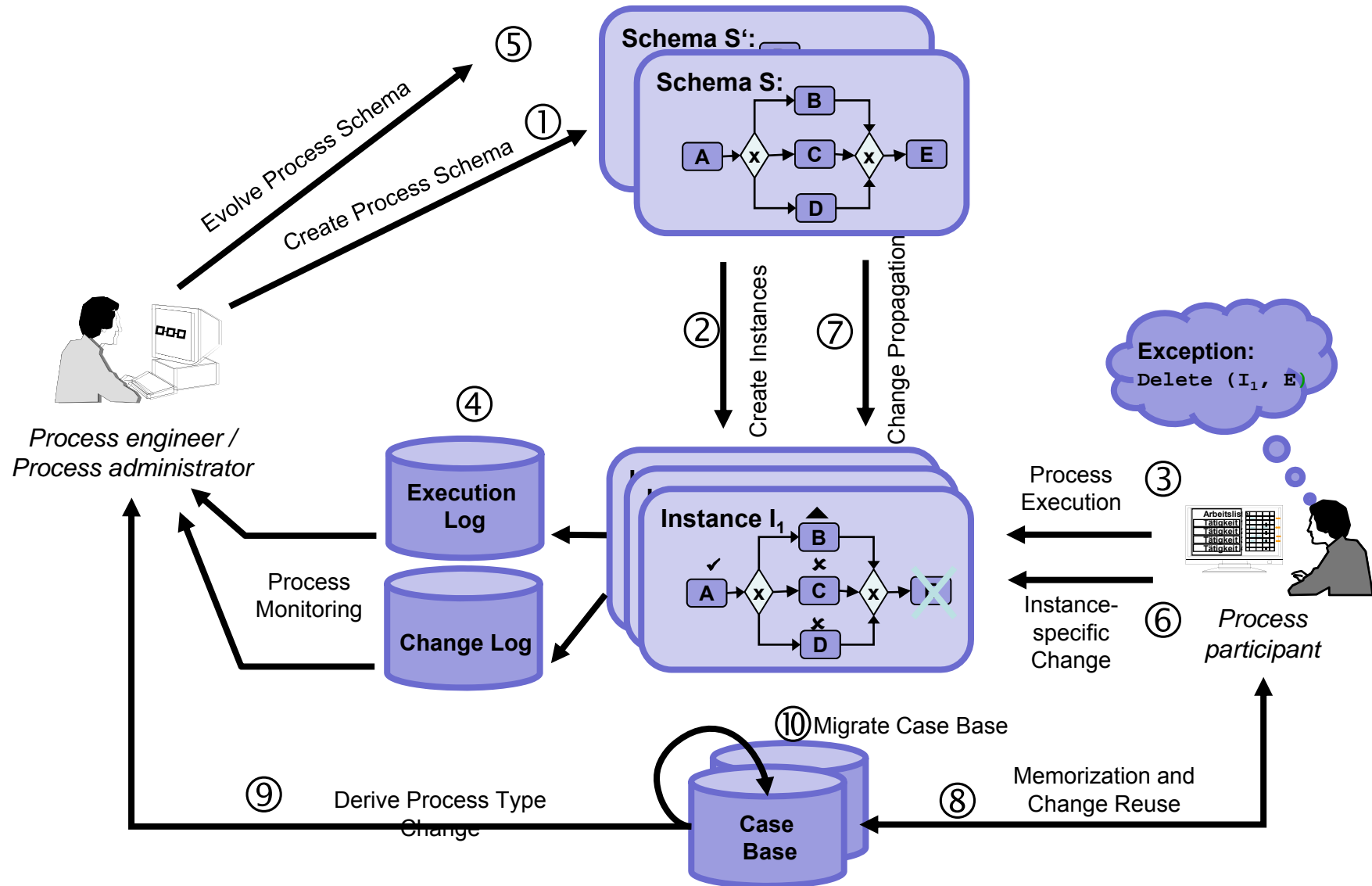
Case  $c_3$  $\text{freq}_{c_3} = 35$ 

$\text{qaSet}_{c_3} = \{$   
 (Does the patient have fluid in the knee?, 'A significant amount'),  
 (Does the patient have an acute effusion of the knee?, 'Yes')  
 $\}$   
 $\Delta_{c_3} =$   
 $\langle$ Insert( $S_I$ , Follow-Up Examination,  
 Non Operative Therapy1, XOR-Join2),  
 Insert( $S_I$ , Puncture, Follow-Up  
 Examination, XOR-Join2) $\rangle$



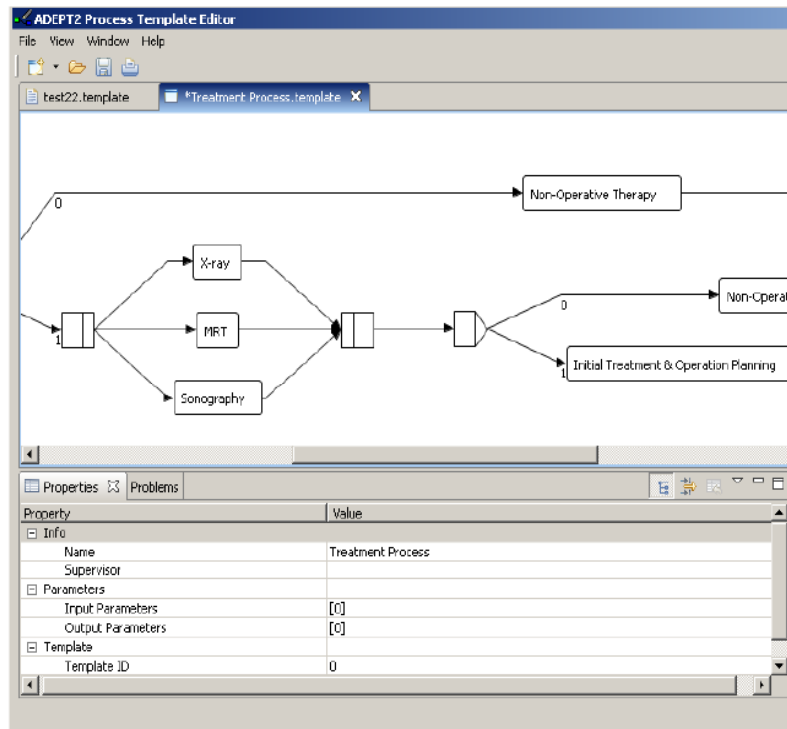
$\Delta_S = \langle$ CondInsert( $S_I$ , Follow-Up Examination, Anamnesis & Clinical Examination, Discharge & Documentation),  
 CondInsert( $S_I$ , Puncture, Follow-Up Examination, Discharge & Documentation) $\rangle$

# Ad-hoc Changes: User Assistance and Change Reuse



# Ad-hoc Changes: User Assistance and Change Reuse

## Proof-of-Concept Implementation



Developed as part of the ProCycle project funded by Tiroler Wissenschaftsfond

The top screenshot shows the CBR Tool interface with the following details:

- Basic Information:** Name: Effusion of knee; Description: Patient has a significant amount of fluid in his knee and suffers from acute effusion of the knee.
- Action:** A table of ADEPT Change Operations:

Operation Type	Details
Insert	SI, Follow-Up Examination, Operative Treatment, XOR-Join2
Insert	SI, Puncture, Follow-Up Examination, XOR-Join2

The bottom screenshot shows a questionnaire and a table of similar cases:

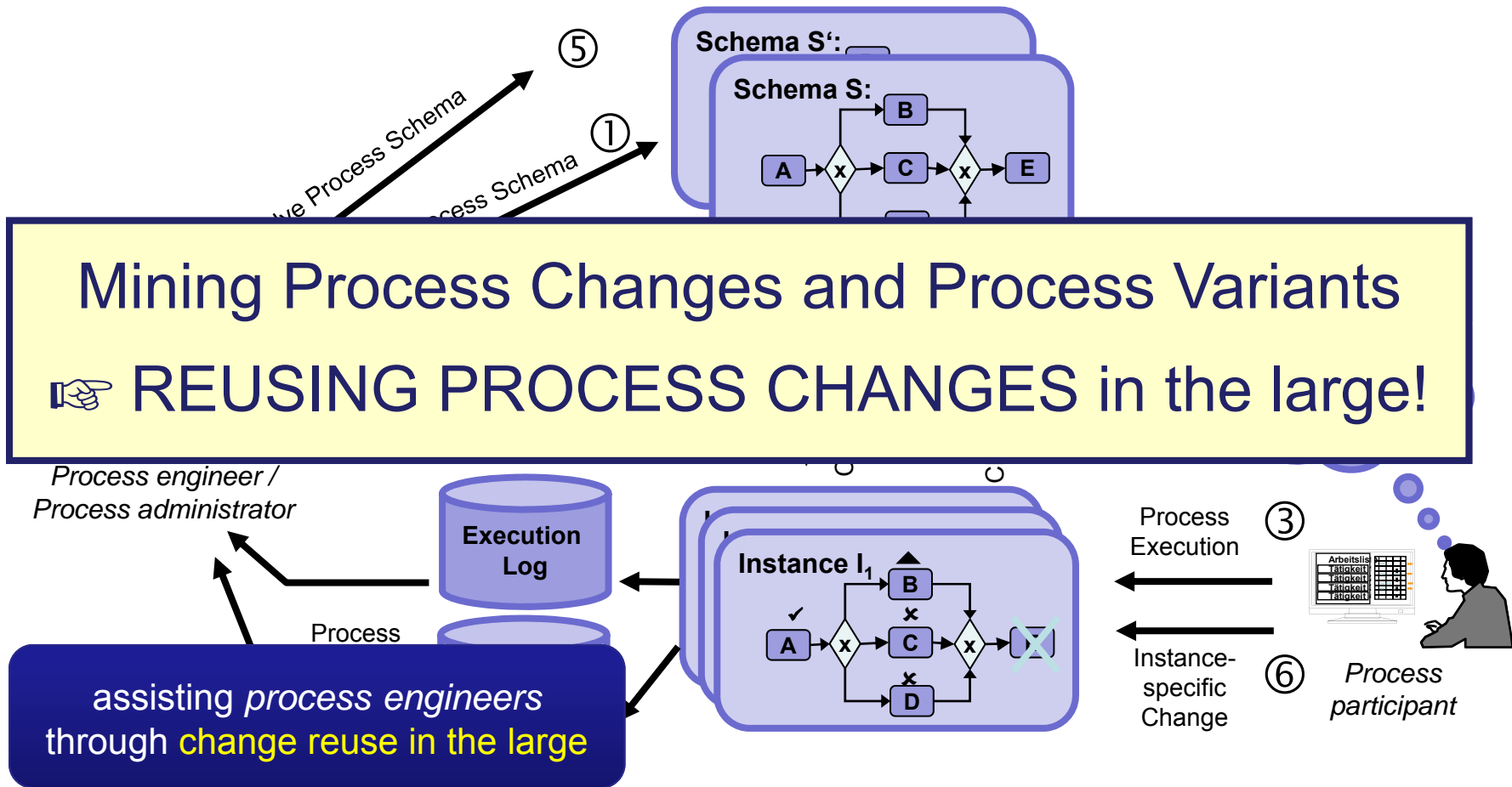
Please answer some of these questions

Question	Answer
Does the patient have a cardiac pacemaker?	Other Answer
Does the patient have an acute effusion of the knee?	?
Does the patient have fluid in his knee?	A significant amount

Similar cases

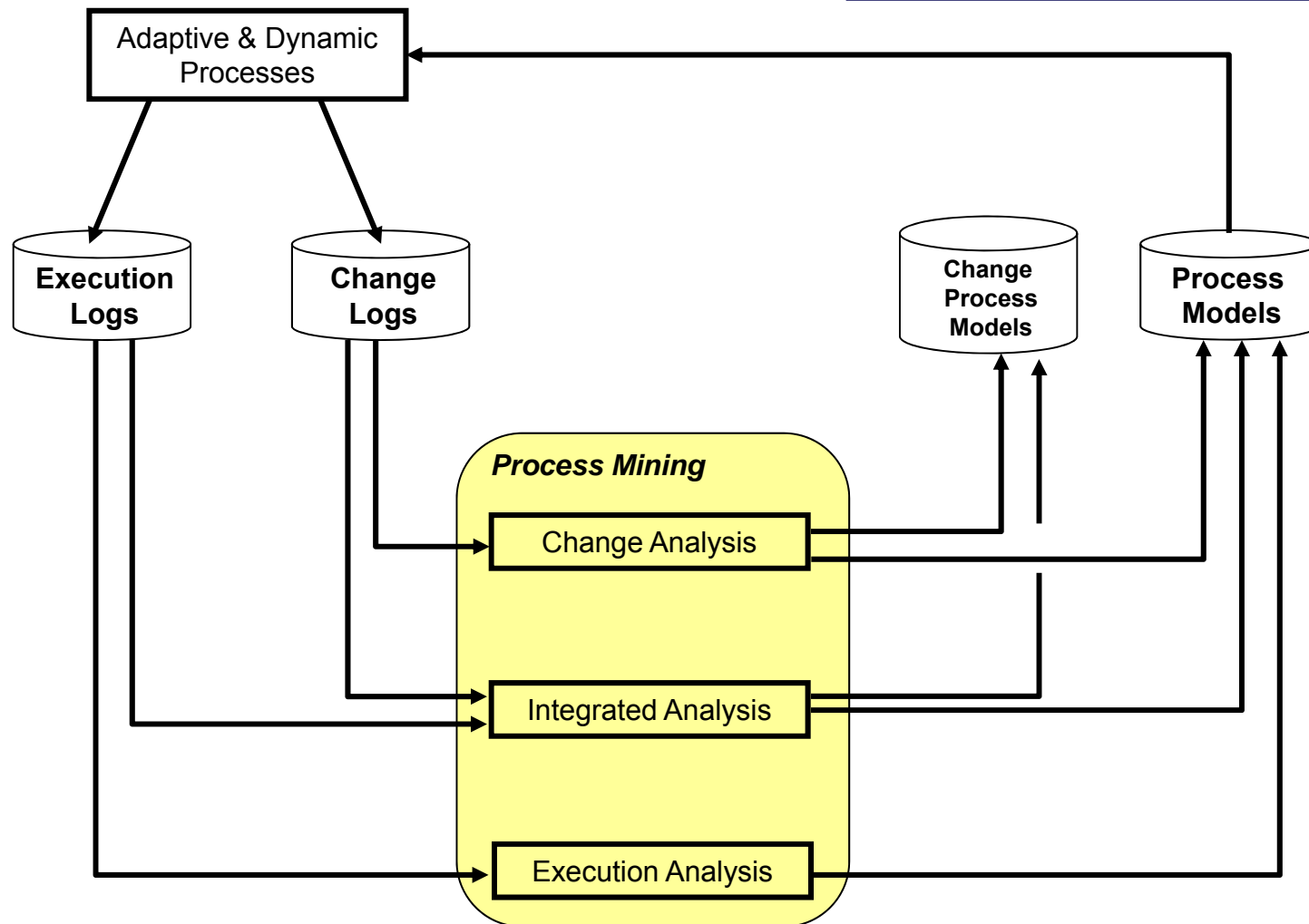
Title	Reuse Counter	Application C...	Control Cont...	Global Similarity	Reputation
Effusion of knee	0	75	100	88	0
Perform Puncture	0	75	33	54	0
Do not perform MRT	0	0	100	50	0

# Introduction: Lifecycle Support for Dynamic Processes



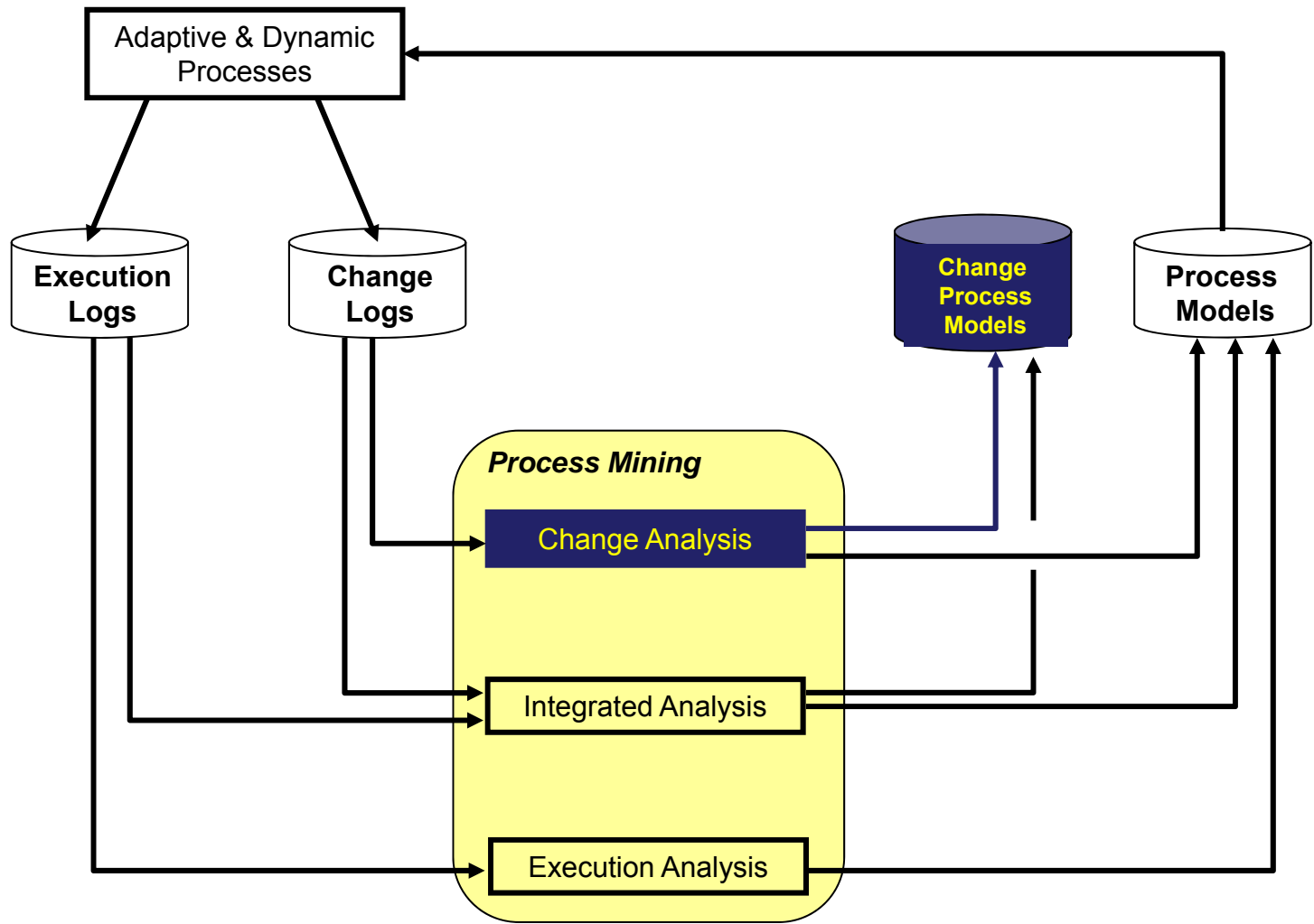
# Reuse Process Adaptations in the Large through Learning

## Execution & Change Analysis

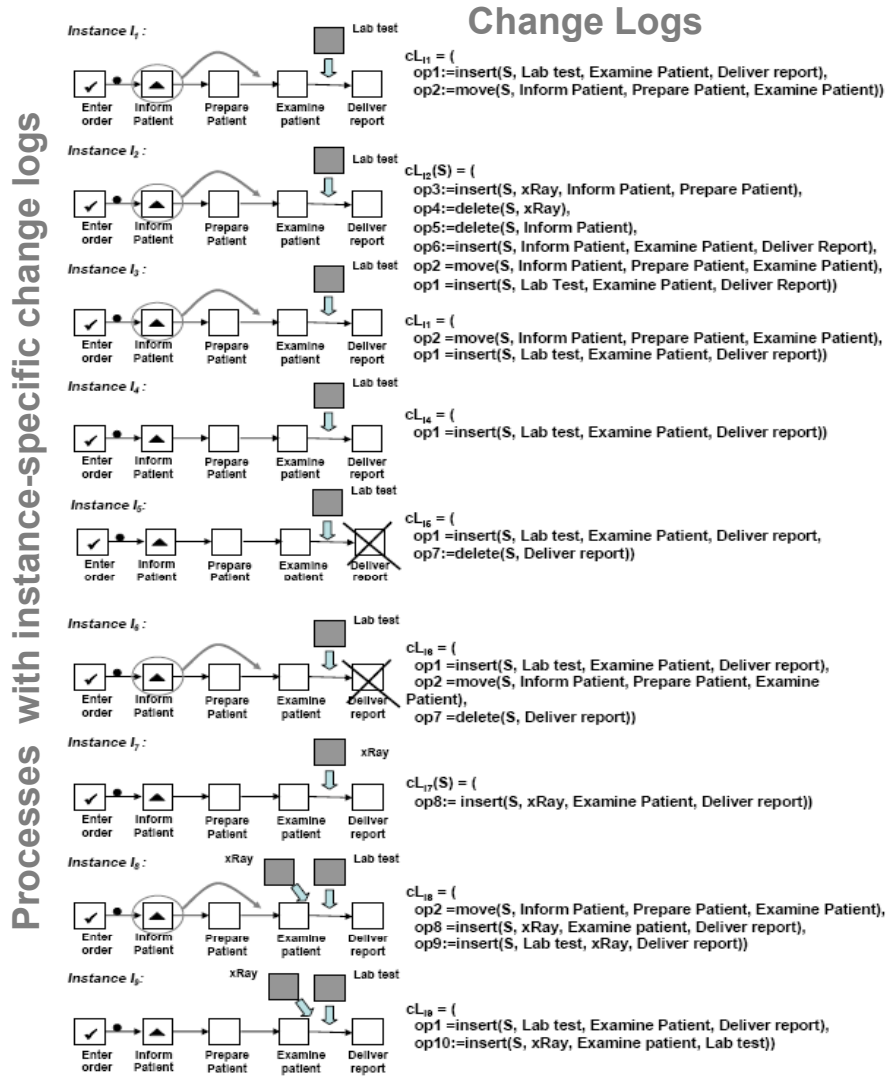


# Reuse Process Adaptations in the Large through Learning

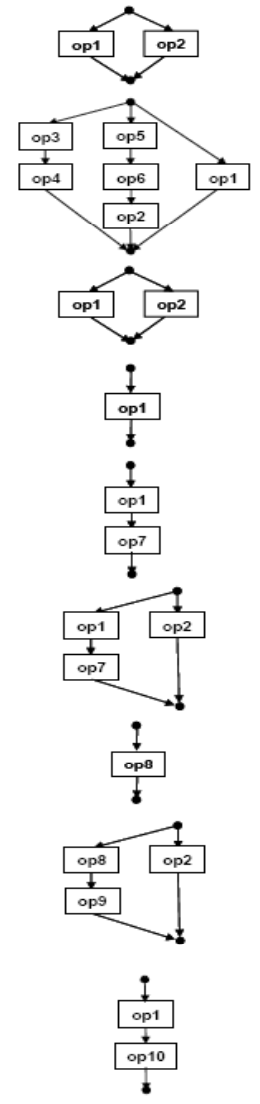
## Execution & Change Analysis



# Change Analysis: Applying Process Mining Techniques to Change Logs

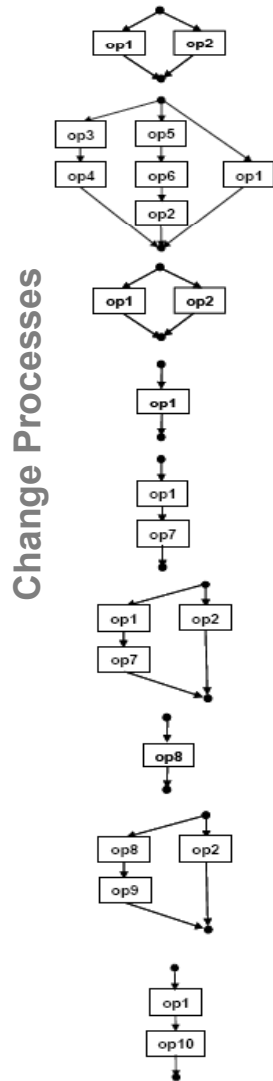


Phase I

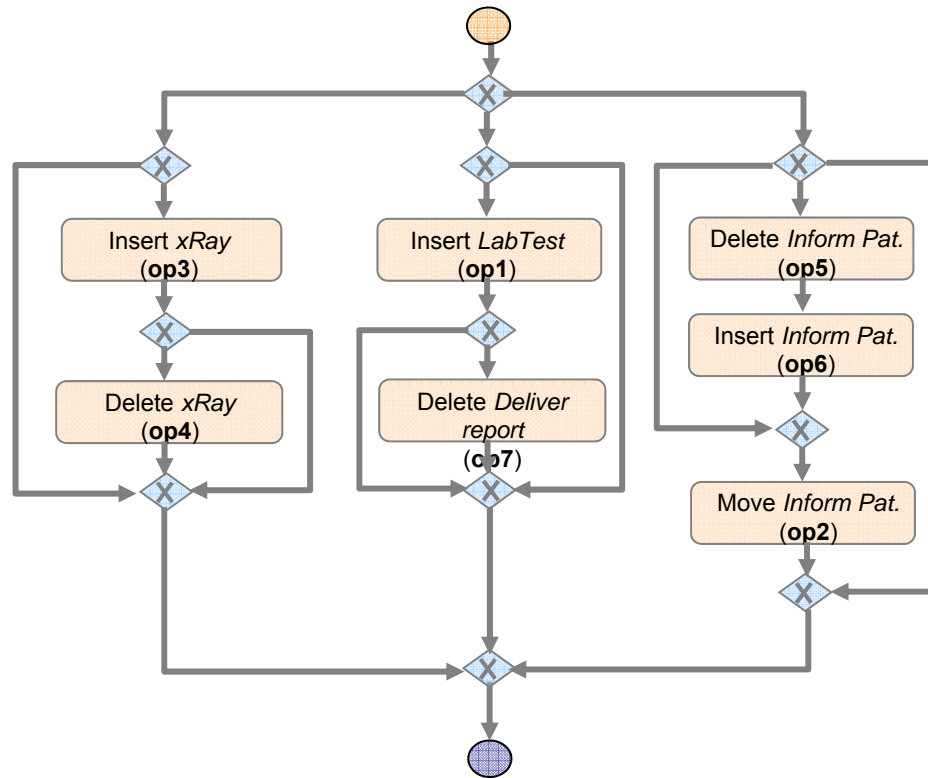




# Change Analysis: Applying Process Mining Techniques to Change Logs



Phase II



*The discovered meta change process covers all changes applied to at least one of the given fluid process instances.*

# Change Analysis: Applying Process Mining Techniques to Change Logs

The screenshot displays the ProM software interface. On the left, a sidebar titled "ProM import" shows a list of filters under "Choose filter:". The "Adept Demonstrator" filter is selected. Below the filter list, the memory usage is shown as "4.1 / 5.6 MB".

The main window, titled "Results - Settings for mining File", displays a process flowchart. The flowchart starts with "Process Start 9.0" and branches into three paths:

- Path 1: "Process Start 9.0" (40) → "INSERT x-Ray complete 4.0" (1.0) → "DELETE x-Ray complete 1.0" (1.0) → "Process End 9.0" (1.0)
- Path 2: "Process Start 9.0" (8.0) → "INSERT Lab test complete 8.0" (2.0) → "DELETE Deliver report complete 2.0" (2.0) → "Process End 9.0" (2.0)
- Path 3: "Process Start 9.0" (4.0) → "DELETE Inform patient complete 1.0" (1.0) → "MOVE Inform patient complete 5.0" (5.0) → "Process End 9.0" (5.0)

There are also cross-connections between paths, such as from "DELETE x-Ray complete 1.0" to "DELETE Deliver report complete 2.0" (3.0) and from "DELETE Deliver report complete 2.0" to "DELETE Inform patient complete 1.0" (8.0).

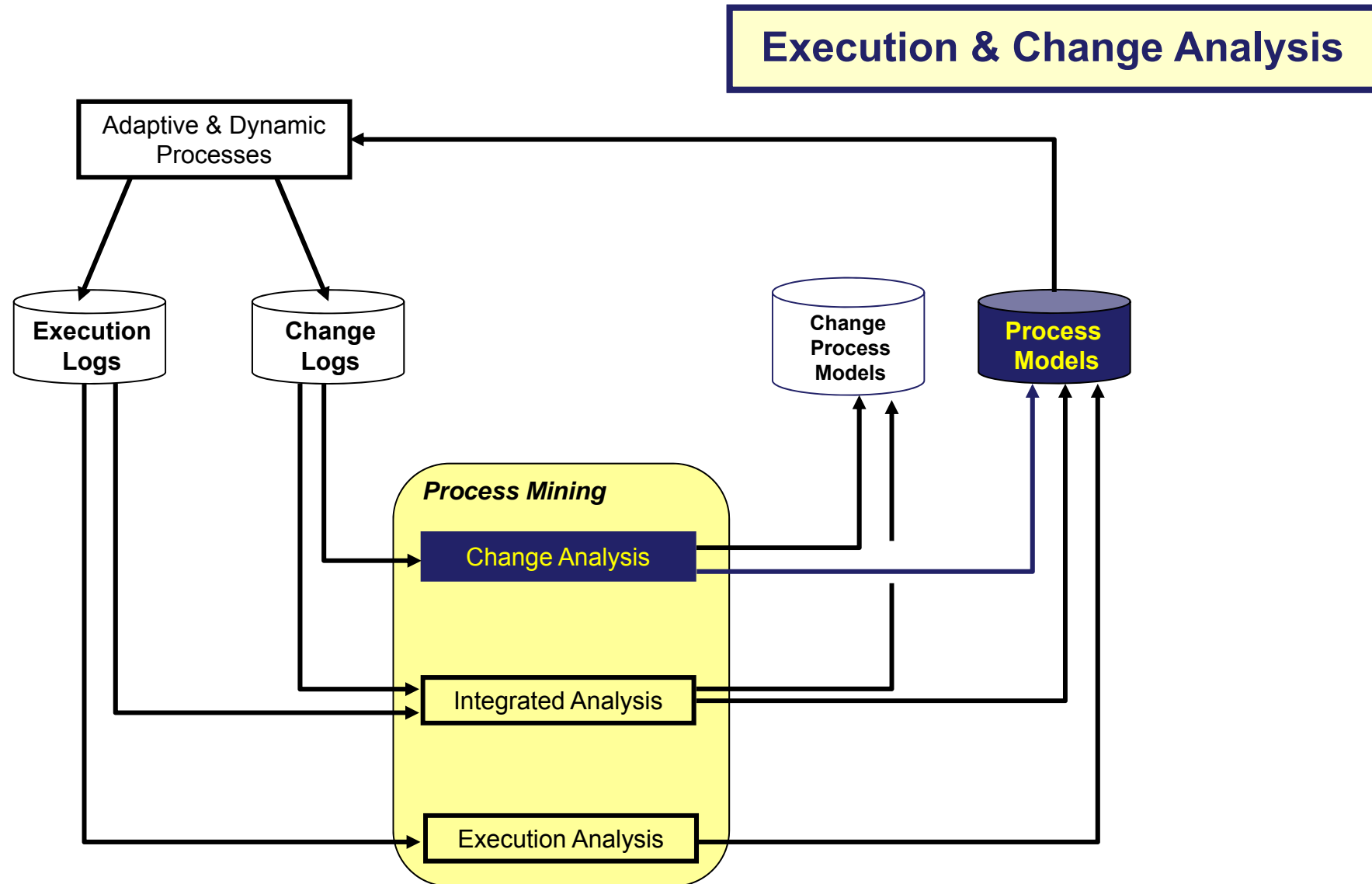
A blue callout bubble points to the flowchart with the text "Change Mining Plugin in ProM". A green callout bubble points to the filter sidebar with the text "Change Logs Imported from ADEPT".

The bottom of the window shows a log window with the following text:

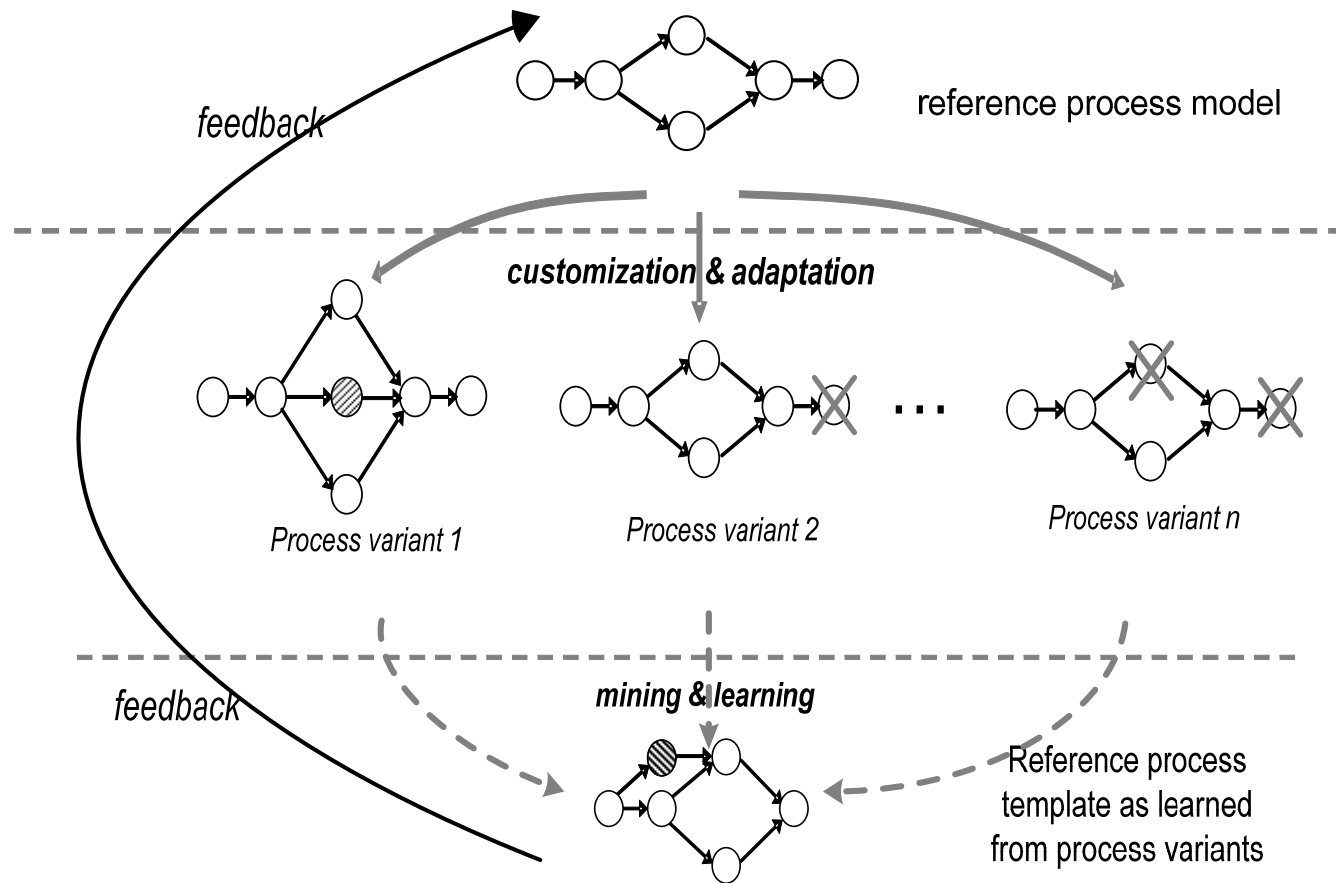
```
* Starting main UI
ProM Framework Initialized
Start process mining.
Mining duration: 391 milliseconds
Process mining finished.
```

At the bottom right, there are buttons for "Normal", "Warning", "Error", and "Debug". The zoom level is set to "Zoom: 127 %".

# Reuse Process Adaptations in the Large through Learning



# Discovering a Reference Process Model by Mining Process Variants



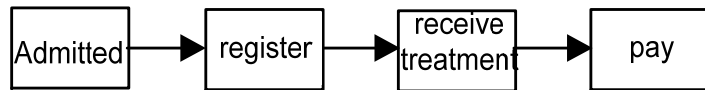
## Discovering a Reference Process Model by Mining Process Variants

### Basic Goal:

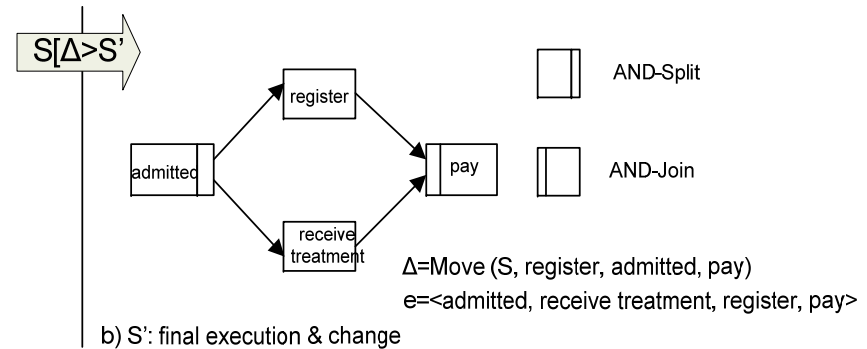
How to *discover a reference process model*  
by mining a collection of *process (instance) variants*  
in order to  
*reduce the amount of future process adaptations?*

# Discovering a Reference Process Model by Mining Process Variants


## Bias and Distance



a) S: original process model



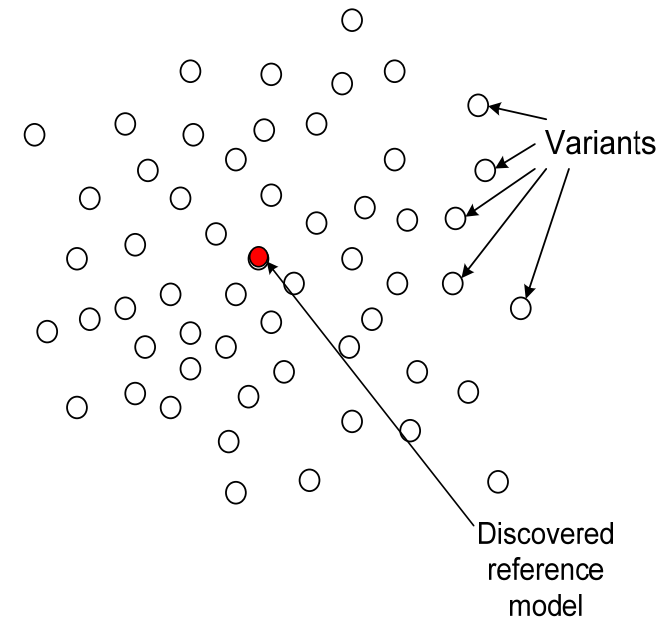
### □ Important measures:

- bias
- change distance  measure the complexity of process changes

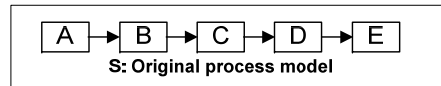
## Discovering a Reference Process Model by Mining Process Variants

### Reformulated Basic Goal:

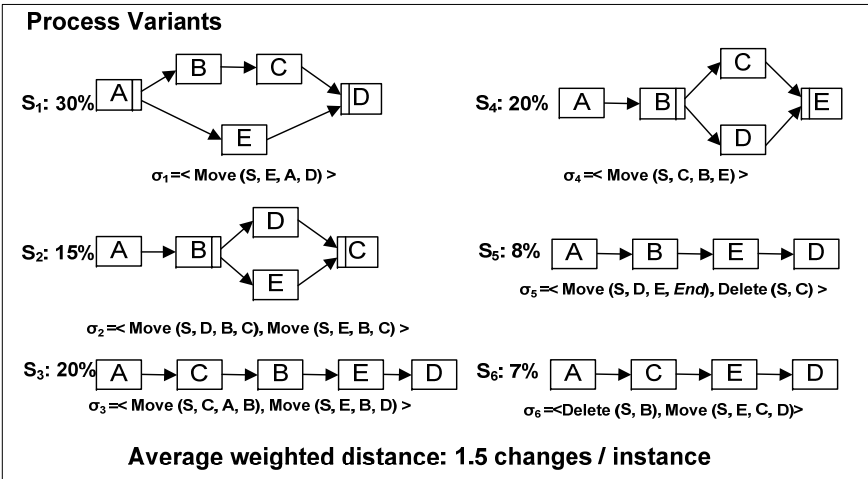
How to *mine a collection of process variants* such that the *discovered reference process model* has *minimal average distance to the variants*?



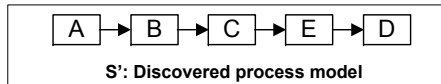
# Process Variants Mining: A Cluster-based Approach



↓ Process Adaptations



↓ Change Mining



↓

Biases after process evolution	
I <sub>1</sub> : 30%	$\sigma'_1 = \langle \text{Move}(S', E, A, D) \rangle$
I <sub>2</sub> : 15%	$\sigma'_2 = \langle \text{Move}(S', D, B, C), \text{Move}(S', E, B, C) \rangle$
I <sub>3</sub> : 20%	$\sigma'_3 = \langle \text{Move}(S', C, A, B) \rangle$
I <sub>4</sub> : 20%	$\sigma'_4 = \langle \text{Move}(S', C, B, E) \rangle$
I <sub>5</sub> : 8%	$\sigma'_5 = \langle \text{Delete}(S', C) \rangle$
I <sub>6</sub> : 7%	$\sigma'_6 = \langle \text{Delete}(S', B) \rangle$

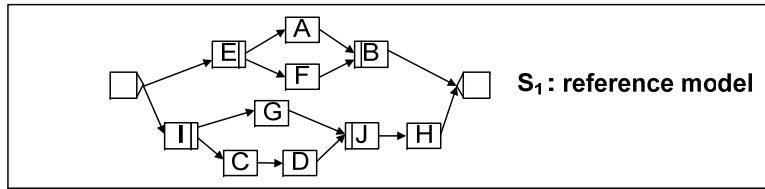
**Average weighted distance: 1.15 change/ instance**



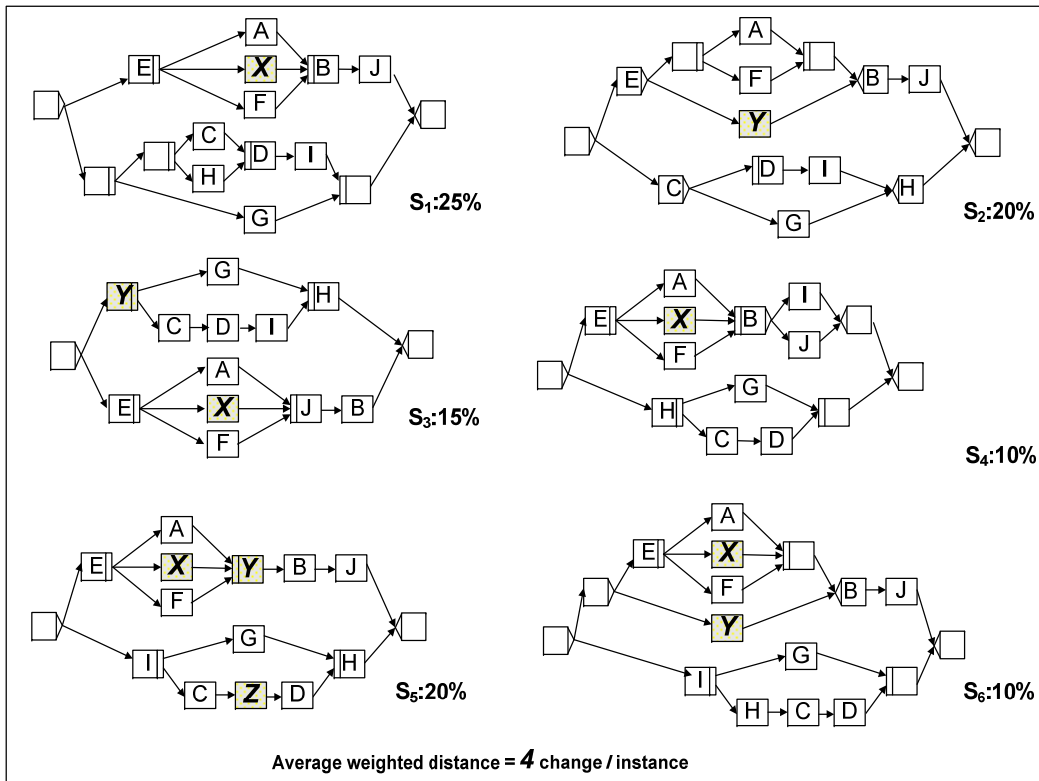
## Process Variants Mining: A Cluster-based Approach

- ❑ This 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 costly
    - A spaghetti-like process structure may result
  - 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.

# Process Variants Mining: A Heuristic Approach



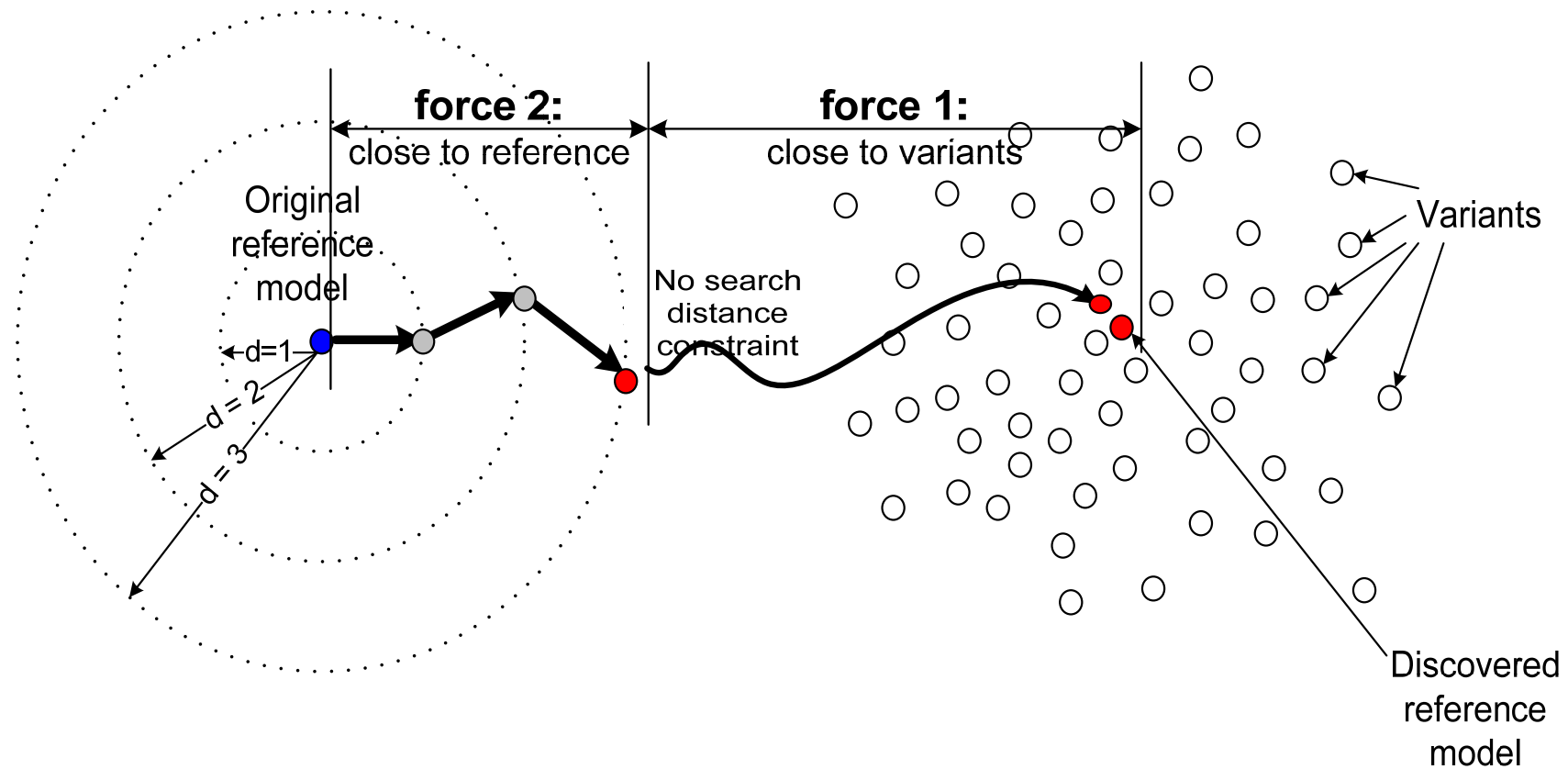
↓  
Process configuration



**Possible starting point:**

*Can we find a model closer to the variants by applying at maximum **two changes** to the old reference model?*

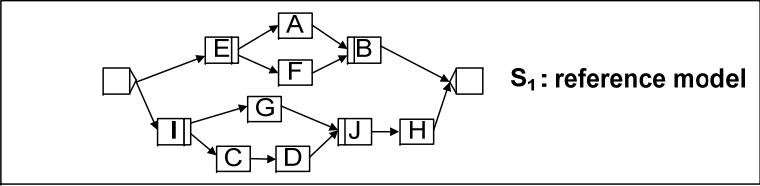
## Process Variants Mining: A Heuristic Approach



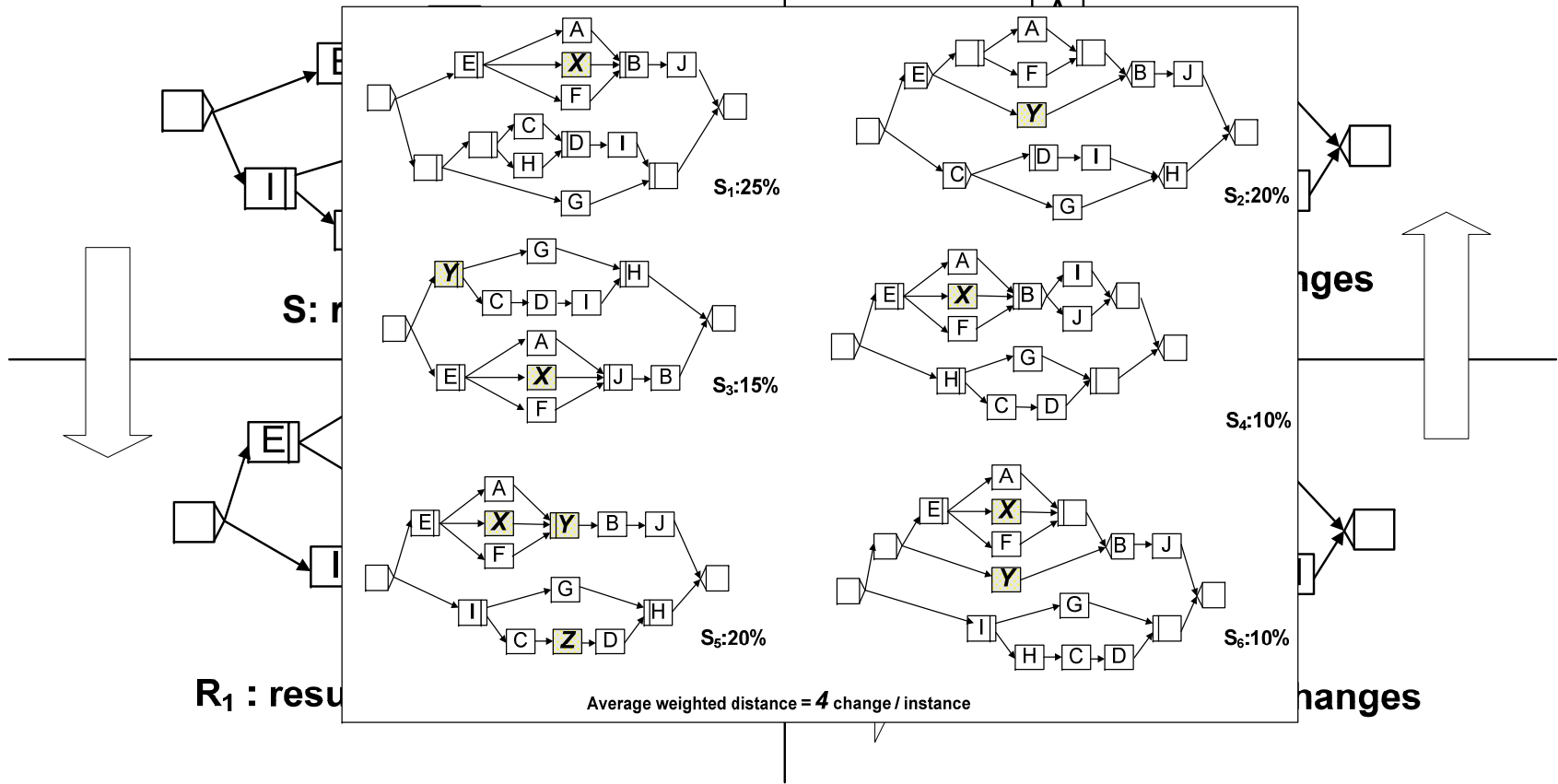


# Process Variants Mining: A Heuristic Approach

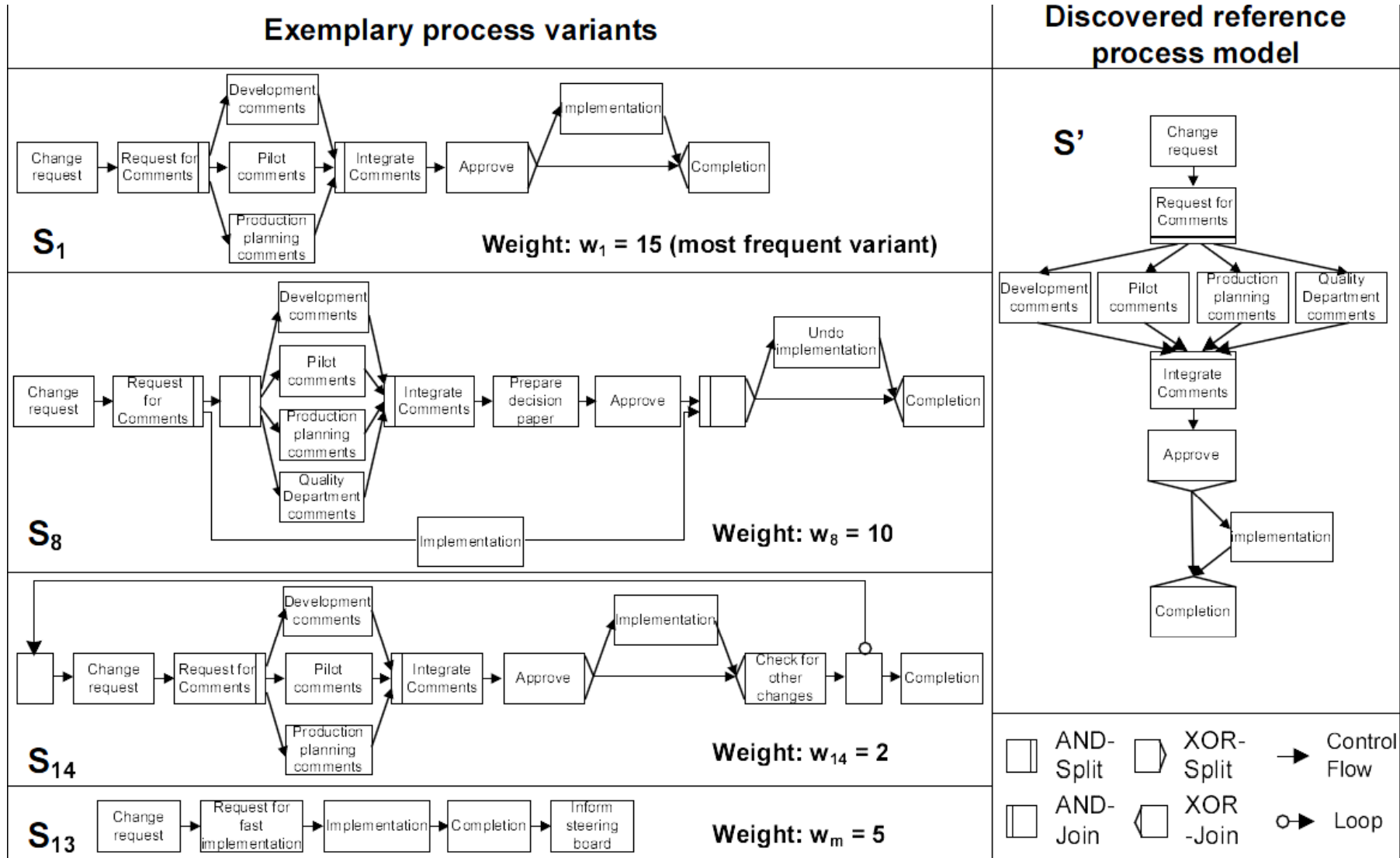
**Example:**



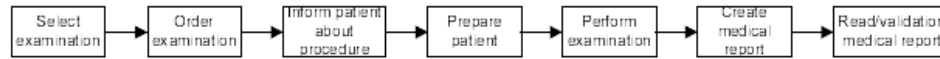
↓  
Process configuration



# Process Variants Mining: Automotive Case Study

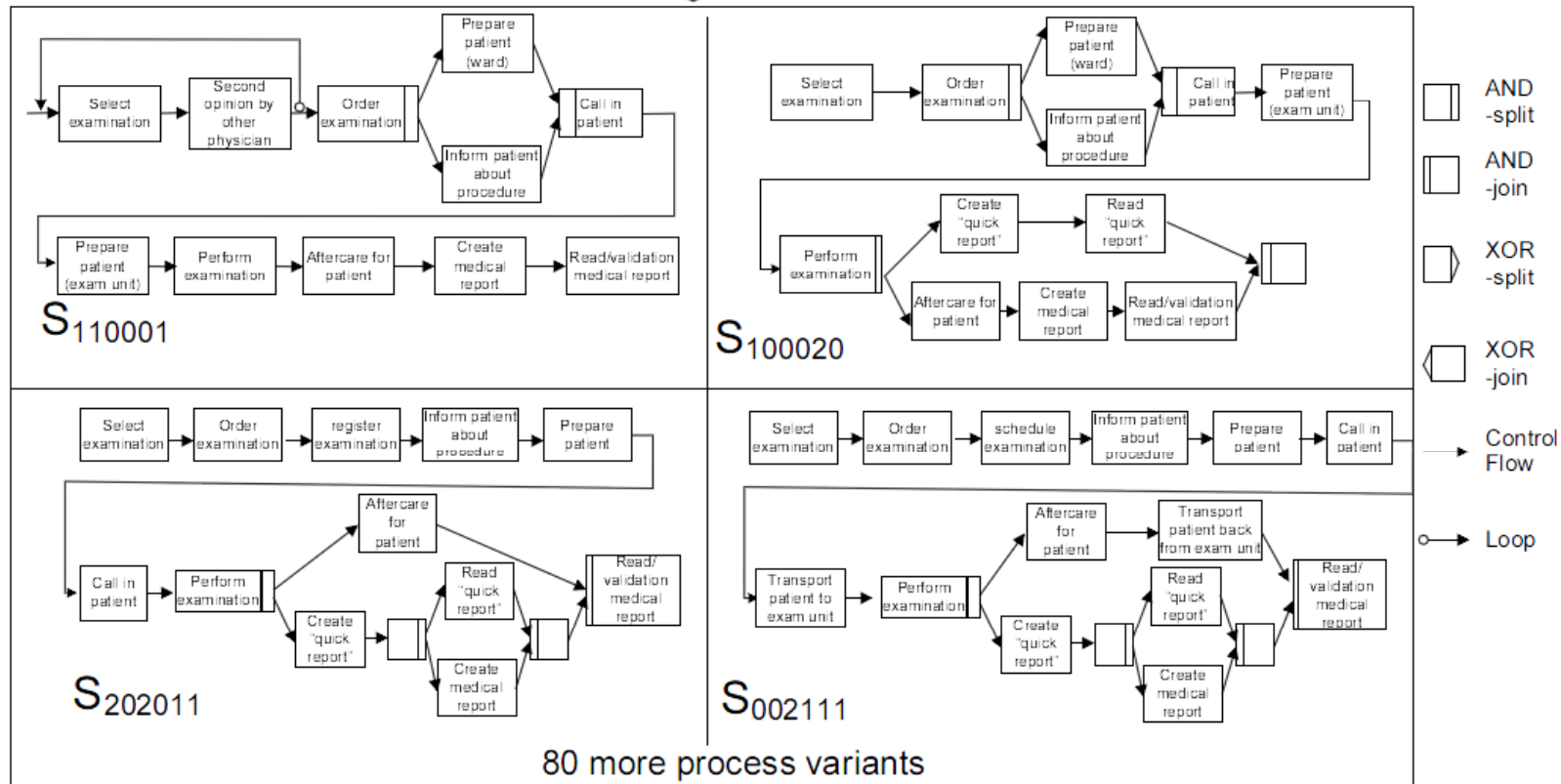


# Process Variants Mining: Healthcare Case Study

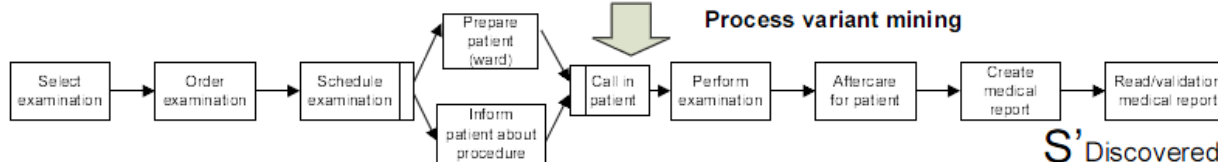


**S** Original reference model

Process configuration



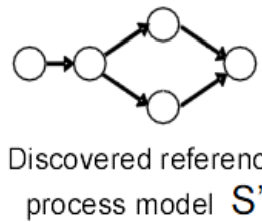
Process variant mining



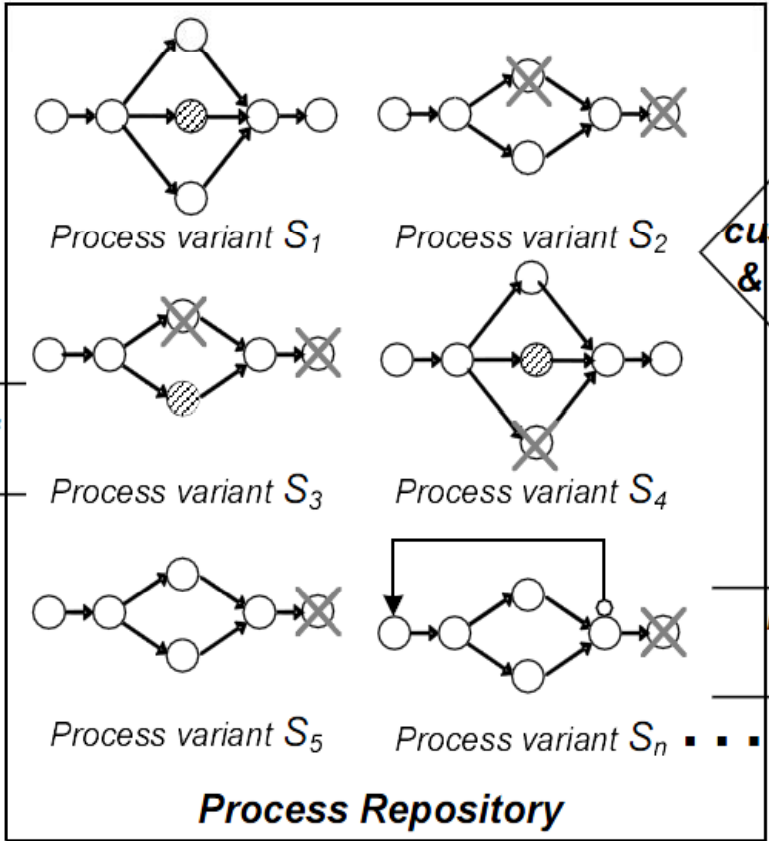
**S'** Discovered reference model

# Process Variants Mining: Scenarios

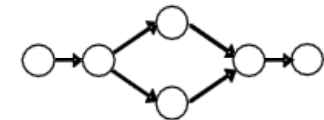
*Scenario 1: No original reference process model available*



**mining & learning**

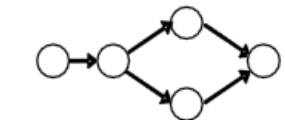


*Scenario 2: Original reference process model known*



Original reference process model  $S$

**Process improvement**



Discovered reference process model  $S'$

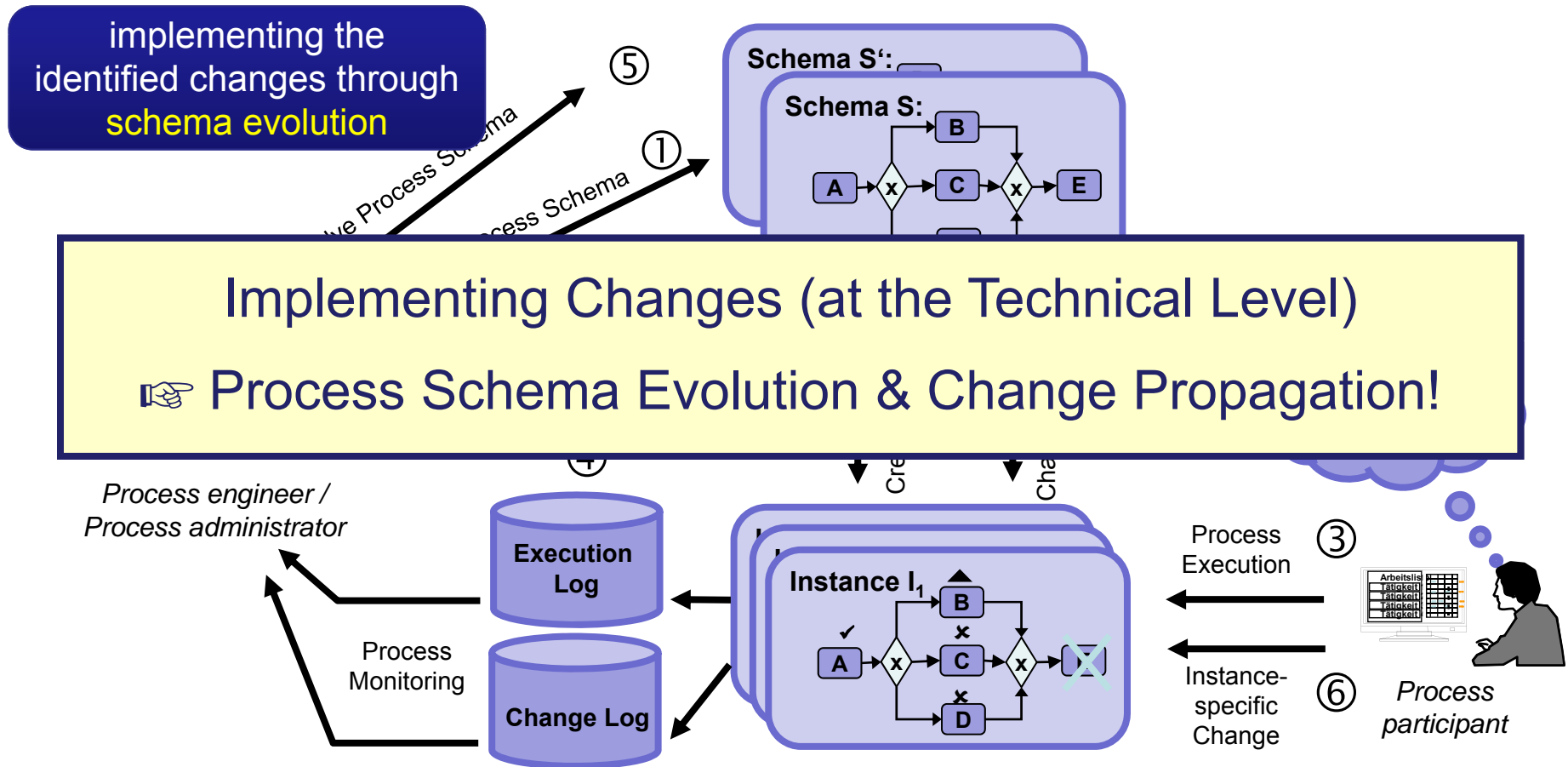
**customization & adaptation**

**mining & learning**

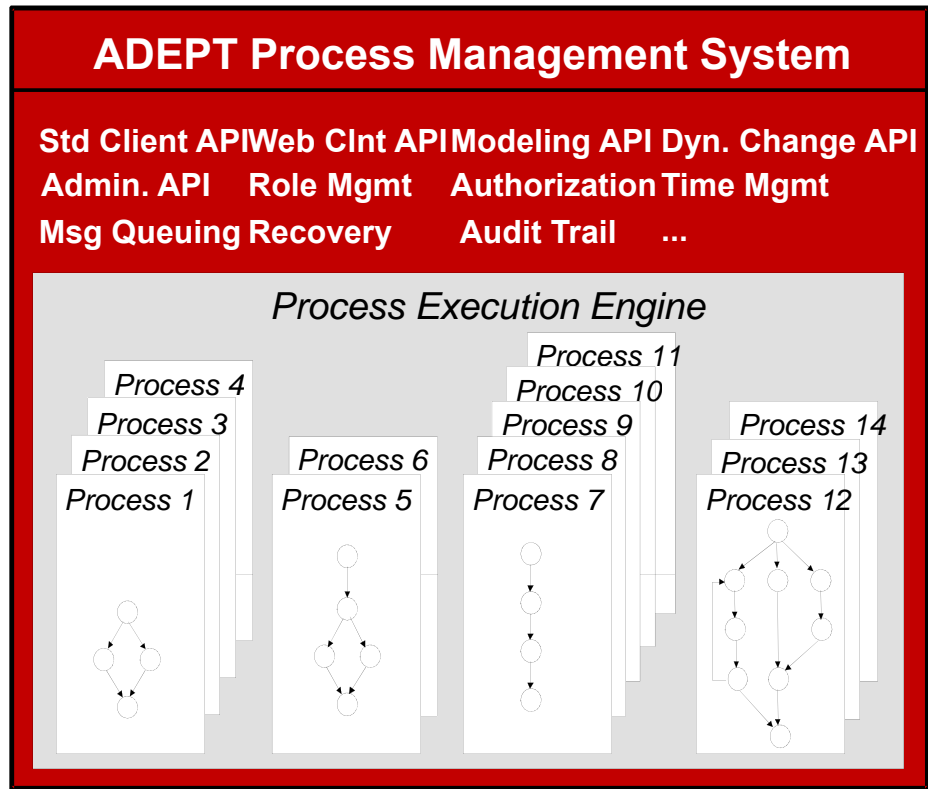
**Goal:** Discover a (new) reference process model which requires less configuration efforts



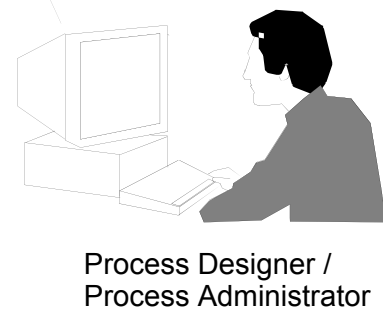
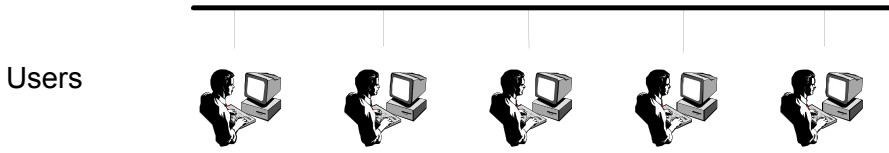
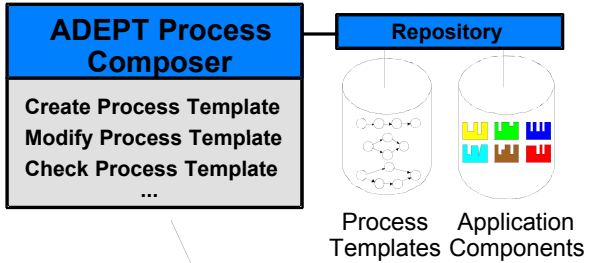
# Introduction: Lifecycle Support for Dynamic Processes



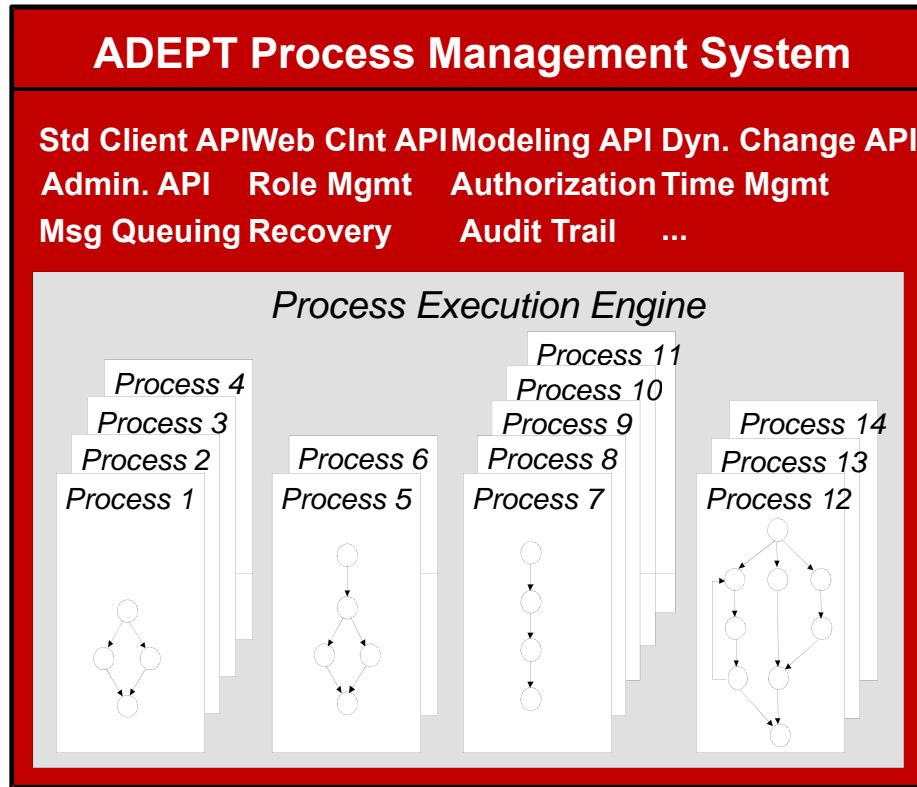
# Implementing Process Changes through Schema Evolution



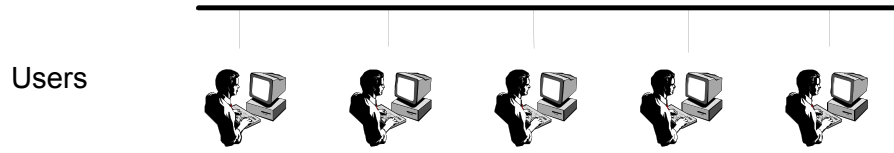
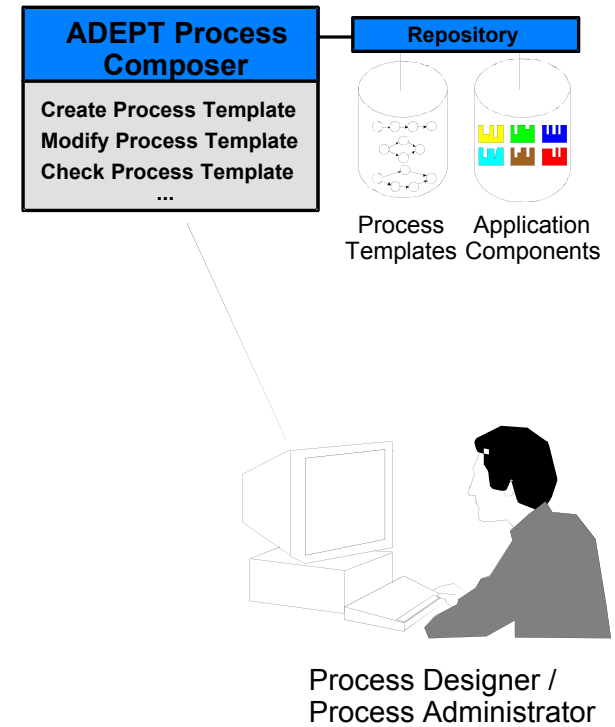
**Need to Change a Business Process**



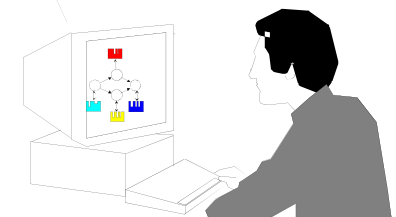
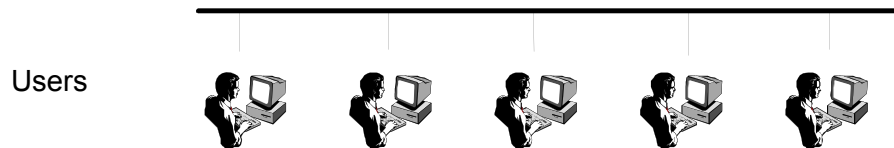
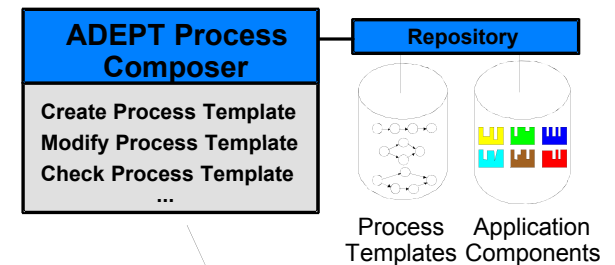
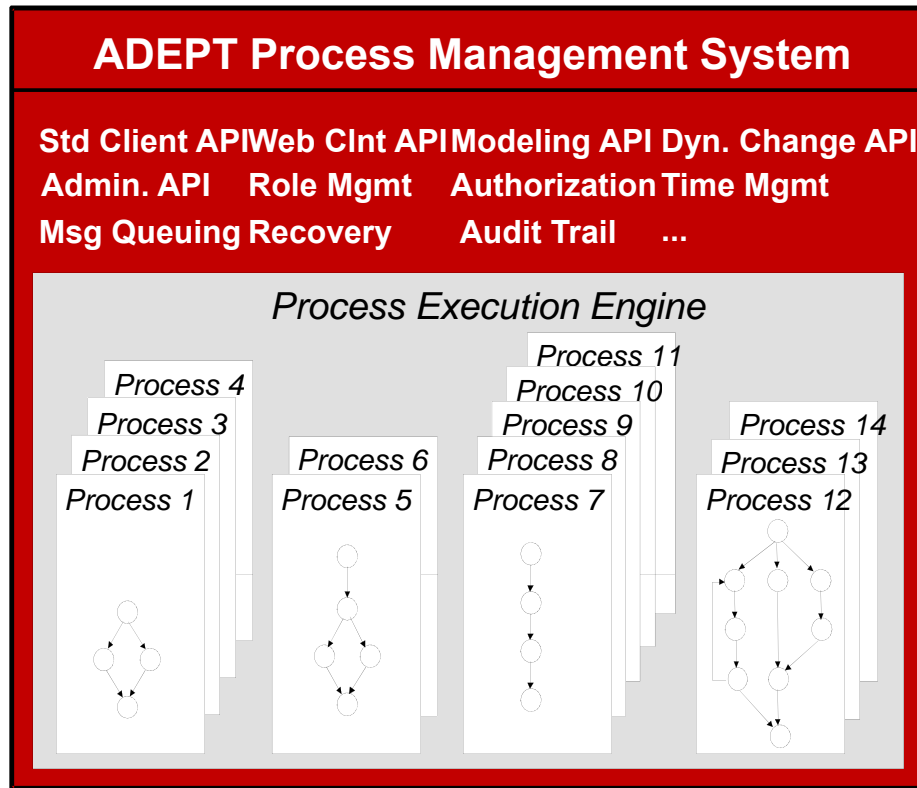
# Implementing Process Changes through Schema Evolution



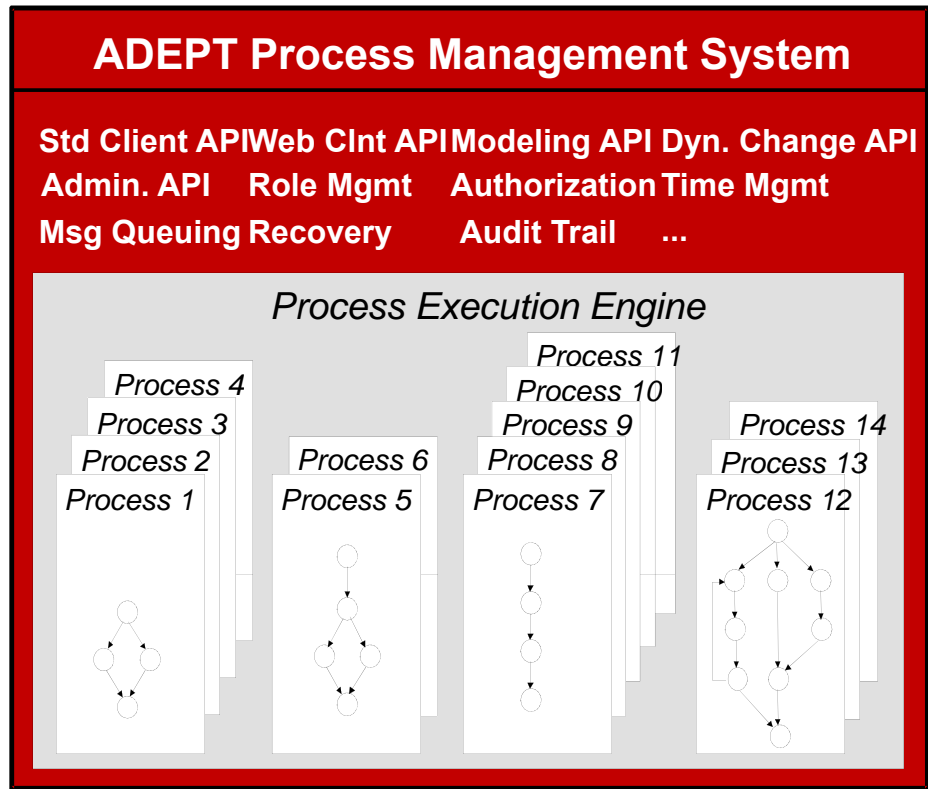
## Process Designer Checks Out "Active" Process Template



# Implementing Process Changes through Schema Evolution



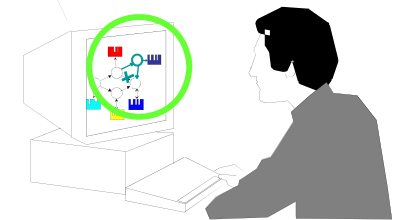
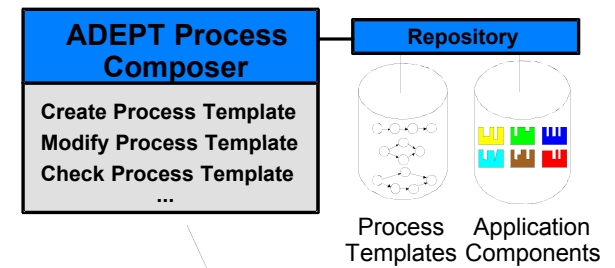
# Implementing Process Changes through Schema Evolution



Users

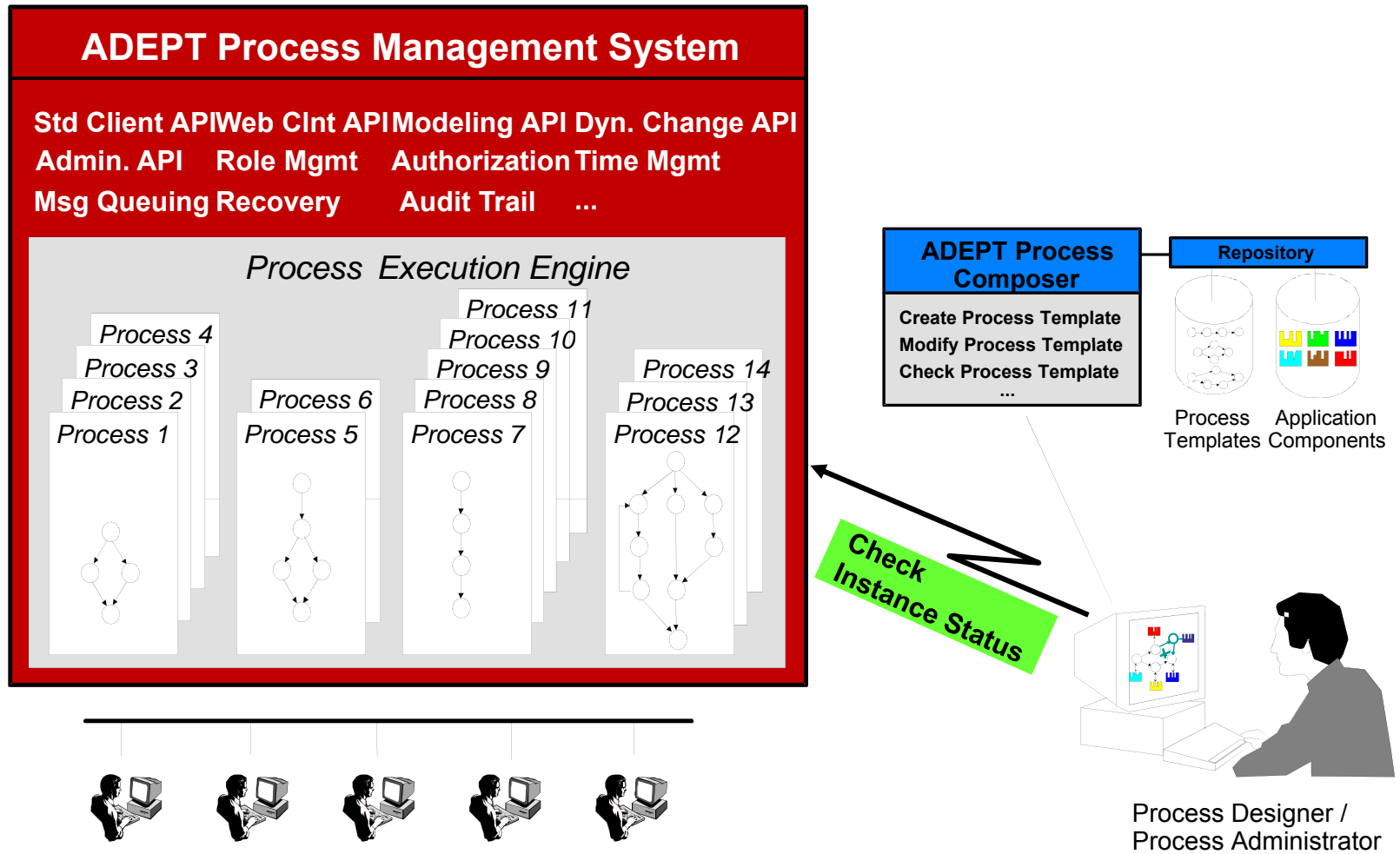


**Process Designer Performs Schema Changes**

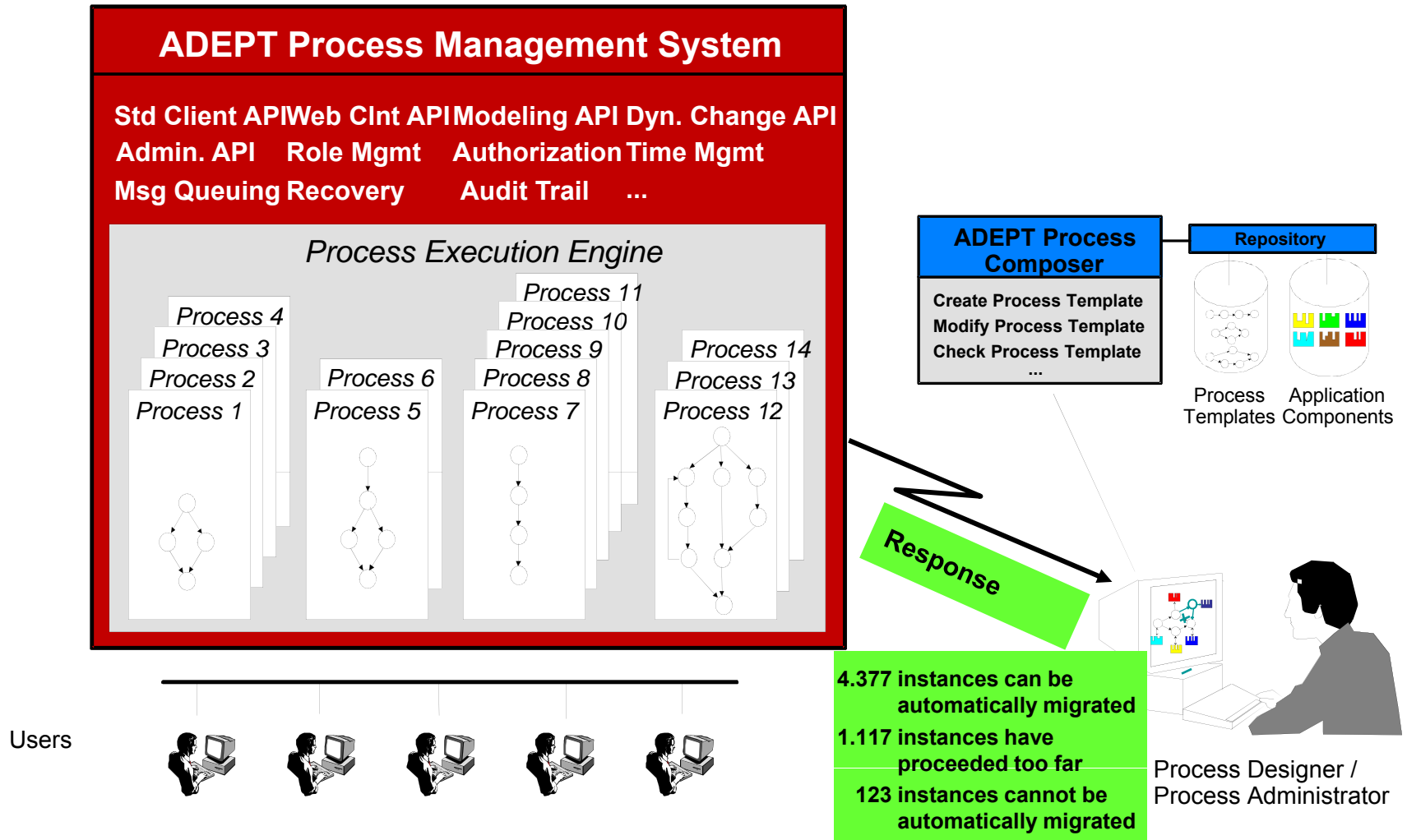


Process Designer /  
 Process Administrator

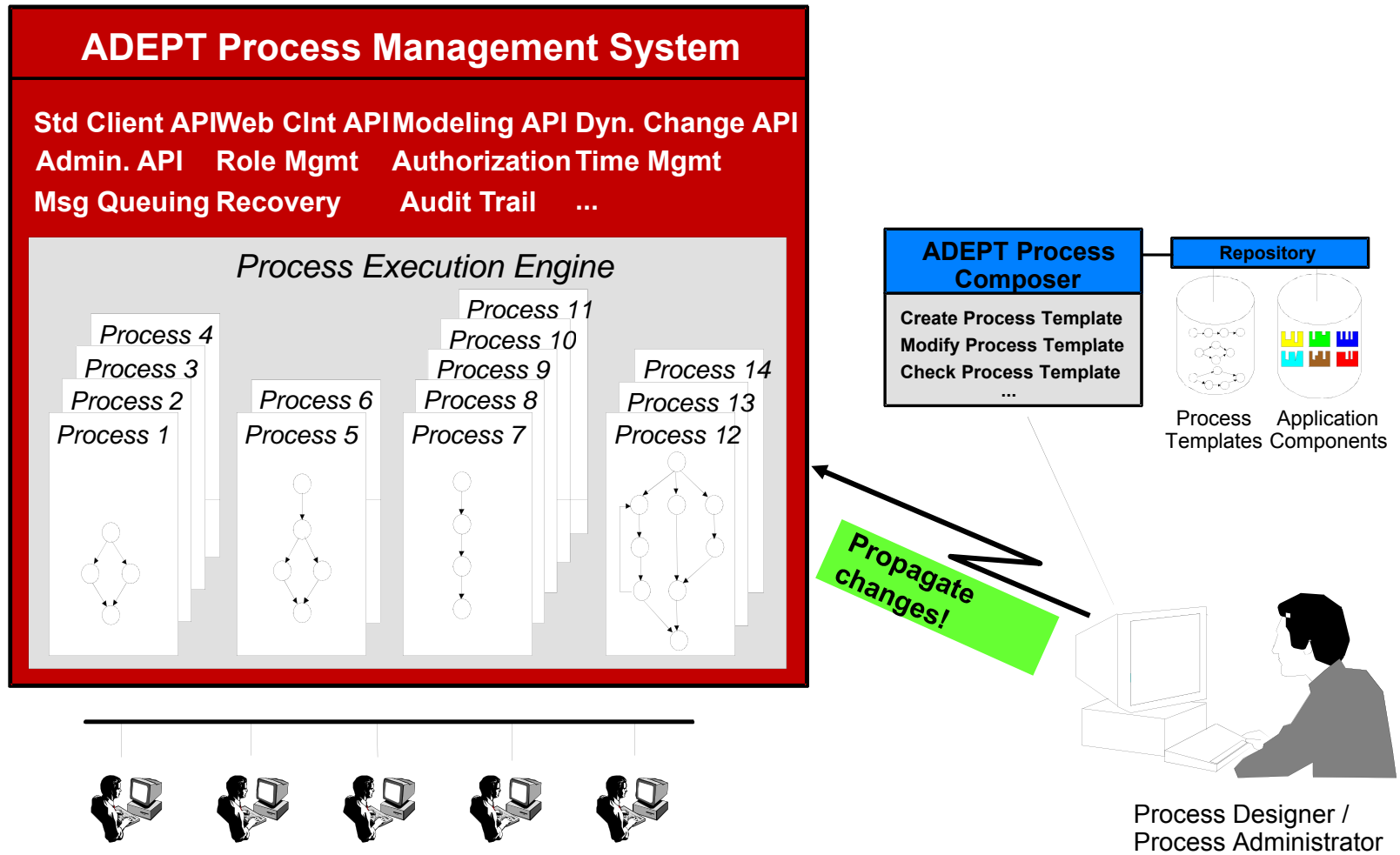
# Implementing Process Changes through Schema Evolution



# Implementing Process Changes through Schema Evolution

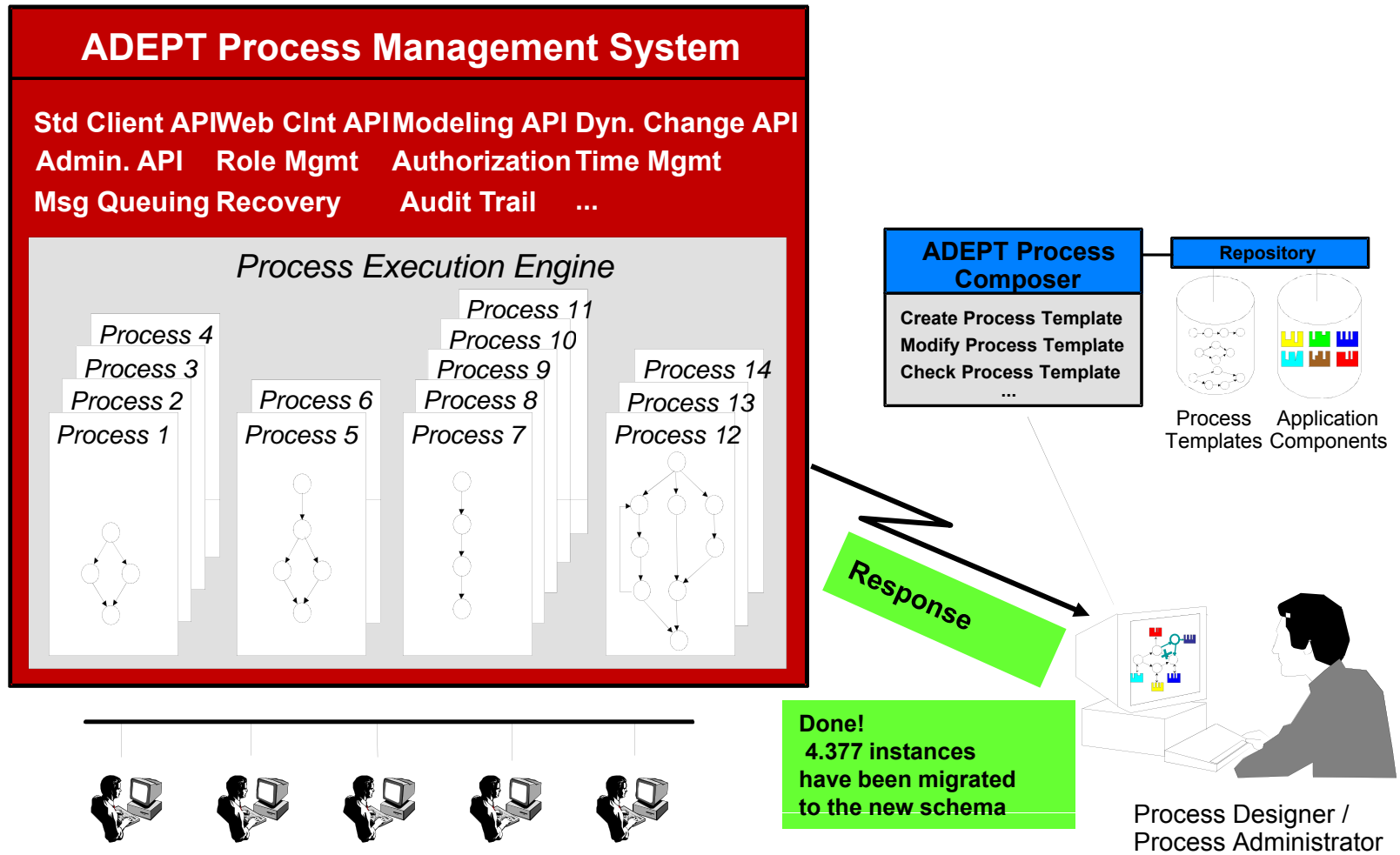


# Implementing Process Changes through Schema Evolution





# Implementing Process Changes through Schema Evolution





The image displays three overlapping windows from the AristaFlow software suite:

- AristaFlow Process Template Editor:** Shows a BPMN diagram with a task 'Fill out Order Form' and an event 'XOR Predicate'. The left sidebar lists an activity repository with various reusable components like 'de.aristaflow.db.SQL' and 'de.aristaflow.form.Form'. The bottom pane shows 'Node Basics' for the 'Fill out Order Form' task.
- AristaFlow Test Client:** A window for testing process instances. It features a toolbar, a table of active instances, and an 'Approve' dialog box. The dialog shows input data for parameters: 'Article\*' (STRING, 'Laptop'), 'Motivation\*' (STRING, 'Old one is too slow...'), and 'Price\*' (INTEGER, '1.500').
- AristaFlow-Klient - supervisor (supervisor):** A web-based interface for process supervision. It shows a task list for 'Arbeitsbereich' with one task 'Aufgaben (1)'. The main area displays a form titled 'Receive customer request and collect data (FORM)'. The form includes sections for 'Customer Data' (name, street, city) and 'Customer Request' (product, quantity). The 'Requested product' field contains 'The Hitchhiker's Guide to the Ge...'. Buttons for 'Confirm', 'Suspend', 'Reset', and 'Fail and discard' are visible at the bottom.