DBIS EXPERIENCE
processes for a flexible world

ulm university universität ulm

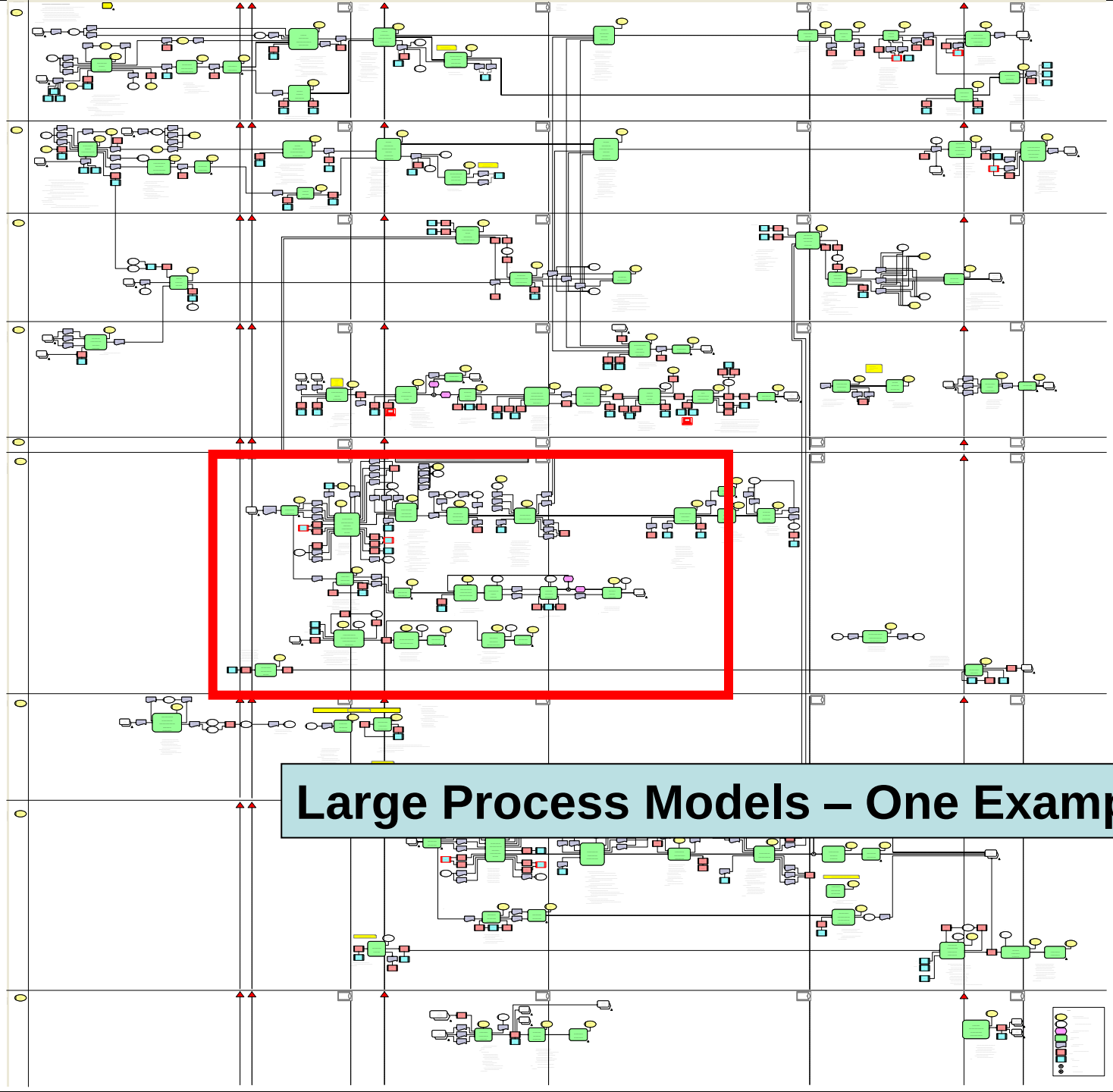
Visualizing Large Business Process Models

Challenges, Techniques, Applications

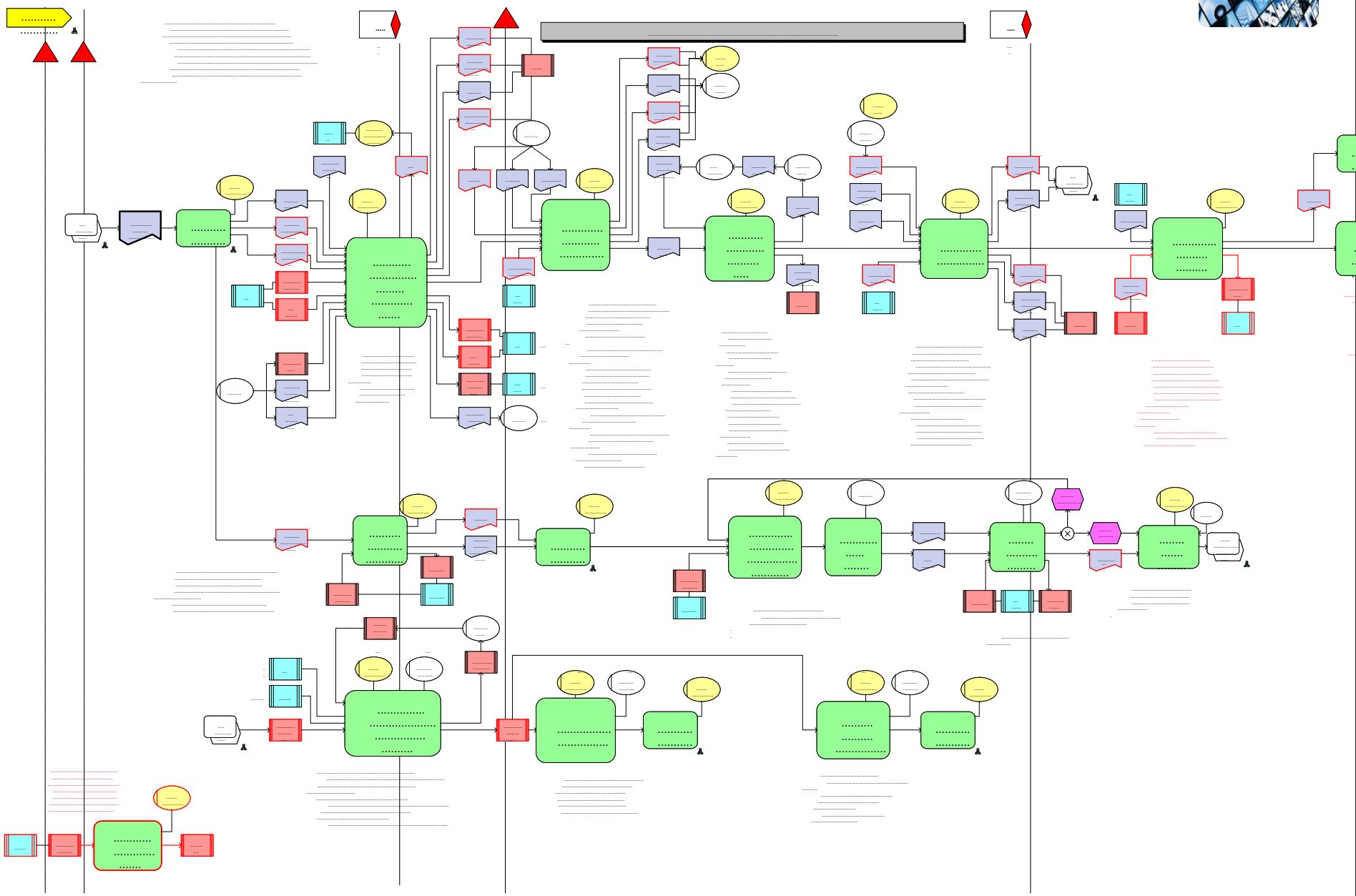
Prof. Dr. Manfred Reichert

Agenda

- Motivation
- Abstracting Process Models
- Adapting Visual Appearance
- Apdapting Display Form
- Outlook
  - Navigating in Complex Business Processes
  - Updatable Process Model Abstractions
  - Gesture-based Interaction with Process Models

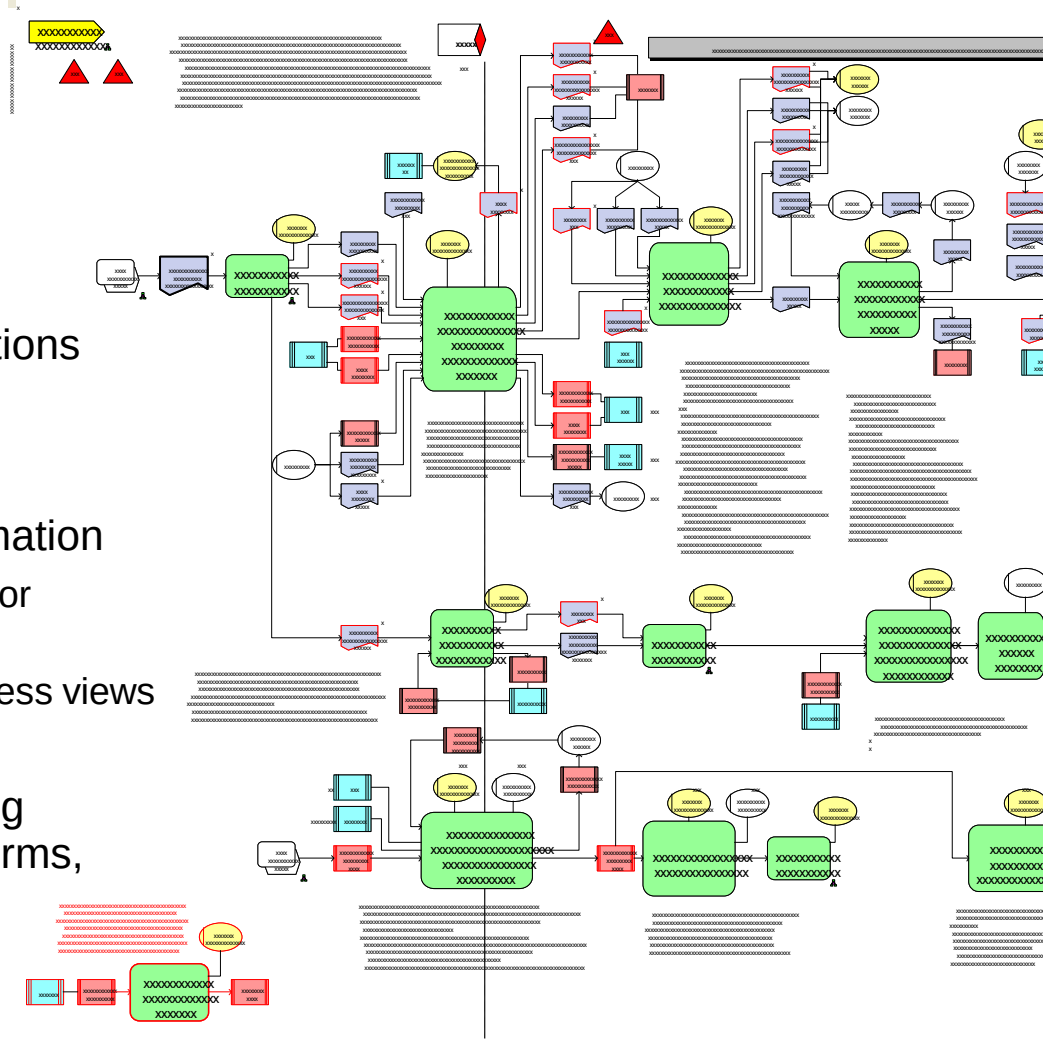

Large Process Models – One Example!

Motivation

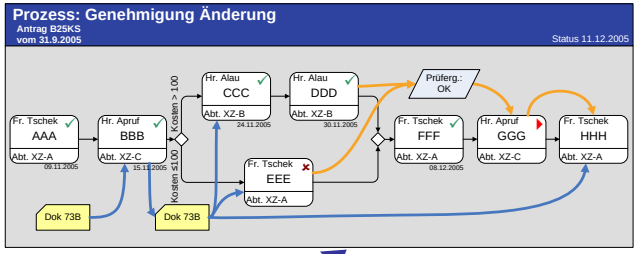


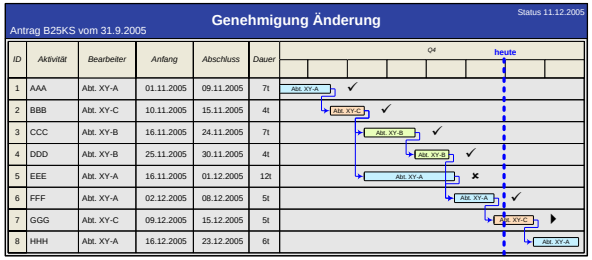
Motivation

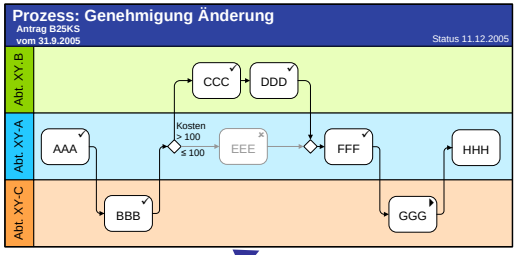
- Problems of this process diagram:
  - Too many symbols
  - Too many different symbols
  - ► Affects comprehensibility!
- Challenges:
  - Enable configurable process notations
    - form & color
    - displayed attributes
  - Enable configurable level of information
    - Abstract from process information or eliminate unnecessary one
    - Derive role- / context-specific process views (i.e., process model abstractions)
  - Enable different forms of displaying process models (e.g., diagrams,forms, trees)



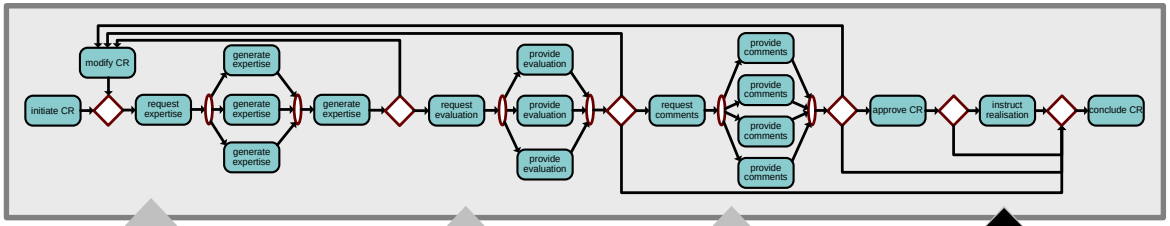
Process Visualization
What is needed?



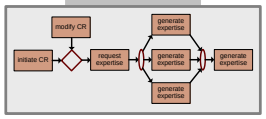




Visualiza omponent



direct import



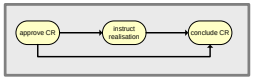
manual (remodelling)

black box
(legacy application)

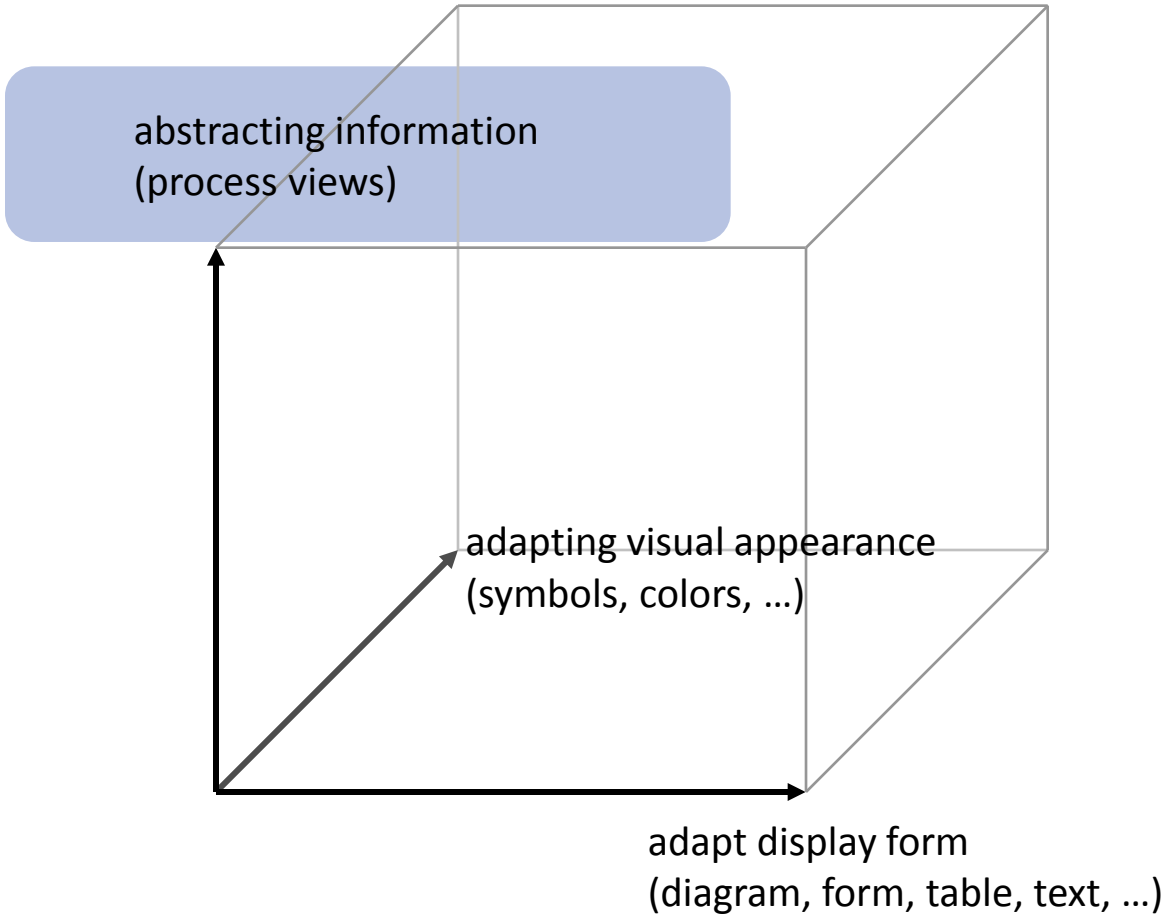
Mining

log-data

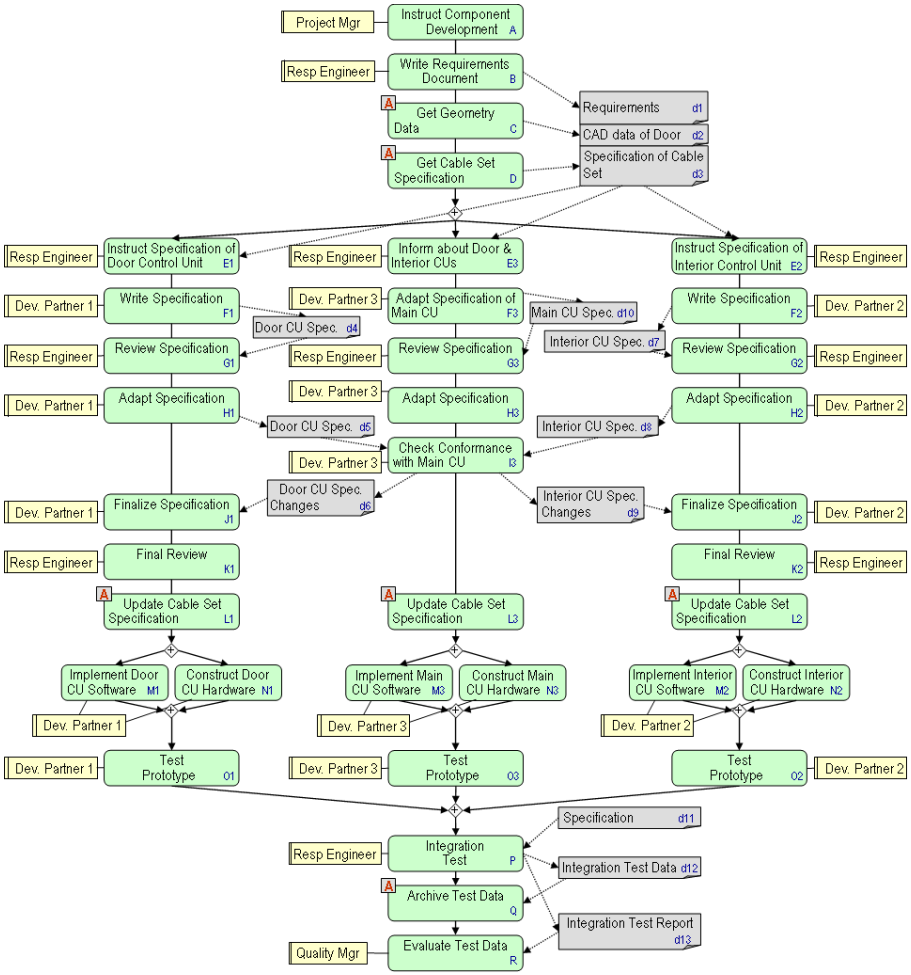
indirect

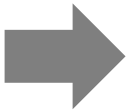


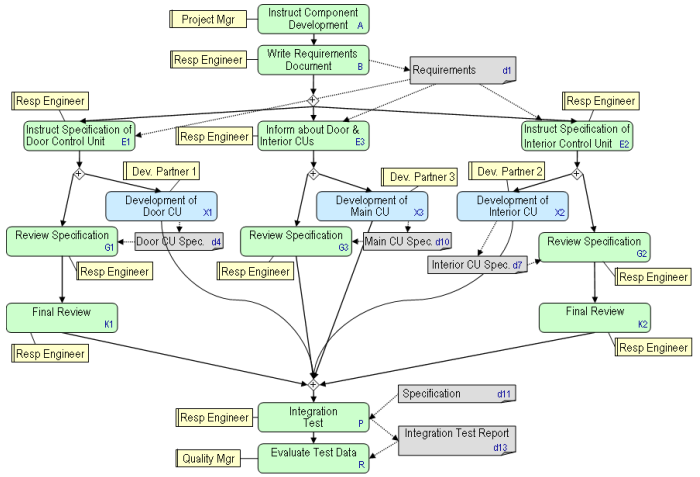
Process Visualization Dimensions



Process Visualization
Abstracting Process Models: Goals







Process Visualization
Abstracting Process Models: Goals

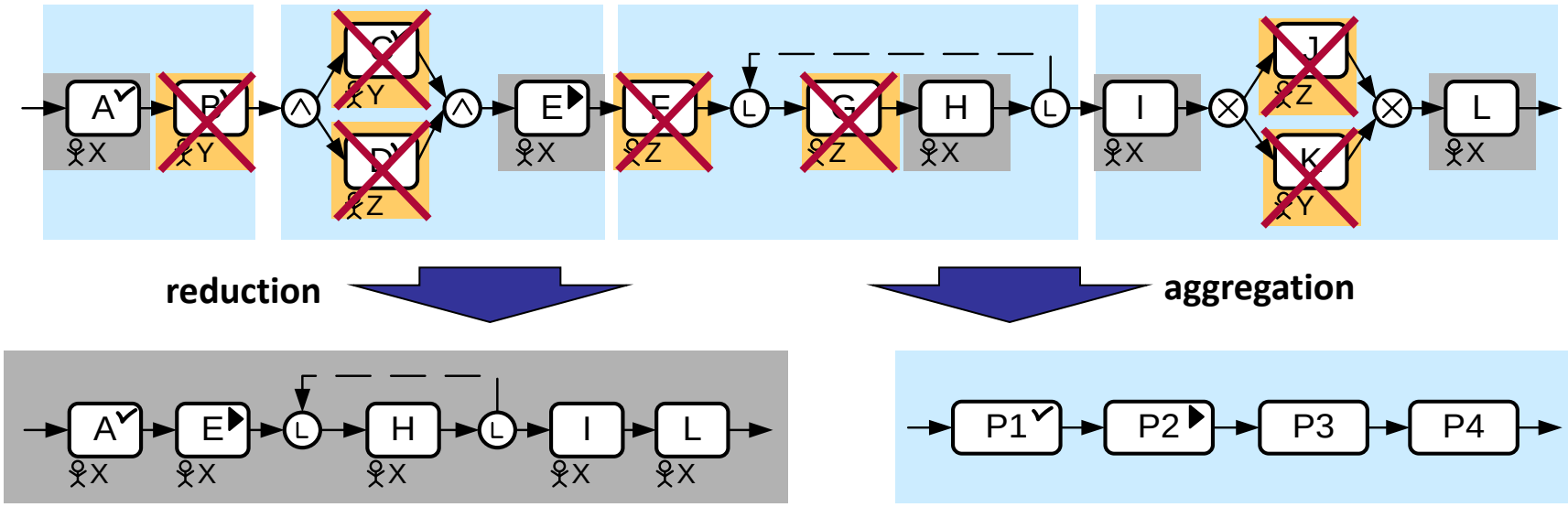
Goals:

- Decreasing the complexity of (large) process models
- Eliminating or abstracting process information

► **Personalize process models through process views**

- **Process views** should be …
  - easy to define (e.g., based on high-level operations)
  - dynamically built if required

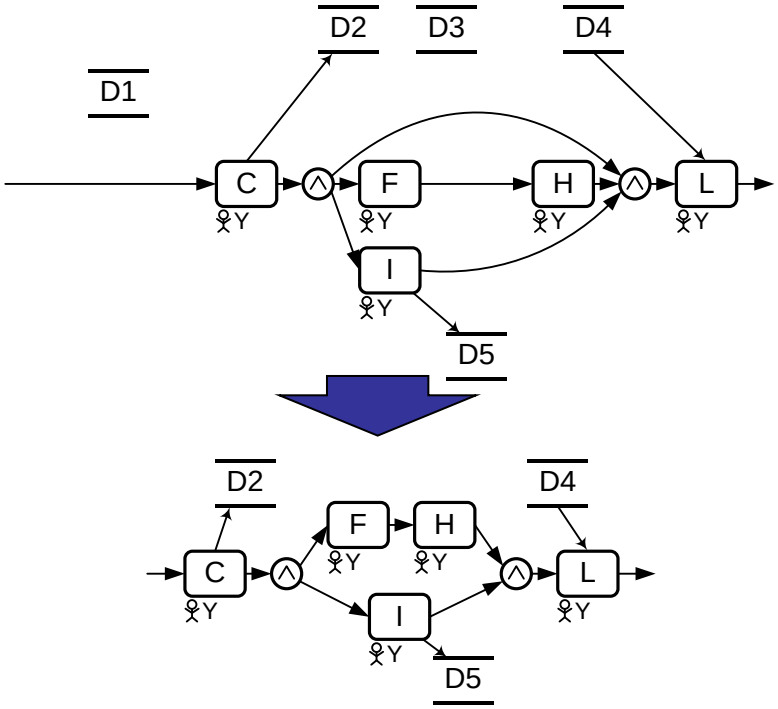
„Only show my activities!"
„Do not display technical activities"
„Aggregate completed parts"



Process Visualization

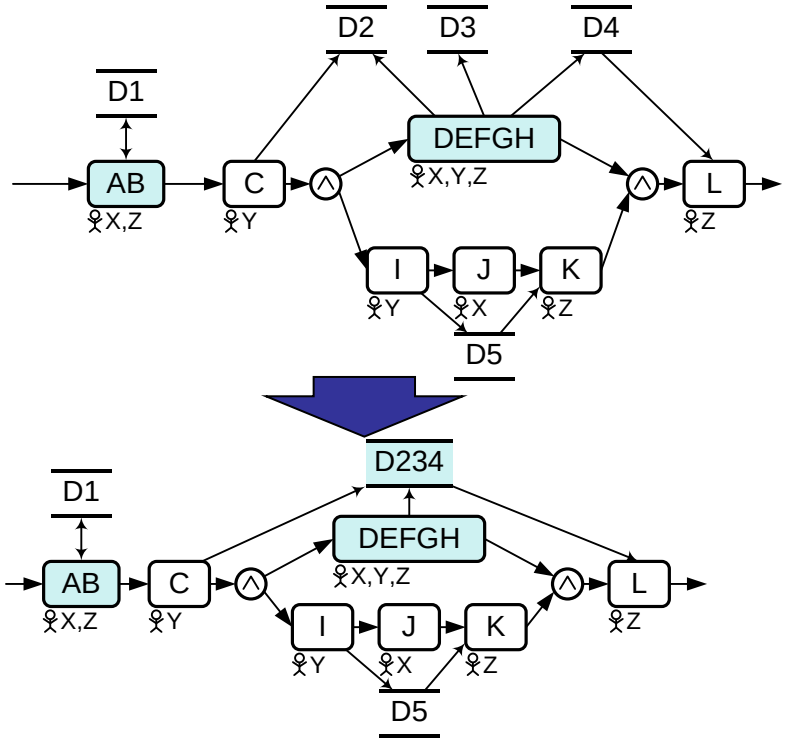
Abstracting Process Models: Fundamental Techniques

Reduction



- Eliminate activities
- Simplify the resulting schema
- Remove adjacent satellite objects

Aggregation

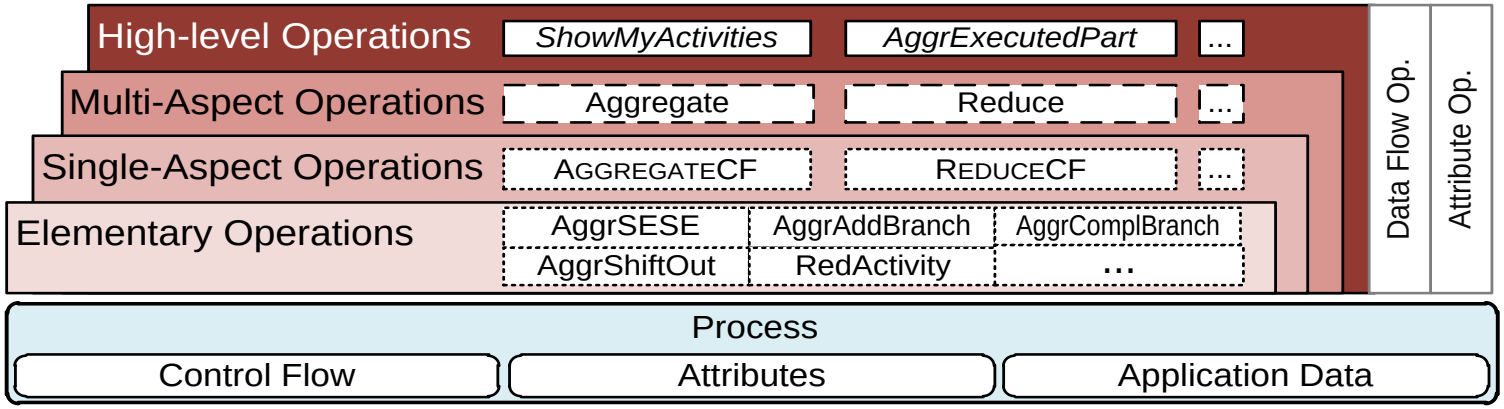


- Aggregate activities
- Aggregate adjacent objects if required

Process Model Abstraction and Process Views in Proviado

Process Visualization
Abstracting Process Models: The Proviado Approach

- A multi-layer approach for abstracting process models and building process views

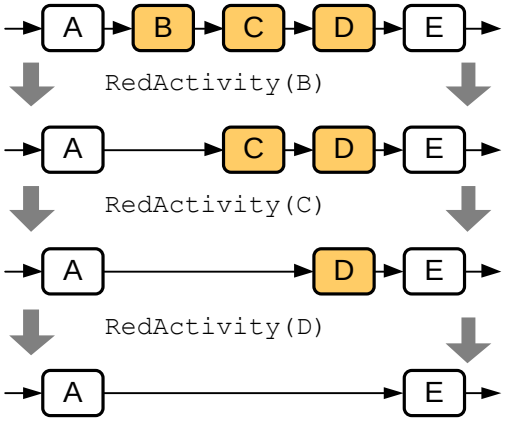


Process Visualization

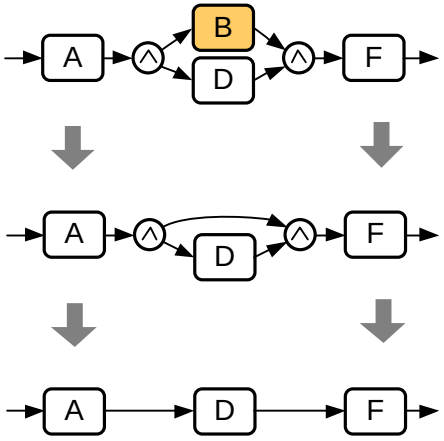
Abstracting Process Models: Elementary Operations

Elementary reduction operations:

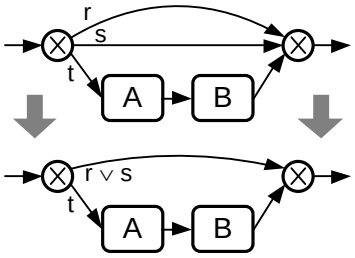
ReduceCF({B,C,D})

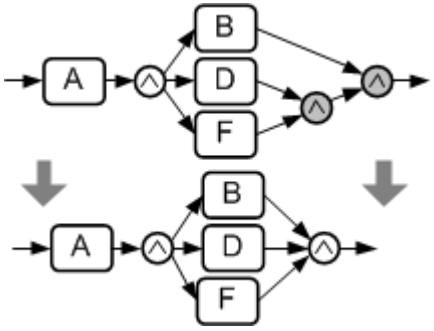


RedActivity(B)

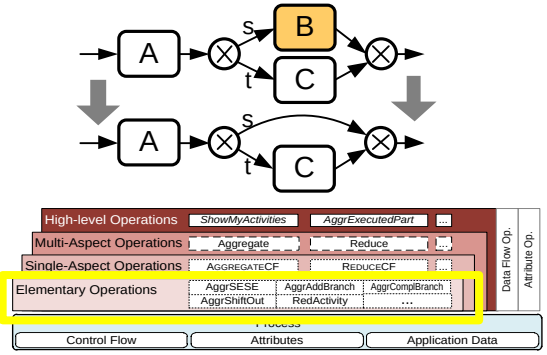


Further refactorings:





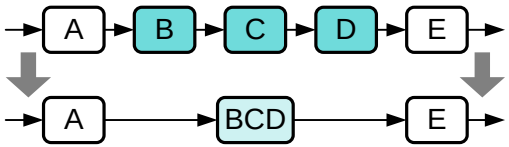
but …



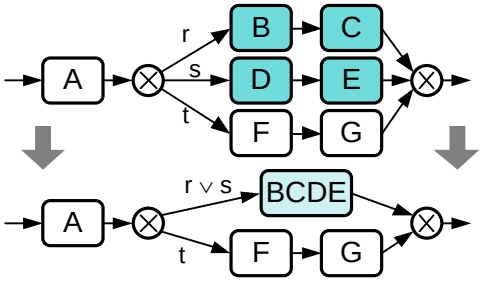
Process Visualization
Abstracting Process Models: Elementary Operations

Elementary aggregation operations

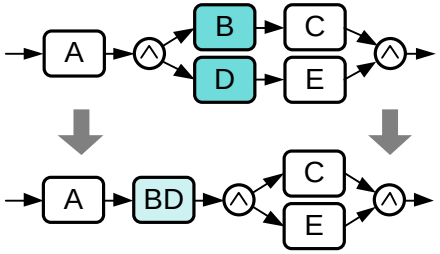
`AggrSESE`



`AggrComplBranches`

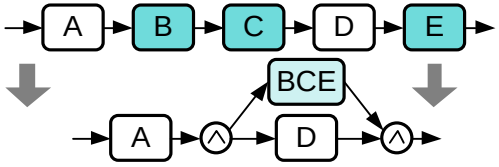


`AggrShiftOut`

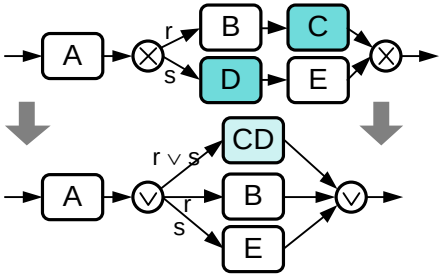


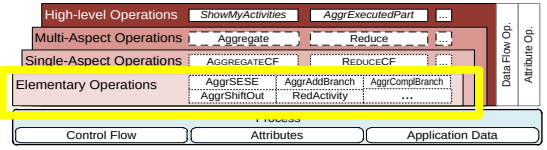
Non-connected activity sets

`AggrAddBranch`



`AggrAddBranch`

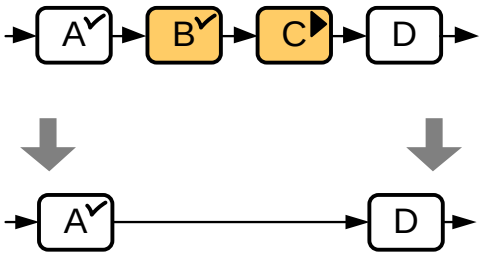




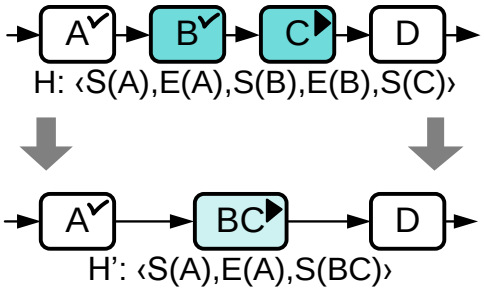
Process Visualization
Abstracting Process Models: Elementary Operations

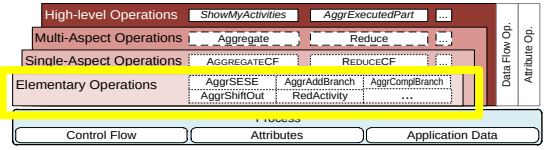
Elementary operations applied at the process instance level

`ReductionSESE`



`AggrSESE`



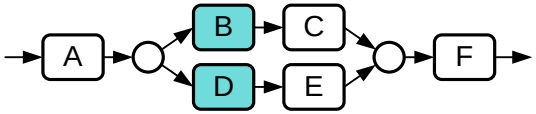


Process Visualization
Abstracting Process Models: View Properties

Let's have a closer look at aggregations!

Process Visualization
Abstracting Process Models: View Properties

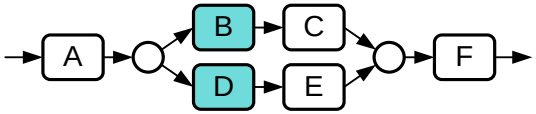
Example: Aggregate activities from set {B,D}



How to aggregate these two activities?

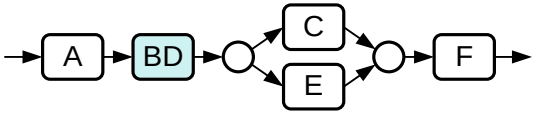
Process Visualization
Abstracting Process Models: View Properties

Example: aggregate activities from set {B,D}



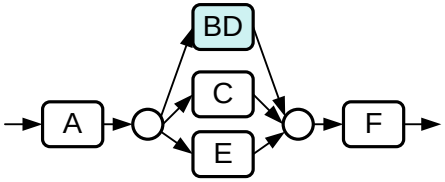
Alternative 1:

AggrShiftOut



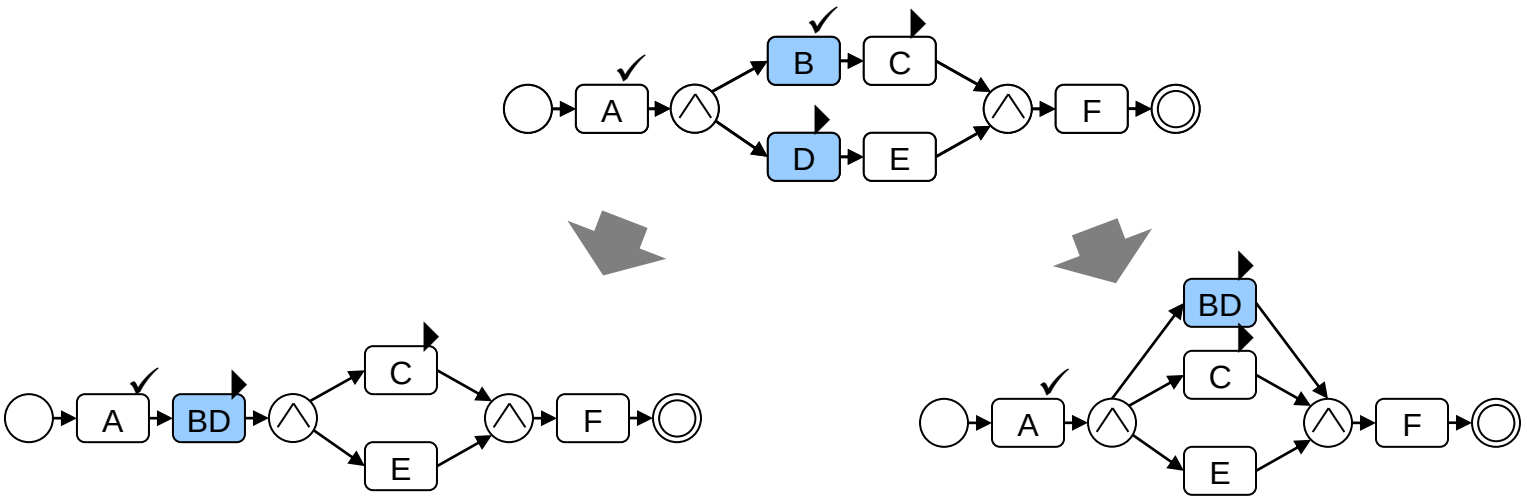
Alternative 2:

AggrAddBranch



► Well-defined properties to characterize the resulting view!

Process Visualization
Abstracting Process Models: View Properties

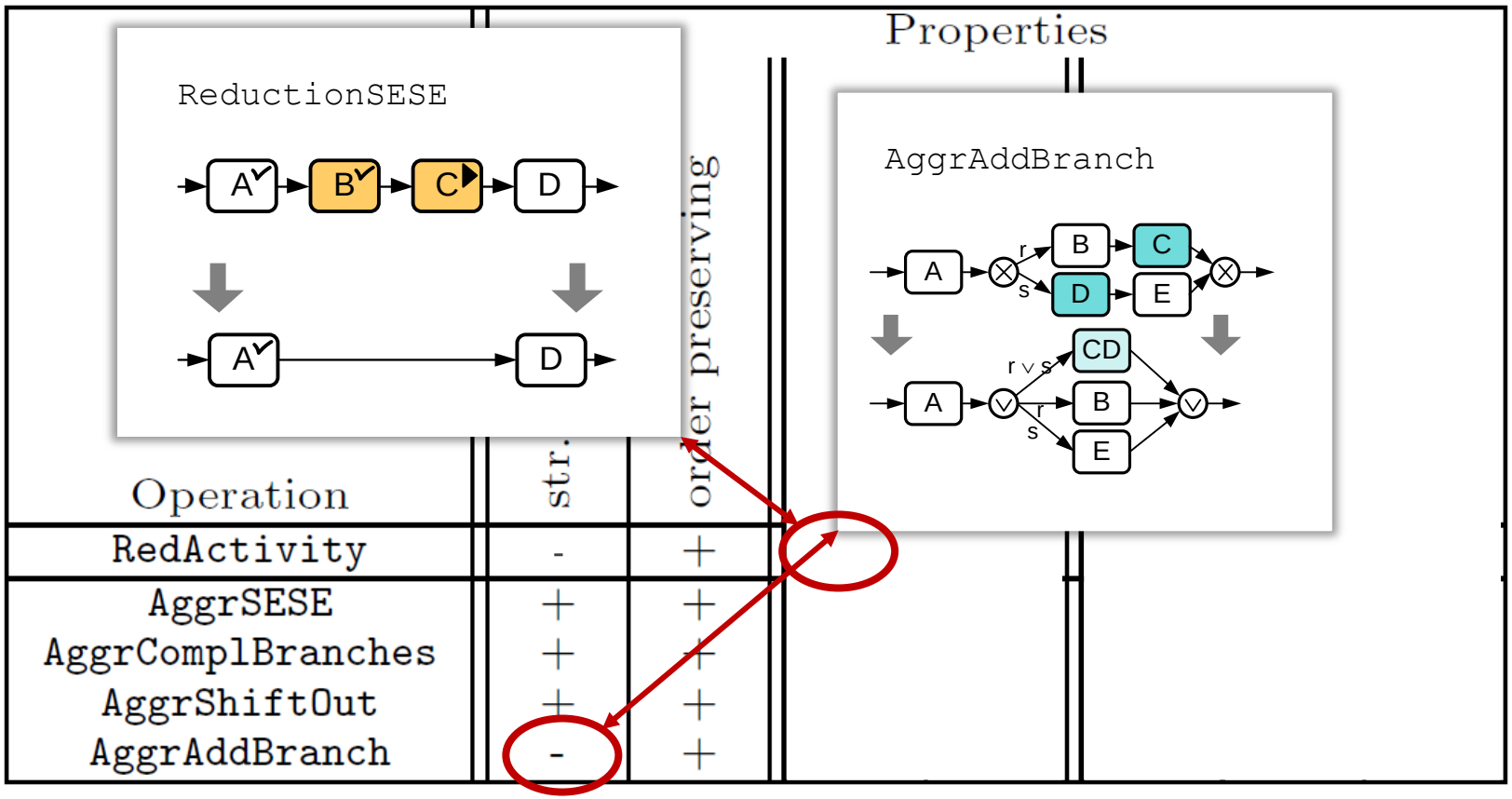


- dependency-generating
- inconsistent process state

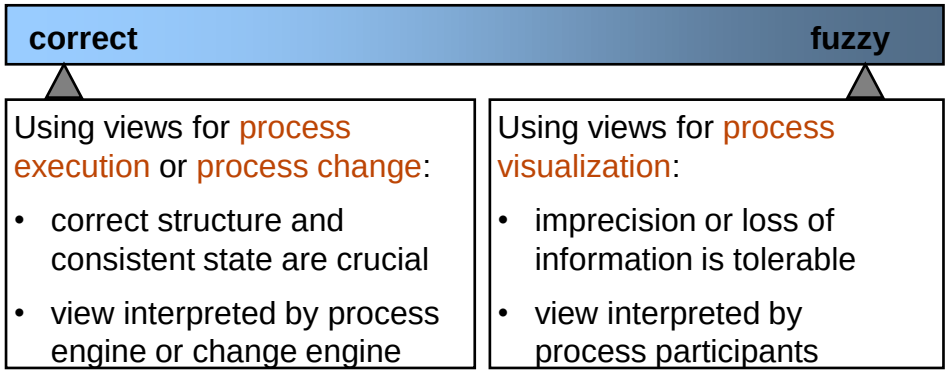
- dependency-erasing
- consistent process state

Process Visualization
Abstracting Process Models: View Properties

Properties of view-building operations:



Process Visualization
Abstracting Process Models: View Parameterization

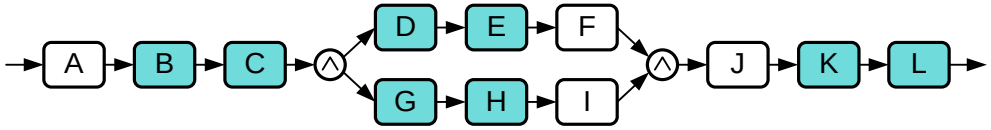


- Depending on the respecitve use case there exist different requirements regarding the properties of a process view
- Proviado addresses this issue by enabling **parameterizable process views**, i.e., the degree of imprecision or tolerable information loss may be flexibly chosen

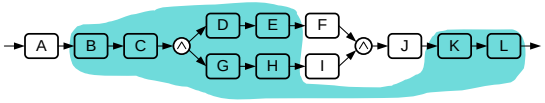
Process Visualization

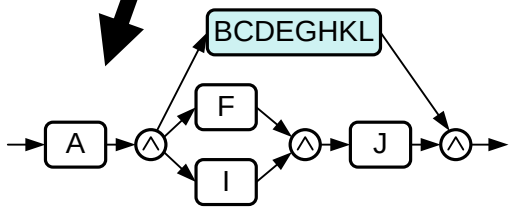
Abstracting Process Models: View Parameterization

3 strategies for aggregating an activity set (AggregateCF):



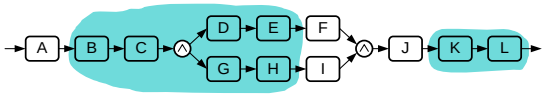
strategy = as-is



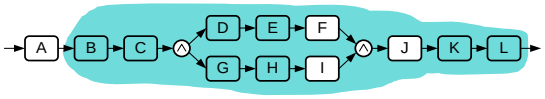


AggrAddBranch

strategy = subdivide

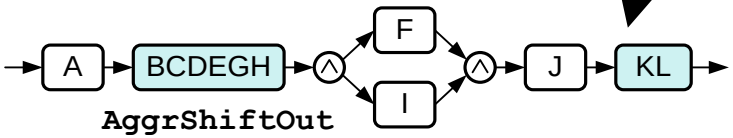


strategy = expand



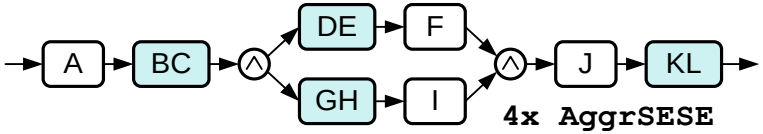
AggrSESE

dependencies = non-erasing



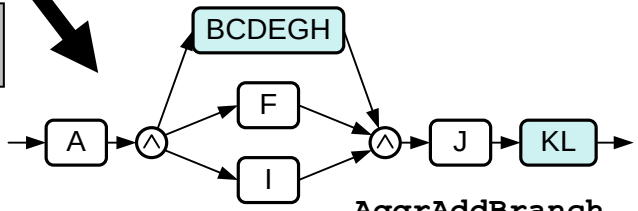
AggrShiftOut & AggrSESE

dependencies = preserving

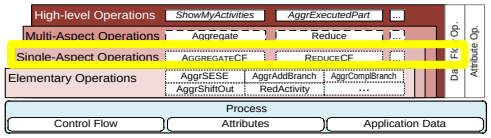


4x AggrSESE

dependencies = non-generating

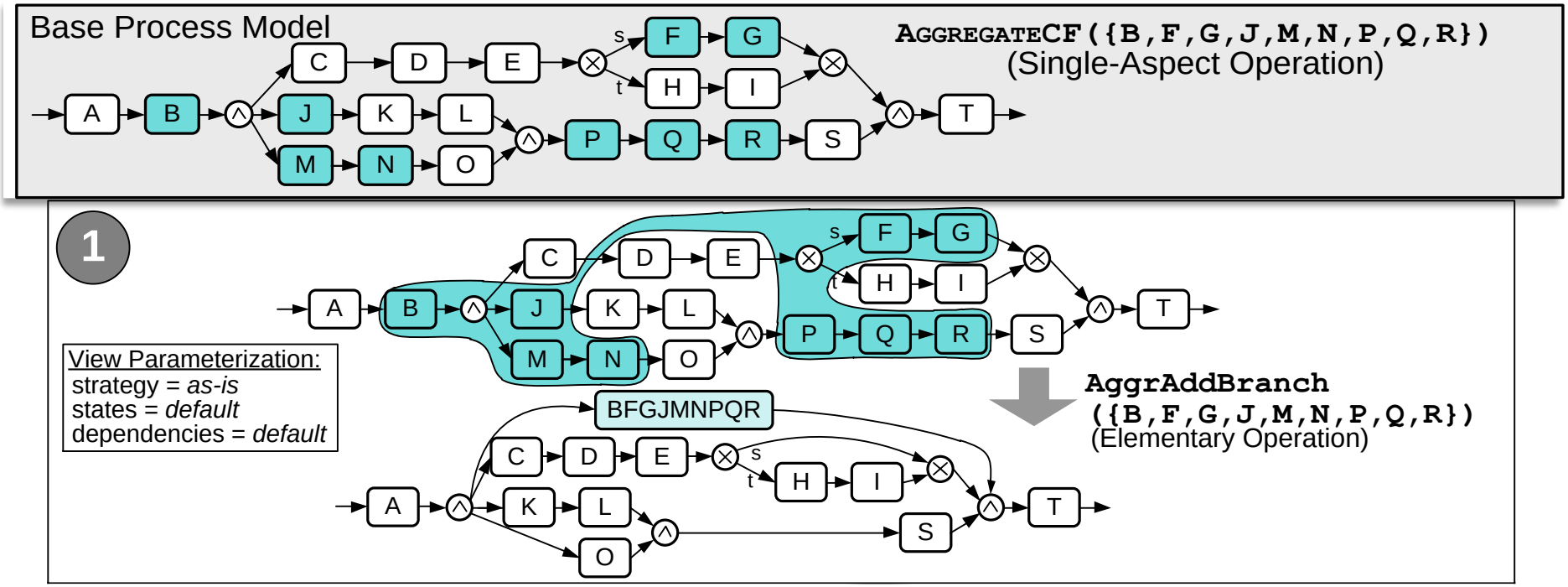


AggrAddBranch & AggrSESE



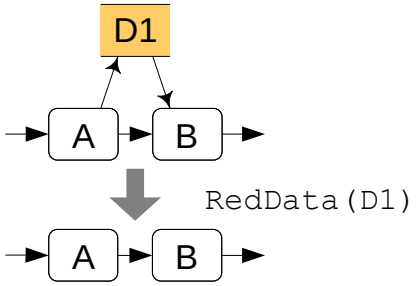
Process Visualization

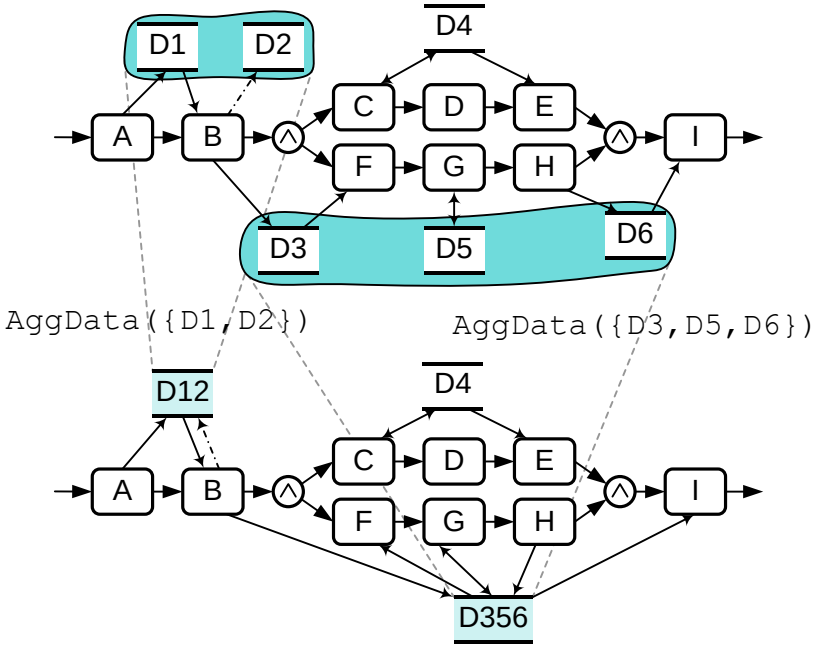
Abstracting Process Models: Single-Aspect Operations



Process Visualization
Abstracting Process Models: Other Process Aspects

Elementary operations for reducing and aggregating data elements: *RedData* and *AggData*

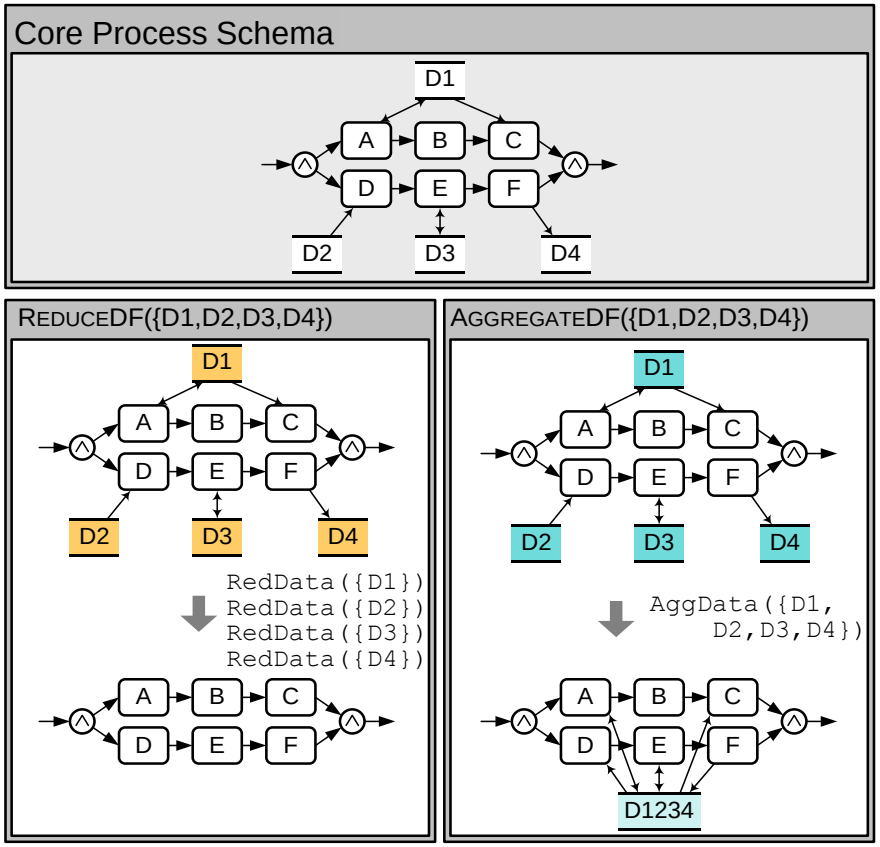




Process Visualization

Abstracting Process Models: Other Process Aspects

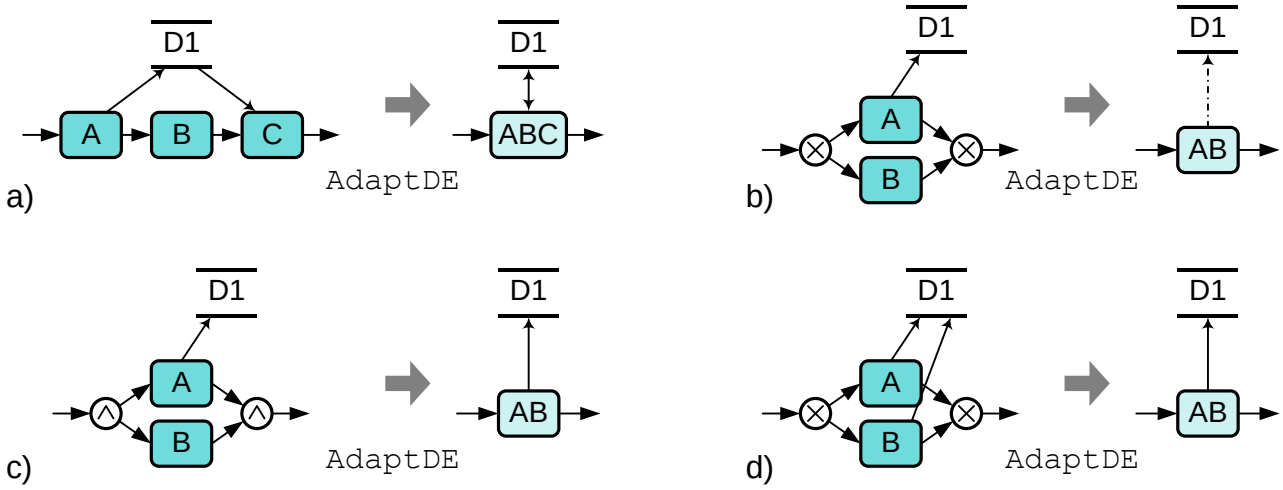
Single-aspect operations for abstracting data flow:
REDUCEDF* and *AGGREGATE DF



Process Visualization

Abstracting Process Models: Other Process Aspects

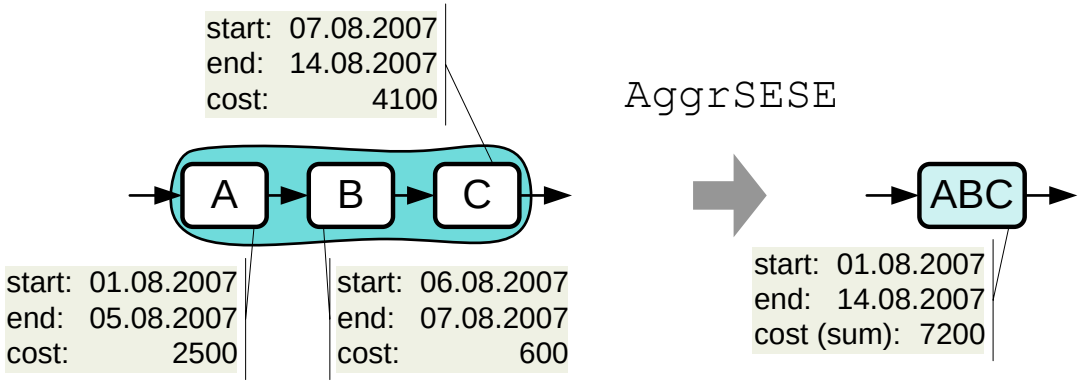
Adapting data flow edges in the context of control flow aggregations: *AdaptDE*



- Resulting data flow edge depends on the given control flow structure

Process Visualization
Abstracting Process Models: Other Process Aspects

Transforming attribute values in the context of an aggregation

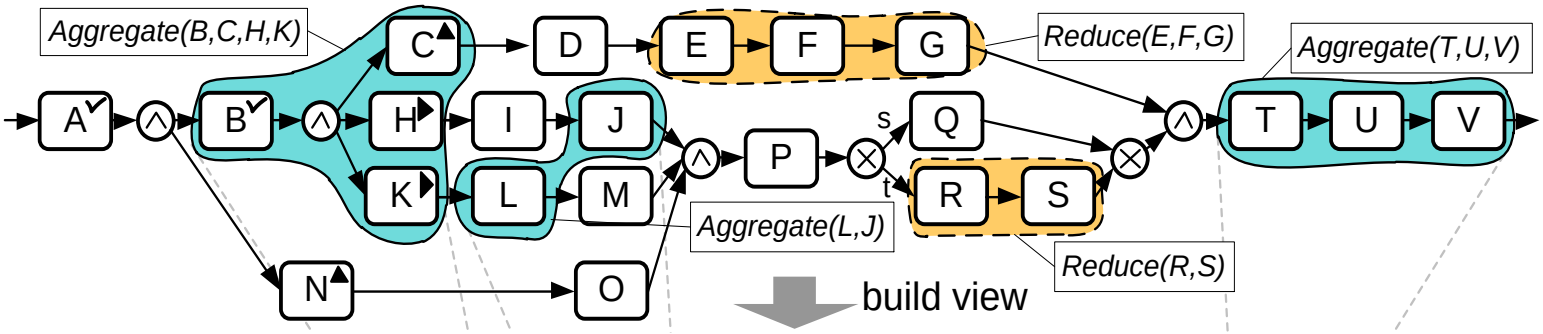


Transformation functions, e.g.,

start = min(ni.start)
end = max(ni.end)
cost = sum(ni.cost)

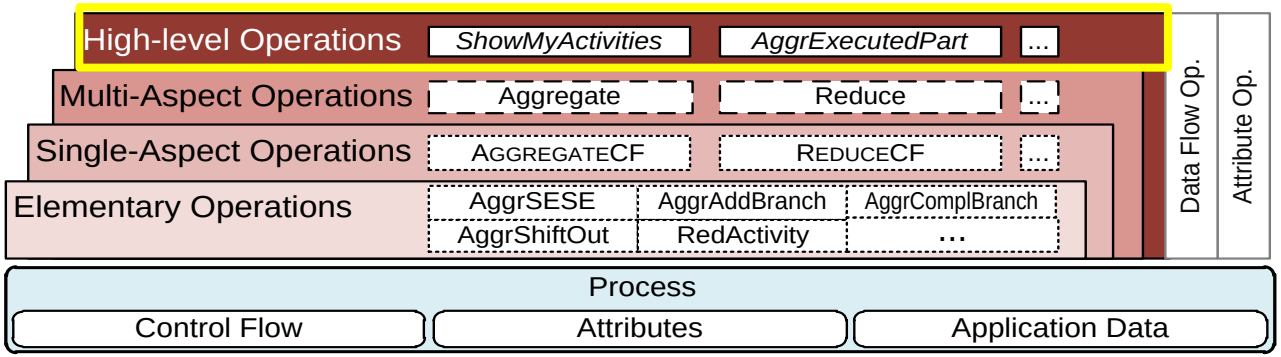
Process Visualization
Abstracting Process Models: Combining Operations

- Building Process Views by Combining Multiple Operations



Process Visualization
Abstracting Process Models: High-level Operations

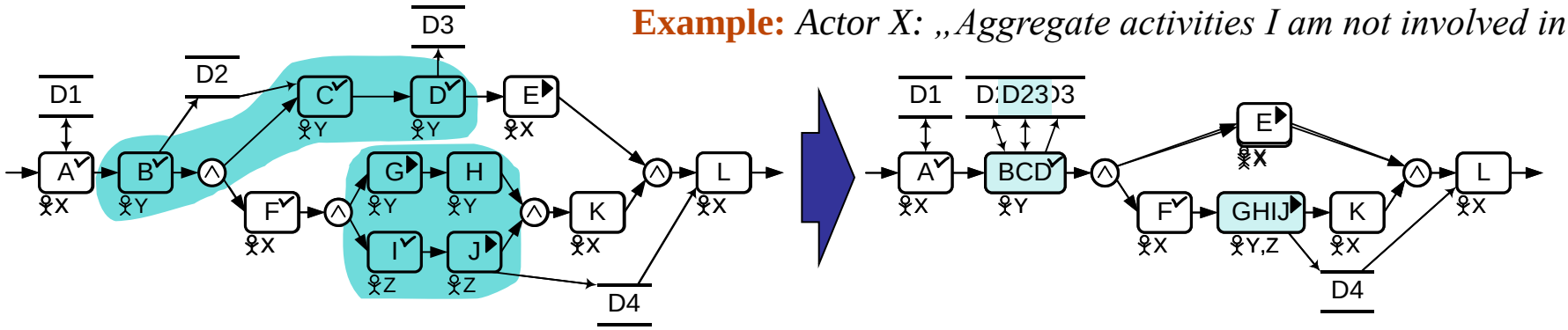
A user-friendly definition of process views requires high-level view creation operations



Example of a high-level operation: *ShowMyActivities*

- Eliminate all activities the current user is not involved in

Process Visualization
Abstracting Process Models: High-level Operations



1. Choose high-level view operation and set its parameters
2. Map high-level operation to multi-aspect operation(s)
3. Map multi-aspect operations(s) to single-aspect operations
4. Determine corresponding single-aspect operations and apply them to the original process model; apply refactorings if applicable

AggrForeignActivities(*User* = *X*)

Aggregate({*B,C,D,G,H,I,J*}, Param = {…})

AGGREGATECF({*B,C,D,G,H,I,J*},
Param = {strategy = subdivide, …})
AGGREGATEDF({*D2,D3*})

`AggrShiftOut`({*B,C,D*}) `AdaptDE`({D2})
`AdaptDE`({D3})

`AggrSESE`({G,H,I,J}) `AdaptDE`({D4})

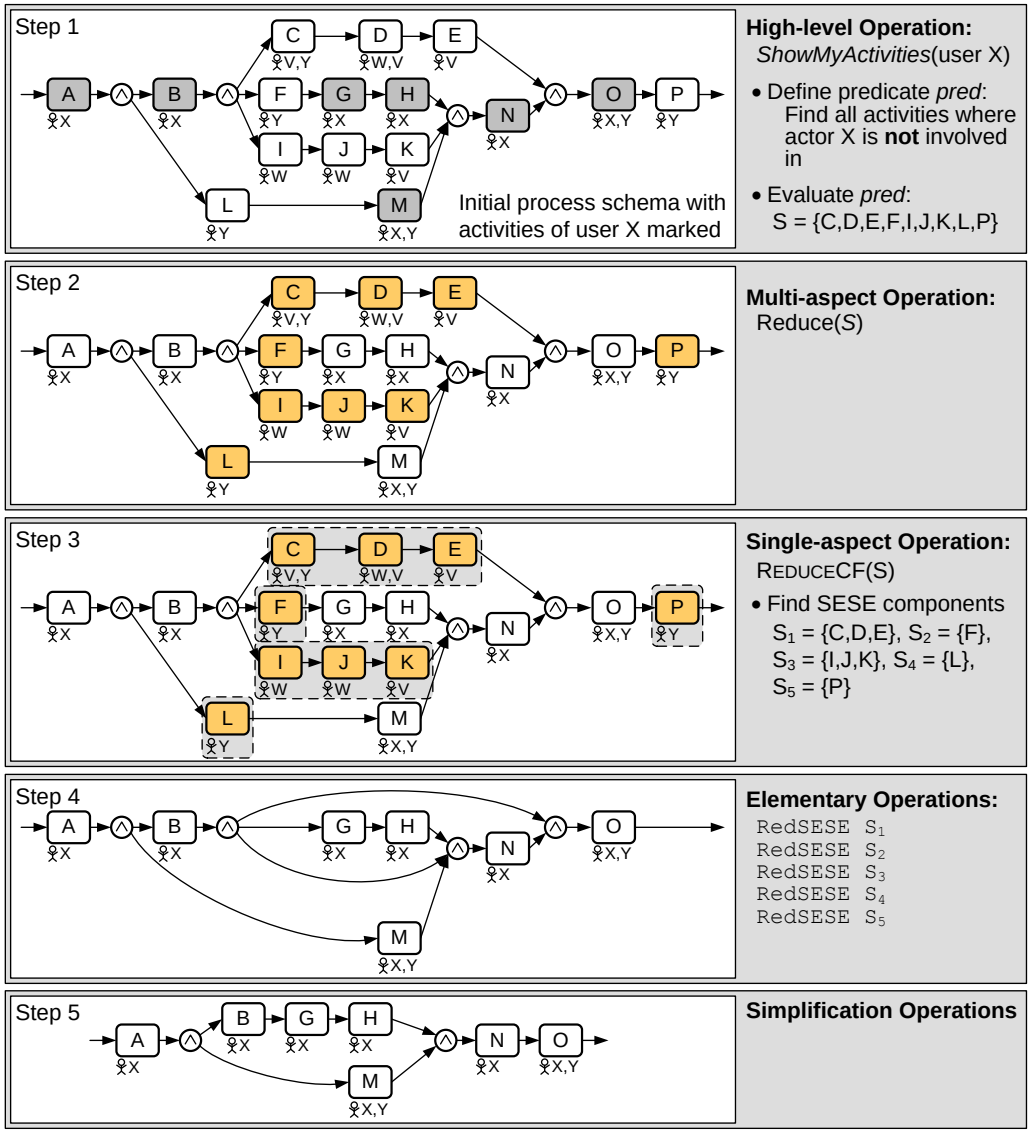
`AggrData`({D2,D3}, …)

Process Visualization

Abstracting Process Models: High-level Operations

Example:

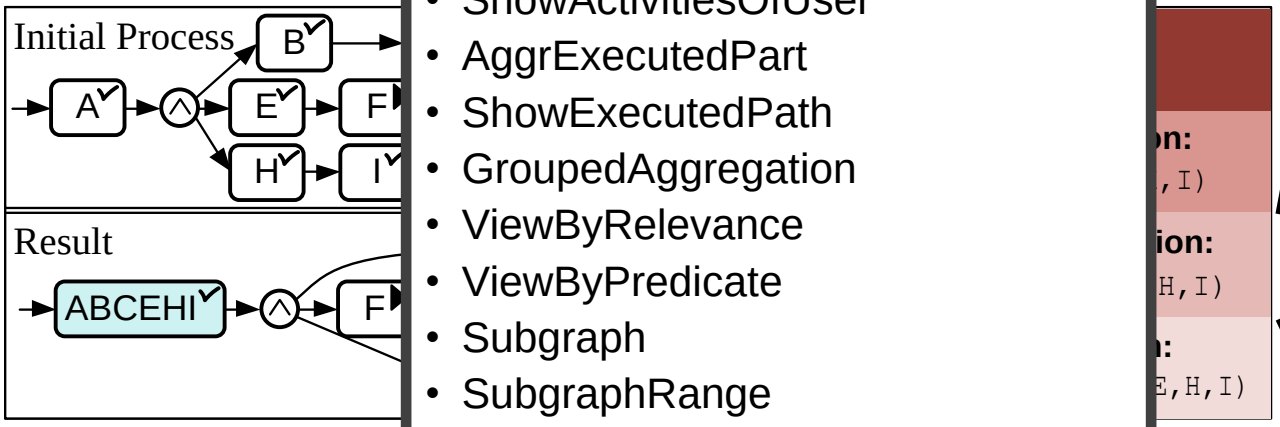
ShowMyActivities



Process Visualization
Abstracting Process Models: High-level Operations

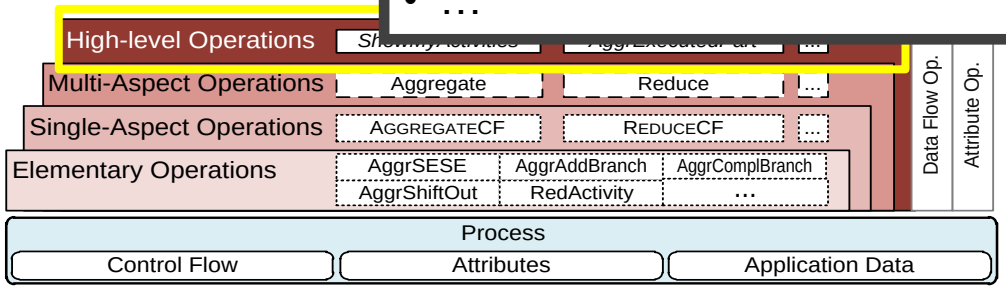
Example:

AggrExecutedP



High-level Operations in Proviado:

- ShowActivitiesOfUser
- AggrExecutedPart
- ShowExecutedPath
- GroupedAggregation
- ViewByRelevance
- ViewByPredicate
- Subgraph
- SubgraphRange
- CutProcess
- …



Parameters

Strategy:
- *as-is*
- subdivide
- expand

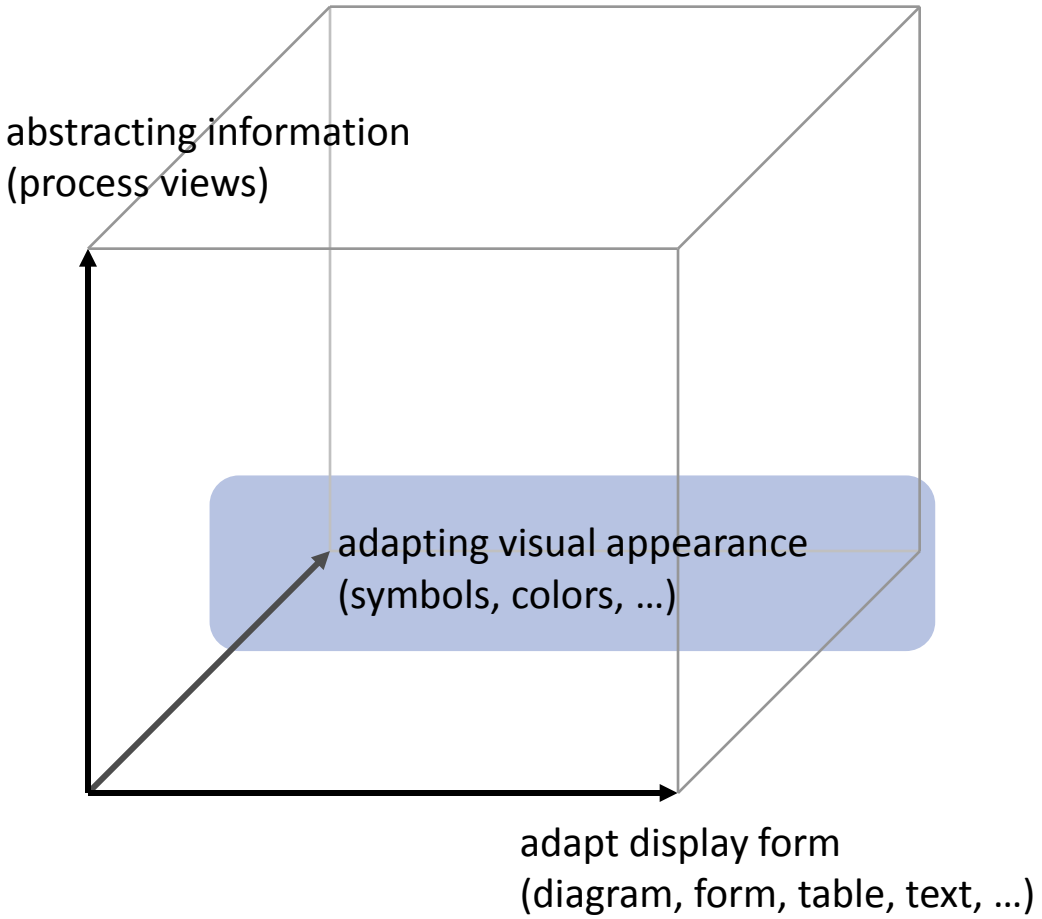
states:
- *inconsistent*
- consistent

dependencies:
- *any*
- preserving
- non-erasing
- non-generating

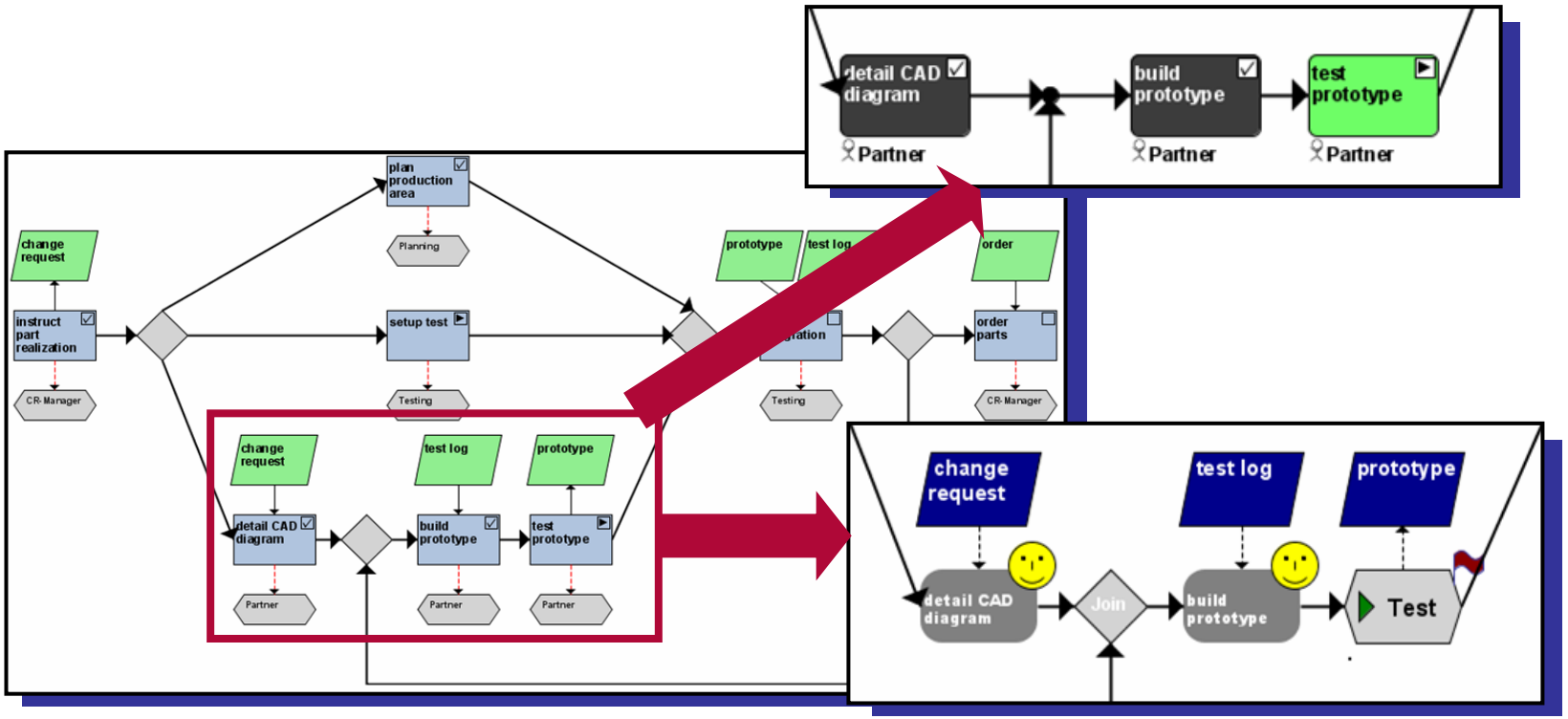
Proviado …

- offers a powerful mechanism for creating and visualizing process model abstractions (i.e., process views)
- enables a high degree of flexibility based on parameterizable high-level view-building operations
- considers all process aspects when creating a process view
  - control flow
  - data flow
  - attributes
  - run-time information
- has a well-defined formal foundation

Process Visualization Dimensions

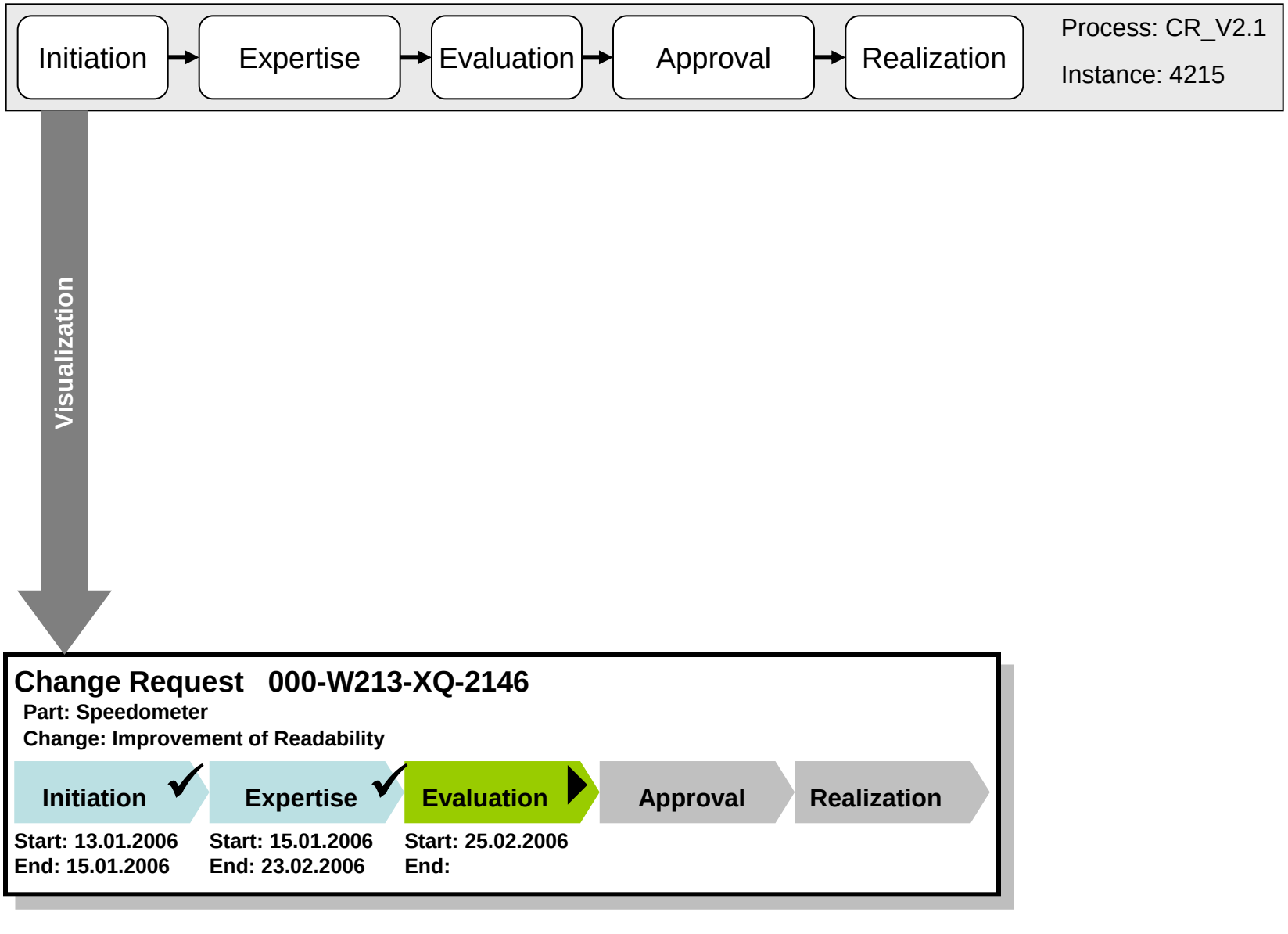


Process Visualization
Visual Appearance of a Process Model



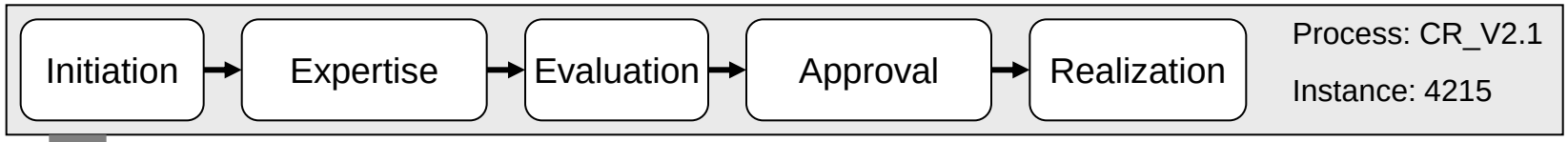
Process Visualization

Visual Appearance of a Process Model: Issues



Process Visualization

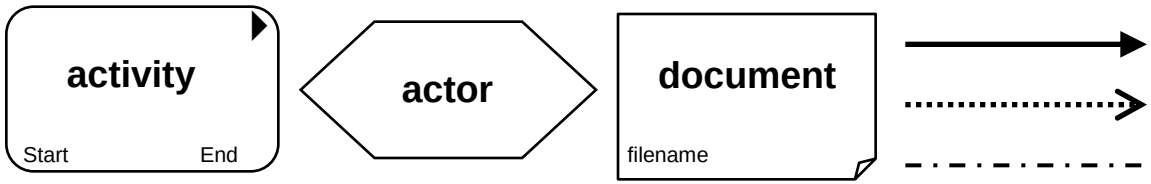
Visual Appearance of a Process Model: Issues

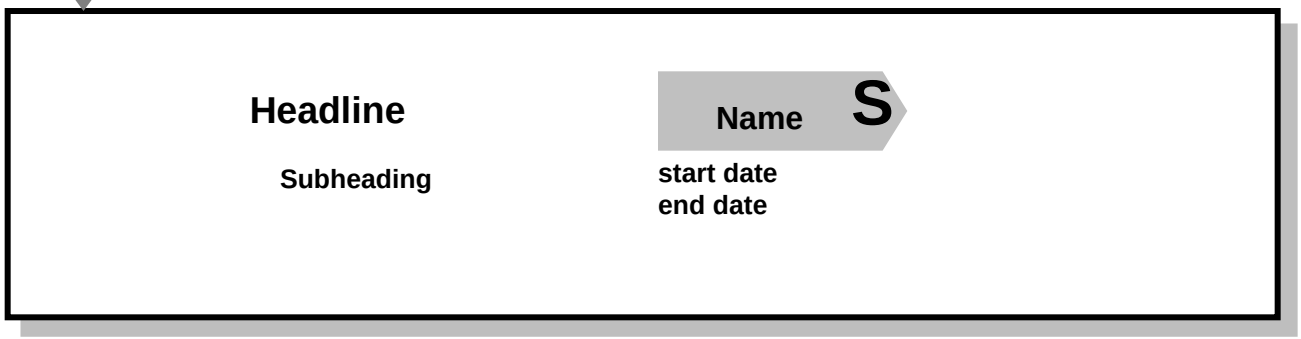


Visualization

1. Which visual notation to use?

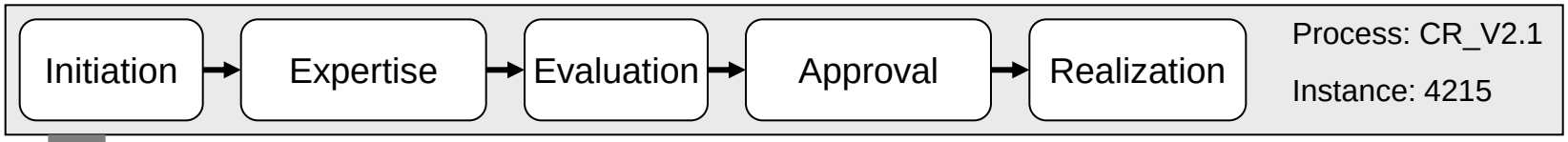
- A *template mechanism* is required that allows for the flexible definition of the
  - *visual appearance* (e.g., geometry) of process objects
  - *placeholders* for attributes or (status) symbols





Process Visualization

Visual Appearance of a Process Model: Issues

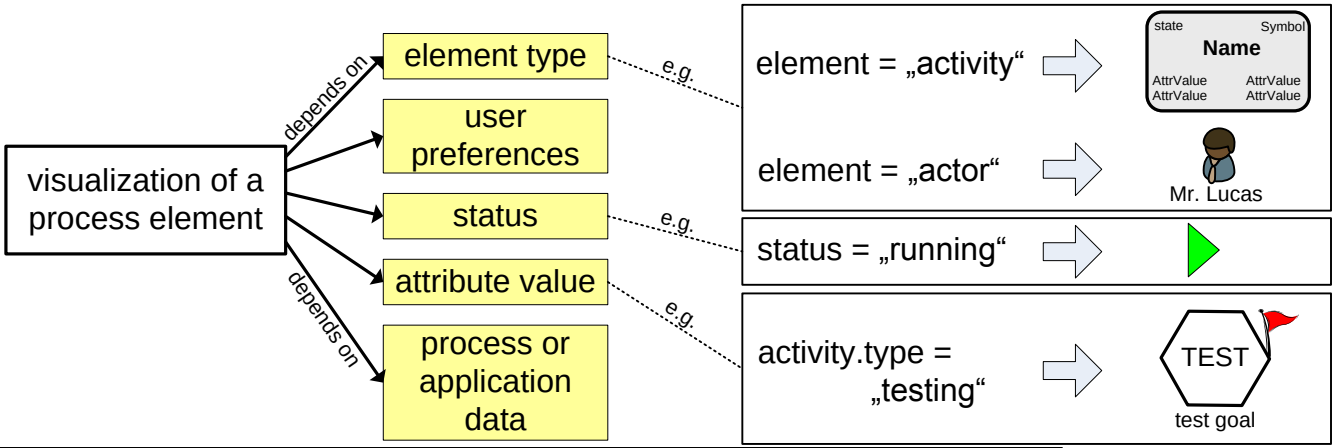


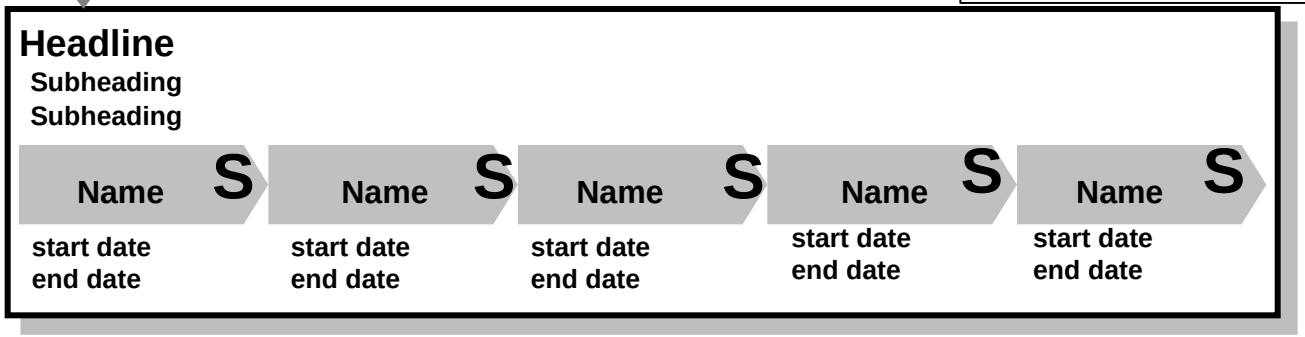
Visualization

1. Which visual notation to use?

2. When to use which visualization?

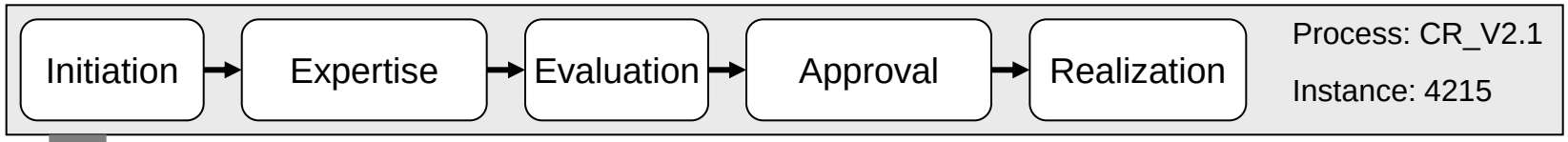
- Selection of a concrete *visualization template* may depend on the type of a process element, attributes, process data, users, etc.





Process Visualization

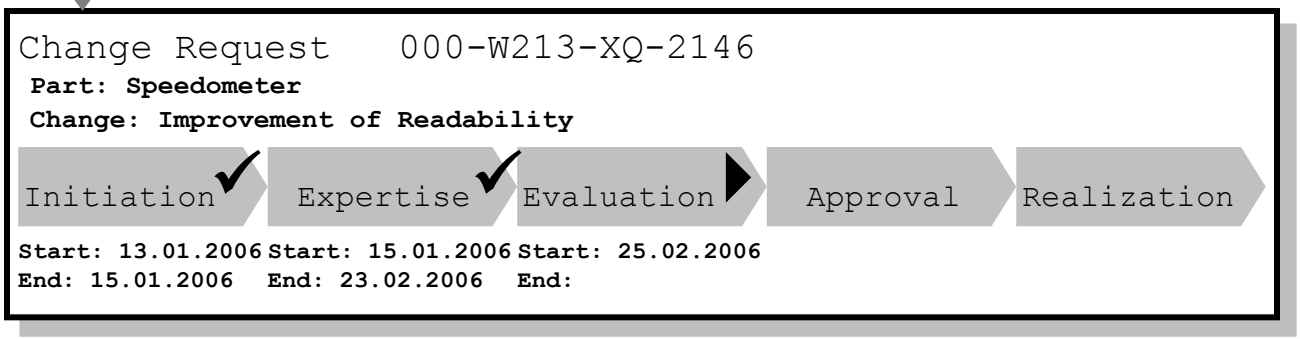
Visual Appearance of a Process Model: Issues



Visualization

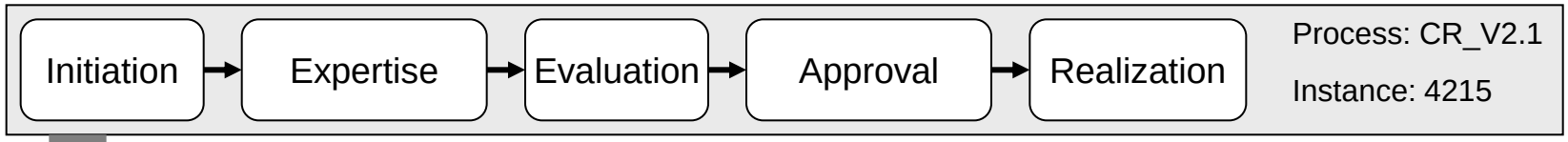
1. **Which visual notation to use?**
2. **When to use which visualization?**
3. **Which data? Where to add and how?**

- Abstracting and formatting process data
  - Formatting, e.g., date values
  - Abstracting, e.g. costs: "high " instead of 1.000.000 €
- Filling template parameters with concrete process data



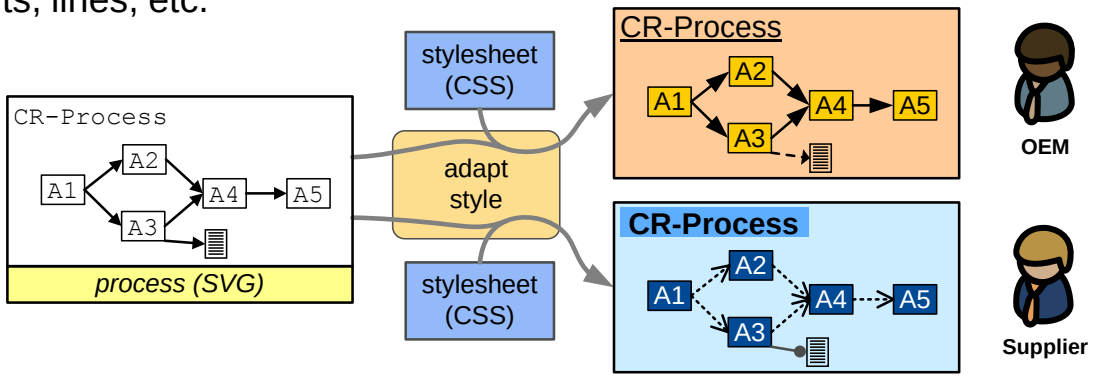
Process Visualization

Visual Appearance of a Process Model: Issues



Visualization

1. **Which visual notation to use?**
2. **When to use which visualization?**
3. **Which data? Where to add and how?**
4. **How shall the visualization style look like?**
   - Colors, fonts, lines, etc.



Change Request 000-W213-XQ-2146

Part: Speedometer

Change: Improvement of Readability

Initiation ✓	Expertise ✓	Evaluation	Approval	Realization
Start: 13.01.2006	Start: 15.01.2006	Start: 25.02.2006		
End: 15.01.2006	End: 23.02.2006	End:		

Process Visualization

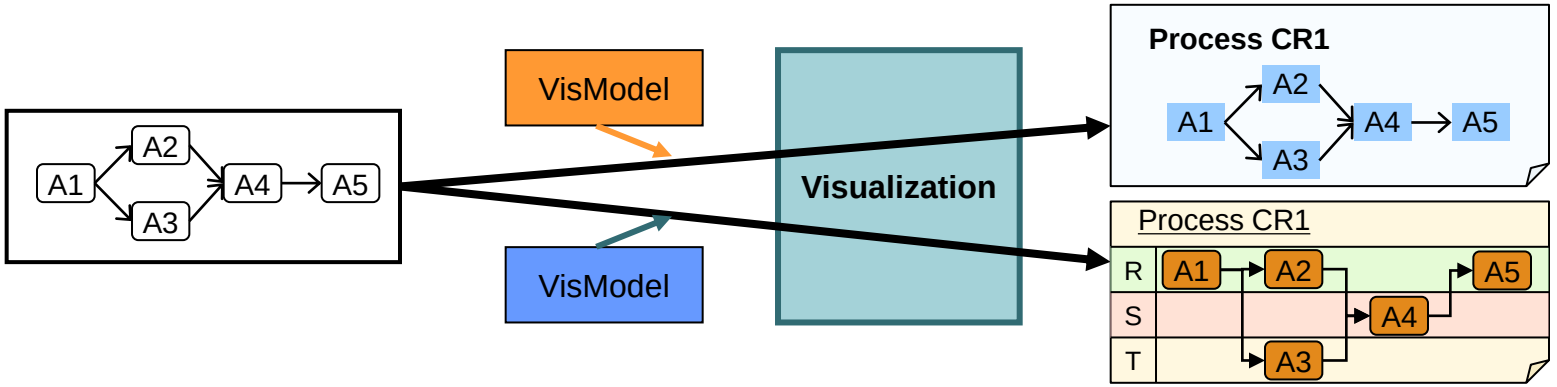
Visual Appearance of a Process Model: The Proviado Approach

How we addressed these issues in the Proviado visualization framework …

Process Visualization

Visual Appearance of a Process Model: The Proviado Approach

Basic principle: separate process data from its presentation



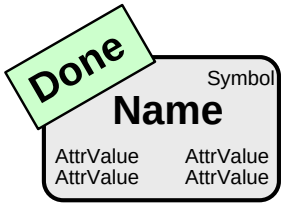
Visualization Model

- logical description of all information required for creating a particular process visualization, e.g., symbols, display form, layouting, …

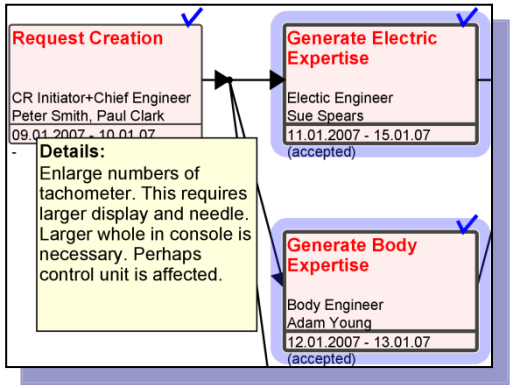
Process Visualization

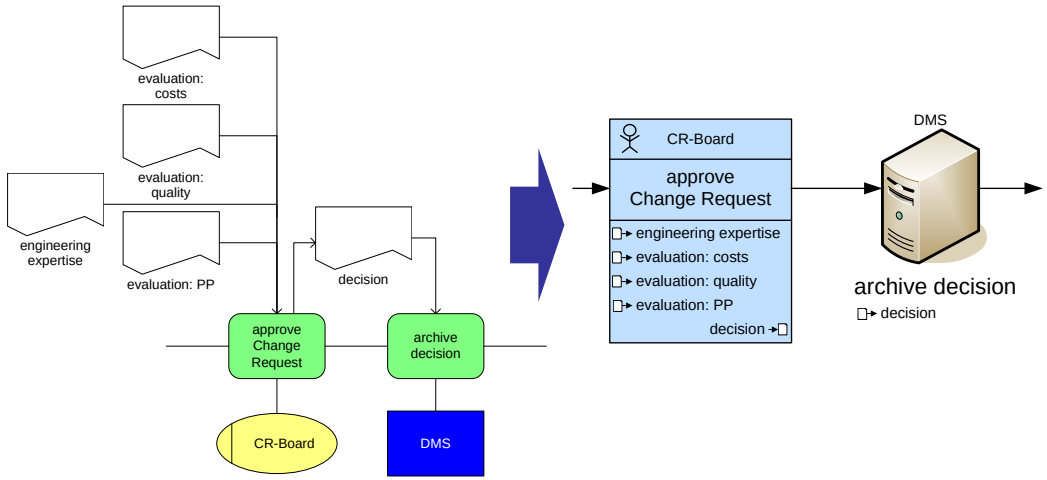
Visual Appearance of a Process Model: The Proviado Approach

Examples of visualization templates



Mr. Lucas





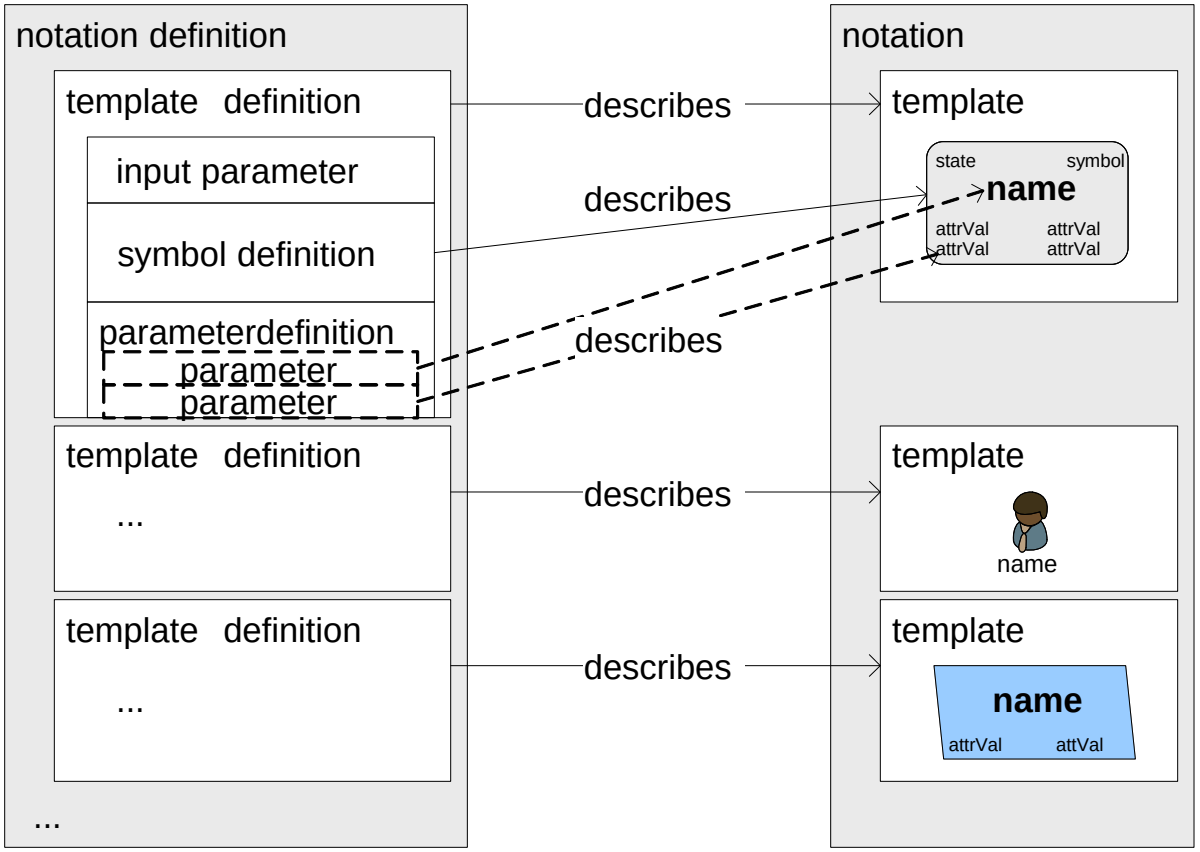
Visualization template defines

1. symbol to be used
2. data to be displayed
3. application context

Process Visualization

Visual Appearance of a Process Model: The Proviado Approach

Schema of visualization templates

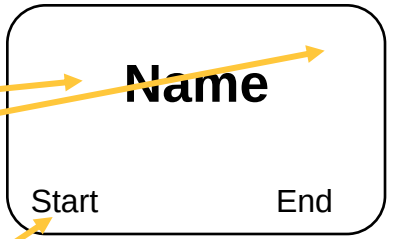


Process Visualization

Visual Appearance of a Process Model: The Proviado Approach

Defining a visualization templates for process elements

```
<template id="default_act">
  <inputs>...</inputs>
  <graphic>
    <symbol>...</symbol>
    <parameter name="Beschreibung"
      location='g/text[@name='actname']"
      value="act.getName()" />
    <parameter name="Status"
      location='g/g[@name='status']">
      <choice>
        <when test="act.getState()=RUNNING">
          <use xlink:href="#act_state_RUNNING" />
        </when>
        <when test="act.getState()=COMPLETED">...</when>
        ...
      </choice>
    </parameter>
    <parameter name=starttime"
      location='g/text/[@name='starttime']'
      value="formatDate(act.getStart(), 'dd/mm/yyyy')"/>
    ...
  </graphic>
</template>
```



Symbol shapes are defined using SVG

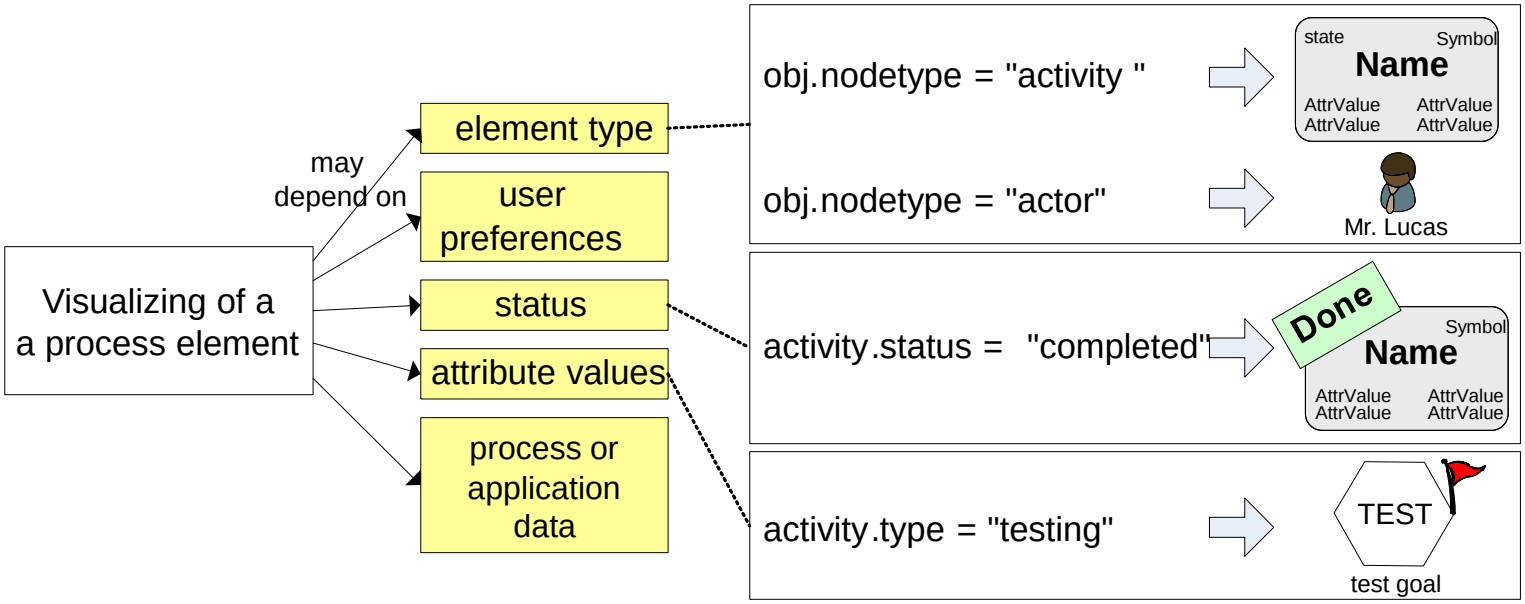
Parameter locations inside the symbol are specified using XPath

JavaScript can be used for calculating parameter values

Process Visualization

Visual Appearance of a Process Model: The Proviado Approach

Defining the application context of visualization templates

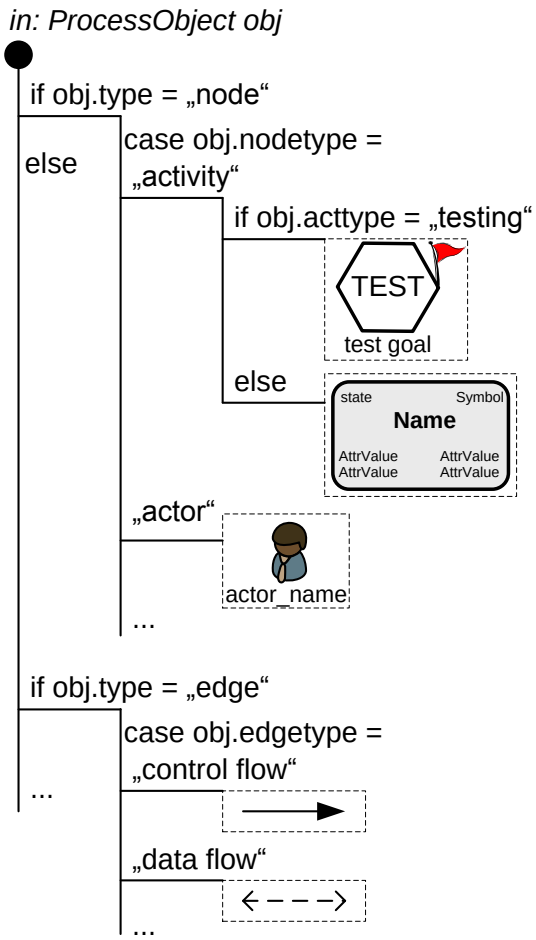


context rules

Process Visualization

Visual Appearance of a Process Model: The Proviado Approach

Defining the application context of visualization templates

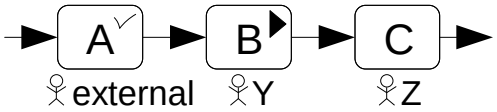



```
<if test="self.type=ACTOR">
  <template id="actor">
    <inputs>
      <input name="actor" value="self"/>
    </inputs>
  </template>
</if>
<if test="self.type=ACTIVITY">
  <choose>
    <case test="self.type='testing'">
      <template ref="testing_act">
        <inputs>
          <input name="act" value="self"/>
        </inputs>
      </template>
    </case>
    <otherwise>
      <template ref="default_act">
        <inputs>
          <input name="act" value="self"/>
        </inputs>
      </template>
    </otherwise>
  </choose>
</if>
```

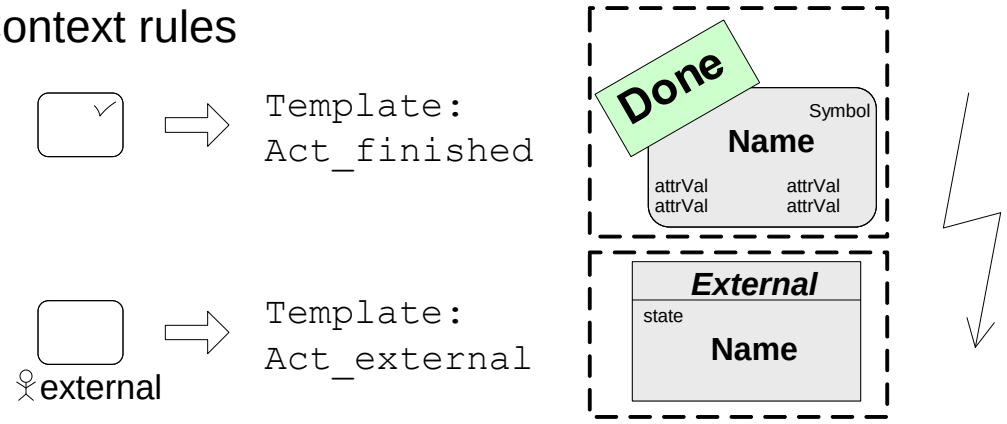
Visual Appearance of a Process Model: The Proviado Approach

Avoiding conflicting context rules

Logical process



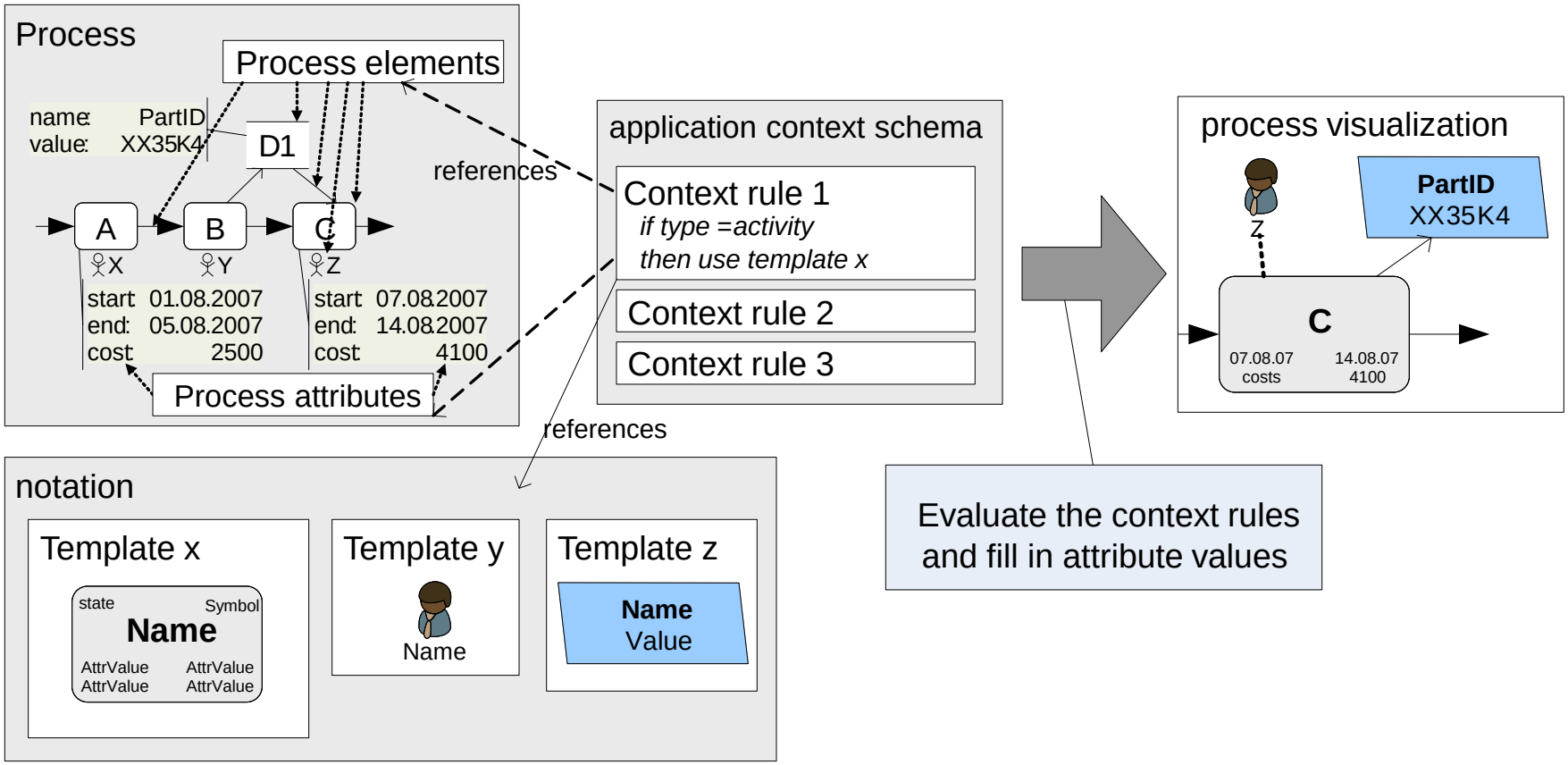
Context rules



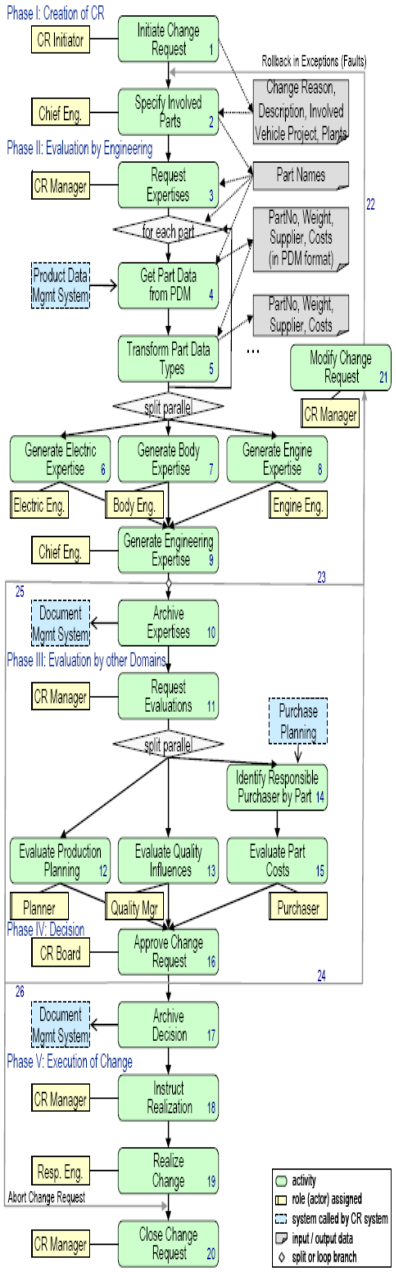
Process Visualization

Visual Appearance of a Process Model: The Proviado Approach

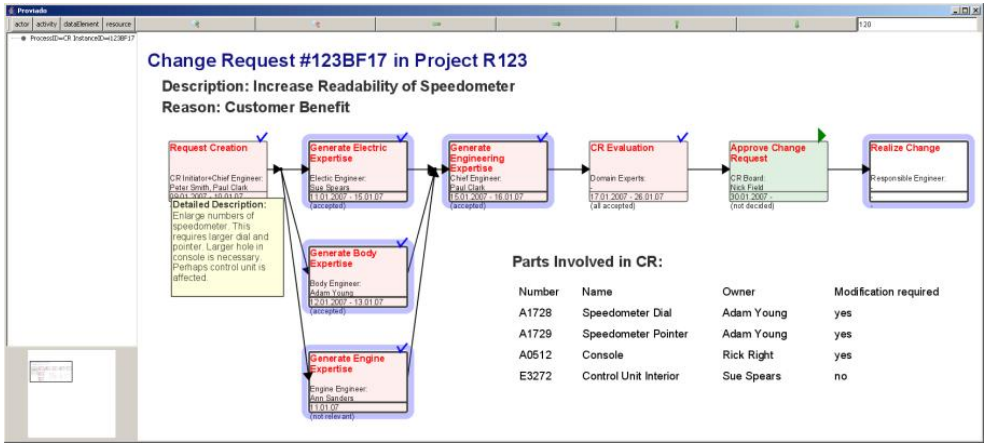
Creating a process visualization



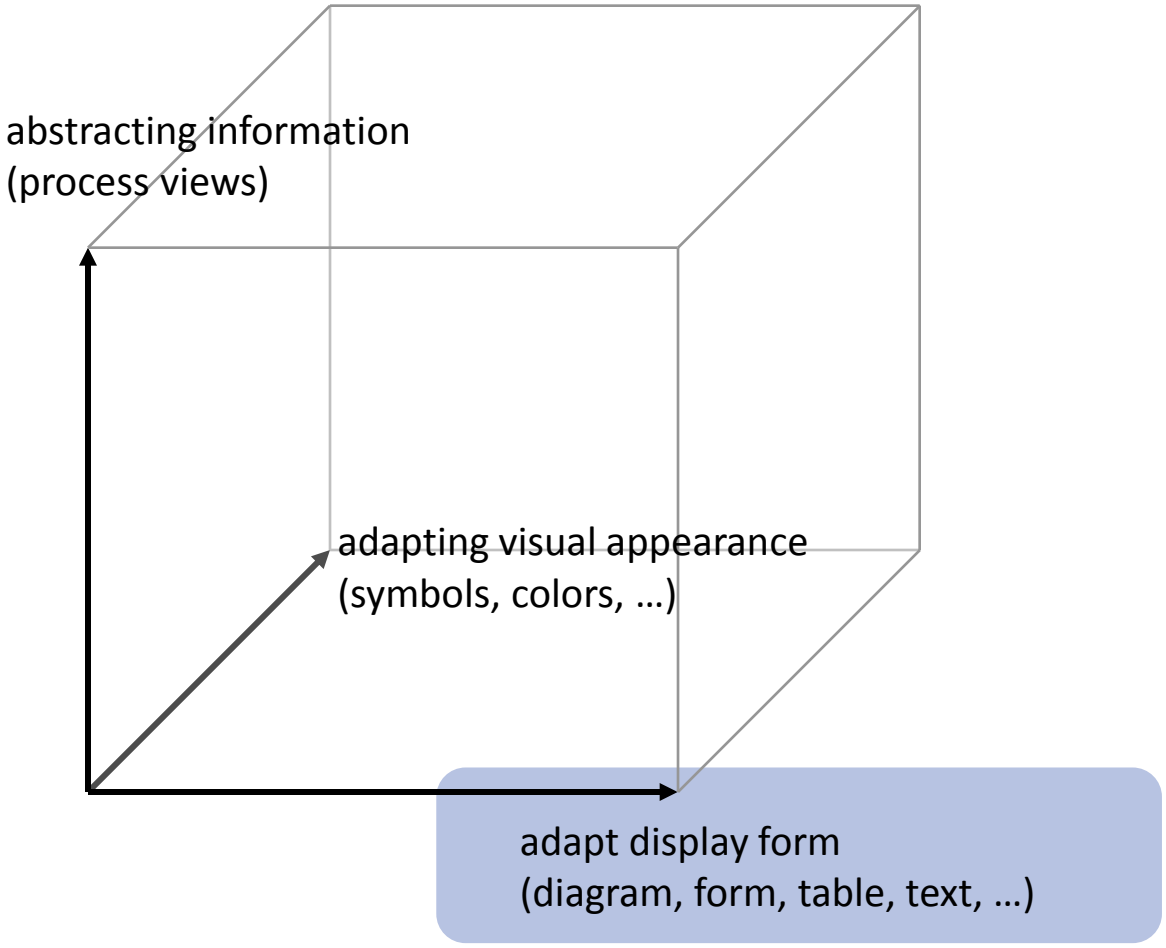
Process Visualization
Combining the Dimensions



Personalized Visualization

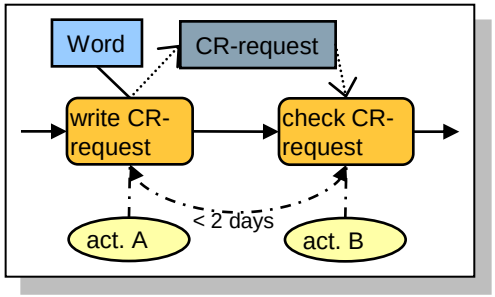


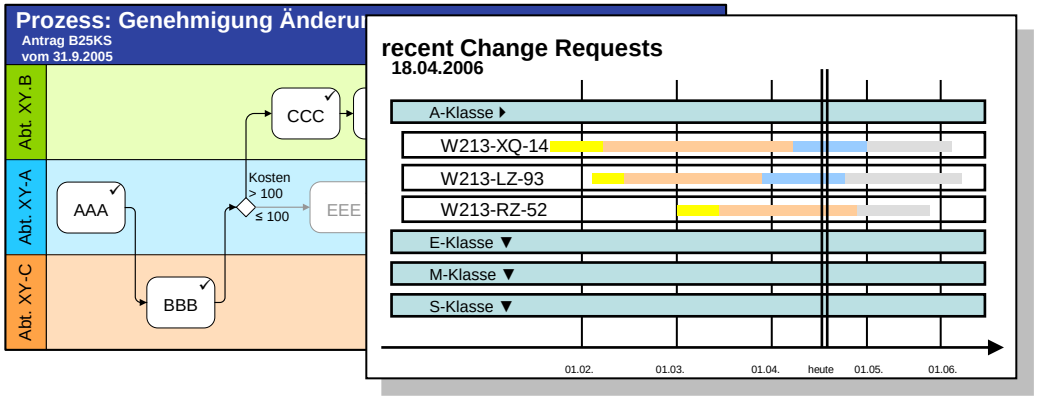
Process Visualization Dimensions

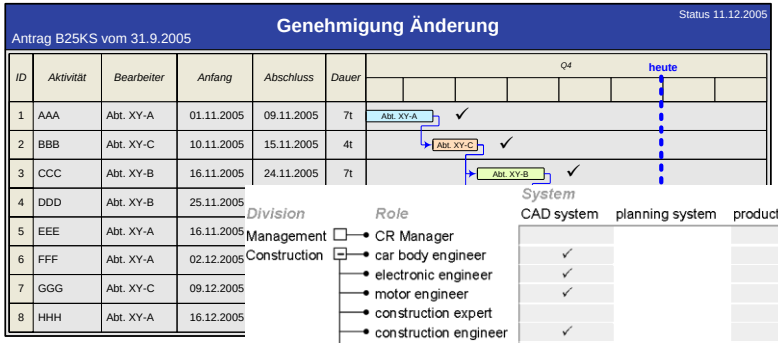


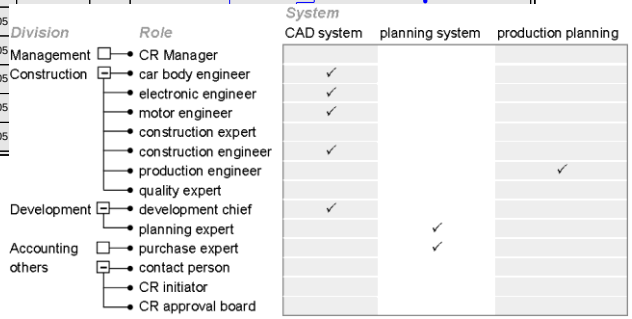
Process Visualization
Display Form of a Process Model

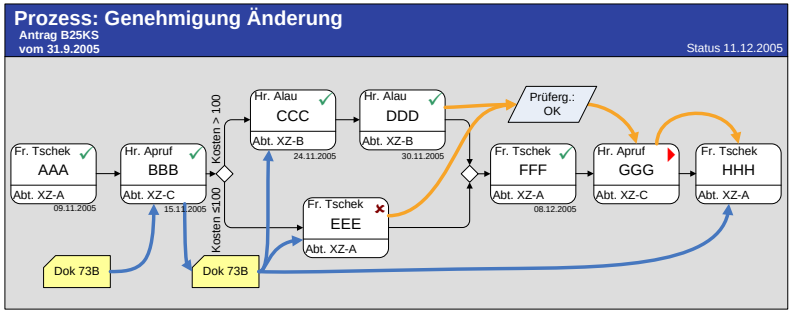
Goal: Experiment with other ways of displaying processes





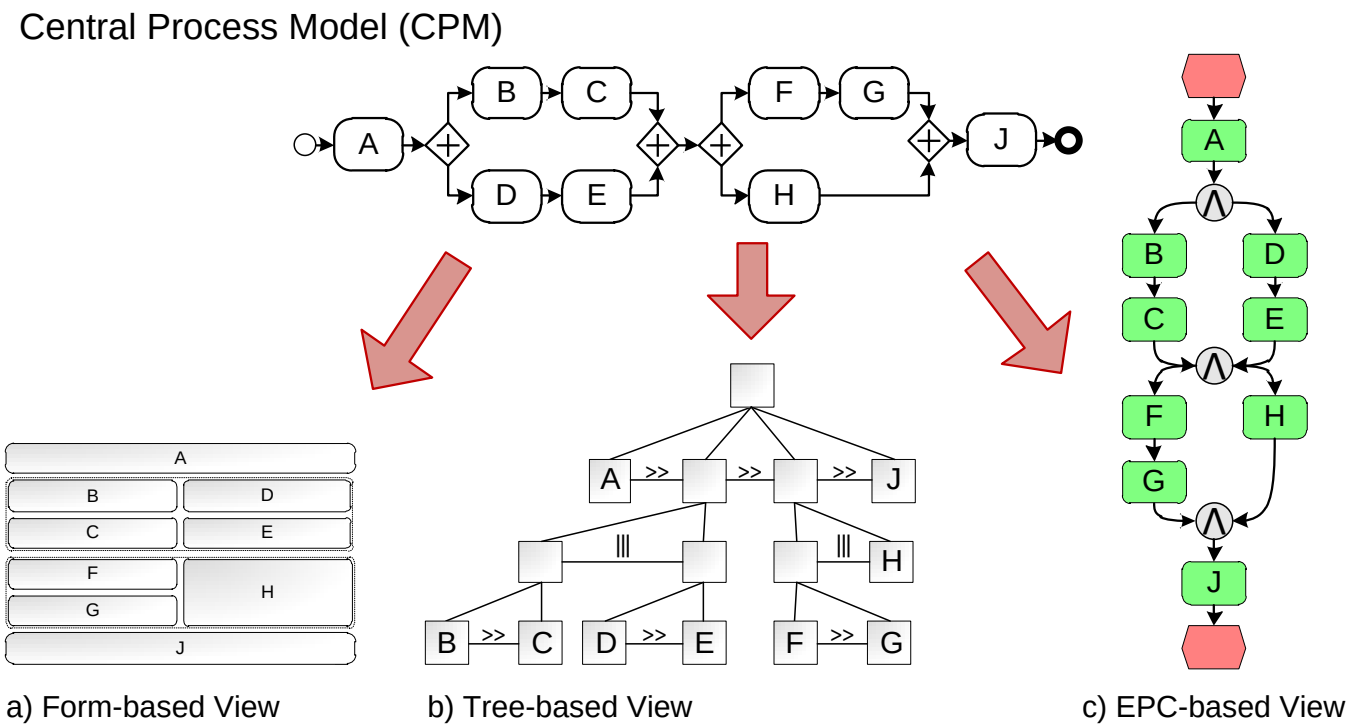






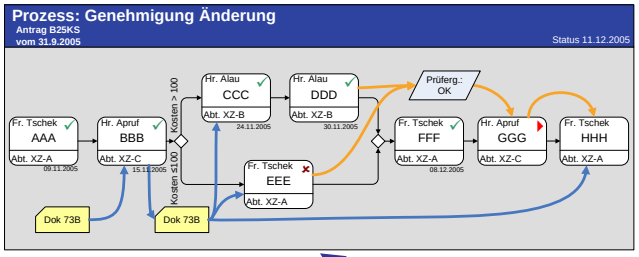
Process Visualization
Display Form of a Process Model

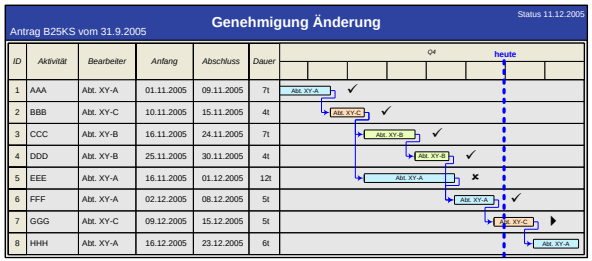
Some concrete work we did in the proView project

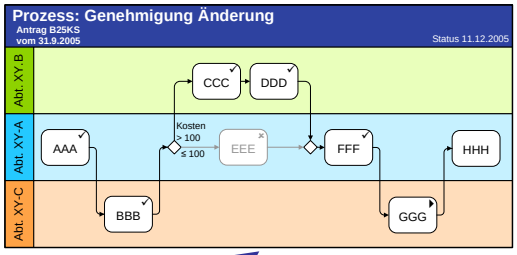


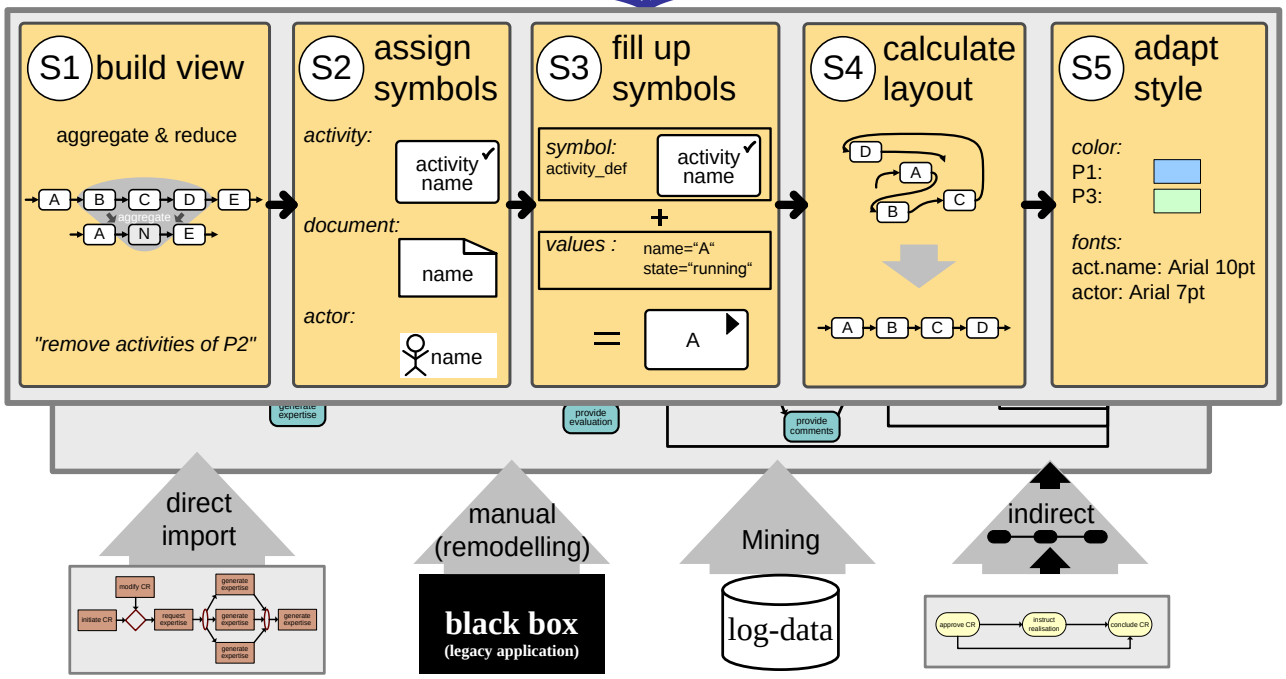
Process Visualization

Summary: What we achieved in Proviado?





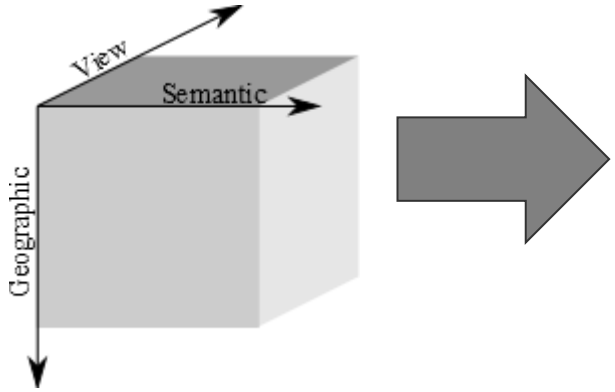




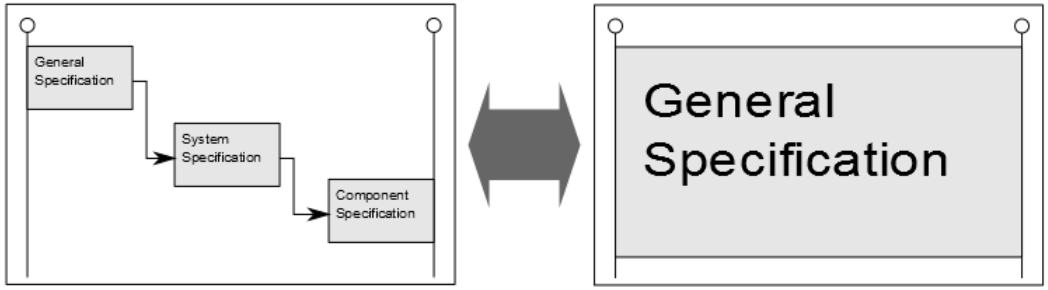
Outlook

Some topics we are currently working on ….

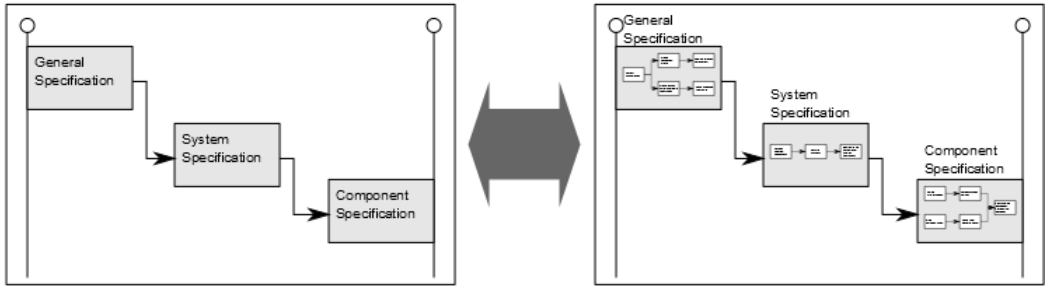
Navigating in Complex Business Processes



(a) geographic navigation dimension

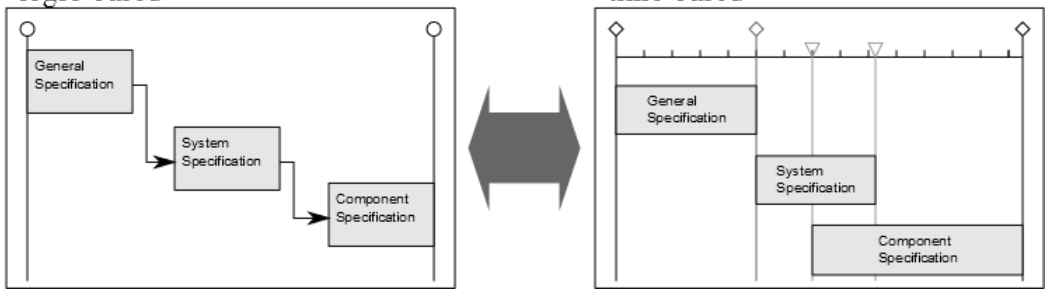


(b) semantic navigation dimension



(c) view navigation dimension

"logic-based" "time-based"



Updatable Process Model Abstractions (Process Views)

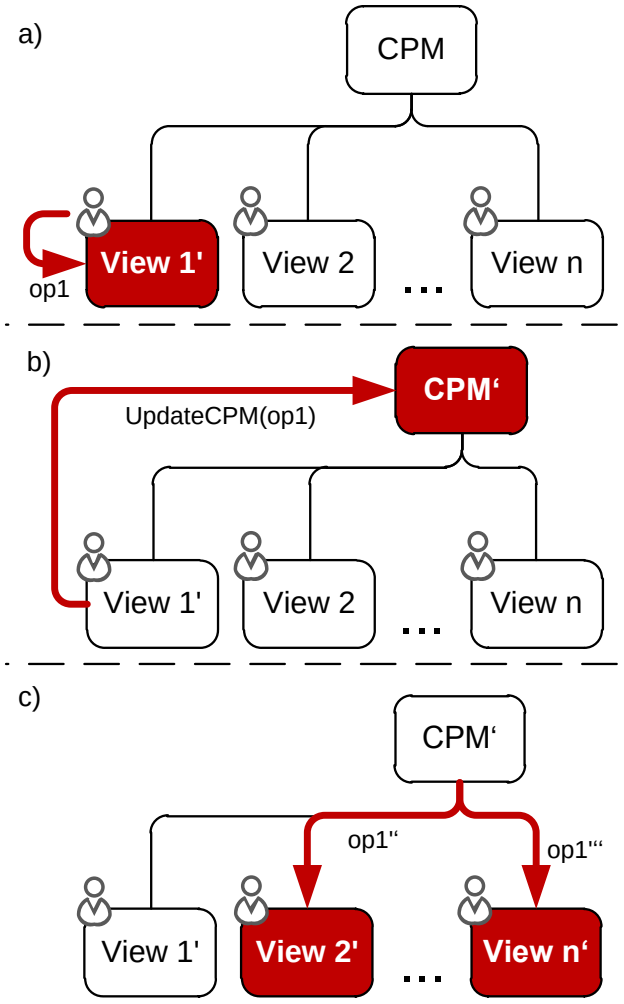
- Focus of this presentation has been on the personalized visualization of large process models
- Another fundamental issue: How to enable domain experts to change and evolve large process models!

Proviado → **proVie** Create Update Execute Appearance

Updatable Process Model Abstractions (Process Views)

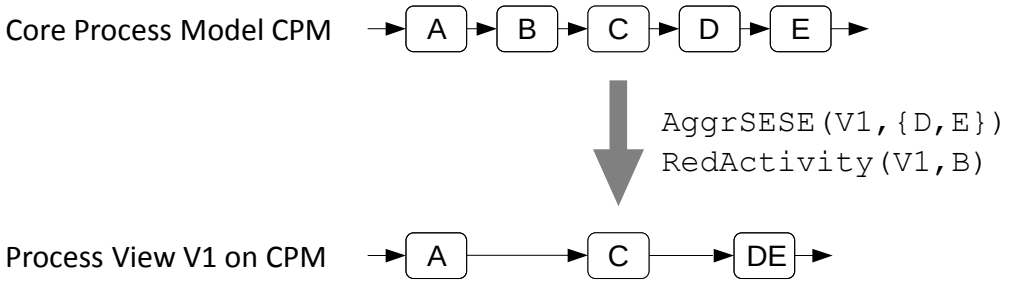
proView

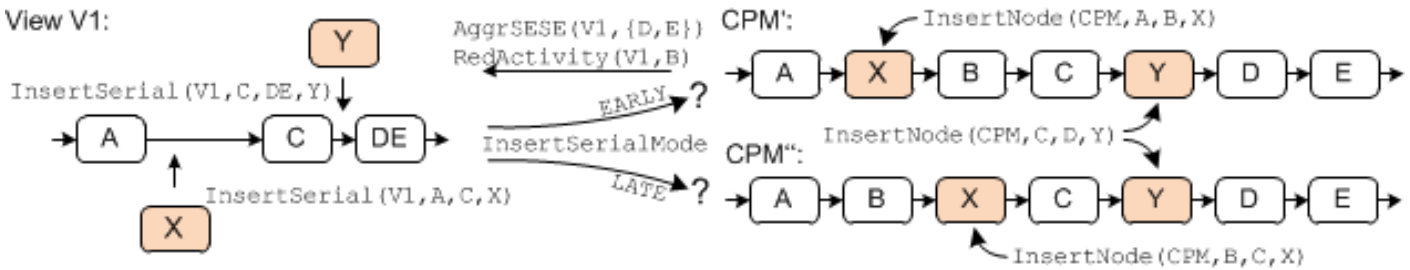
- Basic Idea: Using process views not only for visualization purpose, but also as interface for changing the underlying core process model (CPM)
- Updates of a process view then have to be correctly propagated to its CPM as well as all other views on this CPM
- Necessitates a formal foundation



Updatable Process Model Abstractions (Process Views)

proViewW



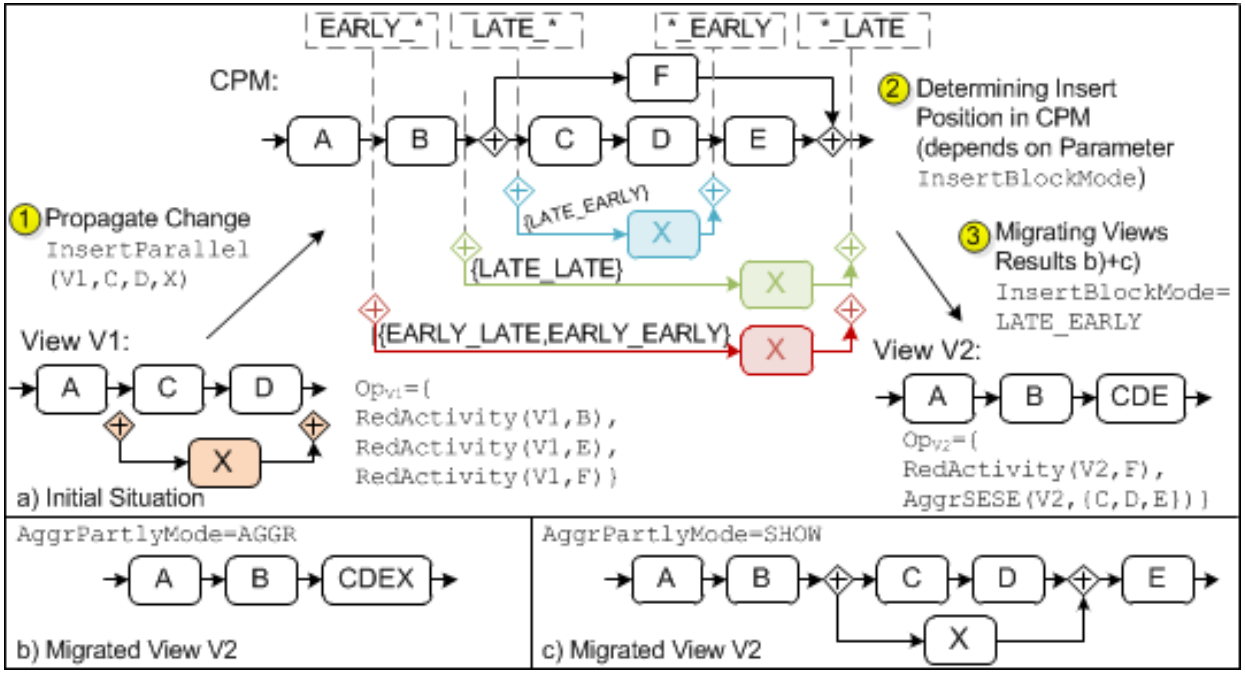


Ambiguities when propagating view changes to the CPM

Updatable Process Model Abstractions (Process Views)

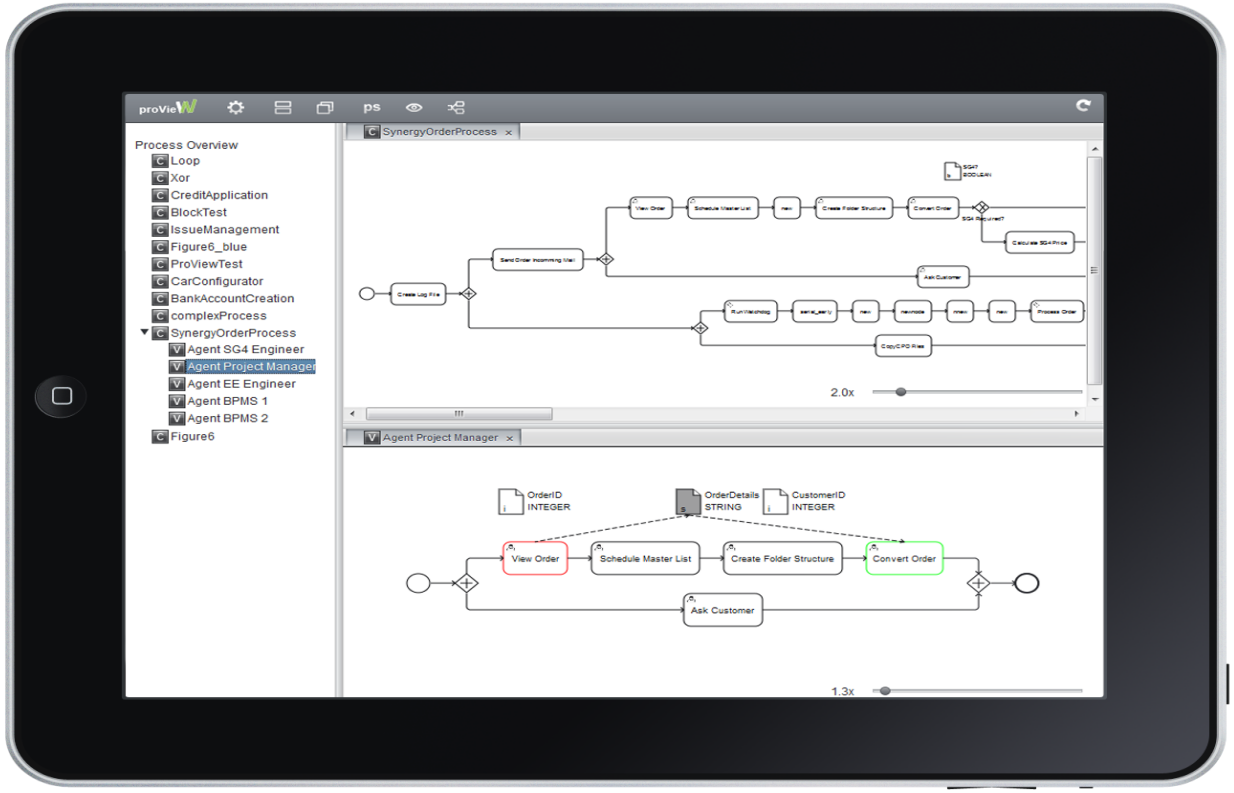
proViewW

Updating a CPM and related views after a view update!

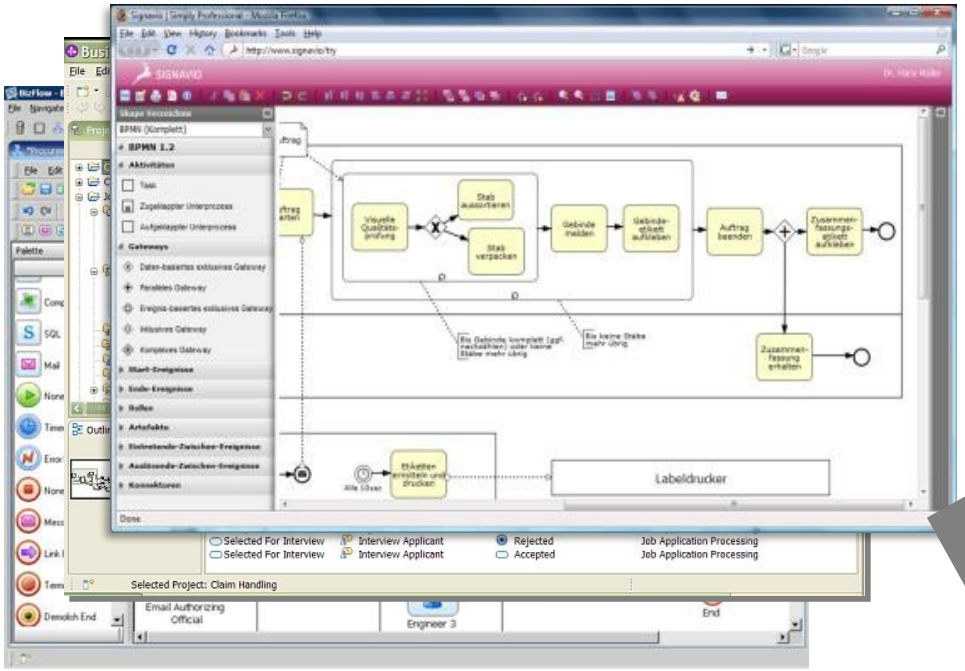


Updatable Process Model Abstractions (Process Views)

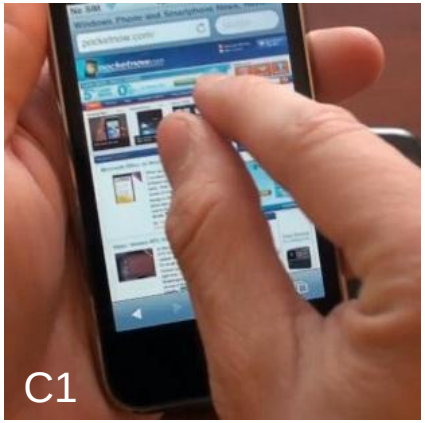
Vist the proViewW DEMO this Tuesday (17:00)



Gesture-based Interaction with Process Models



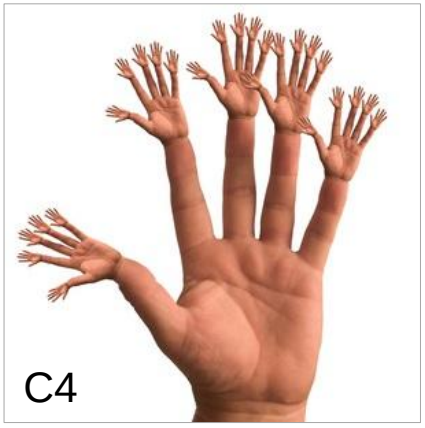
Gesture-based Interaction with Process Models
Characteristics of Multi-touch Applications



Screen Occlusion

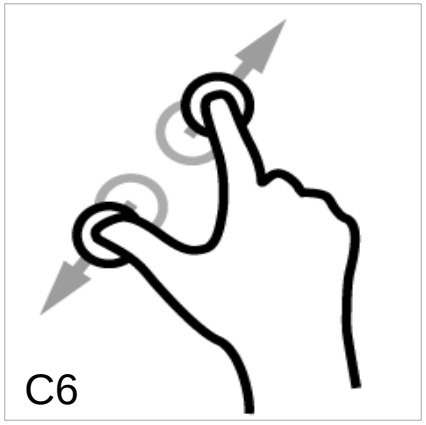
Handling Precision

Fatigue Extremities



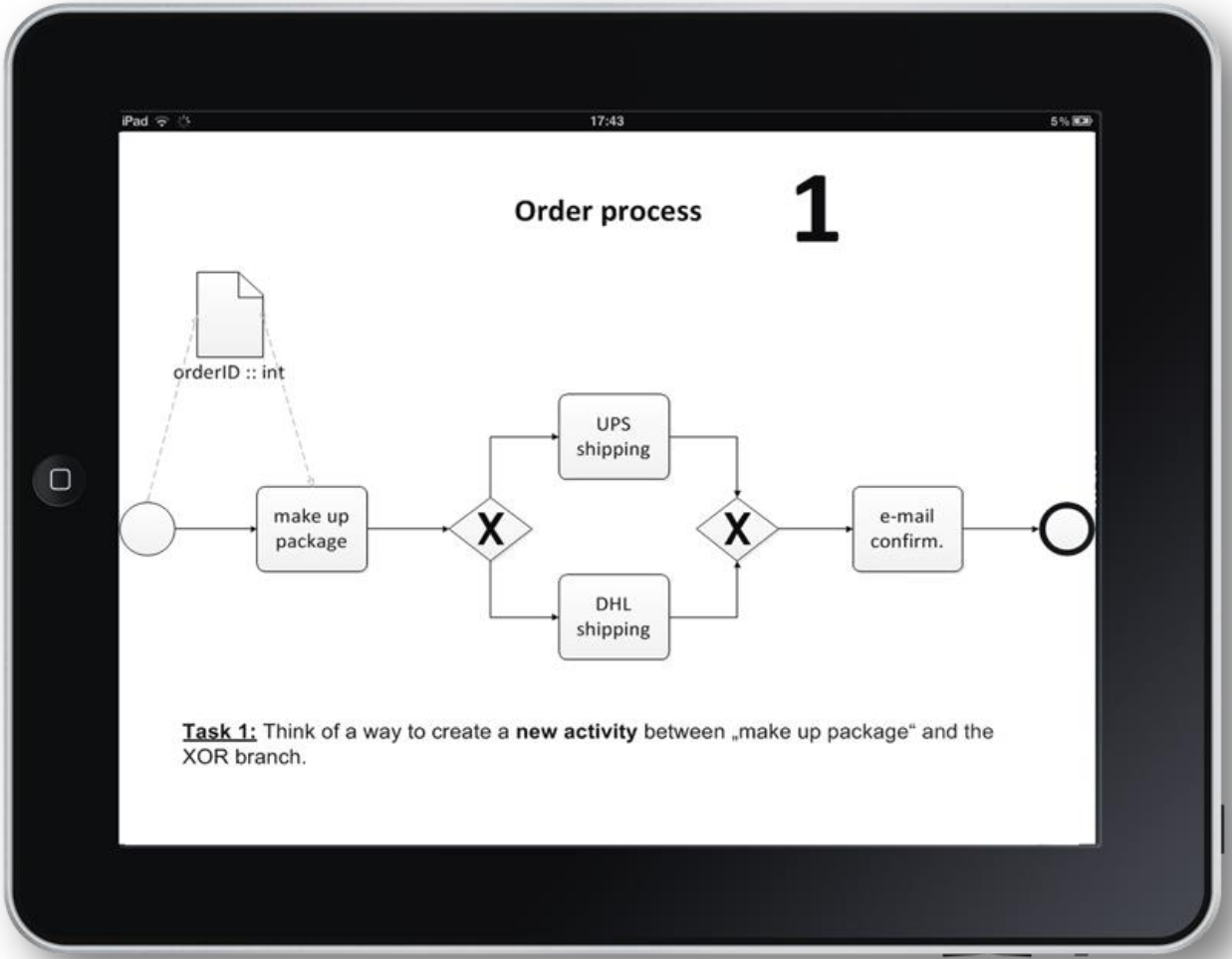
Supported Fingers

Concurrent Users



Interaction Concepts

Gesture-based Interaction with Process Models
First Experiments on Core Gesture Sets



Experiment on Multi-touch Modeling – Results

Picture-based Interaction