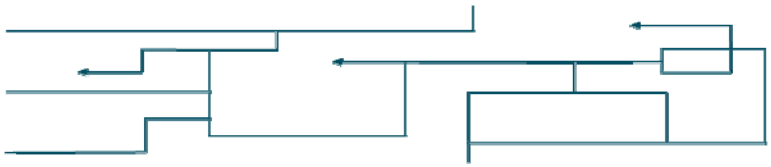
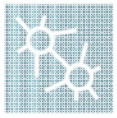
DAI-Labor
TU Berlin

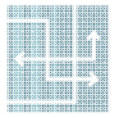


Towards Transformations from BPMN to Heterogeneous Systems

Tobias Küster and Axel Heßler


CC ACT
Agent Core Technologies


BPMN is the new standard modelling notation for all kinds of business processes, and many tools provide a transformation to executable BPEL code. But what about other languages?

We present a BPMN modelling and transformation tool, which is generic enough to support not only BPEL, but which can easily be extended with export features targeting other languages, too.

Content

DAI-Labor
TU Berlin

- Introduction
- The Visual Service Design Tool
- The Transformation Framework
- Transformation to BPEL
- Conclusion

Content

DAI-Labor
TU Berlin

- Introduction
  - Motivation
  - The Problem
  - Our Approach
- The Visual Service Design Tool
- The Transformation Framework
- Transformation to BPEL
- Conclusion

Introduction

DAI-Labor
TU Berlin

- DAI-Labor, TU Berlin
  - Prof. Dr.-Ing. Sahin Albayrak
  - *"Smart Services and Smart Systems"*
  - AI, Agents, Networks, Services, Security, …

  http://www.dai-labor.de

- The SerCHo Project (Service Centric Home)
  - services for the home of tomorrow
  - service creation and deployment

  http://www.sercho.de

Introduction: Motivation

DAI-Labor
TU Berlin

- The Business Process Modeling Notation...
  - is a standardised, intuitive process notation.
  - communicates processes between stakeholders, domain experts, and developers.
  - unifies business process modelling.
  - provides a graphical notation for BPEL.
- What about other executable languages?

Introduction: The Problem

DAI-Labor
TU Berlin

- Most BPMN editors are specialized on creating executable BPEL code
  - e.g. *Intalio BPMS*, *eClarus SOA Architect*, …
- This has some advantages:
  - editor support for e.g. BPEL expression syntax
  - code can easily be generated and deployed
- But: Loss of language-independence
  - for a transformation to another language the diagram has to be recreated in another editor

Introduction: Our Approach

DAI-Labor
TU Berlin

- Use sheer BPMN in metamodel and editor
- Transformations can be 'plugged in'
  - additional export / import features
  - additional editing support for these languages
- Make it easy to create new export features
  - third-parties can provide a transformation to *their* execution language

Content

DAI-Labor
TU Berlin

- Introduction
- The Visual Service Design Tool
  - Background
  - Basics
  - Features
- The Transformation Framework
- Transformation to BPEL
- Conclusion

The Visual Service Design Tool: Background

DAI-Labor TU Berlin

- Motivation: Having a tool to '*bridge the gap*'
  - visual creation of distributed systems
  - transformation to Multiagents-Systems (MAS)
    - mapping to MAS still under development
    - export to BPEL as a by-product of this project
- To be used in the SerCHo-Project at TU Berlin
  - goal: creating and combining ambient home services
  - integrating heterogeneous service technologies
- First version implemented as a diploma thesis

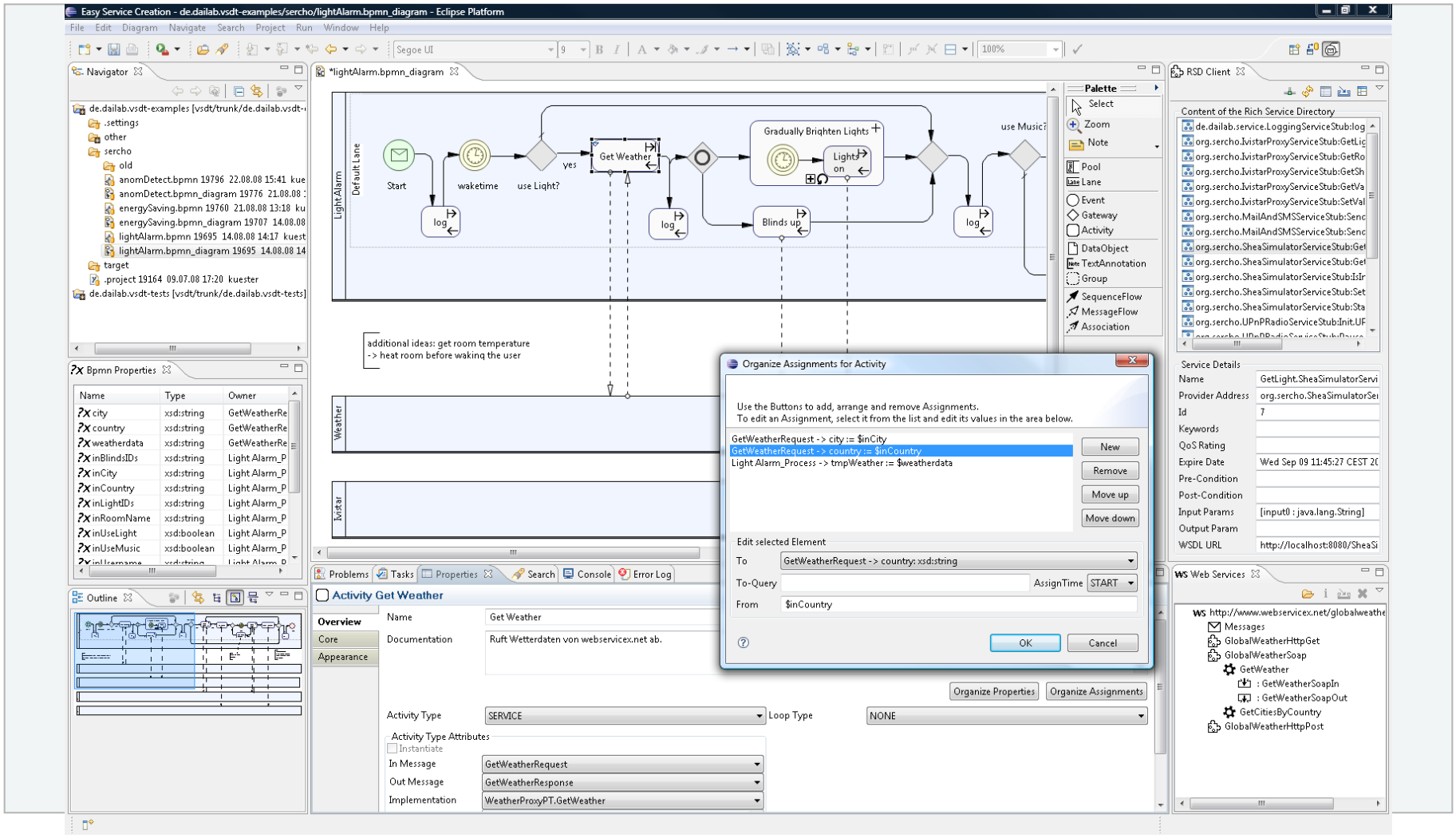
The Visual Service Design Tool: Basics

DAI-Labor TU Berlin

- Metamodel derived from specification
  - fairly Complex, but hidden from the user
  - extension for structures (Sequence, If-Else, ...)
    - used for structure recognition and transformation
- Diagram editor created with Eclipse GMF
  - some customization /additions for improved usability
    - special thanks to *eclipse.modeling.gmf* and the team of the *Eclipse STP* BPMN editor!

The Visual Service Design Tool: Screenshot

DAI-Labor
TU Berlin



The Visual Service Design Tool: Features

DAI-Labor TU Berlin

- Being independent of BPEL has downsides, too
  - reduced editing support, e.g. no syntax validation
- Has to be compensated with plugins, e.g.
  - Web Service View: import existing Web services
  - BPEL Expression Editor (planned): Enter and validate an XPath expression
- Export triggered via Eclipse Export Wizard

Content

DAI-Labor
TU Berlin

- Introduction
- The Visual Service Design Tool
- The Transformation Framework
  - Overview
  - Transformation Stages
  - Transformation Example
- Transformation to BPEL
- Conclusion

DAI-Labor TU Berlin

- Transf. from graph to code is a difficult task
  - concept. mismatch, different expressive power
  - capture global semantics of control flow
  - create readable code one can work with
  - unstructuredness results in added complexity
- Topic of many research papers

Wouldn't it be a good thing to make this part reusable?

The Transformation Framework: Overview

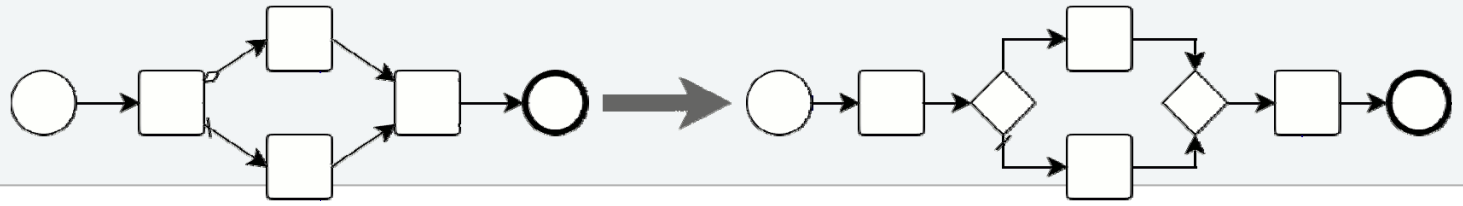
DAI-Labor TU Berlin

- Primary goal: easy to extend and to reuse
- Split up into several stages
  - Validation, Normalization, Structure Mapping, Element Mapping, Clean Up
  - implementing special Interfaces
  - typically realized as a top-down pass or using graph transformation rules (based on EMT)
- Individual stages can be reused or extended for use in other transformations

The Transformation Framework: Stage 1-2

DAI-Labor TU Berlin

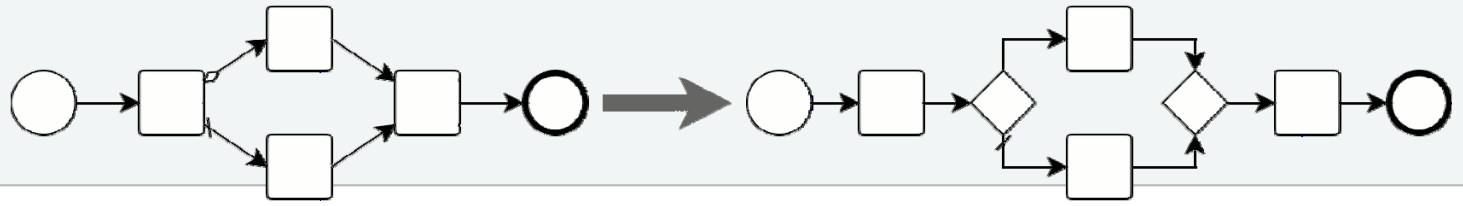
- Validation
  - check constraints, e.g. multiplicities
  - optional: check expression syntax
- Normalization
  - put diagram into "normal form"
  - much fewer cases to consider later on
    - Example:



The Transformation Framework: Stage 2

DAI-Labor TU Berlin

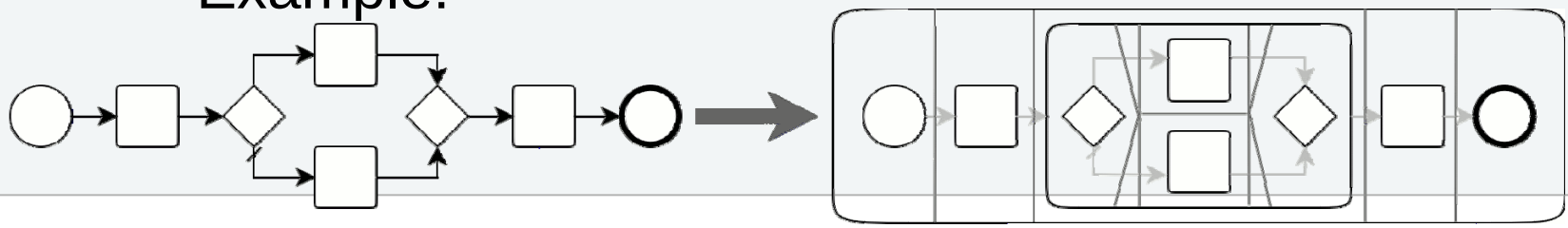
- Normalization Rules
  - insert a Gateway, if an Event or Activity has multiple incoming or outgoing Sequence Flows
  - split a Gateway having both multiple incoming and outgoing Sequence Flows in two
  - insert a no-op Task between two Gateways or an Activity with Event Handlers and a Gateway
  - add a final Gateway, merging all branches
  - move a Gate's Assignments into a separate Task



The Transformation Framework: Stage 3

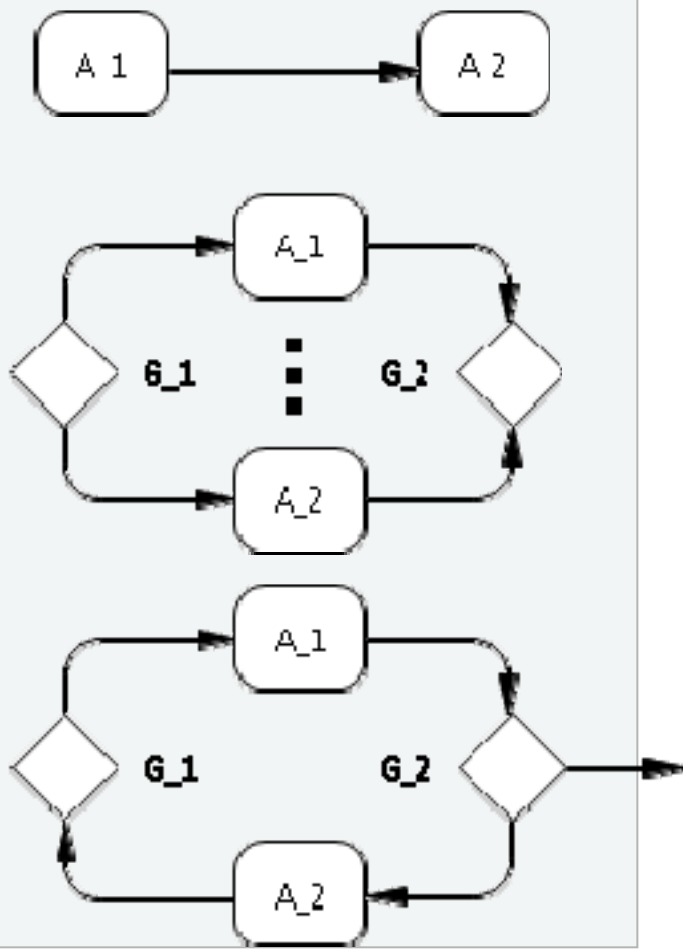
DAI-Labor TU Berlin

- Structure Mapping
  - transformation from graph to block structure
    - following *Structure Identification* strategy
    - realized using a relatively small set of rules
  - can handle slightly unstructured workflows
  - uses extension of the metamodel to integrate recognized structures directly into the model
    - Example:



The Transformation Framework: Stage 3

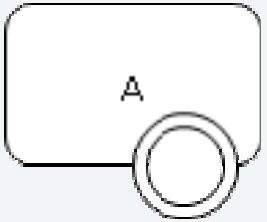
- Sequence Rule
  - pre: not more than one inc./outg. Seq. Flow, no event handlers
  - put Flow Objects in Sequence structure, reroute Seq. Flows
- Block Rule
  - look up other branches going from G1 to G2
  - put Flow Objects and conditions in Block structure, reroute SF
- Loop Rule
  - put Flow Objects and conditions in Loop structure, reroute SF

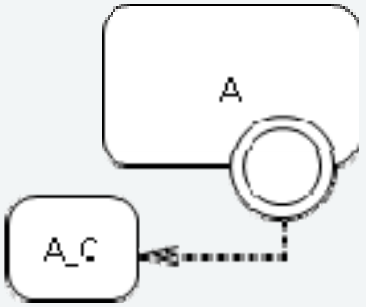


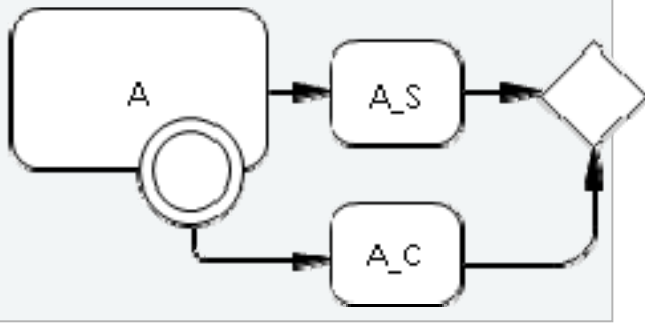
The Transformation Framework: Stage 3

DAI-Labor TU Berlin

- Event Handler Block Rule
  - wrap A into EH Block structure
- Event Handler Comp Rule
  - if A in EH Block, wrap Event and AC into EH Case structure, add to EH Block
  - detach Event from Activity
- Event Handler Skip Rule
  - if A in EH Block, wrap Event and AC into EH Case, add to EH Block
  - refer to AS as "element to skip"
  - detach Event, remove lower SFs







The Transformation Framework: Stage 4-5

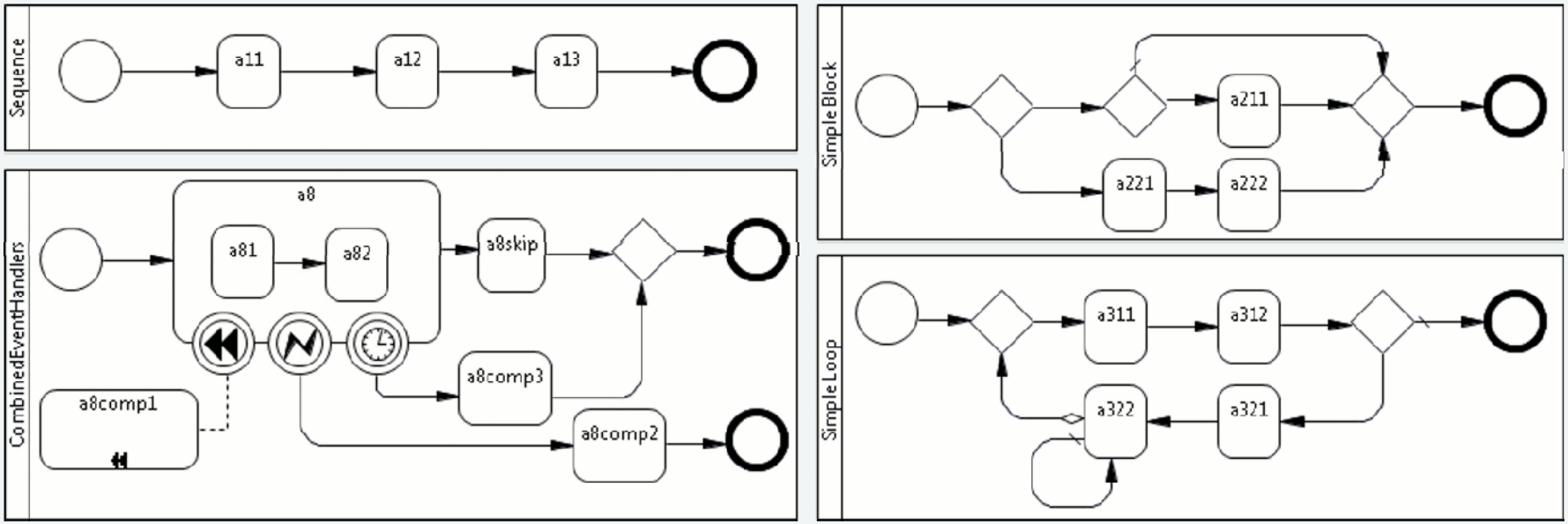
DAI-Labor TU Berlin

- Element Mapping
  - create the target model (e.g. a BPEL process) from the normalized, structured process diagram
  - map each element, e.g. a Task, to its equivalent in the target language
- Clean-Up
  - improve readability of generated code, remove redundant code
  - flatten nested sequences, resolve singleton sequences, ...

The Transformation Framework: Example

DAI-Labor
TU Berlin

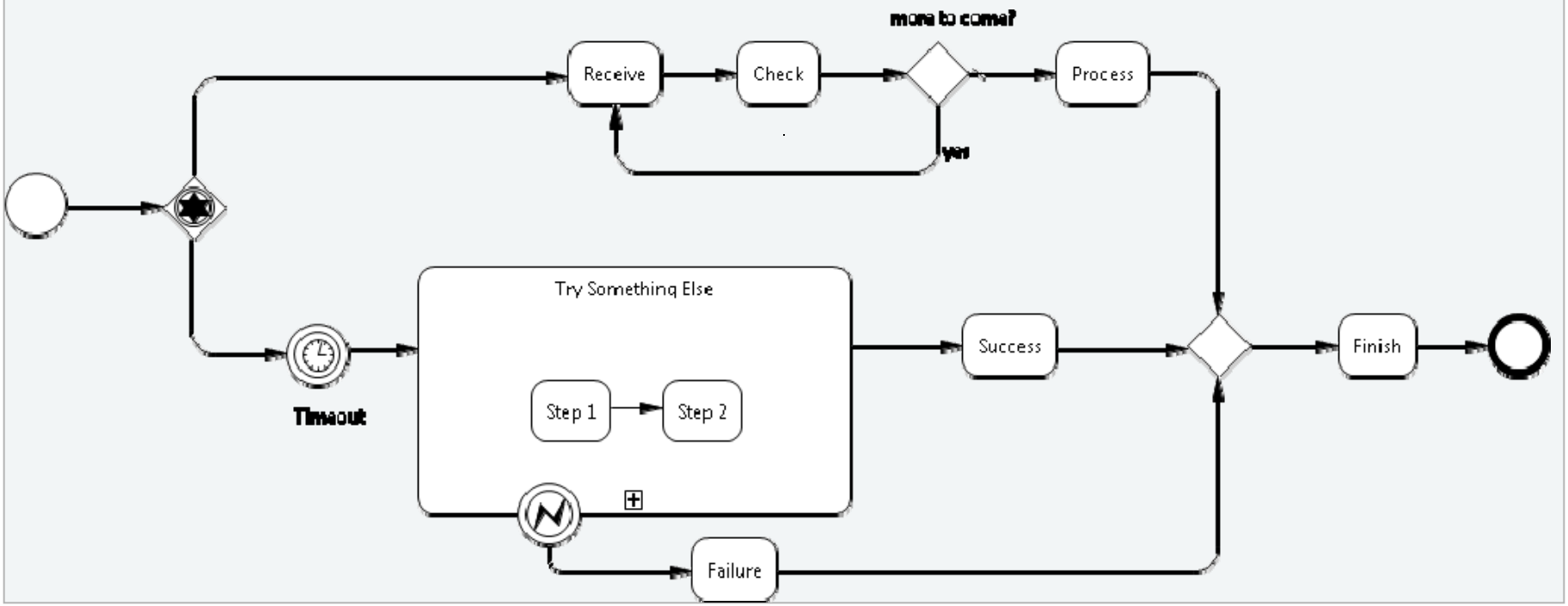
- Examples of transformable workflows



Intermezzo: Example, Step 1

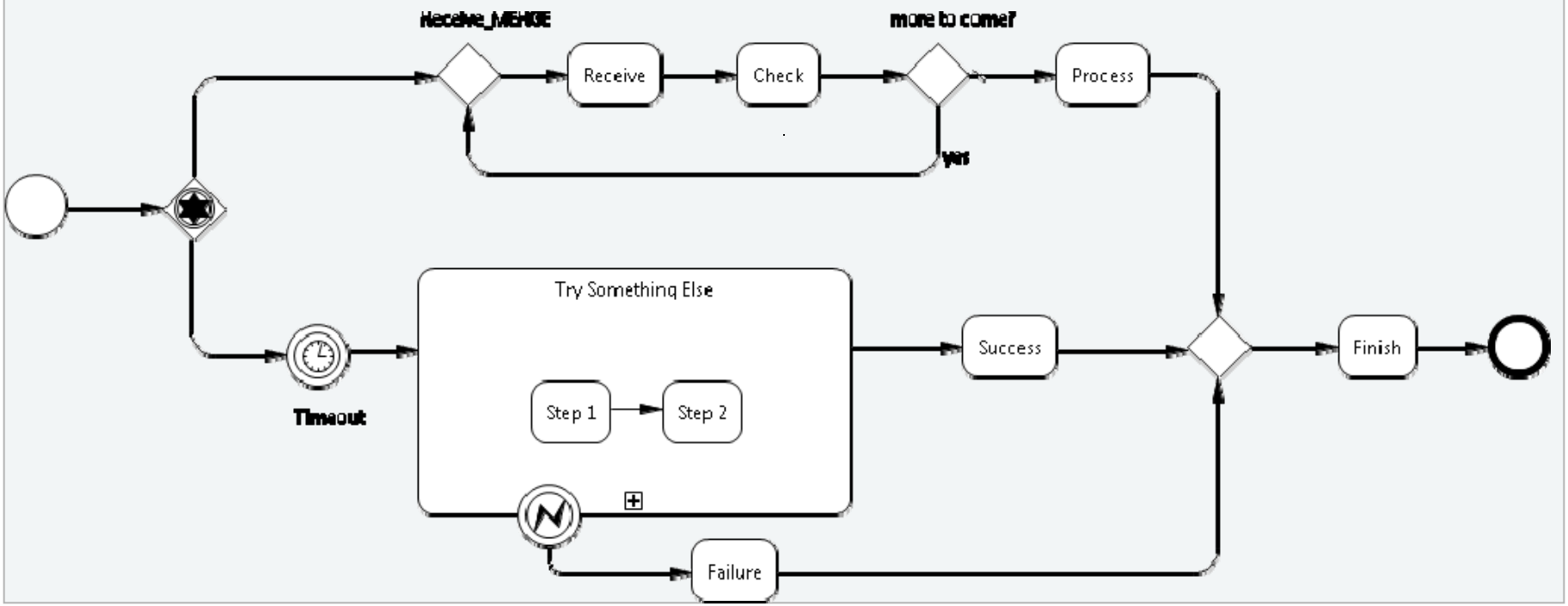
DAI-Labor
TU Berlin

Our starting workflow:



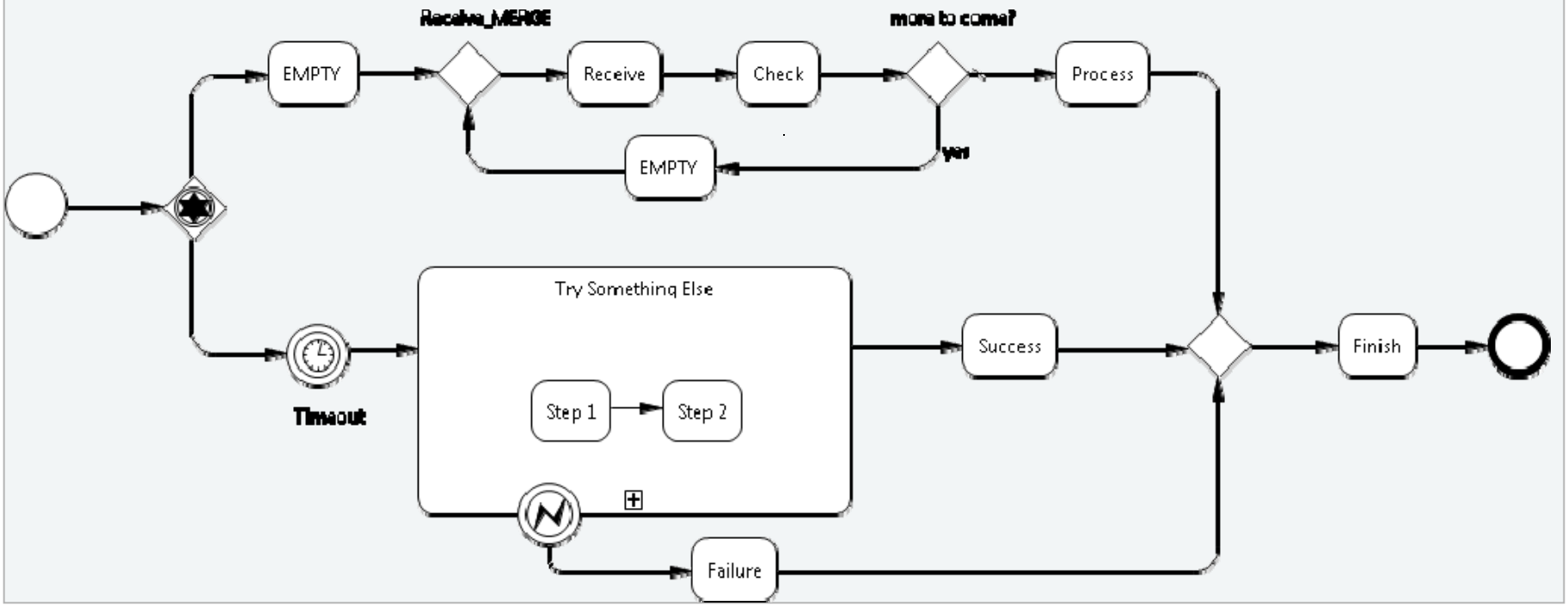
Intermezzo: Example, Step 2

First, insert a Gateway before the Task 'Receive' to simplify the identification of the loop.



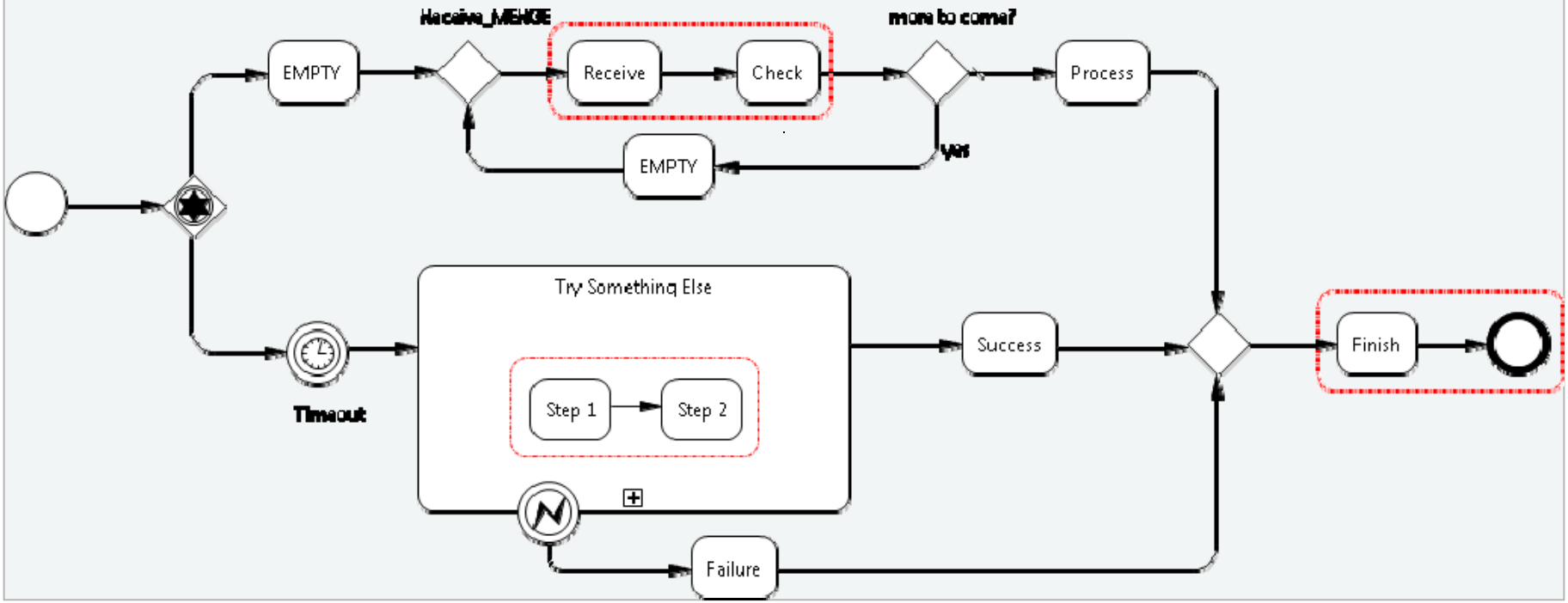
Intermezzo: Example, Step 3

Now, two 'Empty' Activities are inserted between the consecutive Gateways.



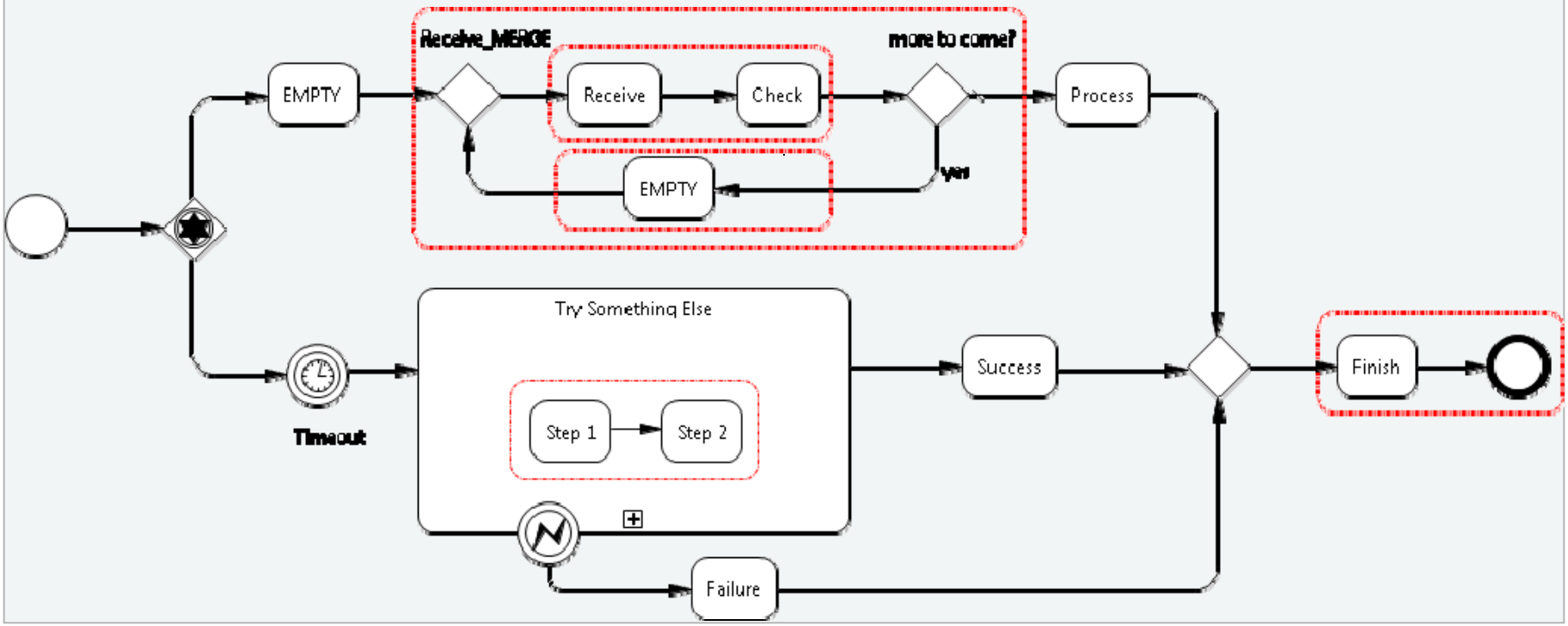
Intermezzo: Example, Step 4

That was the Normalization. The Structure Mapping starts with identifying three Sequences…



Intermezzo: Example, Step 5

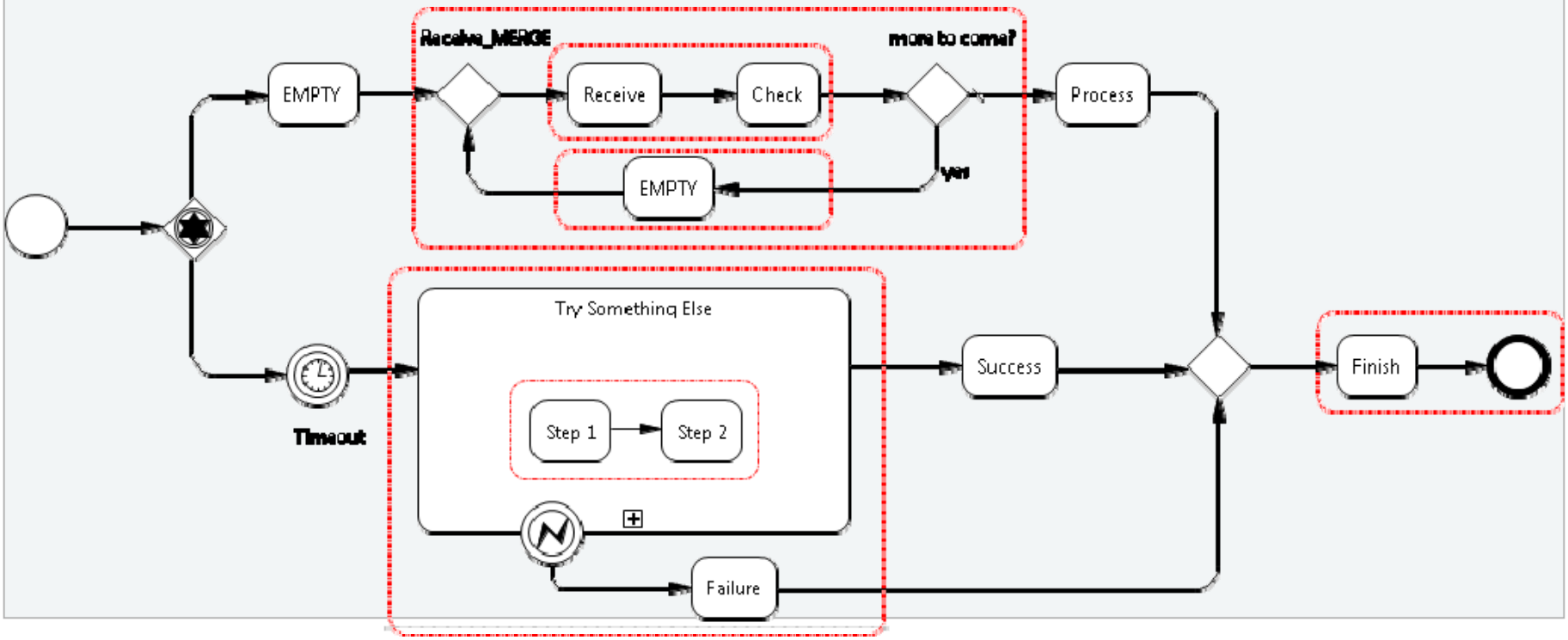
Now that the Gateways are only one Flow Object apart, the Loop is identified.



Intermezzo: Example, Step 6

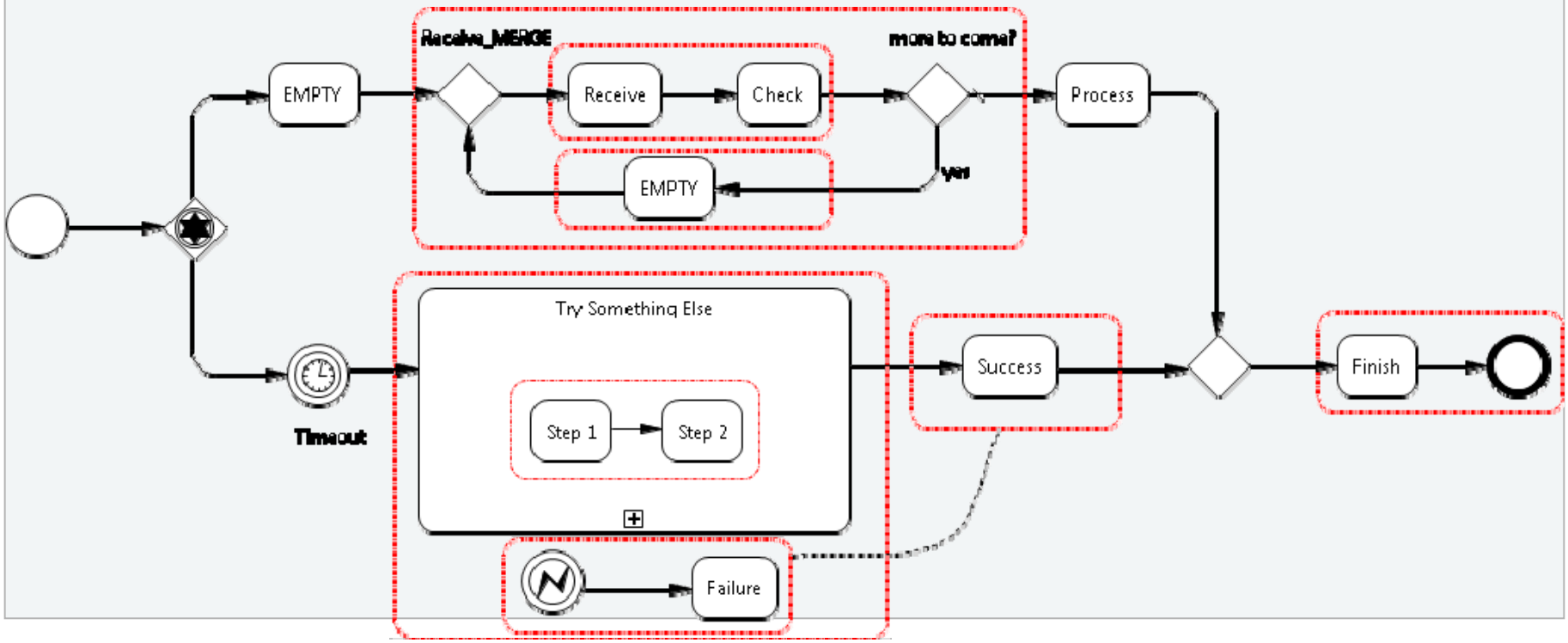
DAI-Labor
TU Berlin

Each Activity with an attached Intermediate Event is recognized as an Event Handler Block.



Intermezzo: Example, Step 7

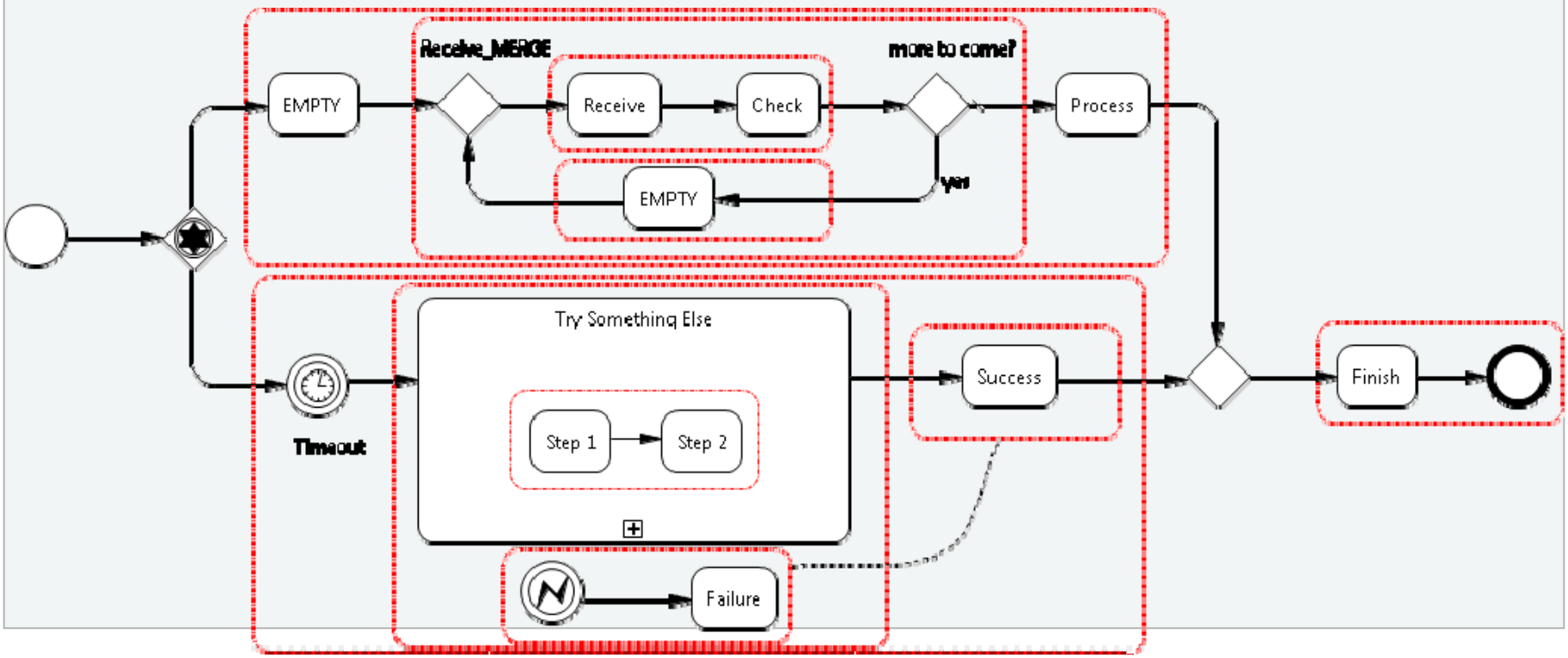
The Compensation Flow is detached from the Activity and put into the Event Handler Block, together with a reference to the Activity to skip.



Intermezzo: Example, Step 8

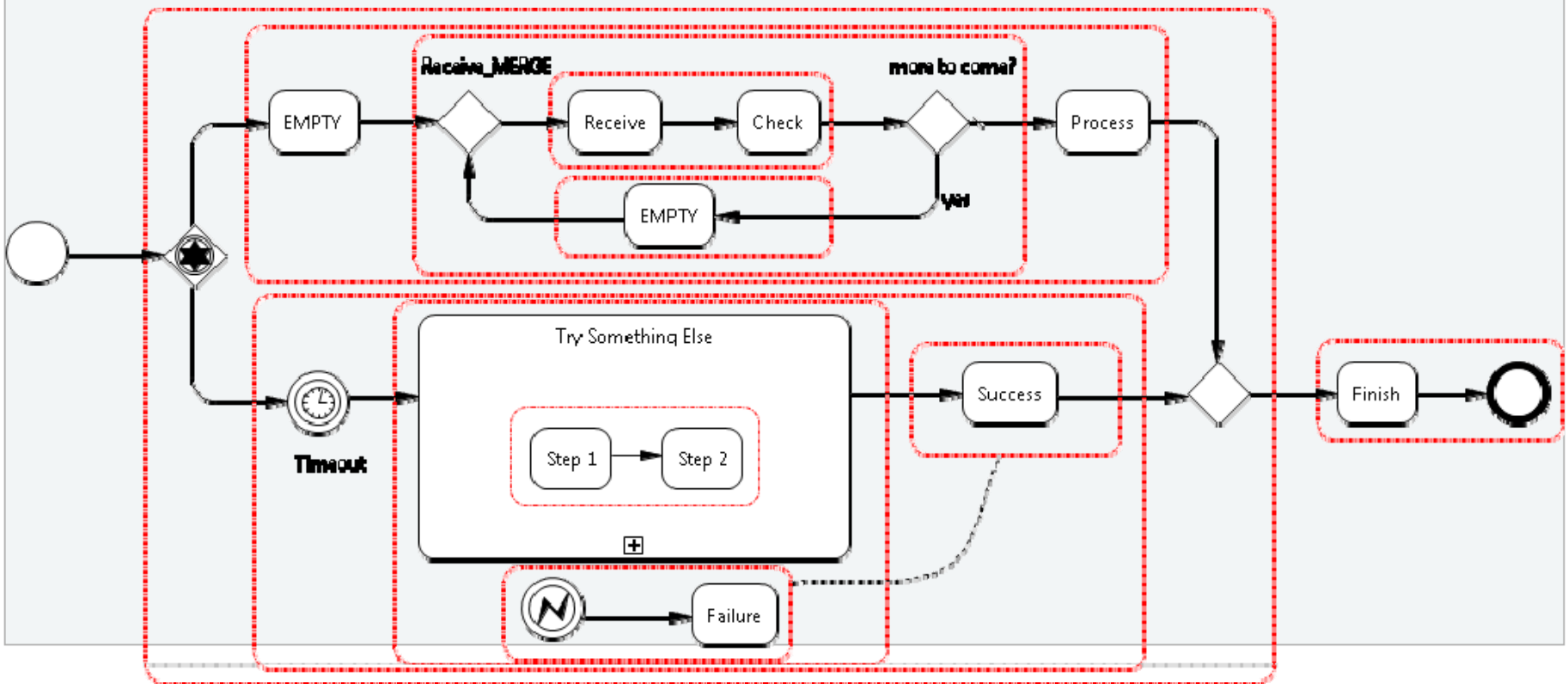
DAI-Labor
TU Berlin

Two more Sequences are wrapped around the branches of the large decision block…



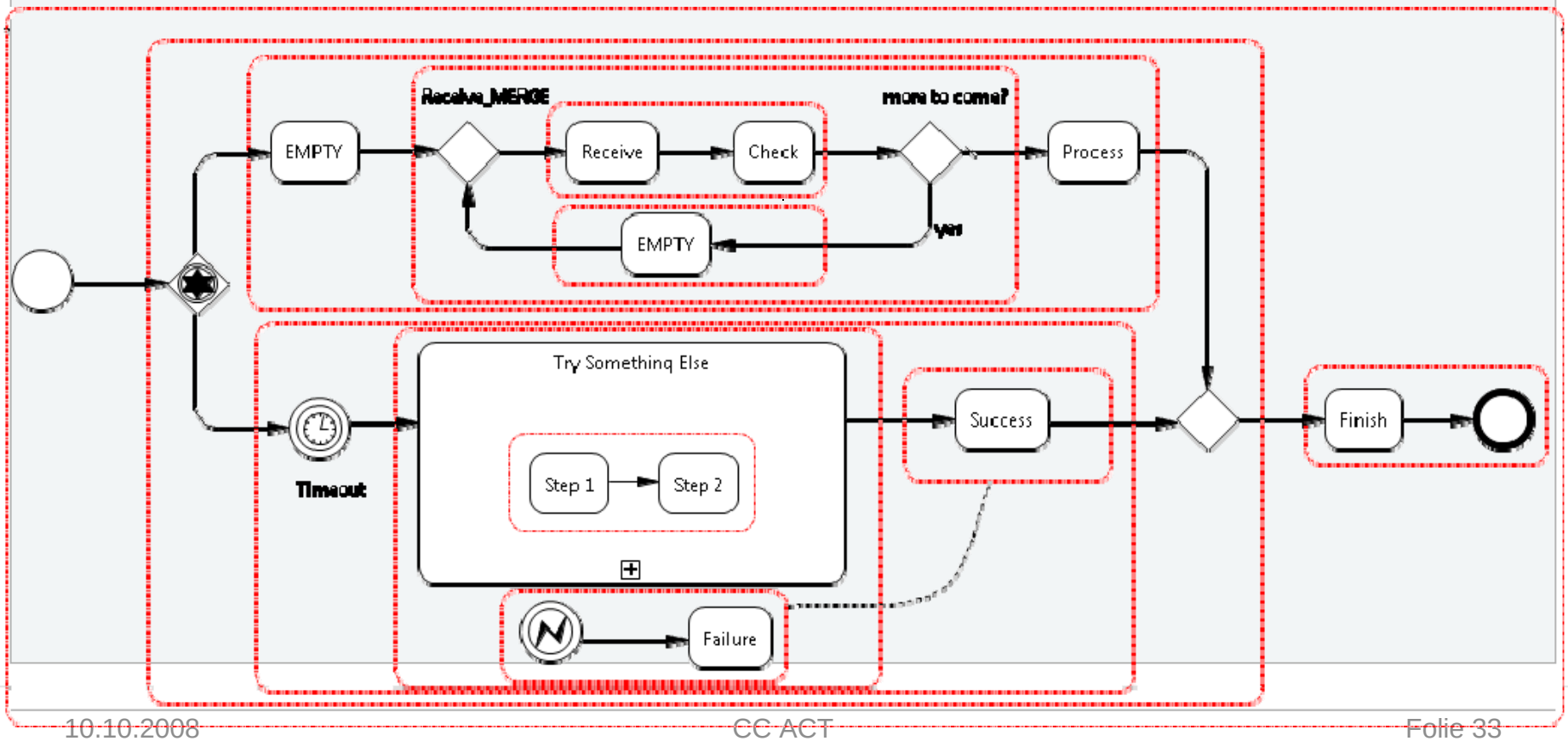
Intermezzo: Example, Step 9

…and in the next step the block itself is identified.



Intermezzo: Example, Step 10

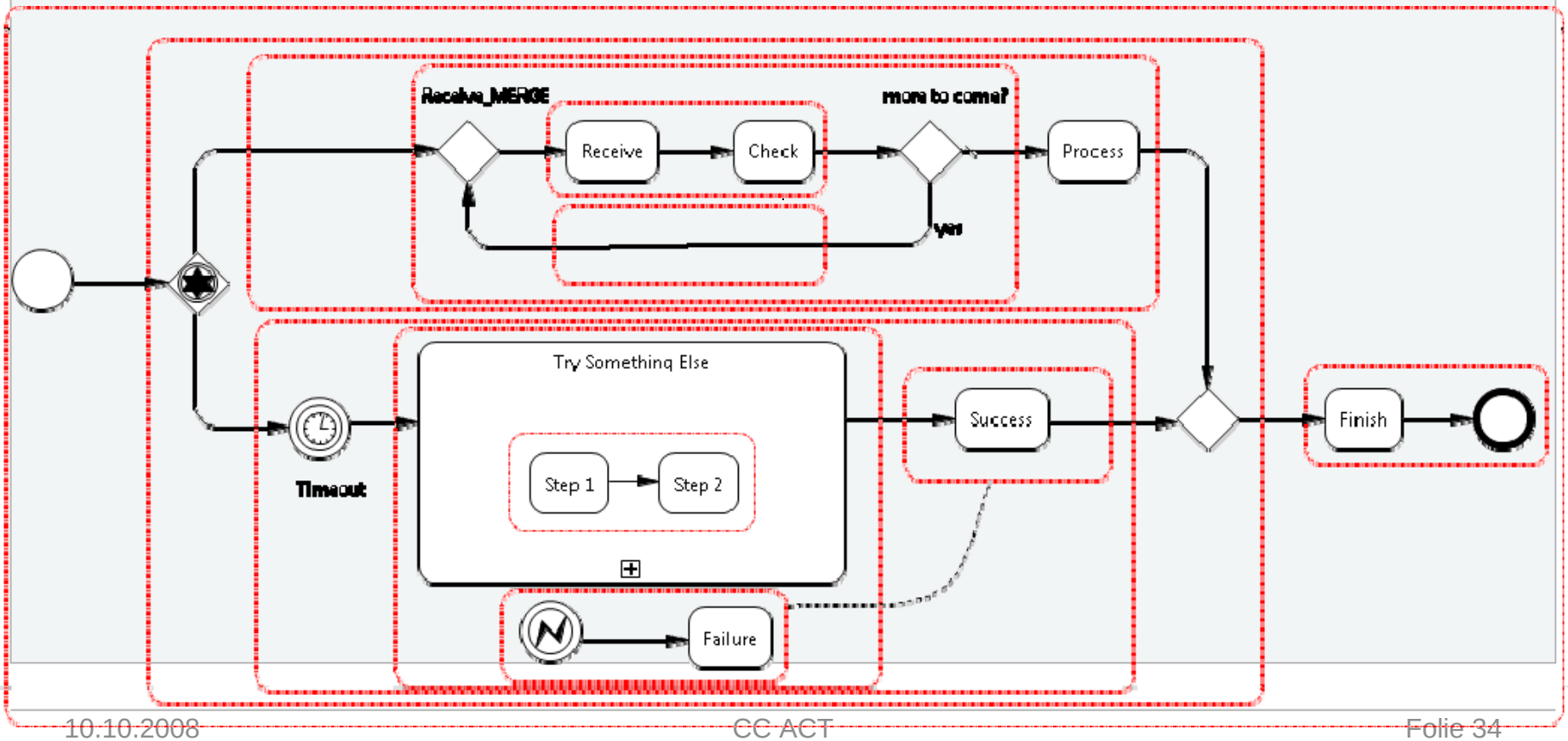
By identifying the outermost sequence the Structure Mapping is finished.



Intermezzo: Example, Step 11

DAI-Labor
TU Berlin

Finally, the 'Empty' Activities inserted earlier are removed.
The result is the input for the Element Mapping stage.



Content

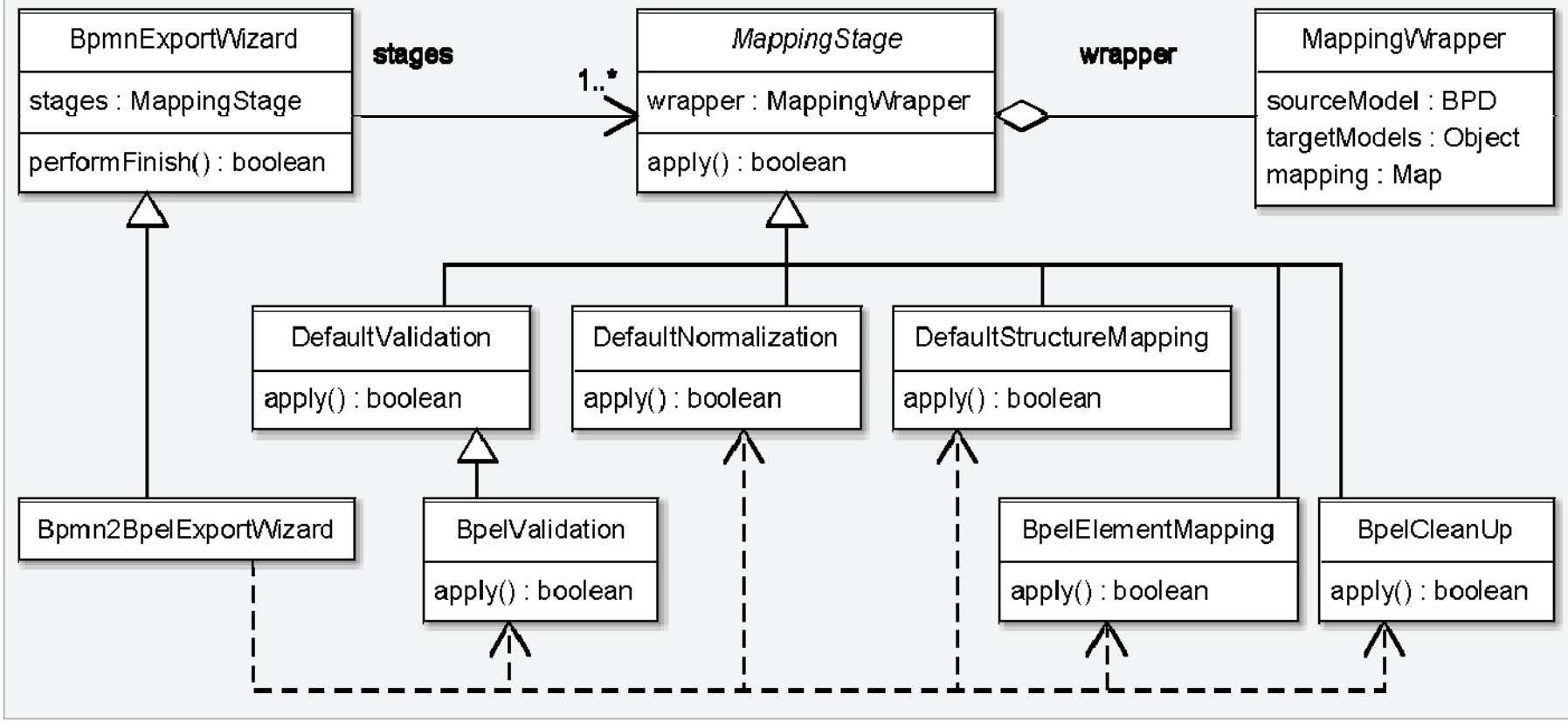
DAI-Labor
TU Berlin

- Introduction
- The Visual Service Design Tool
- The Transformation Framework
- Transformation to BPEL
  - Stages
  - Coverage
  - Tool Demonstration
- Conclusion

Transformation to BPEL: Stages

DAI-Labor
TU Berlin

- Transformation Stages in the BPEL case



Transformation to BPEL: Coverage

DAI-Labor TU Berlin

- Covers nearly the whole mapping
  - Event Handlers, Event-based XOR, OR, …
  - for some elements the mapping is not clear
    - Rule Event, MI Loop, Independent Subprocess, …
    - mapped as described in the specification, but will require some manual rework
  - creates WSDL file holding desc. of orchestr. services
    - **`<binding>`** and **`<service>`** blocks have to be added manually
- Variable name substitution:
  - Given Process **`p`** with Property **`x,`** Expression **`$x+1`** is changed to **`bpws:getVariableData('p_ProcessData','x')+1`**.

DAI-Labor
TU Berlin

Tool Demonstration

Content

DAI-Labor
TU Berlin

- Introduction
- The Visual Service Design Tool
- The Transformation Framework
- Transformation to BPEL
- Conclusion
  - Conclusion
  - Future Work

Conclusion

DAI-Labor
TU Berlin

- Introducing the *'Visual Service Design Tool'…*
  - pure BPMN editor, mappings and language-specific editing support can be plugged in
  - advantage: One BPMN diagram can be exported to arbitrary target languages
- … and a versatile Transformation Framework
  - easy creation and integration of new transformations
  - reusable mapping of graph structure
  - can create readily executable BPEL code
  - more to come!

Conclusion: Future Work

DAI-Labor
TU Berlin

- Future Work
  - further improvement of structure mapping
  - better support for (complex) data types
    - widely disregarded in the BPMN specification
    - both in the editor and during the transformation
  - import from executable code to BPMN
    - still problems with back-transformation of event handlers
  - transformation to further languages
    - focus: Multiagent-Systems
- Long-term goal: Transforming BPMN to Heterogeneous Systems

Contact

DAI-Labor
TU Berlin

DAI-Labor
TU Berlin

Dipl.-Inform. Tobias Küster

Researcher
Competence Center
Agent Core Technologies

+49 (0) 30 / 314 – 74 033
+49 (0) 30 / 314 – 74 003
tobias.kuester@dai-labor.de

www.dai-labor.de

DAI-Labor · Technische Universität Berlin · Sekretariat TEL 14
Fakultät IV - Elektrotechnik und Informatik
Ernst-Reuter-Platz 7 · D -10587 Berlin

DAI-Labor
TU Berlin

Dipl.-Inform Axel Heßler

Researcher
Competence Center
Agent Core Technologies

+49 (0) 30 / 314 – 74 028
+49 (0) 30 / 314 – 74 003
axel.hessler@dai-labor.de

www.dai-labor.de

DAI-Labor · Technische Universität Berlin · Sekretariat TEL 14
Fakultät IV - Elektrotechnik und Informatik
Ernst-Reuter-Platz 7 · D -10587 Berlin