Business Process Management

Orchestrations, Choreographies, and Verification

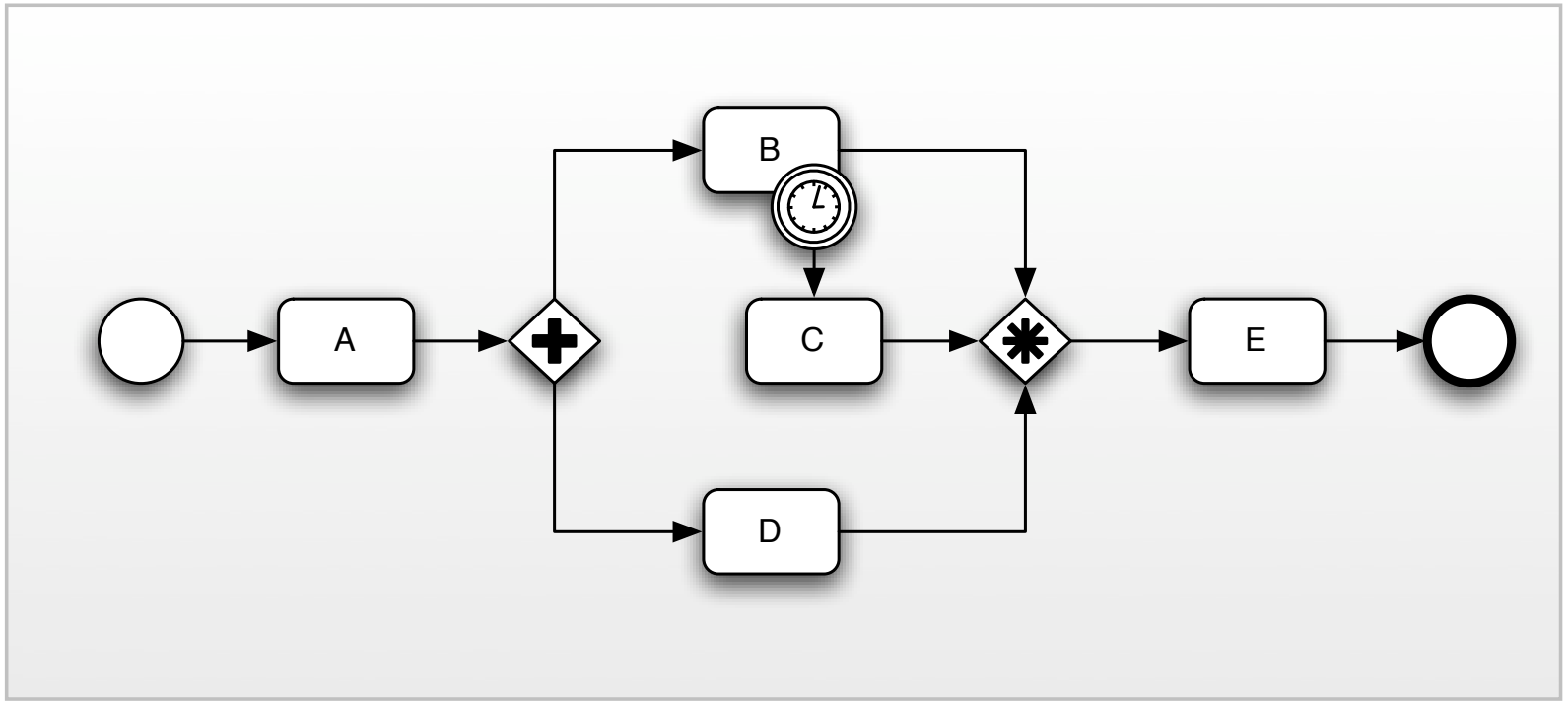
Frank Puhlmann
Business Process Technology Group
Hasso Plattner Institut
Potsdam, Germany

Mapping Graphical Notations

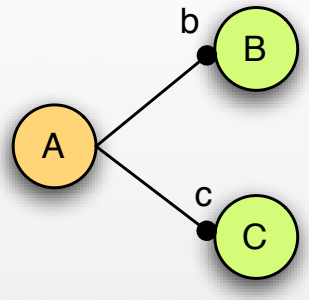
- The Pi-Calculus can be used as a formal foundation for graphical notations; e.g.
  - UML Activity Diagrams
  - BPMN
- Allows for the execution, monitoring, and analysis of these "informal" notations

BPMN2Pi Mapping Steps (Single Pools)

- Assign all flow objects an unique Pi-Calculus agent identifier
- Assign all sequence flows an unique Pi-Calculus name
- “Extend” the Pi-Calculus agents corresponding to the Workflow patterns

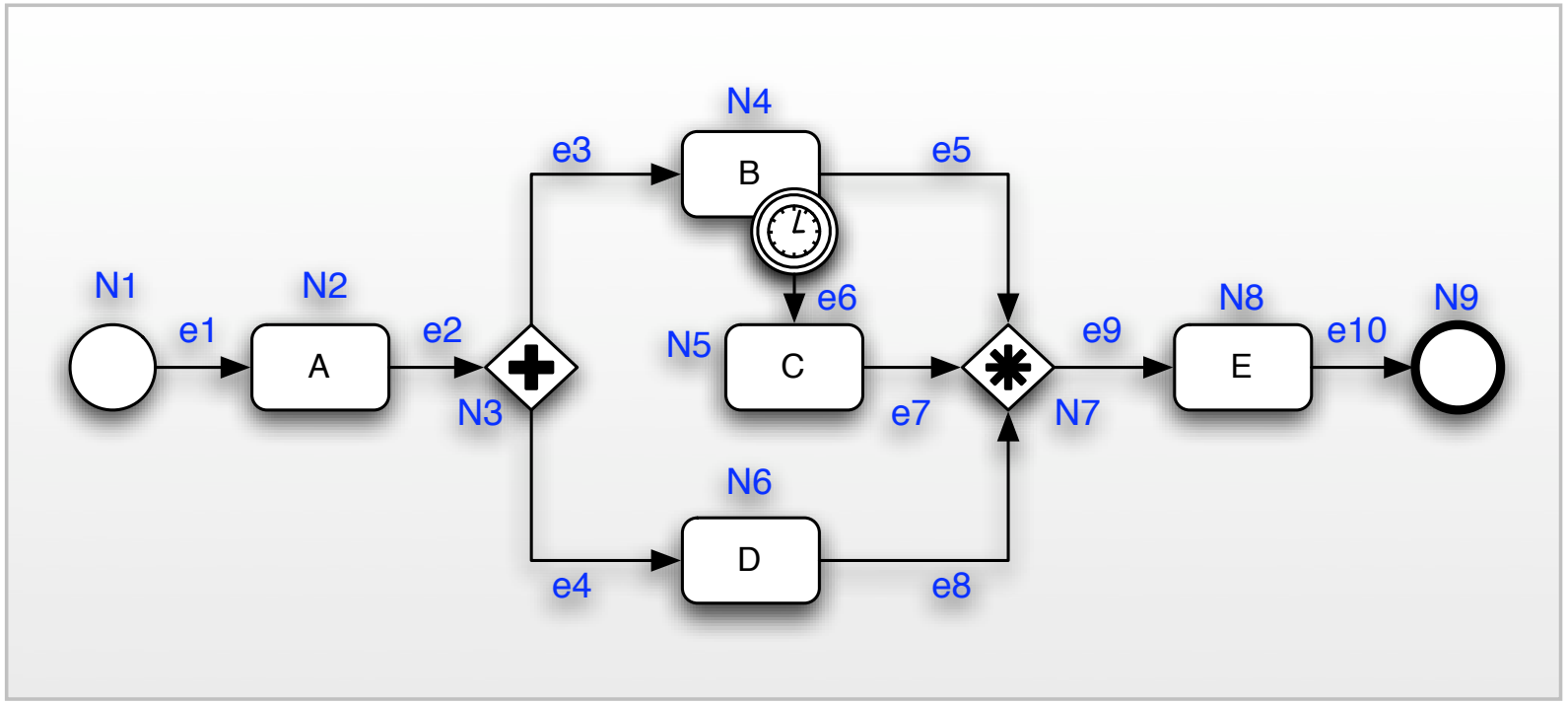


BPMN Example (1)



$$A = (\mathbf{v}check)(A_1|A_2)$$
$$A_1 = \tau_A.check(v).[v = \top]\bar{b}.\mathbf{0}$$
$$A_2 = ir_{\mathcal{E}}.\bar{c}.\overline{check}\langle\bot\rangle.\mathbf{0} + \overline{check}\langle\top\rangle.\mathbf{0}$$
$$B = b.\tau_B.B'$$
$$C = c.\tau_C.C'$$

Event-based Rerouting (Simple Version)

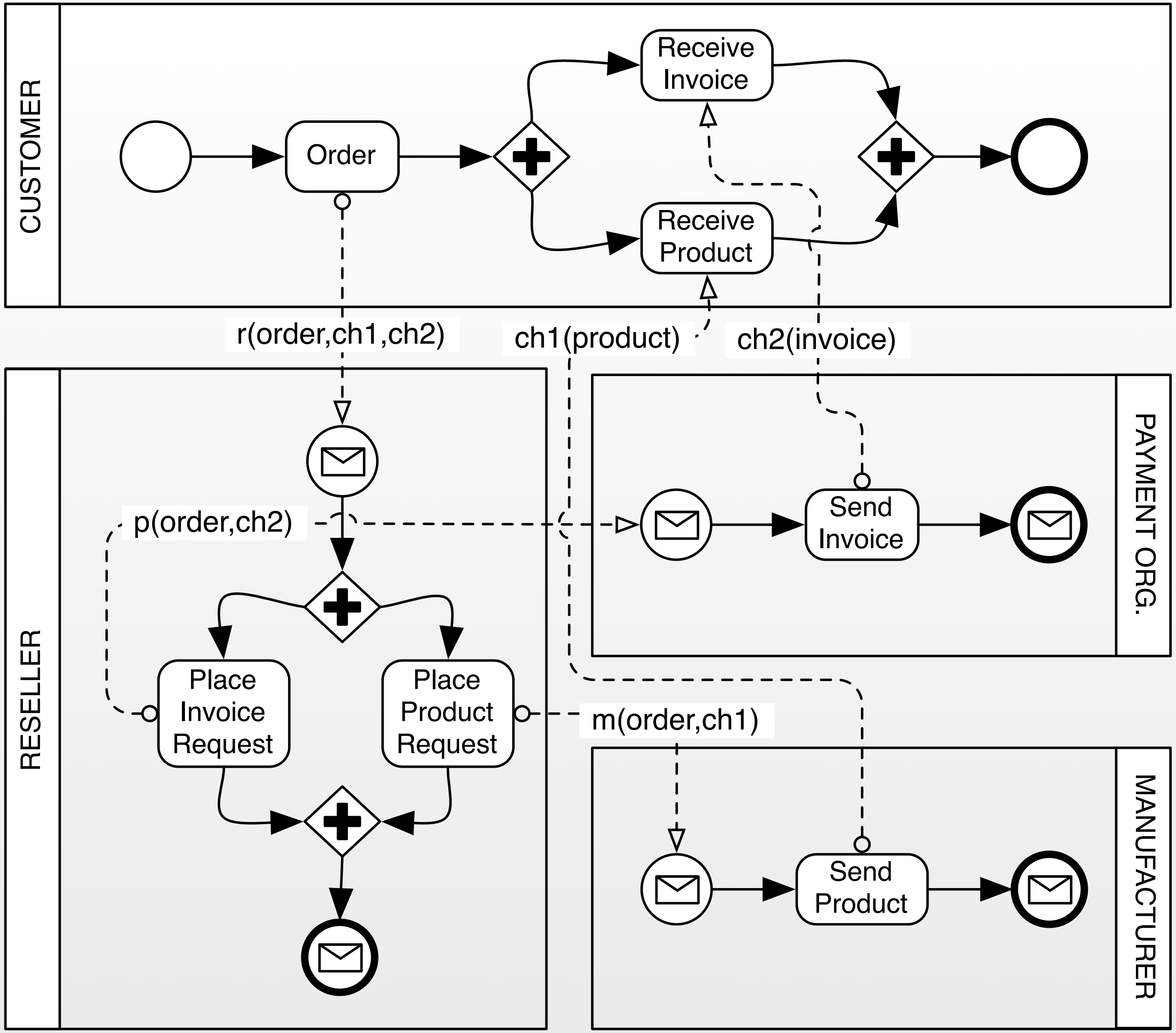


BPMN Example (2)

Choreographies

- Formalized business processes can be combined to choreographies
- Questions:
  - How to represent message flows?
  - How to represent dynamic binding?
  - How to represent correlations?

Example



CUSTOMER
Order
Receive Invoice
Receive Product
r(order,ch1,ch2)
ch1(product)
ch2(invoice)
RESELLER
p(order,ch2)
Place Invoice Request
Place Product Request
m(order,ch1)
PAYMENT ORG.
Send Invoice
MANUFACTURER
Send Product

Dynamic Binding and Correlations

- Idea:
  - Pi-Calculus names are used to represent message flows between a number of processes
- A combination of link passing mobility and scope extrusions realizes dynamic binding directly

Correlations

- A can invoke B several times
- Correlations managed by the restricted name ch:

$$A \stackrel{def}{=} \nu ch\ \overline{b}\langle ch\rangle.(ch(r).A' \mid A)$$

$$B \stackrel{def}{=} \nu r\ b(ch).(\tau.\overline{ch}\langle r\rangle.\mathbf{0} \mid B)$$

Send Interaction Pattern

- Send:

$$A \stackrel{def}{=} \langle\cdot\rangle.\overline{ch}\langle msg\rangle.\mathbf{0}$$

- Static binding:

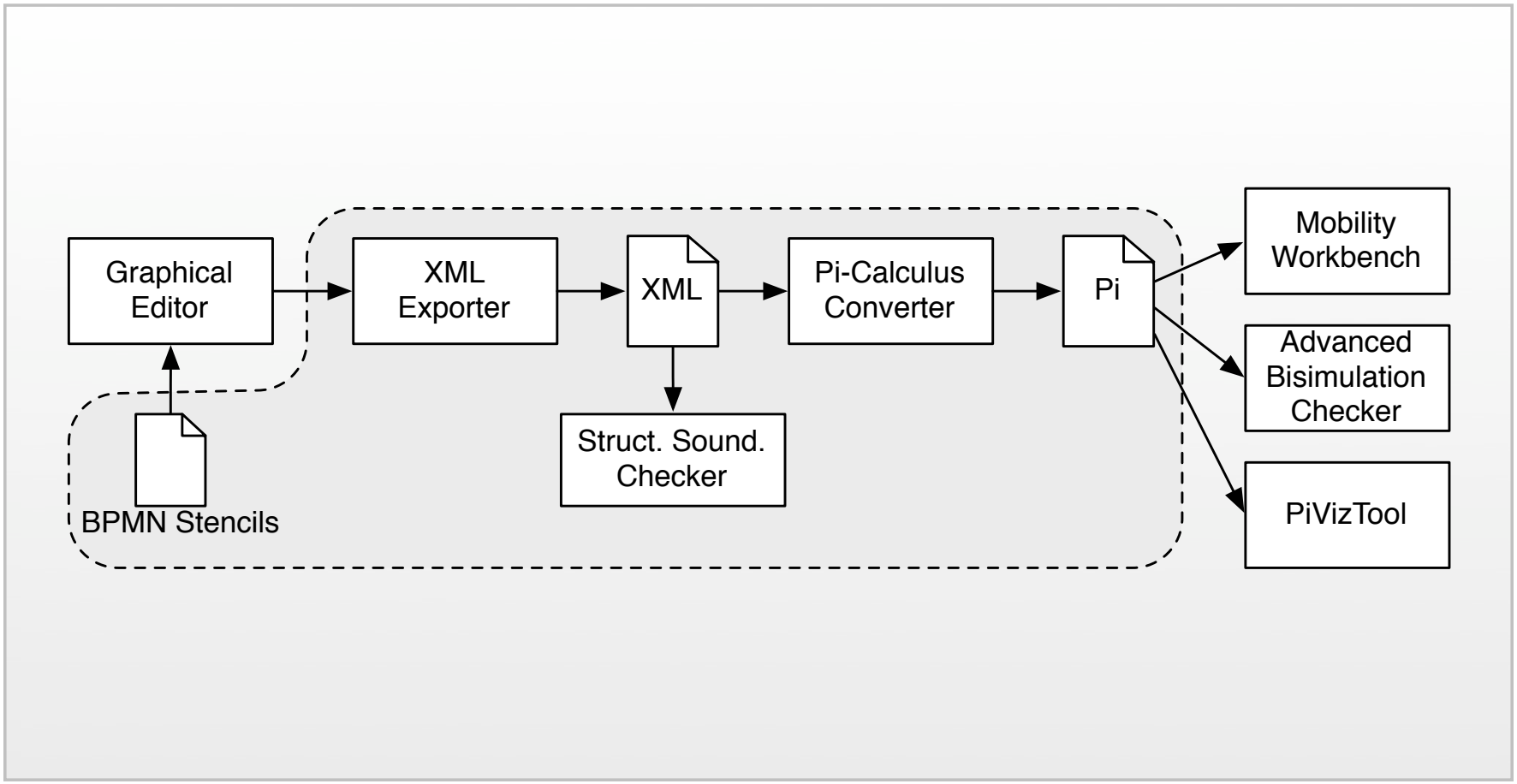
$$I \stackrel{def}{=} \nu ch\ (A \mid E)$$

- Dynamic binding:

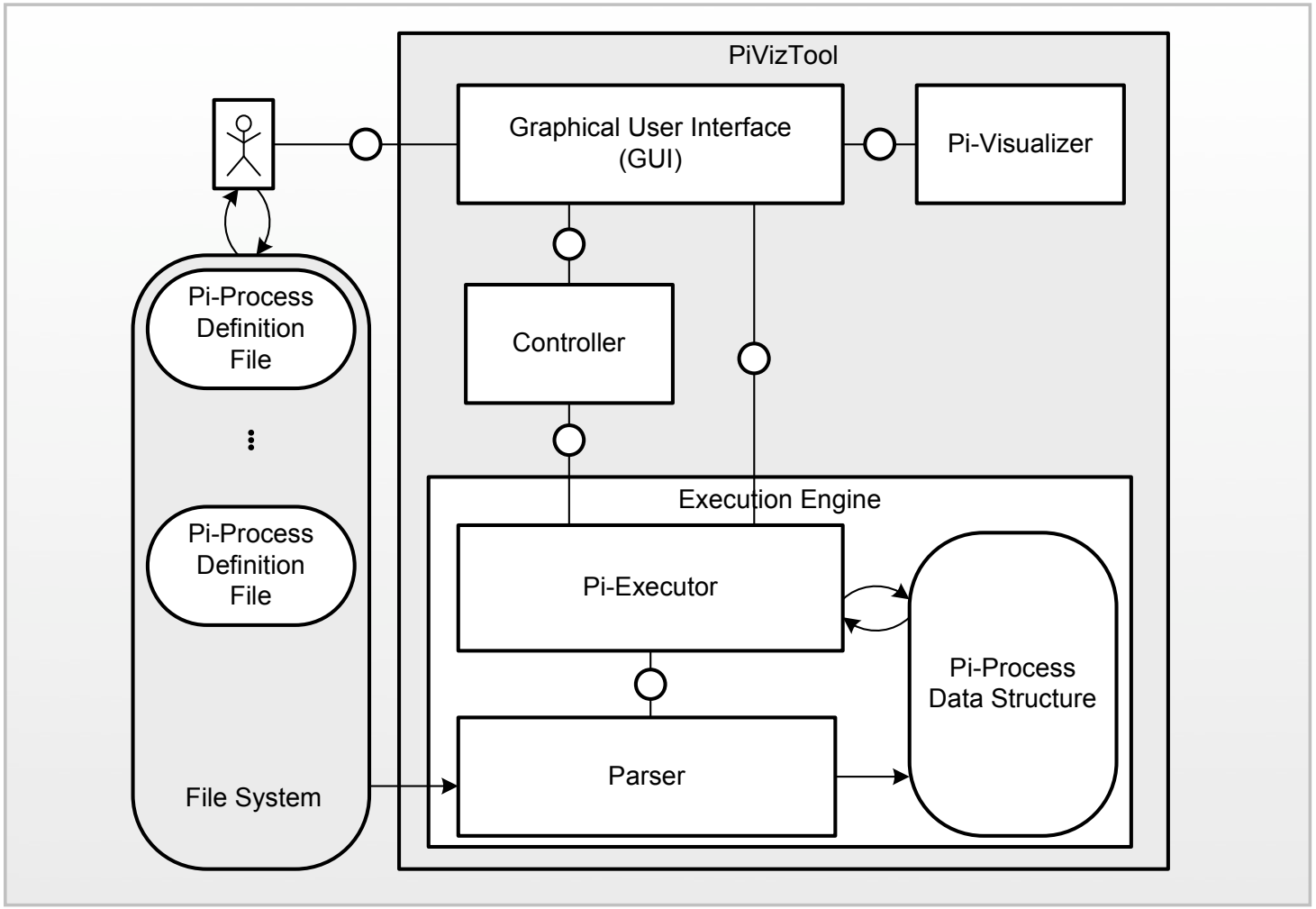
$$I \stackrel{def}{=} \nu lookup\ (lookup(ch).A \mid \mathbb{E})$$

Tool support

- BPMN to pi-calculus mapper
- Graphical pi-calculus simulator optimized for the BPM domain (PiVizTool)
- Reasoners


Graphical Editor
XML Exporter
XML
Pi-Calculus Converter
Pi
Mobility Workbench
Advanced Bisimulation Checker
PiVizTool
Struct. Sound. Checker
BPMN Stencils

Tool Chain


PiVizTool
Graphical User Interface
(GUI)
Pi-Visualizer
Controller
Execution Engine
Pi-Executor
Pi-Process
Data Structure
Parser
Pi-Process
Definition
File
Pi-Process
Definition
File
File System

PiVizTool

Verification

- Formalized business processes can be checked according to
  - Different kinds of soundness
  - Compatibility
  - Conformance

Reasoning about Soundness using Bisimulation Equivalences

- Idea:
  - Use bisimulation to prove invariants of the formalized BPDs
  - Invariants are denoted as „trivial“ agents
- Question:
  - Where to start?

Observables

- What can we observe?
- Reductions
  - Intra-actions
  - Internal actions
- Interactions with the environment?
  - Start Event, End Event, Service Invocations?

Action Semantics

- We‘re interested in observing „certain“ names:
  - All free names of a system
  - These can interact with the environment via matching input and output prefixes not contained in the system
  - Requires a different semantics with a labeled transition system

$$\alpha ::= \overline{x}\langle y \rangle \mid x(y) \mid \overline{x}\langle \mathbf{v}\, z \rangle \mid \tau$$

The LTS Actions

Bisimulation

- Let P and Q be two related agents. If P can evolve to P', then also Q must be able to evolve to Q' such that P' and Q‘ are again related. If the same holds for the opposite direction, starting from Q, the two agents are called bisimilar or bisimulation equivalent.

Weak Bisimulation

- A weak bisimulation relates more agents by stating that an action of P can be weakly mimicked by Q (and vice versa):
  - If P has an action alpha, then also Q has an action alpha enclosed in sequences of tau
  - The length of the tau sequences can be zero (i.e. it includes the previous definition)

Structural Soundness

- According to the definition of a workflow net:
- A business process is structural sound if
  - there exists exactly one initial node,
  - there exists exactly one final node, and
  - each node is on a path in between initial and final node.

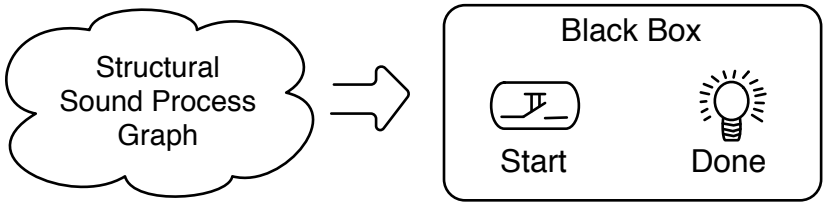
Lazy Soundness

- Key concept:
  - Each structural sound business process should always be able to deliver the result, regardless of the internal actions
- Invariant:

$$S_{LAZY} \stackrel{def}{=} i.\tau.\overline{o}.\mathbf{0}$$

Observation of Lazy Soundness

- Idea: Observation of the Start and End-Events:



- Questions:
  - Waited long enough?
  - Captured all possibilities?

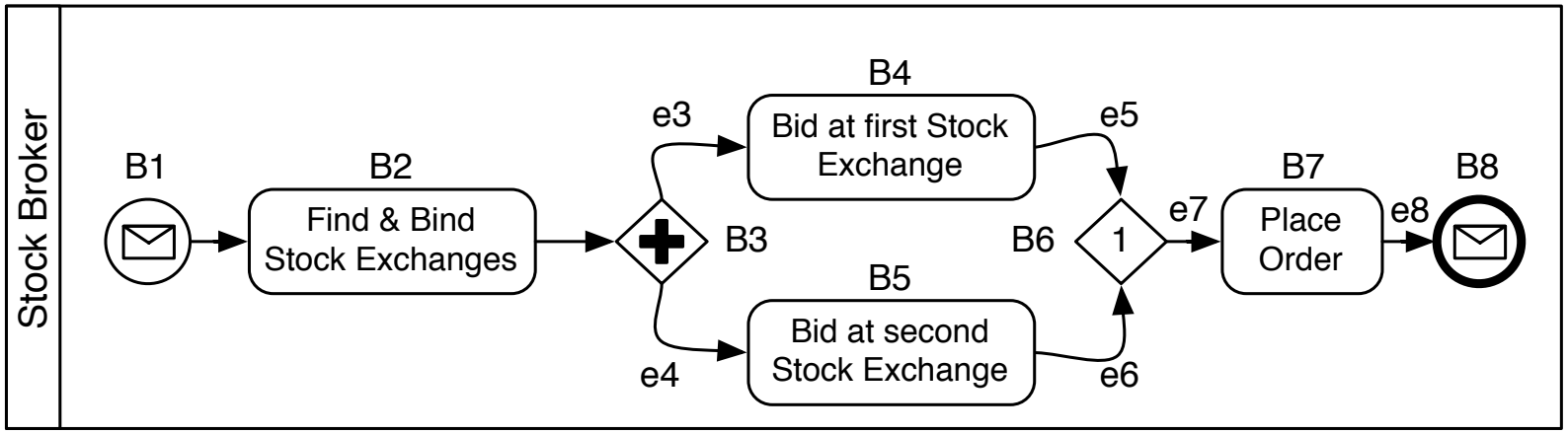
Proving Lazy Soundness

- Lazy soundness can be proved:
  - Map the corresponding business process to agents
  - Annotate the agents representing the initial and the final node with „i“ or „o“ accordingly
- Decide weak bisimulation equivalence between S_LAZY and the mapping

Notes

- Lazy Soundness does not coincidence with existing soundness properties
- Allows activities to be active after the final node has been reached!
  - These are called clean-up, or lazy activities
- Dead activities might be contained
- Requires the distinction between the point in time where a business process delivers the result vs. the moment it terminates

Example

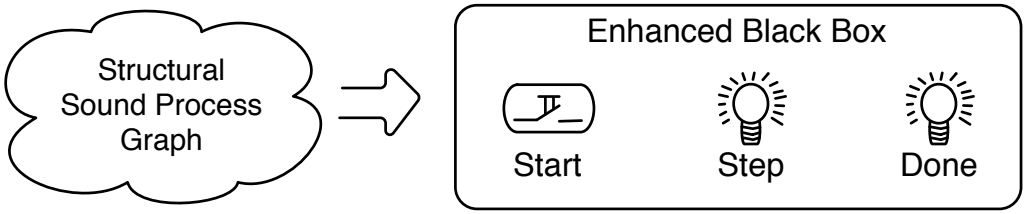

Stock Broker
B1
B2
Find & Bind
Stock Exchanges
B3
e3
e4
B4
Bid at first Stock
Exchange
B5
Bid at second
Stock Exchange
e5
e6
B6
1
e7
B7
Place
Order
e8
B8

Existing Soundness Properties

- Weak Soundness:
  - The delivery of the result denotes the termination of the business process
  - Invariant: The final activity is observed exactly once, and no other activity can be observed after the final node
- Relaxed Soundness:
  - All activities participate in the business process
  - Invariant: Each activity can be observed at least once

Extension of the Black Box

- The black box has to be extended:



- Bisimulation used for weak soundness (must)
- Simulation for relaxed soundness (can)
- Soundness is a combination of weak/relaxed sound

Further Verification

- Compatibility:
  - Lazy soundness can be extended to „Interaction Soundness“ representing a compatibility notion with support for dynamic binding
- Conformance:
  - Bisimulation can be used as a conformance notion

The End.