

Rule-based business process modeling and execution

Linking rule enforcement to
process execution

A model-driven approach

“a model-driven approach to enterprise computing in a networked economy”

- I. How to include Business Rules in Enterprise Models?
- II. How to link rule enforcement and application execution?

I. How to include Business Rules in Enterprise Models?

BR Approach: avoid duplication of logic

The Business Rule Approach [Ross]

- business logic reflects business policy
- observation: business logic is **uplicated** in many enterprise information systems
- result: changes in business policy result in an **avalanche of required application updates**

...by separating rules from applications

A business rule is

- an atomic unit of business logic
- susceptible to change
- crosscuts application components
- a constraint, derivation or reaction [Wagner]

“Avoid logical duplication of business rules to ensure

1. independent development
2. independent evolution

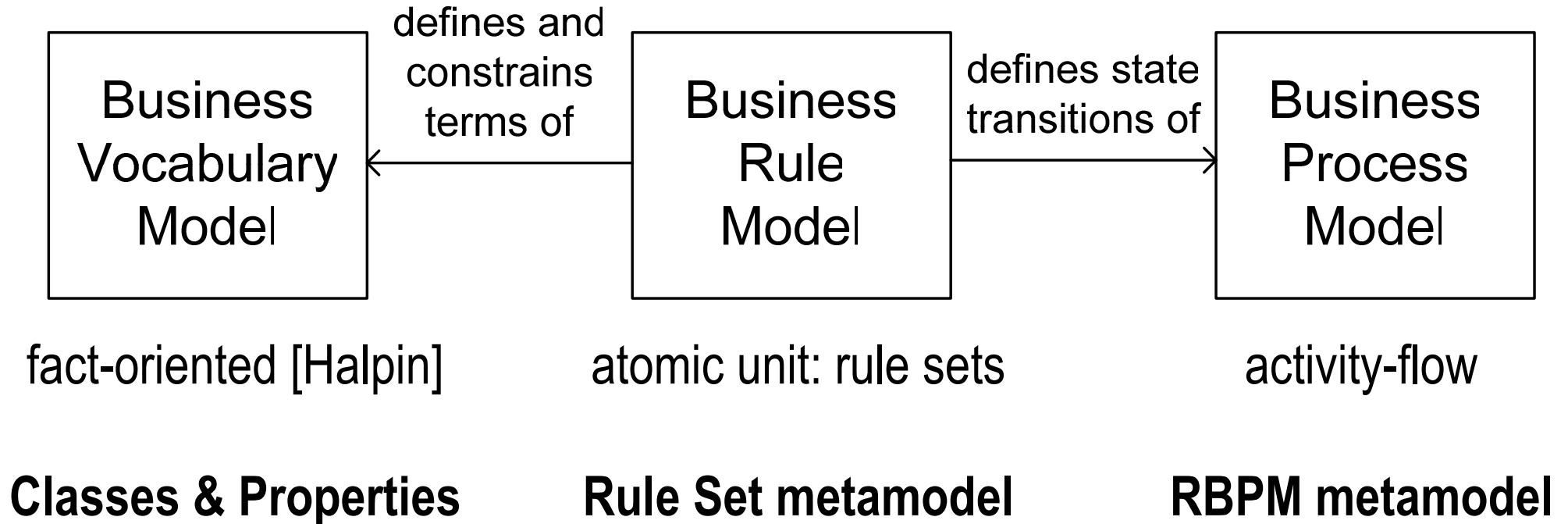
of business rules and applications”

The advantages

- software development advantage
 - + shortens development cycles
 - + reduces the IT bottleneck
 - + facilitates integration
- business advantage
 - + captures hidden business policies
 - + communicates business policies with stakeholders
 - + automates customization

Business Rules bridge the Business-IT gap

Business Rules in Enterprise Models

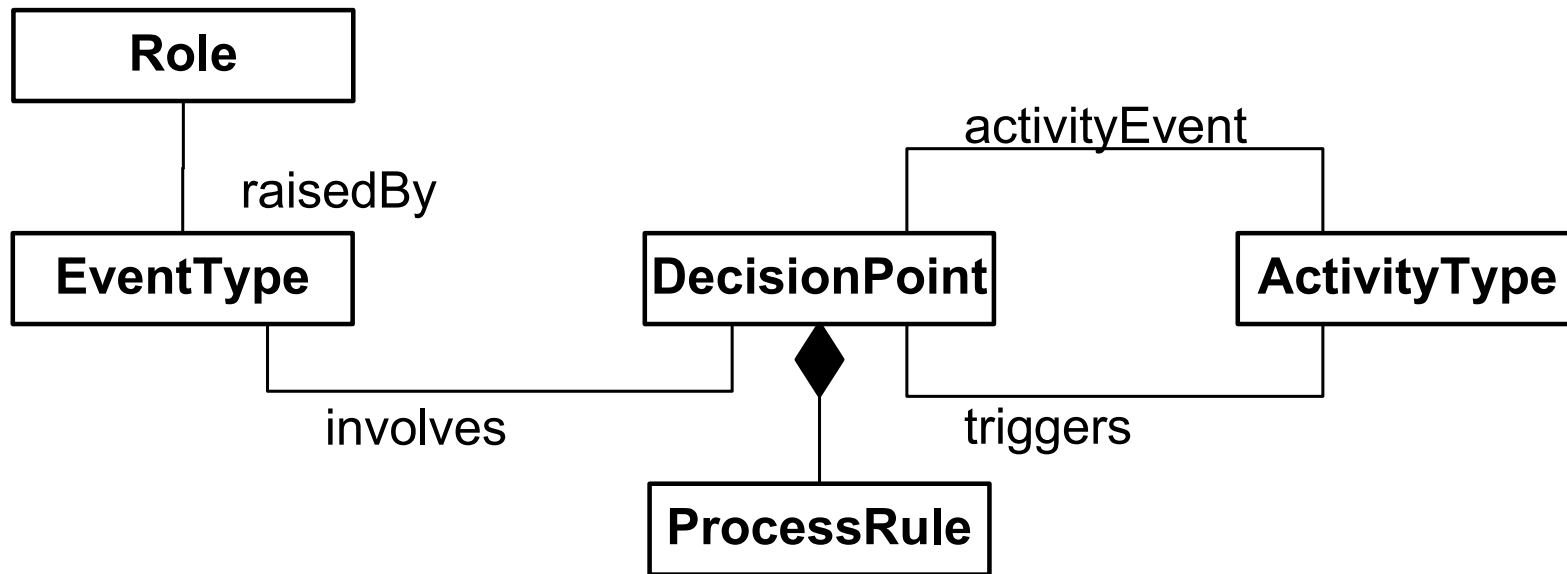


A rule-based business process metamodel

There are three perspectives to process modeling
[Van Der Aalst]

- 1. The activity-flow perspective**
- 2. The case data perspective**
3. The resource perspective

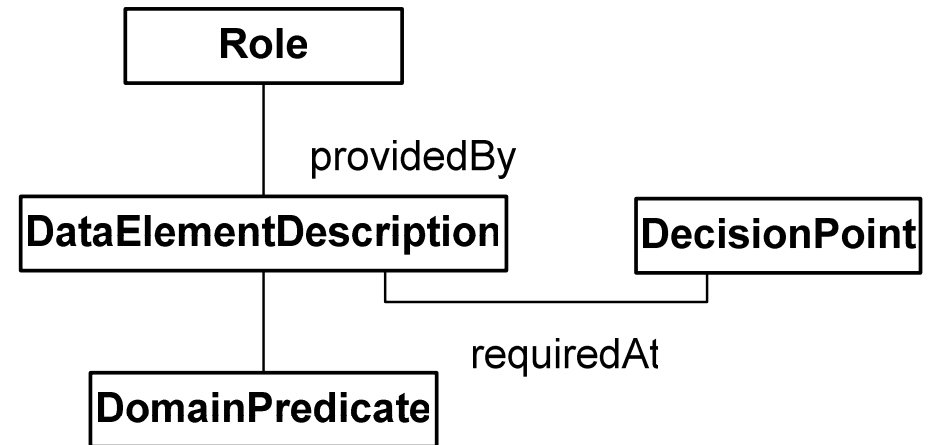
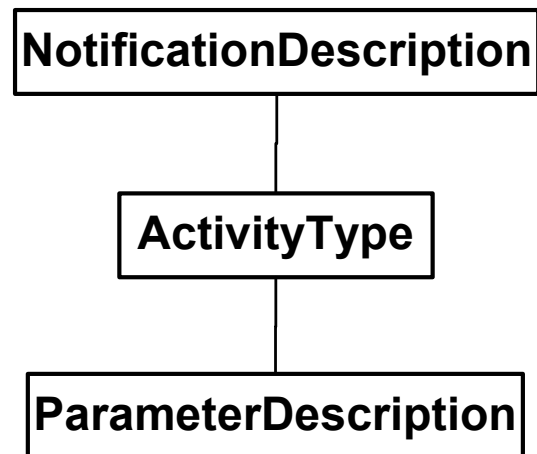
RBPM metamodel: activity flow



- + complex events (AND, OR, NOT)
- + long-running activities (activity events, activity states)

RBPM metamodel: case data

- fact-level granularity
- case data requirements
- dynamic case data acquisition (no fixed messages)



- NotificationDescription contains *EventType*, *DataElement-Description* of notification messages
- *ParameterDescription* of activity messages

The case for process rules

- + easy integration with rule-based data semantics
- + complex events
- + long-running activities
- + dynamic case data acquisition (no fixed messages)
- no formal validation facilities, however...
- comprehensibility at stake, however...

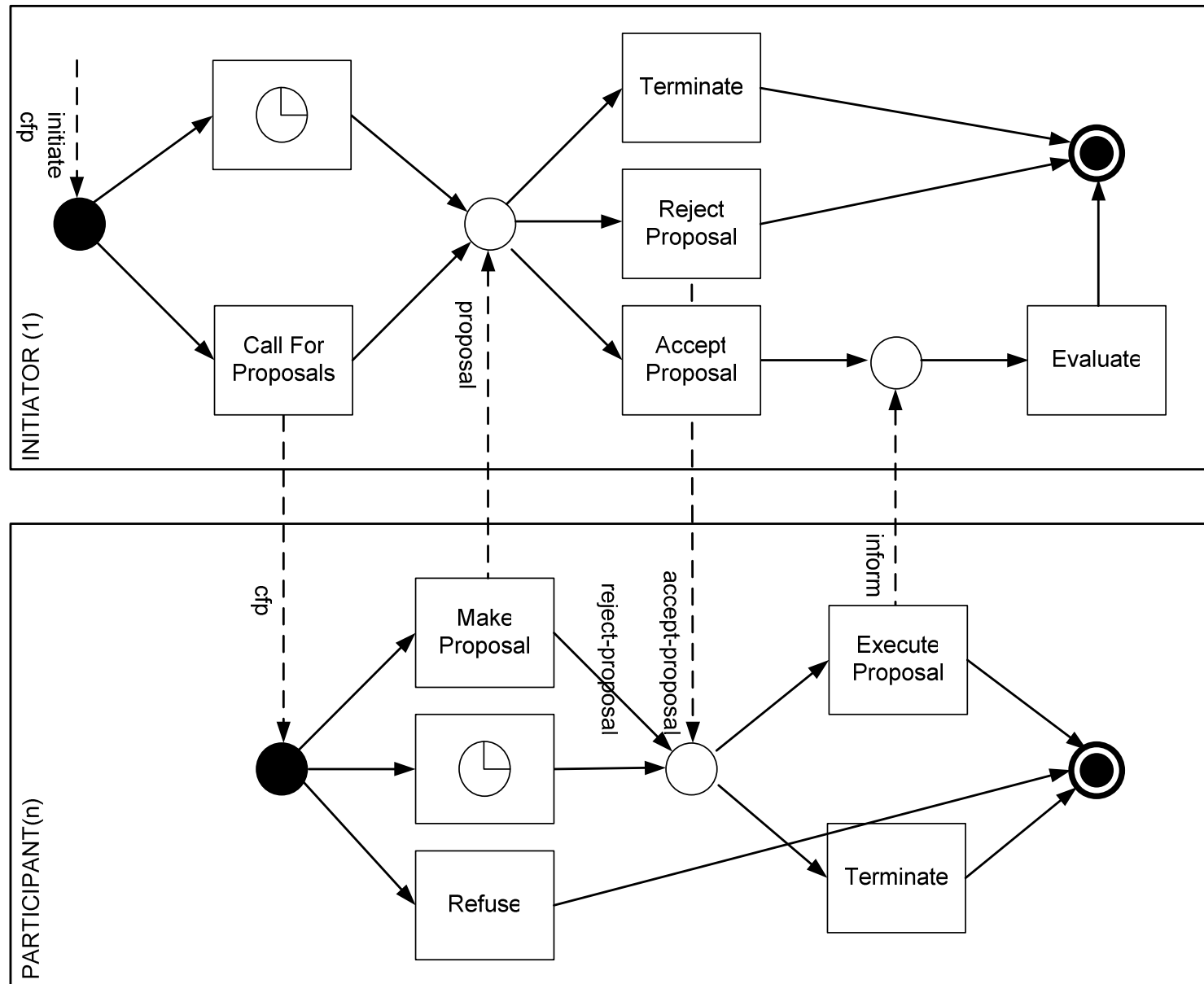
An example

The Contract Net Interaction Protocol

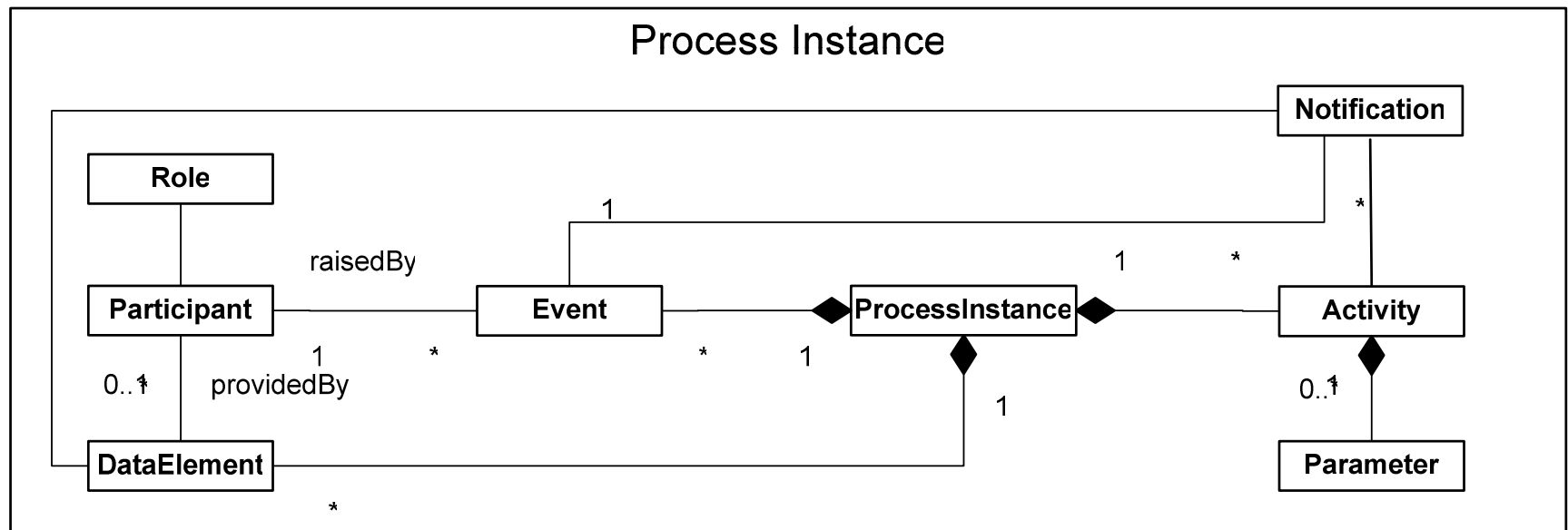
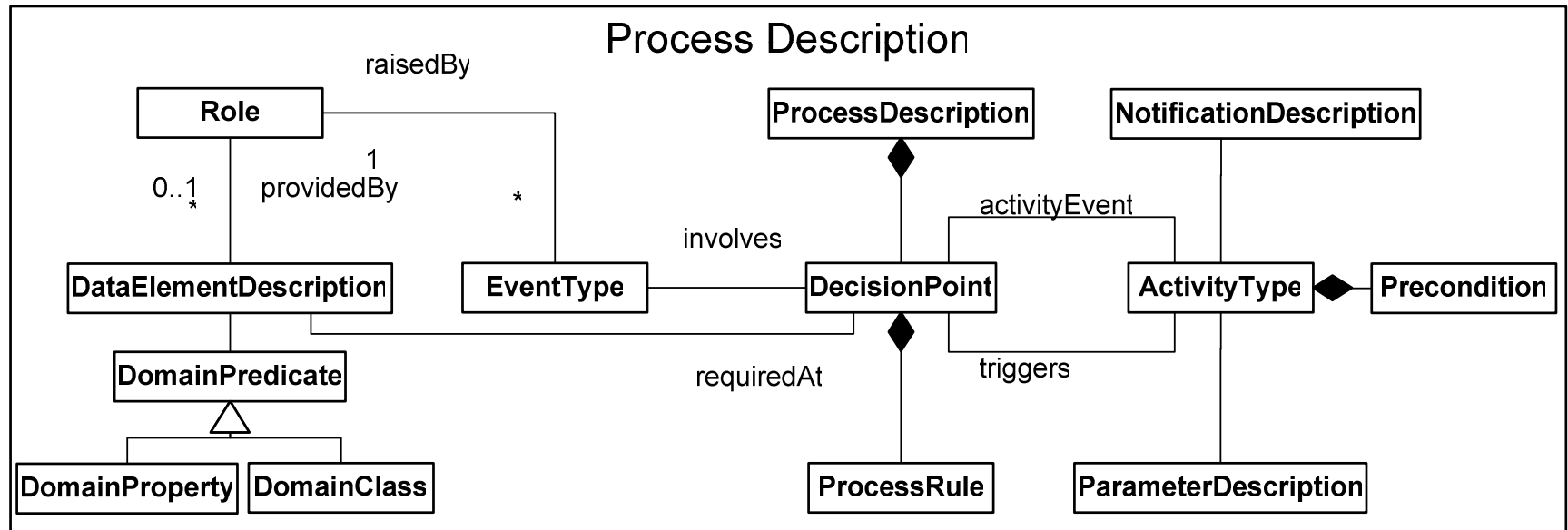
- non-trivial: one-to-many business process
- realistic: a pattern for real-life B2B interactions
- a well-understood negotiation protocol [Smith]

*“An initiator puts out a contract to tender via a CFP
participants can make a proposal
the initiator accepts / rejects proposals
the participant performs the proposal”*

A business process visualization



RBPM metamodel



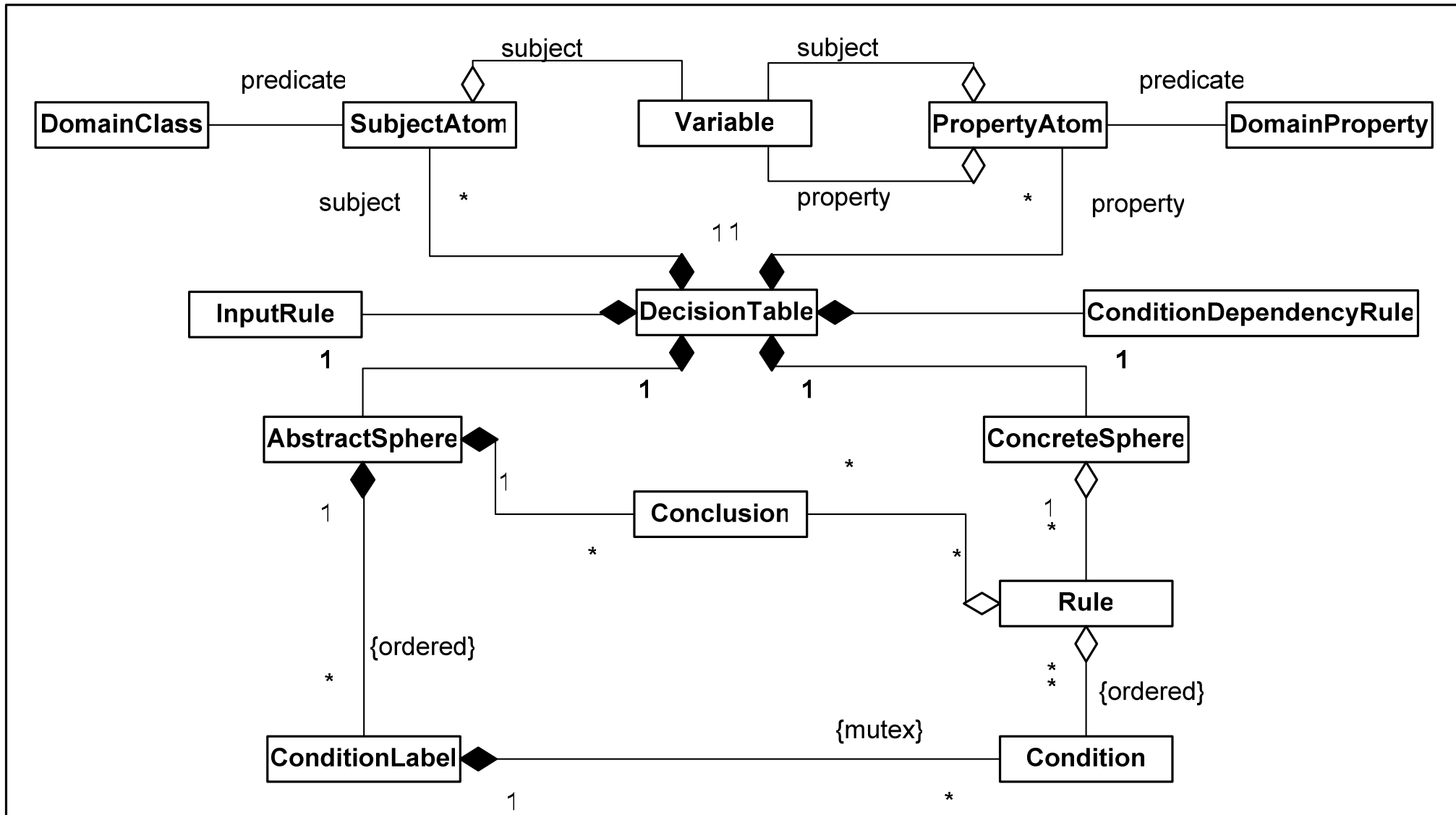
Why rule sets?

- Rule sets rather than rules are the atomic unit of logic
 1. Grouping of rules (modularization)
 - e.g. derivation rules that define one property
 - e.g. process rules pertaining to the same decision point
 2. Rule dependencies (e.g. default logics)
 3. Different numbers of rules can have equivalent semantics
- Rule set metamodel, inspired by Decision Tables
- A transformation from input rules to table rules

Decision Table with Process Rules

reaction(Event, Activity)							
1. etTimerActivity	Y						N
2. etCFPsucceeded	Y				N		-
3. etCFPfailed	-				Y	N	-
4. proposal	Y			N	-	-	-
5. bestProposal	Y		N	-	-	-	-
6. price, reservationPrice	Pr ≤ RP	Pr > RP	-	-	-	-	-
1. atAcceptProposal, P	x	.	.	-	-	-	-
2. atRejectProposal, P	.	x	x	-	-	-	-
3. atTerminate	.	.	.	x	x	-	-
	1	2	3	4	5	6	7

Rule set metamodel



II. How to link rule enforcement and application execution?

But... how to link rules and applications?

Independent development requires strong de-coupling.

	programming paradigm	abstraction level	approach
[D'Hondt]	OO <> LP, bi-directional	application-level	rule enforcement as AOP crosscutting concern, user-defined join points
[Dietrich]	OO > LP, unidirectional	process-level	rule-based agents, 1 fixed interaction point
RBPM	OO > LP, unidirectional	process-level	rule-based process execution, 5 fixed interaction points

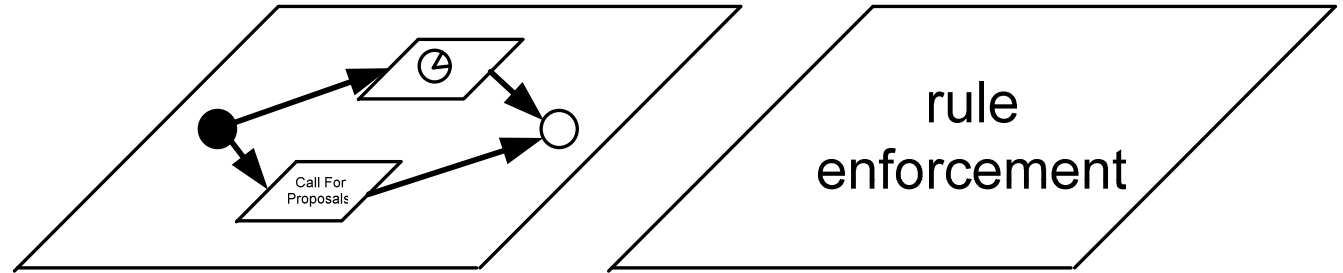
But... how to link rules and applications?

Commercial solutions: tight coupling between processes/applications and rules:

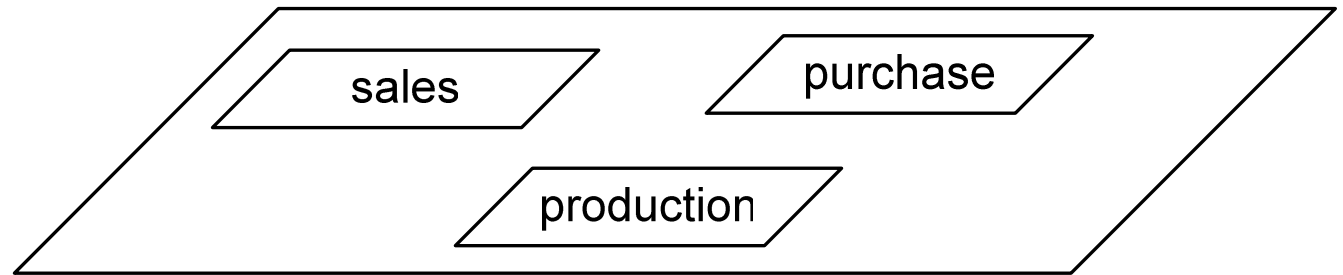
- **MS BizTalk:** decision shape represents call to rule engine
- **IBM WebSphere:** Business Rule Beans are triggered from EJBs
- **ILOG JRules:** bi-directional calling facilities, integrates with several process engines

Process-level BR enforcement: architecture

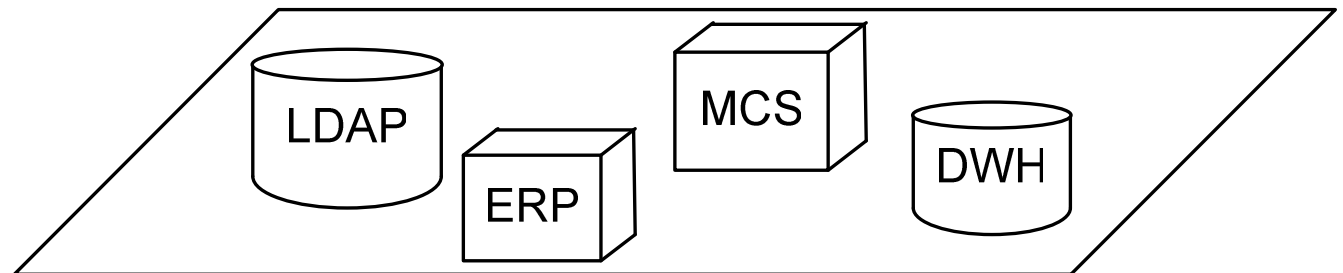
business
process and
rule layer



service and
component
layer



application
layer

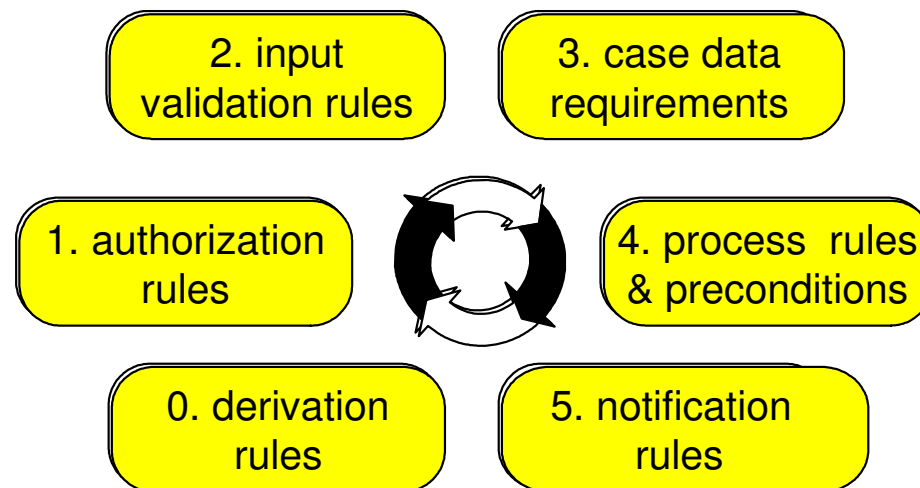


Rule-based process execution: messages

	<i>content</i>	sent to
EVENT MESSAGES	<i>Event and DataElements</i>	process participants (ProcessAgents)
ACTIVITY MESSAGES	<i>Activity and Parameters</i>	service representatives (ActivityAgents)
QUERY/DATA MESSAGES	<i>Query or DataElements</i>	process participants / service or application layer

Event processing cycle

“on a ready-to-go event, the system will check the event against the rules, and if it is valid, it will perform a MakeProposal activity”



these execution-level rules are generated or interpreted from the Enterprise Model

Implementing the RBPM framework

- processOntology, ruleSetOntology metamodels
- Rule-based Vocabulary Model and Process Model
- schema language: RDFS
- SWRL syntax
- *Reification: Parameter and DataElement* refer to `rdf:Statement` triples
- A rule generator generating execution-level rules, or meta-interpreted rules
- Inspired by [Dietrich]

Event processing cycle

Prolog queries derived/generated from the Enterprise Model:

- **Class and property predicate derivations**
- **authorized(+Event, +Participant)**
- **constraintViolation(-DataElement, -Constraint)**
- **requirement(+Decisionpoint, -DataElementType, -Participant)**
- **reaction(+Event, -Activity)**
- **notification(+ActivityEvent, -Notification)**

Agent-based process engine

DispatcherAgent for bootstrapping new processes

ProcessAgent manages process instances
(implements the event processing cycle)

ActivityAgent manages service invocations

CNP Initiator: a screenshot

The screenshot displays the CNP Initiator interface with the following components:

- Incoming Messages (IN):**
 - etInitiateCFP - elInitiateCFP1
 - etCFPsucceeded - activityEvent2
 - etProposal - event2
 - timerActivitySucceededEventType - activityEvent4
 - etAcceptProposalSucceeded - activityEvent6
- Outgoing Messages (OUT):**
 - atCallForProposals - activity1
 - timerActivity - activity2
 - etCallForProposals - notification1
 - atAcceptProposal - activity3
 - etAcceptProposal - notification2
- Prolog Queries:**

```
notificationMessage('http://econ.kuleuven.be/cnpProcessDescription_initiator1#notification1', 'f:/stijngoedertier/ideas/vorte/java/jad
notificationEvent('http://econ.kuleuven.be/cnpProcessDescription_initiator1#notification1', Event)
rdf('http://econ.kuleuven.be/cnpProcessDescription_initiator1#event1', 'http://econ.kuleuven.be/processOntology#eventType', EventT
file_unload('f:/stijngoedertier/ideas/vorte/java/jade/msg/beb2d2ad-5beb-e31a-580e-758245ce4e5c.txt)
Notification message created: 'http://econ.kuleuven.be/cnpProcessDescription_initiator1#notification1'
message: f:/stijngoedertier/ideas/vorte/java/jade/msg/beb2d2ad-5beb-e31a-580e-758245ce4e5c.txt
event: 'http://econ.kuleuven.be/cnpProcessDescription_initiator1#event1'
event type: 'http://econ.kuleuven.be/cnpProcessDescription#etCallForProposals'
```
- Control Buttons:** Kill Agent, Dump Triple Base

CNP Participant: a screenshot

Incoming Messages

IN

- etCallForProposals - event1
- etMakeProposalSucceeded - activityEvent2

Outgoing Messages

OUT

- atMakeProposal - activity1
- etProposal - notification1

Prolog Queries:

```
incorporateDataElements('f:/stijngoedertier/ideas/vorte/java/jade/msg/96490b45-e910-6552-50a6-1140d0d5e5a4.txt', Event)
rdf('http://econ.kuleuven.be/cnpProcessDescription_initiator1#event1', 'http://econ.kuleuven.be/processOntology#event1', Event)
reactions('http://econ.kuleuven.be/cnpProcessDescription_initiator1#event1', Activities)
rdf('http://econ.kuleuven.be/cnpProcessDescription_participant1#activity1', 'http://econ.kuleuven.be/processOntology#activity1', Activity)
activityMessage('http://econ.kuleuven.be/cnpProcessDescription_participant1#activity1', 'f:/stijngoedertier/ideas/vorte/java/jade/msg/96490b45-e910-6552-50a6-1140d0d5e5a4.txt', Activity)
```

Kill Agent **Dump Triple Base**

References

- [Dietrich] J. Dietrich, A. Kozlenkov, M. Schroeder, and G. Wagner, “Rule-based agents for the semantic web.” *Electronic Commerce Research and Applications*, vol. 2, no. 4, pp. 323–338, 2003.
- [D’Hondt] M. D’Hondt and V. Jonckers, “Hybrid aspects for weaving object-oriented functionality and rule-based knowledge,” in *AOSD ’04: Proceedings of the 3rd international conference on Aspect-oriented software development*. New York, NY, USA: ACM Press, 2004, pp. 132–140.
- [Halpin] T. A. Halpin, “A Fact-Oriented Approach to Business Rules,” in *ER*, 2000, pp. 582–583.
- [Ross] R. G. Ross. *Principles of the Business Rule Approach*. Addison-Wesley, 2003.
- [Smith] R. G. Smith, “The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver.” *IEEE Trans. Computers*, vol. 29, no. 12, pp. 1104–1113, 1980.
- [Wagner] K. Taveter and G. Wagner, “Agent-Oriented Enterprise Modeling Based on Business Rules.” in *ER*, ser. Lecture Notes in Computer Science, H. S. Kunii, S. Jajodia, and A. Sølvberg, Eds., vol. 2224. Springer, 2001, pp. 527–540.

Questions