A Contract Layered Architecture for Regulating Cross-Organisational Business Processes

Mohsen Rouached (*mohsen.rouached@loria.fr*)

Olivier Perrin (*olivier.perrin@loria.fr*)

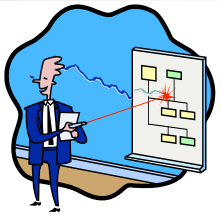
Claude Godart (*claude.godart@loria.fr*)

LORIA-INRIA-UMR 7503

Outline

- Business Process Management (BPM)
- Objectives
- BPM: Workflows vs contracts
- Our Approach
  - A Layered Architecture
  - Event-based Model
  - Event Calculus
  - Commitments
  - A Formalisation for Contract Clauses
- A summary of the approach
- Conclusion & Future Work

Introduction



- Cross organisational business processes
- Distributed and unpredictible business environments
- Business process modeling and analysis
- Formal models in business process management
- Entreprise Informations systems and business monitoring activities

Objectives

- Improve the efficiency of cross-organizational cooperative processes
  - Distributed architecture
  - Autonomy and privacy of partners without central control
  - Externalisation of business processes
- Create flexible processes that are adapted to cooperative environments and unpredictible changes
  - Dynamic adaptation
  - Temporal specification
  - Pro-active monitoring

BPM: Workflows vs Contracts

- Workflows
  - A process based approach
  - Concentrate on **how** to manage processes
  - Modeling of dependencies between business processes
  - Centralised architecture (often)
  - Changes management implies process model changes

- Contracts
  - An event based approach
  - Concentrate on **what** to manage in processes
  - Responsiblities of partners (rules, rights and obligations)
  - Distributed architecture
  - Reconfiguration and easy adaptation

Our Approach

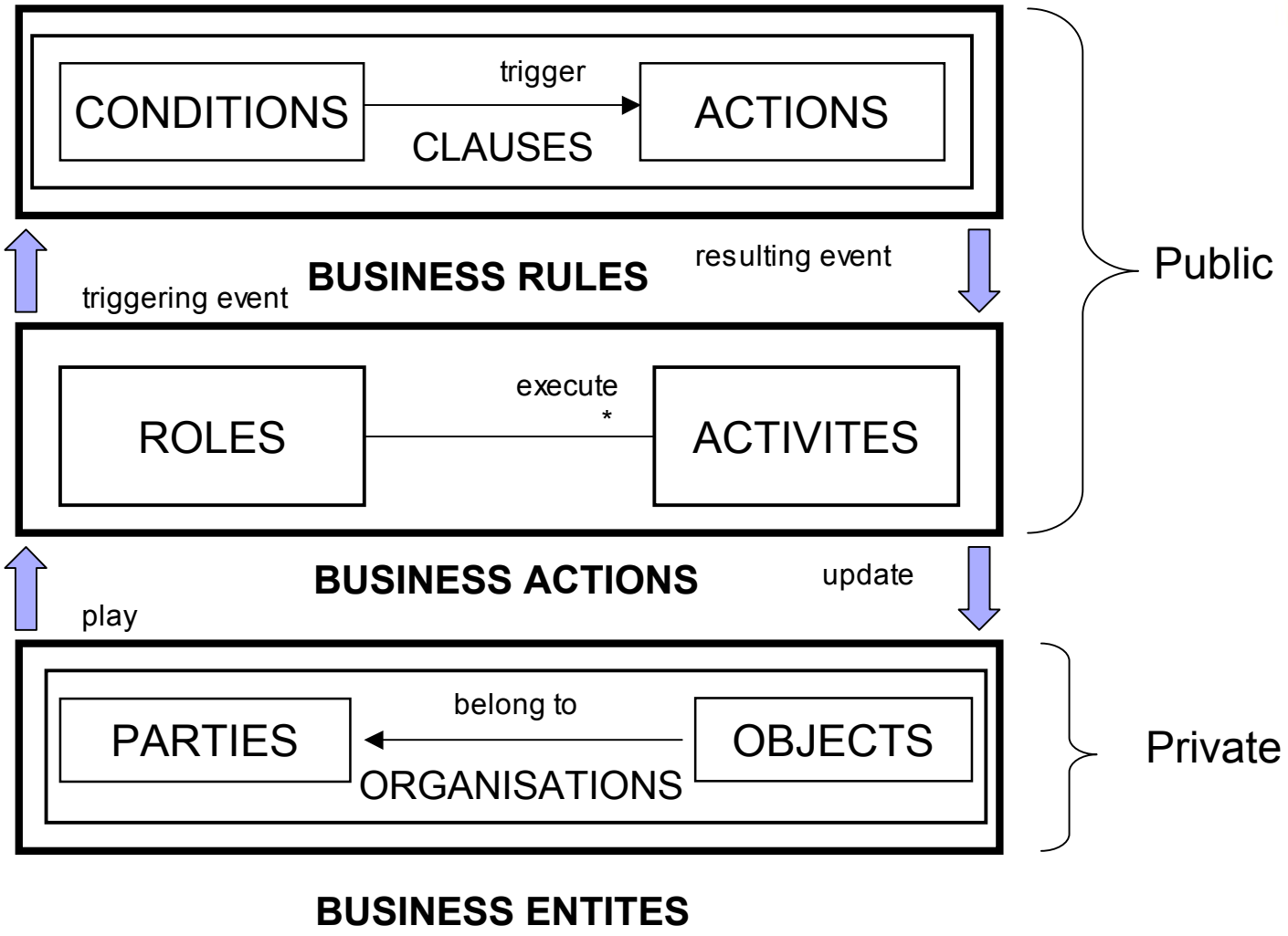
- Identify business partners and objects necessary for contract execution,
- Extract actions mentioned in the contract document,
- Identify role(s) to be played by each participant,
- Determine activities to be executed by each role and their consequences in terms of generated events,
- Provide a formal specification for contract clauses, which permits to ensure temporal abstractions to enforce non-conforming actions at the **pro-active monitoring** stage.

An e-contract model for BPM

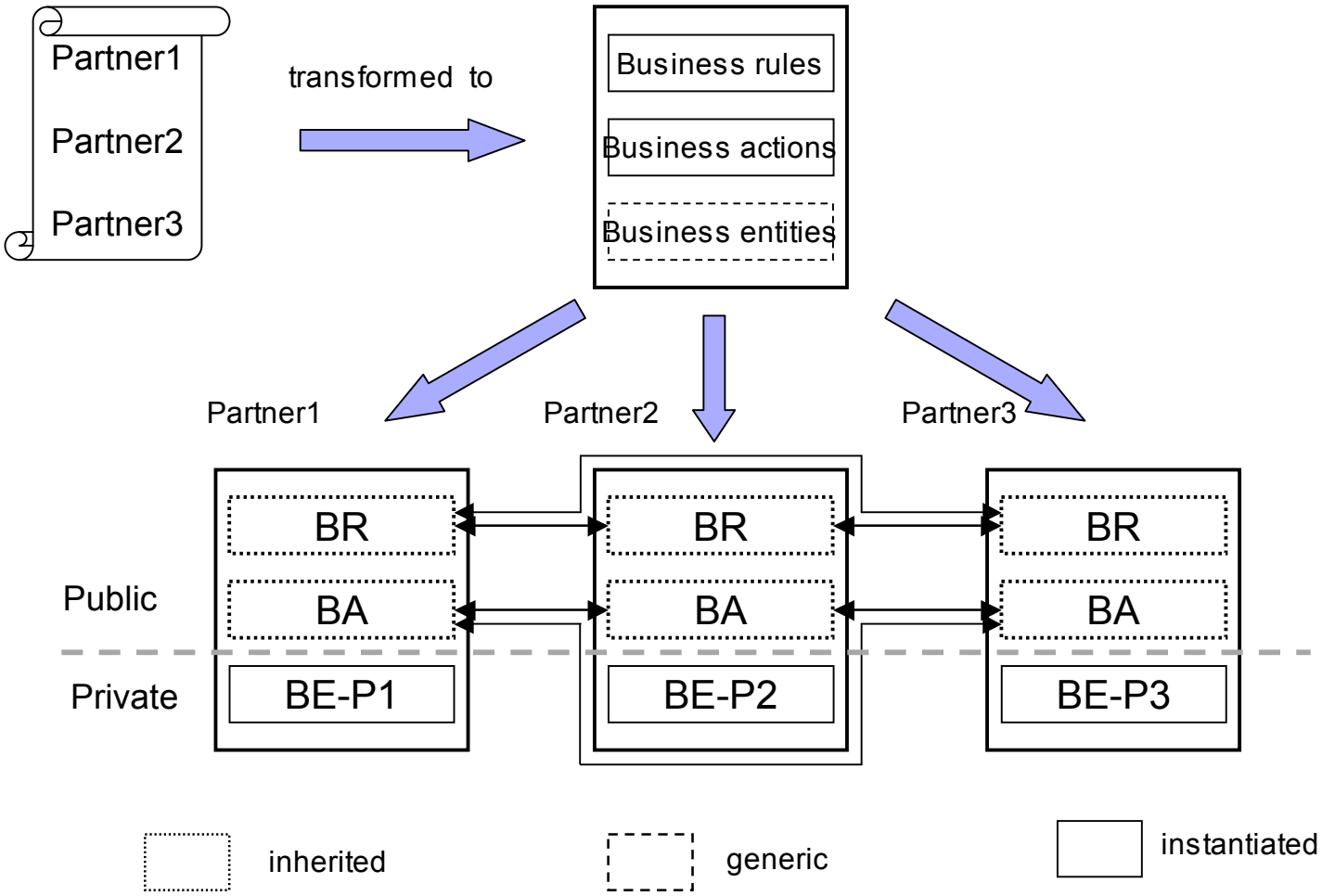
- Proposals
  - A three layers architecture for e-contract modeling
  - An event-based mechanism expressed in the Event Calculus
  - A Formalism for reasoning about e-contracts using Commitments expressed in the Event Calculus

- Goals
  - Abstraction
  - Dynamic adaptation
  - Distribution
  - A well defined semantics to reason about e-contracts, which helps for achiving the pro-active monitoring goals (Event Calculus+Commitments).

Three Layer Architecture for e-contract Enforcement



Distributed Architecture for e-contracts



An event-based model

- An event is a significant occurrence in time or instantaneous (punctual).

- Events are relevant to roles within a context determined by the contract. This context has attributes such as:
  - repetition operators (always, only)
  - detection mode (reported, immediate)
  - composition operators (and, or, all, sequence)
  - counting operators (cardinality, at least, at the maximum)
  - negation operators (except, unless)
  - temporal management (after, before, at the same time, every)
- Relationships between events:
  - Causality
  - Concurrence
  - Composition (conjunction, disjunction, sequence)

Event Calculus

A first-order formal language for specifying properties of dynamic systems which change over time using predefined predicates including:

- **Happens(e, t, ℜ(t1,t2))**−occurrence of an event ***e*** of instantaneous duration at some time ***t*** within the time range **ℜ(t1,t2)**
- **HoldsAt(f,t)**−fluent ***f*** holds at time **t**.
- **Initiates(e,f,t)**−fluent ***f*** starts to hold after the event ***e*** at time ***t***.
- **Terminates(e,f,t)**−fluent ***f*** ceases to hold after the event ***e*** occurs at time ***t***

Commitments [Singh99]

- Definition

C(x,y,a,t,sx,sy)

x: debtor, y: creditor, a: action, sx et sy: sanctions

- Operations
  - Create(e,p1,C)

$$Initiates(e,C(p1,p2,p),t) \leftarrow Happens(e,t) \wedge Create(e,p1,C(p1,p2,p))$$

  - Release(e,p2,C)

$$Terminates(e,C(p1,p2,p),t) \leftarrow Happens(e,t) \wedge Release(e,p2,C(p1,p2,p))$$

  - Cancel(e,p1,C)

$$Terminates(e,C(p1,p2,p),t) \leftarrow Happens(e,t) \wedge Cancel(e,p1,C(p1,p2,p))$$

A Formalisation for Contract Clauses

Using commitments expressed in the Event calculus, contract clauses are expressed as follows:

- Prohibitions :

  Create(F(p1,p2,p),p2,C(p2,p1,A(¬ p)))

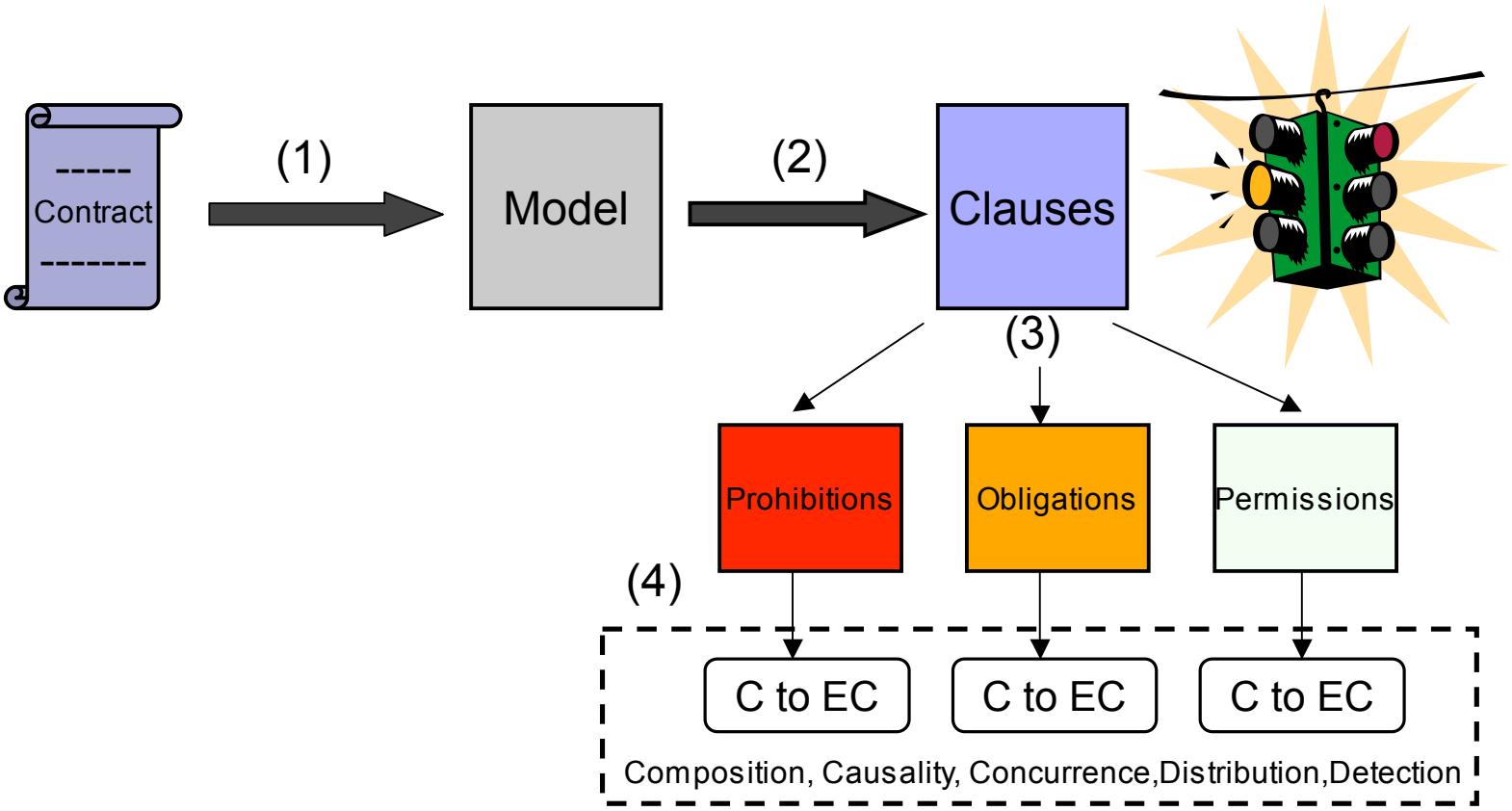
- Permissions :

  Release(P(p1,p2,p),p1,C(p2,p1,A(¬ p)))

- Obligations :

  Create(O(p1,p2,p),p2,C(p2,p1,A(p)))

A summary of the approach



Conclusion & Future Work

- A three layer architecture to model multi-party contracts
  - Autonomy
  - Privacy
  - Distributed nature
- A Formalisation of the monitorable contract model (event+Event calculus+Commitments)
  - Externalisation
  - Temporal abstractions
  - Pro-activity
- In our future work, we plan to:
  - work on further details for complex events management,
  - integrate the solution in existing web services architectures such as ebXML or RosettaNet,
  - study correctness requirements in business process protocols.