Distributed Orchestration v.s. Choreography:

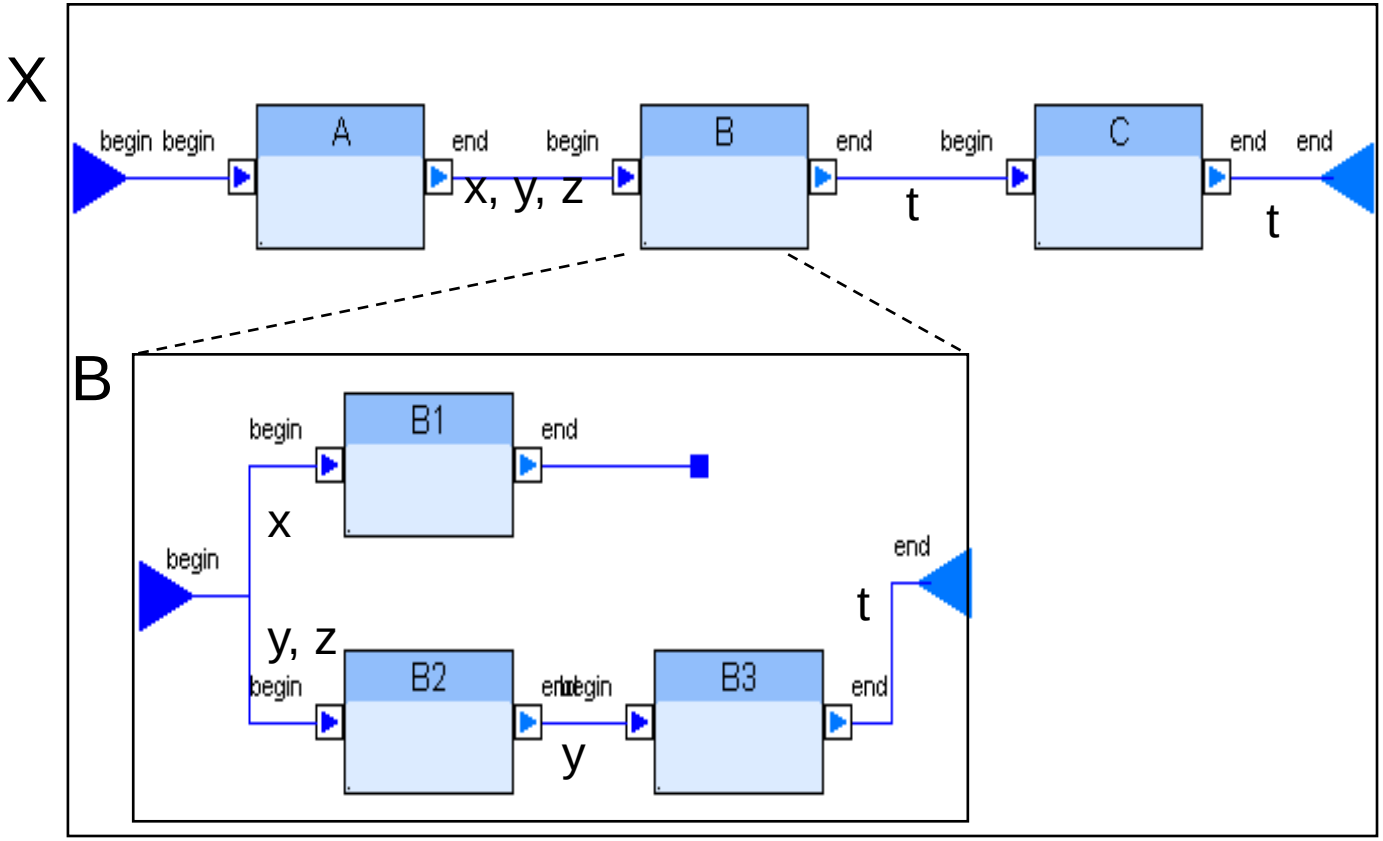
The FOCAS Approach

Gabriel Pedraza, Jacky Estublier
Grenoble University. France

ICSP May 2009

Equipe Adèle

Abstract Process Engine Language (APEL)



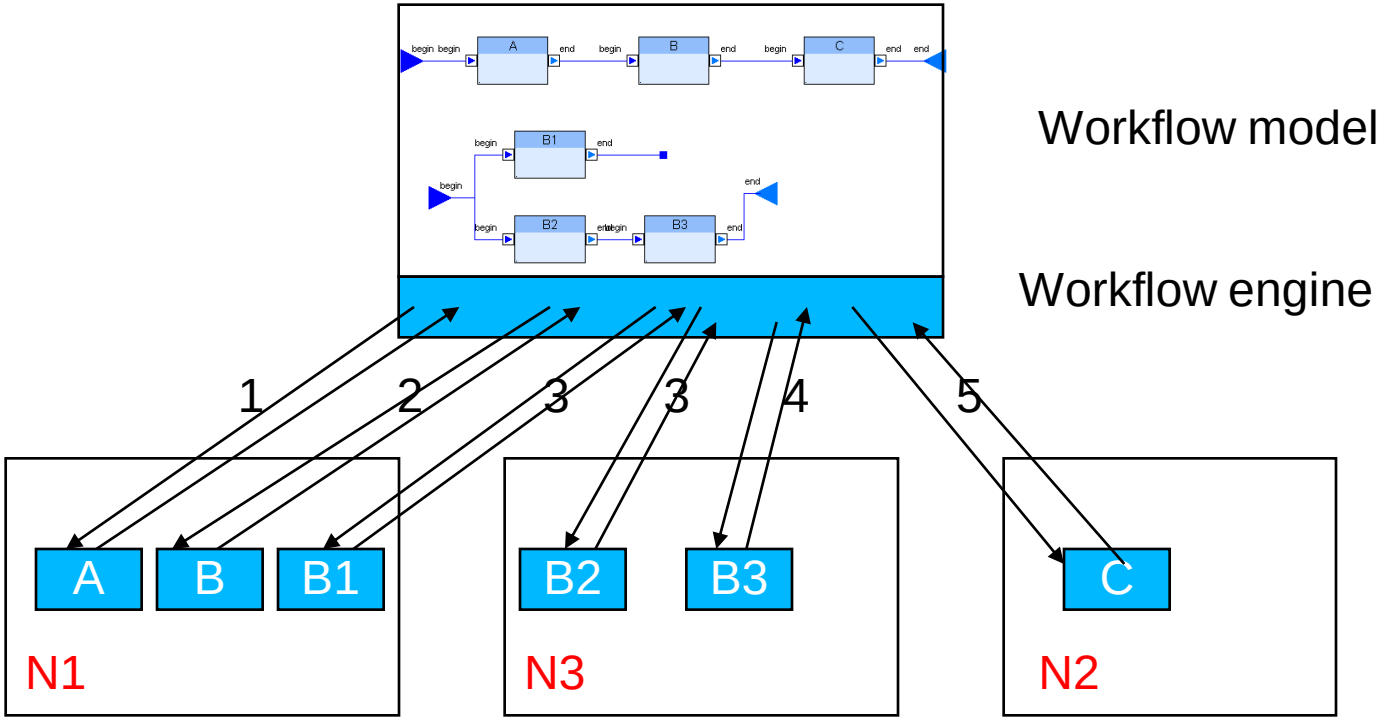
Abstract Process model

Abstract models

- Why using abstract models :
  - Describe the « business » process ,
  - Easy to read and understand,
  - Hide « irrelevant » details,
  - Independent on any implementation,
  - Can be instantiated differently in different contexts,
  - Can be reused ….

- But cannot execute ! (useless ?)

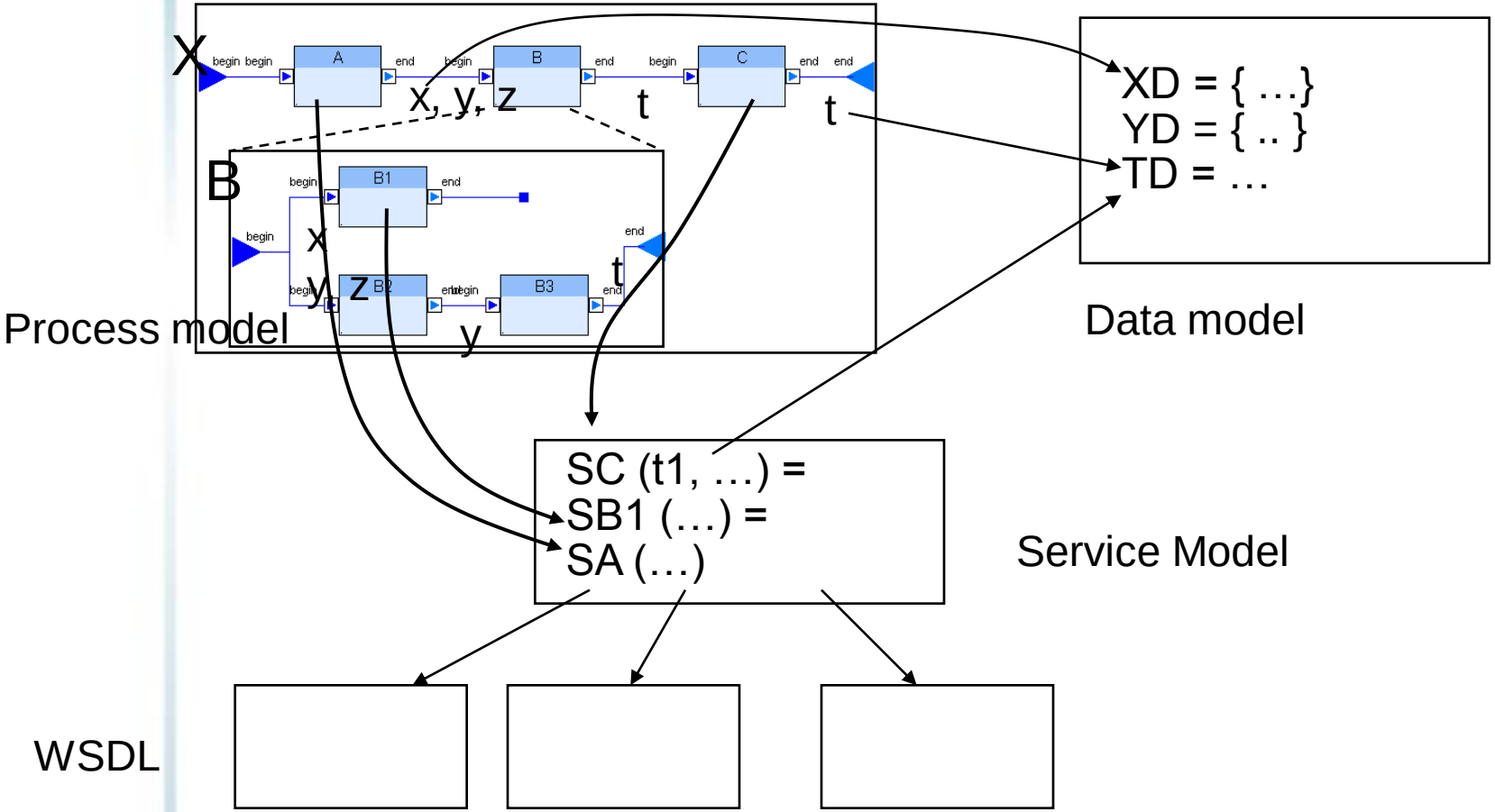
Orchestration: a process for calling WS



Requires to know at least:
- Activity <-> WS.
- Data <-> parameter

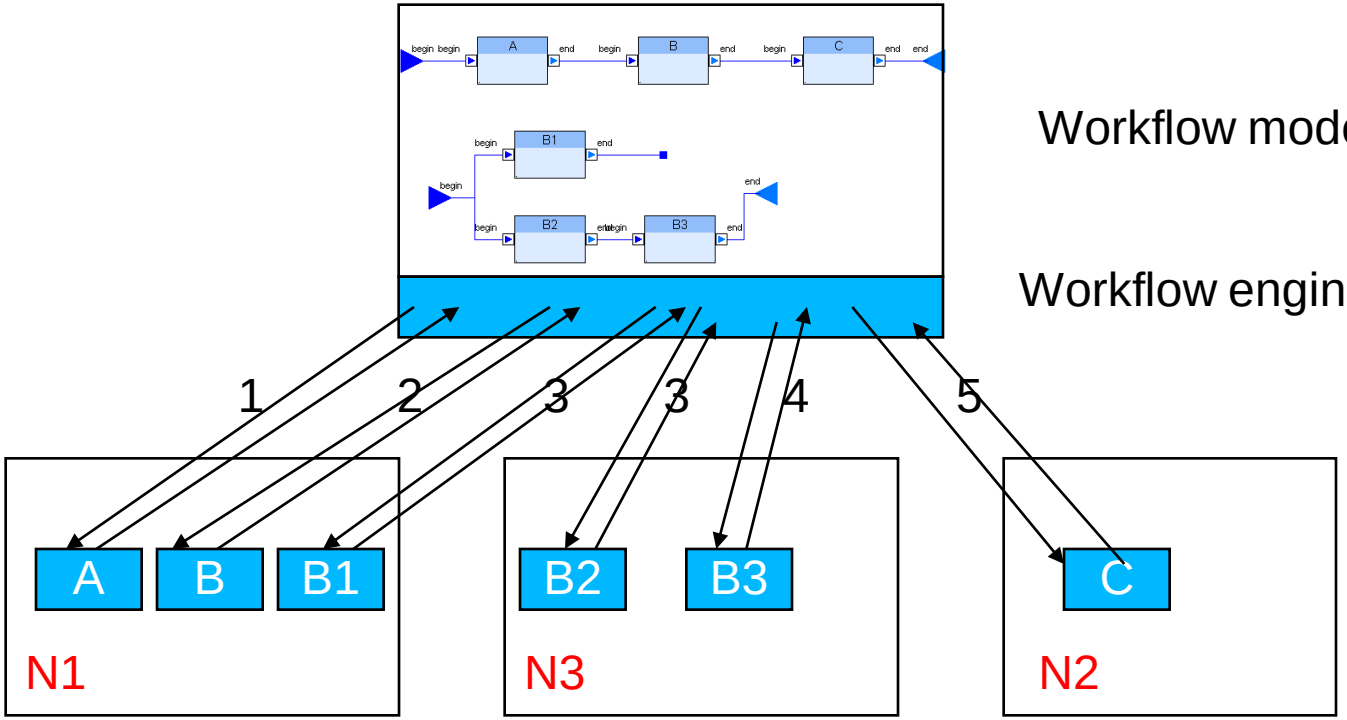
And to know at least:
- Activity + data <-> method call,
- Formats,
- Protocols.

Model composition: Abstract process execution



Executes … but still an academic exercise

Orchestration: a workflow for calling WS



Requires to know at least:
- Activity <-> WS.
- Data <-> parameter

And to know at least:
- Activity + data <-> method call,
- Formats,
- Protocols.

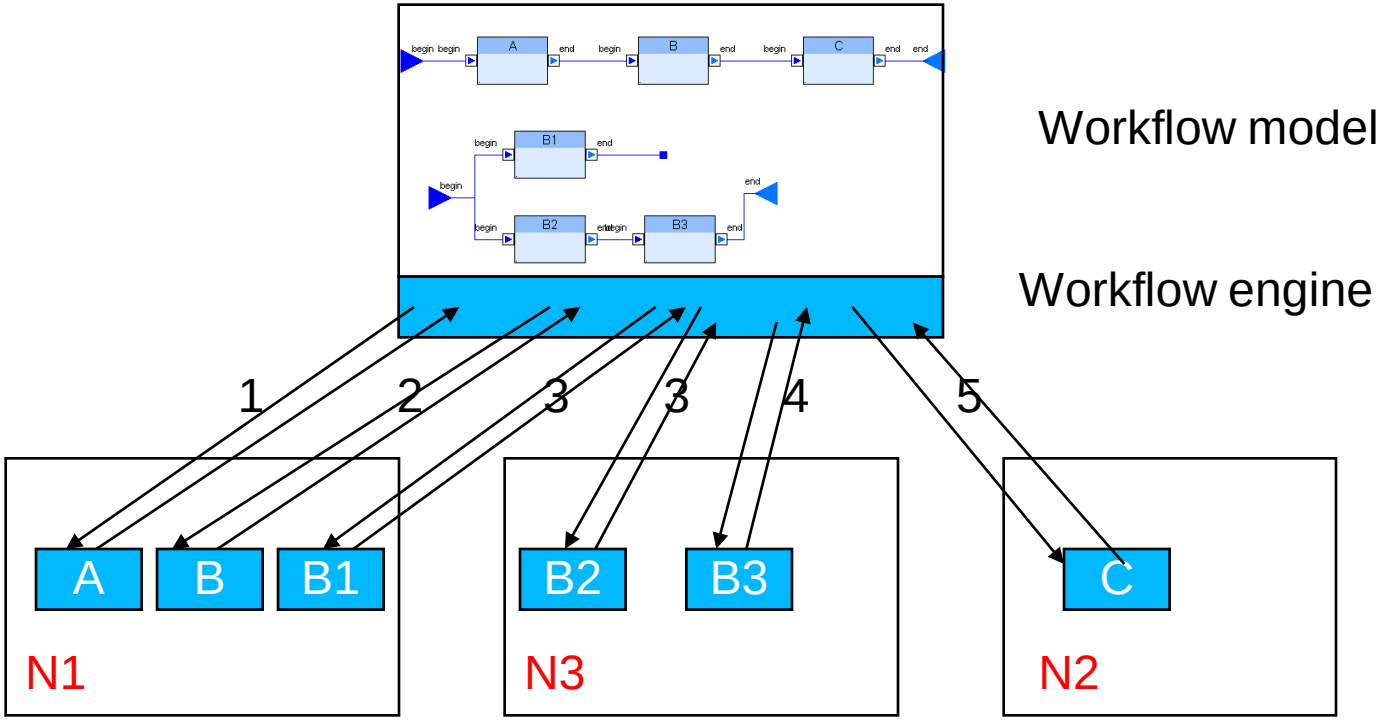
But also:
- distribution constraints.
- security properties.
- error recovery,
- performance issues,
- …..

Non Functional properties : Annotations

- Non functional properties can be expressed as annotations on the abstract model.

- Exemple of annotation available today :
  - Security,
  - Transaction,
  - Choreography (process distribution).

Orchestration



Pros : Centralized

Easy to understand

Simple to design, administrate, dynamic selection, error recovery

Cons : Centralized

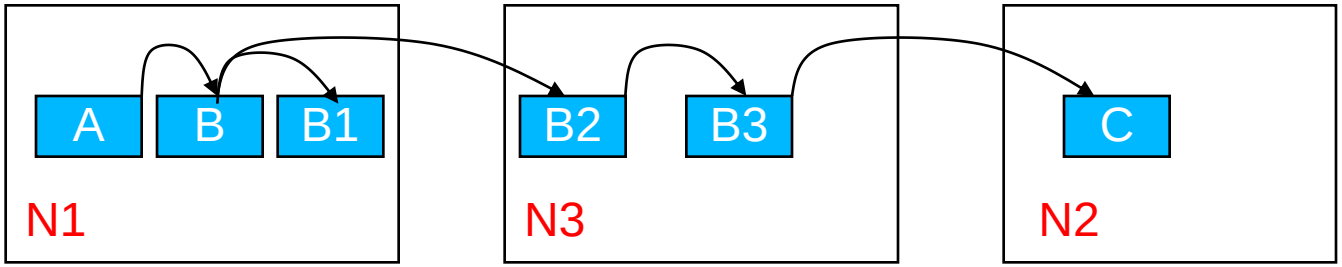
Bottleneck : scalability issues.

Choreography

Pros :

Scalable.

Availability, efficiency, Improved security, flexibility



Cons :

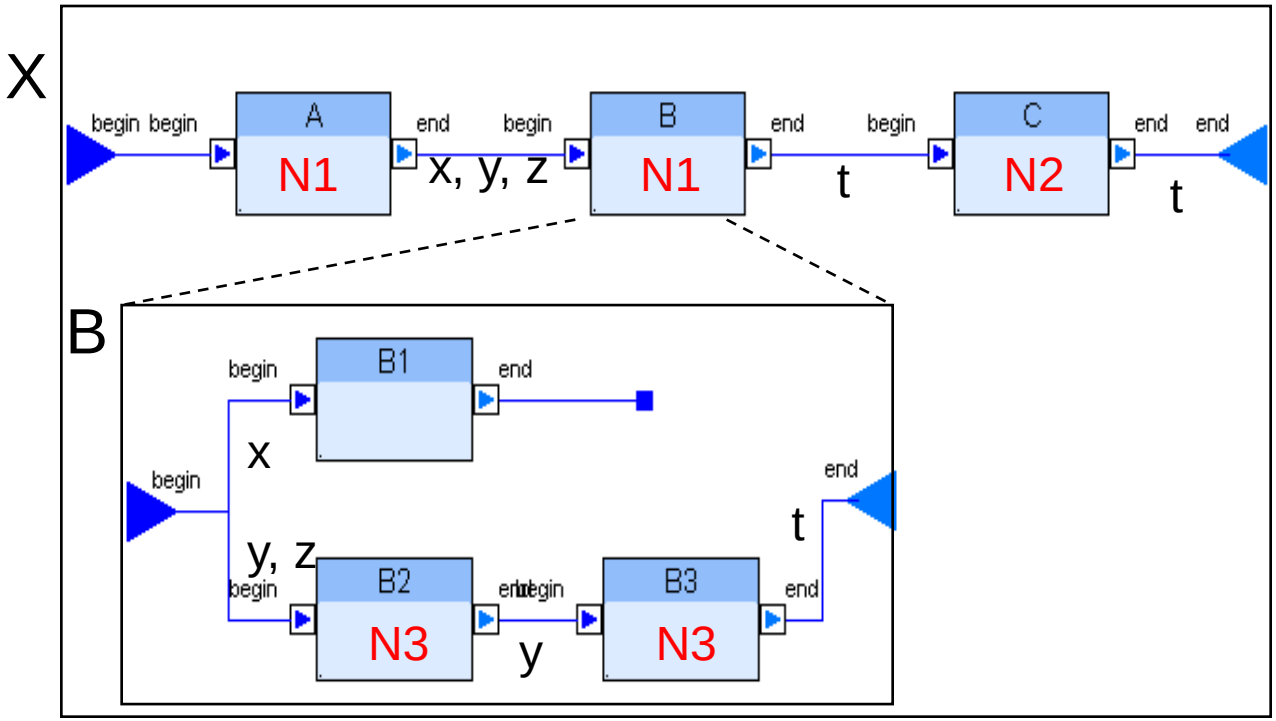
Difficult to design and understand

Difficult to implement, control, administrate, ….

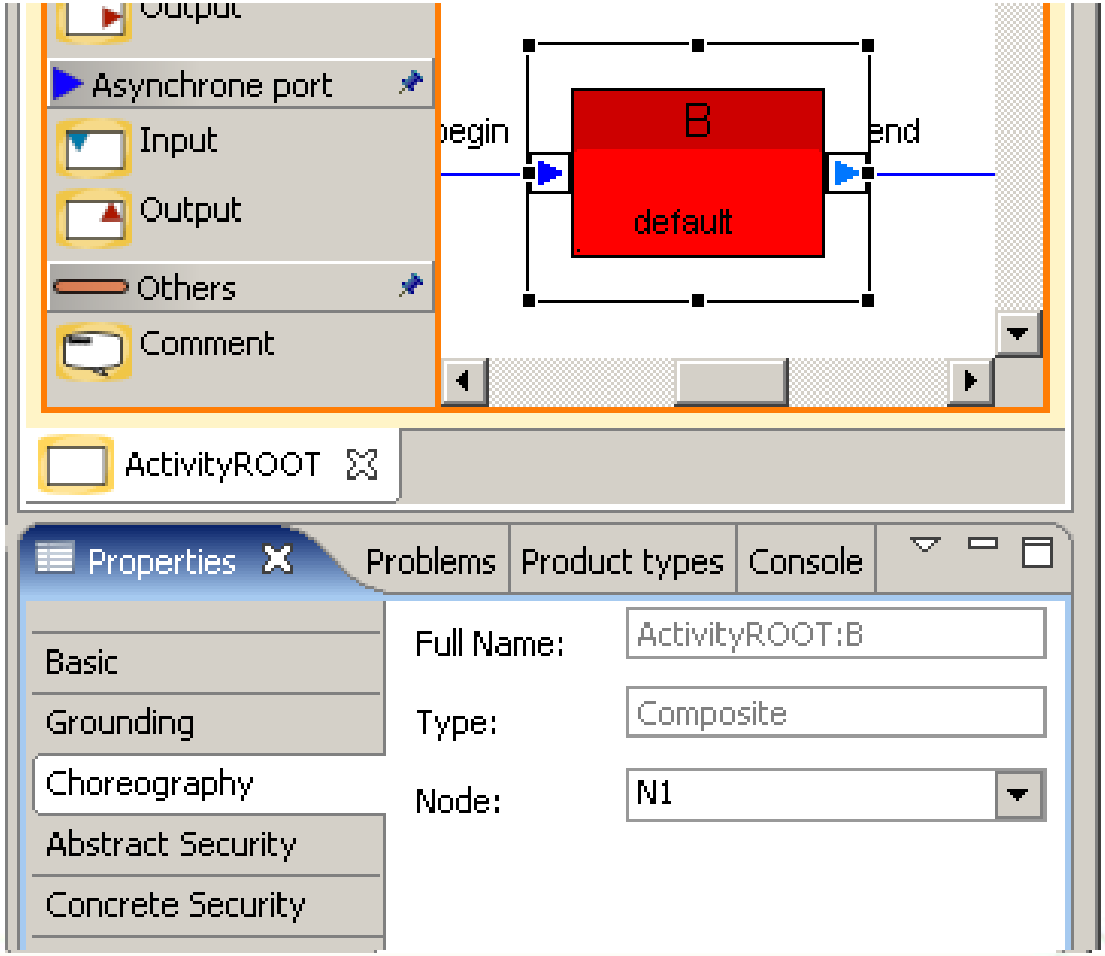
Requires to execute a « routing » algorithm on each server.

Distributed orchestration

- A distribution annotation on an orchestration model.

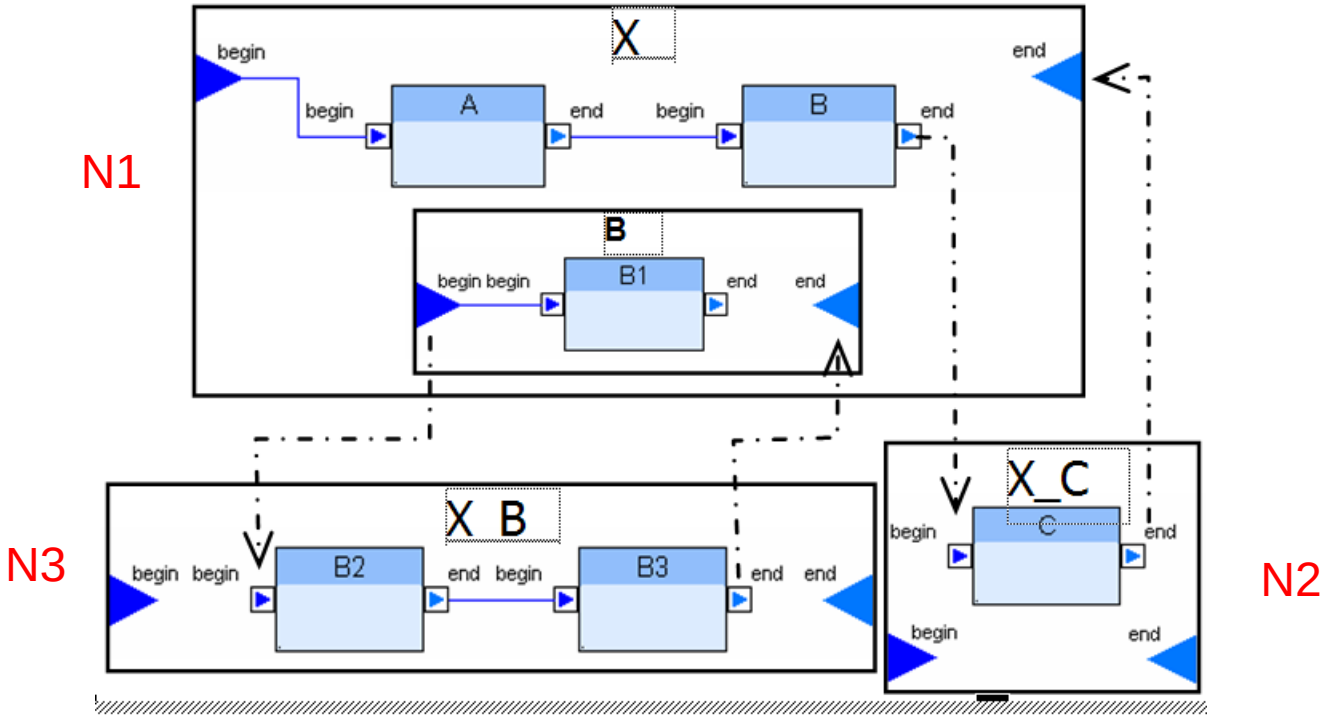


Distribution annotation : FOCAS/Eclipse

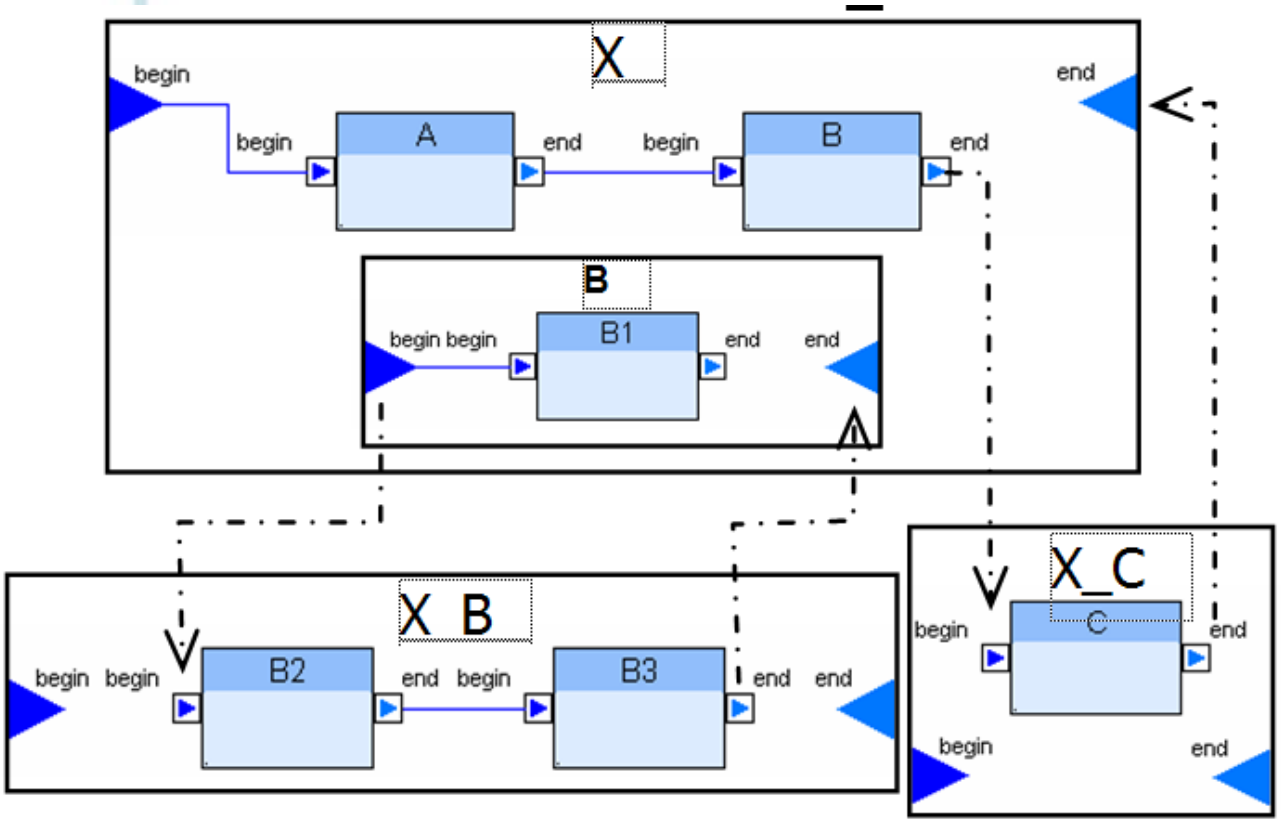


From Orchestration to Choreography

- Transform the central process : one sub-process per node



Abstract Choreography



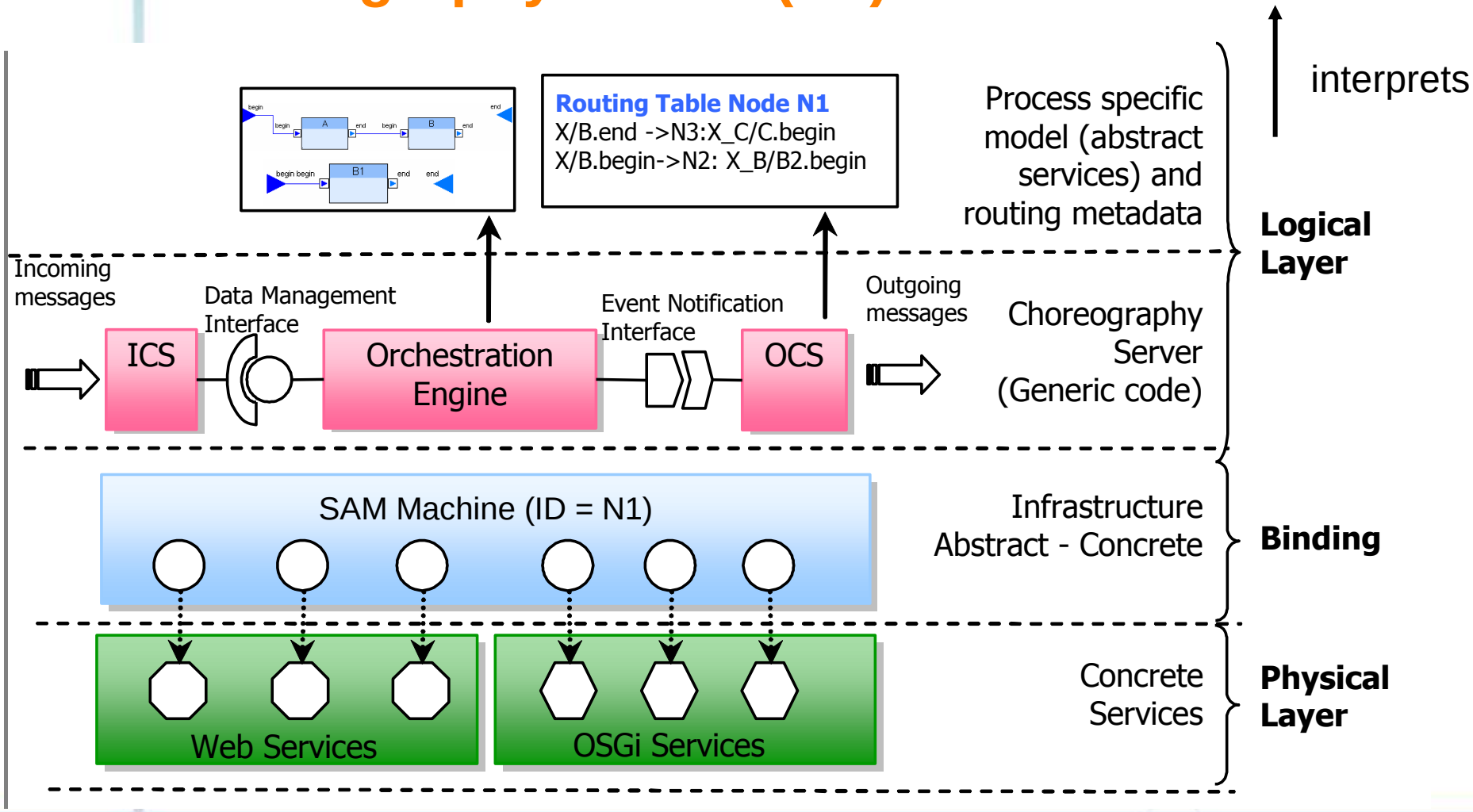
Routing Table Node N1
X/B.end ->X_C/C.begin
X/B.begin->X_B/B2.begin

Routing Table Node N2
X_B/B3.end -> X/B.end

Routing Table Node N3
X_C/C.end -> X.end

Deployment Plan
PUT X_C in N3
PUT X_B in N2
PUT X in N1 ..

Choreography Server (N1)



Deployment server : SAM

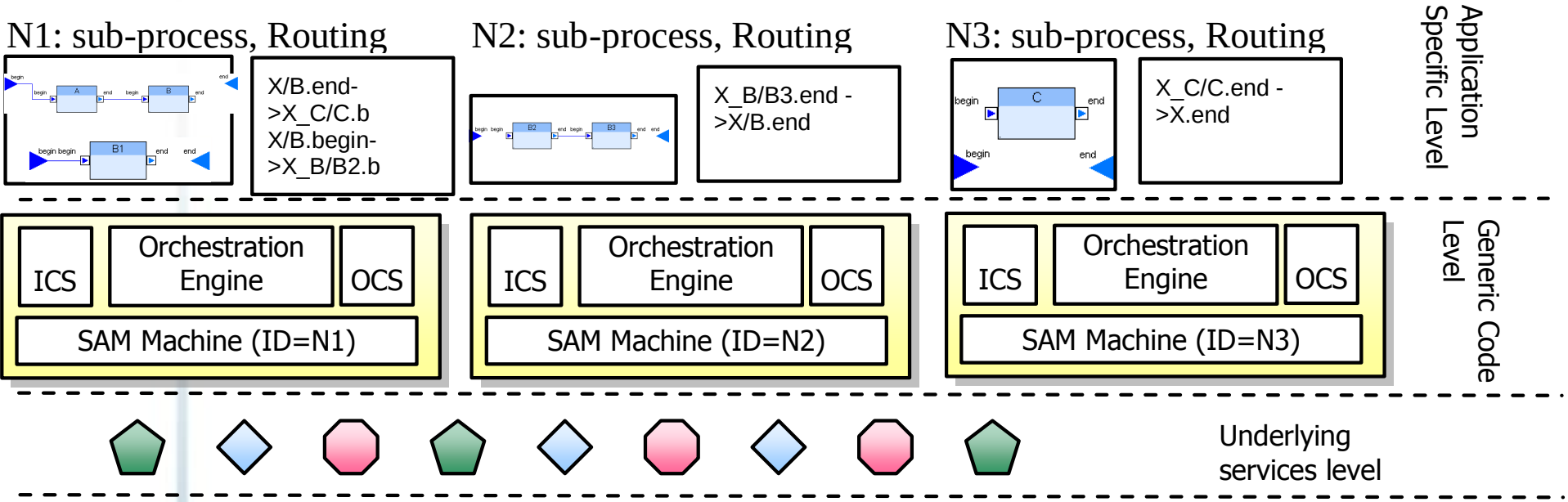
- Service Abstract Machine.
  - Subsumes current SOA platforms (currently: OSGi, iPOJO, AXIS, uPnP, DPWS, SNMP).
  - Natively distributed. SAMs are discovered dynamically.
    - N3:X_C/C.begin (t).
  - Routing and deployment tables can be changed dynamically
  - Choreography topology can be changed dynamically
  - Process model can be changed dynamically

Choreography server

- The workflow engine is unmodified: identical on each node.
  - Reuse existing workflow engines
- The Choreography server is made of two generic components
  - Output Choreography Server (OCS). Interprets routing tables.
  - Input Choreography Server (ICS). Starts activities.

- Each Choreography node is identical (process independent).
  - Can be installed once for all
  - Can run any process
  - Can run any number of process instance

Choreography



Conclusion: Orchestration v.s. Choreography

- Orchestration. good for design:
  - Easy to understand and communicate
  - Business model, error recovery, dynamic selection etc.
- Choreography. good for execution:
  - Efficient, Scalable, Adaptable to various contexts
  - Fully dynamic

- Annotations can bring the best of both camps:
  - Designing an orchestration (a centralized process),
  - Executing a choreography,
  - Without any change in the model,
  - Without any change in process engine, editor, tools …
  - A few new meta data (model)

Conclusion2: Enhancing process technology

- From Orchestration to choreography
  - A transformation that enforces the same process semantics
  - Does not change the process model
  - Does not change the PML environment (interpretors, editors, …)

- Annotation
  - Are also abstract
  - Distribution annotation
  - Provides large dynamic capabilities

- Can be applied to any process model and engine
- Any process can be executed in a distributed way whatever the formalism (if abstract).

- A practical way to apply separation of concerns to process technology.