Tutorial slot 2: Coordination Delegates Realization

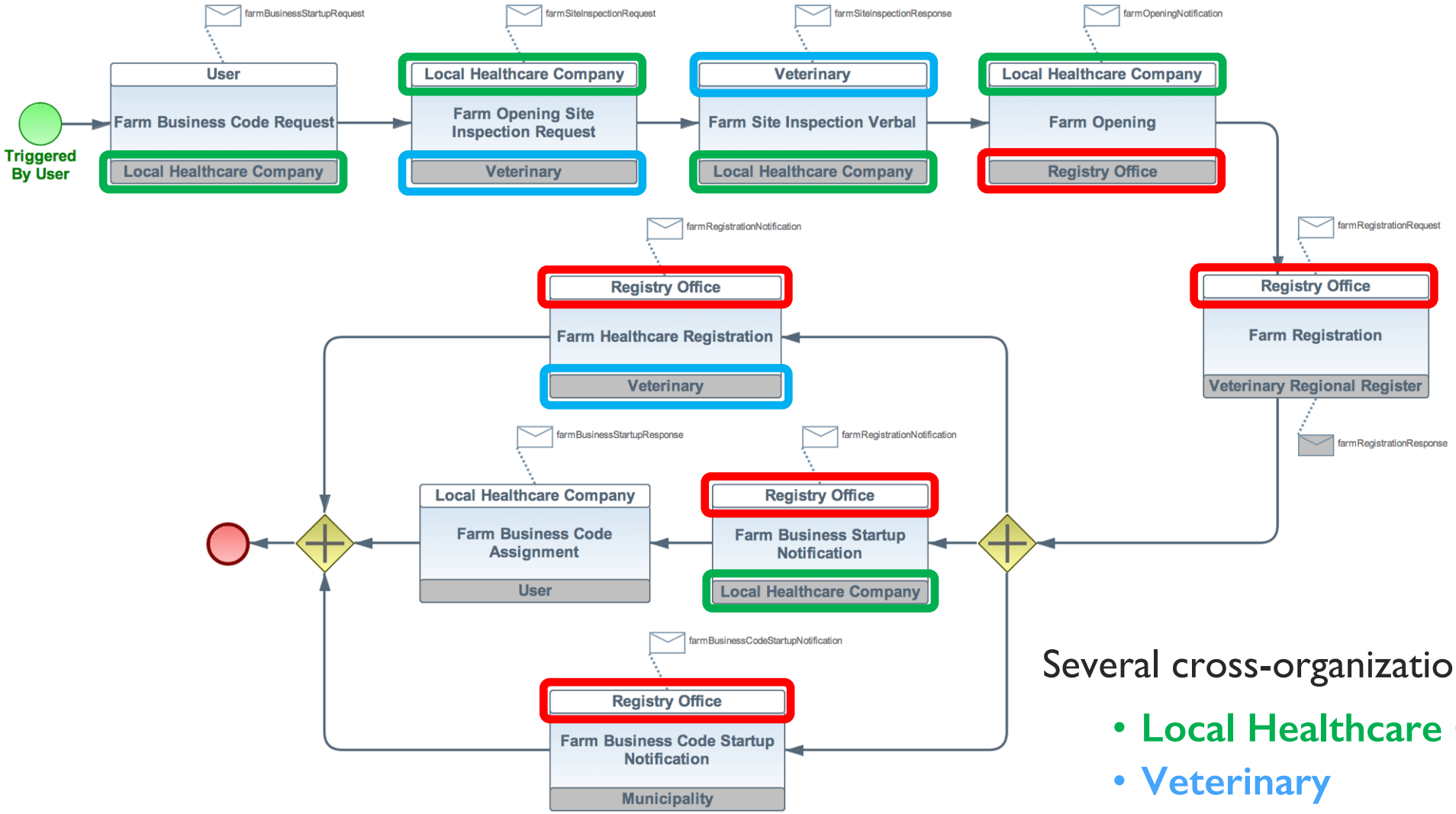
Massimo Tivoli
University of L'Aquila
massimo.tivoli@univaq.it

Outline

- **Running example**
- **Reference architectural style**
- **Coordination delegates realization**
  - Interaction patterns
  - Implementation
  - Coordination logic

Farm Business Startup Choreography

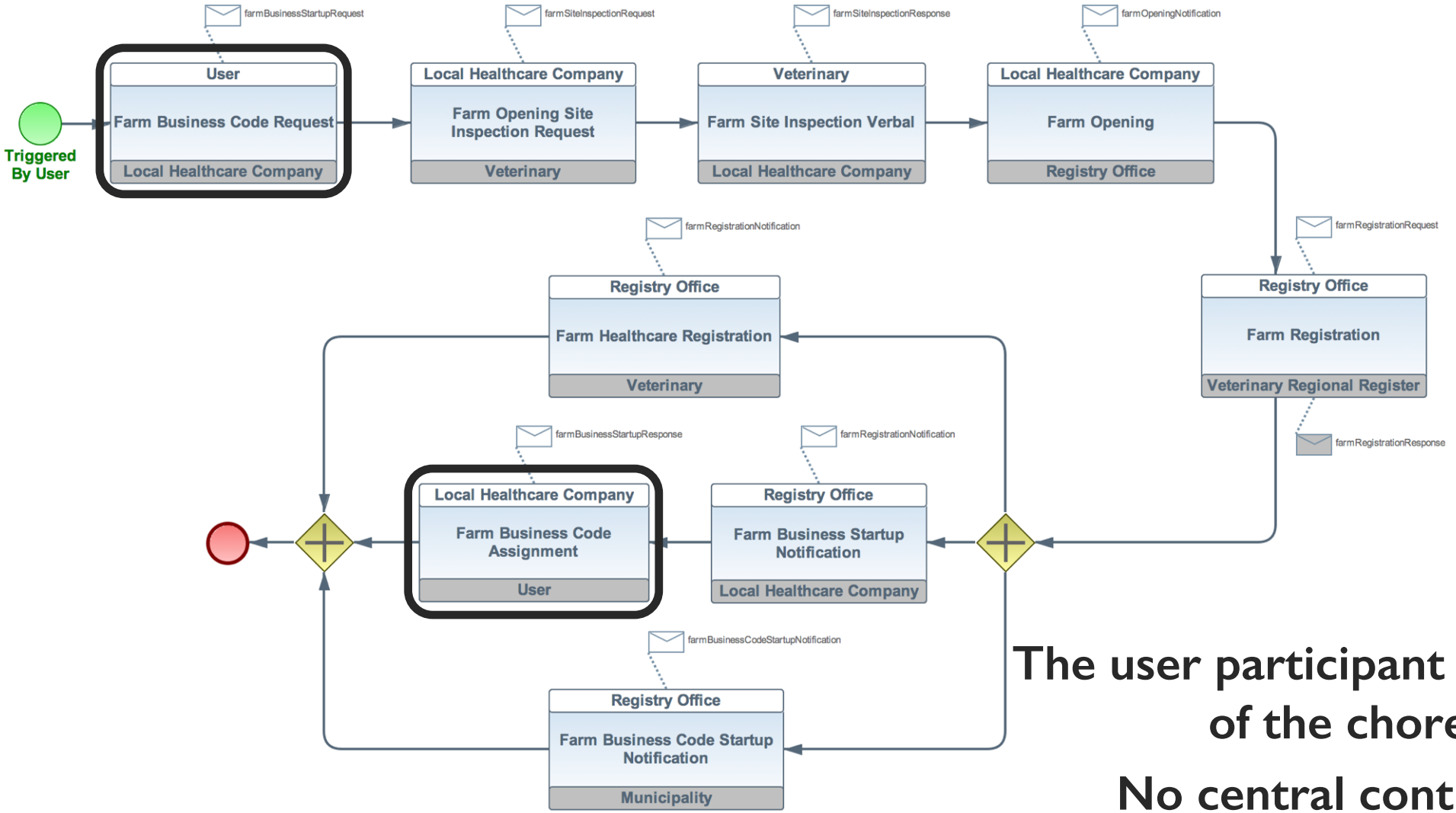
Farm Business Startup Choreography



Several cross-organizational Prosumer participants:

- Local Healthcare Company
- Veterinary
- Registry Office

Farm Business Startup Choreography

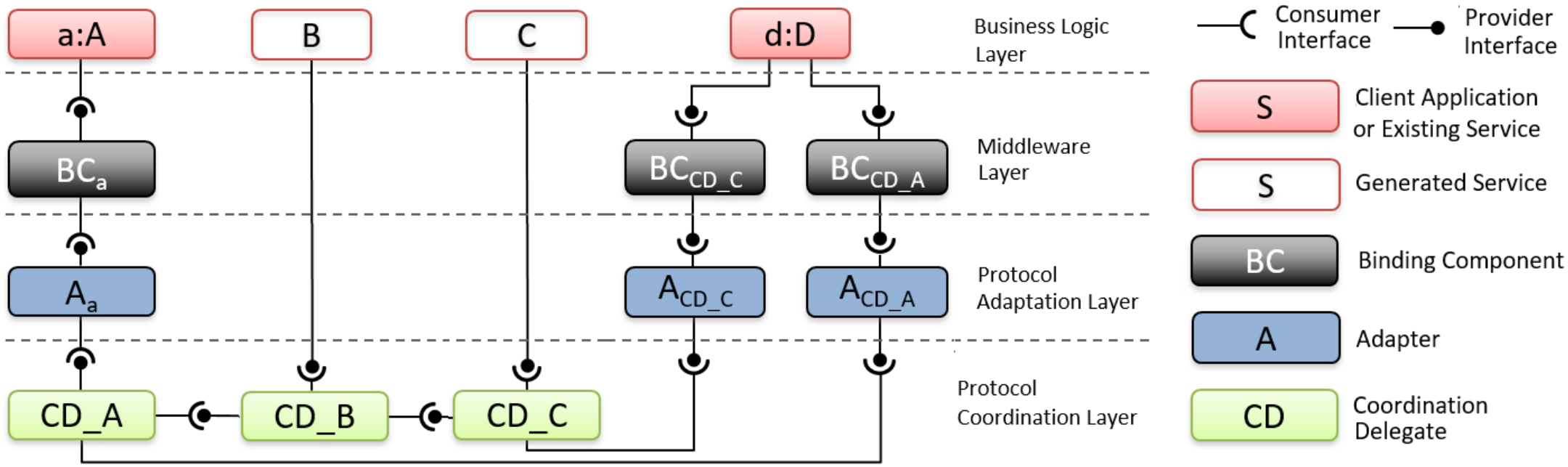


The user participant is a "simple" peer of the choreography

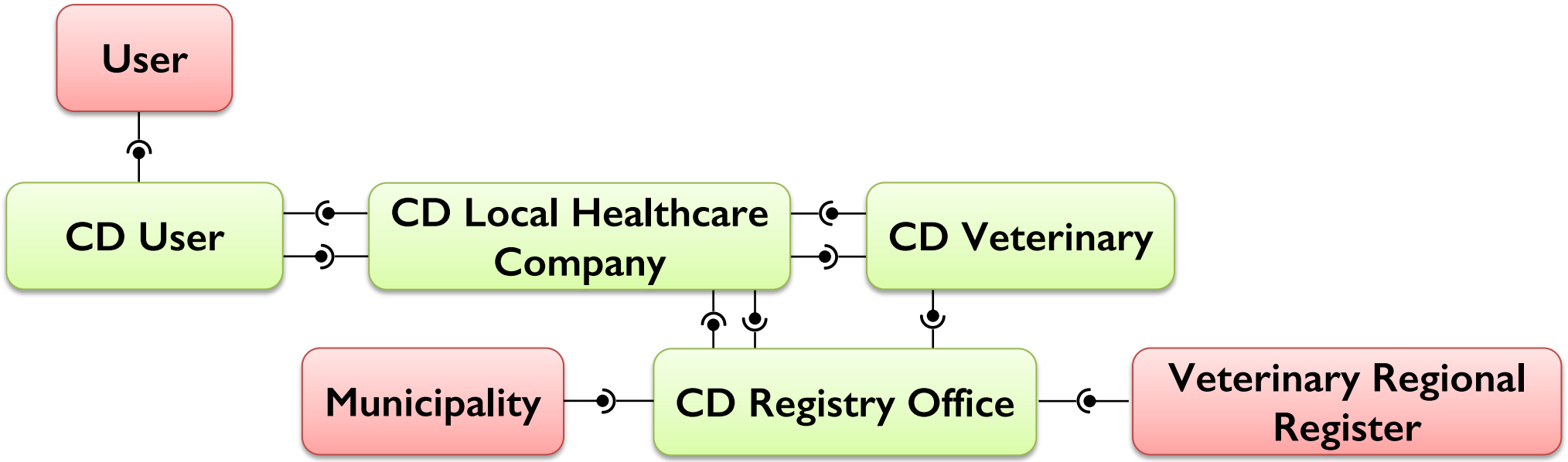
No central control mechanism!

Reference Architectural Style

CHOReVOLUTION – architectural style



Farm Business Startup Architecture



Coordination Delegates

Coordination Delegates - Overview

Fully decoupling of:

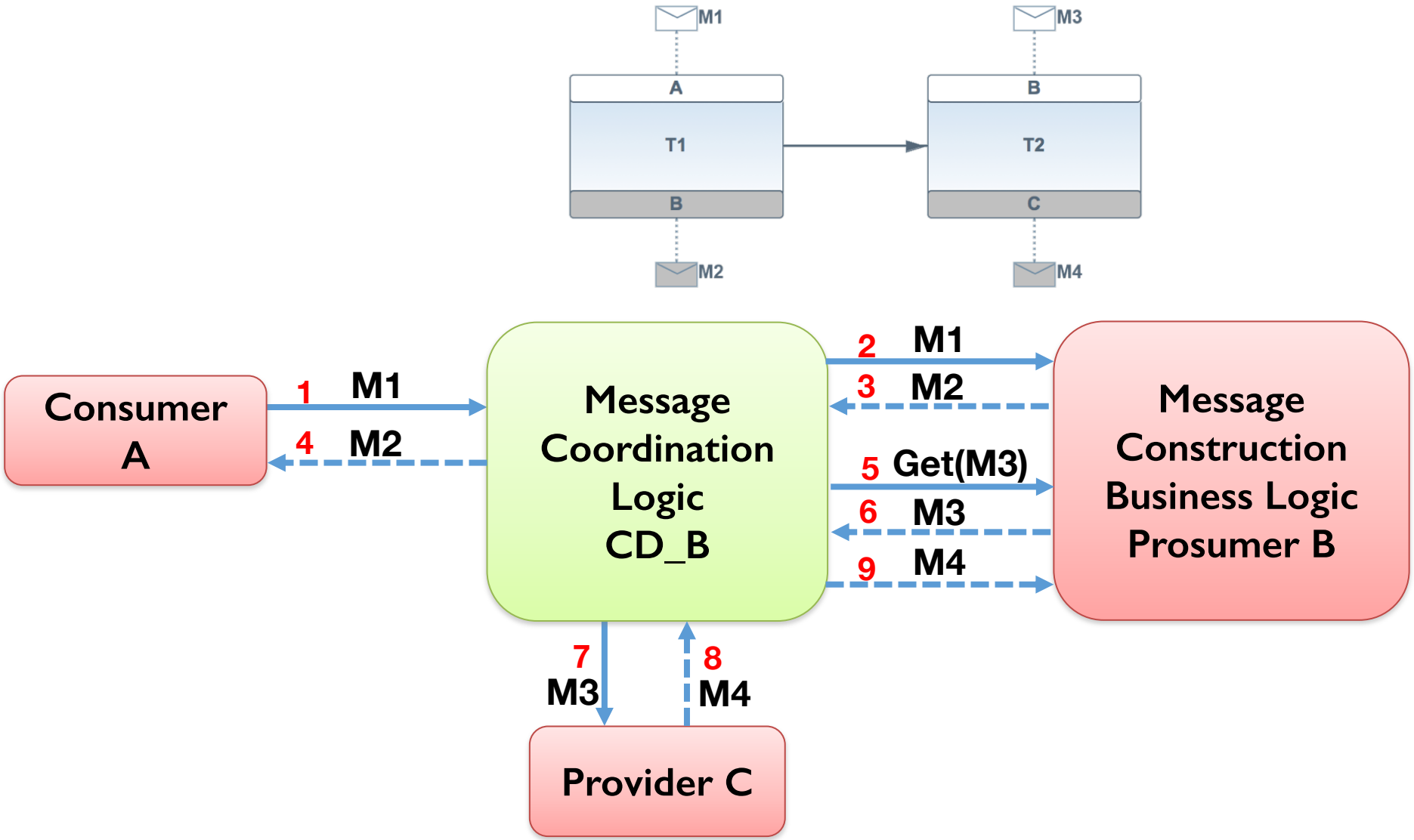
- **Coordination Logic**
  - Ensures that messages are exchanged according to the control flows prescribed by the specified choreography
  - Automatically generated by the Synthesis Processor

- **Business Logic**
  - Logic used for the construction of a message that has to be sent to another service
  - Skeleton automatically generated by the Synthesis Processor, to be filled by a developer exploiting the message persistence API

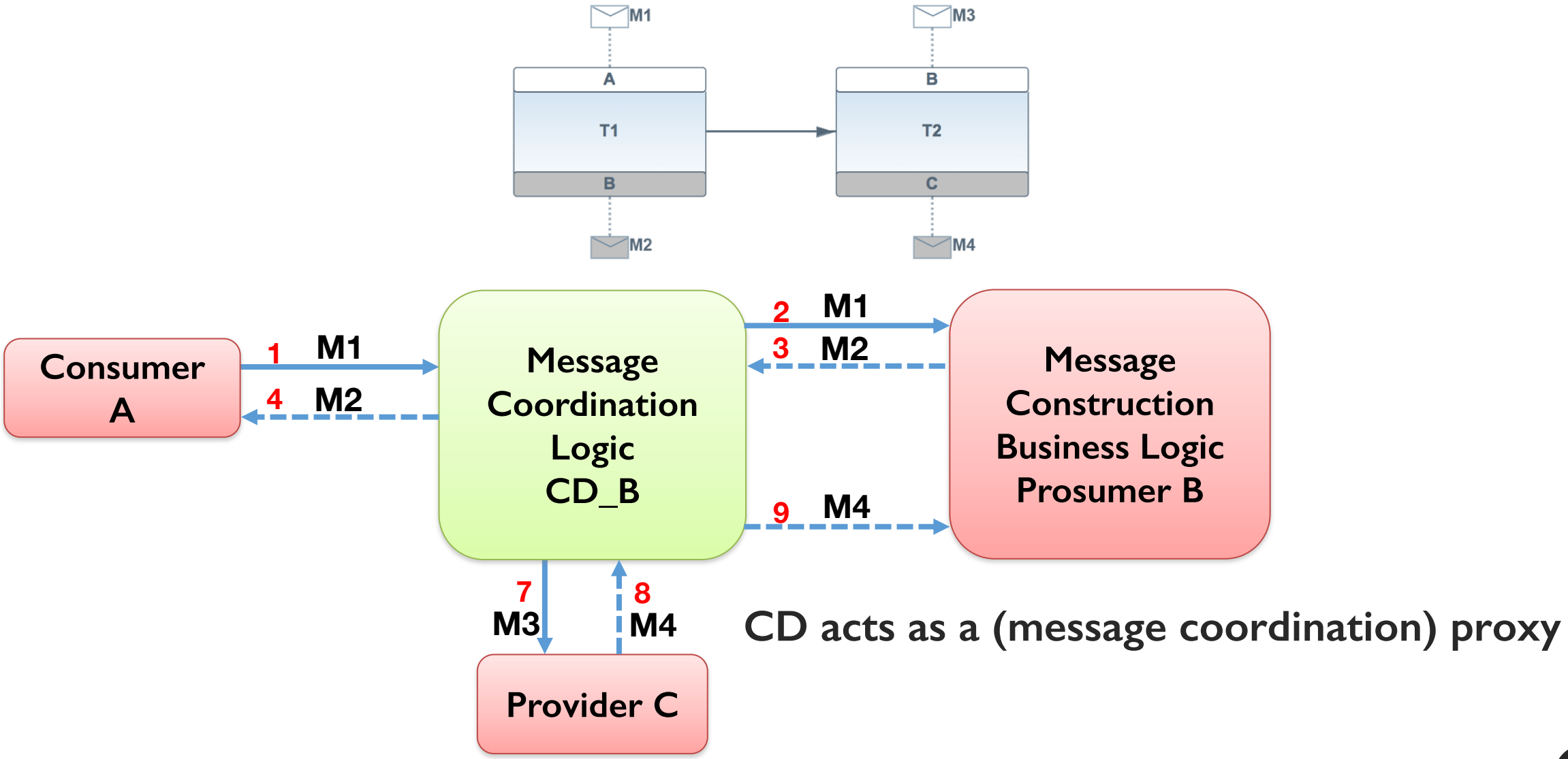
→ **Developers have only to implement the business logic internal to single choreography tasks**

Coordination Delegate Interaction Patterns

CD Interaction Pattern - most general case - I

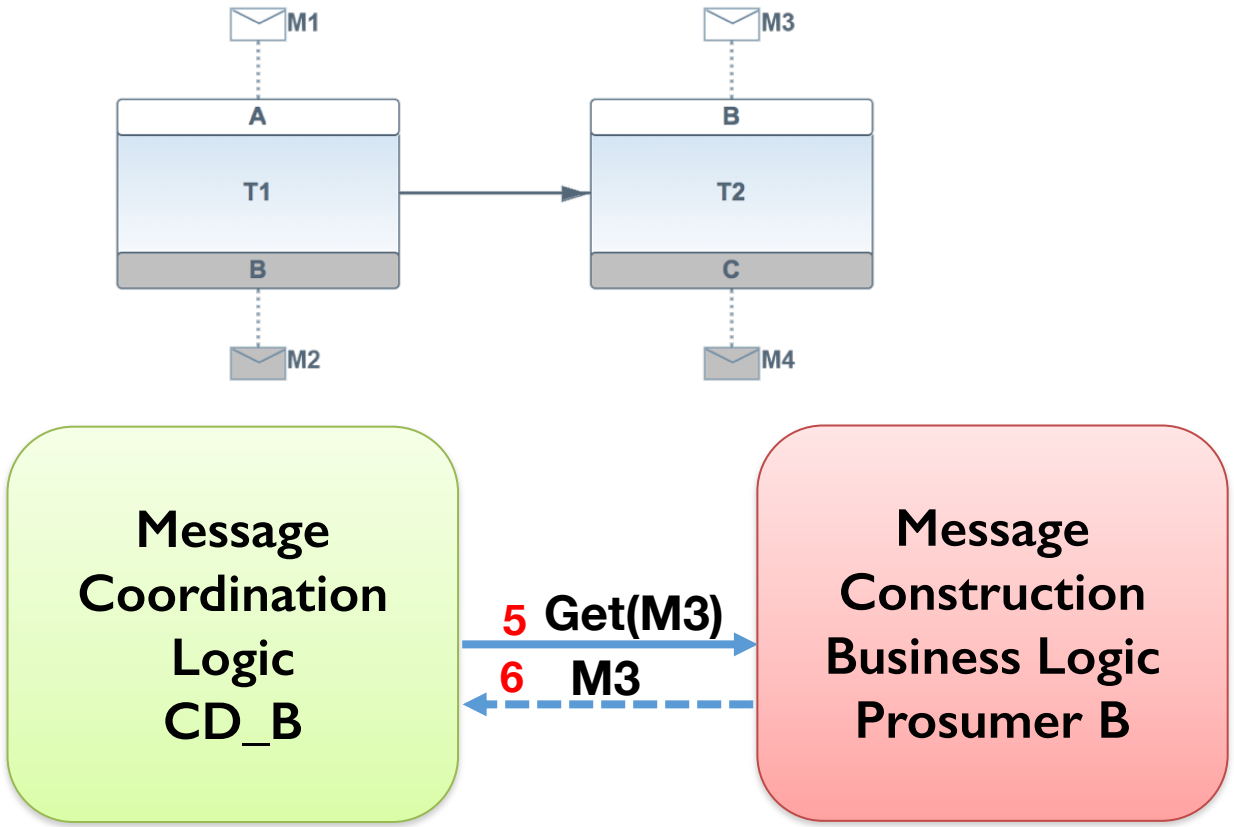


CD Interaction Pattern - most general case - 2



CD acts as a (message coordination) proxy

CD Interaction Pattern - most general case - 3

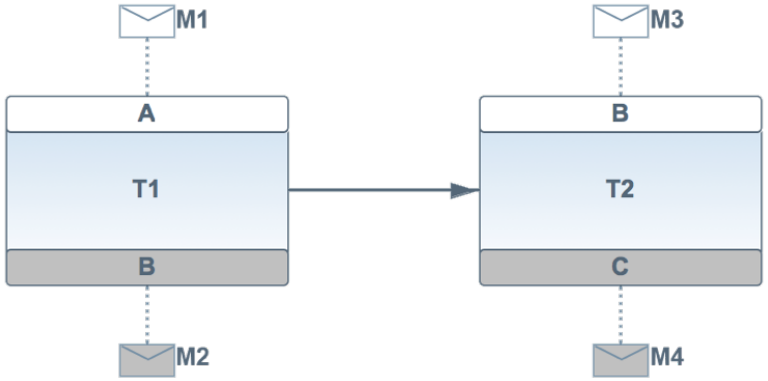


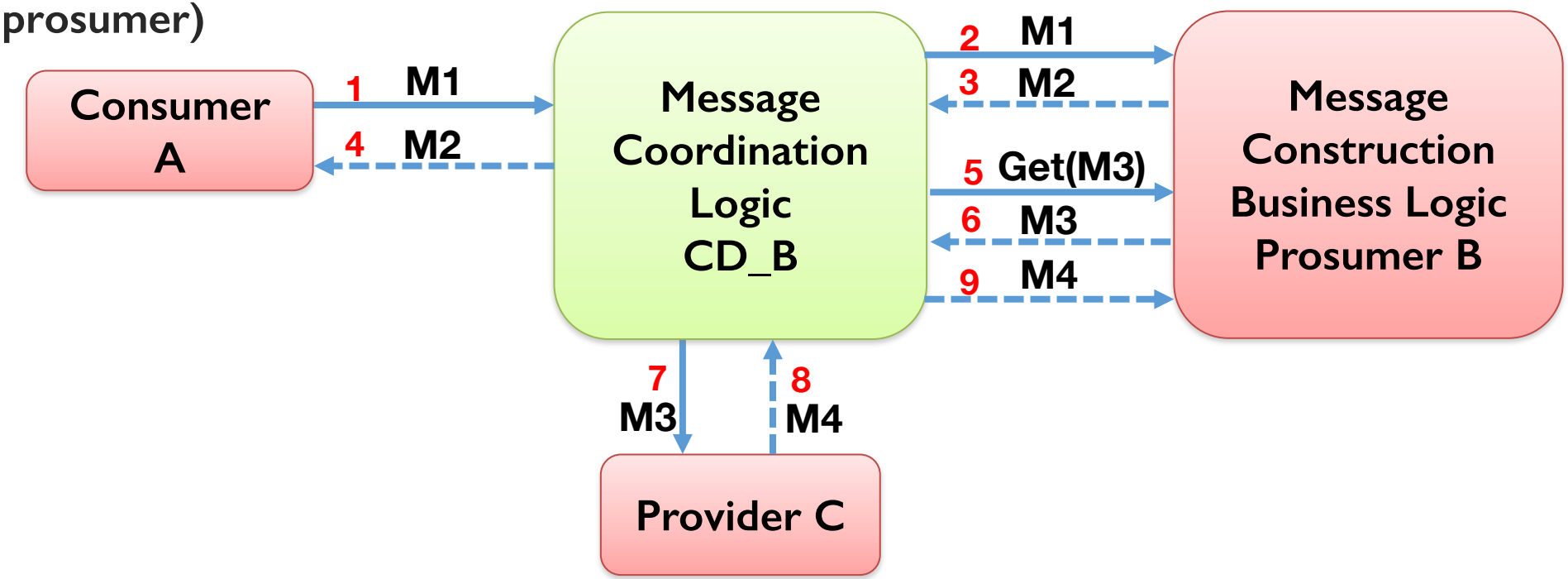
CD asks the prosumer for constructing a message

CD Interaction Pattern - most general case - 4

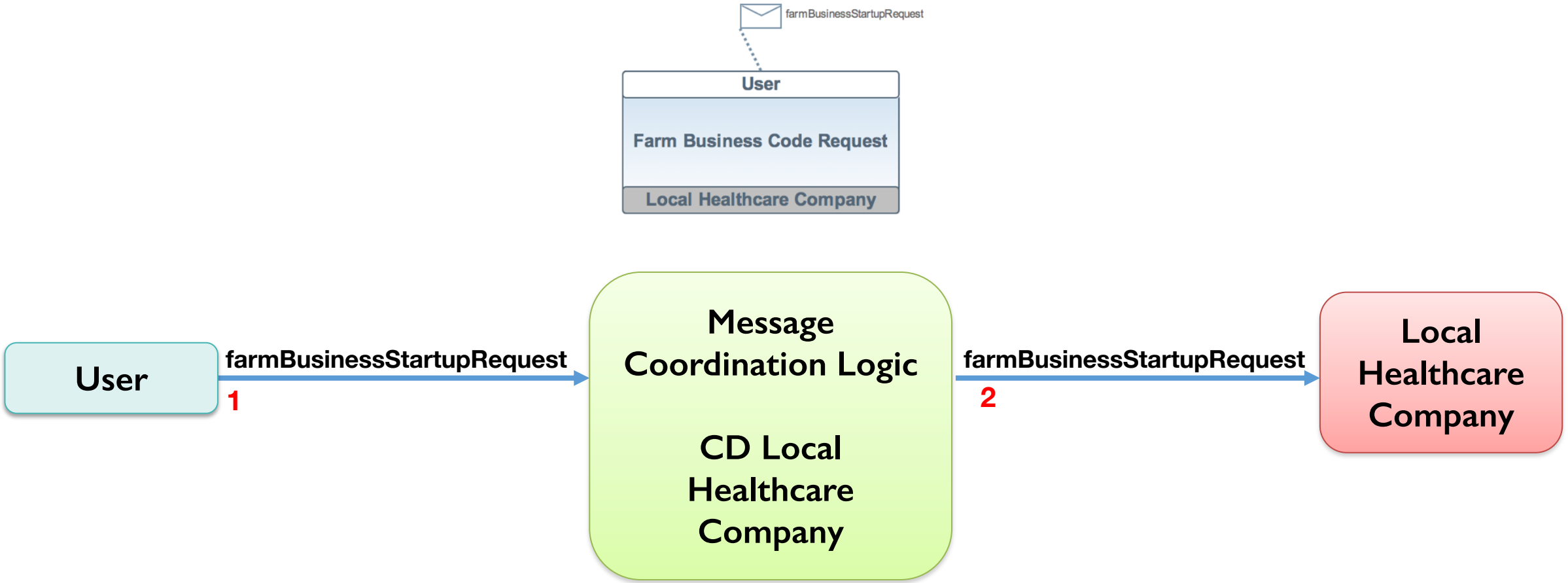
Different interaction scenarios can be derived based on:

- number of messages exchanged
- nature of the involved services (consumer, provider, prosumer)

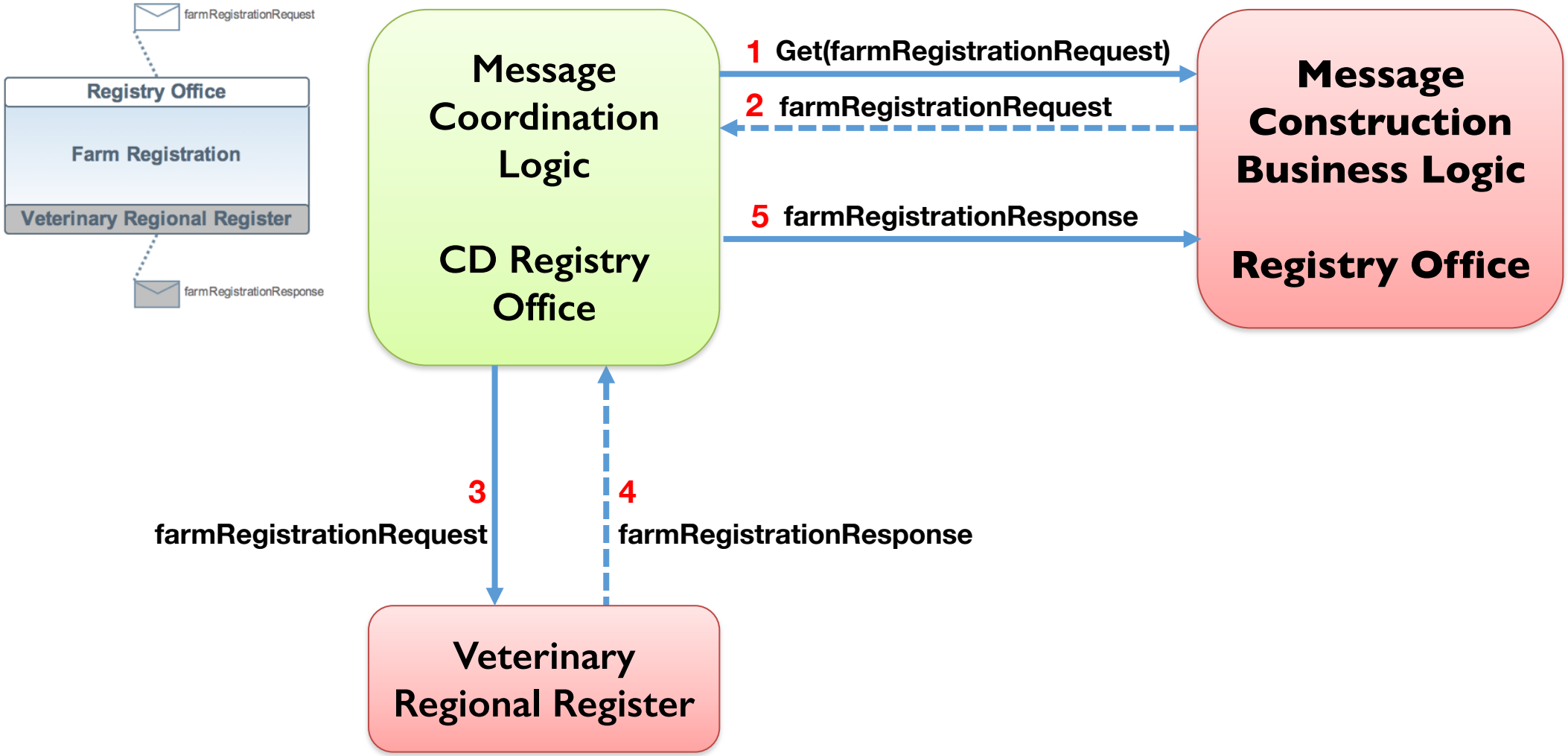




CD Interaction Pattern - client-prosumer/provider



CD Interaction Pattern - prosumer-provider/prosumer



Coordination Delegate Implementation

Coordination Delegate - Implementation

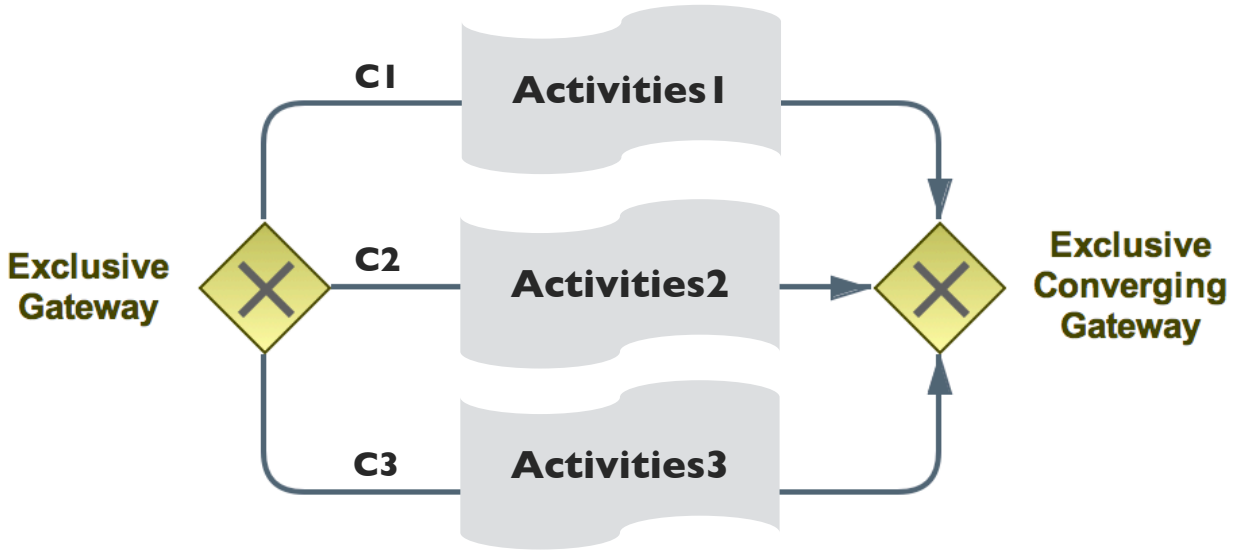
- **Coordination Logic**
  - BPEL process
  - Automatically generated by the Synthesis Processor

- **Business Logic**
  - Web service

Coordination Delegate - Coordination Logic

Coordination Delegate – Coordination Logic - Implementation - Choreography elements - BPEL Mapping

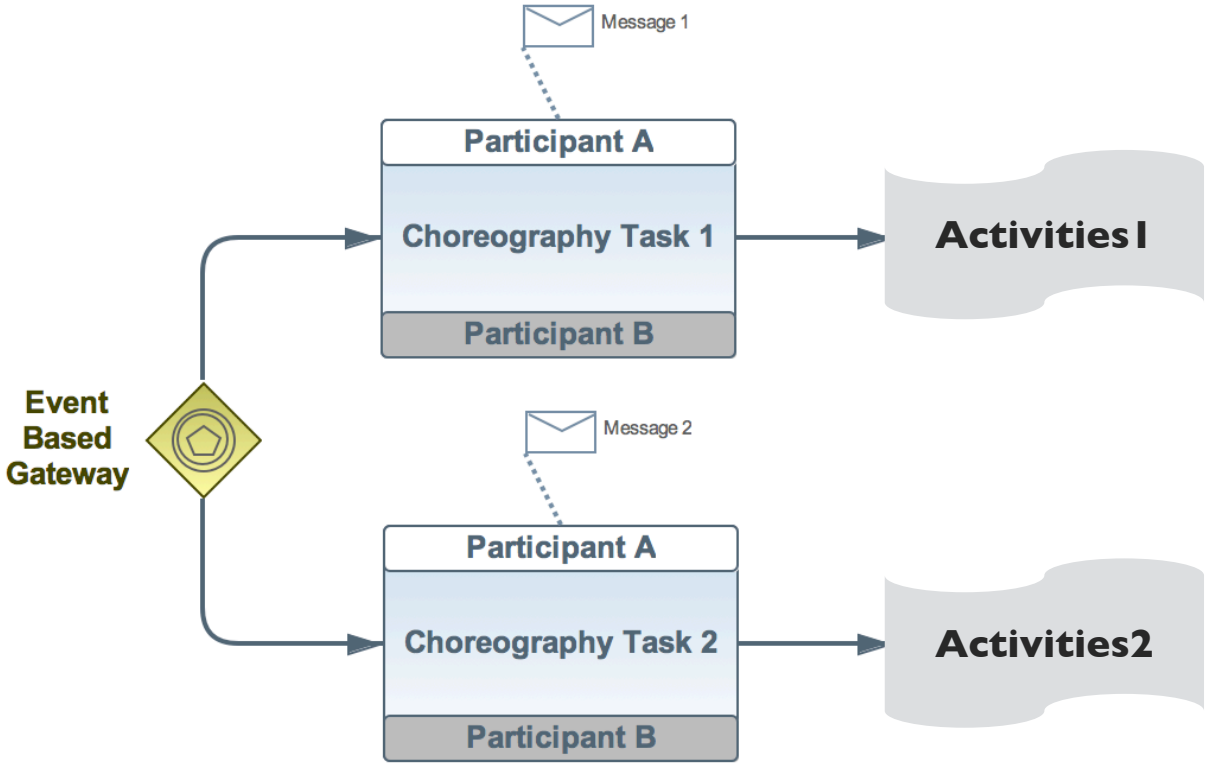
- **Exclusive Gateway**
  - outgoings mapped to <if>, <elseif>, <else>
  - conditions are processed and then set as <condition> of <if> or <elseif> activity




```
<if> <condition> [C1] </condition>
        [Activities1]
    <elseif> <condition> [C2] </condition>
        [Activities2]
    </elseif>
    <else>
        [Activities3]
    </else>
</if>
```

Coordination Delegate – Coordination Logic - Implementation - Choreography elements - BPEL Mapping

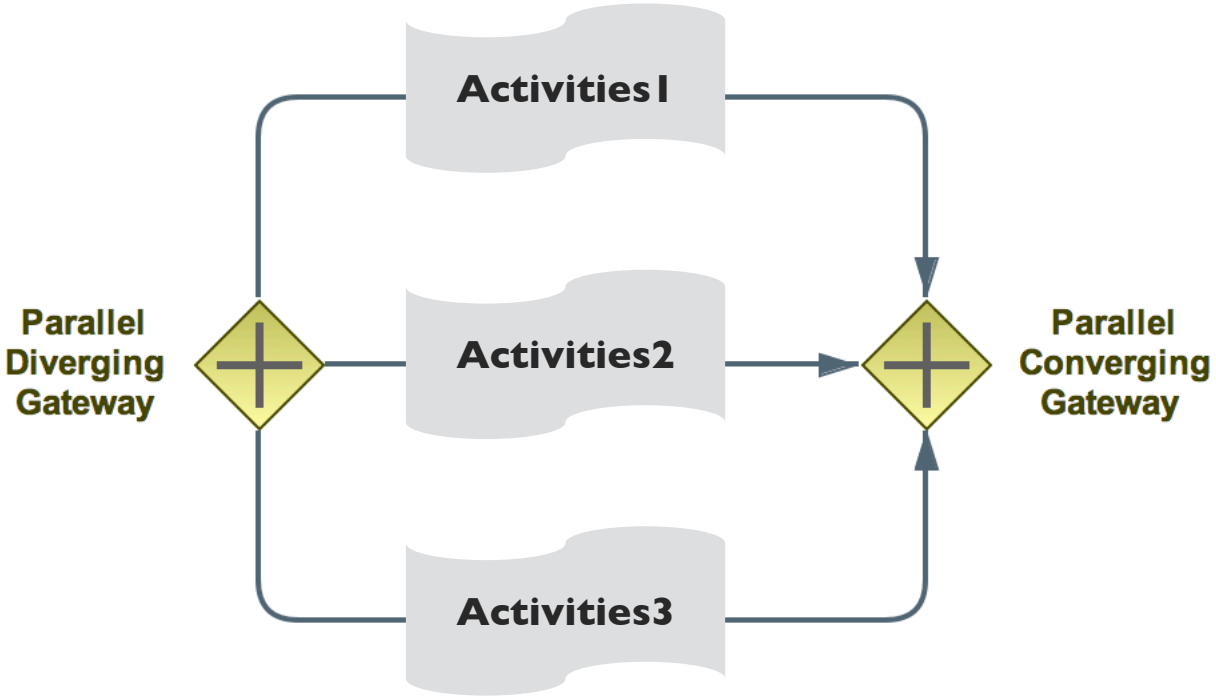
- **Event Based Gateway**
  - mapped to <onMessage>




```
<pick createInstance="[instantiate? 'yes':'no']">
      <onMessage operation="[Choreography Task 1-operation]"
            partnerLink="[Choreography Task 1-operation-interface]"
            portType ="[Choreography Task 1-port type]"
            variable="[Message 1-variable]" >
                  [Activities1]
      </onMessage>
      <onMessage operation="[Choreography Task 2-operation]"
            partnerLink="[Choreography Task 2-operation-interface]"
            portType ="[Choreography Task 2-port type]"
            variable="[Message 2-variable]" >
                  [Activities2]
      </onMessage>
</pick>
```

Coordination Delegate – Coordination Logic - Implementation - Choreography elements - BPEL Mapping

- **Parallel Gateway**
  - mapped to <flow>
  - the related activities are enclosed in a <sequence>




```
<flow>
    <sequence>
        [Activities1]
    </sequence>
    <sequence>
        [Activities2]
    </sequence>
    <sequence>
        [Activities3]
    </sequence>
</flow>
```

Thank You!

Any questions?

You can find me at:

massimo.tivoli@univaq.it