Enabling Scalable Cloud Service Choreographies

Marco Aurelio Gerosa

Carlos Eduardo Moreira dos Santos (Cadu)

Baile Project
FLOSS Competence Centre
University of São Paulo (USP)

HP Labs - November/2011

USP

Who we are

- **Marco Aurelio Gerosa**
  - Professor in the Computer Science Dept. of the University of São Paulo. Main areas of interest: Software Engineering and CSCW.

- **Cadu**
  - PhD student at University of São Paulo. He is currently studying on how to provide a middleware for ultra-large-scale service choreographies. Prof. Fabio Kon is his advisor.

Who we represent

- **Baile**
  - A joint research project involving the University of São Paulo and HP
- **GSD**
  - Distributed Systems Group
- **CCSL**
  - FLOSS Competence Center
- **DCC**
  - Department of Computer Science
- **USP**
  - University of São Paulo

University of São Paulo (USP)

- Latin America Largest University
- Total area: 76 millions m^2
- 89,000 students
- 5,800 professors
- 16,000 administrative staff
- 2,300 doctorate degrees per year
- > 25% of Brazilian scientific production

Source: http://www5.usp.br/en/usp-em-numeros/

* USP is the most relevant university in Latin America according to several rankings

Computer Science Department

- **40 full-time professors**
- **250 undergrads**
- **265 graduate students (190 masters + 75 PhD)**
- **Some research areas**
  - Database systems, distributed systems, software engineering, computer theory, optimization, artificial intelligence, vision and image processing, bioinformatics, musical computing, etc.

CCSL CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE
FLOSS Competence Center
IME-USP

Software Systems Group

- **Distributed systems professors:**

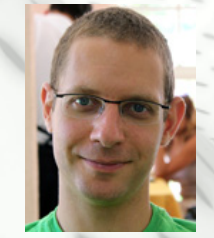

Fabio Kon


Alfredo Goldman

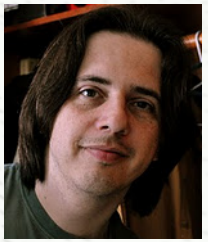

Marco Gerosa


Daniel Batista

- **~ 10 doctoral students**

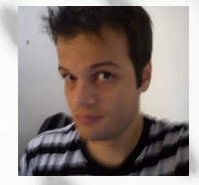
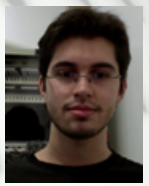




- **1 postdoc**

- **~ 20 masters students**

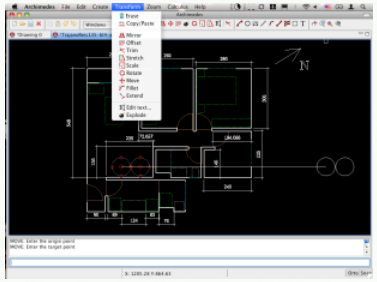




- **~ 10 undergrads**

USP

Some research projects

AcMus



- Borboleta (Telehealth with smartphones)
- Agile Methods for Software Development
- Qualipso (Quality in Open Source)
- Groupware Workbench (software components for groupware development)
- Xflow (Mining software repositories)
- InteGrade (Opportunistic Grid Computing)
- BAILE/CHOReOS - Web Service Choreographies

TANGO

vraptor
powered by Caelum

USP

Agenda

- **Back to the theme:**
  - *Enabling Scalable Cloud Service Choreographies*

- **Agenda:**
  - Future Internet view
  - Service choreographies
  - BAILE project
    - A middleware for service choreographies
    - Choreography analysis
    - Prototype demo

Future Internet

- **The growing scale of the Internet demands new theories and technologies.**
- **Initiatives:**
  - NSF Future Internet Design (FIND) initiative: http://www.nets-find.net
  - European Projects: http://www.future-internet.eu, http://www.choreos.eu
  - China Next Generation Internet (CNGI): http://www.cstnet.net.cn/english/cngi/cngi.htm
  - Japan initiative: http://akari-project.nict.go.jp/eng/overview.htm
- **No common definition yet**

Future Internet comprises

- **Internet of Content**
  - Content may be combined, mixed or aggregated to generate new content and may be cached or live, static or dynamic, monolithic or modular.
- **Internet of Services**
  - Features are exposed as services that can be integrated into other systems or used to dynamically create new systems.
- **Internet of Things**
  - A global network infrastructure, linking physical and virtual objects through the exploitation of data capture and communication capabilities.

Some requirements

- Scalability
- Interoperability
- Mobility
- Awareness and Adaptability
- Security, Privacy & Trust

Scalable Internet

Today's internet	Toward the Future Internet
1 billion PCs (2008)	1.78 billion PCs (2013)
647 million smartphones (2010)	1.82 billion smartphones (2013)
5 exabytes of data (2005)	990 exabytes of data (end of 2012)
10,000 services (2007)	Billions of services
10 billion terminals (2010)	100 billion terminals (2015)
Consumer traffic of 12.7 exabytes/month	Consumer traffic of 42 exabytes/month (2014)

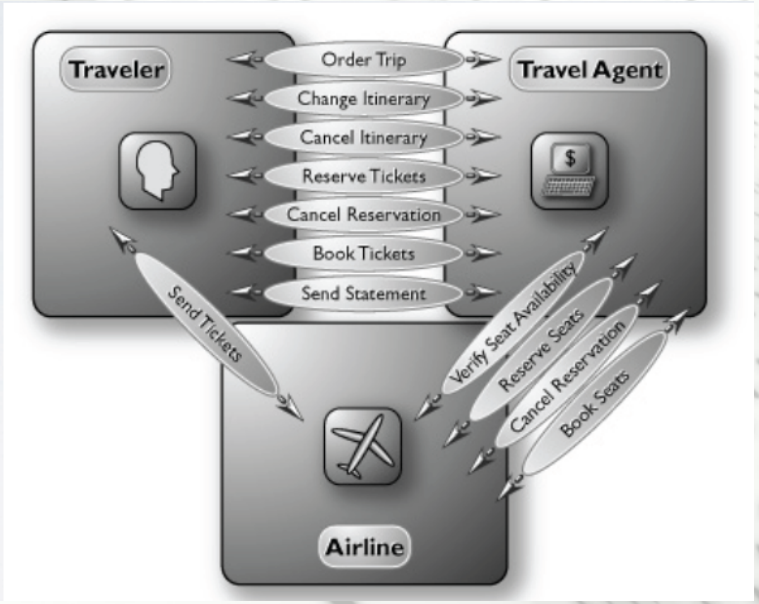
Data extracted from [Issarny et al., 2011]

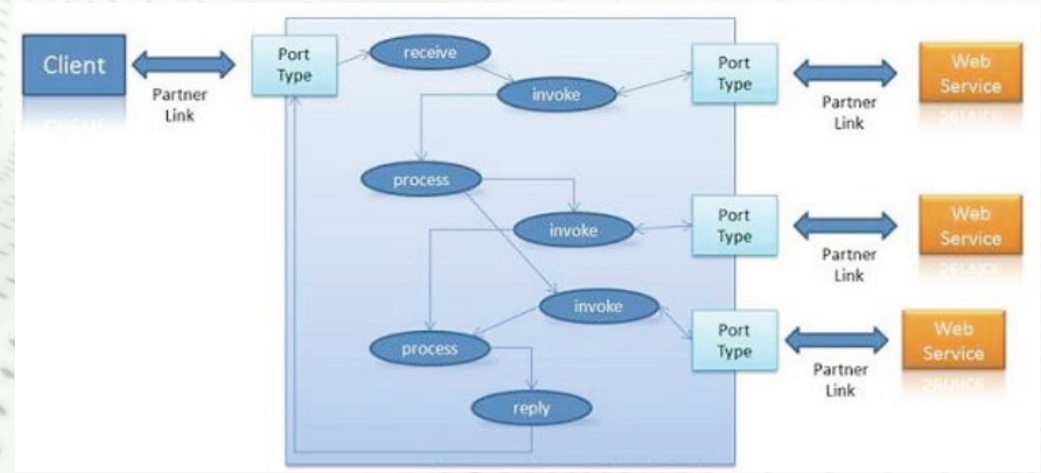
Interoperable Internet

Today's internet	Toward the Future Internet
Islands of interconnected elements	Internet-scale connection of highly heterogeneous elements (vehicles, sensors, mobiles devices, home appliances, etc.)
Service/content mashups leading to the provision of new, diverse services by prosumers	Global-scale services/content mashups creating new services/content with different types and formats
Wide-spread usage of smart mobile devices with limited resources	Global scale usage of mobile devices with ever-growing capabilities. The majority of the connected devices will be mobile.
Service-oriented software development is mostly a static process	Software development is a completely dynamic user-centric process.

Approaches for service composition

- **Service compositions deal with order and interdependencies among services, the flow of information, transactions etc.**
- **Strategies: service orchestration / choreographies**
  - Service orchestration - centralized control
  - Service choreography - peer-to-peer collaboration without a central node of control



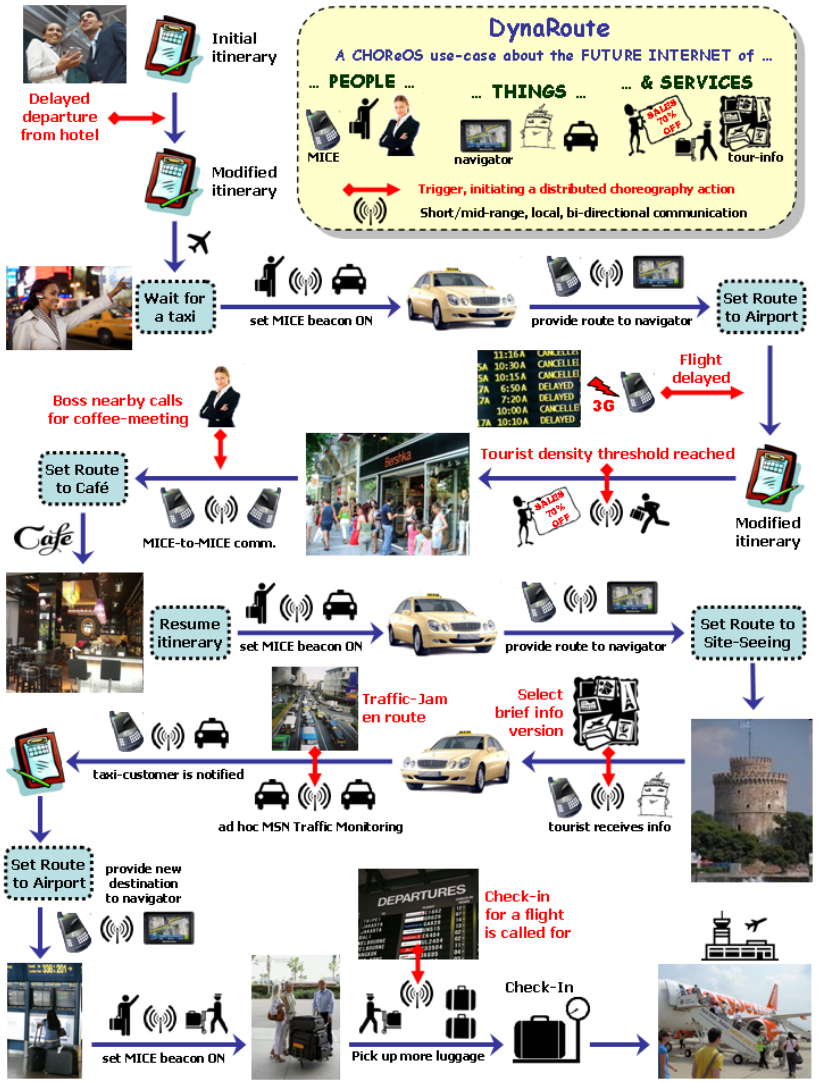


History

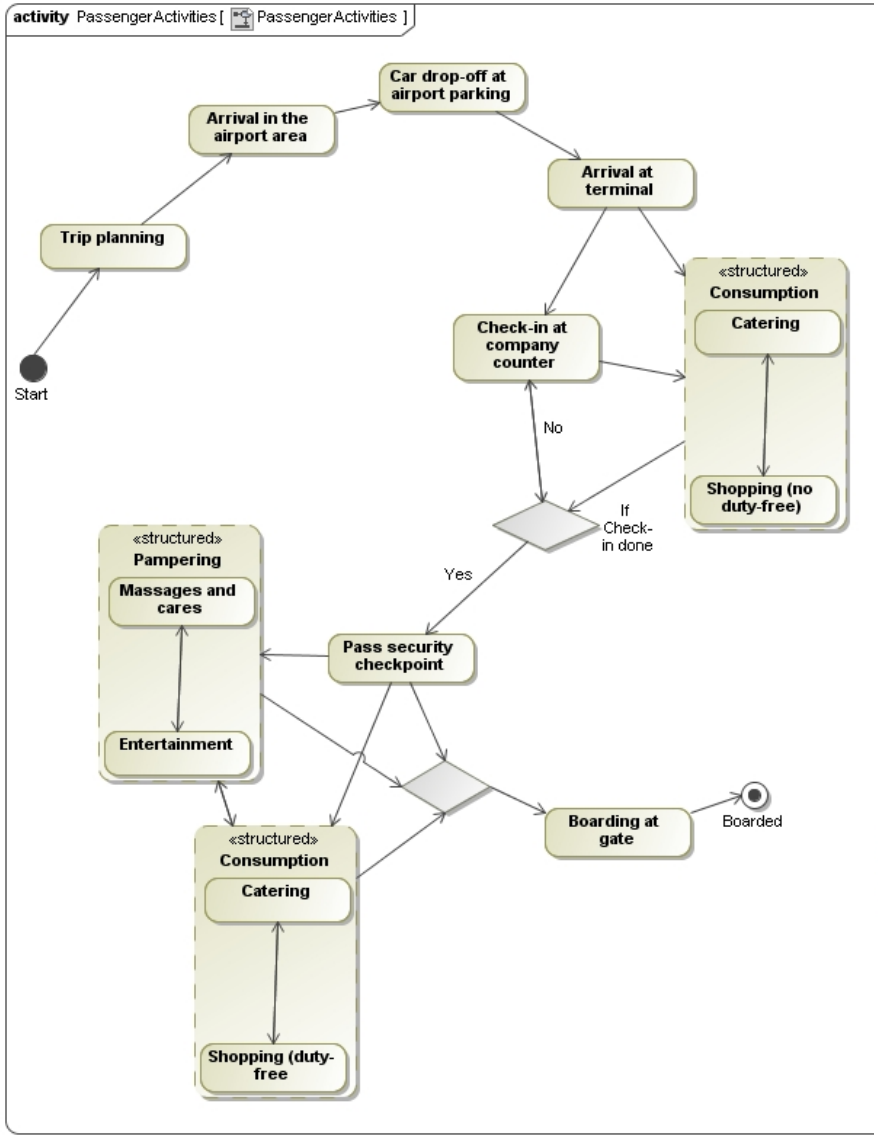
- **Web services adoption has grown during the last years; its usage has become a *de facto* standard for communication among high-level Internet systems.**
- **Orchestration is widely used for composing web services.**
  - But its centralized approach to composition has scalability and reliability problems.
- **Choreographies are Web services compositions organized in a decentralized, distributed manner, with no single point of failures.**
  - Global perspective
  - No central coordination node
  - Cross-enterprise business
  - Roles

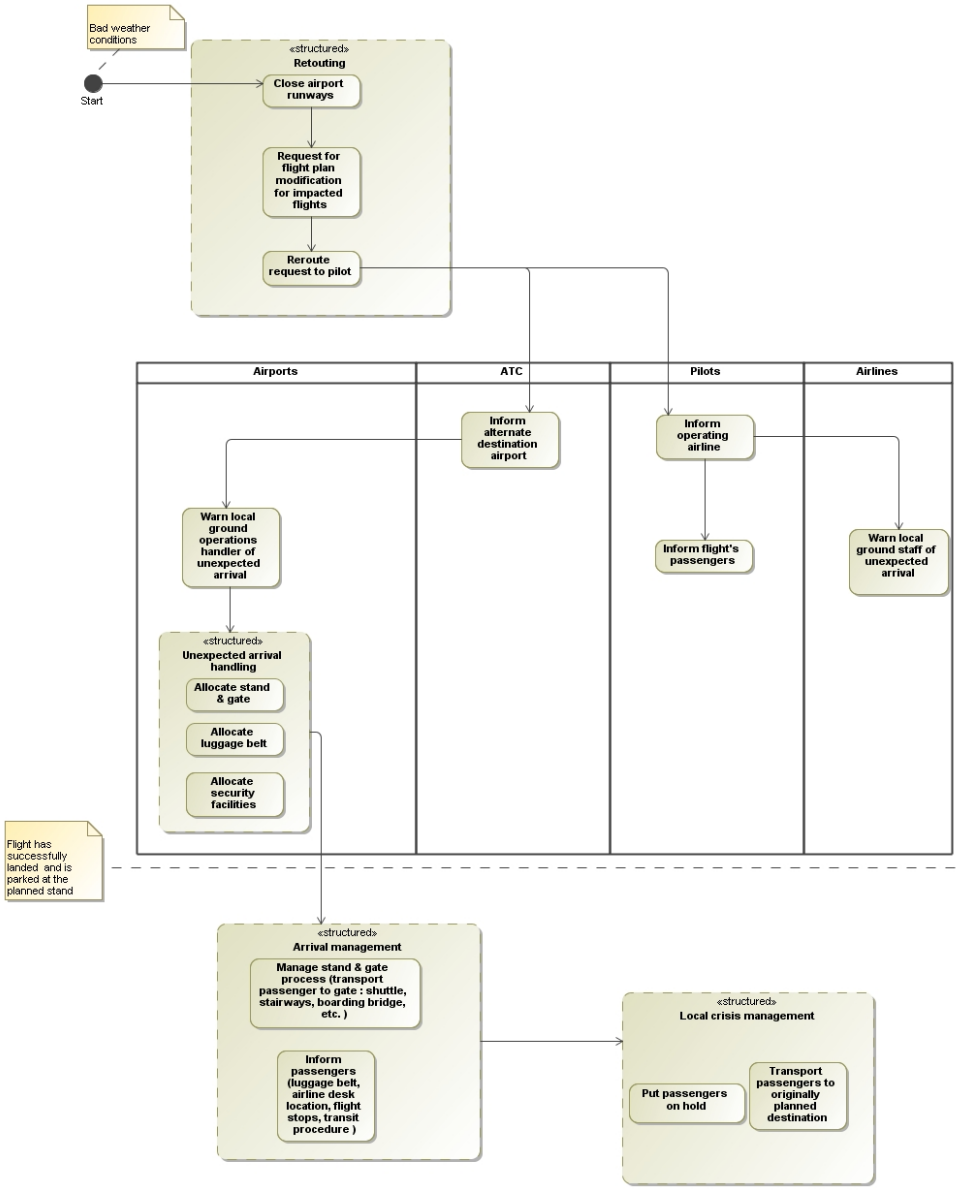
CCSL CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE
FLOSS Competence Center IME-USP

Scenarios for service choreographies

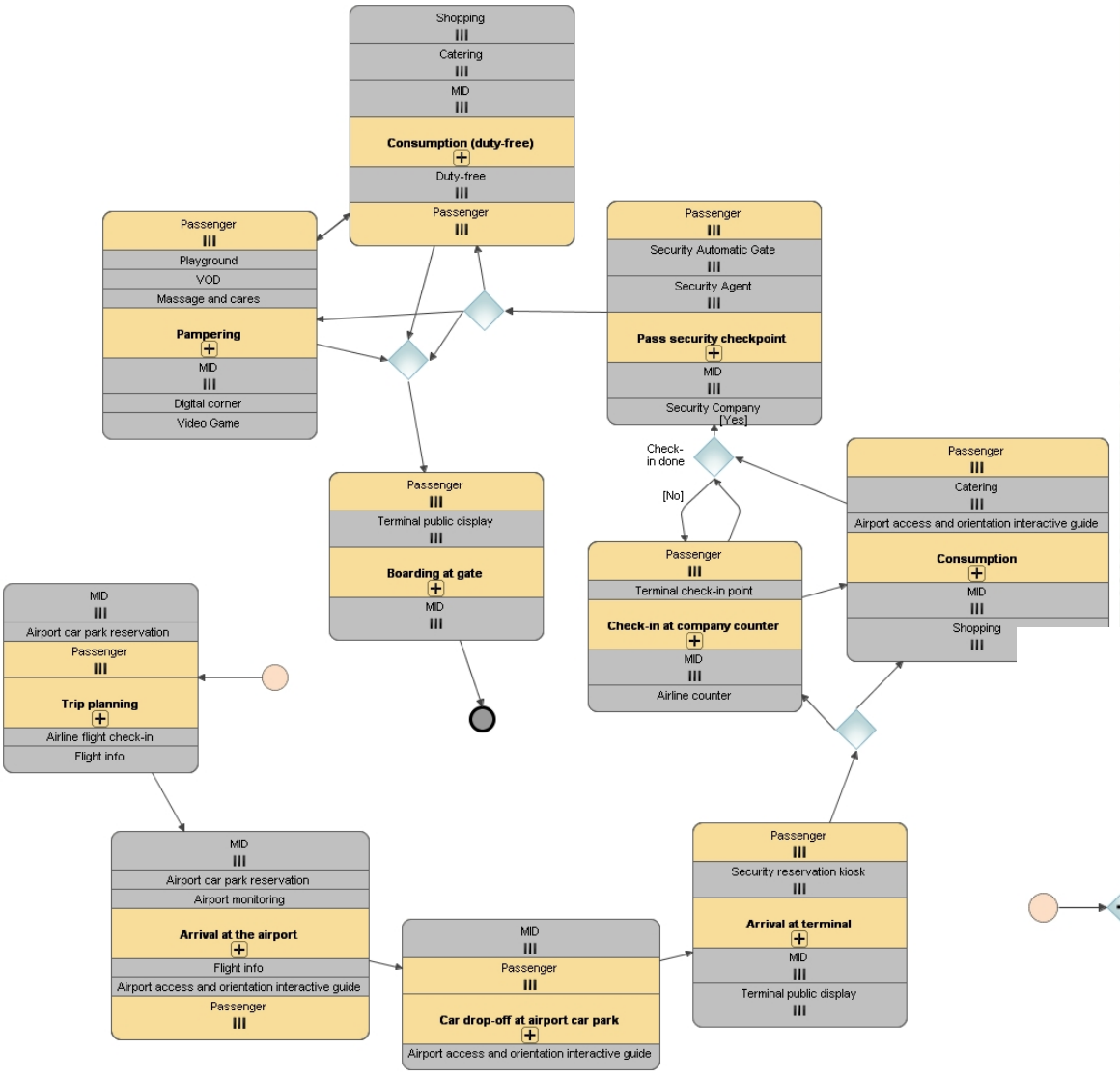


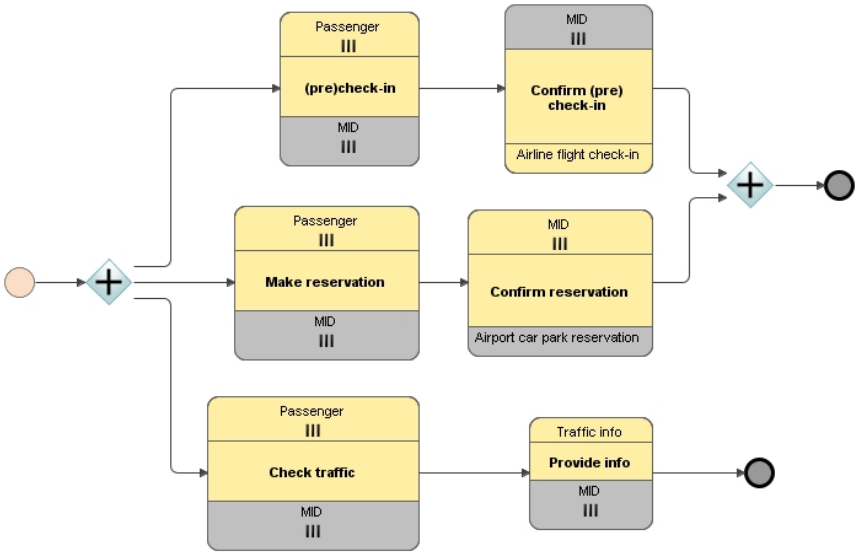
Choreography scenarios



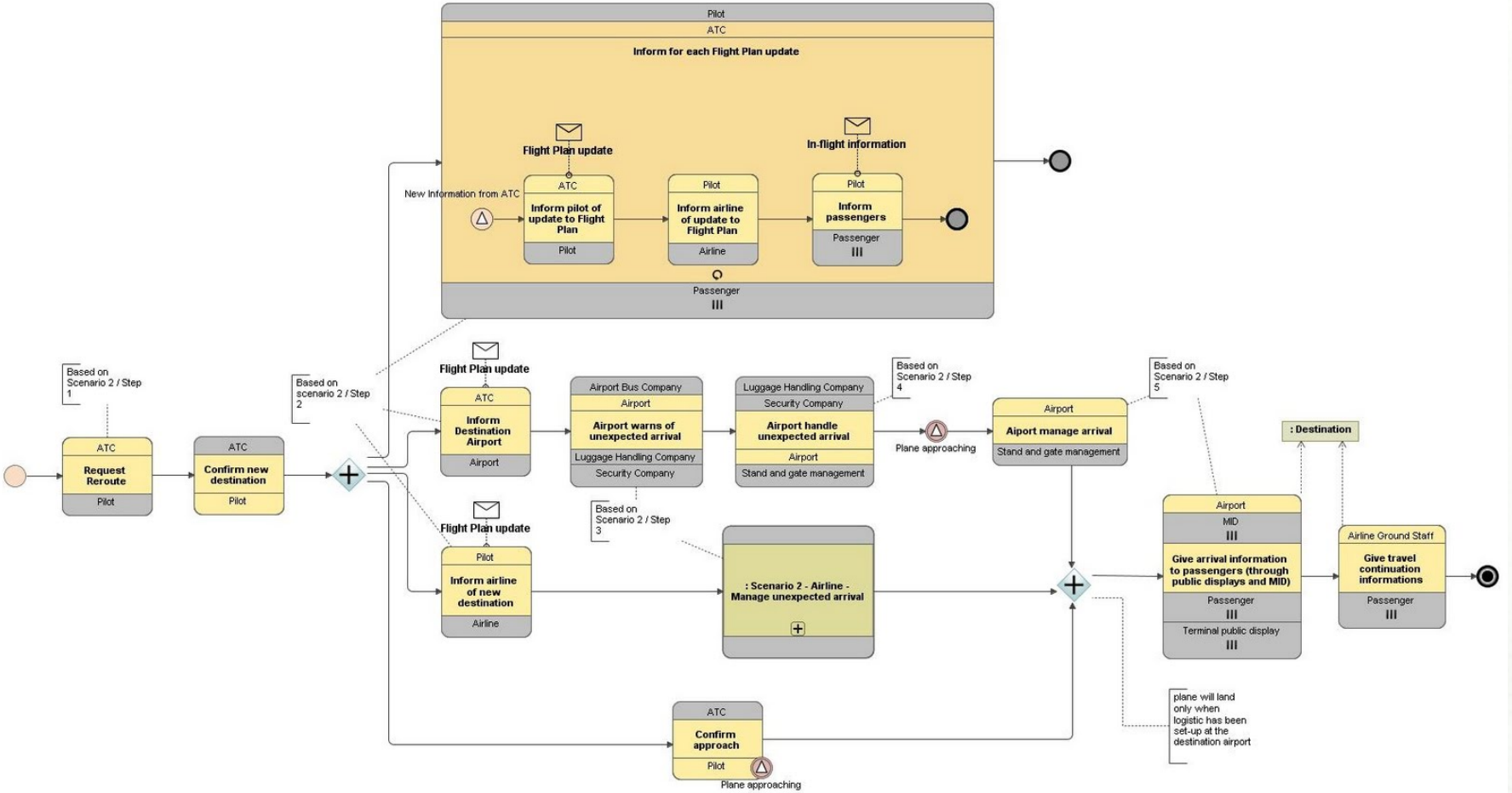


BPMN2 model


Shopping
Catering
MID
Consumption (duty-free)
Duty-free
Passenger
Passenger
Playground
VOD
Massage and cares
Pampering
MID
Digital corner
Video Game
Passenger
Security Automatic Gate
Security Agent
Pass security checkpoint
MID
Security Company
[Yes]
Check-in done
[No]
Passenger
Terminal public display
Boarding at gate
MID
Passenger
Catering
Airport access and orientation interactive guide
Consumption
MID
Shopping
Passenger
Terminal check-in point
Check-in at company counter
MID
Airline counter
MID
Airport car park reservation
Passenger
Trip planning
Airline flight check-in
Flight info
MID
Airport car park reservation
Airport monitoring
Arrival at the airport
Flight info
Airport access and orientation interactive guide
Passenger
MID
Passenger
Car drop-off at airport car park
Airport access and orientation interactive guide
Passenger
Security reservation kiosk
Arrival at terminal
MID
Terminal public display

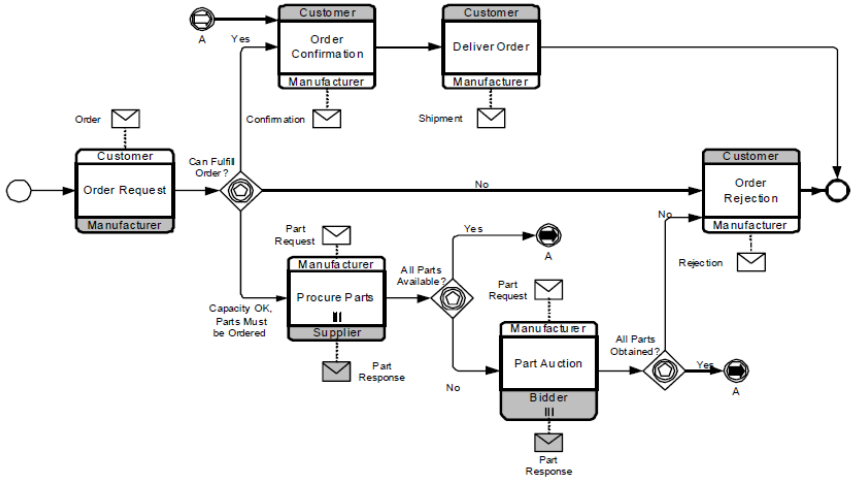

Passenger
(pre)check-in
MID
MID
Confirm (pre) check-in
Airline flight check-in
Passenger
Make reservation
MID
MID
Confirm reservation
Airport car park reservation
Passenger
Check traffic
MID
Traffic info
Provide info
MID

BPMN2 model


Pilot
ATC
Inform for each Flight Plan update
Flight Plan update
In-flight information
New Information from ATC
ATC
Inform pilot of update to Flight Plan
Pilot
Pilot
Inform airline of update to Flight Plan
Airline
Pilot
Inform passengers
Passenger
Passenger
Based on Scenario 2 / Step 1
ATC
Request Reroute
Pilot
ATC
Confirm new destination
Pilot
Based on scenario 2 / Step 2
Flight Plan update
ATC
Inform Destination Airport
Airport
Airport Bus Company
Airport
Airport warns of unexpected arrival
Luggage Handling Company
Security Company
Luggage Handling Company
Security Company
Airport handle unexpected arrival
Airport
Stand and gate management
Based on Scenario 2 / Step 4
Plane approaching
Airport
Aiport manage arrival
Stand and gate management
Based on Scenario 2 / Step 5
: Destination
Flight Plan update
Pilot
Inform airline of new destination
Airline
Based on Scenario 2 / Step 3
: Scenario 2 - Airline - Manage unexpected arrival
Airport
MID
Give arrival information to passengers (through public displays and MID)
Passenger
Terminal public display
Airline Ground Staff
Give travel continuation informations
Passenger
ATC
Confirm approach
Pilot
Plane approaching
plane will land only when logistic has been set-up at the destination airport

Scenarios for service choreographies

- Complex workflow involving multiple enterprises
- Interaction among several distributed orchestrations
- Fault tolerance
- Data intensive computing [Baker et al., 2009]



A. Barker, P. Besana, D. Robertson, and J. B. Weissman, “The Benefits of Service Choreography for Data-Intensive Computing,” in *In Proceedings of the 7th international workshop on Challenges of large applications in distributed environments*, 2009, p. 1--10.

BAILE Motivation

- Choreographies have a good potential to deliver higher fault tolerance, adaptability, configurability, global optimization, and freedom to grow.
- This was identified some time ago (early 2000s) but all standardization initiatives failed up to now (WSCI, WS-CDL). (2011: OMG BPMN2)
- Many companies currently use ad hoc approaches to choreographies developed in-house.
- However, as current ad hoc choreographies get larger and more intricate, they can easily become unmanageable.
- The CHOReOS and BAILE projects will tackle these issues.

Challenges

- **How to support the**
  - Coordination of an ultra large number of services widely distributed and moving on a highly heterogeneous and changing network, composed of heterogeneous devices (from tiny scale sensors/actuators to infrastructure servers).
  - Execution of services with varying load in dynamically evolving applications.

- **Our approach: The use of a middleware to enable scalable service provisioning based on Grid and Cloud computing for service choreographies**

Our research project

- **Scalable Web Service Choreographies for the Future Internet**

- European Commission funding:
  - CHOReOS – http://www.choreos.eu
- HP Brasil funding:
  - BAILE – http://ccsl.ime.usp.br/baile

- **Some research lines:**
  1. Development of a cloud-enabled middleware for service choreographies
  2. Verification & validation of choreographies
  3. Choreographies topology analysis
  4. Performance analysis for large scale choreographies

(1) Development of a cloud-enabled middleware for service choreographies

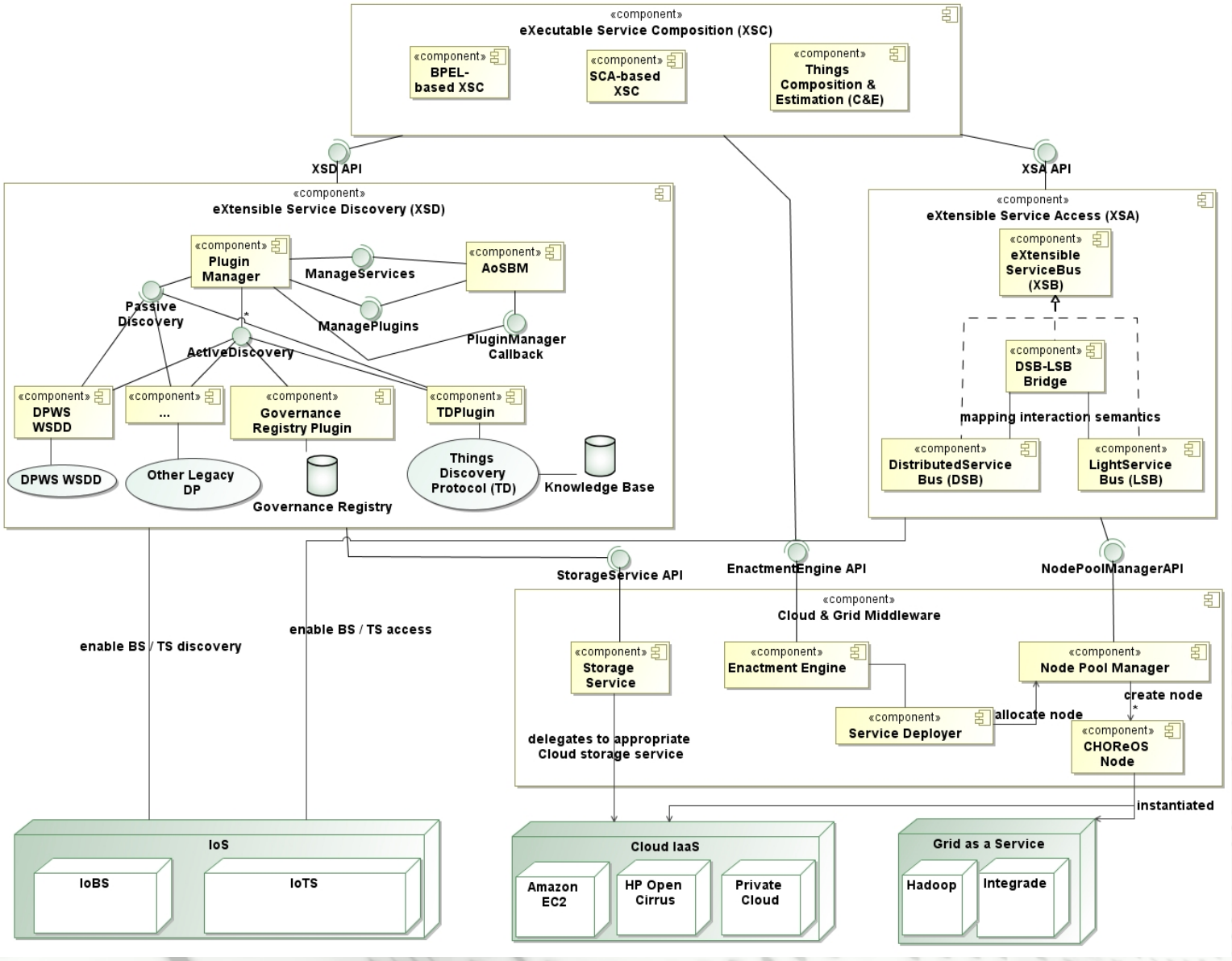
Middleware requirements

- **1 -** Compile a choreography
  - Given a BPMN2 Choreography description and the description of the web services generate the executable bytecode for this choreography.
- **2** - Enactment of a Choreography
  - Given a choreography executable bytecode, enacts that choreography by (1) creating the required web services, (2) deploying the web services whose executable code is available, (3) assigning web services to the roles of the choreography, (4) enabling the choreography to receive messages.
- **3** - Instantiation of Choreographies in a Cloud Infrastructure
  - Instantiate the choreography middleware and the nodes of a choreography in virtual machines of a given cloud infrastructure, e.g., Amazon EC2, Open Cirrus, or OpenNebula. This must be performed in a way that maximizes the QoS as perceived by the final users and minimizes resource consumption.

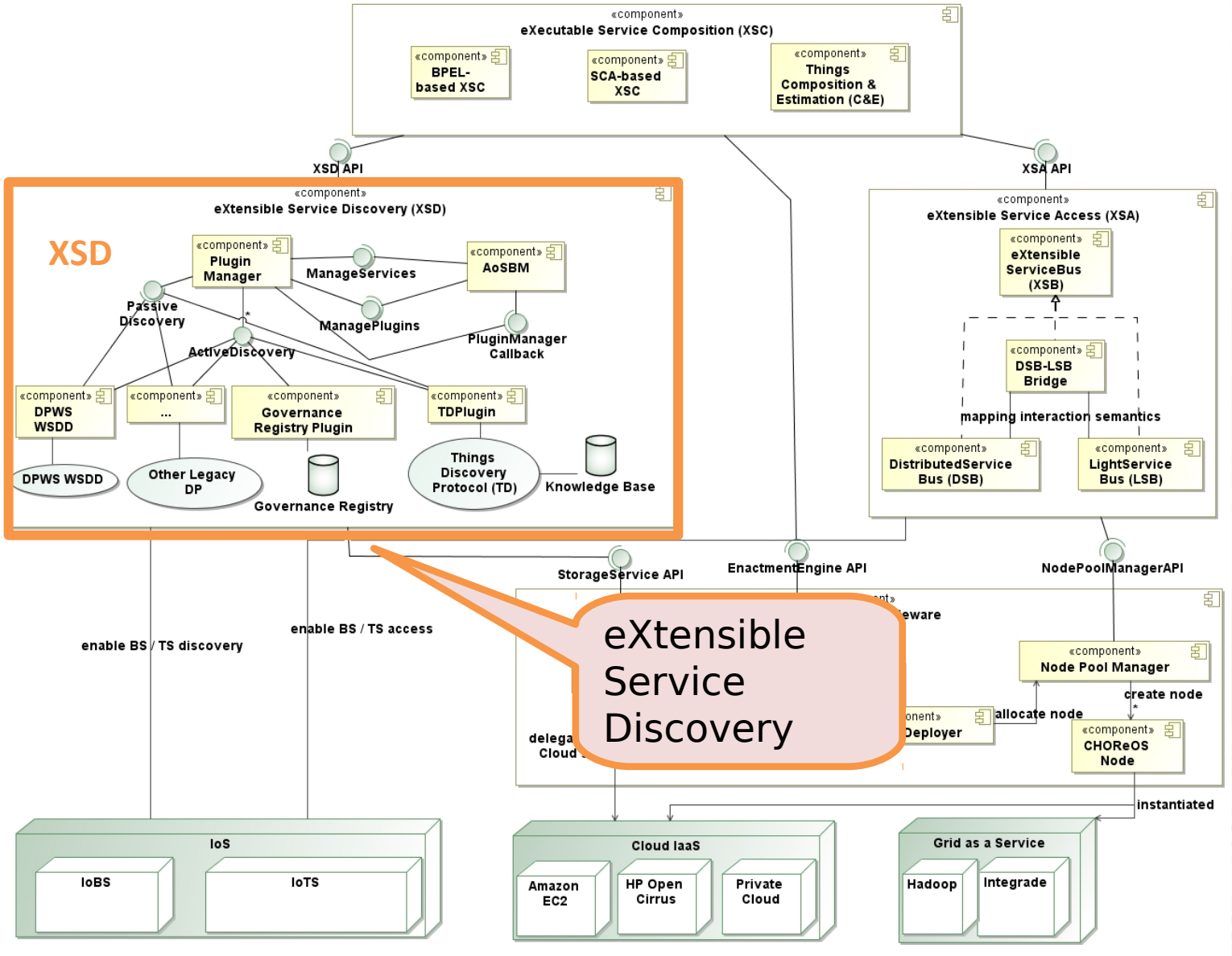
Choreographies and the cloud

- Our goal in the BAILE project is to study how scalable a choreography middleware can get.
- We aim at developing choreographies with
  - tens to thousands of services
  - running on hundreds to thousands of nodes
  - servicing thousands to millions of users
  - running computationally/data intensive tasks
- This involves efforts in
  - Software Architecture and Engineering
  - Investigation of novel methods for creating, managing, and processing choreographies

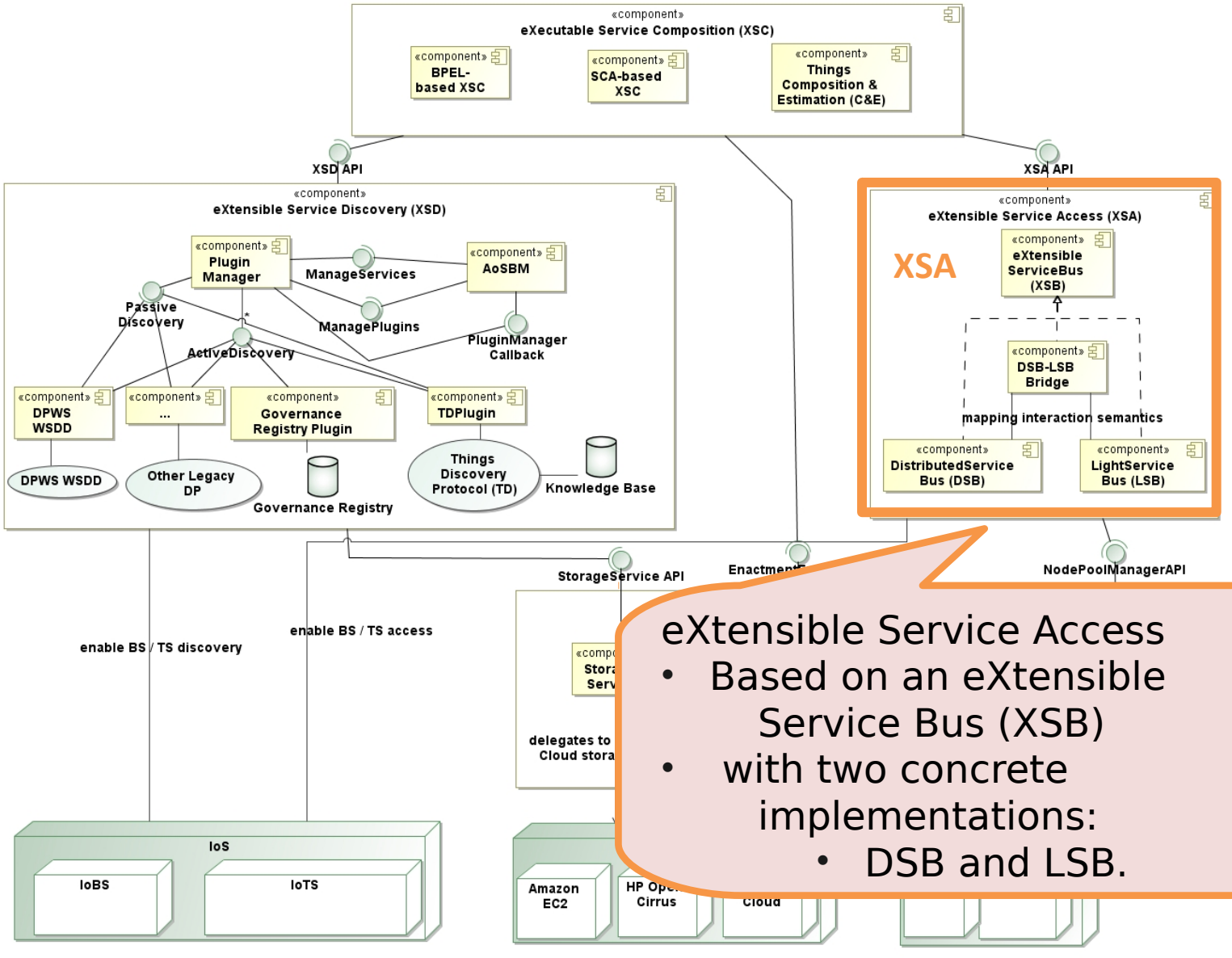
Middleware Architecture



Middleware Architecture



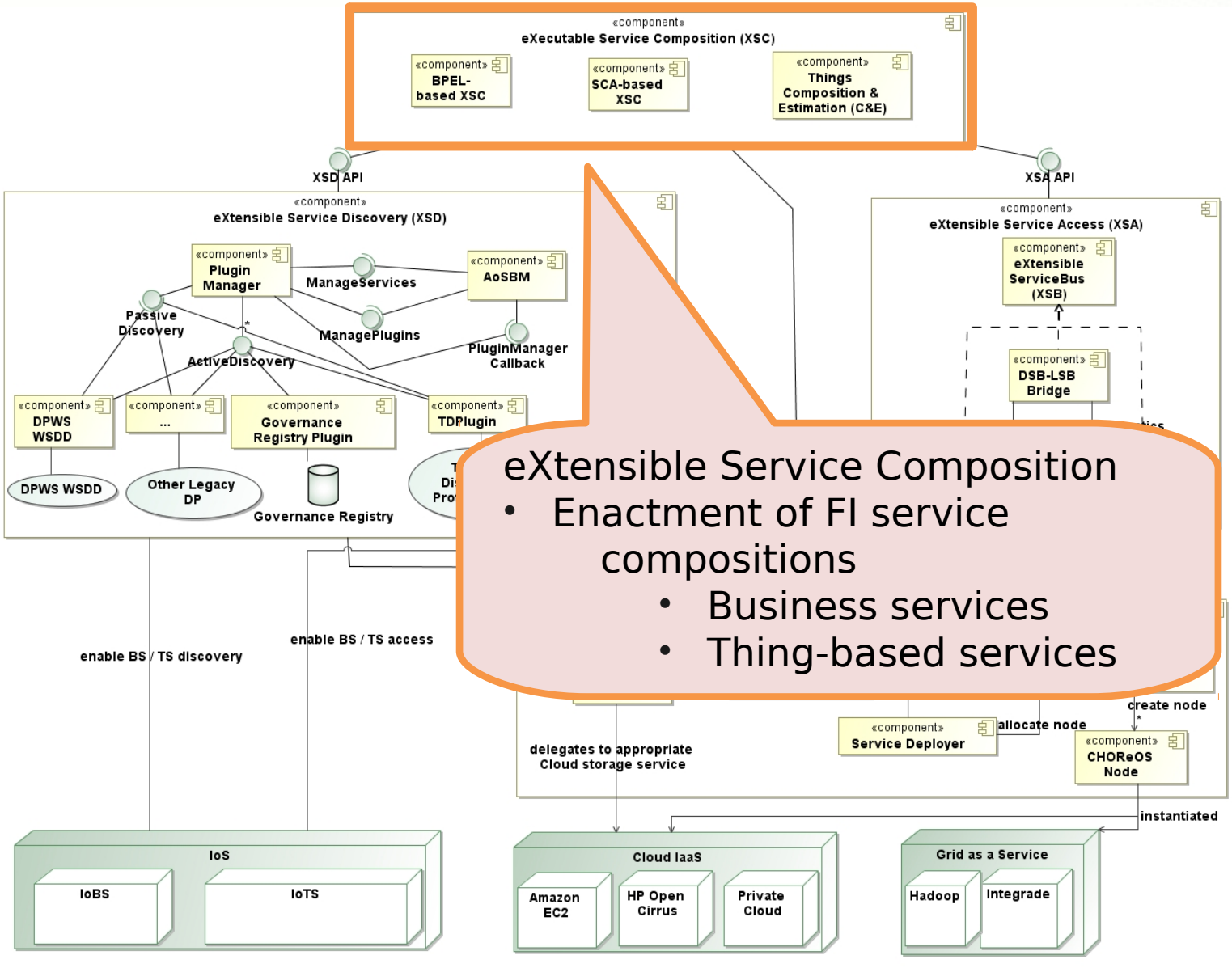
Middleware Architecture



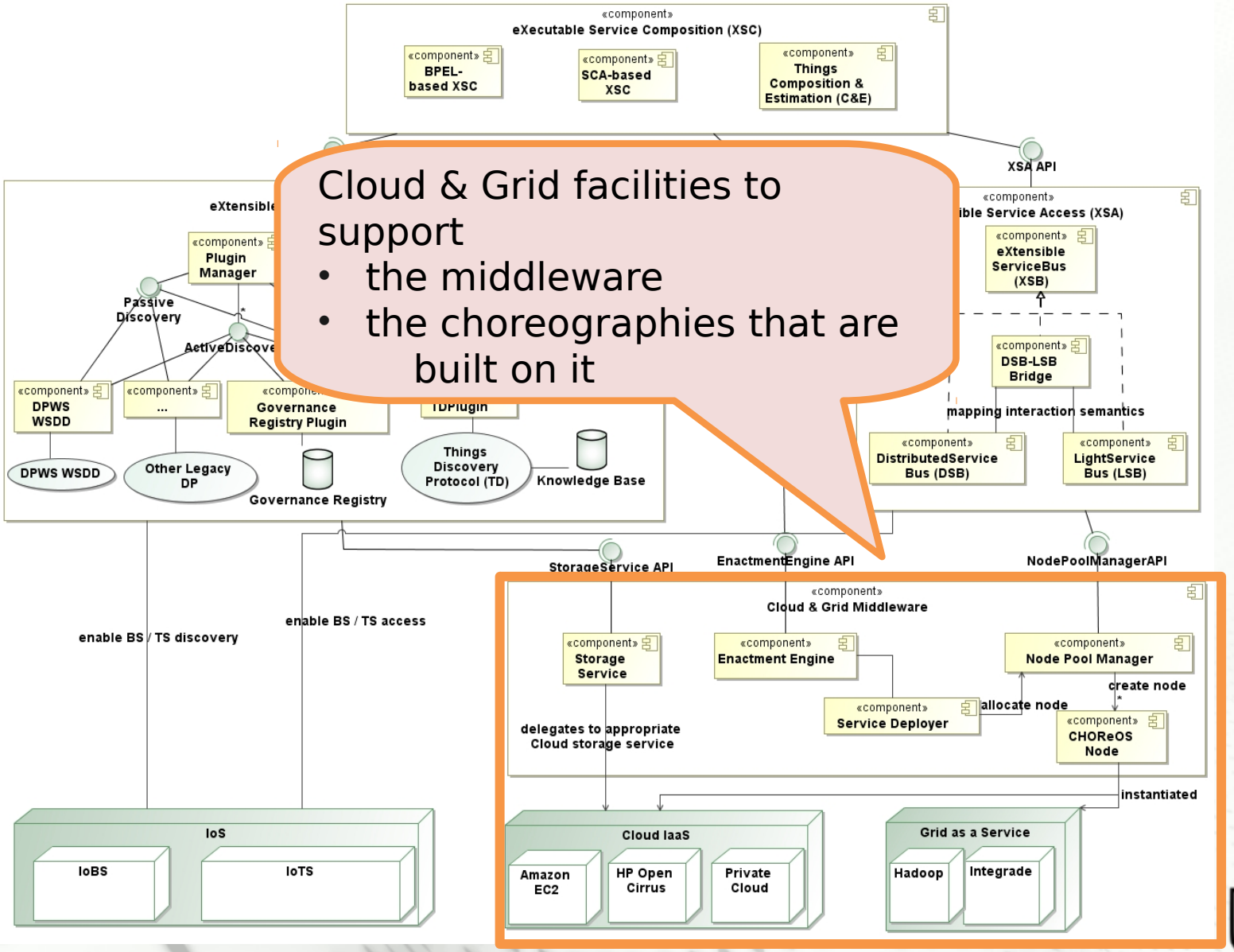
eXtensible Service Access

- Based on an eXtensible Service Bus (XSB)
- with two concrete implementations:
  - DSB and LSB.

Middleware Architecture



Middleware Architecture



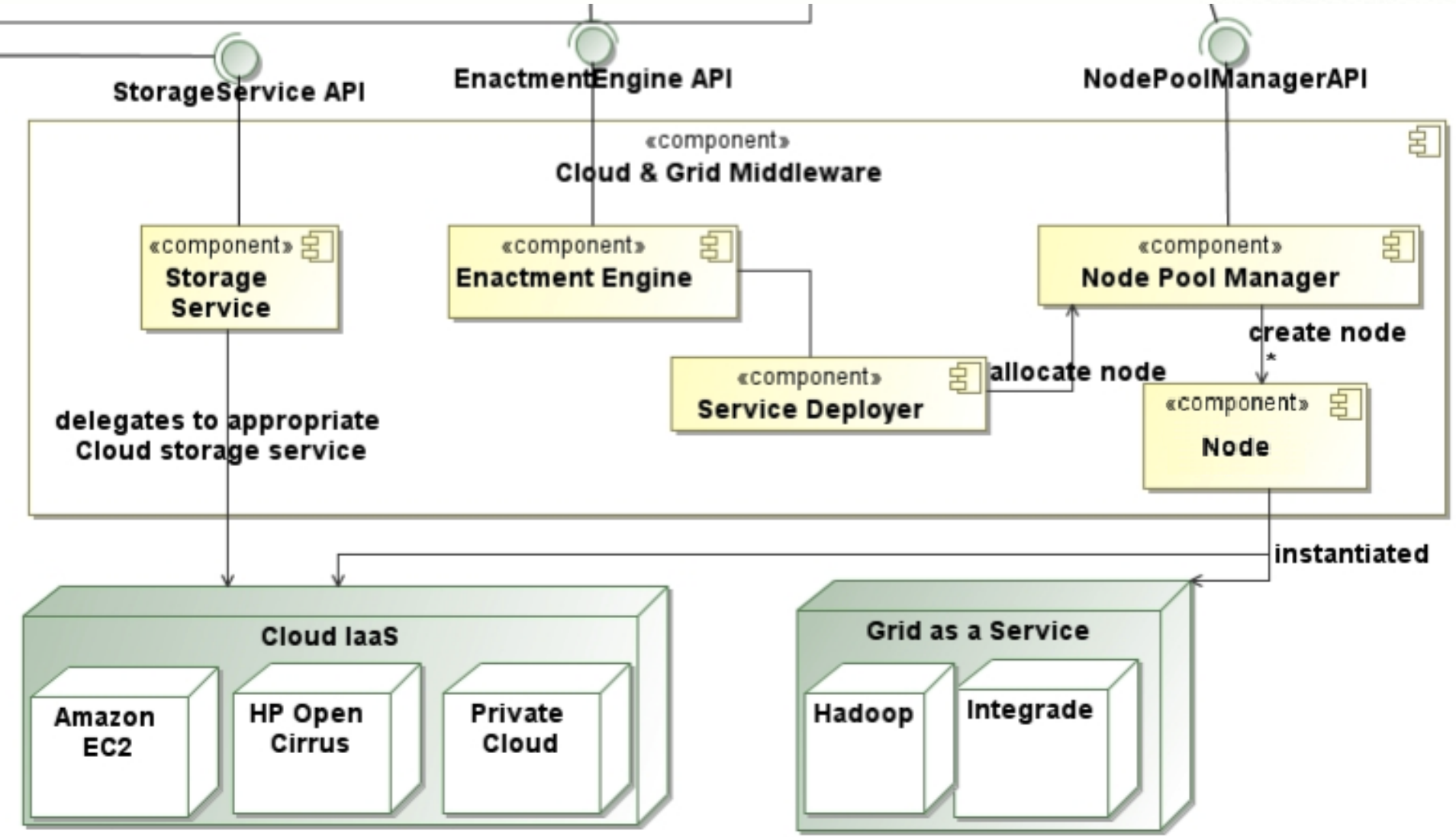
Cloud & Grid Middleware

- **Cloud: to execute large quantities of services, replicate services, and balance the load from millions of requests from thousands of users**
- **Grid: for CPU-intensive or data-intensive applications**

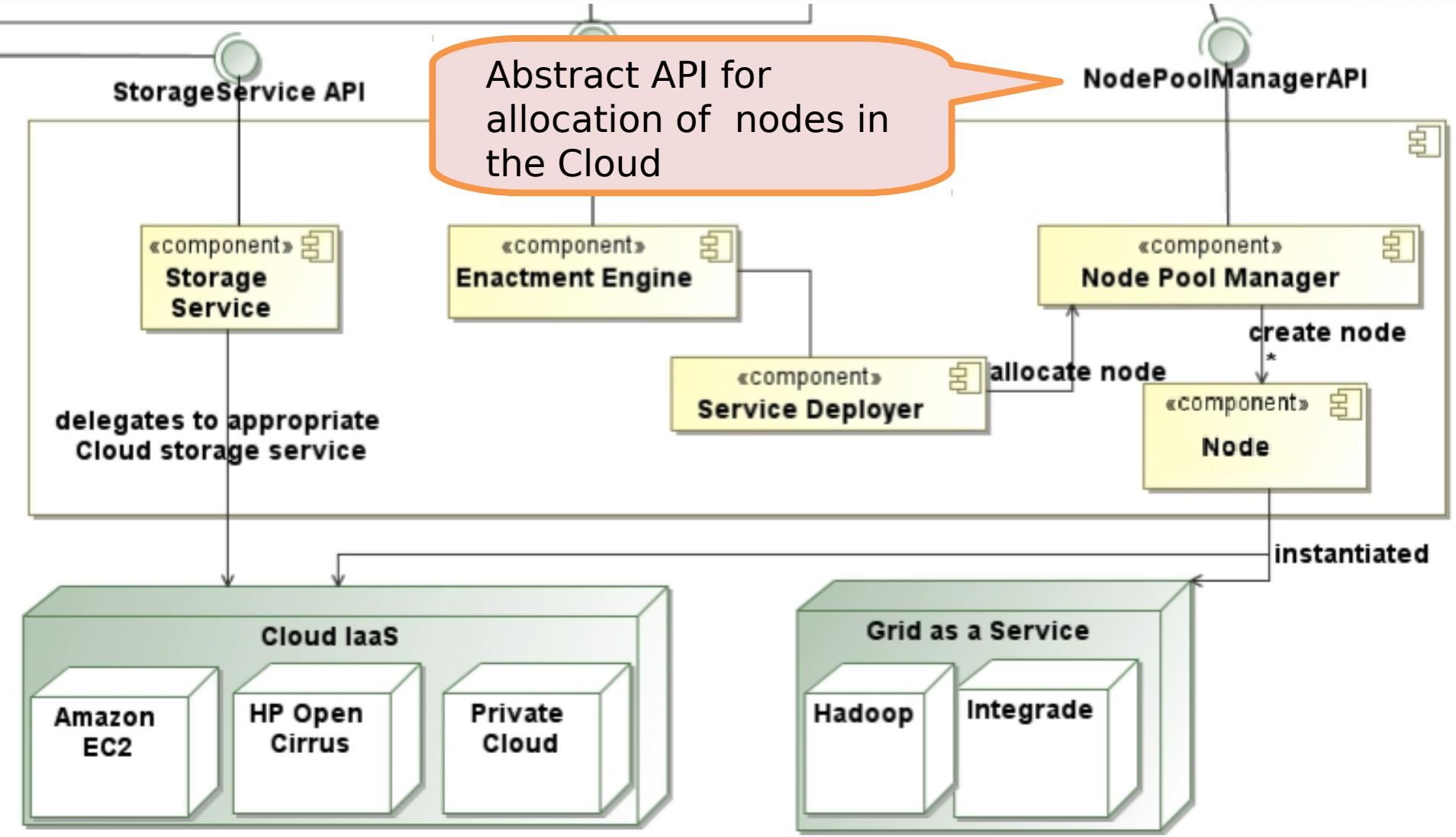
Two-way contribution:

1. **Cloud will provide a means to support scalable choreographies**
2. **Higher-level abstraction for the execution of complex, distributed, service compositions**

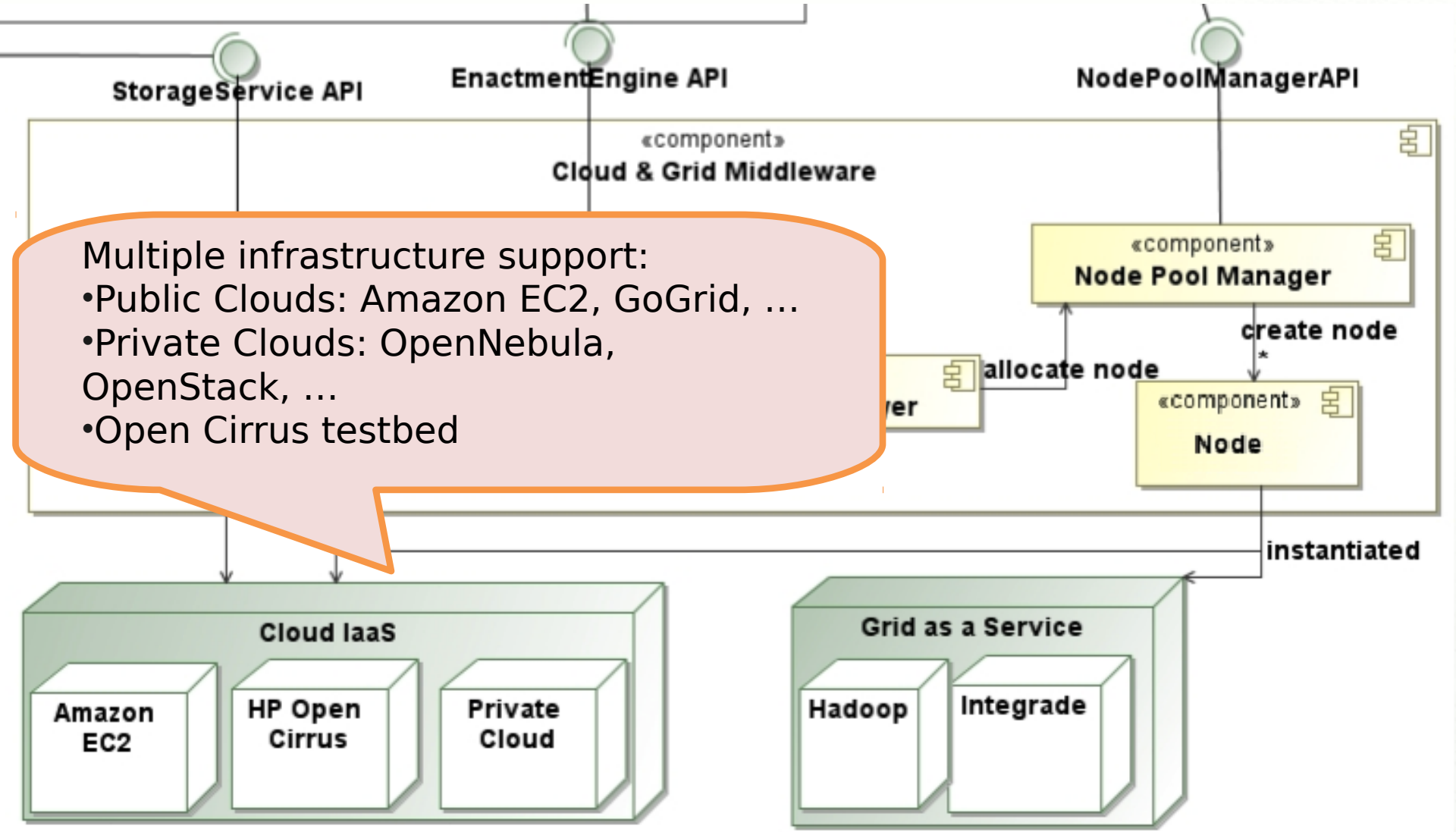
Cloud & Grid Middleware



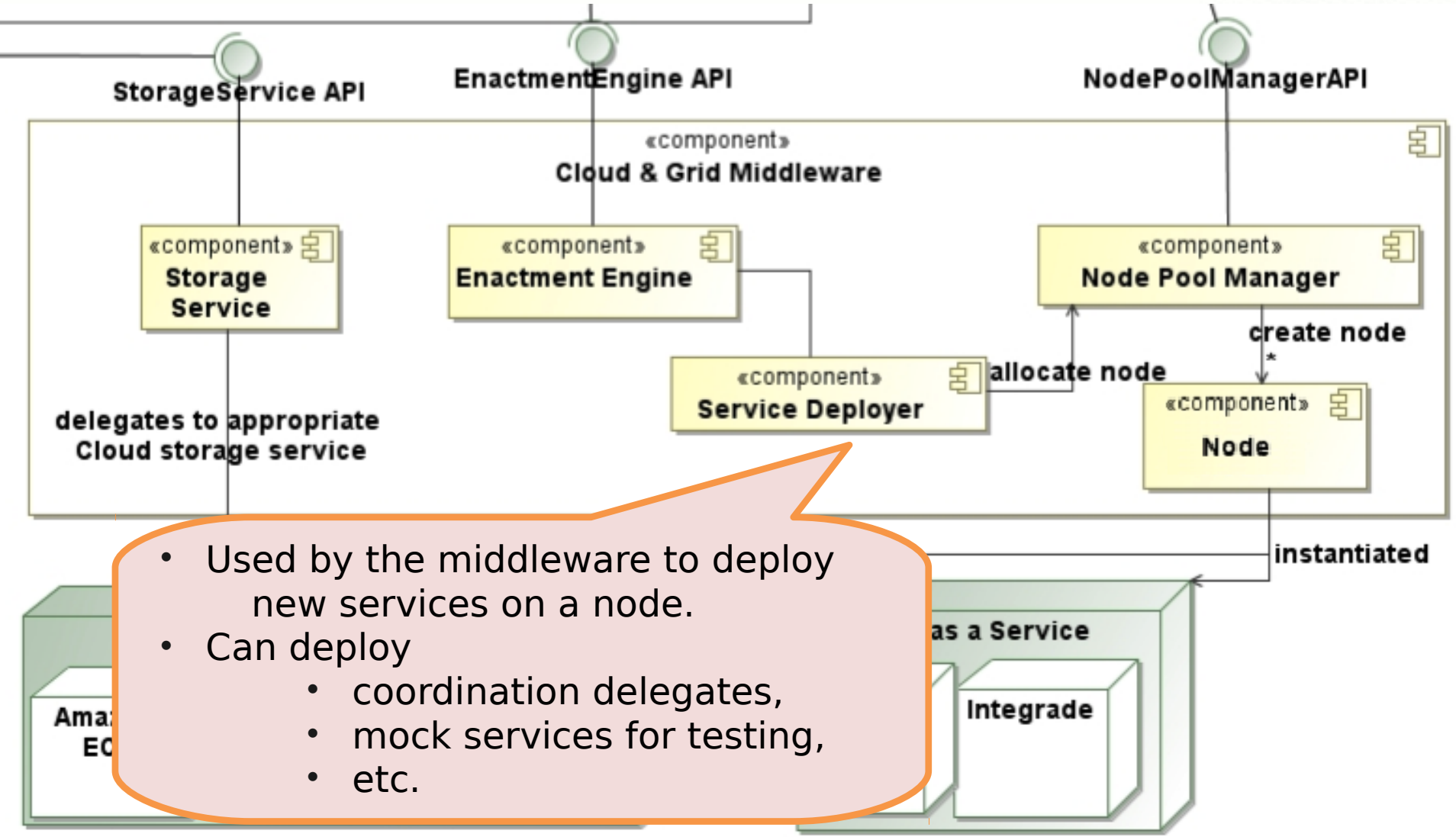
Cloud & Grid Middleware



Cloud & Grid Middleware

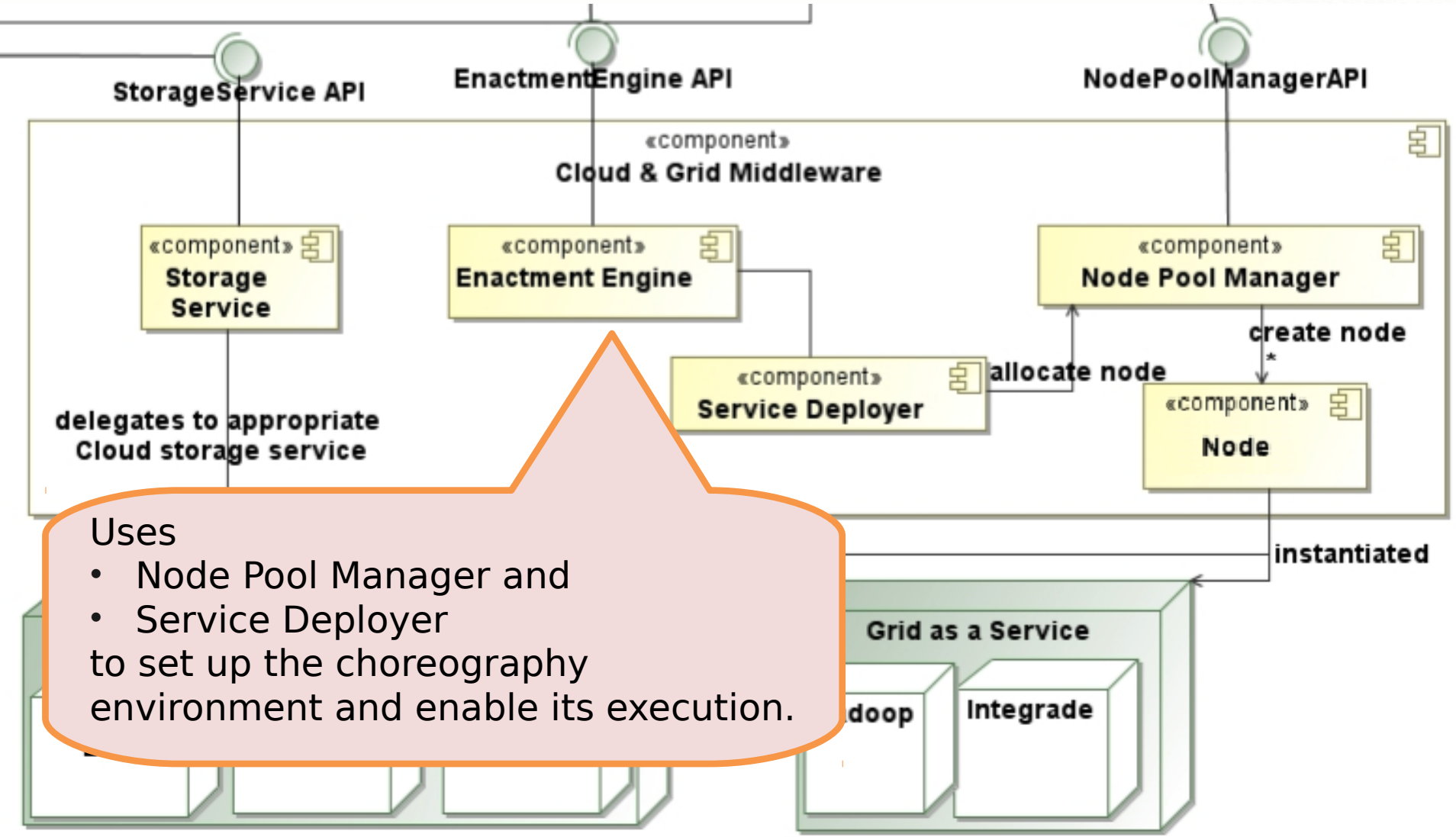


Cloud & Grid Middleware



- Used by the middleware to deploy new services on a node.
- Can deploy
  - coordination delegates,
  - mock services for testing,
  - etc.

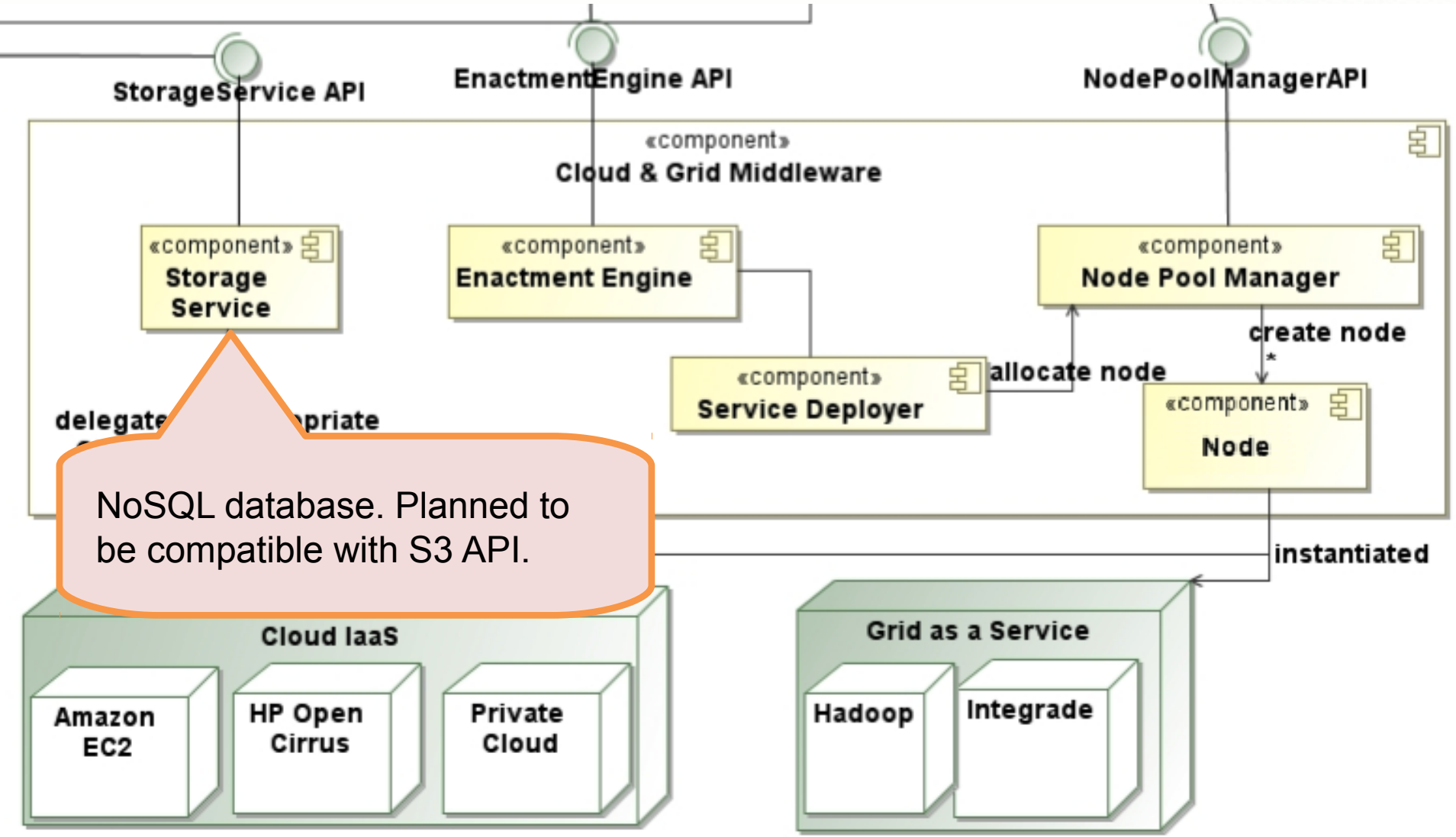
Cloud & Grid Middleware



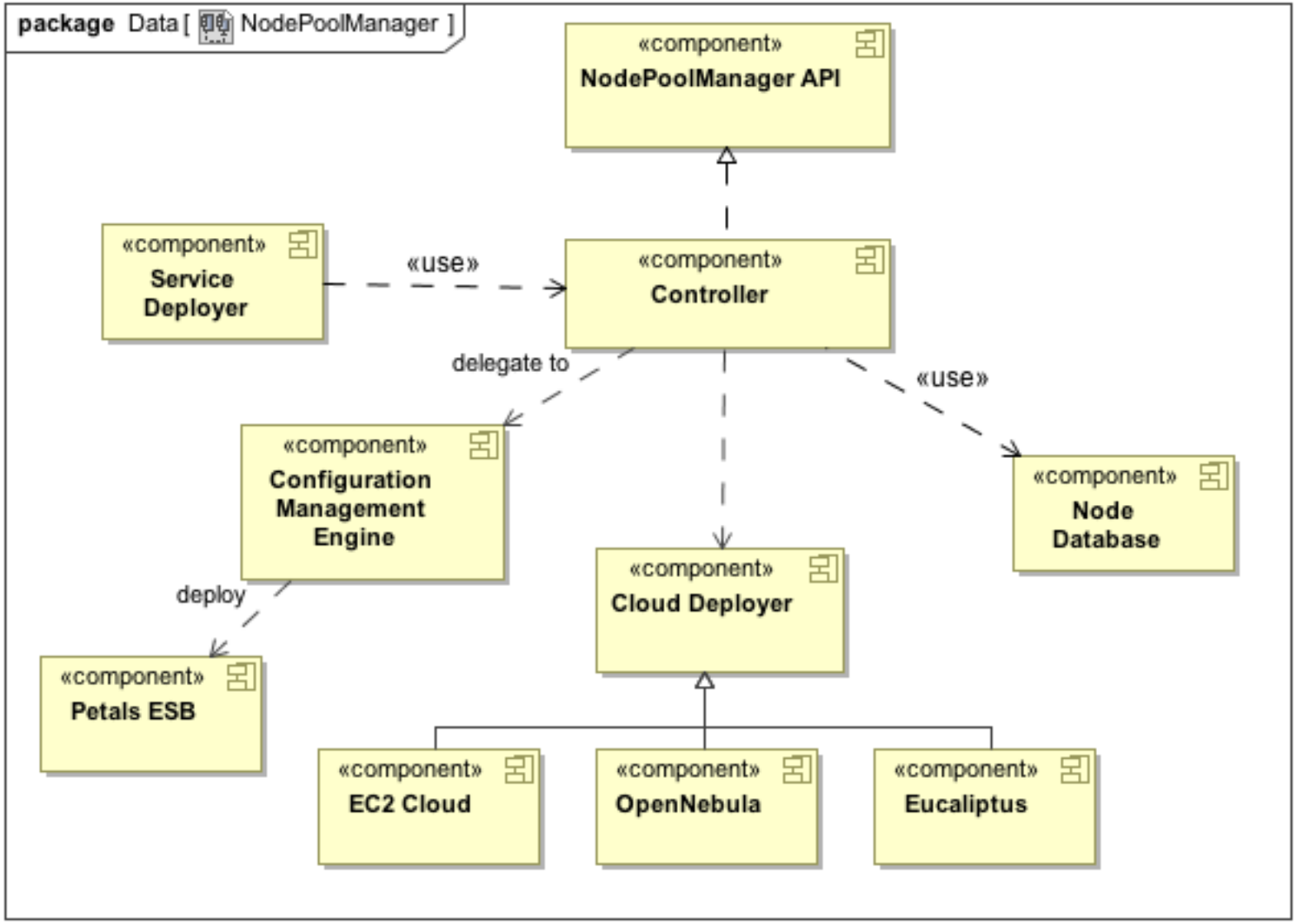
Uses
- Node Pool Manager and
- Service Deployer

to set up the choreography environment and enable its execution.

Cloud & Grid Middleware



Node Pool Manager



Implementation Status

- Node Pool Manager implementation for Amazon EC2
- Automatic deployment of DSB on Amazon and Open Cirrus

Node Pool Manager Demo

(2) Verification & validation of choreographies

A Testing Framework

- **A framework and tool for automated testing of choreographies.**
- **With this, we intend to:**
  - abstract the choreography and its elements to simplify coding of automated tests;
  - implement a conversation monitoring service to support integration tests;
  - mock web services functionalities;
  - Invoke SOAP and REST services dynamically
  - Support scalability testing of web services and choreographies

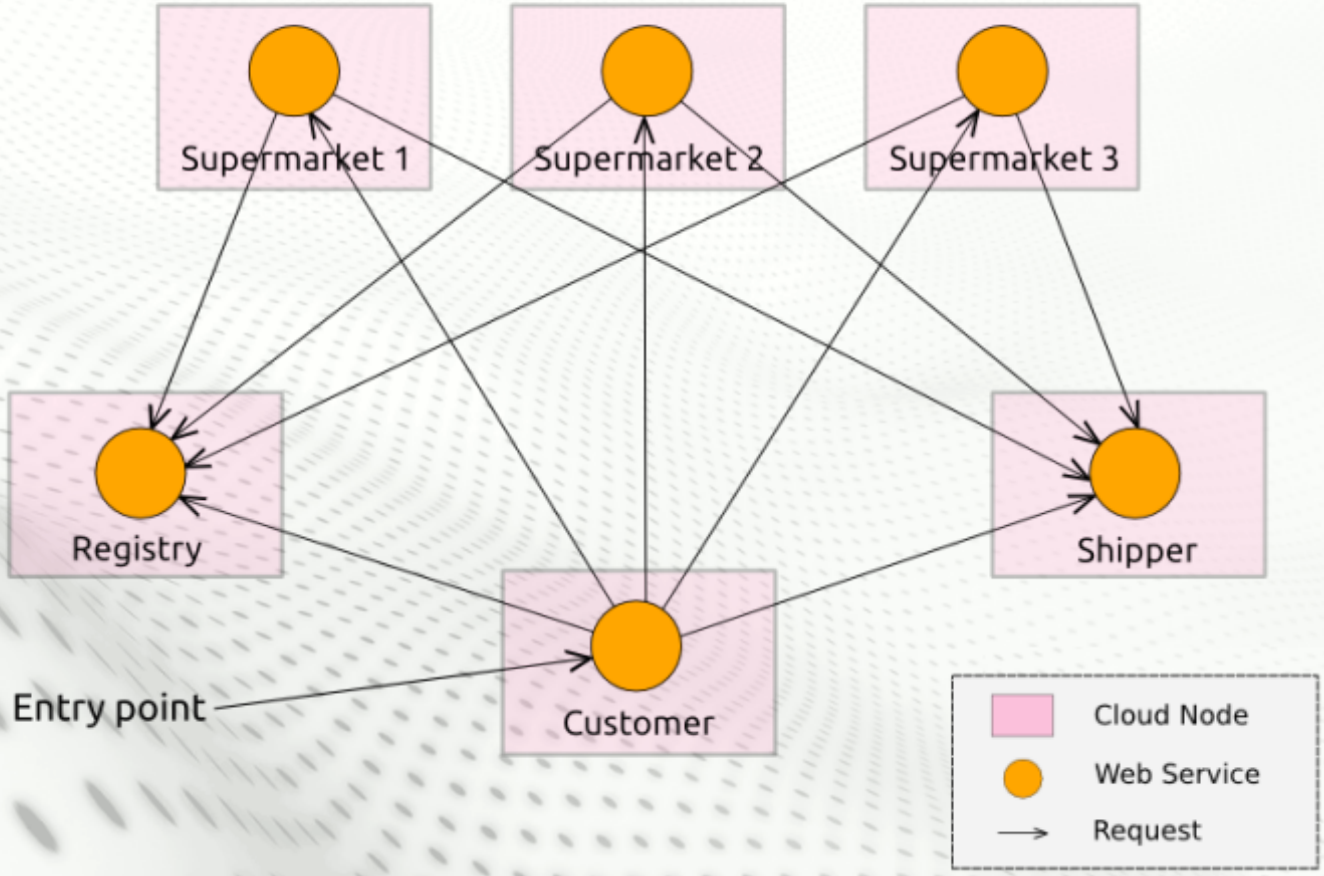
Sample Choreography

The example choreography implements a distributed shopping service:

1. A customer provides to the choreography a shopping list;
2. The price of each item list is queried in different supermarket services to find which one has the lowest price;
3. The choreography returns to the customer the total cheapest price of its list and provides features for purchasing and delivering the items.

Sample Choreography

The choreography is deployed on 6 nodes of Open Cirrus.



Sample Choreography

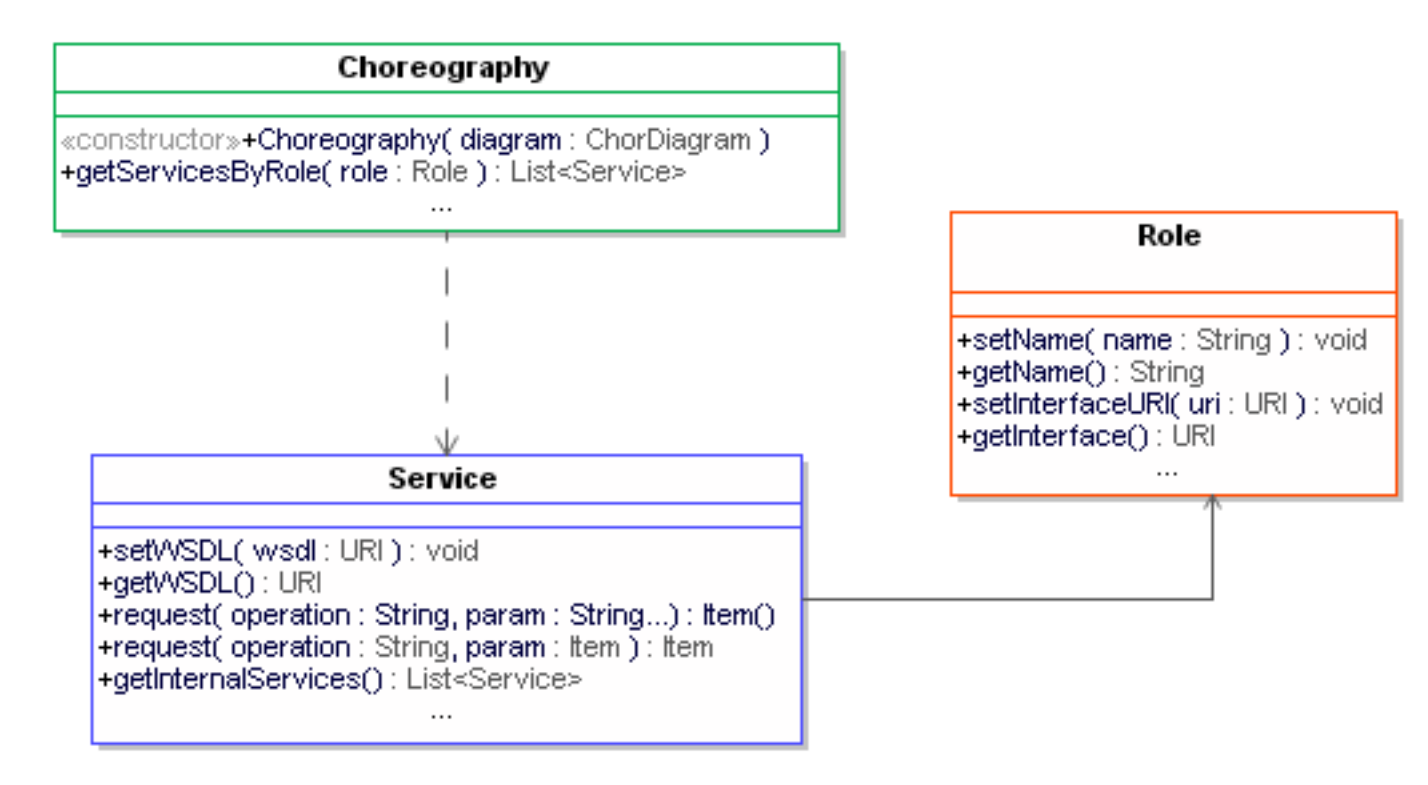
Assessment

1. "Manually" enacted a choreography (with scripts).
2. With the help of *Rehearsal* TDD framework, we perform 5 tests to verify if the choreography is running correctly (1 compliance and 4 acceptance tests).
3. In the future, we will also be able to perform scalability tests to verify if the choreography continues to work correctly, meeting QoS requirements, as its scale grows.

Sample Choreography

Abstraction of Choreography

Interact with the choreography elements (roles, services, messages) through Java objects.



Sample Choreography

Test set up

```
// Choreography creation
Choreography futureMarket  = new Choreography();

// role creation: new Role(“<role name>”, “<role wsdl”>”);
Role supermarket = new Role("supermarket", "supermarket?wsdl");

// service creation
Service store1 = new Service();
store1.setWSDL("store1?wsdl");
store1.addRole(supermarket);

// adding the services and roles into the choreography
futureMarket.addRole(supermarket);
futureMarket.addService(store1);
```

Sample Choreography

Service Role Compliance Testing

Apply a generic test suite to verify if the participants are playing a role properly.

SMRoleTest.class => JUnit test cases that must be applied in all participants that want to play the supermarket Role.

Compliance test => through the **assertRole** feature, the tests are applied:

```
@Test
public void servicesMustBeCompliantWithTheSupermarketRole() {
    List<Service> supermarkets = futureMarket
                                        .getServicesForRole("supermarket");

    for (Service service : supermarkets)
          assertRole(supermarket, service,  SMRoleTest.class);
    }
```

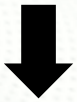
Sample Choreography

Acceptance Testing

Test cases for assessing the choreography workflow from the end-user perspectives:

Register supermarket

receives a supermarket endpoint and register it into the registry service



Get price of product list

receives a customer product list and returns the cheapest total price and an order id for purchasing

Purchase

receives a customer order id and returns the shipper name

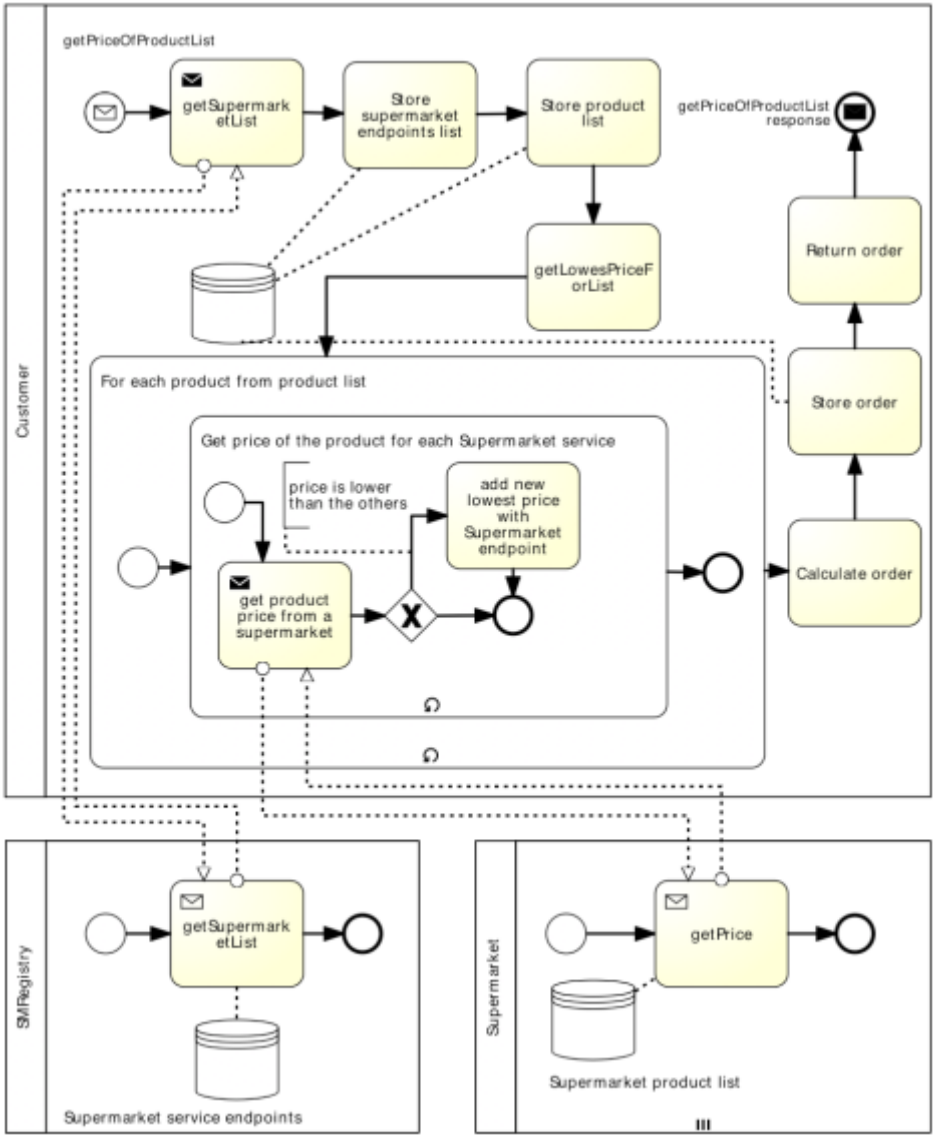
Get delivery data

receives a customer order id and a shipper name and returns the delivery data information (day and time)

Sample Choreograpy

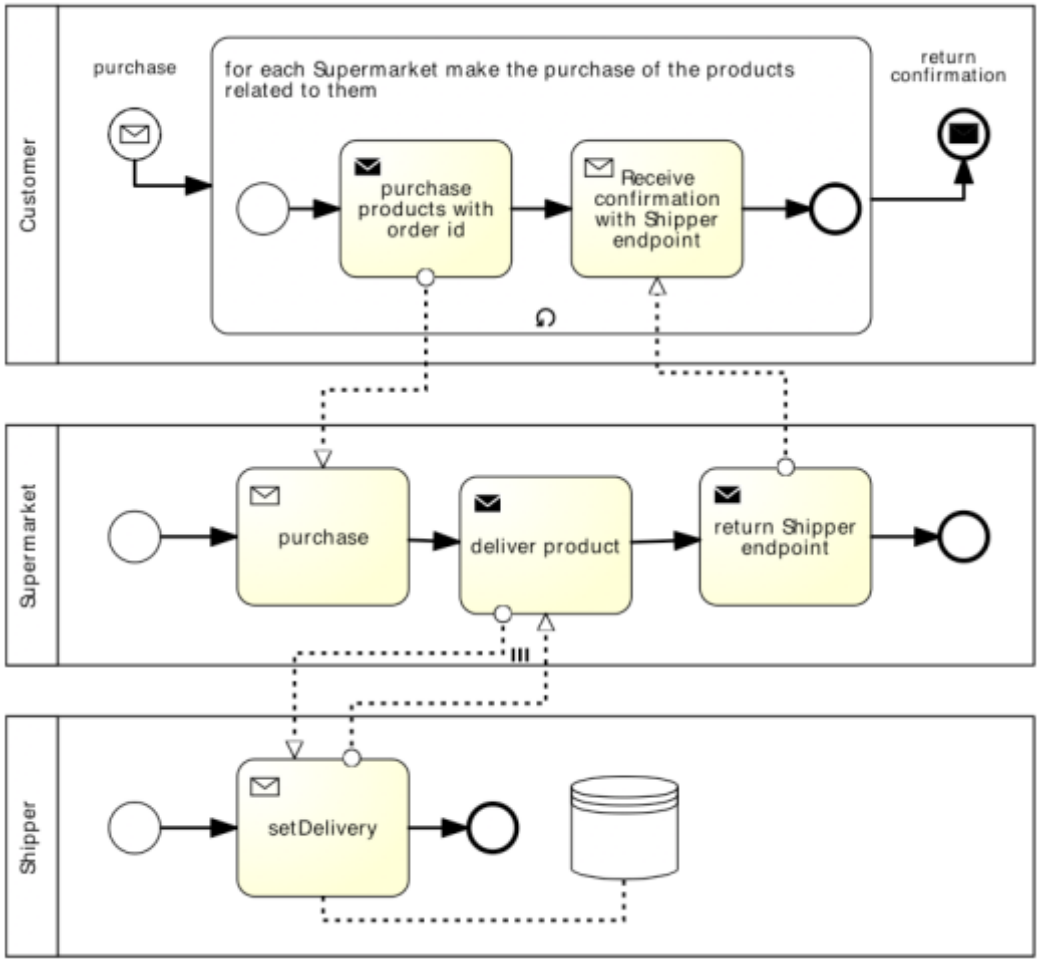
Choreography Testing Demo

Sample Choreography



BPMN2 Model describing the *getPriceOfProducList* operation of the Future Market Choreography

Sample Choreography



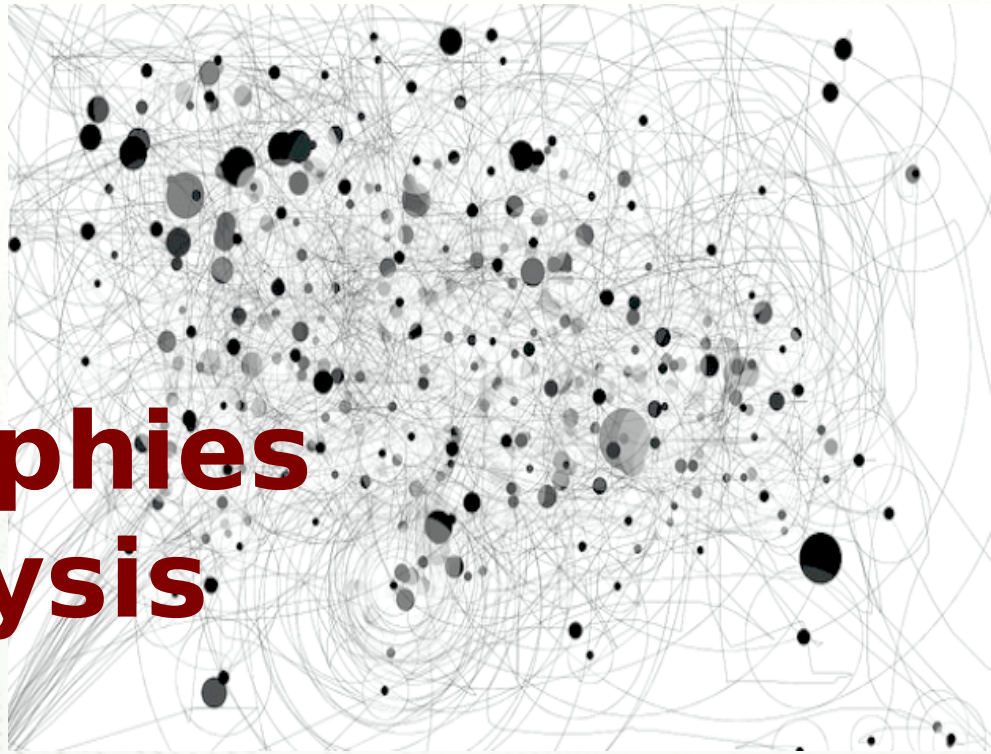
BPMN2 Model describing the *purchase* operation of the Future Market Choreography

Conclusion and Future

- Using the Node Pool Manager, cloud nodes can be manipulated and the DSB can be deployed easily

Next Steps

- Use a different cloud provider
- Development of the Service Deployer and Enactment Engine
- Scalability testing
- Choreography QoS/SLA management



(3) Choreographies topology analysis

Node analysis

- **Services choreography should be designed taking into account the stability and interdependencies of its parts**
- **Some metrics (mainly from SNA – Social Network Analysis)**
  - Degree centrality (number of links incident upon a node)

$$C_{D-}(v) = \frac{deg^{-}(v)}{n-1} \quad C_{D+}(v) = \frac{deg^{+}(v)}{n-1}$$

$$C_D(G) = \frac{\sum_{i=1}^{|V|}[C_D(v^*) - C_D(v_i)]}{(n-1)(n-2)}$$

  - Eigenvector centrality (neighbors and their neighbors)

Algorithm 1: Accelerated power method for eigenvector centrality calculation

Input : An adjacency matrix $A_{i,j}$, where $A_{i,j} = 1$ if the i^{th} node is adjacent to the j^{th} node, and $Ai, j = 0$ otherwise

Output: Eigenvector centrality value for all graph nodes

1 Set $C_E(v_i) = 1$ for all i;
2 Compute $C_E^*(v_i) = \sum_j A_{i,j} * C_E(v_j)$;
3 Set λ equal to the square root of the sum of squares of each $C_E^*(v_i)$;
4 Set $C_E(v_i) = C_E^*(v_i)/\lambda$ for all i;
5 Repeat lines 2 to 4 until λ stops changing;

  - Page Rank

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

Node analysis

- **Betweenness centrality**
  - The chance of a node to be in a shortest paths between other 2 vertices $C_B(v) = (\sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}})$
- **Closeness Centrality**
  - The mean shortest path between v and all other reachable vertices

$$C_C(v) = \frac{|J_v|/(n-1)}{\sum_{t \in J_v} d_G(v,t)/|J_v|}$$

Choreography analysis

- **Overall Stability**
  - Average propagation level of changes performed in the nodes of a network

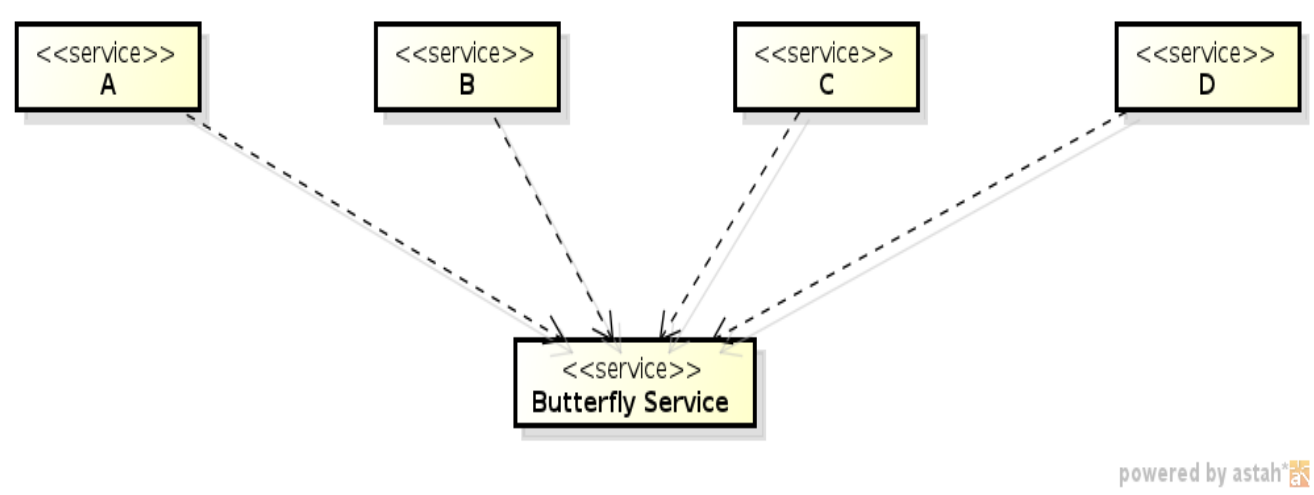
$$AvgImpact(G) = [\sum_{v \in V} \gamma(v)]/n^2$$

$$Stability(G) = 1 - AvgImpact(G)$$

Given a set of services that realize the choreography, choose the ones that maximize choreography overall stability

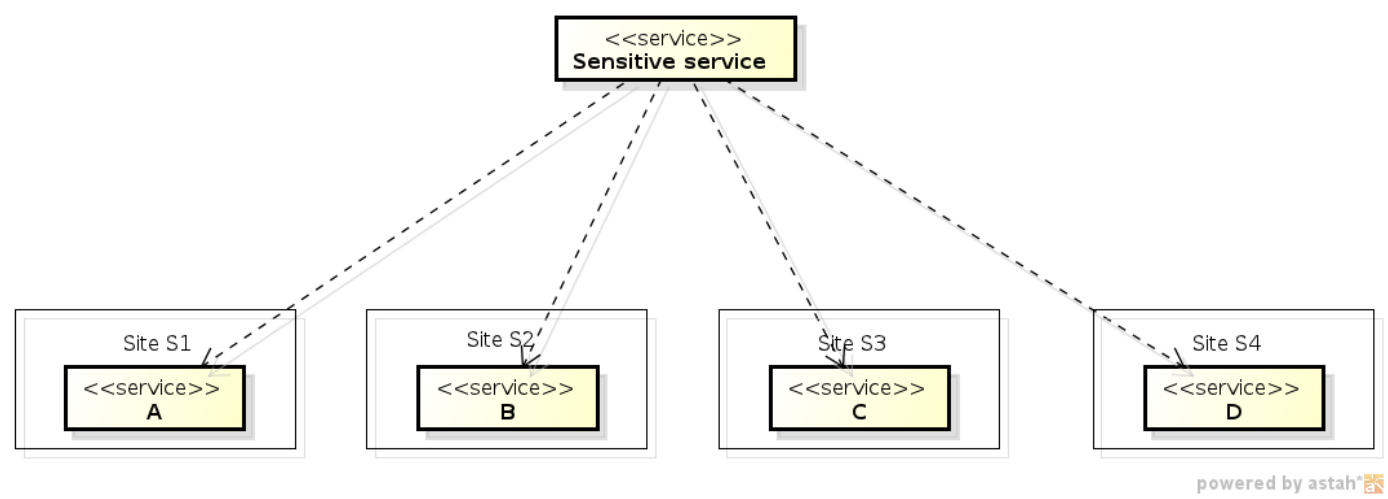
Impact analysis

- **Butterfly services**
  - Likely to cause large ripple-effects
  - Core (prominent) services of the choreography
    - Identify, monitor, backup
    - Unit-test and integration tests are advisable



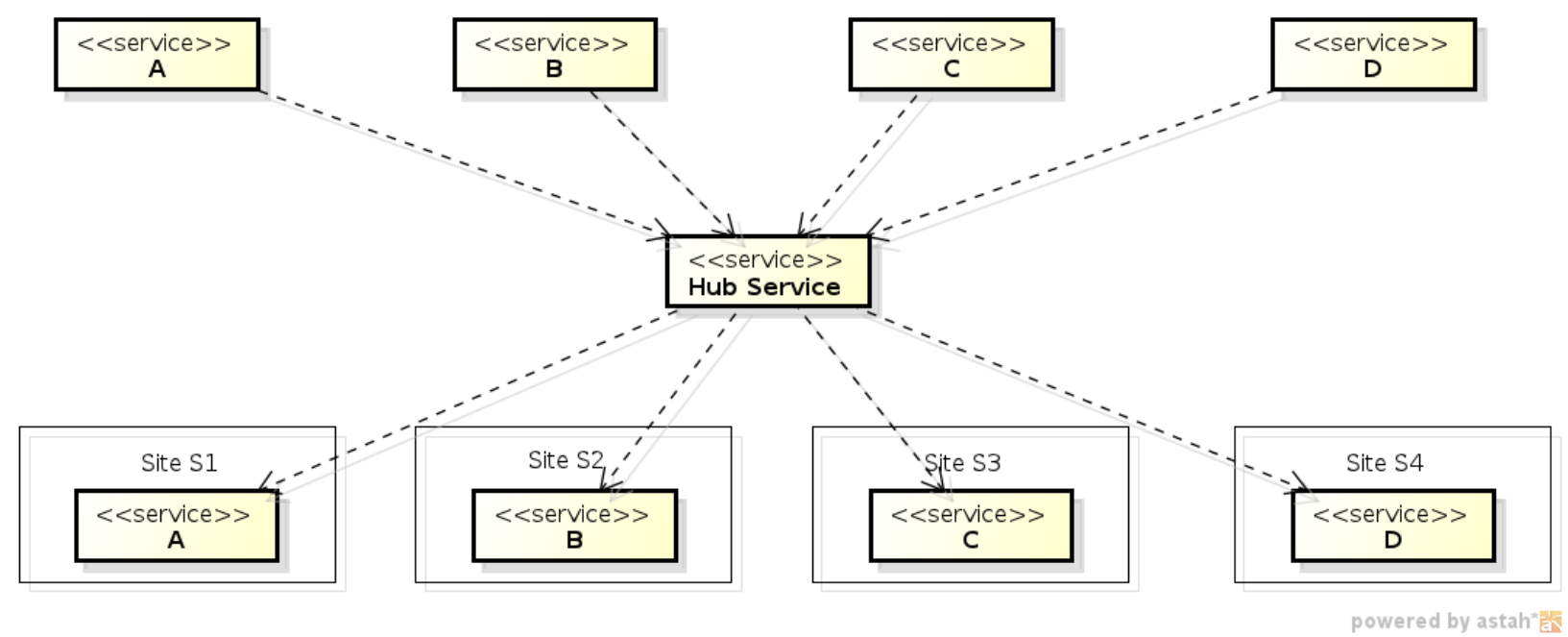
Impact analysis

- **Sensitive services**
  - Frequently suffer from ripple-effect
  - Limit the occurrences of sensitive services during (re)synthesis to improve choreography maintainability and evolvability



Impact analysis

- **Hub services (breakable + butterfly)**
  - Change propagator – Amplifies changes in a choreography



When choosing between equivalent services, do a dependency analysis and point/exclude possible sensitive, butterfly and hub services

Thanks!

Baile Project

IME/USP

- **Visit us:**
  - http://ccsl.ime.usp.br/baile
- **Write us:**
  - Marco Aurélio Gerosa (gerosa@ime.usp.br)
  - Carlos Eduardo Moreira dos Santos (cadu@ime.usp.br)

Some references

- Josuttis, N.: SOA in Practice. O'Reilly Media, Inc. (2007)
- Kokash, N., D'Andrea, V.: Service oriented computing and coordination models. In: Proceedings of Challenges in Collaborative Engineering Workshop, Sopron, Hungary, pp. 95–103. Citeseer (2005)
- Valérie Issarny et al., "Service-oriented middleware for the Future Internet: state of the art and research directions," Journal of Internet Services and Applications 2 (May 25, 2011): 23-45.