

Aligning Service-Oriented Architectures with Security Requirements

Mattia Salnitri¹, Fabiano Dalpiaz², Paolo Giorgini¹

¹University of Trento, Italy

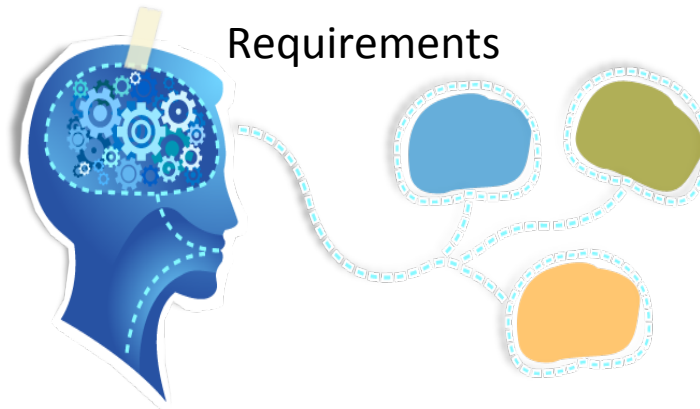
²University of Toronto, Canada



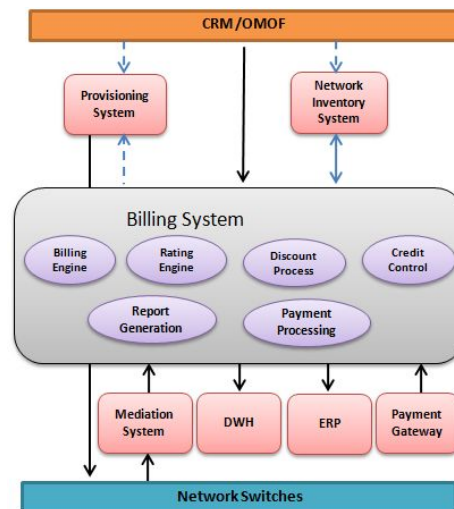
Outline

- Change
 - Requirements
 - Service-oriented Architectures
- Modelling SoA and Security Requirements
- Our proposal
- Example
- Future works and conclusions

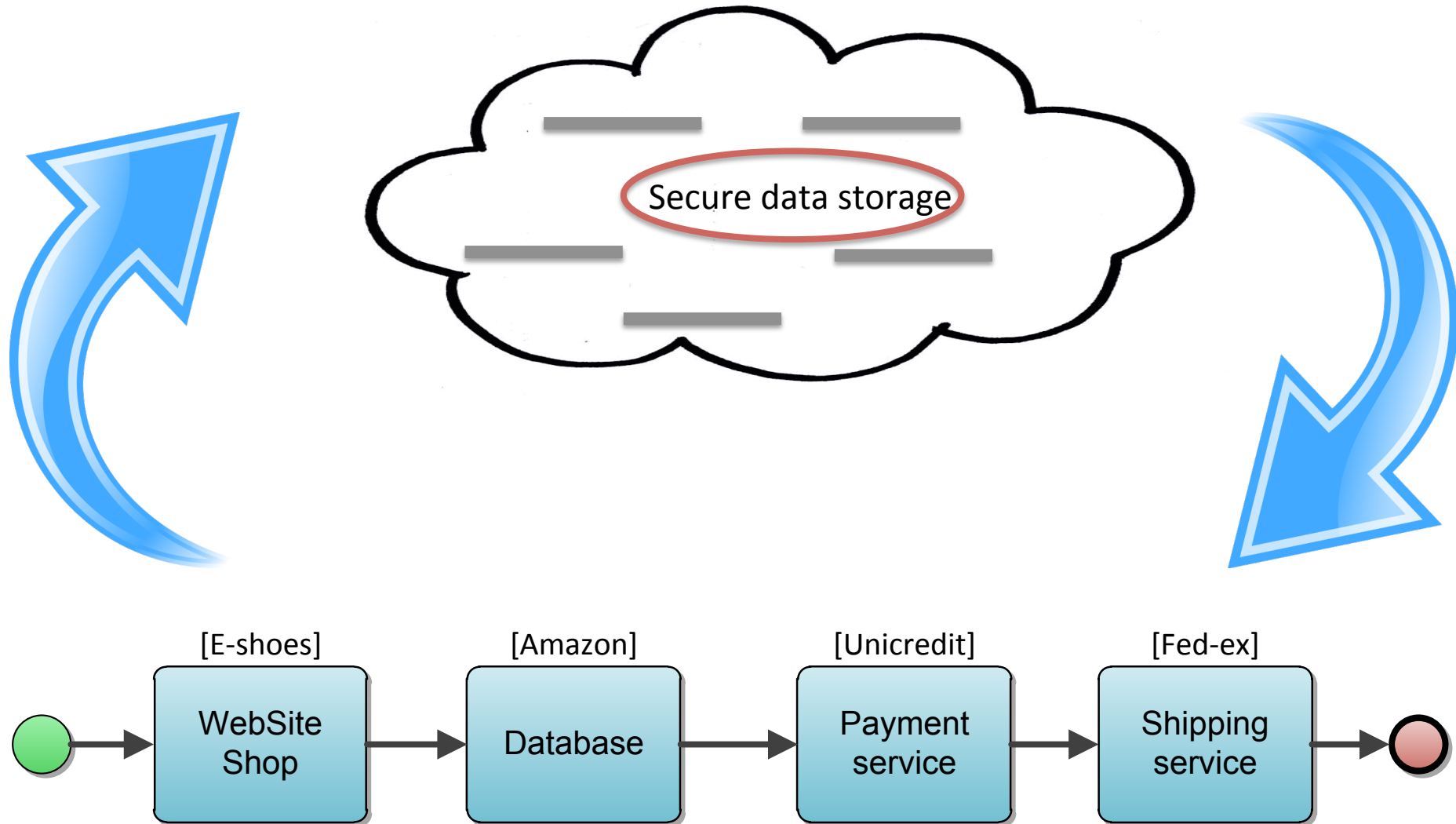
Change



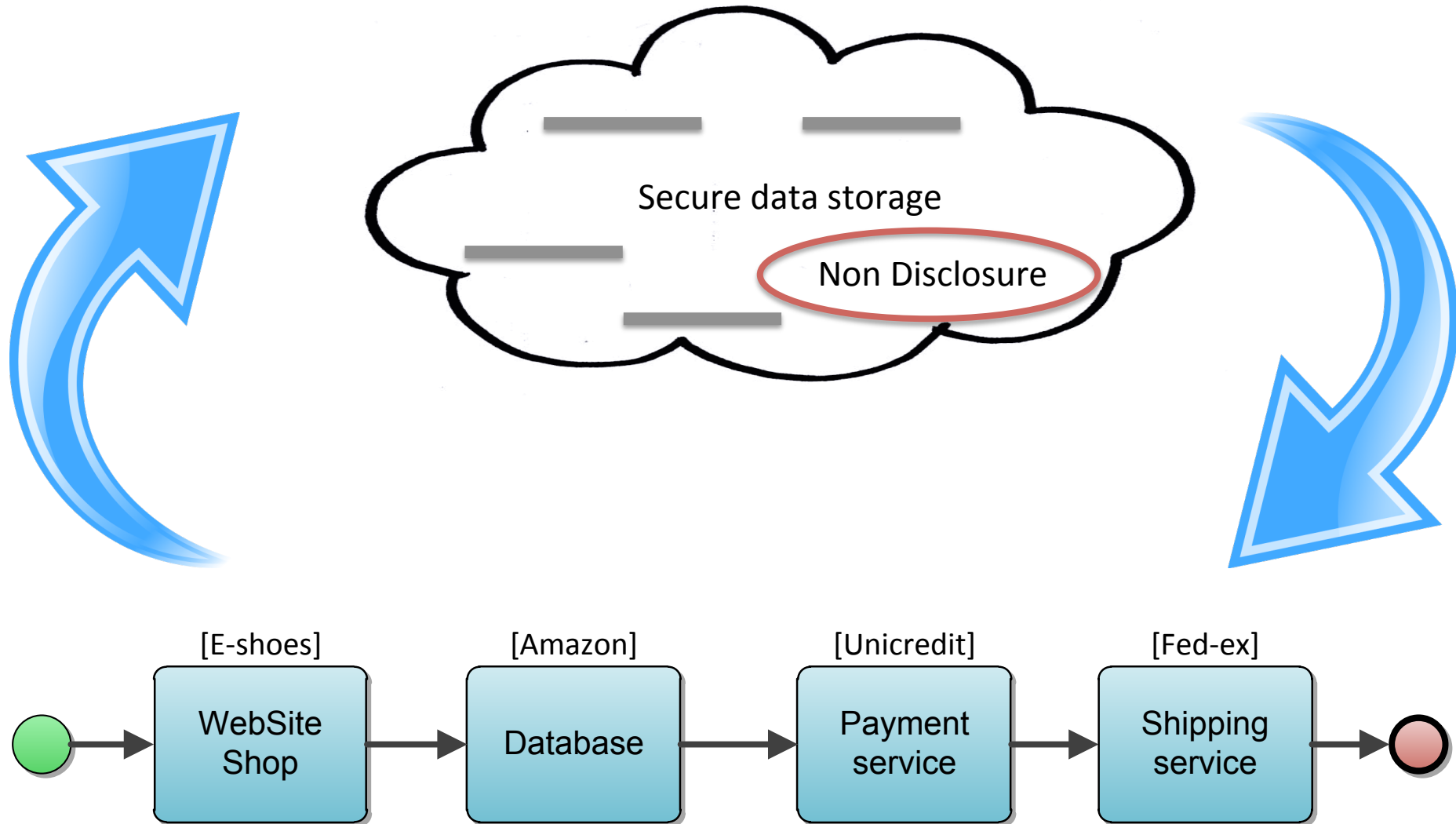
Software



Motivating Example



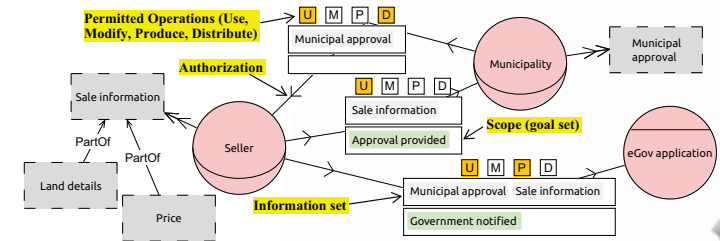
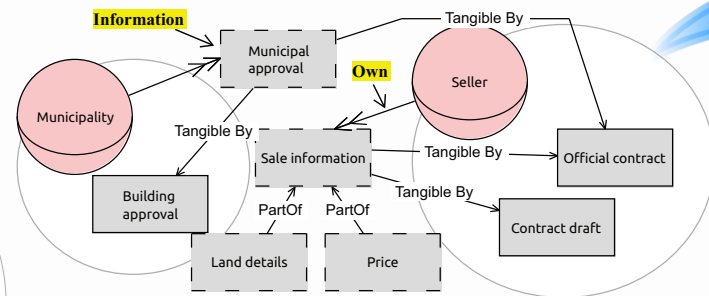
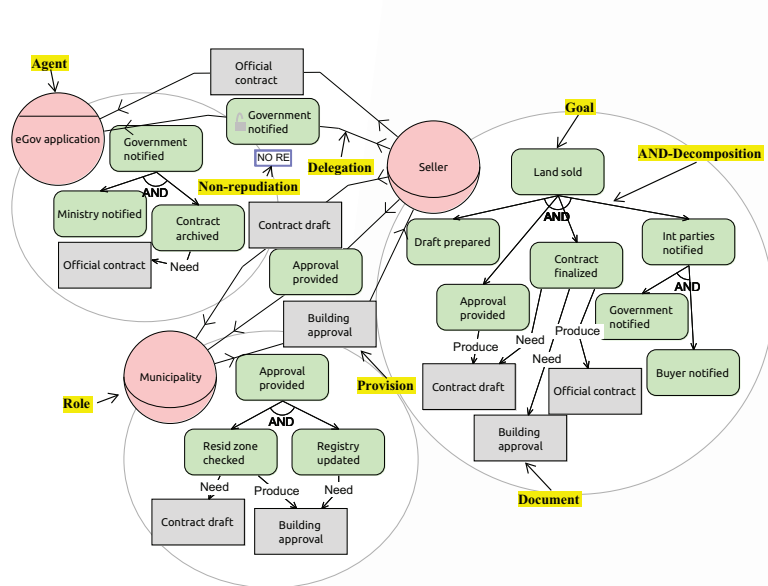
Motivating Example



Context: the Aniketos project

- Secure **service compositions** in a constantly changing environment
 - Design-time:
 - specification of the security requirements
 - Run-time:
 - creation of a secure service composition
 - alignment of requirements with evolved service compositions
- For more information : <http://www.aniketos.eu/>

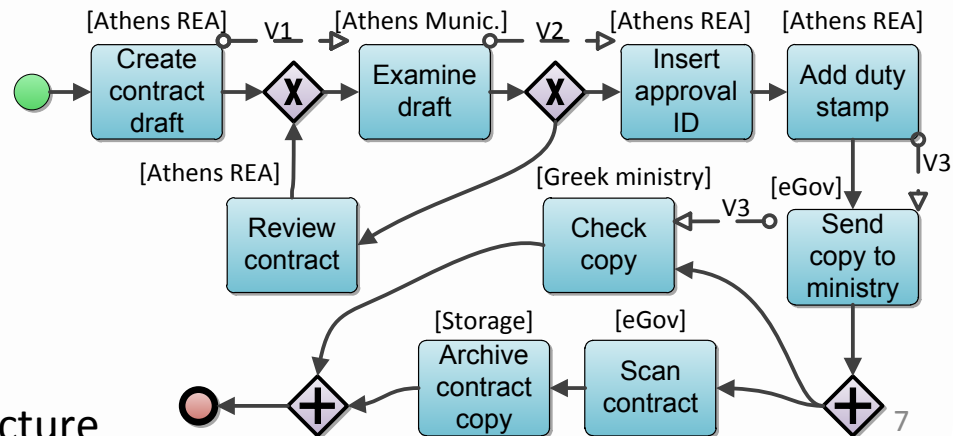
Modelling language overview



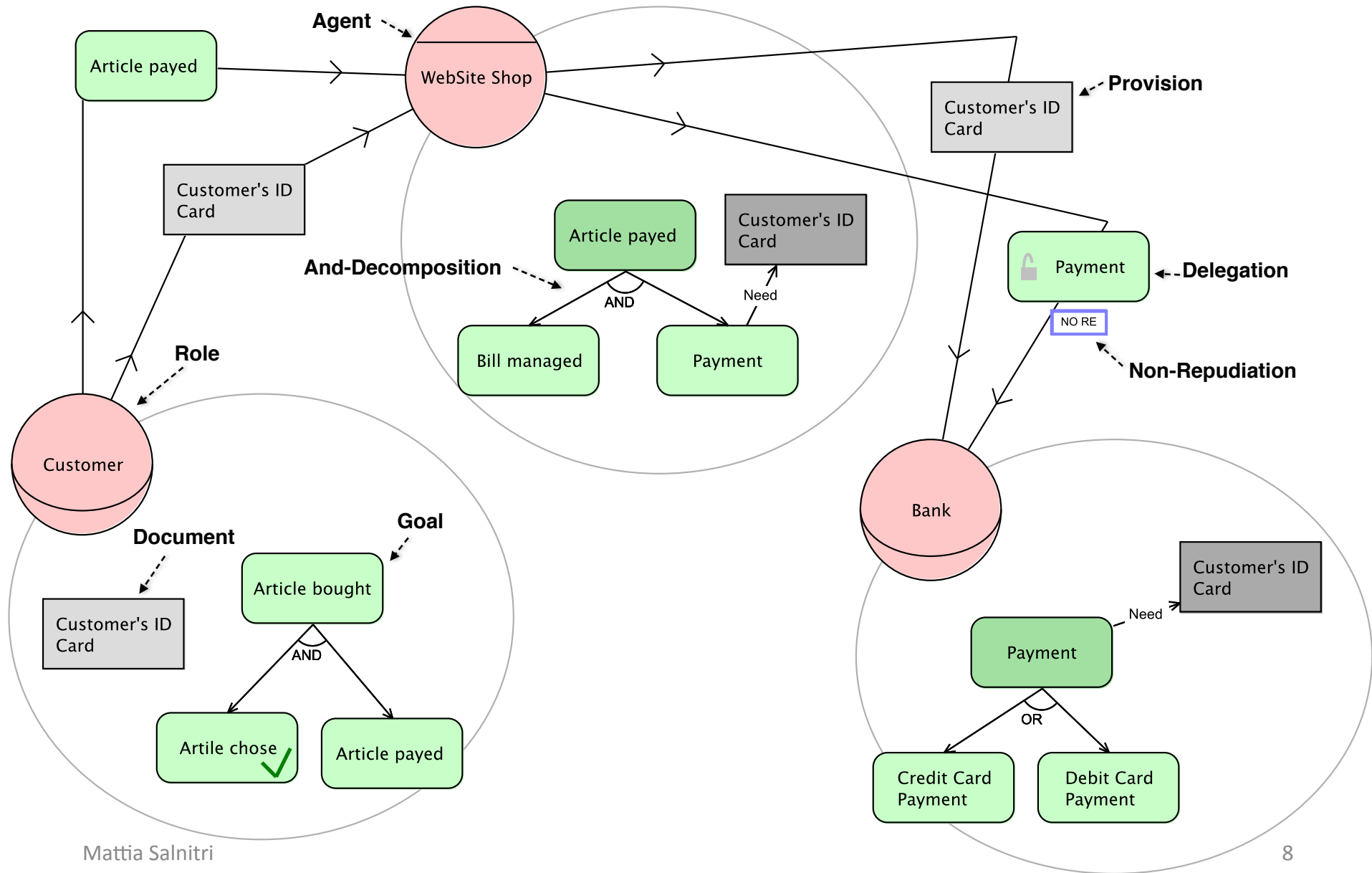
Requirements

Mattia Salnitri

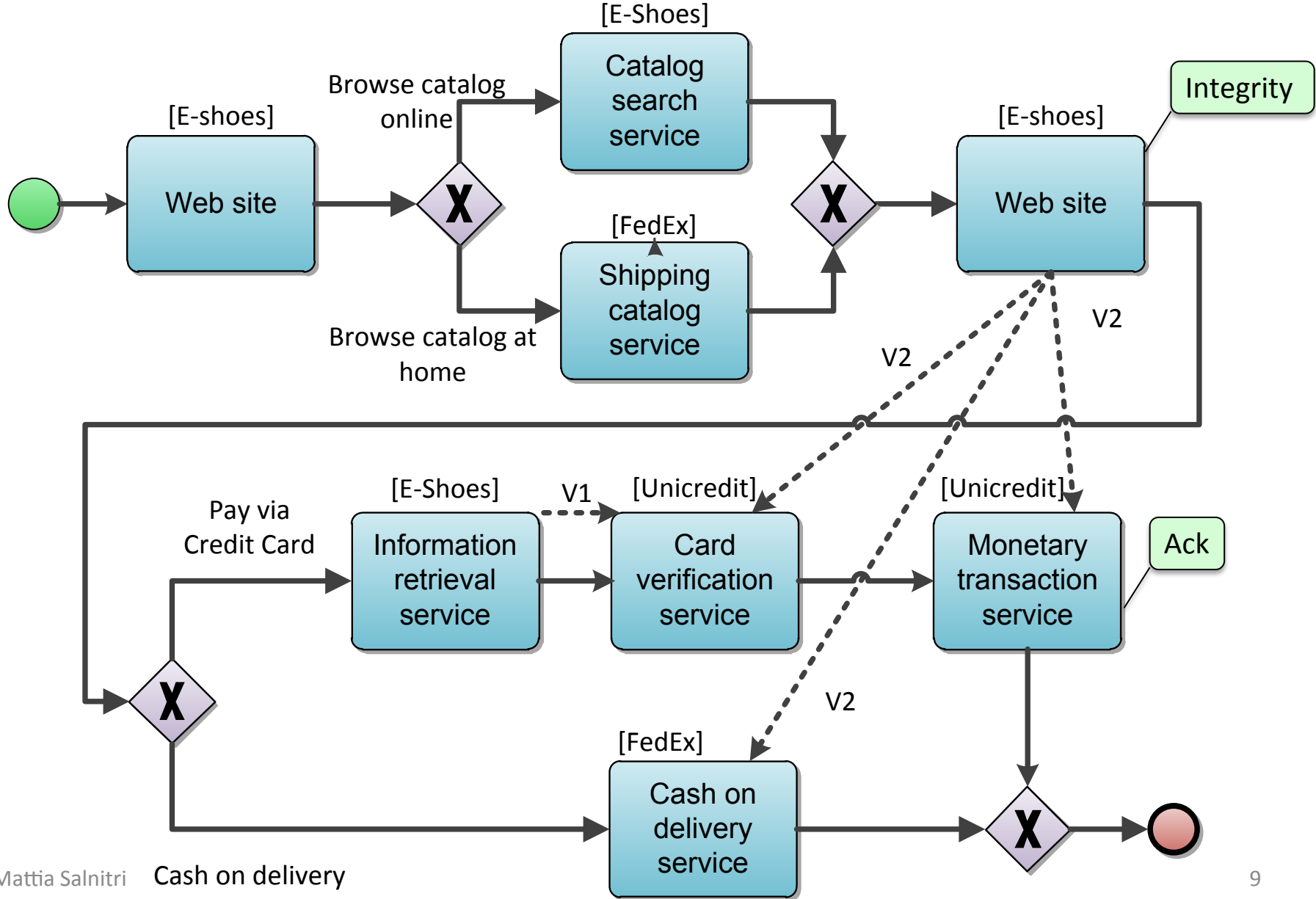
Architecture



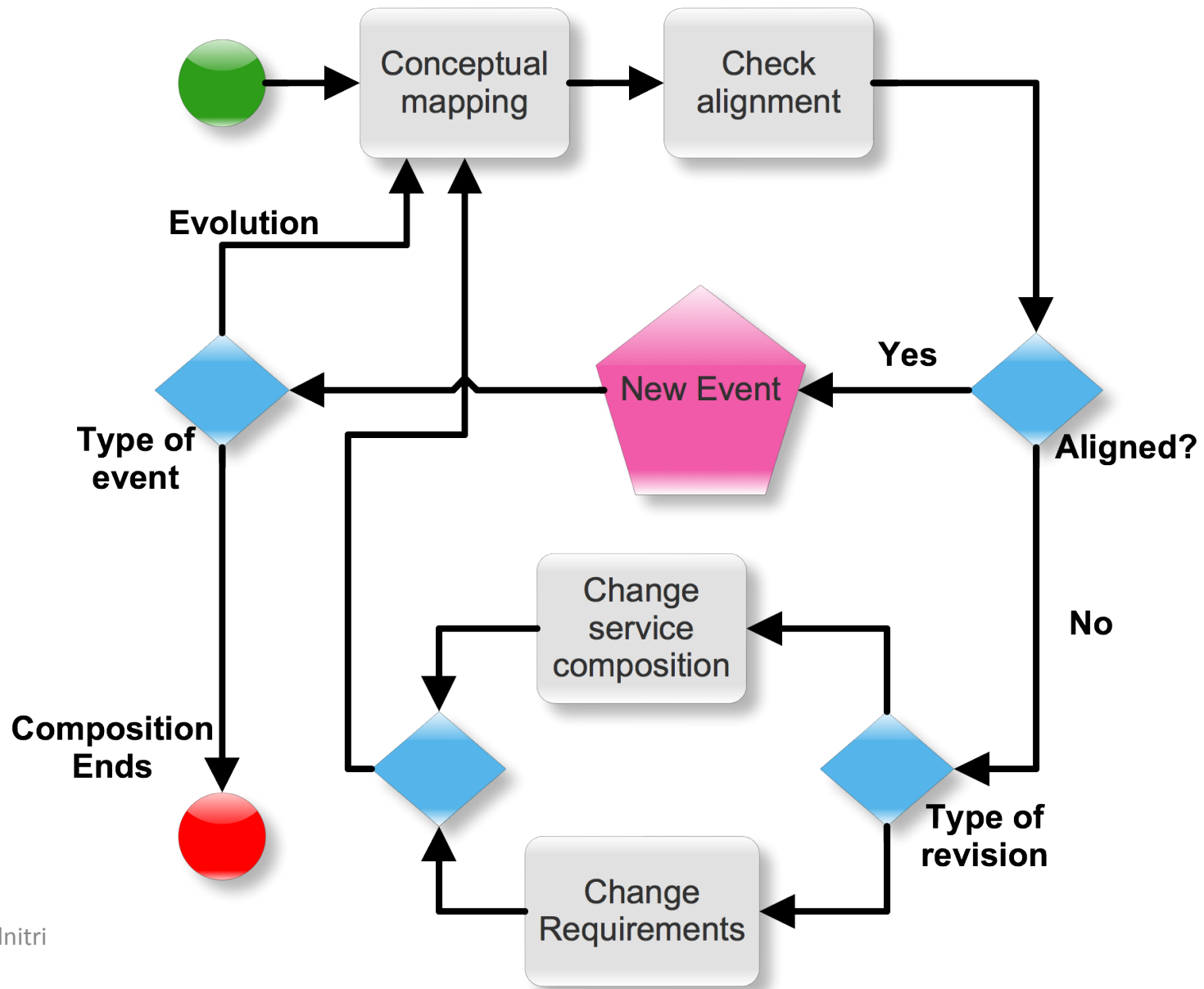
Security requirements with STS-m1



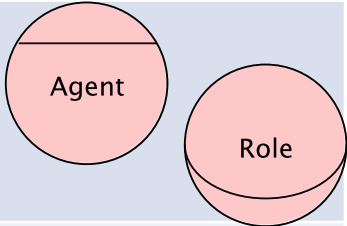
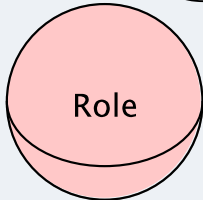
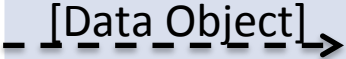
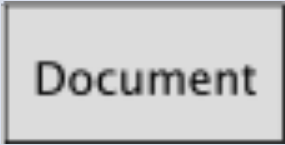


SoA Architecture with Extended BPMN 2.0



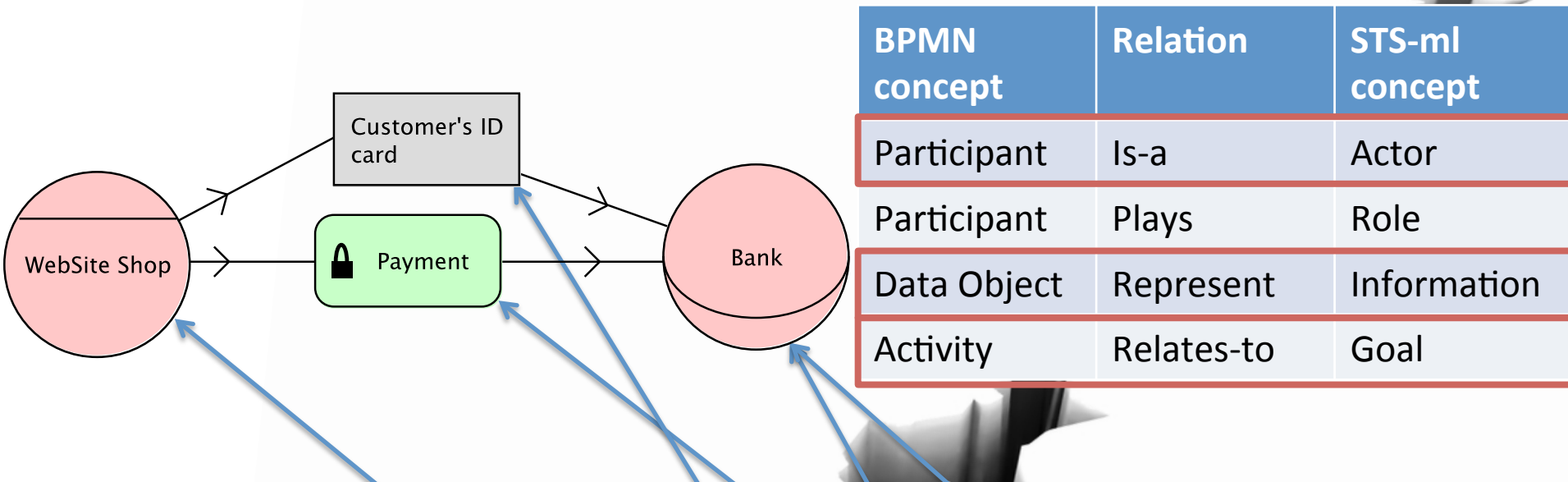
Alignment methodology



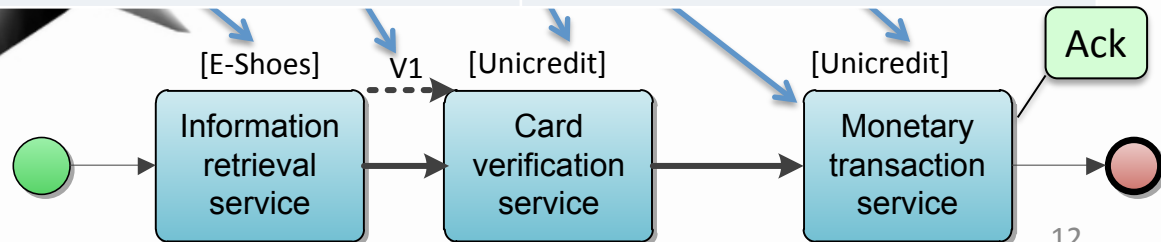
Conceptual mapping

BPMN 2.0 Graphical element	BPMN concept	Relation	STS-ml concept	STS-ml Graphical elements
[Participant]	Participant	Is-a	Actor	
[Participant]	Participant	Plays	Role	
 [Data Object]	Data Object	Represent	Information	
	Activity	Relates-to	Goal	

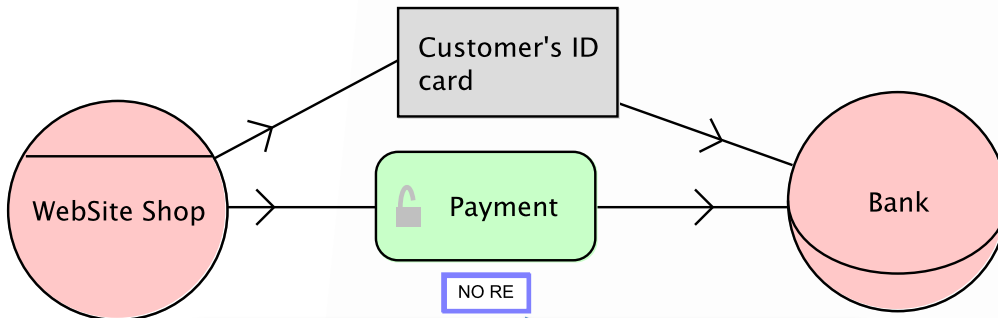
Conceptual mapping example



Business Process Element	Relation	STS-ml Elements
Unicredit	Is-a	Bank
E-shoes	Is-a	WebSite Shop
Money transaction	Relates-to	Payment
V1	Represents	CustomerID



Alignment check example



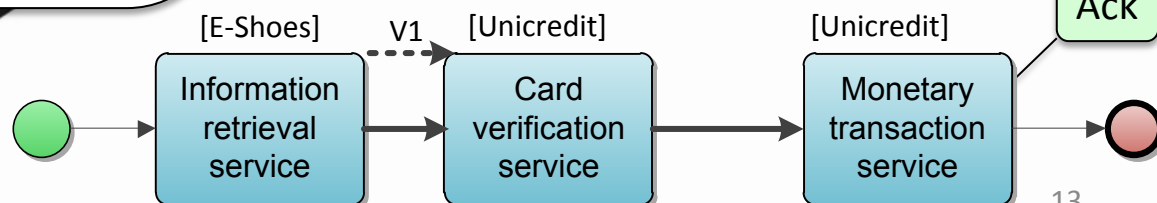
Business Process Element	Relation	STS-ml Elements
Unicredit	Is-a	Bank
E-shoes	Is-a	WebSite Shop
Money transaction	Relates-to	Payment
V1	Represents	CustomerID

Algorithm 5 Non-repudiation verification

VERIFYNR(*actNR*, *perf*, *actCurr*, *found*, *visited*)

```

1  switch TYPEOF(actCurr)
2    case ack :
3      if (actNR ∈ ACKFOR(actCurr) ∧ PERF(actCurr) = perf)
4        then return true
5    case end :
6      return not found
7    case default :
8      if (actCurr = actNR)
9        then found ← true
10  next ← NEXTACTIVITIES(actCurr) \ visited
11  if next = ∅ then return true
12  switch TYPEOF(NEXTELEMENT(actCurr))
13    case gway-excl :
14      return ∧a∈next VERIFYNR(actNR, perf, a, found, visited ∪ actCurr)
15    case gway-incl :
16      return ∨a∈next VERIFYNR(actNR, a, perf, found, visited ∪ actCurr)
17    case activity :
18      VERIFYNR(actNR, perf, GETFIRST(next), found, visited ∪ actCurr)
    
```



Non repudiation

Algorithm 5 Non-repudiation verification

VERIFYNR(*actNR*, *perf*, *actCurr*, *found*, *visited*)

```
1  switch TYPEOF(actCurr)
2    case ack :
3      if (actNR  $\in$  ACKFOR(actCurr)  $\wedge$  PERF(actCurr) = perf)
4        then return true
5    case end :
6      return not found
7    case default :
8      if (actCurr = actNR)
9        then found  $\leftarrow$  true
10  next  $\leftarrow$  NEXTACTIVITIES(actCurr)  $\setminus$  visited
11  if next =  $\emptyset$  then return true
12  switch TYPEOF(NEXTELEMENT(actCurr))
13    case gway-excl :
14      return  $\bigwedge_{a \in next}$  VERIFYNR(actNR, perf, a, found, visited  $\cup$  actCurr)
15    case gway-incl :
16      return  $\bigvee_{a \in next}$  VERIFYNR(actNR, a, perf, found, visited  $\cup$  actCurr)
17    case activity :
18      VERIFYNR(actNR, perf, GETFIRST(next), found, visited  $\cup$  actCurr)
```

Search for an ack,
If needed

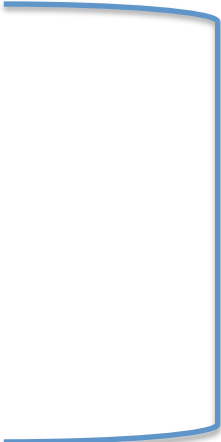
Recursively inspect the
business process

Non distribution

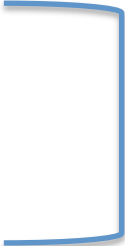
Algorithm 3 Non-Disclosure Verification

VERIFYND($C(\text{deb}, \text{cred}, \top, \text{non-disc}(\text{var}))$, BP , SRS , CM)

```
1   $\text{actByDeb} \leftarrow BP.ACTIVITIESBY(\text{deb})$ 
2   $\text{actByCred} \leftarrow BP.ACTIVITIESBY(\text{cred})$ 
3   $\text{actUsingVar} \leftarrow BP.ACTIVITIESUSING(\text{var})$ 
4   $\text{doc} \leftarrow CM.SEARCH(\text{represents}(\text{var}, *))$ 
5  if  $\text{doc} \neq \text{null}$ 
6    then  $\text{info} \leftarrow SRS.SEARCH(\text{tangible-by}(*, \text{doc}))$ 
7        for each  $i \in \text{info}$ 
8          do  $\text{own} \leftarrow SRS.SEARCH(\text{owns}(*, i))$ 
9              $\text{actByOwner.ADD}(BP.ACTIVITIESBY(\text{own}))$ 
10  $\text{actByOthers} \leftarrow \text{actUsingVar} \setminus \text{actByDeb} \setminus \text{actByCred} \setminus \text{actByOwner}$ 
11 for each  $a_i \in \text{actByDeb}$ 
12 do for each  $a_j \in \text{actByOthers}$ 
13   do if  $\text{var} \in \text{output}(a_i) \cap \text{input}(a_j)$ 
14     then return non-compliant
15 return compliant
```



Retrieve all the
message flows
involved



Examine all the message
flows retrieved

Alignment checking tool

- Security Requirements Compliance Module (SRCM)
 - Run-time/design-time module in Aniketos Framework
 - OSGi - services, apache Karaf compliant
 - It currently supports various security requirements:
 - Non-repudiation
 - Non-distribution
 - Non-modification

Related Works

- **Law compliance** Ghanavati[2009]
 - Design time methodology
 - No security requirements
- **Business process generation** Desai [2004], Zeng[2002]
 - No evolution
- **Auditing Business Process Compliance** Ghose[2007]
 - No requirements
- **Static Compliance-Checking Framework for Business Process Models** Liu [2007]
 - Design time methodology

Future Work

- Extend the list of security requirements supported
- Validate our methodology and SRCM with three Aniketos industrial case studies
- Verification of security properties with symbolic execution techniques
- Alignment between systems



Conclusions

- Key feature
 - we propose **run-time** methodology to **check the alignment of security requirements** with **service compositions**
- Limitations
 - Scalability of semi-automated part
 - Support further security requirements

Thank you!

References

- **Ghanavati[2009]**: Sepideh Ghanavati, Daniel Amyot, and Liam Peyton. Compliance Analysis Based on a Goal-oriented Requirement Language Evaluation Methodology. In Proc. of RE'09, pages 133–142, 2009.
- **Ghose[2007]**: Aditya Ghose and George Koliadis. Auditing Business Process Compliance. In Proc. of ICSOC'07, pages 169–180, 2007.
- **Liu [2007]**: Y. Liu, S. Müller, and K. Xu. A Static Compliance-Checking Framework for Business Process Models. IBM Systems Journal, 46(2):335–361, 2007.
- **Desai [2004]**: N. Desai, A.U. Mallya, A.K. Chopra and M.P. Singh. Processes = Protocols + Policies: A methodology for business process development. Un Proc. Of WWW'05, 2005.
- **Zeng[2002]**: L. Zeng, D. Flaxer, H. Chang and J. Jeng. PLM flow – Dynamic Business Process Composition and Execution by Rule Inference. In Proc. of TES'02, pages 141--150