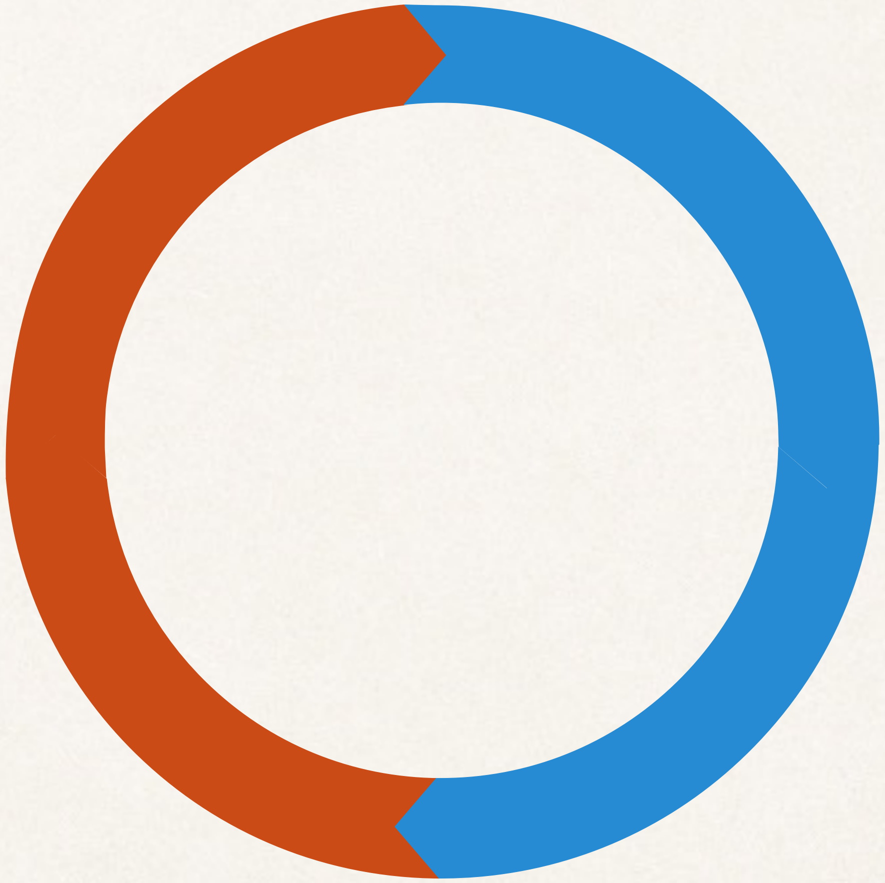
FoCUS, on Implementations of Service-Oriented Computing

Saverio Giallorenzo

FoCUS | A Positive Loop

Theory
Make problems tractable
Formalise new ideas



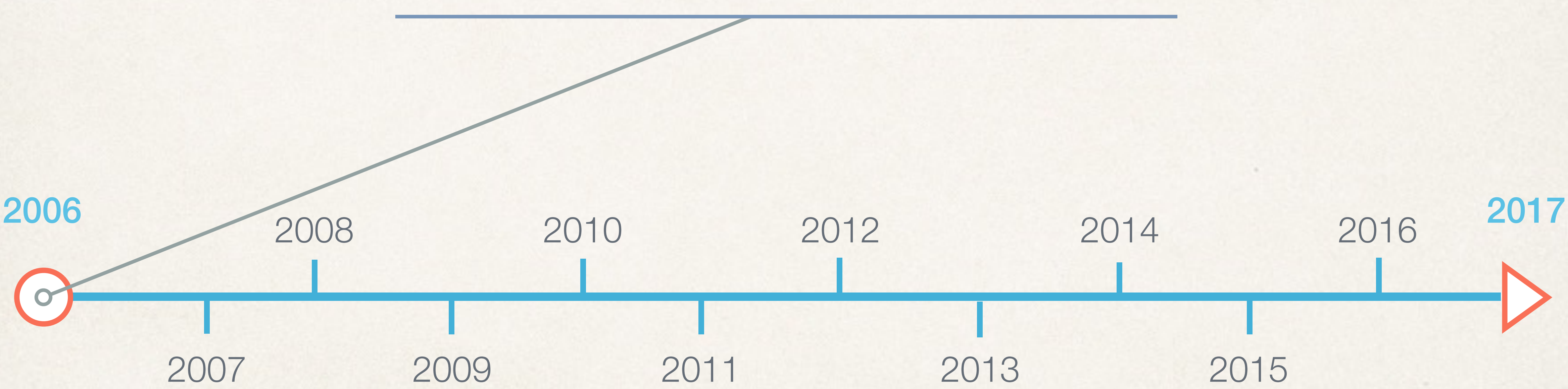
Implementation
Real-world problems
New technologies

Formal Semantics

A case of a Service-Oriented Programming Language

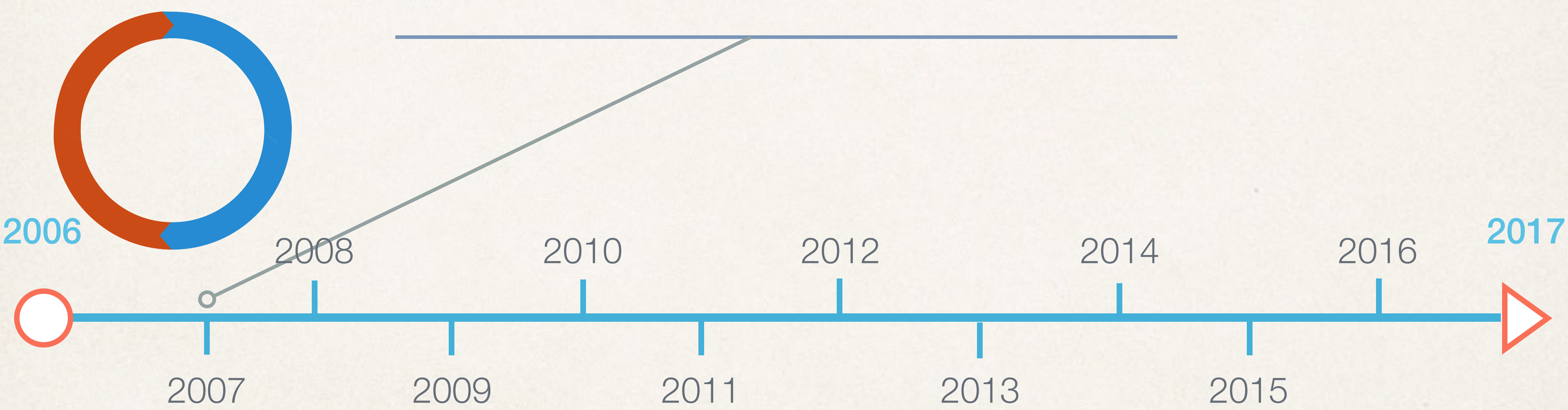
SOCK: A Calculus for Service Oriented Computing*

Claudio Guidi, Roberto Lucchi, Roberto Gorrieri,
Nadia Busi, and Gianluigi Zavattaro



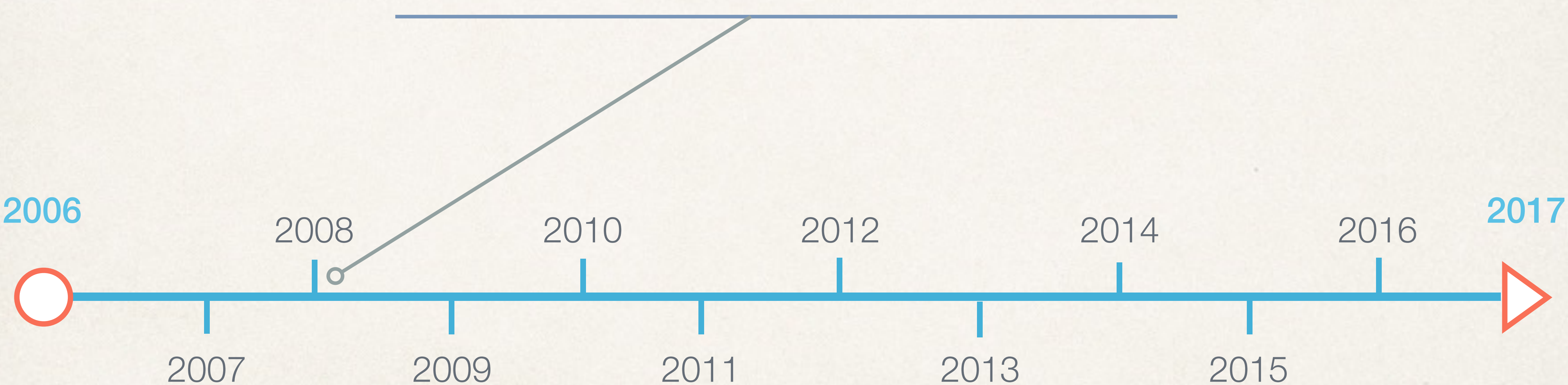
JOLIE: a Java Orchestration Language Interpreter Engine

Fabrizio Montesi[1], Claudio Guidi[2], Roberto Lucchi[2]
Gianluigi Zavattaro[2]



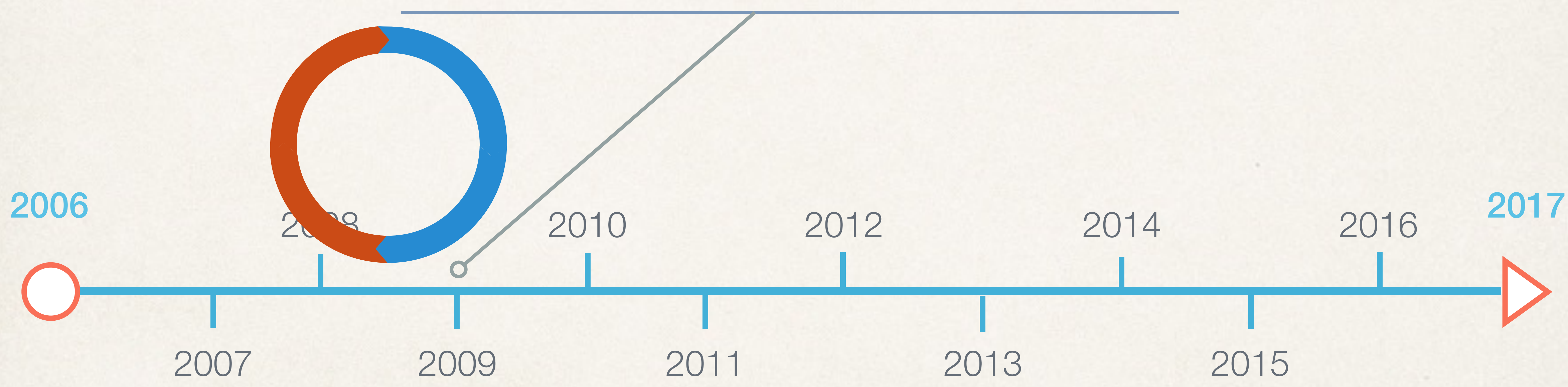
On the Interplay Between Fault Handling and Request-Response Service Invocations*

Claudio Guidi, Ivan Lanese, Fabrizio Montesi, and Gianluigi Zavattaro



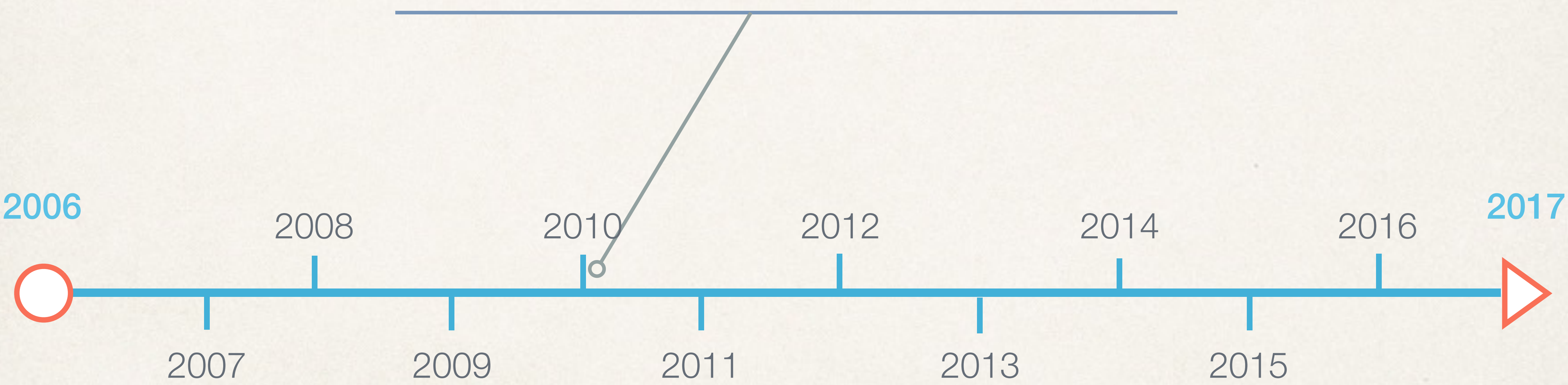
Dynamic Error Handling in Service Oriented Applications*

Claudio Guidi, Ivan Lanese†, Fabrizio Montesi, Gianluigi Zavattaro



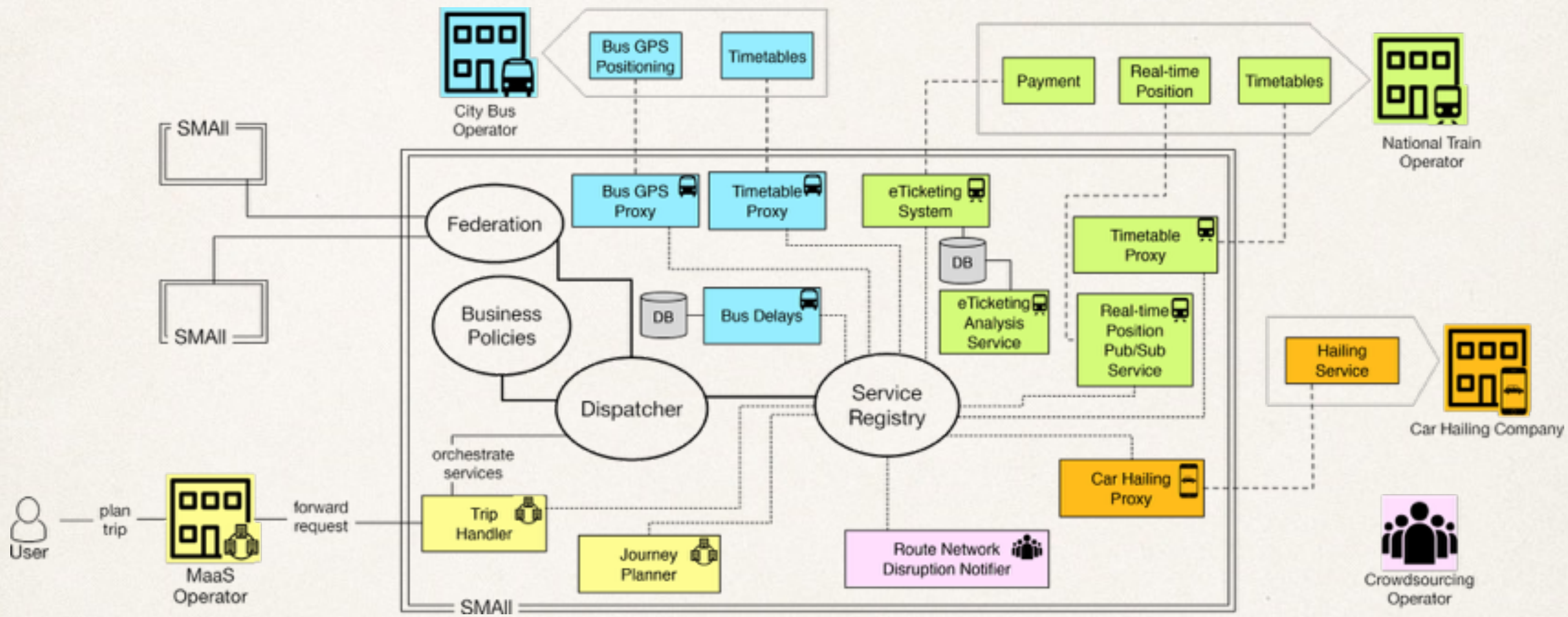
A Framework for Rule-Based Dynamic Adaptation*

Ivan Lanese[1], Antonio Bucchiarone[2], and Fabrizio Montesi[1]



The First Language for Microservices

A Market for Microservices | The SMAll Platform



Microservices for fine-grained:

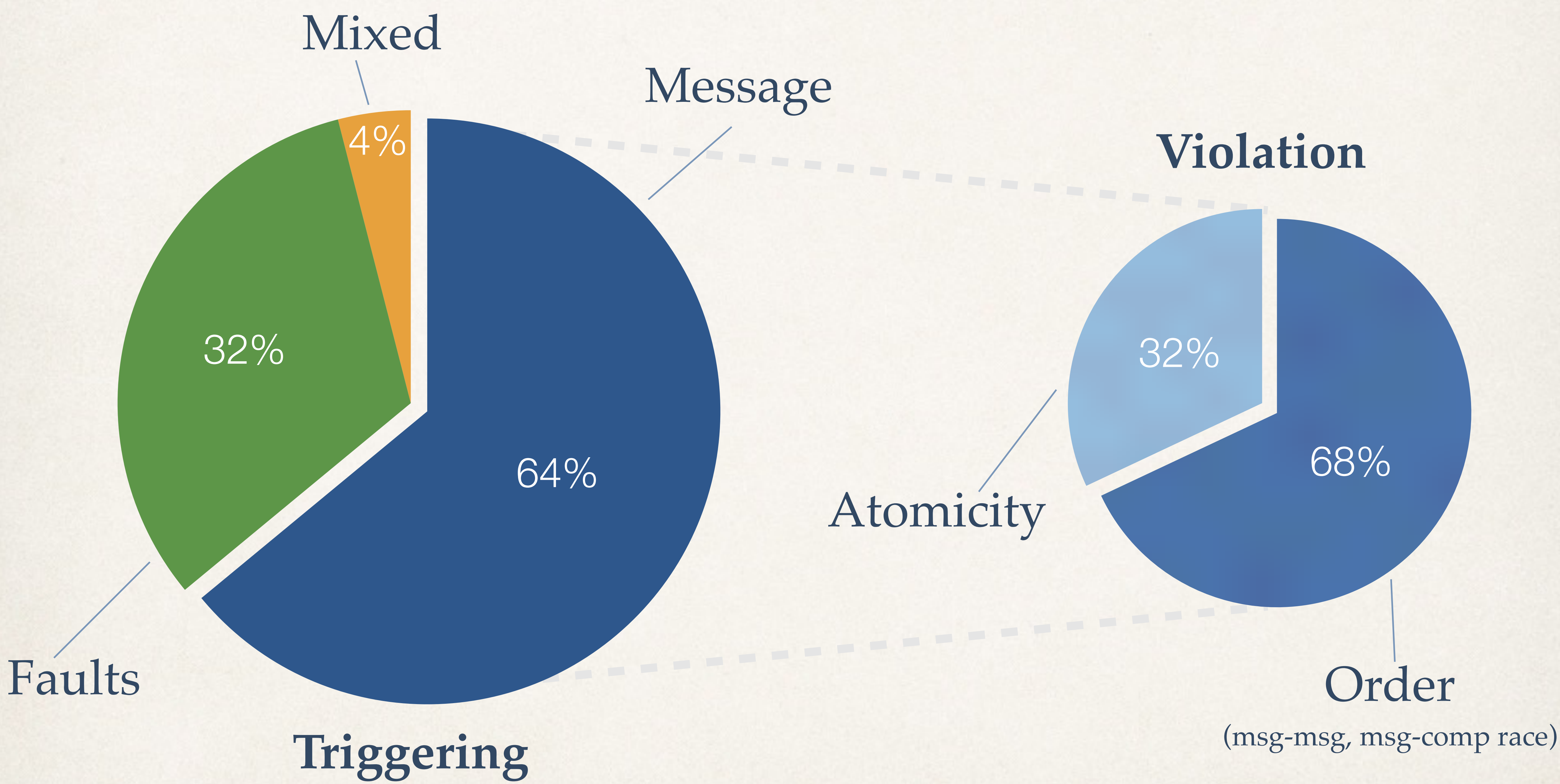
- access policies (avoids some side-effects of multi-tenancy!);
- scalability;
- deployment (multistage continuous integration!).

BONAPARTE, TURREAU, AND VOLNEY AT NICE IN 1793

Bonaparte laid before the others the plan of campaign in Italy which he executed as general in chief, three years later

Global vs Local

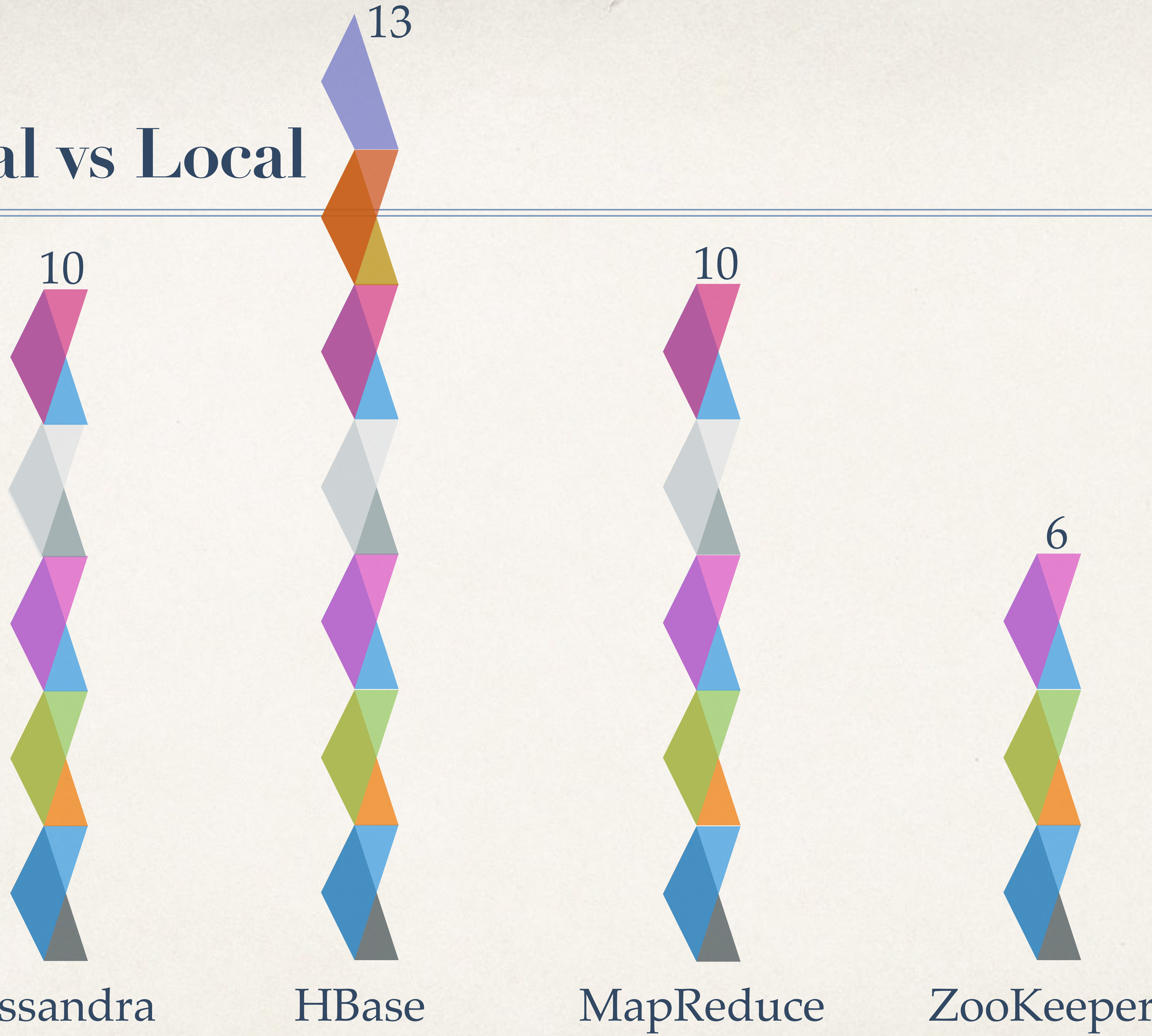
Global vs Local | Timing Conditions



— TaxDC

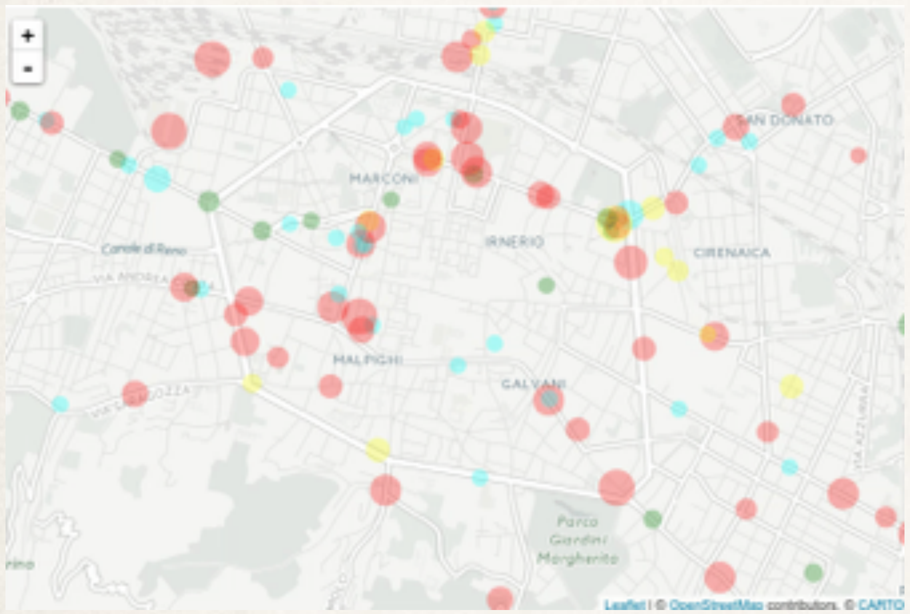
TaxDC: A Taxonomy of Non-Deterministic Concurrency Bugs in Datacenter Distributed Systems

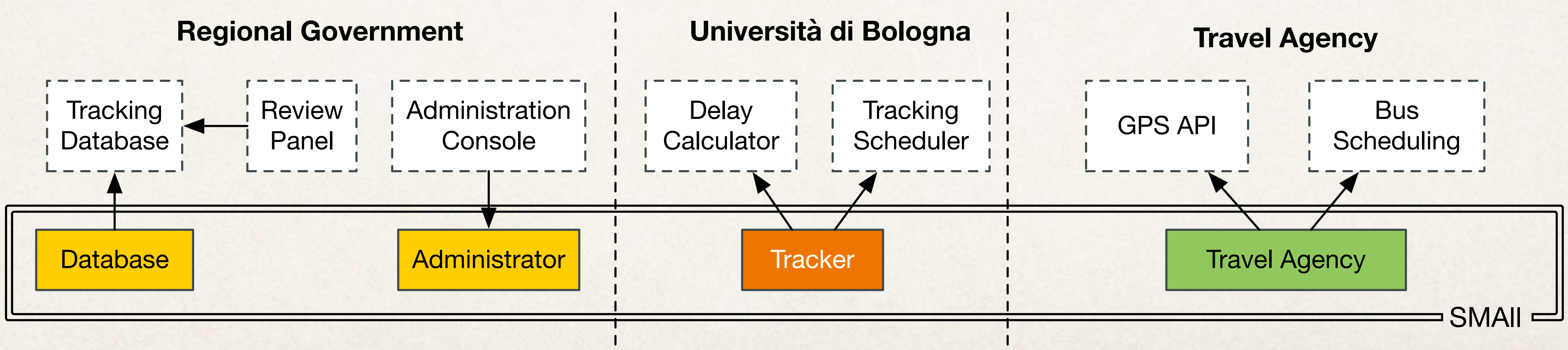
Global vs Local



Choreographies

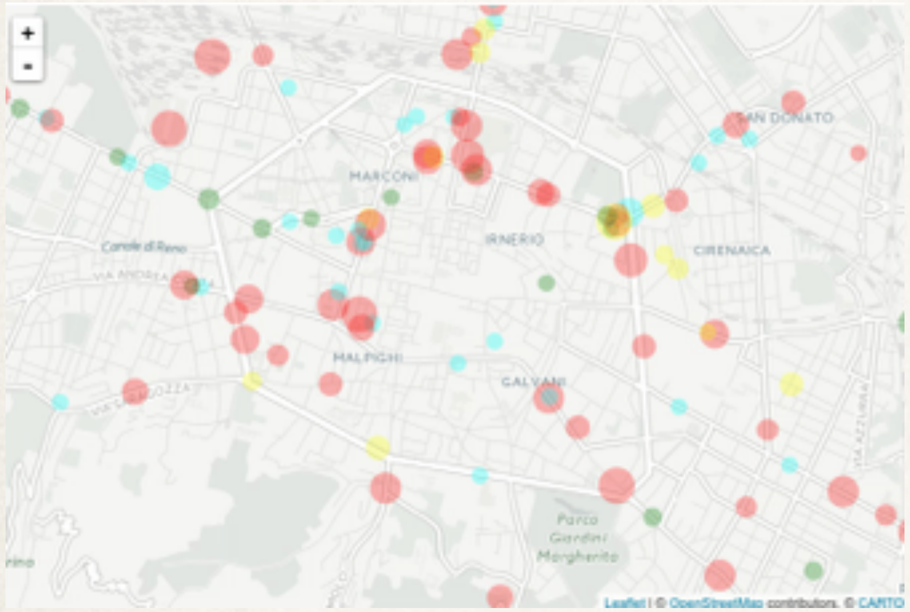
```
line@Admin = getInput( "Insert line to track" );
{
 setLine: Admin( line ) -> DB( line )
 |
 setLine: Admin( line ) -> BusAgency( line )
};
schdl@BusAgency = getBusSchedule( line );
passSchdl: BusAgency( schdl ) -> Tracker( schdl );
hasNext@Tracker = hasNextStop( schdl );
while( hasNext )@Tracker {
 gps@BusAgency = getPosition( line );
 passPosition: BusAgency( gps ) -> Tracker( gps );
 delay@Tracker = calculateDelay( schdl, gps );
 storeDelay: Tracker( delay ) -> DB( delay );
 {
  _@DB = insertDelay( line, delay )
  |
  hasNext@Tracker = hasNextStop( sched )
 }
}
```

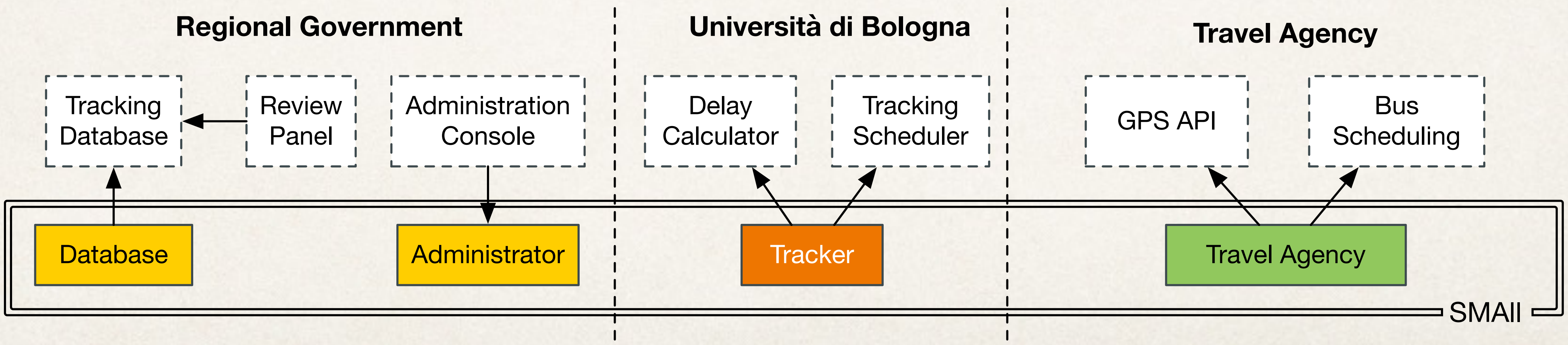




Choreographies

```
line@Admin = getInput( "Insert line to track" );
{
 setLine: Admin( line ) -> DB( line )
 |
 setLine: Admin( line ) -> BusAgency( line )
};
schdl@BusAgency = getBusSchedule( line );
passSchdl: BusAgency( schdl ) -> Tracker( schdl );
hasNext@Tracker = hasNextStop( schdl );
while( hasNext )@Tracker {
 gps@BusAgency = getPosition( line );
 passPosition: BusAgency( gps ) -> Tracker( gps );
 delay@Tracker = calculateDelay( schdl, gps );
 storeDelay: Tracker( delay ) -> DB( delay );
 {
  _@DB = insertDelay( line, delay )
  |
  hasNext@Tracker = hasNextStop( sched )
 }
}
```





Correctness by design and by construction

Choreography → *EPP* → Endpoint Projection

(Correct by design)

(Correct by construction)

```
line@Admin = getInput( "Insert line to track" );
{
 setLine: Admin( line ) -> DB( line )
 |
 setLine: Admin( line ) -> BusAgency( line )
};
```

EPP →

Admin

```
getInput@UI( "Insert line to track" )
{ setLine@Database( line )
  |
  setLine@BusAgency( line ) };
...
```

Database

```
setLine( line );
...
```

BusAgency

```
setLine( line );
...
```

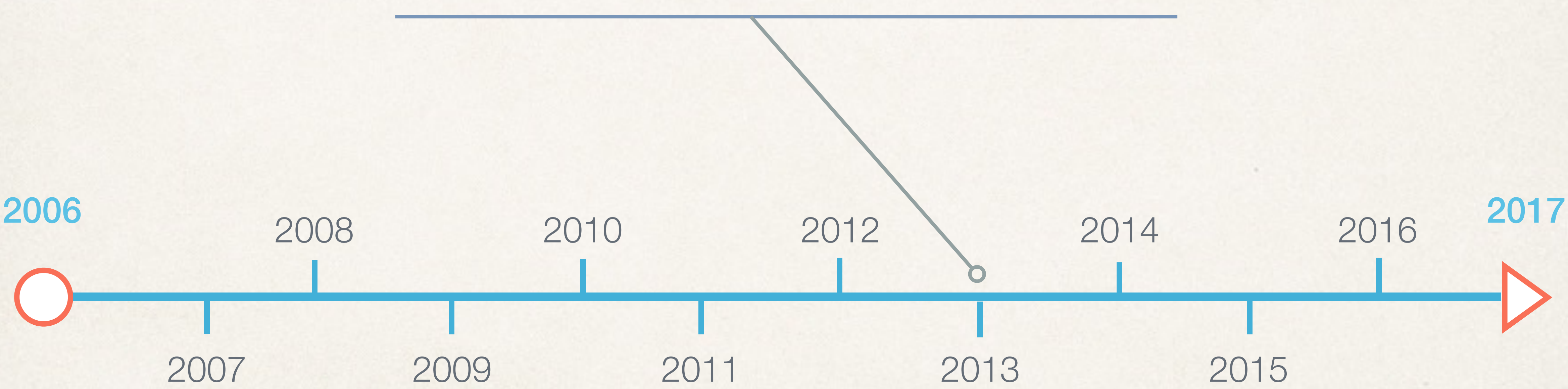
Time for discussion!

Time for discussion!

Dynamic Choreographies*

Safe Runtime Updates of Distributed Applications

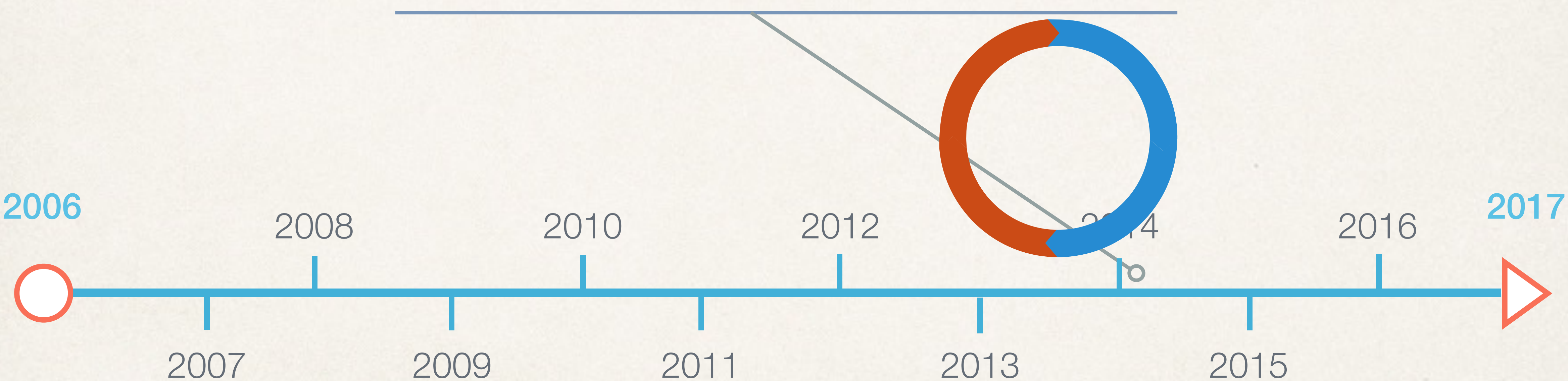
Mila Dalla Preda[1], Maurizio Gabbrielli[2], Saverio Giallorenzo[2], Ivan Lanese[2], and Jacopo Mauro[2]



AIOCJ: A Choreographic Framework for Safe Adaptive Distributed Applications

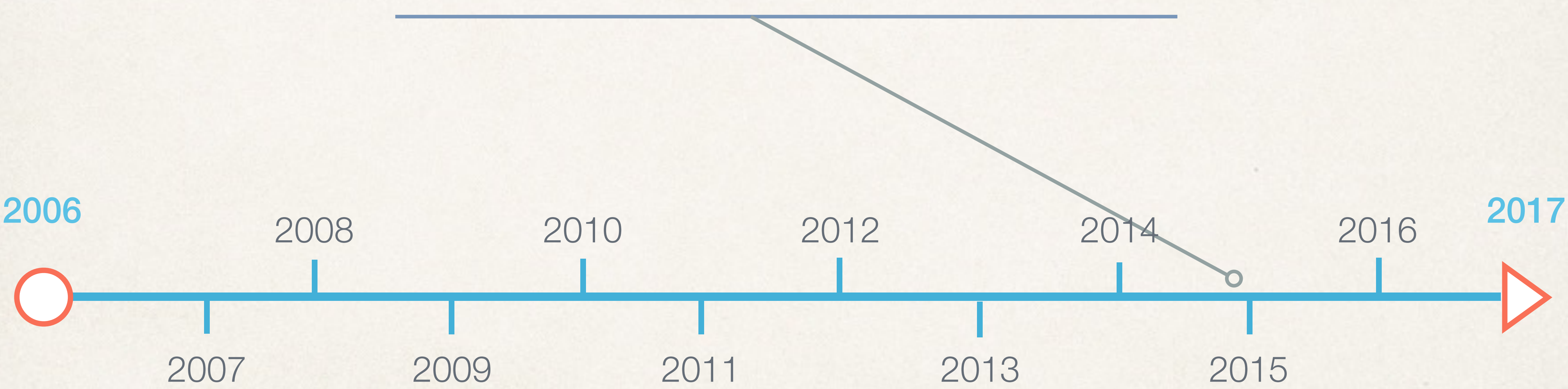
Technical Report

Mila Dalla Preda[1], Saverio Giallorenzo[2],
Ivan Lanese[2], Jacopo Mauro[2], and Maurizio Gabbrielli[2]



Applied Choreographies

Maurizio Gabbrielli Saverio Giallorenzo Fabrizio Montesi



DYNAMIC CHOREOGRAPHIES: THEORY AND IMPLEMENTATION

MILA DALLA PREDA*, MAURIZIO GABBRIELLI†, SAVERIO GIALLORENZO†, IVAN LANESE†,
AND JACOPO MAURO‡