DICE Project

J.I. Requeno, J. Merseguer, S. Bernardi

Universidad de Zaragoza, Spain

DICE Horizon 2020 Project
Grant Agreement no. 644869
http://www.dice-h2020.eu

Funded by the Horizon 2020
Framework Programme of the European Union

DICE Project

- DICE - Developing Data-Intensive Cloud Applications with Iterative Quality Enhancements
- Horizon 2020 Research & Innovation Action
  - Quality-Aware Development for Big Data applications
  - Feb 2015 - Jan 2018, 4M Euros budget
  - 9 partners (Academia & SMEs), 7 EU countries

Universidad Zaragoza

Imperial College London

POLITECNICO DI MILANO

Motivation

- Software market rapidly shifting to Big Data
  - 32% compound annual growth rate in EU through 2016
  - 35% Big data projects are successful [CapGemini 2015]
- ICT-9 call focused on SW quality assurance (QA)
  - ISTAG: call to define environments "*for understanding the consequences of different implementation alternatives (e.g. quality, robustness, performance, maintenance, evolvability, …)*"
- QA evolving too slowly compared to the technology trends (Big data, Cloud, DevOps …)
  - DICE aims at closing the gap
  - Still crucial for competiveness!

Quality Dimensions

- Reliability
  - Availability
  - Fault-tolerance
- Efficiency
  - Performance
  - Costs
- Safety & Privacy
  - Verification (e.g., deadlines)
  - Data protection

Some Challenges in Big Data…

- Lack of quality-aware development for Big Data
  - How to describe in MDE Big Data technologies
    - Spark, Hadoop/MapReduce, Storm, Cassandra, …
    - Cloud storage, auto-scaling, private/public/hybrid, …
- Today no QA toolchain can help reasoning on data-intensive applications
    - What if I double memory?
    - What if I parallelize more the application?

Performance Analysis of Apache Storm Applications using SPNs

José Merseguer

Universidad de Zaragoza, Spain

DICE Horizon 2020 Project
Grant Agreement no. 644869
http://www.dice-h2020.eu

Funded by the Horizon 2020
Framework Programme of the European Union

Context

- Apache Storm
  - Distributed real-time computation system for processing large volumes of high-velocity data
- Real-time data-processing *stream* applications
  - E.g., customization of searches, sentiment analysis in social networks ✉ **Big Data**

The Problem

- Capgemini Research
  - Only 13% companies have achieved full-scale production on Big Data technologies
- Storm specific problems
  - Low-latency processing ✉ Highly demanding performance requirements
  - Youthfulness of the technology

The Need

- *Urgent need for novel, performance oriented, software engineering methodologies and tools capable of dealing with the complexity of such a new environment*

Our Proposal

- Assessment of performance requirements
  - While configuring their Storm designs to specific execution contexts, i.e., multi-user private or public cloud infrastructures

Our Proposal …

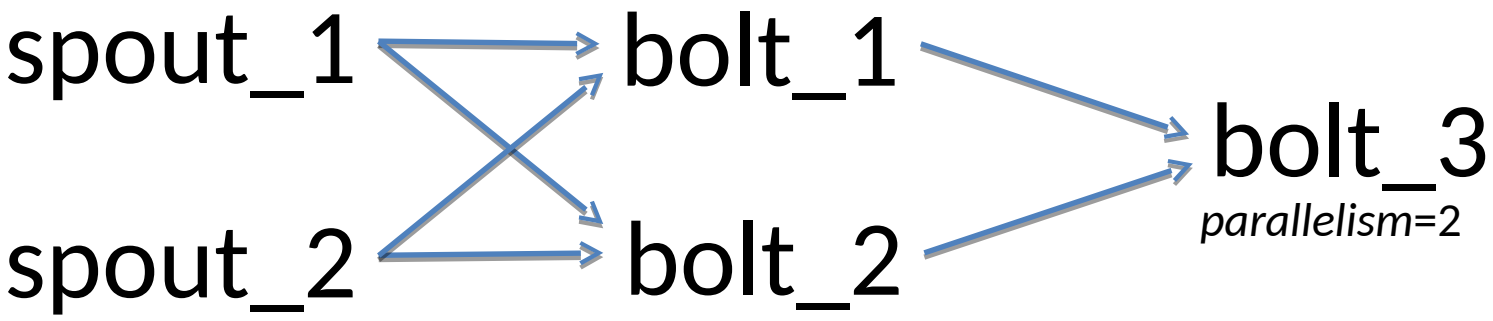
- We have developed a Quality-driven framework for Storm
  - UML modelling of Storm applications
    - We propose a novel **UML profile** ✉ **Domain-specific modelling language**
  - Transformation of the UML Storm models into Stochastic Petri Nets ✉ **performance model**
  - Simulation of the performance model
  - Getting performance results from the simulation

Benefits

- Benefits of our proposal
  - Predict the behaviour of the application for future demands (e.g., response time, throughput or utilization)
  - Impact of the stress situations in some performance parameters
  - Detection of performance bottlenecks

Modelling Storm Applications

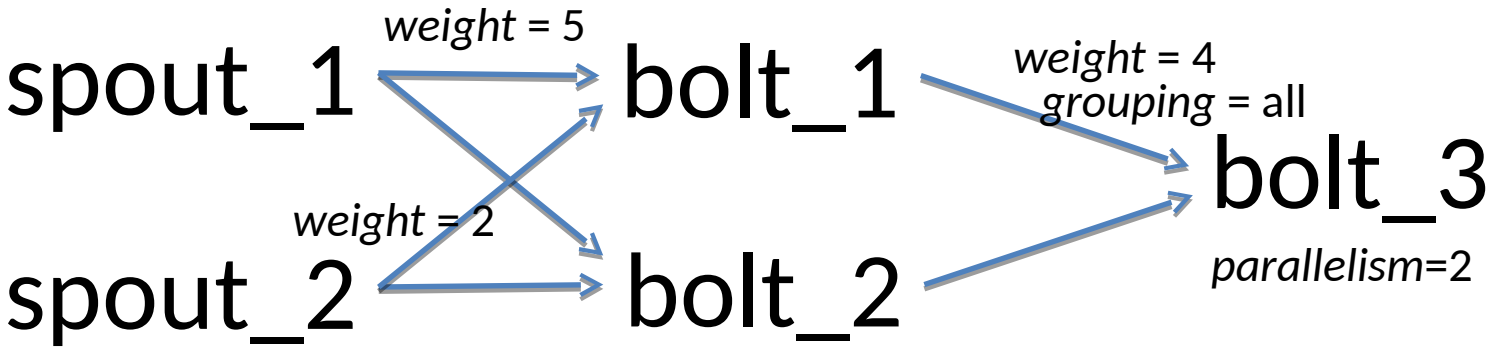
- A Storm application is designed as a DAG
  - Two kinds of *nodes*:
    - **Spouts**, sources of information that inject streams of data into the topology
    - **Bolts**, process input data and produce results



    - ***parallelism***, number of concurrent threads executing the same task (spout or bolt)

Modelling Storm Applications

- *Edges*, define the connections for the transmission of data from one *node* to another:
  - *weight*, number of tuples the next bolt requires for emitting a new message
  - *grouping*, the way a message is propagated to and handled by the receiving nodes (*all*, *shuffle*, subset)

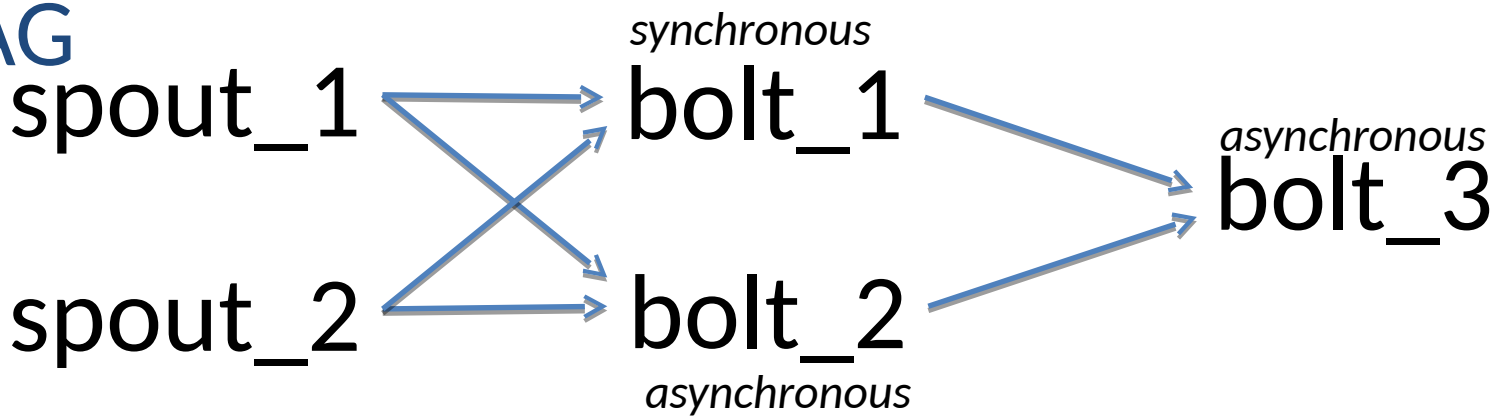


Storm Concepts for Performance

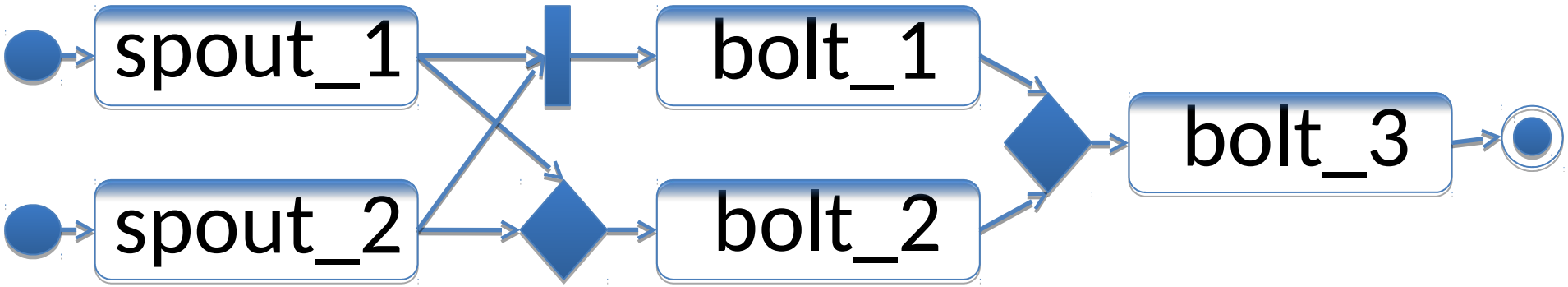
#	Concept	Meaning
1.	*Spout* (task)	Source of information
2.	*Rate*	No. of tuples per unit of time produced by a spout
3.	*Bolt* (task)	Data elaboration
4.	*Weight*	No. of tuples required by a bolt
5.	*Asynchronous policy*	The bolt progresses when at least one input tuple is available
6.	*Synchronous policy*	The bolt progresses when all input tuples are available
7.	*Parallelism*	No. of concurrent threads per task
8.	*Grouping*	Tuple propagation policy (e.g., all)
9.	*Scheduling*	Deployment of tasks

UML Modelling for Storm

DAG



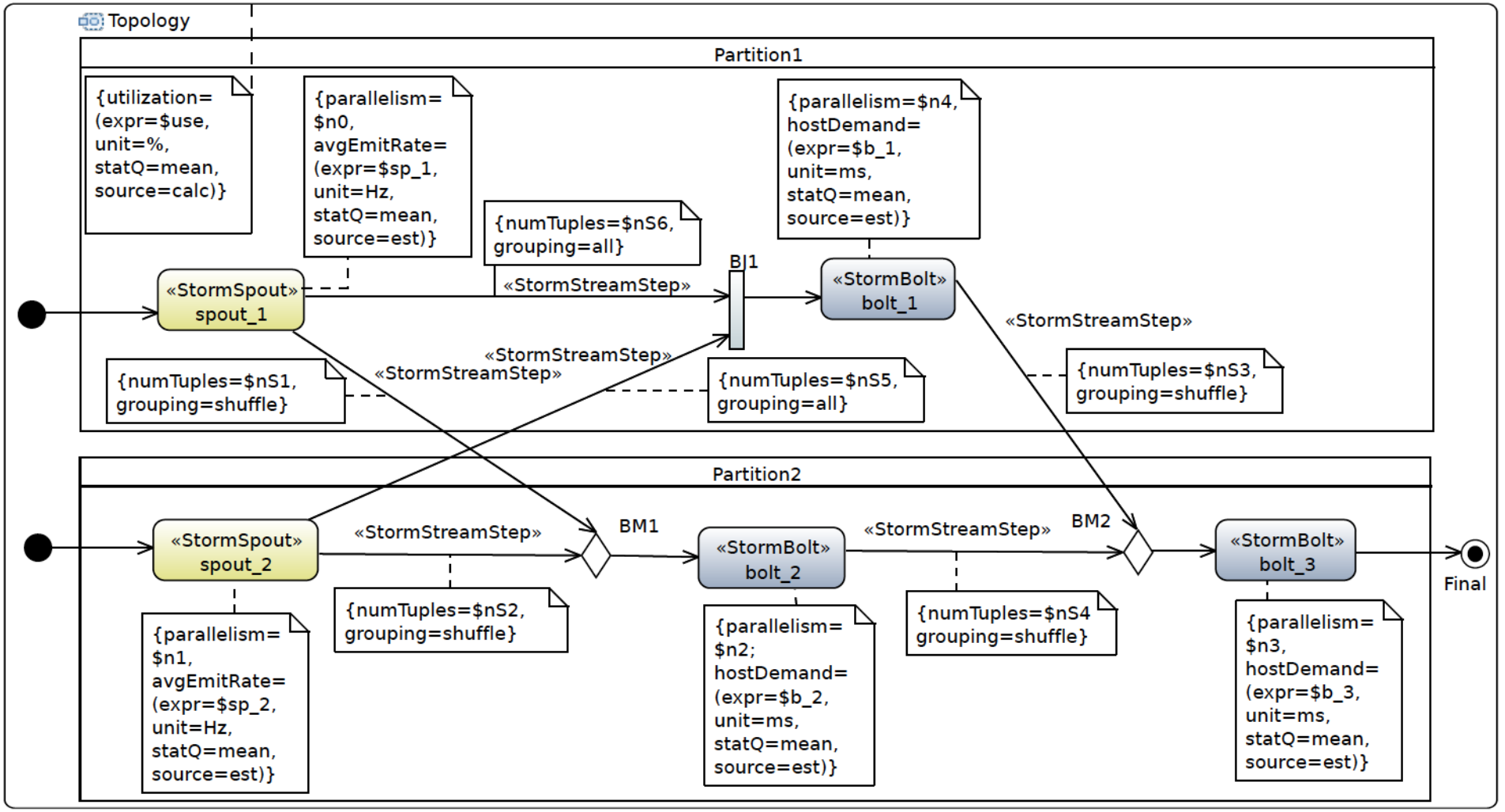
UML Activity Diagram



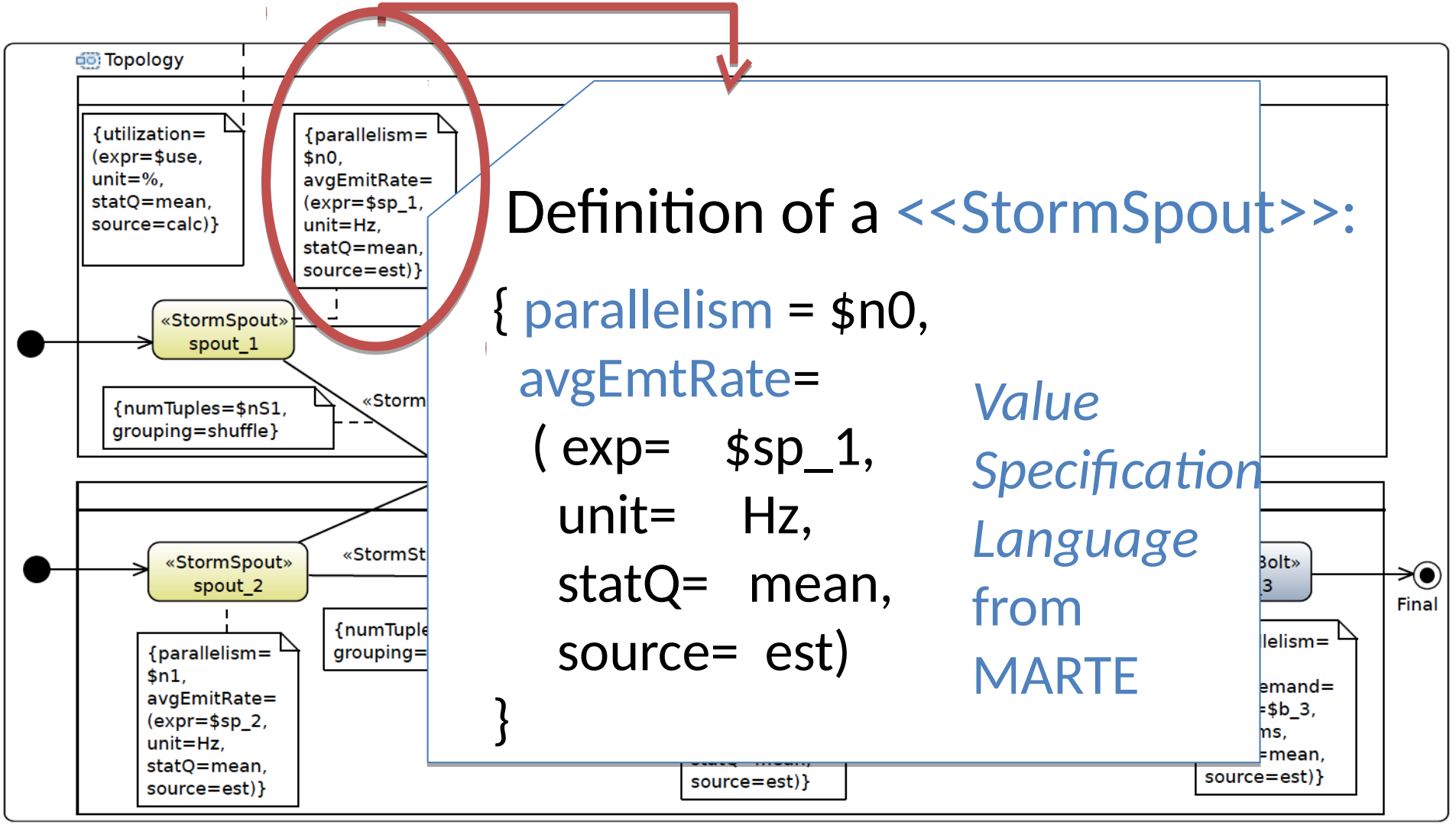
A UML Profile for Storm

Storm Concept	Stereotype	Tag	MARTE inheritance
Bolt	<<StormBolt>>		<<GaStep>>
exec. time		hostDemand	
parallelism		parallelism	
Spout	<<StormSpout>>		<<GaStep>>
emission rate		avgEmitRate	
Stream	<<StormStreamStep>>		<<GaStep>>
weight		numTuples	
grouping		grouping	
Scheduling		resMult	<<GaExecHost>>
		capacity	<<GaCommHost>>

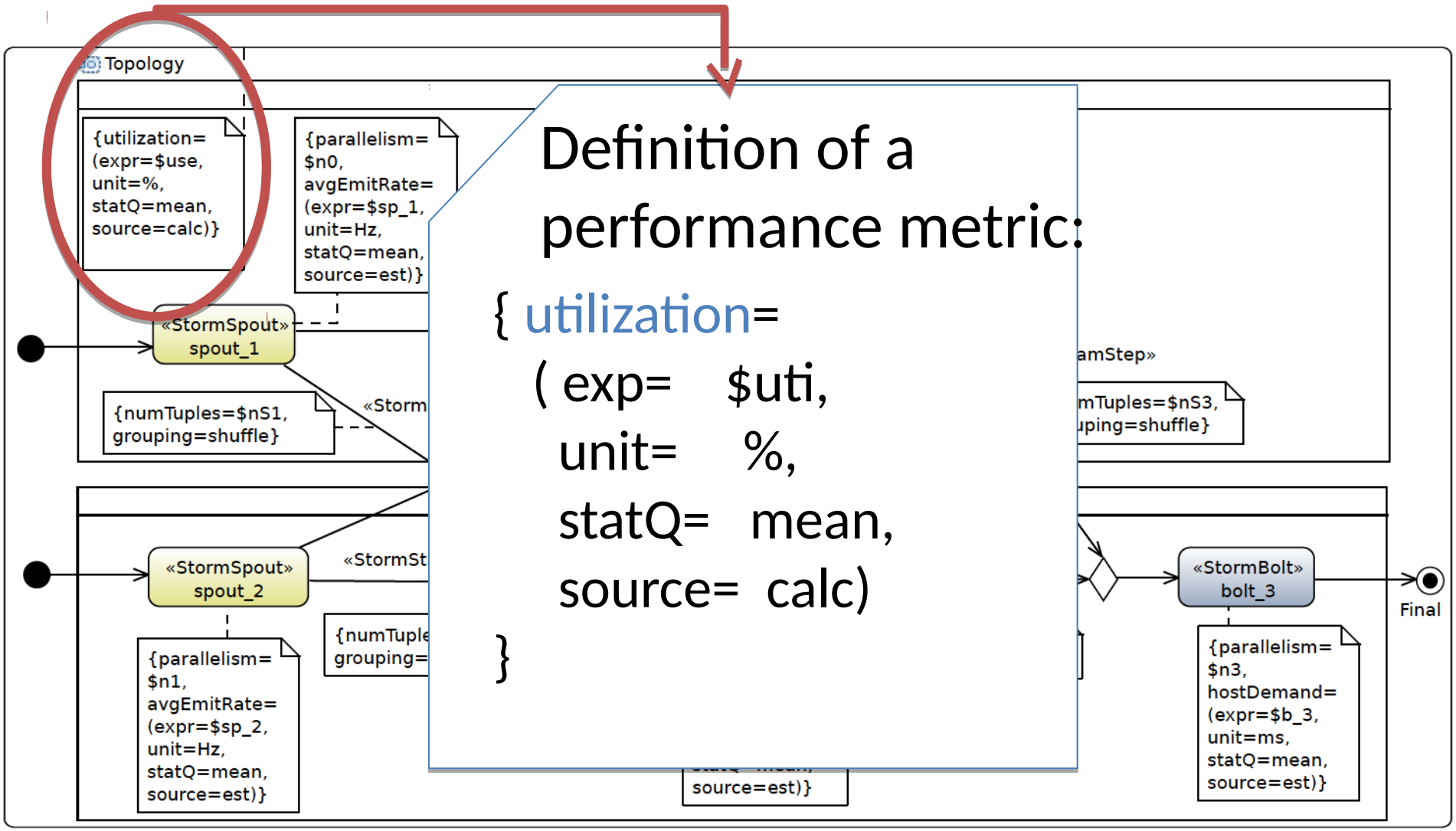
UML Modelling for Storm



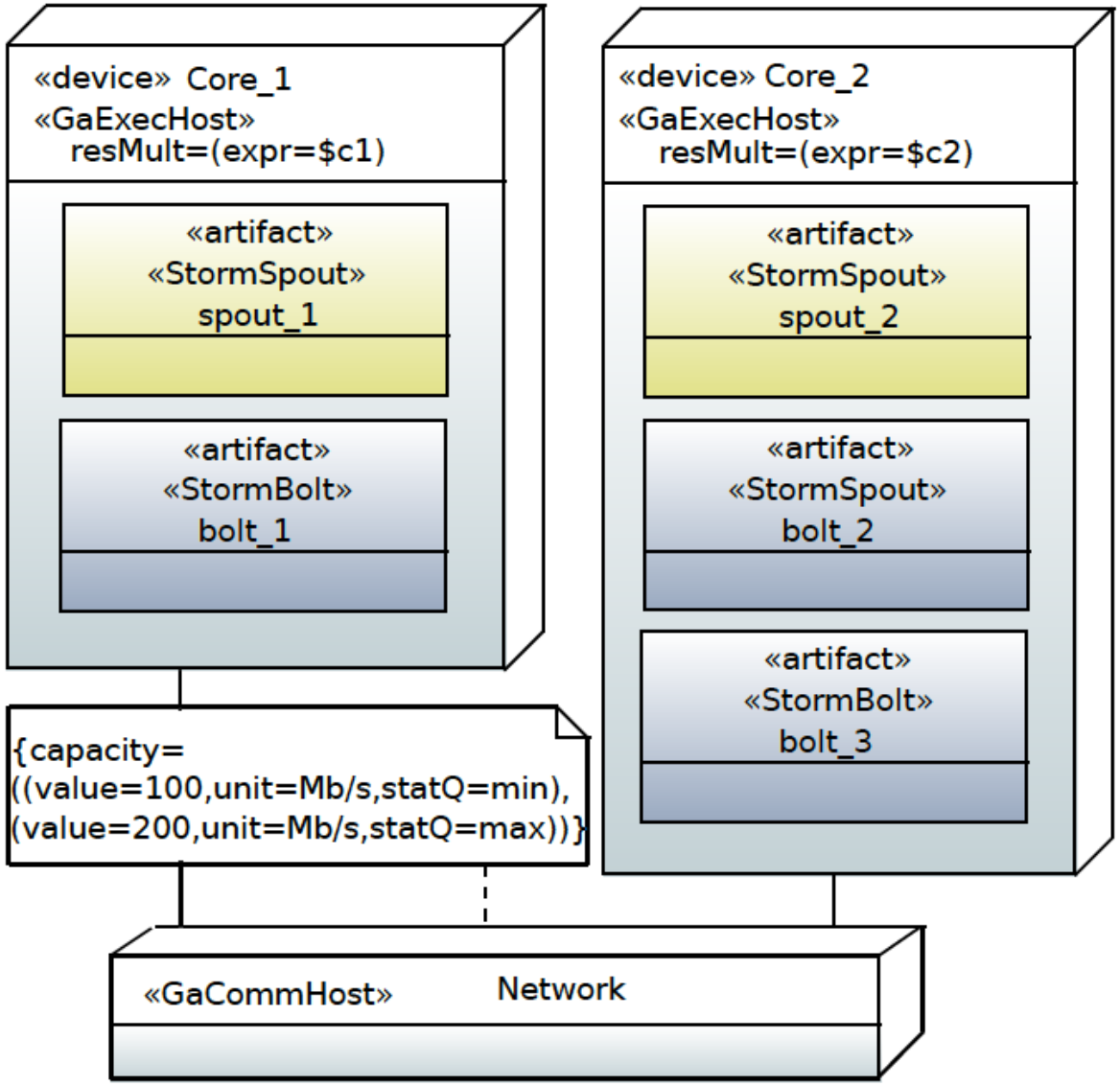
UML Modelling for Storm



UML Modelling for Storm



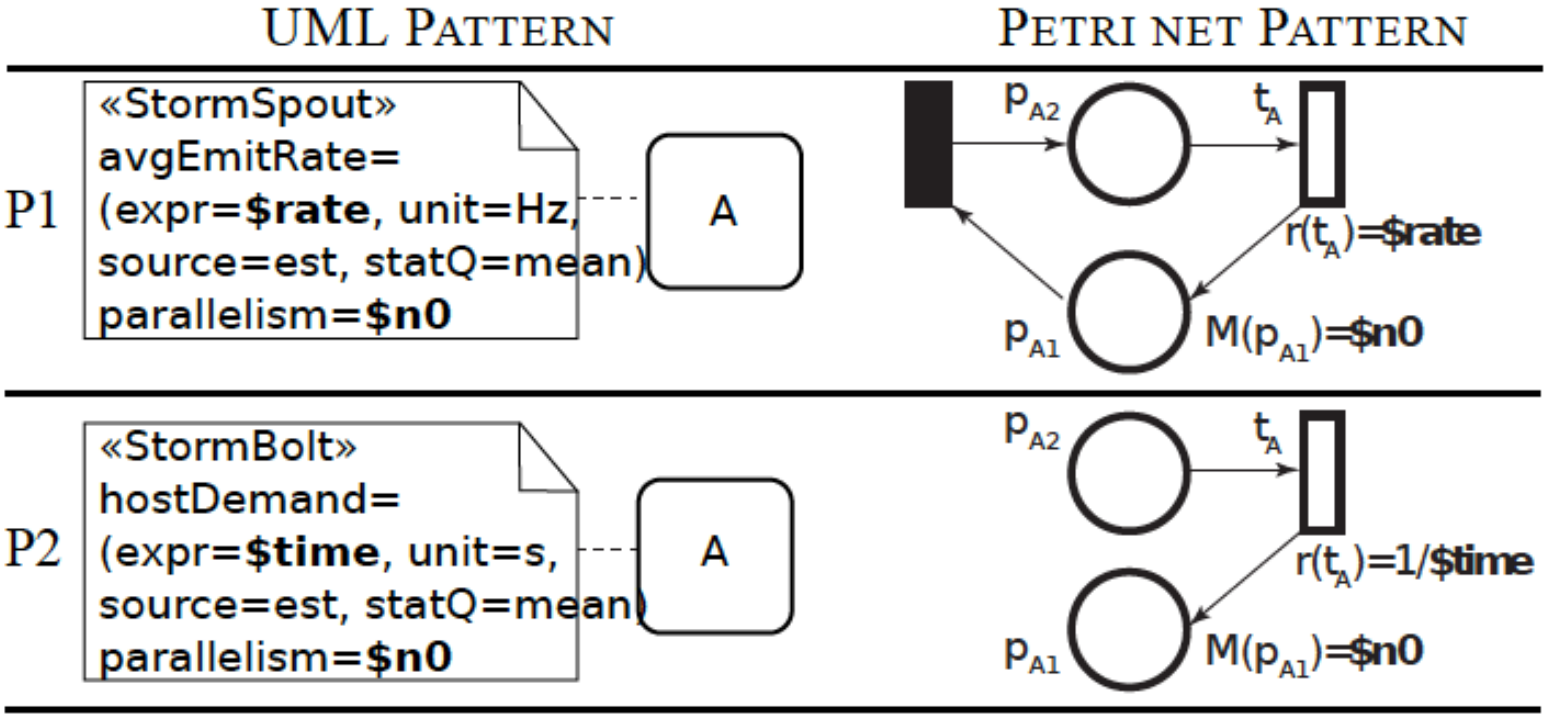
UML Modelling for Storm



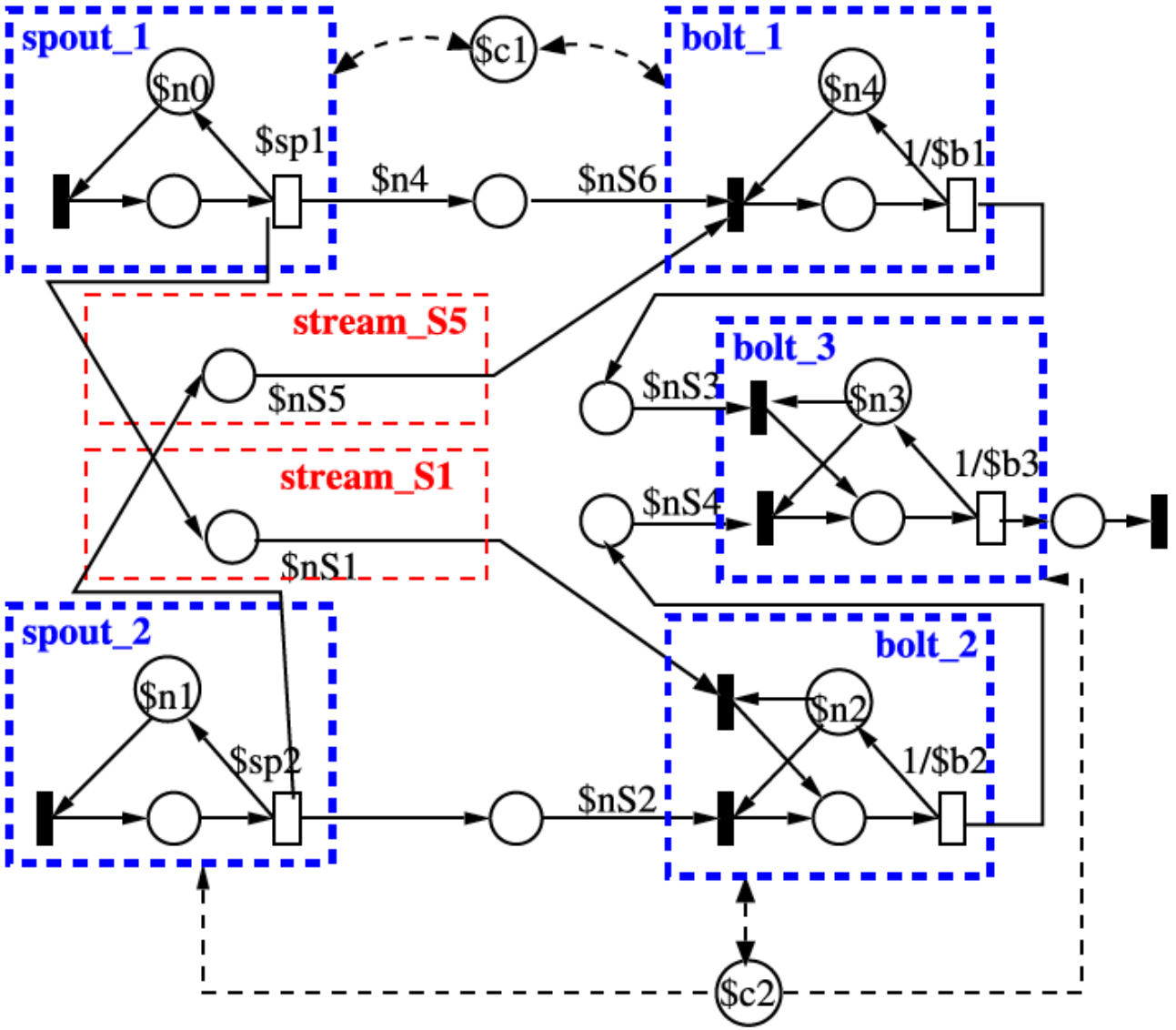
Transformation of the Storm

- For evaluation of the metrics, we need to transform the Storm design into a performance model
- Generalized Stochastic Petri Net (GSPN)
- We propose a set of *transformation patterns* ;
  - Each pattern takes as input a part of the Storm design and produces a GSPN subnet
  - We compose the pieces

Examples of patterns



Final GSPN


spout_1
$n0
$sp1
$c1
$n4
$nS6
bolt_1
$n4
1/$b1
stream_S5
$nS5
bolt_3
$nS3
$n3
1/$b3
stream_S1
$nS4
$nS1
spout_2
$n1
$sp2
bolt_2
$n2
1/$b2
$nS2
$c2

Implementation

- We have implemented in Eclipse (Papyrus Modelling environment):
  - The Storm profile
  - The transformation patterns (using QVT) to PNML
  - Transformation to GreatSPN (using Acceleo)
  - The evaluation of performance metrics (throughput, response time, utilization)
- You can download:
  - [8] DICE Consortium. DICE Simulation Tool, 2017. URL:https://github.com/dice-project/DICE-Simulation/

Validation

- *First*: We used our tool
  - 1) to model the application
  - 2) to transform it into a GSPN
  - 3) to get results
- *Second*: We deployed the Storm application in a real cluster and got results from the Storm monitoring tool
- *Third*: We compared results. **Relative error**
- Metric: **Utilization** of the bolts

Emission rate or host Demand (ms)				%Utilization (%Relative error)		
$\$sp_i$	$\$b_1$	$\$b_2$	$\$b_3$	bolt_1	bolt_2	bolt_3
20	100	100	100	97,6 (2,45)	100 (9,91)	39,0 (3,71)
30	100	100	100	100 (2,91)	100 (4,48)	43,2 (6,05)
40	100	100	100	100 (1,00)	99,5 (0,40)	38,1 (4,50)
50	100	100	100	83,1 (4,61)	78,0 (2,07)	33,1 (2,42)
100	100	100	100	39,4 (0,06)	38,7 (3,27)	17,0 (4,18)
20	20	30	40	41,2 (2,90)	56,4 (5,85)	32,0 (0,26)
30	20	30	40	26,9 (0,68)	40,1 (0,46)	20,8 (2,86)
40	20	30	40	21,2 (6,05)	29,9 (0,19)	16,6 (4,05)
50	20	30	40	16,5 (3,39)	24,7 (3,21)	12,8 (0,09)
100	20	30	40	9,5 (15,7)	11,9 (0,95)	7,3 (12,61)

Thanks

www.dice-h2020.eu

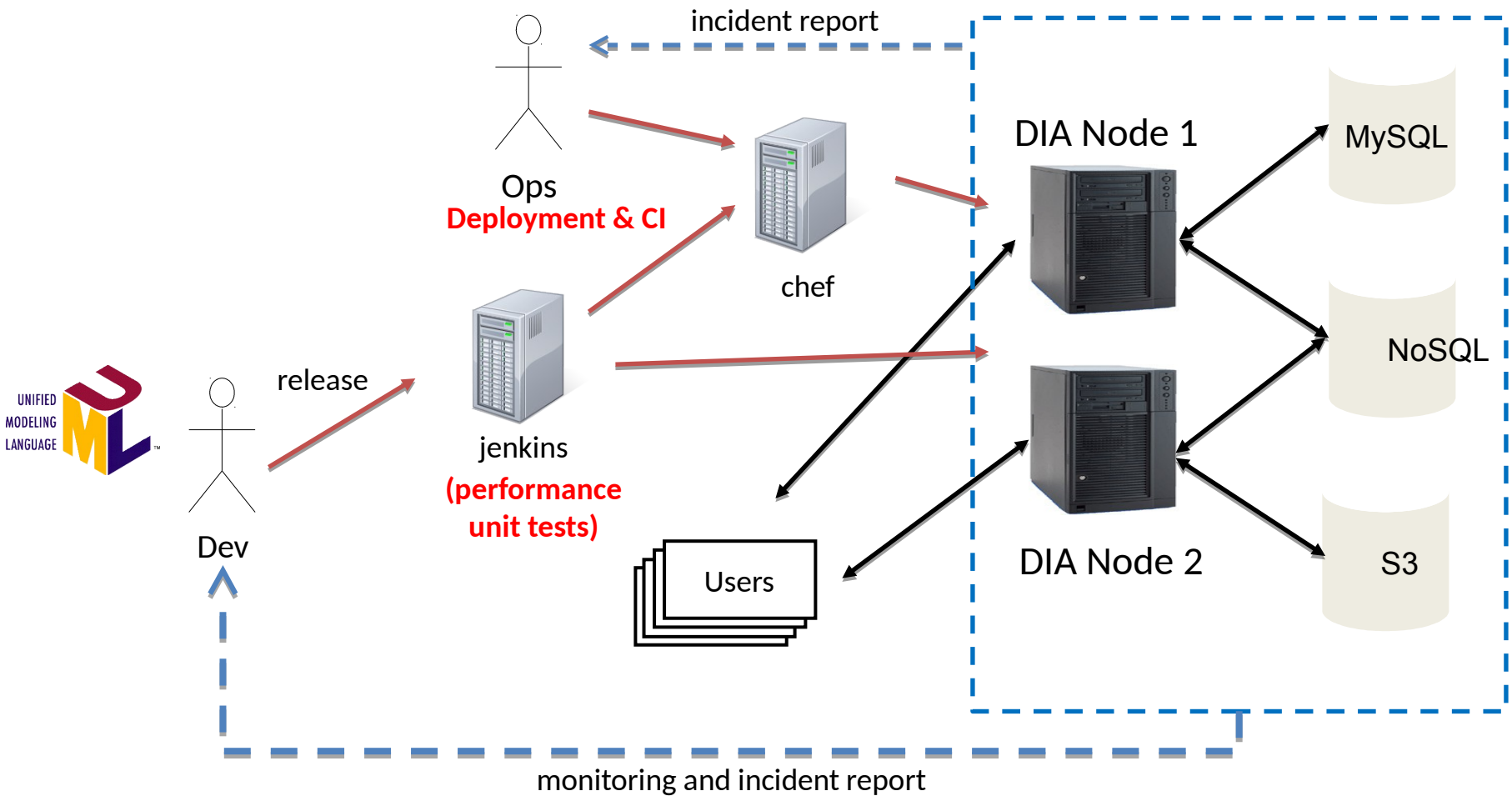
High-Level Objectives

- Tackling skill shortage and steep learning curves
  - Data-aware methods, models, and tools
- Shorter time to market for Big Data applications
  - Cost reduction, without sacrificing product quality
- Decrease development and testing costs
  - Select optimal architectures that can meet SLAs
- Reduce number and severity of quality incidents
  - Iterative refinement of application design

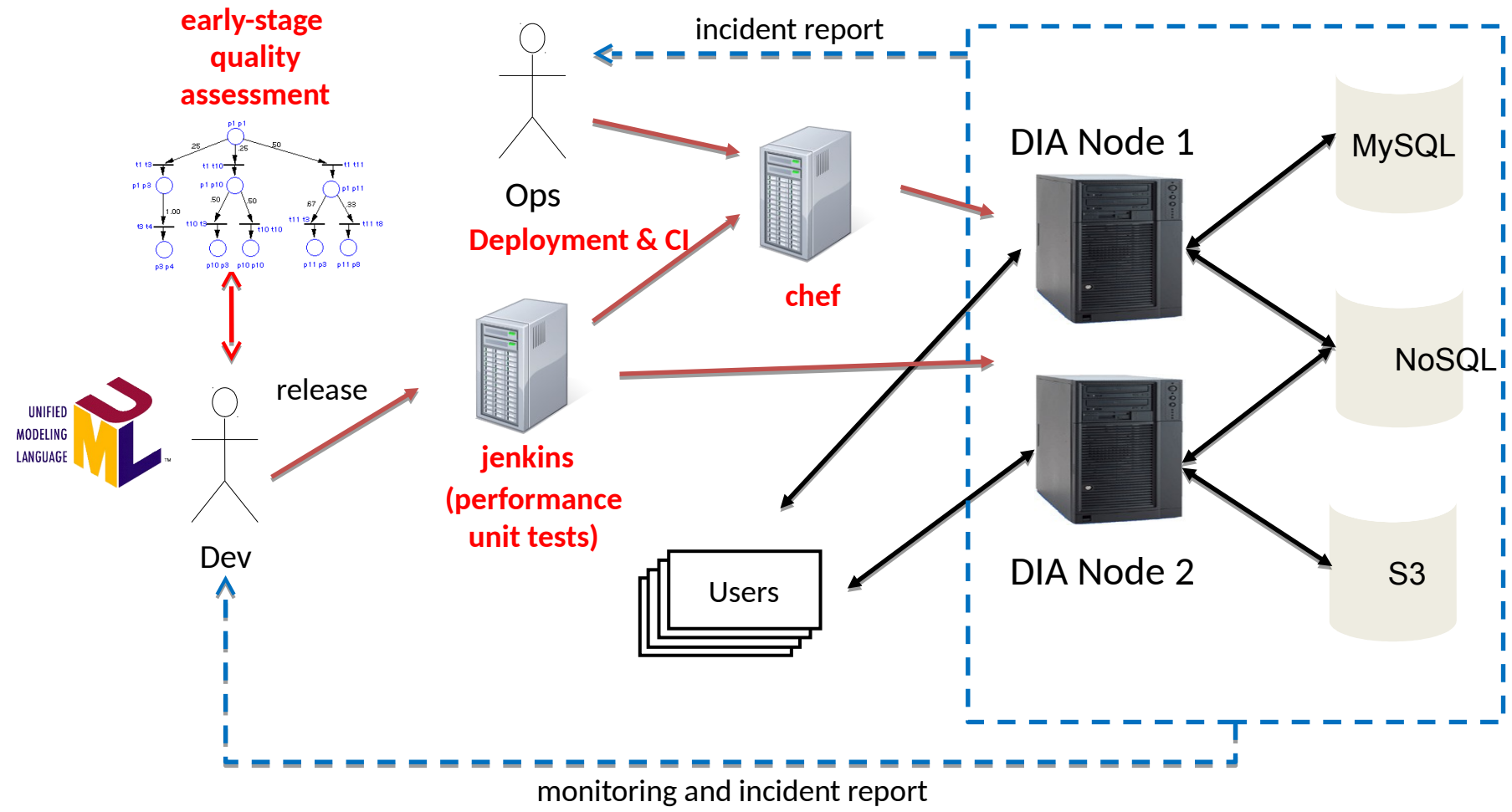
… in a DevOps fashion

- Software development methods are evolving
- DevOps closes the gap between Dev and Ops
  - From agile development to agile delivery
  - Lean release cycles with automated tests and tools
  - Deep modelling of systems is the key to automation

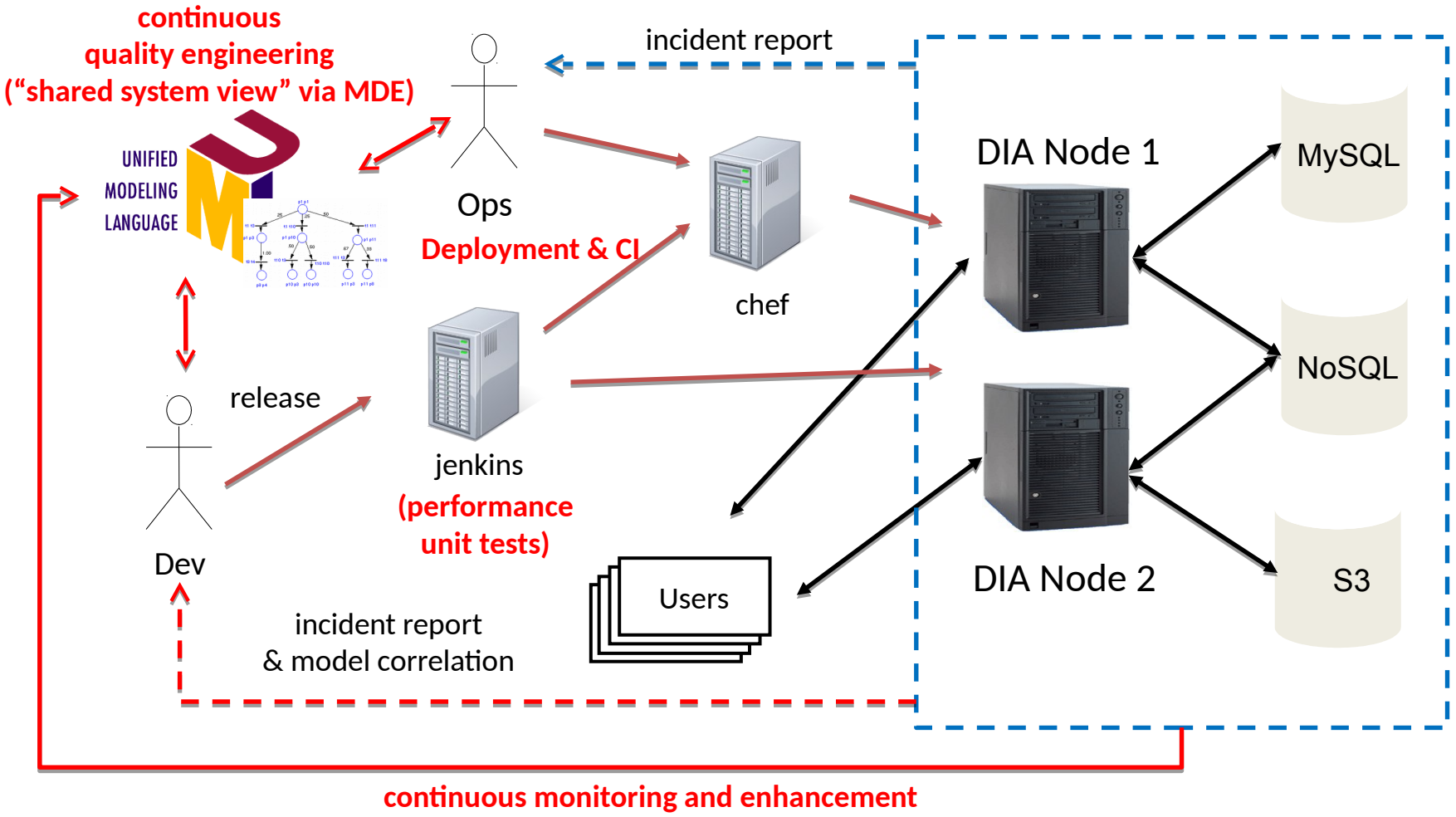
DevOps in DICE: Measurement



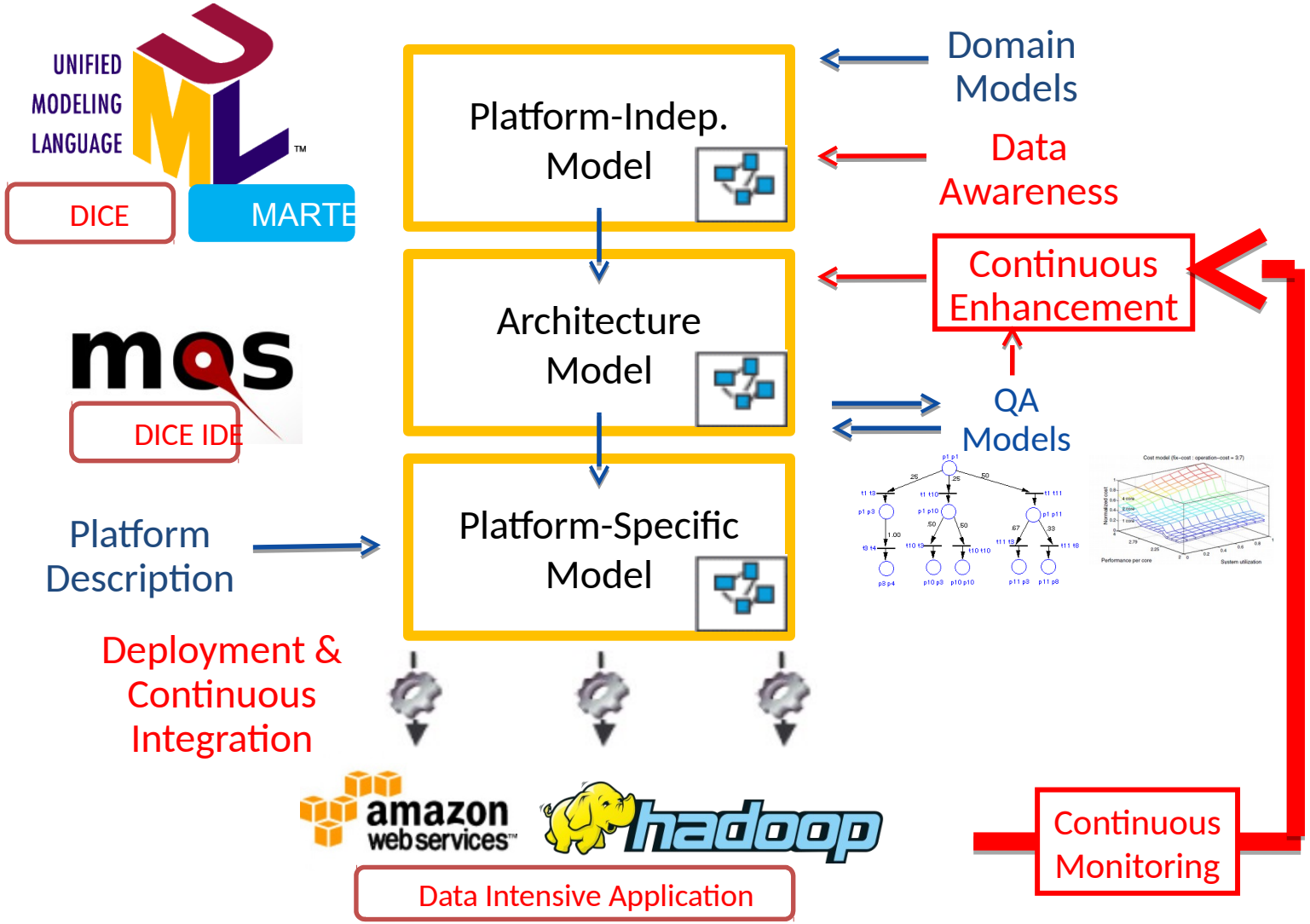
DevOps in DICE: Early-stage MDE



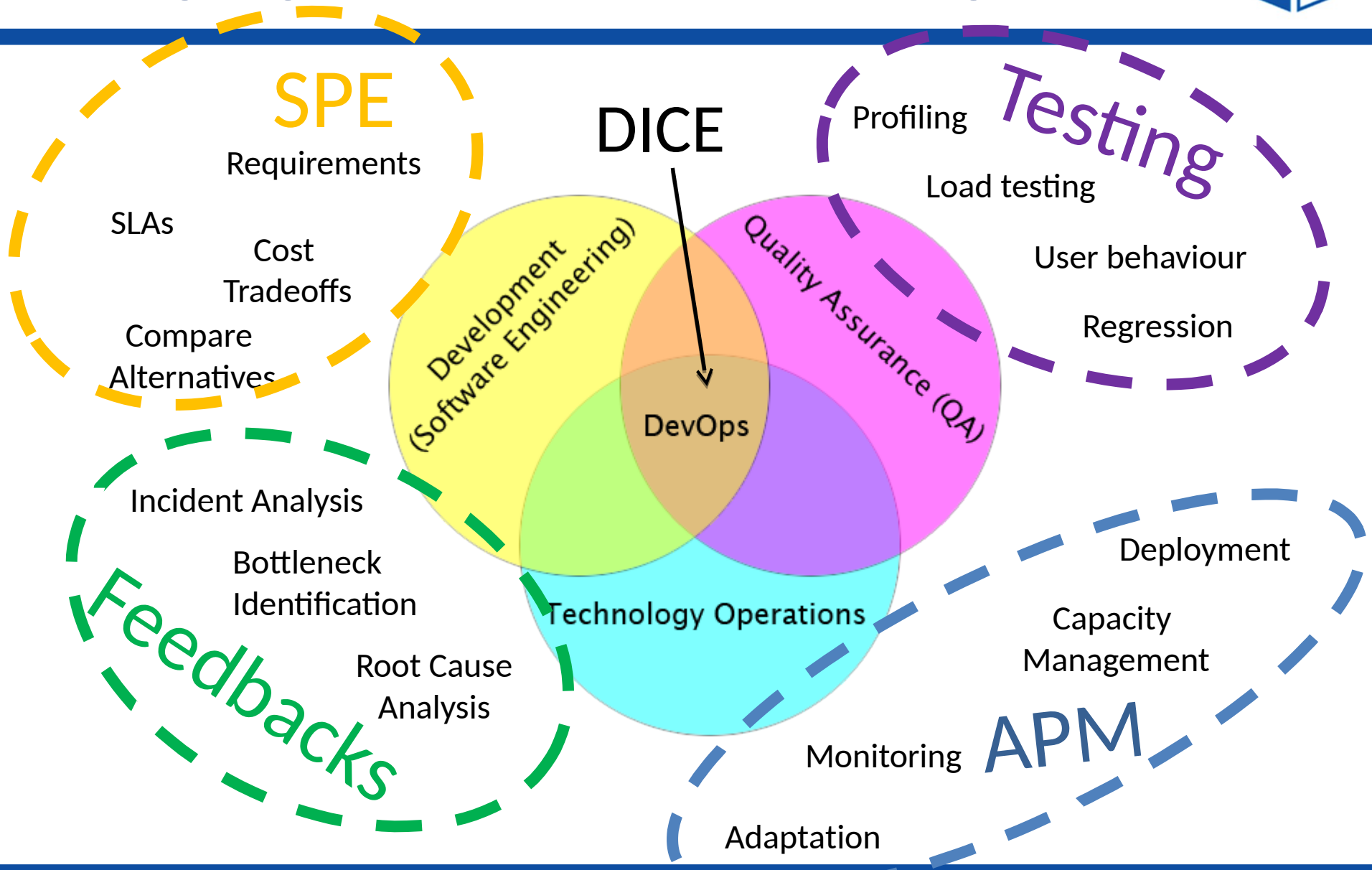
DevOps in DICE: Enhancement



DICE Integrated Solution

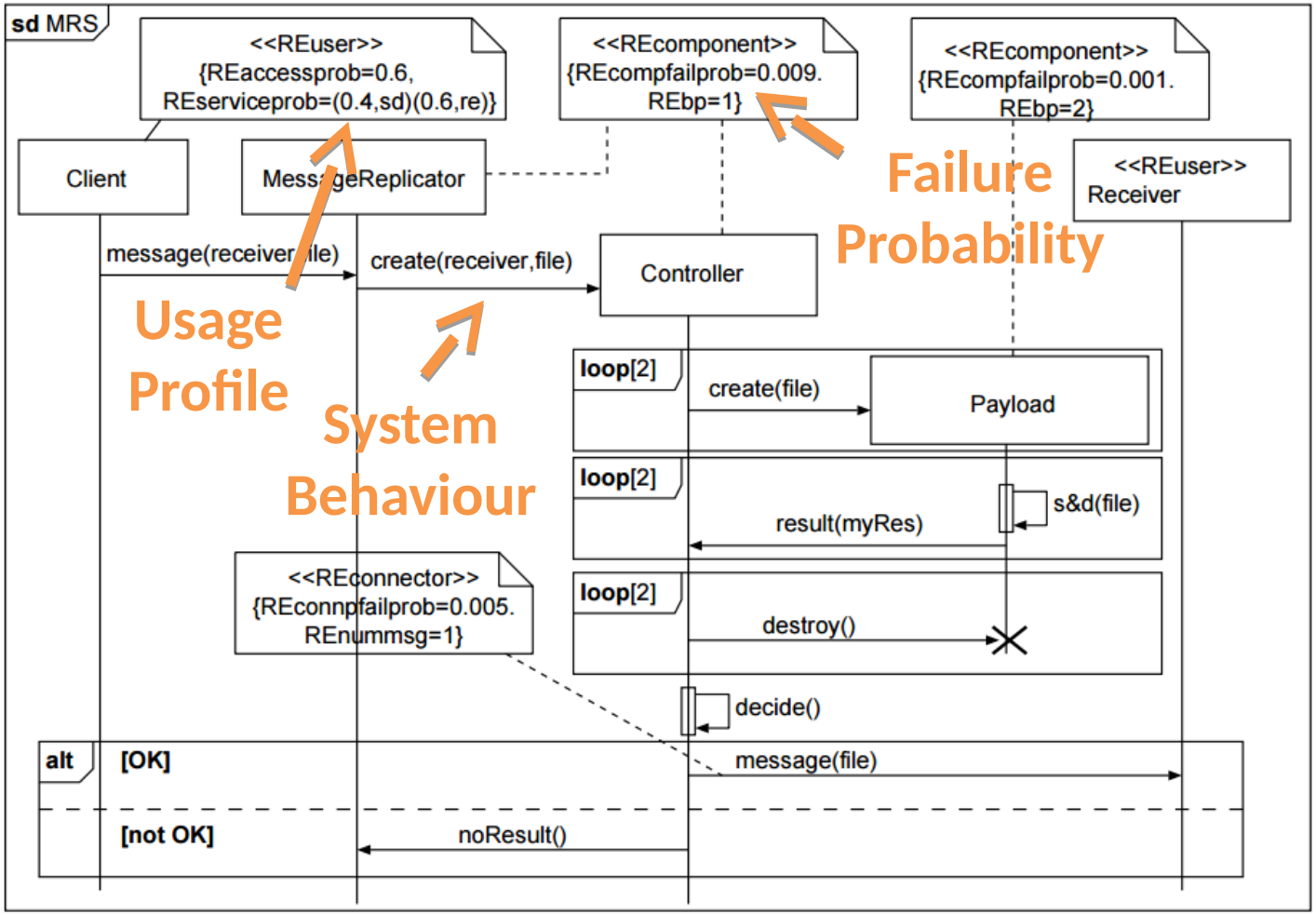


Bringing QA and DevOps together



Quality-Aware MDE

- UML MARTE profile, UML DAM profile, Palladio, …



Quality-Aware MDE

UNIFIED MODELING LANGUAGE™ MARTE

Platform-Indep. Model

Architecture Model

Platform-Specific Model

Domain Models

QA Models

Platform Description

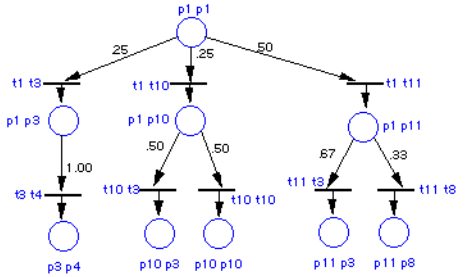
Code stub generation

amazon web services™

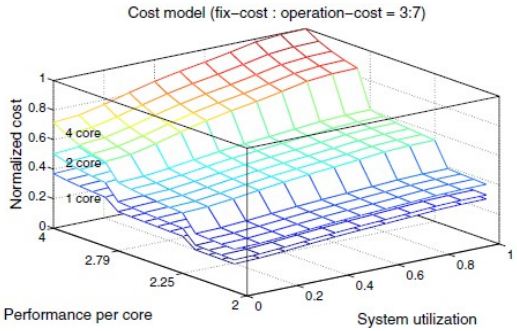
hadoop

Data Intensive Application

Simulation Tools



Cost Optimization Tools



Year 1 Milestones

Milestone	*Deliverables*
Baseline and Requirements - July 2015 [**COMPLETED**]	• State of the art analysis • Requirement specification • Dissemination, communication, collaboration and standardisation report • Data management plan
Architecture Definition - January 2016	• Design and quality abstractions • DICE simulation tools • DICE verification tools • Monitoring and data warehousing tools • DICE delivery tools • Architecture definition and integration plan • Exploitation plan

Demonstrators

Case study	Domain	Features & Challenges
Distributed data-intensive media system (ATC)	• **News & Media** • **Social media**	• Large-scale software • Data velocities • Data volumes • Data granularity • Multiple data sources and channels • Privacy
Big Data for e-Government (Netfective)	• **E-Gov application**	• Data volumes • Legacy data • Data consolidation • Data stores • Privacy • Forecasting and data analysis
Geo-fencing (Prodevelop)	• **Maritime sector**	• Vessels movements • Safety requirements • Streaming & CEP • Geographical information