TESTING BIG DATA APPLICATIONS AT DESIGN TIME AND RUNTIME WITH DICE

Matej Artač (XLAB), Ismael Torres (Prodevelop), Vasilis Papanikolaou (ATC), Giorgos Giotis (ATC)
MOTIVATION

Why DICE?
Outline

• About the DICE project
  • DevOps and testing
• DICE demonstrators
  • Posidonia Operations from Prodevelop
  • News Asset from ATC
• Conclusions
ABOUT THE DICE PROJECT

Developing Data-Intensive Cloud Applications with Iterative Quality Enhancements
Internet of Things feeds Big Data

• IoT is a growing source of data
  • sensors
  • data feeders
  • low power processing

• Big Data
  • storing large volumes of data
  • streaming and off-line analysis
  • finding value in data
What problems EU SMEs face?

- **Learning curves**
- **Initial prototyping**
- **Risk of failure**
- **Fast-paced market**

Customers with legacy data now ask for Big Data technologies

(+) others...
DICE: Quality-Aware DevOps for Big Data

Prototype

Design

“Quality-Aware DevOps for Big Data”

Deploy

Monitor

Enhance

hadoop

cassandra

Spark

STORM
What do we mean by Quality?

- Reliability
  - Availability
  - Fault-tolerance

- Efficiency
  - Performance
  - Costs

- Correctness
  - Privacy & security
  - Temporal metrics
DICE Workflow - Dev

DICE Eclipse IDE

Enhance

Design

Transform to Formal Models

Simulate & Verify

Optimize

User Conference on Advanced Automated Testing
DICE Workflow - Ops

Deploy

Quality Testing

Fault injection

Configuration Optimization

Enhance

DDSM

DICER (TOSCA)
DICE Workflow: Unified Toolchain

- Design
- Transform to Formal Models
- Quality Testing
- Simulate & Verify
- Fault injection
- Optimization
- Enhance
- Enhance
- Enhance
- Deploy
- DICE Eclipse IDE
- Optimize
DICE DEMONSTRATORS

Tools for real SMEs with real use cases
Demonstrators

- Posidonia Operations
  - Maritime Sector

- News Asset
  - News & Media Market

- Tax Fraud Detection Application
  - E-Government Market
Tutorial 1: Posidonia Operations
Tutorial 2: News Asset Platform

Content Creation & Editorial workflow

Editorial planning & Tasks
Aggregation & Verification
Multimedia Asset Management
Cross channel publishing

User Conference on Advanced Automated Testing
DICE- POSIDONIA DEMONSTRATOR

Ismael Torres
About Prodevelop

- Spanish SME with more than 20 years. 80 engineers
- Sectors (civil service, ports, agrairan, insurances)
- R&D (multiplatform mobility, Geospatial technologies, IoT, software engineering)
- 2011 start internationalization (sales delegation Casablanca and Sao paulo)

Highly specialized in the Maritime Industry
POSIDONIA OPERATIONS – Architecture Machines

Base Station

AIS Message

AIS Vessels Tracking

PARSER

Message Broker - Bus

CEP
Complex Event processor

WEB

User Conference on Advanced Automated Testing

© All rights reserved
Posidonia Operations

ALREADY IN PRODUCTION

• 18 ports monitored
• 2 cloud deployments
• 5 on-premises
• +30 artifacts running
POSIDONIA OPERATIONS — New Product

Goal for 2018

• New Product "POSIDONIA Operations ON THE CLOUD"

Main reasons of using DICE

1. Automatic deployment on the cloud
2. Automatic configuration
3. Improve business rules validation (semi-automatic)
HOW WE MADE IT (without DICE)

1. We take some AIS logs
2. Develop a new CEP geo-fencing rule
3. Evaluate (manually) the correctness rate
4. Evaluate (manually) the performance
5. Integrate and deploy in pre-prod
6. Check the throughput (manually)
7. Make some fixes in the rule
8. Iterate from 3. until pro deployment
POSIDONIA OPERATIONS - Challenges

- **Continuous delivery**
  - Continuous evolutionary maintenance
  - Different deployments and port configurations
- **Test - Quality**
  - Ensure reliability of the results, performance, etc.
  - Calculate the maximum throughput to evaluate scalability
    (what happens if the marine traffic increase? what if we add a new rule?)
  - Easily create test deployments to improve quality testing (now, manually and operator dependent)
  - Monitor system and application metrics to evaluate performance (need of application metrics)
  - Calculate correctness rate to evaluate reliability (are we detecting the events correctly?)
DICE METHODOLOGY- Scenarios

Standalone: Simulation
“I wish to focus on a specific DICE tool”

Partial Devops: Deployment + Fault Injection + Monitoring
I want to quickly deploy and tune the DIA
Simulation Tool – Problem to solve

• What happens if we add a new rule/increase marine traffic?

• Which is the maximum throughput of the current configuration?
Simulation Tool – Model

Use a DPIM (Platform Independent Model) to specify the system.
- MDA-UML Models enriched with DICE Profile.

Given a DPIM model, the simulation tool tells us the maximum throughput
- The model specifies:
  - Messages per second
  - Number of AIS Parser and CEP instances
  - Number and cost of each rule
Simulation Tool – Model

These annotations are only descriptive, and have been defined for clarity purposes. For the actual instantiation of the tagged values use the "Properties" tab in the "Properties" view.

User Conference on
Advanced Automated Testing

© All rights reserved
Simulation Tool – Model
Simulation Tool – Model

These annotations are only descriptive and have been defined for clarity purposes. For the actual instantiation of the tagged values use the ‘Profile’ tab in the ‘Properties’ view.
Simulation Tool – Model
Simulation Tool – Configuration

• SIMULATION PARAMETERS:
  • Cost estimation of each step (the actual cost is produced by the monitoring tool)
  • Message rate
Simulation Tool – Configuration

- OUTPUT:
  - Throughput
Simulation Tool – Results

• Scalability prediction at design time
• Iterative enhancement
• Multiple configurations
• Maximum throughput
Simulation Tool – Results

Main objectives of using DICE

1. Automatic deployment on the cloud
2. Automatic configuration
3. Improve business rules validation (semi-automatic)

• GOALS ACHIEVED:
  • PO.1 Simulation and predictive analysis of new event detection rules
  • PO.2 Scalability analysis
DICER-DEPLOYMENT- Problem to solve

• How to reduce the time required to deploy the application?
• how to make deployment easier and not require much technical knowledge?
DICER-DEPLOYMENT - Model

• The Deployment Model (DDSM) is based on the simulation results.
  • Models the physical deployment of artifacts on nodes

• The model specifies:
  • Number of nodes (virtual machines)
  • Number of instances of applications (AIS parser, CEP, Rabbit)
  • Network configuration, installation scripts, etc.

• These tools allow an automatic deployment of the solution, which is essential to perform automatic testing
DICER-DEPLOYMENT

Posidonia - DDSM
DICER-DEPLOYMENT

IDE – DICER

MODEL- DDSM

BLUEPRINT

Model to Text Transformation
DICER-DEPLOYMENT

DEPLOYMENT

Upload

Automatic Deployment
DICER-DEPLOYMENT – Results

• Deployment from scratch: 5 hours (it includes the initial learning curve, create the model, etc.)
• Next deployments: 20 minutes
• Effective time for a new deployment: minutes (automation)
• Continuous Deployment (productivity)
• Deployment diagrams (helps understand the system)
• Automatic Deployment (No experts needed, less error, faster, less cost)
Monitoring- Problem to solve

• How to control the resources used in Real time?
• How to control the Posidonia throughput in real time?
• How to analyze vessels routes inside the port?
• How to use the data obtained to improve the system and to make test?
Monitoring

• DICE monitoring platform (DMon) collects, stores, indexes and visualizes monitoring data in real-time from applications running on Big Data frameworks.
• Monitoring Posidonia Operations “running” provides performance metrics.
• These metrics can be used to redesign the simulation model to improve the results.
Monitoring
Monitoring
Monitoring – Results

- Control about resources used in real time
- Use log files to visualise/control the system in real time
- Use log files as an input for other DICE tools to automatize some test and improve the quality of the system
Fault injection tool - Problem to solve

• What happens to our product, if the system is overload?
• What happens if one of our components stops?
Fault injection tool

- DICE Fault Injection tool (FIT) is used to generate faults within VM.
- FIU installs a third party tool in the VM to produce faults.
Fault injection tool

- CPU load
Fault injection tool

- Memory Load
Fault injection tool - Results

• Test the system behaviour generating different types of faults (CPU and memory overloads)
Anomaly detection tool - Problem to solve

- *Is the time required to execute the different rules normal?*
- *Is there any anomaly in the detection of events?*
Anomaly detection tool

- **Anomaly detection** is the identification of items, events or observations which do not conform to an **expected pattern** or other items in a dataset.

- Anomaly detection is used in the UC to detect anomalies related with the **processing time required** for the different events (rules) that the system analyses.
Anomaly detection tool

Examples of the Rules execution cost

<table>
<thead>
<tr>
<th>Component</th>
<th>key</th>
<th>method</th>
<th>ms</th>
<th>ship</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIS_SENTENCE_LISTENER</td>
<td>2017-06-22T11:53:04.278Z</td>
<td>HANDLE_MESSAGE</td>
<td>209</td>
<td>211636100</td>
</tr>
<tr>
<td>SIMPLE_ANCHOR_OUT</td>
<td>2017-06-22T11:53:04.272Z</td>
<td>UPDATE_ACTIVE</td>
<td>710</td>
<td>305965000</td>
</tr>
<tr>
<td>STOP_OVER_OUT</td>
<td>2017-06-22T11:53:04.271Z</td>
<td>UPDATE_ACTIVE</td>
<td>652</td>
<td>305965000</td>
</tr>
<tr>
<td>SIMPLE_DOCK_STOP</td>
<td>2017-06-22T11:53:04.270Z</td>
<td>UPDATE_ACTIVE</td>
<td>293</td>
<td>305965000</td>
</tr>
<tr>
<td>STOP_OVER_IN</td>
<td>2017-06-22T11:53:04.270Z</td>
<td>UPDATE_ACTIVE</td>
<td>680</td>
<td>305965000</td>
</tr>
<tr>
<td>SIMPLE_DOCK_STOP</td>
<td>2017-06-22T11:53:04.269Z</td>
<td>UPDATE_ACTIVE</td>
<td>295</td>
<td>305965000</td>
</tr>
<tr>
<td>AIS_SENTENCE_LISTENER</td>
<td>2017-06-22T11:53:04.263Z</td>
<td>HANDLE_MESSAGE</td>
<td>9483</td>
<td>305965000</td>
</tr>
<tr>
<td>SIMPLE_ANCHOR_OUT</td>
<td>2017-06-22T11:53:04.262Z</td>
<td>UPDATE_ACTIVE</td>
<td>755</td>
<td>305965000</td>
</tr>
<tr>
<td>STOP_OVER_OUT</td>
<td>2017-06-22T11:53:04.261Z</td>
<td>UPDATE_ACTIVE</td>
<td>618</td>
<td>305965000</td>
</tr>
<tr>
<td>STOP_OVER_IN</td>
<td>2017-06-22T11:53:04.260Z</td>
<td>UPDATE_ACTIVE</td>
<td>686</td>
<td>305965000</td>
</tr>
<tr>
<td>SIMPLE_DOCK_STOP</td>
<td>2017-06-22T11:53:04.259Z</td>
<td>UPDATE_ACTIVE</td>
<td>303</td>
<td>305965000</td>
</tr>
<tr>
<td>AIS_SENTENCE_LISTENER</td>
<td>2017-06-22T11:53:04.246Z</td>
<td>HANDLE_MESSAGE</td>
<td>199</td>
<td>225366000</td>
</tr>
<tr>
<td>AIS_SENTENCE_LISTENER</td>
<td>2017-06-22T11:53:04.245Z</td>
<td>HANDLE_MESSAGE</td>
<td>766</td>
<td>224161160</td>
</tr>
</tbody>
</table>

Definition of anomalies

- AIS_SENTENCE_LISTENER  
  ms > 20000
- RETRACT_OLD_AISGEOMDATA  
  ms > 60000
- SIMPLE_ANCHOR_IN  
  ms > 30000
- SIMPLE_ANCHOR_OUT  
  ms > 2000
- SIMPLE_DOCK_STOP  
  ms > 2000
- STOP_OVER_IN  
  ms > 2000
- STOP_OVER_OUT  
  ms > 2000
- FIRE_ALL_RULES  
  ms > 30000
- HANDLE_MESSAGE  
  ms > 30000
- RETRACT  
  ms > 30000
Anomaly detection tool - Results

- The ADT detects anomalies related with the cost execution, this cost impact directly in the performance of the system.
- The ISF method did not detect all the anomalies (15.5 % from the original 22.4 %) but it had a relatively small false positive count (accuracy of 93.4%).
- It is possible in increase the accuracy of the method by considering a bigger set of data.

<table>
<thead>
<tr>
<th>Metric</th>
<th>CEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labelled anomalies</td>
<td>1447</td>
</tr>
<tr>
<td>Detected anomalies</td>
<td>999</td>
</tr>
<tr>
<td>False positives</td>
<td>58</td>
</tr>
<tr>
<td>Good Anomalies</td>
<td>941</td>
</tr>
<tr>
<td>Percentage labelled</td>
<td>22,4%</td>
</tr>
<tr>
<td>Percentage detected</td>
<td>15,5%</td>
</tr>
<tr>
<td>Accuracy</td>
<td>93,4%</td>
</tr>
</tbody>
</table>
CONCLUSIONS – Benefits of using DICE

1. Automatic deployment on the cloud
   - DICER + Deployment
     - Fully automated deployment in minutes

2. Automatic configuration
   - Simulation + Monitoring:
     - See the effect of different configuration at Design phase

3. Improve business rules validation (semi-automatic)
   - Anomaly Detection + Trace Checking Tool
     - Detect unexpected patterns in the events detected.
   - Fault Injection Tool
     - Evaluate system stability against unexpected failures
THE NEWS ASSET DEMONSTRATOR

Vasilis Papanikolaou, George Giotis
ATHENS TECHNOLOGY CENTER

Who we are
ATC MILESTONES: over the last 30 years

- 1987 - 1996: Fourlis Group (IKEA, intersport)
- 2007: BPO & CS Unit
- 2008: Partner of NESSI ETP
- 2012-2013: BPO division sold to Openbet (UK)
- 2014: TruthNest Social Media Verification
- 2015: ASA Buyout
- 2016 - 2017: Outsourcing & Custom Software Development
Departments & Activities

Media & Content Management

Innovation Lab

Web & Mobile Applications

Custom IT Services

User Conference on Advanced Automated Testing
NEWS ASSET

What News Asset is
News Asset Platform Description

An end-to-end multimedia cross-channel suite for an evolving News Agency
Handles large volumes of media asset (real time and/or archived news content)
Datasets composed of media items such as text, images, reports, articles, videos, etc.

Services that supports the whole life-cycle of managing a news item:
- planning,
- creating, gathering and searching,
- editing,
- producing, distributing and archiving

Old –fashioned non-cloud software
Based on .Net
Storage intensive
Data intensive
Computation services
Centralized architecture

Fat-Client, Business Server and Data Base Server
News reception may vary from 5000 to 20000 items per day
News distribution may be 50000 to 200000 items per day
The default import format is based on standard NewsML format

Concept
Workflow
Keywords
Technical insight
Challenges

- Uncountable media items are produced by heterogeneous sources
- Media eye witnessing reports (e.g. Twitter)
- Immediate access and management of real time news items is crucial
- Handle streams of data, serve a wide range of workloads
Challenges

• Refactoring of the old-fashioned engine
  • related to cloud processing and Big Data technologies

• Reconfiguration
  • revise obsolete architecture with respect to quality-driven metrics

• Manage complexity
  • real-time responsiveness for temporal peaks of high computational demand
NEWS ORCHESTRATOR

The Solution
News Orchestrator Architecture
Trending-Topic Detector Storm Topology

Spouts
- Twitter Streaming API
- Redis Pub/Sub

Entities Extractor → MinHash Extractor → Terms Extractor → Solr Indexer → SOLR

Terms Rolling Count → Terms Intermediate Ranker → Terms Final Ranker → Topics MongoDB Writer → Topics Labeler

MinHash Rolling Count → MinHash Intermediate Ranker → MinHash Final Ranker → Clusters MongoDB Writer → Clusters Labeler

Mongo DB
NEWS ASSET TUTORIAL

News Asset Experience with DICE
DevOps Lifecycle

- **Development Environment**
  - DICER
  - DICE DS
  - Jenkins
  - Continuous Integration

- **Test Environment**
  - Continuous Testing
  - DICE Quality Testing

- **Stage Environment**
  - DICE Simulation
  - DICE Configuration Optimization
  - Continuous Delivery

- **Production Environment**
  - DICE Monitoring
Simulation

**Motive:** Predict the behavior of the system prior to the deployment in the cloud.
Simulation workflow (1)
Simulation workflow (2)
Simulation workflow (3)
Simulation workflow (4)
Simulation workflow (5)
Simulation workflow (6)
Simulation workflow (7)
Simulation workflow (8)
Simulation workflow (9)
Simulation workflow (10)
Simulation workflow (11)
Simulation workflow (12)
Simulation workflow (13)
Simulation workflow (14)
Simulation Benefit

- Configure the Storm design to specific execution context
- Detect performance bottlenecks
- Less than 20% with regard to the prediction error on the utilization metric
Configuration Optimization (CO)

Motive: Find optimal configuration settings wrt computational budget restrictions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>drpc.invocations threads</td>
<td>64</td>
</tr>
<tr>
<td>drpc.max_buffer_size</td>
<td>120000000000</td>
</tr>
<tr>
<td>drpc.port</td>
<td>7772</td>
</tr>
<tr>
<td>drpc.queue.size</td>
<td>128</td>
</tr>
<tr>
<td>drpc.request.timeout.secs</td>
<td>60000</td>
</tr>
<tr>
<td>drpc.worker threads</td>
<td>64</td>
</tr>
<tr>
<td>java.library.path</td>
<td>&quot;/usr/local/libLOPT/local/lib/ser/ser&quot;</td>
</tr>
<tr>
<td>log4j.users</td>
<td>null</td>
</tr>
<tr>
<td>log4j.rootlogger.appendender.name</td>
<td>&quot;AI&quot;</td>
</tr>
<tr>
<td>log4j.rootlogger.chillops</td>
<td>&quot;.\Res\Event&quot;</td>
</tr>
<tr>
<td>log4j.rootlogger.cleanup.age.mins</td>
<td>1040000</td>
</tr>
<tr>
<td>log4j.rootlogger.max.per.worker.log.size.mb</td>
<td>200000000</td>
</tr>
<tr>
<td>log4j.rootlogger.max.sum.worker.log.size.mb</td>
<td>400000000</td>
</tr>
<tr>
<td>log4j.rootlogger.port</td>
<td>80000</td>
</tr>
<tr>
<td>nimbus.ldbstore.class</td>
<td>&quot;org.apache.storm.ldbstore.LocalLdbstore&quot;</td>
</tr>
<tr>
<td>nimbus.ldbstore.expiration.secs</td>
<td>60000</td>
</tr>
<tr>
<td>nimbus.chillops</td>
<td>&quot;\Res\Event&quot;</td>
</tr>
<tr>
<td>nimbus.cleanup.inbox.freq.secs</td>
<td>6000000</td>
</tr>
<tr>
<td>nimbus.code.sync.freq.secs</td>
<td>110000</td>
</tr>
<tr>
<td>nimbus.credential.renewers.freq.secs</td>
<td>60000000000</td>
</tr>
<tr>
<td>nimbus.file.copy.expiration.secs</td>
<td>4000000</td>
</tr>
<tr>
<td>nimbus.impersonation.authorizer</td>
<td>&quot;org.apache.storm.security.auth.authorizer.ImpersonationAuthorizer&quot;</td>
</tr>
<tr>
<td>nimbus.inbox.jar.expiration.secs</td>
<td>360000</td>
</tr>
<tr>
<td>nimbus.monitor.freq.secs</td>
<td>10</td>
</tr>
<tr>
<td>nimbus.queues.size</td>
<td>10000000</td>
</tr>
<tr>
<td>nimbus.seeds</td>
<td>[1]</td>
</tr>
<tr>
<td>nimbus.supervisor.timeout.secs</td>
<td>300000</td>
</tr>
<tr>
<td>nimbus.task.launch.secs</td>
<td>120000000000</td>
</tr>
<tr>
<td>nimbus.task.timeout.secs</td>
<td>30</td>
</tr>
<tr>
<td>nimbus.thrift.max_buffer_size</td>
<td>200000000000</td>
</tr>
</tbody>
</table>
Plugin Configuration, Parameter Selection

- Save current configurations
- Load saved configurations

- Username: gipnd
- Password: ********
- Jenkins URL: http://adrastes.atc.gr:8080
- Job Name: /job/topic-detector/
- Build Token:

**Parameter Selection**

- **Parameter**
  - topology.error.throttle.interval.secs
  - topology.bolts.batch.emit.interval.millis
  - topology.disruptor.wait.timeout.millis
  - topology.disruptor.batch.size
  - topology.disruptor.batch.timeout.millis
  - topology.disable.acknowledge.messaging
  - topology.state.checkpoint.interval.msecs
  - topology.max.spout.pending
  - topology.adder.executors
  - topology.backup.fails

- **Add Parameters**
  - **Parameter**
    - topology.executor.receive.buffer.size: Integer 1024 2048
    - topology.min.replication.count: Integer 1 10
    - topology.worker.shared.thread.pool.size: Integer 1 20
    - topology.max.task.parallelism: Integer 1 50
Service & Experiment Configuration

Plugin Config | Parameter Selection | Service Config | Experiment Config | App Config | Experiments

- **servicename**: Kafka Broker
  - **URL**: http://10.151.64.59:9092
  - **username**: glcpmd
  - **password**: ********
  - **servicename**: Monitoring
  - **URL**: http://10.151.64.45:3001

- **Add Service**

- **Noise**: 1e-5
- **Number of Iterations**: 100
- **Initial Design**: 2
- **Save Folder**: /integrated/reports
- **Config Folder**: /integrated/config
- **Summary Folder**: /integrated/summary
- **Blueprint**: storm-openstack.yaml
- **Config**: topology.yaml
- **Topic**: json-topic
- **Sleep Time**: 10000
- **Metric Poll**: 1000
- **Exp Time**: 300000
- **Replication**: 4
Application Configuration, Experiments
CO Benefit

- **Throughput improvement:**
  - more than twice compared to the default configuration
  - achieved after only 100 iterations
  - took \( 100 \times 10 \text{ min} = 16\text{h} \) to run

- **No expert needed!!**
Quality Testing (QT)

- **Motive:** Stress test the capacity of the ‘trending topic detection’ topology

- Run multiple iterations/experiments
  - Inject constantly increasing stream load
    - With respect to ‘complete latency’ metric threshold

- Evaluate the maximum input rate
  - Twitter Streaming API has limitations
    - Twitter Paid API

```java
TopologyBuilder builder = new TopologyBuilder();
/* Create QT-LIB's Spout factory */
QTLoadInjector qt = new QTLoadInjector();
/* Obtain a spout to inject at prescribed rates specified */
RateSpout qts = qt.getRateSpout();
qts.setArrivalMode(RateSpout.ArrivalMode.ParseCount);
qts.setRateScaler(scaler);
qts.setArrivalFile("counts.txt");
qts.setDataMode(RateSpout.DataMode.ParseJSON);
qts.setBinaryBDoc(true);
qts.setDataFile("test.json");
```
Quality Testing (QT) - Scenario

1. Use QT spouts
2. Integrate QT
3. Set up automatic builds on code updates
4. Deploy topology on Storm cluster
5. Monitor Storm metrics (spout's complete latency, bolts' capacity)
6. Continuous feedback
QT Benefit
CONCLUSION

Final thoughts and remarks
DICE – Automated Testing Encapsulated

• Methodology for using DevOps, testing tools
• Uses a mix of known approaches
  • Formal model analyses (Simulation Tools)
  • Machine Learning approaches (Anomaly Detection, Configuration Optimization)
• Efficient techniques of optimization
• Works offline on a design and online on the runtime
Will it be useful for you?

<table>
<thead>
<tr>
<th>DICE Tool</th>
<th>We use models in our design all the time</th>
<th>UML? No, thank you</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>DICER Deployment</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>Configuration Optimization</td>
<td>✓</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>Quality Testing</td>
<td>✓</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>Fault Injection</td>
<td>✓</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>Monitoring</td>
<td>✓</td>
<td>✓ ✓</td>
</tr>
</tbody>
</table>
Follow us!

- Code repository: [https://github.com/dice-project](https://github.com/dice-project)

- Matej Artač: matej.artac@xlab.si  @matej_artac
- Ismael Torres: itorres@prodevelop.es
- Vasilis Papanikolaou: v.papanikolaou@atc.gr
- George Giotis: g.giotis@atc.gr
- Damian A. Tamburri: damianandrew.tamburri@polimi.it
Acknowledgement

DICE
Horizon 2020 Research & Innovation Action
Grant Agreement no. 644869
http://www.dice-h2020.eu

Funded by the Horizon 2020 Framework Programme of the European Union