DICE: Quality-Driven Development of Data-Intensive Cloud Applications

Overview and goals

- MDE often features quality assurance (QA) techniques for developers

- How should quality-aware MDE support data-intensive software systems?
  - Existing models and QA techniques largely ignore properties of data
  - Characterize the behavior of new technologies

- DICE: a quality-aware MDE methodology inspired by DevOps for data-intensive cloud applications
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]

- European call for software 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 trends in software development (Big data, Cloud, DevOps ...)

©DICE MiSE 2015@ICSE, Florence, Italy, May 17 2015
DataInc example

- DataInc is a small software vendor selling cloud-based environmental software

- DICEnv, a warning system for floods in rural regions
  - monitoring local environmental conditions
  - fetching precipitations data from satellite image stream

- DICEnv exploits Big Data technologies and cloud capacity for online water simulations and MapReduce for batch processing of historical data

- DICEnv is a critical system:
  - is expected to remain up 24/7
  - should quickly ramp up data intake rates, as well as memory and compute capacities, to update more frequently the hazard management control room
DataInc example

- The contract requires delivering an initial version of DICEnv within 3 months serving a small area, increasing coverage on a monthly basis.

- Challenges:
  - How to implement a complex cloud application in such a short time?
  - How to satisfy all the quality requirements?
  - What architecture should be adopted?
Quality-Aware MDE Today

- Platform-Indep. Model
- Architecture Model
- Platform-Specific Model

Domain Models

QA Models

Analytical Models

Cost-Quality Models

Platform Description

Code generation

C++
Java
C#
Quality-Aware MDE Today

Platform-Indep. Model

Architecture Model

Platform-Specific Model

Issues PIM layer:
- static characteristics of data
- dynamic characteristics of data
- data dependencies

DICEnv modeling issues:
- individual dependencies between components and data streams
- relationships between compute and memory requirements
- lack of an explicit annotation for data characteristics

Platform Description

Code generation

©DICE

MiSE 2015@ICSE, Florence, Italy, May 17 2015
Quality-Aware MDE Today

Issues at PSM layer:
- heterogeneity of Big Data technologies
- automatic translation of PSM models into deployment plans

QA tools limitations:
- contention at processing resources with limited features for memory consumption
- fork and joining are complex to be described analytically preserving tractability
An Holistic Approach: DICE

- Platform-Indep. Model
- Architecture Model
- Platform-Specific Model

- Domain Models
- Data Awareness
- Continuous Validation
- QA Models

- Continuous Monitoring

- Big Data
- Amazon Web Services
- Hadoop

- Deployment & Continuous Integration
- Platform Description
- DICE IDE
- DICE
- MARTE

©DICE
MiSE 2015@ICSE, Florence, Italy, May 17 2015
Embracing DevOps

- Software development process is evolving
  - Developer: “I want to change my code”
  - Operator: “I want systems to be stable”
    - ...but code changes are the cause of most instabilities!

- DevOps closes the gap between Dev and Ops
  - Lean release cycles with automated tests and tools
  - Deep modelling of systems is the key to automation
Embracing DevOps

- QA must become lean as well
  - Continuous quality checks and model versioning
- Modelling of the operations
  - Dev needs awareness of infrastructure and costs
- Continuous feedback
  - Forward and backward model synchronisation
  - Tracking of self-adaptation events (e.g. auto-scaling)
- Big data coming from continuous monitoring
  - QA has its own Big data, use machine learning?
Benefits

- Tackling skill shortage and steep learning curves
  - Data-aware methods, models, and OSS 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
DICE Platform Independent Model (DPIM)
DICE Platform, Technology and Deployment Specific Model (DDSM)
DICE Profile: PIM Level

- Functional approach to data to be expanded
  - Data dependencies
    - graph relationships between data, archives and streams
- QA focuses on quantitative aspects of data
  - Static characteristics of data
    - volumes, value, storage location, replication pattern, consistency policies, data access costs, known schedules of data transfers, data access control / privacy, ...
  - Dynamic characteristics of data
    - cache hit/miss probabilities, read/write/update rates, burstiness, ...
DICE Profile: PSM Level

- Need for technology-specific abstractions
  - Hadoop: Number of mappers and reducers, ...
  - In-memory DBs: Peak memory and variable threading
  - Streaming: merge/split/operators, networking, ...
  - Storage: Supported operations, cost/byte, ...
  - NoSQL: Consistency policies, ...

- Generation of deployment plan
  - Proposed Chef + TOSCA extension

- Interest is both on private and public clouds
DICE QA: Quality Dimensions

- Reliability
  - Availability
  - Fault-tolerance

- Efficiency
  - Performance
  - Time behaviour
  - Costs

- Safety
  - Risk of harm
  - Privacy & data protection
DICE QA: Tools

- **Discrete-event simulation**: assess reliability and efficiency in Big Data applications, accounting for stochastic evolution of the environment
  - stochastic Petri nets or queueing networks, rely on simulation

- **Formal verification tools**: assess safety risks in Big Data applications, e.g. find design flaws causing order and timing violations in message and state sequences
  - temporal logic formulae and bounded model checking, satisfiability modulo theories solvers
  - quantifier-elimination techniques to extend temporal logic-based verification
DICE QA: Tools

- **Architecture optimization tool**: find architectural improvements to optimise costs and quality
  - decomposition-based analysis approach
  - resort to fluid approximation of stochastic models

- **Feedback analysis**: automated extraction from the monitored data of key parameters required to define simulation and verification models
  - extract model parameters through log mining and statistical estimation methods
  - breakdown resource consumption into its atomic components on the end-to-end path of requests
○ Horizon 2020 Research & Innovation Action
  ▪ Quality-Aware Development for Big Data applications
  ▪ Feb 2015 - Jan 2019, 4M Euros budget
  ▪ 9 partners (Academia & SMEs), 7 EU countries