CONTINUITY: AUTOMATED PERFORMANCE TESTING IN CONTINUOUS SOFTWARE ENGINEERING

09.11.2017 | HENNING SCHULZ, ANDRÉ VAN HOORN, CHRISTOPH HEGER, ALEXANDER WERT

SPONSORED BY THE

University of Stuttgart

Federal Ministry of Education and Research

NOVATEC
Why Performance Matters

Ellen DeGeneres’ Selfie Crashes Twitter

- 634,000 retweets in half an hour (more than 3 millions overall)
- Disrupted Twitter for more than 20 minutes
Why Performance Matters

Ellen DeGeneres‘ Selfie Crashes Twitter

- 634,000 retweets in half an hour (more than 3 millions overall)
- Disrupted Twitter for more than 20 minutes

A Real-World Example

Load Testing

Realistic user behavior

login

logout

Real user behavior

login

logout

OutOfMemoryError

Development Cycle

Go-live

Motivation

Project Overview

Approach

Status Quo & Vision

Conclusion
Problems of the Classic Load Testing Approach

- High manual effort for maintaining load tests
- Load tests need much time to execute
- There are no suitable load tests
- Complex analysis of performance regressions

Motivation | Project Overview | Approach | Status Quo & Vision | Conclusion
Continuous Delivery Pipelines – How Problems get Worse

High manual effort for maintaining load tests vs. Pipeline automation
Load tests need much time to execute vs. Fast & frequent releases

Implementation
Build
Functional Testing
Performance Testing

Service-focus requires multiple tests vs. There are no suitable load tests
Complex load tests for every release impossible vs. Complex analysis of performance regressions

Motivation
Project Overview
Approach
Status Quo & Vision
Conclusion
ContinuITy Goals

- Automated Extraction and Evolution of Load Tests
- Modularization and Semantification of Load Tests
- Detection of Performance Regressions
- Automated Diagnosis of Performance Regressions

Status quo

Motivation

Project Overview

Approach

Status Quo & Vision

Conclusion
ContinuITy Key Data

- **Start Date**: Sep. 2017
- **Duration**: 2 Years
- **Sponsor**: Consortium
- **Motivation**: Project Overview
- **Approach**: Status Quo & Vision
- **Conclusion**:
System Workflow

Load Test Generation & Execution
- Monitoring Data
- Workload Model Evolution
- Contextual Information
- Workload Model Selection
- Automated Execution
- Test Result

Regression Detection & Diagnosis
- Regression Detection
- Regression Analysis
- Usage Profile + System changes
- Result Preparation
- Result Presentation
- Feedback

Motivation  Project Overview  Approach  Status Quo & Vision  Conclusion
System Architecture

Monitoring
- inspectIT
- Kieker
- Zipkin

Independent Format (OPEN.xtrace)

Load Test Evolution
- WESSBAS
- Modularization
- Model Evolution
- Test Repo
- Model

Regression Diagnosis
- Algorithms
- Detection
- Preparation

Test Execution
- BenchFlow
- Faban
- Cassandra

New Components

Motivation
- Project Overview
- Approach
- Status Quo & Vision
- Conclusion
Load Test Evolution

Motivation  Project Overview  Approach  Status Quo & Vision  Conclusion
Load Test Evolution

Motivation  Project Overview  Approach  Status Quo & Vision  Conclusion
Load Test Evolution

Motivation

API

Project Overview

Workload Model

Approach

Annotation Model

Status Quo & Vision

Conclusion

Test Environment

Production Environments

Transfer load

Executable Load Test

Monitoring Data

Workload Model Generator
Example: Annotation of a WESSBAS Workload Model

**WEESBAS Model:**

- Workload Model
  - Constant Workload Intensity 800
  - Application Model
- Session Layer EFSM
  - Application State ASld5_CartGET
  - Application State ASld8_CartPOST
  - Application State ASld3_DVDGET
  - Application Exit State $5
- Guard Action Parameter List
- Behavior Mix
  - Behavior Model gen_behavior_model0
  - Markov State MSld3_CartGET
  - Markov State MSld4_CartPOST
  - Markov State MSld5_DVDGET
  - Behavior Model Exit State MSld1
- Behavior Model gen_behavior_model1
- Behavior Model gen_behavior_model2

**System Model:**

```plaintext
<http>
&CartGET domain: localhost
port: 8080
path: /dvdstore/checkout
method: GET
encoding: <no-encoding>
protocol: http
parameters:
  - &conversationId parameter-type: REQ_PARAM
    name: conversationId
headers: []
</http>

<http>
&CartPOST domain: localhost
... parameters:
  - &ViewState parameter-type: REQ_PARAM
    name: javax.facesViewState
</http>

<http>
&DVDGET domain: localhost
... ...
```

**Annotation Model:**

```plaintext
inputs:
- !<direct>
  &conversationIdInput data:
    - 7
    - 8
- !<extracted>
  &ViewStateInput extractions:
    - from: CartGET
      pattern: ViewState" value="(.*""
interface-annotations:
- interface: CartGET
  parameter-annotations:
    - parameter: conversationId
      input: *conversationIdInput
- interface: CartPOST
  parameter-annotations:
    - parameter: ViewState
      input: *ViewStateInput
... ...
```
Example: Annotation of a WESSBAS Workload Model

**WEESBAS Model:**
- Workload Model
  - Constant Workload Intensity 800
- Application Model
- Session Layer EFSM
  - Application State ASId5_CartGET
  - Application State ASId8_CartPOST
  - Application State ASId3_DVDGET
  - Application Exit State
  - Guard Action Parameter List
- Behavior Mix
  - Behavior Model gen_behavior_model0
  - Markov State MSId3_CartGET
  - Markov State MSId4_CartPOST
  - Markov State MSId5_DVDGET
  - Behavior Model Exit State MSId1
  - Behavior Model gen_behavior_model1
  - Behavior Model gen_behavior_model2

**System Model:**
&dvdstore interfaces:
- ![http]
  &CartGET domain: localhost
  port: 8080
  path: /dvdstore/checkout
  method: GET
  encoding: <no-encoding>
  protocol: http
  parameters:
    - &conversationId parameter-type: REQ_PARAM
      name: conversationId
      headers: []
    ![http]
    &CartPOST domain: localhost
      ...
    parameters:
      - &ViewState parameter-type: REQ_PARAM
        name: javax.faces.ViewState
    ![http]
    &DVDGET domain: localhost
    ...

**Annotation Model:**
inputs:
- ![direct]
  &conversationIdInput data:
    - 7
    - 8
- ![extracted]
  &ViewStateInput extractions:
    - from: CartGET
      pattern: ViewState" value="(.*)"
  interface-annotations:
    - interface: CartGET
      parameter-annotations:
        - parameter: conversationId
          input: *conversationIdInput
        - interface: CartPOST
          parameter-annotations:
            - parameter: ViewState
              input: *ViewStateInput
            ...

Motivation ✔ Project Overview ✔ Approach ✔ Status Quo & Vision ✔ Conclusion ✔
Example: Annotation of a WESSBAS Workload Model

WESSBAS Model:

System Model:

&dvdstore interfaces:
- ![direct]
  &CartGET domain: localhost
  port: 8080
  path: /dvdstore/checkout
  method: GET
  encoding: <no-encoding>
  protocol: http
  parameters:
  - &conversationIdInput parameter-type: REQ_PARAM
    name: conversationId
  headers: []
- ![direct]
  &CartPOST domain: localhost
  ... parameters:
  - &ViewStateInput parameter-type: REQ_PARAM
    name: javax.faces.ViewState
- ![direct]
  &DVDGET domain: localhost
  ...

Annotation Model:

inputs:
- ![direct]
  &conversationIdInput data:
    - 7
    - 8
- ![extracted]
  &ViewStateInput extractions:
  - from: CartGET
    pattern: ViewState" value="(.*)"
  interface-annotations:
    - interface: CartGET
      parameter-annotations:
        - parameter: conversationId
          input: *conversationIdInput
        - interface: CartPOST
          parameter-annotations:
            - parameter: ViewState
              input: *ViewStateInput
            ...

Motivation  Project Overview  Approach  Status Quo & Vision  Conclusion
Example: Annotation of a WESSBAS Workload Model

**WESSBAS Model:**

- Workload Model
- Constant Workload Intensity 800
- Application Model
- Session Layer EFSM
- Application State ASId5_CartGET
- Application State ASId8_CartPOST
- Application State ASId3_DVDGET
- Application Exit State 5
- Guard Action Parameter List
- Behavior Mix
- Behavior Model gen_behavior_model0
- Markov State MSId3_CartGET
- Markov State MSId4_CartPOST
- Markov State MSId5_DVDGET
- Behavior Model Exit State ASId1
- Behavior Model gen_behavior_model1
- Behavior Model gen_behavior_model2

**System Model:**

&dvdstore interfaces:
- ![http]
  &CartGET domain: localhost
  port: 8080
  path: /dvdstore/checkout
  method: GET
  encoding: <no-encoding>
  protocol: http
  parameters:
  - &conversationId parameter-type: REQ_PARAM
    name: conversationId
  headers: []
- ![http]
  &CartPOST domain: localhost
  ...
  parameters:
  - &ViewState parameter-type: REQ_PARAM
    name: javax.faces.ViewState
- ![http]
  &DVDGET domain: localhost
  ...

**Annotation Model:**

inputs:
- ![direct]
  &conversationIdInput data: 7
- 8
- ![extracted]
  &ViewStateInput extractions:
  - from: CartGET
    pattern: ViewState" value="(.*)"

interface-annotations:
- interface: CartGET
  parameter-annotations:
    - parameter: conversationId
      input: *conversationIdInput
- interface: CartPOST
  parameter-annotations:
    - parameter: ViewState
      input: *ViewStateInput

---

Motivation | Project Overview | Approach | Status Quo & Vision | Conclusion
Example: Annotation of a WESSBAS Workload Model

JMeter Transformation

Motivation  Project Overview  Approach  Status Quo & Vision  Conclusion
Example: Annotation of a WESSBAS Workload Model

JMeter Transformation

Motivation  Project Overview  Approach  Status Quo & Vision  Conclusion
### Vision: Workload Semantification

**Measured in production:**

- Workload group 1
- Workload group 2
- Response time
- Temperature
- Marketing campaign

**Prediction:**

- "Select workload at mondays between 1 and 3 pm"
- "Select workload during marketing campaign X"
- "Select workload causing highest response times"
- "Select workload of tomorrow"

---

**Motivation**

**Project Overview**

**Approach**

**Status Quo & Vision**

**Conclusion**
Where to Find Us

- continuity-project.github.io/
- https://github.com/ContinuITy-Project
- continuity-project.atlassian.net
- @ContinuITy_proj
Summary: ContinuITy Goals

- Automated Extraction and Evolution of Load Tests
- Modularization and Semantification of Load Tests
- Detection of Performance Regressions
- Automated Diagnosis of Performance Regressions