

Scenario-based Resilience Evaluation and Improvement of Microservice Architectures: Artifacts

Dominik Kesim¹, Lion Wagner¹, Jóakim von Kistowski²,
Sebastian Frank¹, Alireza Hakamian¹, André van Hoorn¹

¹ University of Stuttgart, Institute of Software Engineering, Germany

² DATEV eG, Nürnberg, Germany

CONTENTS

1	Introduction	2
2	Resilience Scenarios	2
2.1	Resilience Scenarios Overview	2
2.2	Common Terms	2
2.3	Service Level Objectives	3
2.4	Environmental Changes	3
2.5	Resilience Scenarios	4
2.5.1	Scenario 01: Peak(LinCo)/Ser/Abr	4
2.5.2	Scenario 02: Peak(ExCo)/Ser/Abr	4
2.5.3	Scenario 03: Peak(LinCo)/Ser/NoAbr	5
2.5.4	Scenario 04: Peak(ExCo)/Ser/NoAbr	5
2.5.5	Scenario 05: Failure(CF)/Ins/Ber	6
2.5.6	Scenario 06: Bug/Ins/Ber	6
2.5.7	Scenario 07: Failure(MW)/Bac/Ber	7
2.5.8	Scenario 08: Failure(MW)/Bac/Abr	7
2.5.9	Scenario 09: Depl(GW)/FroBac/NoIdle	8
2.5.10	Scenario 10: Failure(GW)/FroBac/NoIdle	8
2.5.11	Scenario 11: Failure(SerE)/Ser/Ber	8
2.5.12	Scenario 12: Failure(SerE)/Ins/Ber	9
3	Experiment Definitions	10
3.1	Experiment S04 Direct	10
3.2	Experiment S041 Random	12
3.3	Experiment S05 Chaos Monkey	14
4	Additional Experiment Results	16
4.1	Experiment S05'	16
4.2	Experiment S041 Random	19
4.3	Experiment S05 Chaos Monkey	21
5	Workshop Notion and Agenda	23
	References	25

1 Introduction

This document contains the artifacts that were created by our case study “Scenario-based Resilience Evaluation and Improvement of Microservice Architectures: An Experience Report” by Kesim et al. [2].

2 Resilience Scenarios

The following subsections contain information on the elicited resilience scenarios. Section 2.1 contains a compact overview and Section 2.2 explains some common and reappearing terms. The then following subsections present a more detailed description of the scenarios. Like in the paper, we use the scenario template described by Bass et al. [1].

2.1 Resilience Scenarios Overview

ID	Short Name	Source	Stimulus	Artifact	Environment	Response	Response Measure
01	Peak(LinCo)/Ser/Abr	User	Linear increasing load peak (cold start)	Service	Payslip calculation period	All requests are handled correctly and in time	Wage calculation ≤ 1 s, in 99% of the cases, payslip calculation ≤ 20 s (300 Employees)
02	Peak(ExCo)/Ser/Abr		Exponentially increasing load peak (cold start)				
03	Peak(LinCo)/Ser/NoAbr		Linear increasing load peak (cold start)		Not during payslip calculation period		
04	Peak(ExCo)/Ser/NoAbr		Exponentially increasing load peak (cold start)				
05	Failure(CF)/Ins/Ber	Cloud Foundry	Instance terminates	Instance	During wage calculation	User is unaware, calculations are correct and in time, at least one instance is running	Developer gets notified within five minutes
06	Bug/ Ins/Ber	Bug				Developer gets notified	
07	Failure(MW)/Bac/Ber	Middleware Operator	Middleware terminates	Backend	During wage calculation	Abort calculation, caller will be notified	Notification arrives within 1 s in 99% of the cases
08	Failure(MW)/Bac/Abr		Middleware terminates but recovers		Async. payslip calculation	Process is aborted but can be picked up	Restarts and SLOs are satisfied
09	Depl(GW)/FroBac/NoIdle	Deployment	Gateway terminates	Front- and Backend	Not Idle	Frontend shows error, gateway restarts	Downtime of gateway instance is below one minute
10	Failure(GW)/FroBac/NoIdle	Technical Issue					
11	Failure(SerE)/Ser/Ber	Receiving Service	Service terminates	Sending Service S and Receiving Service R	During wage calculation, no instance available	Error message and services R restarts	Downtime is below one minute
12	Failure(SerE)/Ins/Ber				During wage calculation, one instance not available	Calculation correct and in time	Wage calculation response time is below 2 s

Table 1: Scenarios elicited during the workshop

2.2 Common Terms

This subsection explains common terms that appear often in this document.

2.2.1 *Wage Clerks* are the majority of users of the system. They trigger pay slip and wage calculations independently, frequently and in different quantities.

2.2.2 A *Wage Calculation* corresponds to the calculation of a single pay slip of a single employee. It is start by a wage clerk and the main functionality of the payroll accounting system. During the calculation, the central “Calculations” service collects data from almost every other service asynchronously and calculates the resulting pay slip.

2.2.3 A *Pay Slip Calculation* consists of multiple wage calculations. Usually the wage of all employees of a company is calculated for the monthly closing.

2.2.4 In Germany, *Regular Office/Service Hours* are considered to be between 6 am and 6 pm, from Monday to Friday. With the exception of holidays.

2.2.5 The *Pay Slip Calculation Period* indicates the period of the payslip's calculation and serves as a parameter for the payslip calculation.

2.3 Service Level Objectives

In the following presentation of the resilience scenarios we reference some of the major Service-Level-Objective (SLO)s for the critical operations in the payment accounting system. These establish the following objectives:

- (1) The wage calculation is considered to have a response time below or equal to one second in 99% of the cases.
- (2) A pay slip calculation is considered to have a response time below or equal to one second in 99% of the cases. Though, if more than 300 payslips need to be calculated the system needs to respond within 20 seconds.
- (3) The time to notify a service error is below or equal to one second in 99% of the cases.

2.4 Environmental Changes

We consider three types of stimulus. Namely, load (i.e., requests rate or number of concurrent users) fluctuation, resource failure and combination of both. Load fluctuations are further divided into three different types, i.e., linear, exponential and constant load profiles. The following Figure 1 contains exemplary load profiles for these three types and their parameters. They were modeled with LIMBO [3]. The constant load in Figure 1c contains an additional constant noise (mean: 0.5, $\sigma = 0.5$).

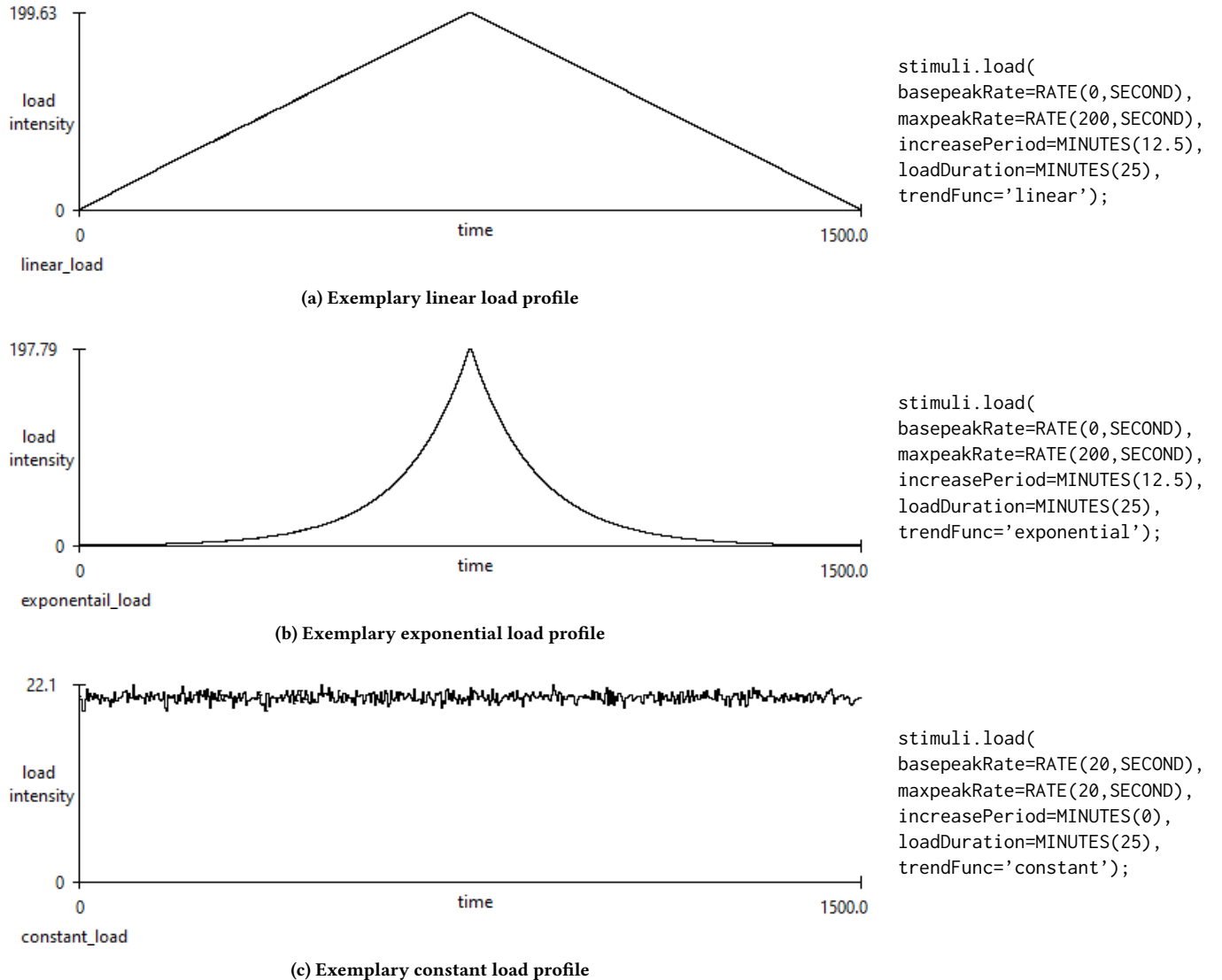


Figure 1: Exemplary load profiles and their LIMBO parameters.

2.5 Resilience Scenarios

2.5.1 Scenario 01: Peak(LinCo)/Ser/Abr

Source

The sources are wage clerks, sending requests to the payment accounting system.

Stimulus

The stimuli are incoming requests to calculate an employee's payslip. The intensity of requests varies over time with a linear increasing workload, as specified in Figure 1a with the following specification (LIMBO profil):

```
stimuli.load(
basepeakRate=RATE(200,SECOND),
maxpeakRate=RATE(400,SECOND),
increasePeriod=MINUTES(30),
loadDuration=MINUTES(4),
trendFunc='linear');
```

Artifact

The services 'Working-Hours', 'Calculations', 'Taxes', 'Employees', 'Social-Insurances', and 'Payments' are affected by the stimulus.

Environment

Pay slip period, during regular service hours.

Response

Requests for performing a wage calculation will be handled correctly and in time. (Enabling autoscaling will be helpful in case of too many requests). Requests for calculating a pay slip will be handled correctly and in time.

Response Measure

The SLOs as specified in Section 2.3 for response times of a wage and payslip calculation are satisfied.

Natural Language Description

During regular service hours in Germany, between six am and six pm, an increasing number of wage clerks send employee information to the payroll accounting service at the same time in order to calculate an employee's pay slip. Thus, the payroll accounting service experiences a linear increasing load which leads to an increase of requests within a 30 second period. The different services involved in the process will calculate the portions of the pay slip. Then, the results will be sent back to the wage clerks. In order to satisfy the SLOs, the payroll account service is expected to return an answer within one second, in case of a wage calculation. In case of a pay slip calculation, the service is expected to return an answer within 20 seconds, if the number of employees is below or equal to 300.

2.5.2 Scenario 02: Peak(ExCo)/Ser/Abr

Source

The sources are wage clerks, sending requests to the payment accounting system.

Stimulus

The stimuli are incoming requests to calculate an employee's payslip. The intensity of requests varies over time with an exponentially increasing workload, as specified in Figure 1b with the following specification (LIMBO profil):

```
stimuli.load(
basepeakRate=RATE(0,SECOND),
maxpeakRate=RATE(300,SECOND),
increasePeriod=MINUTES(12),
loadDuration=MINUTES(0),
trendFunc='exponential');
```

Artifact

The services 'Working-Hours', 'Calculations', 'Taxes', 'Employees', 'Social-Insurances', and 'Payments' are affected by the stimulus.

Environment

Pay slip period, during regular service hours.

Response

Requests for performing a wage calculation will be handled correctly and in time. (Enabling autoscaling will be helpful in case of too many requests). Requests for calculating a pay slip will be handled correctly and in time.

Response Measure

The SLOs as specified in Section 2.3 for response times of a wage and payslip calculation are satisfied.

Natural Language Description

—

2.5.3 Scenario 03: Peak(LinCo)/Ser/NoAbr

Source

The sources are wage clerks, sending requests to the payment accounting system.

Stimulus

The stimuli are incoming requests to calculate an employee's payslip. The intensity of requests varies over time with a linear increasing workload, as specified in Figure 1a with the following specification (LIMBO profil):

```
stimuli.load(  
basepeakRate=RATE(0,SECOND),  
maxpeakRate=RATE(300,SECOND),  
increasePeriod=MINUTES(12),  
loadDuration=MINUTES(0),  
trendFunc='linear');
```

Artifact

The services 'Working-Hours', 'Calculations', 'Taxes', 'Employees', 'Social-Insurances', and 'Payments' are affected by the stimulus.

Environment

Pay slip period, during regular service hours.

Response

Requests for performing a wage calculation will be handled correctly and in time. (Enabling autoscaling will be helpful in case of too many requests). Requests for calculating a pay slip will be handled correctly and in time.

Response Measure

The SLOs as specified in Section 2.3 for response times of a wage and payslip calculation are satisfied.

Natural Language Description

—

2.5.4 Scenario 04: Peak(ExCo)/Ser/NoAbr

Source

The sources are wage clerks, sending requests to the payment accounting system.

Stimulus

The stimuli are incoming requests to calculate an employee's payslip. The intensity of requests varies over time with an exponentially increasing workload, as specified in Figure 1b with the following specification (LIMBO profil):

```
stimuli.load(  
basepeakRate=RATE(0,SECOND),
```

```
maxpeakRate=RATE(299,SECOND),  
increasePeriod=MINUTES(12),  
loadDuration=MINUTES(0),  
trendFunc='exponential');
```

Artifact

The services 'Working-Hours', 'Calculations', 'Taxes', 'Employees', 'Social-Insurances', and 'Payments' are affected by the stimulus.

Environment

Pay slip period, during regular service hours.

Response

Requests for performing a wage calculation will be handled correctly and in time. (Enabling autoscaling will be helpful in case of too many requests). Requests for calculating a pay slip will be handled correctly and in time.

Response Measure

The SLOs as specified in Section 2.3 for response times of a wage and payslip calculation are satisfied.

Natural Language Description

—

2.5.5 Scenario 05: Failure(CF)/Ins/Ber

Source

The Cloud Foundry master.

Stimulus

Internal Error of Cloud Foundry results in the loss of a random service instance.

Artifact

The instances of the following services 'Working-Hours', 'Calculations', 'Taxes', 'Employees', 'Social-Insurances', and 'Payments', 'API Gateway', and 'Eureka' are affected by the stimulus.

Environment

Pay slip period, during regular service hours.

Response

The user may not perceive the loss of an instance, the wage calculation will finish correctly and in time while a sufficient amount of instances are available (at least 1).

Response Measure

The SLOs as specified in Section 2.3 for response times of a wage and payslip calculation are satisfied.

Natural Language Description

—

2.5.6 Scenario 06: Bug/Ins/Ber

Source

Internal bug of a dedicated instance running in Cloud Foundry

Stimulus

The instance will become unavailable due to the bug.

Artifact

The services 'Working-Hours', 'Calculations', 'Taxes', 'Employees', 'Social-Insurances', and 'Payments' are affected by the stimulus.

Environment

Pay slip period, during regular service hours.

Response

The responsible developer will be notified.

Response Measure

A Notification will be sent within 5 minutes.

Natural Language Description

If an instance crashes and throws a catchable runtime exception the respective developer (team) should be notified within 5 min.

2.5.7 Scenario 07: Failure(MW)/Bac/Ber

Source

Middleware operator

Stimulus

The middleware becomes unavailable

Artifact

The services 'Working-Hours', 'Calculations', 'Taxes', 'Employees', 'Social-Insurances', and 'Payments' are affected by the stimulus.

Environment

Pay slip period, during regular service hours.. The system experiences a continuous static load.

Response

Wage and pay slip calculation will be aborted. The caller will be notified.

Response Measure

The SLOs as specified in Section 2.3 for response times of a wage and payslip calculation are satisfied.

Natural Language Description

—

2.5.8 Scenario 08: Failure(MW)/Bac/Abr

Source

Middleware operator

Stimulus

The middleware becomes unavailable, but then restarts to an operating state again.

Artifact

The services 'Working-Hours', 'Calculations', 'Taxes', 'Employees', 'Social-Insurances', and 'Payments' are affected by the stimulus.

Environment

Asynchronously executing pay slip calculation. The system experiences continuous static load.

Response

Process aborts but is able to continue from where it stopped.

Response Measure

Process continues at the point where it stopped and finishes within the SLOs, as specified in Section 2.3, for response times of a wage calculation and pay slip.

Natural Language Description

2.5.9 Scenario 09: Depl(GW)/FroBac/NoIdle

Source

Deployment process initiated by a developer.

Stimulus

API Gateway crashes and becomes unavailable.

Artifact

The services 'Working-Hours', 'Calculations', 'Taxes', 'Employees', 'Social-Insurances', and 'Payments' are affected by the stimulus. Additionally the frontend is affected.

Environment

Pay slip period, during regular service hours. The system experiences continuous static load.

Response

The frontend will display an error message while the API Gateway reboots.

Response Measure

Downtime of the gateway service is less than 1 minute.

Natural Language Description

2.5.10 Scenario 10: Failure(GW)/FroBac/NoIdle

Source

Some kind of technical issue.

Stimulus

API Gateway crashes and becomes unavailable.

Artifact

The services 'Working-Hours', 'Calculations', 'Taxes', 'Employees', 'Social-Insurances', and 'Payments' are affected by the stimulus. Additionally the frontend is affected.

Environment

Pay slip period, during regular service hours.

Response

The frontend will display an error message while the API Gateway reboots.

Response Measure

Downtime of the gateway service is less than 1 minute.

Natural Language Description

2.5.11 Scenario 11: Failure(SerE)/Ser/Ber

Source

Any receiving service R of a wage of pay slip calculation.

Stimulus

Service R does not respond or terminates.

Artifact

The sending service S and Receiving service R. The services are either one of 'Zeiten', 'Berechnung', 'Steuer', 'Mitarbeiter', 'Sozialversicherung', and 'Bezüge'.

Environment

During pay slip or wage calculation, during regular service hours but all instances of R are not available.

Response

An error message is shown and service R gets restarted.

Response Measure

Downtime of the service R should be below 1 minute.

Natural Language Description

—

2.5.12 Scenario 12: Failure(SerE)/Ins/Ber

Source

Any receiving service R of a wage of pay slip calculation.

Stimulus

Service R does not respond or terminates.

Artifact

The sending service S and Receiving service R. The services are either one of 'Zeiten', 'Berechnung', 'Steuer', 'Mitarbeiter', 'Sozialversicherung', and 'Bezüge'.

Environment

During pay slip or wage calculation, during regular service hours but one instance of R is not available.

Response

Calculations are continued correctly and in time.

Response Measure

The SLOs as specified in Section 2.3 for response times of a wage and payslip calculation are satisfied.

Natural Language Description

—

3 Experiment Definitions

The following subsections contain the experiment definitions in the form of Javascript Object Notation (JSON) input files for the ChaosToolkit (CTK).

3.1 Experiment S04 Direct

```

1 {
2   "version": "1.0.0",
3   "title": "S04 - Peak(LinCo)/Ser/Abr/Dir",
4   "description": "The system experiences static workload, while a fault in cloud foundry
5     ↳ leads to a termination of single payslip-service instance",
6   "contributions": {
7     "resilience": "high",
8     "availability": "high"
9   },
10  "tags": [
11    "loadgenerator",
12    "cloud-foundry",
13    "linear-increase",
14    "direct-termination"
15  ],
16  "configuration": {
17    "cf_api_url": "https://api.dev.cfdev.sh",
18    "cf_verify_ssl": false
19  },
20  "secrets": {
21    "cloudfoundry": {
22      "cf_username": "user",
23      "cf_password": "pass"
24    }
25  },
26  "steady-state-hypothesis": {
27    "title": "Verify-Hypothesis",
28    "probes": [
29      {
30        "type": "probe",
31        "name": "call-hypothesis-validator-service",
32        "tolerance": {
33          "type": "jsonpath",
34          "path": "$.body.result",
35          "expect": [
36            true
37          ]
38        },
39        "provider": {
40          "type": "http",
41          "secrets": [
42            "cloudfoundry"
43          ],
44          "url": "http://hypothesis-validator-api.dev.cfdev.sh/api/v1/hypothesis/s0
45            ↳ 0",
46          "method": "GET",
47          "verify_tls": false
48        }
49      ]
50    }
51  }
52}

```

```

49     },
50     "method": [
51         {
52             "pauses": {
53                 "before": 600,
54                 "after": 900
55             },
56             "type": "action",
57             "name": "terminate-app-instance",
58             "provider": {
59                 "func": "terminate_app_instance",
60                 "type": "python",
61                 "secrets": [
62                     "cloudfoundry"
63                 ],
64                 "module": "chaoscf.actions",
65                 "arguments": {
66                     "instance_index": 0,
67                     "app_name": "payslip-service",
68                     "org_name": "cfdev-org"
69                 }
70             }
71         }
72     ],
73     "rollbacks": [
74         {
75             "type": "action",
76             "name": "start-payslip-service-instance",
77             "provider": {
78                 "func": "start_app",
79                 "type": "python",
80                 "secrets": [
81                     "cloudfoundry"
82                 ],
83                 "module": "chaoscf.actions",
84                 "arguments": {
85                     "app_name": "payslip-service"
86                 }
87             }
88         }
89     ]
90 }

```

Listing 1: Experiment Definition of Experiment S04 Direct

3.2 Experiment S041 Random

```

1 {
2   "version": "1.0.0",
3   "title": "S04 - Peak(LinCo)/Ser/Abr",
4   "description": "The system experiences static workload, while a fault in cloud foundry
5     ↳ leads to a termination of a random payslip-service instance",
6   "contributions": {
7     "resilience": "high",
8     "availability": "high"
9   },
10  "tags": [
11    "loadgenerator",
12    "cloud-foundry",
13    "linear-increase",
14    "random-termination"
15  ],
16  "configuration": {
17    "cf_api_url": "https://api.dev.cfdev.sh",
18    "cf_verify_ssl": false
19  },
20  "secrets": {
21    "cloudfoundry": {
22      "cf_username": "user",
23      "cf_password": "pass"
24    }
25  },
26  "steady-state-hypothesis": {
27    "title": "Verify-Hypothesis",
28    "probes": [
29      {
30        "type": "probe",
31        "name": "call-hypothesis-validator-service",
32        "tolerance": {
33          "type": "jsonpath",
34          "path": "$.body.result",
35          "expect": [
36            true
37          ]
38        },
39        "provider": {
40          "type": "http",
41          "secrets": [
42            "cloudfoundry"
43          ],
44          "url": "http://hypothesis-validator-api.dev.cfdev.sh/api/v1/hypothesis/s0
45            ↳ 0",
46          "method": "GET",
47          "verify_tls": false
48        }
49      ]
50    },
51    "method": [
52      {
53        "pauses": {

```

```

53         "before": 600,
54         "after": 900
55     },
56     "type": "action",
57     "name": "terminate-random-instance",
58     "provider": {
59         "func": "terminate_some_random_instance",
60         "type": "python",
61         "module": "chaoscf.actions",
62         "arguments": {
63             "app_name": "payslip-service",
64             "org_name": "cfdev-org"
65         }
66     }
67 },
68 ],
69 "rollbacks": [
70     {
71         "type": "action",
72         "name": "start-all-apps",
73         "provider": {
74             "func": "start_apps",
75             "type": "python",
76             "module": "chaoscf.actions",
77             "arguments": {
78                 "org_name": "cfdev-org"
79             }
80         }
81     }
82 ]
83 }

```

Listing 2: Experiment Definition of Experiment S041 Random

3.3 Experiment S05 Chaos Monkey

```

1 {
2   "version": "1.0.0",
3   "title": "Latency-Injection-payslip-Service",
4   "description": "In this experiment, latency will be injected into the payslip-service.",
5   "contributions": {},
6   "configuration": {
7     "cf_api_url": "https://api.dev.cfdev.sh",
8     "cf_verify_ssl": false
9   },
10  "secrets": {
11    "cloudfoundry": {
12      "cf_username": "user",
13      "cf_password": "pass"
14    }
15  },
16  "steady-state-hypothesis": {
17    "title": "Verify-Hypothesis",
18    "probes": [
19      {
20        "type": "probe",
21        "name": "call-hypothesis-validator-service",
22        "tolerance": {
23          "type": "jsonpath",
24          "path": "$.body.result",
25          "expect": [
26            true
27          ]
28        },
29        "provider": {
30          "type": "http",
31          "secrets": [
32            "cloudfoundry"
33          ],
34          "url": "http://hypothesis-validator-api.dev.cfdev.sh/api/v1/hypothesis/s0
           ↪ 0",
35          "method": "GET",
36          "verify_tls": false
37        }
38      ]
39    ],
40    "method": [
41      {
42        "pauses": {
43          "before": 600
44        },
45        "name": "enable_chaosmonkey",
46        "provider": {
47          "arguments": {
48            "base_url": "http://payslip-service-api.dev.cfdev.sh/actuator"
49          },
50          "func": "enable_chaosmonkey",
51          "module": "chaospring.actions",
52          "type": "python"
53        }

```

```

54     },
55     "type": "action"
56 },
57 {
58     "name": "configure_latency_injection_at_payslip_service",
59     "provider": {
60         "arguments": {
61             "base_url": "http://payslip-service-api.dev.cfdev.sh/actuator",
62             "assaults_configuration": {
63                 "level": 2,
64                 "latencyRangeStart": 2000,
65                 "latencyRangeEnd": 5000,
66                 "latencyActive": true,
67                 "exceptionsActive": false,
68                 "killApplicationActive": true
69             }
70         },
71         "func": "change_assaults_configuration",
72         "module": "chaospring.actions",
73         "type": "python"
74     },
75     "type": "action",
76     "pauses": {
77         "after": 900
78     }
79 },
80 ],
81 "rollbacks": [
82     {
83         "name": "disable_chaosmonkey",
84         "provider": {
85             "arguments": {
86                 "base_url": "http://payslip-service-api.dev.cfdev.sh/actuator"
87             },
88             "func": "disable_chaosmonkey",
89             "module": "chaospring.actions",
90             "type": "python"
91         },
92         "type": "action"
93     }
94 ],
95 }

```

Listing 3: Experiment Definition of Experiment S05 Chaos Monkey

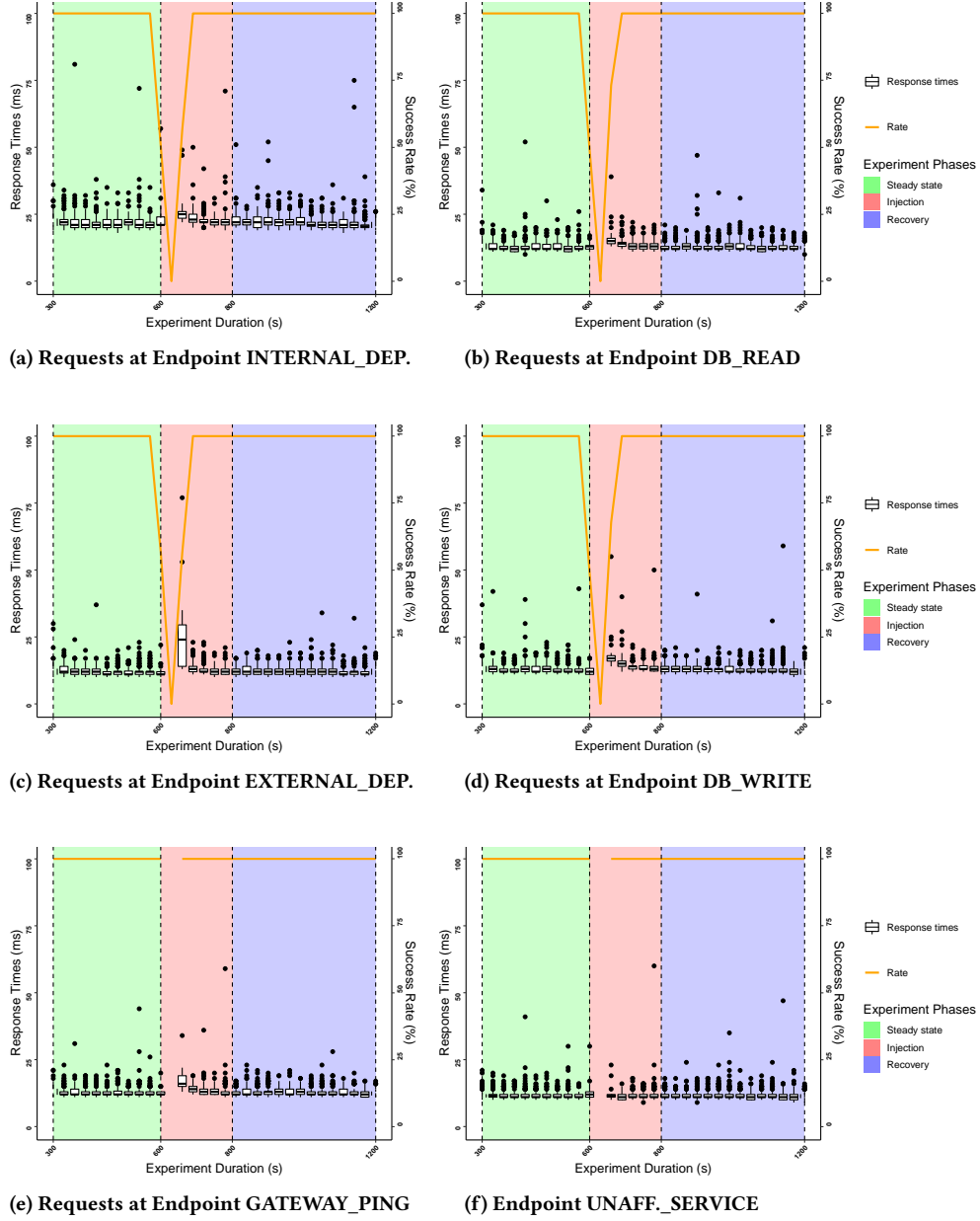


Figure 2: Results of Experiment S05' without Pattern

4 Additional Experiment Results

First, an extended version of the description of Scenario 05' is presented in Section 4.1. This is the same experiment as presented in the paper, but results are shown for all endpoints. After that, the results of the remaining two experiments that were executed but not covered in the paper [2] follow. The code and data that generated the graphs are displayed on CodeOcean [4].

4.1 Experiment S05'

4.1.1 Experiment Results (Without Pattern) Figure 2 shows the three main phases of the experiment: steady state, injection, and recovery. In the steady state, we assume that the system is working as expected, i.e., the response times are satisfying the SLOs. In the injection phase, CTK executes the defined method, i.e., terminating the *payslip-service* instance. In the recovery phase, we assume that the system recovers and returns to a steady-state phase, i.e., the response times satisfies the SLOs. We omitted the warmup and cooldown phase of the load

Endpoint	Steady State								Injection								Recovery							
	w/o Pattern				w Pattern				w/o Pattern				w Pattern				w/o Pattern				w Pattern			
	p_5	\tilde{x}	\bar{x}	p_{99}	p_5	\tilde{x}	\bar{x}	p_{99}	p_5	\tilde{x}	\bar{x}	p_{99}	p_5	\tilde{x}	\bar{x}	p_{99}	p_5	\tilde{x}	\bar{x}	p_{99}	p_5	\tilde{x}	\bar{x}	p_{99}
INTERNAL_DEP.	19	22	22.5	33	19	22	24.0	51	19	21	22.4	32	19	22	24.6	90	19	21	22.1	31	19	22	23.0	34
DB_READ	11	12	13.3	21	11	12	13.0	24	11	12	13.1	20	11	12	13.2	30	11	12	12.9	20	11	12	12.8	19
EXTERNAL_DEP.	11	12	12.6	21	10	12	13.3	23	11	12	12.5	21	10	12	14.1	31	11	12	12.3	19	10	12	12.2	19
DB_WRITE	11	13	13.4	21	11	12	13.1	22	11	12	13.1	20	11	12	13.3	27	11	12	13.0	20	11	12	12.7	19
GATEWAY_PING	11	12	13.3	21	11	12	13.3	24	11	12	13.1	20	11	12	13.6	32	11	12	12.9	19	11	12	12.9	21
UNAFF_SERVICE	10	11	11.9	19	10	11	11.6	19	10	11	11.7	18	10	11	11.6	19	10	11	11.8	18	10	11	11.5	19

Table 2: Statistical summaries of the three experiment phases. p_α : α -th percentile; \tilde{x} : median; and \bar{x} : mean. Values are given in ms and calculated using *Definition 1* of Hyndman and Fan [?].

generator due to readability and analysis purposes, which refers to the overall first and last 300 s. Further, a thirty-second binning was applied, and extreme outliers (>100 ms) are not shown.

The success rates at the endpoints INTERNAL_DEP., DB_READ, EXTERNAL_DEP., and DB_WRITE drop to 0 % as the *payslip-service* is terminated after 600 s and rises back to 100 % as it recovers in about 1.5 min. During this downtime, no response times are recorded since no requests arrive at the *payslip-service*. During the steady-state and recovery phase, the response times are stable at around 20 ms and 15 ms, respectively. During the injection phase, there is a slight increase observable as the *payslip-service* has restarted. The results for GATEWAY_PING and UNAFF_SERVICE show a similar structure. However, the load generator did not record any successful or failed requests during the downtime. Therefore, no success rate could be calculated.

4.1.2 Discussion of Results (Without Pattern) As visible in Figure 2, the response time and success rate values are almost identical in the steady state phase and the recovery phase. This concludes that the system successfully recovered from the failure of the *payslip-service* instance. Furthermore, the increase in the success rate indicates that the *payslip-service* becomes available after 30 s to 60 s. Thus, the Cloud Foundry (CF) platform can re-instantiate the *payslip-service* quickly, leading to a quick recovery of the system.

There are no response times during the unavailability of the *payslip-service*. The response times of the successful requests also behave as expected. Response times are slightly higher at the beginning when the *payslip-service* is re-instantiated, which was expected as normal cold-start behavior. The expectation is that requests to endpoints GATEWAY_PING and UNAFF_SERVICE remain unaffected during the injection because the *payslip-service* is not required to answer the request. Nevertheless, response times at endpoint GATEWAY_PING are affected, which indicates a propagation of the failure effects from the *payslip-service* to the *API-Gateway-service*.

After the injection started, the success rate drops to 0 % at the endpoints INTERNAL_DEP., DB_READ, EXTERNAL_DEP., and DB_WRITE. The CTK terminates the single *payslip-service* instance. The load generator flags all requests as failed, leading to a success rate of 0 %. The plots show neither successful nor failing request responses at the endpoints GATEWAY_PING and UNAFF_SERVICE during injection, which indicates no requests exist in the system. Another possibility is that requests have been dropped. Looking at the raw data tables disproves this argument as there are no dropped requests. Another explanation is that no requests arrived at the system, which leads to a lack of data in the time frame between approximately 600 s and 660 s.

In our hypothesis we stated that the response measure of Scenario 05 still holds, i.e., requests are answered in time (99 % in less than 1 s) and correctly. The measured response times are far below 1 s during the experiment. Thus, our hypothesis regarding the response times is technically fulfilled. However, several requests are not answered at all, which is indicated by the dropped success rate. We consider these as incorrect response. Therefore, we assume that the hypothesis regarding correctness is not fulfilled.

4.1.3 Experiment Results (With Pattern) Each plot in Figure 3 visualizes the system’s response times and success rates with the retry pattern for one endpoint. As in the experiment without pattern Figure 4, each plot is divided into the steady state phase, the injection phase, and the recovery phase. Table 2 shows the associated statistical values.

In general, similar behavior can be observed at all the endpoints. Comparing Figure 2 and Figure 3 shows that the response times in the steady state phase do not vary significantly when the retry pattern is activated. The boxplots show a slightly higher interquartile range in the plot where the retry pattern is integrated. The response times and success rates between steady state and recovery phase do also not differ much. Additionally, in the recovery phase, the mean of response times at endpoint DB_WRITE is even lower than in the experiment without the retry pattern.

During the injection phase, the mean response times are higher in the experiment with the retry pattern than in the experiment without the pattern. However, this does not hold for the endpoint UNAFF_SERVICE, at which the mean response times slightly improved. This behavior also becomes visible in the boxplots. At the beginning of the injection phase, far more high response times can be observed, which leads to a visible spike. Again, this effect is much weaker for endpoint UNAFF_SERVICE.

The plots also show that the success rate does not drop to zero anymore when the pattern is active. For the endpoints INTERNAL_DEP., DB_READ, EXTERNAL_DEP., and DB_WRITE, the success rate drops to approximately 70 %. For the other two endpoints GATEWAY_PING and UNAFF_SERVICE, requests are arriving and the success rate continuously remains at 100 %.

4.1.4 Discussion of Results (With Pattern) Comparing Figure 2 and Figure 3 leads to the conclusion that the system still recovers. In addition, the retry pattern did only slightly affect the response times in the steady state and recovery phase. Their different purposes can

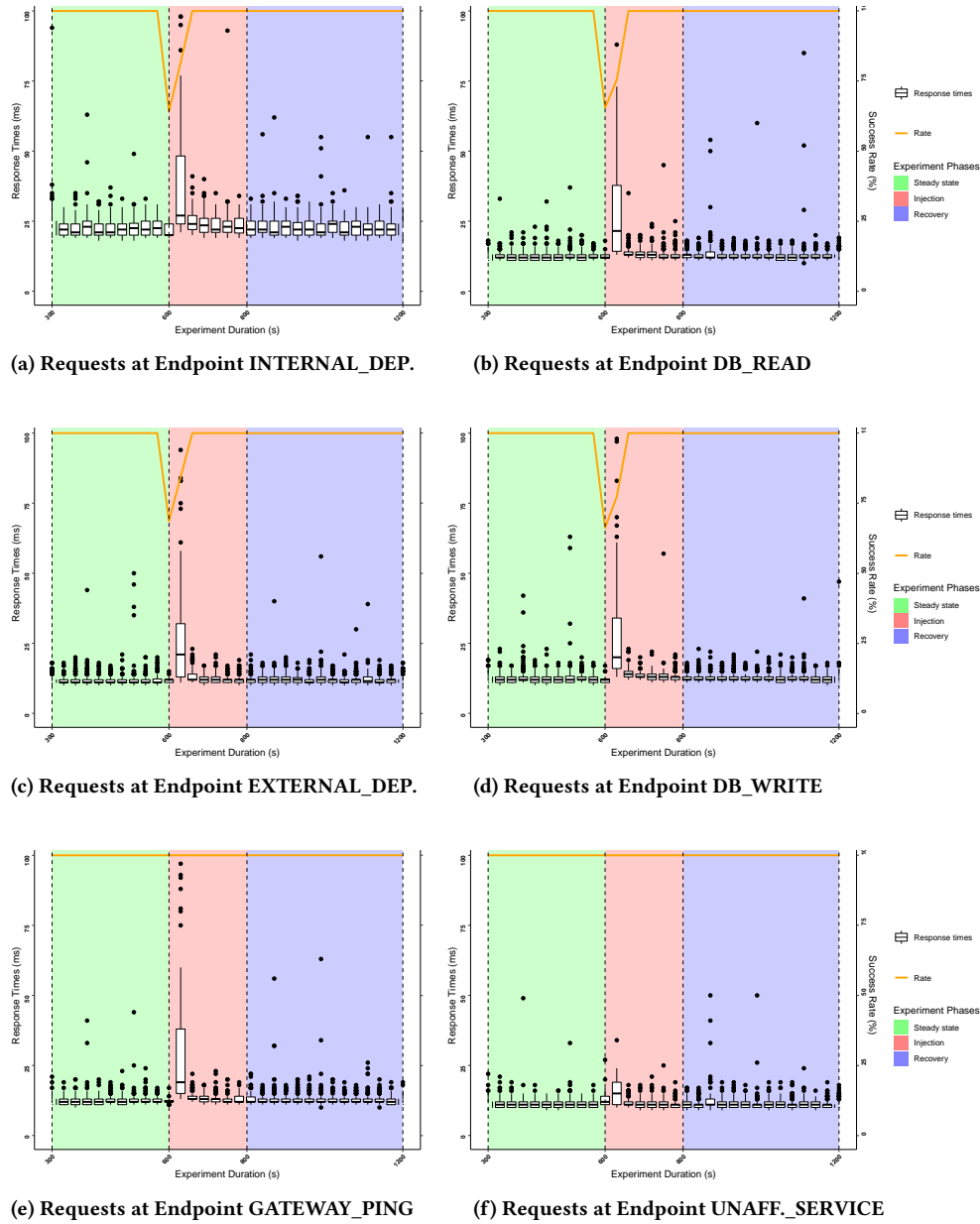


Figure 3: Results of Experiment with activated Retry Pattern for Scenario 05'

explain the general difference in (average) response times between the endpoints, e.g., endpoint EXTERNAL_DEP. does not implement complicated business logic or perform Central Processing Unit (CPU) intensive tasks. Hence, in comparison to the other endpoints, where some calculations are made, the response times are slightly lower. The approximately doubled response times in the steady state phase at endpoint INTERNAL_DEP. are probably related to its internal call to *payslip-service2*.

The application of the retry pattern can explain the response time spikes during the injection. Requests sent shortly before the restart of the *payslip-service* fail, but are retried by the *API-Gateway-service* until the *payslip-service* recovered after approximately 10 s. However, as several retries have been aggregated, the *payslip-service* will have to handle a high amount of requests upon recovery. Thus, the typical cold start behavior and the amount of aggregated requests result in a visible spike in response times.

The endpoints UNAFF_SERVICE and GATEWAY_PING do not depend on the *payslip-service*. This explains the high success rate at these endpoints. For endpoint UNAFF_SERVICE, only the mean during the injection phase shows a small difference of approximately 2 ms in the

experiments with and without retry pattern. This can be the case due to the CF platform using more resources for instance managing and cold start routing to the restarted *payslip-service*.

In contrast to the experiment without the retry pattern, the success rate does not drop entirely. This leads to the conclusion that the retry pattern improved the scenario satisfaction as it increased the percentage of correct responses while keeping the response times below 1 s. A typical trade-off for the retry pattern can be observed between the response times and success rate. Still, another configuration of the retry mechanism, e.g., the number of retries, could be even more beneficial but would require more experiments to be conducted.

4.2 Experiment S041 Random

The following graphs contain the results of Experiment S041 Random.

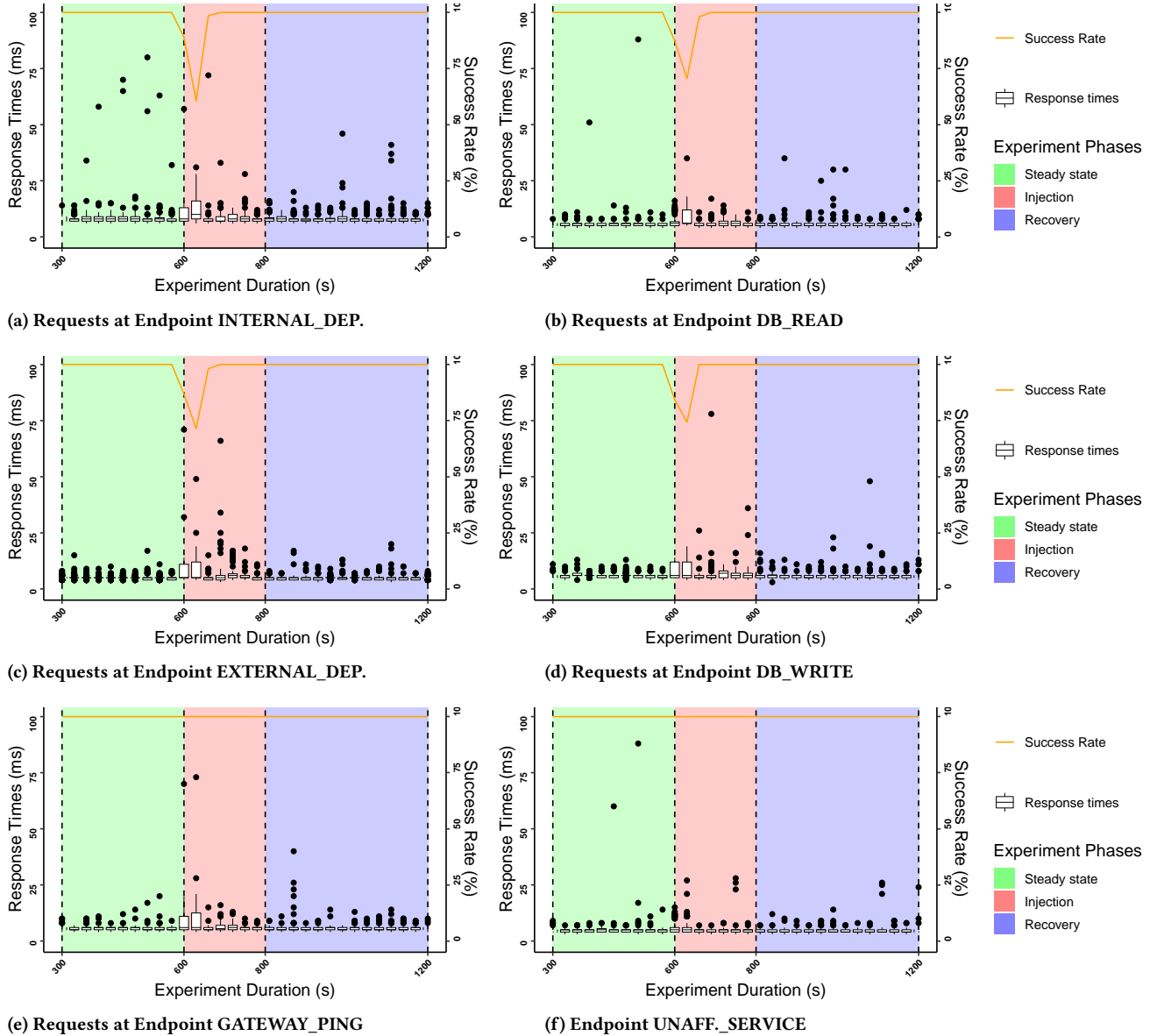


Figure 4: Results of Experiment S041 Random without Pattern

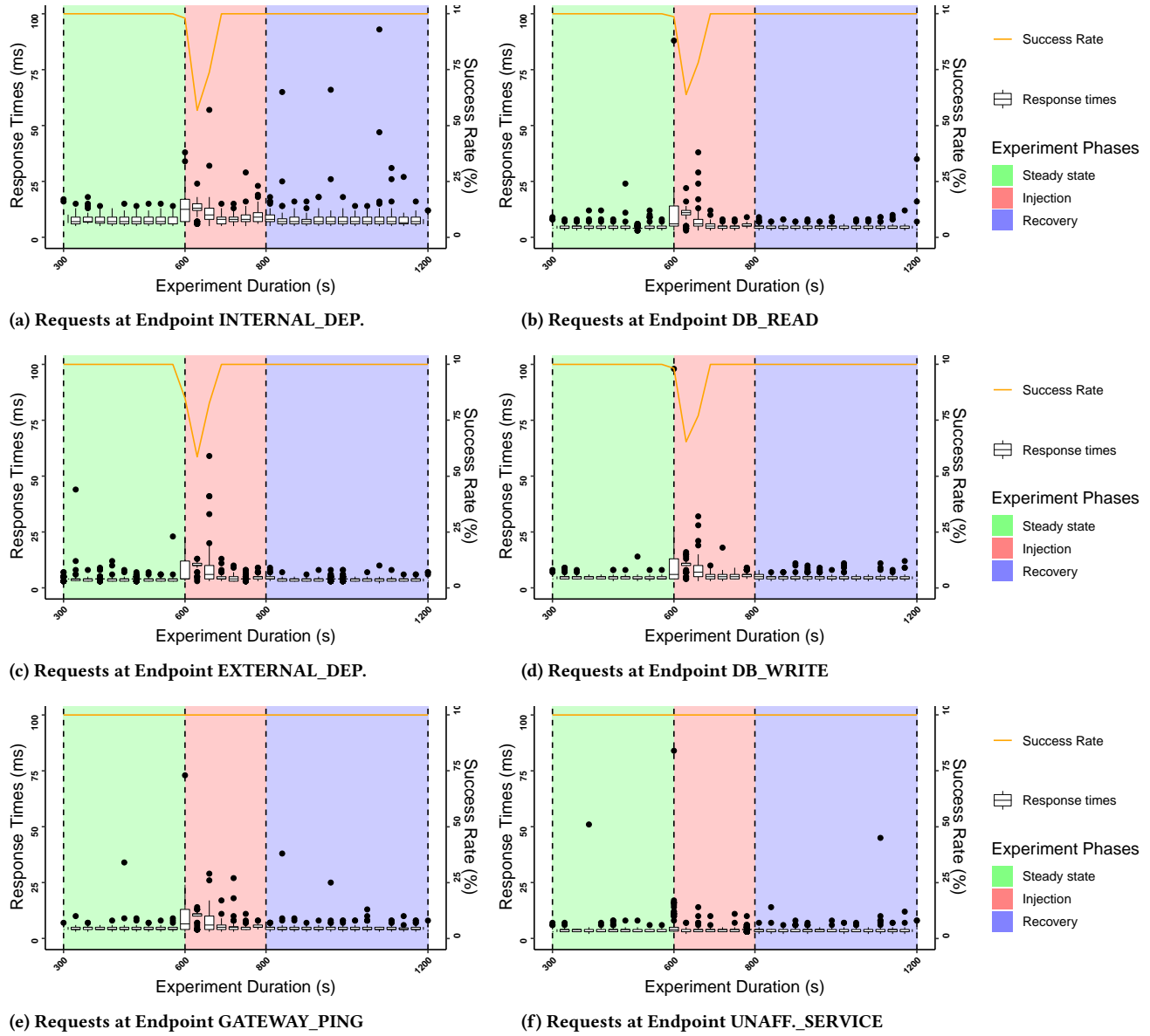


Figure 5: Results of Experiment S041 Random with Retry Pattern

4.3 Experiment S05 Chaos Monkey

The following graphs contain the results of Experiment S05 Chaos Monkey.

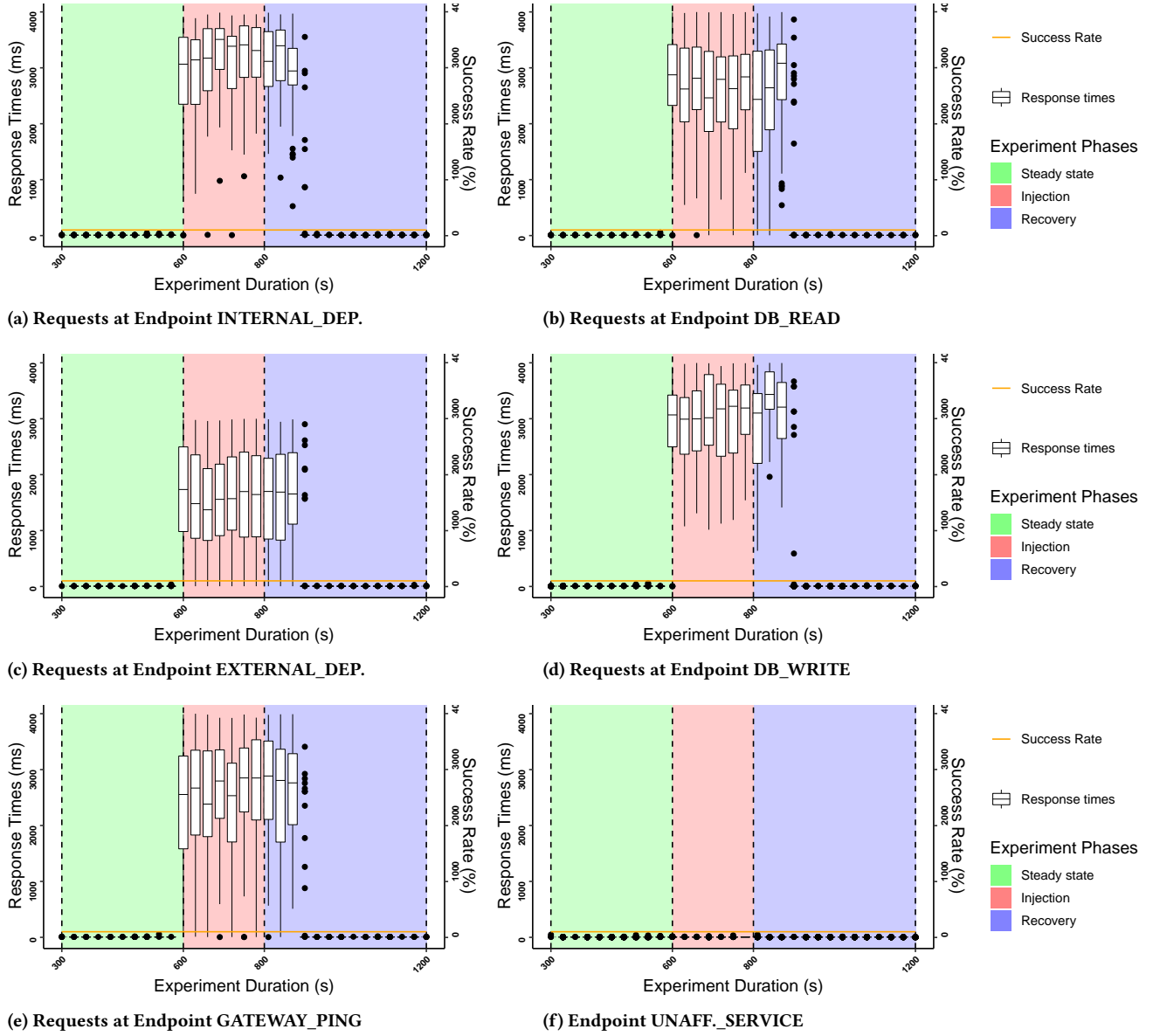


Figure 6: Results of Experiment S05 Chaos Monkey without Pattern

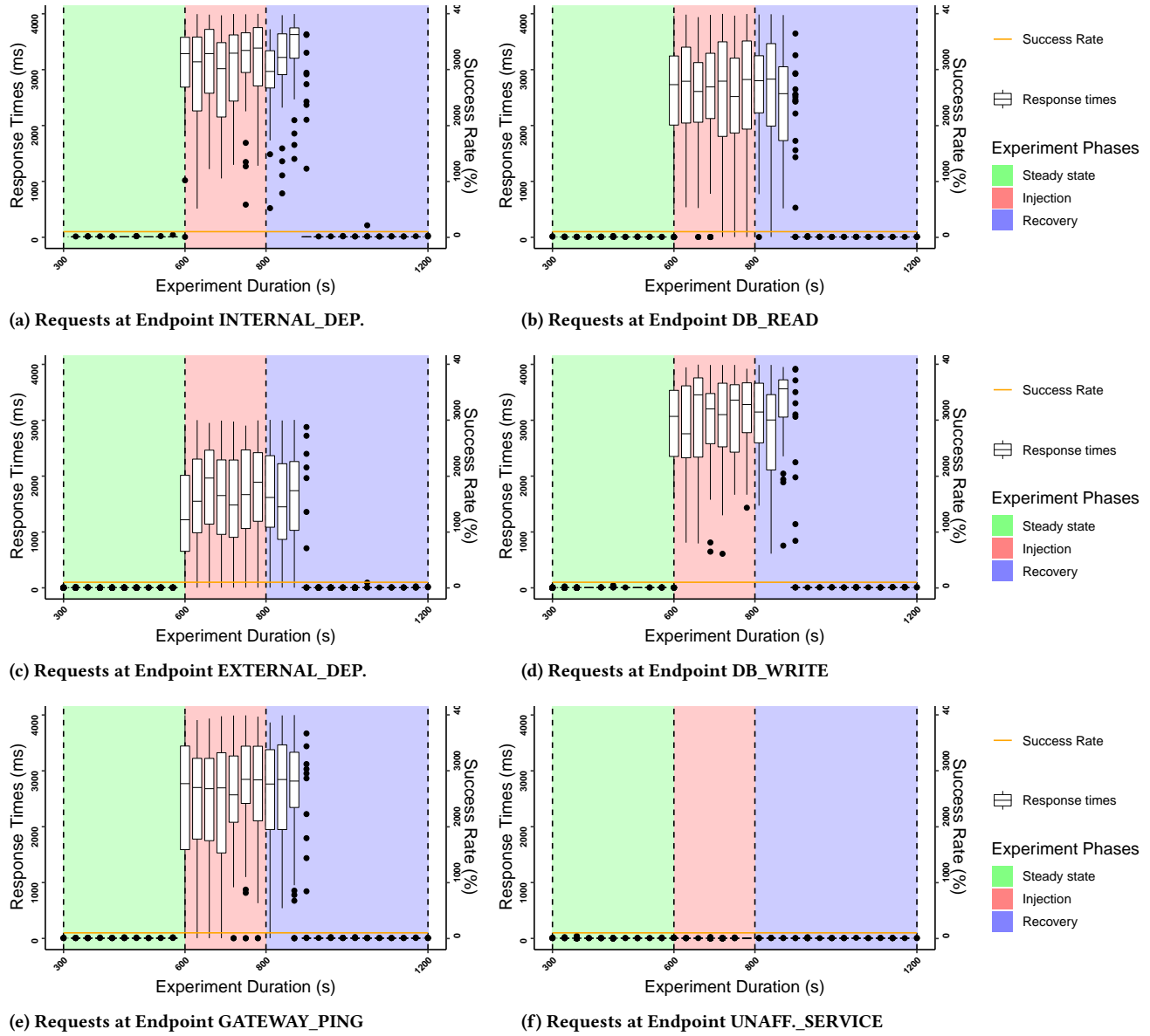


Figure 7: Results of Experiment Chaos Monkey with Retry Pattern

5 Workshop Notion and Agenda

Figures 8 and 9 contain the a short notion and agenda of the Architecture Trade-Off Analysis Method (ATAM)-based workshop. Since our industrial partner was a German company, this document is also in German. Further, it has been censored to not reveal any confidential data.

Workshop Agenda und Inhalt

Sehr geehrte Teilnehmer,

im Rahmen unseres Forschungsprojekts möchten wir mit Ihnen einen gemeinsamen Workshop durchführen. Innerhalb unseres Forschungsprojekts wollen wir auf Basis einer Szenario-basierten Analyse, welche im Fokus dieses Workshops steht, und der Anwendung von automatisierten Chaos Experimenten die Widerstandsfähigkeit (engl. resilience) von [REDACTED] untersuchen und verbessern. Im Rahmen dieses Workshops erhoffen wir uns folgende Ergebnisse zu erzielen:

1. Verständnis für die Architektur von [REDACTED] (Struktur, Verhalten, ...) entwickeln
2. Identifikation von möglichen Risiken und bereits implementierten Lösungen zur Verbesserung der Resilience
3. Identifikation und Priorisierung von Resilience-Szenarien

Im Folgenden finden Sie eine Agenda für den Workshop. Der Workshop findet am,

20.02.2020 von 10 Uhr – 16 Uhr in der [REDACTED] statt

Die von uns definierten Zeitintervalle beinhalten keine Pausen, diese können aber nach Bedarf spontan eingebaut werden. Für den Workshop benötigen wir folgende Hilfsmittel: Whiteboards, Flipcharts, Moderationskoffer, Stifte, Post-its, u. ä.

Session 1 (10:00 – 11:30)

- Einführung
 - o Vorstellen der Teilnehmer + erhoffte Ergebnisse aus dem Workshop
 - o Rollen werden festgelegt (Moderator, Protokollant)
 - o Ziel des Workshops
 - o Regeln für die Teilnehmer
- Architekturbeschreibung
 - o Architekturvorstellung der Moderatoren
 - Moderatoren erläutern Architektur auf Basis von Ihrem Verständnis anhand verschiedener Architektursichten
 - Teilnehmer ergänzen/beschreiben/korrigieren die Architektur
 - o Identifikation von Patterns, die bereits in der Architektur integriert sind

Pause (11:30 – 12:30)

Session 2 (12:30 – 13:30)

- Risikoanalyse
 - o Brainstorming mit Leading Questions: Was kann schiefgehen?
 - o Klassifizierung der Ergebnisse
 - o Risikomatrix erzeugen

Figure 8: Workshop agenda page 1/2

Session 3 (13:45 – 15:45)

- Terminologie der angewandten Methodik (ATAM)
- Identifikation von Resilience-Szenarien
- Priorisierung der Ergebnisse

Abschluss (15:45 – 16:00)

- Retrospektive
 - o Was lief gut?
 - o Was lief schlecht?
 - o Fragen und Feedback

Wir freuen uns auf den gemeinsamen Tag und auf eine aktive Zusammenarbeit!

Mit freundlichen Grüßen,

Dominik Kesim (dominik.kesim@gmail.com)
und Lion Wagner (st148345@stud.uni-stuttgart.de)
Universität Stuttgart

Figure 9: Workshop agenda page 2/2

References

- [1] Len Bass, Paul Clements, and Rick Kazman. 2003. *Software Architecture in Practice*. Addison-Wesley Longman Publishing Co., Inc., USA.
- [2] Dominik Kesim, Lion Wagner, Jóakim von Kistowski, Sebastian Frank, André Van Hoorn, and Alireza Hakamian. 2020. Scenario-based Resilience Evaluation and Improvement of Microservice Architectures: An Experience Report. Universität Stuttgart.
- [3] Jóakim v. Kistowski, Nikolas Herbst, and Samuel Kounnev. 2014. LIMBO: a tool for modeling variable load intensities. In *Proceedings of the 5th ACM/SPEC international conference on Performance engineering* (Dublin, Ireland). Association for Computing Machinery, New York, NY, USA, 225–226.
- [4] Wagner, Lion et al. 2020. Code Ocean Capsule. [\[Clickable Link\]](#).