Building a simulator for the new version of Callbar

Miguel Dias do Rosário Pinto

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Duarte Nuno Jardim Nunes
Prof. Sónia Isabel Ferreira Dos Santos Rafael

Examination Committee

Chairperson: Prof. José Luís Brinquete Borbinha
Supervisor: Prof. Duarte Nuno Jardim Nunes
Members of the Committee: Prof. Rui Filipe Lima Maranhão de Abreu

October 2019
Acknowledgments

I would like to thank to the company that provided us space to work during the project phase, Talkdesk, specially Carlos Morgado for his technical knowledge and guidance and Fernanda Machado for being an amazing host. The team that composed this project for Talkdesk Scope, Ricardo Monteiro, Diogo Lopes and Carlos Silva a special thanks.

This innovative concept opened new paths, ideas and problems so I would also like to emphasize the teachers that accompanied us - Prof. Sónia Rafael, Prof. Pedro Oliveira and Prof. João Ferreira.

I would also like to acknowledge my dissertation supervisor Prof. Nuno Jardim Nunes which was the one that brought this new project to the University of Lisbon.

To all my friends and family that helped me in these last steps of this journey, specially Jéssica Bruno and my parents.

Finally, I would like to thank to all the participants in the early stages of the study that helped us in the user research.
Abstract

The evolution of engineering education has sought to bring students' experiences closer to the needs and challenges of the labor market. Two major trends have gained particular relevance in recent years: the interdisciplinary approach to solve even more complex problems and the approach to highly competitive global markets. The SCOPE project aims to introduce an innovative model of interdisciplinary collaboration across the different engineering specialties at Instituto Superior Técnico (IST), allowing dissertation/project students to collaborate in teams to solve real problems posed by companies. With this in mind I opted to choose Talkdesk as the company to do my work because I made an internship that involved me in the company and led me to help them (and me) to grow. The main goal of this project is to build a simulator tool that is able to integrate the Talkdesk system components. This work will facilitate the possibility to simulate the entire platform for newer versions of Callbar. The part that I developed during this project was the backend of the application where it was necessary to make the connections from our applications to Talkdesk staging environment and the core logic necessary to produce the responses to the requests from the frontend.

Keywords

SCOPE project; Talkdesk; Simulator; Callbar; Reliability; Scalability
Resumo

A evolução da educação em engenharia tem vindo a aproximar as experiências dos alunos às necessidades e desafios do mercado de trabalho. Duas grandes tendências ganharam especial relevância nos últimos anos: a abordagem interdisciplinar para resolver problemas cada vez mais complexos e a abordagem a mercados globais altamente competitivos. O projeto SCOPE visa introduzir um modelo inovador de colaboração interdisciplinar entre as diferentes especialidades de engenharia do IST, permitindo aos estudantes de dissertação/projeto colaborarem em equipas para resolver problemas reais colocados pelas empresas. Com isto em mente optei por escolher a Talkdesk como empresa para fazer o meu trabalho porque fiz um estágio que me envolveu na empresa e me levou a ajudá-los (e a mim) a crescer. O principal objetivo deste projeto é construir uma ferramenta de simulação que seja capaz de integrar os componentes do sistema Talkdesk de forma que, para trabalhos futuros, seja possível simular toda a plataforma para novas versões da Callbar. A parte que eu estive responsável durante o desenvolvimento do projeto foi o backend da aplicação onde foi necessário fazer a ligação da nossa aplicação para os componentes da Talkdesk que estão em ambiente staging assim como toda a lógica necessária para responder aos pedidos efetuados pelo frontend.

Palavras Chave

Projeto SCOPE; Talkdesk; Simulador; Callbar; Confiabilidade; Escalabilidade
# Contents

1 Introduction ......................................................... 1  
   1.1 Goals of the project .................................... 3  
   1.2 Talkdesk ................................................. 3  
   1.3 SCOPE Team ............................................ 4  
   1.4 Division of work .................................... 5  
   1.5 Organization of the Document .................... 5  
2 State of the Art and Technology .......................... 7  
   2.1 Contact Centers ........................................ 9  
      2.1.1 Brief Content of Contact Centers ............... 9  
      2.1.2 CCaaS Architecture Types ................... 10  
      2.1.3 Contact Centers as a Service ............... 12  
   2.2 Chosen Technologies for the Project ............... 12  
   2.3 Applications Related to our Simulator ........... 13  
      2.3.1 WebLOAD ........................................ 14  
      2.3.2 LoadNinja ....................................... 14  
      2.3.3 Apache JMeter .................................. 14  
      2.3.4 LoadView ....................................... 15  
      2.3.5 Locust .......................................... 15  
3 Talkdesk - The Client ............................................. 17  
   3.1 Talkdesk Internal Teams ............................... 19  
      3.1.1 Callbar Team .................................... 19  
      3.1.2 Sales Team ..................................... 19  
      3.1.3 Product Team ................................... 19  
      3.1.4 Cobalt Team .................................... 20  
   3.2 Callbar .................................................. 20  
      3.2.1 Callbar Features ................................ 21  
   3.3 Shadowing and Interview Users ........................ 22
4 System Design and Implementation
   4.1 Introduction .............................................. 27
   4.2 Simulator Features ...................................... 27
   4.3 Simulator Architecture ................................ 28
      4.3.1 Presence System .................................. 29
      4.3.2 RabbitMQ .......................................... 31
      4.3.3 Communication between Frontend and Backend .... 31
      4.3.4 Device states ...................................... 33
   4.4 Design Flows and Information Architecture ............... 34
   4.5 Simulator Interface .................................... 36
5 Conclusion .................................................. 39
   5.1 Reflections on IST-SCOPE .............................. 41
   5.2 Technical Conclusions .................................. 42
   5.3 Future Work ............................................ 42
A Extented Report ........................................... 47
List of Figures

1.1 Scope Team ................................................................. 4

2.1 Gartner Magic Quadrant where the leaders in the market are Avaya, Cisco and Genesys as of May 2018 ................................................................. 10
2.2 Gartner Magic Quadrant where the leaders in the market are Five9, inContact and Interactive Intelligence as of October 2015 ................................................................. 10
2.3 Talkdesk enters in the Gartner Magic Quadrant as a visionary as of October 2018 ................................................................. 11
2.4 Difference between multi-tenant and multi-instance architecture ................................................................. 11
2.5 Insight took from Stack Overflow Developer Survey 2018 [1] ................................................................. 13

3.1 Callbar Login Screen ................................................................. 21
3.2 Dialing screen in the Callbar ................................................................. 21
3.3 Incoming call where it is possible to go the profile of the caller ................................................................. 21
3.4 Contact Center Software Features with Callbar ................................................................. 22

4.1 Example of 2 incoming calls with 3 devices and the time labeled ................................................................. 28
4.2 Example of 3 agents and their occupancy levels ................................................................. 29
4.3 Overview of the Architecture of the system ................................................................. 30
4.4 Devices states and their possible transitions to the other states when a call arrives ................................................................. 34
4.5 Information Architecture ................................................................. 35
4.6 Flowmap of the Simulator interface ................................................................. 36
4.7 Landing Page ................................................................. 36
4.8 Screen that has the possibility to chose a test and the placeholders for the graph and device status ................................................................. 36
4.9 The simulator comes with predefined scenarios which can be chosen ................................................................. 37
4.10 In the custom screen it is possible to adjust all the values according to the user preference ................................................................. 37
4.11 Screen where it is possible to see the graph in real time where the events pop in the right row (accepted or rejected) and the device status being changed according to their actions ................................................................. 37
# Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Program Interface</td>
</tr>
<tr>
<td>AWS</td>
<td>Amazon Web Services</td>
</tr>
<tr>
<td>CCaaS</td>
<td>Contact Center as a Service</td>
</tr>
<tr>
<td>CEO</td>
<td>Chief Executive Officer</td>
</tr>
<tr>
<td>CRM</td>
<td>Customer Relationship Management</td>
</tr>
<tr>
<td>FBA</td>
<td>Faculdade de Belas-Artes</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>IST</td>
<td>Instituto Superior Técnico</td>
</tr>
<tr>
<td>IVR</td>
<td>Interactive Voice Response</td>
</tr>
<tr>
<td>KPI</td>
<td>Key Performance Indicator</td>
</tr>
<tr>
<td>QA</td>
<td>Quality-Assurance</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software as a Service</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>SME</td>
<td>Small and Medium-Sized Enterprises</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>UX</td>
<td>User Research</td>
</tr>
</tbody>
</table>
Introduction

Contents

1.1 Goals of the project .............................................. 3
1.2 Talkdesk .......................................................... 3
1.3 SCOPE Team ..................................................... 4
1.4 Division of work .................................................. 5
1.5 Organization of the Document ................................. 5
The Capstone project is an innovative idea brought to Instituto Superior Técnico (IST) with the purpose to give students a culminating experience to end the academic school and final thesis. This kind of project is a multifaceted assignment because it allows the students to work with colleagues from other areas and universities.

Capstone as the name implies as a core idea that will be the main goal that the students will aim to achieve or solve. For that, it is necessary to adopt the right strategies and think critically to face the problems that arise while developing the ideas. Since this work is done in a team, there are multiple of soft skills that are developed during the project phase, such as cooperation, oral communication, time management, planning and goal definition.

1.1 Goals of the project

This project started with the goal to build a test harness that could provide information and metric analysis for the new version of the Callbar. The provided description for the problem was:

“The Capstone Test Harness for Internet of Things (IoT) Messaging System pretends to operate an IoT system for thousands of devices with strict requirements of latency. The main factors in this system are reliability and scalability to accommodate the production loads. This project wants to build a Test Harness that is capable of reproducing realistic loads in the system. To this end the system must be able to simulate thousands of devices in specific network conditions, with degradation, loss of signal, etc.”

As it’s possible to observe in the description, the proposed goal was to achieve a system that could recreate realistic loads to demonstrate scenarios of possible failures with the release of the new version. With this identified scenarios it would be possible to anticipate and fix this problems before the release, which would be the perfect way to delete bugs in production.

Since the new version wasn’t ready when we started to work on this project, they only had a mock application, we (our team and Talkdesk) adapted the goal to fit with what they had provided and what we could accomplish in the remaining time.

So this project gained a new form and our main purpose changed to build a simulator which is capable of representing the loads that the real system will have in the future with the components that they had to date (which will be described later in Chapter 4).

1.2 Talkdesk

Talkdesk [2] is a company founded in 2011 by Tiago Paiva (who also is the current Chief Executive Officer (CEO)) and Cristina Fonseca both former students of IST. They realized that the paradigm of contact centers was outdated, so they entered in a contest created by Twilio to develop their own ideas
in 10 days using the provided Application Program Interfaces (APIs) allowed in the contest.

It’s a cloud-based contact center software provider in the digital industry of Software as a Service (SaaS). So the main product that Talkdesk sells is precisely an online call center aiming for increased customer experience, reliability, data security, quality and therefore increasing clients’ satisfaction, productivity and profit.

They have several offices in Porto, Coimbra, Lisboa, Salt Lake. Its headquarters are based in San Francisco, in United States of America.

The IST-SCOPE project is closely in contact with Lisbon’s office.

This company will be the focus of our project where we will target their product - Callbar (detailed in Section 3.2) - that will need to be studied, known, addressed and treated as a mix of client and partner.

1.3 SCOPE Team

The multidisciplinary team that worked with Talkdesk is composed by the following four people:

Carlos Silva Student from Faculdade de Belas Artes in Master Degree in Communication Design and New Media - UX/UI Designer;

Diogo Lopes Student from Instituto Superior Técnico in Master Degree in Information Systems and Computer Engineering - Software Developer (frontend);

Miguel Pinto Student from Instituto Superior Técnico in Master Degree in Information Systems and Computer Engineering - Software Developer (backend);

Ricardo Monteiro Student from Instituto Superior Técnico in Master Degree in Industrial Engineering and Management - Project Manager.

Each member of the team will contribute to all phases of the project but different members will adjust their efforts according to their expertise and academic field. The software developers will be responsible for the development of frontend (Diogo Lopes) and backend (Miguel Pinto) of the application; the de-
signer (Carlos Silva) will be responsible for the creation and visualization aspects of the user interface concerning the results of the tests; and the project manager (Ricardo Monteiro) will be responsible for the team’s planning and results analysis resulting from the application project.

### 1.4 Division of work

In Table 1.1 it is described what was the work that each one of the elements of the team mainly did through the SCOPE project. We all helped each other in our tasks but we had our major part attributed to a specific element in the team as it is possible to see in the table.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Table 1.1:</strong> Division of work through the members of Talkdesk SCOPE team</td>
<td></td>
</tr>
<tr>
<td><strong>User Research</strong></td>
<td>Each element of the team contributed equally to this task.</td>
</tr>
<tr>
<td><strong>Project Management</strong></td>
<td>We all had part on this and deciding which tasks we should prioritize but it was <strong>Ricardo Monteiro</strong> who managed Jira Platform and created epics, stories and the rest of the elements to reflect the work that we were doing.</td>
</tr>
<tr>
<td><strong>Frontend Design</strong></td>
<td><strong>Carlos Silva</strong> main task was to design the wireframes and pick the right elements that matched closely with Talkdesk Cobalt components.</td>
</tr>
<tr>
<td><strong>Frontend Implementation</strong></td>
<td><strong>Diogo Lopes</strong> did his major work in this part, where he translated the concepts and ideas that Carlos Silva had to actual code.</td>
</tr>
<tr>
<td><strong>Backend Implementation</strong></td>
<td>My main focus on the project was to design, create and implement the backend of our application.</td>
</tr>
</tbody>
</table>

### 1.5 Organization of the Document

This thesis is organized as follows: Chapter 1 introduces the project and explain what are the goals of the project, who are the teams behind the project and the division of work; Chapter 2 covers the background of Talkdesk and the reasons behind their exponential growth, our picks for the technological stack and related applications; Chapter 3 describes the meetings that we attended during the research phase of the project as well as a detailed explanation of what is Callbar and how we proceeded to use shadowing; Chapter 4 describes the architecture of our simulator mainly the part that was done by me (backend); Finally in Chapter 5 it’s the conclusion of the project and features to be added and improved in the future.
2

State of the Art and Technology

Contents

2.1 Contact Centers ...................................................... 9
2.2 Chosen Technologies for the Project .............................. 12
2.3 Applications Related to our Simulator .......................... 13
2.1 Contact Centers

Contact Centers [3] are a central point from which all customer contacts are managed. From there it is possible to have multiple call centers do diversify the channels used to enter in contact with the clients, creating an omni-channel base for the enterprise’s overall Customer Relationship Management (CRM). Contact Centers are able to have inbound and outbound calls (or an hybrid version).

The traditional Contact Centers had their infrastructure on premises which required to have specialized staff and have additional costs maintaining the infrastructure. With the appearance of the Cloud it is possible to outsource these tasks and have the ability to have elastic computing, which reduces the costs comparing to the traditional way. Another advantage of Cloud infrastructure is that it allows to access anywhere with a single restriction on Internet access. So, for example an agent wouldn’t necessarily need to be in the facilities to answer and make calls.

Contact Center as a Service (CCaaS) are built natively in the cloud so the providers maintain and develop the software which gives more flexibility and better customer experiences. The main purpose of having the infrastructure on the cloud besides the previous advantages is that it allows to have intelligent routing, this means that it is possible to have the calls assigned to the most suitable agent, which is frequently denominated Skills-Based Routing.

2.1.1 Brief Content of Contact Centers

Contact Centers as stated before started by having on premises infrastructures to host their software which required a bigger investment. The major players, as we can see in the Figure 2.1, until the late 2000, were Avaya, Cisco and Genesys.

The first companies that saw the opportunity to host the contact center software in the cloud were inContact, Five9 and Interactive Intelligence (Figure 2.2). The agents could now use any computer with access to internet to make outbound and inbound calls. Gartner included in this round new benefits such as:

- Lower IT maintenance costs;
- Ability to IT headcount;
- Flexibility for agents to work from home;
- Ability to have elastic computing - scale up and down according to the demand of the contact center.

With the entry of these new companies the market started to change from the on premise infrastructure to the cloud. In these years inContact was purchased by NICE and Interactive Intelligence
was bought by Genesys. This meant that the big companies wanted to accompany the new trends in technology and more important reduce the costs while maintaining or increasing revenue.

With the release of the second wave of CCaaS the companies were struggling with the deployment of new features and improvements over what was deployed in the cloud. This meant big downtime events and difficulties integrating with other 3rd party applications. These were the times were Talkdesk entered in the market above other players since their software was built natively for the cloud. Advantages like small downtime and rapid integration of new features and 3rd party CRMs were the main points of sale for many clients and Gartner (see Figure 2.3).

### 2.1.2 CCaaS Architecture Types

The CCaaS is maturing with all the players in the market that keep to evolve and provide solutions that cope with clients perspectives. Because of that different types of architectures are designed according to the companies needs and possibilities. Two make a distinct from the others (Figure 2.4), which are:

- **Multi-tenant** - All users share a common (single) software instance. This means that multiple customers use a single instance of an application running on a single instance of an operating system on a common hardware platform;
- **Multi-instance** - on the other hand, is where multiple customers run their own separate instance
of the application and operating system, running on a separate virtual machine, all on a common hardware platform. So, each user receives its own virtualized software instance on shared hardware.

It’s important to understand that both architectures leverage common hardware and other resources for economies of scale, for example, provisioning tools, portals, data centers, racks, common equipment and servers, which results in competitive pricing and financial benefits for subscribers.

**Figure 2.4:** Difference between multi-tenant and multi-instance architecture

While the multi-tenant may be cheaper because there is only one instance for multiple tenants and the
cost to add new features is lower, it lacks the ability to customize the application according to costumer needs, which is where the multi-instance wins.

2.1.3 Contact Centers as a Service

The future for companies that aim to put their products in the cloud will be brighter since this is where technology is going. The market is expected to grow $50B by 2025 [4] and there is only 36% contact centers that are using the cloud [5], so there is an enormous space to gain market share and innovate.

The first companies that adopted this kind of technology were the Small and Medium-Sized Enterprises (SMEs) since they had less invested in on-premises contact centers. But with the improvements and advantages that cloud contact centers have to offer, such as:

- Uptime agreement close to 100% (Talkdesk already offers 100% uptime);
- Flexibility to give calls to agents for them to work - this enables a new paradigm of intelligent call routing, for example how should the rules be defined to route the calls to certain agents;
- Integration with 3rd party applications;
- Interactive Voice Response (IVR), call recording;
- Reporting that gives feedback on how the contact center is developing and if it hits the goals.

All this advantages give clients benefits while improving their implementation timeline when adopting new CCaaS [6]. It also gives speed and agility, with the ability to update the customer experience as expectations change and as new capabilities are introduced. Since this solutions are easier to run and very scalable the businesses that opt to use this service reduce their costs in the infrastructure by 15 to 35 percent.

2.2 Chosen Technologies for the Project

Our project had the requirements to build an application that was scalable and could handle the loads of thousands of devices when simulating the scenarios. We also had time constraints, as it’s normal in business environment projects. So we had to comply with these requirements and deliver in the time frame, for that, through the amount of programming languages that are available today we opted to go with:

- Python - to implement the backend of the application;
- Javascript, HTML and CSS - for the frontend of the application as well as the bridge from the backend to frontend.
We chose JavaScript, HTML and CSS since we were most familiarized with them and they are the most common languages today. We can see in Figure 2.5 that these 3 languages are in the top choice of most users in the development world.

<table>
<thead>
<tr>
<th>Language</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript</td>
<td>69.8%</td>
</tr>
<tr>
<td>HTML</td>
<td>66.5%</td>
</tr>
<tr>
<td>CSS</td>
<td>65.1%</td>
</tr>
<tr>
<td>SQL</td>
<td>57.0%</td>
</tr>
<tr>
<td>Java</td>
<td>45.3%</td>
</tr>
<tr>
<td>Bash/Shell</td>
<td>39.8%</td>
</tr>
<tr>
<td>Python</td>
<td>38.8%</td>
</tr>
<tr>
<td>C#</td>
<td>34.4%</td>
</tr>
<tr>
<td>PHP</td>
<td>30.7%</td>
</tr>
<tr>
<td>C++</td>
<td>25.4%</td>
</tr>
<tr>
<td>C</td>
<td>23.0%</td>
</tr>
<tr>
<td>TypeScript</td>
<td>17.4%</td>
</tr>
<tr>
<td>Ruby</td>
<td>10.1%</td>
</tr>
<tr>
<td>Swift</td>
<td>8.1%</td>
</tr>
</tbody>
</table>

Figure 2.5: Insight took from Stack Overflow Developer Suvey 2018 [1]

For the backend the main reason we chose Python was because of our familiarity with this language. We had one better choice which would be GoLang by Google. This language is more recent than Python and have many advantages over it, such as more performance, scalability (Concurrency and Parallelism) and it’s a language designed to produce systems at a large scale. Since this is a statically typed language it gives more control to the programmer building and developing the systems. But since we had limited time to achieve the goals for this project and none of us had the experience on this language we opted to go to a known language and open the opportunity for the system to be implemented in GoLang in a future iteration for better performance.

2.3 Applications Related to our Simulator

There are several applications in the market that simulate and create the load to an application to test, categorize and describe its behaviour. In the next subsections it will be described briefly some of the applications that were researched through the process of creating our own and influenced us to create and develop some features such as the ability to load test the scenarios (custom scenarios or pre-
defined) and to customize the parameters of each scenario through a web Graphical User Interface (GUI). The distinct part is that the simulator has integrated Talkdesk components and it's adjusted to their needs to assess the potential problems that might arise in future versions.

2.3.1 WebLOAD

WebLOAD [7] is a load and performance testing tool specifically designed for web applications. It accommodates the enterprise-grade segment in the load performance tools since it's very robust. Its main features are ease of use and flexibility, so it allows to be used with several protocols and technologies which facilitates the work when doing large load assessments to an application.

The dashboards are divided in three sections:

1. Main Dashboard - The Main Dashboard provides a high-level view of test results, including failure counters, failures and errors over time, and other useful measurements;

2. Transactions details Dashboard - A Transactions details dashboard lets you view the performance of transactions throughout the session as well as success/failure over time;

3. Log dashboard - The Log dashboard lets you correlate log events with the timeline progress of the session.

2.3.2 LoadNinja

LoadNinja [8] is a tool that lets the user to create scriptless load tests that improves the time spent in generating them. Its best advantage is that it can reduce test script creation time by 60% using artificial intelligence in recognizing objects in applications and by using their component InstaPlay Recorder which eases the process of loading the tests. It also allows to analyze in real time to tackle performance and bottlenecks faster. A quick summary of their features is: Record and Playback Scripts; Load tests on real browsers; Analyze the performance results; Full coverage with the load tests; Possibility to test in the local network; And it automates within the pipeline CI/CD.

2.3.3 Apache JMeter

Apache JMeter [9] is an open source application developed in Java that can be loaded in the server and in the network to analyze and check the performance under different scenarios. It supports a wide range of protocols and supports variable parameterization, assertions (response validation), per-thread cookies, configuration variables and a variety of reports. Apache JMeter works by adding plugins to increment its functionalities.
2.3.4 LoadView

LoadView [10] is a tool similar to LoadNinja in the way that it runs the tests in a real browser instead of simulated ones. It’s a tool that needs to be payed, but their price model goes in the way of pay for what you use, since it is 100% cloud based. The advantage of being based in the cloud is that it can be deployed in minutes and it can be fully configured from there.

2.3.5 Locust

Locust [11] is also an open source tool for load testing. This framework allows to load test an application using a well known code language - Python. It’s all customizable through code, scalable and distributed, and the great benefit is the parameters can be changed through a web GUI.

In short, we can see that through the advantages that CCaaS offer, more and more companies are entering the market with similar products and offering better Service Level Agreement (SLA) conditions. With the requirements we have for the project as well as the time constraint for project completion, we chose familiar programming languages - Python, JavaScript, HTML and CSS. We are aware that there is at least one better language, being GoLang, but we chose not to pick it since none of us had experience with it.

Through applications related to our theme and project goals, such as WebLOAD, LoadNinja, Apache JMeter among others, we have been able to get the common points to build an interface for us with the same features that are offered in the market adapted to the needs of Talkdesk. We can integrate the customization of the scenarios through the different parameters and the possibility to see in real time the evolution of the system state.
Talkdesk - The Client

 Contents

3.1 Talkdesk Internal Teams ............................................... 19
3.2 Callbar ................................................................. 20
3.3 Shadowing and Interview Users ................................. 22
During the first semester our work involved getting to know Talkdesk internal teams as well as their product - Callbar. In the next section it will be detailed who are those teams and what they do. In the following section it will be described what is Callbar.

3.1 Talkdesk Internal Teams

Almost every week during the first semester we had meetings scheduled with the teams that support, create and sell - Callbar. Those teams are: Callbar, Sales, Product/Presence System and Cobalt. In the next subsections I’ll summarize what were the topics that we talked during this meetings and what do the teams do in their work day.

3.1.1 Callbar Team

This team is responsible to develop the application that all agents will use to communicate with their costumers, in the call center that uses the Talkdesk platform.

It was stated to us that our work will focus in the new version that will be released, so there is no application yet to be tested, only interface mocks. Since our work is dependent on their work our final product used the component that they could provide in the available time-frame, which is the orchestrator to manage the routing configuration and to create interactions.

3.1.2 Sales Team

In this meeting it was presented to us how the sales team sells the Talkdesk product to their potential clients and what were the greater pillars that they focus on selling the product to their clients:

1. It’s easy to implement;
2. It’s easy to use;
3. It’s a platform that was made to be configurable and not programmed in the client side;
4. It enables the supervisor to analyze the Key Performance Indicators (KPIs) of his business.

3.1.3 Product Team

The product team is responsible for creating new features (and future versions) and resolving/fixing bugs that exist in the current production version. During this session it was shown how the current version works, mainly how to do calls, how to use the feature to transfer calls (blind - without a previous chat to the person that is receiving the call and warm - with a conversation to give the context to the next agent
that will pick the call - transfer) and how does the User Interface (UI) looks and feels when using the product.

In the second part of the meeting it was shown the overview of the architecture that they have and what they want to do with the release of the new API - Presence - that will enable them to have more control on their side without being dependent of external vendors. They also gave context about the current version, what are the main problems that they want to mitigate with the release of the next version. In Section 3.2 I’ll go in depth how does the current and the future versions work and will work.

3.1.4 Cobalt Team

In this meeting we got presented to a team that does the bridge between the engineering side of the Callbar and the UI side. The main purpose of them is to create and adapt components to add to the product maintaining the same guidelines and styles.

This team provides a way for other teams to uncouple the work of creating components that need to meet criteria to maintain the same guidelines for cohesion of work.

3.2 Callbar

Callbar [12] (Figure 3.1, 3.2 and 3.3), Talkdesk’s product, is a software that enables users to make and receive calls from anywhere in the world with internet connection on their desktop. This application was done to be unobtrusive and always accessible in case there is work to do with the application. The main benefits of using Callbar are:

- Always accessible, being normally in the bottom of the screen to be easier to make actions within;
- Give context every time there is an incoming call to reduce the time spent by the agents getting the context of the call. It allows to view the previous history that the client had as well as detailed information about him to ease the process of the current call;
- This is a lightweight application because it wants to be unobtrusive and enables the agent to do other work without being worried about the response time of the applications to answer to the customer needs;
- One of the great benefits is to fully integrate with known CRMs, like Zendesk and Salesforce, which gives the agent lots of detailed information about the client;
- It supports Click-to-Call [13] (a feature that enables the users to call directly numbers in web-pages without having to copy-paste to the application. The required software is an extension that intelligently scans the sites searching for numbers), screen pops, call logging.
3.2.1 Callbar Features

These are the features that are currently available to anyone that uses Callbar:

1. Make and receive calls - This is the core of the application, it works as a normal phone (or softphone) where you type the number you want to call and where you can see who is calling to you;

2. Set agent availability - There are at least 4 states in which an agent can be: Available, Away, After Call Work, Offline. Swapping between the states is done automatically by the application when a user is logged in or logged off; when he receives a call and answer it; and when he terminates the call and is registering the work that he did. It is also possible to define custom states to better translate the agent availability to the company that he is in;

3. Mute or hold a call - This features can be used to search for information while the client is waiting;

4. Transfer a call - There are two types of transfers that can happen while in a call with the Callbar application: Warm Transfer and the Blind Transfer. As explained in Section 3.1.3 the difference lies that in Warm Transfers there is a conversation previous to the transfer to give context to the next agent that will pick the call;

5. Initiate a call conference - When it is necessary to integrate another agent, for example, a technical one in the call to solve some problems while the first line support keeps track of it;
6. Merge or create new contacts - This is a feature to tidy up the contact list and ease the navigation and search project;

7. Pause the call recording - If there is any sensitive content that the client doesn’t want to be recorded, this feature comes in handy by preventing that what he says doesn’t get stored;

8. Add notes to a caller’s activity history - It allows to keep track of what is said during the call to facilitate the work of the agent when he is in the state - After Call Work;

9. Access the contact’s profile page in the integration of the users choice - One of the greatest selling points of Callbar, is that it allows to integrate seamlessly with CRMs to make the job of the agent when searching for information about certain clients easier.

![Figure 3.4: Contact Center Software Features with Callbar](image)

### 3.3 Shadowing and Interview Users

During our first stage of the project we planned and did user research through shadowing and small interviews (after the shadowing was completed). Our foundations for this assessment were given by the tutors which accompanied us during this phase of the project.

We used shadowing as a way to understand how the users used Callbar because we wanted to simulate the behaviour of the users in later iterations of the product. In the first version of the Scope project we also aimed to create different scenarios with their behaviours to have multiple scenarios for the Quality-Assurance tests that we pretended to do in the application. It also is a technique that we adopted because it allows us to closely write and understand what the agent does in his daily life, not what his role description says. When our Scope suffered changes we used the results of our analysis to create profiles for our Simulator, which gave us variability in the answering of the calls (it’ll be addressed later in Section 4.2).

Briefly, the 10 recommendations that we followed are [14]:

1. Proactively engage issues with shadowees ahead of time;
2. Prepare for embodied shadowing;

3. Take classes or hold discussion on the emotional side of qualitative methods;

4. Pack a “shadow kit”;

5. Plan to follow the rules, at first;

6. Play around with strategies for notetaking;

7. Dance in the doldrums - this means to take rests during the work;

8. Locate or create social support;

9. Mitigate the anticipation of shadower-as-betrayer;

10. Exit the field mindfully;

Categorizing this recommendations 1, 2, 3 and 4 belong to **Arriving**; 5, 6, 7, 8 belong to **Shadowing**; 9 and 10 belong to **Leaving**.

The purpose of these recommendations were to give us an easier time and to give confidence to the shadowee that we were not there to judge them but to get context to build a better application for them.

This User Research (UX) was divided in two steps: **Shadowing** (Observation Research) and **User Inquiry** (Interview Research). In both of the research methods our focus was to gather information related to workflows, methods, and behaviours of the participants. This would be anonymous and the participant would be briefed about the project and its objectives in the start and at the end the data will be verified with the consent of the participant to be used in the project.

The Shadowing had a minimum of time of 30 minutes where the average of our work rounded 45-50 minutes. There was no interactions while the work was being conducted and the actions that the participant took were registered one in each line.

The User Inquiry would take approximately 30 minutes and would go from a top down approach where first were asked broad questions and than the questions were narrowed to the context of the previous answers or other subjects that the participant deemed relevant to explain to us.

This interaction with actual users of the Callbar gave our work a broader range of topics to cover and implement in future versions of our Simulator. It was also an interesting experience to be able to watch what are the main tasks of a certain team and to network with new colleagues. It also facilitated us to better identify and parameterize the scenarios to give context and information similar to the Callbar environment.
System Design and Implementation

Contents

4.1 Introduction ............................................................... 27
4.2 Simulator Features ...................................................... 27
4.3 Simulator Architecture ................................................ 28
4.4 Design Flows and Information Architecture ......................... 34
4.5 Simulator Interface .................................................... 36
4.1 Introduction

The main goal of the project, as stated in Section 1.1, is to create a simulator that is capable of representing the loads that the real system will have in the future with the components that they had to date.

Despite the changes that we had in our scope project we adapted well and created a product that can be used by Talkdesk to include the future components when they get released.

This simulator allows the users to define or to chose a predefined scenario to run and at the end retrieve results and analyze them if they deem so.

The following sections will describe in detail the What, How and Why of the Simulator.

4.2 Simulator Features

Taking into account what we learned in the sessions with the teams and the knowledge that we got from the shadowing we defined some features that we must have to create the initial form of the application. We wanted to have the possibility to create scenarios to give freedom to the users to explore the metrics that we selected to parameterize the scene. We gave the possibility to change the following parameters:

- Number of Devices;
- Number of Calls;
- Call Duration (ms);
- Call Pickup Delay (ms);
- Call Standby (ms);
- Profiles.

The Number of Calls and Number of Devices is the core of the simulator, with them we can have more or less calls and devices being that the number of calls is the number of total incoming calls and which any device that is in the state waiting for a call can answer. The Call Duration is the time that each call will have in the simulator with a corresponding agent; The Call Pickup delay is how much time the agent takes to pick the phone; And the Call Standby is the time between incoming calls. In Figure 4.1 is an example explaining what is each one of the custom parameters that we made available. The reason for choosing this metrics is that the variability that we could create depended on the time interacting with the calls.
The *Profile* item that we made available to customize has 4 options: *Shy, Champion, Normal* and *Sleepy*. To each of the options we associated a call answer percentage which is the percentage at which the agent answers a call.

The values are:

- Shy - 10% rate - corresponds to a *shy* agent which doesn’t answer to many calls;
- Champion - 90% rate - corresponds to an extroverted agent and very talkative;
- Normal - Value retrieved from the Normal Distribution - from the name this is the average of the agents, it has an approximate 50% chance of accepting the call;
- Sleepy - Value retrieved from the Exponential Distribution - this agent has a low rate at the beginning of the day and that rate increases as the day goes on.

### 4.3 Simulator Architecture

In this section I’ll explain what is the architecture that is being used in the simulator and why is it with that structure. The backend of the application and its core to make the communication with the Talkdesk components were made by me. My colleague (frontend developer) engineered the frontend with the help of the Designer Lead. So, also in this section, there will be my individual contributions to this project.
4.3.1 Presence System

In the beginning of the process it was presented to us by the Product Team (Section 3.1.3), namely Tiago Sousa, the **Presence System**. The Presence System is a new component that is still being developed by Talkdesk that will allow teams like Quality-Assurance (QA) to test the application without being dependent from providers (for example Twilio). This core component is the bridge which will allow to change agent status to make them available to answer calls or to subscribe to notifications whenever there’s a change in presence. With this piece it will also be possible to add a new metric that will have relevance when measuring the availability of the agents to answer to the calls which is the **Occupancy** of each one.

The occupancy will be filled with the weights of each channel that the agent is using in the moment that a call arrives. This means that each channel (any communication channel that the agent is using inside Callbar application which are included integration that might happen in the future), for example, Messages, Callbar or any other social network will leverage the occupancy level (Figure 4.2). With this component it would be possible to add the following operations to the simulator:

![Figure 4.2: Example of 3 agents and their occupancy levels](image)

- Retrieve a single agent presence status (any agent for the given account);
- Subscribing notifications for an agent presence status change;
- Subscribing notifications for the presence status change of all agents of an account;
- Updating any single agent status;
• Modifying an agent occupancy.

Our system was modelled in the beginning to accommodate this component since it was a critical piece to define the states of the devices when they were in call or not. But since the developments suffered some time changes we adapted our infrastructure with a simple module that could decide and define the states of the agents when a call arrives, which includes our Profile parameter.

So, in Picture 4.3 we can observe the overview of the architecture of the system that was built. In the bottom of the image (blue and purple components) are the pieces that we developed during this project which are divided in two: Frontend and the Backend. The Backend has the list with the device objects and their respective states and it has also the consumer which is a separate thread that is created in the beginning of the program which will be in loop consuming the messages that appear on the queue. In the top of the image it’s the RabbitMQ exchange with the associated queue for our system.

![Figure 4.3: Overview of the Architecture of the system](image)

Since we have a message broker and messages are being published and consumed we are in a scenario of Producer/Consumer where the messages go through the RabbitMQ.
4.3.2 RabbitMQ

RabbitMQ is a messaging broker that gives applications a common platform to send, receive and store messages while in the life cycle of an application. It provides: **Reliability**, **Flexible Routing**, **Clustering**, **Highly Available Queues**, and many more features that were necessary to use through our development.

The reasons that made us use RabbitMQ as our platform for messaging is because Talkdesk already use it as a core component, it is Open Source with a big and pro-active community and it is widely adopted from small start-ups to large enterprises. Since the message exchange was already deployed in an Amazon Web Services (AWS) by Talkdesk we didn’t have much space to do tests and evaluate different configurations since this was the scenario they gave us.

The way that the messaging system is setup is with a direct type exchange. This means that every message that is published to the exchange with a certain routing key, in our case is #, will be directed to the queue that has that binding key that matches.

In this way every time an interaction is created the process will be this:

1. POST the interaction file that has the properties to create one incoming call in the system to the Orchestrator (this component is outside of the SCOPE project but it’s in charge of creating the interactions and distributing them hover the other components that we abstracted here in the project);
2. Exchange routes the message (direct type) to the key that is binded with the #;
3. The consumer that is a thread that was created in the beginning will listen and consume the messages as they arrive to the queue;
4. The messages are then processed by the backend and the response to the request is created.

4.3.3 Communication between Frontend and Backend

As we are developing a Simulator, we need information in real time to be passed between the backend and frontend. For this purpose we opted to go with Flask-SocketIO which is a library compatible with the language that we chose (Python 3) and allows for low latency bi-directional communications between them (frontend and backend). With this library we can transmit events between them as soon as there are any new activity to be presented in the interface of the Simulator.

The first request will be handled by the function `handle_my_custom_event()` (Listing 4.1) which is binded to receive events that have the tag `devices`. The request will have a JSON object in the body with the parameters defined for the scenario to be run in the Simulator. This method will have the following order of operations:
1. Clear the structures that store the devices object

2. Create the devices that were defined by the user

3. Send an update to the frontend to display the list of the devices

---

**Listing 4.1: handle function for creating devices**

```python
1 clear_devices.dict()
2 create_devices(int(json_response["number_devices"]),
3     json_response["profiles"],
4     json_response["call_duration"])
5 send()
```

After this first three steps it’s when the interactions start to be created with the parameters defined and the amount desired (Listing 4.2. The messages will be retrieved from the RabbitMQ queue by the consumer thread that was created in the beginning and than they will be parsed and transmitted to the frontend through the SocketIO library. So the following steps are:

4. Enter the loop that will produce the calls (line 2);

5. Create the interaction (line 7);

6. Retrieve the output from the queue - which is a message in JSON format that gives endpoints to accept or reject the call (comes from the Talkdesk Orchestrator) (line 10);

7. Update the structures that contain the devices to PENDING (line 13 to 18);

8. Send the information to the frontend with the updated status (line 20);

9. Use the delay to pick up a call and define a timer to better simulate the environment (line 23 to 29).
Listing 4.2: while loop described in steps 4 to 9

```python
i = 0
while i < int(json_response["number_calls"]) :
    print("---------------------")
    print("New Interaction")
    print("---------------------")
    # Creating the call
    create_interaction()

    # Accepting or rejecting the call
    output = consumer.get_queue().get()

    # Changing the status of device to pending
    for device in device_dict[WAITING_CALL]:
        device.state = PENDING

    # Changing all the devices in the waiting list to the pending
    device_dict[PENDING] += device_dict[WAITING_CALL]
    device_dict[WAITING_CALL].clear()

    # Waiting time until answering or rejecting the call
    time.sleep(int(json_response["call_pickup_delay"])/1000) % 60

    assignment_routing(output, json_response["call_duration"]) send()

    # Waiting time until the next call
    time.sleep((int(json_response["call_standby"]) / 1000) % 60)

    i += 1
```

### 4.3.4 Device states

I’ve mentioned about the state of the devices before and now I’ll explain them. There are three states that the devices can be (Figure 4.4): Waiting Call, Pending and On Call.

As we can see in Figure 4.4 the transition between them depends on the incoming calls and the
actions that were taken for that call. Every device is initialized with the state of Waiting Call, this means that the device is ready to receive calls when they are assigned to it. After a call is assigned the device will transit to the state Pending. In this state the device will have two possibilities: accept or reject the call. If the call is rejected the device will return to the Waiting Call state, if it accepts the call it will transition to the On Call which represents when a device is with a call. After the defined period for the duration of the call it will transition back to the Waiting Call.

4.4 Design Flows and Information Architecture

In this section it’ll be shown the information architecture and the flows of the actual usage of the interface.

The Figure 4.5 shows the information that is presented in the initial screen as well as the two components that the interface has: Customization Screen (Runner) and Simulator Screen (Visualizer). In the Runner component we can have two separate components which are: the Presets Scripts where it’s defined predefined scenarios with parameters values that we concluded that are relevant; and the Custom Script which gives freedom to the user to insert his own values for his custom scenario and execution in the simulator. The Simulator Screen is divided in three sections: the real time timeline where the events pop when they are received; the device status where it is possible to see the changes in their status and how many devices are in the current testing scenario; and the test actions - which are the stop function and the new test to execute a new scenario.

The Figure 4.6 represents the Flowmap of the interface since the landing page where the loading wheel appears through the phase where the simulator is already running showing the incoming calls and the actions that the devices took to that calls. Since this image represent the wireframes that were created some small changes and fine tuning were applied during the development of the interface. The layout in the bottom section where the devices are got some name changes and the action section
too. Briefly, what the image describes is the following: first the user lands in the landing page when he accesses to the web application. Then after the page loads it shows a mock of where the 3 main components will be (that were described in the information architecture image - Figure 4.5). When the user clicks in the action section it will be presented a window where it is possible to select scenarios or insert values for the parameters. After that when the scenario is defined, it starts to run showing in real time the information that comes from the backend with the defined values that the user selected.
4.5 Simulator Interface

In this section, it will be presented the face of our project, where it is possible to see the screens with the colors already selected and the simulator running with the input parameters.

Figure 4.6: Flowmap of the Simulator interface

Figure 4.7: Landing Page

Figure 4.8: Screen that has the possibility to chose a test and the placeholders for the graph and device status

Figure 4.7 and Figure 4.8 are the first screens that the user sees. In Figure 4.8 it’s possible to see
the placeholders for the components to give a grasp and a sense of how the simulator will be when it starts to run.

**Figure 4.9:** The simulator comes with predefined scenarios which can be chosen.

**Figure 4.10:** In the custom screen it is possible to adjust all the values according to the user preference.

Figure 4.9 and Figure 4.10 appear when the user click in the placeholder that has Run Test, with the corresponding screens in each of the tab (Preset and Custom). The user can see the values that the parameters have when selecting each of the predefined scenarios; and in the custom tab he can insert the values that he wants to run his custom script.

**Figure 4.11:** Screen where it is possible to see the graph in real time where the events pop in the right row (accepted or rejected) and the device status being changed according to their actions.

In Figure 4.11 we can see the simulator in action where in the top left corner are the actions that are possible to execute, the pause and end test button, information about the execution time and calls already completed; In the bottom we can navigate and filter the devices by their state; And finally in the top right corner the graph that is being constantly updated with the events that the frontend receives.
Conclusion

Contents

5.1 Reflections on IST-SCOPE .................................................. 41
5.2 Technical Conclusions ......................................................... 42
5.3 Future Work ................................................................. 42
In this chapter it will be made a reflection of how the work went, what are our accomplishments as a team and individual and how the SCOPE Project and the developed application can be improved in the future.

5.1 Reflections on IST-SCOPE

This project aims to bring students closer to the actual market and the state of the art. It introduces a new model of interdisciplinary collaboration to solve problems posed by companies. With this in mind we had to answer to a problem where Talkdesk needed a platform that could test their newer product without being dependent on providers such as Twilio. This means that when it was necessary to make a call it wouldn’t be needed to make an actual call but encapsulate this behaviour in such a way that the system would keep functioning and its behaviour would keep intact. This problem was also raised because every time Talkdesk wanted to test their product the costs were significantly high because Twilio charges for each call made.

So our answer for this problem was to encapsulate and mock the calls in our Simulator that could show information in real time about the events that occurred in the system and if possible give some feedback of what happened in each scenario defined by the user.

The Scope of the project suffered some changes along the way because of time constraints. Our simulator targeted the new version of the Talkdesk, which isn’t publicly available and that's one of the reasons why our scope suffered changes because some components that were to be included weren’t ready at the time of the development. It started to include the integration with the newly Presence API which could manage the status and define a occupancy level for devices. It got later reformulated to create a simulator in which we would manage the status of the devices internally when the calls were received and the appropriate action was taken. The research that we did in the first part of the project was expecting different requirements than in the later phase. We firstly accounted to include the behaviour of the agents in the application such as their reactions in short time and long time while answering the calls. We worked on the research that we made and turned the information into four different profiles that could fit the behaviour of the agents (shy, champion, normal and sleepy).

After this research we than started to divide the work between us where I took the development of the backend of the application. The backend of the application was responsible to create the interactions of the devices with the given input parameters that were inserted by the user. It collected the messages from RabbitMQ that were put by Talkdesk’s components, namely the Orchestrator, and created the events in real time to be passed to the frontend where they were displayed.

Since this project is an innovative model we were the first ones to get in touch with it and it was a great experience to have contact with Talkdesk and professors from a different university - Faculdade de
Belas-Artes (FBA). Aside from the fine tuning that this project needs, such has defined objectives and materials to be delivered it went very well. The experience that the students gain from entering in contact with the exposed market and from the workers in the company is very valuable and it's something that all students should have before getting a job.

We could reach the goals that we defined in the beginning of the project such as the development of an application that could be used by Talkdesk in their tests of the Callbar. We also gained soft-skills during this project, mainly, coordination, team work and critical thinking.

### 5.2 Technical Conclusions

During this project the most time consuming part was to integrate our application with Talkdesk components, since they work in a specific way and with certain parameters. Tweaks were necessary by Talkdesk’s Software Engineers to make it work and insert us in their staging environment to be able to use the Orchestrator piece.

In terms of technologies adopted, with relevant experience with GoLang, it would be my choice to build the backend of our Simulator since it performs very well comparing to other languages, mainly Python.

The results that we got with this web application met the criteria that was defined primarily the encapsulation of the calls on the Talkdesk’s side to remove the dependable factor from the providers, in this case Twilio. The application in future iterations is able to integrate the Presence System by removing the data structures that were added to store the devices states.

### 5.3 Future Work

It is noteworthy that this simulator is built to work with Talkdesk core components, so any use outside of this scope to test other applications won’t work. For the future of our simulator we defined some features that could be added:

- Possibility to hover on D3 elements to give more information about them;
- After the scenario has just run it would be presented a report with the metrics and a analysis of the results that were collected during the execution. This could identify possible problems and solutions. A further analysis by the user would be necessary on this document;
- When the Presence API is in a stable state where it can be used integrate with this application and remove the internal management of the status. This would also give to the application more parameters for customization since it would be possible to change the occupancy of each device;
• Add a custom profile where it would be possible to define the peaks of the function. This peaks would define the percentage in which an agent with this profile would accept a call. This is similar to the normal distribution but defined by the user.

With this in mind, there is always more space to improve as new features are added to the application, but this were the ones that were identified during the development of the project.
Bibliography


In this Appendix it’s the extended report that we presented and delivered in the final phase of the project.

Have you ever tested your application? Did you spend many hours fixing problems?

Figure A.1: Report - Page 1

Figure A.2: Report - Page 2
**Introduction**

- Problem definition
- Our Solution
- Value Proposal

**Project Roadmap**

- Architecture
- Device state

**Value Proposal**

- Internal Improvement
- Dependency on twilio
- Gathering data
- Predict Overwhelming Scenarios

---

**Figure A.9:** Report - Page 9

**Figure A.10:** Report - Page 10

**Figure A.11:** Report - Page 11

**Figure A.12:** Report - Page 12

**Figure A.13:** Report - Page 13

**Figure A.14:** Report - Page 14