Journal of Polish Safety and Reliability Association
Summer Safety and Reliability Seminars, Volume 9, Number 3, 2018

Walkowiak Tomasz
Wroclaw University of Science and Technology, Faculty of Electronics, Poland

Marcin Pol
Wroclaw University of Science and Technology, Faculty of Computer Science and Management, Poland

Dependability aspects of language technology infrastructure

Keywords

dependability, language technology infrastructure, natural language processing, micro-service architecture, CLARIN-PL

Abstract

The paper presents dependability analysis of CLARIN-PL Centre of Language Technology (CLT). It describes infrastructure, high availability aspects and micro-service architecture used in CLARIN-PL applications. Micro-services architecture improves dependability in respect to availability and reliability and to some extent safety. It is comprised of the mechanisms of reliable communication of applications, replication, recovery, and transaction processing. CLT has also a set of components for failure detection, monitoring and autonomic management, and distributed security policy enforcement.

1. Introduction

The concept of service dependability [1] was introduced to provide a uniform approach to analysing all aspects of providing a reliable service: hardware faults, software errors, human mistakes and even deliberate user misbehaviour. Dependability is defined as the capability of systems to deliver service that can justifiably be trusted [1].

Dependability is an integrative concept that encompasses: availability (readiness for correct service), reliability (continuity of correct service), safety (absence of catastrophic consequences), confidentiality (absence of unauthorized disclosure of information), integrity (absence of improper system state alterations), maintainability (ability to undergo repairs and modifications) [2].

Micro-service architecture[3] is a recent style of developing applications that consist of a set of "cohesive, independent processes interacting via messages" [13]. Therefore, each service is independently created and implemented. It allows to overcome problems that we can find in traditional "monolithic" applications like how to maintain large applications, how to fix the errors and remove the failures [6].

CLARIN[1] (Common Language Resources and Technology Infrastructure) is a pan-European research infrastructure intended for the humanities and social sciences. CLARIN-PL[2] Language Technology Centre (CLT) has been created as the Polish node of the CLARIN research infrastructure. It is aimed to support researchers and students in the fields of Humanities, Social Sciences and also Computer Science in work with natural language engineering and text mining. The platform brings researchers into a manageable, secure cloud environment. It is a tool that promotes open, centralized workflows by enabling capturing of different aspects and products of the research lifecycle, including developing a research idea, designing a study, storing and analysing collected data. In this paper we present the dependability aspects of CLT deployed as a set cooperating micro-services.

CLARIN-PL micro-services are designed in the way to have a result (for example: processing of one, small text file) in time less 6 seconds, even if the LTC is busy with processing huge corpora.

[1] https://www.clarin.eu/

[2] https://www.clarin-pl.eu/

2. CLT infrastructure

CLARIN-PL infrastructure consists of three layers (Fig. 1): web applications, repositories and core micro-services.

Web applications are aimed to communicate with users to perform given set of tasks in CLT. Web applications could be developed in two different styles as Single Page Applications that communicates with other *parts* of CLARIN-PL by REST services or as multitier applications, where the server side mediates communication with CLT components. For example: WebSty[3] and LEM[4] are SPAs whereas Inforex[5] and Mewex[6] are 3-tier applications developed in PHP.

The second layer consists of two repositories: D-Space and NextCloud. They allow to store corpora and results of processing. Additionally D-Space repository is an authorisation manager, which gives users possibilities to use all applications from the first layer with one login and password.

The third tier delivers core of micro-services. It includes single authorisation for the CLARIN-PL platform, access to pipelines of language and machine learning tools (NLPServices) and a monitoring module.

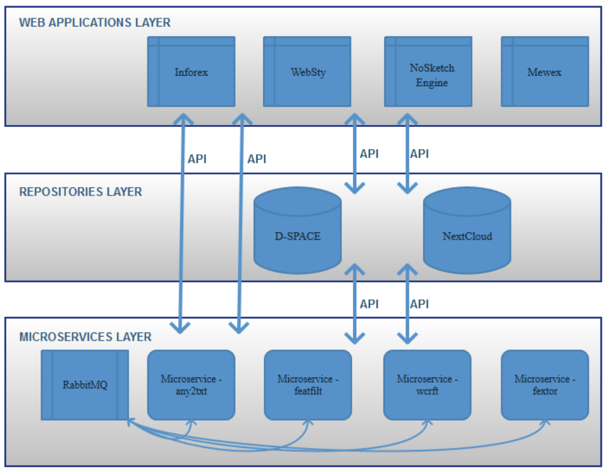

Figure 1. CLARIN-PL three layers infrastructure

3. Usage of infrastructure

CLARIN-PL Platform focuses on NLP research tools. It gives openness, unique identifiers and research data management. The high degree of flexibility means that it is possible to easily customise projects to fit a variety of needs, from small ones to large research collaborations. SS&H researchers can process large text corpora and easily publish or share results. CLT brings easy deposit and sharing functionality allowing processing and exporting results to web annotation application or search tools.

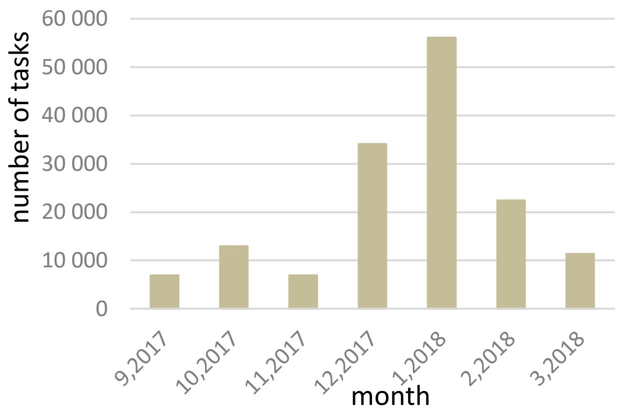

Figure 2. Number of tasks processed each month by CLT services (from September 2018 to March 2018)

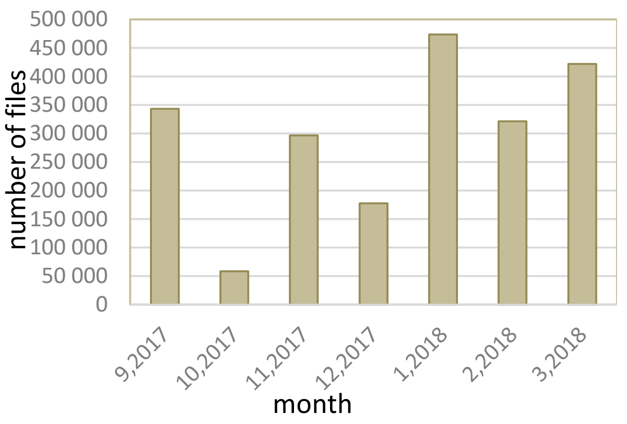

Figure 3. Number of files processed each month by CLT services (from September 2018 to March 2018)

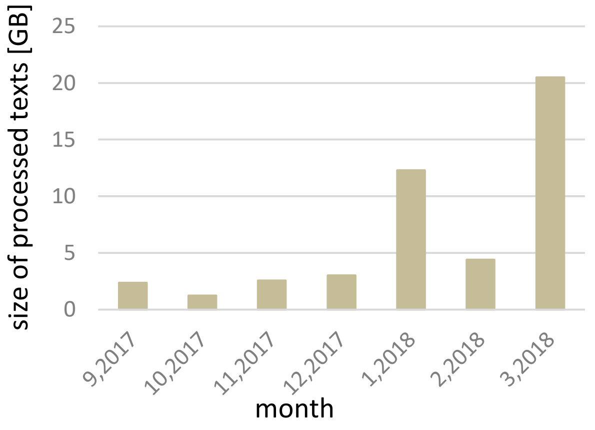

Figure 4. Size of texts processed each month by NLP services (from September 2018 to March 2018)

The main reason in building CLARIN-PL infrastructure was in offering SS&H researchers a more direct

[3] http://websty.clarin-pl.eu/
[4] http://lem.clarin-pl.eu/
[5] https://inforex.clarin-pl.eu/
[6] https://mewex.clarin-pl.eu/

way of using NLP tools in all phases of their projects. So far, the platform is focused mainly on Polish.

Selected statistics of usage of NLP mirco-services over the last 7 months are presented in Fig. 2-4. In average, each day ca.: 750 tasks, 10 000 files (texts) and 230 MB of texts are processed by users in CLT. The amount of processed texts each day is equal to amount of 190 pieces of Nobel Prize Henryk Sienkiewicz "Quo Vadis" novel. It shows that CLT is extensively used by researchers and therefore dependability aspects of the infrastructure is very important.

4. High availability

4.1 Hardware aspects

The CLT is deployed in a private cloud. Hardware consists of nine servers in a mixed rack/blade architecture. Each server has from 192 to 224 GB of RAM, which gives a total of almost 2 TB. Each server has two Intel (R) Xeon (R) CPUs E5-2665@2.40GHz, which let you run up to 16 threads per processor. In total, it gives power of 324 processes in parallel. Data storage subsystem is built on IBM Storwize V7000 with redundant dual-active intelligent FC 8Gb controllers and dual-active iSCSI controllers. Storage is using RAID10 volumes. Data is protected by backup with deduplication mode. All system is protected by UPS. [12]

4.2 Virtualisation

Servers are managed by XENServer[7] that allows to run and manage a large number of virtual machines. Virtualization provides a disaster recovery mechanism ensuring that when a virtualized system crashes, it will be restored as quickly as possible.
We use complete automation tool for managing Xen server pools which utilize the XAPI[8] management interface and toolstack. Our software suite provides complete High availability features within a given pool. The overall design is intended to be lightweight with no compromise of system stability. High availability is provided with built in logic for detecting and recovering failed services. We have two virtual machine servers, with automatic failover, provide safe environment to run services. Service is defined as the application and underlying operating system.

Features of CLARIN-PL scripts for XenServer:

- auto-start of any failed VMs,
- auto-start of any VMs on after reboot ,
- detection of failed hosts and automated recovery of any affected VMs,
- detection and clean up orphaned resources after a failed host is removed - Removal of any failed hosts from pool with takeover of services.

4.3 Failover

Modified by CLARIN-PL version of D-Space[9] repository is stored on XenServer virtual machine. Single point failure at the data storage subsystem does not affect running D-Space repository service instance at all. Single point failure of the primary application server will initiate reconnecting to redundant second controller to another application server and restarting of the D-Space repository service. The policy described above applies for the digital repository and the data and metadata as well. The digital repository software source code is publicly available and is stored in multiple places on multiple machines.

4.4 Backup policy

Data backup is implemented on DS3500 Storwize V7000[10], ProtecTIER 6710 IBM System with deduplication mode. System is configure to create complete data snapshot every Sunday.

The content of the digital repository is backed up to the ProtecTIER every week (for the last month) including daily incremental updates using standard backup tools and can be restored using automatic tools. All backups follow standardized ways of using MD5 checksums for determining the consistency and we use automatic monitoring tools at various levels. All backups follow standardized ways of using MD5 checksums for determining the consistency and we use automatic monitoring tools at various levels. Any corruption of datasets creates error logs; and backups are kept to restore data. Automated database backups happen every day whilst online, with a retention period currently set to 7 days. Additionally, our long-term archive partner provides multiple backups and redundancy.

[7] https://xenserver.org/
[8] https://xapi.com/overview/
[9] http://www.dspace.org/
[10] https://en.wikipedia.org/wiki/IBM_Storwize_family

5. Maintainability

Virtualization makes the CLT management more convenient and efficient. The resources (memory, CPU, disk) could be attached to any machine on demand and changed according to needs. Operating systems are independent from the hardware in the virtual environment so they can be easily moved to another server as a reaction to any failure or resource shortage.

The aim of the monitoring module is to control the CLT state and allow a fast reaction on faults or system overloads. Monitoring is performed in full form on different levels, starting from hardware, through virtual machines, language and machine learning tools up to user web applications.

The CLT is high availability cluster with a distributed setup. According to a best practice scenarios for large and complex environments we do monitor servers with Icinga 2[11]. It is complete solution to monitor system logs, application logs, log files, and syslog data, and alerting you when a log pattern is detected.

It is built to be fast. Thanks to its multithreaded design it is performance oriented. It can run thousands of checks each second without any sign of CPU strain. CLARIN-PL infrastructure has complete monitoring of application servers – including JBOSS[12], Websphere[13], Weblogic[14], ActiveMQ[15], and Tomcat[16]. We implemented effective application server monitoring with the following benefits:

- increased security,
- increased server, services, and application availability,
- increased awareness of network infrastructure problems,
- fast detection of network outages and protocol failures,
- fast detection of failed processes, services, cron jobs, and batch jobs,
- audit compliance and regulatory compliance for example Data Seal of Approval[17] or CoreTrustSeal[18].

The administrators receive an email in ten minutes after the error or warning occurs. In addition summary of all warnings and logs are send to administrators via an e-mail every Sunday.

6. Authorisation and security aspects

6.1. Security

CLARIN-PL takes a proactive approach to security of research data and user data. Regular penetration testing is carried out to ensure service is secure against attack. All previous penetration tests have failed to breach the service. Recommendations issuing from tests have been implemented.

6.2. Authorisation and federated Login

To get access to the CLARIN-PL repository or applications, users must set up a free account with the D-Space (they can login via federation identity using shibboleth[19]).

LTC authorization is accomplished through the private federated login. It is done by generating a random string of 129 characters token, using a cryptographic generator. Then the token is assigned to the authorization process and it is stored in the user's D-Space database. To allow Clarin applications to use the token, it is placed in the http cookie named "clarin-en-token". It is available throughout the wildcard clarin-pl domain.

When user is running a federated application such as Nextcloud, the presence of the previously mentioned cookie is checked. If the cookie is in the browser, verification is performed by calling D-Space microservice. It is checking if the token is associated with the logged in user. D-Space returns to the application the user name and then logs into the application that requests the login. In the case of an error in the verification or absence of the cookie, the D-space will be redirected user. The scheme is shown in *Figure 5.*

The centralized identity solution was created to help deal with user and data security. The users and the applications access to the data within login and password. Once logged to the system, users are logged to all applications.

[11] https://www.icinga.com/
[12] https://en.wikipedia.org/wiki/WildFly/
[13] https://www.ibm.com/cloud/websphere-application-platform
[14]http://www.oracle.com/technetwork/middleware/weblogic/overview/
[15] http://activemq.apache.org/
[16] http://tomcat.apache.org/
[17] https://www.datasealofapproval.org/
[18] https://www.coretrustseal.org/
[19] https://shibboleth.net/

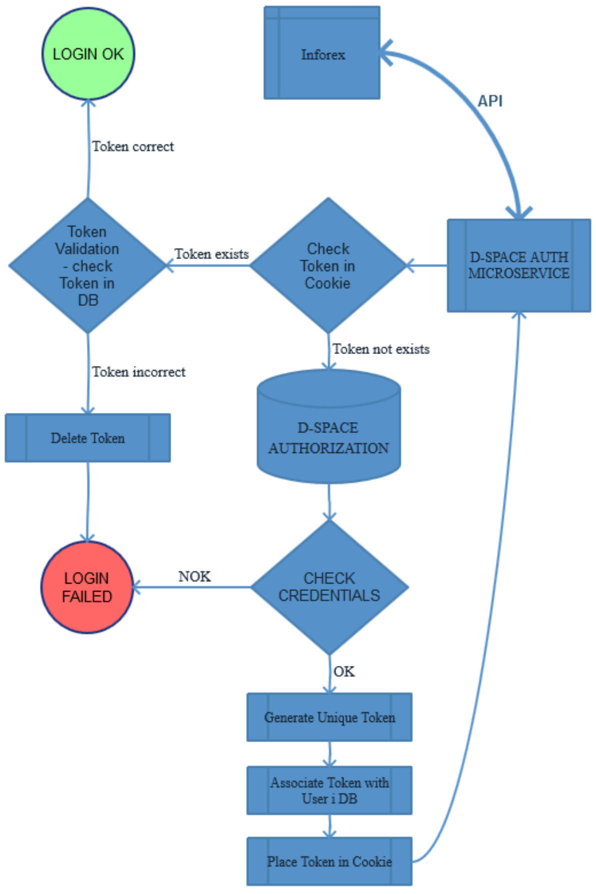

Figure 5. Authorisation schema

7. Response time and scalability

7.1 Response time statistics

Users can perform tasks of different complexity. As it could be seen on *Figures 6* and *7* most of tasks have a short processing time (less then 2s), however still there are tasks with much longer processing time. The largest processing time was more than 25 hours. The processing time is a function of corpora sizes (number of files and file sizes) and the task complexity. The size of the largest processed file was more than 2 GB. The largest corpora consisted of ca. 260 000 files. The most common are one file corpora, the median is equal to 53.

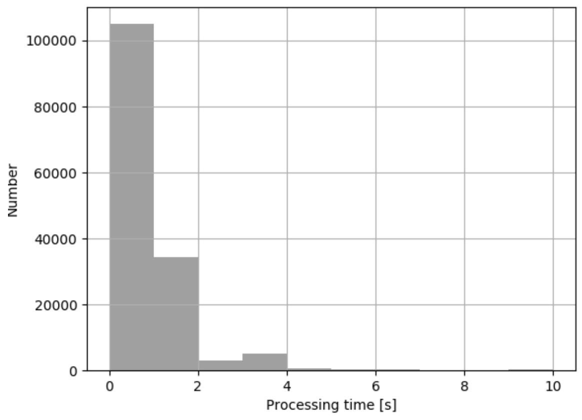

Figure 6. Processing time histograms for times smaller than 10s

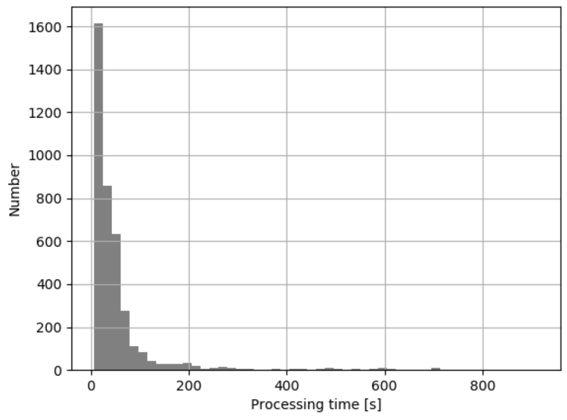

Figure 7. Processing time histograms for times smaller larger than 5s and smaller than 1000 s.

7.2. Synchronous and asynchronous

The most common approach of communication with web application by REST API (REST) [20] is synchronous. It works in standard blocking input/output way. Each incoming request is assigned to separate thread from the server's thread pool. The request thread is blocked until the response is not returned to the client [4]. In case of requests of small response time it is a very useful solution. However, when response time rises it can cause problems and errors. First of all, number of threads on a server side is limited so increasing the response time could result in approaching this limit. Secondly, the response longer then a client HTTP

[20]https://en.wikipedia.org/wiki/Representational_state_transfer

timeout results the timeout limit on the client side (usually equal to 189 s) and breaking the connection and therefore failing of receiving results [9].

Second solution, asynchronous one works in different way. Flow of the program does not block input/output (non-blocking IO). Request processing is delegated from the request thread to another thread not bonded with the thread pool in order to handle another requests in the meantime. It solves multi-threading problem with long-running calls, which might easily exhaust the number of available thread in the thread pool [4].

To fulfil different requirements the CLT web services has two interfaces: synchronous and asynchronous one [10].

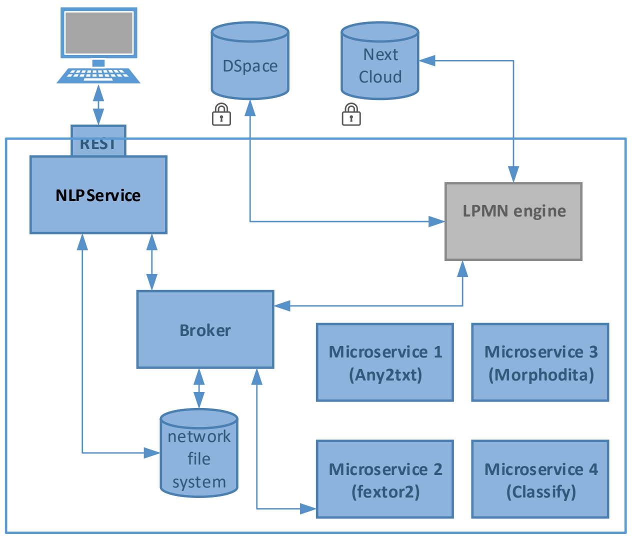

Figure 8. NLP services architecture

7.3 Micro-service architecture for scalability

The processing of texts by a chain of NLP and ML tools is done in micro-service [13] like architecture (Fig 8). Communication between micro-services is done by a queening system. We have used the AMQP [8] protocol for lightweight communication mechanisms and open source RabbitMQ [7] broker for a queuing system. Each NLP micro-service collects tasks from a given queue and sends back messages when results are available. Almost each (more frequently used) NLP micro-service is deployed on a separate virtual machine. Therefore, it is easy to scale up the system just by duplicating a virtual machine as a reaction to a high number of requests for a given micro-service. The workflows of NLP and ML tools are described in the Language Processing Modelling Notation (LPMN) and processed by a LPMN engine [11]. It allows to process texts in on corpora in parallel.

7.4. Service response time

The main aim of CLARIN-PL infrastructure is to fulfil user needs. It is not only the users functional requirements, but also to measure them in the sense of their usability. It has been proven [4] that if user will not receive answer for the service in less than 10 seconds he/she will probably resign from active interaction with the service and will be distracted by other ones. That is why it is so important to provide maximum service response time between below 6 seconds for small tasks.
As it was shown in 7.1, the LTC processes tasks of different sizes and computational complexity. Therefore, there is a need to prevent huge tasks from blocking small one. The LPMN engine has a built in scheduling algorithm that prevents large files and large corpora (corpora with large number of files) from blocking the NLP micro-service queue. The engine checks the queue size and if it exceeds a defined threshold (different for large files and for large corpora) the processing (sending tasks to the queue) is delayed for a given amount of time. Since checking the queue size is costs several ms of processing time (amount import for large corpora with very short texts), the engine checks the queue size not often then each 10 ms and has a simple prediction of the queue size algorithm.
As a result, a simple task (for example: processing of one, small text file) are processed by LTC in time less 6 s even if the LTC is busy with processing huge corpora. The experiments, shown the delay caused by the scheduling algorithm is less than 1% of overall processing time.

8. Conclusion

The paper presents a dependability analysis of CLARIN-PL Language Technology Centre (CLT) as the Polish node of the CLARIN research infrastructure. A dependable coordination of micro-services allows to fulfil users functional requirements. The CLT infrastructure integrates several dependability and security mechanisms in order to enforce reliability, integrity, confidentiality, and availability in a modular way.
The paper presented infrastructure that promotes open, centralized workflows by enabling capturing of different aspects and products of the research lifecycle, including developing a research idea, designing a study, storing and analysing of collected data.
It showed that CLT is extensively used by researchers and therefore dependability aspects of the infrastructure is very important. The amount of processed each day texts is equal to 190 pieces of Nobel Prize Henryk Sienkiewicz “Quo Vadis” novel.

References

[1] Avizienis, A., Laprie, J. & Randell, B. (2000). Fundamental concepts of dependability. *Proc. 3rd IEEE Information Survivability Workshop*, Boston, Massachusetts, 7–12

[2] Caban, D. & Walkowiak, T. (2012). Preserving continuity of services exposed to security incidents. *Proc. The Sixth International Conference on Emerging Security Information, Systems and Technologies*, SECURWARE 2012, IARIA, 72–78.

[3] Dragoni, N., Giallorenzo, S., Lluch-Lafuente, A., Mazzara, M., Montesi F., Mustafin, R. & Safina, L. (2016). Microservices: yesterday, today, and tomorrow, CoRR, vol abs/1606.04036

[4] Nielsen, J. (1994). *Usability Engineering*. Morgan Kaufmann, San Francisco.

[5] Pałczyński, M. & Walkowiak, T. (2015). Synchronous vs asynchronous processing in high throughput web applications. *The 15th International Conference Reliability and Statistics in Transportation and Communication, RelStat' 15*. 195–202.

[6] Richardson, C. (2017). *Micro-service architecture pattern*. microservices.io. Mar. 15, 2017.

[7] Videla, A. & Williams, J. (2012). *RabbitMQ in action. Distributed messaging for everyone*. Manning.

[8] Vinoski, S. (2006). Advanced message queuing protocol. *IEEE Internet Computing*, 10(6), 87–89.

[9] Walkowiak, T. (2014), Behavior of Web servers in stress tests. *Advances in Intelligent Systems and Computing* 286, 467–476.

[10] Walkowiak, T. (2016). Asynchronous System for Clustering and Classifications of Texts in Polish. *Advances in Intelligent Systems and Computing* 470, 529–538.

[11] Walkowiak, T. (2018). Language processing modelling notation - orchestration of NLP microservices. *Advances in Intelligent Systems and Computing* 582, 464–473.

[12] Walkowiak, T. & Pol, M. (2017). The impact of administrator working hours on the reliability of the Centre of Language Technology. *Journal of Polish Safety and Reliability Association* 8(1). 167–173.

[13] Wolff, E. (2016). *Microservices: Flexible Software Architectures*. Addison-Wesley