Setup and Implementation of an Automated Testing Pipeline for a DataOps Use Case

Bachelor's Thesis (T3201)

presented to the
Department of Computer Science

at the
Baden-Wuerttemberg
Cooperative State University
Stuttgart

by
Oliver Rudzinski

submitted on
September 7th, 2020

Project Period (CW)	25/2020 – 36/2020
Matriculation Number, Course	`5481330`, TINF17A
Training Company	Hewlett Packard Enterprise
Internship Company	DXC Technology
Project Supervisor	Dipl.-Ing. Bernd Gloss
University Supervisor	Jamshid Shokrollahi, Ph.D.

Erklärung

Declaration of Authorship

Ich versichere hiermit, dass ich meine Bachelorarbeit mit dem Thema:

I hereby declare that I am the sole author of this bachelor's thesis on the topic:

Setup and Implementation of an Automated Testing Pipeline for a DataOps Use Case

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

and that I have not used any sources other than those listed in the bibliography and identified as references.

Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

I further declare that the electronically submitted version of this thesis is identical to the printed version.

_____, _____ _____

Ort Place | Datum Date | Unterschrift Signature

Abstract

Contents

List of Acronyms

AI	Artificial Intelligence
BI	Business Intelligence
CI/CD	Continuous Integration & Deployment
DWH	Data Warehouse
ELT	Extract-Load-Transform
ETL	Extract-Transform-Load
DAMA	Data Management Association
MBA	Market Basket Analysis
ML	Machine Learning
MVP	Minimum Viable Product
SPC	Statistical Process Control
UI	User Interface
VCS	Version Control System

List of Figures

List of Tables

List of Source Code Excerpts

1 Introduction

1.1 Relation to Project Environment

1.2 Project Scope

1.3 Task Definition

1.4 Chapter Overview

2 State of the Art

The discipline of testing can be found throughout the entire scope of DataOps [1, p. 42]. However, it does not provide specific testing measures or frameworks. This is because data analytics solutions are tailor-made based on the specific needs of their areas of application. Moreover, available DataOps testing foundations remain a set of guidelines and abstract requirements that need to be applied and customized individually within each DataOps solution.

In this chapter, general DataOps and testing foundations are deduced in order to understand the need for DataOps testing and aid the design process of a use-case-specific testing framework which is to follow in Chapter 3.

2.1 Introduction to DataOps

In order to understand the requirement of testing within DataOps, it is important to understand the principles and processes of DataOps itself. In general, DataOps is an approach within building and conducting data analytics which combines established methodologies originating from DevOps, Statistical Process Control (SPC) as well as *agile* software development [2, p. 24]. Several components from each of these methodologies are taken and applied to building and conducting data analytics. These processes aim to eliminate analytics issues found in the traditional development process of such solutions. These issues include but are not limited to slow development and adaptation of analytics solutions [3], error-prone analytics results, repetitive manual processes [2, pp. 11 sqq.], etc. Testing is a common component that supports these DataOps processes.

During the rise of DataOps, other terms, including *MLOps*, *AI*Ops, etc., emerged. It is to mention that all data-related *Ops* underlie the general DataOps methodology and focus on specific subsets of data analytics applications.

2.1.1 SPC Heritage: Data Analytics Pipeline

Common data analytics solutions work by means of a pipeline: Data is acquired from various sources and flows through various steps of transformation, conversion, sanitization, and analysis before resulting in a valuable outcome, e.g., an analytics report. This can be compared to a manufacturing production line. For instance, raw materials from several input points are navigated through a number of steps, resulting in the output product. Issues that might occur during the production flow need to be recognized immediately. It does not suffice to notice issues at the end of a manufacturing process. This is why Statistical Process Control (SPC) is applied to the entire production line. It verifies that each step is conducted correctly and identifies deviations to expected, pre-defined values [4, p. 1]. Applicable tools can then perform recovery measures or stop the process entirely.

This methodology can be directly applied to data analytics pipelines [2, p. 27]. Each step should check if its input, processing, and output is valid and does not carry issues that might lead to unforeseeable problems during further analysis [5]. This could help solving the problem of incorrect analytics results since reverse-engineering the origin of the problem is often harder than performing individual checks and fallout measures [6]. These checks and measures are part of DataOps testing.

2.1.2 DevOps Heritage: CI/CD Pipeline Duality, VCS and Environment Management

DataOps' namesake, DevOps, originates in software development and aims to eliminate manual repetitive processes by automating them. It introduced Continuous Integration & Deployment (CI/CD) pipelines that take over processes taking place between solution development and deployment. This results in automatic building, testing, and deploying of software solutions [7, pp. 21 sqq.]. This does not only remove repetitive processes but also eliminates so-called *siloed organizations* (i.e., dedicated engineers, testers, operation teams, etc.) depending on each other during a development iteration [2, p. 56].

In DataOps, enabling CI/CD creates a pipeline duality. This duality is visualized in Figure 2.1.

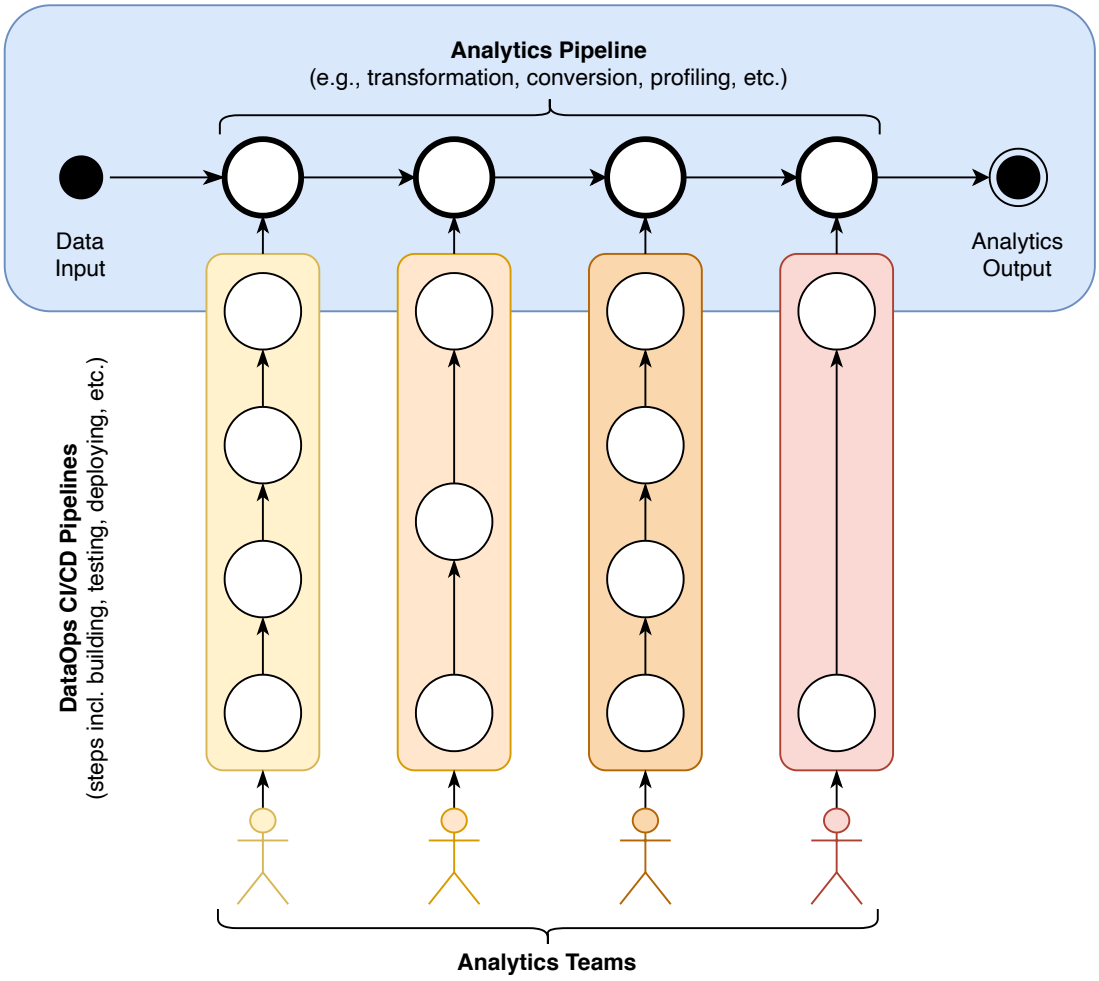

Figure 2.1: DataOps Pipeline Duality (per [2, pp. 38 sqq.])

The data analytics (or data operations) pipeline is also called *Value Pipeline* since it is in charge of answering questions through analytical insights [2, pp. 32 sq.]. Its horizontal representation visualizes its continuous flow. On the contrary, the vertical pipelines, also called *Innovation Pipelines* represent the DataOps CI/CD pipelines [1, p. 66]. Whenever a new feature is developed for any stage of the pipeline, a number of preliminary steps is performed before finally deploying the solution into production [2, p. 33]. As with data analytics in general, the design of such pipelines highly depends on the analytics and quality requirements. In the scenario of Figure 2.1, the individual stages require different CI/CD steps and might also be worked on by different teams within the superordinate project.

As with DevOps, DataOps CI/CD ties in with the project's Version Control System (VCS). This allows for collaboration between developers as well as development environment management [8]. Usually, a developer uses an individual sandbox to make changes to the common source code. This sandbox is a highly isolated en-

vironment which can be used without impacting the development process of other developers. It includes the current common source code as well as processes for installing and running all required dependencies [2, p. 41]. When committing a change to the VCS, it triggers the corresponding CI/CD pipeline which performs its checks and reports potential issues. Otherwise, the updated source code becomes the new common source code since the deployment into production has been conducted successfully.

2.1.3 Agile Heritage: Fast-Paced, Iterative Development

One problem within data analytics is that traditionally developed solutions cannot keep up with the demand of changing requirements. The development process is slow such that valuable and time-dependent information cannot be processed on time [3]. In software development, agile development mostly replaced traditional waterfall-orientated processes. *Agile* in DataOps context means that development and improvement is iterative and fast-paced. It requires a Minimum Viable Product (MVP) which is continuously improved by means of previously mentioned SPC and CI/CD processes [2, pp. 19 sq.].

2.2 Introduction to Testing

The previous section introduced where testing aspects can be found within DataOps. This section presents the current state of the art for software and data quality testing. Both disciplines will be evaluated and used within the testing framework design process.

2.2.1 Why Do We Need Testing?

In general, software development relies on testing principles to holistically and objectively validate the expected performance of a piece of software. Nowadays, organizations depend on data analytics more than ever [9]. Business Intelligence (BI) and Data Warehouse (DWH) solutions are designed to utilize data for business-required decision making [10]. While these systems are expected to generate value, many companies lose trust in their data analytics because it might be prone to unforeseeable errors [11]. This is because BI and DWH systems rely on high-quality data in order to provide representative analytics results and business insights [9]. Unfortunately, data quality issues of various sorts and manifestations lead to the systems generating false

and potentially misleading reports [9][12][13]. Testing the solution for its expected performance at various analytics steps might help solve the underlying issues or even exhaust them completely.

From a pure software perspective, it is desired that the solution does not break or crash during analytics performance for an unforeseeable reason. Applicable software tests could recognize such issues prior to production deployment, reducing or completely removing crucial bugs inside the software [14, pp. 105 sqq.].

As previously mentioned in Section 2.1.2, the pipeline duality ensures continuous flow of both production-grade data analytics as well as improvement and enhancement of the solution. Both pipelines require individual testing measures. This is referred to as "The Duality of [DataOps] Testing." [2, p. 40]

2.2.2 Value Pipeline: Data Quality Testing

Since the Value Pipeline is in charge of the business-critical real-time analysis of various data, the solution-in-use must guarantee the recognition and handling of data quality issues prior, during, and after each individual processing step by means of its SPC capability. In other words, while the underlying analytics software is static, the variance of data is arbitrarily large and needs to be handled properly. In order to define this kind of event handling, unified data quality dimensions are required.

The Data Management Association (DAMA) in the United Kingdom defined "The Six Primary Dimensions for Data Quality Assessment" in 2013 [15, pp. 7 sqq.]. They consist of:

Completenes describes the proportion of the stored data against the potential of being *complete* by means of a use-case-specific completeness definition [15, p. 8][16].

Uniqueness is achieved when each unique data record only exists once inside the entire database at hand [15, p. 9].

Timeliness is the degree to which data represents the reality from the required point in time [15, p. 10].

Validity describes a data item corresponding to its expected (and therefore, predefined) format, schema, syntax, etc. This definition should also include a range of expected or acceptable variation thresholds [15, p. 11]. Testing for data

schematics is one processes which allows for definitive objective differentiation between good and bad data [17].

Accuracy is the degree to which data *correctly* describes the actual object or event existing in the real world [15, p. 12].

Consistency describes the absence of difference when comparing multiple representations of the same real-life object against its actual definition [15, p. 13].

As mentioned inside the data quality dimension definitions, areas of data quality are mostly achieved when data complies with the corresponding data governance definitions. These allow for measuring the data quality [17]. These definitions build the foundation for a valid testing framework.

In practice, checks based on these dimensions have to be embodied inside the data analytics solution. Based on mentioned, use-case-specific requirements, a data flow can be checked by means of those dimensions, leading to recovery measures inside the system, or a system failure with appropriate error reporting.

2.2.3 Innovation Pipelines: Software Testing

Section 2.1.2 describes the Innovation Pipelines as DataOps-specific CI/CD pipelines. Apart from the build and deployment process, a major part of these pipelines is represented by software testing. Whenever a developer intends to update the codebase, a number of tests of various levels is conducted, visualized in the pyramid graphic in Figure 2.2 below.

Depending on the type of software, the types and levels of testing can vary. For data analytics solutions with User Interfaces (UIs) outside the scope, the three foundational levels suffice.

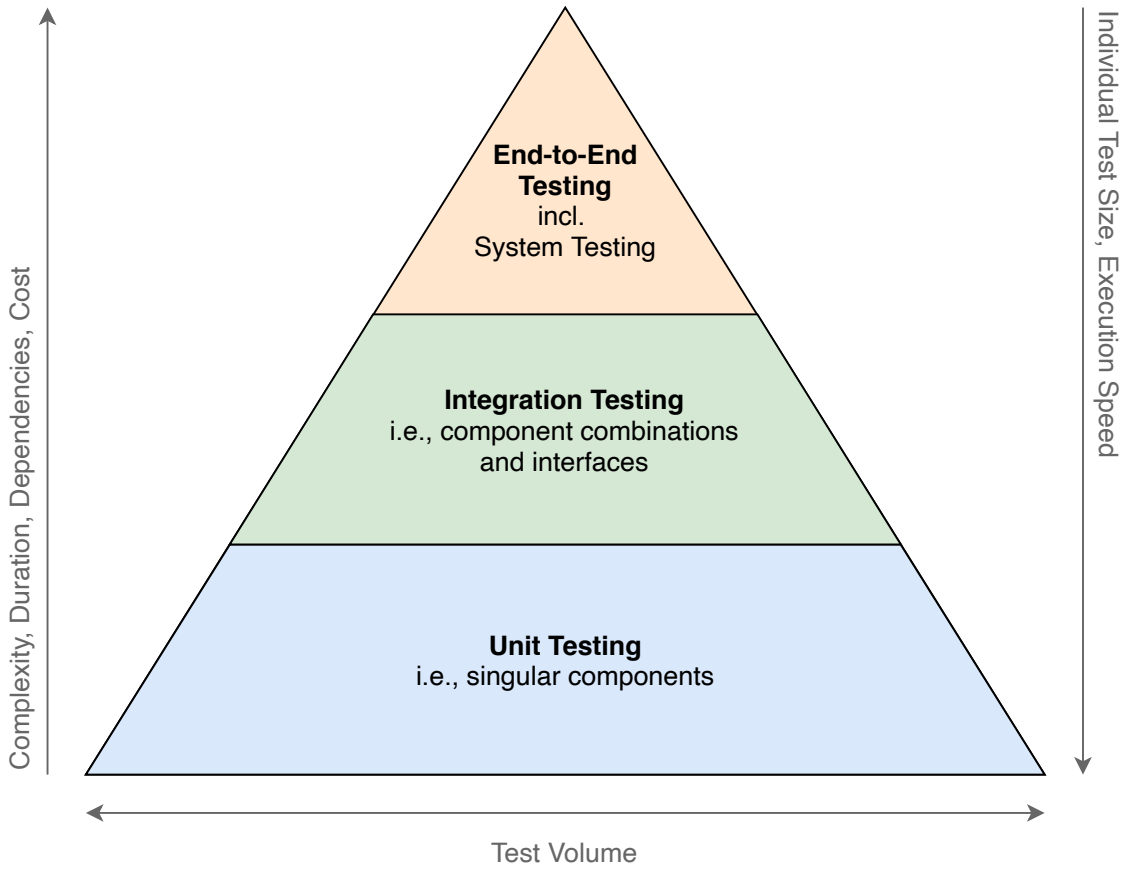

Figure 2.2: Software Testing Level Pyramid

Unit Tests take individual components (i.e., *units*) from the source code and check their behavior. Unit tests are pieces of code that invoke the chosen test unit and compare its actual result with the expected outcome [18, sec. 1]. They are characterized as granular, thus highly voluminous, repeatable, isolated, and idempotent [18, sec. 2].

Integration Test verify that multiple combined units are working together as expected. They focus on the testing of interfaces that connect singular components [19, p. 66]. Integration tests are performed in a less isolated environment, resulting in more outside dependencies [18, sec. 3].

End-to-End Tests describe the highest level of software testing [19, p. 67]. For the sake of simplicity, system tests are treated as a part of end-to-end tests within this work. End-to-end tests are performed under (close-to) deployment requirements. This creates an idealized real-life scenario. Thus, end-to-end testing is the least isolated level and serves as the final test stage before actual deployment. [19, p. 67].

Those three layers represent the *Fuctional Testing* type since they are built based on the functional requirements of a software solution [19, p. 69]. Its counterpart, *Non-Functional Testing*, covers aspects like load, performance, security, etc. [19, p. 70].

Apart from that, other testing methods can be applied to each testing layer. *Smoke Tests* are a subset within the entire test suite that cover core functionality required for the solution to *just* run. Such tests can help to recognize the complexity of a software issue [20, sec. 5]. Conducting *Regression Tests* is also important. They aim to uncover errors with pre-existing components when other components were newly included, changed, or removed [19, p. 70][21]. This is especially crucial with data analytics solutions since historic data still needs to be processable after updating the analytics engine [16]. To achieve regression testing in practice, unit tests can be written in a more abstract manner and re-run for the entire software solution, requiring older unit tests still to pass [21].

Putting everything into DataOps perspective, the Innovation Pipelines need to cover the change of the analytics solution. These CI/CD pipelines need to embody traditional software tests to verify the correct behavior of the application. This also includes testing the SPC capability of the given solution [22]. For that reason, pathological test data needs to be provided that is able to invoke a majority of events during the analytics process [2, p. 42]. In case of tests failing, the corresponding CI/CD Innovation Pipeline can report the issue back to the developer, allowing for further understanding and correction.

All in all, these tests are expected to invoke the majority of outcome possibilities, leading to a high test coverage and correct performance validation [5].

3 Testing Framework Design Process

4 Analytics Pipeline DataOps Enablement

4.1 Actual State Analysis: MBA Data Analytics Pipeline

4.2 DataOps Enablement Requirements

4.3 Architecture Design

4.4 Modus Operandi

4.5 Implementation

5 DataOps Testing

5.1 Testing Strategy

5.2 Testing Architecture Design

5.3 Implementation

6 Solution Evaluation

7 Conclusion

Bibliography

[1] J. G. Schmidt and K. Basu, *DataOps – The Authorative Edition*. Austin, TX: Panther Publishing, 2019, ISBN: 978-0-980-21694-3.

[2] C. Bergh, G. Beghiat, and E. Strod, *The DataOps Cookbook*, 2nd ed. Cambridge, MA: DataKitchen, 2019.

[3] J. Lockner, "What is DataOps?", *IBM Big Data Analytics Hub*, Dec. 2019. [Online]. Available: https://www.ibmbigdatahub.com/blog/what-dataops (visited on 07/15/2020).

[4] S. Knoth, "Statistical Process Control", *European University Viadrina Frankfurt (Oder)*, Jan. 2002. DOI: 10.1007/978-3-662-05021-7_11.

[5] "Add DataOps Tests for Error-Free Analytics", *DataKitchen*, Apr. 2020. [Online]. Available: https://medium.com/data-ops/add-dataops-tests-for-error-free-analytics-741ee48bd5cc (visited on 07/20/2020).

[6] T. C. Redman, "To Improve Data Quality, Start at the Source", *Harvard Business Review*, Feb. 2020. [Online]. Available: https://hbr.org/2020/02/to-improve-data-quality-start-at-the-source (visited on 07/15/2020).

[7] A. K. Kaiser, "Reinventing ITIL® in the Age of DevOps", in. Berkeley, CA: Apress, 2018, ch. Introduction to DevOps, pp. 1–35.

[8] A. L. Davis, "Version Control", in *Modern Programming Made Easy: Using Java, Scala, Groovy, and JavaScript*. Berkeley, CA: Apress, 2020, pp. 127–130, ISBN: 978-1-4842-5569-8. DOI: 10.1007/978-1-4842-5569-8_16.

[9] Munawar, N. Salim, and R. Ibrahim, "Towards Data Quality into the Data Warehouse Development", in *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, 2011, pp. 1199–1206.

[10] M. Souibgui, F. Atigui, S. Zammali, S. Cherfi, and S. B. Yahia, "Data quality in ETL process: A preliminary study", *Procedia Computer Science*, vol. 159, pp. 676–687, 2019, Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 23rd International Conference KES2019, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2019.09.223.

[11] BI-Survey.com, *Data Quality and Master Data Management: How to Improve Your Data Quality*. [Online]. Available: https://bi-survey.com/data-quality-master-data-management (visited on 07/15/2020).

[12] J. Freudiger, S. Rane, A. E. Brito, and E. Uzun, "Privacy Preserving Data Quality Assessment for High-Fidelity Data Sharing", in *Proceedings of the 2014 ACM Workshop on Information Sharing Collaborative Security*, ser. WISCS '14, New York, NY: Association for Computing Machinery, 2014, pp. 21–29. DOI: 10.1145/2663876.2663885. [Online]. Available: 11.

[13] T. C. Redman, "Assess Whether You Have a Data Quality Problem", *Harvard Business Review*, Jul. 2016. [Online]. Available: https://hbr.org/2016/07/assess-whether-you-have-a-data-quality-problem (visited on 07/15/2020).

[14] G. O'Regan, "Software Testing", in *Concise Guide to Software Engineering: From Fundamentals to Application Methods*. Cham: Springer, 2017, pp. 105–121, ISBN: 978-3-319-57750-0. DOI: 10.1007/978-3-319-57750-0_7.

[15] N. Askham, D. Cook, M. Doyle, H. Fereday, M. Gibson, U. Landbeck, R. Lee, C. Maynard, G. Palmer, and J. Schwarzenbach, "Defining Data Quality Dimensions", *DAMA UK*, Oct. 2013. [Online]. Available: https://www.whitepapers.em360tech.com/wp-content/files_mf/1407250286DAMAUKDQDimensionsWhitePaperR37.pdf (visited on 05/17/2020).

[16] S. Shen, "7 Steps to Ensure and Sustain Data Quality", *Towards Data Science*, Jul. 2019. [Online]. Available: https://towardsdatascience.com/7-steps-to-ensure-and-sustain-data-quality-3c0040591366 (visited on 07/15/2020).

[17] I. Schieferdecker, "(Open) Data Quality", in *2012 IEEE 36th Annual Computer Software and Applications Conference*, 2012, pp. 83–84.

[18] R. Osherove, "The Art of Unit Testing", in, 2nd ed. Shelter Island, NY: Manning Publications, 2013, ch. The basics of unit testing.

[19] A. S. Mahfuz, "Software Quality Assurance", in. Boca Raton, FL: CRC Press, 2016, ch. Testing, pp. 59–71, ISBN: 978-1-4987-3555-1. [Online]. Available: https://learning.oreilly.com/library/view/software-quality-assurance/9781498735551 (visited on 07/17/2020).

[20] A. Tarlinder, "Developer Testing: Building Quality into Software", in. Crawfordsville, IN: Addison-Wesley Professional, 2016, ch. The Testing Vocabulary. [Online]. Available: https://learning.oreilly.com/library/view/developer-testing-building/9780134291109 (visited on 07/17/2020).

[21] A. P. Mathur, "Foundations of Software Testing", in, 2nd ed. Dehli, Chennai: Pearson India, 2013, ch. Test Selection, Minimization, and Prioritization for Regression Testing, ISBN: 9789332517660. [Online]. Available: https://learning.oreilly.com/library/view/foundations-of-software/9788131794760/ (visited on 07/17/2020).

[22] "Add DataOps Tests to Deploy with Confidence", *DataKitchen*, Apr. 2020. [Online]. Available: https://medium.com/data-ops/add-dataops-tests-to-deploy-with-confidence-4efde90869a6 (visited on 07/20/2020).