Design and development of a highly configurable solution to manage and supervise batch executions

OUERGHI Yassine

To all ...

Contents

Abstract

TODO: Abstract/Summary

1 General Frame of the Project

Introduction

In this chapter, I will start by presenting the host company, along with a presentation of the problem I am trying to solve, afterward I will introduce my proposed solution.

1.1 Presentation of the host organization

Human Resources field is a vast fertile ground for innovation, this is where Advyteam chose to operate, by creating an HR Solution named "Accretio", suited for all kind of enterprises, it spans different areas of expertise, divided in several modules which the client can choose from. Modules such Core HR functionallities, Talent Management, Leave Management, and others.

Currently, Accretio is an install based application, however, the company is actively porting the application to a SAAS (Software as a service) based one, with support of multi-tenancy, automatic scaling, etc. Accretio uses the microservice architecture, which makes creating new services, and plugin in new services relatively easy.

Advyteam is a young company, established 2011 by three human resources tunisian specialists, its headquarters is situated in Paris, France.

Besides its Human resource management system accretio, Advyteam also provides certification programs in Human resources throught its Advyteam Academy, and offers training and coaching in almost all of the human resources feilds.

1.2 Project Background

With its effort to port the application to a SAAS software model, the company's engineers have faced several problems, one of these problems is how to give the ability to a user of the application to define, launch and monitor batch jobs.

A batch job is a computer program or set of programs processed in batch mode. This means that a sequence of commands to be executed by the operating system and submitted for execution as a single unit.[UNI18]

Currently, the application have several built-in batch jobs, such as calculating the employees payrolls, leaves, evaludations, etc. However, those batches are neither extendible nor configurable.

Say we have 2 clients, they both need to generate their employees payrolls, yet they both have different bussiness rules to calculate the salary, in this case we may end up writing 2 separate batches for each client.

Furthermore, each time a client comes up with a new use case for batch processing, or a new buissness rule to be excuted in a batch job, he have to revert to the Advyteam developers to write a new specific batch, which could be very tideous in a SAAS architecture with tens or hundreds of clients.

In the next section, I will present the different solutions that we came up to try to solve the above problem.

1.3 Proposed Solution

The most challenging aspect of this project, is how to provide to the client, a mean to insert his own logic into the application, with little or no help from the Advyteam team.

Before we dig into the different solutions, we need to have a very basic knowledge of how a batch work, we will dive deeper on how a batch work in future chapters.

A batch job is mainly composed of at least one step, each step itself is divided into three other steps;

- A reading step, from which we will read the data that will be processed, this "data" is the list of the registration numbers of the employees that would be proceesed.

- A processing step, where the data that has been read in the previsous step be will processed, this step will hold the client buissness logic.

- A writing step, where you can write the changes that you have made in a datasource, generate reports... etc.
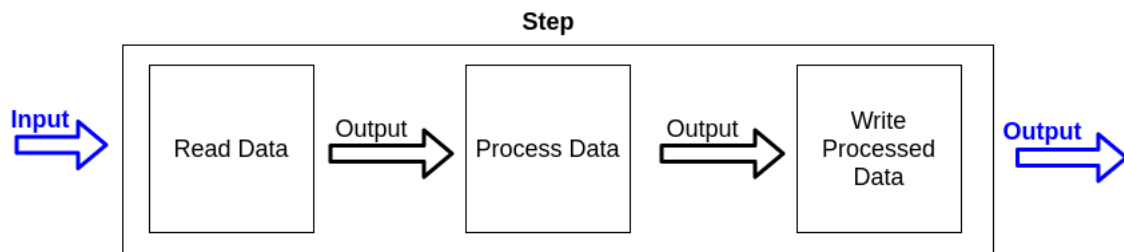

Figure 1.1: Simple Batch Job Execution Components

We need to find out a way where the client can enter his own logic for all of the three steps mentioned above, then somehow insert them and execute them into our application.

I will detail in the next subsection the solutions that we came up with and the choice we made;

1.3.1 RESTful APIs calls

The first solution we came up with, is to create a single batch job that consumes a RESTful API for each step, where we can call for the reading, processing and writing operations.

However this solution mandates that the client expose these RESTful APIs, and host it somewhere on his own servers, we cannot garentee that all of our clients would be able to host a RESTful APIs.

Another drawback is this solution would be very inefficient, we would bombard the client servers with our calls, and the processing would be done on their servers aswell.

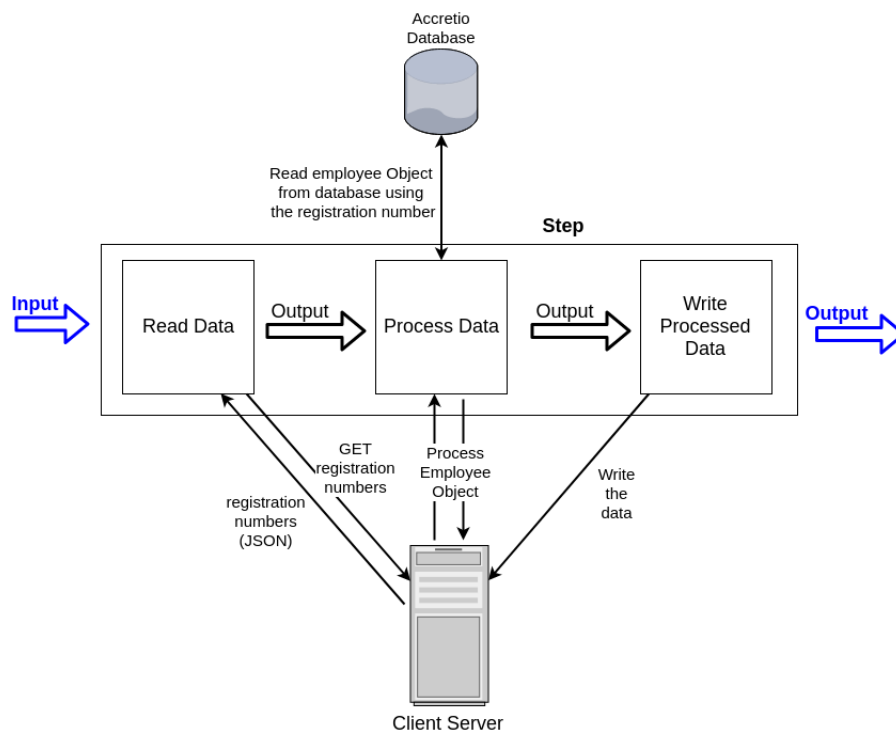This solution is quite simple, but presents lots of inefficiencies.

Figure 1.2: RESTful APIs Calls Solution Diagram

1.3.2 Rule Engine

We could also use a rule engine (such as drools), give the client the ability to define the rules that he would like to apply, and execute those rules when selecting the employees population (reading step), and also in the processing step.

As you can notice, this solution is very limitted to the number of possiblities that the client can chose from (finite list of rules), and would be able to define very simple and basic operations, simple filtering in the reading step, and basic math operations in the processing step.

The client would have no control over the reading and writing step, which is another great limitation, and no controls over the data structure to be used.

1.3.3 JAR Implementation

We decided to go Java's way, by defining a specification, and let the vendor (client in our case) implement this specification [Ora]. The solution we went for is to create two seperate jars:

- Specifications Jar: Also called "Template Jar", which will be handed to the client, where he has to implement it in order to write his own business logic. This jar will contain a set of interfaces and abstract classes, for each of the three steps mentioned above. The classes that the client wrote, would then be autowired in a runner jar.

- Runner jar: Will hold the batch steps definitions and configurations, and would execute the methods that has been wrote by the client in our environment, by autowiring/injecting the classes that have been implemented.

After implementing the Template Jar, the client only needs to upload the generated Jar, through the frontend application form, and specifies a set of options, such the datasource that would be used in the reading step, whether the job should be scheduled, number of retries ..., he can then manage those jobs, by starting them, stoping, monitoring, view generated reports, and so on.

We will go through the job configuration options in details in later chapters.

Figure 1.3 below, shows a visual representation of the solution, note that the squares marked in red, represents where the client's business logic will be injected, and executed.

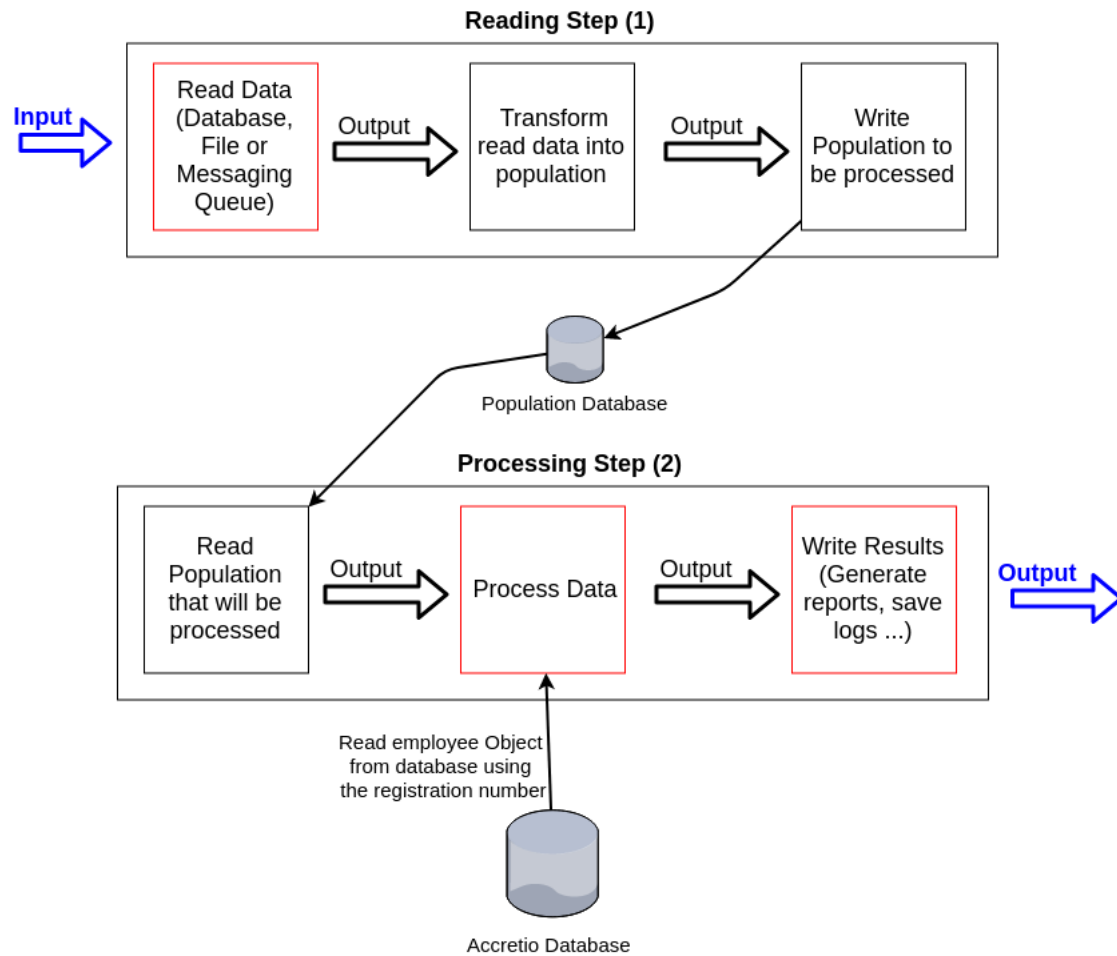**Figure 1.3:** JAR ImplementationSolution Diagram

Conclusion

In this chapter, I presented the frame on which this project is based on, by introducing the host company as well as the needs behind the project. Finally, I ended this chapter by presenting the different proposed solutions, and the one that I went with. The following chapter will be devoted to the analysis and the design of the application.

2 Analysis and Specification

Introduction

The first prerequisite you need to fulfill before beginning construction (development) is a clear statement of the problem that the system is supposed to solve.[McC04]

In this chapter, I will describe the adopted methodology along with the functional requirements and non-functional requirements. Lastly, I will present the system actors and use cases.

2.1 Adopted Development Methodology

A system development methodology refers to the framework that is used to structure, plan, and control the process of developing an information system. One system development methodology is not necessarily suitable for use by all projects. Each of the available methodologies is best suited to specific kinds of projects, based on various technical, organizational, project and team considerations. [fMMSCOoIS08]

For this project I have opted to use the Kanban method, the main reason for this choice is that this project require alot for researching, and many factors are not known beforehand, such which solution will be used, what messaging system will be used, and so on.

The tasks changes frequently, unplanned features or use cases would be added on the fly if a valid Proof of Concept was introduced. Kanban is suited for these kind of projects.

Kanban provides a method to continually adapt in order to smooth out kinks in the arrival of new development work. In this way, it allows the organization to avoid crises and respond more quickly and easily to issues that do arise.[ALR10]

In the last chapter I will be introducing the different Kanban artifacts, such as the backlog, however for the next section, I will be discussing the functional requirements.

2.2 Functional Requirements

Functional Requirements describe what a software is supposed to do and offer to its users, they are the first step toward a solution. We have a single actor in this particular application, which the administrator. The following, represents the requirements that the system must offer to its users:

- Manage Job Configurations:
  - Create a Job Configuration by uploading jar file (that contains the client business logic), and setting the running configurations.
  - Edit/Delete job configuration.
  - Start a Job Batch using a Job Configuration
  - Stop a running Job Batch
  - Restart a stopped Job Batch
  - Retry a failed/locked Job Batch
  - View a Job Configuration lauching history
  - Download a Job Configuration launch log
  - Download a Job Configuration launch report (if report generating is enabled when creating the job configuration)
- Manage Reports:
  - Create Job Configuration reports template
  - Create Specific reports template
  - Manage Report Template Parameters
  - Generate Report using a Template and parse parameters
- Dashboard summarizing jobs executions and states:
  - Number of job executions per date
  - Job executions status (completed, on progress, stopped, locked, failed)
  - Number of reports generate per report tamplate
  - Visualize all job executions logs, with the ablity to filter on the results

2.3 Non-Functional Requirements

Non-functional requirements detail constraints, targets or control mechanisms for a given system. They describe how, how well or to what standard a function should be provided. For example, levels of required service such as response times; security

and access requirements; technical constraints; required interfacing with users' and other systems.[Com10]

In addition to the features and functions that the system will provide, this application needs to satisfy various Non-Functional Requirements:

- Security:
  Unauthorized access to the system and its data is not allowed. Only authenticated users can perform an operation, such as creating a new job, lauchning a job, or viewing job execution.

- Scalability
  The application is part of a SAAS platform, thus scalability is a must. This module is built with a microservice architecture in mind, so scalability came predefined.

- Performance
  Batch jobs requires some important amounts of computational power, execution should be seamless, response time of the application must not bother the user in any way.

- Event-driven
  Job executions might take too long to finish, so executions should be event-driven, using the Asynchronous pattern to manage response time and continuous availability when long-running job executions are required to participate in the application process.

2.4 Requirements' Modeling

For a better understanding of the specification of the required needs of the application, I opted for the realization of some Use Cases Diagrams to have a better understanding of the needs.

In our solution, we only have one actor, which is the administrator that will launch and monitor all the job batches.

2.4.1 General Use Case Diagram

In this section, I will introduce the main functions of our microservice through a use case diagram. The use case diagram defines the expected activities from the administrator with regard to the application.

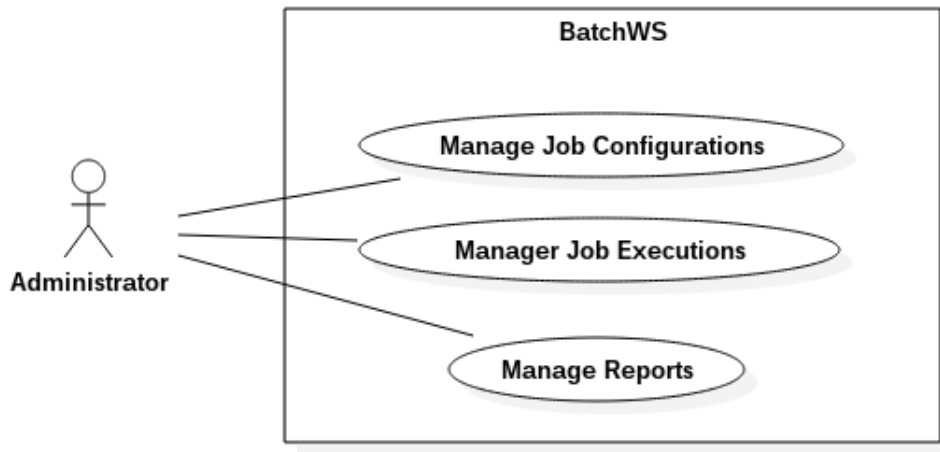*Figure-2.1* below illustrates the main use case diagram.

Figure 2.1: Main use case diagram

Description   In this pharagraph I will define each use case written in the diagram above, later in this chapter, I will go through each use case in a detailed diagrams. The administrator has to be logged in, in order to execute any action.

- Manage Job Configurations:
  Managing job configurations consist of creating configurations, by uploading the business jar, specifing different job batch configurations such as the data source, number of retries, etc.

- Manage Job Executions:
  The administrator can also manage the job executions, by launching, stopping, restarting jobs, check job histories, download job executions logs, etc.

- Manage Reports:
  After a job has been finished, the administrator can view the job execution in a detailed downloadable report, that he create its template. He can also check the reports that has been generated during the writing step, if he choose to during the creating of the job configuration. The administrator is the one reponsible of creating the report templates, and can even specify the template parameters for the detailed reports.

2.4.2 Detailed Use Case diagrams

As stated in the above paragraph, this section will detail the complex use cases, for a better understanding of the system.

2.4.2.1 Manage Job Configurations

Figure-2.1 and *Table-2.1* below represent respectively, the detailed diagram of the Job Configurations Management use case, and its description.
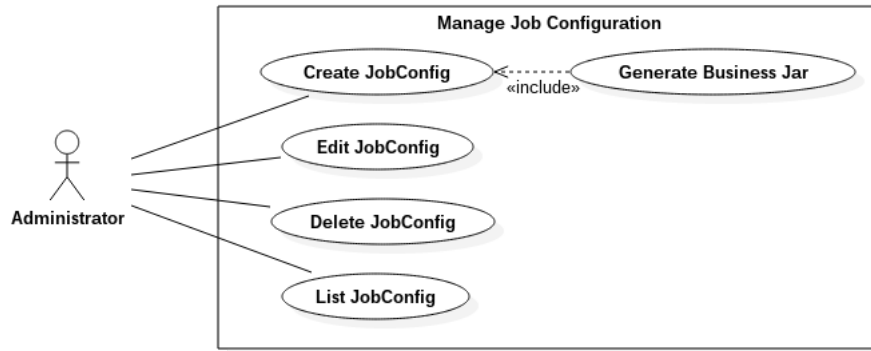

Figure 2.2: Manage Job Configuration Use Case Diagram

Table 2.1: Manage Job Configuration Use Case Description

Title	Manage Job Configuration Use Case.
Intention	Handle the Job Configuration management.
Actors	Administrator.
Pre-Condition	Successful authentication.
Sequences Definition	After the administrator is logged in, and after successfully implementing the template jar with his business logic, he can then upload this jar into the platform, when creating a job configuration, where he can also specify several other parameters that would be used when launching a job the uploaded jar. He can edit or delete a job configuration at any given time.

2.4.2.2 Manage Job Executions

Figure-2.3 and *Table-2.2* below represent respectively, the detailed diagram of the Job Executions Management use case, and its description.
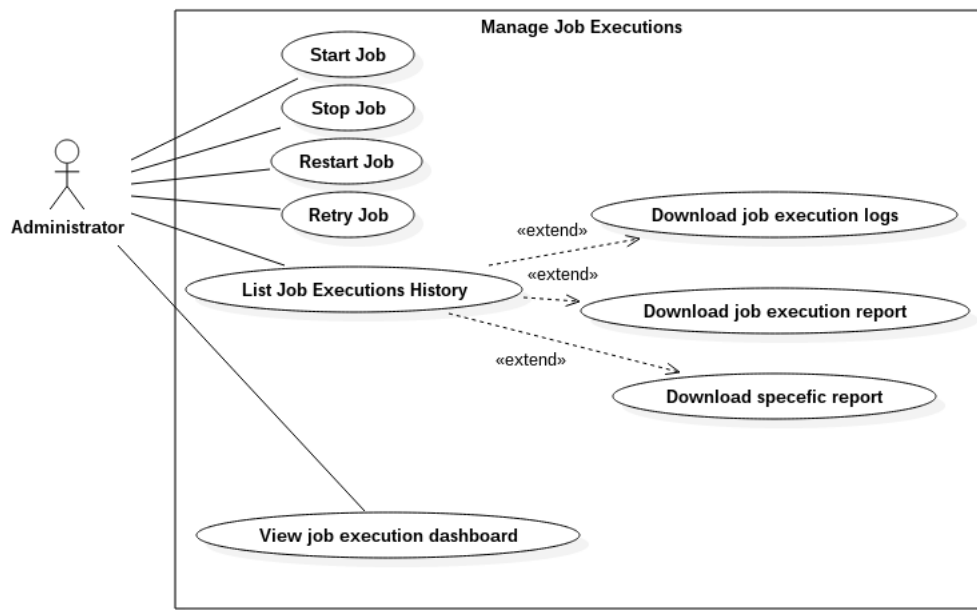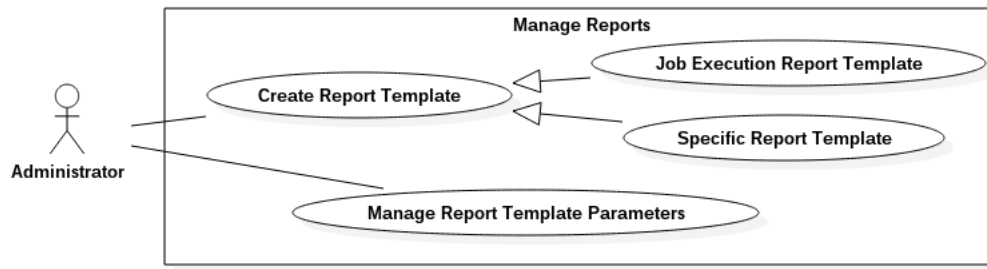
Figure 2.3: Manage Job Executions Use Case Diagram

Table 2.2: Manage Job Executions Use Case Description

Title	Manage Job Executions Use Case.
Intention	Handle the Job Execution management.
Actors	Administrator.
Pre-Condition	Successful authentication.
Sequences Definition	After creating a job configuration, the administrator can then use this newly created configuration to launch jobs. He can also monitor and manage these job executions at any given time. Managing consist of starting, stopping, restarting a job, a job can fail for any reason, in this case the administrator have the ability to retry this failed job. Results of a job execution can be aquired in several ways, from a dashboard containing some important stats to logs and report generation.

2.4.2.3 Manage Reports

Figure-2.4 and *Table-2.3* below represent respectively, the detailed diagram of the Manage Reports use case, and its description.

Figure 2.4: Manage Report Use Case Diagram

Table 2.3: Manage Report Use CaseDescription

Title	Manage Report Use Case.
Intention	Handle the Report management.
Actors	Administrator.
Pre-Condition	Successful authentication.
Sequences Definition	For a better understanding of a job execution, an administrator can generate a report, that he created its template. There are two types of reports, Job Execution reports, and specefic reports. For the latter, the administrator can specify the list of the parameters that would be injected, in contrast to the job execution report, which have a predefined paramters list.

Conclusion

In this chapter I went through the functional and non-functional requirements. Moreover, I made an analysis of these requirements by identifying drawing the global Use Case diagram and its description, along with the detailed diagram for each use case. In the next chapter, I'm going to build on top of this analysis, and specify the design of the application.

3 Design

Introduction

3.1 Architecture

3.2 Class Diagram

3.3 Sequence Diagrams

3.4 Entity relation diagram

Conclusion

4 Implementation

Introduction

4.1 Development Environment

4.2 Technologies

4.3 Solution

4.3.1 Spring Batch Steps

4.3.2 Asynchronous Communication

4.3.3 Configuration

4.3.3.1 System

4.3.3.2 Batch

4.3.4 Report templating & generation

4.3.5 Logs managment

4.4 Overview of the achieved work

4.5 Entity relation diagram

Conclusion

5 TODO

Introduction

5.1 Kanban charts

5.2 Perspective

Conclusion

Bibliography

[ALR10]   David J. Anderson and Janice Linden-Reed. Getting started with kanban for software development. https://dzone.com/refcardz/getting-started-kanban, August 2010.

[Com10]   Higher National Computing. D77f 35: Systems development: Structured design methods, September 2010.

[fMMSCOoIS08] Centers for Medicare & Medicaid Services (CMS) Office of Information Service. Selecting a development approach. https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf, March 2008.

[McC04]   Steve McConnell. *Code Complete 2nd Edition.* Microsoft Press; 2nd edition, 2004.

[Ora]   Oracle. Jsrs: Java specification requests. https://jcp.org/en/jsr/overview.

[UNI18]   INDIANA UNIVERSITY. What is a batch job? https://kb.iu.edu/d/afrx, February 2018.