# Using a Workflow Management Platform in Textual Data Management (Postprint)

**Authors:** Triet, Ho Anh Doan, Sven, Bingert, Ramin, Yahyapour, Triet, Ho Anh Doan

**Date:** 2022-11-28T00:00:00+00:00

## Abstract

This paper provides a brief introduction to the workflow management platform Flowable and its application in textual data management. Relatively new, Flowable's first release was on October 13, 2016. Despite its short time on the market, it appears to have quickly gained attention, currently boasting 4.6 thousand stars on GitHub. The focus of our project is to construct a large-scale text analysis platform that integrates numerous diverse text resources. Currently, we have successfully connected to four distinct text resources and acquired over one million works. Some resources are dynamic, meaning they may add new data or modify existing data. Therefore, it is necessary to maintain synchronization of both metadata and raw data on our side with the source resources. Additionally, to comply with FAIR principles, each work is assigned a persistent identifier (PID) and indexed for search purposes. In the final step, we conduct standard analyses on the data to enhance our search engine and generate a knowledge graph. End-users can leverage our platform to search through our data or access the knowledge graph. Furthermore, they can submit their analysis code to the system. The code will be executed on a High-Performance Cluster (HPC), and users can subsequently receive the results. In this context, Flowable can leverage PIDs for digital object identification and management to facilitate communication with the HPC system. As one may have already observed, the entire process can be expressed as a workflow. A workflow, encompassing error handling and notification mechanisms, has been created and deployed. Workflow execution can be triggered manually or at predefined time intervals. Based on our evaluation, the Flowable platform demonstrates considerable power and flexibility. Further utilization of the platform is already planned or implemented for numerous other projects.

## Full Text

## Preamble

**RESEARCH PAPER**

**Using a Workflow Management Platform in Textual Data Management**

Triet Ho Anh Doan†, Sven Bingert & Ramin Yahyapour
Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen, 37077 Göttingen, Germany

**Keywords:** Flowable; workflow; text analysis; knowledge graph; persistent identifier

**Citation:** Doan, T.H.A., Bingert, S., Yahyapour, R.: Using a workflow management platform in textual data management. 4(2), 398-408 (2022). doi: 10.1162/dint_a_{00139}

**Received:** August 16, 2021; **Revised:** November 28, 2021; **Accepted:** February 5, 2022

---

## Abstract

This paper provides a brief introduction to the workflow management platform Flowable and its application in textual data management. Flowable is relatively new, with its first release on October 13, 2016. Despite its short time on the market, it has quickly gained significant attention, currently boasting 4.6 thousand stars on GitHub. The focus of our project is to build a platform for large-scale text analysis by integrating numerous text resources. We have successfully connected to four different text resources and obtained more than one million works. Some resources are dynamic, meaning they may add more data or modify existing content. Therefore, it is necessary to keep both metadata and raw data synchronized with the source repositories. To comply with FAIR principles, each work is assigned a persistent identifier (PID) and indexed for search purposes. Finally, we perform standard analyses on the data to enhance our search engine and generate a knowledge graph. End-users can utilize our platform to search the data or access the knowledge graph. Furthermore, they can submit their analysis code to the system, which will be executed on a High-Performance Cluster (HPC) with results delivered later. In this context, Flowable leverages PIDs for digital object identification and management to facilitate communication with the HPC system. As one may notice, the entire process can be expressed as a workflow. A workflow incorporating error handling and notification has been created and deployed. Workflow execution can be triggered manually or at predefined time intervals. Our evaluation demonstrates that the Flowable platform is powerful and flexible, with further applications already planned or implemented for many of our projects.

**Corresponding author:** Triet Ho Anh Doan (Email: triet.doan@gwdg.de; ORCID: 0000-0002-7247-9108)

## 1. Introduction

Göttingen University Library, founded in 1734, is one of the five largest libraries in Germany [1]. It maintains a vast collection of text resources stored across different repositories, such as GDZ[1], Goescholar[2], and Textgrid[3].

Since these resources are scattered across multiple repositories, accessing them presents a significant challenge. Furthermore, access to raw data is still managed manually. If users require a raw text resource, they must first contact a librarian, who then contacts the responsible person at the corresponding repository to retrieve the requested data. This process makes it impossible to build a fully automatic text analysis workflow.

Therefore, we have developed a simpler way for users to access data. Our service enables users to perform a wide range of actions, including full-text search, resource download, and data analysis on a High-Performance Cluster. Behind the scenes, this service collects data from various repositories and makes it available to users. Since data are available in different formats at different locations and accessed via different methods, this data gathering process is more complicated than it sounds. Following high-cohesion and loose-coupling design principles, each action—such as data downloading, standardizing, and indexing—is treated as an independent module. Depending on specific use cases, appropriate workflows are developed by connecting these modules. The first step in workflow creation is selecting a format. We decided to use Business Process Model and Notation (BPMN), a highly standardized and well-supported workflow language.

The paper is structured as follows: Section 1 explains the current scenario and project motivation. Sections 2 and 4 provide insights into Flowable and the project architecture, respectively. Section 5 discusses how Flowable can support Canonical Workflow Framework for Research (CWFR) concepts. Finally, Section 6 presents the conclusion.

## 2. What is Flowable?

To avoid confusion among the many Flowable products, it is important to clarify that this paper discusses only Flowable open source[4], a workflow management

[1] https://gdz.sub.uni-goettingen.de/
[2] https://goedoc.uni-goettingen.de/
[3] https://textgrid.de/en/
[4] https://flowable.com/open-source/

platform and Java business process engine. Figure 1 [Figure 1: see original paper] provides an overview of the Flowable architecture, which encompasses four main components: Modeler, Task, Identity Management (IDM), and Admin. Each component has its own role in the system and can be accessed under different user rights configured via the IDM component.

All components expose various REST API endpoints for both public and cross-component communication. However, REST API is not the only interaction method, as Flowable also includes an intuitive web user interface (UI). Through the UI, users can fully exploit all Flowable features. Workflow developers can work with the Modeler component, while users who want to execute workflows can focus on the Task component. Workflows can be easily created via drag-and-drop actions in the web UI. Behind the scenes, all workflows are serialized into XML with BPMN schema. The IDM component handles identity management, either using a local database or connecting to a Keycloak[5] instance to enable Single Sign-On (SSO). Finally, system configuration is managed in the Admin component.

As depicted in Figure 1, five engines run in Flowable: BPMN, CMMN, DMN, Form, and Content engines. Consequently, Flowable supports not only BPMN but also Case Management Model and Notation (CMMN) and Decision Model and Notation (DMN). Figure 2 [Figure 2: see original paper] shows an example BPMN workflow developed using Flowable's Modeler component. The workflow contains two User tasks, one Script task, and one Decision task. Thanks to the Form and Content engines, workflow developers can create forms and link them to User tasks to capture user input. The Script task executes custom scripts in the chosen programming language, and the Decision task can make decisions based on a predefined DMN model.

## 3. Other Workflow Management Systems

Before selecting Flowable, we considered other workflow management systems, with the most prominent candidates being Apache Airflow, Prefect, Camunda, and Activiti.

Apache Airflow and Prefect are powerful workflow management systems written in Python that share many similarities. To create workflows, users must write Python code to define tasks and their connections, forming a Directed Acyclic Graph (DAG). Although these systems are well-developed, they do not suit our use cases. The requirement to create workflows through code is challenging because our workflow creators are domain knowledge experts rather than programmers. Additionally, using DAGs to represent workflows is inappropriate for complex workflows because loops are not permitted in DAGs. Furthermore,

[5]https://www.keycloak.org/

unlike BPMN, DAG is not a standardized workflow language. Finally, tasks requiring human interaction are not well-supported.

Flowable, Activiti, and Camunda share similarities as they originate from the same codebase. Initially, there was only Activiti; in 2013, Camunda split from the Activiti project [4]. Some years later, another fork from Activiti created Flowable. Instead of DAGs, these systems represent workflows using BPMN, and users can create workflows simply by dragging and dropping in their web browsers. Although all three systems meet our needs, we chose Flowable because it is more developed than Activiti [5] and has many more GitHub stars than Camunda.

---

## 4.1 System Architecture

As described in Section 1, storing data across different repositories makes it extremely difficult for users to access them. Therefore, we developed a system to facilitate this process. Figure 3 [Figure 3: see original paper] illustrates our system architecture, which contains three main components discussed in this paper: connector, knowledge extractor, and web server.

### 4.1.1 Connector

Connectors serve as bridges between the system and data sources. For each data source, we develop a corresponding connector responsible for all related actions, including authentication, metadata standardization, requesting Persistent Identifiers (PIDs), downloading raw data, indexing data into Elasticsearch, and keeping the system synchronized with data source changes. All metadata are stored as-is in Elasticsearch. Additionally, we define a core metadata set including title, author, abstract, and other fields, which are extracted from data sources and indexed into Elasticsearch with proper data types to enable searching.

According to the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) standard, each repository item should have a unique identifier [6]. However, not all items from all data sources have identifiers, and formats vary between sources. To ensure all items in our system can be uniquely and consistently identified, connectors retrieve items from data sources and assign PIDs generated and managed by the European Persistent Identifier Consortium (ePIC)[6].

### 4.1.2 Knowledge Extractor

Data obtained by connectors are stored in central data storage. We selected Elasticsearch, a feature-rich, powerful, and reliable platform with a strong focus

[6]https://www.pidconsortium.net/

on search capabilities. After connectors index data into Elasticsearch, the knowledge extractor can read the data and run further analyses to gain insights. One analysis is named-entity recognition, where entities are linked together to build a graph stored in the Neo4j graph database. Other current analyses include topic modeling, sentiment analysis, and word cloud generation, though this list will expand over time. Depending on data size, the knowledge extractor can run analyses locally or submit jobs to the HPC. Finally, analysis outputs enrich the graph database and metadata stored in Elasticsearch to improve search functionality.

#### 4.1.3 Web Server

The web server is the sole entry point for client interaction with the system. Users can access the system via a web UI or a Python package developed alongside the system. Supported features include searching for text resources, retrieving raw data, viewing entity graphs, and submitting analyses for HPC execution. As illustrated in Figure 3, the web server can also trigger connectors and the knowledge extractor. While system administrators can send commands directly to the web server via REST API, these commands typically originate from Flowable. When users want to run analyses on stored text, they send requests to the web server, which transforms and forwards them to the HPC. Users can check job status anytime via the web UI. When the HPC needs to retrieve text, it must send requests to the web server since the HPC resides outside the system and cannot access Elasticsearch directly.

---

### 4.2 Role of Flowable in the System

As described above, many system tasks can be represented as processes or workflows containing various linked actions. For example, importing a new data source requires steps such as downloading data, extracting metadata, and pushing data to Elasticsearch. While the web server can trigger these actions independently, Flowable handles linking them together. Therefore, Flowable serves as the system orchestrator.

It is important to note that Flowable is a multi-tenant application. Consequently, the instance shown in Figure 3 is not dedicated to our system but rather a Software-as-a-Service shared with other systems. We have developed several workflows in Flowable for our system, one of which is shown in Figure 4 [Figure 4: see original paper]. Although designed and deployed using Flowable version 6.5.0, it should work seamlessly with the latest version (6.7.1).

At the workflow' s start, the administrator selects which resource to import or update, then the pipeline begins with three sub-processes: file downloading, metadata extraction, and data indexing. For each sub-process, Flowable sends a request to the web server, which triggers the corresponding component. After sending the request, the Flowable workflow pauses and waits for information

from the web server. When the task completes, the web server notifies Flowable of the status (success or failure). As shown in Figure 4, the next sub-process begins only after the previous one succeeds. If the web server provides no information within 24 hours, the sub-process automatically fails. In all cases, the administrator receives an email regarding the workflow' s final status.

Since data sources change at different frequencies, automatic versions of this workflow execute at varying intervals depending on the specific resource.

---

## 5. Flowable and the Canonical Workflow Framework for Research

The CWFR concept addresses current limitations in processing pipelines. As described in [7], CWFR adds a layer on top of the workflow execution machinery and the technical workflow framework. Its basic elements include recurring patterns, a library of canonical steps, and libraries of domain-specific specialized packages per step. Integrating these elements requires "glue"—a connection layer. Here, the concept of FAIR Digital Objects (FAIR-DO or CWFR-DO) becomes relevant. If each step' s output is assigned a PID and enhanced with mandatory metadata, the next processing step can read this metadata and use the digital object accordingly. The connection between processing steps is thus the appropriate use of PIDs and required metadata (e.g., PID Kernel Information [8]).

The CWFR concept can be implemented using open-source Flowable software. As described in Section 2, Flowable serves as the technical runtime environment for BPMN-described workflows. In the CWFR picture, it covers at least the workflow execution machinery and framework. Through its interfaces, Flowable can include other arbitrary technical workflow frameworks or tools and functions implemented by researchers. Flowable enables automation of processing steps, supports iteration, and easily integrates user interactions via forms. Processing steps can be described as scripts (e.g., in Java) or as API calls to user-defined functions, representing either recurrent canonical steps or dedicated domain-specific steps.

### 5.1 CWFR-DOs in Flowable

Minting PIDs for CWFR-DO creation can be achieved either as part of user-defined functions or via a dedicated step in the Flowable BPMN workflow. For the latter, a sub-workflow can be designed to aggregate metadata (via forms or automatically) and conduct an HTTP API call to the PID system. This sub-workflow could then be called from any point in the data processing workflow. Metadata collected can be handled within Flowable to reduce user interactions at later processing stages. A scientific workflow could start by reading PIDs of initial digital objects through user interaction or file input. The initial PIDs and

newly created PIDs pointing to CWFR-DOs can be maintained in the active workflow' s memory and used at different workflow steps.

### 5.2 CWFR with Flowable in Practice

Flowable, as a technical foundation for CWFR, can be provided by a central IT service provider, relieving scientific staff of runtime environment maintenance responsibilities. The design and implementation of canonical steps, including CWFR-DO PID minting, should be conducted by IT experts from the service provider. Libraries of specialized packages can be built and maintained through collaboration between the service provider and scientific communities. Since workflow design in Flowable is straightforward and well-documented, researchers can also be assigned to create new workflows.

---

## 6. Conclusion

This paper presents a system that enables users to easily access text data from multiple repositories and run their own analyses on an HPC. Designed using a microservice approach, the system comprises various components and third-party services, including Flowable. Flowable serves as the system orchestrator, coordinating independent actions via workflows to achieve specific goals, such as importing new data sources. Finally, together with PIDs, Flowable can effectively support the CWFR concept.

---

## Author Contributions

**Triet Ho Anh Doan** (doanhoanhtriet@gmail.com): Conceptualization, Methodology, Software, Writing–Original draft.
**Sven Bingert** (sven.bingert@gwdg.de): Conceptualization, Project administration, Writing–Review & editing.
**Ramin Yahyapour** (ramin.yahyapour@gwdg.de): Supervision, Funding acquisition.

---

## References

[1] Goettingen State and University Library. Available at: https://www.uni-goettingen.de/en/about+the+goetting en+state+and+university+library/56613.html. Accessed 13 July 2021
[2] Architecture–Flowable open source documentation. Available at: https://flowable.com/open-source/docs/ cmmn/ch07-architecture/. Accessed 27 July 2021
[3] Flowable applications–Flowable open source documentation. Available at:

https://flowable.com/open- source/docs/bpmn/ch14-Applications/. Accessed 28 July 2021
[4] Meyer, D., Simmons, D., Weidner, T., C. O. Team: Camunda engine evolution since Activiti Fork. Available at: https://camunda.com/blog/2016/10/camunda-engine-since-activiti-fork/. Accessed 24 November 2021
[5] Top 10 advances Flowable made since Activiti. Available at: https://www.flowable.com/blog/2021/02/top-10-advances-flowable-made-since-activiti/. Accessed 24 November 2021
[6] Lagoze, C., et al.: Open archives initiative—protocol for metadata harvesting —v.2.0. Available at: https:// www.openarchives.org/OAI/openarchivesprotocol.html. Accessed 2 August 2021
[7] Hardisty, A., Wittenburg, P.: Canonical Workflow Framework for Research (CWFR)—position paper—version 2, December 2020. Working paper. Available at: https://osf.io/9e3vc/.
[8] Weigel, T., et al.: RDA recommendation on PID kernel information. Available at: https://www.rd-alliance.org/ group/pid-kernel-information-wg/outcomes/recommendation-pid-kernel-information. Accessed 2 August 2021

## Author Biography

**Triet Ho Anh Doan** obtained his Master's degree in 2018 in Applied Computer Science at the University of Göttingen, Germany. Since then, he has worked for the Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen as a researcher, software architect, and software developer. He has designed and developed many systems, such as a long-term preservation system for digitized library collections and a search engine for persistent identifiers. He is currently pursuing his Ph.D. at the same university with the topic "Big Data Infrastructure for Analysing Digitalised Library Collections." His research interests include research data management, persistent identifiers, system architecture, and scientific workflow development.

**Dr. Sven Bingert** completed his Diploma in 2003 in Physics at the Karlsruhe Institute for Technology, Germany, and obtained his Ph.D. degree in 2009 in Astrophysics from the University of Freiburg, Germany, as a member of the Kiepenheuer Institute for Solar Physics. After four years as a Postdoc in the Coronal-Dynamics Group at the Max Planck Institute for Solar System Research, Dr. Bingert joined the Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen, Germany. Since 2021, he has been the Deputy Head of the eScience working group. His main research topics are research data management, persistent identifiers, and development of scientific workflow services.

**Prof. Dr. Ramin Yahyapour** holds a doctoral degree in Electrical Engineering. His research focus is on efficient resource management and its application to service-oriented infrastructures, clouds, and data management. Since October 2011, he has been the Managing Director of Gesellschaft für wissenschaftliche

Datenverarbeitung mbH Göttingen, Germany. At the same time, he became a full professor for Practical Computer Science at the Georg-August-Universität Göttingen. He is active in several national and international research projects, such as scientific coordinator of the FP7 integrated project SLA@SOI and executive member of the CoreGRID Network of Excellence. He is an organizer and program committee member of several conferences and workshops, as well as a reviewer for multiple journals.

---

*Note: Figure translations are in progress. See original paper for figures.*