

URL-Archiver

Final Presentation

January 8, 2024

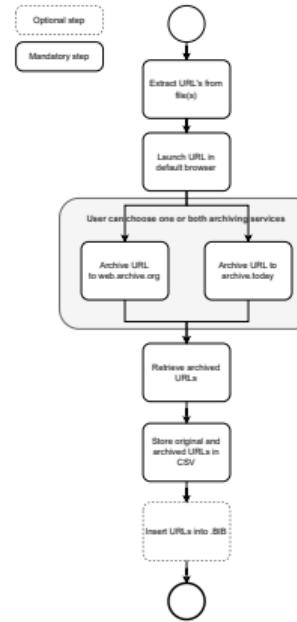
N. Dora, A. Vejseli, K. Wampfler | School of Engineering and Computer Science



Table of Content

Topics Covered

- Solution Overview
- Live-Demo
- Conclusion
- Questions?



About what was this project again?



Problem Statement Recap "URL-Archiver"

The ever changing Internet

- problem statement

- **ever-changing Internet** (websites are taken down or changed)
 - **Forensic** (e.g. legal report about a statement on the Internet)

- solution

- a java application which **searches hyperlinks**
 - and **archives** them with an archiving service



Project Goal

- Develop a software that...
 - ...searches hyperlinks within documents
 - ...is capable of **archiving websites**
 - ...can **store** current and archived links in a file
 - ...is **easy and intuitive** to use



Project Setup Review

Initial Situation

- What we knew:

- the end result has to be a **platform independent java application**
- we can only use **FLOSS-licensed libraries**
- the software must support **archive.ph** and **WayBackMachine**
- the code must be publicly accessible



Project Setup Review

Initial Situation

■ What we did first:

- project setup
- we **researched** both archiving services and what's possible
- we designed the rough **process for user interaction**
- we decided on a **software design pattern**



Project Setup Review

Installation / Project Setup

- creation of a new **github repository** for public accessibility
- setting up a **jira project** (with epics and first stories)
- installing the **latest java version**



Solution Overview

Command line application

Why a command line and not a GUI application?

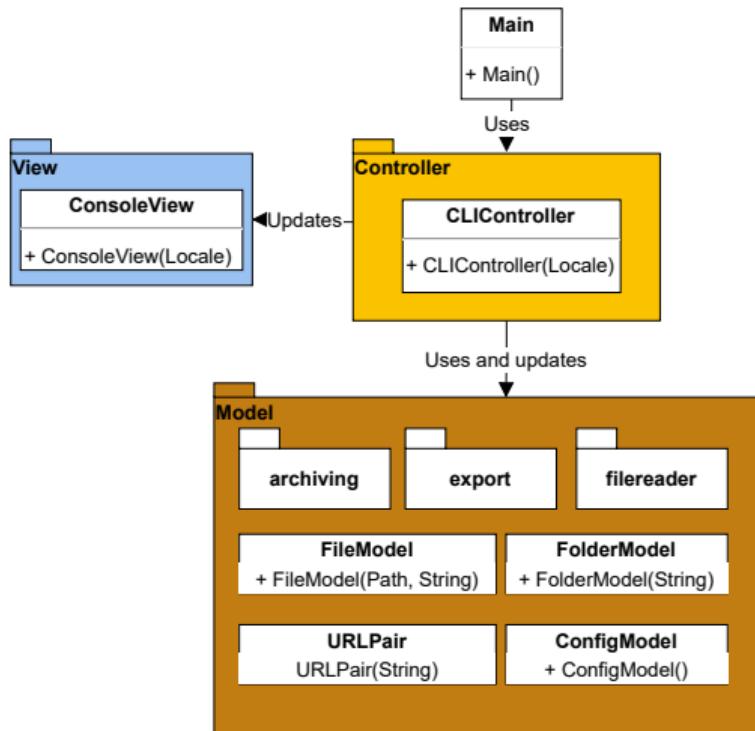
- **Focus on Logic:** Emphasis on functional depth over visual design.
- **Technical Audience:** Suited for users proficient in command-line environments.
- **Efficiency in Development:** Streamlines the development process by focusing less on UI design.
- **Flexibility and Scalability:** Provides a stable base for future feature expansion.



Architecture

Employing the **MVC Pattern**

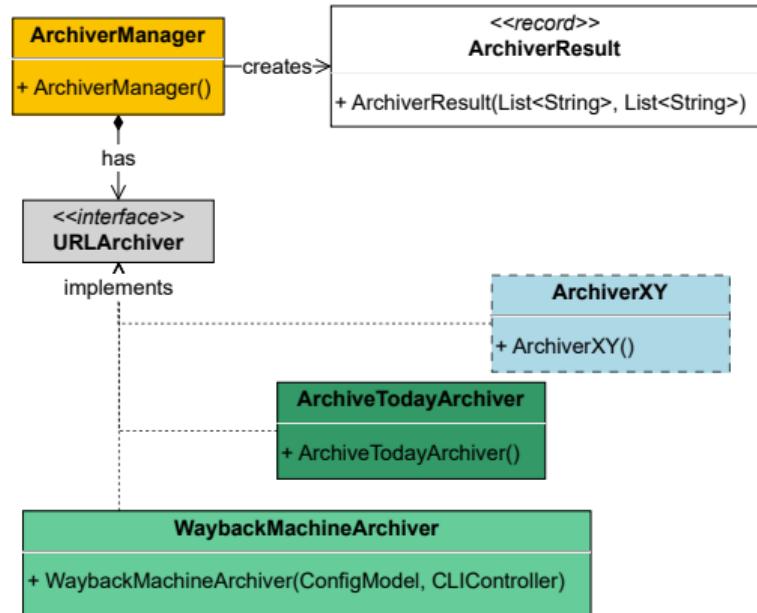
- Modular design for **easy extension**
- Separate data, view, and control flow
- Facilitates the potential **addition of a GUI**



Architecture

Factory Pattern

- **File Reader:** More input file types? No problem.
- **Archiving Services Integration:** An additional service can be added very easily.
- **Selenium Web Drivers:** Support another browser? Nothing easier than that!
- **Export functionality:** Easily add support for additional file types.



Enhancing Functionality and Flexibility

Configuration, Internationalization, and Testing

- **Configuration File Usage:** Preserves settings between program sessions.
 - **Stored Settings:** Includes Wayback Machine API keys and default browser.
- **i18n Implementation:** Lays groundwork for multilingual support.
- **Unit Testing:** Comprehensive tests written for all relevant classes.



Ease of Use: Installation Simplified

Custom Scripts for Cross-Platform Compatibility

- **Simplified Compilation and Installation:** Tailored scripts for Windows, Linux, and macOS.
- **User-Friendly Scripts:** Enable straightforward application compilation and execution.
- **Flexible Installation Options:** Users can choose to compile then run, or do both in one step.

```
1 #!/bin/bash
2
3 # Check for Java installation
4 if ! type java > /dev/null 2>&1; then
5   echo "Java is not installed or not in the PATH. Please install Java 21."
6   exit 1
7 fi
8
9 # Check for Maven installation
10 if ! type mvn > /dev/null 2>&1; then
11   echo "Maven is not installed or not in the PATH. Please install Maven."
12   exit 1
13 fi
14
15 # Build the project with Maven
16 mvn clean package
17
18 # Check for build success
19 if [ $? -eq 0 ]; then
20   echo "Build successful."
21
22 # Create 'bin' directory if it doesn't exist
23 mkdir -p bin
24
25 # Move the JAR file to the 'bin' directory
26 mv target/URL-Archiver-1.0-RELEASE-jar-with-dependencies.jar bin/
27 else
28   echo "Build failed."
29 fi
```

Live-Demo

SCRUM Review

Product Goal

- ✓ Accepts **directories, Unicode text (.BIB, .TEX, .HTML, etc.), or .PDF files** as input.
- ✓ Extracts **all URLs** from the files.
- ✓ Can **open URLs** in a web browser.
- ✓ Can **archive URLs** to archive.today and the Wayback Machine.
- ✓ Retrieves archived URLs.
- ✓ Outputs a **CSV file** with key-value pairs of original and archived URLs.
- ✓ Can add archived URLs to a **.BIB file**.
- (✓) The program code should be **minimal, modular, and self-explaining**.
- ✓ **User manual, Installation instructions and Software documentation** are provided.

SCRUM Review

Scrum Methodology



Nicolin Dora

Product Owner & Developer Scrummaster & Developer



Abidin Vejseli



Kilian Wampfler

Developer

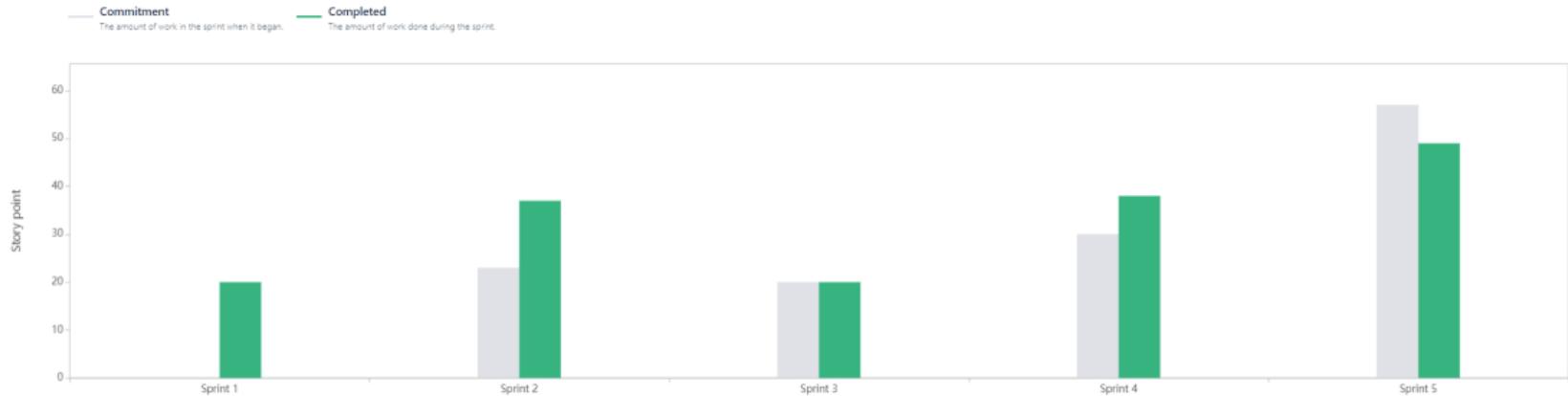
SCRUM Review

Product and Sprint Backlog Management

Projects / URL-Archiver / Reports

Velocity report

[How to read this report](#)



SCRUM Review

Sprint Goals & Approach

- **Sprint 1** Implement input handler for files (any unicode file e.g. .bib, .txt, .html and .pdf) and basic user guidance (Menu, Error messages).
- **Sprint 2** Implement a function to scan a provided text in order to identify and extract any URLs contained within it and upgrade our current console-based interface to enable users to easily open any extracted URL using their default web browser.
- **Sprint 3** Develop and implement a fully automated URL submission system that integrates with the Wayback Machine and Archive Today to ensure at least a 98% success rate in URL archiving.
- **Sprint 4** Enhance the system's stability and usability by resolving identified Selenium bugs across Linux, macOS, and Edge browsers, documenting the sprint process and licenses, conducting a thorough code review, and establishing a new configuration management file, aiming for zero critical bugs at sprint closure and readying the system for seamless URL archiving integration in subsequent sprints.
- **Sprint 5** Complete application refactoring for asynchronous archiving and .BIB file URL integration, ensuring no critical bugs and preparing for seamless future enhancements.
- **Sprint 6** Deliver a finalized application design, improved code quality, and complete documentation, with all components ready for review.

SCRUM Review

Delimitation and Delivery



SCRUM Review

Tools and Communication

Our project's success was also due to the effective use of tools like:



L^AT_EX JUnit 5

- Java, Maven, JUnit 5
- IntelliJ IDEA, GitHub, LaTeX
- Microsoft Teams and BigBlueButton



SCRUM Review

Conclusion



Lessons learned

Team

- Basic code structure
- Code Reviews more often
- Weeklys are important



Conclusion

Conclusion: Project Goal

- ✓ Develop a software that...
 - ✓ ...searches hyperlinks within documents
 - ✓ ...is capable of **archiving websites**
 - ✓ ...can **store** current and archived links in a file
 - ✓ ...is **easy** and **intuitive** to use



Future Work

What can be done to improve the application further

- Add more **archiving services** (such as Memento Time Travel)
- Support more **input file types** (e.g. .docx)
- Integrate a **graphical interface**
- Implement **multilingual support**



Questions?