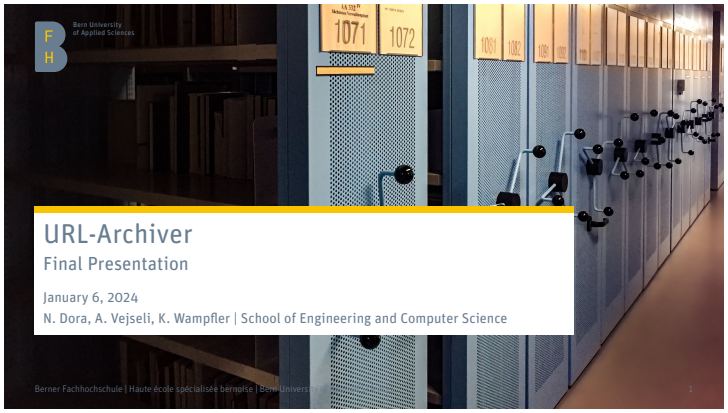


URL-Archiver

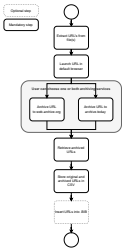
Final Presentation

January 6, 2024

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

Table of Content

Topics Covered



Template

Template with columns

Problem Statement Recap

Problem Statement Recap "URL-Archiver"

The ever changing internet

Kilian:
- Rückblick Project Setup: Erfassung der Ausgangssituation, Themen-Analyse, Stakeholder(-Management), Organisation, Installationen, etc.
- Eine kurze Repetition der Problemstellung und des Lösungswegs?

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

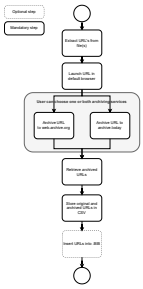
Initial Situation

- 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 whats 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?

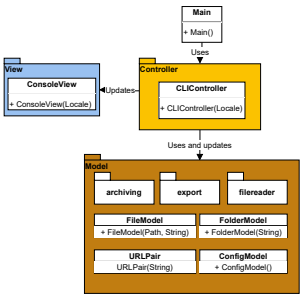
- **Focus on Logic:** Emphasis on functional depth over visual design.
- **Technical Audience:** Suited for users proficient in command-line environments.
- **Efficiency in Development:** Streamlines the development process by focusing less on UI design.
- **Flexibility and Scalability:** Provides a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

1. We have decided in favour of a command line program and against a GUI. For various reasons.
2. On the one hand, because we could focus on the logic and didn't have to build a responsive GUI.
3. On the other hand, our stakeholders are in the technical world and should therefore have few problems with a CLI application.
4. With a CLI application we create a stable basis to build on.
5. With the MVC pattern we can add a GUI in the future with little effort.

Architecture

Employing the **MVC** Pattern

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

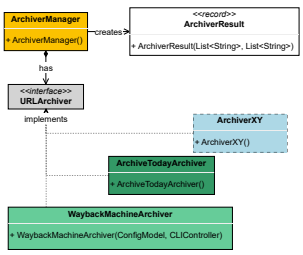


1. Speaking of the MVC pattern. As already mentioned, this pattern allows us to easily extend the application in the future.
2. As the view is separated from the model by the controller.
3. This means that these parts can be easily exchanged without having to make major changes to the others.

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.



1. We have also used the factory pattern where appropriate.
2. Specifically with the archiving services, the Selenium web drivers, when exporting to a CVS and Bibtex file and when reading the files.
3. As you can see on the right, using the archiving services as an example, this allows us a simple extension.

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.

1. We have also made various decisions to improve the functionality and flexibility of the application.
2. We save configurations in a configuration file.
3. We have laid the foundation for multilingualism...
4. ... and are testing all relevant classes with unit tests; there are currently 107 unit tests in total.

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.

```
#!/bin/bash

# Check for Java installation
if ! type java > /dev/null 2>&1; then
    echo "Java is not installed or not in the PATH. Please install Java 21."
    exit 1
fi

# Check for Maven installation
if ! type mvn > /dev/null 2>&1; then
    echo "Maven is not installed or not in the PATH. Please install Maven."
    exit 1
fi

# Build the project with Maven
mvn clean package

# Check for build success
if [ $? -eq 0 ]; then
    echo "Build successful."

    # Create 'bin' directory if it doesn't exist
    mkdir -p bin

    # Move the JAR file to the 'bin' directory
    mv target/UML-Archiver-1.0-RELEASE-jar-with-dependencies.jar bin/
else
    echo "Build failed."
fi
```


1. To make it as easy as possible for users to get started with the application, we have created compilation and run scripts.
2. Of course bash scripts for Linux and macOS and PowerShell scripts for Windows.

Live-Demo

SCRUM

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

Lessons learned

Team

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

Lessons learned (personal)

Kilian Wampfler

- good refresher for java development
- the team worked well together
- my junit tests could be improved (consistency & quality)
- my git commit messages according to convention

Lessons learned (personal)

Nicolin Dora

Lessons learned (personal)

Abidin Vejseli

Questions?

Problem Statement Recap "URL-Archiver"

The ever changing internet

- **ever-changing internet** (websites are taken down or changed)
- not a problem for everyday internet users but…
- …what about **documents** (e. g. theses, documentations etc.)
  - invalid links to content-relevant information
  - invalid quote links
  - old documents may contain numerous broken hyperlinks
- another case: **Forensic**
  - e.g. **legal report** about a statement on the Internet

Problem Statement

Simple Solution

- Archive the **actual state** of the linked websites
  - Wayback machine
  - archive.today
  - Memento Time Travel
  - and more…
- But who has the **time** and **motivation**?
  - …to search each hyperlink
  - …to manually archive each website

Project Goal

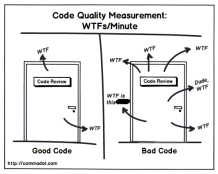
- **Develop a software that…**
  - …**searches hyperlinks** within documents
  - …is capable of **archiving websites**
  - …can **store** current and archived links in a file
  - …is **easy** and **intuitive** to use

Functional Requirements

- **The software must be…**
  - …a **Java application** that is compatible with multiple platforms
  - …Free and Open Source Software licensed **(FLOSS)**
  - …capable of archiving websites on **archive.ph** or/and **WayBackMachine**
  - …capable of generating a **CSV-file** with key-value (URL, archived URL)

Non-Functional requirement

- Coding Norm
- publicly accessible
- Project-Method: **SCRUM**
- public documents in English
- project report with **LaTeX**

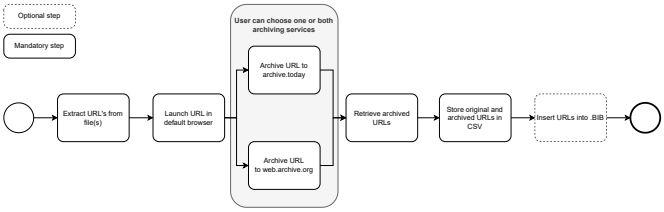


Solving The Problem

Process Model

Workflow for URL Extraction and Archiving

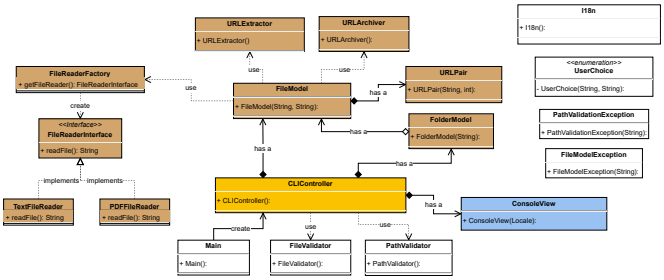
- User can **skip URLs**, **launch them**, **access help** or **quit** the program at **any time**



1. Thank you for coming from my side too.
2. First lets talk about how an user can archive urls.
3. ——————————————————.
4. For this you can see here a rough overview of how a user navigates through the application to archive URLs.
5. The user can open the current URL in their default browser, access the help section, or view the archived URLs at any time.
6. The user does not need to archive a url, he can skip the archiving process.
7. Now to the more technical part.

High Level Class Diagram

Overview

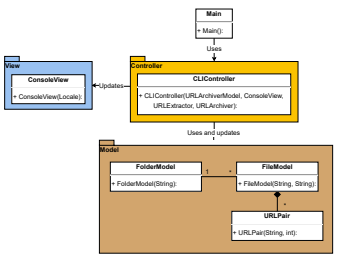


1. To demonstrate how we meet the requirements for our application, which Kilian presented earlier, I will now provide a high-level overview of our structure and its individual components.
2. It is important to note that this is our current structure, subject to expansion and adaptation as the project advances.
3. The Main class serves as the application's entry point.
4. CLIController interprets user commands, validates paths and types, and selects the data model.
5. FileModel manages individual file data, and stores URL associations.
6. FolderModel deals with file groupings within folders, integrating multiple FileModel instances.
7. URLPair links original to the archived URL.
8. I18n enables the application to support multiple languages, enhancing global usability.
9. UserChoice defines possible user actions with corresponding i18n keys, aiding in international user interaction.
10. ConsoleView acts as the communication hub, utilizing UserChoice for navigation and i18n for language support.
11. Now I would like to discuss some aspects of the architecture in greater detail.

Architecture

Employing the **MVC** Pattern

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

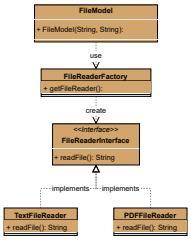


1. As you may have observed, we utilize the MVC pattern, among other techniques, to ensure the scalability of the application.
2. This involves separating the business logic from the presentation using a controller.
3. This enables us to easly switch out the Command Line Interface with a Graphical User Interface.
4. But that's not all!

Architecture

Factory Pattern

- **Scalable** and **robust** design
- Ensures **maintainability** and clear code structure
- Commitment to **best programming practices**

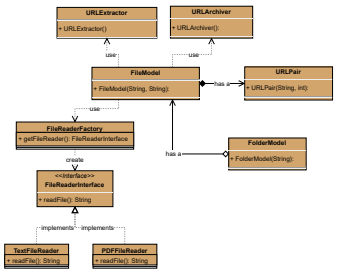


1. To enhance our application's flexibility further, we've implemented a factory class that reads and processes different file formats.
2. With a single call to the getFileReader() method from the FileReaderFactory, specifying the file type, we retrieve an appropriate FileReader.
3. Currently, our application can handle all UNICODE text files and PDFs.

Key Components of the Data Model

Overview

- **FileModel**: Represents and handles individual files
- **FolderModel**: Manages a collection of FileModel instances
- **URLPair**: Links extracted URLs with their archived counterparts
- **FileReaderFactory**: Organizes and maintains URLPairs
- **URLExtractor**: Used for URL extraction
- **URLArchiver**: Used for URL archiving



Now, let us discuss our data model. It is consists of the following classes:

1. The FileModel class manages the details of individual files. Each FileModel potentially containing multiple URLPair objects.
2. The FolderModel is applied when a user provides a folder path, storing the folder's structure and containing a list of FileModel objects.
3. The FileReaderFactory is our system for providing the right reader for a file, based on its type. This ensures that each file is read correctly and efficiently.
4. With the URLExtractor, our model extracts URLs from text, and the URLArchiver takes care of the URL archiving process.
5. This setup not only facilitates a clear division of tasks among the different components but also streamlines the process of archiving and retrieving URLs.
6. And now to the technologies we use for development.

Technologies I/II

- **Java**: The primary programming language for our application
- **LaTeX**: Used for documentation and presentation
- **JUnit 5**: Utilized for unit testing
- **PDFTextStripper (PDFBox library)**: Used for extracting text from PDF documents
- **Selenium**: Web automation tool used for tasks like inputting URLs, handling **captchas**, and retrieving archived URLs due to the absence of an official API from archive.today
- **Maven**: Essential for compilation, dependency management, and building the project

1. The application is developed in Java, while the report, presentations, and user manual are authored in LaTeX.
2. For testing, we utilize JUnit 5, and for text extraction from PDFs, we rely on PDFTextStripper.
3. The archiving process on Archive.today is executed using Selenium to navigate the lack of an API and to address captcha challenges.
4. Additionally, we use Maven for building the application and managing dependencies.

Technologies II/II

- **Archive.today**: One of the services used by the URL-Archiver to archive URLs
- **Wayback Machine**: One of the services used by the URL-Archiver to archive URLs
- **JetBrains IntelliJ IDEA**: Used to develop the program and create the report and presentation in Latex
- **GitHub**: A platform for code hosting and collaboration, allowing us to manage our program's development and version control
- **Atlassian Jira**: A tool for tracking progress and managing our project goals and tasks

1. Furthermore, we rely on Archive.today and additionally on the Wayback Machine to archive URLs.
2. As our development environment, we use JetBrains' IntelliJ IDEA.
3. We use Github to store our code and for version control.
4. To manage and plan our project, we use Jira.

Project Management with SCRUM

Our Scrum Team

Nicolin Dora
Product Owner & Developer

Abidin Vejseli
Scrummaster & Developer

Kilian Wampfler
Developer

Other Roles

Dr. Simon Kramer
Stakeholder & Customer

Frank Helbling
PM-Advisor

Our Scrum Adaptions

Sprint

Sprint	Sprint planning	Daily Scrum	Sprint Review	Sprint Retro
Two Weeks Sprint Goal (SMART)				

Our Scrum Adaptions

Sprint Planning

Sprint	Sprint planning	Daily Scrum	Sprint Review	Sprint Retro
	First Monday Estimate User Stories Define Goal Choose User Stories			

Our Scrum Adaptions

Daily Scrum

Sprint	Sprint planning	Daily Scrum	Sprint Review	Sprint Retro
		~~Daily~~ Twice a week Current state MS Teams		

Our Scrum Adaptions

Sprint Review

Sprint	Sprint planning	Daily Scrum	Sprint Review	Sprint Retro
			Last Day Completed? Problems? Testing Poductincrement	

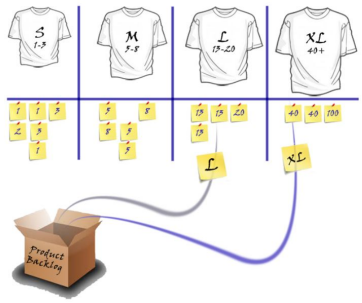
Our Scrum Adaptions

Sprint Retro

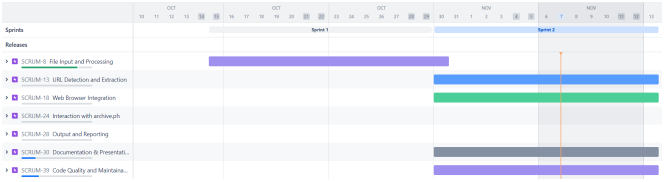
Sprint	Sprint planning	Daily Scrum	Sprint Review	Sprint Retro
				Last Day Feedback Do's & Dont's

Estimation Method

T-Shirt Sizes

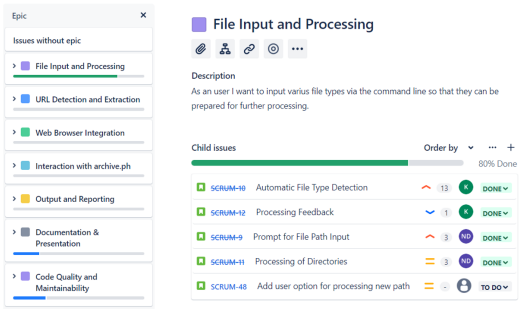


Velocity



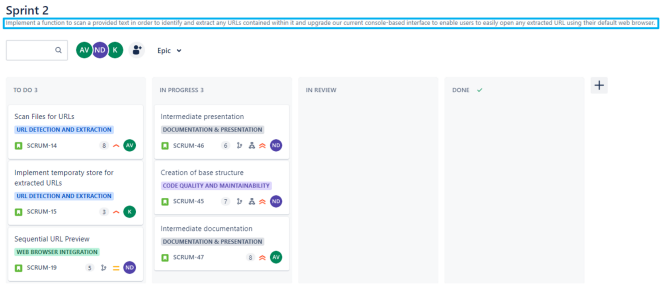
Backlog

Epics



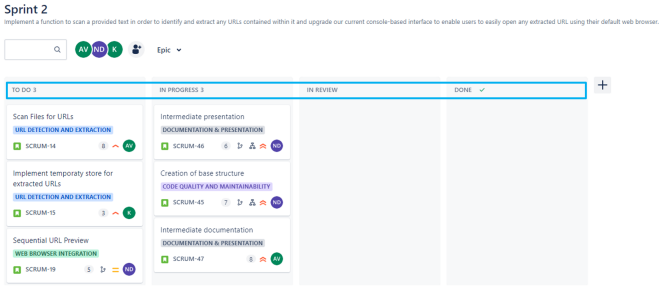
Spint 2

Sprint Goal



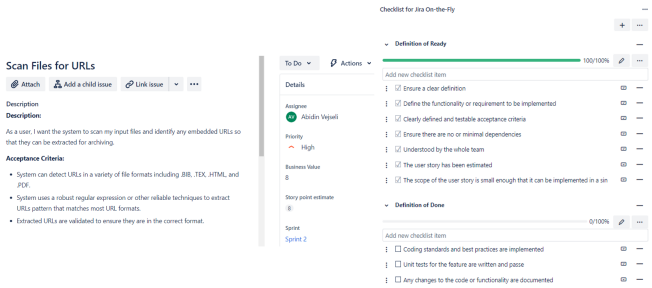
Spint 2

Categories



User Story

Description, Acceptance Criteria, DOR and DOD



Our learnings

Coding	Scrum
Branch per User Story	Create Tasks
Commit Messages	User Story good size
Code Reviews	Weeklys are valuable
Basic code structure	

Questions?