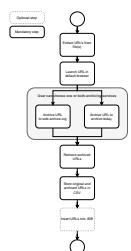


Table of Content

Topics Covered

- ▶ Problem Statement
 - Project Goal
 - Requirements
- ▶ Solving The Problem
 - Process Model
 - Architecture
 - Data Model
 - Technologies
- ▶ Project Management with SCRUM
 - Scrum Team
 - Our Scrum Adoptions
 - Scrum implementation
- ▶ Questions?



Problem Statement

Problem Statement "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



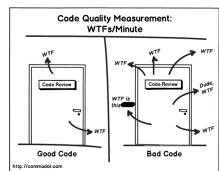
Functional Requirements

- 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

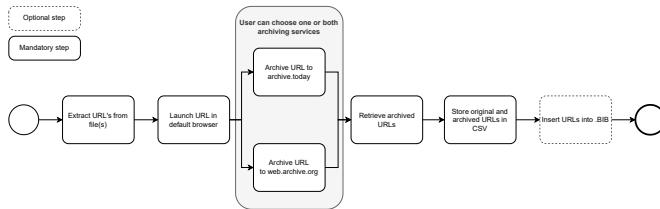


Solving The Problem

Process Model

Workflow for URL Extraction and Archiving

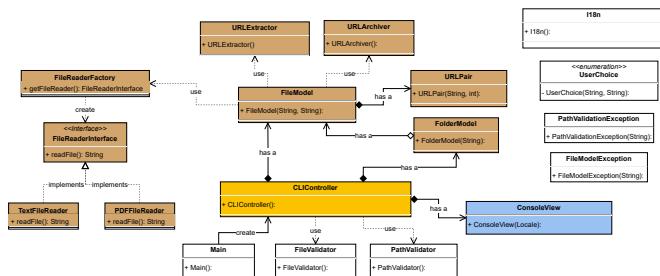
- User can skip URLs, launch them, access help or quit the program at any time



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

High Level Class Diagram

Overview

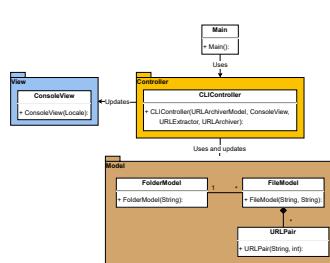


- To demonstrate how we meet the requirements for our application, which Kilian presented earlier, I will now provide a high-level overview of our structure and its individual components.
- It is important to note that this is our current structure, subject to expansion and adaptation as the project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

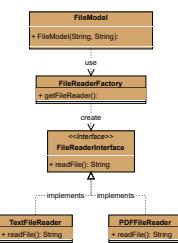


- As you may have observed, we utilize the MVC pattern, among other techniques, to ensure the scalability of the application.
- This involves separating the business logic from the presentation using a controller.
- This enables us to easily switch out the Command Line Interface with a Graphical User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

Architecture

Factory Pattern

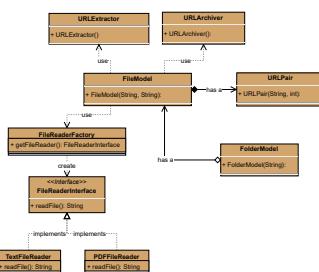
- Scalable and robust design
- Ensures maintainability and clear code structure
- Commitment to best programming practices



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 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 Adoptions

Sprint

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

Our Scrum Adoptions

Sprint Planning

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

Our Scrum Adoptions

Daily Scrum

Sprint	Sprint planning	Daily Scrum	Sprint Review	Sprint Retro
		<ul style="list-style-type: none">• Daily• Twice a week• Current state• MS Teams		

Our Scrum Adoptions

Sprint Review

Sprint	Sprint planning	Daily Scrum	Sprint Review	Sprint Retro
			<ul style="list-style-type: none">• Last Day• Completed?• Problems?• Testing• Product increment	

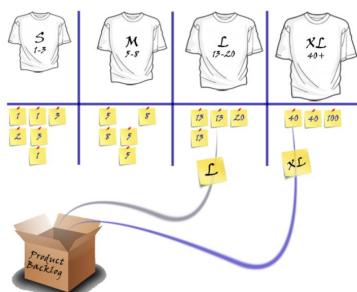
Our Scrum Adoptions

Sprint Retro

Sprint	Sprint planning	Daily Scrum	Sprint Review	Sprint Retro
				<ul style="list-style-type: none">• Last Day• Feedback• Do's & Don'ts

Estimation Method

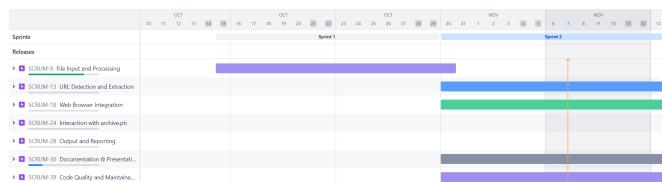
T-Shirt Sizes



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences

25

Velocity



Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences

26

Backlog

	Q1001-10 Store URLs Line Number or Content	URL DEFINITION AND USE	90% BOLD	
	Q1001-11 Complete a List of URLs	URL DEFINITION AND USE	90% BOLD	
	Q1001-12 Immediate Archiving Upon Decision	URL DEFINITION AND USE	90% BOLD	
	Q1001-13 Track Archiving Progress	URL DEFINITION AND USE	90% BOLD	
	Q1001-14 Store User Decisions for Reporting	URL DEFINITION AND USE	90% BOLD	
	Q1001-15 Automate URLs Scanning	URL DEFINITION AND USE	90% BOLD	
	Q1001-16 User Interaction for Capabilities	URL DEFINITION AND USE	90% BOLD	
	Q1001-17 Automatic Retrieval of Archived URLs	URL DEFINITION AND USE	90% BOLD	
	Q1001-18 Generate PDF file	URL DEFINITION AND USE	90% BOLD	
	Q1001-19 Integrate Archived URLs into Superset Hess	URL DEFINITION AND USE	90% BOLD	
	Q1001-20 User Manual	DOCS AND TUTORIALS	90% BOLD	
	Q1001-21 Intermediate presentation	DOCS AND TUTORIALS	90% BOLD	
	Q1001-22 Final presentation	DOCS AND TUTORIALS	90% BOLD	
	Q1001-23 UML diagram	DOCS AND TUTORIALS	90% BOLD	
	Q1001-24 Installation manual & script	DOCS AND TUTORIALS	90% BOLD	
	Q1001-25 Requirements document	DOCS AND TUTORIALS	90% BOLD	
	Q1001-26 Implementation documentation	DOCS AND TUTORIALS	90% BOLD	

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences

27

Epics

The screenshot shows the Jira interface for the 'File Input and Processing' epic. On the left, there's a sidebar with a tree view of epics: 'Issues without epic' (selected), 'File Input and Processing', 'URL Detection and Extraction', 'Web Browser Integration', 'Interaction with archive.ph', 'Output and Reporting', 'Documentation & Presentation', 'Code Quality and Maintainability'. The main area is titled 'File Input and Processing' with a sub-section 'Description' containing the text: 'As an user I want to input various file types via the command line so that they can be prepared for further processing.' Below this is a table titled 'Child issues' with the following data:

ID	Name	Status	Order
SCRUM-10	Automatic File Type Detection	DONE	13
SCRUM-12	Processing Feedback	DONE	1
SCRUM-9	Prompt for File Path Input	NOT DONE	3
SCRUM-11	Processing of Directories	NOT DONE	3
SCRUM-48	Add user option for processing new path	TO DO	1

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences

28

Sprint 2

Sprint Goal

Sprint 2

The screenshot shows the Jira sprint backlog for Sprint 2. At the top, it says '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.' Below this is a table with columns: 'TO DO', 'IN PROGRESS', 'IN REVIEW', and 'DONE'. The 'TO DO' column contains tasks like 'Scan Files for URLs' (under 'URL DETECTION AND EXTRACTION'), 'Implement temporary store for extracted URLs' (under 'URL DETECTION AND EXTRACTION'), and 'Sequential URL Preview' (under 'WEB BROWSER INTEGRATION'). The 'IN PROGRESS' column contains tasks like 'Intermediate presentation' (under 'DOCUMENTATION & PRESENTATION') and 'Creation of base structure' (under 'CODE QUALITY AND MAINTAINABILITY'). The 'IN REVIEW' and 'DONE' columns are currently empty.

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences

29

Sprint 2

Categories

Sprint 2

This screenshot is identical to the one above, showing the Jira sprint backlog for Sprint 2. The main difference is that the 'TO DO' and 'IN PROGRESS' columns now have a light blue background, indicating they belong to the 'URL DETECTION AND EXTRACTION' category. The other columns ('IN REVIEW' and 'DONE') remain white.

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences

30

User Story

Description, Acceptance Criteria, DOR and DOD

Scan Files for URLs

Checklist for Jira On-the-Fly

To Do Actions

Details

Assignee: Aladin Vujelj

Description: As a user, I want the system to scan my input files and identify any embedded URLs so that they can be extracted for archiving.

Acceptance Criteria:

- System can detect URLs in a variety of file formats including BB, TEX, HTML and PDF.
- System uses a robust regular expression or other reliable techniques to extract URLs pattern that matches most URL formats.
- Extracted URLs are validated to ensure they are in the correct format.

Definition of Ready

Add new checklist item 100/100%

- Ensure a clear definition
- Define the functionality or requirement to be implemented
- Clearly defined and testable acceptance criteria
- Ensure there are no or minimal dependencies
- Understood by the whole team
- The user story has been estimated
- The scope of the user story is small enough that it can be implemented in a single iteration

Definition of Done

Add new checklist item 0/100%

- Coding standards and best practices are implemented
- Unit tests for the feature are written and pass
- Any changes to the code or functionality are documented

Sprint: Sprint 2

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences

31

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?