Continuously Extensible Information Systems: Extending the 5S Framework by Integrating UX and Workflows

Prashant Chandrasekar

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Edward A. Fox, Chair
Noshir Contractor
Chris Franck
Steve Harrison
Chang-Tien Lu

May 11, 2021
Blacksburg, Virginia

Keywords: Workflows, UX Integration, 5S, Knowledge Graph

Continuously Extensible Information Systems: Extending the 5S Framework by Integrating UX and Workflows

Prashant Chandrasekar

(ABSTRACT)

In Virginia Tech's Digital Library Research Laboratory, we support subject-matter-experts (SMEs) in their pursuit of research goals. Their goals include everything from data collection to analysis to reporting. Their research commonly involves an analysis of an extensive collection of data such as tweets or web pages. Without support – such as by our lab, developers, or data analysts/scientists – they would undertake the data analysis themselves, using available analytical tools, frameworks, and languages. Then, to extract and produce the information needed to achieve their goals, the researchers/users would need to know what sequences of functions or algorithms to run using such tools, after considering all of their extensive functionality. Our research addresses these problems directly by designing a system that lowers the information barriers. Our approach is broken down into three parts. In the first two parts, we introduce a system that supports discovery of both information and supporting services. We, firstly, describe the methodology that incorporates User eXperience (UX) research into the process of workflow design. Through the methodology, we capture (a) what are the different user roles and goals, (b) how we break down the user goals into tasks and sub-tasks, and (c) what functions and services are required to solve each (sub-)task. We, secondly, identify and describe key components of the infrastructure implementation. This implementation captures the various goals/tasks/services associations in a manner that supports information inquiry of two types: (1) Given an information goal as query, what is the workflow to derive this information? and (2) Given a data resource, what information can we derive using this data resource as input? We demonstrate both parts of the approach,

describing how we teach and apply the methodology, with three case studies. In the third part of this research, we rely on formalisms used in describing digital libraries to explain the components that make up the information system. The formal description serves as a guide to support the development of information systems that generate workflows to support SME information needs. We also specifically describe an information system meant to support information goals that relate to Twitter data.

Continuously Extensible Information Systems: Extending the 5S Framework by Integrating UX and Workflows

Prashant Chandrasekar

(GENERAL AUDIENCE ABSTRACT)

In Virginia Tech's Digital Library Research Laboratory, we support subject-matter-experts (SMEs) in their pursuit of research goals. This includes everything from data collection to analysis to reporting. Their research commonly involves an analysis of an extensive collection of data such as tweets or web pages. Without support – such as by our lab, developers, or data analysts/scientists – they would undertake the data analysis themselves, using available analytical tools, frameworks, and languages. Then, to extract and produce the information needed to achieve their goals, the researchers/users would need to know what sequences of functions or algorithms to run using such tools, after considering all of their extensive functionality. Further, as more algorithms are being discovered and datasets are getting larger, the information processing effort is getting more and more complicated. Our research aims to address these problems directly by attempting to lower the barriers, through a methodology that integrates the full life cycle, including the activities carried out by User eXperience (UX), analysis, development, and implementation experts. We devise a three part approach to this research. The first two parts concern building a system that supports discovery of both information and supporting services. First, we describe the methodology that introduces UX research into the process of workflow design. Second, we identify and describe key components of the infrastructure implementation. We demonstrate both parts of the approach, describing how we teach and apply the methodology, with three case studies. In the third part of this research, we extend formalisms used in describing digital libraries to encompass the components that make up our new type of extensible information system.

Dedication

For Amma and Appa. Thank you for your sacrifices and for providing with every and all opportunities. To my partner-in-crime, Ayshwarya: In every which way, this effort is incomplete without you. To Praveen, Akka and Athimber: Thank you for your inspiration and unbridled love. To Mambalam parents and family: Thank you for your positive energy and support. I am eternally grateful and indebted to you all.

Acknowledgments

This work was funded in part through The Social Interactome of Recovery: Social Media as Therapy Development, NIH Grant 1R01DA039456-01; Global Event and Trend Archive Research, NSF IIS-1619028 and 1619371; CRISP Type 2/Collaborative Research: Coordinated, Behaviorally-Aware Recovery for Transportation and Power Disruptions (CBAR-tpd), NSF CMMI-1638207, subcontract to VT; and CINES project: subcontract from UVA on NSF OAC-1835660: Collaborative Research: Framework: Software: CINES: A Scalable Cyberinfrastructure for Sustained Innovation in Network Engineering and Science. I would like to acknowledge my advisor, Dr. Fox. He has been a wonderful teacher and has patiently guided me in my research. My research process was challenging at a personal level. He provided me with calmness and perspective and helped me overcome what I was going through. I hope to take forward all the lessons he has taught me. Thank you, Dr. Fox.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Imagine a scenario where a subject-matter-expert (SME), e.g., a sociologist, wants to study how different cohorts tweet for a specific kind of event, relative to a new theory. If the SME is familiar with the process of collecting tweets and working with the raw data, then they can do so quickly, before moving on to test their theory or tackle their research goal. If not, they must adopt a different plan that starts with addressing the challenge of collecting data. To collect data, they would do one or both of:

- go through the literature to find openly accessible datasets related to their research, or
- go onto Twitter and manually scan for relevant tweets; copy the text, user details, and other information one tweet at a time; and store them in some format.

After collecting the data, they still have the task of going through the tweets, filtering out non-relevant ones, and, possibly, extracting certain semantic information or derived measures, like tweet topics or tweet sentiment, through manual effort. What if there are no datasets openly accessible? What if testing their theory requires advanced analysis? What if they have to go through millions of tweets manually? The extent to which the situation becomes complicated depends on the scope of the research problem: the number of data sources, the complexity of analysis, the amount of analysis, etc.

Researchers can take it upon themselves to scour the web, learn a new language or framework

or tool for their data collection and analytical needs, and spend a great deal of time. Given the expectation of using "big data," and the pace at which newer methods get built, this approach is not scalable. Furthermore, in long-term studies, researchers are forced to learn new technology and methodology that relates to their work. They face serious data analysis knowledge barriers.

One hope for SMEs, to aid with these challenges, is that they can leverage analytical solutions, frameworks, and workflow-based engines that allow researchers to produce and share their solutions. Some engines and data analysis workflow repositories cater to a particular research domain. The interface for these workflow solutions assumes that the intended user knows how to break down their problem into tasks, and knows what libraries or data mining functions they need to call upon to solve each task. We cannot expect all SMEs to have this background knowledge. Without such background knowledge, and task-oriented problem solving skills, these tools may not prove to be helpful to the SMEs. However, a typical SME supervisor has a view that the primary effort exerted by SMEs should be on research work that requires their domain-specific expertise.

At the Digital Library Research Laboratory at Virginia Tech, we want to provide an opportunity for SMEs to conduct their research despite the data/text mining knowledge barriers.

There are many critical technical challenges we face as we pursue this effort. Firstly, we need to build a system that scales and extends with the evolving nature of SMEs' exploratory needs and the advancements in data mining solutions. Secondly, we need to find a way to represent the relationship between information goals/states and developer-built data mining-centric workflow services that address previously identified goals. That representation also must facilitate reasoning about, as well as generating, workflows for newer goals. Thirdly, we need to capture and identify the properties and components of a class of workflow-based information systems that help SMEs conduct their research despite their knowledge

barriers. Building this extensible system and defining it formally will help future researchers and developers build similar systems by following our research approach.

We propose and teach a methodology that introduces the User eXperience (UX) research process into the process of workflow design. The methodology makes the integration of these two design processes possible. As a result, we can identify and extract goals, (sub-)tasks, services, and, most importantly, their associations. When correctly mined and represented, we believe that these associations will help us infer relationships among information states and goals. We propose a knowledge graph built on the knowledge base of these associations, and a reasoner based on algorithms that leverage the knowledge to produce results that satisfy the SMEs' information needs. When implemented, the knowledge graph will support generation of workflows for information goals (e.g., queries), which will produce desired results when executed by a workflow engine. The workflows represent the functionality of an information system that supports the data needs of SMEs. We showcase the power of the methodology and the use of a suitable knowledge graph for three case studies. We then formally describe a class of such information systems that should aid developers in building instances that specialize on their category of information states and workflow services.

1.1 Overview of Chapters

The rest of this document is organized as follows:

1. Chapter 2 outlines related efforts for different aspects of this research.
2. Chapter 3 outlines research questions and conjectures/hypotheses.
3. Chapter 4 describes the first part of our research solution, a methodology to design extensible information systems.

4. Chapter 5 describes the second part of our research solution, including an identification of core components of such information systems.

5. Chapter 6 employs formalisms to describe a class of workflow-based information systems.

6. Appendix A provides supplemental information on the system solution developed by students in the graduate CS5604 course taught during Fall 2020.

7. Appendix B provides supplemental information on the knowledge graph proof-of-concept built by a student team in the Spring 2021 CS4624 class.

Chapter 2

Review of Literature

In this chapter we discuss previous attempts to lowering knowledge barriers for SMEs. We then go over alternate representations of workflow-related information and why they fall short to out requirements. Following that, we discuss related efforts that one could employ to "reason" with information represented via our chosen data structure. Finally, we discuss the 5S framework that we use in the final part of our research to describe a class of information systems.

2.1 Workflow Systems and Other Efforts To Support SME Information Needs

In this section, we discuss efforts by the research community toward reducing the knowledge barrier for SMEs. We can group these efforts into the following categories:

1. publications and online notebooks, and

2. workflow management systems and repositories.

The first group of efforts is geared towards encouraging reproducibility. The second is helpful for audiences that are more familiar with technology and development processes. As you will

see below, while both sets of efforts are beneficial in their way, they do not directly address the principal pain point.

2.1.1 Related Effort Category 1: Publications and Online Notebooks

Conference proceedings, journals, and workshops provide a platform for researchers to share their work. A natural benefit of sharing each others' work is that readers can learn from other authors' experiments. The readers could gain enough knowledge to attempt the same experiments if they so desire. These articles serve as a useful resource for SMEs, which enables them to carry out analyses for their research. However, as is well known, there is no standard schema enforced upon the authors of these articles. Without a standard, inconsistencies arise, which are compounded by changes in environment and technology, such as software version shifts. It may also be the case that the articles might not provide all of the information the SMEs need to replicate the experiments.

Furthermore, Gil et al. observe that computational methods are getting more and more complex. Therefore, they are more difficult to explain if the intended goal is to teach others [25][23].

To bridge that gap, and to promote transparency and reproducibility, the idea of publishing code along with literary work was adopted. Electronic tools like the Jupyter notebook became a popular medium for sharing code [37]. Jupyter notebooks are executable documents that allow authors to present code, rich text, figures, tables, and overall results, all in one document. When properly written, the document can be shared such that researchers can re-run the document by themselves. However, if poorly written, the code might only execute in the presence of specific libraries. If the author does not provide a way to indicate that, or

to package it in the document, the code will not execute. If the document contains sufficient details, it could be interpreted with ease. As is the case with traditional publications, there are no set standards on what should or should not be included in the document. The same is true for content presentation. While Jupyter notebooks have their merits, we still are faced with many of the same deficiencies we saw with traditional publications.

2.1.2 Related Efforts Category 2: Workflow management systems and frameworks to create workflows

The evolution of workflow management systems (WMSs) has been a natural consequence of advances in computer technology, an increase in digital sensors, and as a by-product, an increase in the volume of observational data and any data collected through automation. When Jim Gray talks about the Fourth paradigm and the "Transformed Scientific Method," he notes the change in how science is conducted and the increased investment into information management systems to support science. He calls for an effort to not only archive journals and publications, but to build an ontology for the data and the processes. He recommends that the scientific community foster digital data libraries for both data and literature [33].

Liew et al. reference Taylor et al., as well as Goble and De Roure and Gorlach et al., when stating that WMSs: (a) support collaborative research, (b) construct workflows without concern about resources and workflow execution, (c) automate steps for reproducibility and simulation-based studies, (d) integrate resources (data and processing power) from heterogeneous sources, and (e) optimize workflow execution [27, 29, 38, 49]. Workflow management systems, if built keeping the SMEs' interests in mind, can indeed provide these benefits.

Many WMSs exist, each with varying characteristics. Pegasus is a popular WMS [12]. It uses Wings for workflow composition and provenance tracking [22]. Pegasus takes care of

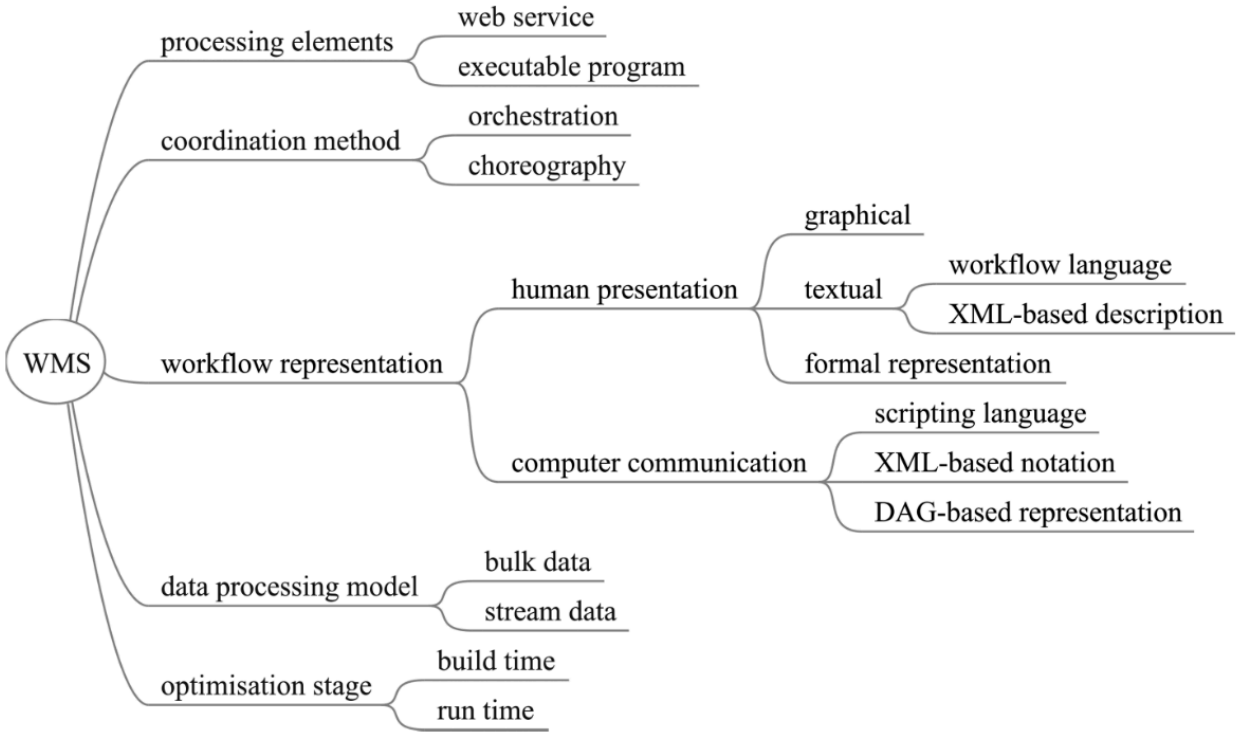


Figure 2.1: Taxonomy of workflow management system characteristics

resource mapping and workflow execution. It integrates various catalogs (libraries) that all contribute in the workflow planning process. Like Pegasus, Apache Taverna is an open-source WMS [63]. It is part of a suite of tools built by the myGrid team [35]. Taverna comes with a workbench, where one could design workflows by selecting services that they want to execute, and in their preferred order. Taverna has a library of 3500-plus services to select from. Workflows are published into a popular workflow repository known as myExperiment [10]. It also comes with a player called the Taverna Player that allows one to "replay" workflows via the browser.

Similarly, there are many more powerful WMSs such as KNIME, Kepler, Galaxy, etc. [4, 5, 39]. Liew et al. built a taxonomy of architectural characteristics that one can use to evaluate and compare WMSs [38]. This taxonomy can be found in Figure 2.1. When we compare

the WMSs, we find many similarities in their design. These WMSs support research in various domains such as earth science, astronomy, chemistry, and geology. The focus of the developers of these WMSs is to provide the resources for conducting their experiments. These systems empower researchers to conduct experiments that support up to a million tasks and process petabytes of data, while integrating libraries deployed on various platforms. One of the main aims of these WMSs is to help the users conduct their experiments without having knowledge of workflow execution and optimization. However, the onus is still on the user to create the workflow. That being said, it is not always the case that they have to create a workflow from scratch.

Users of these systems are provided an interface to search, download, edit, and re-run published workflows. These WMSs have a repository that showcases the workflows that have been built by developers, engineers, and scientists. The Wings platform takes it a step further by additionally sharing workflow templates. Each template is designed to run a particular task [24]. The purpose of sharing the template is two-fold: (a) It helps educate the viewer regarding the different steps required to complete a task, the constraints that these tasks impose on the input data, and the properties of the output; and (b) it provides the ability for users to create their own workflows from the templates.

Pegusus uses the Wings platform for workflow design. Wings has a repository of workflows and tasks. Taverna workflows are shared on the myExperiment platform. Kepler has a workbench that allows search in a similar fashion. WMSs address the issue of workflow creation by allowing users to search and browse through workflows, workflow templates, individual processes, and other workflow-related artifacts. Domain-specific WMSs share, and provide search on, algorithms and datasets commonly found in that domain.

However, these interfaces are not particularly helpful for an audience of SMEs who might have little to no information on the tasks and files required. That knowledge barrier is not

addressed by the WMSs mentioned above.

Note that there are similar problems found in other environments where routines of various kinds are to be integrated. There is a long list of such environments, including:

1. IDEs and other tools to build programs using selections from libraries or sets of routines (e.g., Netbeans [52], Eclipse[13]),

2. R and other combination programming language / environment systems [55],

3. no-code or low-code application development platforms with associated collections of components (e.g., Mendix, Outsystems)[41],

4. Docker and other tools that facilitate the creation, deployment, and execution of applications using containers,

5. mashup enablers and similar tools that connect resources (e.g., Presto, Convertigo, and Caspio Bridge),

6. mobile operating system app stores (e.g., Google Play, Apple Store), and

7. integrators or connectors (e.g., Zapier) that work with APIs or separate WWW services.

While in the following we use the term "workflow" to refer to a connected collection of components, and the integration to be done by WMSs, the approach described can apply as well in all such environments.

This situation provides the motivation and basis for our research and our approach.

2.2 Workflow Representations

In our research effort, we design a hypergraph-based representation (component of the workflow-based information system) that stores the associations between workflow services and connects information states. In this section, we discuss the constraints of using alternate workflow representations.

The end goal of the information system is to provide a workflow for every information goal requested by the SME. One way we could do that is to represent the goal <-> <sequence of workflow services> association as a record in a data structure like a dictionary. In the dictionary, we would have a record for each goal, where the "key" would be the goal, and the "value" would be the sequence of services. The Common Workflow Language (CWL) and Workflow Description Language (WDL) are the most commonly used workflow representations in all of the workflow management systems that we have discussed before [1, 36].

```
workflow myWorkflowName {
    File my_ref
    File my_input
    String name

    call task_A {
        input: ref= my_ref, in= my_input, id= name
    }
    call task_B {
        input: ref= my_ref, in= task_A.out
    }
}
```

Figure 2.2: Example workflow defined using Workflow Description Language. [36]

Both of these languages or representations require that workflows are defined or "declared" as

```
#!/usr/bin/env cwl-runner

cwlVersion: v1.0
class: Workflow
inputs:
  tarball: File
  name_of_file_to_extract: string

outputs:
  compiled_class:
    type: File
    outputSource: compile/classfile

steps:
  untar:
    run: tar-param.cwl
    in:
      tarfile: tarball
      extractfile: name_of_file_to_extract
    out: [extracted_file]

  compile:
    run: arguments.cwl
    in:
      src: untar/extracted_file
    out: [classfile]
```

Figure 2.3: Example workflow defined using Common Workflow Language. [1]

a contiguous block as shown in Figures 2.2 and 2.3. Using a dictionary-based representation would facilitate that type of user query. However, with this representation alone, we lose associations among user goals. All we would have is a dictionary with an entry for each goal. It could be the case that two goals share services or that one goal is a sub-goal of another. However, we lose the chance to make these inferences with this representation. Furthermore, representing the workflow as a "contiguous block" makes it challenging to maintain the dictionary of workflows. For instance, let us assume that we need to modify a service between two existing services in a workflow. To make this change in the dictionary, we would have to go through every item of the dictionary and search for the services' sequences before modifying them. That is one of the main consequences of representing workflows as a contiguous block. Modifying the "block" becomes cumbersome.

We could represent workflows as either Control-flow graphs or Data-flow graphs [38]. Control-flow graphs are made up of tasks as nodes and edges specifying the order of execution. In data-flow graphs, tasks (edges) represent the flow of the data (nodes). In either case, workflows are represented as directed-acyclic-graphs or directed-cyclic graphs (to support iteration). Regardless of which abstraction one chooses to adopt, the constraint we face is that we need to represent not one but multiple workflows for a particular end goal (state of information). We want to represent task precedence or data dependency AND multiple paths to a particular information/node state. Regardless of the abstraction we select, we cannot represent both types of information using a "binary" edge.

2.3 Reasoner Algorithms / Implementations

In our research effort, we design a context-free-grammar-based Reasoner (component of the workflow-based information system) that generates workflows based on SME information

needs. In this section, we look at alternative reasoners that one could adopt in their efforts to mine and derive associations between information states.

There are Semantic Web-based reasoners that might derive connections between services and information states represented in a knowledge graph [61]. RDF knowledge stores like Apache Jena provide a set of default reasoners [51]. For instance, you have the "transitive reasoner" that traverses classes and properties such as "rdfs:subPropertyOf" and "rdfs:subClassof." Additionally, Jena and other RDF stores or knowledge graph databases provide rule-based reasoners. These are like the reasoner that we have implemented (implementation described in Appendix B) [56]. The reasoner can then traverse the graph and form associations. Then, using a graph querying language, an SME or user can find links between information states and links between services.

There also have been instances of AI planning-based reasoning used for workflow generation. In the early stages of the development of the Pegasus workflow management system, the developers used AI planning for grid computing [11, 26]. The focus was to address the challenge of providing distributed computation. The scientists and developers first construct an abstract workflow. Their description of the abstract workflow includes metadata and constraints for files and operations, respectively. Pegasus generates a workflow by selecting appropriate application components, assigning the required computing resources, and overseeing the successful execution, optimized based on the estimated run-time. The preconditions include constraints on feasible hosts and data dependencies on input files. The "plan", which is the output of the AI planner, represents an executable workflow. Their system utilized an existing planner, Prodigy, and represented their concrete workflow construction, with resource mapping, as a planning instance problem [57]. As a result of their efforts, they have supported scientists conducting experiments in various settings [53, 54]. The Pegasus approach is closest to our approach, regarding using AI planning to generate

a workflow. However, they operate at a lower level. They work with the Workflow Manager and generate a concrete workflow plan based on computing resources, data flow, etc. This coordination is how they claim that they can support and optimize workflows of over 100 tasks. It could be beneficial, in the long term, to build scalable information systems to integrate with a system like Pegasus. In the near future, it would be most beneficial to explore AI-planning-based reasoning. We could represent our planning domain and problem using PDDL (Planning Domain Definition Language) [32]. AI Planning uses state-space and searching algorithms such as BFS, A*, Dijkstra's, etc., to search solution plans to solve planning problems. There has been some recent work in building an alternative "planning graph" that improves the complexity of the search algorithm by reducing the state space [44]. The authors of [44] state that this addresses the expressiveness and complexity issues raised in the AI planning book "Automated Planning: Theory and Practice" [21]. Future implementations of the reasoner could optimize the information system's inner workings, thereby improving the support provided to SMEs.

2.4 Formal Descriptions of Digital Libraries: 5S

In our research, we aim to develop a formal description of a workflow-based information system. The formal description of a workflow-based information system or digital library will help capture all the components and the processes connecting them. We use the 5S formal framework to create this description [19]. In this section, we describe aspects of the 5S framework.

5S stands for five constructs – Streams, Structures, Spaces, Scenarios, and Societies – that, when described, help us capture all aspects of a digital library system. With Streams, we can capture all of the information that passes through the system. With Structures, we

capture organizational aspects of the data, the library as a whole, and everything in between. Spaces help us identify and define any stored or presented information in n-dimensional space, using any mathematical space. Scenarios represent every opportunity of interaction between components within the library framework or interaction with any entity external to the library framework. The "actors" that participate in the interactions, both internal and external, are identified in Societies, as are their relationships. "Digital Library Applications" showcases the use of 5S in five different areas of information needs. As noted by the authors, 5S can describe information systems in general, thereby having broader applicability [18].

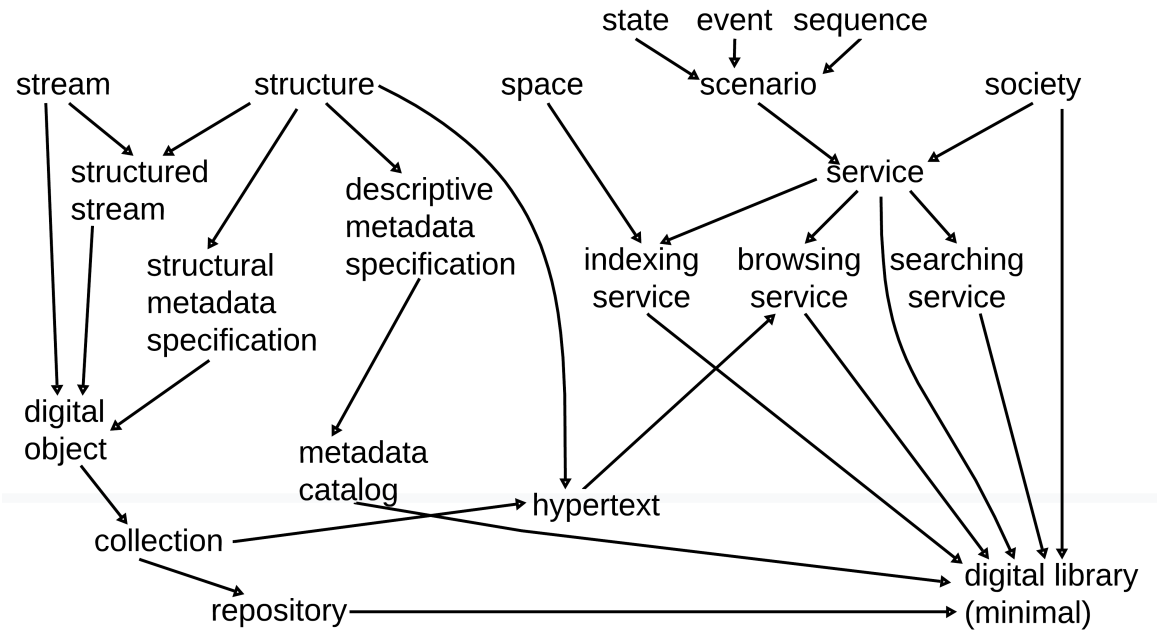


Figure 2.4: Constructs that together make a minimal digital library.

Figure 2.4 identifies the constructs that, in combination, help define a "minimal" digital library. A minimal digital library is "the minimal set of components that make a digital library, without which, in our view, a system/application cannot be considered a digital library". The formal definition of a minimal digital library and its constructs has allowed researchers to extend or instantiate minimal versions of digital libraries for various domains [19]. They have been able to do so by defining new constructs and/or instantiating existing

constructs to apply to the domain in question. Over the years, researchers have described many such (extensions of) minimal digital libraries. In my view, these could function as "design patterns" as described in the software engineering world [43, 45]. In the same manner that design patterns serve as a solution template for a class of problems, these minimal digital library descriptions guide developers looking to build digital library solutions to solve a class of problems.

2.4.1 Minimal Digital Library: Specific components/concepts

The definitions for specific components of the minimal digital library description are, as found in [19]:

1. State: A state is a function, from labels L to values V. A state set S consists of a set of state functions $s : L \rightarrow V$.

2. Transition Event: A transition event (or simply event) on a state set S is an element $e = (s_i, s_j) \in (S \times S)$ of a binary relation on state set S that signifies the transition from one state to another.

3. Scenario: A scenario is a sequence of related transition events $< e_1, e_2, ..., e_n >$ on state set S such that $e_k = (s_k, s_{(k+1)}), for\ 1 \leq k \leq n$.

4. Service: A service, activity, task, or procedure is a set of scenarios.

5. Descriptive Metadata Specification: Let $L = \cup D_k$ be a set of literals defined as the union of domains D_k of simple datatypes (e.g., strings, numbers, dates, etc.). Let also R and P represent sets of labels for resources and properties, respectively. A descriptive metadata specification is a structure $(G, R \cup L \cup P, F)$, where:

(a) $F : (V \cup E) \rightarrow (R \cup L \cup P)$ can assign general labels $R \cup P$ and literals from L to nodes of the graph structure;

(b) for each directed edge $e = (v_i, v_j)$ of G, $F(v_i) \in R \cup L$, $F(v_j) \in R \cup L$ and $F(e) \in P$;

(c) $F(v_k) \in L$ if and only if node v_k has outdegree 0.

6. Metadata Catalog: Let C be a collection (which are a set of digital objects) with k handles in H. A metadata catalog DM_C for C is a set of pairs $(h, dm_1, ..., dm_{kh}), where\ h \in H$ and the dm_i are descriptive metadata specifications.

Our 5S-based description, outlined in Chapter 6, will be built using these constructs. The description should aid developers looking to build a workflow-based digital library or information system.

Chapter 3

Research Overview

Our research derives from considerations of digital libraries, but can readily be generalized to a broad range of types of information systems. Further, though to many the scope of "information discovery" is confined to content external to an information system, i.e., that it manages, we view an even broader scope, indeed a recursive one, where the system itself – including artifacts that guide its construction, and its various services, service integrations, data, knowledge, and components – also can be explored.

3.1 Research Questions and Conjectures

To complete our research, we need to find solutions to many questions. Our solutions will be shown effective through three case studies, to highlight their generality. Accordingly, the scope of our hypotheses is assumed to be relative to those three situations, ensuring they are falsifiable, though we conjecture their scope is over a much larger class of information systems, at least including digital libraries. The central research questions and related conjectures are as follows:

R1: How do we improve the process of designing an information system **to facilitate information discovery** by engaging SME, UX, and DevOps personnel in a collaborative co-design effort?

1. **R1.1:** How do we identify (a) information that SMEs are looking for, and (b) background information such as output format desired and input files they might have?

2. **R1.2:** How do we associate the SME goals with steps that would form a workflow?

C1: Students[1], taking up the roles of UX researchers and developers, trained in our methodology, will be able to build a workflow-based information system that SMEs could use to answer their goals.

R2: How do we implement such an information system, using knowledge graphs supported by a query language and capability?

C2: If we develop an exhaustive knowledge graph, based in part on workflows, then a reasoner can translate any previously identified SME goals in that domain into workflows that when executed yield appropriate results for each goal/query.

R3: How do we describe, so as to facilitate its implementation, such information systems?

C3: We conjecture that it suffices when devising such an information system for a particular domain for it to include:

1. what constitutes a minimal digital library;

2. three additional components defined according to the 5S framework [19] – specifically a knowledge graph infrastructure, services registry, and reasoner; and

3. a services pool and knowledge graph instance constructed through our collaborative methodology involving UX, SME, and DevOps experts.

We hypothesize this is the case for the 3 case study domains chosen.

[1]We refer specifically to majors in computer science, who are seniors or graduate students.

Input and collaboration between SMEs, UX researchers, and DevOps personnel informs our discovery for research questions 1 and 2.

3.2 Overall Research Contributions

Through our research, we devise the following:

1. A methodology that identifies and extracts user goals and workflow services to support the (sub-)tasks that make up the goal;
2. A knowledge graph built to generate a set of workflows designed to provide the answer for information queries;
3. A demonstration of the methodology and the building of a knowledge graph by way of case studies;
4. A working prototype of components of a workflow-based extensible information system; and
5. A 5S-based description of such information systems, that include a set of workflows supporting SME information needs.

3.3 Research Plan Overview

We break the research into three parts. In the first part, we describe the methodology that captures the SME needs, and through collaboration with developers and UX-researchers, breaks down their information goals into smaller units. The developers identify these units as being sub-goals, for which they can build services. We have applied the methodology in

various case studies. We also taught the methodology to student teams in undergraduate and graduate courses. In the second part, we describe a knowledge graph-based representation that organizes the information from the artifacts (produced from the methodology) and a reasoner for that graph that supports the SMEs' queries for information. Through student teams' efforts from CS5604: Information Storage and Retrieval, a proof-of-concept of the knowledge graph and the reasoner was built. We report on feedback provided by student teams and by a class of SMEs. Finally, in the third part, we describe the digital library using 5S (Streams, Structures, Spaces, Scenarios, and Societies) formalisms [17, 18, 19, 46]. We identify and define the digital objects and the components of a workflow-system-based digital library. Part 1 of the research is discussed in Chapter 4, part 2 in Chapter 5, and part 3 in Chapter 6.

Chapter 4

Research - Part 1: Workflow Design

This chapter describes the first part of our effort to build a workflow-based information system solution that supports SMEs. We discuss the methodology that will help design a set of workflows built to answer specific SME needs.

Figure 4.1 provides an overview of the steps of the methodology along with the information that is produced (and described) through each step. The outcome of executing each step brings us closer to provide a complete description of the workflows that define the information system. We can use the 5S framework to capture this information [19]. 5S stands for five constructs – Streams, Structures, Spaces, Scenarios, and Societies – that, when described, help us capture all aspects of a digital library[1]. We address the formal descriptions of the digital library to support this methodology and the description of the workflow-based information system, produced through this methodology, in the third part of the research (see Chapter 6).

In the first step of the methodology, we identify the SME goals using the expertise of a UX researcher. In the second step, we consult with the developer, analyst, or scientist, and break down the goal into specific tasks. In the third step, using the experience of the developer, and the information captured in the previous steps, we describe all implementation-specific functions and entities that are requisite for solving each task. Using all of the information, we construct a model of the workflow. We should store these workflows in a knowledge base

[1] Background information on 5S can be found in Section 2.4 of Chapter 2.

Steps of Methodology / 5S Description of workflows	5S-Society	5S-Scenario	5S-Structure	5S-Stream	5S-Space
Step 1: Characterize user role(s) and goal(s)	User persona: a) user role; b) area of expertise; etc.	Task(s) (if extracted)		Goal(s)	
Step 2: Breakdown of goal(s) in task(s) and sub-task(s)	User persona: a) user role; b) area of expertise; etc.	Task(s) and sub-task(s)	Graph depicting relationship between goals, tasks and sub-tasks	Goal(s)	
Step 3: Identify functions and build services	User persona: a) user role; b) area of expertise; etc.	Task(s) and sub-task(s); Libraries; Services; Functions; etc.	Graph depicting relationship between goals, tasks and sub-tasks	Goal(s); Input data; Output data	Models; Visualization; etc.

Figure 4.1: Methodology that describes workflows. The color indicates the step of the methodology when we produce or extract the artifacts. Blue indicates artifacts produced from step 1 of the methodology. Similarly, orange is for step 2 and green for step 3.

with a schema that facilitates search. The SMEs will be able to search and thereby benefit from the outcome of the methodology after we capture all of the information and store it in the appropriate data structure/schema.

In the remaining sections of the chapter, we go through each step in detail. We additionally report on our efforts to teach the methodology to students in three classroom settings and the application of the methodology for various case studies.

4.1 STEP 1: Identify User Roles, Corresponding Goals and, if Available, Tasks to Achieve Them

Our methodology begins with the process of UX research [**?**]. A UX researcher conducts the User eXperience research process and is tasked with studying users, their habits, and behaviors, as well as forming design-informing models using various standard practices and procedures. These include contextual inquiry and analysis, persona building, and usability testing. Everything begins with understanding the user, their role, and goals. The design process does not end after this. A UX researcher might then work with a UX designer, a User Interface (UI) designer, a product designer, and the development team, as a part of an exhaustive, but elegant, process that ensures sound product design. In our case, the "product" that we are building is a set of workflows and workflow-related information that we forward to the workflow engine. The designers, mentioned above, come into play primarily when building an interactive system that allows a user to interact with an interface. In an interactive system, the design of the user interface plays a significant role in designing and developing every other part of the system. For our research, we do not focus on how we present the set of workflows or how the SME might interact with an interface that would

trigger them.

The UX research process is conducted to specifically answer the following questions.

1. Who is the user who will be using the system?
2. What are the goals they are looking to accomplish via the system?
3. How do they currently accomplish those goals?

With the user-related information, we can construct what is known as a "user persona." In a persona, we can list the user's role. The user role we focus on for all of our users is that of an "SME in their area of research."[2] We can further enrich the details of the persona by identifying research interests, datasets they frequently work with, and other information related to their research. Defining a persona gives the implementation team a point of reference when building a solution.

While it might not be essential to capture a rich persona for the sake of building the workflow, capturing the research goals (or problems), on the other hand, is very important. In some cases, an SME might even provide some restrictions and require us to follow a set of steps to achieve the goal.

We can map the persona-related information to the Society aspect of our description. A "state," as defined in 5S, can be used to represent goals. These states are then later connected based on the developer-built services that move them between states. We describe these in detail in Chapter 6.

Once we have identified specific research goals that are of interest to the different user personas, we need to come up with a plan to solve them. We address this in Step 2 of the

[2] The methodology can work with a variety of other types of users too, but "SME" will be used to represent all such. Other types of users of particular interest in digital libraries include: curator/collection manager, cataloger, system manager, system operator, harvester, explorer, anonymous guest, and data scientist.

methodology.

4.2 STEP 2: Break Down Research Goals Into Tasks and Tasks Into Sub-Tasks

In this step, we break the goal down into units knows as tasks. Tasks can be further broken down into sub-tasks. As defined by Hartson and Pyla in their UX book, a task or a sub-task refers to things that the user does in order to achieve their goal [31]. A sub-task can be thought of as a task by itself. We can produce this breakdown through the expertise of engineers and developers who are familiar with solving such goals and tasks. The UX researchers represent the SME in their discussions with the developers. The goals extracted through Step 1 of the methodology are described, by the UX researcher, in a manner that can be understood by developers and engineers. As a result, a developer and engineer familiar with solving a goal or task, can break down the goal into tasks and sub-tasks. In some cases, the identified goal needn't be broken down further. For example, if one of the goals of an SME is to collect tweets about an event, and they provide the UX researcher with an immediately usable search query, the developer may decide that this goal can be solved without breaking it down into smaller units. In that case, we treat the goal as a task and move on to the next step of the process. So, when and to what extent do we break down the goals?

When the developer identifies that the goal requires multiple steps to solve, they break down the goal into a set of tasks. The primary purpose of identifying the tasks is to guide the overall design such that the final design of the workflow should support the task and sub-tasks identified.

A hierarchical relationship is formed between goals, tasks, and sub-tasks. The meaning of the latter relationship is this: doing a sub-task is a part of doing a task.

Prior to this step, we were provided with SME goals that might not have been described to an extent such that we could facilitate support. The process of breaking down goals into tasks is not straightforward. Through the expertise of the developers, we were able to process the goals and produce a set of "solve-able" units (or tasks). We now have to find a way to solve these user tasks. We do that in Step 3 of our methodology.

4.3 STEP 3: Identify Functions To Solve Each (Sub-) Task and Model Workflow

At this stage, we have identified all aspects of the workflow that need support from functions. It is in this step that we decide what set of functions we want to employ to complete each (sub-)task. We again borrow the definition of a function from [31], as a thing that the system does. This step requires the most active collaboration between the UX researchers and the developers. To design a solution for a task, the developer has many options of functions or algorithms from which they could choose.

The UX researcher interacts with the developer to exchange user expectations on output and the available information on inputs. One of two things happens:

1. The SME has briefed the UX researcher on their requirements, and the information is in the UX-er's possession. In this case, this information is passed on to the developer, and the developer makes a decision accordingly.

2. Alternatively, the UX researcher gets limited information due to the unfamiliarity of

the SME with the various stages of implementation.

Based on the discussion, the developer might have to include an Input/Output service as a part of the workflow to solicit input from the SME regarding the implementation. During this period, the developer also identifies other implementation-specific parameters. Finally, the developer designs a set of services, each of which encapsulates an algorithm or function that is requisite for the specific task to be complete.

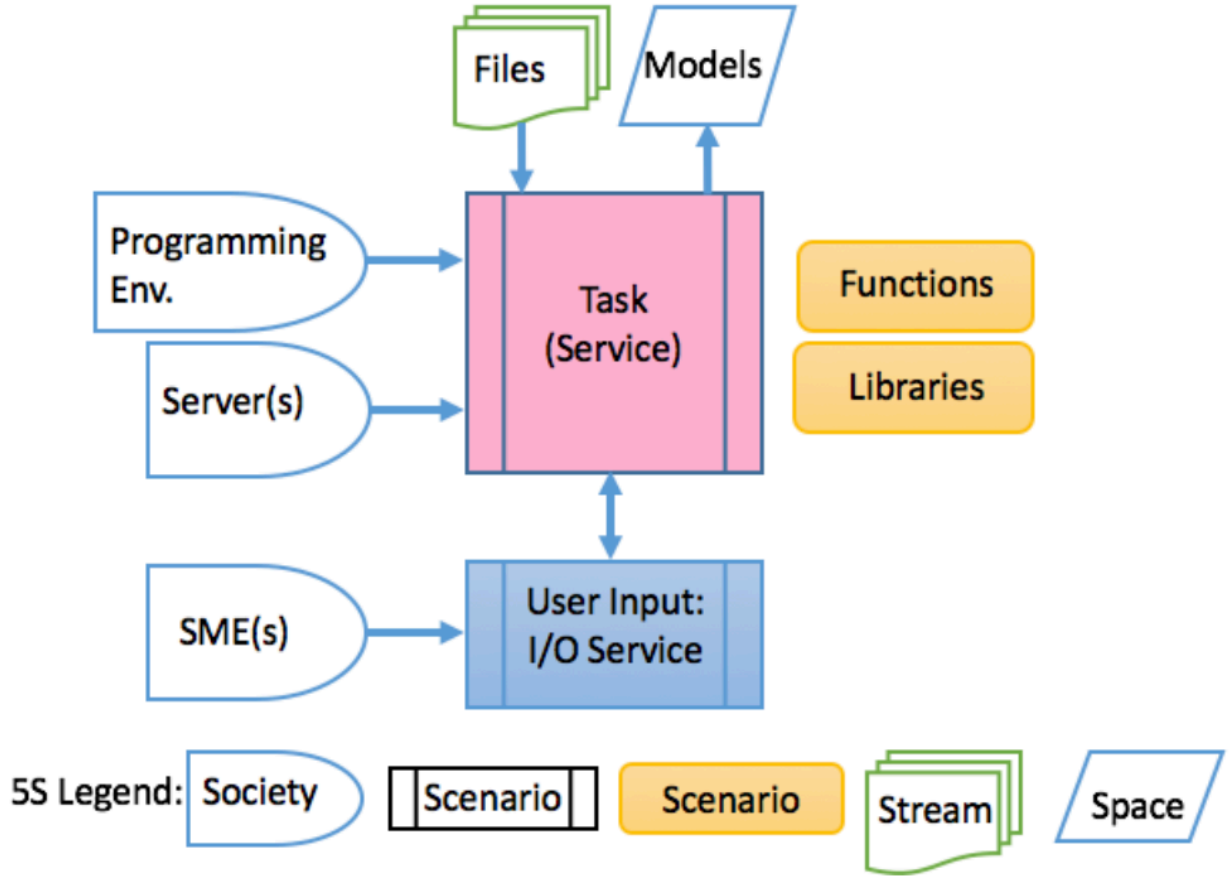


Figure 4.2: Example template showing the implementation specific information that is captured in Step 3 of the methodology.

In the end, the developer combines all of the services to form a workflow based on what they have learned, and on the representation of the structure of the decomposition of the goal into tasks and sub-tasks. We can derive the workflow sequence by doing a topological ordering

of the graph structure. Through this step of the methodology, workflows are specified that represent the combined execution of all of the sub-tasks and tasks that make up the goal.

4.4 Summary

We have (1) a process to take the user goals and produce a solution that is built specifically for them, and (2) a unified representation of SME goals, workflows that support those goals, and all the mappings in between. Representing user goals and workflows in this manner removes the knowledge barrier and terminology mismatch generally experienced when using standard workflow management systems. The workflows can address specific SME information goals. From the SME's point-of-view, they only need to request the desired information, and the information system implementation should generate a workflow. This generated workflow, when executed, will produce the information that they requested.[3] The information supported by the information system workflows is dependent on the stakeholder/research community and its problems/needs. As SME needs evolve over time, new services/workflows can be designed and incorporated, making the information system extensible.[4]

4.5 Teaching of Methodology

We have been able to apply the methodology in the context of a classroom. The opportunity allowed us to teach the methodology to students / student teams so that they can apply it to their process of building system solutions.

[3]The user interface and execution of the workflow by the workflow engine are out of the scope of this document.

[4]This can be done by making incremental changes to the unified representation, rather than building something new and independent of prior work.

4.5.1 CS4624: Multimedia, Hypertext, and Information Access

CS4624 is a capstone Virginia Tech Computer Science class where undergraduate students work on a term-long project defined and managed by a client. The clients represent SMEs from different fields with problems ranging from app-building to data wrangling, to data analysis and visualization. This class provided us with another opportunity to showcase how the methodology can be applied to build information systems for various contexts. More importantly, it allowed us to see if someone (or in this case, a team of students) other than myself could employ the methodology effectively. It was an opportunity for us to see if we could teach the methodology.

We introduced the methodology for two iterations of the class: CS4624 Spring 2020 and CS4624 Spring 2021. We used the Spring 2020 iteration of the class as a pilot study. Our experiences from the Spring 2020 iteration impacted how we administered the methodology for Spring 2021.

CS4624 Spring 2020: Pilot Study

For the Spring 2020 iteration of the class, we introduced the methodology part way into the semester. At this stage, the teams were already building parts of their solution. We aimed for the teams to use the methodology to help them in their development process. We taught the methodology through a presentation during a class session. Following the presentation, each of the seventeen teams had to complete and submit an assignment. In the assignment, they had to implement and produce artifacts from each step of the methodology. They identified their project goals, broke them down to tasks and sub-tasks, and identified the services they would need to build for their information system solutions. The student teams completed a set of surveys. Through the surveys, we learned that many of the students

understood the methodology. Through this class and this exercise, we observed how the student teams applied the methodology. We additionally learned that it would be beneficial if we had introduced the methodology sooner in the course.

CS4624 Spring 2021: Study

Accordingly, in the Spring 2021 offering of CS4624 we introduced the methodology at the end of week 2 of the course, right after the students' selections of term projects had completed. As in the case of the previous iteration of the class, the student teams in Spring 2021 CS4624 were each building solutions that varied in scope and theme. Therefore, our motivation to teach the students the methodology was so that they could use the steps of the methodology to experience a "lite" form of user empathy and organize their development efforts. We wanted the student teams to focus on executing Steps 1 and 2 of the methodology. Due to some teams building mobile apps or databases or conducting data mining, we understood that they would find it hard to describe services (Step 3 of the methodology) involved in their development. Eighteen teams instrumented steps of the methodology to learn about the SME/user requirements and break down the user goals they identified into tasks and sub-tasks. At the end of the class, we asked the students to evaluate the methodology for its usability, learn-ability, and impact in their development effort.

Evaluation by student teams

We constructed our evaluation to assess the following aspects of the methodology:

1. Is the methodology easy to use?

2. Is it easy to learn?

To learn (1) and (2), we employed an adapted version of the System Usability Scale (SUS) that was created to learn about user experiences when using a system [6].

Question
1. Please briefly explain the methodology as best as you can.
2. What did you learn from the methodology?
3. How did you use the methodology in making decisions about your team's approach? How did it influence your solution?
4. If you did not use the methodology, how different would your approach and/or solution have been?
5. Write one thing you would change to improve the methodology.
6. If you could implement this methodology over, what would you do differently?

Table 4.1: Qualitative Questions on Methodology

We additionally asked the students to rate the overall solution's quality, and through qualitative, open-ended questions, assess the degree to which the methodology contributed to their design decisions. Table 4.1 lists the sets of qualitative questions of the survey that we administered at the end of the class.

Figures 4.3, 4.4, 4.5, and 4.6 provide a breakdown of the responses to selected survey questions. As shown in these figures, the student teams gave us favorable feedback regarding their experience using the methodology. For future iterations of the class, we aim to get higher ratings from the students.

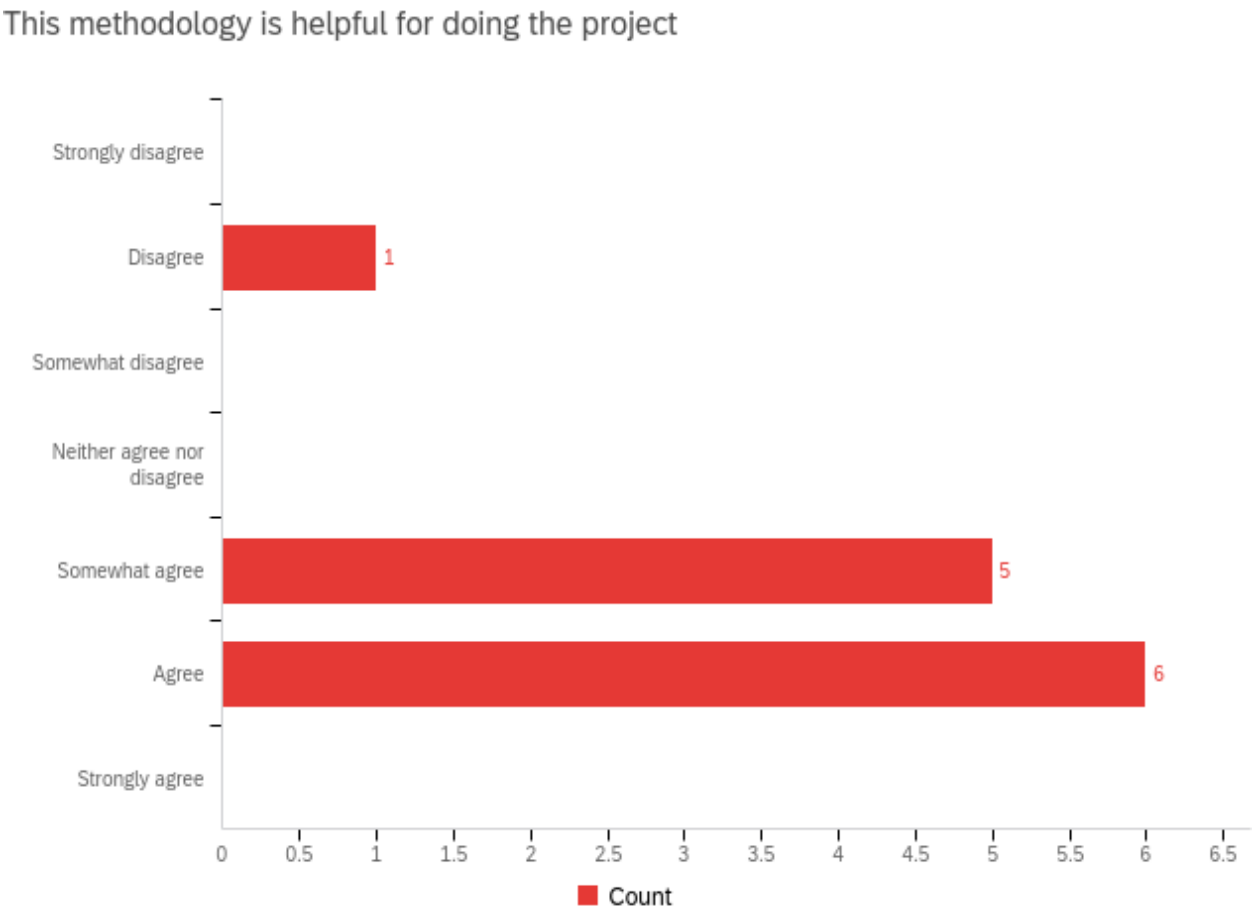


Figure 4.3: Responses from participants for the question: The methodology was helpful for doing the project.

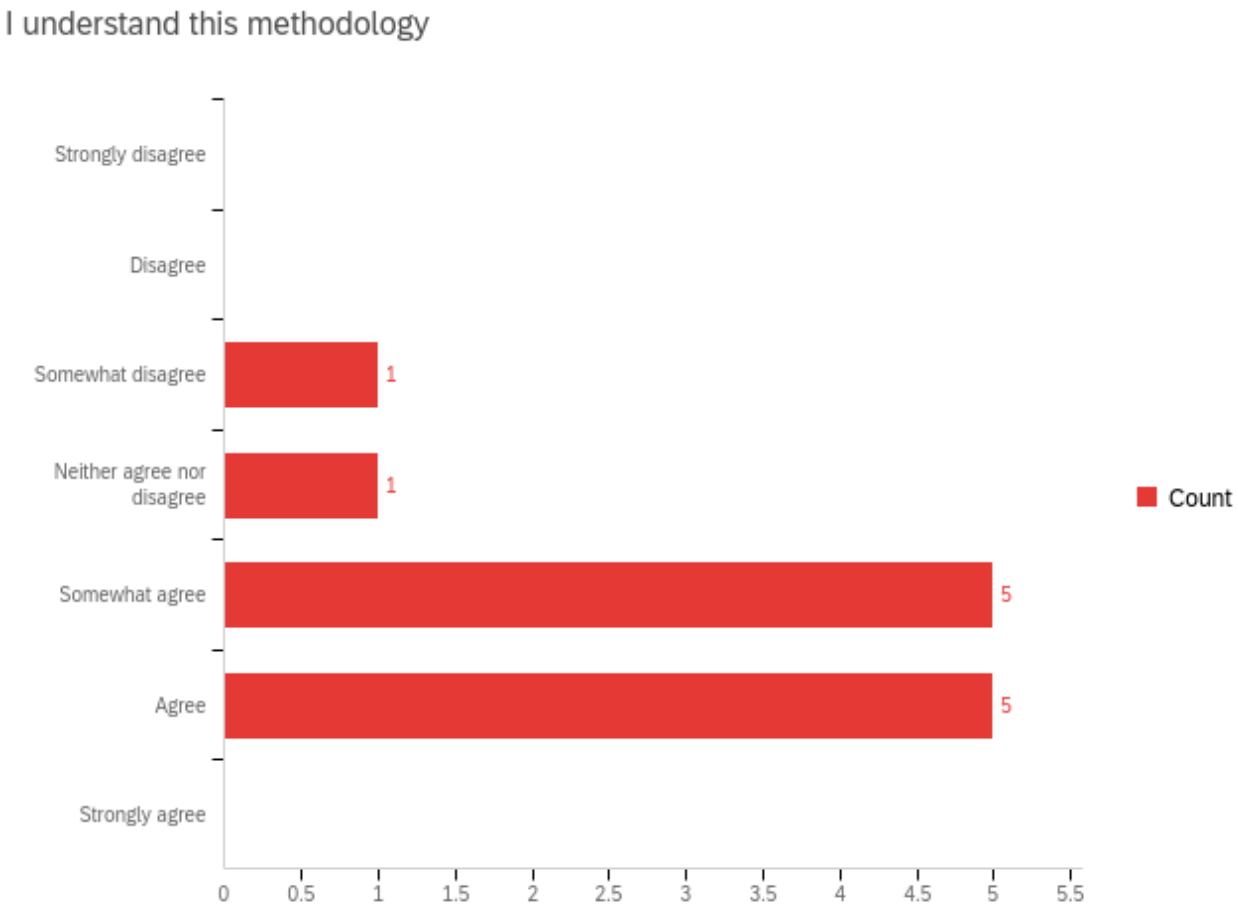


Figure 4.4: Responses from participants for the question: I understand this methodology.

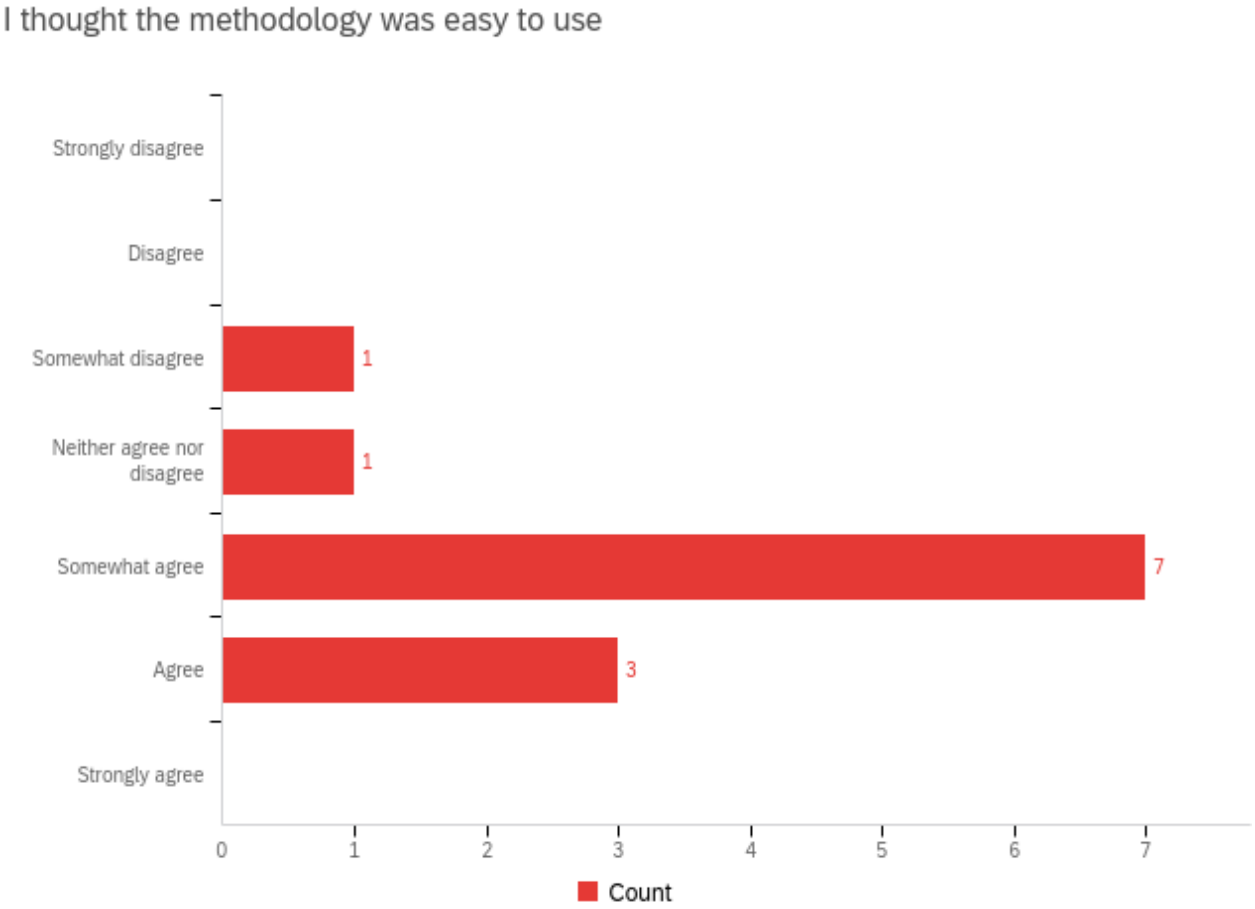


Figure 4.5: Responses from participants for the question: I thought the methodology was easy to use.

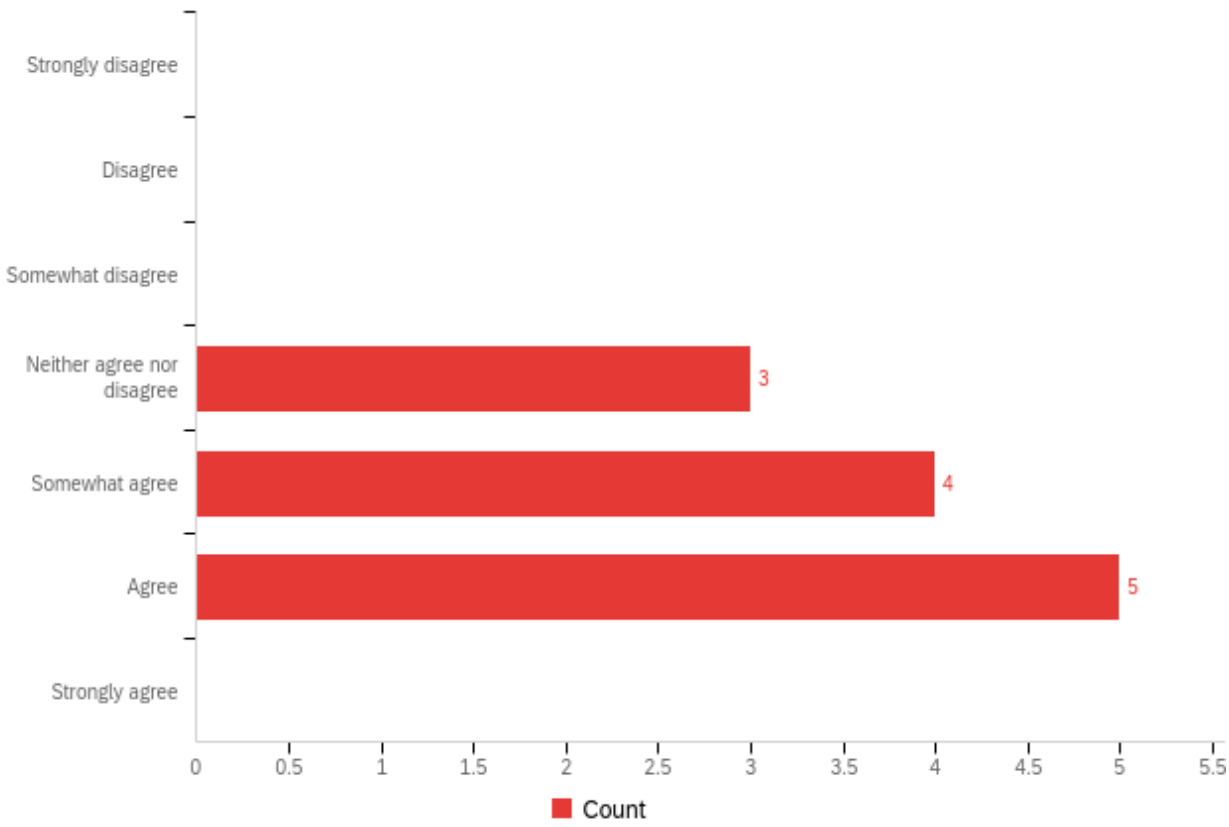


Figure 4.6: Responses from participants for the question: If it turns out that I often am building systems or doing projects like for this course, then I think that I would like to use this methodology frequently.

4.5.2 CS5604: Information Storage and Retrieval

Like CS4624, CS5604 is a problem-based learning course. As described in the course catalog, students in the course are tasked with "analyzing, indexing, representing, storing, searching, retrieving, processing and presenting information and documents using fully automatic systems." They are to collectively build a state-of-the-art information storage, retrieval, and analysis system that supports searching, browsing, and analyses of three types of large collections of documents: (1) tweet (TWT) collections, (2) web page (WP) corpora, and (3) the local collection of electronic theses and dissertations (ETDs). The information needs of the subject-matter-experts (SMEs) would determine the range of search queries supported.

Setup

Figure 4.7 provides an overview of the components of the prospective system.

The figure also shows five different sets of team efforts that fit together to build the components of the system. These include three types of "content'' teams, one front-end interface team, and one integration team. It is the responsibility of each of the "content'' teams to communicate with the SME who works with their type of content. The members of each content team execute the steps of the methodology. Through that process they identify SME information goals, expectations regarding output formats, and the formats of available inputs. They forward this information to the front-end and integration teams. They translate this information into design and development requirements. We shared this figure with the class during the second week of the course.

The course was set up like a flipped classroom. The problem structure and the solution approach encouraged intra- and inter-team collaboration and learning.

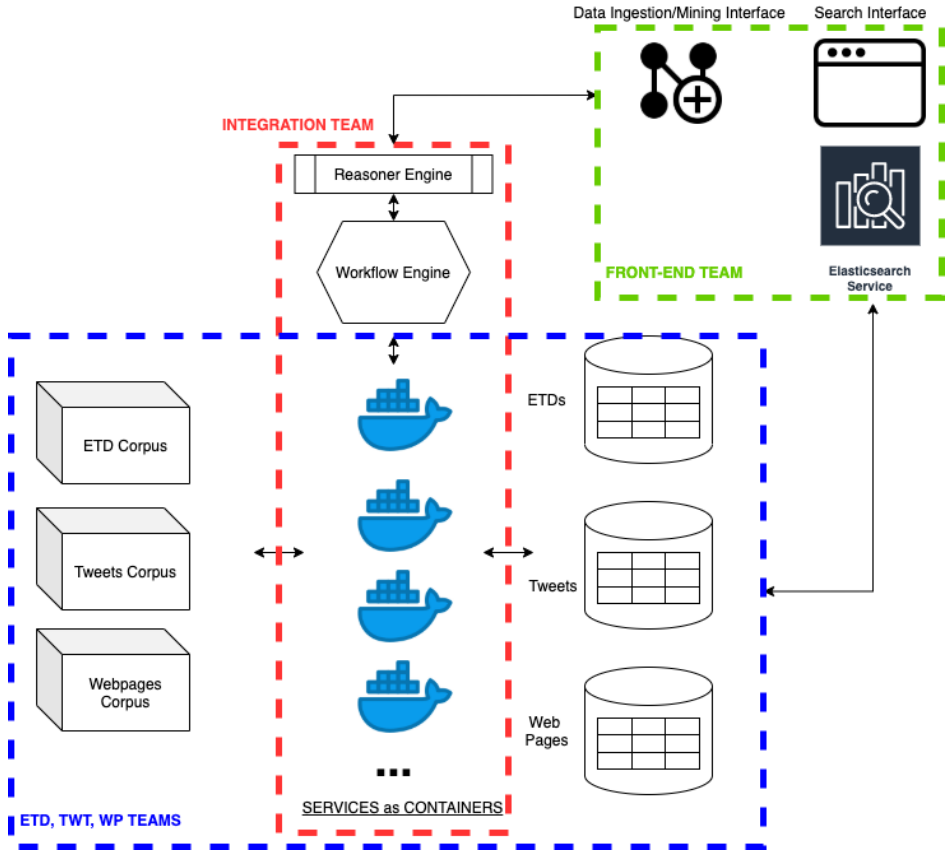


Figure 4.7: Class IR System Architecture

Execution of Methodology

Each of the three teams working on their respective content played the role of UX researchers. They then interviewed their respective SMEs (i.e., one SME for each team). As mentioned previously, the content teams shared this information with the front-end team. The front-end team was responsible for building what would support the user experience. So they needed to be aware of the SME preferences and expectations on output. Afterward, the content teams played the role of the developers. They broke down the user goals into tasks and sub-tasks. For the final part of the methodology (identifying services), the content teams collaborated with the integration team. The integration team was responsible for building the infrastructure for all of the development efforts to be deployed as services. The integration team also built a CFG-based reasoner and a service registry to support SME exploration. They communicated with the front-end team and built APIs that provided a response to SME queries from the UI. Appendix A provides a more detailed description of the components of the final workflow-based information system.

Outcome

The three teams working on their respective content identified 17 (five from the ETD SME, four from the WP SME, and eight from the TWT SME) information goals. They developed a total of 22 (eight from the ETD team, four from the WP team, and ten from the TWT SME team) services, which they containerized and deployed in collaboration with the integration team. The integration team registered the services and connected them based on the relationship between information goals. The front-end team built an interface for users for two types of functions: (1) browse and search through the collections, and (2) request an information goal. Requests were routed and served and the reasoner determined the se-

quence of services to execute to satisfy the user requirements. The reasoner generated the workflow and represented it via an executable document. This document was then passed on to the workflow manager to execute. The response was sent back to the SME via the interface that the front-end team built.

Evaluation by student teams

As was the case with the CS4624 case studies, we constructed our evaluation to assess three aspects of the methodology:

1. Is the methodology easy to use?

2. Is it easy to learn?

3. Does it help produce a good solution?

To learn (1) and (2), we employed an adapted version of the System Usability Scale (SUS) [6], same as before. For (3), we asked the students to rate the overall solution's quality, and through qualitative, open-ended questions, assess the degree to which the methodology contributed to their design decisions.

As for the CS4624 study, Table 4.1 lists the sets of questions of the survey that we administered at the end of the class.

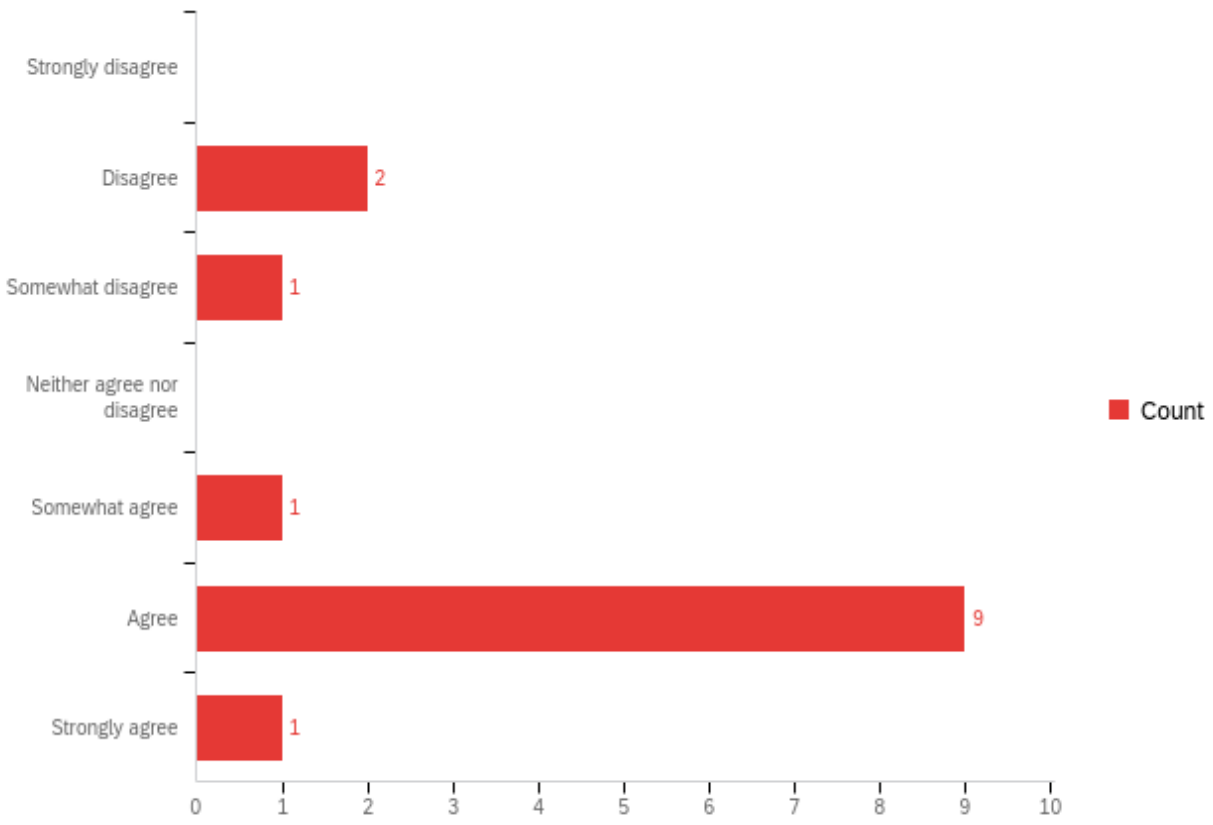


Figure 4.8: Responses from participants for the question: I thought the methodology was easy to use.

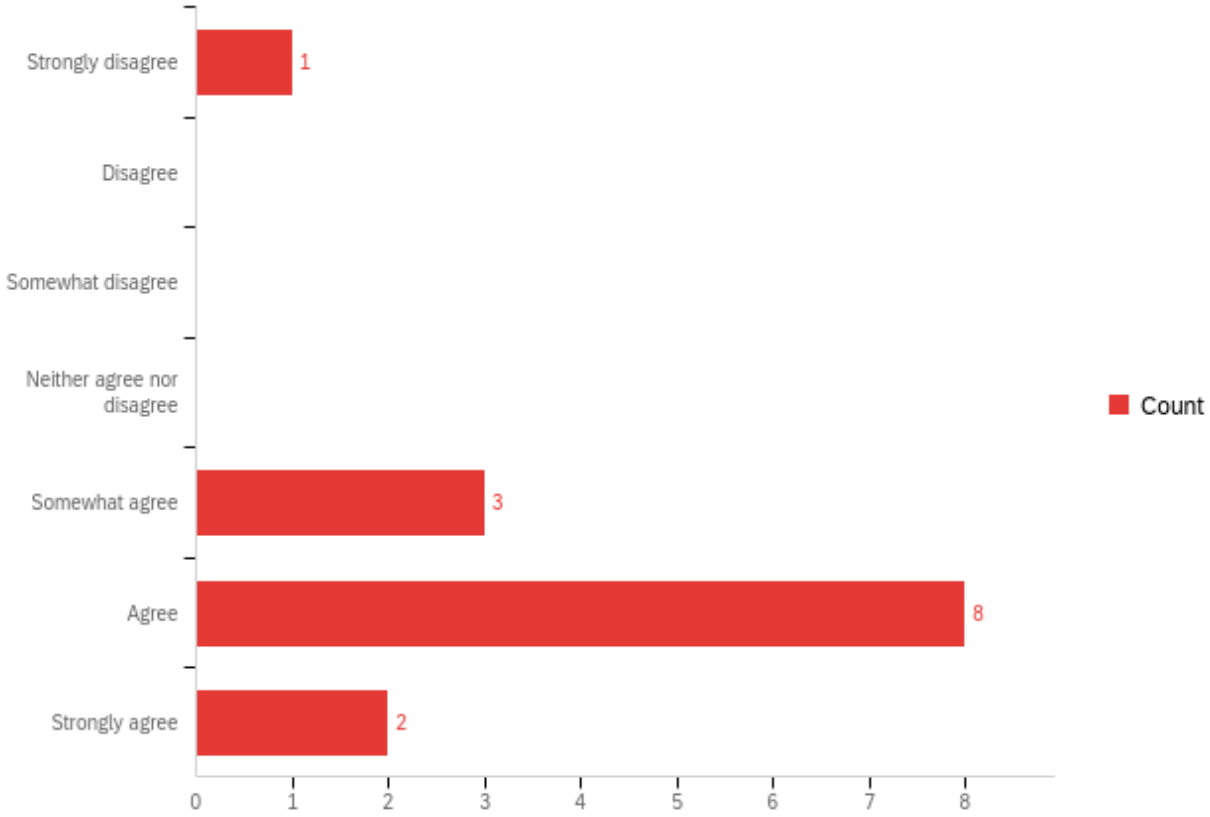


Figure 4.9: Responses from participants for the question: I understand this methodology.

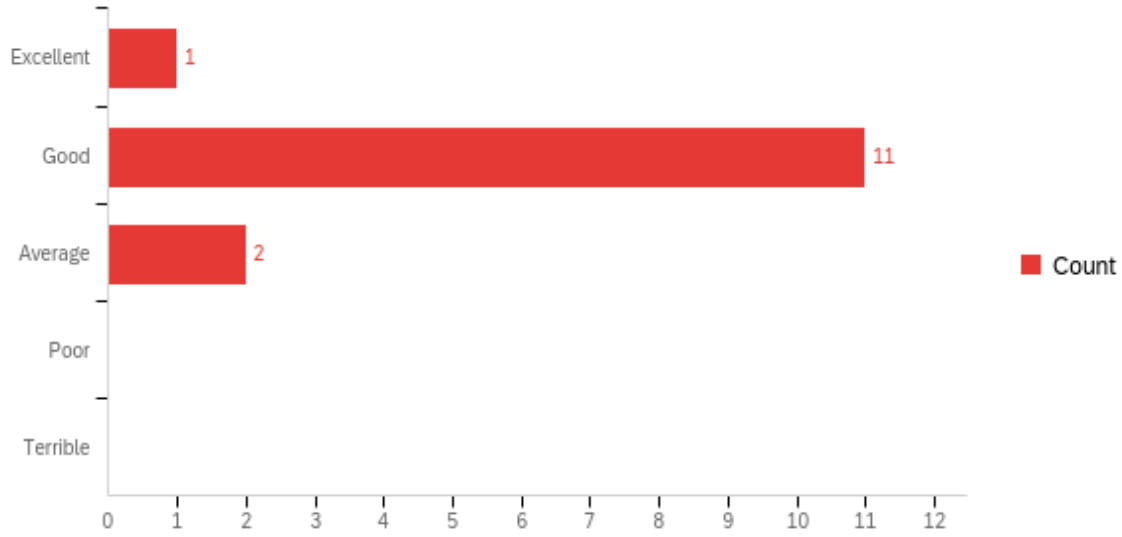


Figure 4.10: Responses from participants for the question: How would you rate the quality of the overall system solution?

Discussion

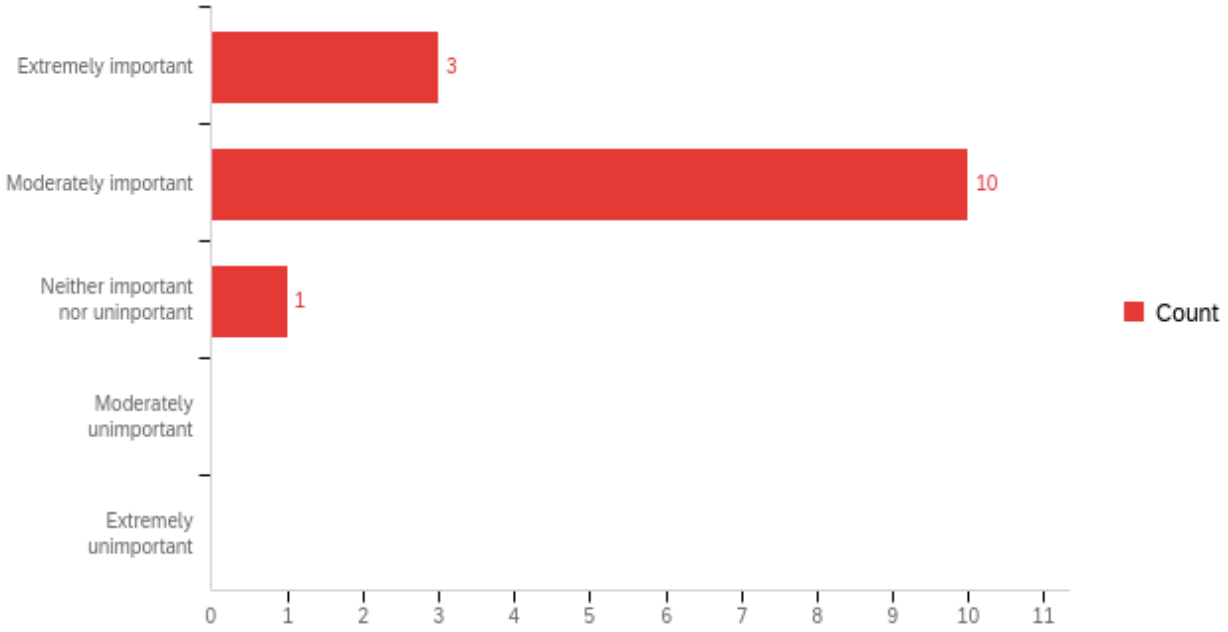


Figure 4.11: Responses from participants for the question: How important was the methodology in influencing your project solution?

Figures 4.8, 4.9, 4.10, and 4.11 provide a breakdown of the responses to selected survey questions. The feedback shows a favourable response from the students about the methodology that we employed as an engine for them to learn about building modern information retrieval systems.

Table 4.2 showcases a subset of textual responses to the question "How did you use the methodology in making decisions about your team's approach? How did it influence your solution?" The textual responses convey that the methodology significantly contributed to each team's design and implementation. Given that, and looking at the student's self reporting of the quality of the solution, we can conclude that the students believed that through the methodology, they were able to build a good quality solution. More specifically, it conveys that the methodology positively impacted their decision making process.

Textual Feedback
1. "The methodology helped a lot when we were creating our services. We developed services in such a way that it could be integrated into the system and work along with services that were developed by other teams. We made sure that the output file for each of our services would be in a format that would support airflow."
2. "It influenced our solution by providing an ultimate goal and leaving the details to the developer which encouraged creativity and research."
3. "The methodology was most useful in the beginning for finding out the scope of the project and defining separation of tasks. Creating the table of goals and tasks influenced how we separated our workflow, but I think this that should be communicated better is how our goals relate to the infrastructure. I think the main place where this approach breaks downs was the teams' understanding of the infrastructure. Since the infrastructure tools were being learned/setup as we were building, it was difficult to get a clear sense of how all the pieces would finally fit together. I think this will be less of a problem next semester if you continue using Airflow, or switch to a similar workflow manager."
4. "It very heavily influenced our initial design and how we designed our services (small, compact, do exactly 1 task)."
5. "We used the methodology to keep everyone on track with end points so the data is in the correct format for each service. The major influence it had was the integration of each service was well defined and everyone had to curate their services to deliver for the next service."
6. "Using the methodology influenced the design of services, thus division of work. As we understood the architecture more, we fleshed out the workflow artifacts and adjusted to microservices."

Table 4.2: Qualitative Feedback on Methodology

Evaluation by SME

We also needed to evaluate the SME's perspective on the solution.

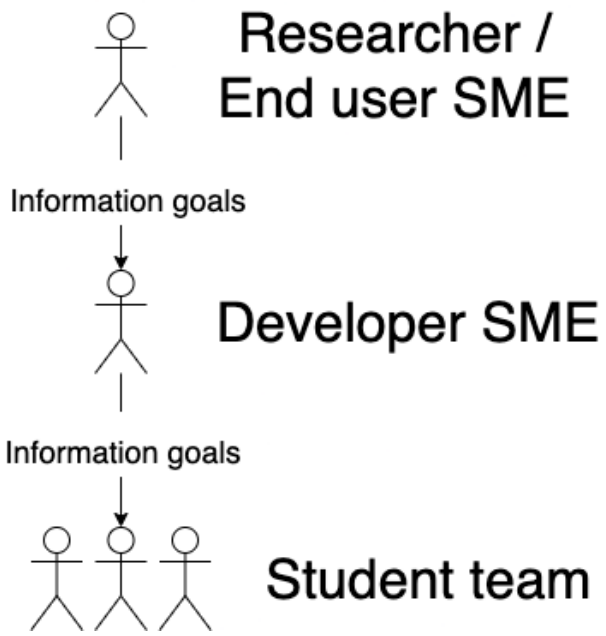


Figure 4.12: Two types of SMEs for the ETD part of the CS5604 class project. The goals for the student teams were from the developer SME. The developer SME goals where based on the researcher SME goals.

Figure 4.12 shows two types of SMEs for the ETD-specific part of the class project. We interviewed both the developer SME and the researcher SME to gather their information goals. The student team working on ETDs built workflows for the goals identified by the developer SME. It is important to note that the information goals for the student team, provided by the developer SME, were based on what the developer wanted to build. The developer SME goals were dependent on the information goals of the researcher SME.

The breakdown of the developer SME goals to task and then services, built by the student team.

We interviewed the two types of SMEs to understand the following:

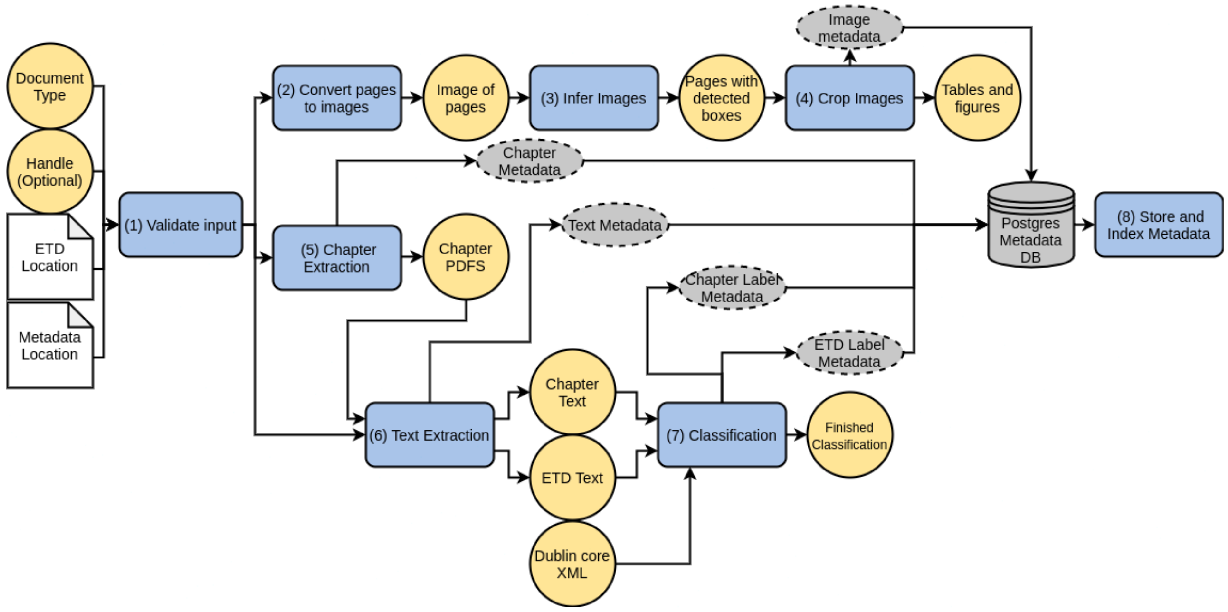


Figure 4.13: Data processing pipeline constructed by the ETD team in the CS5604 course. The rectangles with rounded edges represent the workflow services that the team successfully deployed on the graph-based service registry. [15]

1. From developer SME: Validation and Influence - Did the workflows built by the student team work as expected, and were they helpful for their (developer SME) development effort?

2. From researcher/end user SME: Influence: Does the existence of the workflow and services make it possible for a class of SMEs to use the system? Without these services, how does the way you address the goals change?

The following subsections give answers to these questions.

Developer SME

I provided the developer SME with the figure of the set of workflows that the student team built (Figure 4.13). I asked them to identify the workflows that work and those that don't.

Question
1. Best entry point chapter for reading an ETD, based on a concept.
2. Literature review on a topic
3. Enable experimental analysis of the corpus of ETDs
4. Determine where an ETD was downloaded from
5. Return a set of ETDs containing keywords in generated chapter summaries
6. Search/Sort/Group ETDs by descriptive metadata elements (e.g., return a set of ETDs from the same university)

Table 4.3: Researcher/End user Goals

All but one workflow worked as per their requirement. Next, I asked them how they would use the workflow for their work.

Figures 4.14 and 4.15 shows a set of workflows that the SME is developing (or could potentially develop in the future) that uses the information produced from workflows built by the student team. The design of the student team workflows informed the development efforts of the developer SME.

Researcher/End user SME

Table 4.3 lists the information goals for a class of researchers or end users who want to work with ETDs. I provided the SME with a list of workflows. I shared the input for each workflow and the output produced. We went through every information goal listed in the table. The researcher SME shared if the current set of workflows could extract the information goal. When applicable, they shared what workflows were missing. From my discussion with the researcher SME, we learned that except for having a user interface that would organize the information, all but one of the information goals could be satisfied with the combination of workflows built by the student team and the workflows that the developer SME would build on top of them. For goal 3, the researcher required a Jupyter notebook-style interface, for

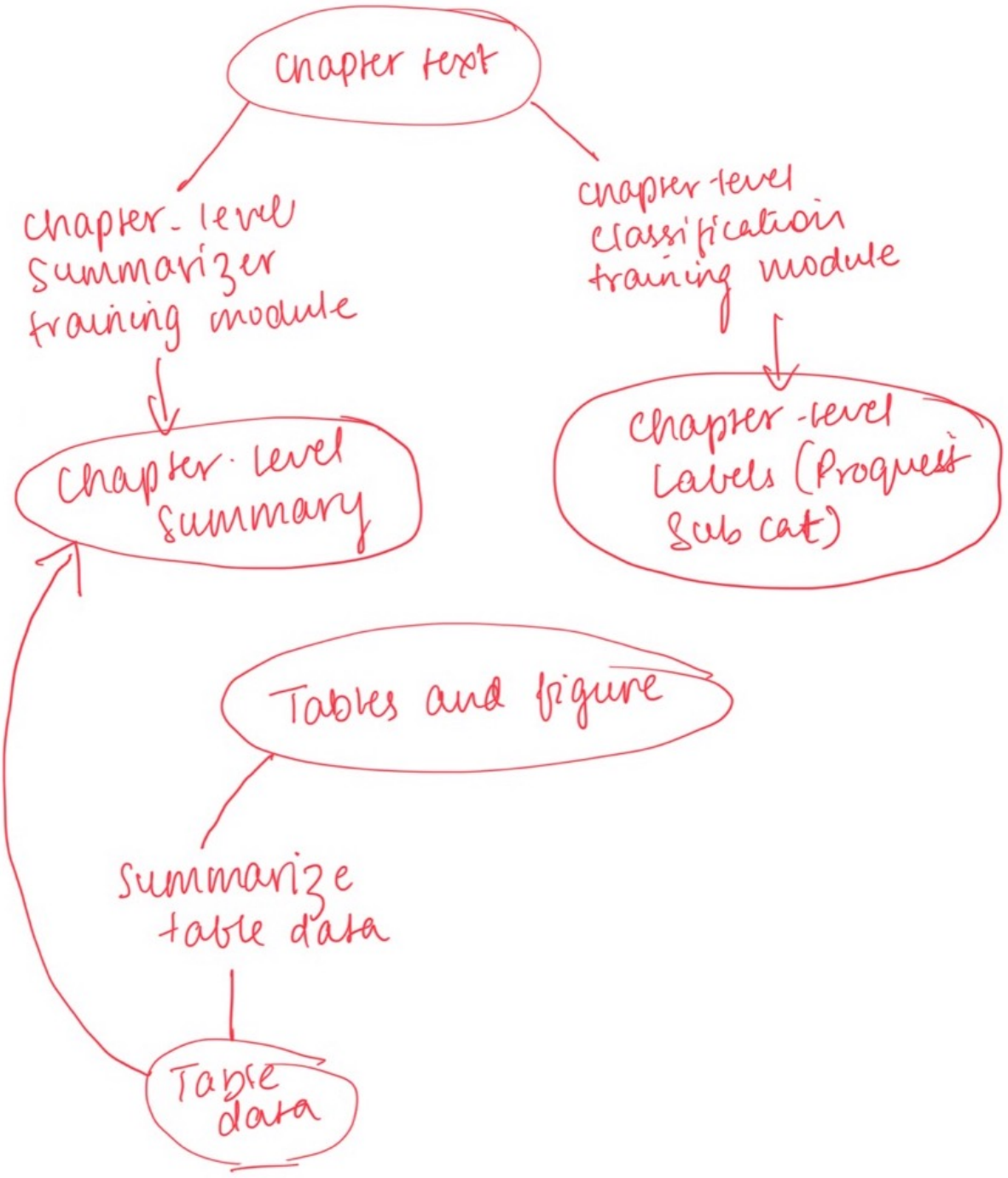


Figure 4.14: A set of workflows that the Developer SME is working on (or planning to work on) that uses the information produced ("Chapter text") by the workflows created by the student teams. This particular set of workflows generate chapter-level summaries and chapter labels, as well as extract data from tables and figures.

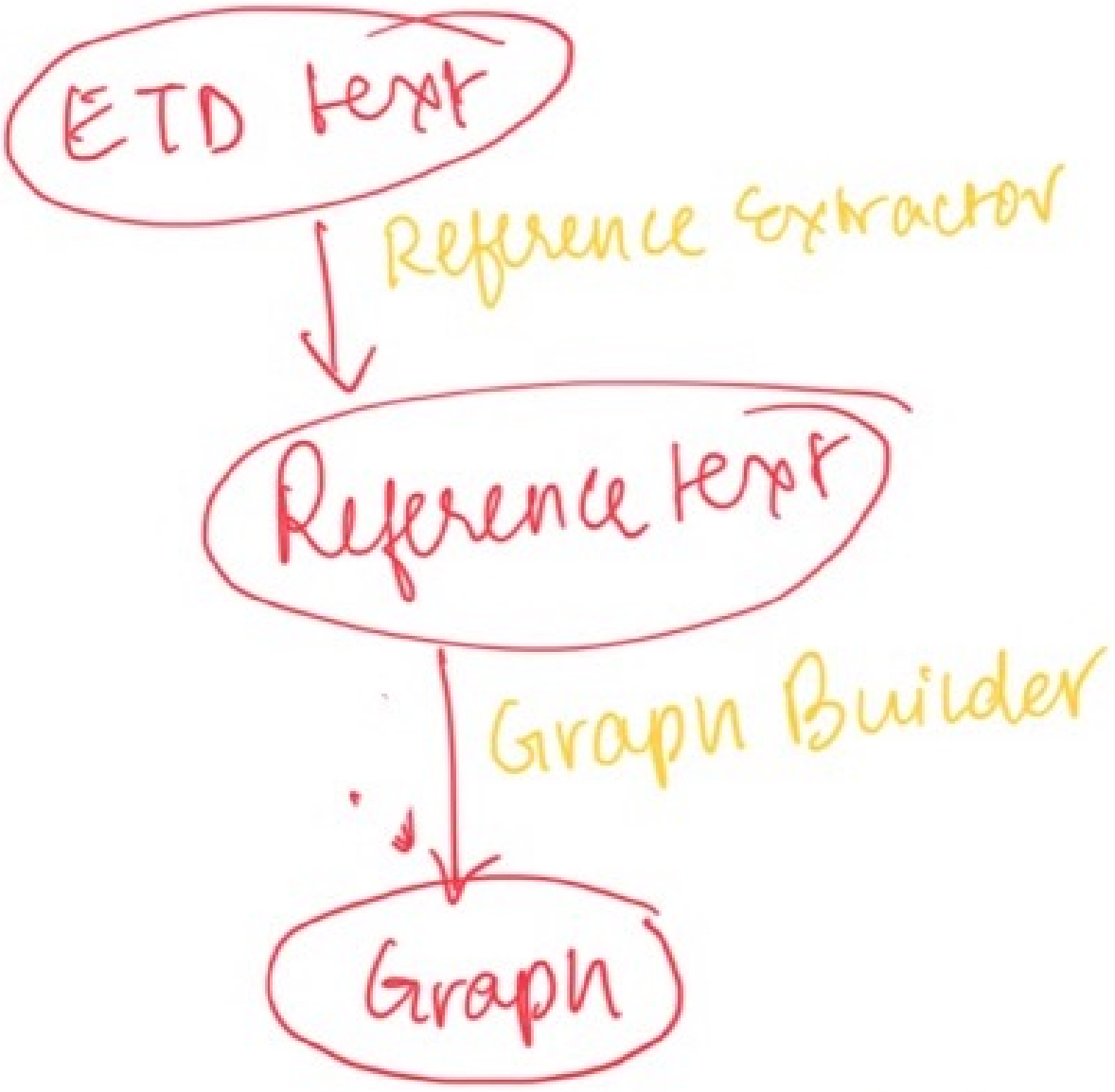


Figure 4.15: Another workflow that the developer SME identified as a future work that would generate a citation graph based on the References section of an ETD.

which a workflow service did not exist.

In conclusion, the researcher SME noted that for some of these goals, without the workflows, it would be close to impossible or highly tedious to extract information from long documents such as ETDs.

4.6 Methodology Instrumented via Case Studies

One of the many functions of Virginia Tech's Digital Library Research Laboratory is to serve as data and information consultants for SMEs. Through various research efforts, we have provided to SMEs solutions such as dataset preparation, advanced analysis, and visualization. It is through our interaction, over the years, that we have noticed a pattern in SME needs and tasks that we have had to support. The patterns are regarding the type of data or the type of analyses that are of interest to SMEs. These experiences have inspired and motivated this research, so they serve as compelling examples to showcase the power of the research solution. Each case study description provides insight into the users' characteristics and information needs that the desired system should support.

4.6.1 Tourism Project

Led by principal investigator Dr. Florian Zach, an assistant professor in the Department of Hospitality and Tourism Management at Virginia Tech, this project focuses on studying state tourism websites. The dataset is monthly snapshots of all US state tourism websites (managed at the state level by its tourism office) from the beginning of their existence. As explained by Dr. Zach, "The website is a representation of how the state tourism office thought it could best attract visitors at the time of the snapshot. These websites essentially

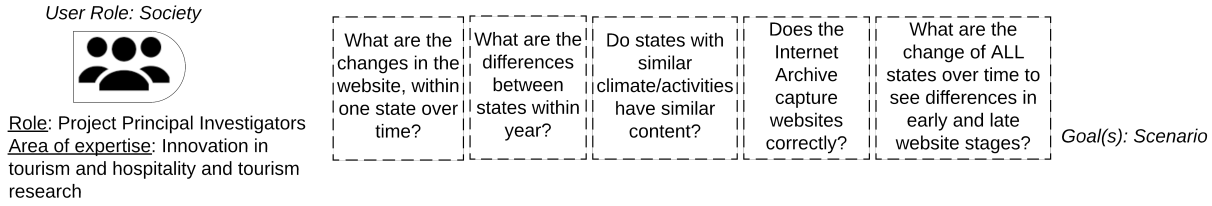


Figure 4.16: User roles and goals identified for the Tourism project

tell us how the tourism offices interpreted potential visitors' search and information consumption behavior." We interviewed Dr. Zach as a subject matter expert. Putting on the hat of the UX researcher, we were able to identify his roles and goals. This can be seen in Figure 4.16.

The student team for the Spring 2020 undergraduate capstone class, CS4624: Multimedia, Hypertext, and Information Access, took on the role of the analyst and developers to instrument Step 2 and Step 3 of the methodology [47]. An essential task of all the goals was data ingestion. Without the data being stored in a format that would allow for mining, none of the questions posed by the SME would be answerable. This effort was the team's focus. They were able to ingest and store the data in a workable format.

Current Status: While the student team was unable to directly address the goals identified in Step 1 of the methodology, they helped by making the data accessible. A team (USstateTourism) in the Spring 2021 class CS4624 has extended this work [58]. They also have followed the methodology, guided by explanatory materials I provided, as well as repeated discussions with them regarding how to proceed. Future teams will be able to make progress from this point.

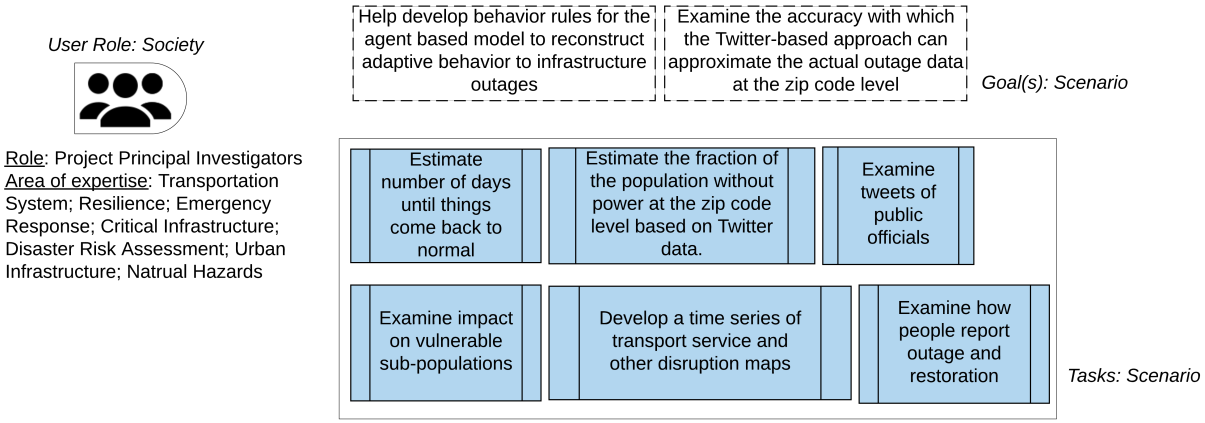


Figure 4.17: User roles, goals, and tasks identified in the CRISP-funded project

4.6.2 CBAR-tpd Project

The CRISP-funded project[5] is staffed by a group with research interests spanning from environmental policy, planning, and management; to transportation resilience and evacuation modeling; to critical infrastructure resilience and disaster risk assessment; to studying behaviors on Twitter during natural disasters. Each researcher has goals that relate to their domain. However, all of it converges to the goal (for the project as a whole) to obtain ALL parameters to accurately model and simulate the behavior of systems, families, individuals, EMT teams, etc. before, during, and after a (hurricane) disaster event.

One of the main contributions comes from looking at how essential services, vulnerable populations, and first responders behave.

Our first step was to understand specifically what the researchers or SMEs wanted to learn. After interviewing the SMEs, we identified their information goals; these are shown at the top of Figure 4.17. The SMEs for the CRISP-funded project additionally provided us with information on how they planned to solve the goals, which we identified as tasks.

[5]CRISP Type 2/Collaborative Research: Coordinated, Behaviorally-Aware Recovery for Transportation and Power Disruptions, CBAR-tpd, NSF CMMI-1638207, PI Dr. Murray-Tuite.

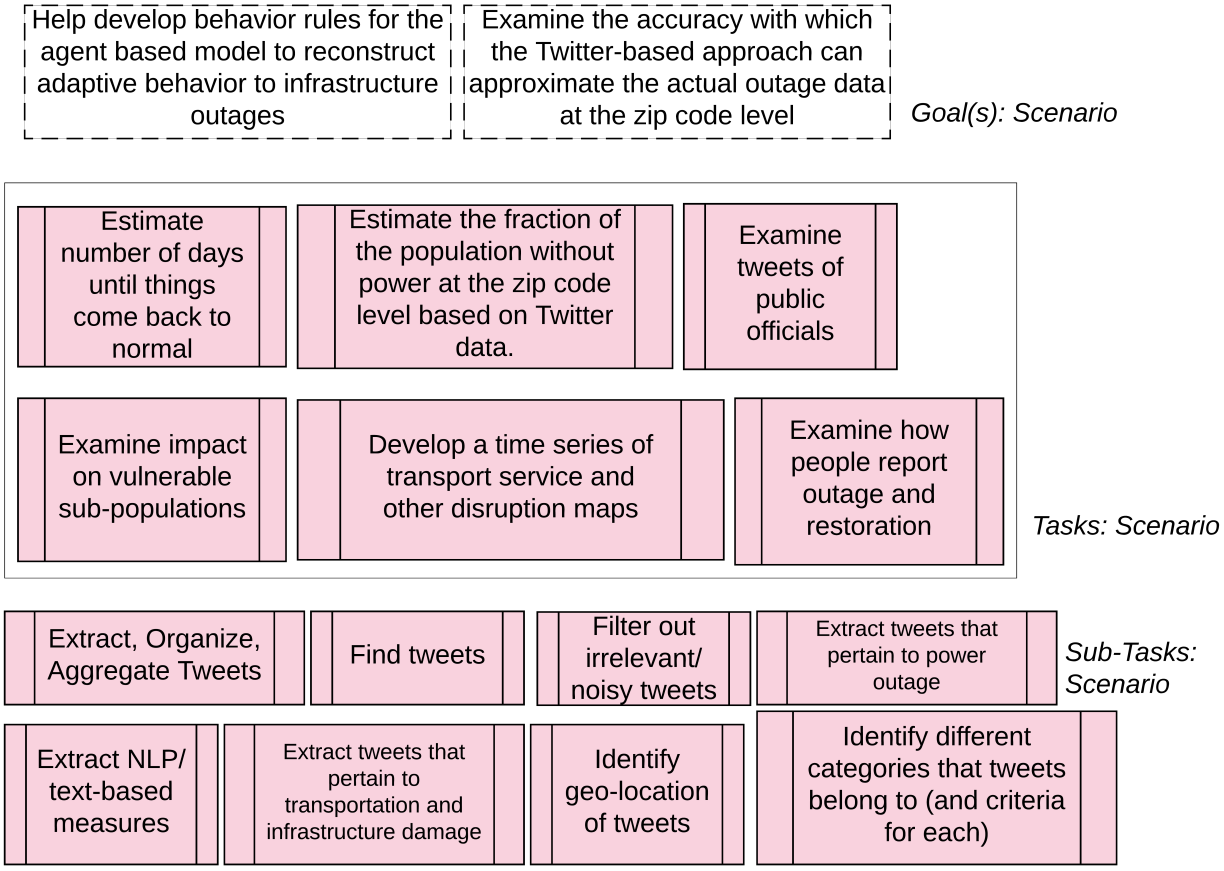


Figure 4.18: CRISP-funded project related Tasks and Sub-Tasks

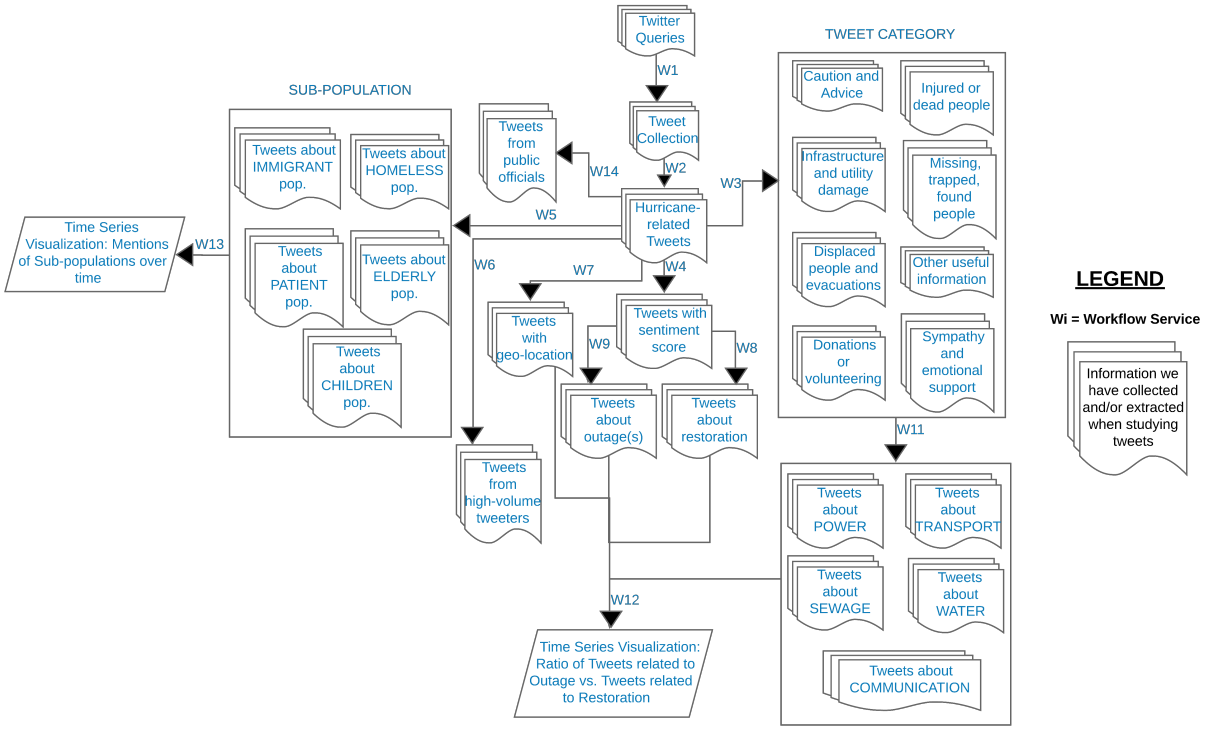


Figure 4.19: CRISP-funded project - Graph showing how the goals are broken down

The analysts and developers in the project were familiar with the mining of tweets related to natural disasters. They broke down every task into sub-tasks. Figure 4.18 showcases these sub-tasks. They were also able to capture the hierarchical relationship between the sub-tasks and tasks and goals. Figure 4.19 shows the associations.

Then, the team developers worked on devising programs, models, and visualizations as they constructed components to support the tasks and sub-tasks that they identified in Step 1 and Step 2 of the methodology.

Current Status: The developers have completed the building of components. As a part of the development effort, student teams in CS4624 built models and visualizations. Their efforts were presented in ISCRAM 2021 as a poster [48, 50]. For the SMEs to query their information goals we would need to create containers for these services and represent the information goals and services in the knowledge graph. This would represent the scope for future work by the developers of the project.

4.6.3 IMLS-funded ETD Project

The IMLS-funded project on Electronic Theses and Dissertations (ETDs) is an effort to provide various societies with the ability to mine and search through an extensive collection of ETDs. The funded period of this project is from 8/1/2019 through 7/31/2022. The final system will include an interface that will allow users to navigate through ETD documents that are typically long, and will retrieve precise answers [62]. The system will also provide an infrastructure of state-of-the-art models explicitly built to analyze ETDs and extract information such as ETD chapters, definitions, figures, tables, etc. For the CS5604 (Information Storage and Retrieval) course, the student team working with the ETD collection completed instrumenting the different steps of the methodology. They subsequently developed the

workflow services in a knowledge graph-based service registry.

They built a total of 8 services (shown in the rectangle in Figure 4.13 with rounded edges).

Current Status: Currently, the developers are developing workflow services to support newer goals. They will similarly register their newer ETD-mining-services into the service registry.

Chapter 5

Research - Part 2: Implementation

In this chapter, we describe our efforts to conceptualize two key components of workflow-based information systems. As a result of the methodology we have: (a) a set of information goals for each type of user, (b) a breakdown of goals into a structured sequence of tasks, and (c) a set of services that support each task.

In this section of the research description, we address Research Question 2: ***How do we implement such an information system, using knowledge graphs supported by a query language and capability?***

Moving forward from the previous chapter, how do we facilitate information discovery of what we have extracted in a manner that is accessible to SMEs?

More specifically, we want the system solution implementation to support the following types of SME queries:

1. Q1: Given an information goal as a query, what is the workflow to derive this information?

2. Q2: Given a data resource as input, what information can we derive using this data resource?

For Q1, our system solution should generate a workflow that would produce the desired result when executed by a workflow engine.

For Q2, our system solution should leverage the goal-task-workflow services relationships and produce a list detailing the information that can be derived[1].

5.1 Hypergraph-based Knowledge Graph

To be able to support the two types of information queries, we need:

1. to build a knowledge base to capture and store the artifacts/information,
2. a representation that organizes them in a manner that maintains the various associations (goals to tasks, tasks to services),
3. a processing component or an engine to compile these associations and deduce relationships between information goals as well as intermediate states of information, and
4. a system solution that should scale to more goals and more tasks.

We are describing the characteristics and components of a knowledge graph [14], and of a service and a system that leverage that knowledge graph.

The SMEs would view the knowledge graph where the nodes represent the end "state" of information they want to obtain. The relationships between the nodes represent events / operations. Thus, we define a workflow as a sequence of events / operations that changes the state of information. From the SME's point-of-view, all they need to do is select the node representing their interest. Under the hood, the *very same representation* enables generating a set of paths that represent data analysis-based workflows, which, when executed, deliver

[1]In the future, one could add a functionality to the system that would execute the workflows in the background and not only store what information one could derive, but additionally store the derived information (from the workflow execution). If the information were stored in a filesystem, we could maintain the mapping of the information state to the location in the filesystem for easy access.

the information requested by the SMEs. The representation of the knowledge eliminates the need for the SME to know the details of the operations. Additionally, the structure's flexibility allows us to represent data analysis workflows of any form, eliminating the need for an additional knowledge base and a middle layer to translate user queries to workflows. We can capture and represent both the SME information needs and workflows together. The scope of information supported by the graph is flexible and is dependent on the research community and problems the knowledge graph-based system is looking to support.

This research proposes to model the SME goals and workflows using a hypergraph-based knowledge graph [20].

Unlike other abstractions, a hypergraph allows us to specify n-ary relations or "hyperedges" within a graph. This property allows us to represent multiple data dependencies of a task with just one (hyper)edge. As a result, every edge incident to a node of interest to an SME represents a different path (or workflow) to achieve that state of information. It would be very cumbersome to capture this relationship using only traditional "binary" edges.

Figure 5.1 and Figure 5.2 are examples of two hypergraph representations of information goals and their relationship to one another via services.

Let us consider a toy example as shown in Figure 5.1. This is a graph representation of information goals / states of information (shown as alphabets) connected to one another by services (shown as numbers). All of this is identified from the methodology. From this representation, we can observe that there are three different workflows to derive information goal "a". All three workflows can derive that information goal. Based on the input provided, a workflow is selected.

The structure for the hypergraph shown in Figure 5.2 was similarly identified through steps of the methodology. We describe this particular case study in detail in Section 4.6.2. The

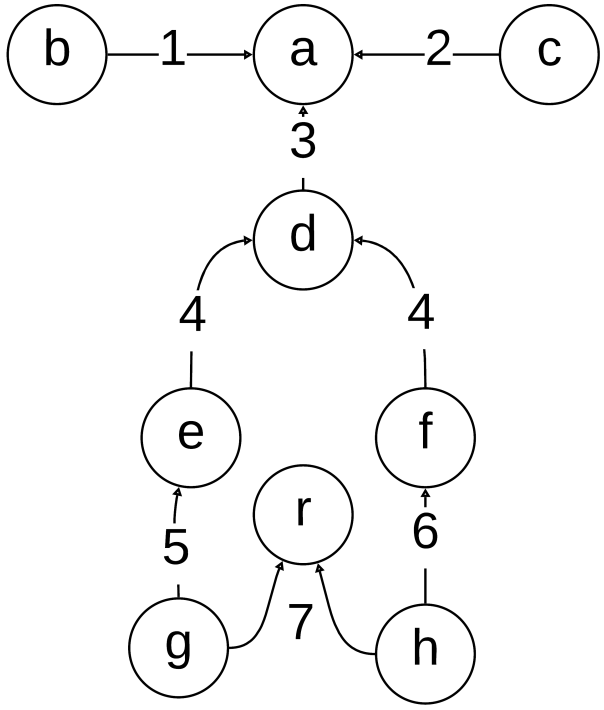


Figure 5.1: Toy example of workflow services connecting the various states of information. Nodes represent information goals. (Hyper-)Edges represent the workflow services that perform the transformations.

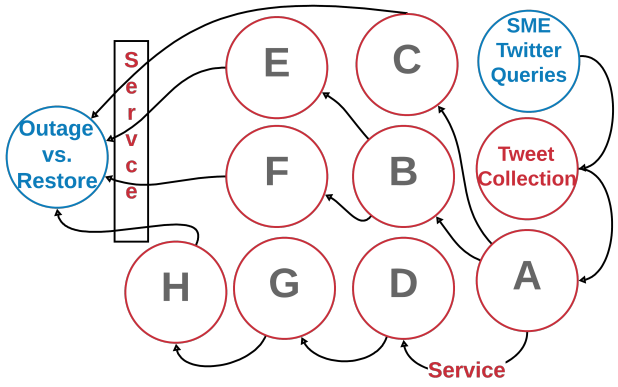


Figure 5.2: A graph representation of one of the workflows from the CRISP-funded project case study. The workflow is to produce a comparison of tweets that report outages, against tweets that report restoration of services, during a hurricane. As in the prior figure, the nodes represent the information goals and the edges represent the workflow services to produce that information.

Node/Information Goal	Description
A	Hurricane-related tweets
B	Tweets with sentiment score
C	Tweets with geo-location
D	Tweets classified into categories (Caution and advice; Injured or dead people; Utility damage; Sympathy emotional support; etc.)
E	Tweets about outage
F	Tweets about restoration
G	Tweets classified into categories (Power; Transportation; Sewage; Water; Communication)

Figure 5.3: Description of Nodes for Case Study Graph from Figure 5.2

SME wanted to know about outages reported on Twitter during a hurricane. Based on the developer's expertise on data mining of Twitter data, this information goal was broken down into steps that involved text mining. Figure 5.3 shows the information that the developer derived and used in the process of extraction related to the information goal of the SME.

With our proposed representation of the hypergraph, we can provide the two types of information of interest to the SME via algorithms that we discuss in the following sections.

5.2 Algorithm for Workflow Generation

We next consider how to generate workflows, using the knowledge graph. We use the term "reasoner" to describe the generator, since in some cases, an inference process, possibly aided by an optimizer, may be needed to produce a "best" workflow that supports a particular information goal.

Let us again consider the toy graph in Figure 5.1.

Let us say the SME wants the information goal "a". There are three possible workflows,

broken down as:

1. a = Service 1, or

2. a = Service 2, or

3. a = Service 3 + Service 4 + Service 5 + Service 6

We can achieve this breakdown if we go down the three different "paths" leading up to the information goal "a". When these workflows, or sequences of services, are executed by any workflow engine, we can send the generated information back to the SME. We can similarly construct workflows to generate **any** of the goals in the graph. To generate a sequence, we recursively go "up" the graph starting with the node representing the information requested by the SME. The recursion continues until we reach nodes with no parent. Since there are three hyperedges incident to the information goal "a", there are three possible "paths" one could take, and therefore three different workflow sequences (as we have highlighted above).

This process of generating a workflow is similar to the recursive replacement traversal of a context-free grammar for generating sentences [8, 9].

A context-free grammar is a type of formal grammar that consists of a set of rules known as "production rules." We can use these rules to generate and describe all patterns of strings in a context-free language [8]. A CFG has the following components:

1. Terminal Symbol: Characters or letters that appear in the strings of the language.

2. Non-terminal Symbol: Variables that are placeholders for terminal symbols.

3. Production Rule: Rules for replacing non-terminal symbols.

We start with a string with non-terminal symbols, and based on the rules, we generate a string that only has terminal symbols. Let us consider an example that is shown in Wikipedia [60].

In this example, we want to construct a language, where each string has only two characters from the alphabet: α and β. The production rules are:

$S \rightarrow \{AA\}$

$A \rightarrow \alpha$

$A \rightarrow \beta$

Terminal Symbols = α, β

Non-Terminal Symbol = A (S is a special type of non-terminal symbol that only appears in the initial string.)

To generate a two alphabetic character string, we begin with the rule that has the S symbol. So currently, our string is: AA

We have two replacement rules. Therefore, valid strings in the language can be any of [“$\alpha\alpha$”, “$\alpha\beta$”, “$\beta\alpha$”, “$\beta\beta$”].

To summarize, to generate a sentence from a CFG, we begin with the start symbol. We then successively expand each leftmost non-terminal until we replace all non-terminals. This recursive replacement of non-terminals is similar to the traversal of the directed hypergraph. Thus a CFG sentence is like a workflow. Algorithm 1 describes the algorithm.

As highlighted by Mckenzie et al., this solution has two specific limitations [40]:

1. The recursive process might fail to halt.
2. The process does not uniformly generate a sentence at random.

Thus, normally, this algorithm would be considered naïve.

Algorithm 1 Generates a sentence given the start symbol

```
1: procedure GenerateWorkflow(startSymbol)
2:     sentence = ''
3:     randprod = (self.prod[startSymbol])
4:     for sym in randprod: do
5:         if sym in self.prod: then
6:             sentence+ = self.GenerateWorkflow(sym)
7:         else
8:             sentence+ = sym+' '
9:         end if
10:    end for
11:    return sentence
12: end procedure
```

The first problem would occur if the grammar has a production of the form: S $\rightarrow$ S S | *a*. However, this situation does not arise for our productions because the workflows are not cyclic. There will never be a circumstance where a non-terminal in the left-hand side of the production (which in our case is an output produced from a service) appears in the right-hand side of the production (which in our case represents an input to the service).

The second problem does not apply to us either. We are looking to generate **all** "sentences" (or workflows) beginning with the particular "start symbol" (or information goal). Thus, because we generate all possible workflows, random generation is not needed.

Accordingly, we can use this simple process to generate a workflow based on the information goal requested by SMEs. As a valid string produced by the CFG is a workflow, a set of workflows for each case study could be considered as a "language" for that case study.

5.2.1 Scalability

In the introductory chapter of this dissertation, Chapter 1, we talked about the (lack of) scalability in SME current practices to solve their information goals and keep up with the

latest developments in data mining-related research.

Fortunately, our workflow-based information system architecture can support crowd-sourcing efforts involving UX researchers, developers, and other SMEs. This ensures scalability. As more and more contributions occur, the knowledge graph grows, and the range of problems that SMEs can solve grows even faster. This is how the system scales.

In terms of system performance, we can refer to [20]. When the reasoner is called upon to generate a workflow for an SME goal, it has to generate a path in the hypergraph. In the solution given above, the path is generated using breadth-first traversal. Given the non-recursive nature of the productions, the algorithm should scale linearly with the size of the graph (or $\mathcal{O}(V+E)$ where V = the number of information goals and E = number of services). In short, the cost of each traversal is linear relative to the size of the graph.

5.3 Algorithm for Related Information Identification

A second interest of SMEs would be to know "What information can I derive from my data?" Through our experiences interviewing SMEs, we found this question commonly occurring. Often, an SME has curated a large collection that is of interest to them and others. However, they do not know what to do with the data. They are unsure what information can be extracted or inferred as it is not their job to keep track of state-of-the-art data-mining efforts. For instance, in the CRISP-funded project, the SME had an initial set of goals. These goals evolved, based on what data we extracted, and conversations with the developer about what one can extract from Twitter data.

We want a data structure, like a dictionary, where we would store, for every node/information goal, all of the nodes that we can reach. We would need to update this as newer hyperedges

(or workflow services) are added to the graph. Having a graph structure helps to provide an answer to this question. Let us say that we add a new workflow service, or hyperedge, to a graph. We would need to update our records to include the new information goal that we can reach through this new hyperedge. A directed hyperedge consists of vertices/nodes V and a set of hyperarcs H, where a hyperarc is represented as the pair <S, v>, where S is a non-empty subset of V and v ∈ V [3]. So if a node u is "reachable" to all elements of S, then it is reachable to v. For an entry in the dictionary or table, we update to include node/information goal v. This process is how we will store and maintain the connections between information goals. Please refer to Algorithm 2 to see how we use a dictionary to maintain the connection between information goals.

Algorithm 2 Maintain a dictionary where key = information goal; value = list of information goals you can reach from it

```
1: procedure MaintainGoalReachableDict(NewGoal, [INP = listofinputs], currentDictionary)
2:     for each inputGoal in INP do
3:         for goal, reacheableGoals in currentDictionary.items(): do
4:             if inputGoal in reacheableGoals: then
5:                 reacheableGoals.append(NewGoal)
6:                 currentDictionary[goal] = reacheableGoals    ▷ Update dictionary
7:             end if
8:         end for
9:     end for
10: end procedure
```

5.4 Summary

Through part two of the research, we have been able to take user information needs, and construct hypergraph-based knowledge graphs that can be used to generate workflows. We propose representing the hypergraph as productions of a CFG. Based on the recursive rules of a CFG, we can answer two fundamental questions that an SME would have about their

information needs. We developed algorithms to (a) generate a workflow for an information need, and (b) provide a list of information one can derive from data at hand (Algorithms 1 and 2). Through this effort, we have provided the data structure and set of algorithms that allow SMEs to get the information they want without having to familiarize themselves with libraries or tools. Appendices A and B provide descriptions of efforts by student teams that built the various components of the workflow-based information system across two different classes. Appendix A describes the first effort to build the components of a workflow-based information system. It includes the construction of the knowledge graph, the services registry, and the CFG-based reasoner. The main shortcoming of the implementation is that the service registry and, as a result, the graph, are manually populated. This shortcoming is addressed in the effort described in Appendix B. In Appendix B, the student team constructs a set of APIs that includes the API to add nodes and edges to the graph.

Chapter 6

Research Part 3: Description

Figure 6.1 showcases what we have learned from Part 1 and Part 2 of the research. The top of the figure outlines the artifacts produced from Part 1 of the methodology. The bottom of the figure highlights the system architecture and components of the workflow-based information system that we built in Part 2 of the research.

In this chapter, we detail our research efforts to formally describe a class of information systems or digital libraries that capture the solution space for the problem of supporting SME information needs. A formal description of a digital library is beneficial in guiding and organizing system-building efforts and building interoperable systems [18]. The goal of expressing a formal description for this class of workflow-based information systems is to have the same effect. We believe that future efforts to build such systems (that are workflow-based and that capture SME goals and workflow relationships) will reap benefits from having this description.

Figure 6.1 has the knowledge graph (a graph-based representation of artifacts produced from the methodology) and the reasoner at the center. When an SME requests some state as their information goal, the reasoner analyzes the knowledge graph and generates a workflow for the set of services or transformations that will produce the requested information. The reasoner forwards this information to the Workflow Manager or Workflow Management System (WMS). The WMS is responsible for orchestrating and executing the workflow, which consists of individual services. This component will connect with a Services Registry. The

registry will index and store all of the services built by developers. The range of information support from this Digital Library will be dynamic and based on the domains of the (a) ever-changing needs of the SME or researchers, (b) the advances in models/algorithms deployed as services by the developers, and (c) the type and quality of content created by the curators.

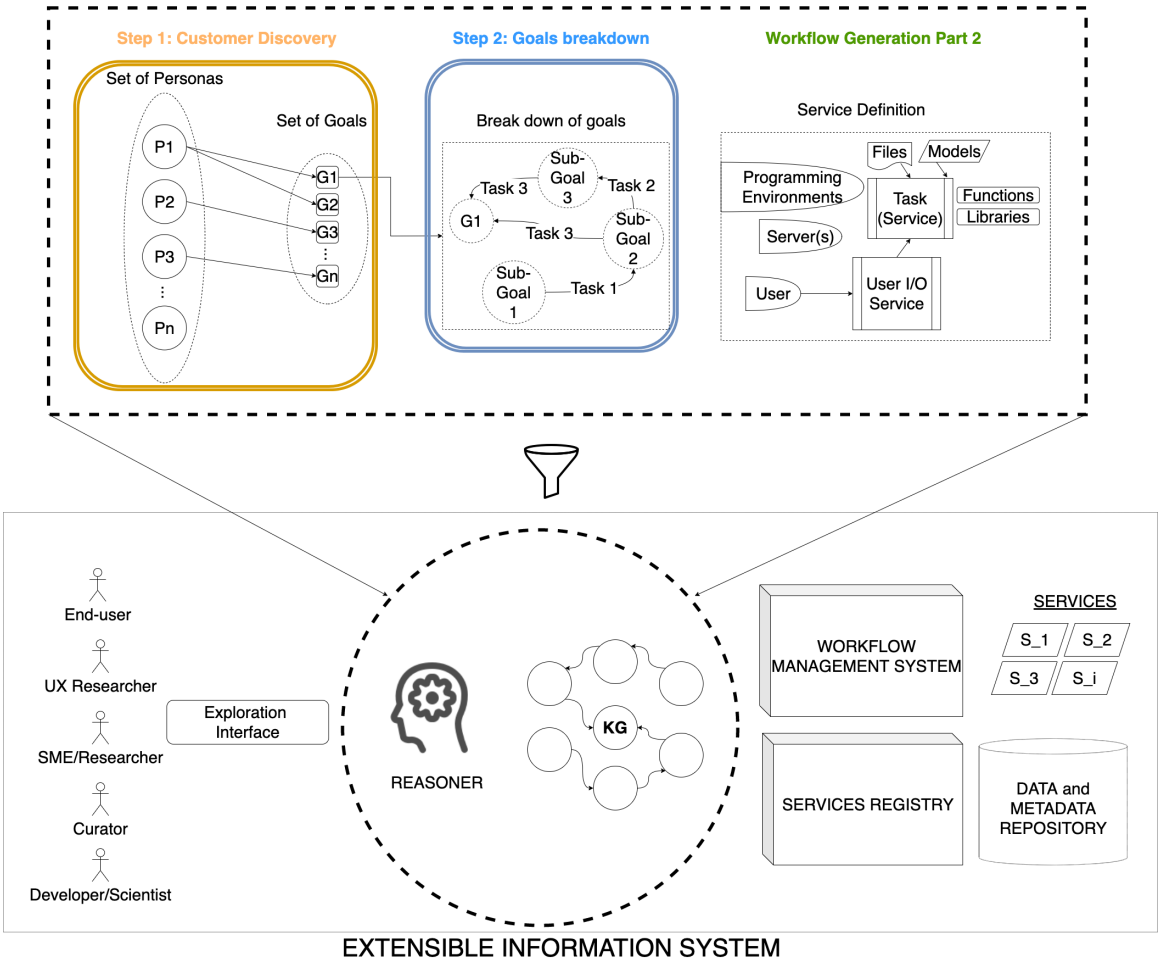


Figure 6.1: The artifacts produced from the different steps of the methodology are shown at the top. They lead to the components of the new Digital Library, shown at the bottom.

6.1 Key Concepts and Components

The components of a workflow-based system operate on two specific concepts:

1. Information state, and

2. Transition event

An information state is a set of functions or predicates. The domain of a function is a set of digital objects. If Q is a finite set of functions or predicates, then a state $s \in 2^Q$. A transition event is an operation that transforms a state to the next. An "input" to the workflow-based information system or WDL is an SME goal. A goal is a state of information desired by an SME. In response to the information goal, the WDL generates a set of workflows, each of which represents a sequence of transition events or operations. When the workflow is executed, the WDL produces the information requested by the SME.

The components of the WDL that operate on these concepts are:

1. A service registry or catalog that stores descriptions of the list of operations or transition events;

2. A knowledge graph that serves as a knowledge base that maintains the relationships between information states;

3. A reasoner that analyses the "conditions" wherein the operations or events, stored in the registry, are to operate on the states of the knowledge graph and returns a workflow representing a sequence of such events; and

4. A workflow manager that executes the operations to produce the output.

6.2 Minimal Workflow-based Digital Library (WDL)

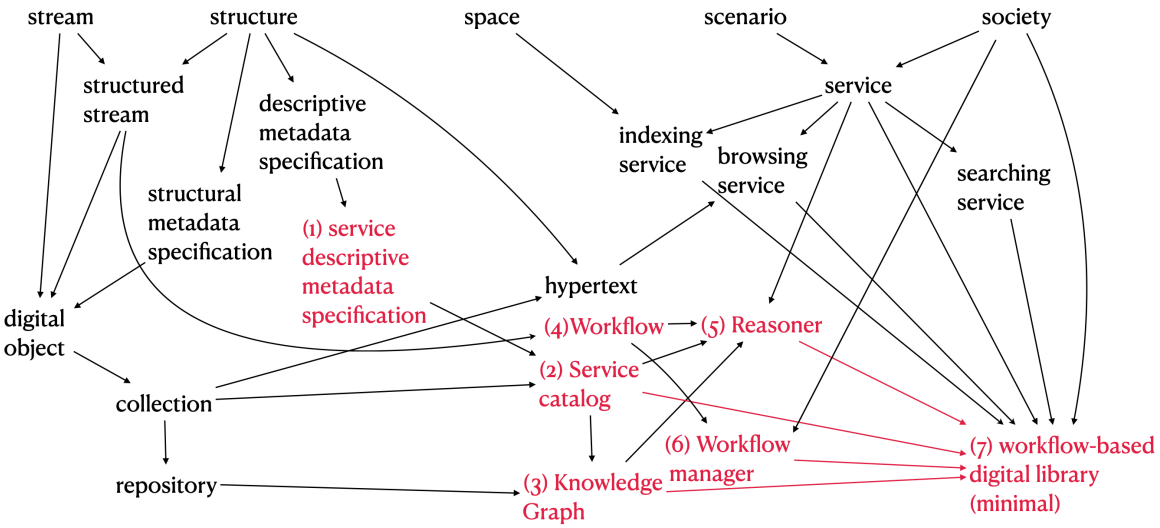


Figure 6.2: The concepts in black are what make a minimal digital library. The concepts in red are those that are specific and requisite when defining a workflow digital library. Arrows indicate that a concept is formally defined in terms of previously defined concepts that point to it.

Figure 6.2 showcases the essential components that, together, make a minimal workflow-based digital library.

6.2.1 WDL Components/Constructs

Please find below the set of formal descriptions of the components of the workflow-based digital library.

1. Service Descriptive Metadata Specification:

   A Service Specification is a descriptive metadata specification for Services. A descriptive metadata specification is a structure that emphasizes the semantic relationships. A Service Specification is a structure $(G, R \cup L \cup P, F)$, where:

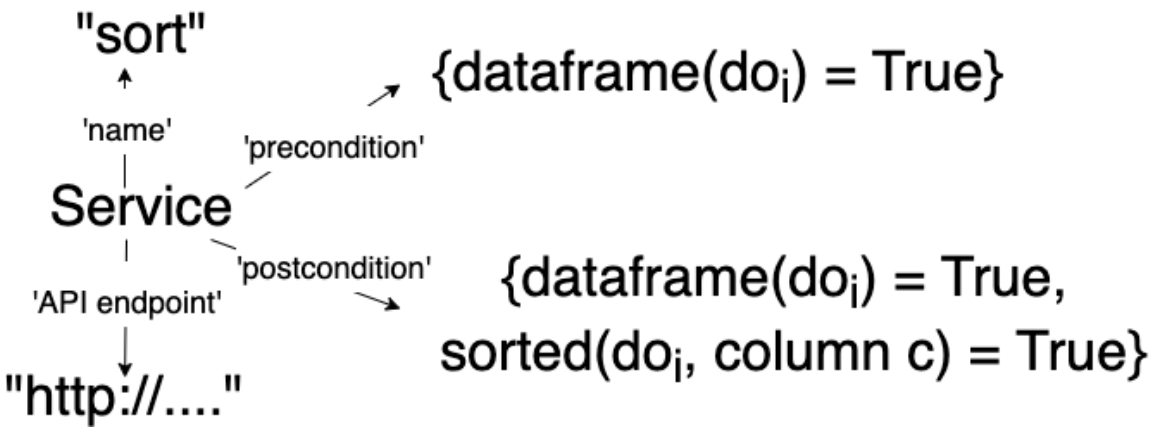


Figure 6.3: Example of Descriptive Metadata Specification for a Service

(a) R represents a set of labels for resources.

(b) $L = \cup D_k$ is the set of literals defined as the union of domains of simple data types (e.g., strings, numbers, dates, etc.).

(c) For each directed edge $e = (v_i, v_j)$ of G, $F(v_i) \in R \cup L, F(v_j) \in R \cup L$ and $F(e) \in P \in \{'name', 'precondition', 'postcondition', 'APIEndpoint'\}$

Here: IF $F(e) = \{'precondition', 'postcondition'\}$, THEN $F(v_j) \in$ information state, $s \in 2^Q$, (where Q = finite set of functions or predicates). IF $F(e) = \{'name', 'APIEndpoint'\}$, THEN $F(v_j) \in L$.

Let us consider a service (called "Sort Column Service"). Its function is to take a dataframe as input and produce a dataframe with one of the columns sorted in ascending order. Figure 6.3 shows a metadata description for the service. The description includes a precondition and a postcondition as well as the other properties mentioned in the formal definition. All services included in the digital library will have a description with these fields populated.

2. Service Catalog/Registry:

(a) Let C be a collection of Services with k handles in H.

handle	dm: name	dm: precond	dm: postcond	dm: API
sorting handle h_s	"sort"	{dataframe(do_i) = True}	{dataframe(do_i) = True, sorted(do_i, column c} = True	"http://...s"
filtering handle h_f	"filter"	{dataframe(do_i) = True}	{dataframe(do_i) = True, filtered(do_i, query q} = True	"http://...f"

Figure 6.4: Example entries for the Service Catalog/Registry

(b) A Service Catalog or Registry for the collection C is a set of pairs $(h, dm_1, dm_2, ..., dm_i, ...)$, where $h \in H$ and each dm_i is a service descriptive metadata specification.

Figure 6.4 provides an example for a tabular form of the catalog/registry. The table has two entries, each a Service Descriptive Metadata Spec. for a Service.

3. Knowledge Graph: A Knowledge Graph is a repository with a graph structure G = (V, E), where:

   (a) V is a set of information states,

   (b) E is an edge between (v_i, v_j), where $v_i, v_j \in V$ if there exists a Service with handle, h_i, with a precondition state set, t, such that $t \subseteq v_i$ and with a post-condition state set, u, such that $u \subseteq v_j$.

4. Workflow: A workflow is a structured stream representing a sequence of Services. The definition of a structured stream is that of a function that assigns sub-sequences to the stream, S.

5. Reasoner:

   A Reasoner is a Service that takes as input a Planning instance, Π, and produces a workflow, W, that is a solution that achieves goal state, g.

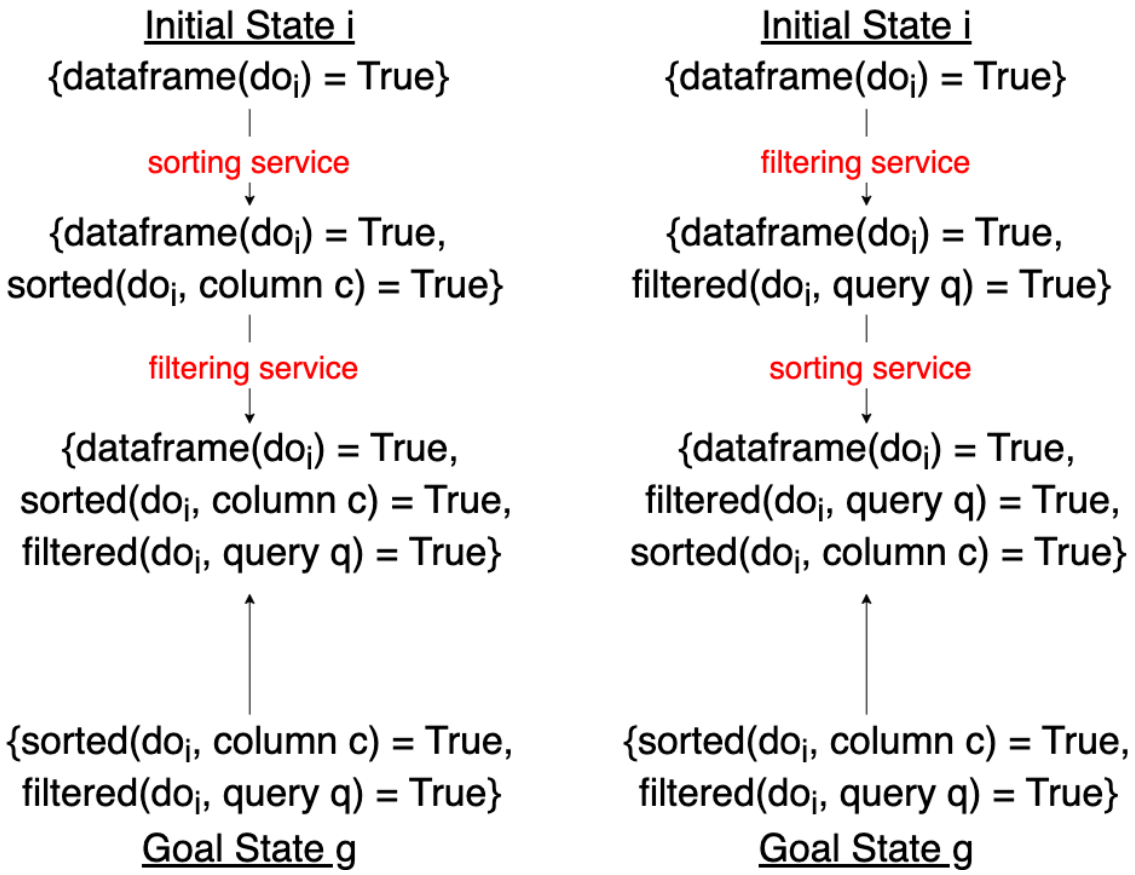


Figure 6.5: Example workflow sequence derived by the Reasoner, given the initial state and the goal state, based on the Knowledge Graph and the Service Catalog

(a) A planning instance or a planning problem is represented by a tuple Π=(KG, i, g), in which KG= the knowledge graph, which specifies the domain knowledge; $i \subseteq 2^Q$ is the initial state specification; and $g \subseteq= 2^Q$ is the goal state. Here Q = finite set of state functions.

(b) A workflow is a sequence of Services $w =< s_1, s_2, ..., s_n >$ which, when executed by a workflow engine, transform from the initial state to achieve the goal state.

Figure 6.5 shows an example of two possible workflows for a given initial state and goal state and the Knowledge Graph based on the Service Catalog.

6. Workflow Manager: A workflow manager is a Society that implements a sequence of Services or a Workflow. As a reminder, a society is a set of entities and the relationships between them. The entities include humans actors as well as hardware and software components, which either use or support digital library services.

7. Workflow-based Digital Library: A Workflow-based Digital Library is a tuple (WDL) = (SC, RE, WM, KG, Serv, Soc), where:

   (a) Reasoner (RE) is a service in the WDL that generates a workflow;

   (b) Service Catalog (SC) is a catalog of workflow services;

   (c) Knowledge Graph (KG) is a graph-based repository of information states;

   (d) Workflow Manager (WM) is a service manager that executes workflows;

   (e) Serv is a set of services containing at least indexing, searching, and browsing; and

   (f) Soc = $(SM \cup Ac, R)$, where SM is a set of service managers responsible for running DL services, Ac is a set of actors that use those services, and R is a set of relationships among $SM \cup Ac$.

In the next section, we showcase how to build an example of an instance of a WDL. One could follow the same process to build a specialization from WDL.

6.3 Twitter-centric Workflow-based Digital Library

Several of our case studies (described earlier) involve supporting information needs that relate to Twitter data. Therefore, we describe a Twitter-centric workflow-based digital library (TWDL) as an instance of WDL and define characteristics that pertain to digital objects defined in terms of tweet data.

To describe a Twitter-centric workflow-based DL (TWDL), we need to first formally describe the information in Twitter. We borrow a suitable definition from [65]. The authors define the information in Twitter as a "Twitter Heterogeneous Information Network" because "Twitter data contains heterogeneous entities and multiple types of relationships".

Twitter Heterogeneous Information Network: A Twitter heterogeneous information network is defined as an un-directed graph $G = (V, E, W, S)$, where $V = T \cup F$.

T refers to a set of tweet nodes, and $F = F_1...F_M$ refers to M other types (e.g., term, user, and hashtag) of nodes, called feature nodes. $E \subseteq V \times V$ represents the set of edges, which are all undirected. W denotes the set of weights of nodes and edges. $S = \{l(v)|v \in T\}$ refers to a set of geographic locations of tweet nodes, where $l(v) \in R^2$ represents a tuple consisting of the latitude and longitude of tweet node v.

Each of the un-directed edges in E describes a relationship between tweet nodes and feature/tweet nodes. For instance, a tweet node could be a "reply" to another tweet node. Similarly, a user node (which is a feature node) would have an "authorship" relationship with the tweet node.

As mentioned above, a Twitter-centric workflow-based DL (TWDL) is a workflow-based DL that operates in the domain of the Twitter Heterogeneous Information Network (THIN). Therefore, we can define it as such.

The formal definition of TWDL has the following criteria/constraints on the definitions used to define workflow-based DL:

1. State Functions: A state set in a digital library is defined as a set of functions that operate on digital objects. The functions in a TWDL operate on "statements." Statements are triples (source node, edge, target node) representing the edge relationship between tweet nodes and/or feature nodes from the Twitter Heterogeneous Information Network (THIN).

2. States: Given a finite set of edges/functions, Q, representing THIN, a state s (as defined for WDL) is $\in 2^Q$. This state is a sub-graph or "sub-THIN".

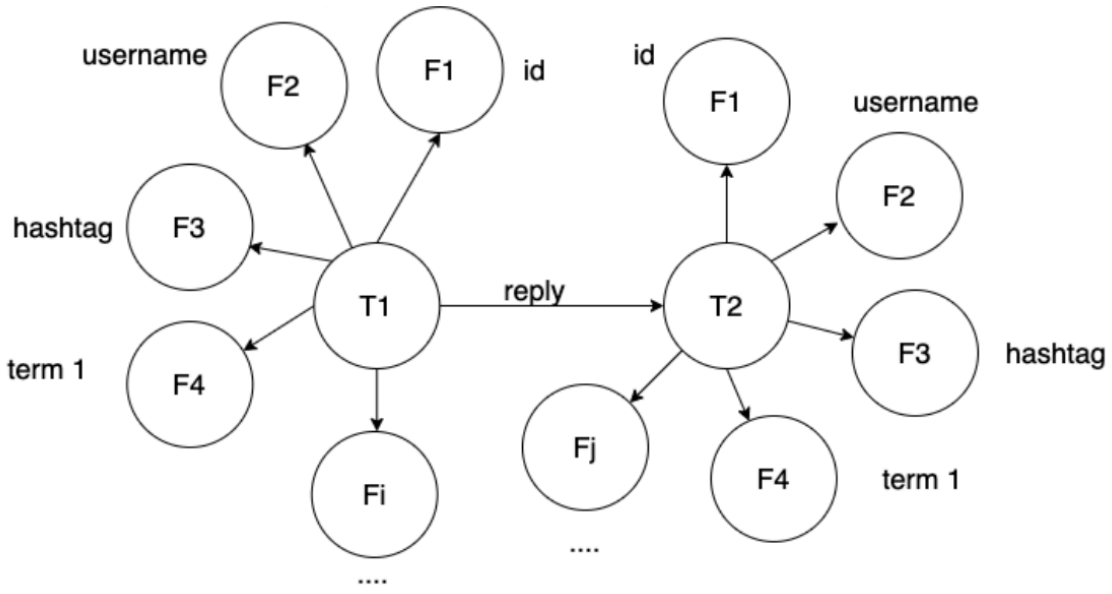


Figure 6.6: Example of a set of nodes in the Twitter Heterogeneous Information Network (THIN)

Example state S = {

(T, id, 0505),

(T, username,"@cfc"),

(T, hashtag, "#ucl"),

(T, term1, "all"),

(T, term2, "english"),

(T, term3, "final"),

...

}

Figure 6.7: Example of a state describing a node in THIN

Figure 6.6 shows an example of nodes in the network. Figure 6.7 shows an example of an "information state" that describe a sub-graph in the network.

We can define the digital objects for TWDL from this information. The set of services for TWDL is constrained by services that operate on THIN. These include, but are not limited to, services that perform network generation/mining and image and text mining.

The contents of the Knowledge Graph and the Service Catalog are based on THIN-centric digital objects and services. Regarding the formal description of a minimal TWDL, we can borrow the descriptions of the Knowledge Graph, Reasoner, and Service Catalog from WDL.

To similarly describe a minimal workflow-based digital library for ETDs and web pages, we can follow the same process. We first need to define the relevant types of digital object formally. Then we can identify the different state functions that operate on the ETD- or web page-based digital objects. That would allow us to build components of the information system or digital library specific to them.

Chapter 7

Conclusion

7.1 Summary

Our research began with the goal of supporting subject-matter experts (SMEs) in their research initiatives. More specifically, we considered facets of their research that require data analysis-centric tasks that can be automated. We have designed, implemented, and formally described a new type of extensible workflow-based information system (called WDL) to support such SMEs, as well as a broad range of other users. We devised a teaching methodology wherein UX researchers, SMEs (including end users, curators, data scientists, DL researchers, etc.), and developers work together to build a set of workflow artifacts such as workflow goals, workflow tasks, and services to support each task. This methodology helps integrate UX research efforts into the workflow design efforts to make the latter more useful.

To evaluate the usability and learnability of the methodology, we conducted one pilot study and two user studies, each of which involved students in a course centered around problem-based learning. In one case study, a set of student teams coordinated their efforts to build a service-centric information retrieval system. The methodology guided the teams to build services to support (sub-)tasks that, when applied in sequence, would yield a workflow that would address the SME goals. The teams also helped implement/integrate key parts of the WDL infrastructure: a service registry, a reasoner, and a workflow manager. We describe

these efforts in detail in Appendix A.

In another case study, student teams applied the methodology to their respective projects. They built user personas and identified goals early on in the course. Our motivation to introduce the methodology for this course was to organize the student teams' development efforts as they built features to support the (sub-)tasks they identified. We hoped that the student teams would find the methodology to be easy-to-learn and easy-to-use. Evaluation results confirmed this.

In another case study that involved teams collaborating to build an information retrieval system, we found that the students did indeed find the methodology easy to learn and use. Moreover, through qualitative feedback, the students reported that the methodology significantly impacted their solution's design and their development efforts. In the other case study, some students reported that the methodology was easy to learn and easy to use. However, it was hard for us to evaluate the impact of the methodology on the solution design since each team in the class was working on individual projects with different scopes and types of implementation. Through these case studies we were able to validate our conjecture as students took up the roles of UX researchers and developers, instrumented our methodology, and were able to build a workflow-based information system that SMEs could use to answer their goals.

Our next effort was to design two essential components that allowed SMEs to find information without learning the data mining effort executed by the workflows required to produce it: (1) a reasoner and (2) a knowledge graph-based goal-task-workflow representation. We devised a hypergraph-based representation of the knowledge. Hypergraphs allowed us to capture multiple ways to derive information and while maintaining an association between information goals. The graph was extensible in that as developers built more workflow services, there would be multiple ways to obtain or "reach" a particular information goal. Appendix

A presents an implementation of the hypergraph using a combination of PostgreSQL tables. Appendix B presents an alternative system built using GRAKN's knowledge graph. Once constructed, we needed a way for an SME to explore this graph and for the graph representation to enable discovery for workflows that generate SME goals. We built upon a notion that SME goals and workflow services should be linked the same way entities in a sentence (in a language) are connected to one another. We built a context-free-grammar-based reasoner where the hyperedges represent the grammar productions, and a "workflow" would be a sentence. From a representation standpoint, the benefits of this approach are that CFG engines are relatively common, and it would be reasonably easy to translate productions in the grammar to other workflow languages if required. Additionally, we can conceivably create a "language" for each case study or represent each SME/researcher line of inquiry as a language. This would allow for easier sharing of information. As a result, the SMEs can query their information goal, and the CFG-based reasoner will generate a "sentence" or a workflow that would satisfy their goal when executed. Our design of the Knowledge Graph and Reasoner and its implementation confirmed our second conjecture that these two components help translate any previously identified SME goals in that domain into workflows that, when executed, yield relevant results for each goal/query.

Finally, our research looks to take what we have learned and experienced via the different case studies and the solution we have built earlier to define a formal model to represent a (sub-)class of digital libraries to support information exploration via workflows. The model represents characteristics of the system and unambiguously defines components of a workflow-based information system. As per our conjecture for the third and final research question, we were able to identify components that make up a minimal workflow-based digital library. We employ the 5S framework that provides formalisms to define digital library components. Our model will guide development teams and help them build consistent and

interoperable solutions.

We have published an article that describes the design process of building a digital library that supports information exploration [16]. We have filed a provisional patent application that focuses on the workflow design methodology. We aim to convert it to a utility patent within the period allowed.

7.2 Future Work

In terms of future work, we plan to introduce the methodology for future iterations of the CS4624 course. Before teaching the student teams the methodology, we plan to gather information on students' current knowledge and exposure to UX research and build workflow services for a service-oriented architecture. We would like to see how prior exposure impacted the student's understanding and instrumentation of the methodology. Additionally, we would evaluate if the methodology reinforced or complemented their existing workflow building and UX research experiences.

When we do introduce the methodology in upcoming semesters, an area of optimization would be to scale the SME interview process. This optimization would aid the UX researchers in building consistent personas and identify goals. An approach I am considering is designing and employing a conversational AI-based chatbot to elicit information from SMEs. AI-powered chatbots have shown the ability to frame survey questions in a more personalized manner, which improves response quality [64]. We can also compare the information we learn via the chatbot against traditional interviewing by UX researchers.

We also plan to publish the lessons we have learned (via student evaluation) from our experience teaching the methodology.

We can stitch together the efforts from the student teams from the two classes (as described in Appendices A and B). First, we need to build a user interface that would allow developers to add and define services to the service registry by calling the previously mentioned API. We need to integrate the infrastructures as well. Once that is done, we can improve upon specific information system components, such as the reasoning (or workflow generation) aspect of the information system.

Bibliography

[1] Peter Amstutz, Michael R. Crusoe, Nebojša Tijanić, Brad Chapman, John Chilton, Michael Heuer, Andrey Kartashov, Dan Leehr, Hervé Ménager, Maya Nedeljkovich, Matt Scales, Stian Soiland-Reyes, and Luka Stojanovic. Common Workflow Language, v1.0. 7 2016. doi: 10.6084/m9.figshare.3115156.v2. URL https://figshare.com/articles/dataset/Common_Workflow_Language_draft_3/3115156.

[2] Apache Software Foundation. Apache airflow. https://airflow.apache.org/docs/stable/. Accessed: 08-August-2020.

[3] Giorgio Ausiello and Luigi Laura. Directed hypergraphs: Introduction and fundamental algorithms—a survey. *Theoretical Computer Science*, 658:293–306, 2017.

[4] Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Kilian Thiel, and Bernd Wiswedel. Knime - the Konstanz information miner: Version 2.0 and beyond. *SIGKDD Explor. Newsl.*, 11(1):26–31, November 2009. ISSN 1931-0145. doi: 10.1145/1656274.1656280. URL https://doi.org/10.1145/1656274.1656280.

[5] Daniel Blankenberg, Gregory Von Kuster, Nathaniel Coraor, Guruprasad Ananda, Ross Lazarus, Mary Mangan, Anton Nekrutenko, and James Taylor. Galaxy: A web-based genome analysis tool for experimentalists. *Current Protocols in Molecular Biology*, 89(1):19.10.1–19.10.21, 2010. doi: 10.1002/0471142727.mb1910s89. URL https://currentprotocols.onlinelibrary.wiley.com/doi/abs/10.1002/0471142727.mb1910s89.

[6] John Brooke. Sus: a "quick and dirty" usability. *Usability evaluation in industry*, page 189, 1996.

[7] Cao, Yusheng and Mazloom, Reza and Ogunleye, Makanjuola. Cs5604 (information retrieval) fall 2020 front-end (fe) team project. http://hdl.handle.net/10919/101526. Accessed: 25-May-2021.

[8] N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956.

[9] Noam Chomsky and David W Lightfoot. *Syntactic structures*. Walter de Gruyter, 2002.

[10] David De Roure, Carole Goble, and Robert Stevens. The design and realisation of the Experimentmy virtual research environment for social sharing of workflows. *Future Gener. Comput. Syst.*, 25(5):561–567, May 2009. ISSN 0167-739X. doi: 10.1016/j.future.2008.06.010. URL https://doi.org/10.1016/j.future.2008.06.010.

[11] Ewa Deelman, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Sonal Patil, Mei-Hui Su, Karan Vahi, and Miron Livny. Pegasus: Mapping scientific workflows onto the grid. In *European Across Grids Conference*, pages 11–20. Springer, 2004.

[12] Ewa Deelman, Karan Vahi, Mats Rynge, Gideon Juve, Rajiv Mayani, and Rafael Ferreira da Silva. Pegasus in the cloud: Science automation through workflow technologies. *IEEE Internet Computing*, 20(1):70–76, January 2016. ISSN 1089-7801. doi: 10.1109/MIC.2016.15. URL https://doi.org/10.1109/MIC.2016.15.

[13] Eclipse Foundation. Eclipse ide for java developers. https://www.eclipse.org/downloads/packages/release/kepler/sr1/eclipse-ide-java-developers. Accessed: 15-February-2021.

[14] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48:1–4, 2016.

[15] Fan, Jiahui and Hardy, Nicolas and Furman, Samuel and Manzoor, Javaid and Nguyen, Alexander and Raghuraman, Aarathi. Cs5604 fall 2020: Electronic thesis and dissertation (etd) team. http://hdl.handle.net/10919/101511. Accessed: 15-February-2021.

[16] Edward A. Fox and Prashant Chandrasekar. How should one explore the digital library of the future? *Data and Information Management*, 0(0):–, 2021. doi: doi:10.2478/dim-2021-0003. URL https://doi.org/10.2478/dim-2021-0003.

[17] Edward A. Fox and Ricardo da Silva Torres. *Digital Library Technologies: Complex Objects, Annotation, Ontologies, Classification, Extraction, and Security*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2014. ISBN 9781627050302. doi: 10.2200/S00566ED1V01Y201401ICR033. URL http://dx.doi.org/10.2200/S00566ED1V01Y201401ICR033.

[18] Edward A. Fox and Jonathan Leidig. *Digital Libraries Applications: CBIR, Education, Social Networks, eScience/Simulation, and GIS*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2014. ISBN 9781627050326. doi: 10.2200/S00565ED1V01Y201401ICR032. URL http://dx.doi.org/10.2200/S00565ED1V01Y201401ICR032.

[19] Edward A. Fox, Marcos André Gonçalves, and Rao Shen. *Theoretical Foundations for Digital Libraries: The 5S (Societies, Scenarios, Spaces, Structures, Streams) Approach*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2012. ISBN 9781608459100. doi: 10.2200/S00434ED1V01Y201207ICR022.

[20] Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. Directed hypergraphs and applications. *Discrete applied mathematics*, 42(2-3):177–201, 1993.

[21] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.

[22] Y. Gil, V. Ratnakar, J. Kim, P. Gonzalez-Calero, P. Groth, J. Moody, and E. Deelman. Wings: Intelligent workflow-based design of computational experiments. *IEEE Intelligent Systems*, 26(1):62–72, Jan 2011. ISSN 1941-1294. doi: 10.1109/MIS.2010.9.

[23] Yolanda Gil. Human tutorial instruction in the raw. *ACM Trans. Interact. Intell. Syst.*, 5(1), March 2015. ISSN 2160-6455. doi: 10.1145/2531920. URL https://doi.org/10.1145/2531920.

[24] Yolanda Gil. Teaching big data analytics skills with intelligent workflow systems, 2016. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12547.

[25] Yolanda Gil and Daniel Garijo. Towards automating data narratives. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, IUI '17, page 565–576, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450343480. doi: 10.1145/3025171.3025193. URL https://doi.org/10.1145/3025171.3025193.

[26] Yolanda Gil, Ewa Deelman, Jim Blythe, Carl Kesselman, and Hongsuda Tangmunarunkit. Artificial intelligence and grids: Workflow planning and beyond. *IEEE Intelligent Systems*, 19(1):26–33, 2004.

[27] Carole Anne Goble and David Charles De Roure. Myexperiment: Social networking for workflow-using e-scientists. In *Proceedings of the 2nd Workshop on Workflows in Support of Large-Scale Science*, WORKS '07, page 1–2, New York, NY, USA, 2007. Association

for Computing Machinery. ISBN 9781595937155. doi: 10.1145/1273360.1273361. URL https://doi.org/10.1145/1273360.1273361.

[28] Google and Cloud Native Computing Foundation. Kubernetes. https://github.com/kubernetes/kubernetes. Accessed: 10-October-2020.

[29] Katharina Görlach, Mirko Sonntag, Dimka Karastoyanova, Frank Leymann, and Michael Reiter. *Conventional Workflow Technology for Scientific Simulation*, pages 323–352. Springer London, London, 2011. ISBN 978-0-85729-439-5. doi: 10.1007/978-0-85729-439-5_12. URL https://doi.org/10.1007/978-0-85729-439-5_12.

[30] Haney, Casey D. and Pastore, Camellia and Teshome, Bethlehem and Pant, Ashutosh. Migrating cs5604 applications from cs container cluster to digital library research laboratory. http://hdl.handle.net/10919/103273. Accessed: 12-May-2021.

[31] Rex Hartson and Pardha Pyla. *The UX Book: Process and Guidelines for Ensuring a Quality User Experience*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2012. ISBN 0123852412.

[32] Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, and Christian Muise. An introduction to the planning domain definition language. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(2):1–187, 2019.

[33] Tony Hey, Stewart Tansley, and Kristin Tolle. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, October 2009. ISBN 978-0-9825442-0-4. URL https://www.microsoft.com/en-us/research/publication/fourth-paradigm-data-intensive-scientific-discovery/.

[34] Hicks, Alexander and Thazhath, Mohit and Gupta, Suraj and Long, Xingyu and Poland,

Cherie and Hsieh, Hsinhan and Mahajan, Yash. Integration and implementation (int) cs 5604 f2020. http://hdl.handle.net/10919/101544. Accessed: 15-February-2021.

[35] Information Management Group, Department of Computer Science, The University of Manchester. escience lab. http://www.mygrid.org.uk. Accessed: 10-October-2016.

[36] Jeff Gentry, Chris Llanwarne, Mike Lin. Open workflow description langugage. http://openwdl.org. Accessed: 01-July-2019.

[37] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter development team. Jupyter notebooks ? a publishing format for reproducible computational workflows. In Fernando Loizides and Birgit Scmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90. IOS Press, 2016. URL https://eprints.soton.ac.uk/403913/.

[38] Chee Sun Liew, Malcolm P. Atkinson, Michelle Galea, Tan Fong Ang, Paul Martin, and Jano I. Van Hemert. Scientific workflows: Moving across paradigms. *ACM Comput. Surv.*, 49(4), December 2016. ISSN 0360-0300. doi: 10.1145/3012429. URL https://doi.org/10.1145/3012429.

[39] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the Kepler system: Research articles. *Concurr. Comput. : Pract. Exper.*, 18(10): 1039–1065, August 2006. ISSN 1532-0626.

[40] Bruce McKenzie. Generating strings at random from a context free grammar, 1997.

[41] Mendix Tech BV. Mendix: Low-code application development platform. https://www.r-project.org/. Accessed: 15-February-2021.

[42] Meno, Emma and Vincent, Kyle. Twitter-based knowledge graph for researchers. http://hdl.handle.net/10919/98239. Accessed: 10-January-2021.

[43] Steven John Metsker and William C. Wake. *Design Patterns in Java(TM) (2nd Edition) (Software Patterns Series)*. Addison-Wesley Professional, 2006. ISBN 0321333020.

[44] Nirwan, Debby. How i implemented algorithm in python planning graph. https://tinyurl.com/jdx6dkss. Accessed: 15-March-2021.

[45] Wolfgang Pree. *Design Patterns for Object-Oriented Software Development*. ACM Press/Addison-Wesley Publishing Co., USA, 1995. ISBN 0201422948.

[46] Rao Shen, Marcos André Gonçalves, and Edward A. Fox. *Key Issues Regarding Digital Libraries: Evaluation and Integration*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2013. ISBN 9781608459124. doi: 10.2200/S00474ED1V01Y201301ICR026. URL http://dx.doi.org/10.2200/S00474ED1V01Y201301ICR026.

[47] Shere, Danya and Ayub, Ahmad and Mueller, Rebecca and Fabian, Lexi and Shah, Akshat. CS4624: Multimedia, hypertext, and information access: Us state tourism websites project. http://hdl.handle.net/10919/98257. Accessed: 15-July-2020.

[48] Song, Ziqian and Thackaberry, Taylor and Bogemann, Kayley and Burchard, Shane and Butler, Jessie and Spencer, Austin and Li, Liuqing and Wernstedt, Kris and Murray-Tuite, Pamela and Fox, Edward A. Twitter disaster behavior. https://fox.cs.vt.edu/talks/2021/20210524ISCRAM2021PosterPuertoRicoTweets.pdf. Accessed: 28-May-2021.

[49] Ian Taylor, Matthew Shields, Ian Wang, and Andrew Harrison. *The Triana Workflow Environment: Architecture and Applications*, pages 320–339. Springer London, London, 2007. ISBN 978-1-84628-757-2. doi: 10.1007/978-1-84628-757-2_20. URL https://doi.org/10.1007/978-1-84628-757-2_20.

[50] Thackaberry, Taylor and Bogemann, Kayley and Burchard, Shane and Butler, Jessie and Spencer, Austin. Twitter disaster behavior. http://hdl.handle.net/10919/98251. Accessed: 25-May-2021.

[51] The Apache Software Foundation. Apache jena: Inference. https://jena.apache.org/documentation/inference/index.html, . Accessed: 15-August-2019.

[52] The Apache Software Foundation. Apache netbeans. https://netbeans.apache.org/, . Accessed: 15-February-2021.

[53] The Pegasus Project. Pegasus workflow gallery. https://pegasus.isi.edu/workflow_gallery/, . Accessed: 12-May-2021.

[54] The Pegasus Project. Pegasus application showcase. https://pegasus.isi.edu/application-showcase/, . Accessed: 12-May-2021.

[55] The R Foundation. The r project for statistical computing. https://www.r-project.org/. Accessed: 15-February-2021.

[56] Vaticle Inc. Grakn.ai. https://grakn.ai/. Accessed: 15-Jan-2020.

[57] Manuela Veloso, Jaime Carbonell, Alicia Perez, Daniel Borrajo, Eugene Fink, and Jim Blythe. Integrating planning and learning: The prodigy architecture. *Journal of Experimental & Theoretical Artificial Intelligence*, 7(1):81–120, 1995.

[58] Verelly, Abhinav and David, Gruhn and Bhattarai, Ashutosh and Grishaw, Shane. Us state tourism. http://hdl.handle.net/10919/103269. Accessed: 25-May-2021.

[59] Weedon, Daniel B. and Olsen, Daniel. Knowledge graph. http://hdl.handle.net/10919/103271. Accessed: 25-May-2021.

[60] Wikipedia contributors. Context-free grammar — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Context-free_grammar&oldid=934576648, 2020. Accessed: 16-October-2019.

[61] Wikipedia Foundation. Semantic reasoner. https://en.wikipedia.org/wiki/Semantic_reasoner. Accessed: 15-August-2019.

[62] William A. Ingram, Edward A. Fox, and Jian Wu. Imls grant: Opening books and the national corpus of graduate research. https://www.imls.gov/sites/default/files/grants/lg-37-19-0078-19/proposals/lg-37-19-0078-19-preliminary-proposal.pdf. Accessed: 20-July-2020.

[63] Katherine Wolstencroft, Robert Haines, Donal Fellows, Alan Williams, David Withers, Stuart Owen, Stian Soiland-Reyes, Ian Dunlop, Aleksandra Nenadic, Paul Fisher, Jiten Bhagat, Khalid Belhajjame, Finn Bacall, Alex Hardisty, Abraham Nieva de la Hidalga, Maria P. Balcazar Vargas, Shoaib Sufi, and Carole Goble. The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Research*, 41(W1):W557–W561, 05 2013. ISSN 0305-1048. doi: 10.1093/nar/gkt328. URL https://doi.org/10.1093/nar/gkt328.

[64] Ziang Xiao, Michelle X. Zhou, Q. Vera Liao, Gloria Mark, Changyan Chi, Wenxi Chen, and Huahai Yang. Tell me about yourself: Using an ai-powered chatbot to conduct conversational surveys with open-ended questions. *ACM Trans. Comput.-Hum. Interact.*, 27(3), June 2020. ISSN 1073-0516. doi: 10.1145/3381804. URL https://doi.org/10.1145/3381804.

[65] Liang Zhao, Feng Chen, Jing Dai, Ting Hua, Chang-Tien Lu, and Naren Ramakrishnan. Unsupervised spatial event detection in targeted domains with applications to civil unrest modeling. *PloS one*, 9:e110206, 10 2014. doi: 10.1371/journal.pone.0110206.

Appendices

Appendix A

Airflow, Registry, UI, and Other Development Aspects

During Fall 2020, the CS5604 graduate course built components of a workflow-based information system. This chapter describes tools, frameworks, and the implementation of concepts/components that are at the core of a workflow-based information system.

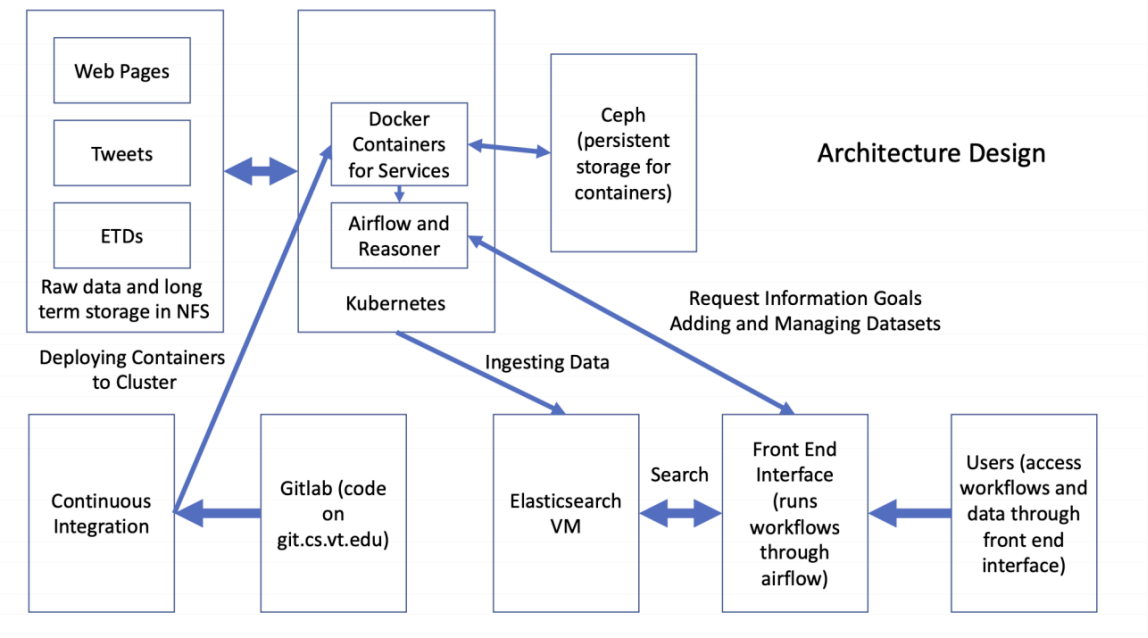


Figure A.1: Information System Architecture built by CS5604 team

Figure A.1 shows the architecture and the components of the workflow-based information system.

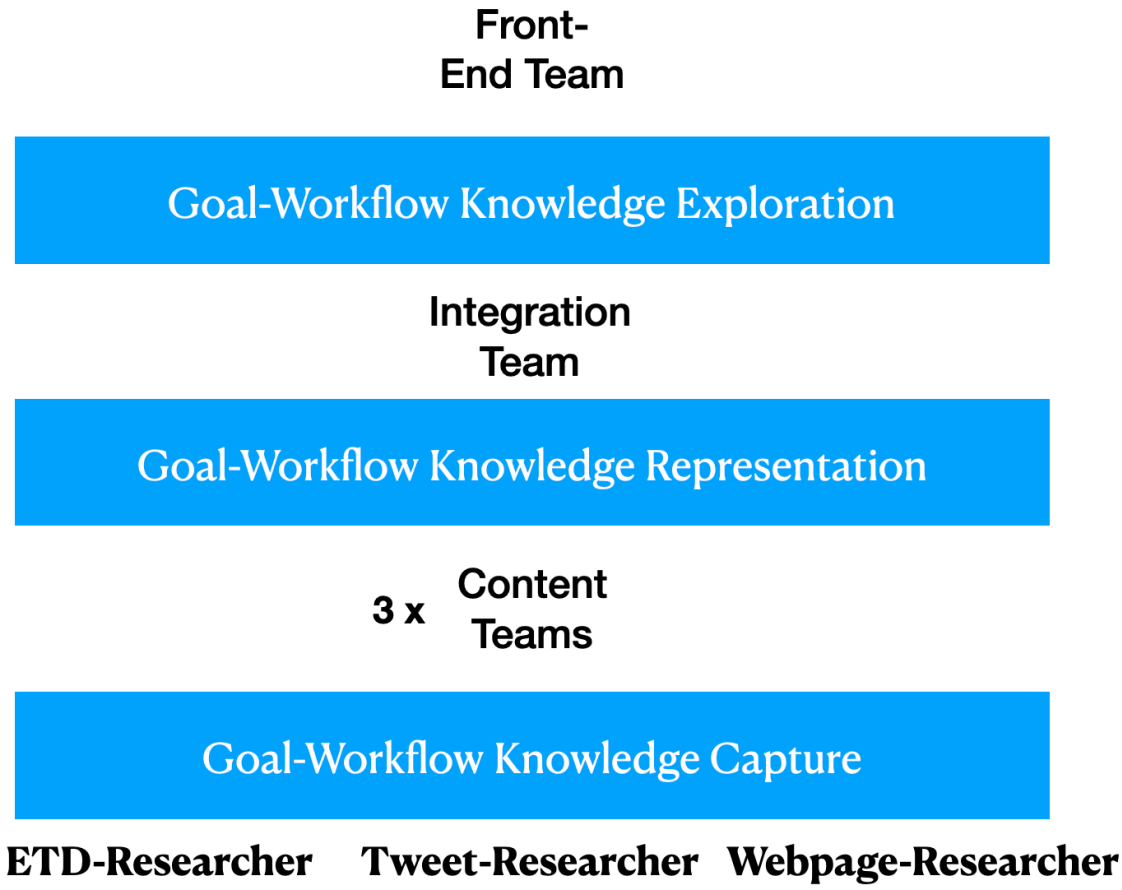


Figure A.2: CS5604: Teams and how they interacted with one another

Figure A.2 showcases how the teams from CS5604 collaborated to build the information system. The Content teams identified SME information goals and built a set of services that they deployed using Docker. The Integration team used a container cluster to host these services. They additionally stored information on how these services connected with one another (please see the Services and Service Registry (bullet number 3) below for more details). Finally, they developed APIs that would allow for querying user goals and executing workflow services. The UI researcher interface built by the Front-end team called these APIs.

We will go through each component that the student teams built.

1. User Interface (UI): The CS5604 class had a team of students working on building the front-end interface for the system. The system would have two types of interfaces. One type of interface would support the search of documents indexed via ElasticSearch. The other was for curators and researchers to execute data-mining workflows. In this chapter, we focus on the interface built for curators and researchers.

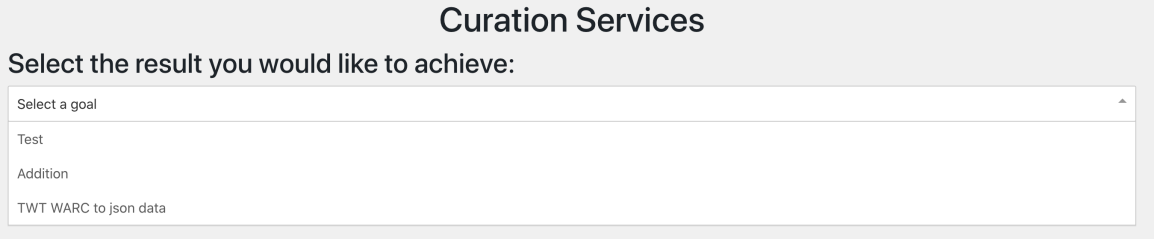


Figure A.3: UI Interface dropdown showing the list of SME goals that the researchers can select. [7]

   The researcher interface starts with a dropdown box (as shown in Figure A.3). The UI forwards the researcher's selection and displays the generated workflow to the user (Figure A.4). From then on, the UI allows SMEs to either execute the entire workflow or steps of it. The team built this interface in React.js. The next component, the Reasoner, is responsible for supporting all UI requests.

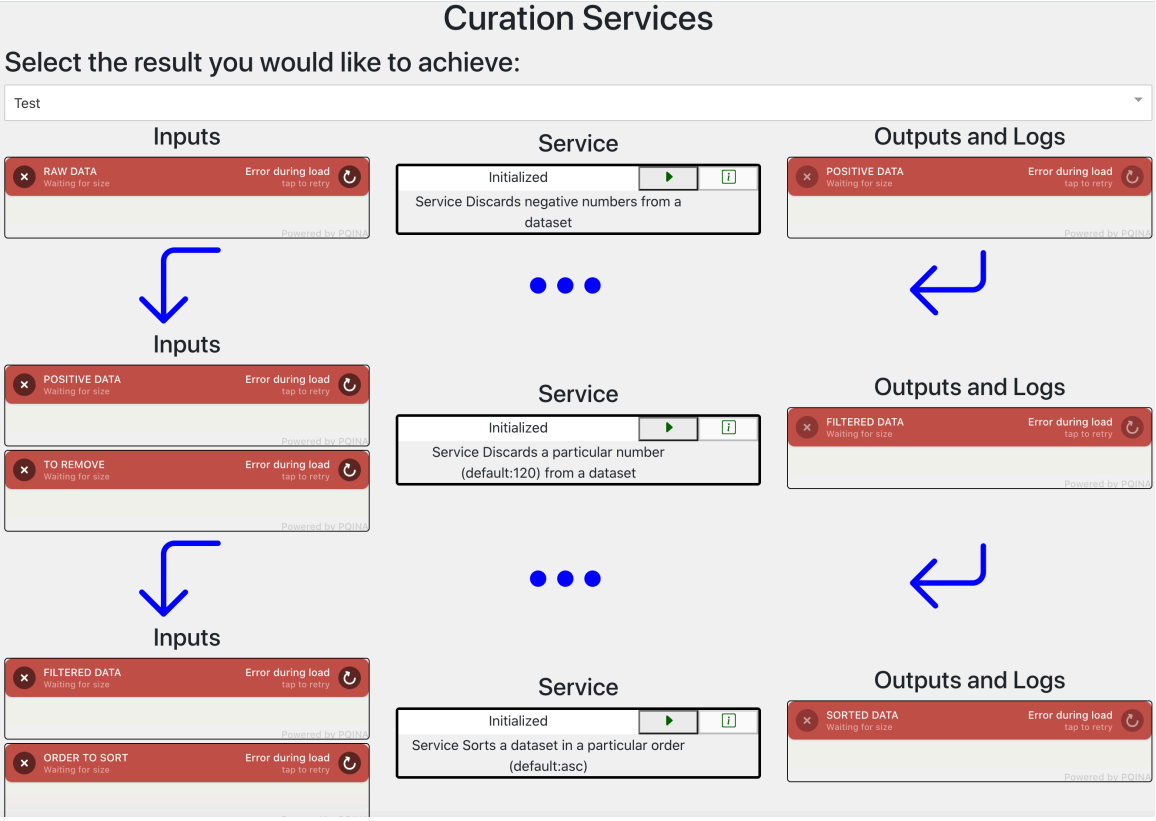


Figure A.4: UI Interface showing the workflow generated based on the user selection. [7]

2. Reasoner: The Reasoner serves as a middle-ware component for this information system.

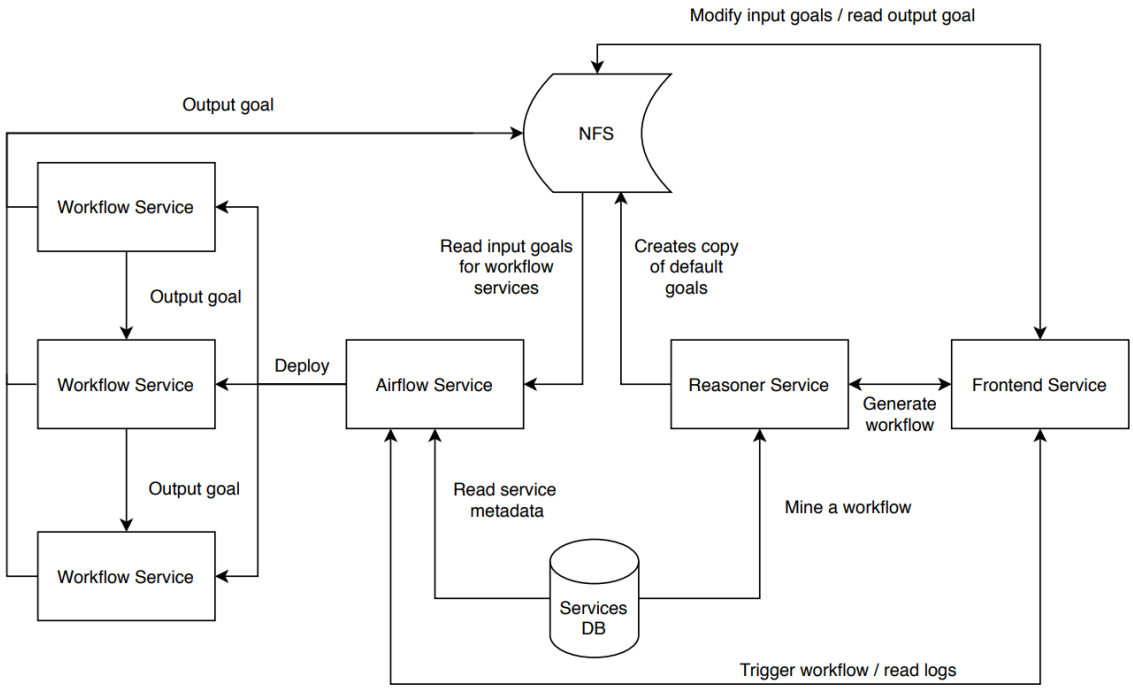


Figure A.5: Components of the information system and how they communicate with one another when responding to the UI. [34]

Figure A.5 shows how the Reasoner service connects with the UI and the rest of the components of the information system. The Reasoner provides two APIs:

(a) /generateGrammar: This is an internal API called upon whenever any new service is registered. Services and the tables for services are explained below. A new context-free grammar is generated every time a new service is registered. The second Reasoner API then uses this generated grammar to generate a workflow.

(b) /generateWorklow: This API endpoint takes in the SME/researcher information goal as input and generates a workflow based on the production rules generated using the previous endpoint. It also generates an "Airflow document" which repre-

sents the sequences of services to execute by the workflow manager. The workflow manager, Apache Airflow, takes this document and executes the sequence of services. The response is saved in a location in file storage, which is shared with the UI for the SME/researcher to locate and download.

3. Services and Service Registry: The Content teams interviewed the SMEs and broke down the goals into tasks and sub-tasks via Steps 1 and 2 of the methodology. They then built a service to support each task and package it via Docker. Following that, the Content teams uploaded the images to a container registry. Finally, they provide details of the images that were stored in PostgreSQL tables representing the Service Registry.

   Figure A.6 shows the three tables that hold the information about the services and the (sub-)goals they each support. The Reasoner uses these tables when generating the grammar and, as a result, when generating a workflow. You can find examples of entries to these tables in the report done by one of the teams [15] (Tables 7, 8, and 9).

   The ContainerCluster team in the Spring 2021 class was responsible for moving the containers created as a part of the CS5604 team efforts (in the VT CS cluster) to the container cluster in the DLRL and to be managed there via Kubernetes. Figure A.7 shows the final structure of the DLRL cluster. The migration involved two steps: First, run the images in the VTCS cloud locally, and secondly, migrate them to DLRL cluster. The effort also included creating and providing access to persistent volumes for the containers to safely access and store data. You can find details of their effort in their class report [30].

4. Apache Airflow (Workflow Manager): As mentioned previously, the Reasoner constructs workflows and represents them in an "Airflow document."

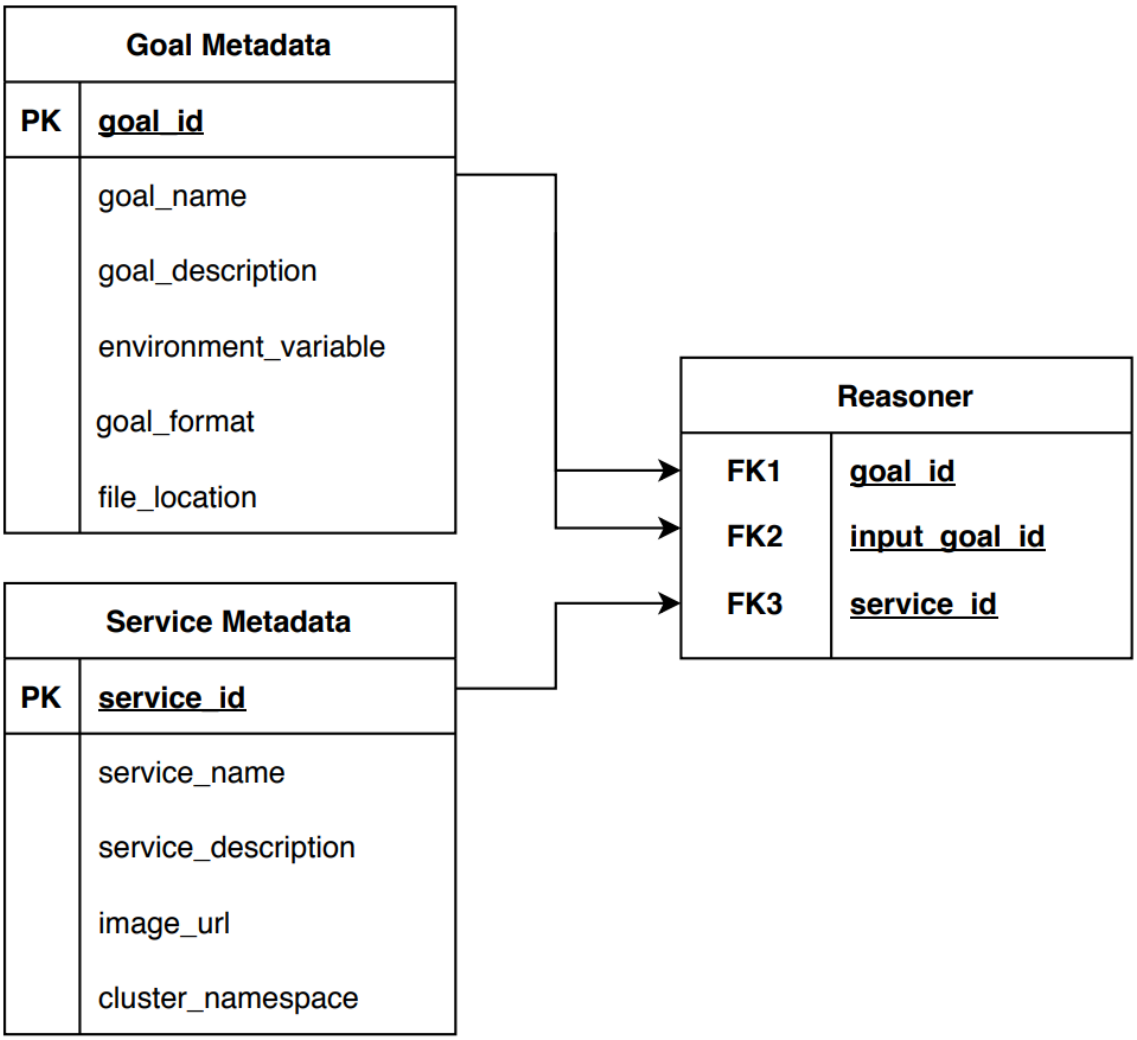


Figure A.6: ER-schema describing the Service Registry [34]

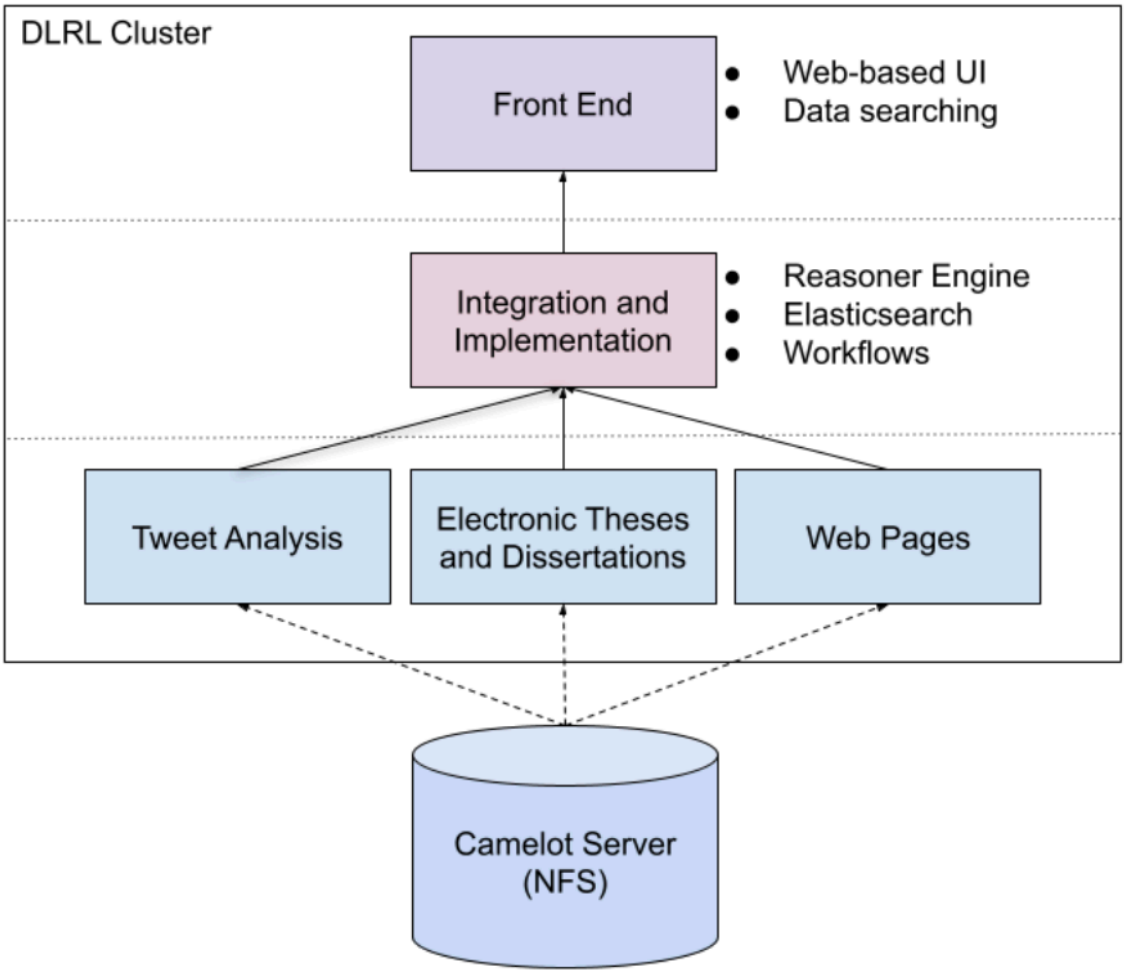


Figure A.7: DLRL Container Cluster build by CS4624 ContainerCluster team. [30]

```
{
   "requestedGoal":"6",
   "workflow":[[1, 2, 3]],
   "workflowId":["ZVpeMlEH"],
   "serviceMetadata":[[{
            "service_id":1,
            "service_name":"service1",
            "service_description":"Discards negative numbers from a dataset",
            "image_url":"container.cs.vt.edu/cs-5604-fall-2020/int/team-int-repo/service1:latest",
            "cluster_namespace":"cs5604-int-test",
            "owned_by":"INT"
         },
         .
         .
      ]
   ],
   "goalMetadata":[[{
            "goal_id":1,
            "goal_name":"goal1",
            "goal_description":"RAW DATA",
            "goal_format":"<Text file>",
            "file_location":"/mnt/camelot-cs5604/int/dataset.txt",
            "environment_variable":"RAW_DATASET",
            "owned_by":"INT"
         },

         .
         .
         .
      ]
   ],
   "workflowMetadata":[[{
            "goal_id":2,
            "service_id":1,
            "input_goal_id":1
         },
         {
            "goal_id":4,
            "service_id":2,
            "input_goal_id":2
         },
         .
         .
         .
      ]
   ]
}
```


Figure A.8: Document produced by the Reasoner after generating a workflow. This document is passed on to Apache Airflow for execution. [34]

The document is executed, and the workflow is scheduled and monitored using Apache Airflow. Apache Airflow is an open-source platform built using Python to schedule, manage, create, and run workflows [2]. Workflows are represented as Directed Acyclic Graphs (DAGs). Each node in the graph represents a task to be performed in the workflow. You can learn more about the core components of Airflow in the report by the CS5604 team at [34] (Figure 4). Figure A.8 shows an example Airflow Document generated by the Reasoner. The information in the document contains all the information that Airflow needs to spawn containers to initiate execution [28].

Appendix B

Appendix B: GRAKN.AI Knowledge Graph and Visualization

As described in Appendix A, the students' efforts in CS5604 helped build a workflow-based information system. But, there was still some work remaining. More specifically, the system built in CS5604 did not allow developers to automatically register the services they have created and uploaded to a Service registry. One could only manually add new services and information into the Service Registry. We needed to ensure that we can update the Service Registry and the Knowledge Graph. The main focus of the efforts for the Spring 2021 CS4624 Knowledge Graph team was the following:

1. Build a knowledge graph that represents the relationship between information goals and services;
2. Build a search API for users to be able to search the entities in the Knowledge Graph;
3. Build an API that generates a path between two information states; this path would represent a workflow or a sequence of transformations or services; and
4. Build an API for users to add information goals and for developers to add services connecting them.

The student team used GRAKN.AI to build the knowledge graph database [56]. GRAKN.AI

is a scalable knowledge graph that allows us to express hyperedges and thereby helps us build a hypergraph, as we conceived in our research.

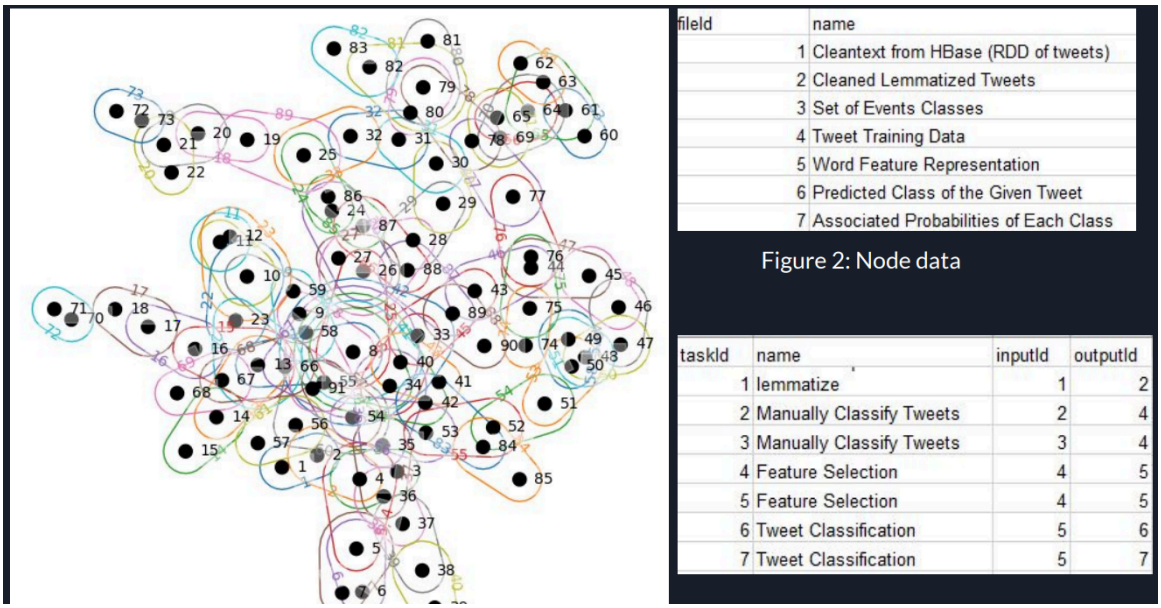
Figure B.1 shows the schema for the knowledge graph. GRAKN.AI allows its users to create two types of nodes: an entity node and a relationship node. As shown in the schema, an information goal or a "file" node is an entity node. A service or a "task" node is a relationship node. GRAKN.AI allows us to assign two "roles" for the file nodes: an "inputfile" and an "outputfile." Following that, in the schema definition of the task node, we relate both the inputfile and the outputfile. It is as simple as that. With that definition, we can establish connections between information states and services that transform one state to the next. The schema is a little light when it comes to the metadata of a file and a service. This schema is flexible and will be modified in the future to have all the fields that were present in the Service Registry built by the CS5604 team. With this schema, the students loaded the knowledge graph database with tasks and files that the CS4624 Spring 2020 team had identified after mining the efforts from the GETAR project [42]. The team was able to load this data using an API that they made available as a part of the deliverable. They built a search API that would return a set of files and tasks that matches the query string. Given the scenario that an SME researcher wants to derive an output file given a specific input, the team built an API that would take these criteria and generate a path of services or tasks that connect the two information files.

As mentioned previously, the team had built an API for developers to add new tasks or services to the knowledge graph. But, given that it is a graph, the developers would need to know if they are adding a task to an existing component of a graph or a disconnected component altogether. For that purpose, they would need a visualization of the graph in its current state. The development of this visualization defined the second half of the team's effort. The team explored various graph visualization packages and ways to represent

```
define

file sub entity,
    has name,
    has fileId,
    plays inputfile,
    plays outputfile,
    plays parent,
    plays child;

task sub relation,
    has name,
    has taskId,
    relates inputfile,
    relates outputfile,
    plays parent,
    plays child;

instanceOf sub relation,
    relates parent,
    relates child;


name sub attribute,
    value string;

fileId sub attribute,
    value long;

taskId sub attribute,
    value long;
```

Figure B.1: Schema for the GRAKN knowledge graph database. [59]

hypergraphs. They finally employed the HyperNetX library that provides the classes and methods for visualization of complex network data.



fileId	name
1	Cleantext from HBase (RDD of tweets)
2	Cleaned Lemmatized Tweets
3	Set of Events Classes
4	Tweet Training Data
5	Word Feature Representation
6	Predicted Class of the Given Tweet
7	Associated Probabilities of Each Class

Figure 2: Node data

taskId	name	inputId	outputId
1	lemmatize	1	2
2	Manually Classify Tweets	2	4
3	Manually Classify Tweets	3	4
4	Feature Selection	4	5
5	Feature Selection	4	5
6	Tweet Classification	5	6
7	Tweet Classification	5	7

Figure B.2: Visualization of the hypergraph knowledge graph via HyperNetX library. [59]

Figure B.2 shows us a hypergraph visualization. More specifically, it shows us the nodes and hyperedges of the GRAKN.AI knowledge graph database. Typically, a developer would come in and look at the graph and then use an interface to add their task or service to be a part of the graph.

Currently, the visualization does not have an interface for the developer to add their task or service. That is where the connection between the visualization and the knowledge graph is missing. In terms of future work, the first step would be to connect the two and build a form-based interface for developers to add services to the graph.