Epidemiology Experimentation and Simulation Management through Scientific Digital Libraries

Jonathan P. Leidig

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, Co-Chair
Madhav V. Marathe, Co-Chair
Pete H. Beckman
Keith R. Bisset
Ali R. Butt
Henning S. Mortveit

July 20, 2012
Blacksburg, Virginia

Keywords: Computational Epidemiology, Digital Libraries, Simulation and Experimentation Management, 5S Formalisms

Copyright 2012, Jonathan P. Leidig
Advances in scientific data management, discovery, dissemination, and sharing are changing the manner in which scientific studies are being conducted and repurposed. Data-intensive scientific practices increasingly require data management related services not available in existing digital libraries. Complicating the issue are the diversity of functional requirements and content in scientific domains as well as scientists’ lack of expertise in information and library science.

Researchers that utilize simulation and experimentation systems need digital libraries to maintain datasets, input configurations, results, analyses, and related documents. A digital library may be integrated with simulation infrastructures to provide automated support for research components, e.g., simulation interfaces to models, data warehouses, simulation applications, computational resources, and storage systems. Managing and provisioning simulation content allows streamlined experimentation, collaboration, discovery, and content reuse within a simulation community. Formal definitions of this class of digital libraries provide a foundation for producing a software toolkit and the semi-automated generation of digital library instances.

We present a generic, component-based SIMulation-supporting Digital Library (SimDL) framework. The framework is formally described and provides a deployable set of domain-free services, schema-based domain knowledge representations, and extensible lower and higher level service abstractions. Services in SimDL are specialized for semi-structured simulation content and large-scale data producing infrastructures, as exemplified in data storage, indexing, and retrieval service implementations. Contributions to the scientific community include previously unavailable simulation-specific services, e.g., incentivizing public contributions, semi-automated content curating, and memoizing simulation-generated data products. The practicality of SimDL is demonstrated through several case studies in computational epidemiology and network science as well as performance evaluations.
Acknowledgments

I wish to thank my co-advisors, Dr. Edward A. Fox and Dr. Madhav Marathe, for the constant dedication, direction, and support they have provided. They have given me a framework for discovering my academic interests and a means to completing my research. I reflect in amazement on the expertise, knowledge, external networks, and time you have shared with me. I thank you for all you have taught me regarding academia, research, teaching, writing, and life. I thank Drs. Keith Bisset, Ali Butt, and Henning Mortveit for demonstrating and instilling a passion for excellence in research and development. I thank Dr. Pete Beckman for hosting an exciting summer of research and the chance to appreciate life at ANL. Thank you all for devoting significant time towards my career and dissertation. I am grateful for our association and the opportunity to complete my doctoral studies, working with you.

I had the privilege of working for five years alongside two wonderful laboratories at Virginia Tech. I thank all of the students, staff, and faculty associated with the Network Dynamics and Simulation Science Lab (NDSSL) and the Digital Library Research Lab (DLRL). I appreciate all of the time spent collaborating, discovering, researching, and writing papers. I also thank the Department of Computer Science; DTRA R&D Grants 1-09-1-0017, 1-11-1-0016, and 1-11-D-0016-0001; NIH MIDAS 2U01GM070694-09 and 3U01FM070694-09S1; NSF PetaApps OCI-0904844, NetSE CNS-1011769, and SDCI OCI-1032677; and Swiss TPH for financial and travel support.

I thank my friends Gerrit, AJ, and Devin, along with close friends and academic colleagues too numerous to enumerate here, for forging a common path through life in Michigan and Virginia. It would be unfair to only list a subset of you all here, while I would need to write a book to fully thank you individually. I thank these companions, some for sharpening my faith, some for sharing my passions, and some simply for sharing themselves.

I thank all of my parents, siblings, and grandparents for the decades of love, instruction, sacrifices, and time you have devoted to me. Thank you for believing in my abilities and eventual success. You have imparted what it means to be an honorable professor, family member, servant, and witness. Thank you for growing and growing up with me.

Finally, I thank my wife, Karly, for bending her life and plans to mine, sharing my burdens, and joining in my accomplishments. You have been the rock that supported all of my efforts in life and academia. Thank you for your patience, understanding, and tireless work beside me. Ultimately, I attribute my accomplishments to my faith and daily walk with Jesus Christ.
Dedication

To my wife for her unfailing faith, hope, and love; to my parents for their direction and purpose; and to my siblings for their unwavering confidence and support.

This dissertation is mine only because you lifted me high enough to reach for it.
# Contents

1 Introduction 1

1.1 Computational Epidemiology Simulation 2

1.2 Problem Formulation 3

1.3 Digital Library Overview 4

1.4 Justification of Effort 5

1.4.1 How is it Different from a Database? 5

1.4.2 Why is a Digital Library Needed? 6

1.5 Hypothesis 7

1.5.1 Objectives 7

1.5.2 Thesis 7

1.5.3 Research Contributions 8

2 Related Work 10

2.1 Researchers, Users, and Roles 11

2.1.1 Users, Researchers, and How to Understand Scientists 11

2.1.2 Education 13

2.2 Notable Related Projects 13

2.3 Institutional Digital Libraries 15

2.4 Scientific Data Management 17

2.4.1 Concerns 18

2.4.2 Scientific Content Repositories 18


<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4.3 Standards</td>
<td>19</td>
</tr>
<tr>
<td>2.5 Summary</td>
<td>21</td>
</tr>
<tr>
<td>3 Computational Epidemiology and Network Science</td>
<td>22</td>
</tr>
<tr>
<td>3.1 Integrated Modeling Environments</td>
<td>22</td>
</tr>
<tr>
<td>3.2 Models and Software</td>
<td>22</td>
</tr>
<tr>
<td>3.2.1 Public Health Disease Models</td>
<td>23</td>
</tr>
<tr>
<td>3.2.2 Simdemics</td>
<td>24</td>
</tr>
<tr>
<td>3.3 Case Studies</td>
<td>24</td>
</tr>
<tr>
<td>3.3.1 Case Study: CINET DL</td>
<td>25</td>
</tr>
<tr>
<td>3.3.2 Case Study: Open Malaria DL</td>
<td>26</td>
</tr>
<tr>
<td>3.3.3 EpiSimdemics and Simstructure DL</td>
<td>29</td>
</tr>
<tr>
<td>3.4 Datasets</td>
<td>29</td>
</tr>
<tr>
<td>3.5 Metadata</td>
<td>30</td>
</tr>
<tr>
<td>3.6 Experiments and Studies</td>
<td>32</td>
</tr>
<tr>
<td>3.7 Need for SimDL</td>
<td>32</td>
</tr>
<tr>
<td>3.8 Simulation and Experiment Supporting Digital Libraries</td>
<td>34</td>
</tr>
<tr>
<td>4 Digital Library Foundations</td>
<td>37</td>
</tr>
<tr>
<td>4.1 Formal Definitions for Scientific Contexts</td>
<td>40</td>
</tr>
<tr>
<td>4.1.1 DL Management Systems</td>
<td>42</td>
</tr>
<tr>
<td>4.1.2 DL Systems</td>
<td>42</td>
</tr>
<tr>
<td>4.1.3 DL Instances</td>
<td>42</td>
</tr>
<tr>
<td>4.1.4 Takeaway</td>
<td>42</td>
</tr>
<tr>
<td>4.2 Background Definitions</td>
<td>44</td>
</tr>
<tr>
<td>4.3 Content</td>
<td>46</td>
</tr>
<tr>
<td>4.4 Societies</td>
<td>48</td>
</tr>
<tr>
<td>4.5 Scenarios and Services</td>
<td>49</td>
</tr>
<tr>
<td>5 Simulation Digital Library Framework</td>
<td>53</td>
</tr>
</tbody>
</table>
5.1 A Simulation Supporting Digital Library
   5.1.1 Computational Scalability
   5.1.2 Scientific Data
   5.1.3 Data Integrity
   5.1.4 Quality of Service
   5.1.5 Integration and Framework Extensions

5.2 The Software Architectural Overview

5.3 Service Organization

5.4 Interoperability
   5.4.1 Functionality, Services, Scenarios
   5.4.2 Users
   5.4.3 Architecture

6 Simulation-Specific Services
   6.1 Service Layers
   6.2 Novel Simulation-Specific Services
   6.3 Infrastructure Communication
   6.4 Curate
   6.5 Digital Object Collection Ranking
   6.6 Filter
   6.7 Get DLID
   6.8 Get Metadata
   6.9 Incentivize
   6.10 Log and Register
   6.11 Memoization
   6.12 Provenance Register and Resolve
   6.13 Search in Scientific Collections
   6.14 Supporting Metadata Functionality
      6.14.1 Generating Metadata Description Sets
6.14.2 Extracting Metadata Values .............................................. 88
6.14.3 Forming Metadata Records ................................................. 92
6.14.4 Indexing Metadata Records ................................................. 92
6.14.5 Summary ................................................................. 94

7 Evaluation ................................................................. 96
7.1 Methodology and Evaluation .................................................. 97
  7.1.1 Formal Descriptions ...................................................... 97
  7.1.2 Implementations ......................................................... 97
  7.1.3 Interoperability .......................................................... 97
7.2 Framework Evaluation ......................................................... 98
  7.2.1 DLMS Completeness ....................................................... 98
  7.2.2 DLS Software Toolkit Effectiveness ..................................... 99
  7.2.3 DLS Software Toolkit Efficiency (Service Performance) ............ 99
  7.2.4 Service Generality ........................................................ 101

8 Conclusions and Future Work .................................................. 104
8.1 Summary ................................................................. 104
8.2 Hypothesis Review ............................................................ 105
8.3 Future Work ............................................................... 105
  8.3.1 Functional Extensions .................................................... 106
  8.3.2 Cloud, Grid, and Cluster Supporting DL Services ................... 107
List of Figures

1.1 Information lifecycle for scientific content within SimDL. ................. 2

3.1 Overview of a model schema (left), input terms for a selected level of the schema (foreground), and input configuration file being generated (right). ............ 28

3.2 Digital object model. ................................................. 30

3.3 Structured workflow of content as produced by the research environments that SimDL supports. ................................................................. 30

3.4 Broader context of services and system components in SimDL. .......... 36

4.1 Conceptual simulation digital library universe based on the DELOS Reference Model using 5S-encoded formal definitions. ................................. 39

4.2 Metamodel of SimDL’s architectural design. ................................. 41

4.3 Definitional dependencies for the minimal simulation-supporting digital library. . 43

4.4 Representation of scientific document workflow. See [87] for additional information regarding the workflow......................................................... 46

4.5 Organization of the minimal simulation-supporting services and communication. 50

4.6 DL modeling and generation architecture guided by DL generation theory provided in [49]. ................................................................. 51

5.1 Prototypical simulation-based research infrastructural environment. .... 54

5.2 Information workflow in conducting and archiving an experiment. ........ 56

5.3 SimDL internal data flow architecture. ............................................. 59

5.4 SimDL component execution control flow. ........................................... 59

5.5 Possible science supporting services. .................................................. 63

5.6 SimDL service dependencies. .......................................................... 64
6.1 Organization of the content workflow, communication interface for content access and service exposure, low-level services, layered high-level services, and user communities. ................................................................. 68

6.2 5S graph of collaboration and related services. .................................................. 70

6.3 Message passing in a blackboard JINI-space coordinated SimDL environment. ... 71

6.4 Direct invocation of SimDL services in a SimDL infrastructure. ....................... 71

6.5 Metadata description set term organizational structure. ............................... 89

6.6 Metadata description set for input configuration files to the EpiSimdemics model. This XML representation is simply an alternative, concise translation from the RDF-triplestore used to store the metadata records. Each of these lines of XML is stored in a triple, e.g., (‘id:0’, ‘dc:creator’, ‘user:leidig’). ................................. 93
List of Tables

4.1 Informal service definitions [52]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
4.2 Formal service definitions developed from initial work proposed in [52]. . . . . . . 45
4.3 Standard low level formal symbols as described in [50]. . . . . . . . . . . . . . . . 45
4.4 Informal service definitions for simulation science. . . . . . . . . . . . . . . . . . 50

5.1 SimDL service and broker classes. . . . . . . . . . . . . . . . . . . . . . . . . . 61
5.2 SimDL service and broker classes (continued). . . . . . . . . . . . . . . . . . . . 62

6.1 Blackboard-based DLBroker service. . . . . . . . . . . . . . . . . . . . . . . . . . 70
6.2 Formal service definitions for curating (deletion of content operation). . . . . . . 73
6.3 DLCurate service. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
6.4 DLCurateDORanker service. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78
6.5 Formal service definitions for filtering. . . . . . . . . . . . . . . . . . . . . . . . . 78
6.6 DLFilter service. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78
6.7 Formal service definitions for getting a DL identifier. . . . . . . . . . . . . . . . . 79
6.8 DLGetDLID service. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79
6.9 Formal service definitions for getting metadata. . . . . . . . . . . . . . . . . . . . 79
6.10 DLGetMetadata service. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79
6.11 Formal service definitions for incentivizing. . . . . . . . . . . . . . . . . . . . . . 80
6.12 DLIncentivize service. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 80
6.13 Partial statistics in CINET incentivizing. . . . . . . . . . . . . . . . . . . . . . . . 81
6.14 Formal service definitions for logging. . . . . . . . . . . . . . . . . . . . . . . . . 82
6.15 DLLog and DLRegister service. ................................................. 82
6.16 Formal service definitions for memoization. ................................. 83
6.17 DLMemoization service. .......................................................... 83
6.18 Formal service definitions for provenance tracking. ......................... 84
6.19 DLProvRegister service. .......................................................... 84
6.20 DLProvResolve service. ............................................................ 84
6.21 Formal service definitions for searching. ...................................... 86
6.22 DLSearch service. ................................................................. 87
7.1 DLMS definition completeness. ................................................... 99
7.2 DLMS definitions and DLS toolkit comparison. .............................. 100
7.3 Evaluation of the DLBroker in milliseconds. .................................. 100
7.4 Evaluation of the DL services with time resolution in milliseconds (with small, medium, and large indexes). .............................. 102
7.5 Service deployment efforts for the CINET case study. ....................... 103
Chapter 1

Introduction

In the broad scientific community, there exist numerous large collections of data intensive experiments. In particular, the computing, biology, mechanical engineering, and physics domains tend to produce large datasets of computer generated output from experimentation systems. Acquiring statistically significant results from a single set of input parameters may require thousands of experiment runs to be conducted. Each of these runs may take a matter of seconds to several weeks to complete. In addition, the equipment, computing systems, human efforts, and other resources make experiments expensive to run and especially wasteful to rerun. This cost makes it difficult and expensive for experiments to be replicated if data is not preserved. Even for collaborators working in similar areas, replicating experiments with the exact same inputs, machines, and software tools may be immensely difficult or prohibited. In many cases, the exact environment and input settings are not recorded in a manner that would allow for an exact duplication of an experiment over long-term time frames. In some groups, data might be kept organized only while actively used and analyzed by researchers. In many domains and organizations, data are also very infrequently shared externally or managed in a manner that allows for repurposing. Funding and intellectual property concerns may discourage sharing of information with potential collaborators. Thus, few systems are in place for collaborators willing to work cooperatively on similar problems.

A digital library (DL) that makes extensive use of stored metadata on the methodology of data generation will allow enhanced reuse of existing experiments as well as the ability to reference them later. Although digital libraries for the scientific community exist, these systems primarily focus on storing published papers interpreting data instead of archiving data itself. In fact, there are numerous examples of archiving valued scientific content with less importance placed on storing the data that higher level content is based upon [32, 86, 100]. It is currently difficult for scientific claims made in high-impact papers to be verified. The SIMulation Digital Library (SimDL) is a theoretical framework, service toolkit, and practical instantiation that allows information in the epidemiology and network science simulation domains to be stored, managed, reused, and curated throughout a structured lifecycle, see Fig. 1.1. This is accomplished through the customized digital library which maintains information concerning contextual scenarios, underlying datasets, inputs,
results, analyses, and related content for each experimentation. [31] discusses the concept of an information ecology composed of an organization’s entire information environment. SimDL closely follows this concept by addressing the scientific workflows, user activities, information usage, data sources, information products, and information management priorities of simulation-based research groups in several domains. In addressing these concepts, a software instance deployment of SimDL provides the minimal functionality required to support a range of simulation-based scientific activities. In this work, I have produced a generic software toolkit that may be used to produce SimDL instances. This generic software may be utilized in multiple domains, e.g., epidemiology and network science, for data management services.

Figure 1.1: Information lifecycle for scientific content within SimDL.

1.1 Computational Epidemiology Simulation

Epidemiology encompasses the study of health patterns in a population and factors that contribute to health. Computational epidemiology involves the development and utilization of computer models to understand the spatio-temporal diffusion of diseases through populations. Animal health simulations require models for avian flu, swine flu, and mad-cow diseases. Human health
simulations include communicable diseases (e.g., influenza), vector-borne diseases (e.g., malaria), and immunopathology. Social network simulations include human manipulation, fads, marketing, and computer virus diffusions. These types of simulations are used to test public health policies, develop intervention strategies, and predict network diffusion end-states.

Organizations such as the Center for Disease Control and Prevention (CDC), Centre for Research on the Epidemiology of Disasters (CRED), European Centre for Disease Prevention and Control, Health Protection Agency, World Health Organization, and public health organizations in numerous countries study, among other things, the health state of a population and factors contributing to changes in the health state. Computational approaches can be applied for these studies to simulate how a disease disperses throughout a human population. Studies are conducted, in general, through the use of a simulation software tool that requires information on the population structure, agent behavior, disease transmission, and a model of the disease. With this information, a simulation may study the characteristics of the disease spread. These studies may be generalized to the simulation of the diffusion process of a contagion across a given network. The generalized approach allows software tools to simulate animal health, human health, human manipulation, computer states, and adaptive agent behavior. These simulation software tools require specialized datasets on the population of agents as well as parameters to the scenario-specific models. After a simulation, the set of results produced by the software tool may include information on the state of the simulation at given time points as well as the end state of the scenario. Public health officials and agricultural administrators rely on simulations of a scenario considering potential interventions to set public policy. Network science researchers may utilize the same approach to model the parallel scenarios of computer virus diffusion, transportation flows, or social network marketing. The results provided by computational experiments are used to set these health policies, define response strategies, and raise awareness of issues. To serve this purpose, managed experiment collections are needed to provide researchers with a mechanism to collaborate for the purpose of developing comprehensive solutions to modeled historical and envisioned real-world scenarios.

1.2 Problem Formulation

Recent collaborations between a set of computational epidemiology institutes have identified the need for digital libraries to support simulation infrastructures. Digital libraries may be leveraged to support data management, data provisioning, information satisfaction, user interface requests, user interactions, and computation submission functions for modeling and simulation systems. Simulations within the computational epidemiology domain allow public health officials to experiment and analyze potential policies for various contexts. This experimentation process makes use of a sequence of digital content types; simulation schemas, input parameter configurations, raw datasets, result summaries, analyses and plots, documentation, publications, and annotations. Content may be captured at each stage of the scientific process, forming stage-specific collections of digital objects. Different users are interested in generating and accessing content from specific collections in order to support specific user roles and tasks. Many standalone simulation applications
and online repositories interact with users through a highly customized, static web or command line interface for providing input parameters and returning results. Grid, cloud, and volunteer computational resources form the backend of computational epidemiology simulation and analysis applications. Digital libraries (DLs) may be incorporated into the simulation infrastructure to enable cohesive access to content and provide seamless functionality between system components. A DL front-end to a simulation system may automate: the process of conducting analyses; the generation of plots; organization and management of simulation-related metadata and experimental collections; dissemination of studies and findings; provision of simulation-related content; and a mechanism for collaboration between simulation researchers, software developers, study designers, analysts, and policy setters.

Formal descriptions of a DL framework, software toolkit, services, and content enable a semi-automated process to generate and verify digital libraries tasked with supporting a simulation system. Descriptions of each DL element must be clear as services (e.g., data management) are provided by a non-human, automated software infrastructure. By supporting the formal definition and rapid generation of simulation supporting digital libraries, we envision the semi-automated development of digital library systems; increased collaborations within and between research institutions; and effective management, discovery, reuse, and curation of existing content. Though our SimDL case study systems are within the computational epidemiology domain, domain and contextual information are encapsulated inside schemas to allow extensions into other simulation fields or non-simulation experimentation processes. As such, we interchangeably term this class of digital libraries as scientific, experiment, and simulation supporting digital libraries.

1.3 Digital Library Overview

SimDL is defined as a digital library tasked with at least minimally supporting computational epidemiology. SimDL is a fully formalized digital library management system with a complete software toolkit implementation. Digital libraries have been concisely stated as “a system that provides a community of users with coherent access to a large, organized repository of information and knowledge” [100]. In this particular instance, the community of users encompasses numerous computational and social scientists interested in computational epidemiology or network science for purposes ranging from the simulation of network dynamics to the refinement of public health policies.

Borgman et al. expanded the above definition to “digital libraries are a set of electronic resources and associated technical capabilities for creating, searching, and using information... The content of digital libraries includes data, metadata that describe representation, creator, owner, reproduction rights, and metadata that consist of links or relationships to other data or metadata, whether internal or external to the digital library” [16]. A vital aspect to SimDL is its ability to communicate with existing epidemiology simulation tools and infrastructures to create new content. Metadata is leveraged for linking a provenance sequence of a scientific workflow including input data,
simulation results, network analyses, generated reports, multimedia files, and publications.

Borgman et al. again refined the definition by adding: ‘‘digital libraries are constructed - collected and organized - by [and for] a community of users, and their functional capabilities support the information needs and uses of that community. They are a component of communities in which individuals and groups interact with each other, using data, information, and knowledge resources and systems. In this sense they are an extension, enhancement, and integration of a variety of information institutions as physical places where resources are selected, collected, organized, preserved, and accessed in support of a user community’’ [16]. Many communities would greatly benefit by having a digital library archiving the information generated through experimentation. In the modeling and simulation communities, there are many context-specific applications that could make use of such a digital library. Simulation environments typically funded by the DOD, DOE, and NIH include transportation and epidemiology, along with numerous other discrete event applications [9]. A collaborative group of individuals in each of these separate areas could make use of an archive of all simulations conducted through the use of SimDL. Development with the library-focused aspect of a digital library facilitates a transdisciplinary approach to bringing collections, activities, research efforts, and researchers together throughout the scientific workflow process [103]. SimDL strives to support collaboration within a transdisciplinary group with varied interests relating to communal content.

1.4 Justification of Effort

Archives provide the ability to utilize epidemiology experiments at later points of time to extract or repurpose information from the existing base of simulations. Since large-scale climate, epidemiology, or transportation simulations can take days or weeks to conduct, such a digital library could prove very beneficial in reducing resource consumption and wait time. The storage of the computational epidemiology experimental information yields benefit of reducing duplication of previous experiments and faster turnaround times for existing data. Even within small research groups, SimDL may reduce duplication of experiments caused by a lack of effective communication. Collections of epidemiology simulations allow for comparison and validation of real world data to states predicted by a model. This allows researchers to predict future real world states and fine-tune epidemiological models. The digital library promotes managing and sharing of scientific content across user roles along with dissemination to other researchers.

1.4.1 How is it Different from a Database?

SimDL provides high-level services over underlying collections of content. Databases are widely used to structure collections of data at a low level of complexity. Database management systems (DBMS) strive to maintain and utilize collections of data, while, resource planning packages add service layers to a DBMS [124]. In a similar fashion, digital libraries make use of underlying
databases and DBMS that provide the ability to access stored content in order to provide an extra layer of services exceeding simple storage. Further development of an initial storage structure into a service-driven repository is required when needs are not met by a simple structure or DBMS. Digital libraries provide the ability for users to develop representative metadata indexes linked to data object collections. Metadata indexes enable search for documents without a detailed knowledge of the entire collection and its structure. Composite digital objects are used to link content from multiple collections. At its lowest level, SimDL incorporates an underlying storage structure including databases and a DBMS (e.g., Oracle and MySQL). The two storage layers, content and metadata, provide the basis for low- and high-level services.

SimDL is designed to extend existing simulation systems. It interacts with simulation software to launch new simulations and stores returned results. With the necessary simulation input and result data structures, the system is capable of automating the capture of simulation-based experimentation. The data format and structure of an experiment determines the storage required to capture these simulations in the digital library. In addition to simulation data, the semantics behind the experiment can be automatically extracted and included in metadata records. This metadata allows the association of experimental data with higher-level analyses, graphs, plots, and reports. As additional content is manually produced, digital objects can be linked to the relevant set of experiments. Search is enabled over the data by determining with domain knowledge the similarity of a query to the existing range of experiments and identifying the set of potentially related digital objects. This approach requires the usage of customized database and filesystem storage for simulation content to reduce the storage footprint for large collections. To promote collaboration between researchers, contributors, study designers, end-users, and resource managers, memoization and incentivization reporting is provided in SimDL. In short, SimDL is not simply a retrievable collection of documents but a tool for bringing together collections, services, and researchers throughout the entire experimentation lifecycle of information creation, diffusion, usage, and preservation - a need suggested by [103].

### 1.4.2 Why is a Digital Library Needed?

Computational epidemiology requires a transdisciplinary group of researchers to solve problems. The approach generally requires systems modeling, simulation software construction, statistical analysis, and public health interpretation and validation. As experimentation is continuous, it is important to make all information on a given experiment available to the set of researchers working on various aspects of the scientific workflow. A digital library may be used to automate collection development, collection management, and distribution of scientific materials to collaborators. A simple storage structure would not meet the objectives as a digital library is required to provide support for higher level services and abstractions. SimDL meets these needs through its middleware services layer which makes use of the underlying storage system and domain information to provide insightful connections between objects in the collection. Existing DL software is not capable of supporting large-scale, complex data collections and providing science-based services.
1.5 Hypothesis

1.5.1 Objectives

Current socio-technical system modeling environments leveraging High Performance Computing (HPC) are complex systems for specialized domains. The use of these systems is primarily accessible to a set of highly specialized technical personnel fluent in the domain and usage of the software application. Widespread access to simulation and analysis content is enhanced with the inclusion of a digital library to the modeling infrastructure. The vision for SimDL is to extend an infrastructure’s existing information procurement and delivery mechanism to provide a wider set of researchers with access to HPC-models and analytical tools. Similar to the effect of search engines on altering research and analysis patterns, SimDL makes access to HPC resources seamless through content organization and accessibility mechanisms.

The goal of this effort is to provide collaborative access to scientific content in a comprehensive and meaningful way. The proposed digital library solution should adequately capture enough metadata information to allow for a set of experiments to be reproduced on a similar computing system. This information must contain the environmental conditions, software tools used, and input parameters for experiments. The results and analysis produced from a given set of experiments are linked with experimental information. The preservation of the above information allows services to be built around thorough investigations of the computational epidemiology domain. The SimDL framework has a domain-free design and software implementation. It targets the provisioning of services which automate human-intensive tasks regarding scientific content. We designed, implemented, deployed, and tested the SimDL framework to meet the objectives of the simulation community, prove our thesis, and answer related research questions.

1.5.2 Thesis

The hypotheses of this dissertation are:

- Simulation-related content, users, user tasks, DL services, and toolkit implementations can be defined in formal notation.
- The SimDL framework can provide efficient and effective automated services to support simulation-based content, users, tasks, research, and infrastructure, as specified in the formal definitions.

These hypotheses lead to research questions including:

- What constitutes scientific simulation content, communities, and anticipated tasks?
• What necessary services can SimDL provide that are absent or inefficient in existing DL software?
• How are the efficiency and effectiveness of SimDL services evaluated?

1.5.3 Research Contributions

The SimDL framework and approach taken provides a foundation for formalizing minimal needs within a scientific context, minimal software toolkit implementations for supporting several SimDL instances, abstractions for adding new services, and infrastructure for servicing requests. Research groups in other domains could use this approach to define, design, and implement similar DL instances in other fields based on the SimDL specifications described in this dissertation.

This research makes the following contributions to DL research communities:

• digital libraries described better than before (e.g., 5S implementation details, scientific & simulation context, and minimal epidemiology and network science requirements);
• formal 5S definitions for a class of scientific digital libraries;
• automation for scientific services and SimDL instance construction;
• existing services & collections for building additional services;
• definitions and implementations of novel simulation-based digital library services;
• efficient indexes of structured simulation content;
• metadata and curation processing grammars;
• potential DL instance offloading to virtualized HPC resources; and
• foundation for a simulation-services registry.

For the network science and computational epidemiology community, this work contributes:

• a deployable SimDL generation framework to support simulation infrastructures and work-flows;
• semi-automated metadata description set generation tools;
• fully automated metadata extraction and record generation tools;
• preservation, archival, and curation grammars and tools;
• partial data management and sharing plan fulfillment;
• multiple fully-fledged practical SimDL instances; and
• evaluation of the benefits and performance of SimDL for the scientific community.
Chapter 2

Related Work

Two decades of research in the field of digital library have been productive. Several open source digital library packages have been produced. Examples of these packages include DSpace[106], Eprints[139], Fedora[119], and Greenstone[59]. Each of these software packages have been utilized in hundreds of case studies across a variety of domains. In certain situations, such as full text document repositories, the existing software performs admirably. However, these DL environments are neither suited nor capable of providing DL functionality to support simulation science.

The digital library community is in the early stages of supporting the diverse contexts, efforts, workflows, activities, roles, and content found in the research community. The scientific data management community is broadly conducting research in data management, archival, access, sharing, and curation. However, progress towards a well defined simulation-supporting framework or open source digital library has been elusive. Producing SimDL required advances to existing digital library theory, approaches, and technologies. Historically, there have been few scientific-specific DL service implementations. Until now, there has been no available science-supporting or simulation-supporting digital libraries. In this work, we have produced SimDL, a generic, extendable, and formally-defined DL framework with the minimal set of simulation-supporting services.

In [70], Tony Hey describes an emerging paradigm of data-intensive science. Science has progressed to a data exploration age from previous empirical, theoretical, and computational paradigms. In recognition of this change, funding agencies such as NSF have now required data management planning [73]. Simple reuse of available open source DL packages is not feasible due to the increasing data collection limits and rate of science. This is demonstrated by the petabytes of data produced by CERN’s Large Hadron Collider, Australian Square Kilometre Array of radio telescopes, and astronomy’s panSTARRS array of celestial telescopes. New search algorithms are required that utilize technologies such as MapReduce for distributing metadata indexes and distributing computations. Beyond an assortment of science-specific services, a knowledge driven research infrastructure is called for. The end goal of these efforts is open science, or transparency in experimental methodologies, observations, and collections of data [12]. It is assumed that this
will eventually led to public availability and reusability of scientific data, public accessibility and transparency of scientific communication, and web-based tools to facilitate scientific collaboration. This work also claims that scientific research and data belong to and should be managed as a ‘bazaar’ from the village, cathedral, and bazaar analogy. Some progress has been made in data-intensive science, demonstrated in changing models of chemistry research revolving around data access and virtual collaborations. Historically, journals have recorded the ‘minutes of science’ through publication repositories [149]. This worldview has led to a system of dark data, or data that has been produced but is invisible and lost to science [67]. In data-supporting digital libraries, the goal is to record the methodology and content of scientific efforts, not simply records of ‘finished’ products.

Digital library research started in the early 1990’s at locations including University of California, Harvard University, Indiana University, New York University, University of Michigan, and University of Virginia [58]. Some of this was based on efforts in earlier decades, e.g., University of Michigan’s Michigan Terminal System, University of Colorado’s TAXIR system, and University of Michigan’s MICRO Information Management System. This work deviates from this tradition of DL theoretical research and DL instance delivery. Historically, DLs were delivered through the following process, a) planning development; b) purchasing hardware resources; c) planning network infrastructure and hardware; d) planning storage systems; e) developing systems and UIs; f) managing information, metadata, and document formats; g) building collections; and h) integration with other electronic services and functionality [35]. In this work, we are following an alternative path of formally defining aspects to a class of DLs, implementing a software toolkit from these definitions, and deploying case studies through the software. Our previous publications have presented background reviews on pertinent scientific data management systems, studies analyzing and assisting scientists, simulation environments, and formal reference models [87, 88, 117].

2.1 Researchers, Users, and Roles

2.1.1 Users, Researchers, and How to Understand Scientists

There have been several notable efforts and surveys to understand how to support science by investigating scientists, researchers, and research organizations. In [159], a survey of 400 digital library managers reported that 47.5% spent time creating metadata records; 40% developing policies; and 32% managing controlled vocabularies, thesauri, and taxonomies for a domain. 58% of digital librarians have major concerns with understanding scientific workflows, 58% with the quality of existing metadata structures and record formats, and 56% have concerns with how search functions can make use of metadata. Additional analysis identified significant concerns with controlling vocabularies, standardization, interoperability, extensibility, multilingualism, staffing, outreach, and training tools for metadata. A separate survey identified principal costs for groups that utilize DLs, i.e., .40 spent on commercial and produced content, .23 on equipment and infrastructure, .18 on DL personnel, and .07 on content creation systems [58]. Surveys into the patterns of information use in the life sciences have identified a large gap in the actual practices by scientists and the guidelines of
funding agencies [112]. This study may be utilized to improve understanding of information use and exchange in the life sciences, including epidemiology. These efforts inform the higher level evaluations of scientific digital libraries and support the development of a reusable SimDL.

There have been numerous case-studies specific to understanding researchers in data-intensive projects. [15] models individuals as data authors, managers, scientists, users, and funding agencies and describes the key responsibilities and practices of each group. The MESSAGE project was a scientific effort covering pollution sensors. [37] summarizes a case study of the data-related aspects of the work, e.g., datasets must be considered from a diverse set of researcher perceptions based on user roles and biases. [42] describes the research practices at the University of Michigan. The publication covers what types of data management activities researchers engage, how researchers think they could improve their practices, and in which DL services they are interested. [47] identifies roles that library science liaisons may engage in within an organization, i.e., analyzing dataset requirements, management planning, teaching practices to students, and setting preservation standards. [68] looks at the manner in which scientists and scholars are currently producing, storing, and disseminating digital objects; the drawbacks of these practices; and how libraries could better manage intellectual output. [77] takes a similar outlook on the dangers of scientists applying information management techniques in an ad-hoc manner as opposed to first making use of the traditional experiences of library information scientists in managing digital objects. FISHnet attempted to produce a freshwater information sharing network [135]. In FISHnet, researchers were studied to discover how they use and manage data as well as their concerns for data protection and copyright. [85] presents a case study of the receptivity of certain types of scientific researchers to library involvement in curation practices and the attributes that lead to receptivity. [101] argues that data repositories may have a larger impact than the institutional repositories that have not quite had the initially anticipated impact on changing research patterns across all domains. A number of UK-based research data repositories initiatives and aspects are highlighted. [102] presents experiences into understanding how user, i.e., researcher, participation in data curation efforts can be encouraged. [120] assessed current academic data management practices and discovered that researchers do not need physical storage capacity but assistance with funding agency requirements, grant proposal explanations, finding data-related services, and publication support. [84] presents challenges to librarian practitioners including content appraisal and selection, retention, description, preservation, authenticity, and compliance. [134] presents a series of interviews across 30 scientific projects to discover issues relating to the provision and adoption of e-infrastructures. [151] presents a study into how soft-support instead of additional technical solutions can be utilized to solve the ‘people-problem’ side of scientific data management. They found that researchers organize data in ad-hoc manners, store data without security and backup, are positive about sharing data but slow to do so, and equate backups with long-term preservation. [48] looks into how scientific researchers can be modeled and compared based on similar research activities. These case studies provide insight into the current gaps between the desire of researchers to preserve scientific content, suitable DL systems, and available training.
2.1.2 Education

Across multiple domains, there has been a systematic push to educate the next generation of scientists in data management and information sciences. Many of the current efforts have been funded by academia and organizations such as NSF and JISC. Mantra offers research data management training online at the Ph.D. level\(^1\). DataTrain provides data management training modules for post-graduate courses in Archaeology and Social Anthropology at the University of Cambridge\(^2\). DATUM provides research data management training for postgraduate students in the health studies discipline\(^3\). DMP-ESRC provides monitoring and planning for economic and social research in data-rich investments\(^4\). DMTpsych provides postgraduate training for research data management in the psychological sciences\(^5\). The Applied Quantitative Methods Network (AQMeN) aims to build capacity in the use of intermediate and advanced level quantitative methods amongst the social science community\(^6\). [115] describes areas in which information specialists can make contributions to research practices in Biology. In addition, a graduate curriculum for Biological Information Specialists is also presented. These efforts, and others, compose a laudable effort to shift perceptions and user expertise along with technological advancements.

2.2 Notable Related Projects

Previous work has detailed the notable scientific research data initiatives in Australia, France, Netherlands, UK, and Italy, and is briefly summarized here \([10]\). In Australia, PANDORA provides collaboration and coordination on collection policies as well a collaborative archiving. The National Library of Australia’s Digital Objects Management system manages long term preservation and access to digital content. There is also a national Australian Digital Resource Identifier scheme. In France, preservation technologies are developed by the Institut National de l’Audiovisuel. In the Netherlands, the national library, Koninklijke Bibliotheek, is conducting long-term preservation studies and has produced a digital deposit system for electronic publications (DNEP) with a goal of researching preservation. The Dutch Ministry of the Interior has produced a Digital Preservation Testbed. The Netherland Institute for Scientific Information Services has developed archives tasked with curation and providing access to primary research data in biomedicine, social sciences, history, and languages. In the UK, the British Library produced the Digital Library Store, DLS. The Joint Information Systems Committee Digital Preservation Focus (JISC) has funded digital preservation initiatives in academia and is responsible for many of the projects related to SimDL. The Data Archive for the Social Sciences has a history of storing research data back to the 1970s. The CAMiLEION research projects aims to provide emulation and migration as

---

\(^1\)http://datalib.edina.ac.uk/mantra
\(^2\)http://www.lib.cam.ac.uk/preservation/datatrain
\(^3\)http://www.northumbria.ac.uk/sd/academic/ceis/re/isrc/themes/rmarea/datum
\(^4\)http://www.data-archive.ac.uk/create-manage/projects/JISC-DMP
\(^5\)http://www.dmtpsych.york.ac.uk/
\(^6\)http://aqmen.ac.uk
preservation strategies. The Cedars research projects provide persistent identifiers to data. In Italy, PRESTO from Radiotelevisione Italiana provides preservation for audiovisual media and mediums. Internationally, the Networked European Deposit Library has defined preservation metadata for digital publications. The Electronic Resource Preservation and Access NETwork ERPA-NET projects were efforts in the preservation of scientific digital objects back in the early 2000s.

Efforts to enhance scientific data management practices have been a major focus for e-Science and cyberinfrastructure groups over the last several years. Major sponsors of research and projects in this area include NSF, ARL, and JISC. Several examples of scientific data management include earthquake simulation repositories [78], embedded sensor network DLs [18], community earth systems [38], D4Science II [23], mathematical-based retrieval [160], chemistry systems [96], national research data plans [82], and science portals [107]. Each of these projects attempt to provide e-Science software to support data management, arching, management, access, and sharing.

Several recent projects demonstrate the variety and current state of the art in scientific data management, preservation, and dissemination. FISHlink was a JISC funded project to demonstrate how diverse freshwater biology data can be combined, repurposed, and reused\(^7\). The project provided metadata attribution, provenance information, sharing data, and integration through vocabularies. A project in gravitational waves produced a data management and data sharing plan for big science, specifically astrophysics collaboration research groups [56]. The US National Virtual Observatory (NVO) is a large scale system that provides access to data and computing resources [114]. NVO makes it easy to locate, retrieve, and analyze data from archives and catalogs worldwide. Its goal is to make it possible for astronomical researchers to find, retrieve, and analyze astronomical data from telescopes across the globe. The project had success in building a large community of research organizations and professional scientists in this domain. It is one of the few projects that has built efficient services for massive data sources. However, the software and infrastructure utilized here has not been generalized and reused in the broader scientific community. The NASA Planetary Data System archives and distributes NASA’s scientific data from missions and observations\(^8\). This system has fully integrated datasets organized into nodes, subnodes, and data nodes distributed among dozens of organizations around the United States. Similarly, NASA’s Earth Observing Systems project provides thousands of datasets to the general public\(^9\). The ESO Very Large Telescope and Science Archive System additionally performs routine computer aided feature extraction from raw data and allows data mining on both data and extracted parameters\(^10\). Unfortunately, few other scientific domains maintain as much community support and governmental funding to develop substantive big science portals.

Several additional projects are worth noting as they each presented a small subset of the design decisions also used in the SimDL framework. According to Greenberg, “data sought for repository deposition is being generated at a much faster rate than metadata can be created for representing, organizing, and accessing data” [57]. Their approach to this problem lead to the development

\(^7\)http://www.fishlinkonline.org  
\(^8\)http://pds.jpl.nasa.gov  
\(^9\)http://eospso.gsfc.nasa.gov  
\(^10\)http://archive.eso.org/cms
of Dryad. Dryad is a repository that automatically generates DOIs, file description metadata, and Dublin Core terms. However, user information metadata generation is semi-automated and descriptions are manually acquired. This repository takes several approaches we reuse in SimDL, e.g., XML Schema metadata representations. However, Dryad does not provide the functionality required by the simulation community, e.g., fully automated metadata extraction. The National Geospatial Digital Archives produced guidelines for file formatting and archiving based on experiences in building large digital data archives from multiple data providers [40]. Specifically they describe issues such as versioning, file formats, handling proprietary formats, dataset size, ownership, compression, generating collection-level metadata, generating digital object (DO)-level metadata, and format registration. The Data-PASS project aims to systematically identify, archive, appraise, and acquire social science datasets for preservation purposes [62]. They were able to demonstrate that a community is capable of archiving large amounts of content within a set of loose partnerships when a medium for collaborating and managing data is present. [146] describes the differences in document repositories and data repositories and demonstrates how data repositories are required to capture, in addition to data, information about the data, its creators, overviews, and various aspects related to proof (e.g., data validity). [144] describes the Australian National Data Service, a large-scale framework that supports creation, capturing, managing, and sharing of research data. This is done through a collection registry, persistent identifiers, policies, schemas, metadata catalogues, harvesting guidelines, and search services. [17] details how ‘little science’, e.g., habitat ecology researchers, are beginning to use embedded sensor networks and DLs to capture, organize, and manage large numbers of DOs. The effort describes how data practices evolve with data-intensive generation mechanisms. The work also discusses curation and dissemination policies, lack of domain data standards, lack of interest in utilizing other groups’ data, and concerns about ownership and quality. [22] describes the DILIGENT approach to DL generation where basic services are combined into application-specific DL instances. [21] further elaborates by describing the integration of DL and grid technologies and resources within the Earth Science community. DILIGENT is notable in that it allows communities to build specialized DLs that support the whole lifecycle of content production and use in user defined workflows. Ongoing work in content-based image retrieval (CBIR) has produced algorithms for automatically extracting and indexing features found in multimedia, such as scientific data images. One CBIR project in fish species identification extended the 5S framework to provide a formal foundation for their work and allow for application-specific definitions of CBIR digital libraries, see [110, 111]. Our SimDL framework takes a similar approach and also builds upon facets of each of the other projects.

### 2.3 Institutional Digital Libraries

Digital library systems greatly differ in terms of their scope, functionality, and scalability. Digital libraries for individual units or projects are globally prevalent and domain-specific, e.g., a Geoscience department’s DL for managing rocks, minerals, and core samples [141]. These DLs are heavily tailored for a domain, provide complete coverage for required services, and are generally
not reusable in other areas. Institutional digital libraries attempt to provide general services for a wider community while not being able to provide domain-specific services such as image-based search, e.g., Purdue University Libraries. These DLs are typically found in universities where the goal is to provide a minimal level of research data management for faculty without the ability to support domain-specific metadata. In [156], the issues with institutional DLs are discussed, e.g., bottlenecks preventing community acceptance of a DL and lack of data sharing. The University of Edinburgh DataShare project provides a system of academic data sharing and management\footnote{http://datashare.is.ed.ac.uk}. The Institutional data management blueprint describes a DL for managing generic institutional data in higher education institutions [19]. [34] describes the University Digital Conservancy (UDC), the University of Minnesota’s DL. The UDC is charged with data stewardship, archiving, educating scientists, building scientist portals, managing data through policies, and gaining TRAC repository certification. The developers of UDC are attempting to support e-Science and data services through the university repository. It remains to be seen how these types of DLs are able to support the full diversity of scientific domains at a reasonable level of content granularity.

It is not entirely clear which human-intensive tasks institutional DLs must minimally support. [127] discusses the pressure on researchers to provide quality data management and sharing plans. It also discusses university reluctance to incentivize and support the generation of facilities for research data management and curation. Funding organizations wish to add value to expensive content, repurpose data across disciplines, produce transparency and reproducibility of published works, and release funded work in the public domain. However, the activities that lead to such a DL is still unclear for universities and individual researchers. [138] presents an extensive survey of approaches to e-Science and available services in both institutions and libraries. Multiple institutional DLs are then discussed in detail. One finding is that 9% of organizations were attempting to generate institute-wide e-Science DLs, 25% were focusing on supporting individual units only (e.g., departments), 61% were planning a hybrid structure of general institution-wide DLs with additional specialized unit DLs, and 49% were planning on collaborating with other institutes to produce a partnered DL that would be deployable at multiple institutions. With the SimDL framework, we produced a context-free unit digital library that could be used across domains (similar to an institutional DL) while allowing each context and infrastructure to directly dictate the domain-specific metadata schemas in use. The automation, or reduction, of institutional DL generation efforts by organizations without extensive resources or DL expertise is closely related to the Virtual Digital Library Generator [3]. The VDL Generator provides DL designers with a service to define a DL, produce a logical component plan, and yield deployment plans to be replaced with physical components. In comparison to the VDL Generator, SimDL concentrates on providing and integrating software components for formally defined individual services. We use formalisms to precisely characterize the DL to highlight the emergence of simulation-supporting services in our framework and in comparable related works.
2.4 Scientific Data Management

There are multiple domain-specific services and projects that were developed to support an aspect of scientific data management. Many groups have undertaken efforts to create, maintain, and utilize data management and sharing plans, particularly organizations associated with the Digital Curation Centre [36]. These data management plan efforts have produced a large number of tools for describing management plans but have not yet provided a consensus on quality tools in this area. The Data Management in Bio-Imaging project is similar to SimDL in that it attempts to support high-throughput bio-imaging systems [5]. The project had success in producing a ‘Bio-Imaging’ service that tracked images, software, software input parameters, and results, as well as linking the repositories of two scientific projects. While this effort parallels SimDL’s support of simulation systems, the minimal SimDL toolkit requires an expanded set of services not available in that work. The History Archaeology Linguistics Onomastics and Genetics (HALOGEN) pilot project attempted to support repurposing of data management for research data between humanities and genetics researchers [140]. The project included support for a content lifecycle similar to SimDL, albeit without the services required for simulation data. However, it provides a case study in joint data management plans and allowing collaboration between researchers in different domains. The Collaborative Assessment of Research Data Infrastructure and Objectives (CARDIO) project aimed to integrate data management plan tools. The produced toolkit integrates data management planning tools including DAF, AIDA, LIFE and DRAMBORA. This is related to our work in integrating individual scientific services into a larger toolkit but is concentrated on providing a user selectable data management planning service through multiple service implementations. The MaDAM and MiSS project aimed to provide a university-wide data management service and infrastructure initially integrated for Life and Medical Sciences researchers [121]. The service provides data capture, storage, and curation. The results for deploying the service for the Life and Medical Sciences lead to inputs for applying the service to other domains, but progress has not yet been made in applying the service to the whole university or across domains. The Supporting Productive Queries for Research (SPQR) project utilized linked data, RDF triples, and ontologies to allow multiple types of content and media, from classical antiquities, to be queried as complex objects [76]. There is a direct parallel with our scientific workflow generation of a stream of related digital objects. This project attempted to link DOs with similar content whereas SimDL links DOs based on provenance information from a structured workflow. The ARCHER project produced a tool for supporting sensors and scientific instruments that allows researchers to personally manage the collections of content they have generated and share content through workspaces [2]. While this data management package allows for the management of experiments by individual researchers, it does not provide the set of minimal services required to support backend simulation infrastructures. Karasti describes the LTER program’s approach to providing global curation services within communities willing to contribute to open data, e.g., long-term ecological research projects [80]. While these techniques, tools, and approaches have been produced, the DL community has not produced a generic toolkit for mashing these services and approaches. Additionally, services that

\[\text{http://www.dcc.ac.uk/projects/cardio}\]
are appropriate for one domain are not necessarily generalizable for science as a whole. As such, a methodology for precisely defining services is required to inform service selection in DL building efforts.

2.4.1 Concerns

Progress in developing a class of science-supporting digital libraries has been hindered by a variety of concerns and requirements from diverse scientific communities. A DL that is acceptable for one community might not address the concerns of other groups. There are a host of external threats to digital collections [130]. These include media failure, hardware failure, software failure, network problems, media and hardware obsolescence, software obsolescence, human error, natural disaster, attack by hackers, lack of funding to continue maintenance, and demise of the hosting institution. Content publishers often have concerns over confidential records, privacy, copyright, restricted access, data corruption, and data validity [79]. Harvey presents the following range of additional problems with data management systems [65]. Technology obsolescence complicates preservation services as computers and software are updated frequently and lead to the inability to access data. Technology fragility leads to inaccessible data when bytestreams are changed or corrupted. Curation is hindered by a general lack of understanding of what constitutes good practices, inadequate technologies, low organizational priority, skill-sets of scientists, and uncertainties about the best organizational infrastructures. Anderson presents an organization of a multitude of scientific data issues [1]. This organization describes domain-specific, organizational content management, content policy, and technical issues, and how these concerns form into barriers to developing generalizable scientific data management systems. The direct and indirect cost of managing and preserving data is difficult to estimate and depends on long-term forecasting of staffing, equipment, acquisition and deposit interfaces, file formatting, processing, metadata indexes, access processing, refreshing, versioning, copying, migration, storing, corruption monitoring, intensity of utilization, media replacement, and disaster recovery [11]. It is often unclear how to set policies in large-scale data production environments for content selection criteria, purging, advisory panels, and scope of access, e.g., remote sensed data [41]. Different domains have a variety of specific challenges and concerns related to instrumentation, content generation mechanisms, infrastructures, and acceptance of community standards [118]. DLs mitigate many of these concerns. However, DLs must be placed within the larger socio-technological environment to identify the aversion to and implausibility of DL integration in certain settings.

2.4.2 Scientific Content Repositories

A limited number of DL solutions are available to provide repositories for data underlying scientific publications. Common DL software such as DSpace, Fedora, Eprints, and Greenstone can be configured to use domain-specific metadata schemas and provide a generic store of an arbitrary attached digital object. However, these types of deployments are not robust in satisfying research
processes and scientific infrastructures with domain-specific extensions and modules. Dryad provides some support for data and metadata via Dublin Core application profiles and DSpace [57]. Several institutional repositories have been utilized for sharing content in small science efforts across multiple domains [27]. Project StORe provided a middleware for resolving articles found in output repositories with original and derived data located in source repositories. [123] The Purdue Libraries’ distributed institutional repository has been utilized in investigations of the relationships between raw data, structured data, publications, and scientific communities [156]. Universities in Australia have had some success in building repositories for raw data and collaborative research activities, e.g., ARROW, DART, ARCHER [145]. This work found that implementing multiple repositories is the best suited approach to separately support collaboration on research and publications. In the future of e-Science, multiple flavors of DLs and repositories are going to be needed to manage datasets, software, and publications based on the requirements of individual domains. Warner proposes an interoperability framework to join systems such as SimDL with aDORe, ArXiv, DSpace, and Fedora [152]. From these examples, we can surmise that scientific communities generally recognize the need for DLs of scientific content. It is unlikely that a single DL will be suitable for a broad range of potential scientific DL instances. The scope of many of these DL efforts is more restricted than our goals in producing a DL for context-free, simulation-based research.

### 2.4.3 Standards

While current efforts in the DL community do not indicate the emergence of a standard institutional DL in the near future, technologies and standards underlying scientific DLs implementations is increasing. Dahl provides a summary of web and metadata standards [29]. Sample web standards for automation include MARC to transfer records; XML markup languages to structure and communicate data; LDAP, Shibboleth, and Athens to authenticate users; OpenURL to transfer bibliographic information; and Z39.50, SRU/W, OpenSearch, and OAI for search and retrieval protocols. Metadata standards include XML style syntax; the Library of Congress’ MARCXML framework for expressing full MARC records in XML; Metadata Object Description Schema (MODS) for a simplified version of MARC expressed in XML; the Dublin Core and VRA core set of standard metadata elements; Encoded Archival Descriptions (EAD) for encoding archival types; and Metadata Encoding and Transmission Standard (METS) for an XML standard to combine descriptive, structural, and administrative metadata together within a single record. Rhyno details licensing, discovery services, and underlying constructs in DLs [126]. Licensing options include the GNU General Public License GPL, Creative Commons, GNU Lesser General Public License LGPL, Artistic License, Berkeley System Distribution BSD license, Apache Software License, MIT License, NCSA License, Mozilla Public License MPL, Netscape Public License NPL, and OCLC Research Public License. To describe metadata schema, Global Information Locator Service (GILS) describes a standard for organizations to describe collections. Content Standard for Digital Geospatial Metadata (CSDGM) provides terms and definitions for documentation of digital geospatial data. Learning Object Metadata (LOM) from the IEEE Learning Technology Standards Committee is used for education objects. The Online Information Exchange (ONIX)
defines publisher-generated metadata similar to the MARC standard. The Gateway to Educational Materials (GEM) is based on Dublin Core and adds controlled vocabularies for education including subjects, resource type, pedagogy, audience, and format. XMLFile from Virginia Tech is a tool that exposes a set of XML files in a directory as an Open Archives Initiative (OAI) data provider. XML authoring tools include Amaya, Pollo, Xerlin, KOffice, OpenOffice Writer, Cooktop, and Jedit. Popular underlying databases include MySQL, PostgreSQL, SAP DB, Interbase/Firebird, HSQL, Berkley DB, DB2, and Oracle. Object and XML stores include eXist, 4Suite, and Ozone. These technologies have been utilized in many DLs and provide general purpose mechanisms for storing, indexing, discovering, and sharing content.

Miller describes current metadata standards and practices [105]. Metadata is often described in terms of administrative, structural, and descriptive lines, where administrative metadata is composed of technical, preservation, rights, and usage terms. Common metadata services include identifying, retrieving, searching, browsing, metadata sharing, metadata harvesting, metadata aggregating (e.g., mapping and crosswalks between multiple schemas), metadata conversion and processing, and assessing quality. Metadata takes on several levels of granularity, including collection-level metadata in MARC as well as domain-specific, item-level terms. Efforts in defining a metadata description set may reuse of widely accepted, standard terms found in the Dublin Core Metadata Initiative, Library of Congress Subject Headings (LCSH), medical subject headings (MeSH), and the Dewey Decimal Classification DDC. Metadata schema can be analyzed based on accepted definitions of about-ness, of-ness, is-ness, facets, exhaustivity, specificity, and ambiguity of possible values. Ambiguity is often tackled through synonym rings, authority files, lists, taxonomies and classification, thesauri, subject headings, and vocabularies. Controlled provenance vocabularies defined by DCMI include is version of, has version, is replaced by, replaces, is required by, requires, is part of, has part, is referenced by, references, is format of, has format, and conforms to. Alternatively, MODs defines provenance vocabularies for preceding, succeeding, original, host, constituent, series, otherVersion, otherFormat, isReferencedBy, references, and reviewOf. Additionally, VRA 4.0 defines relatedTo, partOf, largerContextFor, formerlyPartOf, formerlyLargerContextFor, componentOf, componentIs, partnerInSetWith, preparatoryFor, basedOn, StudyFor, studyIs, and many more relations. To improve metadata quality and interoperability, the authors suggest organizations ‘‘1) use DC and other standard elements correctly, 2) include sufficient contextual information and access points, 3) enter data values that are machine-processable and linkable, distinguish administrative and technical metadata from descriptive metadata, 5) document your local practices.’’ Developing a metadata schema requires considering the schema’s context, defining functional requirements, developing element sets, identifying general terms and collection-specific terms, establishing controlled vocabularies and encoding schemes, developing content guidelines, documenting in a scheme or application profile, and creating a metadata registry or namespace. Widely used metadata practices include the use of linked data and RDF triplestores. Their work also notes the emerging differences in linked data, databases, digital collections, DLs, digital archives, digital repositories, institutional repositories, and content management systems.
2.5 Summary

These projects and SimDL provide services for scientific content generation, curation, representation, and management. However, SimDL focuses on structured simulation processes. SimDL addresses providing communication interfaces for external components, request submission, analysis submission, and data management for simulation systems. Additional workflows are integrated with SimDL through added DL components that make use of staged content (e.g., additional analysis processes or data mining may make use of existing datasets). These parallel e-Science DL efforts aim to bring together interoperable e-Infrastructures to generate worldwide virtual research environments. The focus of SimDL is to provide DL services related to computational technologies for individual researchers, institutions, and collaborators, not semantic integration of resources. The scope includes automated DL generation, and simulation related content production and management.
Chapter 3

Computational Epidemiology and Network Science

Modeling and simulation studies involve a statistical design, where a simulation model covering cells, or scenarios (i.e., input parameter configurations), of the design is executed. Conducting experiments consists of producing a model, developing software, generating a set of configurations, running the simulation application with a configuration on HPC resources, acquiring results, conducting analyses, and human interpretation of the results. Similarities exist in computational epidemiology simulation workflows that can be supported and partially-automated through a DL. My efforts in the application of digital library technologies towards the field of computational epidemiology were presented to the public health community through multiple conference presentations [89, 90, 91, 93, 94, 95].

3.1 Integrated Modeling Environments

As published in [8], modeling and simulation research is conducted in integrated technological environments. This type of environment is ideal for the utilization of simulation-supporting digital libraries that supports multiple models, software, data, and researcher roles. The work published in that article provided the impetus to develop SimDL. See [8] for the full description of these environments.

3.2 Models and Software

SimDL aims to support a diverse set of simulation applications. The initial case studies that guided the development of SimDL were primarily composed of computational epidemiology models and network science analyses packages.
3.2.1 Public Health Disease Models

Typical simulation models simulate the spread of diseases throughout human, animal, and plant populations. These simulations are used to predict the effects of potential strategies to mitigate the spread of an infectious disease to prevent epidemics. Differential equation-based models simulate a macro level view of epidemics. Agent-based modeling is used to provide detailed results at the high-granularity of individuals in the population. Three classes of computational epidemiology models that are initially supported by SimDL include contagious diseases, vector organism-based diseases, and bacterial infections within human tissues. The distinguishing features of these models stem from the type of disease transmission and propagation, e.g., person-to-person, vector-borne, and within-host.

Contagious Disease Models

The NIH Models of Infectious Disease Agent Study (MIDAS) research groups have produced a wide-ranging collection of infectious disease models. Several of these models use the EpiSimdemics [9] and EpiFast [14] simulation platforms. These two platforms have been used in multiple studies covering diseases that include H5N1, H1N1, smallpox, and pertussis (whooping cough). These diseases are spread by social contacts and environmental surfaces. The simulation models make use of synthetic populations of the United States based on census data, physical locations, and human activity modeling. Conducting a study using these platforms involves generating a population, implementing a disease model in software, developing an input configuration of required parameters, and executing the simulation software on large high-performance computing resources (e.g., clouds, grids, and clusters). Studies using these models are used by NIH and CDC policy makers to set mitigation policies during an epidemic (e.g., vaccine and antiviral distributions, school closures, and quarantines) [97]. Similar models consider the distribution of healthcare workers within given locations and the spread of diseases throughout the area [28].

Vector-Borne Disease Models

A parallel corpus of disease models, specific to malaria, have been produced through support by the Bill and Melinda Gates Foundation and many others. These vector-borne disease models include OpenMalaria [137]. This model also uses agent-based modeling to simulation the movement of individuals and their interactions with infectious vectors (i.e., mosquitoes). Studies using malaria models are typically conducted for the benefit of developing countries at high-risk to malaria. Public health officials in developing countries directly benefit from web-portals by reviewing public content produced in simulation systems. The findings of computational epidemiology researchers are captured through the digital library and may be used to guide pharmaceutical and healthcare delivery in developing and rural areas, similar to efforts in [39, 60, 122]. It is important to note that these disease models and simulations are based on observations, surveys, and datasets gathered in the field by local health officials. The number of disease models produced, along with versioning of
individual models, has led to the need within this group for generic interfaces, submission systems, and data management components.

**Immunopathology Models**

Within a human, models can be used to predict the inflammation and regulatory immune pathways as individual cells interact with foreign bacteria. The ENteric Immunity SImulator (ENISI) models immunopathology (e.g., ulcer, inflammation, diarrhea, etc.) considering microbes, mucosal immune responses, and multiple cell-types in tissues such as the gut [154]. This application can be used by mucosal immunologists to test and generate hypothesized mechanisms for clinical enteric disease outcomes given in vitro (laboratory) observations. Populations are represented by groups of individuals with the same cell-type and tissue site location. Each cell-type (e.g., macrophage, dendritic cell, CD4+Thelper cell) that occupies a specific immunological state is represented as a population whose individuals move among tissue sites. Each cell changes its location depending on type-specific rules. When in the same location, cells are considered in contact and may change immunological state depending on a specific set of rules for interaction. Individual movement and interactions are specified, the simulation is run, and high-level behaviors are observed. Experiments are conducted by modifying specifications, to represent specific laboratory conditions, and observing the effect on the net immune response. An example study using this tool reproduced a typical inflammatory response to B.hyodysenteriae as well as the immunopathological effects of autoimmunity to commensal bacteria. Analysis of simulations have highlighted key pathways by which chronic inflammation persists following B.hyodysenteriae elimination and epithelial cell death.

### 3.2.2 Simdemics

My research involving SimDL was initially started by the need to support the Simdemics modeling and simulation platform. These applications include EpiSimdemics [9], EpiFast [14], and EpiNet [25]. As published in my earlier work [7], the Simdemics class of modeling applications allows for a large body of scientific content to be produced after substantial efforts in developing multi-agent computational epidemiology models. In addition, a large number of network science algorithms and measures have been added to the research infrastructure that hosts Simdemics. This class of network science packages led to the production of a SimDL case study specific to network science.

### 3.3 Case Studies

The research has been practically demonstrated in case studies. Epidemiology studies are conducted by comparing the results of simulations constructed with slightly varying input parameters, i.e., scenarios. SimDL services were designed and implemented to support the workflows of content produced by studies conducted within the NDSSL [6] and Swiss Tropical and Public Health Institute
The architectural components in each of these digital library implementations include collections of diverse content, metadata record collections describing individual digital objects, a set of middleware services, metadata collection APIs, broker APIs, brokered connections to user interfaces, and brokered connections to other infrastructure components. These case studies are representative samples of simulation workflows similar to those seen in wet-laboratory management systems and high-throughput bioinformatics systems.

Development and extension of SimDL services was initially an effort between two computational epidemiology research groups at Virginia Tech, USA and the University of Basel, Switzerland. Each group required SimDL to be integrated in different types of environments, namely a cyber-infrastructure for multiple simulation models as well as a simulation infrastructure local to an individual institution. The diversity of disease models, available computational systems, and user requirements led to the development of a brokering system for DL communication. Several decades of experience by each group identified the minimal set of required services for a scientific digital library. The resulting SimDL has been integrated with varying levels of coupling to brokers and other infrastructure components. SimDL has proven useful in forming collections from simulation workflows for both environments. These collections have formed repositories of content useful for infrastructure components such as user interface presentation and graph analysis platforms. SimDL has since been integrated in case studies including CINET and OpenMalaria.

3.3.1 Case Study: CINET DL

One case study SimDL instance supports a cyberinfrastructure for network science research and computer science education (CINET). Researchers in many domains deal with large network graphs, e.g., bioinformatics, transportation, public health, and social sciences. Educators teach graphs and networks in many domains. In computer science, the core curriculum consists of several lecture hours regarding graphs and trees in the area of discrete structures. Many other departments offer graph theory courses along with other courses that deal with large network graphs. In addition, there are many graduate network science courses that use the same types of content for research and education. These types of courses require real-world, large-scale graphs, analyses algorithms, and computational resources. We have developed a cyberinfrastructure for network science education and research. CINET manages large network graphs, datasets, distributed network analyses algorithms (e.g., GALIB, NetworkX, and GDSCalc), and coursework materials. Educators can use the CI as a medium for conducting class projects, storing lecture materials, discussing best practices with other educators, and using pre-existing graphs. CINET also has a research and experimentation UI and portal that allows users to use CINET computing resources to conduct experimentation using scalable analysis software on existing network graphs. The cyberinfrastructure may be used as a laboratory material in order to use real-world, large-scale datasets in teaching and researching these topics.
DL Description

A SimDL instance was needed for information management, workflow support, and providing a variety of services. The cyberinfrastructure (CI) required services to manage network graphs, analysis algorithms, results, and logs of user behavior. APIs and brokers were needed to support connection to user interfaces and a blackboard system, i.e., Simfrastructure. Through the DL services, the CINET system is able to receive graph and software contributions from users, compute network analyses, reuse content, and incentivize the network science community. The aims for this community were to work collaboratively on generating a repository for datasets, conducting research, and guiding educational activities.

3.3.2 Case Study: Open Malaria DL

Modeling and simulation research groups in the public health domain often duplicate efforts in building highly similar simulation infrastructures. Study designers and policy makers using these systems infrequently have proper technical training in building properly formatted simulation input configurations, connecting to computational systems, moving datasets to appropriate systems, and invoking simulation models through command line interfaces. A generic job submission system may support these research groups by automating many of the technical, routine tasks required to conduct computational studies and allowing new models to be plugged-in to an existing infrastructure. The job submission system provides a generic user interface portal with connections to simulation engines, analysis processes, computational resources, and data management components. Schemas that describe simulation models’ input configurations are used by the interface to present simulation models to users and structure the design of experimental simulation studies. The submission system may be used to design studies, submit studies to the simulation system, gather results, execute analysis scripts, archive datasets, and present the results and analyses back to the user through the interface. Using simulation model schemas has allowed the submission system to support multiple malaria and infectious diseases simulation models while minimizing human-intensive efforts to connect new simulation models.

A generic Job Submission System (JSS) has been designed and implemented to provide the type of infrastructure desired by the malaria-focused public health community. The primary model initially supported by the JSS and digital library instance is described in [137]. The infrastructure includes generic components for exposing disease models, providing a user interface, building input configurations, designing studies, requesting and executing simulation, gathering results, launching analyses and visualization applications, and data management. The user interface makes use of a collection of schemas that describe individual simulation models to allow study designers to interact with disease-specific models. Data management is provided through a digital library that provides services to collect, index, manage, archive, curate, and search multiple collections of content as it is produced by the simulation infrastructure. The types of content stages included model schemas, input configurations, sub-configurations, etc. The required basic services include storage, management, access, browse, and retrieval. The collections are formed by capturing files
as they are produced by the simulation-based scientific workflow.

The JSS infrastructure consists of components for simulation model software, analysis software, computational resources, interfaces, and digital libraries. Each component uses brokers to communicate with the other components through the centralized blackboard. Entities may be added to individual components and directly utilized by the rest of the infrastructure. As an example, a new simulation model may be added to the model component and will automatically be exposed through the user interface, executed on compute resources, and managed by the digital library.

A centralized blackboard and component brokers manage the communication of messages and data in the JSS infrastructure. In addition to passing messages between the interface and digital library, the infrastructure also automates the experimentation workflow. This approach is based on existing systems outside of the public health domain that compose models into workflows [33, 99].

After receiving a simulation request, the brokering system is used to locate an available computational system with the appropriate simulation software. The input files are transferred to the identified system and the simulations are launched. Progress updates are sent back to the user interface as individual simulation runs are completed for progress monitoring purposes. After the simulation process has finished, results are transferred to result repositories for archival and forwarded to the analysis system. The raw result data is used by the analysis broker to invoke analysis software that have been pre-defined to support a particular simulation model. Modelers and policy makers generally do not have extensive technical background or training. The infrastructure’s automated workflow abstracts the need for novice users to fix formatting errors, transfer data to remote machines, and execute command line applications and analyses.

SimDL was designed to support the experimentation workflow for multiple models, including [137], within a reusable model-independent design. Fig. 3.1 displays the interface for gathering input parameters to a malaria model. Collaborating researchers had developed simulation systems which were complex for non-model developers to execute. The ontologies were found to aid in a model’s UI presentation through the generic browser. A model-independent UI was built to rapidly expose new disease models and present input, result, study design, analysis, and document collections. Brokers also were developed to support the UI in submitting input configurations, compliant with XML Schemas, to available computational systems.

**DL Description**

SimDL provides services including harvesting, indexing, searching, filtering, and archiving. Through blackboard connections, SimDL receives schemas, input configurations, subconfigurations, and study designs as they are saved by users. The content managed by SimDL resides in databases and file systems on computational platforms. The five types of simulation content produced by the simulation workflow, form the five collections managed by SimDL (e.g., schemas that describe simulation models). In addition, SimDL manages the full factorial or sampling study designs.
Sets of input parameters, or input subconfigurations, may be saved to the digital library. This allows domain experts in one field to contribute high-quality blocks of parameters that may be reused by novices without training in all areas of the modeling domain. As an example, malaria models require sets of parameters for human populations, mosquito vectors, disease characteristics, monitoring processes, and public health interventions. Thus, entomologists, social scientists, and local health officials may collaborate asynchronously in designing high-quality studies.

A study consists of multiple full-factorial experiments that include a base file, multiple parameters to investigate, a list of values for each parameter, and list of previous subconfiguration input values to reuse. After a study design has been developed, the interface verifies the correctness of the parameter values, generates the input files, and submits a simulation request to the blackboard system. Additional components will receive the simulation request and automate the process of producing results and generating the analyses and plots applicable to the model schema being used. The interface also provides a central portal for accessing the results and automated analyses.
As all of the information displayed in the interface is provided by the blackboard, generic or diverse customized interfaces may be accessed through different URLs and connected with the infrastructure. Lastly, language support for internationalization efforts are provided by a dictionary lookup for all interface terms and labels.

3.3.3 EpiSimdemics and Siminfrastructure DL

The simulation applications showcased in early sections are well suited for support by SimDL, i.e., EpiSimdemics [9] and EpiFast [14]. These applications are already integrated with the Siminfrastruc- ture architecture and brokering system. As SimDL has been integrated with the same architecture through a DL broker, supporting these applications is a logical extension of the existing efforts in producing SimDL case studies. In the Siminfrastructure environment, SimDL can provide services to store, manage, retrieve, and compare the sequence of digital objects produced through simulations. It also provides services such as searching, memoizing, incentivizing, reusing, reproducing, and discovering. Providing these information services would require replacing existing execution brokers with DL extended versions. The modified simulation workflow invokes appropriate SimDL services when processing certain events in each simulation workflow stage.

3.4 Datasets

The initial candidate datasets to be stored in SimDL center around epidemiological experiments related to the previously discussed models. This requires storage of the shared and individual simulation inputs as well as the entire set of outputs for each individual simulation. Going forward, any simulations conducted through SimDL with Siminfrastructure, the infrastructure supporting the Simdemics simulation software metadata are captured by prompting the user for a context-specific module of the specific scenario being simulated. A future version of SimDL may provide a connection interface to link any external experimental simulation software system repository to SimDL, but this generality is not pursued in the initial implementation. Stored datasets are investigated for indexing and to extract required metadata values. The location of a compressed derivative of each file in the dataset is then stored by a Siminfrastructure data manager and linked with the SimDL. On retrieval, each compressed file may be individually referenced and returned from the identified location. Computer generated experimental data are already in a digital format and make for suitable digital objects. This eliminates the need for digitization that often is a large part of digital library efforts. As the output of experiments will likely be quite comprehensive, the data may have to be compressed in some format. Some data, such as runtime system variables or artifacts, could possibly be curated in storage if it can be determined that it will not be used or referenced at a later date. Since the data is produced through some computing system, it is likely already in some structured organization of output, see Fig. 3.2 for a digital object representation. The information involved in scientific research processes may be formed into a structured stream.
of content. This stream is based on the research workflow that produces these digital objects, see Fig 3.3.

Figure 3.2: Digital object model.

Figure 3.3: Structured workflow of content as produced by the research environments that SimDL supports.

3.5 Metadata

Structured data resulting from an experiment can be stored cohesively as a single complex digital object (DO) file with attached metadata. This metadata partially takes the form of the system information necessary to rerun the experiment as well as additional variables that are useful in robust searching. To be specific, the metadata must capture information on the computing hardware the experiments were run on. In addition, information on the software used to run the experiment and the input parameters to the software system are necessary to conduct and differentiate specific experiments. Since various types of experiments are to be stored within the same digital library, the metadata must be sufficiently expressive to store all of the possible experiments. Hardware system
related information may be common across numerous experiment sets. Thus, it is primarily the
software and input information that must be carefully constructed though hardware may affect the
results in some distributed agent-based systems. This metadata section provides enough information
for a simulation scenario to be precisely duplicated a second time. Additional metadata is required
to allow search on variables not included with the input or environment variables. To promote
insightful search, context-specific metadata is used to capture scenario specific information into the
metadata.

An agent-based discrete event simulator, such as in Simdemics-based systems, may be capable of
running simulations in numerous contexts covering varying phenomena. For example, Simdemics
is capable of modeling public health epidemics [4] as well as the spread of malware in wireless
networks [25]. It would be undesirable to maintain separate digital libraries collections for each
countext and equally undesirable to force incompatible contexts into the same space. To maintain
numerous context-specific experiments within the same digital library, a context-specific module is
required to allow management of various contexts. This module contains the entire set of metadata
fields associated with a simulation scenario, including the input and environment variables. This
metadata module contains significant variables that are extracted from the input and output files.
These variables are defined within the context-specific module by the user group with domain
knowledge of the specific context. After the selection of input and output variables, additional
context-specific and context-free variables related to the contextual scenario are incorporated into
the metadata. These additional variables are primarily used to enhance the search functionality and
are not used for experiment duplication. For known and described modules, SimDL is able to store
context-specific cells and provide search functionality over the identified input and output variables.

The raw content generated by large-scale infrastructures is intractable to support in real time;
metadata plays a key role in providing DL services. Producing metadata records covering content is
done through automated metadata indexing in external data sources. The metadata indexes used
for high level services are all contained in RDF triplestores. Low-level services utilize RDBMS
for metadata representation. While metadata record terms are highly dependent on the content
being represented, I have defined a four-part metadata schema structure to be used for all metadata
records. The standard record structure contains Dublin Core terms, infrastructure terms, and formal
terms based on formal definitions.

By combining automatically harvested and user supplied metadata, SimDL is able to provide search
functionality over a context with a designer defined amount of metadata storage. Significant input
and output variables may be searched to match a user query with existing experiment sets. It may be
necessary for a module designer to alter the module design, and modification of the module scheme
may be required. With this approach, context-specific significant variables may be stored twice as
they appear in digital objects as well as in the metadata. After the metadata module development
process, a given metadata scheme would be capable of ‘indexing’ all data files related to a cell with
significant input, output, and user supplied variables for the particular cell’s contextual-scenario.
3.6 Experiments and Studies

As published in [6], computational epidemiology experiments were conducted to parameterize a simulation application and undertake predictive scenario studies. The effort was in the context of public health interventions in the course of an epidemic (e.g., anti-viral distributions and school closures). My efforts in this large study led to the identification and specification of digital library services needed to support this community.

3.7 Need for SimDL

A digital library supporting scientific domains must manage a sequence of content including large social network files, complex input configurations, geo-temporal output results of disease spreads, statistical plots, publications, and geographical diffusion video clips. Simulation groups in academia and national laboratories, using cluster, grid, and cloud resources, generally produce terabytes of simulation results when conducting studies in a variety of simulation domains. Their epidemiological research environments produce massive quantities of scientific, highly-numeric digital objects. The sequence of content scales from several gigabytes for small studies to terabytes for a national large-scale study. However, traditional DL services do not support simulation tasks. Data management is needed for content including public health research, medical and census records, synthetic populations, graphs and networks, and simulation workflows. Service algorithms and implementations cannot rely on previous work in developing full-text indexing and algorithms that utilize term frequencies, thesauri, dictionaries, and partial-matching scores. Simulation efforts in HPC often involve multiple experts from different domains and locations in collaboration on developing models, software, and experimental studies. An average modeling and simulation group might include computational experts, software developers, statisticians, mathematicians, medical or disease experts, entomologists, and public health officials. Only a select group of individuals will be able to connect to HPC resources, transfer data to necessary locations, design a correctly formatted input set, invoke the simulation model executable, and perform analyses. Web-interfaces and DL-automated human intensive processes assist in exposing simulation systems to non-technical users but often only provide a restricted subset of available simulation features and possible study designs. Modeling and simulation research groups are commonly hindered by the lack of sufficient digital library systems for simulation workflows and numerical research content. These groups frequently do not have personnel with expertise in data management and information retrieval. Non-uniform content management practices have hindered collaborations between institutions and failed to support cohesive efforts within labs.

As seen from the preceding examples and highlighted publications, large simulation systems, such as Simdemics, are capable of executing large, complex studies. Simulation models require large datasets (e.g., human population networks at the continental level), HPC systems (e.g., grids and clusters), and large storage to store generated content. Petascale storage systems are in place for a variety of domains including the internet archive, scientific research centers such as CERN, and
national laboratories. As petascale simulation applications become prevalent, scalable management and retrieval methods are required by research groups without prior experience or expertise in designing and implementing systems for this context. The parameterization of study scenarios is often a drawn-out, human-intensive process. Executing an array of scenarios with enough replications to achieve statistical significance often requires a non-trivial, costly amount of HPC resources. Research organizations and individual researchers generally do not have the desire to manage the entire workflow of data products systematically. Given the current set of available data management solutions, managing complex objects remains a difficult process. Organizations also must face the problem that too much data is generated to keep everything. Consistent curation is required at the institutional level whereas many organizations rely on arbitrary deletion decisions by individual data producers. The current class of digital library solutions does not support the information product preserving and repurposing functions required by large-scale simulation environments. SimDL is able to manage experiments and related content. The experimentation design process is streamlined by reusing portions of the existing corpus of input configurations. Backend processes may be automated with several SimDL services, e.g., memoization (using instead of recomputing existing results). Curation recommendations are provided based on rules parameterized for each collection by suggesting what should be curated, what curation activity should be done, and quantitative support for why the recommendation was made. Incentivization reports are generated for contributors of software, content, resources, and human-intensive efforts in multiple user roles. With this set of minimal services, SimDL is the first digital library system capable of supporting large integrated research environments.

As published in [87], SimDL aims to provide a generic, extensible component-based digital library. SimDL is customized by tool builders through domain-specific simulation schemas and extended through the addition of modular components. The utilization of schemas allows SimDL instances to support simulation applications for an unrestricted set of domains and contexts that have structured input requirements and a waiting simulation launching broker. Non-simulation scientific content may be handled by SimDL instances sans the simulation submission component (e.g., wet labs or instruments producing digital data points along with environmental input conditions from high-throughput methods).

SimDL was designed out of a need to automate the holistic management of scientific data produced by multiple simulation applications at multiple institutes. Modelers and simulation developers also desired fully automated deployment of a basic DL instance and low amounts of continued maintenance. The management of scientific data requires policy decisions for data preservation, curation, and distribution mechanisms within the digital library. Shared access to a digital library hosted by one institution allows privileged experts at multiple locations to directly and indirectly collaborate, communicate, cooperate, and coordinate research efforts through the DL. The automation of conducting simulation studies is accomplished by the integration of SimDL into an existing simulation infrastructure. Until scientific digital libraries become mainstream in the simulation community, DLs will likely continue to be integrated into a pre-established experimentation process and existing software system. Customization of the simulation-launching component may be required to provide an existing simulation engine with a functional set of inputs (e.g., transmitting XML input files to a
Simulation and Experiment Supporting Digital Libraries

As published in [87], my efforts have produced SimDL instances and services in several fields. Many institutions and research groups across a broad spectrum of domains make use of simulation applications to produce experimental data. Simulation efforts may benefit from employing a generic, extensible digital library that makes use of a domain descriptive schema to manage content related to the experimentation process. Simulation applications in the field of computational epidemiology are initially targeted due to interest by research institutions in this domain. The domain schema containing metadata concerning the required input parameters to the simulation application may be used to customize a generic digital library, interface, and data management process. Multiple schemas are likely necessary to represent the domain semantics of a simulation system for several contexts. The use of schemas to represent contextual information, such as input parameters, allows simulation supporting digital libraries to cover domains in addition to computational epidemiology. While data management practices for simulation related content varies greatly between domains and research groups, a generic digital library permits extensive reuse of prior DL development efforts to support structured simulation systems in a non-restricted set of eScience domains.

Efforts in developing SimDL aim to provide a generic management system for simulation groups that can be tailored to support individual simulation models [87, 88]. SimDL provides a generic component-based framework with services that are customized for a set of simulation models through the use of model ontologies. The scope of SimDL is to provide a DL that may be integrated in HPC cyberinfrastructure architectures or an individual lab with services for epidemiology simulation study management, organization, and search. In designing SimDL services for epidemiology, we have developed a scheme in which many of the services are likely to be useful for other classes of simulations.

Digital libraries can be integrated with public health research infrastructures to support this community. DLs can be used to manage existing simulation inputs and results to allow full or partial reuse of existing input configurations and results. Workflows can be developed within an infrastructure using SimDL to capture content as it is produced at each simulation workflow stage. Processes can be automated through a DL for tasks such as archiving existing input configurations, mapping inputs to result files, managing results to be sent to analysis scripts, generating metadata description records for documents, user notification of events (e.g., results or analyses availability), and reuse of content. DLs also serve as a logical space for organizing content based on ontologies and distributing results, analyses, publications, and related documents.

Currently, there is a notable lack of deployable DL options for research institutions. Existing DLs lack the capabilities to communicate with cyberinfrastructure components, provision of numeric-based service implementations, automatically create metadata records, support simulation tasks, and federate across simulation systems, models, and model versions. My goal in developing SimDL was
to generate a formal, reusable, and component-based DL framework to be utilized by simulation
groups in generating DL software and instances. The focus of this work is on automatically
managing scientific content while enabling scalable discovery and retrieval. As published in [92],
SimDL provides at least the minimal set of functionality required by computational epidemiology
research groups. Formal definitions provide a foundation for the SimDL class of digital libraries.
The formal definitions guided the development of SimDL software and instances. The definitions
allow DL services to be implemented and reused between systems and instances.

SimDL provides a complete digital library with novel services and implementations to directly
support the epidemiology and network science research communities. The process for constructing
SimDL included formally defining content, services, users, and user tasks; identifying the minimal
set of simulation-specific services based on user modeling; implementing simulation-specific ser-
vices; defining a framework for composing generic and simulation specific services; developing
multiple instantiations of the framework for a cyberinfrastructure and local institutional infrastruc-
ture; and evaluating the effectiveness of the set of functionality provided by the framework. A
formalized description of SimDL and simulation content was presented in [87]. Formal definitions
of simulation-specific services have been formulated, implemented in software, and evaluated.
These service definitions describe the minimal set of user tasks required by simulation-supporting
DLs. The definitions are used to describe interactions between users and the infrastructure as well
as patterns of tasks. A broader set of potential service components to include in SimDL are shown
in Fig. 3.4. However, efforts to produce SimDL have concentrated on identifying, defining, and
implementing the minimum required functionality.

Installations of the SimDL framework and software are currently in place for (1) a cyberinfras-
tructure for network science applications including epidemiology, (2) two public health research
institutions, and (3) several simulation models. A specific cyberinfrastructure, CINET, and several
local infrastructures at research institutions provide the initial set of environments for SimDL to
support. These environments were selected as representative samples of possible use cases of
SimDL. A brokering system is used for SimDL communication with other infrastructure compo-
nents. Evaluations of the sufficiency of formally-defined services, effectiveness at supporting user
tasks, and performance have been conducted.
Figure 3.4: Broader context of services and system components in SimDL.
Chapter 4

Digital Library Foundations

Many digital library systems have been developed since around 1991. The digital library community and DL projects generally do not utilize existing underlying foundations or theories that have been found useful in other domains, e.g., databases.

Minimal DLs are expected to provide a set of DL services that meet the anticipated scenarios of a society in a given context. User societies in SimDL are composed of computational and social scientists interested in computational epidemiology and network science for purposes ranging from the simulation of network dynamics to the refinement of public health policies. Metadata structures and provenance tuples link sequences of input data, simulation results, network analyses, generated reports, multimedia files, and publications. Archives of simulation-related content with related DL services hold the promise of changing the way studies are conducted in simulation-based fields. The SimDL framework has been previously described in [87, 88], while some services were defined in [117].

Previous research efforts in digital library foundations have produced two digital library reference models, the 5S framework [52] and the DELOS Reference Model [24]. The 5S framework formally defines DL-related terms using precise tuples [53]. The 5S framework consists of Societies, Scenarios, Spaces, Structures, and Streams. Various aspects of a DL, its content, its users, and its services, can be formally defined using combinations of each of the 5S terms. In the 5S framework, a society is a high-level component of a DL that describes the information needs of users. It defines the administrators who are responsible for DL services, the actors who use those services, and the relationships between them. An example of society in DL includes service managers who are responsible for managing the services provided by a DL. Scenarios detail the behavior of DL services. Scenarios may be used to describe the external DL behavior from the user’s point of view, e.g., searching for documents by type in a DL. Spaces define logical and presentational views of DL components, e.g., vector space models used to rank documents related to queries. A structure specifies the organizational aspects that are used in organizing DL content, e.g., metadata specification representations. Streams describe the communication of information within the DL, e.g., bytestreams of video materials. The 5S framework was formulated with formal descriptions of
core DL concepts. The 5S framework has been used to generate multiple digital libraries including NDLTD, ETANA, and AlgoViz, as described in [43, 45, 54, 136]. Additional research in 5S has focused on quality evaluations (5SQual) [109], a descriptive language (5SL) [51], and graphical representations (5SGraph) [162]. The framework has since been extended through the addition of subsequent definitions tailored to describe aspects of digital libraries within a particular scope, e.g., content based image retrieval [111]. Common services required by many DLs involve indexing, searching, and browsing content as well as query and annotation processes [52]. Producing formal descriptions of digital libraries is facilitated through reuse of the existing set of definitions.

The DELOS Reference Model is a DL foundations effort that parallels the 5S framework. Aspects of a digital library that can be formally defined in 5S can be represented by one of the sections of the reference model. The reference model consists of six high-level terms including content, functionality, user, quality, policy, and architecture. The DELOS reference model describes the digital library universe in terms of digital library management system (DLMS), digital library system (DLS), and digital library (DL). A DL instance is a practically running system that is hosted to allow societies to interact with content. The DLS is the software implementation used to deploy a class of DLs. The DLMS is the representation of the conceptual problem, solution, and DL approach, possibly defined in formal definitions. These definitions include content, users, functionality, architecture, and policy. The DELOS reference model may be used to define content and services as well as to define relations between terms in concept maps of the DL universe [24]. Furthermore, the DELOS functionality working group has researched methods of defining compositions of functions, an essential task in defining the minimal set of SimDL services.

The two models provide frameworks for describing a DL and all of its aspects. In this work, we use the 5S framework to encode formal definitions for a SimDL DLMS, see Fig. 4.1. The 5S framework is utilized to precisely describe DL aspects. The DELOS model is utilized for a separation of the concerns associated with formal definitions, DL software, and DL instances. The DELOS model’s separation of people and services into DLMS, DLS, and DL concepts is one of many potential societal organizations. In addition to formal definitions, semantics, e.g., ontologies and taxonomies, describe the functionality and policies in SimDL. Different societal arrangements of the definitions and semantics would lead to different SimDL architectures. We extend the use of the 5S framework to support services based on 5S definitions of content and users. We made use of the DELOS universe concept to guide our formalization effort. The software DLS and SimDL instances (DLs) are then produced from these definitions. SimDL may be described considering each of these system classes. SimDL as a ‘DLS’ has been provisioned in a software toolkit. Its application as a ‘DL’ includes active deployment in epidemiology and network science infrastructures. The innovative aspects to SimDL include its conception as a DLMS. DLMS aspects include its users, functionality requirements, content definitions, service definitions, service design, automated policies, and architectural design. Our goal was to fully design SimDL as a DLMS, create a useable DLS software implementation of the DLMS design, and deploy the DLS in two practical DL prototypes. The following sections will provide formal definitions for the content, users, and services of SimDL.

The end-users, designers, system administrators, and application developers described in DELOS
can be mapped to formally defined societies in 5S. Similarly, functionality is mapped to services. The other top-level terms are mapped to combinations of 5S concepts, e.g., content is mapped to streams and structures. The following examples demonstrate the potential for mapping from one of the formal frameworks to the other. The service manager user shown in the society concept of the 5S framework can be defined in the ‘user’ concept in the DELOS model. The user concept covers the various actors that interact with DL. The example of searching for documents by type shown in the scenarios concept in the 5S framework can be defined in the ‘functionality’ concept in the DELOS model. The functionality concept defines the function as a specific querying service that a DL offers to its users. The vector space model described in the spaces concept of the 5S framework can be defined in the ‘content’ and ‘architecture’ concepts of the DELOS model. The vector space model represents content including the metadata that the DL handles and may be utilized in architectures for making data available to users. The example of metadata specifications shown in the structures concept in 5S framework can be defined as a type of structure defining ‘content’ in the DELOS
model. The streams of video materials in the stream example can be defined in the ‘content’ concept in the DELOS reference model. Thus, it is possible to ‘translate’ our formal definitions into the DELOS framework albeit with a loss of the mathematical precision found in the 5S framework. For this work, precise formal definitions were desired to guide software implementations, reason over service compositions, and produce a service registry. In other contexts, general definitions may be desired to allow software developers leeway in the behavior and implementation of services.

By designing and implemented SimDL as a DLMS and DLS, we are producing a practical toolkit based on formal theory and abstractions. The DLMS toolkit provides a solution to the well-defined class of problems described earlier. The DLMS design and provided DLS implementation may be used by simulation groups to deploy DLS instances within simulation workflows. Modifications to the DLS will be required by simulation-groups to utilize site-specific databases, support infrastructure components, add functionality, and optimize service implementations.

One innovation of this work is in SimDL as a DLMS and the functionality it provides to simulation infrastructures. Many of SimDL’s services are found in available digital libraries. However, existing service implementations are not suitable for simulation content, tasks, and infrastructures, as they were motivated by a fundamentally conflicting context. Examples of these services include logging content use for incentivizing, comparing numerical studies, and searching scientific collections. Other SimDL services solve new problems that were not previously envisioned in traditional DL efforts. These services include incentivizing, memoization, semi-automated metadata description set generation, metadata extraction, metadata record generation, and record indexing. This effort is aimed at advancing computation-based reproducible science. The benefits it provides to simulation-based research include verifiable results and claims, replicable experimentation, provenance tracking, managed collections, practical services, novel science-specific service implementations, and usage reporting for content and software.

Precise formal definitions of DL services are paired with additional informal descriptions. Service descriptions include actors, activities, components, socio-economic and legal issues, and environments. These descriptions are constructed based on the taxonomy of DL terms provided in [50]. The inputs, outputs, pre-conditions, post-conditions, and implementation details for each service are provided, along with relational operators. 5SL, the language used for formal definitions within the 5S framework, provides a means of encoding the metamodel for the minimal simulation supporting DL. The metamodel describes the architectural component of the DLMS, see Fig. 4.2. The preceding references and the previous body of work in related DL projects support my selection of the 5S framework for the purposes of formalizing the SimDL framework.

4.1 Formal Definitions for Scientific Contexts

Digital libraries have recently been applied in a diverse set of scientific domains requiring sets of services distinguishable from traditional domains. Providing suitable coverage and implementation for scientific digital library services remains a time-consuming effort. Ad-hoc, institute-specific
service implementations introduce the possibility for incompatibility, reduced interoperability, and redundant efforts. Formal frameworks provide a method to formalize the design, structure, and implementation details for required services and existing reusable implementations. I have formally defined scientific content, users, and services for simulation-related domains. Definitions are used to guide service development through a generic scientific digital library toolkit. The definitions and toolkit support DL generation and evaluation for specific domains.

Current scientific digital library efforts are increasing the functionality available to the scientific community through domain-specific implementations [18, 38, 78, 96, 160]. As digital libraries are further expanded across academic institutions and scientific domains, coordination will be needed between digital librarians to reduce duplicated effort. The functionality of scientific applications requires DL services not found in existing DLs (e.g., support for fully numeric content and large-scale scientific content providers). Existing DL services require re-implementation to support scientific content due to deficiencies and mismatches between required and existing services (e.g.,

Figure 4.2: Metamodel of SimDL’s architectural design.
lack of comparable term frequencies and document frequencies in highly numeric data files). New methods for selecting, managing, and accessing scientific digital objects are required. The community of digital librarians developing digital library systems for scientific institutions would benefit from a formal foundation to precisely define concepts and effectively collaborate while remaining cognizant of domain-specific considerations.

4.1.1 DL Management Systems

In the first step, we apply the 5S notation of formal definitions to describe the DLMS layer, see the inner circle in Fig. 4.1. A proposed DL’s content, users, services, and service implementations are defined. The definitions include mathematical notations, input content, output products, pre-requisites, post-requisites, pre-conditions, post-conditions, and implementation details. SimDL required simulation-specific services for storage architectures, browsing and retrieval, content filtering, similarity scoring, incentivizing users, and supporting simulation infrastructures [88]. Formal definitions of simulation users, content, and service requirements were developed early in the DL design process [87].

4.1.2 DL Systems

The DLMS service definitions are used by DL developers to produce the required DLS software toolkit implementation, see the middle layer in Fig. 4.1. We currently use the pairs of service definitions and implementations to generate a service toolkit, seed a potential service registry, identify services pertinent to a required instance, and reuse implementations. We also make use of these definitions in the design of evaluation studies. As an example, formal definitions enable comparisons between scientific DL requirements, services defined in the DLMS, and service coverage provided by the DLS toolkit.

4.1.3 DL Instances

The define-design-develop approach was applied in developing services for the case studies and a fingerprint digital library to support content-based image retrieval and fingerprinting algorithm analysis [117]. See [88, 117] for descriptions of the case study prototypes and formal definitions of the users, content, and services for these scientific digital library systems.

4.1.4 Takeaway

Our work in developing formal definitions, implementations, and formal evaluations of DL instances has direct benefits to scientific DL development efforts. The definitions provide precise
implementation requirements for service developers and DL instance builders. Our ongoing efforts are in producing the DLS toolkit and service registry to promote future reuse of service products through their context and implementation details. Formal definitions allow for formal analysis of interoperable and workflow services. This is conducted by reasoning over potential service compositions based on input and output behavior, pre- and post-conditions, etc. The definitions and toolkits provide a starting point for service reuse and generalization in the scientific DL community.

The contribution of this work is a formal specification of the minimal set of services in an epidemiology simulation supporting DL based on the case study requirements and a working system, SimDL, with implementations for these services. An overview of the minimal SimDL functionality, as defined in formal definitions, is shown in Fig. 4.3. With formal definitions, a collaborating digital library research community could produce a registry of available, reusable DL services. As seen in the search service definition, formalizations are required to differentiate, group, and form hierarchies between multiple implementations of similar services. The definitions provided in this work are a starting point for formally describing required functionality in a representative sampling of possible scientific DL efforts. The 5S framework was used to describe inputs, outputs, pre-conditions, post-conditions, and the desired functionality of a minimal set of SimDL services. Further efforts are required to produce a standard for formally describing implementations of these services in the 5S notation and to further define the composition of specific function implementations. With the addition of more specific definitions of service input formats, output formats, and implementation algorithms, we envision future efforts towards a DL service registration system based on the 5S framework notation for managing scientific DL services.

![Figure 4.3: Definitional dependencies for the minimal simulation-supporting digital library.](image-url)
Table 4.1: Informal service definitions [52].

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acquiring</td>
<td>Takes a set of digital objects, belonging to a collection not in the DL, and incorporates them into some collection of the DL.</td>
</tr>
<tr>
<td>Authoring</td>
<td>Creates a digital object and incorporates it into some collection of the DL.</td>
</tr>
<tr>
<td>Cataloging</td>
<td>Incorporates a metadata specification into a set of metadata specifications describing a digital object.</td>
</tr>
<tr>
<td>Converting</td>
<td>Takes a digital object and produces a version of it (by changing its streams, structures, or structured streams).</td>
</tr>
<tr>
<td>Describing</td>
<td>Produces a description of a digital object (in terms of a metadata specification) and incorporates this description into the object’s set of metadata specifications.</td>
</tr>
<tr>
<td>Filtering</td>
<td>Given a set of digital objects - and either (1) a query, a threshold (a real number), and an index; or (2) a category and a classifier - produces a subset of the original set, in which the objects either match the query with a weight higher than the threshold or belong to the specified category.</td>
</tr>
<tr>
<td>Indexing</td>
<td>Given a collection, produces an index for it.</td>
</tr>
<tr>
<td>Logging</td>
<td>Produces a log entry from some event from a scenario of some service.</td>
</tr>
<tr>
<td>Requesting</td>
<td>Given a handle of a digital object, returns the respective object.</td>
</tr>
<tr>
<td>Searching</td>
<td>Given a query, a collection, and an index for that collection, returns for each object in the collection a real number indicating how well the query matches with the object.</td>
</tr>
<tr>
<td>Submitting</td>
<td>Either incorporates (1) a new object into the collections of the DL; (2) a new metadata specification into the set of metadata specifications of a digital object; or (3) a new operation into the set of operations of a service manager.</td>
</tr>
</tbody>
</table>

4.2 Background Definitions

Previous work in 5S has produced a body of formalisms we reuse to describe SimDL. The development of SimDL builds heavily on previous digital library designs. As such, formal definitions for simulation supporting digital libraries build greatly upon existing formal definitions for underlying services. Table 4.1 contains an informal description of these existing services and provides a foundation for simulation-specific services. Table 4.2 provides a partial formal definition for each of these basic services with which simulation-specific services interact. See [52] for lower-level symbol definitions and pre- and post-conditions not elaborated here. Each of these services rely on content definitions described in previous work [50], e.g., collections \( Coll \), catalogs \( Cat \), repositories \( R \), digital objects \( DO \), and indexes \( I \). At the lowest level, common symbols are consistent across service definitions, as partially shown in Table 4.3. In scenario or service definitions, several relational operators are reused, e.g., precedes and reuses. Relations between societies are used to describe human and computing users and their associations, e.g., includes, invokes, and associated. Relations between content, societies, and services are also encoded in 5S, e.g., executes, recipient, participates_in, uses, and runs.
Table 4.2: Formal service definitions developed from initial work proposed in [52].

<table>
<thead>
<tr>
<th>Service</th>
<th>User input</th>
<th>Other input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acquiring</td>
<td>{do_i : i \in I}, C_i</td>
<td>None</td>
<td>C_k</td>
</tr>
<tr>
<td>Authoring</td>
<td>None</td>
<td>None</td>
<td>do_i</td>
</tr>
<tr>
<td>Cataloging</td>
<td>h_i, ms_{ik}</td>
<td>(h_i, mss_{ip})</td>
<td>(h_i, mss_{iq})</td>
</tr>
<tr>
<td>Converting</td>
<td>C_i</td>
<td>None</td>
<td>C_k</td>
</tr>
<tr>
<td>Describing</td>
<td>None</td>
<td>do_i</td>
<td>ms_{ik}</td>
</tr>
<tr>
<td>Filtering</td>
<td>q, t, C_k</td>
<td>I_{C_i}, cl_{C_i}</td>
<td>{do_j : j \in J}</td>
</tr>
<tr>
<td>Indexing</td>
<td>C_i</td>
<td>None</td>
<td>I_{C_i}</td>
</tr>
<tr>
<td>Logging</td>
<td>None</td>
<td>e_i</td>
<td>log_entry_i</td>
</tr>
<tr>
<td>Requesting</td>
<td>h_i</td>
<td>None</td>
<td>do_i</td>
</tr>
<tr>
<td>Searching</td>
<td>q, C_i</td>
<td>I_{C_i}</td>
<td>{do_k, w_{qk}}</td>
</tr>
<tr>
<td>Submitting</td>
<td>do_i, C_c, ms_{ic}, DM_{C_d}, op_k</td>
<td>None</td>
<td>C_k, DM_{C_d}</td>
</tr>
</tbody>
</table>

Table 4.3: Standard low level formal symbols as described in [50].

<table>
<thead>
<tr>
<th>Concept</th>
<th>Symbol</th>
<th>Concept</th>
<th>Symbol</th>
<th>Concept</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital Object</td>
<td>do</td>
<td>Collection</td>
<td>C</td>
<td>Annotation</td>
<td>ann</td>
</tr>
<tr>
<td>Set of annotations</td>
<td>ans</td>
<td>Handle</td>
<td>h</td>
<td>Binder</td>
<td>bi</td>
</tr>
<tr>
<td>Actor</td>
<td>ac</td>
<td>Hypertext</td>
<td>Hyptxt</td>
<td>Metadata Specification</td>
<td>ms</td>
</tr>
<tr>
<td>Set of Metadata Specifications</td>
<td>mss</td>
<td>Classifier</td>
<td>Class_{C_i}</td>
<td>Category</td>
<td>c</td>
</tr>
<tr>
<td>Cluster</td>
<td>clu</td>
<td>Transformer</td>
<td>tfr</td>
<td>Space</td>
<td>sp</td>
</tr>
<tr>
<td>Weight</td>
<td>w</td>
<td>Index for Collection C</td>
<td>I_{C}</td>
<td>Query</td>
<td>q</td>
</tr>
<tr>
<td>Stream</td>
<td>stm</td>
<td>Structure</td>
<td>st</td>
<td>Structuring Function</td>
<td>\psi_{ij}</td>
</tr>
<tr>
<td>Threshold</td>
<td>t</td>
<td>Event</td>
<td>e</td>
<td>log entry</td>
<td>log_entry</td>
</tr>
<tr>
<td>Rating</td>
<td>r</td>
<td>Operation</td>
<td>op</td>
<td>Set of Categories</td>
<td>{C_i}</td>
</tr>
</tbody>
</table>
4.3 Content

Scientific and simulation-related content greatly differs from traditional digital objects. A simulation workflow produces multiple stages of simulation content that require data management support. The stages of content include collections for model schemas, model ontologies, user input parameter configurations, output results, output summaries, standard analyses, documentation, annotations, and publications [87]. A digital library managing these collections must support text, binary, data, image, and video files, as well as hypertext.

Research data is not well handled by typical DL services. A single experiment from one user may consist of hundreds of gigabytes of results requiring management as a single digital object. As the largest digital objects begin to grow into terabyte range for general scientific research, typical policies for distribution, transfer, and curation must be re-thought as some content may be regenerated at an acceptable computational cost. In massive-scale computing applications, e.g., weather and climate modeling at Argonne National Laboratory, simulations produce petabytes of results per simulation study. Indexing and searching DL services for numerical and graph files are currently inefficient and generally based on previous work in full-text collections. SimDL approaches these problems by using ontologies and metadata harvesting scripts to generate individual model-specific metadata records.

Figure 4.4: Representation of scientific document workflow. See [87] for additional information regarding the workflow.

Allowing a digital library’s users access to an entire provenance stream allows claims to be supported, organizes the existing body of previous work, and allows data mining of collections related to a simulation system. See Fig. 4.4 for an Open Provenance Model representation of the content.
definitions [108]. The provenance metadata depicted as directed edges in Fig. 4.4 are tracked in SimDL within the provenance services. The following definitions provide a representation of the content supported by simulation-specific services, as presented in [87]. These definitions build upon the set of previous 5S definitions (i.e., handles, streams, structures, digital objects, complex objects, and annotations).

**Definition 1:** A *schema* is a digital object of tuple $sch = (h, sm, S)$ where

1. $h \in H$, where $H$ is a set of universally unique handles (labels);
2. $sm$ is a stream;
3. $S$ is a structure that composes the schema into a specific format (e.g., XSD structure of elements and attributes of restricted values).

**Definition 2:** An *input configuration* specification matching an XSD schema is a tuple $icfg = (h, sm, ELE, ATT)$ where

1. $ELE$ is a set of XSD elements;
2. $ATT$ is a set of XSD attribute values for an element $ELE_i$.

**Definition 3:** A *sub-configuration*, a subset of an input configuration, is a tuple $sub-icfg = (h, sm, icfg, ELE, ATT)$ where

1. $icfg$ conforms to an XSD schema and $icfg \supseteq sub-icfg$;
2. $ELE$ is a set of XSD elements where $ELE \subseteq icfg$’s $ELE$;
3. $ATT$ is a set of XSD attribute values for an element in $ELE$.

**Definition 4:** A *result* is an output from a simulation or experimentation process, corresponding to a specific $icfg$, and is a tuple $res = (h, sm, icfg, proc)$ where

1. $icfg$ is the matching input configuration;
2. $proc$ is the process used to generate the result from $icfg$.

**Definition 5:** *Analysis* of a set of experiments is a complex object consisting of textual or numeric documents and images (e.g., plots and graphs) and is defined as a tuple $ana = (h, SCDO = DO \cup SM, S, icfg)$ where

1. $DO = do_1, do_2, ..., do_n$, where $do_i$ is a digital object;
2. $SM = sm_1, sm_2, ..., sm_n$ is a set of streams;

3. $S$ is a structure that composes the complex object $cdo$ into its parts in $SCDO$;

4. $icfg$ is the input configuration and one-to-one mapping to a raw dataset.

**Definition 6:** An experiment is a complex object consisting of the full range of information constituting an experiment within a domain and is defined as a tuple $exp = (h, SM, sch, icfg, ana, D, A)$ where

1. $SM = sm_1, sm_2, ..., sm_n$ is a set of streams;

2. $D = d_1, d_2, ..., d_n$ a set of additional documents, e.g., summary or publication;

3. $A = an_1, an_2, ..., an_n$ a set of annotations describing the overall experiment and individual digital documents.

### 4.4 Societies

User and system modeling has been the focus of research activities in the social aspects of computing. The Zachman framework produces a matrix of rows representing different people’s perspectives on a project where columns represent what they are seeing from that perspective (e.g., data, activities, locations, time, and motivation) [66]. Perspectives are influenced by the activities associated with a user role, e.g., planner, owner, architect, designer, builder, functioning system. The following communities maintain inter-community relationships through activities as described through a set of community-specific digital library scenarios [87]. These user roles do not include the complete set of stakeholders outside of the DL context, e.g., modelers and system developers who produce new simulation models.

1. **Tool builder:** formally define a proposed DL; locate, reuse, and assemble existing components; generate novel components; deploy a DL instance; integrate with a simulation or digitized experimentation infrastructure

2. **System and DL administrators:** set data management policies; clean datasets or metadata; curate content; manage accounts; evaluate existing components

3. **Related systems:** submission and retrieval with experimentation applications and high-performance computing architectures; analysis of submissions and retrieval with analyst oriented software; validation of inputs

4. **Study designer:** generate new model schemas and transform to updated versions; enter or load input configurations; query, browse, and load sub-configurations; save configurations and sub-configurations; validate configurations; submit configurations for execution; monitor experiment progress
5. **Analyst:** submit new analysis requests; view automated or requested analyses

6. **Annotators:** mark annotations on streams of content or individual documents

7. **Explorer:** query and browse collections or experiment streams of documents

Societies are defined in 5S as a tuple $society: (C, R)$, community and relationships between communities that form around related activities. $R = \{r_1, r_2, ..., r_m\}$ where each tuple $r_j = (e_j, i_j)$, $e$ is communities, $i$ activity.

### 4.5 Scenarios and Services

We aim to formally define a suite of DL services to be implemented in SimDL and other simulation management systems. Formal definitions provide a means of explicitly specifying the requirements for services. Formal definitions allow for registered, existing services to be identified and reused in multiple DL implementations. Requirement identification and development may use formal definitions to guide efforts. Definitions may be used to prove sufficiency and completeness of services without specifying implementation decisions. With formal definitions of services, we envision a future registration system for simulation-specific services described by languages such as UDDI, OWL-S, and the 5S framework. A service registry would be useful in rapidly prototyping DL service layers that support HPC infrastructures and scientific applications as described in [61, 104].

The following set of services are essential for a minimal simulation supporting digital library. Existing service definitions and notation are reused where possible but are mainly limited to text-based versions of simulation-specific services, e.g., generic indexing and search. See Fig. 4.5 for an overview of the minimally required simulation-specific DL services and broker connections. Several services have been defined and implemented that are specific to simulation-supporting digital libraries, see Table 4.4. These services either do not exist in existing DL software or require implementations that are not compatible with existing software. Each of these services is elaborated with a description, goal, set of scenarios, and formal definition within the following section.

The 5S approach generates tailored DL instances based on metamodels, 5SL definitions, the 5SGen digital library generation suite, and a component pool of existing software, see Fig. 4.6. We have formalized SimDL, extended 5SL to encode SimDL, and generated the component pool and DLS for SimDL manually. This was required as suitable service implementations did not preexist for handling simulation processes. The interaction between services is vital to build a hierarchy of increasingly complex services. For future uses, the services and definitions provided here will be reusable by future DL Implementers through the SimDL toolkit implementation and matching formal definitions.

The class of large infrastructure environments targeted in this work contain metadata databases describing simulation-related content in multiple forms and collections internal to tightly coupled collaborator systems. Similar systems for managing distributed metadata databases would benefit
Figure 4.5: Organization of the minimal simulation-supporting services and communication.

Table 4.4: Informal service definitions for simulation science.

<table>
<thead>
<tr>
<th>Service Brokers</th>
<th>Supporting Metadata</th>
<th>Searching in Science</th>
<th>Incentivizing</th>
<th>Memoization</th>
<th>Provenance Tracking</th>
<th>Curating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blackboard-based Communication</td>
<td>Given a specific simulation process (1) semi-automates the generation of metadata description sets, (2) describes the methodology to extract metadata values from files, (3) describes the expected metadata record format, (4) automatically processes files as they are generated by a simulation system, (5) automatically produces properly formatted metadata records, and (6) indexes the metadata records in a simulation metadata index and catalog.</td>
<td>Given a query and index of metadata covering scientific collections, returns a listing of similarity scores related to the pertinence of each digital object with the query.</td>
<td>Logs and tracks the usage of content, software, and services and provides reports to producers.</td>
<td>Takes an input configuration for a simulation request, identifies duplicate or overlapping simulations, and returns the matching completed simulation results.</td>
<td>Maintains links between related content based on a defined scientific workflow.</td>
<td>Takes a collection, metadata index, and curation rules and recommends content from the collection and index, as the rules dictate, for arching, deletion, migration, and preservation.</td>
</tr>
</tbody>
</table>
from similar curation services at the infrastructure level, see distributed hash tables and P2P networking from [128]. Previous work in developing SimDL to provide interoperability for multiple infrastructures has uncovered the need to assist individual organizations within the scientific community to automate the: design of metadata databases, capture of metadata for scientific products, and long-term management of content. As suggested in [158], there is a breadth of potential implementations when allocating responsibilities between DBMS, the surrounding infrastructure, and applications. Our approach implements automated functionalities at the level of infrastructure services in contrast to widely-practiced human-intensive database administration (DBA) functions, e.g., indexing and tuning, commonly implemented at the database level. Digital libraries provide numerous infrastructure-level services such as searching, browsing, archiving, preserving, annotating, and recommending. DLs manage scientific content and make use of metadata databases and indexes to provide these higher level services. These can make use of existing techniques to improve efficiency, such as index self-tuning [148] and periodic automated index management [63]. Event-based triggers at the infrastructure-level may be used in self-healing in terms of disk space faults, and preventing failures, as identified in [158]. Our curation framework produces event-based digital library systems that process infrastructure events (e.g., input file saved, result generated, disk space warning issued) and perform automated infrastructure-level curation services (e.g., generate an input file metadata record, archive results, and remove low-priority content based on existing policies). In cloud computing and large data-intensive simulation systems, the automation of these tasks reduces the effort and time requirement for database administrators and scientists in tasks tangential to their primary responsibilities.
Formal definitions of each service are provided along with the service description in the ‘Simulation-Specific Services’ chapter.

Finally, SimDL is a 4-tuple $(R, DM, Serv, Soc)$, where

- $R$ is a repository consisting of scientific content collections (e.g., input configurations and results);
- $DM = \{DM_{C_1}, DM_{C_2}, ..., DM_{C_K}\}$ is a set of metadata catalogs for the collections $\{C_1, C_2, ..., C_K\}$ in the repository;
- $Serv$ is a set of services containing simulation-specific services for at least metadata extraction, metadata record indexing, searching, curation, provenance tracking, and incentivizing, as well as the set of services found in original, non-simulation digital library definition; and
- $Soc$ is a set of societies including administrators, analysts, annotators, explorers, external infrastructural components, study designers, and tool builders.
Chapter 5

Simulation Digital Library Framework

5.1 A Simulation Supporting Digital Library

SimDL aims to fill the gap between commonly available DL software and the requirements of simulation environments. The digital library is intended to be integrated with a larger simulation system and is not distributed as a standalone system, see Fig. 5.1. SimDL is based on a broker system in which multiple external brokers exist for DL communication. As such, SimDL’s architecture consists of internal services and broker connection APIs. The digital library broker coordinates the data flow from external infrastructure brokers and provides a content access point for available services. SimDL APIs and brokers allow for component communication and integration within simulation environments. Ontologies describing individual simulation models are used to form a network of related repositories. Repositories with various stages of model-specific content are integrated through links which form a domain meta-ontology from individual ontologies. Domain specific metadata from multiple simulation models are harmonized for inter-model interoperability and search.

SimDL provides common services found in many digital library implementations as well as novel context-specific services. The services provided by SimDL are primarily implemented in the middleware agent which is supported by the user interface and storage system. The fundamental services in SimDL, and digital libraries in general, are the ability to store and retrieve information. This is supported by input connections to Simfrastructure, the user contribution upload interfaces, and the ability to deposit content to the storage system. Administrators require the ability to modify or manage the collection’s state as well as SimDL itself. Search functionality requires an indexing scheme and metadata storage. Metadata is leveraged to provide a higher-level view of simulation data and related files without investigation of each digital object itself. Indexing for a set of experiment scenarios allows for queries over the range of experiments. Third party access is provided to support extended tools that may leverage SimDL’s services. The search functionality commonly provided by current digital library implementations provides the ability...
to search metadata and full-text search in some cases. SimDL requires a new style of search functionality as widely-used standards for full-text document mining and search are not generally relevant to scientific content, with the notable exception of scholarly publications.

The SimDL framework provides DL service abstractions for producing a DLS to fulfill a DLMS design. The service software implementations are modular and capable of building on lower-level services. The metadata schemas utilized by services are implemented as RDF triplestores, allowing arbitrary metadata formats and records to be sent to the DL for storage. External components are able to modify metadata schema and index or search content without changes to a structured data model or DL. This also allows multiple projects, workflows, and content types to use the same DL implementation. The DLBroker communication workflow maps service requests to DL services. With this abstraction, multiple DLs tailored to different domains or projects may be connected to the same infrastructure. The exposed services are domain-free. The incentivizing service allows customizations through dynamically set incentivization report and metric templates. The search service utilizes query model arguments of the similarity and ranking aspects of searching. The DL services have a high level of modularity. This is demonstrated in the dependencies and with lower-level services in the higher level collection building and information satisfaction services. The services can execute independently within this space. The design is useful beyond the CINET and SimInfrastructure case studies. The SimDL DLMS and software toolkit is available to be utilized in other contexts with arbitrary data models without changes to the internal metadata indexing scheme.
5.1.1 Computational Scalability

In addition to file storage issues, the computational availability of data management systems is constrained by hard limits. Providing SimDL services will require processing for each user query as the metadata for the total set of experiments may have to be ranked. In addition to the computational power required to deploy SimDL for a research group, Simdemics will require considerable computation to perform an experiment. Simulation wall-time may span a few minutes to several weeks depending on the number of agents, complexity of behavior, range of time steps, and experimental runs. Assuming these experiments would be run regardless of the existence of SimDL, the digital library framework may reduce the computational effort should users select to retrieve existing simulations or multiple experiments cover the same input ranges as requested for slightly different scenarios.

5.1.2 Scientific Data

Experimental data in the epidemiology and network science contexts require a data model with special considerations. The investigated scenarios are based on the diffusion of a given phenomenon across a sizable dynamic network. In these simulations, agents and behavior may be represented as nodes and interactions as edges in a network. This structure provides demographic rich datasets as each node may contain detailed information on the agent being represented. The network may scale from dozens of computer systems to billions of humans or poultry. These dynamic networks change over time. Some studies may require capturing large entire networks at given points of time. At a minimum, the initial network of agents is required in storage. In the CINET case study, a similar type of content exists in the form of large-scale networks from multiple domains. Specific discrete events in epidemiological simulations, such as agent activity or geo-temporal travel behavior, may require storage for each event and agent pair that may occur in simulations. It should be noted that even within epidemiology, the diverse set of known and modeled diseases may require a range of simulation techniques and separate software solutions. For example, simulating a human-to-human airborne disease scenario requires models, inputs, datasets, and disease diffusion that are different from that of a vector-borne scenario. A digital library for epidemiology simulations may require connections to multiple simulation software tools that are individually specialized for respiratory, gastrointestinal, sexually transmitted, contamination, and insect vector-borne disease scenarios. Considering the combined set of scenarios, the multiple disease models, transmission vectors, and possible intervention strategies, it is clearly unsuitable for a single simulation tool to attempt to simulate all possible experiment sets. Although each tool generally simulates the diffusion process of a phenomenon across a network of agents, the process for simulating each scenario may require specialized methods. Due to the stochastic variance and popularly known butterfly effect, the exact inputs to the system are required to provide consistent and deterministic output production. The complete storage of inputs and the resultant simulation outputs provides a rich environment for data mining, content reuse, and repurposing. Fig. 5.2 describes a simplified version of the information workflow of data products as generated in the case study systems.
5.1.3 Data Integrity

The validation of results depends on the quality of the modeling conducted for the underlying network, agent behavior, and phenomenon diffusion over the network. Unfortunately for researchers, datasets surrounding disease occurrences across the globe are incomplete, and there is the lack of an acceptable dataset standard. To complicate matters, a disease may not have a standard vocabulary for items associated with the disease. It may be impossible to intelligently compare the validity and quality of results coming from multiple simulation tools which use separate agent and phenomenon datasets. Unlike publications, data are not usually directly evaluated for validity or potential reuse [150]. Within a group, standardized and accepted procedures are typically in place for common data. Those outside of a research group may be granted access to a dataset without knowledge of the dataset’s integrity. The dissemination and reuse of scientific data would be quite valuable for long running and computationally-intensive simulations. Metadata models are to either include or contain input files that include all simulation parameters. Investigations on the integrity of the selected simulation parameters and underlying datasets would require additional and complimentary efforts outside the scope of SimDL. The role of SimDL in this process is to maintain the available metadata relating to data validity and integrity. Integrity information beyond the scope of SimDL (e.g., the generation of synthetic populations) is not captured during the simulation process. However, this may be added by extending the utilization of the provenance registration services in non-minimal case studies. Descriptions of additional integrity-related information may be added by attaching additional files to experiment sets through the use of related metadata.
5.1.4 Quality of Service

As SimDL is planned for research group deployment, its implementation must address several issues. To fulfill storage requirements, the selected data formats and storage scheme should require as little space as possible to provide input coverage. Storage of content will require a sufficient storage space allocation as well as permanent availability as provided through the ‘data manager’ component. To assist in preservation, data backup options are necessary for replication. Data may be compressed with widely available compression tools to limit the storage space and file transfer sizes. Network traffic should be limited through the use of compression and transmission of only the set of requested files though it is assumed that sufficient bandwidth is available for transmission of large files. The metadata and index structure should be minimized to the level of sufficiently meeting the requirements of all services. The metadata selected should be appropriate to context-specific simulation scenarios. Thus, various types of simulation domains and content types will require slightly different metadata schemes. To promote general acceptance of the results and analyses, the context-specific modeling of inputs should adhere to informal institutional guidelines. A bound on the system search time is required to ensure usability of the searching functions. This computational restriction may culminate in a limit on open user connections to the maximum serviceable amount per the targeted deployment system. This requirement is related to the computational power devoted to performing search on the index and metadata items. Performing large-scale, complex simulations requires a large amount of processing. It is necessary for large amounts of computational power to be available for the purpose of ensuring that users are not unnecessarily delayed in retrieving datasets relating to dynamically conducted simulations. Should the digital library fall suspect to a malicious denial of service attack, SimDL must recover and rollback to earlier metadata and index backup files. As the digital library will require continued development and integration of yet to be developed infrastructures, the system design is modular and extensible enough to permit different styles of brokers or wrappers.

5.1.5 Integration and Framework Extensions

A key component of this effort is to link the digital library to the simulation software capable of producing parts of the collection. The boundary between these two independent systems determines which software is responsible for certain functionality. For example, users may use software executables directly and produce simulation sets separate from SimDL. Several Simdemic interfaces have their own implementation of query input matching to identify exact matches to previous simulation runs. Ranking mechanisms are an integral part of a digital library and provide discovery over the corpus of registered content. For experiment sets in Sim Infrastructure not known by SimDL, a call would be made to rerun a previous experiment, at which point SimDL recognizes an exact match and returns a link to the existing simulation results in SimDL. Regardless of whether new computation is required or not, SimDL is eventually provided with the required simulation outputs and analysis produced by applications as well as logs of the users who interacted with them.

Integration is achieved by registering an infrastructure’s blackboard system with SimDL, through
the SimDL properties file. SimDL then monitors this blackboard, receives service requests, and places results back onto the blackboard. Additional modifications are needed in existing external components to redirect existing processes, workflows, and functions in order to send requests to the DL.

The framework can be modified through the addition of new services, modification of existing service implementations, and the parameterization of services. To add a new service, the name of the service needs to be registered in the DLServiceType enumeration (one line of code). The service code itself is added to the code base as a standalone class that extends the DLService interface and provides an `execute()` method. Lastly, the DLBroker’s service selection algorithm must be updated to correctly process requests from the blackboard and forward the request to the new service (two lines of code). Existing service implementations are modified by optimizing the `execute()` method within the service class. Parameterization of the incentivization and curation services require additions to the reports, statistical measures, partial Boolean decision clauses, and curation operators. While not elaborated here, the procedure for completing this process is clear in the technical documentation and code. There is no current formal definition registration service or process.

### 5.2 The Software Architectural Overview

The digital library maintains a set of thirteen services that may be invoked directly or through a predefined blackboard and DLBroker. The digital library code includes a set of services that each utilize the DLDataItem to return generic results back to the DLBroker. There are dependencies between these services. The dependencies generate a multi-tiered digital library with more complex services building upon lower-level services. See Fig. 5.3 for a depiction of the data flow between classes within SimDL. See Fig. 5.4 for a depiction of how control flow proceeds between SimDL components.

A centralized blackboard and component brokers manage the communication of messages and data in the SimDL infrastructure. In addition to passing messages between system components and the digital library, the infrastructure also automates the experimentation workflow. This approach is based on existing systems outside of the public health domain that compose models into workflows [33, 99]. See Fig. 5.1 for an overview of the style of infrastructure SimDL has been integrated with in each case study.

After receiving a simulation request, the brokering system is used to locate an available computational system with the appropriate simulation software. The input files are transferred to the identified system and the simulations are launched. Progress updates are sent back to the user interface as individual simulation runs are completed for progress monitoring purposes. After the simulation process has finished, results are transferred to the digital library for archiving and forwarded to the analysis system. Results are directly linked with input configurations and model schemas used in previous steps. This provenance information could be used by the analysis broker to
invoke analysis software that is pre-defined to support a particular simulation model. Modelers and
policy makers generally do not have extensive technical background or training. The infrastructure’s automated workflow abstracts the need for novice users to fix formatting errors, transfer data to remote machines, and execute command line applications and analyses.

Through blackboard connections, SimDL manages content such as input configurations and study designs as they are saved by users, results after simulations terminate, and image plots after analysis scripts are finished. The digital library broker acts as an adapter between the blackboard and SimDL. This indirection allows for other digital libraries and data management systems to be added to the system to complement SimDL. It also allows for more than one digital library to service the same blackboard system. The content managed by SimDL resides in databases and file systems on computational platforms. The five types of simulation content produced by the simulation workflow, see Fig. 3.3, form the five collections managed by SimDL (e.g., schemas that describe simulation models). For each digital object in the collections, SimDL automatically generates a metadata record consisting of simulation model-specific terms, collection-level terms, Dublin Core terms [153], and formal 5S framework terms [53]. The metadata records are then able to be used by other services, e.g., to rank content for user searches and execute automated curation policies. Tables 5.1 and 5.2 contain a brief description of each of the SimDL classes.

It is possible for the architecture of SimDL to be described and implemented with various design styles. Since raw data itself, not a high level summary of results, is involved, it is likely that a rather large amount of data will be ultimately stored in each instance. The scope of the digital library will determine what types of experimental results should be stored. Equally important to the scope of experiments being stored is the way in which the data will be used in the future. Ideally, a researcher may be able to save effort by using previous results instead of running new experiments. To accomplish this, the metadata must represent digital objects results accurately. A search function will allow the user to find previous experiments that are similar to the inputs being investigated. If no matching experiment inputs are found, this may imply that certain sets of inputs have not been previously modeled and simulated in earlier studies. If distantly matching experiments are returned in a ranked list, these experiments may provide a basis for examining the effects that various input parameters for the experimental scenario have on the results of the experiment. When highly matching ranked results are returned, there may be an indication that experiments with this type of inputs have been studied before with available results already available. Thus, using a digital library to archive results may actually lead to a system for identifying viable areas of interesting experimentation.

SimDL initially supports a hierarchal XSD schema consisting of input attributes, elements, restrictions, and documentation. Simulation systems without a hierarchical structure to input parameters will simply have a flat schema consisting of mainly top-level elements. Hibernate’s\(^2\) automation functionality allows database tables to be constructed to match a recently uploaded schema and map records in each table with objects SimDL may handle. This automation produces the SimDL storage layer for a specific schema and allows for multiple DL components to access a schema’s related collections based on the schema itself without interaction from a system administrator or

\(^2\)http://www.hibernate.org
Table 5.1: SimDL service and broker classes.

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLBroker</td>
<td>connects to an assigned blackboard (test, development, production), reads blackboard entries sent to the ‘DLBroker’, processes a sent entry based on the requested DLServiceType, and returns the result to the blackboard.</td>
</tr>
<tr>
<td>DLBrokerTest</td>
<td>contains testing scripts for the broker and all of the DL services and classes.</td>
</tr>
<tr>
<td>DLCurate</td>
<td>executes and evaluates the set of rules defined for a curation activity and returns a listing of digital objects that are suited for that activity. Currently implemented actions include preserving, deleting, archiving, and migrating. Execution of the set of rules will produce a collection of digital objects that are recommended for the attached curation activity.</td>
</tr>
<tr>
<td>DLCurateAction</td>
<td>is an enum representing: preserve, delete, archive, and migrate. These are the possible activities that may be guided by specific DLCurateRules (policies).</td>
</tr>
<tr>
<td>DLCurateBooleanPartial</td>
<td>builds and executes queries over metadata and encodes a restricted set of SQL and RDF operators (see DLCurateOperator). This is used to discover the set of digital objects that match specific metadata criteria. The criteria initially include in, between, &gt;, &lt;, &lt;=, &gt;=, and =. This allows more complex rules to be defined which combines the results from multiple partial decision clauses.</td>
</tr>
<tr>
<td>DecisionClause</td>
<td>builds and executes queries over metadata and encodes a restricted set of SQL and RDF operators (see DLCurateOperator). This is used to discover the set of digital objects that match specific metadata criteria. The criteria initially include in, between, &gt;, &lt;, &lt;=, &gt;=, and =. This allows more complex rules to be defined which combines the results from multiple partial decision clauses.</td>
</tr>
<tr>
<td>DLCurateCombinationType</td>
<td>is an enum representing: arithmetic (mean), geometric (mean) harmonic (mean), summation, and maxContributingClause. These represent the possible mechanisms for combining and ranking the results from multiple partial decision clauses.</td>
</tr>
<tr>
<td>DLCurateDORanker</td>
<td>takes a set of Boolean partial decision clauses, executes each clause, and returns combined results to fulfill the full clause by combining the multiple results as defined by the requested DLCurateCombinationType.</td>
</tr>
<tr>
<td>DLCurateOperator</td>
<td>is an enum representing: in, between, lessThan, greaterThan, equals, lessThanEquals, and greaterThanEquals. These are used to generate an SQL or RDF query statement pairing a specific metadata term with the query or clause inputs.</td>
</tr>
<tr>
<td>DLCurateRule</td>
<td>encapsulate a single rule composed of partial decision clauses, combination type, threshold, and curation action. Each digital object with a combined clause weight that exceeds the threshold is returned in a list suggesting the action be performed on that object.</td>
</tr>
</tbody>
</table>
Table 5.2: SimDL service and broker classes (continued).

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLFilter</td>
<td>builds and executes an SQL query of arbitrary metadata terms and values. The returned data is a listing of digital objects that exactly match all of the query terms.</td>
</tr>
<tr>
<td>DLGetDLID</td>
<td>generates and returns a new unique digital library identifier, DLID.</td>
</tr>
<tr>
<td>DLGetMetadata</td>
<td>queries and returns the full set of available RDF metadata predicates and objects for a single request subject.</td>
</tr>
<tr>
<td>DLIncentivize</td>
<td>executes a requested report and returns formatted text produced by combining all of the report metrics’ results.</td>
</tr>
<tr>
<td>DLIncentivizingMetric</td>
<td>maintains a list of possible metrics (SQL statements) that mine the RDF metadata table. These may be utilized in multiple reports.</td>
</tr>
<tr>
<td>DLIncentivizingReport</td>
<td>defines a single report as a set of specific DLIncentivizingMetrics.</td>
</tr>
<tr>
<td>DLIncentivizingReportType</td>
<td>is an enum representing: provider, graph, software, result, hpc, user, and system. These are the initial set of available reports.</td>
</tr>
<tr>
<td>DLog</td>
<td>inserts a set of RDF metadata triples in a metadata record as a batch.</td>
</tr>
<tr>
<td>DLMemoization</td>
<td>utilizes the DLFilter and DLGetMetadata services to identify a set of digital objects that exactly match a metadata pattern and returns the matching Data Manager SIDs if known by the digital library.</td>
</tr>
<tr>
<td>DLProvRegister</td>
<td>allows the provenance sequence between two digital objects to be assigned.</td>
</tr>
<tr>
<td>DLProvResolve</td>
<td>retrieves the set of digital objects that maintain the queried provenance type (i.e., precedes or follows) with the queried digital object subject.</td>
</tr>
<tr>
<td>DLProvType</td>
<td>is an enum representing: precedes and follows. This allows the provenance for a workflow of digital objects to be assigned.</td>
</tr>
<tr>
<td>DLRRegister</td>
<td>inserts a set of RDF metadata triples in a metadata record as a batch of RDF triple inserts.</td>
</tr>
<tr>
<td>DLDatum</td>
<td>utilizes Java Generics to allow an unspecified data type to be stored. Each DLService implementation returns this object.</td>
</tr>
<tr>
<td>DLSearch</td>
<td>builds and executes an arbitrary weighted Boolean query over the metadata index. The results are a ranked list of digital objects with the similarity score of each object to the query using a modified Rogers and Tanimoto similarity score.</td>
</tr>
<tr>
<td>DLService</td>
<td>provides an interface that all DL services must implement. Specifically, each service must have an execute method with Generics arguments and return type.</td>
</tr>
<tr>
<td>DLServiceType</td>
<td>is an enum representing: DLog, DLRRegister, DLIncentivize, DLSearch, DLFilter, DLMemoization, DLCurate, DLCurateDORanker, DLProvRegister, DLProvResolve, DLGetMetadata, and DLGetDLID. This is used in the DLBroker to direct a service request to a service capable of fulfilling the specific request.</td>
</tr>
</tbody>
</table>
5.3 Service Organization

The services in SimDL are split into ‘collection building’ and ‘information satisfaction’ services. Collection building services are generally services that build and organize collections of content. Information satisfaction services allow content to be accessed and in SimDL also support scientific activities. Fig. 5.5 displays potential science-supporting services beyond the minimal functionality found in SimDL. Within these categories, services are organized into workflows of low-level and high-level services. These terms are not an indication of complexity but whether a particular service interacts with digital object and metadata collections (low-level services) or collections and other services (high-level services). These distinctions lead to directed dependencies between services shown in Fig. 5.6. These dependencies arise as high-level service implementations perform software calls to lower-level services.

<table>
<thead>
<tr>
<th>Collection Services</th>
<th>Information Satisfaction Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creation and Maintenance</td>
<td>Access</td>
</tr>
<tr>
<td>Contributing</td>
<td>Browsing</td>
</tr>
<tr>
<td>Curating</td>
<td>Discovering</td>
</tr>
<tr>
<td>Indexing</td>
<td>Listing</td>
</tr>
<tr>
<td>MD description set generating</td>
<td>Recommending</td>
</tr>
<tr>
<td>Metadata extracting</td>
<td>Retrieving</td>
</tr>
<tr>
<td>Metadata record generating</td>
<td>Searching</td>
</tr>
<tr>
<td>Producing and consuming</td>
<td>Storing</td>
</tr>
<tr>
<td>Reporting</td>
<td></td>
</tr>
<tr>
<td>Commenting</td>
<td></td>
</tr>
<tr>
<td>Tagging</td>
<td></td>
</tr>
<tr>
<td>Rating</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.5: Possible science supporting services.

5.4 Interoperability

A digital library tasked with managing content from an entire scientific method will contain multiple heterogeneous collections, each potentially considered a non-autonomous digital library. The digital library system enables the federation of collections and internal processes dealing with
managed content. Interoperability between collections of related content is provided by leveraging the provenance mapping between sets of items derived from each experimentation stage.

5.4.1 Functionality, Services, Scenarios

SimDL acts as a services provider when integrated within a larger infrastructure. Services provided through the SimDL broker to user interfaces allow access to multiple collections and ease the effort required to manage users, support roles, and allow users to switch roles as their access allows. The automation services utilized by the interfaces in each of the case studies provides a browser with a consistent look and feel across user roles. The digital library’s server API allows customized UIs, designed by tool builders, to take advantage of the underlying services. A simulation-launching component may submit simulation requests and input configurations to a backend staging space. Simulation brokers monitoring the staging space can process the request and launch the experiment on a set of optimized computational resources. Similarly, submitting analysis requests can be automated as a digital library service. Through the use of an analysis request staging space, analysis procedures and software can be invoked for a recently completed simulation’s datasets. SimDL automates the process of creating, inserting into, or querying a database after processing the logic.
contained by a selected schema.

Functionality wrapped in modules is reusable when there exists clear specifications of each service, definitions of underlying digital objects, and pre- and post-conditions as suggested in [142]. Code porting, integrating existing components to serve specific DL functions, is assisted with formal specifications of a desired system. Thus, formally defined services in an existing SimDL instance are reusable in generating other instances. Data resulting from a simulation process is typically analyzed to produce knowledge in regard to a tested scenario. This experimental process provides a provenance chain resulting from the composition of defined functions. Several stages of the experimental process are automated in SimDL by the composition of simulation and analysis submission functions. An intuitive querying function over simulation parameters and user annotations provides access to the stream of digital objects generated by each of the composite functions. SimDL differs from other DLs in that functionality is included through integrating new internal components, not through connecting infrastructures as in D4Science [23].

5.4.2 Users

Multiple experts are required to conduct complex simulation-based research. Similarly, multiple user roles exist for simulation-based digital libraries which support collaboration, participation, and privacy as defined in [75]. In SimDL, collaboration is assisted by capturing and exchanging information produced from complex tasks performed by users at different stages of the experimentation process. The collaboration requires multiple tools to answer research questions by translating a query or simulation request into successive units of work. Although the submission of simulation results to an analysis system may be automated, participation from users is required to launch new simulations, perform customized analyses, draw conclusions, and annotate streams of content. Users with similar and differing roles work on the same content, further existing work between and within streams of content, and switch between consuming digital objects from previous experimental stages to providing content for users in successive stages. Privacy is provided by restricting access to an institution’s users and trusted external individuals or groups. Collections are partitioned to provide access to users of a particular role (e.g., a general public account might allow finding existing documents and publications but not proprietary simulation schemas). In addition to privacy concerns, practical consideration may restrict the number of users with a simulation launching role to reduce the stress on computational and data storage resources. A digital library’s interaction with its infrastructure affects its provided user interoperability. Users under most roles interact with a digital library’s content and are presented with the ability to interact with multiple schemas and versions of schemas across all available simulation applications, domains, and contexts within a single, generic interface. By selecting tabs in the UI, users with a particular role may transition seamlessly to another role (e.g., study designers switching to analysts when viewing the results from a launched simulation).

Tool builders are provided with reusable, formally defined components. The use of a schema automates the UI generation database mapping, simulation launching, and collection management
processes. Human involvement consists of developing an XSD schema for a simulation system and starting the DL generation process.

For study designers, the digital library hides the complexity of launching and analyzing simulations. The designer may reuse and modify existing sub-configurations, submit simulation requests, make use of data management services, and interact with the simulation infrastructure.

Analysts are able to retrieve datasets and summaries of datasets along with the input conditions from which the results are derived. Automated tasks for analysts include the generation of plots, summary statistics, and the potential for extensions to perform standard analysis tasks useful for a specific institute and simulation system. Explorers are able to query and retrieve content across the entire provenance trail for findings without having to interact with each simulation system component. User’s with this role may discover existing content or identify a lack of previous simulations for a queried segment of the simulation system’s multi-dimensional input space.

5.4.3 Architecture

Rich component profiles are described in [142] as a method of allowing reuse of components when building digital library instances. A component profile, 5S descriptions of a component’s related digital objects, and the context for the component’s required functions allow existing components to be discovered and reused. Detailed component profiles would be useful in generating entire digital library instances similar to existing simulation supporting digital libraries that use different schemas and backend simulation systems. Development efforts are reduced as components, composite functions, and entire digital libraries become reusable. In SimDL, automatically generated storage components are customizable to support a schema (e.g., utilizing RDF or RDBMS access methods by parsing a schema’s XSD structure of elements and attributes). Content in the generated storage component becomes accessible to users by generating internal components to provide access to the repositories through a service API leveraged by the generic UI. The simulation-based class of DLs are most useful in providing rich services when integrated into an existing simulation process, data workflow, and infrastructure.

Authorization and Access

Since large scale data intensive experimental efforts require sufficient resources to be continuing efforts, many organizations conducting such endeavors are often reluctant to share the data produced by their work. This may be a factor in determining the actual wide-spread adoption of a digital library storing data, research studies, experimental metadata, or higher-level documents. The collaborators allowed to access archived data may be decided by a standard organizational policy or user level settings. At a minimum, SimDL is able to enforce strict access privileges of data or metadata to authorized users within a user group.
Chapter 6

Simulation-Specific Services

The digital library technologies developed in this effort have advanced the state of the art in scientific DL services. Historically, large-scale scientific content has proven difficult to manage in a scalable manner. The class of DL produced here provides novel service designs or implementations for simulation indexing, searching, metadata capturing, metadata managing, content reusing, curating, provenance tracking, and incentivizing. Formal definitions provide a basis for describing and developing DL systems, functionality, and instances. Previous publications have formally defined SimDL in terms of content, users, scenarios, and services [87, 117]. Example simulation-specific service implementations include metadata description set generation, automated metadata value extraction, metadata record building, archiving, curation, preservation, ontology building, organization, commenting, tagging, rating, recommending, interface support, and fingerprinting services. The produced formal definitions for SimDL define a DLMS with a broader scope that the implemented DLS requires, e.g., definitions for future work in commenting and rating scientific digital objects.

Fig. 6.1 details an expanded list of services that require novel simulation-specific implementations. Of these services, infrastructure connecting, contextual searching, curating, metadata value extraction, metadata record generating, indexing, management, ranking, weighted-scientific searching, memoizing, and incentivizing have been implemented and evaluated. This set of services covers the minimal set of prototype-required services and provides a diverse set of novel services not available in current DL systems. A subset of the high-level services can be directly implemented with the availability of our existing basic database-centric services. The links between the high-level, layered services and activities denote the hierarchy of service dependencies. Note that search is typically a basic service but requires a novel, advanced simulation-specific implementation. The following services have been implemented in the initial SimDL framework: DLLog, DLRegister, DLIncentivize, DLSearch, DLMemoization, DLCurate, DLCurateDORanker, DLProvRegister, DLProvResolve, DLGetMetadata, and DLGetDLID.
Figure 6.1: Organization of the content workflow, communication interface for content access and service exposure, low-level services, layered high-level services, and user communities.

6.1 Service Layers

The assortment of SimDL services produces a system capable of meeting the digital library requirements for maintaining a set of simulation-based experiments. At the collection-building level, SimDL allows users to submit queries, retrieve documents, contribute content or metadata, and modify content or metadata. The information management layer provides the ability to store and retrieve content related to the entire experimental process. This consists of a collection of archives containing various digital objects mapping to each stage of the experimental process. Between the collection-building and information management layers, a middleware layer provides the context-specific inference process whereby SimDL maps user requests to operations on collections of content. The collection-building system provides the storage and retrieval needs of the digital library’s user base. Applying a metadata index-based approach to content management provides simplistic search on extracted actual data. Data mining on the full set of raw data requires extensions into the data management component that manages the collection of raw digital objects. By separating the services into several layers, higher level services may be rapidly generated which
utilize underlying services.

6.2 Novel Simulation-Specific Services

A key aspect of SimDL is the ability to interface with the simulation software to produce new content that has not been previously simulated. The current generation of commercial and open-source digital libraries does not place a strong emphasis on storing scientific data itself apart from publications. Outside of SimDL, no progress has been made at connecting digital libraries with research infrastructures to directly produce workflows of scientific digital objects. Another service not found in common digital library implementations is the ability to make use of domain knowledge when ranking the resultant set of documents related to a given query. As input parameters found in a domain schema are stored in the metadata for a given set of experiments, researchers may query the system to determine if a given type of experiment has been attempted before. Although storage for diverse content is available in digital library systems, e.g., Greenstone [157] and in [143], the sequential workflow of digital objects required in this context is atypical. Metadata and indexing schemes must provide the ability to store parameters for data, graphs, analyses, and reports for each set of experiments. Existing open source digital libraries lack the functionality necessary to provide these context-specific services.

Collaboration services consist of tasks supporting multiple researchers at various stages of the simulation workflow. Scientific collaboration is a composition of generic and simulation-specific services, see Fig. 6.2. Collaboration consists of tasks for annotating, locating, rating, recommending, reviewing, and submitting content. Locating includes processes for user acquisition of content through browsing, searching, and retrieving. Submitting includes processes for indexing, managing, and disseminating content within a DL. Collaboration reuses generic services for annotating, rating, recommending, and reviewing; locating invokes simulation-specific services for browsing, searching, and retrieving; and submitting invokes generic services for indexing and disseminating. Generic versions of these services for text-based documents are defined in [50]. SimDL currently has implementations for all of the collaboration services except for commenting, rating, and tagging. As all of the epidemiology content in a model-specific collection is produced by the same simulation model, ratings for individual documents are currently gratuitous.

6.3 Infrastructure Communication

Service requests may be sent to the DL through a blackboard system. When SimDL is integrated in this type of infrastructure, messages or service requests are placed in the blackboard to be sent to SimDL. SimDL monitors the blackboard it is registered to and processes messages as they arrive. Results from SimDL are then posted back to the DL and the requesting infrastructural component is notified. The services may be utilized by including the DL binary within a project
Figure 6.2: 5S graph of collaboration and related services.

Table 6.1: Blackboard-based DLBroker service.

<table>
<thead>
<tr>
<th>DL Broker</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>reads ‘BlackboardEntries’ from an assigned Blackboard. The entry must have a type set to ‘DL_Request’. The entry specifies the DLService-Type (i.e., which service is requested) and provides the required service arguments.</td>
</tr>
<tr>
<td><strong>Results</strong></td>
<td>places result blackboard entries on the designated blackboard.</td>
</tr>
</tbody>
</table>

directly for better performance or in infrastructures without a blackboard component. In either case, a DLDataItem object will be returned by either the Blackboard or service invocation. Each DL service utilizes Java Generics for receiving service arguments. A DL service is invoked using the interface’s execute method with the following prototype: public <T> DLDataItem execute(T args). In most cases, a HashMap of argument terms is sent as the argument with specific keys required by the service and along with values for each argument term. Table 6.1 details the utilization of the DLBroker. Fig. 6.3 details the SimDL service request message passing architecture. Request messages (i.e., entries) are placed in a common blackboard, received by the DLBroker, and processed in DLServices.

In certain situations, performance overhead is reduced by avoiding message passing through a blackboard-type system, see Fig. 6.4. In several case studies, the SimDL has been included within external source code in pre-compiled binary formats. In one example, a case study’s user interface required quick response times for user interactions and thus precipitated a bypass of the messaging system through precompiled binaries.
6.4 Curate

In systems that support scientific content, research organizations are increasingly giving long-term management of collections serious consideration. However, semi-automated curation of large scientific repositories has yet to be solved and accepted in the scientific community. For this service, a new means of describing curation policies and semi-automating policy execution was needed. The long-term aspects to management include curating, archiving, and preserving. Curation is complex and composed of deciding what content to maintain and under what conditions and duration it is meant to be kept. Archival is the process of preparing, representing, and storing content to be accessed later with a long-term focus on storage media, format representation, and access tools. Archiving was one of the core functions of formal publication mentioned in [129] and an increasingly common method of distributing previous scholarly articles, albeit without underlying data. Preservation is the process of ensuring that the format, meaning, and accessibility
of content is not lost over time. [74] describes the need to ensure information integrity. This entails that, given a bitstream or structured content, processes are needed to manage the object, dereference the object through identifiers, track provenance information tracing its source, and maintain the context of ways in which it was created and can be used. Various operations may be performed on curated collections, i.e., reformatting, copying, converting, migrating, and deleting. Curation is currently a difficult task as humans are poor at being able to ‘identify the valuable’ and ‘destroy the worthless.’ In [74], archivists are pessimistic about the efficacy of the natural selection of digital information, as people generally do a poor job of keeping the context of digital objects but do keep outdated files without sufficient metadata. Evidential value metadata records the founding and substantive activities of an organization. Informational value metadata records significant people, digital objects, and events. Together, evidential and informational metadata may be utilized to provide DL administrators with the ability to cognizantly perform curation activities.

[69] suggests internal and external quality dimensions that can be extended to assist in curation activities. Internal quality dimensions of content may be used to evaluate the importance of a digital object (DO) to existing processes and services, context, and role in stockholder satisfaction. External quality dimensions evaluate a DO’s importance to patrons of a DL system, expectations of the system, and role in providing DL performance that exceeds expectations. Collections may be analyzed using these criteria to gauge the magnitude of DO value, percentage of utilization, percentage of overall utilization change, costs, resources consumed in production, effect of turnaround time, and anticipatory future utilization. DOs may be evaluated to quantify the cost, benefit obtained, integrity, dependability, accuracy, currency, completeness, comprehensiveness, ease of use, and satisfaction of stakeholders of current DL efforts. The goal of these evaluations is to ensure that a DL maintains useful content and functionality to support current and anticipated purposes related to an organization’s mission. This type of analysis is provided by the DLCurate service.

While librarians have a history of curation-related activities and operations, new curation services are needed in DLs. These curation services are specific to science as simulation content is storage-intensive, well structured, and reproducible under certain conditions. As such, a service that allows curation activities to be defined in a set of rules, and a simple grammar to encode these rules was needed in SimDL. The societies requiring these services span collection agents, managers, curators, administrators, policy designers, librarians, maintainers, and repurposees. The curation service is provided though a combination of components including archived documents, handles/DOIs, metadata records, catalogs, rules, grammars, and curation operators. The goals of an automated curation system are to: a) properly define the required information needed to make curation decisions, b) define appropriate metadata description sets, c) automatically collect curation-related metadata based on a proposed metadata extraction grammar, d) encode policies based on rules defined by a proposed curation grammar, and e) automatically apply curation policies to scientific collections. Later portions of this chapter present our framework and theoretical definitions for DL services that produce self-generating metadata collections, self-indexing scientific content through metadata extraction, and self-curating collections through policy automation. Note that self-indexing here refers to the inclusion of values into metadata records regarding specific content, as opposed to popular database indexing for efficient database functions, as demonstrated in [148].
Scientific content produced as output from HPC simulation systems may scale to infeasible storage requirements in many institutions. Large simulation results datasets may be summarized or reproduced in many cases. Content curation provides a means of establishing policies for content removal as well as arbitrary user-specified deletion. Simulation-based *curation* requires processes for identifying removable documents, removing documents, and either removing the document’s index entries or defining a method of re-generating the document. As input, *curation* requires a set of document handles in a metadata index, \( \{h_x, \ldots, h_y\} \); a filter or query for pre-defined operation collection identification, \( \{q, C_i\} \); and a specification for the type of curation operator, \( co \), e.g., archiving, deleting, migrating, and preserving. Specifications for documents that should remain indexed but physically removed for storage reclamation additionally require input for a regeneration method consisting of a software executable, \( sw_i \), and input configuration, \( icfg \). The output of *curation* is a suggestion for a modified collection, \( C_i \mapsto C_i' \); and altered index for the collection, \( I_{C_i} \mapsto I_{C_i}' \). Pre-conditions for *curation* are \( \forall h_i \in \{h_x, \ldots, h_y\} : \exists do_i, C_i : do_i \in C_i \land h_i = do_i : C_i \in Coll \). Post-conditions for *curation* are not strict as it is possible for no DOs to match triggering criteria. In addition, each curation operator will have an alternative set of post-conditions, e.g., deletion produces a reduced collection and migrating increases the number of items represented and stored. SimDL’s default policy is to maintain all information and rely on infrastructure designers to define processes for result summarization and dataset deletion. See Table 6.2 for a description of the behavior of the curation service.

**Table 6.2: Formal service definitions for curating (deletion of content operation).**

<table>
<thead>
<tr>
<th>User input</th>
<th>Other input</th>
<th>Output</th>
<th>Pre-conditions</th>
<th>Post-conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( {h_x, \ldots, h_y} ), ( C_i, co )</td>
<td>( q )</td>
<td>( C_i', I_{C_i}' )</td>
<td>( \forall h_i \in {h_x, \ldots, h_y} : \exists do_i, C_i : q \subseteq I_{C_i} \land do_i \subseteq C_i \land h_i = do_i : C_i \in Coll )</td>
<td>( C' \subseteq C : I_{C_i}' \subseteq I_{C_i} : \forall do_i \in C', do_i \in C )</td>
</tr>
</tbody>
</table>

**Curation Activities**

Digital curation is a set of policies and practices developed for the maintenance and effective management of digital content over time. Curation elicits the need for coordinating preservation, archiving, and long term access activities. Curation in scientific domains “requires a blend of skills and experience, including advanced scientific research and competence in database management systems, multiple operating systems and scripting languages” [72]. Curation is very important in the scientific simulation context because of the sheer volume of data generated. To run experiments, data needs to be processed and transformed. Also, without enough documentation about the data and its meaning, the data is rendered useless for future users. This process of transformation and subsequent documentation is laborious and time-consuming. Also, there are few standards that deal with how to document these different processes. [72] describes various advances in the area of biological data curation and suggests developing a structured vocabulary of data collected from various sources. The Digital Curation Center describes a stepwise digital curation life cycle and recommends undertaking a planned effort to promote curation and preservation throughout the data.
Archiving consists of organizing, preserving, and providing access to selected content. Due to the nature of simulation-based research, there is often a trade-off between saving files, programs, and other materials [71]. Some simulation-based content may not require archiving when data can be regenerated, given a software binary and an input configuration. This caveat found in simulation-systems allows for lossless data deletion, albeit with greatly increased access latency due to the regeneration of files.

Preservation aims to ensure that content remains consistent over its lifecycle. In [55], data is classified into two categories, ephemeral and stable. The classification is based on the capability to regenerate data. Ephemeral data cannot be regenerated or recomputed and must be preserved if valued. In contrast, stable data can be recomputed from other existing data. The decision regarding preservation of stable data is based on other factors, such as economic and financial feasibility, storage limitations, etc. Metadata is generally categorized as ephemeral data. Simulation results and analyses typically are stable data. Policies for conversion (i.e., copying files and programs to each new system as new systems are introduced) and emulation (i.e., saving the data and the program as a bit stream, along with a description of the original machine architecture) [98] may be processed by curation rules. As discussed in [147], the task of preservation should assure the maintenance of the value (both syntactic and semantic) of the digital content at any point of time. The laws of library science are applicable in this regard [125]. [147] gives a detailed list of several different factors that need to be a focus of a comprehensive curation effort along with essential curation strategies designed to foster desirable curation values.

Over long periods of time, research institutions lose the ability to simply keep and maintain all of their produced content. Reliance on individual researchers to properly maintain content is not sustainable, due to matriculation, hardware failures, improper duplication strategies, and lack of dissemination. Institute-specific curation policies are needed to reduce data management costs, disk utilization, wasted regeneration of existing content, and duplication of human and computational costs. Curation policies may be used to automate removal of seldom accessed content and maintain heavily accessed or high-value content to maximize the future potential to reuse datasets. Activity based costing frameworks for digital curation are derived from the OAIS reference model [81]. The cost estimation is based on cost-critical activities identified by the functional break-down provided by the OAIS model. Often, DB administrators have the impossible task of identifying data, produced by others, that may be safely deleted from storage. With a new class of self-curating collections, user and DBA acceptance is required to allow for curation policy automation. Rules are required to prevent accidental deletion of unrecoverable content. Due to changes in institutions and infrastructures, curation policy rules will need to be continually updated. Following is a definition of curation, a proposed curation grammar, and a representative set of example curation rules.

**Formal Definition**

Curation is a tuple defined as $curation = (CDO, G, C)$, where:
• CDO is defined as a tuple denoting the set of digital objects to be considered for curation;
• \( G \) is a set of curation grammar rules, \( \{ g_1, \ldots, g_n \} \), applicable on the set of digital objects; and
• \( C \) is the output of the curation task indicating the sorted/modified collection of digital objects after executing the grammar rules, \( G \), on the input digital objects collection, \( CDO \).

A metadata index, or record collection, may be defined as a tuple \( MDOI = ( C, MD ) \) where:

• \( C \) is a collection;
• \( MD \) is a set of metadata terms, \( \{ md_i, \ldots, md_n \} \);
• \( md_i \) is a tuple defined as \( md_i = ( M, V ) \);
• \( M \) is a set of metadata types, \( \{ m_1, \ldots, m_n \} \), where each \( m_i \) is a scalar alphabetic value string indicating a metadata type (e.g., file size, file type, file extension, format, regeneration cost, replicability, percent diff, usage, and lastUpdated); and
• \( V \) is a tuple, \( ( v_1, \ldots, v_n ) \), or a vector of values \( v_1 \) to \( v_n \) associated with each metadata type from \( M \) for each digital object in a collection.

A partial decision clause is a tuple \( pdc = ( MDOI, W, T ) \) where

• \( MDOI \) is a metadata index;
• \( W = \) is a set of weights, \( \{ w_1, \ldots, w_n \} \), assigning a value, \( w_i \in [0, 1] \), to be associated with metadata type \( m_i \) in \( M \) within \( MDOI \); and
• \( T \) is a function of a metadata type, within the domain of \( M \), that returns an acceptable set of values or a singular threshold value associated with a particular metadata type.

Partial decision clauses are specific to metadata terms and are used to build curation policy rules. These decision clauses are set at the collection level to enable weighting and comparison of metadata terms for ranking, similarity searching, and curation policies.

A Boolean importance value function is a tuple \( bivf = ( MDOI, PDC, IV ) \), where

• \( MDOI \) is a metadata index for a collection \( C \);
• \( PDC \) is a set of partial decision clauses, \( \{ pdc_0, \ldots, pdc_n \} \), providing a weighting for each metadata term in \( M \); and
• \( IV \) is a set of weighted values, \( \{ iv_0, \ldots, iv_n \} \), where \( iv_i \) is the score of \( do_i \in C \) in relation to a given set of partial decision clauses.

The Boolean importance value function may be utilized to rank a set of documents based on storage cost, regeneration cost, reproducibility, etc.
Curation Grammar

The curation grammar was constructed based on an enumeration of potential curation scenarios. From the set of curation scenarios and formalized definition of curation, an abstract set of vocabulary terms and a rule encoding structure were developed.

Example Grammar Vocabulary

- \( \text{has}(m) \): Returns true if a metadata structure contains a metadata type \( m \).
- \( \text{value}(m) \): Returns the value of metadata type \( m \). If \( m \) does not exist, returns \text{null}.
- \( \text{contains}(m, \text{value}_\text{set}) \) : If \( \text{value}_\text{set} \) contains \( \text{value}(m) \), returns \text{true}, otherwise, it returns \text{false}.
- \( \text{threshold}(m) \) : Executes and returns the threshold value for a metadata type. The threshold could either be a single value or a set of values, depending upon how \( T_M \) is defined.
- \( \text{delete} \) : Marks object for deletion.
- \( \text{preserve} \) : Marks object for preservation.
- \( \text{archive} \) : Marks object for archiving on external storage.
- \( \text{THRESHOLD} \) : Globally defined threshold value, e.g., minimum importance value required for preservation to prevent disk space issues.

Example Grammar Rules

1. Delete content over a certain size: \( \text{if value(file_size) > threshold(file_size)} \rightarrow \text{delete} \)
2. Delete unnecessary files by type: \( \text{if contains(file_extension, \{tmp, DS_Store\})} \rightarrow \text{delete} \)
3. Preserve a set of human generated files by extension: \( \text{if value(file_extension) \in \{set_of_file_extensions\}} \rightarrow \text{preserve} \)
4. Archive older, unused content: \( \text{if (value(usage) < threshold(usage)) \\&
\&
(value(lastUpdated) > threshold(lastUpdated))} \rightarrow \text{archive} \)

With the methodology for describing curation policies as provided by the curation grammar and rules, digital libraries, such as SimDL, may periodically apply the current set of curation policies on a collection, or due to an external event (e.g., low disk space). Items deemed suitable for removal may be removed from a collection as well as the metadata index. The curation policies and strategies of an institution are likely to change over time. Given this, a grammar for describing new policies and succinctly encoding curation rules is pertinent. After the curation service has processed a given
Table 6.3: DLCurate service.

<table>
<thead>
<tr>
<th>DLCurate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>requires no arguments as all known curation rules are executed.</td>
</tr>
<tr>
<td>Results</td>
<td>contains a ArrayList&lt;HashMap&lt;String,String&gt;&gt; data item consisting of four indexes. Each index in the list contains a HashMap&lt;String,String&gt; with keys for ‘curationType’, ‘DLID’, and ‘Score’. This HashMap contains a list of the IDs for a set of digital objects. The value of ‘curationType’ key is a DLCurateAction enum value (i.e., preserve, delete, archive, or migrate). Thus, each HashMap contains the set of digital objects recommended for the corresponding curation action.</td>
</tr>
</tbody>
</table>

collection, sets of DOs are recommended for a set of curation operations, i.e., deletion, archiving, migration, and preservation. These actions are not performed automatically to prevent the accidental loss of data. Hence, a DL administrator must take the handles for the recommended actions and perform these actions manually. See Table 6.3 for documentation on the curate service.

### 6.5 Digital Object Collection Ranking

The collection ranking service is a lower level service utilized by the curation service. This service applies a set of Boolean clauses to a collection and returns a ranked list of content based on the metrics found in the clauses. This service is repurposed in the incentivization service’s reports to give contributors a metric on their data products. See Table 6.4 for more details on the behavior of the service.

### 6.6 Filter

Filtering is often required in lists displayed in user interfaces. In fact, it is the only service listed as a minimal requirement in all of SimDL’s case studies. Filtering is conducted to return an unranked list of content handles that exactly match a search criteria. The distinction between filtering and search is that search returns ranked lists of documents partially matching a query. In simulation contexts, filtering may be used to restrict results to particular users, groups, user roles, and sub-documents. In this last case, sub-documents might refer to a particular sub-set of input configurations parameters or results from a particular study. The formal definition of filtering is included in Table 6.5. Here, \( q \) is a \( k \)-term query, \( C \) is a collection, \( ids \) are in a set of DO identifiers, and the set of output DO’s metadata values exactly match the query.

See Table 6.6 for a description of the behavior of the filtering service.
Table 6.4: DLCurateDORanker service.

<table>
<thead>
<tr>
<th>DLCurateDORanker</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>requires an ArrayList&lt;HashMap&lt;String, Object&gt;&gt; representing a list of Boolean partial decision clauses and a HashMap&lt;String, String&gt; with a ‘combinationType’ key with an associated value for DLCurateCombinationType. Each partial clause in the list is encoded by a HashMap&lt;String, Object&gt;. This partial clause’s HashMap must contain a ‘weight’ key and associated value of type Double representing the importance of this clause. The partial clause’s HashMap must also contain an ‘object’ key and associated value of type DLCurateBooleanPartialDecisionClause. The combination type indicates the method used to combine the multiple ranked lists for each partial clause and is selected from the DLCurateCombinationType enum.</td>
</tr>
<tr>
<td><strong>Results</strong></td>
<td>contains a HashMap&lt;String, Double&gt; with keys containing unique DLIDs for digital objects. The related values are the importance value of each digital object based on the provided partial decision clauses.</td>
</tr>
</tbody>
</table>

Table 6.5: Formal service definitions for filtering.

<table>
<thead>
<tr>
<th>User input</th>
<th>Other input</th>
<th>Output</th>
<th>Pre-conditions</th>
<th>Post-conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0, ..., q_k$</td>
<td>$C_i, I_{C_i}$</td>
<td>${id_0, ..., id_n}$</td>
<td>$k &gt; 0 : \forall md \in q, md \in I_{C_i}$</td>
<td>$\forall md \in q, q_{md} = id_{md}$</td>
</tr>
</tbody>
</table>

### 6.7 Get DLID

The service to get a DL identifier is a lower level service utilized by all of the collection building services. It simply produces unique identifiers to be used as handles for DOs in specific stages of the simulation workflow. See Table 6.7 for the formal definition, where $h$ is a unique DO handle and $DB_{seq}$ is a DBMS sequence generator, e.g., of Oracle sequences.

This service supports the automated collection building services within an infrastructure. The identifier is utilized in numerous external components in the case studies to retrieve specific metadata and o link records to raw data being stored in the metadata broker. See Table 6.8 for a brief description.

Table 6.6: DLFfilter service.

<table>
<thead>
<tr>
<th>DLFfilter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>requires a query HashMap&lt;String, String&gt; with keys consisting of RDF metadata ‘predicate’ keys and values consisting of RDF metadata ‘object’ values.</td>
</tr>
<tr>
<td><strong>Results</strong></td>
<td>contains an ArrayList&lt;String&gt; where each string is a unique DLID ‘subject’ found in the RDF metadata.</td>
</tr>
</tbody>
</table>
Table 6.7: Formal service definitions for getting a DL identifier.

<table>
<thead>
<tr>
<th>User service input</th>
<th>Other input</th>
<th>Output</th>
<th>Pre-conditions</th>
<th>Post-conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>DBseq</td>
<td>h</td>
<td>$h \not\in C$</td>
<td>$h \in DB_{seq}$</td>
</tr>
</tbody>
</table>

Table 6.8: DLGetDLID service.

<table>
<thead>
<tr>
<th>DLGetDLID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>requires no arguments.</td>
</tr>
<tr>
<td>Results</td>
<td>contains a HashMap $&lt;$String,String$&gt;$ where the ‘DL_ID’ key is associated with a newly generated unique DLID.</td>
</tr>
</tbody>
</table>

6.8 Get Metadata

The service to get the metadata for a particular digital object is a low level service. Given an RDF triple subject, the service returns the full listing of predicates and objects related to the subject. The formal definition is found in Table 6.9 where a query and Collection are transformed into a set of metadata from the metadata index. It is utilized by the search, filtering, and incentivizing services.

Table 6.9: Formal service definitions for getting metadata.

<table>
<thead>
<tr>
<th>User input</th>
<th>Other input</th>
<th>Output</th>
<th>Pre-conditions</th>
<th>Post-conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q$</td>
<td>$C_i, I_{C_i}$</td>
<td>${md_0, \ldots, md_j}$</td>
<td>$\exists q, C$</td>
<td>$q \in md : \forall md_j, md_j \in I_{C_i}$</td>
</tr>
</tbody>
</table>

See Table 6.10 for a description of the services behavior.

Table 6.10: DLGetMetadata service.

<table>
<thead>
<tr>
<th>DLGetMetadata</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>requires a query HashMap $&lt;$String,String$&gt;$ with a ‘subject’ key and associated value of the RDF triple ‘subject’ for which to get metadata.</td>
</tr>
<tr>
<td>Results</td>
<td>contains a HashMap $&lt;$String, String$&gt;$ where each key is an RDF metadata ‘predicate’ and the associated value is the metadata ‘object’.</td>
</tr>
</tbody>
</table>

6.9 Incentivize

Community building is enhanced with services that encourage and reward humans and systems to contribute content, software, resources, activities, and efforts. As such, the high-level incentivization service generates reports along these lines. While considerable research has gone into growing online communities, these services do not exist to encourage scientists to contribute and utilize the type of applications and infrastructures supported by SimDL. The service makes use of logs and
tracks the usage of content, software, and services; and provides reports to producers and system administrators. The societies utilizing this service include collection agents, administrators, authors, publishers, and contributors. The activities supported through incentivizing include publicizing the usefulness of contributions, marketing resources, and advertising systems as a repository of entities. The components involved include collections, resources, activity logs, metrics, statistics, and reports. The formal signature of incentivizing is given in Table 6.11 where a query covers a specific collection $C$, set of reports $R$, and set of statistics or metrics $M$ for each report. The data products are a (possibly null) bytestream encoding a textual report.

<table>
<thead>
<tr>
<th>User service input</th>
<th>Other input</th>
<th>Output</th>
<th>Pre-conditions</th>
<th>Post-conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q$</td>
<td>$C_i, R, M$</td>
<td>$str$</td>
<td>$\exists C_i, R, M : \forall M_m, M_m \in R_r$</td>
<td>$R \rightarrow str$</td>
</tr>
</tbody>
</table>

The set of statistics provided for content producers and resource contributors in the CINET case study is provided in Table 6.13. These statistics are built from lower level logging of each service that touches a particular digital object and the context related to the access event. The service creates simple reports on the utilization activities and rates of various components. More complex reports execute multiple queries, e.g., that your usage has gone up over time, the kinds of people who are using contributed content, how high a rating a contribution received, and how you compare to the top ranked person. The CINET case study includes seven report templates that include 45 statistics. The seven reports are targeted to providers, graphs, software, results, HPC, users, and system infrastructure. See Table 6.9 for a further description of the service.

<table>
<thead>
<tr>
<th>DLIncentivize</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>requires a query HashMap&lt;String, String&gt; with a “reportType” key and associated value of a type of report to generate as selected from DLIncentivizingReportType.</td>
</tr>
<tr>
<td>Results</td>
<td>contains a HashMap&lt;String, String&gt; where the “result” key is associated with a single String containing the requested report.</td>
</tr>
</tbody>
</table>

Table 6.12: DLIncentivize service.

### 6.10 Log and Register

Logging and registering are considered the same service in SimDL with a slight differences in semantics. Logging activities are restricted to the automatic recording events that produce content, e.g., a simulation configuration, contribution, experimentation or result is generated. Registering is the explicit semi-automatic or manual recording of DOs in the DL after they have been processed by another entity. Both of these activities leverage the same underlying collection building services,
Table 6.13: Partial statistics in CINET incentivizing.

<table>
<thead>
<tr>
<th>Category</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provider</td>
<td>number of graphs provided&lt;br&gt;number of software applications provided&lt;br&gt;average uses per provided graph&lt;br&gt;average uses per provided software&lt;br&gt;cumulative uses total&lt;br&gt;breadth vs. depth ratio: number of studies / number of replications</td>
</tr>
<tr>
<td>Graph</td>
<td>datestamp provided&lt;br&gt;studies used in&lt;br&gt;count of individual replications&lt;br&gt;usage distribution graph over time&lt;br&gt;bibliometrics - downloads/use last x weeks, last x months&lt;br&gt;breadth vs. depth: used in many measures vs. thorough parameterization&lt;br&gt;percentage of measures used with this graph</td>
</tr>
<tr>
<td>Software</td>
<td>datestamp provided&lt;br&gt;total times used in studies&lt;br&gt;use distribution over time&lt;br&gt;number of products generated&lt;br&gt;what graphs they were paired with&lt;br&gt;percentage of graphs used with this measure&lt;br&gt;percentage extent of parameterization</td>
</tr>
<tr>
<td>Results</td>
<td>times reused&lt;br&gt;times ‘memoized’&lt;br&gt;reuse distribution over time&lt;br&gt;bibliometrics - downloads last x weeks, last x months, etc.&lt;br&gt;generation / computation time&lt;br&gt;HPC cycles needed to generate the result&lt;br&gt;storage footprint&lt;br&gt;potential collaborators recommendation</td>
</tr>
<tr>
<td>HPC</td>
<td>jobs submitted&lt;br&gt;number of replicates&lt;br&gt;number of studies executed&lt;br&gt;percentage of studies executed on this platform&lt;br&gt;running time&lt;br&gt;data size in/out&lt;br&gt;memoization count, ratio, etc.</td>
</tr>
<tr>
<td>Systems</td>
<td></td>
</tr>
<tr>
<td>Users</td>
<td>track and provide feedback on end-users or personalization</td>
</tr>
<tr>
<td>System</td>
<td>job throughput rates, job throughput delay, expected completion time, wait</td>
</tr>
</tbody>
</table>
known as DLLog and DLRegister. Thus, the differences are only in the processes leading to the service call. The key requirement for these services is to allow the DL to monitor Siminfrastructure research-based events. The formal definition is provided in Table 6.14. See Table 6.15 for a description of the service.

Table 6.14: Formal service definitions for logging.

<table>
<thead>
<tr>
<th>User service input</th>
<th>Other input</th>
<th>Output</th>
<th>Pre-conditions</th>
<th>Post-conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>DO, {md_0, ..., md_m}</td>
<td>(C_i, I_{C_i})</td>
<td>none</td>
<td>(\exists C_i, I_{C_i} )</td>
<td>{md_0, ..., md_m} (\in I_{C_i}'), (DO \in C_i')</td>
</tr>
</tbody>
</table>

### 6.11 Memoization

Conducting large simulation study designs requires large amounts of resources, e.g., HPC systems, cycles, and scratch space. In addition, they often take days of human-intensive efforts to design and a delay of several days or weeks while the study is executed. The memoization service implements a dynamic programming approach to linking input configurations to results for a given simulation system. Thus, the design space of a simulation study or request that overlaps with existing content may leverage the DL’s repository of existing results. This service provides researchers with shorter study turnaround times, immediate access to results, and discovery of similar existing studies. It also lessens consumption of HPC systems and frees cycles for novel computations. The service provides a means of storing and reusing simulation results while avoiding repeated deterministic calculations. Memoization is a high-level service relying on collection building, getting metadata, metadata extraction, and filtering services. The formal definition is provided in Table 6.16 where a set of metadata terms is matched with the set of data manager IDs having identical metadata terms. See Table 6.17 for additional details. In the SimDL toolkit implementation, a list of identifiers is provided for matching raw digital objects found in the data manager storage mechanism. The calling service may access the raw data files by secure copying or through the blackboard system as appropriate.

Table 6.15: DLLog and DLRegister service.

<table>
<thead>
<tr>
<th>DLLog</th>
<th>DLRegister</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>requires a metadata HashMap(&lt;\text{String, String}&gt;) containing metadata term “predicate” keys and associated metadata value “object” values. The RDF “subject” value is not needed as the DL automatically generates a new “subject” DLID and attaches to each metadata record.</td>
<td></td>
</tr>
<tr>
<td>Results</td>
<td>contains a HashMap(&lt;\text{String, String}&gt;) where the “DLID” key is associated with the String representation of an integer containing the number of RDF metadata triples added in this logging activity. A non-negative result indicates successful execution.</td>
<td></td>
</tr>
</tbody>
</table>
Table 6.16: Formal service definitions for memoization.

<table>
<thead>
<tr>
<th>User service input</th>
<th>Other input</th>
<th>Output</th>
<th>Pre-conditions</th>
<th>Post-conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$DO, q, {md_0, ..., md_i}$</td>
<td>$C_i$</td>
<td>${h_0, ..., h_j}$</td>
<td>$md, h \in C_i$</td>
<td>$\forall md, h_{md} = q_{md}$</td>
</tr>
</tbody>
</table>

Table 6.17: DLMemoization service.

<table>
<thead>
<tr>
<th>DLMemoization</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>requires a query HashMap $&lt;$String,String$&gt;$ with keys consisting of RDF metadata ‘predicate’ keys and values consisting of RDF metadata ‘object’ values.</td>
</tr>
<tr>
<td>Results</td>
<td>contains an ArrayList $&lt;$String$&gt;$ consisting of data manager ‘SID’s.</td>
</tr>
</tbody>
</table>

6.12 Provenance Register and Resolve

Provenance tracking provides the ability to discover the source and generation process for a DO. The service utilizes a metadata index of provenance links between DO handles to capture the workflow of content. Simulation systems produce a stream of content at multiple stages including model schemas, input configurations, results, analyses, documentation, annotation, and publication. Preserving provenance information from findings and results is necessary for simulation researchers. Provenance is maintained by defining each relationship between content and harvesting content collections for each stage of the experimentation process. Similar to indexing a document $do_i$ in an index $I_{C_i}$, provenance information can be encapsulated in an index structure where two sets of documents, $\{do_i, ..., do_k\}$ and $\{do_j, ..., do_m\}$, are indexed in $P_{\{do_i, ..., do_k\}, rel_{i}, \{do_j, ..., do_m\}}$. Provenance streams in a collection may be defined by the tuple $P = (h_{\{do_i, ..., do_k\}}, \{rel_{i}, ..., rel_{k}\}, h_{\{do_j, ..., do_m\}})$, where

1. $h_{\{do_i, ..., do_k\}}$ represents handles for a preceding set or type of digital objects from $C_i$, a collection in $Coll$;
2. $rel$ is a provenance relation, where $rel_i$ is a relation between $do_i$ and a subset of $(do_j, ..., do_m)$;
3. $h_{\{do_j, ..., do_m\}}$ represents handles for a successor set of digital objects from $C_j$, a collection in $Coll$; and
4. $\{do_i, ..., do_k\} \in C_i$; and $\{do_j, ..., do_m\} \in C_j$.

The formal definition is completed in Table 6.18 where two DOs are linked by a provenance relationship term. See Table 6.19 for more details on the provenance registration service. See Table 6.20 for more details on the provenance resolver service.
Table 6.18: Formal service definitions for provenance tracking.

<table>
<thead>
<tr>
<th>Registering</th>
<th>User service input</th>
<th>Other input</th>
<th>Output</th>
<th>Pre-conditions</th>
<th>Post-conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>do_i, do_j, rel</td>
<td>C_i</td>
<td>none</td>
<td></td>
<td>do_i, do_j ∈ C</td>
<td>(do_i, rel, do_j) ∈ C'</td>
</tr>
</tbody>
</table>

Table 6.19: DLProvRegister service.

<table>
<thead>
<tr>
<th>DLProvRegister</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>requires a query HashMap&lt;String,String&gt; with keys ‘subject’, ‘dlprovtype’, and ‘object’. The values associated with ‘subject’ and ‘object’ must be valid assigned DLIDs. The value associated with ‘dlprovtype’ is selected from DLProvType and should be read as a left to right statement (e.g., ‘subject DLID’ ‘precedes’ ‘object DLID’).</td>
</tr>
<tr>
<td>Results</td>
<td>contains an Integer describing the number of new provenance-related metadata triples that were registered (e.g., 0 or 1).</td>
</tr>
</tbody>
</table>

6.13 Search in Scientific Collections

Search over scientific collections required novel techniques and algorithms to process highly numeric, scientific content with little to no free text. In the course of several decades of information retrieval research, several thousand weighting schemes have been introduced and tested, see [26]. Salton experimented with term weights in the 1960s though most active schemes have been produced from the 1970s [131]. Salton, Allen, and Buckley summarized the 20 years of weighting work conducted within the SMART system [132]. In this work, over 1,800 term weighting factors were tested experimentally. A detailed overview of similarity functions (e.g., cosine) is given in [155]. Historical work in this area has proposed approaches to improving search, e.g., change the document

Table 6.20: DLProvResolve service.

<table>
<thead>
<tr>
<th>DLProvResolve</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>requires a query HashMap&lt;String,String&gt; with keys ‘subject’ and ‘dlprovtype’. The value associated with ‘subject’ must be a valid assigned DLID. The value associated with ‘dlprovtype’ is selected from DLProvType and should be read as a left to right statement (e.g., find results X, where ‘subject DLID’ ‘precedes’ X).</td>
</tr>
<tr>
<td>Results</td>
<td>contains an ArrayList&lt;String&gt; containing RDF metadata triple ‘objects’ that have the requested DLProvType relationship ‘predicate’ with the requested ‘subject’.</td>
</tr>
</tbody>
</table>
collection through document transformation, reformulation of the query, modifying term weights, query splitting, query expansion, and (Rocchio and pseudo) relevance feedback. This treatment of discovery services is split into searching, ranking, browsing, and dissemination.

At the core of the discovery services is an RDF-triplestore metadata index. With this index, activities such as filtering, providing access, archiving, requesting, searching, and selecting are supported. The index is utilized to discover a set of handles or DOIs in a catalog that are likely to be relevant to a request. Ranking is a complex process typically relying on probabilistic information retrieval, degree of similarity, metric spaces, and indexing services. With known metadata indexes, we can evaluate the accuracy, pertinence, relevance, precision, and recall scores of search implementations.

**Browsing** scientific content is similar to browsing textual documents with the added assumption that documents will be displayed in a faceted design based on model and workflow stage. SimDL has implemented faceted browsing for a clustered document space based on the simulation model source and the stage of the workflow producing each type of content as described by existing ontologies. Browsing requires as input an initial simulation facet, $\text{anchor}_f$, and potential targets, $\text{Hyptxt}_j$. Browsing produces as output a set of digital objects, $\{\text{do}_i : i \in I\}$. The pre-condition for browsing is that the anchor must exist in the network of hyperlinks, $\text{anchor} \in \text{Hyptxt}_j$. The post-condition restricts the resulting digital objects to existing documents in the collection valid for $\text{Hyptxt}_j$, $\exists C \in \text{Coll} : \{\text{do}_i : i \in I\} \subseteq C$.

Existing search functions are well suited for full-text documents and textual metadata. Simulation content requires search over numerical metadata and summarized content. The metadata for typical SimDL content consists of numerical values for metadata terms. As an example, the metadata schema for a collection of input files to a simulation model will largely consist of selected input terms. Traditional similarity metrics break down in this situation as the semantics and similarity of multiple metadata values are non-trivial to determine. Search implementations relying on text-based metrics, such as term frequencies, thesauri, dictionaries, stemming, and term co-occurrence, break down in scientific digital libraries. However, $k$-length queries may be segmented into $k$ Boolean clauses for numerical metadata terms. In this type of querying system implemented for SimDL, a similarity score between a document and query is provided by a weighted summation over the $k$ clauses as presented in [88]. The formalism for scientific search is identical to full-text search as a query ($q$), collection ($C_i$), and index ($I_C$) are required to produce a set of weighted results, $\{(\text{do}_i, w_{qk}) : q_k \in q\}$. A detailed formal definition of the technical aspects used to implement a search service are required to differentiate between full-text and numerical-based search mechanisms.

In ranking and searching of configurations, datasets, analyses, or documents in experiment supporting DLs, it is useful to rank documents for queries based on annotations or selected input and analysis parameters. Query processing in a high-dimensional space scheme might rank documents based on how each document’s icfg parameter values match a Boolean or weighted query. While this ranking scheme is sufficient in providing a generic search mechanism, it is intended as a starting point for interested developers to customize search functions based on context specific factors accompanying a context schema. Through semantic ties across the multiple collections of experiment-based content, it is possible to retrieve the stream of documents that constitute a
provenance trail of research findings. The search functions used in SimDL are a high-level service relying on previous collection building activities and available metadata indexes. The query agent, i.e., human or infrastructure component, specifies a data type, query model, and $k$ query terms. The internal algorithm uses Boolean weight functions to rank scientific metadata terms. The exact form is

$$\text{similarity}(Q, D_j) = \frac{\sum_{i=0}^{k} p_i \cdot v_{i,j}}{\sum_{i=0}^{k} p_i}$$

where

- $p_i =$ user defined parameter weight, default $1/k$ and
- $v_{i,j} =$ Boolean match of term $i$ in RDF metadata for $j$.

This implementation scales linearly for metadata index scales and completes in .14 seconds on a small collection of several hundred items. The current implementation has a limitation that $k$ separate queries and combination are performed and will need to be optimized in distributed implementations. The strength of the algorithm is that it works well with scientific metadata schemas and highly numeric content. The formal definition is completed in Table 6.21. See Table 6.22 for a description of the search service.

**Table 6.21: Formal service definitions for searching.**

<table>
<thead>
<tr>
<th>User service input</th>
<th>Other input</th>
<th>Output</th>
<th>Pre-conditions</th>
<th>Post-conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0, ..., q_k, w_0, ..., w_k$</td>
<td>$C_i$</td>
<td>${(h_0, w_0), ..., (h_n, w_n)}$</td>
<td>$\forall q_i, q_i \in I_{C_i}$</td>
<td>$\forall h_i, h_i \in C_i : \forall w, 0.0 \leq w \leq 1.0$</td>
</tr>
</tbody>
</table>

*Dissemination* of scientific content consists of exposing content and providing the ability to retrieve specific content. *Exposing* is a composition of general services for acquiring, cataloging, indexing to support search functions, and classifying to support faceted browse functions. *Retrieval* consists of requesting a document, $\exists C \in Coll : do_i \in C$; identifying handles for the requested document, $h_i = do_i$; and provision of the document as output, $\{do_i\}$. The pre- and post-conditions for retrieval are $\exists do_i, C : do_i \in C \land h_i = do_i$ and $do_i = h_i$, respectively. Exposing and retrieval are fully supported in our implementation of SimDL.

The use of schemas by SimDL allows digital library services to make use of contextual information within a domain. Discovery is improved as DLs understand the environment in which queries are performed. This understanding is accomplished by restricting the search space to a context defined by an input schema and accompanied by additional domain information (e.g., the how, what, where, and why of anticipated queries). Similarity is based on distance in a high dimensional space that includes input parameters, annotations, and simulation-related metadata. By restricting a DL’s focus to content within a context, similarity ranking may be intelligently assisted by domain experts. Contextual ranking becomes important as content is searched from large systems and collections. The institutes involved in SimDL’s development had previously limited metadata to basic information describing simulation runs and a brief tag entered by study designers or analysts without advanced experience in computing or data management. Pairing user annotations with
Table 6.22: DLSearch service.

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLSearch</td>
<td>requires a complex structure to represent the weighted query, ArrayList&lt;HashMap&lt;String, String&gt;&gt;. Each item in the ArrayList represents a single weighted search clause (e.g., a four term query might have: metadata predicate “graphProvidedBy”, metadata object “jleidig”, and weight “.25”). Each clause is encoded in a HashMap with keys “term” associated with the RDF predicate term identifier value (graphProvidedBy), “value” associated with the RDF object value (“jleidig”), and “weight” associated with a string representing a double value for the individual’s importance. The weights are combined based on an arithmetic mean. Thus, a straightforward weight assignment of a k-clause query might include weight values between 0.0 and 1.0 with a summation of all of the weights equal to 1.0. Alternatively, a 3-clause query with weights 2.0, 3.0, and 4.0 would have a total weight of 9.0 and possible similarity scores of 0.0, .22, .33, .44, .55, .66, .77 and 1.0.</td>
</tr>
<tr>
<td>Results</td>
<td>contains a HashMap&lt;String, Double&gt; where each key is a ‘DLID’ for a digital object and the associated value is a numerical value representing the similarity of the digital object to the query.</td>
</tr>
</tbody>
</table>

SimDL’s schema-based contextual information results in improved querying and discovery in comparison to previous browsing functions. By automating the RDF-based storage design to match a contextual schema, the storage structure mirrors the semantics of a context and allows processes with access to the schema and storage system (e.g., data mining algorithms) to intelligently perform operations on existing data. Though requiring additional effort from a domain and simulation system expert, DL component and service optimization may occur by taking advantage of contextual information to alter the weights in a Boolean query.

### 6.14 Supporting Metadata Functionality

Supporting metadata using this approach involves processes for generating metadata description sets (schemas), extracting metadata values, forming metadata records, and indexing metadata records. In simulation-based science, hierarchical multi-resolution access to data is required and calls for careful metadata planning. “Data analysis must ultimately be done at the highest available spatial and temporal resolution, yet there is far too much data for all of it to be based at this level. Therefore, the analysis process often requires a number of intermediate stages” [113]. Extracted metadata can be considered on a plane one level above raw data. An RDF metadata graph can be indexed in simple DL storage and linked to data stores of DOs. The overall approach is to store a raw level of data, a level 2 description with details on the DO’s generation process, level 3 data approximation with extracted metadata, and finally a level with human-generated tags.
Digital libraries universally require processes to index metadata for supported collections. The performance of DL services is greatly influenced by the system-wide method of content indexing. Individual services typically cannot take advantage of optimally defined metadata indexes. What is needed is a framework for indexing content in graphs, defining the metadata required by services, and connecting components to interact with SimDL. Here, I present a graph-based index and services framework for digital libraries. The design uses graphs and formal definitions to support services at typical and metamodel levels for powerful, general capabilities. From this, I develop a metadata model in which formal terms are defined within the 5S framework.

We previously described a framework that produced interoperable metadata description sets and harmonized model vocabularies, presented a software implementation of the framework, and demonstrated the performance of the software prototype of an existing digital library. This framework was presented in [95]. The publication contains a complete discussion of the metadata description set generation framework, description sets and records, description set generation software, harmonization and automated metadata extraction. The discussion presented here provides a treatment of building description sets, extracting metadata from content, building metadata records, and indexing the records.

### 6.14.1 Generating Metadata Description Sets

Metadata description sets, also called metadata schemas, provide a structure for representing content through metadata terms and values as defined by the Dublin Core community [153]. The task of developing standards compliant metadata description sets needs to be automated for scientific researchers, who typically lack expertise in information science. This may be provided through semi-automated generation software and automated extraction tools. The approach used here is similar to automated relational index suggestion tools found in the Microsoft Index Tuning Wizard, IBM’s DB2 advisor, and Oracle’s Tuning Pack. Previously developed metadata description set generation software [95] takes several representative examples of simulation input and output files; recommends terms for potential inclusion in a metadata schema; allows user selection of the important terms; adds Dublin Core terms, infrastructure terms, and formal terms; and generates a metadata schema. Fig. 6.5 displays a representation of the organization of simulation-based metadata terms. See [95] for a discussion of methods and software for generating description sets.

### 6.14.2 Extracting Metadata Values

Content generated by the execution of simulation models must be automatically processed due to the volume of digital objects produced. Fortunately, metadata extraction need not require human-intensive efforts as ontologies describing a domain and type of object may be used to build a metadata description set for each type of content. Pairing the description set with a metadata extraction script supports automated, run-time metadata record harvesting. Metadata extraction for a specific type of document requires an ontology, $ont$; metadata description set, $MD_s$; extraction
script, \textit{ext}_s; and a document harvested by a content broker, \textit{do}_i. The output of the extraction process is the inclusion of \textit{do}_i in the collection index, \textit{I}_{do_i}. The pre-conditions are an existing collection, \textit{C}_i \in \text{Coll}, and script \textit{ext}_s to extract content from \textit{do}_i to \textit{I}_{do_i}, as described in \textit{MD}_s. The post-condition is a modified collection index, \textit{I} \Rightarrow \textit{I}'', where \textit{I}_{do_i} \in \textit{I}''. Metadata extraction in SimDL is handled by extraction scripts paired with a model ontology and is provided by simulation model developers. It is assumed that all input files or output files from a specific version of a simulation model may be processed by the same extraction script.

Automated metadata extraction remains an open problem in information retrieval and management. Minimally, the goal of automated metadata extraction is to process files of a given type, determine metadata values for specific metadata terms, generate a metadata record with a given structure, and index the metadata record within a metadata collection for the given type. Even without the semi-automated development of quality metadata description sets through generation software, it is possible to implement automated metadata extraction for any static description set. The difficulty in metadata extraction lies in producing a well described process for extracting and continuing the eminence of the current description sets’ structure, extraction procedures, and record formatting.

An additional DL component is needed consisting of a grammar and software implementation to form an automated metadata extraction process. Users will need a language (grammar) for defining how to automatically extract (data mine) actual values from an individual digital object for all of the metadata terms described by a given metadata schema, and to generate a complete metadata

---

**Figure 6.5:** Metadata description set term organizational structure.
record for the digital object. A software implementation takes as input: a) a metadata schema for a specific file type, b) an automated extraction specification written in the extraction grammar, and c) a digital object of the file type; then it produces a metadata record for the digital object.

**Formal Definition:** Formal definitions of metadata extraction and curation allow for precise understanding, implementation, and use within the automated curation framework. The following definitions build on previously specified formal 5S definitions [44].

**Metadata extraction** is a tuple defined as $ME = (\{DO\}, M, G, Mdr)$ where

- $\{DO\}$ is a set of input files, $\{do_1, \ldots, do_n\}$;
- $M$ is a set of metadata terms, $\{m_1, \ldots, m_k\}$, requiring metadata extraction;
- $G$ is a set of grammar rules, $\{g_1, \ldots, g_r\}$, for metadata extraction into a specific record structure; and
- $Mdr$ is a set of metadata vectors, $\{v_1, \ldots, v_n\}$, structured in records, as returned from the metadata extraction tool.

$MDO = \{mdo_1, \ldots, mdo_n\}$ where,

- $mdo_i = \{K, T\}$;
- $K$ is a set of ‘$t$’ metadata types, $\{k_1, \ldots, k_t\}$, whose values need to be extracted from the file;
- $T$ is a set of file types. A file type is a combination of file extension and file structure, $\{e, s\}$;
- $e =$ file extension; and
- $s =$ file structure, e.g., row-based or column-based.

**Metadata Extraction Grammar**

A metadata extraction grammar is needed to explicitly describe the process of mining metadata values from a structured file. The grammar described here is based on the standard types of structured files used as inputs and outputs to simulation systems, e.g., TXT, CSV, RDF, and XML. These files are based on a single simulation software version and are structured in row or column orientation. In some cases, metadata values will appear next to term identifiers, while other value determination may require processing (e.g., column summation). Processing XML files requires a different set of operations, as the structure of an XML file may be leveraged to access values. The metadata extraction grammar builds upon the following formal vocabulary for terminal symbols. Note that grammar production rules merit a separate lengthy discussion. This grammar
was constructed based on a variety of structured file exemplars used at the Network Dynamics and Simulation Science Laboratory at Virginia Tech.

**Grammar Vocabulary**

- **delimiter**: The delimiter between tokens in a file
- **extract**: Extracts and returns the token at the current location in the file
- **goto_row(r)**: Move pointer to beginning of row \( r \) in the file
- **goto_column(c)**: Move pointer to beginning of column \( c \) in the file, with columns separated by delimiter
- **goto_next_token(t)**: Move pointer to next token \( t \) in file
- **get_value**: Extracts and returns the associated value for current token
- **sum_column(c)**: Extracts and sums up all the values in column \( c \) and returns the result
- **next_token_in_row**: Advances pointer to next token in row (if it exists)
- \( v_{k_i} \): Metadata value for term \( k_i \)

Note that this partial vocabulary is required to describe the provided example grammar rules and an extended version provides additional functionality, e.g., \( \text{average_column}(c, row_i, row_j) \).

**Example Grammar Rules:** With this grammar, a set of metadata extraction rules may be produced to describe the methodology for mining values from a specific type of simulation-related file as specified in a single metadata description set. The set of rules may be applied on the set of files \( F \) to generate metadata records for these digital objects. The following are two representative rules as encoded in the grammar. The first example processes a structured file, e.g., simulation input file, that contains pairs of input terms and input values co-located on the same line. The second example processes a column-oriented file, e.g., simulation output file, that contains data points to be totaled per column. Note that single value selection rules, summation rules, etc., may be combined on a term-specific basis, i.e., each \( k_i \) is described by a separate rule.

- **IF** \( e = \text{‘txt’} \) **and** \( t = \text{‘row-based’} \) (a uniform text file of row-based \{label, value\} pairs) \( \rightarrow \) delimiter = ‘;’; For each row \( x \),
  - goto_row(x)
  - if extract \( \in K \), next_token_in_row; \( v_{k_i} = \text{extract} \);
- **IF** \( e = \text{‘txt’} \) **and** \( t = \text{‘column-based’} \) (a uniform text file of column-based labels for summation across all rows) \( \rightarrow \) delimiter = ‘TAB’; For each column \( x \),
  - goto_column(x)
  - if extract \( \in K \), \( v_{k_i} = \text{sum_column}(x) \);
6.14.3 Forming Metadata Records

Metadata records are generated by leveraging the automated metadata extraction process to produce a set of descriptive metadata for a DO. The metadata is then added as predicates and objects to the DO’s subject identifier in the RDF metadata index. The completeness of records, and conformance to a description set depends entirely on the automated metadata extraction service’s ability to discover and extract metadata values from a DO. As such, the conformance of metadata records is directly related to the current metadata extraction’s mining protocols, i.e., what terms and term values to search for within a DO. As the metadata record building process is automated, the collection completeness is guaranteed for all DOs processed by Simfrastructure. SimDL uses the description set generation and harmonization software and is currently in use to manage epidemiology simulation models and related content. Metadata description sets for each of the three malaria and influenza models were generated, see Fig. 6.6 for an example. A metadata broker was developed to harvest input, output, and result term values as well as to create metadata records each time the simulation model software is executed. The model-specific metadata schemas and records are used in all index, browse, search, discovery, dissemination, verification, and preservation services of SimDL. The metadata records are produced in RDF form to allow queries of specific records. Metadata produced through the framework provides description set interoperability between research groups, simulation systems, multiple models, and stages of the simulation workflow.

6.14.4 Indexing Metadata Records

The RDF Triplestore is a database for storing and retrieving triples of metadata records in a graph. Each triple contains a subject, object, and predicate or relationship between the subject and object. RDF Triplestores are composed of a graph of individual RDF triples, i.e., (subject, predicate, object). Network graphs provide a means of logically managing metadata for a collection. Overlay networks provide a means of indexing connections between digital objects and related metadata values. Service-specific overlay networks can be developed in a digital library through motif-identification algorithms. Motifs in graphs are small networks, or subgraphs, with a defined structure. In SimDL, the pairing of individual input configurations with the related simulation results produced from the configuration forms a motif. Motif identification is useful in enumerating building blocks of subgraphs and the functional properties of each type of motif. Multiple approaches exist for motif identification including color coding, back-tracking, and dynamic programming. Parallel subgraph identification algorithms, e.g., PARSE [161], are required for large graphs such as metadata indexing for collections. Initially, the indexing and searching services were designed with Oracle RDF processing implementations. While distributed, scalable motif-identification methods are available in our case studies; the SimDL toolkit provides a non-parallel RDF search service for small, medium, and large sized collections. Our aim is to produce a graph-based mechanism for indexing all types of content. Graphs provide a generic method of organizing content. The power of formalized search can be demonstrated in two contrasting domains.

Metadata Structure and 5S Graph Representation: Content may be described by Dublin Core [153],
Figure 6.6: Metadata description set for input configuration files to the EpiSimdemics model. This XML representation is simply an alternative, concise translation from the RDF-triplestore used to store the metadata records. Each of these lines of XML is stored in a triple, e.g., (‘id:0’, ‘dc:creator’, ‘user:leidig’).
domain-specific, infrastructural, and 5S terms, and can be represented in a graph structure. The metadata schemas for SimDL systems include 5S and Dublin Core terms. Simple 5S terms define basic concepts. SimDL includes formally defined terms such as input file, analysis type, study designer, and simulation model parameters. In addition to simple terms, relationships and term hierarchies can be encoded with 5S. For example, the Dublin Core terms of audience, contributor, creator, publisher, and rightsHolder are formally described using 5S society and scenario definitions in SimDL. The metadata terms for SimDL may be indexed in a graph to support sub-graph identification for building metadata overlay reasoning.

Graph Construction: Every document in managed collections has a metadata record. Complete metadata records have a metadata value for each of the metadata schema terms and are used for search purposes. The metadata schemas these projects contain could be defined in RDF or XML Schema formats. Individual records are stored using RDF or in database tables.

RDF graph-based services require a labeled, undirected 5S graph \( G = (V, E) \), a set \( L \) of labels, and a labeling function for edges \( l : E \rightarrow L \). Individual documents and users are nodes \( v_i \in V \). Metadata terms are represented by edges \( e_i \in E \) with a label \( l_{e_i} \) and connect nodes with matching metadata values. Note that \( G \) is a sparse graph in SimDL as few documents have identical metadata values for any given term. However, as \( k \) metadata terms form \( k \) potential edges between a pair of nodes, \( |E| \leq \frac{kn(n-1)}{2} \).

6.14.5 Summary

Generating interoperable metadata terms and description sets between simulation models and institutions is a non-trivial process. Similarly, cross-community collaborations on building domain vocabularies and ontologies are not simple or straightforward endeavors. Determining semantic relationships between models is technically difficult and has historically been unsuccessful. Often, modeling groups within a domain compete for funding and disagree on modeling paradigms. As an example, there are disagreements whether high-level differential equations or disaggregated agent based models are more appropriate for given domains. There are also disagreements on the semantic meaning of commonly used terms in a domain vocabulary. Further, the technical use of term values vary widely between simulation model implementations. The large and diverse nature of sub-fields within a domain, such as computational epidemiology, generally means scientists cannot easily create meta-ontologies for a domain. This leads to fractured ontologies covering sub-fields of a domain with only partial overlaps. Further analyses of modeler produced description sets and ontologies are required before we attempt to build a single, cohesive vocabulary, ontology, or term namespace for the simulation models and domains in epidemiology.

We have employed several tools to mitigate these challenges in the short term. Model vocabularies are constructed within a domain as research groups produce description sets. Taxonomies and ontologies are formed in each description set through 5S definitions of term structures and relationships. Simulation efforts frequently involve multi-institutional, multi-national collaborations.
Model developers may provide multiple XML schemas of input, result, and analysis terms in different languages when producing description sets. These multi-lingual schemas have been shown to provide model-specific multi-lingual thesauri for model terms in our previous internationalization efforts [117]. Importantly, semantic relationships between model terms are typically easier to define between multiple models developed by the same simulation institution, as scientists are intimately familiar with each model. Thus, a two-tiered harmonization approach may be most appropriate to define semantic matches within an institution to consolidate terms for the more difficult process of inter-institutional harmonization.

A major limitation of the current description set generation framework is the reliance on domain-modelers to choose good terms from the suggested terms. Description sets may require modification at a later time if the description set lacks terms needed for future digital library services. Future extensions to the framework may involve user profiles and required services per user role to refine the set of suggested terms. Similarly, the inclusion of unnecessary terms recommended by the system and approved by the user will make poor use of metadata storage space. There is no guarantee a modeler will adhere to good term criteria, defined in [20], when generating description sets without previous training or expertise in term selection.

The framework consists of a semi-automatic production of metadata description sets and a process for automated term value extraction and record generation in simulation systems. The framework was used to define the metadata implementation used in SimDL. It also has potential for additional implementations in scientific data management efforts by other groups in non-epidemiology domains. We have developed a metadata description set generator and harmonization implementation of the framework. This software has enabled digital library services for the computational epidemiology modeling and simulation community that is currently hindered by a lack of ontologies and semantically linked metadata schemas. The contributions of this portion of the work are a framework and software implementation to efficiently produce description sets (containing Dublin Core, model, infrastructure, and 5S terms), harmonize terms between models, and automatically generate records for content. Future efforts are needed to incorporate the 5S Framework Term Wizard for adding 5S definitions into RDF description sets by modelers without expertise in formal definitions. Additionally needed are studies to evaluate the harmonization software after the generation framework has been used to produce a larger corpus of diverse description sets.
Chapter 7

Evaluation

Crucial to the adoption of the SimDL framework are positive results from case studies and an evaluation of the performance, coverage, and scalability of SimDL and its services. However, digital library evaluation is recognizably an extremely complex and difficult task [46, 133]. [30] suggests an approach to evaluate elements and criteria. Evaluation elements are described in multiple entities such as network components, technical infrastructure, information content, information services, and use-case support. Evaluation criteria define the measurement for each element, i.e., extensiveness, efficiency, effectiveness, service quality, impact, usefulness, and adoption. The efficiency measurement may be broken down into details of a service implementation’s metadata functions, database functions, input and output processing, and logical processing. Evaluations of metadata functions utilizing RDF triplestores generally consists of measurements with the Lehigh University Benchmark (LUBM), real data from UniProt, or domain-specific data. In previous quality of service metrics for application cluster efforts [116], database function measurements consisted of the time taken for a request to connect to the DB, time taken for the request to be served by the DB, time taken for the DB to initialize, time taken for the DB to shutdown, time taken for the DB backup to complete, per process CPU and memory utilization, number and duration of users connected to the DB, memory allocated at startup, DB checkpoints, and memory per transaction. The remaining efficiency measurements are considered in terms of the average space and computation requirements needed to execute the service. As there are no existing parallel service implementations, efficiency of SimDL is assessed by acceptable thresholds for service performance. Performance results for services must be reasonable and not too inefficient to prevent widespread adoption in case study infrastructures. Thus, the end-to-end processing for a service should take no longer than a limited number of seconds for services exposed to end-users. A formal evaluation may be utilized to show the descriptive power in formalisms and the complete description of the SimDL framework. They also may prove that a set of services fulfills a particular use case scenario and the full set of case study scenario requirements. It is expected that SimDL may save time and effort for end-users in tasks such as designing a study, conducting studies, identifying similar work, managing and repurposing previous work, and conducting standard analyses. However, end-user evaluation studies are not conducted due to the focus on evaluating system performance.
7.1 Methodology and Evaluation

As published in [92], the following three components may be used to guide the evaluation of the suitability of SimDL for scientific settings.

7.1.1 Formal Descriptions

Theoretical foundations for users, content, and services provide a means of explicitly defining digital libraries. With these definitions, services can be implemented and reused between DL efforts. Here, the 5S framework notation is used to define SimDL while making use of previous definitions. Content including schemas, model-ontologies, input configurations, sub-configurations, results, analyses, and experiments that have been formally defined for SimDL [87]. User roles were also defined for SimDL including tool builders, administrators, systems, study designers, analysts, annotators, and explorers. Formal definitions of simulation-specific services were proposed in parallel work [87]. Thus, full formal descriptions of users, content, and services have been produced.

7.1.2 Implementations

Formal definitions of services, in 5S notation, include an informal description, required inputs, produced outputs, service pre-conditions, and post-conditions. The process for developing a service includes task identification, definition, and implementation. After service definition, developers previously had leeway to implement the service however necessary, as long as the input format, output format, and conditions are met. With the inclusion of implementation details in 5S notation, the defining and implementing stages have merged to allow practical constraints to inform formal definitions of the approach taken. Further work is required to determine if the current definition notation is sufficient for guiding the complete implementation of services.

7.1.3 Interoperability

SimDL supports user, architecture, and functionality interoperability through simulation-specific services. SimDL provides an API to allow user interfaces to interact with content collections. Multiple generic interfaces were built using this API in epidemiological and network science case studies. Thus, users have consistent interface presentations between models and systems. Architectural interoperability is provided through the brokering system communication with external components and infrastructures. With architectural interoperability, SimDL may be integrated into multiple simulation infrastructures that can provide adaptors to the brokering system. Functional interoperability is attempted through formal definitions of the inputs, outputs, pre-conditions, and post-conditions of services. With formal definitions of service inputs and outputs, it is possible to
reason over compositions of interoperable services, interoperability validation, and interoperability service registries.

7.2 Framework Evaluation

The SimDL framework is evaluated in its multiple forms as a digital library management system, DL system, and instance. The minimal framework effectiveness is achieved when the full set of DLMS functionality requirements are covered in a fully expanded DL instance. As a DLMS, SimDL is evaluated based on completeness and effectiveness of its defined components. The service completeness is demonstrated by comparing the provided coverage found in the DLS toolkit to the anticipated functionality requirements. The desired functionality was gathered from use cases of the predefined user roles [87]. The effectiveness of the DLMS design is determined by whether its definitions led to a DLS that provides the functionality required. As a DLS, SimDL is evaluated in terms of the implementation performance and deployment effectiveness. To provide effective DL instances, the DLS must contain effective and efficient services. The Boolean effectiveness rating of a service is assigned based on whether the DLS provisions an appropriate service in the DL prototype that is able to fulfill a given set of functionality requirements. The minimally acceptable DLMS effectiveness is achieved if the full set of DLMS functionality requirements are covered in a fully expanded DL instance. The efficiency was determined by measuring the response time, computational cost, internal space requirements, and parallel scalability limits of each service within the DL prototype instances. The deployment effectiveness considers the usefulness of the DLMS and its DLS implementation for developers within simulation groups. Specifically, the lines of DLS software code that are used in each prototype are counted. As the software reuse was different in each prototype instance, the lines of code (LOC) measurement was normalized with number and complexity of the DLMS functionality required by each prototype. In addition, the lines of ‘wrapper’ code needed to make use of the DLS within each infrastructure was counted. This evaluation determined if SimDL as a DLMS and DLS is reasonably effective and generalizable across deployment instances.

7.2.1 DLMS Completeness

DLMS completeness is evaluated by comparing the DLMS definitions to the functionality required in DL instances from use case modeling. Simply put, does the set of DLMS descriptions fully specify a class of DLs that would lead to DLSs and DLs desirable for a community? See Table 7.1 for further details on this evaluation based on expected user roles presented in [87]. Refer to earlier descriptions on formalisms for the full specification of each DLMS definition. The minimal set of SimDL DLMS definitions was guided by and fully specifies the anticipated user requirements for the modeled user roles. While additional services may be provided to support extended functionality, the described DLMS covers the complete set of minimal functional requirements, as shown in 7.1.
Table 7.1: DLMS definition completeness.

<table>
<thead>
<tr>
<th>Use cases</th>
<th>Relevant DLMS definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building tools</td>
<td>Communicating, getting DL identifiers, generating schemas</td>
</tr>
<tr>
<td>Administration</td>
<td>Curating, ranking</td>
</tr>
<tr>
<td>Supporting systems</td>
<td>Communicating, memoizing</td>
</tr>
<tr>
<td>Designing studies</td>
<td>Registering content, registering provenance</td>
</tr>
<tr>
<td>Analyzing</td>
<td>Supporting metadata,</td>
</tr>
<tr>
<td>Annotating</td>
<td>Registering metadata</td>
</tr>
<tr>
<td>Exploring</td>
<td>Searching, filtering, discovering, getting metadata, resolving provenance</td>
</tr>
</tbody>
</table>

### 7.2.2 DLS Software Toolkit Effectiveness

The toolkit must provide support for the set of functionality and content defined in the DLMS. While case study DL instances often require a subset of the full set of available functionality, the toolkit must implement the functionality required by the set of instances it is tasked with supporting. Extended formal definitions in a DLMS have neither a software DLS implementation nor case study instance that requires the functionality described in the definition. These definitions are considered ongoing extensions or provisional efforts and not considered in the DLS effectiveness. The suitability of DLS service implementations as measured in the DLS toolkit efficiency evaluation. Services provided in a toolkit are assumed to be of sufficient quality to be considered an effective implementation of the DLMS requirements. Table 7.2 details the effectiveness of the SimDL software toolkit to the formal definitions as described in Fig. 5.5 and Fig. 5.6. Many tasks are high-level abstractions of a single DL service, e.g., contributing, indexing, storing, and provenance registering are separate tasks supported by the DLRegister service. Note that the DLGetMetadata service returns data manager identifiers, allowing components to access and retrieve raw digital objects directly. In every case, the provided software toolkit provides services to implement an identified minimal functionality.

### 7.2.3 DLS Software Toolkit Efficiency (Service Performance)

The efficiency of the DLBroker is given in Table 7.3. Each measurement is the mean of 1,000 performance tests. The blackboard connection time is only incurred once at DL startup. The remaining costs are incurred with each DL service request.

The efficiency of each DLService is provided in Table 7.2.3 as a reference point. The results are given on small (i.e., 100 item collection), medium (i.e., 1,000 item collection), and large (i.e., 100,000 item collection) metadata indexes to show the scalability of the current implementations. Each value is the average time spent in a portion of code over 1,000 executions of the service. In alignment with Dublin Core, RDF triplestore metadata indexes were generated to contain fifteen
Table 7.2: DLMS definitions and DLS toolkit comparison.

<table>
<thead>
<tr>
<th>Formally and Informally Described Tasks</th>
<th>DLS Toolkit Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browsing</td>
<td>DLSearch, DLGetMetadata</td>
</tr>
<tr>
<td>Community growing</td>
<td>DLIncentivize</td>
</tr>
<tr>
<td>Comparing</td>
<td>DLMemoize, DLGetMetadata</td>
</tr>
<tr>
<td>Consuming and retrieving</td>
<td>DLGetMetadata</td>
</tr>
<tr>
<td>Curating</td>
<td>DLCurate, DLDORanker</td>
</tr>
<tr>
<td>Discovering</td>
<td>DLSearch, DLFilter</td>
</tr>
<tr>
<td>Disseminating</td>
<td>DLSearch, DLFilter</td>
</tr>
<tr>
<td>Incentivizing</td>
<td>DLIncentivize</td>
</tr>
<tr>
<td>Indexing</td>
<td>DLRegister</td>
</tr>
<tr>
<td>Logging</td>
<td>DLRegister, DLProvRegister</td>
</tr>
<tr>
<td>Metadata extracting</td>
<td>External component-specific codes</td>
</tr>
<tr>
<td>Metadata record generating</td>
<td>DLRegister</td>
</tr>
<tr>
<td>Metadata schema generating</td>
<td>Off-line software and UI</td>
</tr>
<tr>
<td>Reusing</td>
<td>DLMemoization, DLGetMetadata</td>
</tr>
<tr>
<td>Producing and contributing</td>
<td>DLRegister</td>
</tr>
<tr>
<td>Provenance tracking</td>
<td>DLProvRegister, DLProvResolve</td>
</tr>
<tr>
<td>Searching</td>
<td>DLSearch</td>
</tr>
<tr>
<td>Storing</td>
<td>DLRegister, DLProvRegister</td>
</tr>
<tr>
<td>Usage tracking</td>
<td>DLIncentivize, DLGetMetadata</td>
</tr>
<tr>
<td>Workflowing</td>
<td>DLProvRegister, DLProvResolve</td>
</tr>
</tbody>
</table>

Table 7.3: Evaluation of the DLBroker in milliseconds.

<table>
<thead>
<tr>
<th>Blackboard connection time</th>
<th>Processing and service selection overhead</th>
<th>Blackboard posting time</th>
<th>Blackboard retrieving time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,500</td>
<td>0.1</td>
<td>33.2</td>
<td>68.5</td>
</tr>
</tbody>
</table>
elements. For the discovery services, all queries were composed of two terms, in alignment with
the current average term-length of popular search engines, e.g., Google. Here, $n$ represents the
number of DOs in the collection and the metadata index, $r$ denotes the number of rules or reports
(composed of multiple clauses), and $c$ gives the number of clauses (DB queries) in a rule. I have
optimized these algorithms to scale with regards to $n$. However, in environments with smaller,
static collections and complex reports or queries, it may be possible for alternative algorithms to be
optimized in terms of $r$ and $c$.

7.2.4 Service Generality

We expect a reduction in software development efforts when integrating and utilizing the SimDL
toolkit in comparison to customized, institute specific software. This effort can be measured in
reusable lines of code (LOC), LOC needed to parameterize a service for a specific infrastructure,
development time, and ease of use. The minimal software implementation is roughly 2,000 lines of
code in Java sans DL properties initialization files, instance-specific parameterization of reports,
statistics and metrics, DB connections, curation policies, and DB setup. User studies are not a
part of this evaluation. See Fig. 7.5 for the LOC effort and reductions related to the integration of
the SimDL toolkit instead of reimplementation. These estimates are for SimDL development and
integration LOC only. Roughly ten high-level statements are needed to invoke each service from an
external component. The parameterization is an estimate based on the minimal number of reports
and metrics for standard DO files described in the preceding chapter, see Fig. 6.13. Depending on
the exhaustiveness of the desired incentivizing and curating functions, the actual requirements will
be several times this amount.
Table 7.4: Evaluation of the DL services with time resolution in milliseconds (with small, medium, and large indexes).

<table>
<thead>
<tr>
<th>Service (index size)</th>
<th>Overall cost</th>
<th>Input processing</th>
<th>Database or metadata cost</th>
<th>Processing cost</th>
<th>Internal memory bound - $\mathcal{O}{f(x)}$</th>
<th>Complexity - $\mathcal{O}{f(x)}$ bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLCurate (s)</td>
<td>220.6</td>
<td>0</td>
<td>28.2</td>
<td>192.3</td>
<td>rcn</td>
<td>rcn</td>
</tr>
<tr>
<td>DLCurate (m)</td>
<td>257</td>
<td>0</td>
<td>30.3</td>
<td>226.3</td>
<td>rcn</td>
<td>rcn</td>
</tr>
<tr>
<td>DLCurate (l)</td>
<td>460.5</td>
<td>0</td>
<td>236</td>
<td>224.5</td>
<td>rcn</td>
<td>rcn</td>
</tr>
<tr>
<td>DLCurateDORanker (s)</td>
<td>223.6</td>
<td>0</td>
<td>26.4</td>
<td>197</td>
<td>n</td>
<td>cn</td>
</tr>
<tr>
<td>DLCurateDORanker (m)</td>
<td>236</td>
<td>0</td>
<td>29.3</td>
<td>206.4</td>
<td>n</td>
<td>cn</td>
</tr>
<tr>
<td>DLCurateDORanker (l)</td>
<td>484.7</td>
<td>0</td>
<td>241.4</td>
<td>243.2</td>
<td>n</td>
<td>n+c</td>
</tr>
<tr>
<td>DLFilter (s)</td>
<td>229.1</td>
<td>0</td>
<td>7.8</td>
<td>1</td>
<td>n</td>
<td>n+c</td>
</tr>
<tr>
<td>DLFilter (m)</td>
<td>206.6</td>
<td>0</td>
<td>11.9</td>
<td>0</td>
<td>n</td>
<td>n+c</td>
</tr>
<tr>
<td>DLFilter (l)</td>
<td>331.8</td>
<td>0</td>
<td>133.3</td>
<td>1</td>
<td>n</td>
<td>n+c</td>
</tr>
<tr>
<td>DLGetDLID</td>
<td>224.6</td>
<td>0</td>
<td>13.2</td>
<td>8.7</td>
<td>l</td>
<td>l</td>
</tr>
<tr>
<td>DLGetMetadata (s)</td>
<td>210.3</td>
<td>0</td>
<td>7.1</td>
<td>.8</td>
<td>l</td>
<td>n</td>
</tr>
<tr>
<td>DLGetMetadata (m)</td>
<td>209.7</td>
<td>0</td>
<td>6.9</td>
<td>1</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>DLGetMetadata (l)</td>
<td>254</td>
<td>0</td>
<td>7.2</td>
<td>38.3</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>DLIncentivize (s)</td>
<td>135</td>
<td>73</td>
<td>55</td>
<td>0</td>
<td>n</td>
<td>rm</td>
</tr>
<tr>
<td>DLIncentivize (m)</td>
<td>135.9</td>
<td>73.2</td>
<td>55.5</td>
<td>2</td>
<td>n</td>
<td>rm</td>
</tr>
<tr>
<td>DLIncentivize (l)</td>
<td>474</td>
<td>75.5</td>
<td>389.4</td>
<td>3</td>
<td>n</td>
<td>n+c</td>
</tr>
<tr>
<td>DLMemoization (s)</td>
<td>226.9</td>
<td>83.5</td>
<td>10.9</td>
<td>123.3</td>
<td>n</td>
<td>n+c</td>
</tr>
<tr>
<td>DLMemoization (m)</td>
<td>260.4</td>
<td>64.5</td>
<td>11.4</td>
<td>178.1</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>DLMemoization (l)</td>
<td>450.4</td>
<td>65.9</td>
<td>115</td>
<td>251.1</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>DLProvRegister</td>
<td>226.27</td>
<td>0</td>
<td>1.55</td>
<td>1</td>
<td>l</td>
<td>l</td>
</tr>
<tr>
<td>DLProvResolve (s)</td>
<td>208.9</td>
<td>0</td>
<td>7.4</td>
<td>0</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>DLProvResolve (m)</td>
<td>204.3</td>
<td>0</td>
<td>7.3</td>
<td>0</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>DLProvResolve (l)</td>
<td>246.5</td>
<td>0</td>
<td>45.4</td>
<td>0.1</td>
<td>l</td>
<td>l</td>
</tr>
<tr>
<td>DLRegister</td>
<td>263.5</td>
<td>0</td>
<td>14</td>
<td>9</td>
<td>l</td>
<td>l</td>
</tr>
<tr>
<td>DLSearch (s)</td>
<td>104.4</td>
<td>0.1</td>
<td>21.5</td>
<td>0</td>
<td>n</td>
<td>cn</td>
</tr>
<tr>
<td>DLSearch (m)</td>
<td>129.4</td>
<td>0</td>
<td>19.3</td>
<td>0</td>
<td>n</td>
<td>cn</td>
</tr>
<tr>
<td>DLSearch (l)</td>
<td>374</td>
<td>.1</td>
<td>25.8</td>
<td>116.8</td>
<td>n</td>
<td>n</td>
</tr>
</tbody>
</table>
Table 7.5: Service deployment efforts for the CINET case study.

<table>
<thead>
<tr>
<th>Service</th>
<th>LOC to Parameterize</th>
<th>LOC Reused for Developers</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLCurate (s)</td>
<td>250</td>
<td>559</td>
</tr>
<tr>
<td>DLCurateDORanker</td>
<td>20</td>
<td>410</td>
</tr>
<tr>
<td>DLFiltre</td>
<td>0</td>
<td>156</td>
</tr>
<tr>
<td>DLGetDLID</td>
<td>0</td>
<td>117</td>
</tr>
<tr>
<td>DLGetMetadata</td>
<td>0</td>
<td>135</td>
</tr>
<tr>
<td>DLIncentivize</td>
<td>88</td>
<td>221</td>
</tr>
<tr>
<td>DLMemoization</td>
<td>0</td>
<td>87</td>
</tr>
<tr>
<td>DLProvRegister</td>
<td>0</td>
<td>124</td>
</tr>
<tr>
<td>DLProvResolve</td>
<td>0</td>
<td>119</td>
</tr>
<tr>
<td>DLRegister</td>
<td>0</td>
<td>154</td>
</tr>
<tr>
<td>DLSearch</td>
<td>0</td>
<td>170</td>
</tr>
</tbody>
</table>
Chapter 8

Conclusions and Future Work

8.1 Summary

Our efforts in the SimDL framework aimed to introduce a new generation of digital libraries that directly connect researcher efforts with simulation environments. The SimDL model allows researchers to take advantage of large existing bases of experimentation by archiving previous work for shared use among collaborators. SimDL may play a key role in the coordinated data management efforts of large cyberinfrastructure environments, particularly those based on computational epidemiology and network science. Future uses of SimDL may expand into general simulation environments.

In this work, we have

- covered the digital library requirements of the computational epidemiology and network science communities;
- generated the SimDL framework for developing and implementing a class digital libraries;
- formalized SimDL as a digital library management system, in the 5S framework, and showed how to translate definitions into alternatives (i.e., DELOS Reference Model);
- produced a digital library system software toolkit from the DLMS;
- deployed multiple digital library instances from the DLS;
- proposed and studied important considerations and an evaluation for digital library case study deployment;
- demonstrated that this process may be utilized to generate software toolkits with reasonable services; and
- shown that this process is capable of supporting simulation-related infrastructures efficiently.
8.2 Hypothesis Review

Looking back at what was accomplished, we have proven both of the hypotheses. From chapters 3 and 4, we deduce that it is possible to describe the current and anticipated needs of the simulation-based research community and formally define relevant content, users, tasks, digital library services, and digital library software. This was demonstrated through utilization of the 5S framework. We have shown that the DLMS and implemented DLS toolkit led to efficient (semi) automated services in support of the defined requirements. Additionally, we described what constitutes scientific digital library artifacts, previously unavailable SimDL services, and evaluations of SimDL performance.

8.3 Future Work

Building on this effort, we can generalize the SimDL framework to a broader science-supporting digital library with appropriately defined minimal functionality. Additional performance, case study, and user study evaluations may answer the following questions.

- How effectively can epidemiology simulation research groups not involved in developing the prototypes make use of SimDL?
- How does SimDL and its services improve efficiency in comparison to existing systems?
- What are the performance gains when comparing designing and running experiments with and without SimDL services?
- How well can educators carry out educational activities with managed content, digital libraries, and simulation infrastructures?
- Are there alternative, standard methodologies to running experiments in simulation groups (other than study designs, results, analyses, data mining, and publication)?
- How well does SimDL support this from end-to-end, based on the effort required to create a study and execute an experiment?
- What additional functions may be provided through SimDL from emerging workflow, digital library, and management systems?
- How suitable is SimDL outside of epidemiological and network science domains?

Formal definitions for content, services, and users have been completed along with many service implementations. Deployments of SimDL have been implemented for managing individual simulation models and cyberinfrastructure. Implementing functionality for extended (non-minimal) services will remain an ongoing development task, and integration requires tight coupling through assumed brokers. Services are currently being organized into a registry for discovery and reuse, albeit in a DL community without a history of registry utilization.
8.3.1 Functional Extensions

Functional extensions and research efforts to improve SimDL are in continual development. While the intention is to provide SimDL as an extensible platform for deploying digital libraries to manage scientific content, several components are required to promote adoption within the epidemiology, simulation, digital library, and larger scientific communities. These components may be added to the software toolkit using the framework to produce an extended version of the minimal SimDL by adding general implementations of further services specific to the epidemiology and network science domain. A next-generation set of components is needed to annotate and recommend digital objects; ask high level semantic search queries; query across SimDL instances; federate SimDL instances with other digital libraries through semantic mapping; provide interoperability for accessing metadata and collections with other digital libraries; and provide a method for communication within the DL system (e.g., message boards). Visualization components may be added to the paired UI to ease input configuration navigation; dragging and dropping sub-configurations; fisheye overviews of a schema; and treemap views of datasets, analyses, documents, and annotations collections. Maintaining user sessions across SimDL instances may provide more coherent use of user credentials, demographics, access rights, preferences, tailored views, expertise, classes of rights (access to content), and roles (access to system functionalities). However, SimDL is not initially envisioned as a platform for semantically combining all worldwide simulation systems. The foremost expectations for a SimDL instance are to holistically provide DL services for interested simulation groups within a domain under multiple contexts.

This framework may be extended with high-level language processing, translating between questions and study designs, automated model selection and parameterization, cloud-based searching implementations, and DL service computation offloading to HPC resources. Additional curation activities may be added by simply parameterizing the curation service. Obvious extensions include translating to new file and metadata formats as a part of migration, archiving older software codes, and emulating operating systems. Scientific content migration may be automated and added to the curation policy triggers and processes. In addition, content regeneration and study reproduction is possible through the existing provenance links and curation services. The provenance tracking service is able to link successive DOs in an arbitrary workflow. This concept may be further fleshed out with a formalization of tracking provenance in dynamic workflows. For the scientific community, a large number of science-supporting services could be added to SimDL. As examples, linear extrapolation of simulation results could be performed for some models based on existing data, and multivariate query-by-sketches could apply data mining to existing collections as demonstrated in [13]. Additional work is needed in defining domain-specific appropriate methods for comparing complex experiment sets in ranked searches. Note that the current search functions allow for the application of complex, weighted queries and may be utilized to conduct experiment comparisons. Novice experimenters lack detailed knowledge of simulation models, prior studies, and how to properly design studies. A service to automate model selecting and parameterization based on high-level statements and queries would greatly benefit this class of users. Additional evaluations on similarity and distance measures would provide insights into alternative discovery functions,
i.e., Cosine, Dice, Euclidean, Forbes, Russel-Rao, Yule, Simpson, Manhattan, Tvensky, Kendall’s (K) tau, and similarity analysis. While the SimDL framework and software toolkit have been used successfully in several case studies, we wish to see further adoption and extensions by other computational epidemiology, simulation-research, and DL research groups.

8.3.2 Cloud, Grid, and Cluster Supporting DL Services

Distributed storage and retrieval implementations have yet to fully utilize available cloud, grid, and cluster infrastructures. This framework is especially suited to such research areas due to the direct availability of HPC platforms to SimDL instances. Future work in this framework would be disappointing without expansion into and advance in scalable and distributed DL computing. The storage, indexing, and discovery of content is a primary concern in digital libraries. As the number of simulation experiments to be stored increases, the storage size will necessarily increase. As additional computational power is added and simulation software is extended, it is likely that the inputs and results of simulation will increase as experiments become more complex. Over time, the input space of applications will be explored as directed by the usage of researchers. With novel functionality and attractive services, widespread adoption and heavy usage of SimDL at a particular research group would generate massive amounts of experimental data. Deterministic discrete-event simulation software employed by SimDL produces a predictably sized output depending on the input files. Should a non-terminating simulation software attempt to store results in SimDL, extra consideration would be required to ensure that the time steps required for the simulation to reach a steady-state are acceptable, see [64]. When storing simulation generated output in particular, the method of automated data collection is a prime concern. There is not a clear standard for the delivery method of simulation results. The interface to the simulation software (e.g., Simdemics) is a key element in storing the simulated data. It is necessary for the information provided to and produced by the simulation software to be matched directly with the digital library storage scheme. In addition, a given scenario may be replicated numerous times with different random seeds or input. Awareness of the considerable storage requirements for a set of replications may be required when launched by researchers. This may lead to compressed data formats for new content and links to similar input fragments across experiments. The framework described in this work may be utilized in its own right and provides a stable platform for investigating these possibilities.
Bibliography


[21] Leonardo Candela, Fuat Akal, Henri Avancini, Donatella Castelli, Luigi Fusco, Veronica Guidetti, Christoph Langguth, Andrea Manzi, Pasquale Pagano, Heiko Schuldt, Manuele Simi, Michael Springmann, and Laura Voicu. DILIGENT: integrating digital library and


