# An Informatics Framework for Testing Data Integrity and Correctness of Federated Biomedical Databases

Mijung Kim,[1] Tahsin Kurc,[2] Alessandro Orso,[1] Jake Cobb,[1]
David Gutman[2], Mary Jean Harrold,[1] Andrew Post,[2] Ashish Sharma,[2] Joel Saltz[2]
[1]College of Computing, Georgia Institute of Technology, Atlanta, GA
[2]Center for Comprehensive Informatics, Emory University, Atlanta, GA

**Abstract**

*Clinical research is increasingly relying on information gathered and managed in different database systems and institutions. Distributed data collection and management processes in such settings can be extremely complex and lead to a range of issues involving the integrity and accuracy of the distributed data. To address this challenge, we propose a middleware framework for assessing the data integrity and correctness in federated environments. The framework has two main elements: (1) a test model describing the dependencies between and constraints on data sources and datasets, and (2) a family of testing techniques that create and execute test cases based on the model.*

## 1. Introduction

A growing number of studies gather and manage information from multiple data types and sources. Logistics, regulations, and data-analysis requirements may not always allow for centralized data gathering and management. Data may be collected from patients recruited at multiple institutions. Even within an institution, data may be collected and processed by different laboratories because of instrumentation and analysis requirements. Federated systems have been developed and employed [1,2,3,4] for these types of studies to support distributed data access and analysis requirements. An important component that is missing in most existing systems is a middleware framework for testing the data integrity and correct operation of a federated environment.

Data sources in a federated environment change over time— data management systems are modified, data models and ontologies are changed, new datasets are gathered, and existing datasets are updated. Federated databases are often managed by different groups; a group may modify their database without informing other groups, causing inconsistencies and breaking dependencies within the federated environment. Errors may arise from both human mistakes and faults in the software. For example, updates to data may introduce hard-to-detect errors. Indeed, such errors occurred in one of our studies, when a subset of the clinical database we accessed remotely was updated erroneously, replacing old diagnosis values with new values that did not match the known progression of the disease. Detecting and tracking this and other types of errors (Section 2 describes additional examples of errors) manually in a federated environment is impractical, and their presence can seriously compromise the results of a clinical research project.

Some databases and ETL (Extract, Translate, Load) processes implement data quality and error checks. In practice, however, most implementations are done as one-off solutions via low-level scripts and programs, which can be difficult to extend or modify for new datasets and additional tests. Moreover, these implementations are targeted at a single instance of a resource and are not designed to test a federated environment. Our goal is to address these challenges by developing and evaluating a framework that can support systematic testing of the data integrity and correct operation of federated environments.

Our framework has two main elements: a *test model*, representing constraints on and dependencies between datasets and data sources in the federated environment, and *a family of testing techniques* that leverage the model to test data integrity and accuracy of the environment. The test model is a set of rules derived from (1) data models of individual data sources and constraints expressed in the data models, (2) relationships among different data models and data sources, (3) business processes (e.g., study protocols), (4) user-defined rules and constraints, and (5) rules and constraints based on domain knowledge. The testing techniques are driven by the test model and assess the federated environment by (1) identifying relevant test scenarios for the environment, (2) creating test cases that realize such scenarios, (3) generating (when needed) suitably tagged synthetic data to enable the scenarios, and (4) executing the generated test cases.

## 2. Motivation and Objective

Our effort is motivated mainly by translational research projects supported by the Atlanta Clinical and Translational Science Institute (ACTSI), a multi-institutional partnership funded by the NIH Clinical and Translational Science Awards program. A common theme among a wide range of studies undertaken by ACTSI investigators is that biomedical data are captured at multiple locations and stored in different systems (e.g., multiple i2b2 [5] instances) hosted by

partnering institutions. It is worth noting that many of the research scenarios supported within ACTSI are common use cases in other clinical research efforts.

To illustrate the issues we target, we consider a specific example: the study of brain tumors conducted in the In Silico Brain Tumor Research Center (ISBTRC)[1] through *in silico experiments* on data collected from a group of patients. Datasets for this study encompass high-throughput omics data, radiology and pathology image data, clinical data, such as diagnosis and survival, and tissue data. The datasets are obtained from public databases (e.g., Rembrandt[2] and The Cancer Genome Atlas[3]), derived from primary datasets (e.g., image analysis results), and collected from subjects at the collaborating institutions.

In the course of the study, the primary datasets from public resources are downloaded to local clinical, imaging, and genomic databases for further analysis and cyclically updated to include new data. These datasets are expected to have gene expression, microarray, mRNA, and miRNA data, radiology and high-power light microscopy image data, and clinical diagnosis and survival outcome for each patient. For subjects at the collaborating institutions, tissue samples are collected following the study protocol. Microscopy image data are obtained from the tissue samples; two modalities of microscopy image data are captured at Emory (brightfield microscopy images by a pathology imaging group for every subject and quantum dot immunohistochemistry images by a nanotechnology center for some subjects). Each subject's de-identified clinical information is maintained in a clinical data-management system. Gene expression datasets are stored in a molecular database, whereas imaging data are managed in radiology and microscopy imaging databases. The primary datasets are analyzed using a suite of analysis algorithms and by human experts (in the case of image data). The omics analysis results are stored in a molecular database, whereas the image analysis results are stored in image markup and annotation databases.

The data gathering and management processes in this study are complex and error prone, as they involve multiple points of data acquisition/generation and multiple data management systems. There are multiple sources of error. For example, in one case, an image was accidentally deleted, leaving analysis results obtained from that image unlinked to any image data. Errors may also arise due to incorrect identifiers and foreign-key relationships across different data generation points and systems. Different institutions and systems maintain local identifiers (e.g., de-identified specimen, patient, image identifiers), which need to be mapped to appropriate global identifiers and stored so that related data products can be linked and queried correctly. This is an error prone process. If a system in the environment does not support such identifiers correctly, or mapping scripts are implemented incorrectly, join queries involving that system will return incorrect results.

As we mentioned in the Introduction, updates to data may introduce hard-to-detect errors. An erroneous update to some of the clinical data in our study, for instance, resulted in incorrect diagnosis values that conflicted with the expected progression of the disease. This is an example of the kind of domain knowledge that a testing framework should incorporate and check.

As a final example, a time consuming inspection of the datasets from the public repositories revealed that some of the required data for some subjects were missing (e.g., image data). The TCGA study included a set of manual annotations generated by a set of neuropathologists; this process depended on the availability of digitized glass slides. In an early in silico experiment, we sought to replicate the results generated by the neuropathologists using a set of custom-developed computer algorithms. We eventually discovered that not all images for slides used in the manual annotations were available for download on the TCGA website, which led to an extremely labor intensive process to track down additional slides. Our framework would be able to capture such a discrepancy early in the process, by checking for the presence of data based on the study protocol requirements, and notify the users of the issue.

The goal of our testing framework is to be able to detect and report these types of errors using a suite of techniques that combine domain knowledge, modeling, and software testing methods.

## 3. Approach

Figure 1 provides a depiction of our framework, which consists of a set of *testing techniques* and a *test model* that are tightly integrated. In addition, it contains a *data-mining* element, which can provide support for extracting rules from the federated environment. In this framework, test cases are run both offline, before deployment, and online, to monitor whether changes to the environment violate any of the dependencies or constraints encoded in the identified rules.

### 3.1. Test Model

The test model is used to provide a specification of the expected correct state and function of a federated environment. It describes constraints, dependencies, and relationships imposed on data and resources. Our approach draws from the principles and practices

[1] http://cci.emory.edu/cms/projects/ISBTRC.html
[2] http://caintegrator-info.nci.nih.gov/rembrandt/
[3] http://cancergenome.nih.gov/

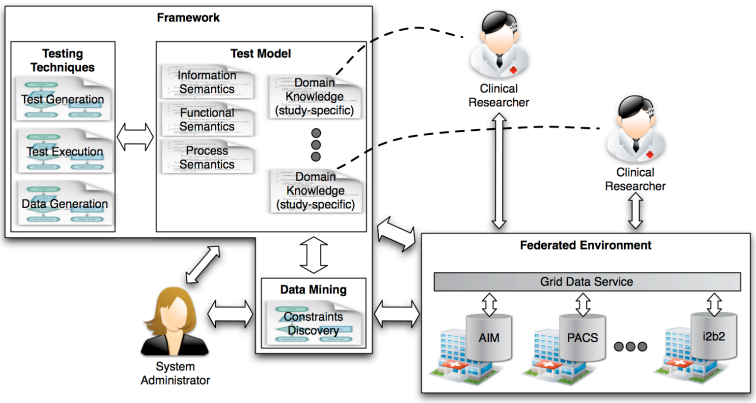


**Figure 1:** Intuitive view of the proposed framework.

implemented in frameworks for data and systems interoperability [6]. The key components of these frameworks are system specifications (based on static, functional, and behavioral semantics), conformance statements, and conformance assertions. Conformance statements are derived from specifications, and conformance assertions are evaluated against the conformance statements to assess the level of interoperability between systems. The test model in our case becomes a key to system specifications and statements expressed as a set of properties and rules in a semantic language (OWL and SWRL [7,8]). Testing techniques evaluate conformance assertions against the specifications and statements. We employ the notions of information, functional, and process semantics to create the test model for a given federated environment.

### 3.1.1. Information Semantics

Information semantics represent the set of properties and rules that can be derived from data models in individual databases and constraints associated with the data models. The first aspect of information semantics is the use of value domains and value sets to express permissible and non-permissible values for data elements. This aspect can be used to create rules to express data value constraints. For example, if a data attribute maintains values on a patient's height, it may have been assigned a value set of [1,8] feet. In that case, the test model could include the following corresponding set of rules: *AttrX.domain=numerical; AttrX.unit=foot; AttrX.range =[1,8]*.

Another relevant aspect of information semantics is the description of relationships between data models within and across databases. In our framework, this aspect results in rules that describe how data elements in different databases are related, whether they are used to store semantically equivalent data, and so on. For example, a rule stating that "Attribute X in Database A is the same as Attribute Y in Database B," encoded as *haveSameValue(DB_A,AttrX,DB_B,AttrY)*, could be created to express that the two attributes should have the same value.

A third aspect of information semantics is the incorporation of domain knowledge. As our understanding of biological systems and diseases progresses, certain relationships, hierarchies, and axioms are deemed as *domain facts* and incorporated into ontologies representing domain knowledge. The test model makes use of domain ontologies to express rules that describe constraints and relationships derived from the domain knowledge. For instance, a domain ontology may define that "Stage II always follows Stage I, and Stage I never follows Stage II for disease Z,". The test model then contains the rule $\forall t1>t2 \Rightarrow DiseaseZ.Stage[t1]<DiseaseZ.Stage[t2]$, where t1 and t2 indicate timestamps.

### 3.1.2. Functional Semantics

Functional semantics describe the expected behavior of a system (e.g., error conditions, returns on successful invocation of the system) when it is interacted with. In our case, functional semantics are used to define rules on the behavior of the system for (1) data loads, updates, and deletes, (2) successful query executions, and (3) error conditions. For instance, if new object identifiers should be generated, when new data objects are loaded to a system, a rule expressing which identifiers should or should not be generated can be defined as part of the test model.

### 3.1.3. Process Semantics

Process semantics describe business processes and relationships between databases and systems that are not part of information and functional semantics. For example, in the study described in Section 2, the study protocol corresponds to the business processes. The protocol describes which data should be obtained for each patient and how many instances of a given data type should be gathered. The process semantics of acquiring microscopy imaging data can lead to a rule stating that "an image data item for the in silico brain tumor study must be of type *InSilicoImageData*, have two bright-field H&E whole-slide images, and have two quantum-dot images." (For clarity, we show these rules in natural language.) Another rule may combine the image data with other data types for the brain tumor study: "A dataset for the in-silico brain tumor study must contain data items of type *InSilicoImageData*, patient survival data, sequence data, and mRNA data."

This set of rules can be used to evaluate if the necessary data elements have been collected for a subject in the brain tumor study. In our framework, process semantics is also employed to derive user-defined rules on dependencies and relationships among two or more databases. For instance, a rule can be defined to specify that: "if there are image-annotation results in the image-annotations database, there must be a corresponding image in the image database."

We have identified these three components for test model creation based on our experience with several use cases. Our main goal with the test model is to provide a high-level mechanism for users and developers to represent constraints, dependencies, and relationships. Nevertheless, we are building the testing system such that application-specific test scripts can be executed for cases the test model is not sufficient.

We plan to examine automated mechanisms and data mining to create rules from data models, datasets, and domain knowledge. Some rules in the test model can automatically be derived from the data models. For example, schema constraints on associations could be used to describe rules on such associations. In data models where data elements are semantically annotated, and associated with value sets for permissible or non-permissible values, rules could be created on such value sets to describe constraints on the data elements. Data mining may also be applied to identify strong correlations among the data that can indicate the presence of latent rules. Such inferred rules may be used to identify anomalies in the data -- data mining was successfully used to this end in previous work (e.g., [11,12]).

### 3.2. Testing Techniques

Our main goal is to assess the integrity and accuracy of the environment's data sources and datasets with respect to the rules in the test model. In this section, we describe how our approach accomplishes this goal and illustrate its application on some of the examples from Section 3.1.

#### 3.2.1. Test Generation and Execution

Tests are typically defined according to a given set of requirements (e.g., coverage of some code elements). In our case, the key aspects to be tested are the data elements and the rules defined over them. Thus, the starting point for our approach is the test model. More precisely, our approach analyzes the test model to identify relevant data elements, discover the rules involving such elements, and generate testing requirements (and ultimately tests) based on such rules. The specific kind of tests generated depends on the type of rule being analyzed. Consider, for instance, the rules presented in Section 3.1.1. The analysis of rule *haveSameValue(DB_A,AttrX,DB_B,AttrY)* would result in offline tests that check that "for corresponding elements a and b in Databases A and B, respectively, a.X and b.Y must have the same value." (The specific number of tests generated would depend on the amount of resources available for testing.) The framework can instantiate these tests and run them on the databases automatically, so as to identify and report violation of this rule. Another example is provided by rule $\forall t1>t2 \Rightarrow DiseaseZ.Stage[t1]<DiseaseZ.Stage[t2]$. In this case, the analysis of the rule would result in a set of online tests that would be implemented as runtime monitors deployed on the federated datasets. The monitors would be triggered by changes that affect the value of one or more *DiseaseZ.Stage* fields, would check that the new value of the fields is greater than the old values, and report a problem if this is not the case.

#### 3.2.2. Data Generation

In some cases, real datasets may be inadequate and prevent some of the tests from being run. For example, a test that checks whether n types of data collected from m different databases are aggregated correctly may need data of exactly the right types in the different databases involved, and this data may not be present. Moreover, for some more sophisticated tests, data with a specific distribution may be needed. Finally, in cases where an accurate test oracle is defined, differential testing may be needed to assess the outcome of a test.

When generating synthetic data sets, the data must typically have characteristics representative of real data and cannot be simply randomly generated. We will leverage existing techniques for generating meaningful, valid synthetic data (e.g., [9,10]) and will extend them to support distributed data sources. Our framework will use information about relationships and dependencies among database schemas, generate synthetic datasets according to this information and on the generated tests, and populate data across the distributed databases. As an example of this scenario, consider another rule from Section 3.1: *AttrX.domain=numerical; AttrX.unit=foot; AttrX.range =[1,8]*. To execute a test that targets this rule, the framework may need to add to a database suitably tagged synthetic patients, whose AttrX's value violates the domain, unit, or range constraints.

## 4. Current State of Testing Framework

We are in the process of implementing a prototype of our framework. Presently, we are building a brain tumor translational informatics test bed that contains a wide range of real data from the brain tumor study described in Section 2. The datasets come from the TCGA[3], Rembrandt[2], and NBIA[4] data repositories and

[4] National Biomedical Imaging Archive http://imaging.nci.nih.gov

from Emory University, Thomas Jefferson University, and Henry Ford Hospital (three of the collaborating institutions in ISBTRC[1]). Table 1 lists the datasets and data management systems for the test bed. The data management systems will be deployed on a set of virtual machines for easier portability and distributed deployment. The test bed will allow us to create scenarios for various types of errors, such as incorrect mappings from local identifiers to global identifiers (i.e., foreign key relationships – the molecular, imaging, and clinical data stored in different systems should be linked through subject and/or specimen identifiers), missing data with respect to the study protocol (e.g., missing whole slide images for some patients), and updates to invariant data elements (e.g., disease diagnoses, dates of death). In parallel, we are incrementally building our test model by defining rules for properties, constraints, and relationships among the databases in the study. We have defined several rules using OWL and SWRL for immutable data elements, the study protocol, and foreign key relationships. Our test bed will eventually provide the end users with (1) a set of test cases that are automatically generated using the test model, (2) a method to automatically execute the test cases, and (3) a report of the test outcome for users.

## 5. Conclusion

Testing is often overlooked in federated settings and accomplished via one-off implementations. An integrated middleware framework can provide a cost-effective solution to this problem. Such a framework should enable researchers and database administrators to: (1) specify a description of the correct state and function of the system as a set of rules expressing dependencies, relationships, and constraints on data sources and datasets; and (2) create, based on the identified set of rules, relevant test scenarios for the federated environment, test cases that realize such scenarios, and suitably tagged synthetic data, when the existing data are not sufficient.

| Type of Dataset | Data Management System |
|---|---|
| **Neuroimaging Data** | |
| Radiology images in DICOM format, imaging metadata | Virtual PACS[5], xNAT[6] |
| Manual annotations provided by neuroradiologists | AIME[5,7] |
| **Molecular Data** | |
| mRNA, miRNA, methylation data, copy number, sequence data | in-house developed database with file system for data files |
| **Clinical Data** | |
| Clinical data (including days to death, diagnosis, year of initial pathologic diagnosis), specimen (e.g., sample type), etc., data | i2b2[8], in-house developed database |
| **Pathology Data** | |
| Whole slide microscopy images as 20x and 40x magnification, image metadata | caMicroscope[9] |
| Computer- and human-generated annotations of pathology images | PAIS[10] |

**Table 1:** Datasets and data management systems in the test bed.

**Acknowledgements**. Partially funded by: Federal funds from the National Cancer Institute; National Institutes of Health Contracts HHSN261200800001E, 94995NBS23, and 85983CBS43; NIH PHS Grants (UL1 RR025008, KL2 RR025009 or TL1 RR025010) from the CTSA program of NCRR; NHLBI R24 HL085343; NIH U54 CA113001; NLM R01LM009239-01A1, and BISTI P20 EB000591; NSF award CCF-0725202.

## References

1. Grethe J.S. et al., Biomedical Informatics Research Network: Building a National Collaboratory to Hasten the Derivation of New Understanding and Treatment of Disease. Stud Health Technol Inform, 2005, 112: p. 100-109.
2. Oster, S. et al., caGrid 1.0: An Enterprise Grid Infrastructure for Biomedical Research. Journal of American Medical Inf. Assoc. (JAMIA), 2008, 15(2): p. 138-149.
3. von Eschenbach, A.C. and K. Buetow, Cancer Informatics Vision: caBIG. Cancer Inform., 2006, 2: p. 22–24.
4. CardioVascular Research Grid. http://cvrgrid.org, 2011.
5. Mendis, M. et al., Integration of Hive and Cell Software in the i2b2 Architecture. AMIA Annual Symp., 2007: p. 1048.
6. Service-aware Interoperability Framework (SAIF). https://wiki.nci.nih.gov/display/SAIF/CBIIT+SAIF+Wiki, 2011.
7. Web Ontology Language. http://www.w3.org/TR/owl-features/, 2011.
8. A Semantic Web Rule Language (SWRL). http://www.w3.org/Submission/SWRL/, 2011.
9. Chays, D., et al., A Framework for Testing Database Applications. ACM SIGSOFT International Symposium on Software Testing and Analysis, 2000: p. 147-157.
10. Chays, D., et al., An AGENDA for Testing Relational Database Applications: Research Articles. Software Testing Verification, and Reliability, 2004, 14(1): p. 17-44.
11. Orso, A. et al., Techniques for Classifying Executions of Deployed Software to Support Software Engineering Tasks. IEEE Trans. on Soft. Eng., 2007, 33(5): p. 287-304.
12. Engler, D., et al., Bugs as Deviant Behavior: A General Approach to Inferring Errors in Systems Code. ACM Symposium on Operating Systems Principles, 2001: p. 57-72.

[5] https://cabig.nci.nih.gov/tools/IMAG_Middleware
[6] eXtensible Neuroimaging Archive Toolkit, http://xnat.org
[7] Annotation and Image Markup, https://cabig.nci.nih.gov/tools/AIM
[8] Informatics for Integrating Biology & Bedside http://i2b2.org
[9] https://cabig.nci.nih.gov/tools/caMicroscope
[10] Pathology Analytical Imaging Standards https://web.cci.emory.edu/confluence/display/PAIS/Algorithm+Validation