

AUTOMATING DATABASE CURATION WITH WORKFLOW TECHNOLOGY

A Thesis presented to the Faculty of the Graduate School
University of Missouri-Columbia

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
GAURAV ASHOKKUMAR SANGHI

Dr. Toni Kazic, Thesis Supervisor

MAY 2005

The undersigned, appointed by the Dean of the Graduate School,
have examined the thesis entitled

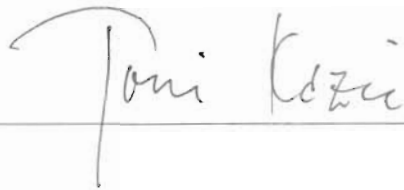
AUTOMATING DATABASE CURATION WITH WORKFLOW TECHNOLOGY

Presented by Gaurav Ashokkumar Sanghi

A candidate for the degree of Master of Science

And hereby certify that in their opinion it is worthy of acceptance.

Dr. Toni Kazic



Dr. Chi-Ren Shyu



Dr. Mary Polacco



ACKNOWLEDGMENTS

I extend my sincere gratitude and appreciation to the many people who made this thesis possible. Special thanks are due to Dr. Chi-Ren Shyu and Dr. Mary Polacco for their kindness in serving on my thesis committee. I'd also like to thank specially to Dr. Chi-Ren Shyu for giving me an enormous amount of knowledge in his lectures on databases. I am also highly indebted to Dr. Toni Kazic, my advisor, for the tremendous amount of knowledge I gained from her and also for the confidence she had shown in me in the most difficult times and for making me believe what I am capable of doing.

I benefited from discussions with Amithreddy Gosukonda, Avanthi Mummaneni, Raman Seth, Sinéad Boyce, Jiahui Jiang, Archana Chikkabel, Nandini Basu, Sumit Sadekar, Bryan Cannon, Julie Stuppy, Thomas Campbell, Arpit Ghoting, Phani Chilukuri, Radu Brumariu, Andrew Gilmore and William B. Wise. This work is supported by a grant to T.K. (GM 56529) from the U.S. National Institutes of Health.

And lastly, I would like to thank my family for giving me moral support, guidance and encouragement right from the beginning of this research thesis and all the way to the finish line.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	ii
LIST OF FIGURES	v
LIST OF TABLES	vi
LIST OF ABBREVIATIONS	vii
ABSTRACT	viii
1 Introduction	1
1.1 Motivation	1
1.2 The Problem	1
1.3 Types of Databases and their Processes	7
1.4 What is Workflow?	11
1.5 Publication Model of Workflow	13
1.6 Proposed Solution	13
1.6.1 Synopsis of Results	14
2 Languages and Platforms	16
3 Results	16
3.1 Process Design	16
3.2 Overview of The <i>Agora</i> 's "Back End"	19
3.3 Agents in the Databases	21
3.4 Automated Workflow	22

3.4.1	Database Structure and Perl scripts	22
3.4.2	Reviewer Selection	24
3.4.3	Review and Revision	27
3.4.4	The Arbiter's Decision	29
3.4.5	Data Inclusion	31
3.4.6	Rebuilding New Databases	46
3.4.7	Adaptive Workflow Model	46
3.4.8	Re-engineering the Workflow of <i>Nomenclature</i>	57
4	Discussion	58
4.1	Future Improvements	63
	BIBLIOGRAPHY	66

LIST OF FIGURES

Figure	Page
1. Previous Curation Workflow in <i>Klotho</i>	4
2. Previous Curation Workflow in END	5
3. Previous Curation Workflow in UM-BBD	6
4. User's View of The <i>Agora</i>	9
5. Data's View of The <i>Agora</i>	10
6. Back End Functions of The <i>Agora</i>	20
7. Database Structure and Operations	25
8. Reviewer Selection Process	28
9. Review and Revision Process	30
10. Including Data in Data Files	32
11. Parsing <i>Glossa</i> Data into Prolog Facts	33
12. Updating Isomer Files	42

LIST OF TABLES

Table	Page
1. Sample Data for a Molecular Structure Deposit	36
2. Data Extracted for a Molecular Structure Deposit	37
3. Data Entries for a Molecular Structure Deposit	38
4. Sample Data for a Term Deposit	39
5. Data Extracted for a Term Deposit	40
6. Data Entries for a Term Deposit	44
7. Data Deletion for a Term Deposit	45
8. Sample Data for a Enzymatic Reaction Deposit	50
9. Data Extracted for a Enzymatic Reaction Deposit	51
10. Data Entries for a Enzymatic Reaction Deposit	56

LIST OF ABBREVIATIONS

<i>abbreviation</i>	<i>full name</i>
END	Enzyme Nomenclature Database
BND	Biochemical Names Database
UM-BBD	University of Minnesota Biocatalysis and Biodegradation Database
JCBN	Joint Commission on Biochemical Nomenclature
CGI	Common Gateway Interface
SQL	Structured Query Language

ABSTRACT

Building and curating databases efficiently is one of the major problems in scientific databases today. The first problem lies with the exponential growth in scientific data. With the never-ending process of research and inventions in the scientific community, the dataset is very large and constantly increasing in growth rate. Manual curation is no longer efficient to handle this amount of data and the database becomes error prone and inconsistent. When approved a datum may necessitate changes in existing data distributed over multiple tables and fields in one or more databases. The second major problem is the processes used to deposit, review, revise and finally accept data into a database. Each database has its own way of allowing people to submit new information, verify the information that was submitted, and depending on the verification process, discard the data or include them in the database. The process of including data in databases itself might be very different and complicated, depending on the relationships that exist among the data and the structure of the database's design. When it comes to building databases of biological information, staffs of Ph.D.-level scientists are needed to curate and maintain data, and each database has its own domain model. All these factors make it expensive to build databases of biological information. Costs could be reduced if the scientists who curate the data are provided with data that are initially reviewed by other experts for accuracy and consistency. Since scientific expertise is distributed around the world, this allows the best experts to contribute knowledge. A common platform that implements a well-accepted work process is needed to support such community curation.

Using The *Agora* as an example, I have applied workflow methodology to automating the human processes involved in the curation process to minimize human intervention and eventually reduce errors in the database. In an attempt to maximize productivity in building and maintaining databases, I emphasized automating

the flow of work in the curation of new data. Workflow technology is a method of understanding the flow of data from the source to its final destination, identifying each of the intermediate stages, and then understanding each stage and automating as many of the processes and sub-processes as possible. I have applied the workflow used in reviewing manuscripts for publication to curating these biochemical data.

The purpose of this thesis is to determine the complex relationships that exist between information technology and community database applications, in an attempt to determine how workflow technology can be optimized to improve curatorial productivity. I studied four different data curation groups, three databases and one group that produces text synopses of the literature. I have managed to design a single framework for data deposits for all the databases so that the process could be automated and simplified. I was able to use this common framework for the synoptic group's workflow, but the task of re-engineering their processes is much harder and continues. The thesis focuses on the "back end" of The *Agora*, which involves the intermediate databases and the final output databases that store different categories of data in the form of different Prolog predicates with their own tracking methods.

This model is flexible enough to accommodate additional processes idiosyncratic to particular groups of curators, such as those for enzymatic reactions, biochemical terms, and molecular structures. Also, future improvements can be easily incorporated into this model without making any major design changes and with straightforward coding procedures. This thesis demonstrates the application of workflow technology to intellectually complex, geographically distributed, multidisciplinary scientific processes.

1 Introduction

1.1 Motivation

In today's world of explosive scientific growth, community databases are becoming increasingly popular for storing and providing huge amounts of data in various forms. However, handling such huge amounts of information efficiently and consistently has become a problem. GenBank increased from 11 billion base pairs of DNA to 45 billion from 2000 - 2004, approximately a 100% increase in data size every year. The same goes for DNA sequences which were around 10 million in the year 2000, and have increased four times to a total of 40 million at the end of four years in 2004 [13]. For the HTML text of the *Enzyme Nomenclature*, the average number of users per week increased from 5515 in year 2000 to 20,392 in the year 2004 [14]. With data and database usage increasing at this tremendous rate, the existing methods for curation and management of data and code are rapidly being outpaced. Considering the rate at which data are being generated for databases, I need ways to speed up the building and maintenance of curated databases.

1.2 The Problem

Building and maintaining scientific databases is complex and expensive because:

- Research and discoveries are increasing scientific data at an exponential rate, and these data must be captured in databases in order to interpret the results of the experiments.
- The relationships among the data are complex.
- Human errors introduced during manual data management makes database inconsistent.

- Processes to deposit, review, revise and include data in databases are different for different databases.
- Scientific related data are present in multiple databases.
- The number of curators and available resources are far smaller than those needed to curate essential scientific data using current methods.

These problems are illustrated by the history and characteristics of the databases that participate in The *Agora* [7]. Each of the participating databases, *Klotho*, *Atropos*, BND, END, and UM-BBD, differ not just in the types of data collected, but especially in the way they were curated. To illustrate, I describe three examples, shown in Figures 1 - 3. *Klotho* contained information on both molecular structures and synonymous names; proposed entries were checked by three different people; and a variety of command-line Prolog predicates and Perl scripts were used to track this process, identify fully and partially checked proposed entries, and accession the new data. For *Enzyme Nomenclature* (hereafter referred to as the *Nomenclature*), the process is much more complicated. Information on possible new enzymes reaches the curators by email, reading the literature, phone calls, or the sending of reprints; a curator drafts an entry ; the draft entry is circulated to the members of the JCBN and modified; the modified entry is posted for public comment; the entry is further modified as needed and either re-reviewed or approved; and incorporated into the JCBN web site. Modification of existing entries, whether major or minor, are (mostly silent) posted on the JCBN web site. To enter the *Nomenclature* data into END, the HTML from the web page was parsed, the parse checked by computer scientists and one biochemical expert, auxiliary data are added, and the data accessioned into END. UM-BBD focusses on reactions metabolizing xenobiotics, many of which have not been sufficiently described to warrant classification in the *Nomenclature*. Students informally review the literature on a reaction and make a web page about it

as part of their coursework. The proposed page is reviewed by the database’s curator, modified if needed, and posted to a web site. Some of the data are accessioned into a supporting database.

Though the processes differ significantly, there are some common elements. Each group acquired data, reviewed them, revised them, and made them publicly available. Another commonality is the way the data from different database interact. Some entries in UM-BBD are ready for incorporation into END. Since *Klotho* wanted to enter structural information for each small molecule in the *Nomenclature*, that group tabulated molecule names used in END and checked these for consistency and synonymy. Many inconsistencies and synonyms were found, which were sent back to the JCBN with suggestions to resolve them. (This effort was the nucleus of BND, which has since grown to include other types of relationships among terms besides synonymy.) The databases differ in the granularity of the data (from small for *Klotho* to very large for END); in how automated data management is (high for *Klotho*, essentially none for UM-BBD, and intermediate for END); who can change the data and how; and whether changes are recorded (fairly well for *Klotho*; relatively little for END; and apparently none for UM-BBD).

It was obvious that the domain model for END could be easily extended to other reactions and processes, provided ways were found to more efficiently manage curation of those data. For example, many reactions implicated in human dementias are enzymatic, but many more are not [16]. But to make this possible, I needed a way to automatically analyze the incoming data and put them efficiently at the right place in the right database. This reduces manual work and as a result, human errors are reduced and completely eliminated to some extent.

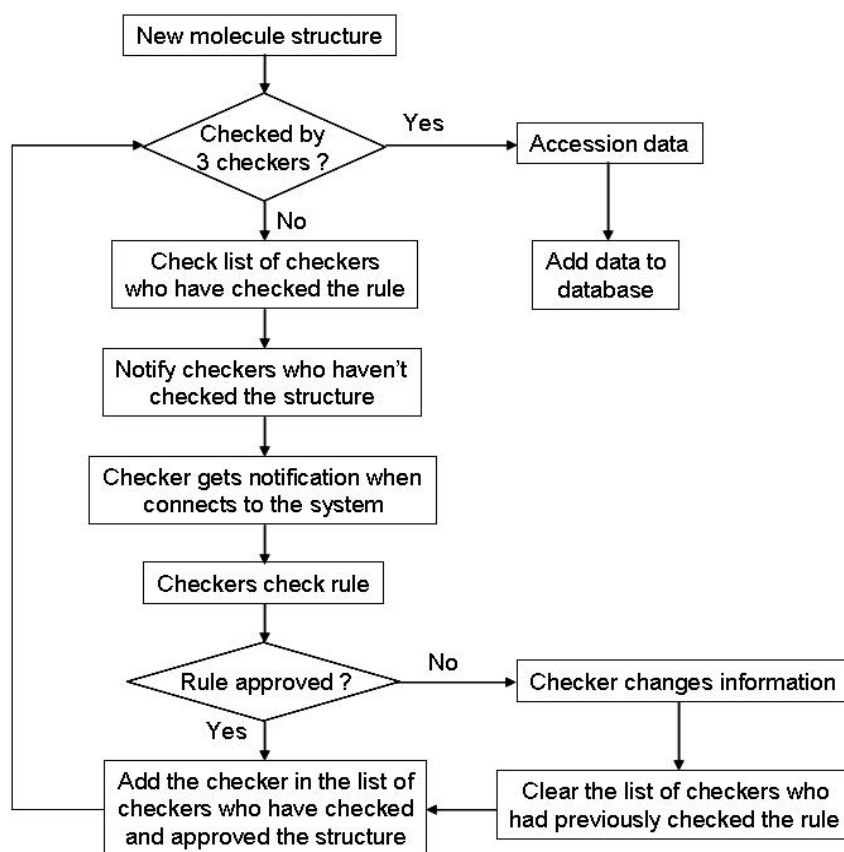


Figure 1: Previous curation workflow in *Klotho*. Everytime a new molecule structure was found, three checkers checked the structure before accessioing and including it in the database. A list of checkers was used for monitoring the progress of checking by each of the checkers. After a structure was approved by any of the checker, the checker was added in the list of checkers who have approved the data. However, if a structure was modified by any of the checkers, the list of checkers who had approved the structure was reset and the checkers were required to check the new structure again and approve it.

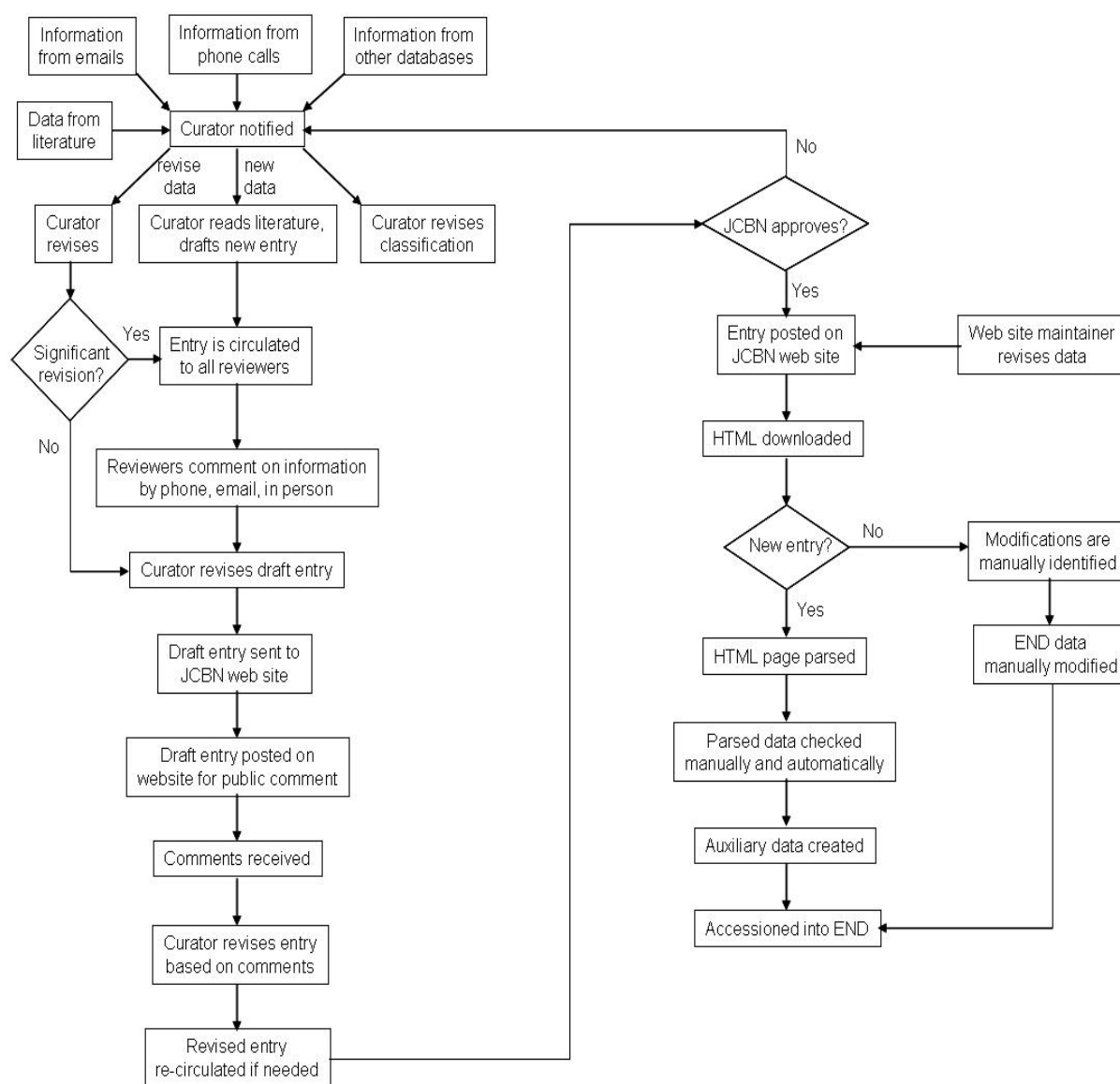


Figure 2: Previous curation workflow in END. The curator received new information in various forms and revised existing data drafted new entry. The entry was reviewed by a set of reviewers and after revision, the entry was sent to JCBN web site for public comments. Based on the comments, the entry was revised and reviewed if needed and then sent to JCBN for approval. On approval, an HTML page for the entry was posted on the web site. If it was a revised data, the modifications were identified and accessioned manually. If the data were new, the HTML page was downloaded and manually parsed to create auxiliary data and then accessioned into END.

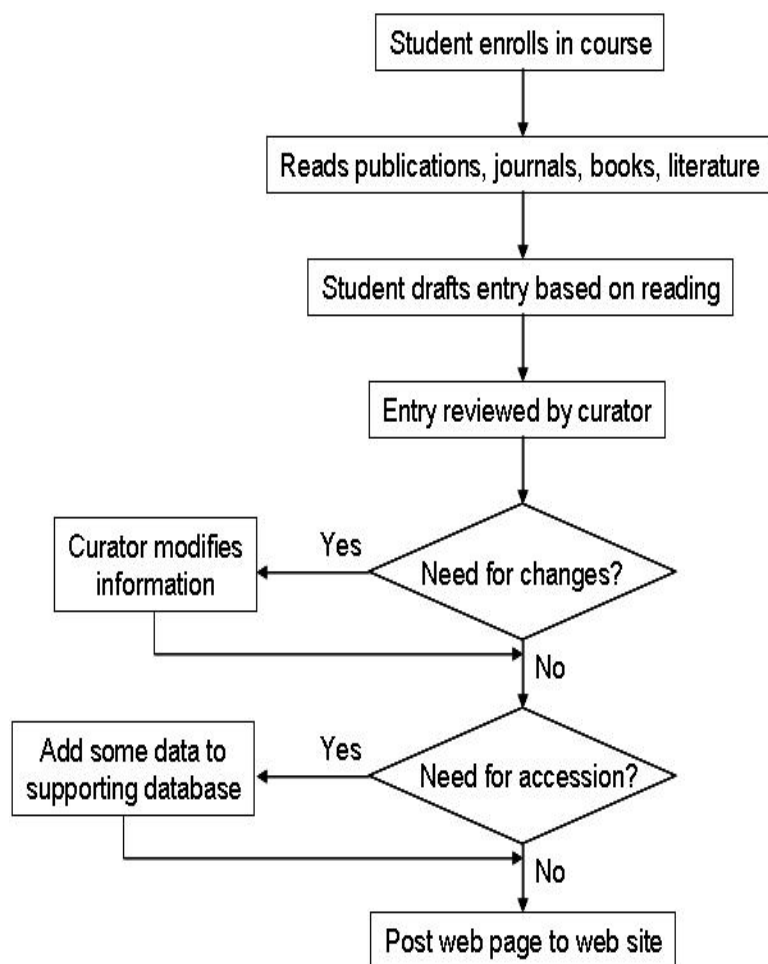


Figure 3: Previous curation workflow in UM-BBD. A student enrolled in a certain course read publications and drafted an entry. This entry was reviewed by the curator and modified if needed. An HTML page was written for the entry and hosted on the web site. Some of the information was accessioned and stored in a database.

1.3 Types of Databases and their Processes

A curated database is a database developed by a curator, or a group of curators, who have very good expertise in the associated domain. A curator finds new information, removes redundancy and inconsistencies from current and submitted data, incorporates annotations, and adds reference information and cross-references to external databases. Thus curation can be defined as the process of verifying and enhancing data in databases. All these processes require extensive scientific expertise and are time consuming, and that makes it very difficult to build and maintain new databases. Especially in the case of biological databases, it is really hard to keep up with new information as it is released. Since this entire process of reviews and revision involves lot of processes, subprocesses and decision making for their proper operation, exorbitant manual effort is required to keep data curation running smoothly, efficiently, and accurately.

Different databases have their own methodology for accepting new data from the public and for curating the data. GenBank is the NIH annotated collection of all publicly available DNA sequences [13]. Submitting new information to GenBank is required to submit papers for publication. Revisions or updates to GenBank entries can be made at any time through emails. These revisions and new additions are curated to a minimal extent and made available to the public community. On the other hand, BioMagResBank (hereafter referred to as BMRB) collects, archives, and disseminates the important quantitative data derived from NMR spectroscopic investigations of biological macromolecules [10]. Relevant data are deposited through a set of forms by the scientists who generate the data, and after the information is submitted, in consultation with these scientists, BMRB resolves any problems with the self-consistency and completeness of the deposition. Unlike GenBank, BMRB has an extensive data curation process performed by the scientists of BMRB.

The *Agora* is an electronic infrastructure for the scientific community to use and curate information on biochemistry, molecular biology, and physiology in multiple, independent databases. The databases that currently participate in The *Agora* are END (the Enzyme Nomenclature Database — a database of enzymatic reactions), BND (Biochemical Names Database — a database of molecule names, terms, and relationships), *Klotho* (a database of molecule structures) and a database of dementia-related information. There are currently three types of deposit which can be made through The *Agora*: new reactions, new terms and relationships for existing molecules, and new structures of molecules. These deposits go in END, BND, *Klotho*, and the dementia databases. The *Agora* is comprised of two main sections: the “front end”, which is the web interface for people to deposit, review, revise and curate the information using browsers; and the “back end”, which manages the curation process for the participating databases. Illustrations of the user’s point of view and the data’s “point of view” in The *Agora* are given in Figures 4 and 5 respectively.

As shown in Figure 4, the user only experiences the “front end” of The *Agora*. The user deposits information and then waits for a response from the reviewers. The reviewers, who are selected in the “back end” of The *Agora*, review the deposited information and send their decision and comments to The *Agora*. If a revision is requested, detailed information on the review is sent to the depositor, who revises the information and submits again. The same reviewers review the revised information and decide the fate of the data. In either case, the information is sent to the arbiter for the related database, who makes the final decision and either includes the data in the database or rejects the data. There are only a few designated experts who are selected as curators for databases.

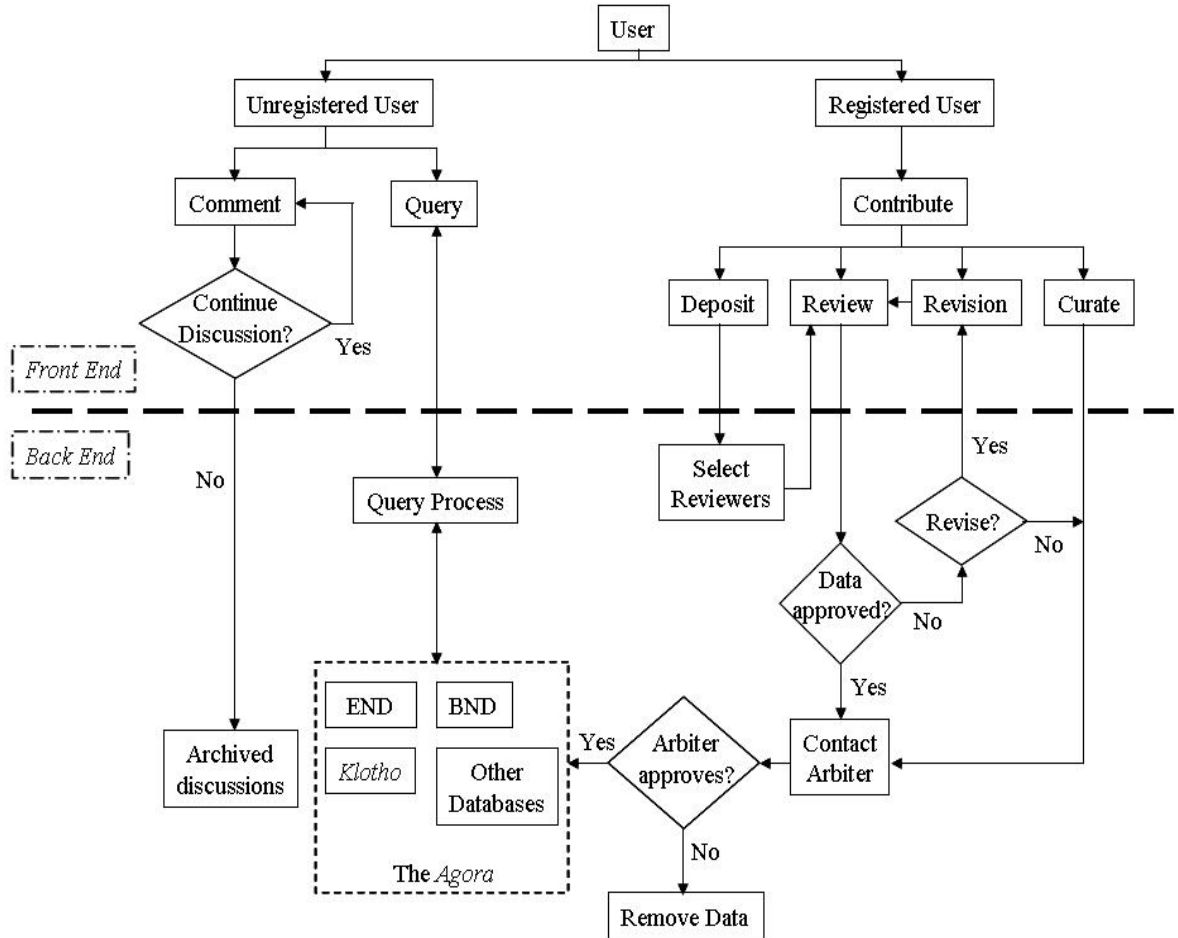


Figure 4: A user’s view of The *Agora*. The dashed line separates the two sections of The *Agora*, the “front end” and the “back end”. The user is aware of the “front end” of The *Agora*, which is the web interface, but is unaware of the processes of the the “back end”. The design of the “back end” is described in Section 1.3.

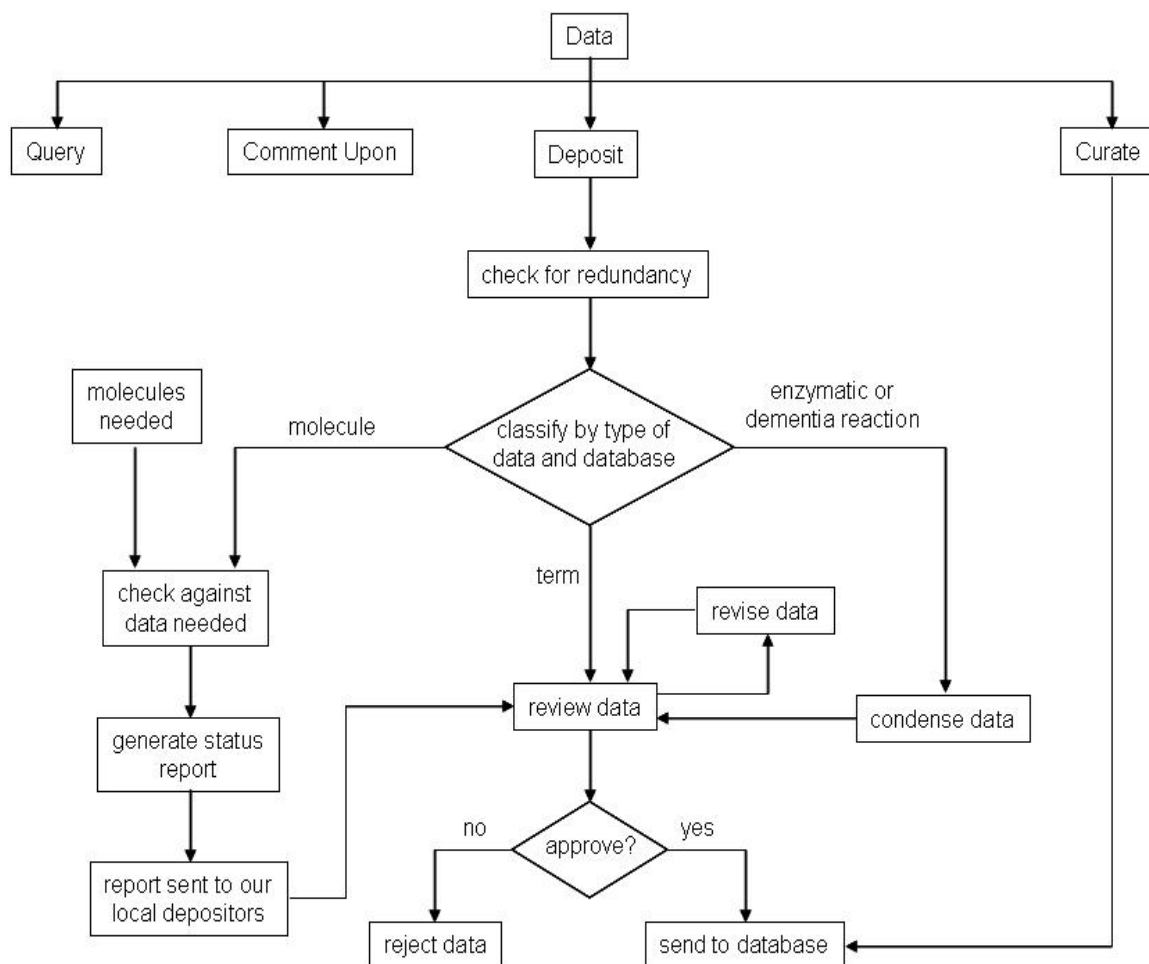


Figure 5: The data's view of The *Agora*. The data present in The *Agora* can be queried, commented upon, deposited, or curated. When deposited, they are checked for redundancy; then for the database to which they are to be deposited; reviewed; revised if needed; and finally included into the database if approved.

1.4 What is Workflow?

Without a thorough understanding of workflow, productivity gains will be incomplete and ineffective. Workflow can be described as the coordinated execution of simple or complex activities (tasks) by human or automatic executors (agents). It is the path and systems used in the linked flow of activities to achieve an aim. The flow defines where inputs are initiated, the location of decision points, and the alternatives in output paths. Workflow is used in systems that perform automatic routing of events or work-items from one agent to another. Clarence Ellis breaks groupware into four categories: keepers, synchronizers, communicators, and agents [5]. Keepers are the members of the group that maintain the information; synchronizers coordinate the subprocesses; communicators maintain the communication between the different subprocesses; and agents do the actual manipulation and analysis of information. In curated databases, keepers are the group of people who collect and curate the data; synchronizers ensures that all processes are executed in the right order; communicators maintain the communication between the depositors, reviewers, and the arbiter; and the agents are the human or automatic agents who organize and manage the data in the databases. Workflow, like groupware, also has these four categories. In a groupware, it is not necessary that all these four categories be present in a process. Even if there are one or more categories missing, a process can still form a groupware. However, this is not the case with workflow. For workflow, it is necessary that all the four categories are present. Synchronizing and coordinating subprocesses in a process or a group of processes forms the most critical part of workflow technology. While automation and task consolidation remain the cornerstone of workflow optimization, a number of additional factors must be considered including the stochastic nature of the workload, the availability of human resources, and the specific technologies being used in the process [15].

Approaches to developing such workflow systems have both commercial and academic origins. Commercial systems have evolved from work on forms-based image processing systems and groupware [9]. Academic research has focused mainly on process modeling and database transaction issues [17]. Workflow management has been proposed in programming-in-the-large for heterogeneous and distributed information system environments. This type of programming is done by larger groups of people or by smaller groups over longer time. This type of programming produces code that cannot be understood without a divide and conquer approach because of its size or complexity. Researchers in software process modeling have developed formal languages for process modeling that have been used to define process analysis, simulation and execution techniques, tools and integrated environments. Process analysis is understanding the process in order to develop ideas for improvement of the process, and simulations are experiments executed on the model over a period of time to test results. Database transaction research focuses on extending traditional transaction semantics to support long duration and/or cooperative transaction models [1, 2, 17]. Workflow implementation has been mostly seen in automating mechanical processes like machine operations or sending paychecks on a timely basis. From the examples of workflow implementation in industry, it is clear that workflow can be re-engineered only in the real work environment. It is necessary to see and understand the current workflow and the processes involved in it, examine the inputs and outputs of each process, relate them to other stages of the workflow, and identify the possible processes which can be automated or modified to improve the overall efficiency of the process. However, workflow technology has yet to be implemented in database curation processes, and this was one of the key issues in this research.

1.5 Publication Model of Workflow

The process of database curation involves the curator reviewing existing data; modifying them depending on the new information in the literature, and adding new data. Since new information is available daily, the data should be made up to date on a regular basis. This examination of data by curators is very similar to the scrutiny of submitted manuscripts by reviewers. In both cases, body of proposed information is studied for accuracy and consistency with other knowledge. For publications, proposed information is generated and structured by a submitter and considered by expert peers. These peers can recommend the information be incorporated into an archive (the journal) as is or following revision, or reject it. Submission and review are performed by all members of the scientific community and is universally accepted by them. Because of this acceptance and the common belief that peer review improves the quality of publications, we decided to adopt this model workflow as the framework for The *Agora*. In The *Agora*, I designed the process of data submission so that the scientific community can submit and verify data before they are included in the database. Letting experts world-wide curate and review information partially curates the data, making the work of database curators less tedious and time-consuming.

1.6 Proposed Solution

The way to faster and more efficient curation of databases is to either provide automated curation of databases, or at least to provide information to curators so that their workload is reduced or made easier. Since completely automated curation would involve communicating with numerous other databases and the literature holding related information in biology and biochemistry fields, I have left this for future development. I decided a more feasible solution is to make work easier for the curators. The ability to define, execute, and track processes is central integrating and making

productive new technologies such as automated workflow in a widely distributed setting [6]. So I decided to design a standard workflow for the curation process of the various databases participating in The *Agora*. Certain designated scientific experts, called the reviewers, would look at the information, references, and comments made by the depositor and check for their validity and authenticity. To help ensure that the information is acceptable in general, I have two different reviewers reviewing the same information and giving their independent suggestions. Their feedback is checked to see if they both agree on the data and if so, the data are passed to another expert, called the arbiter, to look at the information again and approve it. There is only one arbiter for each database, who is a designated expert in the respective field. By using repeated and independent reviews of publicly deposited information, the data are much more accurate and consistent. With added information from depositors and reviewers, the task of the curator is made much easier. Although the tasks of curator could be simplified, the overall process would still be very slow. In order to speed up the overall process, I decided to automate as many processes in the workflow as possible.

1.6.1 Synopsis of Results

I studied four existing workflows. Three were local databases developed in our laboratory. The fourth was a geographically distributed group of people who synthesize information from the literature, or approve the synthesis, or publish the synthesis on a web site. The databases' workflows were easily studied, but I had to rely on descriptions of the workflow of the group by one person. The description changed gradually because the workflow and communication with the reported both changed. These and other factors made it much harder to implement a final version of the workflow. Nonetheless, I managed to successfully implement the workflow technology

and automate it on The *Agora* and partially for the group. The major results can be listed as follows:

- Human agents were replaced by automatic agents in all the possible processes and subprocesses of database curation. These automatic agents were self-triggered and so there was no wait time in proceeding to the subsequent steps of curation. As soon as an event happened, the next step was started immediately, saving a significant amount of time.
- Workload was equally distributed among the best and most experienced personnel at intermediate levels. This ensured that the response could be received within a reasonable amount of time, and selecting the best reviewers guarantees the best results from the review and revision stages.
- Data were added in the database with no errors. As a result, the database was more consistent and robust. Since the data were related to each other at various levels, correcting any errors was a tedious process and involved a lot of manual work in tracking each error and correcting it.
- All relationships between the existing and new data were correctly maintained. The automated process updated each related data structure in the database.
- Workflow technology for remote processes is difficult to design and implement. This is one of the major limitations of workflow technology.
- Applying workflow technology to distributed groups of volunteers is extremely difficult. Determining the current workflow accurately is very difficult; each person can have a different workflow; and there is much less incentive for these groups to make difficult changes.

There are some important limitations of this technology and in its current implementation of The *Agora*:

- Since the arbiter is the final authority, I assume the data coming from the arbiter are correct, and so the database is still prone to errors introduced by the arbiter.
- There are processes, such as review, which could not be automated in the current implementation and so I still need human expertise to review and curate data.

In spite of these limitations, this model can be applied to other processes which are idiosyncratic to a particular group of curators. This makes the model flexible enough for not only implementations in other related processes, but also in including future improvements.

2 Languages and Platforms

I used Perl (version 5.0004) on a Sun E450 running Solaris 8.0. I have used MySQL database for storing intermediate data from the “front end”, and for extracting the final data for including in our databases. I used Quintus Prolog (version 3.4) for our final databases *Klotho*, BND, and END.

3 Results

3.1 Process Design

In order to achieve an optimal solution for the problem, I had to first study the process so that an appropriate model for the workflow could be designed. This designing process involved the following steps.

- *Observing existing workflow.* This was the primary process in the case of *Klotho* and END. Since these two databases had partial workflow and some data management processes already existing in the current implementation, it was necessary to study them so that the new workflow can adapt the existing processes and integrate smoothly.
- *Identify common processes and bottlenecks.* In this step, I studied the common processes involved in the curation of data for each of the participating databases. This helped in identifying ways to design a common workflow for these databases, as well as for potential future databases that are similar to the current databases in terms of data and curatorial processes. During this step, I also identified common bottlenecks in the overall workflow, which were mainly human operated processes, and were the key targets in automation.
- *Curating sample data for databases.* In this step, I ran the curation process for sample data for each of the databases and studied the various processes that were involved in the curation process. This not only helped in identifying the processes, but also revealed some hidden subprocesses that could make substantial difference when automated.
- *Prototyping automation.* Based on the results from sample data curation, the processes and subprocesses were separated into two lists; one that could be automated and one that were not possible to automate immediately or in the near future. All possible processes were automated so as to accomodate information processing for all the different databases.
- *Testing prototype with sample data and revising workflow.* Once the prototype for the new automated workflow was designed, tests were run using the same data sets used in the initial stages to check if all the processes were included

in the workflow and if the data behaved in the same way as before. Repeated revisions of workflow were made to accomodate all the features for each of the participating databases, and to automate any more subprocesses identified during prototyping and revision.

Using these steps, I sketched out the entire deposit, review, and revision process that the data go through before being included in the public databases. This design is shown in Figure 6. The design was made by repeatedly making sample deposits for all the types of data currently supported by The *Agora* so that I could list the functions needed for each type of deposit and how their data were handled.

Although the data are sent to their respective databases, it was observed that the curation process for each type of data could follow a common workflow, as shown in Figure 6. The flow was very straightforward sequential process. Data and processes involving the “front end” of The *Agora* are marked in dashed boxes. These were the processes or decisions made at the “front end” by human agents like the depositor, the reviewer, or the arbiter, and could not be automated at this stage since they involve data coming from experimental results rather than from something that could be searched by automatic agents. Maybe in future, it might be able to automate this process once I have complete information on all the possible resources for a particular type of information. Also, decisions made by the arbiters come solely from their personal experience and expertise, and that process is something that can never be automated. However, the remaining boxes in the Figure involved processing and analysing data in the database, so they could be automated to speed up the process. These processes were handled by humans until now and involved looking up the data present in the database and making decisions based on them. These decisions comprised looking at the type of information submitted, checking when a deposit is completed, selecting appropriate reviewers by looking at the list of available

reviewers, keeping track of reviews and revisions by monitoring the activities in the database, informing the depositors of their deposit’s progress, checking the feedback from reviewers and arbiters and taking appropriate actions, *etc.* These decisions and selection procedures followed a set of rules for each of the data types, and so it was possible to automate them and let automatic agents make the decisions and perform other actions on the data. This has the benefit of instantaneous response to incoming data rather than waiting until a human agent personally looked over the data and decides what needs to be done.

3.2 Overview of The *Agora*’s “Back End”

The process designed for The *Agora*’s “back end” is shown in Figure 5. The figure outlines the processes that act on data contributed *via* or called by The *Agora*. Various operations are performed on the data. Users can run queries on the existing data, submit comments on them, or deposit new data. The data can also be curated by their curators. When a new information is submitted through the “front end” of The *Agora*, the data are checked for redundancy. If the data already exist in the database, the data are discarded and the original depositor notified. If they are new, the database for which the data are deposited is determined. If the data are a new structure of molecule, a local check is made against the data that are needed. *Klotho* maintains a list of molecule structures needed for END. This list is used by our local *Klotho* team, who are generating molecular structures for *Klotho*. If the new data are about a molecule that is in the list of required molecules, a new report is generated and the local team is notified about the new molecule structure so they can promptly review it. If this structure is approved, the molecule is removed from the list of needed molecules and the structure included in the database. If rejected, the information is used by the local team as the basis of a correct deposit. If the information is about a

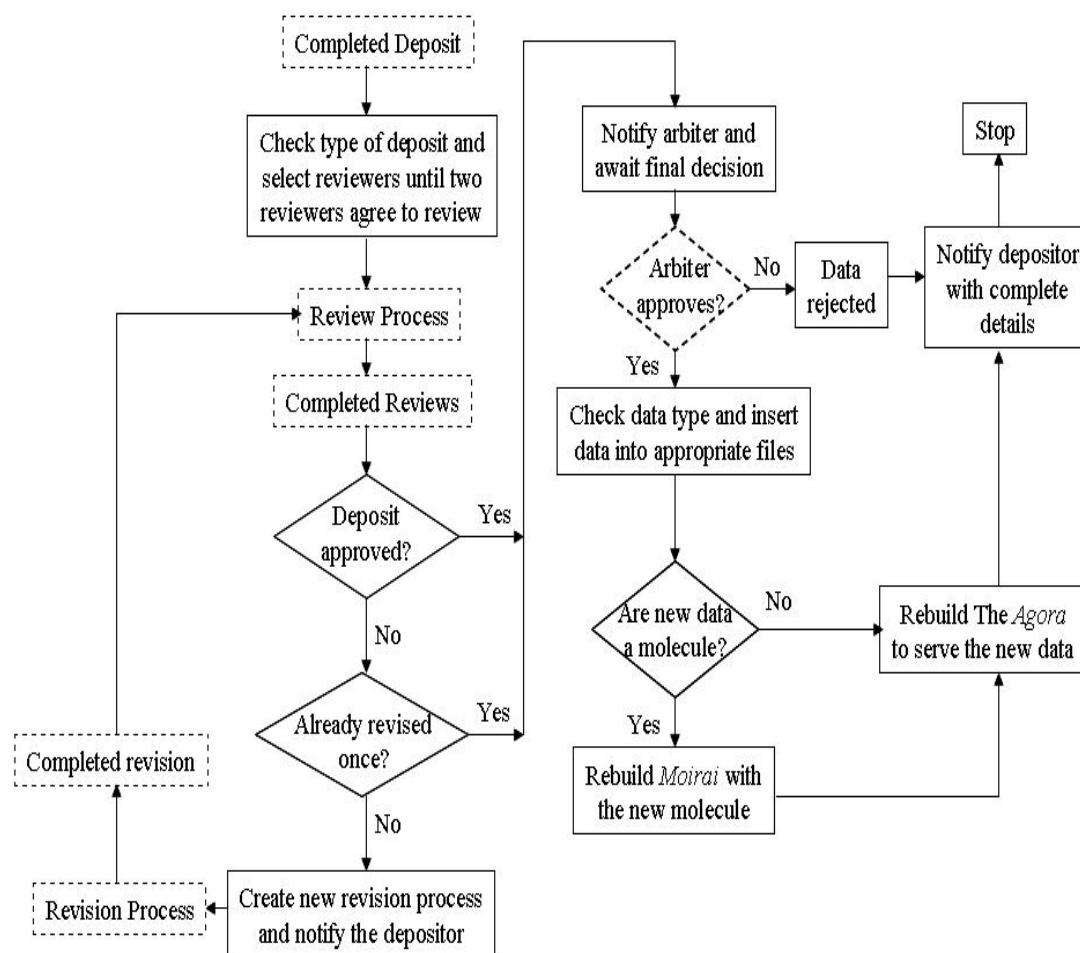


Figure 6: “Back end” functions of The *Agora*. These are the functions performed before the deposited data can be approved and included in the databases or rejected. The dashed boxes represent data coming from the “front end”, or processes that take place in the “front end”. The solid boxes represent data or processes in the “back end”. The data are reviewed by two reviewers, revised by the depositor if needed and reviewed again, and then sent to the arbiter to either approve or reject the data.

synonym or a relationship among existing terms, the information is directly sent to the reviewers. If needed, a revision is made by the depositor, and after another review by the reviewers and approval by the arbiter, the data are included in the database.

For every new deposit made through The *Agora*, the deposited information is to be reviewed by a set of designated people called reviewers, and then depending on their comments and suggestions, it is either approved, or a revision by the original depositor is requested. After revision, the data go back to the same reviewers. After review is completed, the revised deposit and the reviewer's comments are sent to the arbiter. A database arbiter is the final authority to approve or reject the data. If the data are rejected for any reason, all comments from the reviewers and arbiter are sent to the original depositor for their records. If the data are accepted, they are parsed from their *Glossa* bundles into the database's native form for inclusion [8].

3.3 Agents in the Databases

The human agents involved in The *Agora* are the depositor, reviewers, arbiter, and the curators. The automated agent is The *Agora*'s "back end". The depositor is the person who deposits new information on one of the types of data supported by The *Agora*. The depositor uses the "front end" of The *Agora* and submits information using a set of forms. On receiving comments and suggestions from the reviewers, the depositor revises the data originally deposited or provides more information as requested. Reviewers are designated experts who review the information deposited. Arbiters are designated experts who have absolute authority over their respective databases. There is only one arbiter for each database, and this person has the power to approve or reject incoming data on their own judgment. Any member of the scientific community using the The *Agora* can be a depositor or a reviewer. One can be a reviewer only when the arbiter judges one's experience and expertise

and assigns one reviewer status. However, a reviewer can never review his or her own deposited information. On completion of the deposit, the “back end” selects the reviewers who are appropriate for the type of data and notifies them. Once the reviews are complete, the “back end” checks the reviews and either notifies the depositor asking for a revision of the data, or contacts the arbiter to make a decision on the information. Based on the arbiter’s decision, the data are either included in the concerned database if approved, or else archived into The *Agora* if they are rejected. Curators are experts who review the data in the database, remove redundancy, verify automated annotation of data, and additional information.

3.4 Automated Workflow

3.4.1 Database Structure and Perl scripts

A MySQL database is used for storing all the intermediate information about the deposited data coming from the “front end”. The tables used in the intermediate database, and the various scripts operating on these tables is shown in Figure 7.

When a deposit is completed, the information is moved to a table “completed_deposits”, and at the same time, an entry is added to another table “pending_reviews” with default values. This table is used by a Perl script which selects reviewers for this new deposit. This script checks the “pending_reviews” table for any new entries, and if found, checks for the type of deposit, and searches for two appropriate reviewers and updates the values by overwriting the defaults. Using these values, I can keep track of the number of reviewers who have agreed to review the information, the list of reviewers who have refused to review the data, and the time period since the reviewer was notified of their selection. The information about reviewers is stored in a table “reviewers”, which has all the information about each of the designated reviewers for The *Agora*. Once two reviewers agree to review the deposit, their information is

saved in a table “review_details” which is used for our records, as well as when the time comes to notify the reviewers after a revision is completed for this particular deposit. It is customary to normalize tables in a database for better query efficiency. However, in this case, I chose not to normalize “pending_reviews” for better performance. In *The Agora*, the total number of reviewers for all the different types of data was relatively small compared to the number of completed deposits. It was observed that the time for selecting reviewers was less when the list of previously selected reviewers was stored as a single string in the same table, as compared to the time taken for retrieving the list when stored as separate entities in a different table. It is really a tradeoff between normalization and performance. It would be a good idea to normalize the table when the number of reviewers is high.

When the reviews are completed, the information is stored in another table “completed_reviews”, where another script checks the decision made by the reviewers. If both the reviewers agree on the information, the arbiter for the corresponding database is notified, and the information is added as a new transaction for the arbiter. If the reviewers do not agree on the deposited information, the same script checks the table “review_details”, gets the original depositor’s information, and sends an email requesting a revision of the deposit. At the same time, an entry is added in the “revisions” table of the database, as well as a new entry in the “transactions” table for the depositor. When the revision is completed, the data are moved to a table “completed_revisions”, which is checked by the same script used for selecting the reviewers and uses the “review_details” table to notify the same two reviewers about the completed revision and to start the review for the revised information. Once the second review is completed, the script compares the decisions made by the reviewers, and in either case, notifies the arbiter and adds appropriate entries in the tables for the arbiter.

When the arbiter has made the final decision, the information is moved as a new entry in “final_decision” table, along with the decision made by the arbiter. Three separate scripts, one for each of the currently participating databases, *Klotho*, BND, and END, are executed daily that check for new entries in the “final_decision” table. If an entry is found for that particular database, the script extracts the information from this table, and checks if the arbiter has approved the information or not. If approved, the script adds the information to the output database. After adding all information to the output database, the script rebuilds the core application of The *Agora*, which serves the queries and other applications. If the arbiter rejects the data, the script moves the information from the “final_decision” table to another table “archived_data”, which is used for archiving the rejected data for our records and potential future entries. Before the scripts terminate, they remove all other entries from the intermediate MySQL database related to the deposit. The information now is present in either the output database or as an entry in the “archived_data” table in MySQL.

In the entire process, “*session_id*”, a numerical identification assigned to the original deposited information, and the “*user_id*”, a numerical identification for the depositor, reviewers, and the arbiter, are used for tracking information and progress within the intermediate database. Detailed information on the scripts and the intermediate database is explained in the following sections.

3.4.2 Reviewer Selection

When a deposit is completed, reviewer selection begins. The entire process is illustrated in Figure 8. A script checks the type of data deposited. Depending on the three possible types of data — data about a reaction, a molecule’s structure, or terms for a molecule — the script selects two reviewers to review the deposit based

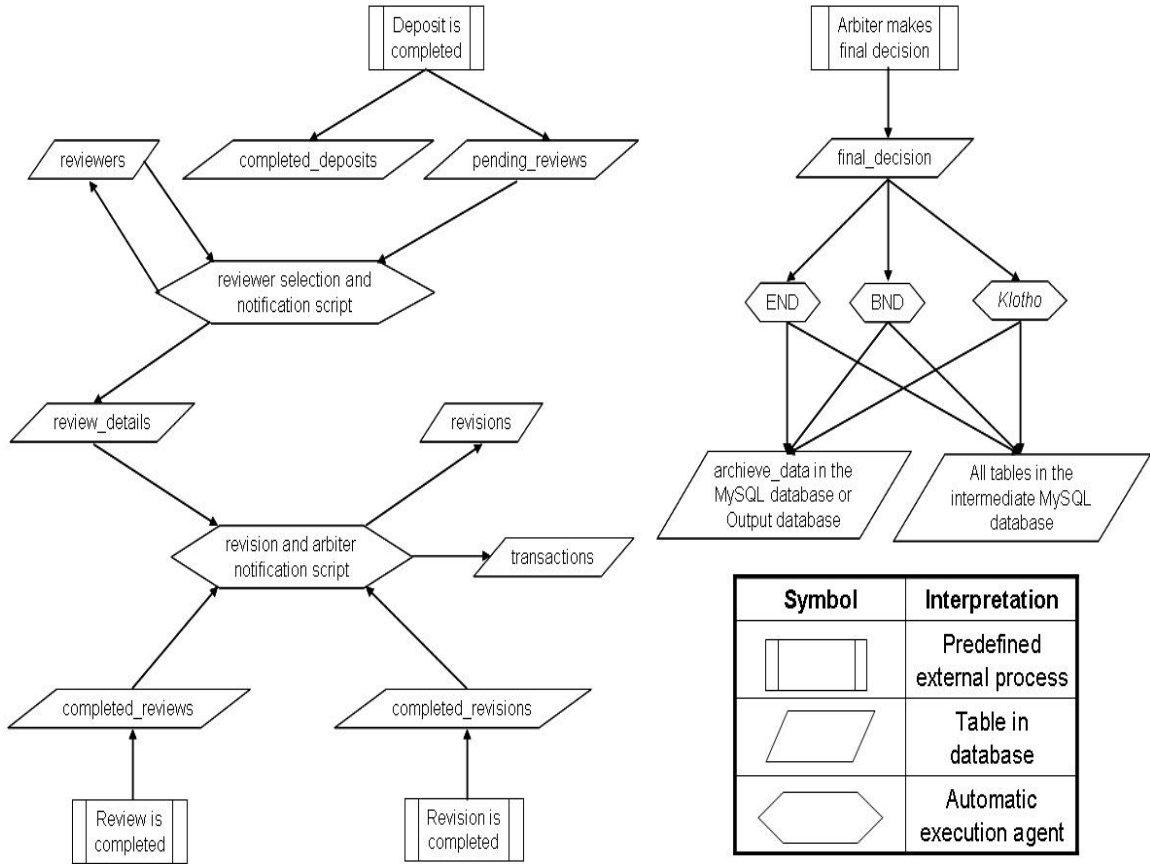


Figure 7: Database structure and operations. The predefined external processes are operations which are carried out in the “front end” of The *Agora*, and the tables are the various tables used in the intermediate MySQL database. The automatic execution agents are the various perl scripts running in the “back end”. An arrow pointing towards a table denotes that the data are being written to the table by the process from where it begins. An arrow pointing out from a table to a process means the data in the table are being used by the process. The three scripts for END, BND, and *Klotho* moves the final data from table “final_decision” to the output database if approved, or to a table “archive_data” in MySQL. At the same time, the scripts remove all entries from all tables in the intermediate database that are related to this final data.

on a set of criteria. These criteria include the reviewer's capability to review the type of data deposited, the number of deposits previously reviewed successfully, the number of deposits that are currently being reviewed by that reviewer, the number of reviews that have still not been completed by that reviewer, and the number of review requests that the reviewer turned down earlier. This assures that all reviewers get an equal opportunity to review deposits and distributes the work load equally to reviewers who are capable of reviewing deposits quickly and efficiently. In the current version of *The Agora*, we do not have any limit on the number of reviews that can be completed by a particular reviewer. We are trying to have better results and better quality of data in the database by having a small coordinated group of experts. However, if the need arises, we might limit the number of reviews that can be done by a single reviewer to not overburden a single individual. Once two reviewers are selected, an email is automatically sent to each of them notifying them of their selection and requesting confirmation that each will review the deposit. The selected reviewers go to the "front end" and log into their accounts and record their decision. Periodic reminders are sent to the selected reviewers if no response is received about their decision within a particular time frame. In the current implementation, I have a time frame of seven days in which the reviewers have to record their decision; failing which, three email reminders are sent to the reviewer on successive days, asking the reviewer to record his or her decision. Each reviewer sees a list of deposits he or she has been selected to review; the reviewer selects a deposit and either agrees to review it or refuses by clicking the appropriate button. A reviewer can defer the decision by not selecting the deposit. If one agrees to review a deposit, a new entry is entered in the reviewer's transaction list and the reviewer can start reviewing the data immediately. If the reviewer decides not to review the deposit, or if the reviewer does not respond by decision deadline even after receiving the reminders, it is assumed

that the reviewer will not review the deposit and the script immediately selects another reviewer for the job and repeats the process. This process continues until two reviewers have agreed to review the deposit. Since some depositors are also qualified reviewers or curators, special care is taken to prevent selecting the original depositor as a reviewer. If none of the eligible reviewers agree to review a deposit, the arbiter for the corresponding database is notified to review the data and either approve or reject them. Once the reviewers have agreed to review the data, periodic reminders are sent to them if their reviews are not received within the preset time frame. In the current implementation, I give reviewers a time period of two weeks to review the data and submit their comments and decision about the deposit. The script is run daily and an incrementing counter is used to check the age of each review. If the counter crosses a predefined limit, the script reminds the reviewer of the delay and requests the reviewer to submit the review. This is to ensure that the decision on the deposited data can be taken as quickly as possible.

3.4.3 Review and Revision

Once the reviewers agree to review the data, the review process takes place using the “front end”. The review and revision process, and the arbiter’s role, are illustrated in Figure 9. The reviewers check the deposited data for accuracy and consistency through various resources, including the references cited by the original depositor. The reviewers are able to review each element of the deposit and can either accept the deposited information if all the information provided is correct, submit a request for a revision by the depositor, make suggestions, or reject the deposit. Once the two reviewers have completed their reviews, their decisions with comments and reviews are stored in a temporary database table (“completed_reviews”). This table is regularly checked automatically and reviewers’ decisions compared. If the reviewers agree on

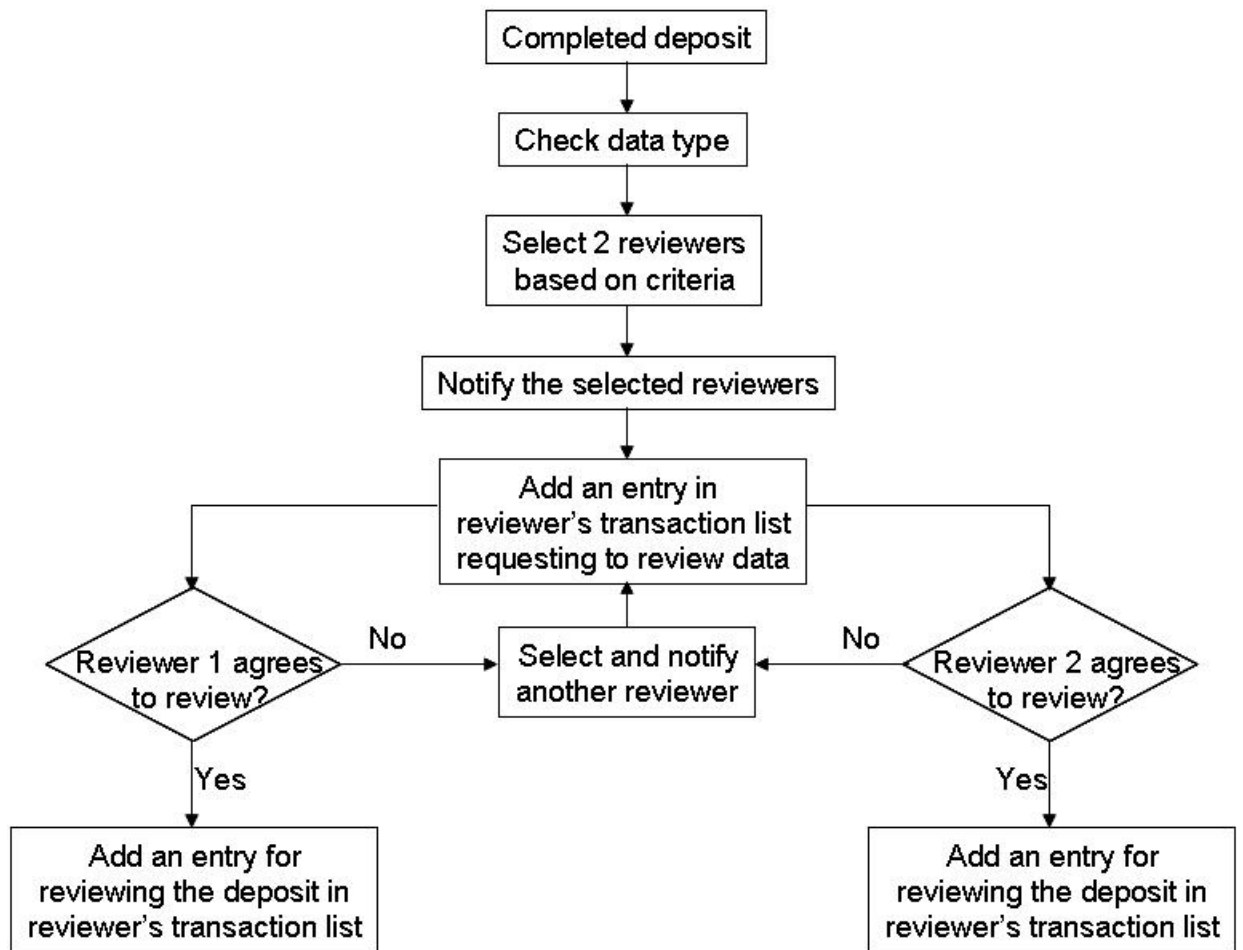


Figure 8: Reviewer selection process. Depending on the type of deposit, I select two reviewers, notify them about their selection, and add a transaction to their list of decisions. If the reviewer agrees, I add a new review process to their transaction list. If the reviewer refuses, I select another reviewer and repeat the process until two reviewers agree to review a particular deposit.

the deposit, the data are sent to the arbiter. However, if the reviewers disagree, their suggested changes or comments are sent to the depositor to make the necessary changes or provide more information. This process of revision takes place only once for each deposit, and is completely identical to the deposit interface initially used by the depositor. The forms used by the depositor initially to submit the information are loaded again, but pre-filled with the values. The depositor can then make the necessary changes, or provide additional information as suggested by the reviewers. Once the depositor has submitted the changes and the revision process is complete, the same reviewers review the new data. Once the reviewers have made their final decision on the revised deposit, the review process is complete and a decision about including the data is made. If the reviewers are still not satisfied with the information, the data are sent to the arbiter to make a final decision.

3.4.4 The Arbiter's Decision

Once the second round of review is completed, the arbiter decides the deposit's fate. The process is shown in Figure 9. A script checks the reviewers' decisions, sends an email to the arbiter of the database for which the data are meant, and a new entry is added to the arbiter's transactions. The transaction involves a side by side comparison of the data submitted by the original depositor, the changes made in the revision process, and the reviews and reviewers' comments. For any particular datum, the arbiter can select any of the values proposed by the depositor or the reviewers, or assign a completely different value. In the first case, if a reviewer suggests a different value, then his or her comments should indicate why. The second case is for situations where the arbiter is not satisfied by any of the proposed values for a datum. After the arbiter has reviewed all the fields and made any necessary corrections, he or she finally decides if the data are now acceptable and whether they can be included in

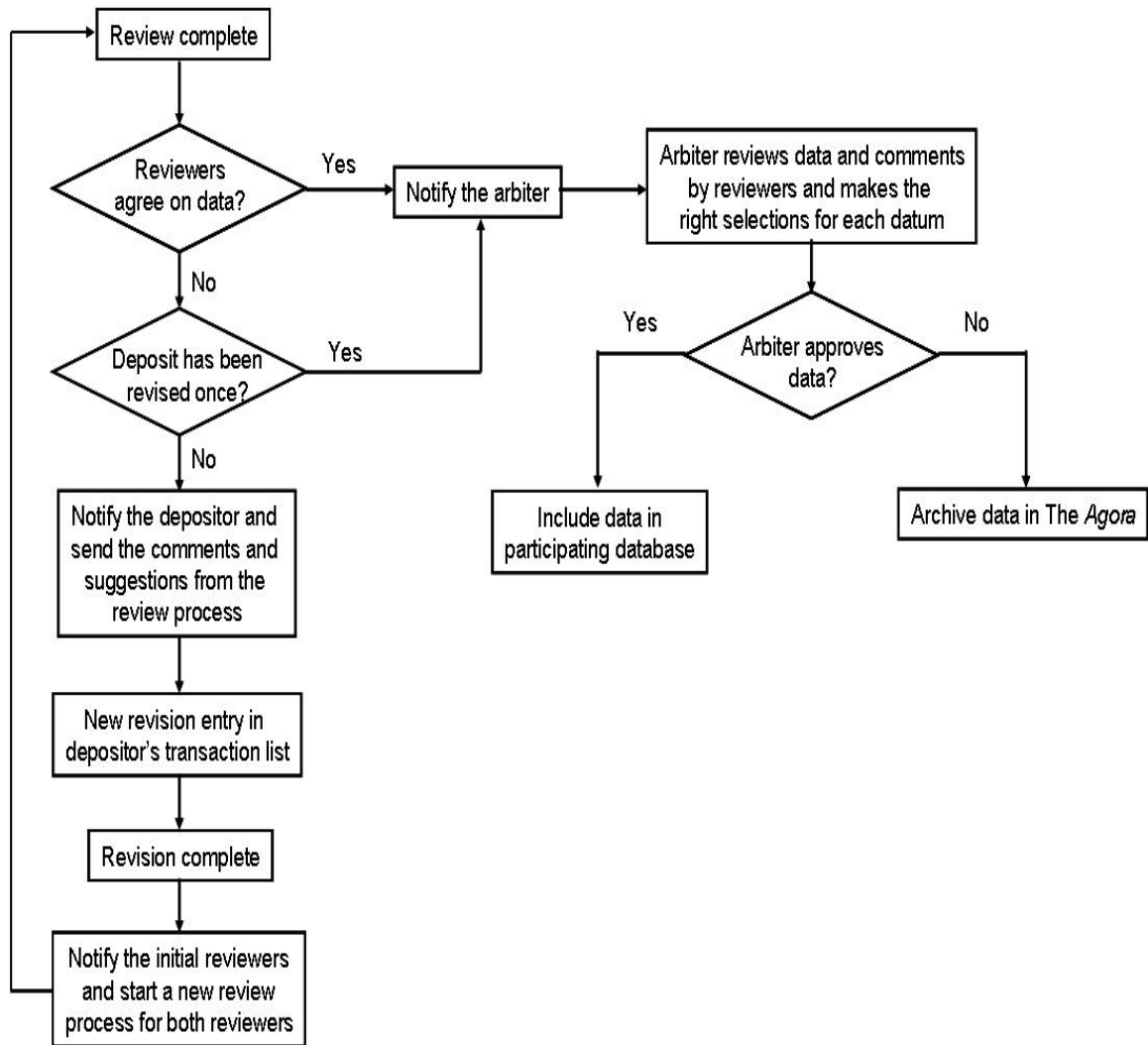


Figure 9: Review and revision process. Once the reviews are completed, the reviewers' decisions are compared. If they disagree, a new revision process is added to depositor's transaction list and the depositor is notified along with the comments and suggestions from the reviewers. Once the depositor completes the revision, the reviewers are notified, and then they examine the new information and decide if the revised data are acceptable. In either case, the arbiter is now informed. If the data are approved, they are included in the respective database or else archived in *The Agora*. Note that there is only one revision for each deposit and that occurs only when the reviewers are not satisfied by the information. If the data are accepted by the reviewers in the initial review, the data is sent directly to the arbiter for approval.

the database or not. If the data are approved by the arbiter, the data are processed for inclusion in the database. An email is sent to the depositor notifying him or her of the deposit's acceptance. If the arbiter rejects the data, they are archived in the intermediate MySQL database and the depositor is notified of the rejection. The most common reason for the data being rejected would be insufficient information available at the time of deposit. By keeping a copy of rejected deposits, it is possible for the curators to review rejected deposits as new information becomes available in the future; if satisfactory then, these deposits can be completed by the curators and included in the databases.

3.4.5 Data Inclusion

The procedure for including the data in the various databases is rather difficult and complicated as compared to the overall procedure. There are two main reasons for this. First, the domain model of the participating databases differs significantly from the model used to store the data in the intermediate databases. For example, a single datum in the intermediate database often produces multiple data in the destination database: *synonym_index* and *synonym_list* are derived from synonym information in term deposits. Also, tracking information for each datum is added to its respective database. Second, approved data must be checked to see if they are already included in the destination databases. A deposit reporting a reaction can re-use molecules, literature citations, and assays that are already recorded. A deposit reporting relationships among terms can add new relationships to existing terms. Thus, only some parts of a new deposit may warrant addition of entirely new facts.

The overall design for including accepted information in the database is shown in Figure 10. As can be seen, if the deposit is accepted, information from the MySQL table field is broken down into components and goes in their respective Prolog files

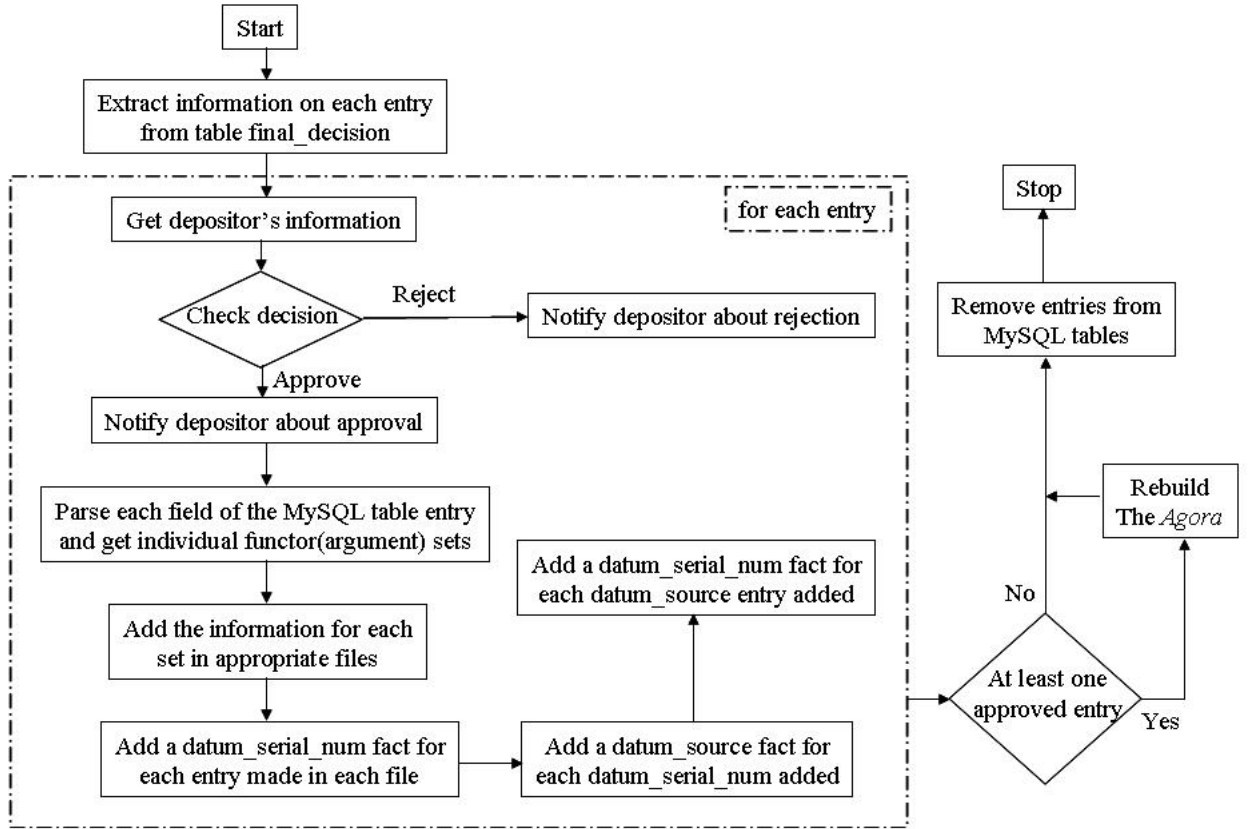


Figure 10: Including data in data files. If the deposit is approved by the arbiter, the information is extracted from the MySQL table and each datum is converted into one or more Prolog facts. Along with these facts, tracking information about each datum is added in the tracking information files. Once all the information has been added, the core application, *The Agora*, is rebuilt. If the data are rejected, I archive the information. In either case, the depositor is notified about the decision of the deposited data.

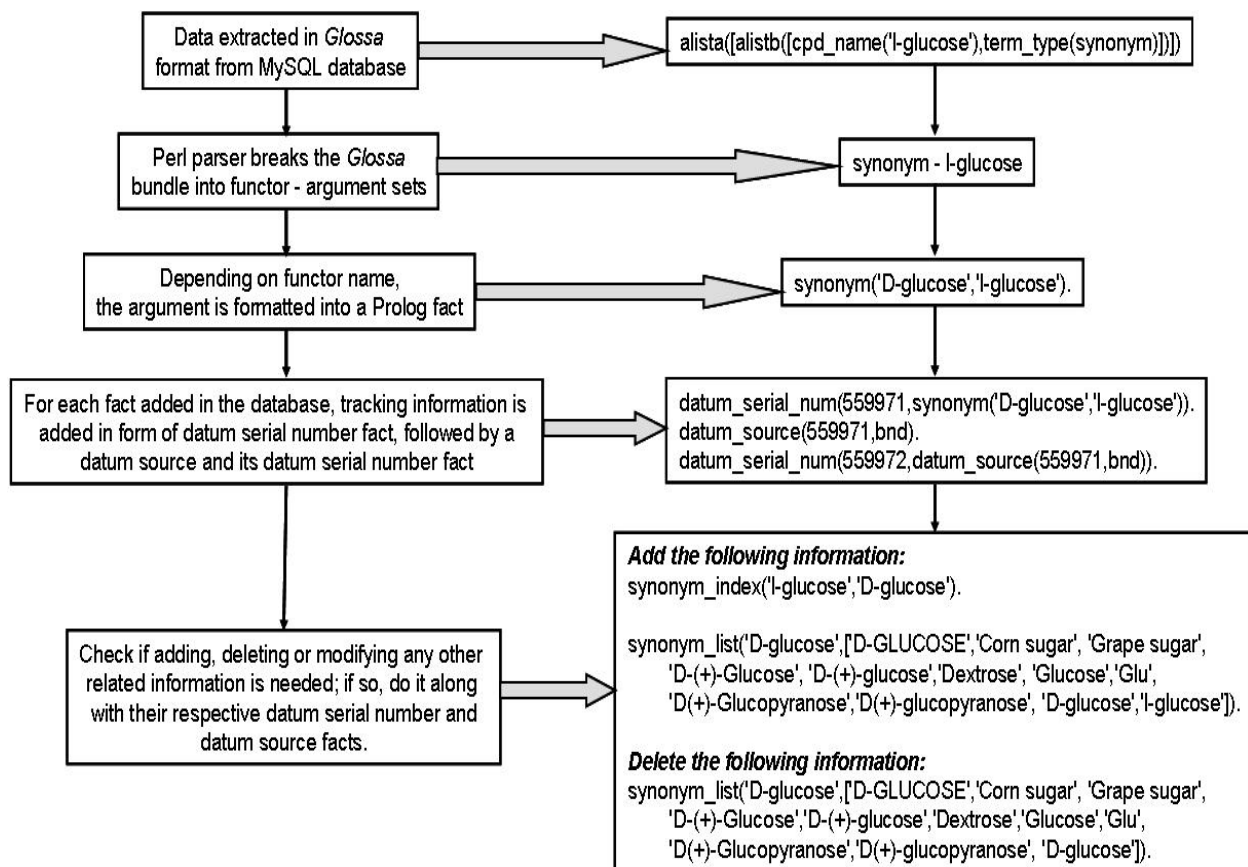


Figure 11: Parsing the *Glossa* data into Prolog facts. The data that exist in the MySQL database in *Glossa* format are broken down into their components and then each component is added to the database in their respective files. I also add the *datum_serial_num* and *datum_source* facts for each Prolog fact created using the parser for tracking purposes. For every deposit, the parser modifies any other related file and might add, delete, or update existing information depending on the type of data and its relationship with existing data.

depending on the type of information. For each addition, a *datum_serial_num* fact is added for identification and tracking purposes, and for the same reason, a *datum_source* fact is added for determining the database to which the information belongs, and a *datum_serial_num* entry for the *datum_source* fact just created. These facts provide very finely grained attribution and management of the data. A Perl parser extracts the bundled *Glossa* information from the MySQL database and converts it into Prolog facts for addition to the relevant files. An overview of this conversion process and a corresponding example are shown in Figure 11.

I shall now discuss the three types of data that are currently supported by The *Agora*. For each type of data, a deposit was made through the “front end”. After the review and revision process, the data were approved by the arbiter. In each type of deposit, I will show how the bundled *Glossa* data are broken down into small sets and then how each datum is entered in the corresponding database.

Molecular structure deposits Including data for molecular structures is easier than term deposits. A sample data entry for a molecular structure deposit is shown in Table 1. In structure deposits, the only essential information is the configuration rule for the molecular structure; the type of molecule it is; and the relevant references. Optional information includes the InChI description and comments on the structure, which go in their respective files. As soon as the depositor enters the configurational rule, *Klotho* checks the rule in accordance to the rules of chemical structure and automatically generates various other output formats used in the web site [4]. These files include the terminal file, Fischer file, and the SMILES string file. During the review process, the reviewers can look at the generated files for the structure of molecule as generated by *Klotho* and then comment. When the structure is approved, information is then extracted as shown in Table 2. An accession number is given to the molecule. The number begins with ‘KL’ and followed by either ‘M’, if it is the entire

structure of a molecule; ‘C’, if it is the general structure of that class of molecules; or ‘S’, if it is a structure of a substituent. The three character prefix is followed by a number which increases sequentially as molecules are added to the database. The files that were generated by *Klotho* are directly moved and used for generating the HTML page where all the information for the molecule is available. The configuration rule for the molecule goes into the Prolog database. The config rule is added to one or another data file. For example, depending on the type of molecule, the structure could be of a carbohydrate, and so the config rule is added to the sugars file. Various stages of parsing a molecule structure deposit are shown in Tables 1 - 3. Besides the references for the information, the most important information in these deposits are the rule for the structure; the type of molecule; and whether it is a structure of entire molecule, class of molecules, or a substituent. The last decides where the information for this deposit should be included if it is approved. These information is extracted from the deposited information and formatted to suit the database. The comments for the structure are also extracted and added to the relevant files.

Along with the additions and modifications to the Prolog database, the script also modifies certain local tracking Prolog facts. As discussed in Section 1.3 and Figure 5, these Prolog facts are used by the local *Klotho* team, who continuously add and review new molecular structures. These facts generate a weekly report which provides each of the depositors with a list of molecule structures that still need to be generated; molecules that are yet to be checked by some of the reviewers; and a list of molecules for which the structures have been approved and are ready to be included in the database. Every time a new structure is accepted, these tracking facts are modified to provide the latest information to everyone whenever the report is next generated.

Once the information is added to the database, a script is executed which automatically generates a new HTML page for the molecule just added and makes it

<i>field</i>	<i>value</i>
deposit_id	1984
deposit_type	mol
survey_path1	user_name(gaurav)::user_id(15)::session_id(1984):: type_deposit(mol)::check_cpd_struct(cpd_name('methanol'), inchi(''),macromol(no),struc_source(pub_ref))
survey_path2	user_name(gaurav)::user_id(15)::session_id(1984):: comm_msg(config(cpd_name('methanol'), cpd_specification([top(methyl),bottom(hydroxyl)]))): type_config_rule(entire)::alistb([type_mol(other),type_mol_text('alcohols')]): structure_org([reference_number_set([reference_number(281), reference_number(),reference_number(),reference_number(), reference_number()]))])
blocks_path	user_name(gaurav)::user_id(15)::session_id(1984):: comments('Methanol is produced from the distillation of wood and is a clear, colorless, volatile liquid with a weak odor that is somewhat sweeter than ethanol.')
final_path	release_permission(permission(approval), sub_journal(),sub_authors(),sub_title())

Table 1: Sample data for a molecular structure deposit in the intermediate MySQL database.

available on the web. This script uses the files that were generated by the *Klotho* during the deposit process and other information that was made available by the deposit.

For this structure deposit, since the structure was for the entire molecule (see Table 1), the accession number assigned to this structure had a prefix of 'KLM' and the complete accession number assigned to this molecular structure was 'KLM0000591'. Since the type of molecule was selected as 'other', the structure was added in the list of general molecular structures in 'general.pl'. The information that was entered in

<i>Type of Information</i>	<i>Information</i>
Molecule name:	methanol
INChi Description:	
Macromol?	no
Structure Source:	pub_ref
Config Rule:	[top(methyl),bottom(hydroxyl)]
Molecule Class:	entire
Molecule Type:	[type_mol(other),type_mol_text('alcohols')]
References:	[reference_number_set([reference_number(281), reference_number(),reference_number(), reference_number(),reference_number()])]
Comment:	Methanol is produced from the distillation of wood and is a clear, colorless, volatile liquid with a weak odor that is somewhat sweeter than ethanol.

Table 2: Data extracted for a molecular structure deposit using the parser.

the database for this deposit is shown in Table 3. In the file for general structures, I include the entire comment as deposited for reference purposes. So, the comment appears in the structure file as a comment just above the actual config rule for the structure and also in 'comments.pl' file for that accession number.

Term deposits A term deposit is more difficult than a molecular structure deposit, but much simpler when compared to a reaction deposit. Unlike a molecular structure deposit, a term deposit has many relationships among the terms, so setting up the correct relationships and updating the database are more complex. A sample entry for a term deposit is shown in Figure 4.

The data that are present in the intermediate MySQL database are in *Glossa* format and the information is bundled together in one large string in each of the fields of the database. A Perl parser was written that extracts the data from the

<i>Data file</i>	<i>Information Entered</i>
general.pl	% Methanol is produced from the distillation of wood and is a clear, colorless, volatile liquid with a weak odor that is somewhat sweeter than ethanol. config('methanol',[top(methyl),bottom(hydroxyl)]).
comments.pl	comment(KLM0000591,'Methanol is produced from the distillation of wood and is a clear, colorless, volatile liquid with a weak odor that is somewhat sweeter than ethanol.').

Table 3: Data entries for a molecular structure deposit. Note that the comment appears in the file in which the config rule is added. This is used for reference purposes. It is a single line beginning with a comment character, % in the case of Prolog. The line has been broken twice to display it here.

MySQL database. The data are checked to see if they were accepted by the arbiter for inclusion in BND. If not, the parsing script notifies the original depositor about the rejection and removes the entry from the intermediate MySQL database. These data are then archived in a separate table of the intermediate database for future consideration. If the data were accepted, the parser notifies the original depositor of the approval and then starts to extract the information for the BND. The parser parses each field from the database and separates the data into functor(argument) sets. The functor is the type of information related to the original deposit and the argument is its value. Depending on the functor type, the parser will store the corresponding value in the related files. For the sample data shown in Table 4, the parser would extract the information as shown in Table 5.

As can be seen from Table 5, all the information is broken down into smaller components and made much simpler. This information is then checked for type of information, like synonym, isomer, Markush term, *etc.*, and then formatted to suit the requirements of its data files. Also, for every new entry in any of the data files, tracking

<i>field</i>	<i>value</i>
deposit_id	2215
deposit_type	syn
survey_path1	user_name(gaurav)::user_id(15)::session_id(2215)::type_deposit(syn):: check_syn(cpd_name('D-glucose'),inchi(''),cpd_struc(yes))
survey_path2	user_name(gaurav)::user_id(15)::session_id(2215):: alist([alistb([cpd_name('L-glucose'), term_type(synonym),term_type(isomer)]), alistb([cpd_name('pyranose'),term_type(isomer)]), alistb([cpd_name('furanose'),term_type(isomer)]),num_terms(1)])
blocks_path	user_name(gaurav)::user_id(15)::session_id(2215):: isomer_survey([cpd_name('L-glucose'),struc_isomer(none), tautomer(none),stereo_isomer(configurational isomer), nonconfig_isomer(none),config_isomer(chiral), chiral(enantiomer)])::alistb([cpd_name('L-glucose'), comments('A primary source of energy for living organisms. It is naturally occurring and is found in fruits and other parts of plants in its free state. It is used therapeutically in fluid and nutrient replacement.')]):: isomer_survey([cpd_name('pyranose'),struc_isomer(none), tautomer(none),stereo_isomer(anomer),nonconfig_isomer(none), config_isomer(none),chiral(none)]):: alistb([cpd_name('pyranose'), comments('The cyclic form of a sugar with a six-membered ring.')]):: isomer_survey([cpd_name('furanose'),struc_isomer(none), tautomer(none),stereo_isomer(anomer),nonconfig_isomer(none), config_isomer(none),chiral(none)]):: alistb([cpd_name('furanose'), comments('A simple sugar containing the five-membered furn ring.')])
final_path	release_permission(permission(approval),sub_journal(), sub_authors(),sub_title())

Table 4: Sample data for a term deposit in the intermediate MySQL database.

<i>Term</i>	<i>Information</i>
Term Name: L-glucose	Term Type: synonym
Term Name: L-glucose	Term Type: isomer
Term Name: pyranose	Term Type: isomer
Term Name: furanose	Term Type: isomer
Term Name: L-glucose	Isomer Information: struc_isomer(none),tautomer(none), stereo_isomer(configurational isomer), nonconfig_isomer(none),config_isomer(chiral), chiral(enantiomer)
Term Name: pyranose	Isomer Information: struc_isomer(none),tautomer(none), stereo_isomer(anomer),nonconfig_isomer(none), config_isomer(none),chiral(none)
Term Name: furanose	Isomer Information: struc_isomer(none),tautomer(none), stereo_isomer(anomer),nonconfig_isomer(none), config_isomer(none),chiral(none)
Term Name: L-glucose	Comments: A primary source of energy for living organisms. It is naturally occurring and is found in fruits and other parts of plants in its free state. It is used therapeutically in fluid and nutrient replacement.
Term Name: pyranose	Comments: The cyclic form of a sugar with a 6 membered ring.
Term Name: furanose	Comments: A simple sugar containing the five-membered furan ring.

Table 5: Data extracted for a term deposit using the parser.

information is added for future use or for back references. Under certain conditions, further checks are made on the information to decide where exactly to insert it. Figure 12 illustrates the problem of placing data so that only the most specific relationship is written, a particular problem with information about isomer relationships. If the term is an isomer term for the primary term, then the information goes into the lowest leaf node of isomer tree as classified by BND [11]. In the sample term deposit, the term L-glucose was deposited as a (structural) configurational isomer, an enantiomer of D-glucose possessing a chiral center property. As *per* the classification of isomers, as one goes down the tree, enantiomer is the lowest leaf node in the classification tree and so the term L-glucose will be entered only in the data related to enantiomers and not in any of the other classes. However, if a term is an isomer that follows two different paths of the isomer classification tree, then the term will be entered in the leaf nodes of each of the paths.

Once the data are separated and cleaned, they need to be added to the files to which they belong. For the sample term deposit from Table 4, the data that are added to the various files are shown in Table 6.

For entering the information in the Prolog database, the information is formatted and textual information added for inclusion in the files. In certain cases, some information needs to be removed from the existing database as well. If the term entered is a synonym, then the corresponding *synonym_list* predicate is modified. If the entry were modified without modifying its tracking information, the tracking information would be of no use for the modified data. If I were to make new additions to the tracking data without retiring old tracking information, then I will have multiple data confusingly leading to a single entry, and one would not be able to identify which information refers to what kind of update to the entry. So I delete the old entry in the tracking data and generate new tracking information every time a datum is created

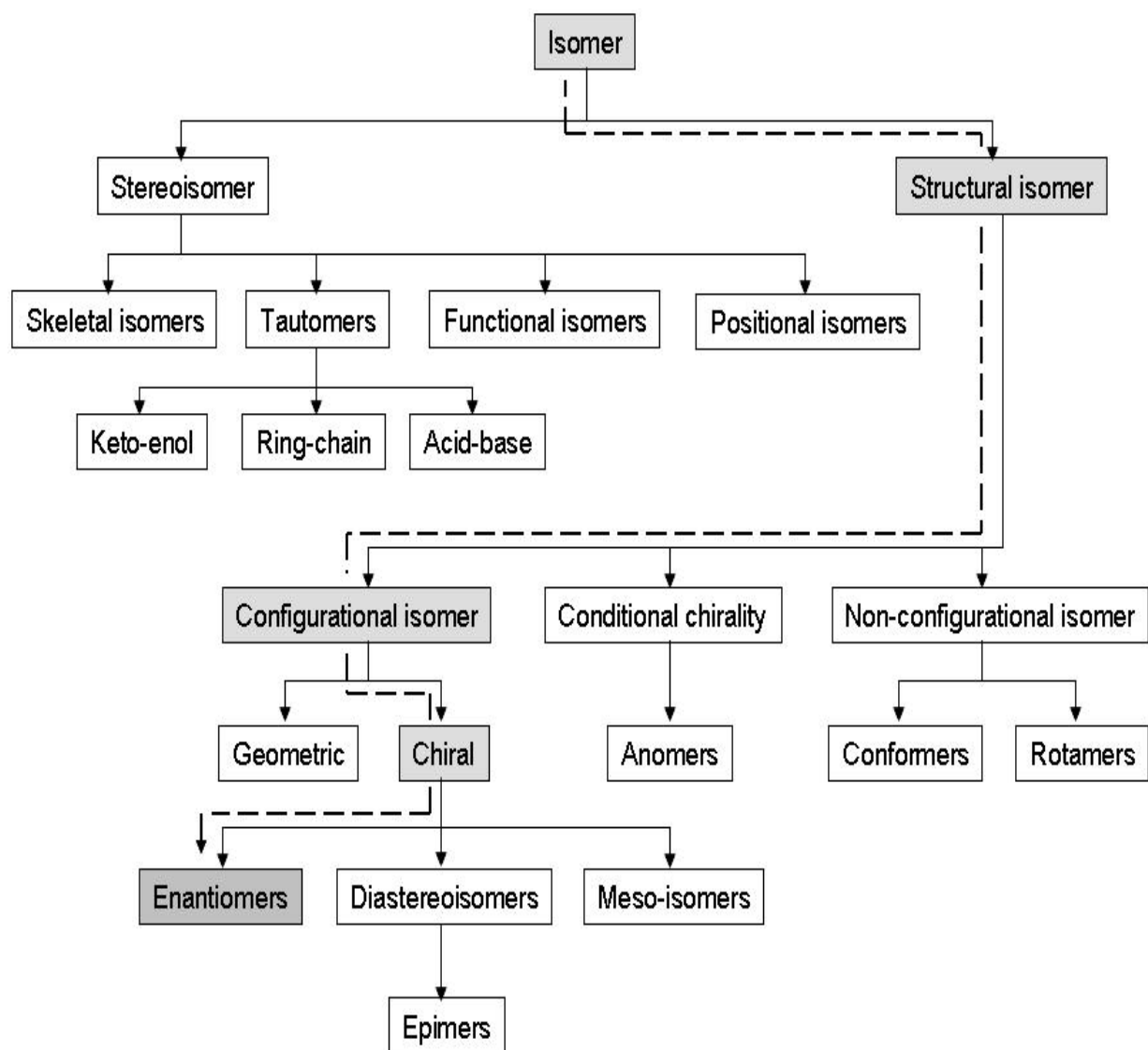


Figure 12: Updating isomer files. The isomer classification of BND is shown above. The code is written so that the isomer term is entered only in the leaf node of the isomer tree. A term which is a structural isomer, a configurational isomer, and an enantiomer possessing a chiral center will end up only in enantiomers and not in any of the parent nodes.

Term: **L-glucose**

synonym.pl	synonym('D-glucose','L-glucose').
datum_serial_number.pl	datum_serial_num(559971,synonym('D-glucose','L-glucose')).
datum_source.pl	datum_source(559971,bnd).
datum_serial_number.pl	datum_serial_num(559972,datum_source(559971,bnd)).
synonym_index.pl	synonym_index('L-glucose','D-glucose').
datum_serial_number.pl	datum_serial_num(559973,synonym_index('L-glucose','D-glucose')).
datum_source.pl	datum_source(559973,bnd).
datum_serial_number.pl	datum_serial_num(559974,datum_source(559973,bnd)).
synonym_list.pl	synonym_list('D-glucose',['D-GLUCOSE','Corn sugar','Grape sugar','D-(+)-Glucose','D-(+)-glucose','Dextrose','Glucose','Glu','D(+)-Glucopyranose','D(+)-glucopyranose','D-glucose','L-glucose']).
datum_serial_number.pl	datum_serial_num(559975,synonym_list('D-glucose',['D-GLUCOSE','Corn sugar','Grape sugar','D-(+)-Glucose','D-(+)-glucose','Dextrose','Glucose','Glu','D(+)-Glucopyranose','D(+)-glucopyranose','D-glucose','L-glucose'])).
datum_source.pl	datum_source(559975,bnd).
datum_serial_number.pl	datum_serial_num(559976,datum_source(559975,bnd)).
enantiomer.pl	enantiomer('D-glucose','L-glucose').
datum_serial_number.pl	datum_serial_num(559979,enantiomer('D-glucose','L-glucose')).
datum_source.pl	datum_source(559979,bnd).
datum_serial_number.pl	datum_serial_num(559980,datum_source(559979,bnd)).
comments.pl	comment('L-glucose','A primary source of energy for living organisms. It is naturally occurring and is found in fruits and other parts of plants in its free state. It is used therapeutically in fluid and nutrient replacement.').
datum_serial_number.pl	datum_serial_num(559987,comment('L-glucose','A primary source of energy for living organisms. It is naturally occurring and is found in fruits and other parts of plants in its free state. It is used therapeutically in fluid and nutrient replacement.')).
datum_source.pl	datum_source(559987,bnd).
datum_serial_number.pl	datum_serial_num(559988,datum_source(559987,bnd)).

<i>Data file</i>	<i>Information Entered</i>
Term: pyranose	
anomer.pl	anomer('D-glucose','pyranose').
datum_serial_number.pl	datum_serial_num(559977,anomer('D-glucose','pyranose')).
datum_source.pl	datum_source(559977,bnd).
datum_serial_number.pl	datum_serial_num(559978,datum_source(559977,bnd)).
comments.pl	comment('pyranose','The cyclic form of a sugar with a six-membered ring.').
datum_serial_number.pl	datum_serial_num(559983,comment('pyranose','The cyclic form of a sugar with a six-membered ring.')).
datum_source.pl	datum_source(559983,bnd).
datum_serial_number.pl	datum_serial_num(559984,datum_source(559983,bnd)).
Term: furanose	
anomer.pl	anomer('D-glucose','furanose').
datum_serial_number.pl	datum_serial_num(559981,anomer('D-glucose','furanose')).
datum_source.pl	datum_source(559981,bnd).
datum_serial_number.pl	datum_serial_num(559982,datum_source(559983,bnd)).
comments.pl	comment('furanose','A simple sugar containing the five-membered furan ring.').
datum_serial_number.pl	datum_serial_num(559985,comment('furanose','A simple sugar containing the five-membered furn ring.')).
datum_source.pl	datum_source(559985,bnd).
datum_serial_number.pl	datum_serial_num(559986,datum_source(559985,bnd)).

Table 6: Data entries for a term deposit.

Term: **L-glucose**

synonym_list.pl	synonym_list('D-glucose', ['D-GLUCOSE', 'Corn sugar', 'Grape sugar', 'D-(+)-Glucose', 'D-(+)-glucose', 'Dextrose', 'Glucose', 'Glu', 'D(+)-Glucopyranose', 'D(+)-glucopyranose', 'D-glucose']).
datum_serial_number.pl	datum_serial_num(123878, synonym_list('D-glucose', ['D-GLUCOSE', 'Corn sugar', 'Grape sugar', 'D-(+)-Glucose', 'D-(+)-glucose', 'Dextrose', 'Glucose', 'Glu', 'D(+)-Glucopyranose', 'D(+)-glucopyranose', 'D-glucose'])).
datum_source.pl	datum_source(123878, bnd).
datum_serial_number.pl	datum_serial_num(151672, datum_source(123878, bnd)).

Table 7: Data deleted for a term deposit.

or modified. As a result, I have only one tracking datum for each other datum. The reason for deleting the old information is that the data were already accepted and hence have been properly incorporated and running in our database. The main reason for tracking this information for a new datum being added or modified is to roll back to a safer state if the database becomes unstable for any reason. Entries in the database that were removed are shown in Table 7.

Reaction deposits Deposits for enzymatic reactions are parsed, checked, and added to END very much as described in Sections 3.4.5 and 3.4.5 for *Klotho* and BND, respectively. The only important differences are that the volume and complexity of the relationships among the information are much greater than for the other two databases. The number of files that need to be modified and updated for the even simplest and smallest enzymatic reaction is another factor which makes the deposit of enzymatic reactions the most difficult deposit to handle in this automated workflow. A sample deposit, the information extracted, and the information that would be entered into END are shown in Tables 8 - 10, respectively. The same pro-

cess will be used for other types of reactions by inserting an initial filter to determine to which reaction database the deposit should be assigned.

3.4.6 Rebuilding New Databases

Once the data are added to the database, the entire core application program serving the community, The *Agora*, is rebuilt to provide the latest information to the users. If the deposit is a new molecule, *Moirai* is rebuilt followed by The *Agora*.

3.4.7 Adaptive Workflow Model

After discussing the different types of data deposits and their management process, it must be very clear that the three databases use the same workflow model for the curation and data management process. This workflow model was designed and implemented at various levels of progress for the three databases. In *Klotho*, there was already a fair amount of automation and existing data, as well as some data management by Prolog. So, to make the workflow adapt to the existing design and smoothly integrate it into the existing data management was a prime factor in the design process. For BND, the case was exactly opposite. There was absolutely no data management or tracking processes for it, and so the design was made from scratch. For END, there was a huge amount of data in the database, but it was only partially tracked and absolutely no automation in the existing design. Same was the case for non-enzymatic reactions as well. Although these databases were far apart from each other, in terms of data size, data management, and tracking abilities, it was still possible to consolidate the three workflows into a single automated workflow model. Similarly, it is possible to implement the same model to adapt to other existing databases with minimal adjustments depending on the type of data and any existing workflow that the database might have.

<i>field</i>	<i>value</i>
deposit_id	2181
deposit_type	rxn
survey_path1	user_name(gaurav)::user_id(15)::session_id(2181):: type_deposit(rxn)::horizontal_rxn_eqn([sinistras ([reactant(stoich(1),cpd_name('adenine')), reactant(stoich(1),cpd_name('5-phospho-alpha-D-ribose 1-diphosphate'))]), dextras([reactant(stoich(1),cpd_name('adenosine 5'-monophosphate')), reactant(stoich(1),cpd_name('pyrophosphate'))]), more(no),leftct(4),rightct(4)]::rxn_name('APRT'):: rxn_ref([reference_number_set([reference_number(278), reference_number(),reference_number(),reference_number(), reference_number()]))::rxn_catalyzed(yes):: rxn_transport(no)::rxn_kinetics(yes)::mech_info_q(yes):: rxn_in_activator(no)::rxn_has_macromol(yes):: rxn_novel_macromol(iunk)::assay_survey([reference_number_set([reference_number(278), reference_number(),reference_number(), reference_number(),reference_number()]), qual_assay(other),qual_assay_text('thermal denaturation assay'), substrate([cpd_name('PRPP'),concentration(conc_value(10), conc_unit(mM)),pH(),cpd_name('Guanine'),concentration(conc_value(10), conc_unit(mM)),pH(),cpd_name(''),concentration(conc_value(), conc_unit(M)),pH(),cpd_name(''),concentration(conc_value(), conc_unit(M)),pH()]),buffer([cpd_name('HCl'),concentration(conc_value(40),conc_unit(uM)),pH(8.0), cpd_name('KCl'),concentration(conc_value(2), conc_unit(uM)),pH(),cpd_name('MgCl2'),concentration(conc_value(5), conc_unit(uM)),pH(),cpd_name('trypsin inhibitor'), concentration(conc_value(100),conc_unit(ug/ml)),pH(), cpd_name(''),concentration(conc_value(),conc_unit(M)),pH(),pH(8.0)], temp(70celsius),stopping_conditions(AMP, PPi), measurement_technique(HPLC analysis),coupled_assay(iunk), essential_mols([cpd_name(''),cpd_name(''),cpd_name(''),cpd_name('')])):: org_survey([reference_number_set([reference_number(278), reference_number(),reference_number(),reference_number(), reference_number()]),

```

org_survey([reference_number_set([reference_number(278),
reference_number(),reference_number(),reference_number(),
reference_number()]),
org_gp(group),other_orgs_q(yes),
genus(Homo),species(sapiens),strain(),approx_org_name())::
phy_survey([reference_number_set([reference_number(278),
reference_number(),reference_number(),reference_number(),
reference_number()]),nominal_pathway_name('Nucleotide metabolism'),
stimulus_sen(yes),disease('Urolithiasis')])

survey_path2 user_name(gaurav)::user_id(15)::session_id(2181)::
rxn_eqn(sinistras([reactant(stoich(1),cpd_name('adenine'),
state('free'),state_text(''),datum_xref(compartment('cytoplasm'),
compartment_text(''),accession(''),rlted_db()),macromol(no)),
reactant(stoich(1),cpd_name('5-phospho-alpha-D-ribose 1-diphosphate'),
state('free'),state_text(''),datum_xref(compartment('cytoplasm'),
compartment_text(''),accession(''),rlted_db()),macromol(no))]),
dextras([reactant(stoich(1),cpd_name('adenosine 5'-monophosphate'),
state('free'),state_text(''),datum_xref(compartment('cytoplasm'),
compartment_text(''),accession(''),rlted_db()),macromol(no)),
reactant(stoich(1),cpd_name('pyrophosphate'),state('free'),
state_text(''),datum_xref(compartment('cytoplasm'),
compartment_text(''),accession(''),rlted_db()),macromol(no))]))::
catalyst_survey([reference_number_set([reference_number(278),
reference_number(),reference_number(),reference_number(),
reference_number()]),is_novel_mol(cpd_name('APRT'),
novel_mol(no)),ec_num(ec_class(),ec_subclass(),
ec_subsubclass(),ec_serial_num()),ec_num(ec_class(),
ec_subclass(),ec_subsubclass(),ec_serial_num()),
sys_name(cpd_name('')),synds([cpd_name('AMP pyrophosphorylase'),
cpd_name('AMP diphosphorylase'),cpd_name('Transphosphoribosidase'),
cpd_name('APRT'),cpd_name('')]),phys_char_q(iunk),
rxn_component(),alistb([cpd_name('Adenine'),stoich()]),
alistb([cpd_name('Phosphate'),stoich()]),
alistb([cpd_name('Ribose'),stoich()]),

```

field value

```
alistb([cpd_name(''),stoich()]),alistb([cpd_name(''),stoich()]]),
rxn_subunit(alistb([cpd_name(''),stoich()]),
alistb([cpd_name(''),stoich()]),alistb([cpd_name(''),stoich()]),
alistb([cpd_name(''),stoich()]),alistb([cpd_name(''),stoich()]]),
is_cloned(iunk),cata_origin(iunk),cofactor_reqd(iunk),
tunable_catalyst(yes),alt_cat_rxns_q(iunk),alt_rxns_q(no),
alt_forms_mol_q(iunk),other_substrate(cpd_name('Hypoxanthine'),
cpd_name('Guanine'),cpd_name(''),cpd_name(''),
cpd_name(''),cpd_name(''))::gp_org_survey([num_orgs(1)])::
other_org_gp_survey([num_orgs(1),num_orgs(0)])::
stimulus_survey([reference_number_set([reference_number(278),
reference_number(),reference_number(),
reference_number(),reference_number()]),stimulus_name(''),
stimulus_src(genotypic),concentration(conc_value(),conc_unit(M)),
stimulus_resp(increased),stimulus_amount(amount(),
amount_unit(-15)),stimulus_mech_known(unknown),
stimulus_name(''),stimulus_src(genotypic),concentration(
conc_value(),conc_unit(M)),stimulus_resp(increased),
stimulus_amount(amount(),amount_unit(-15)),
stimulus_mech_known(unknown),stimulus_name(''),
stimulus_src(genotypic),concentration(conc_value(),conc_unit(M)),
stimulus_resp(increased),stimulus_amount(amount(),
amount_unit(-15)),stimulus_mech_known(unknown),
stimulus_name(''),stimulus_src(genotypic),
concentration(conc_value(),conc_unit(M)),stimulus_resp(increased),
stimulus_amount(amount(),amount_unit(-15)),
stimulus_mech_known(unknown),stimulus_name(''),
stimulus_src(genotypic),concentration(conc_value(),conc_unit(M)),
stimulus_resp(increased),stimulus_amount(amount(),
amount_unit(-15)),stimulus_mech_known(unknown)])
```

<i>field</i>	<i>value</i>
blocks_path	user_name(gaurav)::user_id(15)::session_id(2181):: gp_org_survey([alistb([org_name('Homo sapiens'), genetic_background(),markers([marker(),marker(), marker(),marker())],organ('liver'),tissue('liver'), cell_type('erythrocytes'),cell_line('')])):: other_orgs_survey([alistb([org_name('Mice'), genetic_background(),markers([marker(),marker(), marker(),marker())],organ('liver'),tissue('liver'), cell_type('erythrocytes'),cell_line('')])):: comments('This reaction seems to be the only mechanism through which free adenine is incorporated into its corresponding nucleotide in humans.')
final_path	release_permission(permission(approval), sub_journal(),sub_authors(),sub_title())
decision	Y

Table 8: Sample data for an enzymatic reaction deposit.

<i>Type of Information</i>	<i>Information</i>
Reaction equation:	adenine + 5-phospho-alpha-D-ribose 1-diphosphate = pyrophosphate + adenosine 5'-monophosphate
Reaction name:	APRT
Reaction reference:	Local reference number: 278
Reactant information:	
adenine:	stoich(1),state('free'),compartment('cytoplasm')
5-phospho-alpha-D-ribose 1-diphosphate:	stoich(1),state('free'),compartment('cytoplasm')
adenosine 5'-monophosphate:	stoich(1),state('free'),compartment('cytoplasm')
pyrophosphate:	stoich(1),state('free'),compartment('cytoplasm')
Catalyst:	APRT
Catalyst reference:	Local reference number: 278
Catalyst synonyms:	'AMP pyrophosphorylase','AMP diphosphorylase', 'Transphosphoribosidase'
Catalyst details:	rxn_component(cpd_name('Adenine'),cpd_name('Ribose'), cpd_name('Phosphate')),tunable_catalyst(yes), other_substrate(cpd_name('Hypoxanthine'), cpd_name('Guanine'))
Assay name:	thermal denaturation
Assay reference:	Local reference number: 278
Assay information:	substrate('PRPP',millimolar(10),pH(unsaid)), substrate('Guanine',millimolar(10),pH(unsaid)), buffer_component('trypsin inhibitor',ug_per_mol(100), pH(unsaid)), buffer_component('KCl',micromolar(2),pH(unsaid)), buffer_component('MgCl2',micromolar(5),pH(unsaid)), buffer_component('HCl',micromolar(40),pH(8.0)), pH(8.0),temperature(celsius(70)), stopping_conditions('AMP, PPI'), msemt_tech('HPLC analysis')
Pathway name:	Nucleotide metabolism
Disease:	Urolithiasis
Organism:	genus(Homo),species(sapiens),organ('liver') tissue('liver'),cell_type('erythrocytes')
Comments:	This reaction seems to be the only mechanism through which free adenine is incorporated into its corresponding nucleotide in humans.

Table 9: Data extracted for an enzymatic reaction deposit using parser.

<i>Data file</i>	<i>Information Entered</i>
agora_transaction.pl	agora_transaction(2181,1114368491,deposit,end).
citation.pl	citation(crespillo2003).
assay.pl	assay(561732,'thermal denaturation', [substrate('PRPP', millimolar(10),pH(unsaid)), substrate('Guanine',millimolar(10),pH(unsaid))], [buffer([buffer_component('trypsin inhibitor', ug_per_mol(100),pH(unsaid)), buffer_component('KCl',micromolar(2),pH(unsaid)), buffer_component('MgCl2',micromolar(5),pH(unsaid)), buffer_component('HCl',micromolar(40),pH(8.0)), pH(8.0)]),temperature(celsius(70)), stopping_conditions('AMP, PPi')], msemt_tech('HPLC analysis'),unknown,essential_mols([])).
rxn_assay.pl	reaction_assay(561699,561732).
catalyst.pl	catalyst('APRT',561699,[],[]).
polypeptide.pl	polypeptide('APRT').
catalyst_origin.pl	catalyst_origin('APRT',561699,[cloned(iunk),origin(iunk)]).
dextra.pl	dextra('adenosine 5'-monophosphate',561699,1,[free],[cytoplasm]). dextra('pyrophosphate',561699,1,[free],[cytoplasm]).
sinistra.pl	sinistra('adenine',561699,1,[free],[cytoplasm]). sinistra('5-phospho-alpha-D-ribose 1-diphosphate',561699,1, [free],[cytoplasm]).
disease.pl	disease(561755,['Urolithiasis']).
organism.pl	organism('Homo','sapiens',['']).
rxn_location.pl	rxn_location(561699,[organism('Homo','sapiens','nil','nil'), tissue('liver'),cell_type('erythrocytes'),cell_line(''),organ('liver')]).
event.pl	event(561699,created,1114368491,nil,reaction(561699,'APRT',[]), agora_transaction(2181),'APRT').
comment.pl	comment('This reaction seems to be the only mechanism through which free adenine is incorporated into its corresponding nucleotide in humans.').
reference.pl	reference(crespillo2003,article,[author('Javier Crespillo, Pilar Llorente, Luisa Argomaniz, Celia Montero'),year(2003), article_title('APRT from erythrocytes of HGPRT deficient patients: Kinetic, regulatory and thermostability properties'), journal_title('Molecular and Cellular Biochemistry'), volume(254),start_page(359),end_page(363)]).
rxn_rubric.pl	rxn_rubric(561699,'APRT').
reaction.pl	reaction(561699,'APRT',[]). reaction(561755,'Nucleotide metabolism',[561699]).

<i>Data file</i>	<i>Information entered</i>
cpd_name.pl	cpd_name('adenine'). cpd_name('5-phospho-alpha-D-ribose 1-diphosphate'). cpd_name('adenosine 5"-monophosphate'). cpd_name('pyrophosphate'). cpd_name('APRT').
cpd_index.pl	cpd_index(561699,'adenine'). cpd_index(561699,'5-phospho-alpha-D-ribose 1-diphosphate'). cpd_index(561699,'adenosine 5'-monophosphate'). cpd_index(561699,'pyrophosphate'). cpd_index(561699,'APRT').
evidence.pl	evidence(561699,citation(crespillo2003),agora_transaction(2181)). evidence(561706,citation(crespillo2003),agora_transaction(2181)). evidence(561732,citation(crespillo2003),agora_transaction(2181)). evidence(561735,citation(crespillo2003),agora_transaction(2181)). evidence(561756,citation(crespillo2003),agora_transaction(2181)). evidence(561760,citation(crespillo2003),agora_transaction(2181)).
datum_serial_num.pl	datum_serial_num(561700,citation(crespillo2003)). datum_serial_num(561701,datum_source(561700,end)). datum_serial_num(561702,evidence(561699,citation(crespillo2003), agora_transaction(2181))). datum_serial_num(561703,datum_source(561702,end)). datum_serial_num(561704,reference(crespillo2003,article, [author('Javier Crespillo, Pilar Llorente,Luisa Argomaniz, Celia Montero'),year(2003),article_title('APRT from erythrocytes of HGPRT deficient patients: Kinetic, regulatory and thermostability properties'),journal_title('Molecular and Cellular Biochemistry'),volume(254),start_page(359), end_page(363)])). datum_serial_num(561705,datum_source(561704,end)). datum_serial_num(561706,rxn_rubric(561699,'APRT')). datum_serial_num(561707,datum_source(561706,end)). datum_serial_num(561708,evidence(561706, citation(crespillo2003),agora_transaction(2181))). datum_serial_num(561709,datum_source(561708,end)). datum_serial_num(561710,reaction(561699,'APRT',[])). datum_serial_num(561711,datum_source(561710,end)). datum_serial_num(561712,cpd_name('adenine')). datum_serial_num(561713,datum_source(561712,end)). datum_serial_num(561714, cpd_name('5-phospho-alpha-D-ribose 1-diphosphate')). datum_serial_num(561715,datum_source(561714,end)).

<i>Data file</i>	<i>Information entered</i>
datum_serial_num.pl	<p>datum_serial_num(561716, cpd_name('adenosine 5'-monophosphate')). datum_serial_num(561717,datum_source(561716,end)). datum_serial_num(561718,cpd_name('pyrophosphate')). datum_serial_num(561719,datum_source(561718,end)). datum_serial_num(561720,cpd_name('APRT')). datum_serial_num(561721,datum_source(561720,end)). datum_serial_num(561722,cpd_index(561699,'adenine')). datum_serial_num(561723,datum_source(561722,end)). datum_serial_num(561724, cpd_index(561699,'5-phospho-alpha-D-ribose 1-diphosphate')). datum_serial_num(561725,datum_source(561724,end)). datum_serial_num(561726, cpd_index(561699,'adenosine 5'-monophosphate')). datum_serial_num(561727,datum_source(561726,end)). datum_serial_num(561728,cpd_index(561699,'pyrophosphate')). datum_serial_num(561729,datum_source(561728,end)). datum_serial_num(561730,cpd_index(561699,'APRT')). datum_serial_num(561731,datum_source(561730,end)). datum_serial_num(561733,assay(561732, 'thermal denaturaion',[substrate('PRPP',millimolar(10), pH(unsaid)),substrate('Guanine',millimolar(10),pH(unsaid))], [buffer([buffer_component('trypsin inhibitor', ug_per_mol(100),pH(unsaid)),buffer_component('KCl', micromolar(2),pH(unsaid)),buffer_component('MgCl2', micromolar(5),pH(unsaid)),buffer_component('HCl', micromolar(40),pH(8.0)),pH(8.0)]), temperature(celsius(70)),stopping_conditions('AMP, PPI')), msemt_tech('HPLC analysis'),unknown,essential_mols([])). datum_serial_num(561734,datum_source(561733,end)). datum_serial_num(561735,reaction_assay(561699,561732)). datum_serial_num(561736,datum_source(561735,end)). datum_serial_num(561737,evidence(561732, citation(crespillo2003),agora_transaction(2181))). datum_serial_num(561738,datum_source(561737,end)). datum_serial_num(561739,evidence(561735, citation(crespillo2003),agora_transaction(2181))). datum_serial_num(561740,datum_source(561739,end)). datum_serial_num(561741, dextra('adenosine 5'-monophosphate',561699,1,[free],[cytoplasm])). datum_serial_num(561742,datum_source(561741,end)).</p>

<i>Data file</i>	<i>Information entered</i>
datum_serial_num.pl	<pre> datum_serial_num(561743, sinistra('adenine',561699,1,[free],[cytoplasm])). datum_serial_num(561744,datum_source(561743,end)). datum_serial_num(561745, dextra('pyrophosphate',561699,1,[free],[cytoplasm])). datum_serial_num(561746,datum_source(561745,end)). datum_serial_num(561747,sinistra('5-phospho-alpha-D-ribose 1-diphosphate', 561699,1,[free],[cytoplasm])). datum_serial_num(561748,datum_source(561747,end)). datum_serial_num(561749,catalyst('APRT',561699,[],[])). datum_serial_num(561750,datum_source(561749,end)). datum_serial_num(561751,peptide('APRT')). datum_serial_num(561752,datum_source(561751,end)). datum_serial_num(561753, catalyst_origin('APRT',561699,[cloned(iunk),origin(iunk)])). datum_serial_num(561754,datum_source(561753,end)). datum_serial_num(561756, reaction(561755,'Nucleotide metabolism',[561699])). datum_serial_num(561757,datum_source(561756,end)). datum_serial_num(561758, evidence(561756,citation(crespillo2003),agora_transaction(2181))). datum_serial_num(561759,datum_source(561758,end)). datum_serial_num(561760,disease(561755,['Urolithiasis'])). datum_serial_num(561761,datum_source(561760,end)). datum_serial_num(561762, evidence(561760,citation(crespillo2003),agora_transaction(2181))). datum_serial_num(561763,datum_source(561762,end)). datum_serial_num(561764,organism('Homo','sapiens',['']))). datum_serial_num(561765,datum_source(561764,end)). datum_serial_num(561766,rxn_location(561699, [organism('Homo','sapiens','nil','nil'),tissue('liver'), cell_type('erythrocytes'),cell_line(''),organ('liver')])). datum_serial_num(561767,datum_source(561766,end)). datum_serial_num(561768,event(561699,created,1114368491,nil, reaction(561699,'APRT',[]),agora_transaction(2181),'APRT')). datum_serial_num(561769,datum_source(561768,end)). datum_serial_num(561770,comment(561699,'This reaction seems to be the only mechanism through which free adenine is incorporated into its corresponding nucleotide in humans.')). datum_serial_num(561771,datum_source(561770,end)). </pre>

<i>Data file</i>	<i>Information entered</i>
datum_source.pl	datum_source(561700,end). datum_source(561702,end). datum_source(561704,end). datum_source(561706,end). datum_source(561708,end). datum_source(561710,end). datum_source(561712,end). datum_source(561714,end). datum_source(561716,end). datum_source(561718,end). datum_source(561720,end). datum_source(561722,end). datum_source(561724,end). datum_source(561726,end). datum_source(561728,end). datum_source(561730,end). datum_source(561733,end). datum_source(561735,end). datum_source(561737,end). datum_source(561739,end). datum_source(561741,end). datum_source(561743,end). datum_source(561745,end). datum_source(561747,end). datum_source(561749,end). datum_source(561751,end). datum_source(561753,end). datum_source(561756,end). datum_source(561758,end). datum_source(561760,end). datum_source(561762,end). datum_source(561764,end). datum_source(561766,end). datum_source(561768,end). datum_source(561770,end).

Table 10: Data entries for an enzymatic reaction deposit.

3.4.8 Re-engineering the Workflow of *Nomenclature*

Since the data for END depend on the *Nomenclature*, our workflow design for END extended to the JCBN. However, I would not observe the JCBN's workflow directly, since all of the relevant members work individually, asynchronously, and at a distance. In fact that the workflow for *Nomenclature* is itself in transition. Another contributing factor to the difficulty was the remote location of the database designers and managers. Since the database design and logical view of was made by the original designers of the Enzyme Nomenclature [12], automating the process was time consuming and difficult to understand the stages involved in the curation process. This was because the database designers were located at a remote location and all correspondence with them was made through phone calls and emails. Since such forms of communication always results in communication gaps and misunderstandings, insufficient amount of information provided, delayed responses during email exchange, *etc.*, it made the task extremely difficult. It was not until the key person visited late last fall that I could observe the actual workflow. This proved to differ somewhat from what she had previously described, and was automated as part of the END workflow. These differences can be accommodated by adding an extra subpath to the current workflow to accommodate entry drafting and the public comment period.

4 Discussion

In *The Agora*, I have not only brought workflow technology to database curation, but also automated the processes that were mechanical to reduce human errors. The “back end” of *The Agora* is designed to increase the overall efficiency and performance of the participating databases, making them sturdier and more consistent. However, it took a great deal of observation to identify the key components in the data flow process that could be automated or consolidated. The initial version of *The Agora* was a pretty straight forward database, but it had many drawbacks. The entire process of adding and revising data was manual and it was very time consuming and error-prone. So, the first step in automating the curation process was to identify the stages of manual entry and then automate them. This was achieved by running experiments using sample data sets and then examining the various actions performed on the data, or in response to various sets of data and conditions. During this examination, the processes and subprocesses that slow down the overall workflow because humans must act were identified. Since some of the processes involved human expertise and external searches, it was not possible to automate them in the current version. These processes included verifying information from resources such as journals and publications, which could not be automated at this time. Maybe in future, I might be able to automate this process once I have complete information on all the possible resources for a particular type of information. However, manual processes like file updates and making basic decisions based on the information, which currently require human agents, were replaced by computer agents and automated. During this process of automation, I also identified a few key processes like notification of the current status of data to depositors or reviewers and arbiter. So I automated the communication process to automatically contact the person and providing the relevant information. In the entire design process, one key factor that was considered

was to make the entire automation process completely transparent to the user. The user still used the existing interface to deposit, review, revise or curate data, but had absolutely no idea of the machinery running in the background.

The process of automating workflow is difficult because it involves understanding the entire existing workflow for the current process, its purpose, the input data and the expected outputs, *etc*, before one can list and differentiate between the processes which can be automated and those that cannot. Not every human action can be replaced with an automatic executor, and not every step in the prior process should be retained. In *The Agora*, I designed a workflow for the curation process and then automated all the possible processes and subprocesses for best results and efficiency. The mechanism was designed in such a way that the same technique can be applied to other processes which are similar to database curation. Having one single overall workflow framework was a major issue in our research. It was necessary to have one common workflow logic for similar processes to minimize costs and improve consistency. Keeping this factor in mind, the workflow for *The Agora* was designed so that all the three types of database, namely END, BND, and *Klotho* use the same framework with slight differences in the format of the data and the relationships among them. This framework can now be termed as a general workflow that can be applied to any processes which perform similar operations, not just in database curation, but also in other industrial processes.

I provided for very fine attribution and management of data to have complete details about every new addition or modification to the database. The main reason for tracking information for new data being added or modified is to roll back to a prior state if the database becomes unstable for any reason. By looking at the tracking information file, one can check which datum was modified before the application became unstable, and then make any changes to either correct the datum if the

reason for the error is known, or remove it completely from the database and notify the arbiter. The possible reasons for this kind of failure are that the arbiter overlooked some information which was not in accordance with the rules of data type information, or a syntax error in the datum, or there might be some conflict among the existing data and the one just entered. In the current version of *The Agora*, I do not have any checks for these kinds of errors, but I might enter some checkpoints for making sure that the application runs smoothly in the future.

Although designing the automated machinery was pretty straight forward, there were certain problems faced during its implementation. Most of the processes for the *Klotho* and the Biochemical Names Database were easy to implement as they were designed and implemented from scratch at the same location. I did iterate several times over the new *Klotho* workflow to make sure it provided all the functionalities of the current workflow. However, automating process related to the *Nomenclature* was difficult. The fundamental problems stemmed from the changing relationship between END and the *Nomenclature*. Originally, data were added to END by parsing HTML pages at the JCBN web site, and suggested changes to those pages emailed back to the web site maintainer. As long as I focussed on automating reaction deposit (for all types of reactions, not just enzymatic), design and implementation were relatively straightforward. Only gradually did the curators of *Nomenclature* realize this process could help them, so that they then started explaining the actual workflow they use. It turns out this is not an efficient process and has many problems with data consistency and control. The process of adding to *Nomenclature* has many problems of logical data consistency because the roles of the actors are not clearly defined or enforced. For example, the data can be silently modified by more than one person acting independently of the others. In spite of agreement that this is inappropriate, the situation persists for historical and personal reasons. While I set out to improve

this position by redesigning the workflow, I only gradually learned the details of the various versions of the current *Nomenclature* workflow. Changing the workflow for END to accomodate the current *Nomenclature* workflow would be a significant step backward. What I did not appreciate at the begining was how novel and potentially threatening a different workflow for the *Nomenclature* could be; the genuine strengths of the existing workflow; and how slowly one must sometimes negotiate the process of workflow design. As a result of this work, the current workflow is changing and I am adapting our design to what I understand is the end-point of this evolution. As the main process for the *Nomenclature* workflow is implemented at many remote locations, many errors resulted in the automation design and implementation. This might seem obvious, since the basis of modifying workflow is examining and understanding the existing workflow. Running experiments with sample data sets and observing the process was a problem. The information received through email and telephonic conversations was not good enough to easily automate this process, and misunderstandings, communication gaps, and lack of information on basic assumptions made the task very difficult. To summarize, one can say that automation is simple if studied properly, but very difficult without strong cooperation between users and designers. Not only the end results, but the many odd situations that arise must be analyzed. If the intermediate stages are not understood, it is very difficult to automate even the simplest process. One error at any stage, and the result will not be accurate any more. It is advisable that the entire process is located at one location for easy communication, testing, verification, and management. The automation task can be difficult if the database is distributed across many locations but is still tightly bound together.

The most important factor while designing the automated process is to make it flexible enough for future inclusions and modifications. If this is ignored, it might not

be possible to easily add more advances in the future. Every addition would require re-engineering of the entire process, which could be time consuming and expensive. The process should be modular so that with few adjustments, new features can be incorporated into the system. Most management experts realize that in order to stay on top, an organization must be constantly evolving [3]. Coding a process into a computer system can have the effect of freezing the process at one instance, and may make it costly to change. Clearly, workflow systems need to have specific features to support process evolution. By making the workflow implementation flexible enough I can diminish the barrier of change and make the system more easy to change and continuously evolve into a better system. In The *Agora*, this was accomplished by learning the process of database curation as it existed in its initial stages. A step by step schema was determined and then each step was inspected for any cause that might be slowing the curation process. I identified all the known bottlenecks in the process and subsequently redesigned and coded them for optimum efficiency. Once the current processes were corrected, a detailed analysis of the curation process was completed, all the possible data types and formats were listed, and then the code was redesigned to make it flexible for future changes. Once the workflow was automated and improvements were added, the curation process was repeatedly tested with all the currently supported data samples for the different databases. Once the tests were successfully passed, the workflow was examined again for any further improvements in the workflow or data processing for improved efficiency. Thus, having the workflow automated and future possible expansions with minimum efforts made available in the workflow, The *Agora* can now support continuous evolution and turn itself into a better system.

4.1 Future Improvements

After the successful implementation of the automated design to The *Agora*, I am now planning to improve it further by bringing new functionalities and features to it. Some of the key improvements that could be incorporated are listed as follows:

- *Providing curation of existing data in the participating databases.* This will help in curating information that was included in The *Agora* before the “front end” was released. Curating existing data makes them cleaner and more current.
- *Continued automation of the Nomenclature workflow.* Since the workflow of *Nomenclature* is extremely complicated, and involves large amounts of data with each new information, I would like to automate more processes in *Nomenclature* workflow for faster results and better consistency.
- *Precheck data for stability.* It might be the case that the new data included in the database could crash the core application for some reason, maybe because of a conflict with the existing data in the database. By having some checkpoints or conditions, I can eliminate this possibility before the application is rebuilt and made available to the public.
- *Route review data based on type of information.* In order to improve reviews on deposited information, I would try to select reviewers based on the type of information. For example, for the same type of deposit, I would look at the information and select a reviewer who has more expertise about those data. So, it would be an additional criterion for selecting reviewers for a particular deposit.
- *Automate the review process.* In the current implementation, two reviewers review the deposited information and make their decision about the deposit.

This process consists of manually verifying the deposited information against the references or other sources of information as quoted in the deposit, and then checking for their validity. If I can automate this stage or some of its parts, I will be able to reduce the time in the review process. This can be made possible by writing some robots that can look up publication databases or journal collections, and check for the references mentioned in the deposit. If the deposited information could be found in the references by these robots in conjunction with a text search program, then I can reduce the need for human agents as reviewers, and automate more of the review process. However, since there are a number of such databases, building such a robot and a text search program which can search for all kinds of text, would require time and knowledge of these databases for accurate search procedures.

- *Communicate with remote databases.* Since the automation process has a common design procedure, I am planning to communicate with databases besides the ones that already are a part of The *Agora*. This will allow The *Agora* to automate community deposits to these remote databases. This communication can be made possible using a central hub and linking all the participating databases using a common syntax and semantics of data. Once the new databases are connected to The *Agora*, it would be possible to have the same services of depositing, reviewing, revising and curating information for the new databases as a part of The *Agora*. This way, I will not just automate the processes on the participating databases, but also provide a separate interface similar to that of The *Agora* for the people. This makes it easy for the scientific community to have a common standardized interface for the processes involved in building different types of curated databases at the same time. That will be the time when The *Agora* would really be The *Agora*. In early Greek history, the *agora*

was primarily used as a place for public assembly; later it functioned mainly as a center of commerce. By providing a common service for all databases, it can have a similar center of scientific information.

BIBLIOGRAPHY

1. Alonso, G. and H. J. Schek, 1996. Research issues in large workflow management systems. Technical Report 1996PA-as96-nsfws.
2. Conradi, R., Liu, C., and M. Hagaseth, 1995. Planning support for cooperating transactions in EPOS. *Inf. Syst.* **20**(4):317–336.
3. Donovan, J. J., 1994. *Business Reengineering with Information Technology*. Prentice-Hall, New Jersey.
4. Dunford-Shore, B. H., Sulaman, W., Feng, B., Fabrizio, F., Holcomb, J., Wise, W., and T. Kazic, 1994–present. Klotho: *Biochemical Compounds Declarative Database*. University of Missouri, Columbia, MO, <http://www.biocheminfo.org/klotho/>.
5. Ellis, C. and J. Wainer, 1994. A conceptual model of groupware. In *CSCW '94: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, pages 79–88. ACM Press, New York.
6. Gary, K., Lindquist, T., Koehnemann, H., and L. Sauer, 1997. Automated process support for organizational and personal processes. In *GROUP '97: Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work : The Integration Challenge*, pages 221–230. ACM Press, New York.
7. Jiang, J., Sanghi, G., Kutikkad, G., Bugrim, A., Boyce, S., Slomczynski, J., McDonald, A., Mummaneni, A., Seth, R., Sulaman, W., Fabrizio, F., Chilukuri, P., Feng, D., Engel, T., Wise, W. B., Ellis, L., Shapshak, P., Tipton, K. F., and T. Kazic, 2002–present. *The Agora*. University of Missouri, Columbia, MO, <https://www.the-agora.org>.

8. Kazic, T., Jiang, J., Kutikkad, G., Yao, G., Bugrim, A., and J. Slomczynski, 2000–present. *Glossa Semiotics*. University of Missouri, Columbia, MO, http://www.the-agera.org/glossa/semiotic_list.ps.
9. Khoshafian Setrag, B. M., 1995. *Introduction to Groupware, Workflow, and Workgroup Computing*. J. Wiley and Sons, New York.
10. Markley, J. L., Ulrich, E. L., Doreleijers, J. F., Mading, S., Maziuk, D., Tolmie, D., Wenger, R. K., Miller, Z., Yao, H., and J. Lin, 2004–present. *BioMagResBank*. University of Wisconsin, Madison, <http://www.bmrb.wisc.edu/>.
11. McDonald, A., Seth, R., Richardson, M., Sachs, N., Wise, W. B., and T. Kazic, 2002–present. BND. University of Missouri, Columbia, MO, <https://www.the-agera.org/bnd/>.
12. Mummaneni, A., Boyce, S., Bugrim, A., McDonald, A., Slomczynski, J., Fabrizio, F., Sulaman, W., Akunuri, B., Wise, W. B., Tipton, K., and T. Kazic, 2000–present. END: *Enzyme Nomenclature Database*. University of Missouri, Columbia, <http://www.biocheminfo.org/end>.
13. National Center for Biotechnology Information, 1995. *GenBank*. National Center for Biotechnology Information, <http://www.ncbi.nlm.nih.gov/GenBank/index.html>.
14. of the IUBMB, M., 2004. <http://www.chem.qmul.ac.uk/iupac/usage/summary.html>.
15. Reiner B, Siegel E, C. J., 2002. Workflow optimization: current trends and future directions. *Journal of Digital Imaging* **15**(3):141–52.
16. Shapshak, P., Duncan, R., Torres-Munoz, J. E., Minagar, A., and C. K. Petito, 2002. Preliminary gene expression studies in AIDS: problems and solutions.

In Granda, W. V., ed., *Proceedings of the Second Virtual Conference on Genomics and Bioinformatics*, page *under processing*. North Dakota State University, <http://www.ndsu.edu/virtual-genomics/>.

17. Sheth, A., Georgakopoulos, D., Joosten, S. M. M., Rusinkiewicz, M., Scacchi, W., Wileden, J., and A. L. Wolf, 1997. Report from the NSF workshop on workflow and process automation in information systems. *SIGSOFT Softw. Eng. Notes* **22**(1):28–38.