Proceedings of the Federated Conference on Computer Science and Information Systems pp. 1043–1051
DOI: 10.15439/2018F173
ISSN 2300-5963 ACSIS, Vol. 15

The Evolution of a Healthcare Software Framework: Reuse, Evaluation and Lessons Learned

Alessandra A. Macedo, José A. Baranauskas
Universidade de São Paulo,
Departamento de Computação e Matemática,
Av. Bandeirantes, 3900 - Monte Alegre,
14040-901 Ribeirão Preto-SP, Brazil
Email: {ale.alaniz, augusto}@usp.br

Renato de F. Bulcão-Neto
Universidade Federal de Goiás,
Instituto de Informática,
Al. Palmeiras, Quadra D, Câmpus Samambaia,
74690-900 Goiânia-GO, Brazil
Email: renato@inf.ufg.br

***Abstract*—The literature describes examples of software frameworks providing developers with generic and reusable functionality for building healthcare applications. Using concepts and technologies from Information Retrieval, Machine Learning, and Semantic Web, we present a novel software framework called HSSF (Health Surveillance Software Framework) which aims to facilitate the development of applications to support health professionals in the prevention of chronic diseases. The main contribution of this paper includes lessons learned distilled from (i) the reuse and evolution of the HSSF components on the development of three new health surveillance applications, and (ii) a quantitative evaluation of the HSSF reusability in terms of time spent and artifacts reused on such development task. Lessons learned are summarized as advantages and drawbacks regarding HSSF reusability. The HSSF allows healthcare applications not only to relate scientific research evidences, exams and treatments, but also to incorporate them together into the clinical practice.**

I. Introduction

THE practice of software reuse can potentially make information technology products more efficient for clients and cheaper for the production. Reuse is not limited to the source code or to the machine code; in a broader sense, documents, coding styles, components, models, patterns and knowledge items may be reused.

One way to promote reuse is to create software frameworks as abstraction of functionalities to build and deploy applications. We consider software frameworks as reusable designs of all or part of a software system described by a set of codes, libraries, tools, APIs, and mainly by abstract classes and the way that instances of these classes collaborate [1]. Therefore, the main goal of a software framework is to create new software more efficiently by means of reuse.

As in other knowledge domains, there are various examples of software frameworks targeting the development of healthcare applications. For instance, OpenMRS [2] [3] describes data items such as clinical findings, laboratory test results or socioeconomic data that can be stored by medical systems. OpenMRS consists of a concept dictionary to avoid the need to modify the database structure to add new diseases. This framework supports the programming of new functions without the need to modify the core code when the focus is the reuse of concepts and/or database models. IBM has developed the SpatioTemporal Epidemiological Modeler (STEM), which models infectious and vector borne diseases. STEM provides developers with a plug and play software architecture for the development of simulations of disease spread, e.g. in bioterrorist crises [4].

In terms of reuse of services, the service-oriented architecture (SOA) paradigm is used as a software framework that aims at facilitating the support for clinical decision [5]. At the level of classes reuse, the virtual reality domain also offers frameworks such as (i) ViMeT [6], which focuses especially on the development of applications that simulate biopsy exams, and (ii) SOFA [7] [8], which targets research into medical simulation. Still considering medical simulation, TES [9] carries out computational simulations of transcranial electrical stimulation.

Despite of the large amount of software frameworks support in healthcare, institutions, professionals and patients are still burdened with the amount of information created due to the mass adoption of the Internet and to the availability of several types of documents, including health records, social websites for health, medical images, among others. Furthermore, the information contained in such documents is too complex and semantically rich for traditional search engines to make sense of. This scenario can benefit from health surveillance systems which can recommend information related to the medical records of a given patient [10], or to a similar user, for example, to provide informational and emotional support in on-line social websites for health [11].

This paper (i) presents the Health Surveillance Software Framework (HSSF) as an evolutionary and reusable design of software components to support the development of health surveillance applications [10], (ii) evaluates aspects of the HSSF reuse, and (iii) lists the lessons learned along the evolution of the HSSF architecture.

The HSSF diagram, an object-oriented application framework, allows distinct health surveillance applications to be created by instantiation of its abstract classes. HSSF has been developed by evolving our previous software which were

This work was supported by the São Paulo Research Foundation (FAPESP - Proc. 16/13206-4) and the National Council for Scientific and Technological Development (CNPq)

designed as a reusable set of functionalities for surveillance systems. Hence, we built HSSF by generalizing software components from three previous endeavors: the *Automatic-SL* [12], the *CISS* [13], and the *FREDS* [14].

The *Automatic-SL* system assists healthcare professionals in their decisions recommending Surveillance Levels (SLs) to identify patients' healthcare needs. It can be used to recommend pediatric procedures in primary healthcare, and it also identifies significant risk factors and protective factors associated with the patients and their families. The *CISS* establishes associations between chronic diseases (cardiovascular diseases, diabetes and obesity) reported in scientific papers and a given patient's clinical record with genetic and epigenetics[1] risk factors. Finally, the *FREDS* effort focuses on the definition of conceptual mappings between the content of medical images and the textual information contained in medical records, applied in a scenario of computer-aided diagnosis in thyroid cancer.

By reusing HSSF components, developers have created three new healthcare applications: the *CISS+* [16], the *CISS-SW* [17] and the *QASF* [18] [19]. They answered a questionnaire that helped us understand the reuse of the HSSF mainly in terms of the number of reused artifacts and the time spent on reuse activities. Finally, lessons learned were elaborated from this whole experience with the HSSF components.

As far as we know, there has been no related research to HSSF in terms of two perspectives: (i) the same underlying theory and technology, including Information Retrieval, Machine Learning, and Semantic Web; and (ii) the health surveillance support, i.e. aiding the prevention of chronic diseases by alerting healthcare workers about risk factors through retrieval of published scientific papers with information on epigenetic risk factors, or even classification of patients into risk groups. Most frameworks in the healthcare domain have proposed reusing services (mainly simulations, electronic health records and medical decision support) and data model (ontologies, dictionaries and databases).

This paper is structured as follows: Section 2 the development of the HSSF framework; Section 3 describes the evaluation carried out as an attempt to understand the HSSF reusability. Finally, Section 4 discusses the lessons we have learned along the process, and Section 5 brings final remarks and perspectives for future work.

II. The development of HSSF

Here we briefly present how the HSSF was developed and evolved in terms of its software components architecture and its core systems including the *Automatic-SL* [12], the *CISS* [13], and the *FREDS* [14] systems. Further details can be found elsewhere[2] [10].

[1]People exposed to risk factors (e.g. food shortage) at the beginning of life can have altered the gene expression, which can impact adult life by posing higher risk of developing chronic diseases. Epigenetics studies those changes in the gene expression [15].

[2]Source code with documentation, papers and reports with UML models are available at the official HSSF website – *http://dcm.ffclrp.usp.br/hssf/*.

A. The HSSF architecture

The architecture of the HSSF is comprised of three main layers — *Presentation*, *Business*, and *Storage* — so that each layer contains its own modules to process documents, as depicted in Fig. 1.

The *Business* layer consists of abstract classes and external packages of utilities, both two in the target domain, as well as two connector layers, called *Communication* layers, which provide the required communication components to the *Presentation* and *Storage* layers.

The *Presentation* layer presents different views and templates such as Graphical User Interfaces (GUIs), which allows access by two main types of users: (a) healthcare professionals, who can analyze risk groups automatically classified by means of surveillance services, or who can receive recommendations of papers related to a given patient's clinical record (e.g. during a medical appointment); and (b) researchers interested in investigating the relationship between risk factors, chronic diseases, risk groups and patients' records.

The two *Communication* layers are composed of connectors for tools, ontologies and knowledge sources. The upper *Communication* layer allows the presentation of recommendations to end users (healthcare professionals and researchers) via GUIs. The bottom *Communication* layer integrates the *Business* layer with features provided by the tools (e.g. classifiers) and knowledge sources (e.g. pre-processed collections of scientific papers and ontologies), and also comprises modules for communication with databases.

In the *Business* layer, the *Search For Papers* module interacts with public repositories of scientific papers. This module collects and updates a collection of papers. Currently, the repository crawler uses concepts from ontologies of target domain to focus the crawler on topics of interest. There is no crawler for clinical records because all clinical records of interest are considered to be associated with scientific papers.

Also in the *Business* layer, the *Textual Processing* module is composed of programming utilities and modules for *Paper Processing*, *Clinical Record Processing* and *Natural Language Processing*. *Textual Processing* module processes textual information from a set of clinical records and collected scientific papers, which are all stored in the *Storage* Layer. Each document (clinical record or paper) is processed, so *Paper Processing* and *Clinical Record Processing* modules can identify simple and complex terms. The *Natural Language Processing* module applies natural language processing, such as processing of n-grams, stemming, removal of stopwords and recognition of concepts. The recognized terms are statically weighted and stored. The processing of clinical records is similar to the processing of scientific papers.

The *Concept Recognition* module manipulates linguistic and knowledge resources, which in turn support the association between different lexical concepts. Clinical records can be manipulated in one language and papers can be processed in another language. For instance, the *Concept Recognition* module exploits classes and methods from the Unified Medical Language System (UMLS) [20] to identify concepts

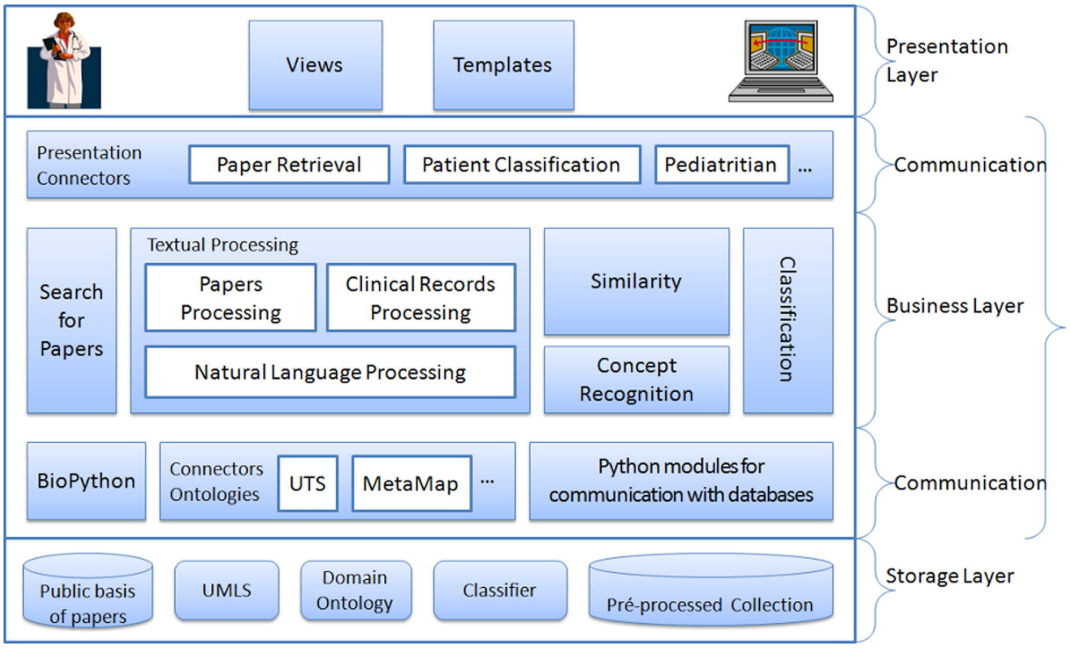

Fig. 1. The HSSF architecture [10].

from health topics such as epigenetics. The overall textual processing of the HSSF, supported by linguistic resources, includes the removal of stopwords, the processing of n-grams, the recognition of concepts, and the computation weights for concepts. The removal of stopwords for the papers and clinical records collections is based on lists of stopwords from programming utilities such as Snowball[3]. The processing of n-grams uses the open source Python NLTK[4] set of modules, linguistic data and documentation for research and development in natural language processing and text analytics.

The processing of clinical records is not identical to the processing of papers: after the processing of n-grams and the identification of concepts, a query array containing the remaining concepts is built for clinical records, whereas a weight matrix is composed of the scientific papers. Both the query array and the weight matrix are submitted to the *Similarity* module, which is in charge of computing similarities. The architecture of HSSF allows the *Similarity* module to calculate similarity measures between papers and clinical records, and it can also apply automatic relevance feedback.

B. The HSSF core systems

This section describes three health surveillance systems that we developed: the *Automatic-SL* [12], the *CISS* [13], and the *FREDS* [14] systems. Together these systems contribute with their main software components so as to build the HSSF architecture previously presented.

The *Automatic-SL* system aims to identify children with developmental problems and therefore assists healthcare professionals in their decisions and in reassessing recommendations as a multidisciplinary team [12]. After each medical appointment at a pediatric care center, this system collects patient information to automatically assign it a Surveillance Level (SL) measure which in turn indicates the type of healthcare procedure and service that a patient needs. Exploiting machine learning techniques, the *Automatic-SL* identifies significant risk and protective factors associated with patients and their families. Therefore, it provides surveillance indications that support preventive care to avoid diseases in adulthood.

As another surveillance service, the *CISS* system retrieves scientific papers that relate chronic diseases to genetic and epigenetic risk factors found in patients' clinical records [13]. From the PubMed repository, a *CISS* module routinely searches and retrieves new scientific papers in the domain of genetic and epigenetic risk factors for chronic diseases. Next, *CISS* processes textual information of that collection of papers for later retrieval of relevant papers according to a clinical record submitted by a healthcare professional.

To associate papers with a clinical record, *CISS* also processes its textual content and then calls a module which calculates the similarity among documents. In turn, this module accesses the pre-processed version of the collection of scientific papers to retrieve papers with the highest degrees of similarity to the clinical records. Selected papers are then presented to a healthcare professional with risk factors associated with the record previously submitted. By using this approach, healthcare professionals should be able to create a clinical routine with families and set up the best possible growing conditions.

Finally, the *FREDS* system was included during the design phase of the HSSF. Aiming to support decision making sys-

[3]A small string processing language designed for Information Retrieval purposes; documentation is available at http://snowballstem.org/.

[4]Documentation is available at https://www.nltk.org/.

tems in terms of diagnosis of diseases, the *FREDS* system establishes conceptual relationships or mappings between microscopic images content and textual information crawled from clinical records [14]. The main idea is to extract complementary information from exams that describe cell components similar to those identified in the microscopic image evaluated by a pathologist. The aim is to contribute with the reduction of the semantic gap between the computational retrieval of medical images and the human interpretation of their content. The *FREDS* system advocates that the semantic mapping can support the generation of knowledge.

The HSSF software framework has then emerged from the experience of developing the three aforementioned health surveillance systems. The orchestration of those systems as software components and their respective (required/provided) interfaces are summarized in Fig. 2.

The *CISS* component provides developers with classes supporting important functionalities: *Scientific Papers Searching*, *Medical Record Processing*, *Article Processing*, *Textual Processing*, *Natural Language Processing*, *Concept Recognition*, and *Similarity*. These functionalities are provided via *ClinicalRecordProcessing* to the *Automatic-SL* component so as it can classify patients according to surveillance level measures automatically computed. The *DocumentProcessing* is used by the *FREDS* component so as it can relate image reports to imaging exams on an automatic way.

The *FREDS* component offers classes to the *Image Feature Extraction*, the *Image Segmentation*, *Image Classification*, and the *Imaging Report Retrieval*. The *ImagingExamProcessing* provides these services. Finally, the *Automatic-SL* component serves a classification functionality to be reused by other applications via *SurveillanceLevelProcessing*.

III. The deployment of HSSF

In previous section, we described how HSSF was originally built by means of its fundamental software components which provide multiple services including clinical record processing, imaging exam processing, among others.

Throughout this section, we present how those components were reused to develop three new health surveillance applications as proofs of concept to the HSSF framework.

A. CISS+

As an evolution of the original *CISS* system, the first application developed by means of the reuse of HSSF components is called *CISS+* [10]. It augments the semantics of terms and concepts of scientific papers and clinical records by means of the use of the UMLS metathesaurus [20] and the MetaMap tool [21]. Experiments with UMLS and MetaMap demonstrated the effectiveness of the concept recognition task with a reduction of roughly 90% of terms.

Also as novelty, the *CISS+* employs automated techniques of relevance feedback to refine queries. The *Similarity* class of the CISS component calculates similarity measures among scientific papers and clinical records and runs automatic relevance feedback using three approaches: (i) using meta-information from the Medical Subject Headings (MeSH)[5] of scientific papers from the PubMed database; (ii) considering the whole set of documents with identified concepts after n-grams processing of medical records as relevant documents; and (iii) considering meta-information from the "Publication Type" field of PubMed papers.

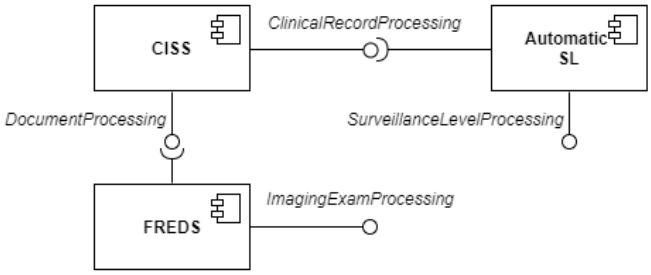

Fig. 2. The component diagram of HSSF.

Hence, the *CISS+* reuses mainly paper searching, textual processing, extended recognition of concepts, and mechanisms of query expansion in the *Similarity* and *Concept Recognition* classes of the CISS component.

B. CISS-SW

Using Semantic Web concepts and technologies, the *CISS-SW* is a search system that enables physicians to retrieve a scientific paper related to a patient's clinical record [17].

After the textual processing, the *CISS-SW* maps terms of papers into RDF triples[6] [22], stores them in a Triple Store[7], and composes a SPARQL [23] query by using the clinical records. With this query, the system retrieves form the triple store the papers related to the clinical records.

Therefore, the new functions of the *CISS-SW* include the processing and retrieval of scientific papers with Semantic Web support. In general, it reuses the classes related to the textual processing of clinical records provided by the HSSF.

C. QASF

A question-answering system returns short and direct answers to users. The *QASF* (Question Answering System in Chronic Diseases) application receives a question about chronic diseases and epigenetics information, and then looks for answers in collections of scientific papers [18] [19]. The aim is to help healthcare professionals to rapidly find focused related answers in the domain of chronic diseases.

The *QASF* architecture essentially consists of three modules: (i) *Question Processing*, (ii) *Answer Processing*, and (iii) *Document Processing* [18].

[5]The U.S. National Library of Medicine's hierarchically-organized terminology for indexing and cataloging of biomedical information.

[6]An RDF triple is a data entity composed of subject-predicate-object, like "John knows Steve" or "Steve is 42". RDF triples are the standard information exchange format in the Semantic Web.

[7]A triplestore is a purpose-built database for the storage and retrieval of RDF triples through semantic queries usually written in the SPARQL syntax.

The *Question Processing* module converts the question a user submits in natural language to a query that helps to search and select answers. This module applies pattern recognition and machine learning algorithms to handle the type and the content of the question, respectively. The *QASF* exploits linguistic and knowledge resources (e.g. WordNet and SNOMED) to support the processing of learning healthcare information.

As the only module reused from the HSSF architecture, the *Document Processing* module retrieves documents (e.g. scientific papers) that might have the answer to the user question. From each candidate document, this module also extracts excerpts that should be the answer.

Finally, the *Answer Processing* module processes all the potential right answers and classifies them according to a similarity value. Currently, the *QASF* considers the cosine between the user question and the candidate answer as the similarity value. Hence, the user is given the "n" first answers.

IV. Evaluation

Considering the software engineering literature, reusability is the degree to which an asset (e.g. a module or component) can be used in more than one software system, or in building other assets. In this section, we describe how we measured the reusability of the HSSF components in developing the systems described in the previous section.

Although the number of reused lines of code is a commonly used unit of measure, a software framework is more than lines of code. In the case of reuse, another well-known measure is the developer hour, which refers mainly to the time spent on searching, analysis and integration/modification. The time spent on these activities must be less than the time that is necessary to develop the artifact to be reused.

In order to evaluate the benefits of the HSSF framework, we requested each developer of the *CISS+* (one participant), *CISS-SW* (one participant) and *QASF* (two participants) applications to answer a questionnaire we elaborated, as illustrated in Fig. 3. The analysis of the corresponding answers was an attempt to understand the reuse of the HSSF components in terms of the time spent and the artifacts reused.

The *QASF* developers answered the questions together, then we have considered only one answer. We have disregarded the amount of time required to locate/search the HSSF as a tool to be reused because all the developers belong to the same group.

A. Questionnaire answers

Regarding the question 1, developers have considered classes, lines of code, and external packages as the software artifacts the most reused assets of the HSSF. From the answers to question 2, we identified that the most useful classes were the *Paper Search* and the *Paper Processing* classes, followed by *Natural Language Processing* and *Concept Recognition*. Regarding the question 3, despite analyzing from 20 to 25 classes, *QASF* developers reused from 1 to 4 classes without modification as well as from 1 to 4 classes with modifications. The *CISS+* developer analyzed between 15 and 19 classes and reused all of them without modifications. However, this person has actively participated in the creation of the HSSF. Finally, the *CISS-SW* developer analyzed from 5 to 9 classes and reused between 5 and 9 classes without modifications.

Considering the answers to question 4, as the *QASF* developers have not found proper documentation, they spent months to understand the model, to integrate the classes, and to understand and extend source code. The *CISS+* developer did not spend time on any of those activities because she is member of the *CISS* development team. Finally, the *CISS-SW* developer spent days reading the documentation and only weeks understanding the model and the classes, integrating these, and understanding and extending source code.

Another interesting result is the number of attempts that each developer made before reusing the HSSF components (question 5). Disregarding the *CISS+* developer, the other developers have only carried out between 1 and 3 attempts before the reuse. This small number of attempts is a good indicator of the HSSF reusability.

The last question of the questionnaire confirms this finding because the two developers have stated that they earned weeks and months regarding the development of their systems when they reused the HSSF. The developers of the *QASF* considered they did not earn much time, but we noticed they did not find any documentation (papers, reports, models, manuals, and help documentation) about the HSSF. Such documentation is available at the official HSSF website. *QASF* was recently created, hence the difficulty of reuse can be due to the need of updating many technologies exploited by the HSSF.

B. The average time spent on reusing HSSF classes

A look at the answers of the questionnaire has motivated us to find an ad hoc measure to quantify the time spent on the reuse of HSSF classes. To assess whether the reuse of classes yields a positive result besides the costs with the modification and integration of the reusable item of the HSSF in the current project, we have defined *the average time needed for classes reuse* as

$$T^{(a)} = \frac{T_c^{(u)}}{C} + \frac{T_c^{(m)}}{C_r} \tag{1}$$

in which:

- $T_c^{(u)}$ is the time associated with understanding of the classes;
- C is the number of classes analyzed;
- $T_c^{(m)}$ is the time spent integrating or extending the classes by source lines of code; and
- C_r is the total number of classes reused without modifications.

We ignored the risk of wasting time on the search for classes because the developers belong to the same team. Therefore, we only considered the questions 3.a, 3.b, 4.b, and 4.d (or 4.e) which are C, C_r, $T_c^{(u)}$, and $T_c^{(m)}$, respectively.

Questionnaire: An attempt to understand the HSSF reuse experience

1. What kind of artifacts did you reuse from CISS? (you can select more than one)

[ ] Models [ ] Classes
[ ] Tools [ ] Lines of code
[ ] Libraries [ ] API
[ ] External Packages [ ] Other:_____

2. If you **reused classes**, which components were more useful for your application? (you can select more than one)

[ ] Natural Language Processing [ ] Paper Processing
[ ] Paper Search [ ] Clinical Record Processing
[ ] Textual Processing [ ] Classification
[ ] Concept Recognition [ ] GUI
[ ] Similarity [ ] Other:_____

3. If you **reused classes**, please answer the following (a., b., c.) questions:

a. How many classes did you **analyze** to reuse from CISS?

[ ] 1 to 4 [ ] 10 to 14 [ ] 20 to 25
[ ] 5 to 9 [ ] 15 to 19 [ ] zero

b. How many classes (without modifications) did you **reuse** from CISS?

[ ] 1 to 4 [ ] 10 to 14 [ ] 20 to 25
[ ] 5 to 9 [ ] 15 to 19 [ ] zero

c. How many classes did you **extend/modify** from CISS to be reused?

[ ] 1 to 4 [ ] 10 to 14 [ ] 20 to 25
[ ] 5 to 9 [ ] 15 to 19 [ ] zero

4. How much **time** did you spend in the following activities to reuse CISS?

a. Reading documentation:

[ ] hours [ ] months
[ ] days [ ] Other: _____
[ ] weeks

b. Understanding/analyzing the model:

[ ] hours [ ] months
[ ] days [ ] Other: _____
[ ] weeks

c. Understanding/analyzing the classes:

[ ] hours [ ] months
[ ] days [ ] Other: _____
[ ] weeks

d. Integrating the classes:

[ ] hours [ ] months
[ ] days [ ] Other: _____
[ ] weeks

e. Understanding/analyzing lines of source code:

[ ] hours [ ] months
[ ] days [ ] Other: _____
[ ] weeks

f. Extending by coding:

[ ] hours [ ] months
[ ] days [ ] Other: _____
[ ] weeks

5. In average, how many **attempts** did you make **before** reusing an artifact from CISS?

[ ] 1 [ ] 4 to 5 [ ] More than 8
[ ] 2 to 3 [ ] 6 to 7 [ ] zero

6. Do you consider the reused artifacts **saved time** of the developing a new application?

[ ] Yes. How much ? Estimative:_____ hours. [ ] No.

Fig. 3. Questionnaire answered by developers who reused HSSF.

Using the answers given by the developers of the *CISS+*, *CISS-SW* and *QASF* applications, we have listed the results for this equation in Table I. This includes the average time (T^a) spent on reusing the classes of the HSSF by developers of the *CISS+*, *CISS-SW* and *QASF* applications. T^a, $T_c^{(u)}$ and $T_c^{(m)}$ are measured in terms of days. C and C_r, in turn, represent numbers of classes.

TABLE I
THE AVERAGE TIME (T^a) SPENT ON REUSING HSSF CLASSES

Developer	C	C_r	$T_c^{(u)}$	$T_c^{(m)}$	T^a
CISS+	19	19	7	6	**0.68**
CISS-SW	9	9	7	7	**1.55**
QASF	25	4	30	30	**8.7**

Table I also shows that the *CISS+* developer spent 0.68 days reusing each class of the HSSF. This was expected because she collaborated with the development of many HSSF classes before developing the *CISS+* application. The *CISS-SW* developer spent 1.55 days reusing each class of the HSSF, whereas the *QASF* developers spent much more time (8.7 days) than the other two developers. This was also expected because *QASF* developers did not find documentation about the HSSF and needed to update HSSF classes and packages.

Besides questions 3.a., 3.b., 4.b., and 4.d. (or 4.e.), some other quantitative answers to the questions can help to measure the HSSF reuse, but the reuse of the model, class, and source lines of code must be related because they consist of artifacts reused at different levels of design abstractions. In other words, our ad-hoc equation previously presented considers independent variables only.

In a near future, we intend to augment this initial effort including other time of reuse of other levels of design abstractions. Another situation to verify includes measures of the topology of the model, which includes connectivity, and between, among others, and should help us to infer coupling of classes, for example. Some literature works have presented result metrics in terms of reuse [24] [25].

V. ANALYSIS, LESSONS LEARNED AND RESULTS

When thinking about reuse, we must consider that the process of creating and updating of a *software framework* should never end. It is fundamental to reuse the framework while aiming to receive the developers' feedback on the development of new versions of the reused framework.

The creation of the HSSF framework has been a long process that has relied on collaborative work including ideas, software requirements, designs and development, augmentations, results and publications. This process started in 2007 with the initial development of the *Automatic-SL* system [12], which was followed by the creation of the *CISS* system [13] and *FREDS* [14]. In the end of 2014, we agreed on a version of the HSSF; and in 2015, we developed two new systems to validate the HSSF, the *CISS+* [16], and the *CISS-SW* [17] systems. Finally, in 2016, we included new classes in the HSSF after we developed the *QASF* system [18] [19].

The HSSF framework provides hot spots that are easy to manipulate such as:

- the insertion of other scientific information resources besides PubMed;
- the use of another domain ontology to create queries and to filter scientific papers from the desired information resources;
- the exploration of other ontologies and/or thesauri aiming at the recognition of medical and biomedical concepts besides UMLS;
- the manipulation of different types of clinical records or other documents in the healthcare domain; and
- the use of other classification and clustering techniques.

Given those hot spots, the less flexible one is the exploration of other ontologies and/or thesauri aiming at the recognition of medical and biomedical concepts besides UMLS. The HSSF carries a multilingual processing (mainly English and Portuguese), so it is necessary to apply a linguistic resource that can relate multilingual concepts and bring semantic relationships. In the healthcare domain, UMLS still represents the best option to recognize concepts, justifying the natural inflexibility. However, an extension of our framework to manipulate other linguistic resources besides UMLS has already been designed.

As frozen spots of the HSSF framework, we can mention abstract classes that allow each hot spot cited, for instance, an abstract class to illustrate ordinary attributes of different sets of documents and another abstract class to represent attributes of different document types. Other frozen spots consist of classes for textual processing (e.g. stopwords elimination and *n*-grams processing) as well as for the identification of relationships among documents.

In 2016, we presented the first version of HSSF as a software framework, which consisted of an architecture and a class diagram depicted elsewhere [10]. After we created the *QASF*, we were able to extrapolate and to update that diagram to integrate *Automatic-SL*, *CISS*, *FREDS* and *QASF*, as depicted in Fig. 4. By analyzing our own reuse experience, we have distilled some lessons learned and also classified these as advantages or drawbacks with the HSSF reuse.

In terms of **drawbacks** of the HSSF reuse, we can consider:

- the need to learn technical aspects of the programming languages, packages and ontologies to generate a steep learning curve that is necessary for the developers to know how the framework works before they can reuse it; and
- the cost in terms of demand of development expertise.

Considering these limitations, the HSSF faces problems in a specific abstraction domain. For years, some authors have advocate that the reuse of software frameworks degrade the performance/efficiency of the application and its security issues [26] [27]. However, HSSF users have not noticed any of these two points yet. Moreover, security is not an essential requirement of the *CISS+*, the *CISS-SW*, and the *QASF* applications.

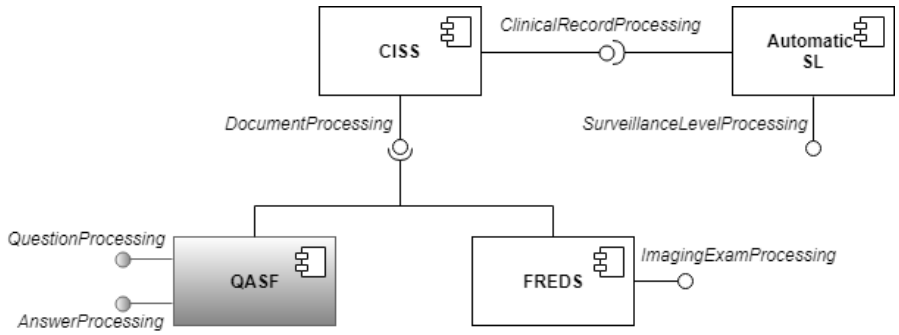

Fig. 4. The HSSF framework extended with two *QASF* services available to developers: *QuestionProcessing* and *AnswerProcessing*.

On the other side, the use of the HSSF offers the following **advantages**:

- the HSSF reduces the time and the energy spent on developing the *CISS+* and *CISS-SW* systems because the developers have the standard infrastructure and diagram of the HSSF providing the organization of modules and the classes of application;
- as a consequence from the previous advantage, developers can devote more time to requirements and user interfaces;
- the source codes of the three systems (*CISS+*, *CISS-SW* and *QASF*) are more organized and well documented because they follow the coding convention of the HSSF, which makes the source codes clean and easy to understand; and
- the use of the HSSF infrastructure affords three well-separated applications and defined business and logic layers from the user interface, making the code cleaner and extensible.

Finally, we believe that the HSSF improves the quality of the applications because the developers focus on the unique requirements of their application instead of spending time on infrastructure. Two of the users of the HSSF consider that their performance during the developing of their applications were improved reusing assets of the HSSF.

VI. Conclusion

The *Automatic-SL*, *CISS* and *FREDS* systems could be abstracted because they had common classes, purposes, and collaborations. For instance, the Relevance Feedback (RF-SL) classifier of the *Automatic-SL* system generates structured information from medical records and transforms it into bags of words; this system also eliminates stopwords and conducts stemming to produce a term-weight matrix. This matrix resembles the concept-weight matrix used by the *CISS* to compute similarity between scientific papers and medical records.

The main difference is that the matrix of the *CISS* uses UMLS concepts instead of simple terms. Nevertheless, the application recognition of concepts and the construction of the weight matrix according to these recognized concepts are perfectly applicable not only to the RF-SL module of the *Automatic-SL*, but also to other classifiers of the latter system. On the other hand, *FREDS* is composed of medical image processing classes that can also be related to textual processing.

Between 2007 and 2012, when the *Automatic-SL* and the CISS systems were created, the HSSF had its first classes designed for reuse with a focus on the processing and classification of healthcare-related information. Some years later, newcomers and specialists of our research group have used the HSSF, which has resulted in three new applications, the *CISS+*, the *CISS-SW* and the *QASF*. The developers of these systems have answered a questionnaire as an attempt to understand the reuse of the HSSF basically in terms of types of artifacts and time spent on reuse. The answers have allowed us to measure the average time that is necessary to reuse HSSF classes and to distill experiences such as lessons learned.

The HSSF is a software framework still under development. In a near future, it will be included a new set of class diagrams with more details for novel types of reuse, allowing much more general applicability. Besides this ongoing work, the results reported herein open new research directions that include:

- extension to the Chronic Disease Ontology (CDO) with knowledge obtained from scientific papers on epigenetics mechanisms and epigenetics risk factors for chronic diseases retrieved by the CISS;
- use of text entailment to map risk factors for chronic diseases;
- integration of new computational tools to map new concepts;
- modelling of an electronic medical record system coupled to *CISS* for use by the pediatric team; and
- investigation into the use of pediatric consensus by the HSSF.

By extending and reusing HSSF capabilities, our main goal is to allow healthcare applications developers to relate science research results, exams and treatments, which may be all incorporated into the clinical practice.

Acknowledgment

This work was supported by the São Paulo Research Foundation (FAPESP – Proc. 16/13206-4) and the National Council for Scientific and Technological Development (CNPq - Proc. 302031/2016-2 and 442533/2016-0).

REFERENCES

[1] D. Roberts and R. Johnson, "Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks," in *Pro. Conf. Pattern Languages and Programming*, vol. 3, 1996. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.46.8767

[2] B. Mamlin, P. Biondich, B. Wolfe, H. Fraser, D. Jazayeri, C. Allen, J. Miranda, and T. W, "Cooking up an open source emr for developing countries: Openmrs - a recipe for successful collaboration," in *AMIA Annual Symposium Proceedings*, 2006, pp. 529–533.

[3] L. P. M. P. Thompson A, Castle E, "Experience implementing OpenMRS to support maternal and reproductive health in northern nigeria," *Stud Health Technol Inform*, vol. 160, pp. 332–336, 2010.

[4] A. Falenski, M. Filter, C. T, A. A. Weiser, J.-F. Wigger, M. Davis, J. V. Douglas, S. Edlund, K. Hu, J. H. Kaufman, B. Appel, and A. K, "A generic open-source software framework supporting scenario simulations in bioterrorist crises," *Biosecurity and Bioterrorism: Biodefense Strategy*, September 2013.

[5] K. Kawamoto and D. F. Lobach, "Proposal for fulfilling strategic objectives of the u.s. roadmap for national action on decision support through a service-oriented architecture leveraging hl7 services," *J Am Med Inform Assoc*, vol. 14, no. 2, pp. 146–155, Mar-Apr 2007.

[6] A. C. M. T. G. de Oliveira and F. d. L. dos Santos Nunes, "Building a open source framework for virtual medical training," *Journal of Digital Imaging*, vol. 23, pp. 706–720, 2010.

[7] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, and S. Cotin, "SOFA: A Multi-Model Framework for Interactive Physical Simulation," in *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, ser. Studies in Mechanobiology, Tissue Engineering and Biomaterials, Y. Payan, Ed. Springer, Jun. 2012, vol. 11, pp. 283–321. [Online]. Available: https://hal.inria.fr/hal-00681539

[8] H. Talbot, N. Haouchine, I. Peterlik, J. Dequidt, C. Duriez, H. Delingette, and S. Cotin, "Surgery Training, Planning and Guidance Using the SOFA Framework," in *Eurographics*, Zurich, Switzerland, May 2015. [Online]. Available: https://hal.inria.fr/hal-01160297

[9] E. T. Dougherty and J. C. Turner, "An object-oriented framework for versatile finite element based simulations of neurostimulation," *Journal of Computational Medicine*, vol. 2016, p. 15p., 2016.

[10] A. A. Macedo, J. Polettini, J. A. Baranauskas, and J. Chaves, "A health surveillance software framework to design the delivery of information on preventive healthcare strategies," *Journal of Biomedical Informatics*, vol. 62, pp. 159–170, August 2016.

[11] L. Jiang and C. C. Yang, "User recommendation in healthcare social media by assessing user similarity in heterogeneous network," *Artificial Intelligence in Medicine*, vol. 81, pp. 63 – 77, 2017, artificial Intelligence in Medicine AIME 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0933365717301185

[12] J. T. Pollettini, S. R. G. Panico, J. C. Daneluzzi, R. Tinós, J. A. Baranauskas, and A. A. Macedo, "Using Machine Learning Classifiers to Assist Healthcare-Related Decisions: Classification of Electronic Patient Records," *Journal of Medical Systems*, vol. 36, no. 6, pp. 3861–3874, 2012.

[13] J. T. Pollettini, J. A. Baranauskas, E. S. Ruiz, M. da Graça Pimentel, and A. A. Macedo, "Surveillance for the prevention of chronic diseases through information association." *BMC medical genomics*, vol. 7, no. 1, p. 7, Jan. 2014. [Online]. Available: http://www.biomedcentral.com/1755-8794/7/7

[14] H. C. Pessotti, L. O. M. Junior, E. G. Soares, and A. A. Macedo, "FREDS: Framework para redução da descontinuidade semântica em imagens médicas," in *Proceedings of the 11th Workshop on Medical Informatics - CSBC 2011*, 2011, pp. 1782–1791.

[15] D. J. P. Barker, "Fetal and infant origins of adult disease," *Monatsschrift Kinderheilkunde*, vol. 149, no. 13, pp. S2–S6, Jun 2001.

[16] A. A. Macedo, J. T. Pollettini, and E. V. Munson, "A chronic illness system using biomedical knowledge sources and relevance feedback," in *IEEE International Sympposium on Computer-Based Medical Systems*, 2015, pp. 244–249.

[17] J. Chaves, J. Pollettini, and A. Macedo, "Relating biomedical information using information mapping supported by semantic web," in *Proceedings of the 15th World Congress on Health and Biomedical Informatics*, ser. MEDINFO 2015, 2015, p. 1p.

[18] L. F. Almansa and A. A. Macedo, "Sistema de informação para perguntas e respostas em doenças crônicas," in *Proceedings of the 16h Medical Informatics Workshop - CSBC 2016*, Porto Alegre/RS - Brazil, July 2016, p. 10p.

[19] L. F. Almansa, G. Rubio, J. A. Baranauskas, and A. A. Macedo, "A question-answering architecture for surveillance information systems on chronic diseases based on knowledge from linguistic resources," *Submitted to Knowledge and Information Systems (KAIS)*, p. 25p., Feb 2018.

[20] O. Bodenreider, "The unified medical language system (UMLS): integrating biomedical terminology," *Nucleic Acids Research*, vol. 32, no. Database-Issue, pp. 267–270, 2004. [Online]. Available: https://doi.org/10.1093/nar/gkh061

[21] *UMLS Reference Manual [Internet]*, National Library of Medicine (US)., 1999.

[22] R. Cyganiak, D. Wood, and M. Lanthaler, "RDF 1.1 concepts and abstract syntax," W3C, W3C Recommendation, Feb. 2014, http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[23] A. Seaborne and S. Harris, "SPARQL 1.1 query language," W3C, W3C Recommendation, Mar. 2013, http://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

[24] H. Koziolek, "Performance evaluation of component-based software systems: A survey," *Perform. Eval.*, vol. 67, no. 8, pp. 634–658, Aug. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.peva.2009.07.007

[25] M. Gupta, Chetna; Rathi, "A meta level data mining approach to predict software reusability," in *International Journal of Information Engineering and Electronic Business*, vol. 5, no. 6, Hong Kong, 2013, pp. 33–39.

[26] M. Fayad and D. C. Schmidt, "Object-oriented application frameworks," *Communications of the ACM, Special Issue on Object-Oriented Application Frameworks*, vol. 40, no. 10, oct 1997.

[27] A. A. Al-Baity, K. Faisal, and M. Ahmed, "Software reuse: the state of art," in *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*, Athens, 2013, pp. 1–7.