Pattern- Based Architecture of a system for systematic review of clinical trials

Author: Margreth Venaely Kileo
Student No: s1951998

Email:venamagie@yahoo.com

Master Thesis Report

University of Groningen

Faculty of Computer Science

Software Engineering and Distributed systems

12/8/2011

Pattern-based architecture of a system for systematic review of clinical trials

Supervisor University of Groningen

dr.R.Smedinga

Supervisor University Medical Center Groningen (UMCG)

Gert van Valkenhoef

ACKNOWLEDGEMENT

This report is the final deliverable of my Master's thesis project performed at the University Medical Center Groningen (UMCG) in the field of pattern - based software architecture. I would like to express my appreciation to those people who supported me to the completion of my MSc studies.

First, I would like to thank Dr. Bert De Brock who connected me with UMCG to conduct my thesis project. Also, I would like to thank my supervisor at the University of Groningen Dr. R. (Rein) Smedinga, J.H.JongeJan and my supervisor at the UMCG Gert van Valkenhoef. Thank you for your time, knowledge and constructive idea to pursue this research.

Furthermore, I want to thank my colleagues at department of Epidemiology for providing me a nice working environment by sharing knowledge and life experience during lunch time. My appreciation will also go to my colleagues at the department of computer science, we shared knowledge which helps me to the success of my studies. And I would like to thank Mr. Mathias Galster and Hamad Kamal for sharing with me their experience on software architecture.

Also I would like to thank NFP (Netherlands Fellowship Program) which gave me the scholarship and opportunity to pursue my master's program in the Netherlands.

My appreciation will also go to National examination Council of Tanzania (NECTA) who accepted my absence and allowed me to study in the Netherlands for two years.

I want to thank my lovely husband Andrew John Mntambo for his encouragements, patience and his care that supported me in my studies. Also, I would like to thank my wonderful son Ivan Andrew for accepting my absence as his mother which gave me more strength in my studies .

Finally, I would like to thank my parents and friends for their prayers, faith and support in pursuing my studies. Thank you all and God bless you.

Abstract

Clinical trials are used as medical evidence for clinicians or medical decision makers to support the treatment decisions and improve patient care. To have reliable evidence to inform their decisions, they should perform a systematic review of the medical literature in order to summarize and analyze the available clinical trials in a systematic way.

However, the clinical trials are scattered to different information systems such as abstract databases and clinical trials registries which are not interoperated to each other. Due to poor data structure in those systems, the clinical trials are not accessible or re-usable in an automatic way. As a result, the process for systematic review is time consuming and error prone due to much manual work in literature searches and screening activities.

To solve these problems, we propose the design of a new system by using the pattern-based software architecture approach in order to support the process of systematic review in an efficient way. As a result, the new system enables the accessibility and re-usability of clinical trials from different sources in a single point of view. Therefore, the pattern-based design increases the efficiency of systematic review by removing manual work in literature searching and screening activities.

Table of contents

1 Introduction

The medical literature is rapidly growing which increases the need for electronic databases that enable clinicians or medical decision makers to support their medical decisions (17).Clinicians or decision makers use the clinical information stored in the electronic databases to perform a systematic review of clinical trials. The systematic review is used as source of evidence for medical decision making by summarizing the available clinical information from different databases in systematic way. There are various challenges for clinicians and medical decision makers to perform the systematic review due to the lack of information system which perform the systematic review in a single point of view. This is because, the clinical information are stored in different information systems which are not interoperated to each other. Moreover, the clinical information is not accessible and re-usable to reviewers because of poor data structure in those systems (11).

In order to solve the existing problems in systematic review we propose the design of a new system that enables the reviewers to perform the systematic review from a single point of view. To design the architecture of this system we will use software patterns designs. Software patterns provide generic solutions to domain problem by enabling the re-use of existing system designs (4).

In this document we will investigate how software patterns can improve the quality of software architecture for a system for systematic review. First, we focus on describing the current status of systematic review of clinical trials and show how the software architecture of existing systems affects the efficiency of systematic review. Thereafter, we will discuss the requirements of the new system and then describe different patterns that can be used to design that system. Then, we will design the architecture of the new system and show how the proposed architecture fulfills the identified requirements. Finally, we will conclude by describing the results of using software patterns in designing the system.

1.1 Organization of the thesis

This thesis is divided into 7 chapters; the first chapter shows the introduction of the research, problem definition and the research questions we are going to answer. The second chapter describes the general understanding of systematic review and it shows the current status for

systematic review of clinical trials. In this chapter, we show how the existing software supports the systematic review and explain how the reviews are published. Also, we discuss how the software architecture of the existing system affects the efficiency of systematic review. Then, we provide the summary of this chapter. The third chapter describes the vision of the new system, its context, stakeholders, use cases, functional requirements and non-functional requirements. In the fourth chapter we analyze different patterns that can be used in our system design. The fifth chapter shows the initial model of the software architecture of the new system designed, elaborated models of complete architecture with the patterns that used to design such a system and sequence diagrams. In the sixth chapter we evaluate the designed architecture and show how it meets the system requirements. Finally, we conclude our research by discussing the results of using software patterns to design a system that can improves the efficiency of systematic review and shows the gaps need to be covered in the future research.

1.2 Problem Definition

Clinical trials provide authoritative medical evidence for informing treatment decisions and improving patient care. Due to that, researchers and medical decision makers are using clinical trials as evidence to support their medical decisions. In the evidence-based medicine approach, the researchers should find the research studies from different literature searches and extract the clinical evidence relevant to the patient's problem or the research question (5).

However, there are various information systems that provide the clinical trials information such as clinical registries and medical literature databases. The medical decision makers should conduct an efficient search of medical literature from those sources in order to find the relevant evidence for their research. Due to that, they have to perform the manual work of combining the search results from different sources to enable the availability of unbiased evidence, because those sources are not interoperating with each other. Thus, it is difficult for them to access the clinical trials information from those sources automatically. As a result, collecting, assessing and analyzing the clinical trials information from those sources is extremely time consuming and difficult (11).

To summarize the available trials, the researchers and medical decision makers perform the systematic review that attempt to identify and synthesize all the available clinical trials from different sources in a systematic way. As a result, the systematic review can be used as a source of information in evidence based medicine that contains information from the clinical trials. The

systematic review involves different steps such as literature searches, screening activities, data extraction and analysis(5). There are various information systems to support systematic reviewing but every step of systematic review uses different information systems. Thus, there is lack of an information system that can perform all steps of systematic review from a single point of view.

In order to understand the steps of systematic review we describe an example of a review performed by a group of researchers to assess the effectiveness of psychological interventions delivered by psychological specialists and generalist clinicians. The objective of this research is to compare the effect of interventions (for diabetes) delivered by generalists' clinicians and psychological specialists (2).

First, the reviewers perform the literature search at Cochrane Library (a database for systematic reviews) and 5645 search results were found. Then, the two reviewers independently perform the first screening of abstracts and titles from the search results and they found 141 articles to be included in their research based on their selection criteria. So the reviewers perform the second screening by reading the full-text of those 141 articles in order to identify the clinical trials which are relevant to their research question. Finally after reading those articles they identify 41 articles which are relevant to their research and they found 35 unique clinical trials from those articles. So the reviewer extract clinical data and perform the analysis, in which the findings show that psychological and general clinicians are similarly effective in delivering psychological interventions (2).

However, there are challenges in performing literature searching and screening activities in an efficient way. Those challenges still exist because there is lack of information systems which can perform the literature search automatically from different sources. As result, the reviewers have to search the literatures from various databases and perform the screening activities manually. Moreover, it is difficult to identify clinical trials because they are published in article format and not in a semantically structured way. As a result, it takes time for reviewers to read those articles and extract clinical data in a trial oriented format ready for the analysis. Thus, the process for systematic review is time consuming and error prone because of manual work of literature searching and screening activities.

The efficiency of systematic review can be improved by designing a system that will provide all sources of clinical trials from a single point of view in an accessible way. This design will

produce quality software architecture that can solve the existing problems of systematic reviewing. The designed architecture will provide the good design practices to support the process for systematic review in an efficient way.

1.3 Research Questions

To reach our goal of improving the efficiency of systematic review the following research questions must be answered.

How can software patterns improve the quality of software architecture of a system for systematic review of clinical trials?

In order to answer this question the following sub-questions have to be answered

- What is the current status for systematic review of clinical trials?
- How does the software architecture of existing systems affect the efficiency of systematic reviews?
- What are the requirements for the new system?
- How can software patterns support the design of the new system based on the identified requirements?
- What are the architectural designs of the new system based on the selected patterns?
- Does the designed architecture meets the identified requirements?

2 Background

In this section we present an overview of the background of our problem statement, in which we discuss the meaning of systematic review and describe the steps for performing the systematic review. Then we discuss the current status in abstract databases and clinical registries. Thereafter, we explain the existing software for systematic review and then describe how these systematic reviews are published. Finally, we explain how the architecture of the existing systems for systematic review affects the efficiency of systematic reviewing.

2.1 What is systematic review?

Evidence-based medicine is the process of systematically finding, appraising, and using research findings as the basis for medical decisions. Evidence based-medicine can be practiced in any situation where there is doubt about a specific clinical diagnosis or comparison of treatments on a certain diseases (5).

One of the most important sources for evidence based medicine is clinical trials. Clinical trials are randomized studies which answer specific research questions, for example to find the best ways to treat a certain disease. So, the clinicians need to assess the available clinical trials from different sources in order to find reliable evidence to inform their decisions (21).

To enable this, a review has to be conducted to find high-quality clinical trials, but the traditional narrative literature reviews did not describe how the research studies were searched selected and appraised. As a result, there were biases and random errors in documenting the clinical trials information. To solve this problem, the systematic reviews were established in the 1990's as the method to assess the available evidence from the medical literature in a systematic way (5). Systematic review is defined as a review which includes a comprehensive, exhaustive search for primary studies, selection of studies based on the eligibility criteria and critical assessment of the included studies (21). In addition, systematic review may include synthesis of the results by meta-analysis, a statistical method to summarize the result of several clinical trials. Thus the systematic review is used to summarize the available evidence before the medical decision makers apply it.

In order to perform a systematic review, the following steps have to be followed:

1) Formulation of a focused review question

2) A comprehensive, exhaustive search and inclusion of primary studies

3) Quality assessment of included studies and data extraction

4) Synthesis of study results: Meta-Analysis (optional)

5) Interpretation of the results and report writing.

These steps are also shown in the Figure 1 below.

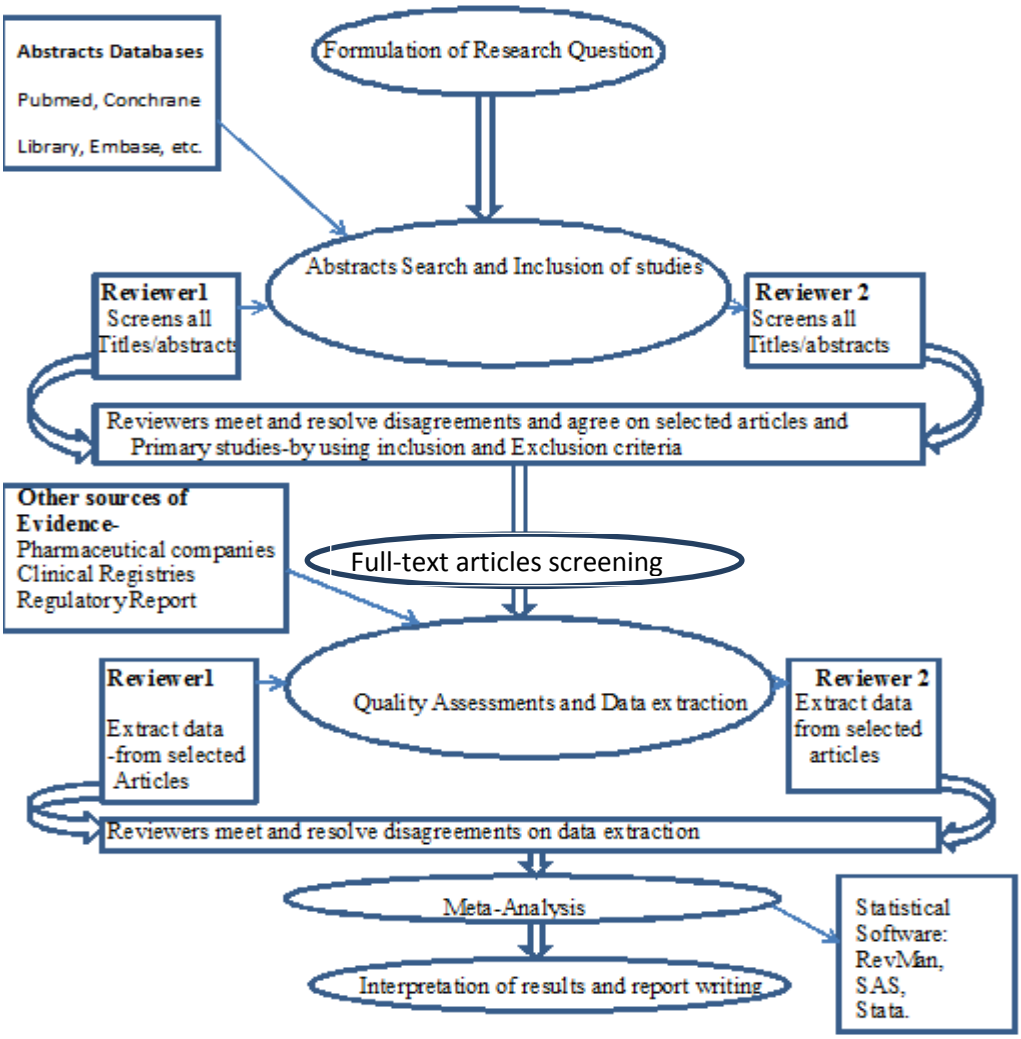


Figure1: Architectural of systematic review (21)

As shown in the Figure 1, the first step in systematic review is formulation of the research question which specifies the patient population or the disease of interests, the intervention and the outcome measures of the trial. For example consider a review on Chinese herbal medicines for the treatment of hepatitis B. The focused question will be "are Chinese herbal medicines

helpful in increasing the response to alpha-interferon (outcome) as compared to interferon therapy alone? “

The next step is literature searches and inclusion of primary studies, this process involves search of abstract/titles of the articles from all relevant databases and sources. To perform the exhaustive search for primary studies, different search filters for specific study are used for example PubMed

Clinical queries filters are filters used to simplify search task in the abstract database called PubMed(15). Although these databases are useful in identifying published clinical trials, there are more other sources like Cochrane Controlled Trials Register (CENTRAL), in which reviewers can search for abstracts. In addition to abstracts databases, there also clinical registries which used to register the clinical trials, so the reviewers also has to search for relevant clinical information in different clinical trial registries. After searching all sources, the reviewer will save all the citations of search results into citation management software to process search results and keep track of search strategies used to find the abstracts.

After conducting the search of abstracts/titles, the next step is screening of abstracts in order to select the abstracts which can be used in the second screening. This process involves two reviewers, who work independently for screening all titles/abstracts .The abstract screening is required because irrelevant abstracts cannot be filtered out automatically. This process may take a long time because the reviewers have to review hundreds or thousands of abstracts in the search results. Clear inclusion and exclusion criteria must be used in order to select the appropriate abstracts relative to the research question. For example, inclusion criteria may depend on how narrow or broad the research question is. If the research question is narrow then few studies will be relevant but if the research question is broad then a large number of studies will be included in performing the systematic review. Example of inclusion criteria that may be used is “studies that include result for patients with type IIIB or IIIC open tibia fractures resulting from acute trauma” (5). After each reviewer has selected the included and excluded abstracts, the two reviewers meet and resolve disagreements of the selected abstracts. The selected abstracts are then ready for the second screening of full-text articles.

In the second screening, the reviewers read full-text articles of selected abstracts and make the final decision about their relevance based on the inclusion and exclusion criteria. They also have to find out whether there are overlapping articles from different sources.

As shown in the Figure 1, the next step of systematic review, both reviewers work again independently to perform the quality assessment of included studies and data extraction from the final selected articles by using paper data extraction forms. Normally, they extract information on study characteristics, methodology, population, interventions and outcomes. After this step, the reviewers meet and resolve disagreements or agree together on the extracted data and primary clinical trials they include in their analysis (5).

The final data after this process is ready for data entry in the statistical software (SAS and Stata) or systematic review software for meta-analysis, in order to synthesize and summarize the results from different studies.

In the final step, systematic reviewers should interpret results and write the report based on the available evidence. The report should include a discussion of the limitations of the primary studies, potential biases in the original articles, strength of the evidence and directions for future research (21).

2.2 What is the Current status for systematic review of clinical trials?

In this section we present the overview of current status of systematic review in abstract databases and clinical registries.

2.2.1 Database of Abstracts

The databases of abstracts were established to support literature searching and the retrieval of articles. These databases use different search strategies such as clinical trial categories, in order to enable easy retrieve of articles which are relevant to research question (15).

The U.S National Library of Medicine (NLM) established the abstracts database called Medline, which contains journals citations and abstracts for the biomedical literature. These abstracts are accessible online through the PubMed search engine. PubMed is free resource developed and maintained by the National Center of Biotechnology Information (NCBI) at the NLM (7).Currently, PubMed comprises over 20 million citations for biomedical literatures from Medline, life science journals, and online books. Thus, it is used by most reviewers in the first step of systematic review by searching for abstracts which are relevant to their research question (7).

In order to enable the search capabilities, PubMed use a number of tools to assist in building search queries, such as MeSH terms and specialized search strategies that reduce the searching task for the reviewer when performing a systematic review, but this may cause inaccurate results

and loss of information. This is because some of the relevant studies will not be included in the search results, even though they are good evidence. So clinical queries filters make the easiest way for researcher to find the information in the PubMed but on the other hand they produce biased results with loss of relevant clinical trials (15).

Also, there are other abstract databases used in literature searches of clinical trials; such as EMBASE and Cochrane CENTRAL. The EMBASE database, published by Elsevier Science has more than 4,000 journals published in 70 countries. EMBASE is more comprehensive than MEDLINE in its coverage of European and pharmaceutical journals. There is an overlap of approximately 34% in coverage between the two databases (7). The CENTRAL register of clinical trials was established by the Cochrane Collaboration in order to serve as a repository for trial reports identified by members of the Cochrane collaboration. Trial reports for CENTRAL are identified by using electronic searches of databases such as PubMed and page-by-page hand searches of journals and conference abstracts. It shows that, it is difficult to collect the trial reports from different sources because there is no single database or combination of databases that includes all trial reports (7).

Therefore, the process for systematic review is time consuming due to screening activities and manual data extraction from articles because the search results are presented in publication format and not study oriented. In other word, the search results are an unstructured data set, which causes difficulties for reviewers to find and use clinical trials in their research. Moreover, theses databases support only literature searches but do not support other steps like data extraction and meta-analysis, so they don't fulfill all the requirements of systematic review.

2.2.2 Clinical registries.

Apart from the problems found in the previous section, there is publication bias in the journals and databases of abstracts due to selection in publishing the trial results. For instance, studies with positive results are three to eight times more likely to be published in journals with higher citation indexes than studies with negatives results. Also well-designed studies that do not have statically significant results may not be published in journals (5).To solve these problems, many medical journals now require the clinical trials to be registered before starting collecting data for research in order to eliminate publication bias by capturing all trials.

The registration of clinical trials was already proposed in 1986, and in the US, the legislation for it was made by FDA modernization Act of 1997 which makes clinical trial registration a legal

requirement. Due to that, in 1998's, the National Library of Medicine initiated the development of clinicalTrials.gov, which was available in 2000 as a web-based system for trials registration (11). This registration of clinical trials helps to remove publication bias because the clinical trials are registered without considering if the results are positive or not.

The clinical trial can be registered in ClinicalTrials.gov through the web based system called Protocol Registration System (PRS) (12). The researcher has to create the study protocol in the PRS and once they finish data collection they can submit their results in the system. The clinicalTrial.gov staff reviews the results before publishing them. The results are submitted in a tabular and text format which results in difficulties in accessing them in an automated way.

Not only Clinicaltrials.gov enables trials registration, also other countries set up their own registries, but all these registries are less sophisticated than ClinicalTrials.gov. Table 1, shows different registries and number of registered trials in those registries (Registry studies column). The 'results' column show whether the results are registered. It shows that, the clinicalTrial.gov is the largest registry with a larger number of studies than other registries (11).Moreover, ClinicalTrials.gov has already made the results of some trials accessible but other registries do not even register the results.

Register	**Registry Studies**	**Results**
ClinicalTrials.gov	86,613	yes (1,156)
ISRCTN register	8,206	no
Australian New Zealand Clinical Trials Registry	3,681	no
Japan Primary Registries Network	3,500	no
The Netherlands National Trial Register	2,027	no
Chinese Clinical Trial Register	727	no
Clinical Trials Registry - India	703	no

Iranian Registry of Clinical Trials	210	no
German Clinical Trials Register	157	no
Sri Lanka Clinical Trials Registry	43	no
Pan African Clinical Trial	14	no

Table1, Trial registration in different Clinical registries. Source from (11)

The International Ministerial Summit on Health research in 2004 established the WHO International Clinical Trials Registry Platform (ICTRP) in order to unify trial registration world-wide. The ICTRP is used to create the network of international clinical trial registries in order to provide a single access point for clinical trials. The ICTRP provides a search portal that collects the results from primary registries and groups together the trials that are registered in more than one registry. However, the search portal provides its results in textual format (11).

In addition to trial registries, the Pharmaceutical Research and Manufacturers of America (PhRMA) established a repository for clinical trials as another source of evidence. The clinical repository is accessible by different pharmaceutical companies to upload their clinical trials results. But all included reports in the database are text based, due to that, it is impossible to automate processing of the trials (11).

Therefore, despite of existence of various registries and results databases there is no single source that is complete because many registries are used to register the clinical trials but they are not accessible due to an unstructured data format (11).

2.3 What are the existing software for supporting systematic review?

Most of the software for systematic review is used in the last step of systematic reviews in order to summarize and perform meta-analysis of the clinical trials. This software used to process and store the body of evidence based on the research questions.

The Cochrane collaboration was established in 1993 as an international network of people to help healthcare providers, policy makers and patients to make a well informed decisions based on the available evidence. Though there are various statistical programs to support Meta -

analysis, Cochrane Collaboration established Review Manager (RevMan) as the official software to prepare and maintain systematic reviews (20).

The main functional requirement for this system is to perform the meta-analysis of clinical trials. The stakeholders of this software are: reviewers who prepare the systematic reviews and Cochrane Collaboration staff that approve the reviews submitted by reviewers (20). RevMan provides the interface to allow storing and sharing of systematic reviews, and submitting reviews to the Cochrane collaboration .Then after the approval of their reviews by Cochrane staff the systematic reviews are published to the Cochrane library of systematic reviews. The Cochrane Library is a database which contains summaries of systematic reviews and results of meta-analysis (24).

In order to prepare the systematic review in RevMan, the reviewer has to create a protocol. However, RevMan does not support other steps for systematic reviews like literature searches and data extractions. So, the reviewer has to find literature from other information systems after creating the protocol in RevMan, because RevMan is not interoperated with other sources of evidence.

As shown in the Figure 2, the reviewer creates a protocol called “Topical Capsaicin for chronic pain in adults” which means, the review will identify the available clinical trials of the treatments for chronic neuropathic pain in adults(6) . After creating this protocol, the reviewer has to manually perform the literature searching and screening activities. Then the information from the search results, such as publication information of the selected studies has to be manually entered into RevMan.

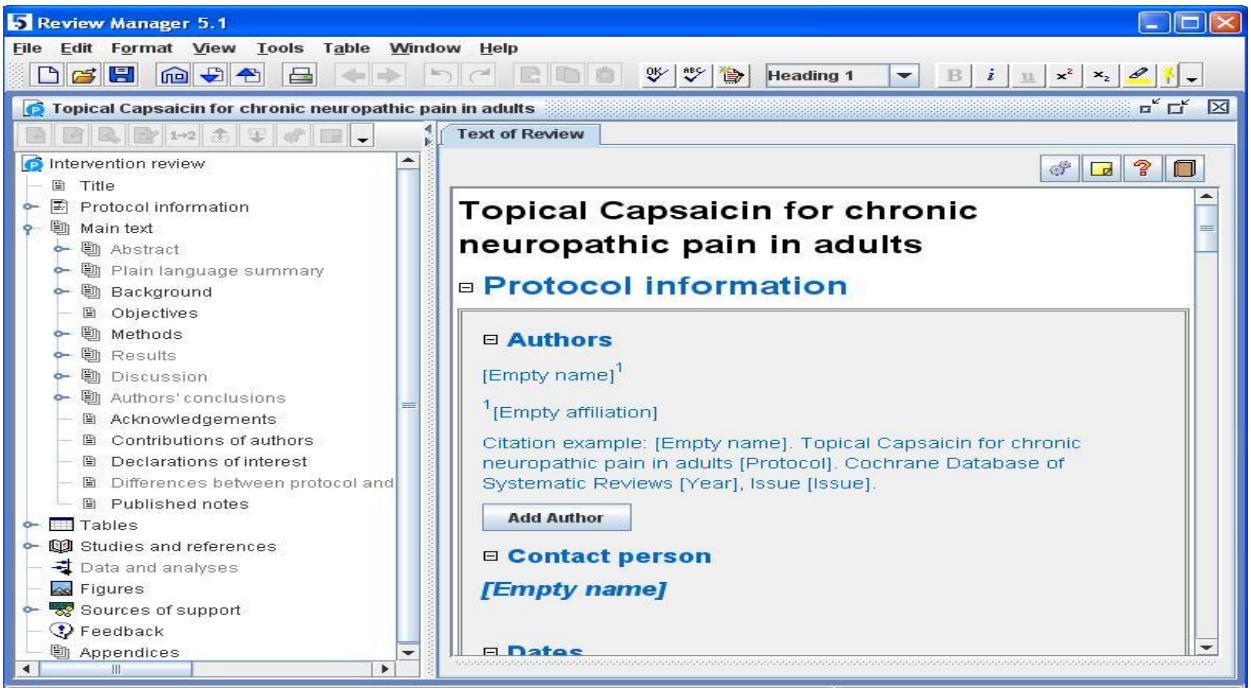


Figure 2: Screen shot for Review Protocol in RevMan (6)

After that, the reviewer has to enter the characteristics of included studies in a table format which show, method of the study, participants' information, interventions and outcomes measures used in the study as shown in the screen shot in Figure 3.

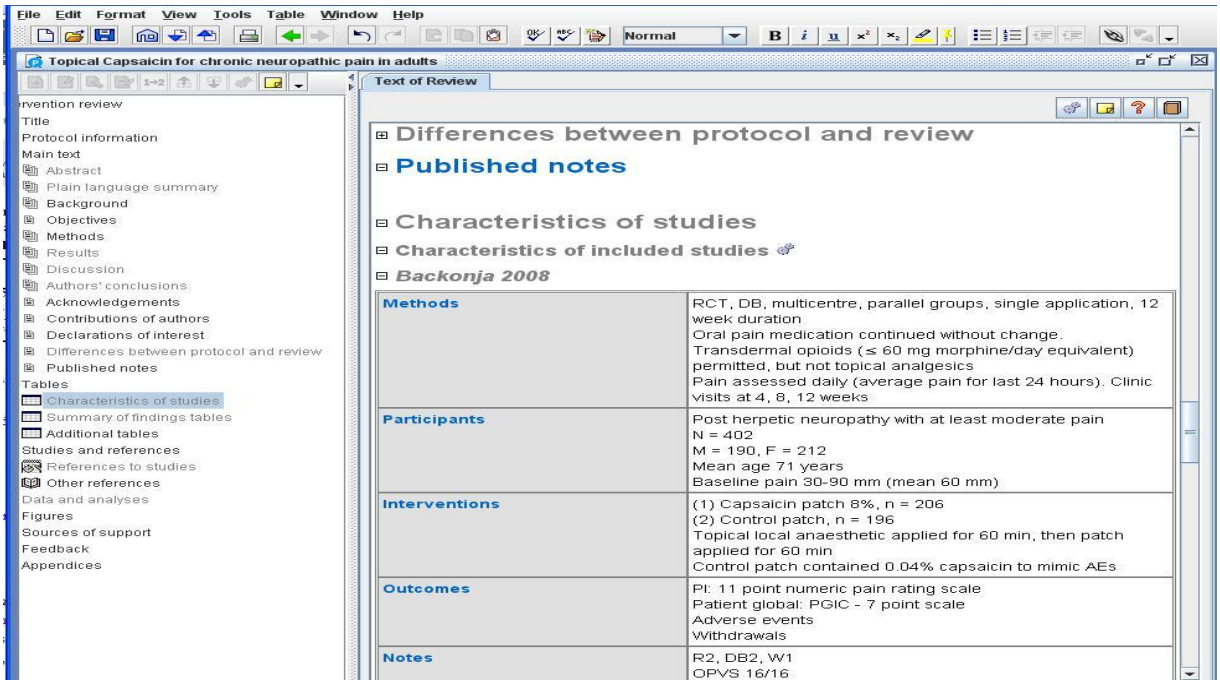


Figure 3: Screen shot for study characteristics in RevMan (6)

Thereafter, the reviewer has to perform data analysis by selecting the appropriate studies for analysis, comparing studies and choosing the outcome measures of every study. Then, the reviewer enter manually the data for every study in the analysis table as shown in the Figure 4, because RevMan cannot extract data automatically from existing study tables. In the example shown in the Figure 4, the reviewer selected two studies Backonja 2008 and Bernstein 1989 that provide the results of using Capsaicin versus placebo as the treatments for the chronic neuropathic pain in adults (6). In this case, every study provides its results in mean, standard deviation and the total number of patients for the given treatments. To enable data analysis, RevMan analyzes all the results of different studies by using built in statistical packages, which are used to generate the analysis report by combining the results of these two studies.

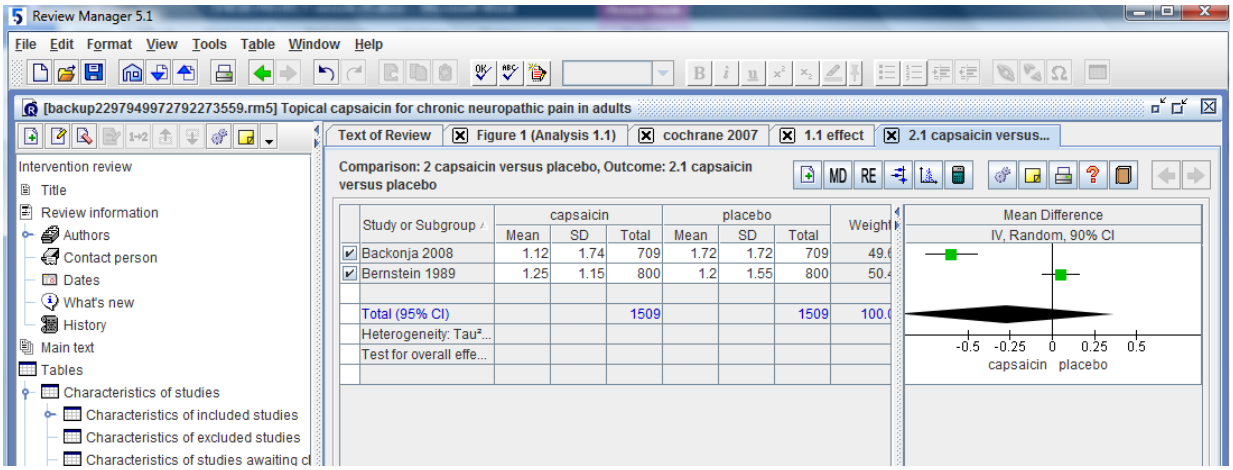


Figure 4: RevMan Screen shot for Data analysis table (6)

Finally, after analysis the RevMan will provide the results in a graph format called Forest Plot as shown in the Figure 4. Then the report for systematic review can be created as full article, which can be used as evidence to support the medical decisions (10). This report can be stored in the Systematic Review database, and after its approval it can be published to Cochrane library.

There are other statistical software like Stata and SAS which can perform meta- analysis, but they cannot create the full article of systematic review, they provide only the statistical functions for Meta -analysis. Due to that, if reviewer needs to use that statistical software, they should first enter their data in software like excel for data management and import data to the statistical software for data analysis. Thus, the RevMan provides more complete support for systematic review.

However, the process for performing systematic review through RevMan is time consuming due to manual work because the reviewer has to enter data manually when creating full article of systematic review. This is because RevMan cannot process and store all the activities of searching and screening the search results. So, it takes a long time for reviewer to collect enough and accurate data for creating the full article of their systematic review. Apart from that, the study results in RevMan are presented in tabular formats specialized for the analysis. Due to this, other reviewers cannot automatically retrieve data from those tables because they are stored with insufficient meta-data. As a result, duplication of effort for reviewers is unavoidable, because the information generated in screening, quality appraisal and data extraction are not stored in a way that can be used by other researchers.

Moreover, there is other system to perform the systematic review such as Aggregate Data Drug Information System (ADDIS) which used to support automated meta-analysis of clinical trials as well as evidence-based decision support (10). However, the ADDIS is used to automate more the last step of systematic review but it does not perform other step of systematic review such as searching and screening activities.

2.4 Publication of reviews

The report of a systematic review can be published into different sources such as medical journals or review databases in order to be re-used by clinicians or decision makers as a source of clinical evidence(11). The reviews stored in those sources are used to inform the treatment decisions or assess the benefits and risks of using alternative drugs in a certain diseases. However, the reviews published in medical journals are in article format. As a result, it takes time for reviewers to extract data from those journals since the data are stored in text format and not machine readable format (11).

The Cochrane Collaboration (11) uses the Cochrane library to publish the reviews in order to increase the accessibility of clinical data through the internet. But, the reviews published in Cochrane Library are in review format not trial oriented format. So, it takes time for reviewers to find relevant information from that database (24).

2.5 How can software architecture of existing systems affect the efficiency of systematic review?

The process for systematic reviews involves different steps and every step has its own information system to meet the requirement for systematic review. Due to that, the architectural

designs of these systems may affect efficiency of systematic review because every system has its own architecture with different requirements and stakeholders.

The software architecture of RevMan is designed to process only the results of primary studies included in the review. This software does not support other steps for systematic review like literature searches and data extractions. So the reviewer has to find literature searches and extract data manually from different sources of evidence before using RevMan for performing the systematic review. Due to that, it may affect the efficiency of systematic review because it will take time for reviewers to collect enough data from different sources.

The problem is not only in RevMan, but also there is lack of application interface which provide all clinical results from different sources of evidence in a single access point. Due to that, there is lack of a structured database which contains the results of clinical trials in a study oriented format.

Another problem is lack of study identification because every nation has its own study identification. Also some of the study in the clinical registries does not have link with the abstract of the study from the databases of abstracts, so researchers have to search manual the clinical abstracts which are relevant to their studies. Due to that, it is difficult to trace the available studies from different sources of evidence (3), (8).

Also there is lack of interoperability between databases of abstracts, clinical registries and statistical software. This is because every source of evidence has different formats for presenting their results. So the reviewer has to perform hand- searching to combine the results from different sources which may result in accuracy errors or results bias and duplication of effort for reviewer in performing literature searches and data extraction (21).

Moreover, there is lack of re-usability of evidence, because study results stored in statistical software or RevMan are in a tabular format without any meta-data to indicate their meaning, so they cannot be re-used by other researchers for re-analysis. Due to that, other researchers cannot expand or extend the available evidence because the results of studies are not accessible in a machine-readable format. Therefore, the available evidence from abstract databases, clinical registries and other sources of evidence are not transparent, accessible and re-usable, because the architecture of their systems does not support those functionalities. As a result, there is inefficiency processes in performing systematic review.

2.6 Summary of Background

The process for systematic reviews involves various steps and some steps are already automated in information systems but other steps are still performed manually, such as literature searches and screening of search results. Other steps like analysis and report writing are not the problems in our case, because so far there is Review Manager or statistical software which can perform those activities.

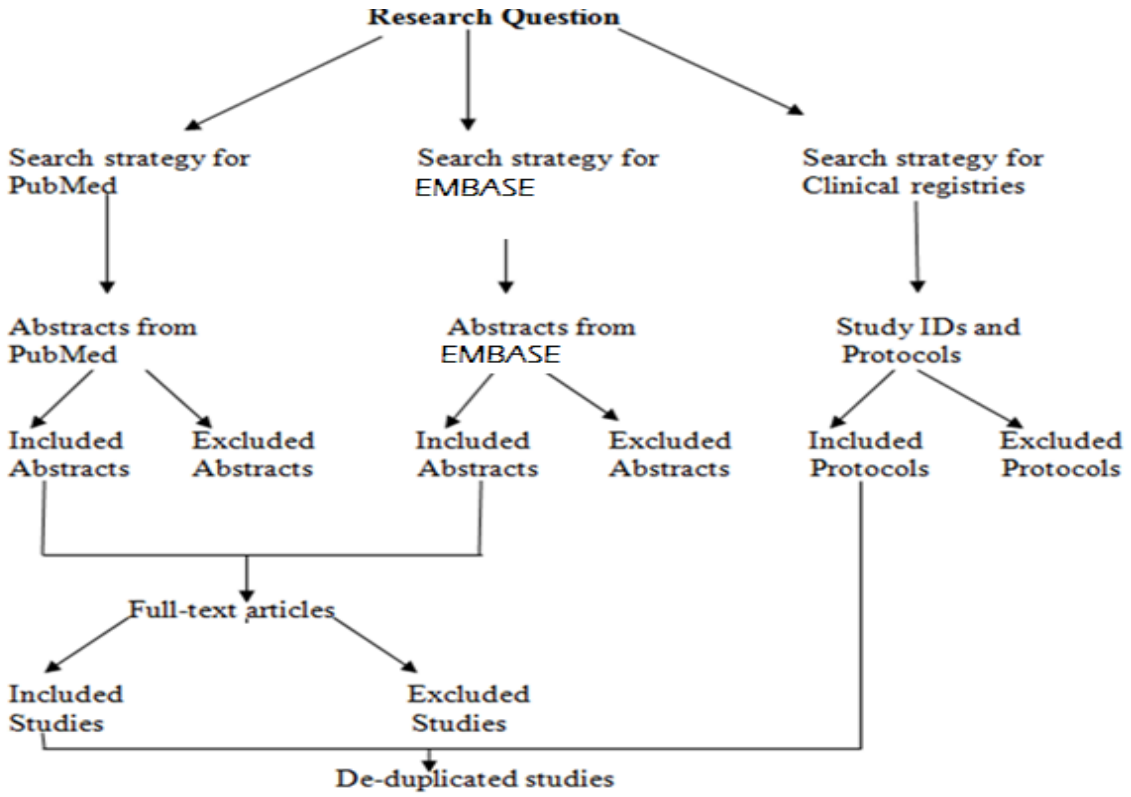


Figure 5: Searching and Screening activities.

The figure 5 shows that, there are various gaps in processing the systematic reviews especially in the literature searching and screening activities. It shows that, after formulating the research question, the literature can be searched in different electronic databases, such as MEDLINE or EMBASE. Not only that, but also the search activities can be performed in clinical registries and different search strategies can be used to obtain the search results from those sources. The problems in performing searching and screening activities are due to the lack of an information system which perform automatically search and process the search results from different sources of clinical trials in a single system. Due to that, it's a challenge for reviewers to perform the literature searches, because the search results from databases and registries are not interlinked to

each other. As a result, the reviewer has to perform search activities manually from those sources.

As shown in Figure 5, the reviewer performs the abstracts screening for every search result from those sources. And according to their selection criteria, they come out with included abstracts and excluded abstracts. Then, the reviewer combines manually the included abstracts from those sources. As a result, there is duplication of abstracts, which causes difficulties for the reviewer to identify those duplicates and eliminate them. Due to that, the reviewers perform manually the de-duplication activities, in order to obtain the included abstracts ready for full-text screening.

Also it is difficult for reviewers to perform the full-text screening because the included abstracts are not always interlinked with their articles. In this process, the reviewers need to read manually the articles but it is difficult for them to select the clinical trials from those articles because those articles are not published in study oriented format and the published clinical trials from clinical registries are not interlinked with their articles. According to their selection criteria the reviewers select the included clinical trials and excluded clinical trials from full-text screening and clinical registries protocols. As result there is much manual work of combining the included clinical trials from databases and clinical registries because these information systems are not interoperated to each other.

Due to that, the reviewers need to perform manually the de-duplication process in order to remove all the duplicated clinical trials and any overlapping information from databases and clinical registries. So, it takes time for reviewers to produce de-duplicated clinical trials which can be used as evidence in the analysis.

Moreover, there is loss of useful information such as excluded abstracts, clinical trials and excluded protocols from clinical registries, because there is lack of information system which can store those information in an re-usable format. As a result, duplication of efforts for reviewers is unavoidable because the reviewers have to repeat the same procedure for searching and screening activities which results repetition of tasks in performing systematic review.

Therefore, the process of systematic review is performed in an inefficient way due to manual work of searching and screening activities, because there is lack of information system which can provide all the literature searches from different sources in single point of view. So the new system for systematic review has to be designed to solve those problems in which all the steps for systematic review can be processed in one system. Due to that, the new system needs to

produce a structured data set which will be accessible by reviewers in order to perform the efficiency systematic review. In this document we will focus on the design of a software architecture which can automate the processes of searching and screening search results in order to improve the efficiency of systematic review.

3 Requirements

Vision

We need to develop an information system that performs all the steps of systematic review in a single point of view. To reach this goal, we develop the software architecture of the system called "Clinical Review System (CRS)" that can solve the existing problems of literature searches and screening activities. To design such architecture, the CRS should provide transparency in accessing the clinical trials information from difference sources of evidence. Also it should allow interoperability with other heterogeneous systems such as databases of abstracts or clinical trial registries and enable the re-usability of data stored in the system as source of evidence.

System Context

The CRS is designed to provide the clinical trials information from different sources of evidence in a single point of view. The design of this system reduces the manual work of performing the systematic review by enabling more automated literature searching and screening activities.

As shown in the Figure 5, the CRS (central rectangle) can interact with two type of users; systematic reviewers and computer scientists. The systematic reviewer can interact with the system through its GUI to perform literature searching and screening activities. Also the system provide API for computer scientists to create algorithms for supporting the automatic processes such as processing abstracts, de-duplication of abstracts and processing the clinical trials.

We need the project manager to coordinate the users of the system and other administration issues. The project managers can interact with the CRS to manage the user's information or reviews stored in the system. Also, the system should enable the developers to extend the existing functions with new features.

Initially, the CRS is designed to perform only searching and screening activities because it can interact with other external system such as ADDIS and RevMan to perform the other steps of systematic review such as data extraction, analysis and report writing.

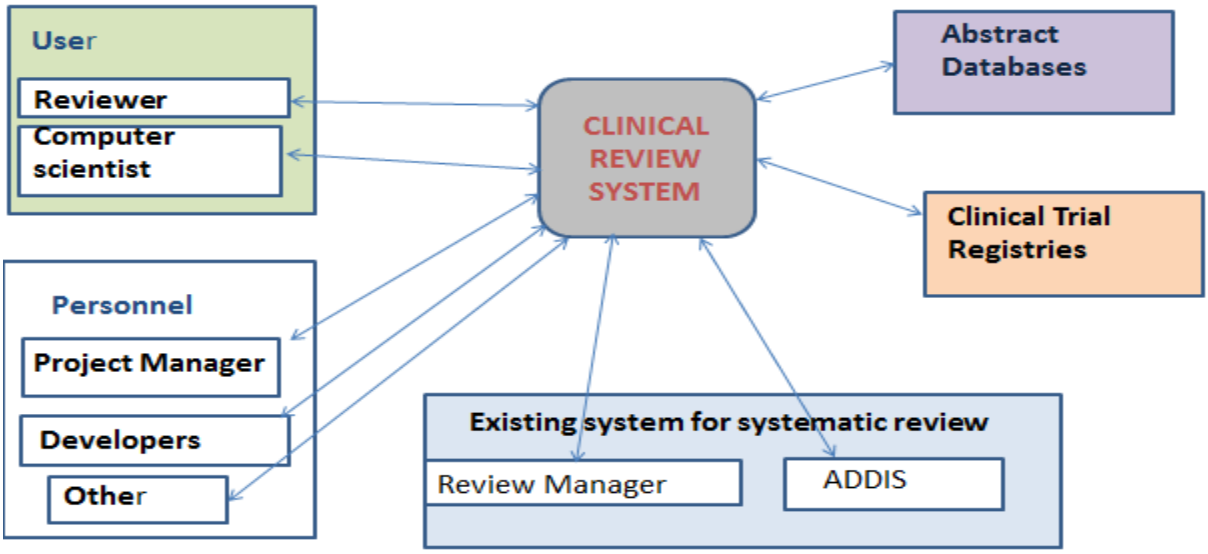


Figure 5: System context of CRS.

3.1 Stakeholders

From the system context we derive the most prominent stakeholders of the system. In the Table 2 we list these stakeholders with their most significant concerns and priorities. For the definition of quality attributes we refer to ISO/IEC FDIS 9126-1, an ISO standard on software quality(16).

Stakeholder (Priority)	Stakeholder's Motivation	Concern	Concern's Priority
Project Manager (Medium)	Project manager is concerned for the long-term success of the project and increasing the market share.	**Scalability**: The system should be able to start from few users but able to scale with many to users including automated ones and the system should be able to manage large databases of trials and abstracts.	High
		Interoperability: The system should be able to interface with existing legacy system and other external systems.	High
Systematic Reviewer(High)	Systematic reviewers have high priory because they are the main users of the CRS.	**Usability**: The system should be easy to use by providing GUI to enable literature searching, and screening activities.	High
		Security: The system should be secured. Sensitive data(email and other private information) should be protected and encrypted	Low

		Performance: The system should provide quick response from every user request, it does not matter how is it performed but it should provide quick feedback to users, such as error message or produce interactive functionality to users.	Medium
Computer Scientist(High)	Computer scientists have high priority because they facilitate the usability of the system in an automatic way which will reduce manual work in performing the systematic review.	**Re-usability:** The system should allow reuse of the system's components through its API to enable automatic literature searches, processing of abstracts and automatic de-duplication of abstracts and studies.	High
		Integribility: The system's component should be more independent and loosely coupled to each other.	High
3rd Party systems (High)	Have high priority because the 3rd party systems used to enable the access of literature searches from different sources in a single point of view. As a result, it will facilitate the automatic process for literature searches	**Interoperability**: Integrated databases should be able to interoperate with the main system to enable literature searches	High
		Modifiability: The system should be able to adapt new features of integrated databases.	Medium
		Re-usability: Other existing system such as RevMan/ADDIS should be able to re-use data stored in the system	High
Developers (Medium)	Are medium because no need for developers to develop new system when system requirement changes.	**Extensibility**: The system should be able to adapt new features based on the needs of the stakeholders.	Medium
		Scalability: The system should be able to grow with increasing need of users and developers while ensuring functionality of old features and new features	Medium

Table 2: Stakeholders and their concerns

3.1.1 Key drivers

Key drivers are the most important quality attributes which the architectural design software has to fulfill. We identify our key drivers by listing the required quality attributes with the highest priority based on the stakeholders concerns. As shown from table above systematic reviewers, computer scientists and integrated databases are the stakeholders with high priority, so we select our key drivers based on the priority of their concerning as shown above.

1. **Usability**: The system should be easy to use by providing a GUI to enable the literature

searching and screening activities.

2. **Interoperability**: The system should be able to interface with existing legacy system and other external systems such as abstracts databases, clinical trials registries or review databases.
3. **Re-usability:** The system should allow reuse of system's components within the system, with other existing system for systematic review such RevMan/ADDIS, or by computer scientists through an API.
4. **Integratability:** The system's components should be independent and loosely coupled to each other.
5. **Scalability**: The system should be able to support large databases for abstracts or trials and it should grow from small number of users initially to a larger number of users later on.

3.2 Use cases

In this section we describe how the reviewers and computer scientists can interact with the System. We present the use case of the whole system, and show the use case which is not solved in the existing system of systematic review. As shown in Figure 6, the use case which is not solved is study identification, but other use cases are not the problem anymore in performing the systematic review because they can be solved by Review Manager or statistical software.
In figure 7, we show more in detail how the reviewer can perform the study identification in the CRS in order to provide the solution to the existing problems.

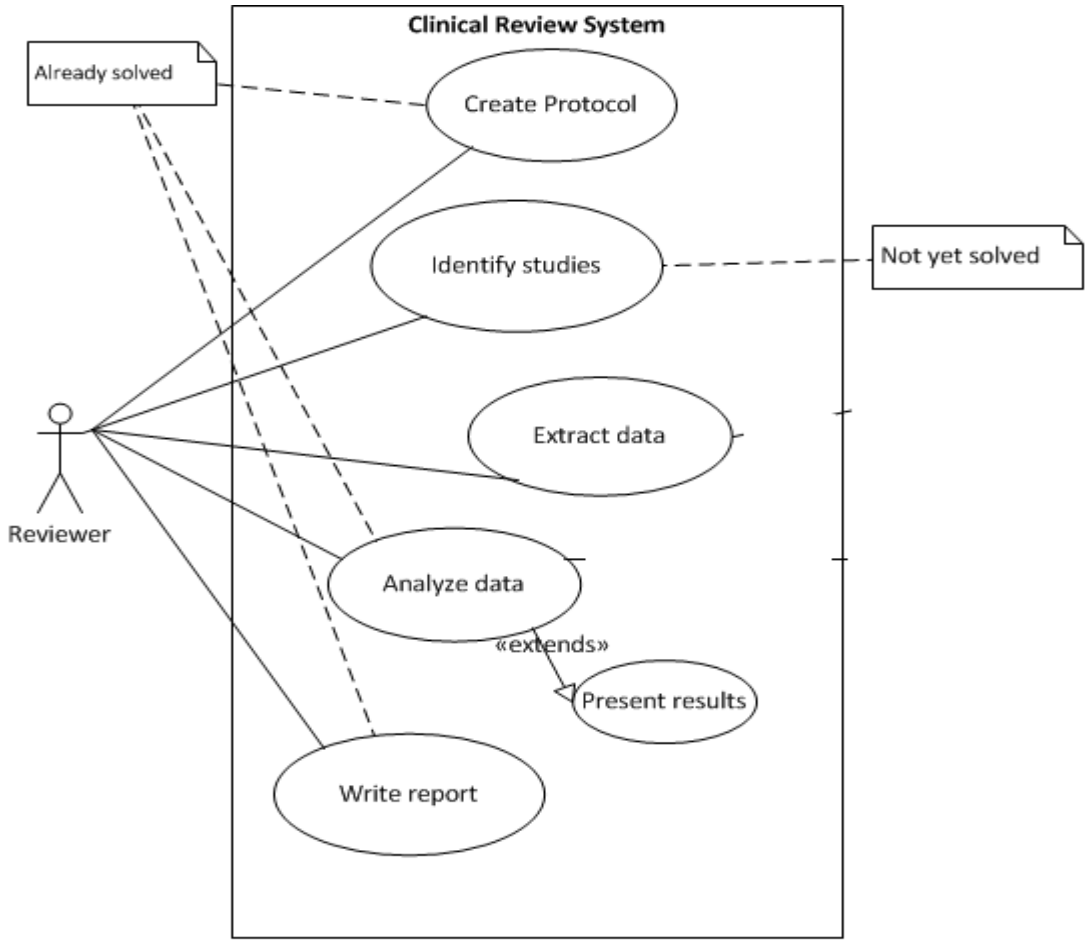


Figure 6: General use cases

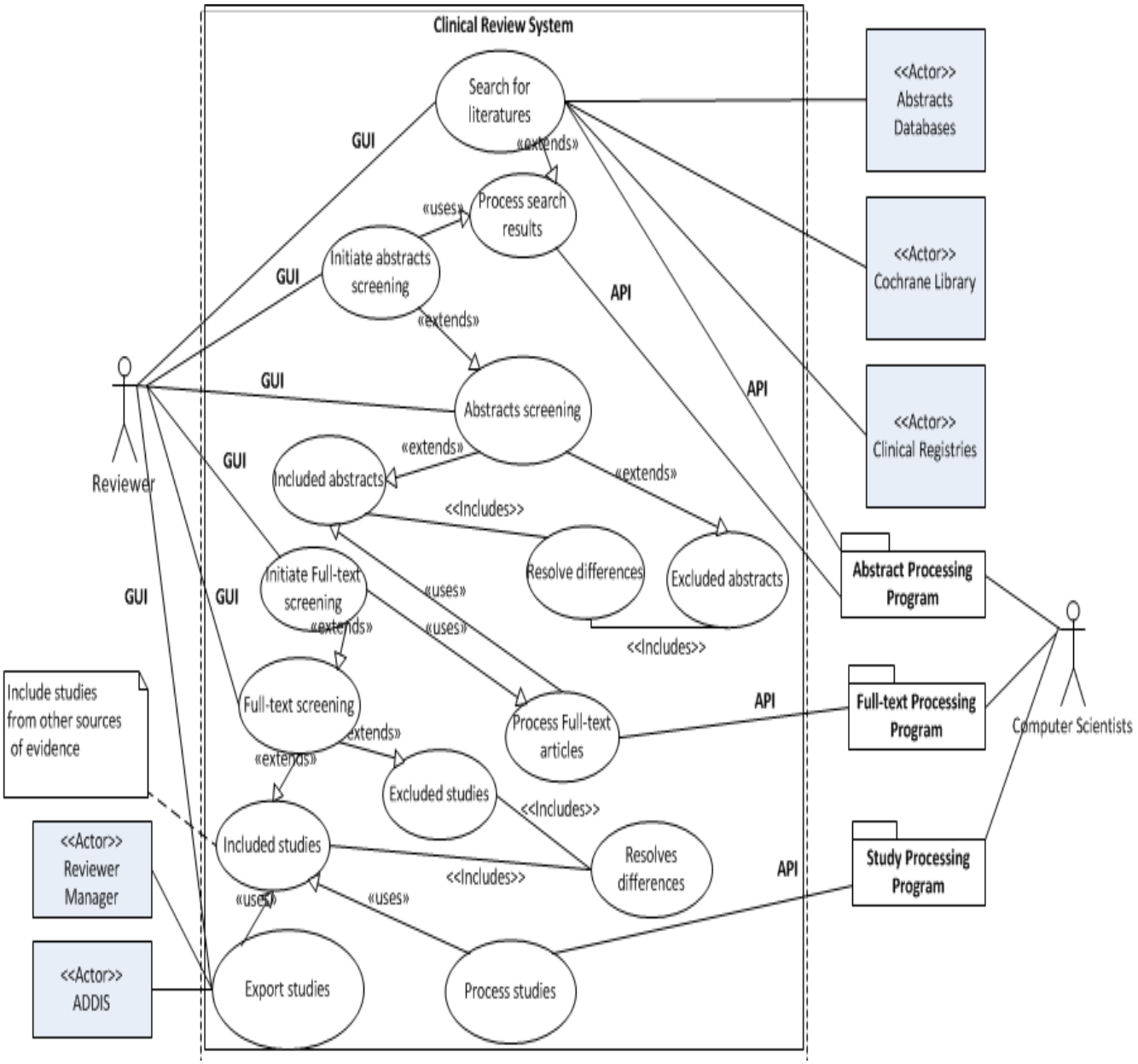


Figure 7: The extended use case of "study identification".

3.2.1 UC_1 Search for Literature

Actor	Reviewer
Triggers	
Pre-conditions	Reviewer has already formulated the research question and created the review protocol.
Post-conditions	The lists of abstracts will be displayed ready for screening activities
Success scenario	1. The reviewer enters the search key words through the GUI to search in different databases and clinical registries.

	2. The system automatically performs the search query in its own database and external sources. 3. The system automatically processes the search results from those database and clinical trials registries 4. The system automatically groups the search results by enabling the automatic de-duplication of entries of the same abstracts from different databases based on their meta-data. 5. The system automatically saves them as frozen data and assigns them a unique ID number, in which they are displayed to Reviewer as search results ready for screening activities.
Alternative Flows	1. Reviewer selects a specific database to perform a special query 2. Reviewer types the special query for the selected databases. E.g. may use filters for specific query such as PubMed clinical query filters or Cochrane filters for RCTs 3. A list of query results is shown with their search strategies or filters.

3.2.1 UC_2a Initiate Abstracts Screening

Actor	Principal reviewer
Triggers	
Pre-conditions	The search results are saved in the system as frozen data and displayed to the reviewers.
Post-conditions	Two reviewers have been assigned to perform abstract screening.
Success scenario	1. The Principal reviewer approves the abstracts lists. 2. Principal reviewer assigns two reviewers to perform abstract screening independently.
Alternative Flows	

3.2.1 UC_2b Abstracts Screening

Actor	Reviewer
Triggers	
Pre-conditions	Two reviewers have been assigned to perform abstract screening independently.
Post-conditions	The two reviewers select included abstracts and excluded abstracts based on their inclusion and exclusion criteria defined in the review protocol.
Success scenario	1. Reviewer clicks the button for abstracts screening through the GUI provided by the system. 2. A list of abstracts is shown and reviewer select the included Abstracts base on their inclusion and exclusion criteria. 3. The system save them as included abstracts and keep a log of

	excluded abstracts with reasons for their Exclusions.
Alternative Flows	

3.2.1 UC_2c Resolve differences (abstracts)

Actor	Reviewer
Triggers	
Pre-conditions	The two reviewer have already selected the included and excluded abstracts in the abstract screening.
Post-conditions	The two reviewers agree on the selected abstracts.
Success scenario	1. The system automatically combines the included abstracts and excluded abstracts from two reviewers. 2. The system identifies differences and resolve them **3.** A list of included agreed on by the reviewers is ready for the full-text screening.
Alternative Flows	The two reviewers dis-agree on the selected abstracts then the third reviewer will resolve their disagreement.
Alternative 2	They can resolve their differences without 3rd reviewer, where they meet together and resolve their differences.

3.2.1 UC_3a Initiate Full-text Screening

Actor	Principal reviewer
Triggers	
Pre-conditions	Two reviewers agree on the included abstracts ready for second screening.
Post-conditions	Two reviewers are assigned to perform full-text screening independently.
Success scenario	1. The Principal reviewer approves the included abstracts 2. Two reviewers perform full-text screening independently.
Alternative Flows	

3.2.1 UC_3b Full-text screening

Actor	Reviewer
Triggers	
Pre-conditions	Two reviewers are assigned to perform the second screening (Review of full-text articles)
Post-conditions	A list of clinical trials are shown.
Success scenario	1. Reviewer clicks the button of Full-text screening. 2. A list of included abstracts with their DOI or URL of articles will be shown and reviewer click the link of every articles

identified.
3. The reviewers read all articles identified in the included abstracts.
4. Reviewer select the included trials based on their inclusion and exclusion criteria, and keep log of excluded clinical trials with the reasons of exclusions

Alternative Flows

3.2.1 UC_4 Resolve differences (identified clinical trials)

Actor	Reviewers
Triggers	
Pre-conditions	The included and excluded clinical trials already identified by two reviewers.
Post-conditions	A final list of clinical trials is created with necessary meta-data.

Success scenario

1. Two reviewers create lists of included and excluded clinical trials.
2. The system identifies and resolves differences of the lists created by two reviewers.
3. A list of included clinical trials from two reviewers will be saved in the system and the system will assign them a unique ID (e.g. Review study ID). The system automatically group together every review ID of clinical trials with their registries ID, DOI or URL of articles, abstracts ID and databases ID.

Alternative Flows

3.2.1 UC_5 Process Clinical Trials

Actor	computer scientists
Triggers	
Pre-conditions	The two reviewers already agree on the included clinical trials.
Post-conditions	Clinical trials processed according to the demand of the reviewers.

Success scenario

1. The Computer scientist creates a Study Processing Program to enable automatic processes of clinical trials through the CRS's API.
2. The Study Processing Program sends the request to CRS the API.

Alternative Flows

3.2.1 UC_6 Export clinical trials

Actor	CRS
Triggers	
Pre-conditions	The reviewers create the review protocol in the RevMan/ADDIS and they need to import the clinical trials from CRS.

Post-conditions The clinical trials exported to Review Manager or ADDIS to perform other Steps of systematic reviews, such as meta-analysis or report writing.

Success scenario

1. The CRS receive the request from the RevMan/ADDIS.
2. The CRS process the request and export the clinical trials to RevMan/ADDIS through its GUI.
3. RevMan/ADDIS receives the file of exported trials.

Alternative Flows

3.3 Functional Requirements

Functional requirements derived from the description of the system context, use cases and deficiencies from the existing systems for systematic review.
We provide our priorities according to the existing problems or challenges in performing systematic review. Due to that, we give high priority to the requirements of searching and screening activities. Other requirements are given medium or low priority because there are already solved in existing systems.

3.3.1 User Account

ID	Requirement	Priority
FR1	Users should be able to create a personal account on the system	High
FR2	Users should be able to update or modify the data in their personal account	High
FR3	Users should be able to remove their account	Medium

3.3.2 Formulation of research question

ID	Requirement	Priority
FR4	Users should be able to formulate the research questions based on patient, intervention, comparison between treatments and outcome measures	Medium
FR5	Users should be able to create a protocol for their research questions.	Low

3.3.3 Searching

ID	Requirement	Priority
FR6	The reviewer should be able to search for abstracts /titles of their trials automatically from abstracts databases, Cochrane library and Clinical registries.	High
FR7	The system should be able to provide search results from different sources and send it to the special program for processing search results.	High
FR8	The system should allow the reviewers to reuse the excluded abstracts to prevent duplication of effort, by providing search results with the status of their abstracts.	High
FR9	The system should be able to keep log of different search	High

	strategies used to obtain the search results and the actual results.	
FR10	The system should use filters for specific study designs(e.g. PubMed Clinical Queries filters , and Cochrane filter for RCTs)	Medium
FR11	System should be able to perform automatic de-duplication and classify abstracts by disease area.	High
FR12	The system should provide API for computer scientists to add or update the program for automating abstracts processes.	High

3.3.4 Abstracts Screening

ID	Requirement	Priority
FR13	Reviewer should be able to screen titles/abstracts and makes abstracts selections for the second screen.	High
FR14	The system should be able to keep log of selected abstracts and criteria used in the abstract screening.	High
FR15	The system should be able to provide GUI for multiple reviewers to work on the same abstracts.	High
FR16	The system should be able to group the abstracts screened by multiple reviewers and enable them to resolve their differences.	High
FR17	The system should be able to assign a unique number for the selected abstracts and group them with their database ID.	High

3.3.5 Full-text screening

ID	Requirement	Priority
FR18	The system should be able to initiate the process by providing lists of abstracts with their articles identified, which contain DOI or URL to access them.	High
FR19	The system should provide API for computer scientists to create algorithms for automating the processes of full-text screening.	High
FR20	Reviewer should be able to get full text of all articles identified and read them.	Medium
FR21	The system should provide GUI for multiple reviewers to work on the same articles to perform full text-screening.	High
FR22	Reviewer should be able to select included studies from the reviewed articles by using exclusion and inclusion criteria's	High
FR23	System should keep log of excluded studies with reasons for exclusions	High
FR24	The system automatically should combine the clinical trials selected from multiple reviewers by enabling them to resolves their differences.	High
FR25	The system should be able to assign a unique ID number of each clinical trials selected and group them with their registry ID ,DOI or URL of referred articles , abstracts ID 's and databases ID's.	High

3.3.6 Process of Included Clinical Trials.

ID	Requirement	Priority

FR26	Reviewer or computer scientists should be able to view the clinical trials with their relevant articles, papers or abstracts references.	High
FR27	The system should provide API for computer scientists to add algorithms which enable automatic processes for clinical trials.	High
FR28	Reviewer should be able to create the clinical trials manually by entering trials information identified in other ways (e.g. from Pharmaceutical companies).	High
FR29	Reviewer should be able to update the information on the selected clinical trials.	High
FR30	The system should be able to store all the clinical trials created or imported from different sources in a well-structured data format	High
FR31	The system should provide GUI for reviewer to export clinical trials to other external systems (Review Manager or ADDIS).	High
FR32	The system should create bibliography of the clinical trials and make it public and accessible to other reviewers.	High
FR33	System should be able to extract data automatically from the imported clinical trials or existing clinical trials.	low

3.4 Non-Functional Requirements (technical NFR)

We list the required non-functional requirements of the system with their priorities based on the stakeholder's concerns so we can know the behavior of the system we are going to design.

3.4.1 Interoperability

The system should be able to interface with legacy systems, other existing applications and external information system.

ID	Requirement	Priority
NFR1	The system should interact with literature databases, clinical registries or other sources to enable automatic literature searches.	High

3.4.2 Usability

The system should be easy to use for literature searches, screening search results, data extraction and data analysis.

ID	Requirement	Priority
NFR2	The system should provide the user interface which enable easy access of literature searches or clinical trials from different sources of evidence.	High

3.4.3 Scalability

The system should be able to support large databases of abstracts or clinical trials. Also it should be scalable as the number of users increases.

ID	Requirement	Priority

NFR3	~10- 100 million abstracts can be concurrently stored or accessed.	High
NFR4	~1- 10 million trials should be concurrently stored or accessed.	High
NFR5	The system should be scalable as the number of users increasing.	High

3.4.4 Re-usability

The CRS should enable the reuse of its software components within the system or with other external systems.

ID	Requirement	Priority
NFR6	Users or other external system should reuse the information stored in the database as independent component to support searching, screening activities and data analysis.	High
NFR7	The system should provide API for computer scientists to develop algorithms for automatic process of search results from different sources of evidence.	High

3.4.5 Integratability

The system components should be able to integrate with each other within the system or with other external systems.

ID	Requirement	Priority
NFR8	The CRS should be able to integrate with the abstracts databases or review databases for automatic search.	High
NFR9	The CRS should be able to import trials from clinicalTrial.com for data extractions.	Medium
NFR10	The CRS should be able to export trials to Review Manager or ADDIS for data analysis	High
NFR11	The software components should integrate with other applications or programs to support automatic processes.	High
NFR 12	The system should be compatible with the existing programs or packages which support automatic processes of abstracts and screening activities.	High

3.4.6 Availability

The system should be available at anytime, anywhere through the internet.

ID	Requirement	Priority
NFR13	Partial hardware or software should not result the failure of the system. To solve this problem fault tolerance mechanism should be applied, such as Ping/echo, heart beat and caching the exceptions	Medium

3.4.7 Security

The user information and data transaction within the system or with other external system should be secured.

ID	Requirement	Priority
NFR14	Secure login: Login credentials should always be transferred over an encrypted link.	High
NFR15	Single point of access: Internal system should be isolated from private network and only be accessible by user through CRS.	High

3.4.7 Performance.
The system should well response to meet user expectations and tolerance.

ID	Requirement	Priority
NFR16	Response times: The system should respond to any user action within 2 seconds. And this response should not be a transaction results but can be a response message like time out or error message.	Medium
NFR17	Data transaction performance: The system should perform transaction within 20 seconds and these transactions should be between two systems or client and the system.	Medium

4 Analysis

In this chapter we analyze different patterns that can be used in our system design based on the Pattern-Driven Architectural Partioning approach.

4.1 Pattern-Driven Architectural Partioning (PDAP) approach

The process for designing the software architecture is not a simple work, but it is a time consuming work and expensive(4). But the only way to simplify the act of design is reusing the existing system designs which are documented as software patterns. Software Patterns are architectural documents that express the relation between a certain contexts (preconditions for the use of the pattern), problem (consideration that may make the system difficulty to design) and solution to support the software engineering and best practices (4). Patterns are used to provide designs solutions to different design problems but those solutions are not always implemented in the same way. This is because, the provided solutions are generic and they are customized to the situation in which they are used (9).

The relation between the documented context, problem and solution statements of a certain patterns helps architects to understand the impacts of applying the solution in his/her system design (13). The applied solution from the patterns may have positive or negative impacts on quality attributes of the system. The quality attributes are used to show the behavior of the system and address the concerns of its stakeholders (4).

The main goal of software architecture is to design the system that meets the requirements of its stakeholders. In order to design such architecture, the architect should answer the following question in early phase of the architecture process (14).

1. How to partitions the system into modules that can be implemented
2. How to satisfy the functional requirements during this partioning
3. The influence of partitioning on the system's quality attributes

Using architectural patterns helps to answers those questions because the architectural patterns have visible partioning that address functional requirements and they describe the impacts of the partition on quality attributes of the system. Also, one architectural pattern can be the initial stage of decomposing the system into architectural elements that can be easily decomposed, implemented and allows changes during the architecting process.

In order to apply the solutions from different patterns in decomposing the system, then the

decomposition should balance both functional requirements and quality attribute of the system. To enable this, **Pattern-Driven Architectural Partioning approach** helps architects to partition the system and come out with the complete architecture that meets its requirements(14). We use this method to design the architecture of CRS in order to design the system that meets its requirements and quality attributes.

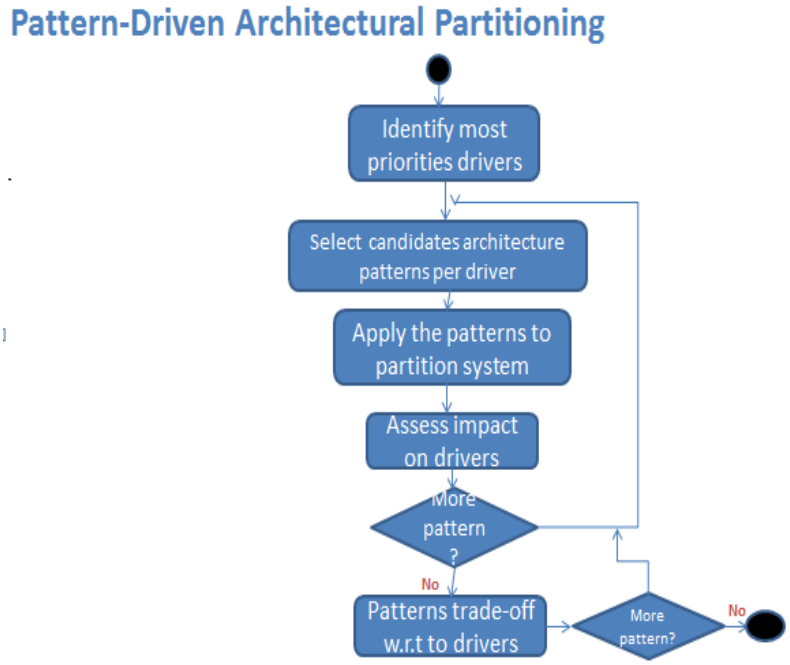


Figure 8: PDAP(18)

The figure 8 shows that Pattern-Driven Architectural Partioning has several steps. The first step is to identify the most prominent architectural drivers. To perform that step, consider both functional requirements and key drivers based on the stakeholder's concerns.

The second step is to select the candidate patterns per driver. In this step the architects should identify the architectural patterns relative to design problems. For every identified pattern, examine the context, problem and solution statements and map them with the existing problem designs. Then find out how that pattern matches with their key drivers.

The next step, as shown above is to apply the solution from the candidate's patterns or combination of candidate pattern and decompose/partition the system based on the available solutions.

After partioning the system, then the next step is to evaluate the impacts of the solution on the key drivers. If the solution has a negative impact on the system design then repeat step 2 above. This process can be repeated if another architectural driver needs to be considered or if another

patterns need to be added to solve the design problems or trade-off issues. If the solutions have positive impact on system design, then decomposition should satisfy both functional requirements and key drivers.

Then, the last step is to verify the entire architecture and find out how every patterns addresses the quality attributes (key drivers).If more patterns are needed in our architecture then repeat step 2 above and If there is no any changes on the designed architecture then the architect end the process.

4.2 How to apply this method in our system design?

We use this method by applying all the steps mentioned above to design our software architecture. First, we identify the **Usability**, **Interoperability**, **Re-usability**, **Integribility** and **scalability**. And we consider these drivers as the most prominent drivers in designing our system based on the stakeholder's concerns (refer section 3.1). In the second step, we identify patterns and find out how that patterns match with our key drivers.

Then, we apply the solution from identified patterns and decompose our system based on the proposed solution. Thereafter, we assess every solution and find out how those solutions have impact to our key drivers. So we decompose our system by adding more patterns to solve the existing design problems.

Refer to chapter 3 and our architectural vision that we need to develop the Clinical Review System that can support all the steps of systematic review in a single point of view. In order to meet that vision, the designed architecture of CRS should have front-end server which is a web-based system that provides a unified interface to interact with multiple users and other external systems. The front-end server of CRS should be able to handle multiple requests from multiple users and enables data accessibility within the system or with other external data sources.

To design such architecture, we identify several patterns from architectural books or articles (4),(9),(23) that can solves our design problems. We apply the solutions from those patterns to decompose our system based on PDAP and capture all the architectural decision we made in designing our architecture. Architectural decisions are the series of decisions performed during the process of architectural design and shows how those decisions have impacts on system design positively or negatively (13). In the next sub-section we present all the decisions we made from identified patterns in a tabular format based on the architectural decision documentation (13).

4.2.1 AD_1 Model-View-Controller Pattern(MVC)

Refer to PDAP, we consider usability as highest priority key driver in designing our system and we select Model-View Controller pattern as the solution that can address that driver. After examine the solution from this pattern, we apply it to decompose the front-end server of CRS. The front-end server decomposed into three components: Model, View and Controller. View and controller are responsible to handle the user interface of the system and the Model is responsible to handle the core functionalities of the system. View can initiate the event to the controller and the controller receives the request and changes it to user action that is understandable by the model. The Controller used to affect changes in the Model and model processes the query and returns the result back to the controller that can be displayed in the view component (4). Finally, we assess this solution and documents the consequences of applying this solution in decomposing our system (see table 3 below).

Table 3: Architecture decision for decomposing the entire system

Issue	We need the interactive system to enable flexible user interfaces.
Decision	We decide to MVC because it separates the core functionalities of the system with the user interfaces. MVC divides an interactive application into three areas, Model responsible for encapsulates core data and functionality, view responsible for display information to the user and Controller responsible for receiving input from the user.
Status	Decided
Group	Interactive application
Assumptions	It is assumed that the CRS use change propagation mechanisms to supports loosely coupling and reduces dependencies between model, view and controller.
Constraints	The MVC should : • Provide easy change of user interface even possible at run time. • Support different “look” and “feel” standards or porting the user interface without affecting the code in the core of the application. • Presenting the same information in different ways.
Position	1.Model- View –Controller 2. Presentation- Abstraction- Control(PAC)
Argument	(1) MVC ensure the usability of the system because it provides clear separation of core functionality and its user interface. Controller and View components are responsible for the user interface and Model component is responsible for the core functionalities of the applications (4). Also MVC enable the reusability of the system because multiple Views can share the same data or functionalities from the model.

	And when data model changes no need to change all their views but change propagation mechanism can be used to notify all Views about the changes. (2) PAC support mult-tasking by providing the separation of concerns in the system but it increases the system complexity due to inter-communication between components(4)
Implications	• Using view and controller increase the complexity of the system because it is difficult to separate the concerns between Controller and view since they have one to one strong relationship which may hinders the re-usability of the view and controller components (14). • There is Inefficiency of data access in view because multiple views can access the model at the same time (4).
Related decisions	AD_2,AD_3
Related Requirements	FR17, FR23,FR NFR 4, NFR 5
Notes	

4.2.2 AD_2 Layers (Relaxed layer pattern)

As shown in the figure 9, the entire system decomposed into three parts, but after assessing this decomposition (refer PDAP approach) we found that we still need to add more patterns in our system designs.

We need decompose our front-end server with flexible user interfaces because the Model-View Controller does not address well the re-usability of view and controller as independent components (4). Due to that, we add layers pattern in our system design to solve the design problems of the Model-View Controller.

This pattern helps to structure application into group of subtasks in which each group is a set at particular level of abstraction. We decompose our system into different layers which can be used by different users as independent layers. Where, each layer should assign its task and specify the services they are going to provide. Also each layer should specify an interface to interact with other layers. And these layers should be loosely coupled to each other to ensure re-usability and integratability between layers (4).

The figure 9 shows that, we decompose our system into 4 layers such as view layer, controller layer, integration layer and Data access layer. And these layers should provide an interface to interact with other sub-system in order to fulfill the system requirements. Also this figure shows that, the multiple front-end servers can access the data from the back-end servers and external

data sources through the Data access layer.

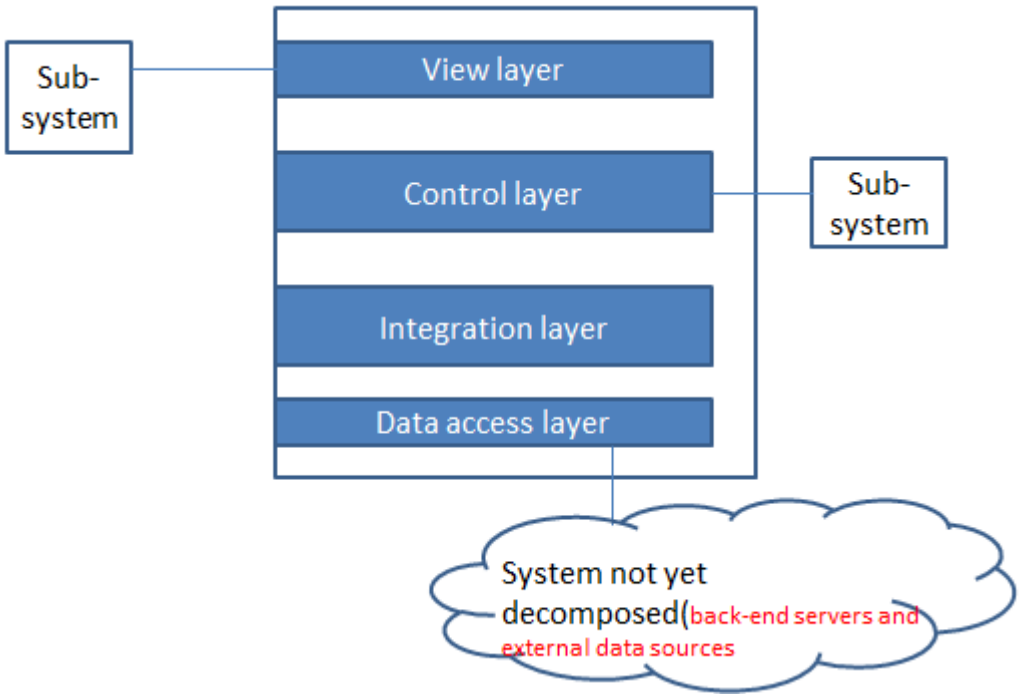


Figure 9: layers pattern

Table 4: Architectural decision for decomposition of front-end servers

Issue	We need to decompose the front –end server of CRS into independent layers in order to solves the design problem of Model-View Controller pattern
Decision	We decide to use Relaxed layer pattern to perform the high level decomposition of the system which enable the system to decouple its functionality into independent components.
Status	Decided
Group	Request control
Assumptions	The system has complex components needs to be decomposed consider view and controller components
Constraints	The Relaxed layer we use provide: • View layer: used to handle multiple requests from clients or subsystems. • Control layer: is concerned with the business process for screening the search results and perform the automatic processes. • Integration layer: provides the environment for system to enable processes integration or data integration • Data Access layer: provides the data access to the users and automatic clients from the system or from other external data

	sources.
Position	Similar to the layers pattern
Argument	We use relaxed layer pattern because it allow the reuse of the individual layers in different context which enable the reusability of system components within the system or with other external system such as RevMan/ADDIS, Abstracts databases and others. As a result it provides clear separation of concerns (4). Also it ensures integratability of the system which allows different users to use products from different vendors in different layers (4).
Implications	The consequence of using layers pattern is that it is not good for high performance because the relevant data must be transferred through a number of intermediate layers (4).
Related decisions	AD_1, AD_3,AD_4
Related Requirements	FR6,FR7, FR12, FR13, FR17,FR21,FR23,NFR 10,NFR 11
Notes	

4.2.3 AD_3 Dispatch View Pattern

We need to decompose the view layer of front-end server in order to ensure more the usability of the system. This is because, the system should be able control the accessibility of multiples views to the core functionalities of the system. Also, we need to ensure the clear separation between the views and controllers concerns. So we select the dispatch view pattern to solve the design problems in the view layer.

This pattern is responsible for view management, navigation, also it generate a view based on the template and dynamic model. This pattern used to handle multiple views of the system and control the requests from the clients to the appropriate views (19).

The figure 10 shows how the solution from this pattern can be applied in decomposing the view layer. The given solution decomposes the view layer into two components dispatch view component and view component. Dispatch view component used to receive the requests from the different subsystem and find the appropriate view of the requests. This component used to provides all views of the system in a single point of access in the view component. The View component is responsible to find the appropriate controller for every request from different sub-system such as GUI client (19).

The table 5 shows the architectural decision in decomposing the view layer and show how that decomposition impacts the key drivers of our system.

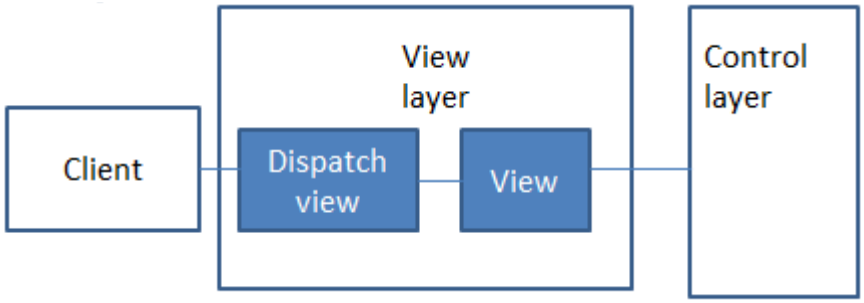


Figure 10: Dispatch view Pattern

Table 5: Architectural decision for decomposition of view layer

Issue	CRS support multiple views for users to access specific data in the model.
Decision	We decide to use dispatch view pattern in order to coordinate the accessibility of multiple views to the model because there is no centralized component for managing access control, content retrieval or view management. And there is duplicate control code scattered through the various views.
Status	Decided
Group	View Management
Assumptions	The CRS use an efficient update mechanism for propagating changes between views. The CRS is replicated to various regions to enable easy accessibility of the system.
Constraints	This pattern has to be used in the front-end servers to control multiple views from the clients.
Position	1.Dispatch View Pattern 2.Service To worker Pattern
Argument	Dispatch view pattern and Service To worker pattern are identical with respect to their components but they differ in division of labor among their components. (1) The dispatch view pattern provides the limited role of view management and it applies when no outside resources is utilized to choose the view. It is relevant to our problem domain because the clients can submit the request directly to a controller with a query that describes an action to be completed (19) . (2) Service To Worker pattern is more sophisticated than dispatch view pattern because it can invoke business services to determine the appropriate view to display. We did not use this pattern because it is not relevant to our domain problem since this pattern is design more in the service oriented domain(1)
Implications	The requests are handled in a central place to control the multiple views to the clients. As a result, it ensures usability because it is easier for the system to manage multiple views of response of different requests form

	the clients. Also the dispatch view Pattern improves separation of concerns because it reduces dependences that may occur between views and the controllers. As a result, it ensures re-usability of the view and controller because the control code for every view of our system can be accessed from single point of access.
Related decisions	AD_1, AD_2
Related requirements	FR17, FR23,FR NFR 4, NFR 5

4.2.4 AD_4 Broker (Trader System) Pattern

We need to add more patterns in our system design (refer PDAP approach) in order to handle accessibility of data from multiple front-end servers to the back-end data servers or external data sources. Also we consider the interoperability as the second priority in designing our software architecture. So, we assume that our front-end servers are replicated to various regions and we need to find out how multiple front-end servers can access the data in the system and from external data sources in a single point of view. In this case we need to decompose our data access layer by adding more patterns which can solve the designs problems on data access.

We decide to use Broker (Trader system) in the data access layer (refer AD_2) in order to provide an environment for multiple front-end servers to access the data in the back-end servers and external data sources (4).This variant used to provide the location of every data servers to the front -end server. In which the front end servers access the data in the back end servers through the services registered on the broker (trader). This pattern has a control mechanism to handle multiple requests from front- end server to the available data –server.

This broker pattern registers the data servers or external data sources as a service by using dynamic registration mechanism. So, it provides an API for the data servers and other heterogeneous systems to interact with the existing system. This is because, they register the data servers and external data sources (heterogeneous systems) as the service that can be accessible by multiple front-end servers remotely (4).

Figure 11 shows that, broker (trader) pattern provides an API for multiple front-end servers (the blue box) to interact with back-end data servers and external data sources which are not yet decomposed.

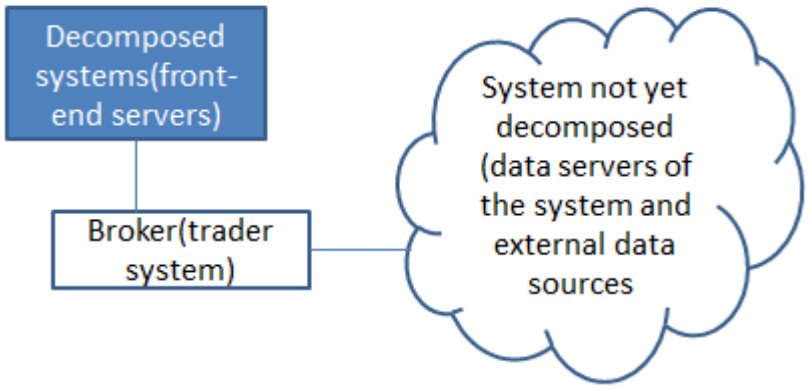


Figure 11: Broker (Trader system) Pattern

Table 6: Architectural decision for decomposition of data access layer

Issue	The CRS needs to interact with different heterogeneous systems and multiple data servers containing abstracts and clinical trials data.
Decision	Broker provides all services from different sources in a single point of view.
Status	Decided
Group	Server communications
Assumptions	CRS have more than one broker in case one broker fail the requested is forwarded to the next broker. Also it is assumed that the CRS has replicated into various regions.
Constraints	• This variant connect front-end servers with back-end servers of CRS, since back-end servers are replicated but also this variant provide the location of those servers in a transparent way. Also this variant can register service from other external systems. • The front-end servers control the request from the client. • The back-end servers control the data servers • This variant provides the control of the inter-communication between these servers.
Position	1.Trader system 2. Re-use of look-up broker (refer AD_7)
Argument	(1) Enable interoperability between CRS and heterogeneous systems .Also it allows scalability of the CRS because if the CRS need to add another server then server registers itself to the Broker. Also it provides the control layer services between servers, in which it protects the system from fake servers because all servers are authenticated here. It provides the data access from data servers any time or anywhere. Also it facilitates the reusability because the system can use the

	service from other external system to fulfill its requirement, such as searching literature in abstracts databases, clinical trial registries and other databases. (2) We did not re-use lookup broker (Refer AD_7) because it not secure because client will be in same level with server.
Implications	Since broker used as single point of data access and then if it is hacked the system will fail, thus the brokers need to be duplicated in order to ensure the availability of the data in the system.
Related decisions	AD_2,AD_5,6
Related Requirements	FR7,FR8,NFR1-3
Notes	

4.2.5 AD_5 Information Aggregation Pattern, Search adapter variation

We decide to add Information Aggregation Pattern (search adapter) pattern in our system design (25) in order to decompose the external data sources and ensure accessibility of data from multiple sources in a single point of view.

This pattern is the variant of runtime pattern which used to provide accessibility of unstructured data. This ability to access unstructured data provide the search results with a reasonable response time because the search solution is implemented via search adapter node (25).

The search adapter node can interact with external data sources though the logic interface or via product API. As shown in the Figure 12, the search adapter node receive the requests, formulate the query or analyze the request and sends it to the external data sources by using brokering capabilities. Then the search adapter return the results from external data sources, aggregate and normalize the search results and send it back to the clients. However, the search results are formatted before sending it back to the clients that request the original search (25).

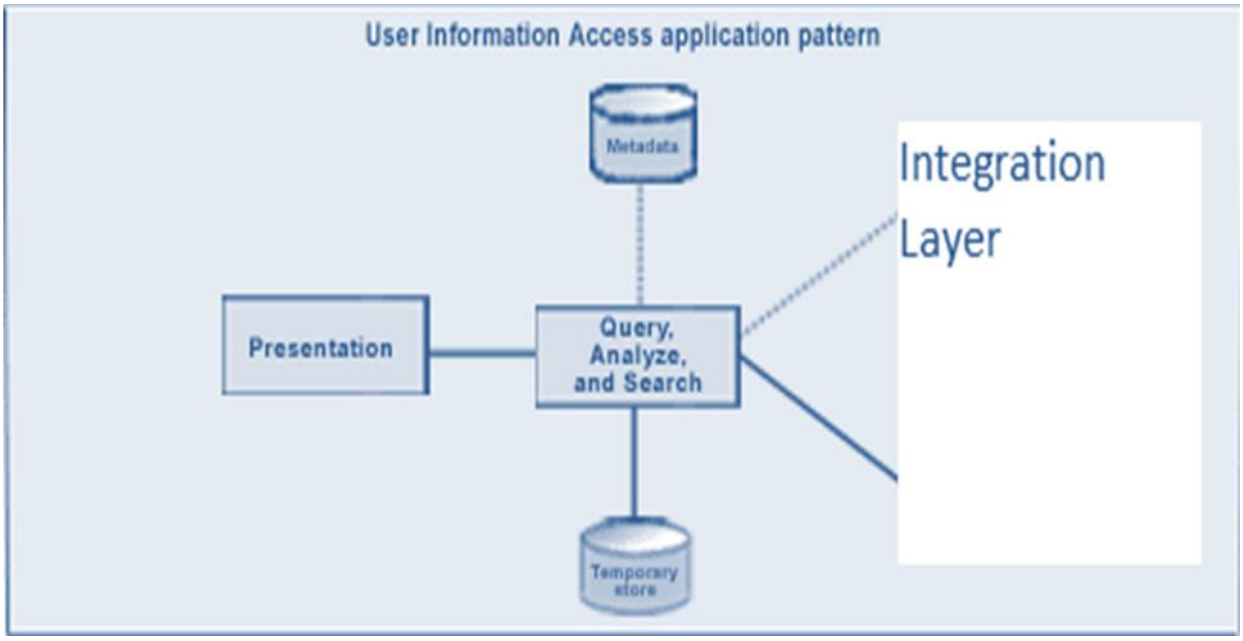


Figure 12: source from http://www.ibm.com/developerworks/patterns/bi/at7-runtime.html

So, we apply the solution from this pattern in our system design. The solution from this pattern provides the accessibility of data from external data sources such as databases of abstracts or Clinical Trial registries through the search adapter.

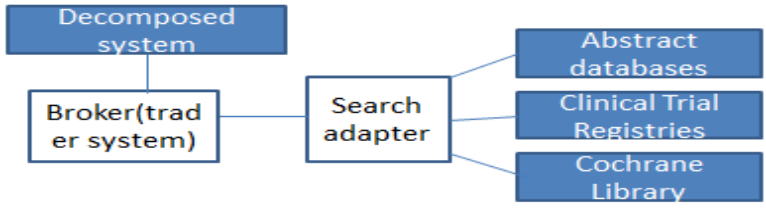


Figure 13: decomposition of external data sources

The figure 13 shows that, the decomposed system such as front-end servers can send the requests to the external data sources through the broker (trader). The broker provides the accessibility of data from different external data sources in a single point of view (search adapter component). The search adapter component used to search, aggregate the search results from different sources and makes it available to decomposed system as they come from single virtual data source. This decomposition shows that, the search adapter can integrate with external data sources by re-writing the requests before sending them to external data sources and also this adapter re-write the search results from external data sources before sending the results to the decomposed

system. The table 7 shows all the designs decisions used to decompose the external databases and how this solution has impacts to our key drivers.

Table 7: Architectural decision for decomposition of external data sources (integrated databases)

Issue	CRS need to perform automatic searches from multiple sources by using brokering capabilities. CRS should be able to combine the search results from multiple sources in a reasonable response time.
Decision	We decide to use this variation because it enables mult-source brokering capabilities via the inclusions of "search adapters" nodes.
Status	Decided
Group	External sources control
Assumptions	The CRS perform the searching activities by brokering those queries and send them to the multiple data sources through the integration layer. Every multiple data sources have search adapters which contain the logic to interface with external data sources. The search adapters then return search result from each individual data index in which the search brokering node aggregate and normalize the search results so that they appear to be from one "virtual source". Then the aggregated results are sent back to the presentation node/web application server node in which results are formatted and send back to the client who send the original search.
Constraints	This pattern act as a broker by sending the search queries to the multiple data sources.
Position	Data integration
Argument	This pattern enables CRS to perform an automatic search by providing search results in a single point of view. Also, this pattern enable user to access multiple data sources within a short response time.
Implications	This pattern provide the search results in a read-only format however it is possible to perform the write access in the temporary storage. As a result, this pattern addresses usability because it provides accessibility of data from different sources in a single point of view. Also it ensures well the interoperability because it uses Broker pattern to register all data sources as services and control them. This decomposition also addresses well the integratability because it allow data transformation before sending or receiving data from those sources.
Related decisions	AD_4
Related Requirements	FR 6-8, FR36-39, NFR 1-3,NFR 10-11
Notes	

4.2.6 AD_6 Shared Repository

We need to decompose the back-end data servers in order to ensure the accessibility of data in a single point of view. We assume that, data are stored in shared memory which can be accessed with multiple users. So, we select Shared repository pattern to decompose the data servers by examining the its problem and solution statements and find out if this pattern address our key drivers. This pattern is used when the software made several software components which need to communicate and exchange data in a single point of access(22).

The figure 14 show the solution provided by this pattern by using the data repository for communication. The repository are known and used by all software components in the system but those components they don't know each other. Component-I produce product V in the repository but component -2 and component -3 use it without knowing who deliver that product. And component-1 delivers its product without knowing the users of that product. So this pattern provides an environment for different software components to share the data in the repository (22).

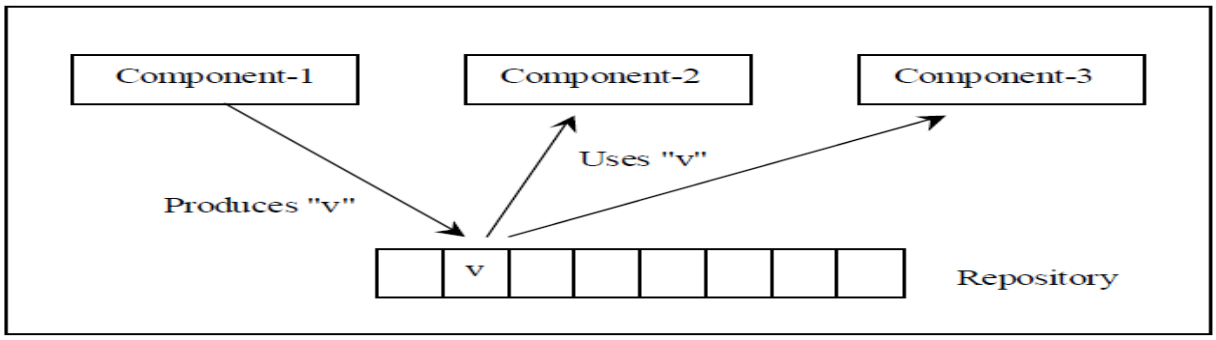


Figure 14: Shared repository pattern

So, we apply the solution from this pattern by providing the data from the shared memory in the central repository as an independent component which can be easily reused or integrated to each other (22). Figure 15 shows that our data repository divided into three repositories, user repository responsible for handling user information, Abstracts repository responsible to handle the selected abstracts and clinical trials are used to handle the clinical trials after full-text screening. Then, the front-end servers can access the data in those repositories through the broker (trader). The broker (trader) registers all repositories and provides their locations to the front-end servers. The front-end servers access the data from different repository in our system through the single point of access of Broker (trader). So, we document the architectural decisions used in

decomposing the back-end data servers of our system as shown from table 8. Also this table shows the impacts of applying this decision in our design compared to different alternatives solutions.

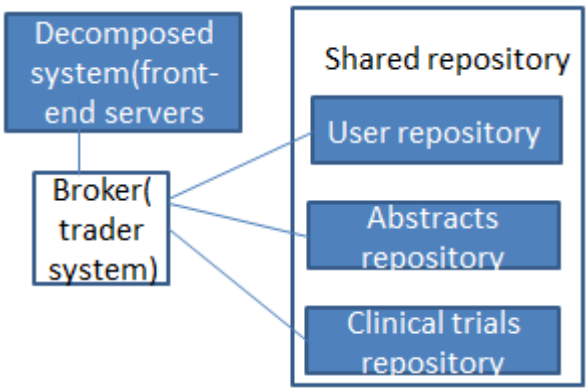


Figure 15: Decomposition of back-end data servers

Table 8: Architectural decision for decomposition of back-end data servers

Issue	The system needs to provide access of data in a single point of view. This means, the system needs to provide the clinical trials information in a structured database in which different users or other systems can access the same data.
Decision	We decide to use shared repository pattern to enable communication of software components based on the shared repository.
Status	Decided
Group	Data sharing
Assumptions	• Repositories implement the fault tolerance mechanism to increase the availability of the data. • We assume that in every local repository data is transferred in a reliable way, fault tolerance and load balancing. • The CRS have front-end servers for handling client request and back-end servers for processing the request and storing the data. So CRS provide interface for front-end servers and back-end servers to enable data sharing between clients and CRS.
Constraints	The shared repository can be used: • When user or subsystems needs to access the information in the user information database. • When reviewer, subsystems or computer scientists need to access the abstracts or trials information for literature searching or screening activities.
Position	1.Shared Repository

	2.Active Repository 3.Blackboard Repository
Argument	(1) It allows the reviewers to access the clinical information from single point of view which increases efficiency in performing the systematic review. This pattern enables many users or other subsystem to share the same data in the databases, as a results it ensures scalability and reusability of clinical data components in the repository. (2) We did not choose Active repository because it does not deliver timely information when user perform query in their repositories (4). (3) Blackboard use control component to control the repository but this is not applied in our system since the actions are user – driven (4).
Implications	Since Broker (trader system) knows all location of servers then clients do not need to know where the servers are located but broker can initiate the global search in their repositories. If the CRS grows and need to add more servers then they have to be registered to broker as well. As results with the help of Trader broker, it supports scalability by allowing many application servers to have access with the same data.
Related decisions	AD_4
Related Requirements	FR2,FR8,FR19,FR32,NFR7,NFR8,NFR9
Notes	

4.2.7 AD_7 Look-up Broker

We decide to add another pattern in our front-end servers in order to handle the multiple requests from the views because the system can be overloaded if the number of users increases. So, we select the Look-up broker to handle multiple requests from different users of the system and provide the accessibility of controllers in a single point of view.

Look-up broker is a variant of the Broker pattern that uses a look-up object to access the remote objects in a distributed environment .The look –up broker is responsible for receiving the requests from multiple clients for registering remote objects and control the communication between clients and remote objects (4). The table 9 below shows the architectural decision for this pattern and how that decision has impact on our system design.

Table 9: Architectural decision for handling multiple requests from users of the system

Issue	CRS needs to be accessed by multiple users over the network.
Decision	We use Look-up Broker in order to handle multiple client requests.

Status	decided
Group	Client communication
Assumptions	CRS is provided through the internet and user or subsystem can use the system as distributed system. CRS have many replicated servers to handle client requests.
Constraints	Look-up broker decompose the system as the thin browser client to handle multiple request from the clients and also it used as load balancer which provide accessibility of core functionalities of the CRS through the single point of view.
Position	1.Lookup broker 2. Client-server
Argument	(1) We decide to use Lookup broker because it support large number of users to access the CRS at the same time. This means it allows many users with many request to access the CRS. Though it has great performance bottleneck but it ensures good scalability. (2) We did not choose client- server pattern because it does not support scalability since the system will be overloaded when multiple users sends their requests to the server.
Implications	Look-up broker has impact the performance of the system, because the client does not have access with the server directly.
Related decisions	AD_1, AD_2
Related Requirements	FR6, FR7, FR22, FR24, FR28, FR30, FR 31, NFR8.
Notes	

4.2.8 AD_8 Façade Pattern

We decide to add another pattern on the decomposition of our back-end data servers in order to hide the complexities of data structure in repositories. So, we select Façade pattern and apply the solution from this pattern to decompose more our repositories (9).

The façade pattern is used to provide a unified interface to a set of interfaces in a sub-system. It defines a higher-level interface that makes the subsystem easier to use (9). As shown in table 10, we document the architectural decision of applying this pattern and shows how this decision have impacts to our system designs.

Table 10: Architectural decision for decomposing more the back-end data servers

Issue	We need to design the interactive system with high-level interface that makes the subsystem easy to use.
Decision	We decide to use the Façade pattern in order to hide the complexity of the system.
Status	Decided
Group	Interactive application
Assumptions	we need this pattern to hide the complexities of data structure in the shared repositories

Constraints	Façade pattern used when: User need simple user interface to access the data from the complex system (9).
Position	Similar to Mediator pattern.
Argument	We use Façade to presents a specialized interface to the client that makes it easier to use, as a result it increase the usability of the system.
Implications	Façade used to hide the complexities of data repositories, due to that, it ensure the usability of our system
Related decisions	AD_8
Related Requirements	FR17, FR23,FR NFR 4, NFR 5
Notes	

4.2.9 AD_9 Wrapper Pattern

We need to add an interface in view layer to handle all the requests from external systems. So, we select Wrapper pattern to decompose such interface. This pattern converts the interface of a class into another interface that client expect. This pattern helps multiple classes to work together despite of their incompatible interfaces (9).The table 12 below shows the architectural decision used in applying this pattern in designing such interface.

Table 12:Architectural decision for decomposition of system's interface for other external system to export the data from existing system.

Issue	CRS has to interact with other external systems or heterogeneous components in order to export clinical trials for data extraction and analysis.
Decision	We decide to use Wrapper pattern because we want to create a reusable class that cooperate with the classes which don't have related or compatible interfaces.
Status	decided
Group	Communication
Assumptions	• It is assumed that our system can interact with other existing system for systematic review such as RevMan/ADDIS
Constraints	The wrapper pattern provides: • Communication with other existing system such as Review Manager or ADDIS
Position	Similar to adapter pattern and Bridge pattern(9)
Argument	This pattern enables the reuse of system components within CRS and with other external system because it converts the components with incompatible interface into the interface the client object expect.

Implications	As a result CRS will increase its re-usability and integribility because the CRS can export data to other external systems without changing their core functionalities.
Related decisions	AD_2,AD_1
Related Requirements	FR 6-8, FR13, FR36-39, NFR 1-3,NFR 10-11
Notes	

Therefore, after identifying the relevant patterns in our design, we create the complete initial model of our architecture as shown in the next chapter. In that chapter we combine all the identified patterns and come out with the complete software architecture that addresses all the system requirements. According to that, we decide to continue with the last step of our architecting process. To perform the last step of the mentioned method above (refer PDAP), we verify the complete architecture by assessing every pattern used in our design and find out how every pattern addresses our key drivers. Due to that, we use chapter 6 to show this step and finally we end our architecture process.

5 Software Architecture

The design of software architecture is the third stage in the software development life cycle. And the key activity of software architecture is the decomposition of the system into architectural elements that fulfill the system requirements and the stakeholder concerns (4).

However, there various design methods for software architecture such as object-oriented Framework, which design the software architecture by combining collections of classes and objects used to create the functionalities of the system. But this method does not always deliver the quality software architecture because their designs solutions do not balance the functional requirements and quality attributes of the system. Other methods are ADD (Attribute-Driven Design) but this method designs the software architecture by decomposing the system based on quality attributes. But this method does not address the impact of patterns with respect to functional requirements and quality attribute of the system (14).

One of the proposed approach for architecting a software system while emphasizing both functional requirements and quality attribute is Pattern-Driven Architectural Partitioning (PDAP)- (refer chapter 4). We decompose our system designs based on the patterns, but the patterns are not isolated solutions; they should integrate with other patterns and the domain environment to design the complete software architecture.

In this section we describe how these patterns integrated to each other in designing the quality architecture that balances both functional requirements and quality attributes. We also elaborate the patterns used to design the complete architecture and show how that architecture integrated with our domain environment. Thereafter, we present the sequence diagrams to show how the objects can interact in the system to meet the system requirements.

5.1 Initial Model

We decompose our system based on the Pattern-Driven Architectural partitioning method by combining all the patterns according to the priorities of our key drivers(refer chapter 4). First we decompose our system based on the Model-View Controller pattern, (refer AD_1). Then, we use Layers pattern to decompose the front-end server of CRS (refer AD_2).We use look-up broker to

handle the multiple requests from different users in accessing the core functionalities of the system in the control layer as shown in figure 16 (refer AD_7).

In the data access layer, we use shared repository pattern to store data in the CRS and Broker (trader) to control the accessibility of data in those repositories and other external data sources through the search adapter components(refer AD_4,5,6).

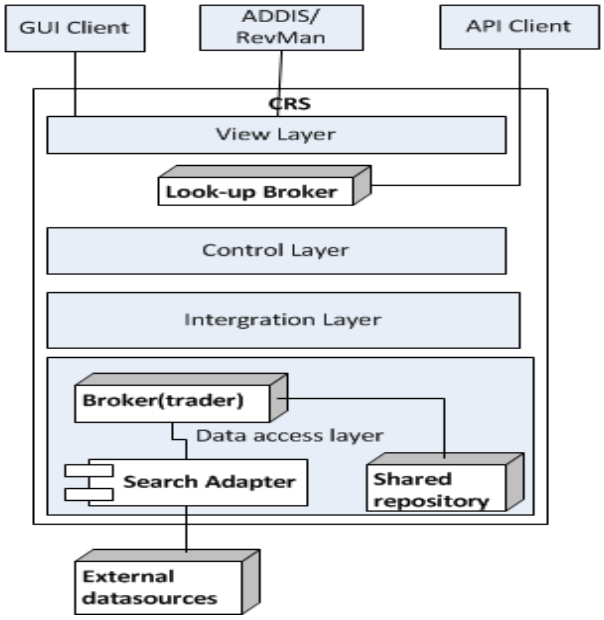


Figure 16: Initial Model

5.2 Description of the entire system

We describe the entire system by partioning the initial model into three parts; front end part, application part and back-end part.

The front –end part of the system is responsible for handling the requests from different users of the system. As shown in the figure 17, the GUI client, ADDIS/RevMan can access the CRS through the view layer. Also in this part, we use the look-up broker pattern to handle multiple requests from the GUI client and other users of the system. The look-up broker provides the API for other special programs to interact with CRS to perform the automatic processes (refer AD_7).

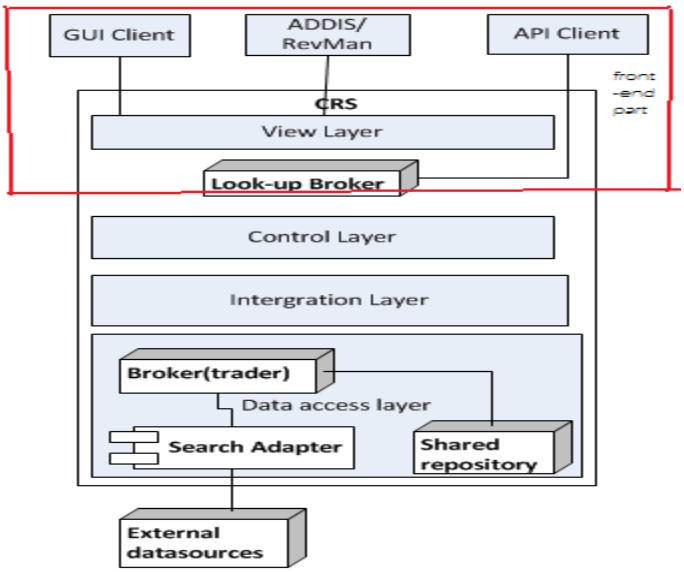


Figure 17: Part 1- Front-end part

The Figure 18 shows the application part of our system, which is responsible for processing the requests from the clients and processing the search results from back-end data servers. In this part, we use the look-up broker to provide the accessibility of controllers of the system in a single point of view (refer AD_2). Also the look-up broker provides the API for computer scientists to automate some of the processes in the control layer.

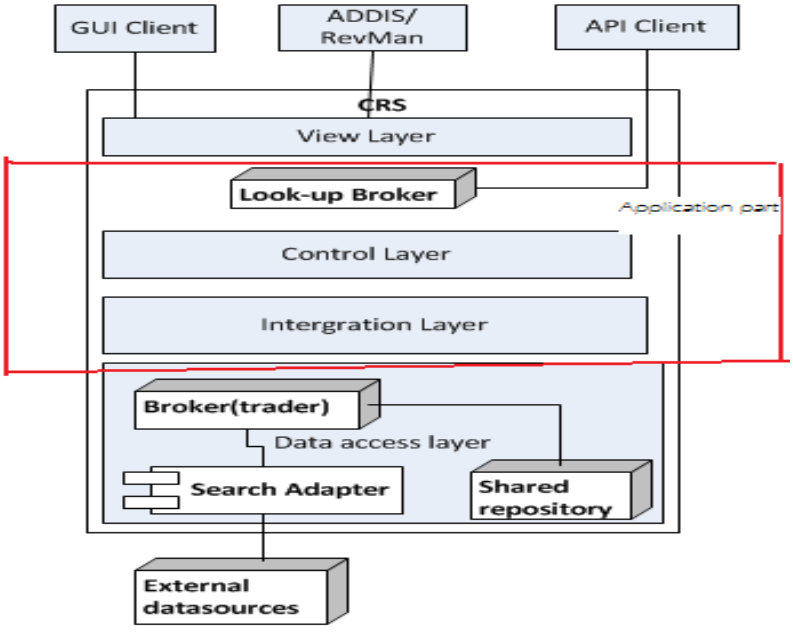


Figure 18: Application part

The backend of our system design is responsible for handling data through data access layer as shown in the figure 19. The data in the CRS are stored in the shared repository as independent components to enable data sharing with multiple clients. To design this part we use the shared repository pattern (refer AD_6). Also, the back-end of our system is responsible to perform automatic literature searches from different sources through the broker (trader). This part, enables the accessibility of data from shared repositories (refer AD_6) and other external sources through the search adapter pattern (refer AD_5) in a single point of view.

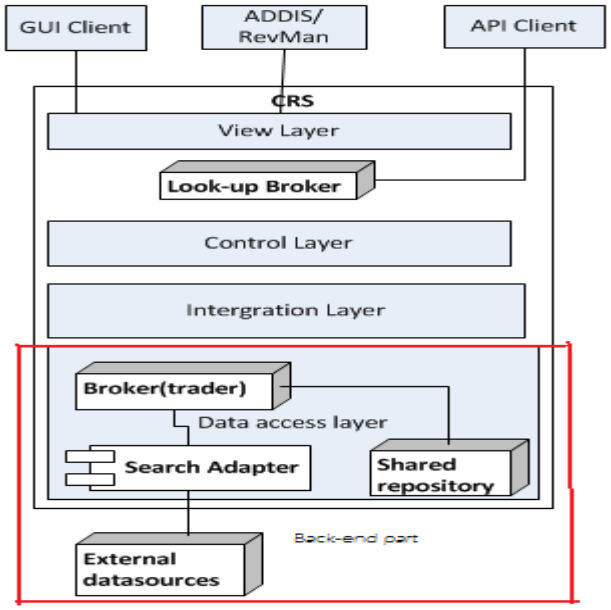


Figure 19: Back-end part

Importance to achieve our key drivers.

Usability- This partition, ensures the usability of the system because the CRS provides the interface for GUI clients and other external users to access the core functionalities of the system through the view layer. Also this decomposition provides the role of separations between user interfaces of the system in the view layer and core functionalities in other layers (refer AD_1, AD_2).

Interoperability- The broker (trader) and search adapter ensures the interoperability of the system because it allows the system to interact with other heterogeneous external data sources such as databases of abstracts or clinical trial registries.

Re-usability- The layered decomposition of front-end server allow the front-end part to receive requests from users and send it to the application part. Application part processes the request and

sends it to the back-end part. Back-end part access the data and send results to the application part that used to process results and send them to the user through the front-end part. Also, the layered decomposition allows the system components to be reused within the system by automated clients such as special programs or with other external system such as RevMan/ADDIS. The use of shared repository pattern to decompose the back-end data server ensures the re-usability of system components because this pattern allow the data to be reused within the system or with other external system (refer AD_2, 6).

Integratability- The layered decomposition allows the different layers in the system to integrate to each other to meet the system requirements (refer AD_2).

Scalability- The application part ensures the scalability of the system because if the system add new controller then it has to register itself to the look-up broker. And also the look-up broker can handle multiple requests from multiple users. Also the back-end part enables good scalability of the system because it is easier to add new data server in the broker (trader system).

5.3 Elaborated model with patterns

In this section we describe the patterns in detail and we modify our initial model in order to satisfy our system requirements and key drivers (refer chapter 3).

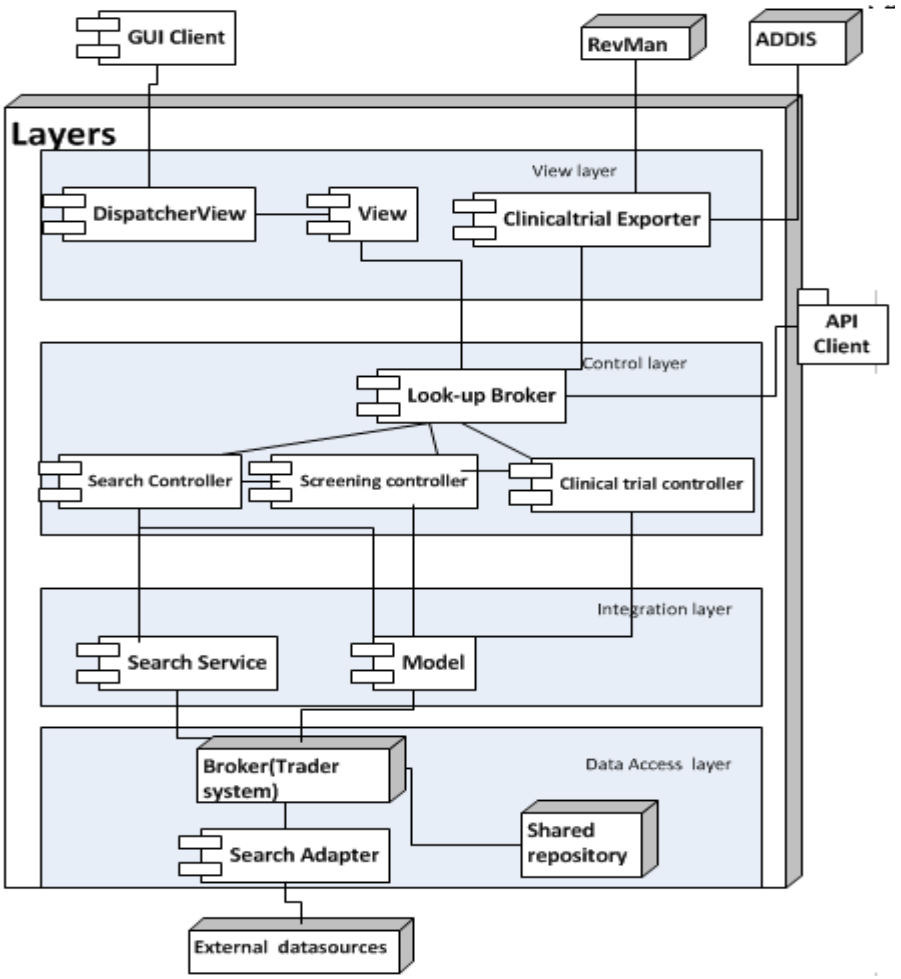


Figure 20: Elaborated model of software architecture

The designed architecture of CRS allows multiple users to interact with the system through the view layer. Figure 20 shows that, the GUI Client can interact with the system through the Dispatch view component (refer AD_3) that used to provide the appropriate view for every request from the clients. Also this architecture provides an interface for RevMan/ADDIS to interact with the system through the clinical trial exporter interface in the view layer (refer AD_9). The view layer and other automated clients can send their requests to the look-up broker that used to handle multiple requests and provide them the appropriate controllers to every request.

We decide to divide the core functionalities of CRS into three controllers such as Search controller, screening controller and clinical trial controller as shown in the control layer. The search controller component is responsible for processing all the search requests from the users and other external system (refer section 3.3.3). The screening controller component is

responsible to perform the screening activities such as abstracts screening and full-text screening (refer section 3.3.4 and 3.3.5). And the clinical trial controller component is responsible for processing the requests for clinical trials such as exporting clinical trials or any query related to the clinical trials stored in the system(refer section 3.3.6).

Also, the designed architecture uses the integration layer to integrate all the controllers with the model component. Model component enables the core functionalities of the system to access the data in the back-end data servers in a single point of access. Also, this architecture provide search service component in the integration layer to handle all the search service in our system and other external data sources such as abstract databases or clinical trial registries.

As shown in the Figure 20 the last layer of our system is data access layer which contains the broker (trader system) to enable accessibility of data from different data repositories and external data sources. Also it uses search adapter component to access the data from external data sources (refer AD_4, 5, 6).

5.3.1 Broker (Trader system) Pattern

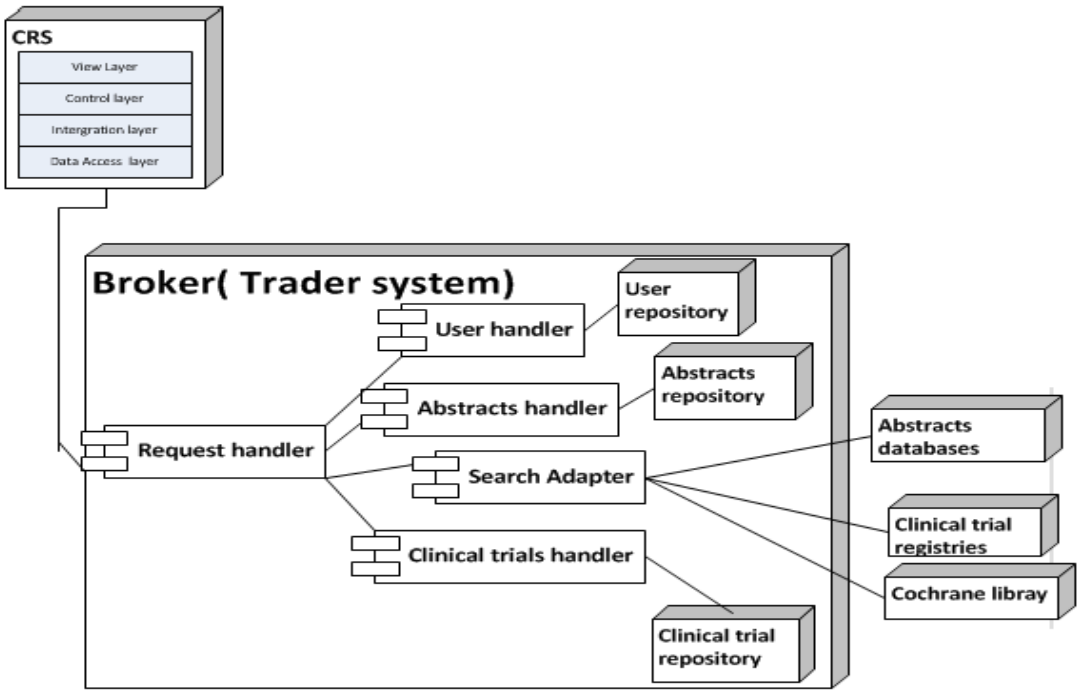


Figure 21: Broker (Trader system) Pattern

Broker pattern (Trader system) is designed to provide the communication layer between the front-end server and the data servers. This pattern can handle multiple requests from front-end server to different repositories and other external systems (refer AD_4). Front-end server

registers its services to the broker (trader system) and these services can be accessed by other replicated front-end servers. For example the front-end server register its search service in the broker and this service can be accessed by multiple front-end servers. As shown in the Figure 21, this pattern use request handler component to handle multiple requests from different front-end servers, and this component allow accessibility of the data in repositories through their handler components. Also, the broker (trader) pattern use the search adapter component to perform the search service from the external data sources such as Abstract databases, clinical trial registries and Cochrane library.

In order to avoid single point of failure of our system, we will use two brokers (trader system) if one broker fail the system will re-direct the requests to the back-up broker. Also the new broker can register to the existing broker, and all the services can be replicated to the new broker.

This pattern ensures interoperability (refer section 3.4.1) because it allows the front-end server to perform searching activities to different heterogeneous systems. Also the solution from this pattern increases the scalability (refer 3.4.3) of the system because it is easier for CRS to add new repository in the broker (trader system). In addition, it can handle multiple requests from front-end servers to different repositories.

5.3.2 Shared Repository Pattern

The shared repository pattern is designed to handle the data stored in the CRS (refer AD_6). Data in CRS are stored in different repositories such as user repository which contain data for user's information and their credentials; abstracts repository which contain the abstracts information from different sources and clinical trials repositories contains the data for clinical trials.

Shared repository pattern enables the data sharing between multiple front-end servers with different repositories through the broker (Trader system) pattern (refer AD_4).

Figure 22 shows that, the user repository contains various databases and all these databases can be accessed through the Façade interface. Façade interface is used to simplify the use of the user repository (refer AD_8).

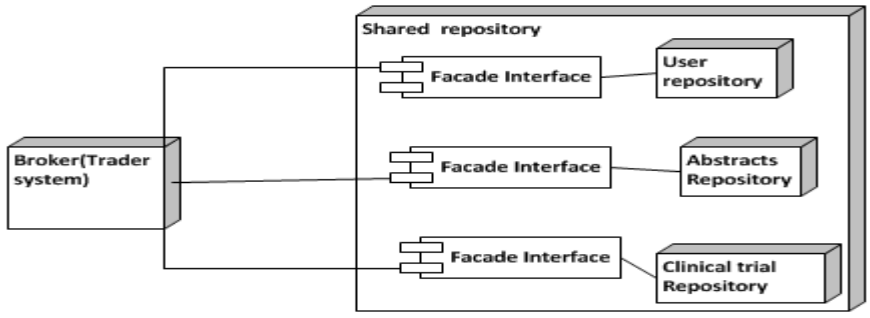


Figure 22: Shared repository

The back-end of our system contains the abstracts repository in which after reviewers perform the screening activities then selected abstracts are stored in the abstracts repository as shown in the figure 22.

Also the figure shows that, the system contains clinical trials repository to provide the central accessibility of clinical trials to different users. This repository provides data sharing by storing the clinical trials as independent component in a trial oriented format. Also, it allows multiple users to access the clinical trials through the broker (Trader system) as single point of access. All repositories in our system are designed with façade pattern in order increase accessibility of data to the clients (refer AD_8).

5.4 Sequence diagrams

We use sequence diagrams to show how objects can interact in system. The sequence diagram shows how objects interact with one another. This diagram allows the specification of runtime scenarios in a graphical manner. In this section we present the sequence diagrams for searching activities, full-text screening activities and the export clinical trials to other external system (refer figure 20).

5.4.1 Sequence diagram for searching activities

The reviewer can writes the search key word in the webpage provided by the client. As shown in the figure 23 the reviewer sends the search request to the GUI client, and the client is responsible to pack the data and formulate the request message to front-end server of CRS. The front-end server is responsible to handle the request from GUI client through its view layer (refer AD_2).

The front-end server forwards the search query to the broker (trader) to enable data accessibility from repositories in a single access. And if the search query need to access data from external

data sources then broker will forward the query to search adapter. The search adapter node has brokering capabilities to re-writes the search query and forwards it to all registered external data sources. After those sources performing the search activities, this search adapter combines the search results and re-writes the search results before returning them back to the broker (trader system). Then the broker will forward the search results to the front-end server in order to process the search results and prepare the result's page. We assume that, the front-end servers are replicated servers and the broker (trader) should know the location of all severs. Then the front-end server will return the webpage containing the search results to the GUI Client. And the GUI Client is responsible to forward the webpage of search results to reviewers ready for performing the screening activities.

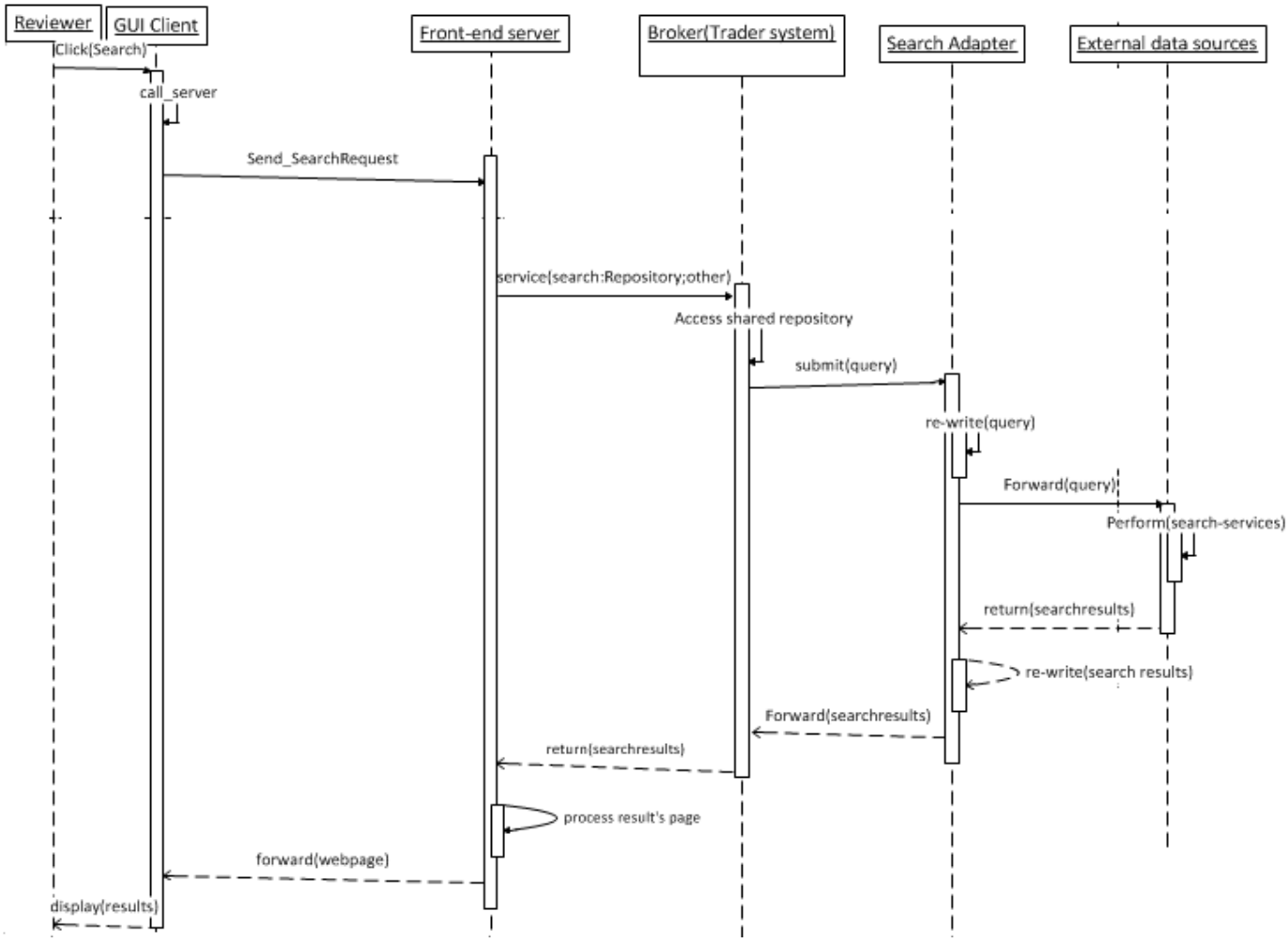


Figure 23: Sequence diagram for searching activities.

5.4.2 **Sequence diagram for screening activities in the CRS**

In this sequence diagram, we show more in detail how CRS can perform the Full-text screening activities. Figure 24 shows that, the reviewers can initiate the screening activities in the system by clicking the button called “screen” through the GUI provided by CRS. Then, the GUI Client sends the request to the Dispatch view component to finds the appropriate view for the given request. The view sends the request to look-up broker and the look-up broker find the appropriate controller for the request. The look-up broker returns the appropriate controller to the view component .The responsible view calls the full-text screening controller method to process the request.

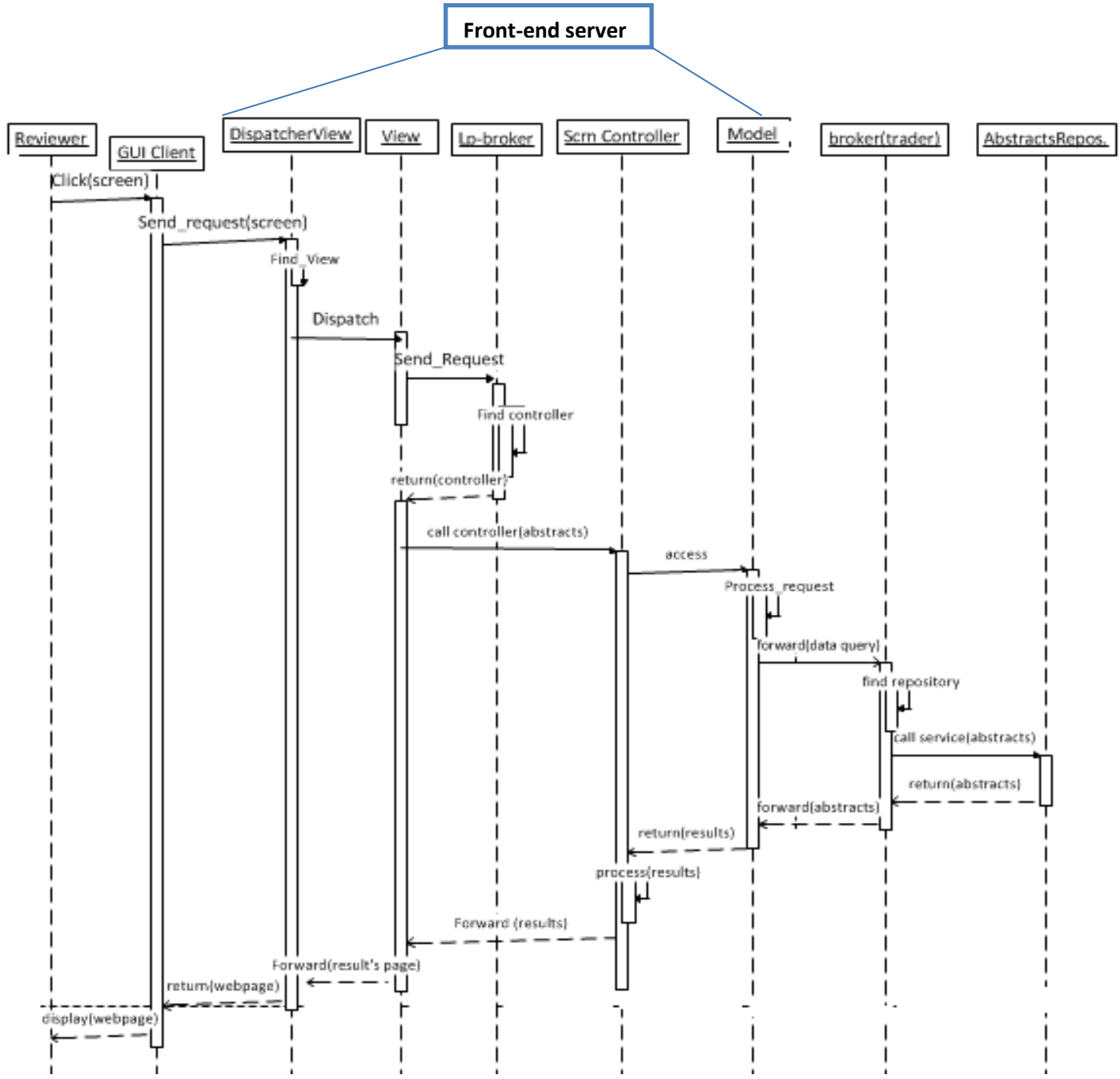


Figure 24: sequence diagram for screening activities.

As shown in the Figure 24, the screening controller can accesses the model, and the model will forwards the data query to the broker (trader). The broker (trader) used to find the abstracts repository from the pool of repositories and call for the service to access the abstracts repository for processing the given request.

After the repository processed the query it returns the results to the broker (trader), which used to forward them to the model component. The Model will return the results to the responsible controller. The controller processes the results and forwards them to the appropriate view. The view will get data and forward the results page to the dispatch view. Finally the dispatch view will forward the results page to the GUI Client and then the GUI Client displays the results page containing the list of included abstracts to the reviewer. The reviewer will open the link of identified article in the given list to start full-text screening activities. We assume that, the reviewer will ready those articles manually and select the clinical trials which are relevant to their research question.

5.4.3 **Sequence diagram for exporting clinical trials**

As shown in the Figure 25, other external systems such as RevMan/ADDIS can interact with the CRS by exporting the clinical trials stored in the repositories to perform other steps of systematic review. The RevMan/ ADDIS sends the request to the front-end server of CRS through the clinical trial exporter (refer figure 20). And that exporter will forward the request to the look-up broker to find the appropriate controller for exporting the clinical trials. The controller will call the model to access the data in the back-end servers. The model will forward the request to the broker (trader), and the broker will find the available clinical trial repository and call the service for exporting the clinical trials. The responsible clinical trial repository will process the request and return the data to the broker and the broker will forward the data to the front-end server. The front-end server wraps and exports the data to RevMan/ADDIS through the clinical trial exporter interface in the view layer.

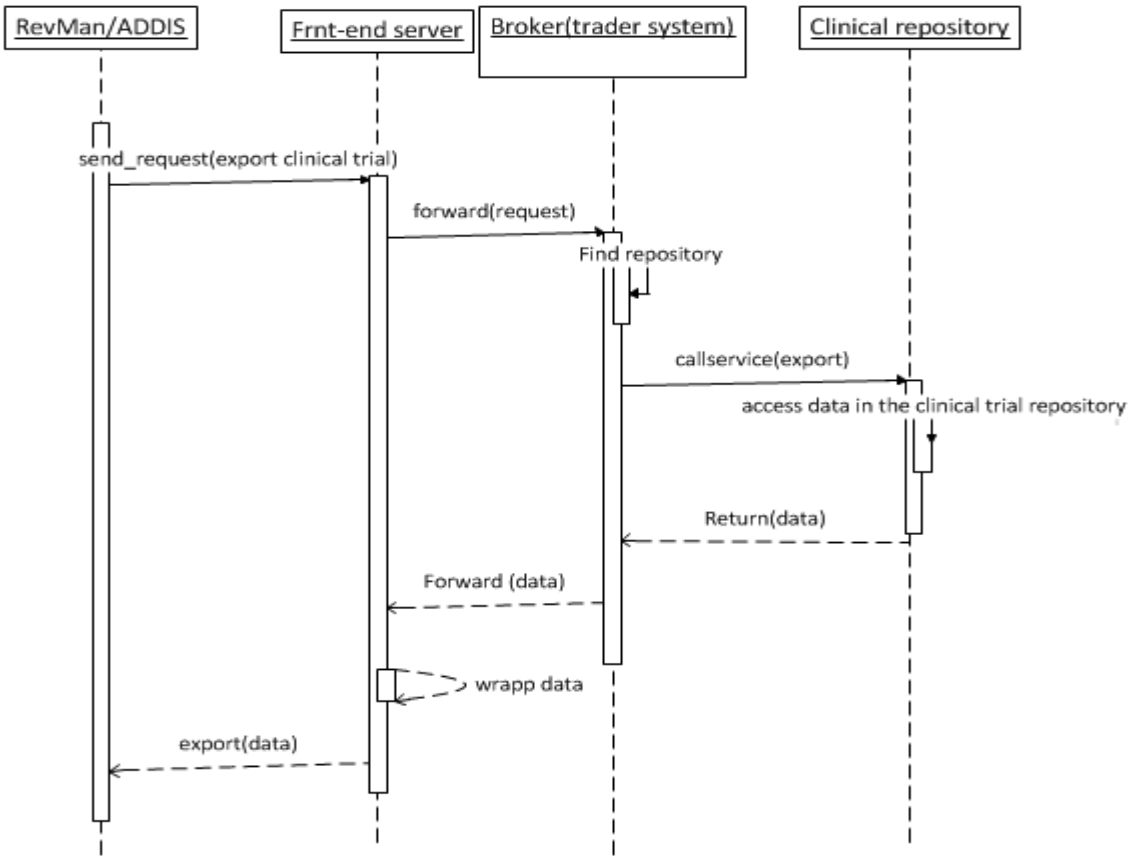


Figure 25: sequence diagram for exporting clinical trials.

6 System Validation and Verification

In this chapter we evaluate how different patterns used in our system design address the key drivers and functional requirements of CRS.

6.1 Key driver's verification

The software patterns form the design solutions which have to be understandable or reusable by other readers. In order to understand those solutions, we need to discuss the trade-off presented by the solutions which help the reader to predict the behavior of the system after applying the patterns (18).

We use Force resolution map (FRM) to evaluate our system design which shows how each patterns address our key drivers. The FRM is a graphical tool used to provide a clear and precise description of how the solution resolves all its forces. This tool helps us to decide if the patterns meet our key drivers and system requirements.

The FRM uses a numerical representation to show how the patterns behave with respect to the forces used in system design. The range of -2 to +2 shows to what extent the force (quality attribute) is addressed (18).

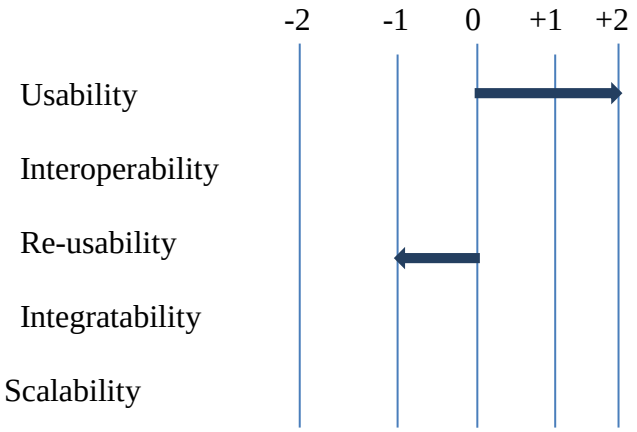
- " -2" shows that the pattern has a strong negative impact on the force.
- "-1" the behavior of the pattern follows the force indication less than other solutions.
- "0" the pattern behaves neutrally to this force which means the pattern have no impact on the given force.
- "+1" the pattern has positive impact on the force but there is other solution that address well the same force.
- "+2" shows that the pattern has a strong positive impacts on the force.

In this section, we map our key drivers (forces) with the identified patterns and find out how these patterns satisfy our key drivers. Our key drivers are Usability, Interoperability, Re-usability, Integribility and Scalability.

6.1.1 Model- View Controller Pattern

The FRM below shows that, the solution from this pattern have strong positive impact to usability of the system than other solution because the system provide the accessible and easy interface to multiple users to access the system from single point of view . Though this pattern has negative impact with re-usability that may increase the complexity of the system because

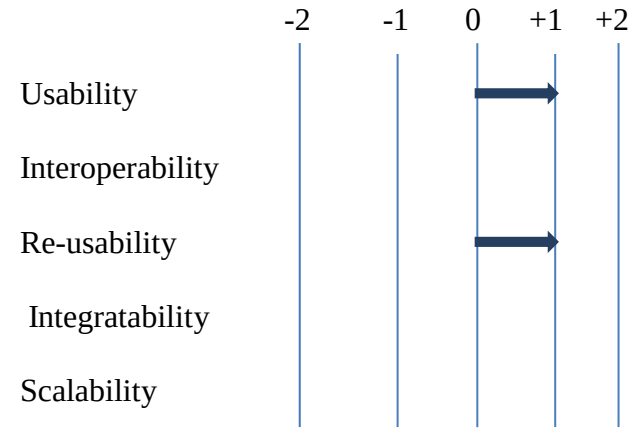
view and controller have strong relationship and they cannot be reused as individual component. The FRM below shows that, this pattern have neutral behaviors to the Interoperability, Integratability and scalability.



6.1.2 Dispatch View Pattern

The FRM below shows that, the dispatch view pattern address well usability because this pattern is used to manage all the views in our system (refer AD_3).The use of this pattern also increases the efficiency of the system because it controls the multiple views to access the same model.

Also the FRM shows that this pattern addresses well re-usability because it provides the role of separation between view and controller (refer AD_3). As a result, it solves the design problem of using Model-View Controller to enables the reusability of view and controller as independent components. The FRM shows that, interoperability, integratability and scalability are neutral forces to this pattern.

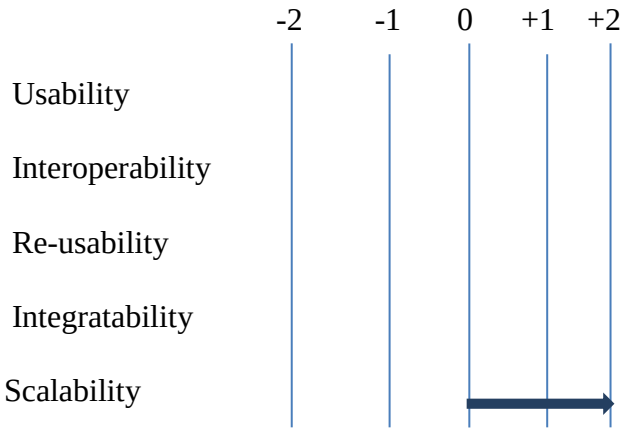


6.1.3 Look-up broker Pattern

The FRM below shows that, this pattern have strong positive impact to scalability because this pattern used to make the CRS more transparent to users (refer AD_7). As a result, it can handle multiple requests from different users of the system and at the same time it provides the all the controllers of the system in single point of view.(see figure 20).

The trade-off point in using the solution from this pattern is that when the look-up broker fails then the whole system may fail because it acts as a single point of access to the system. To solve this problem we decide to use two look-up brokers if one broker fail, then the back-up broker will be responsible to control the request from the clients(refer AD_7).

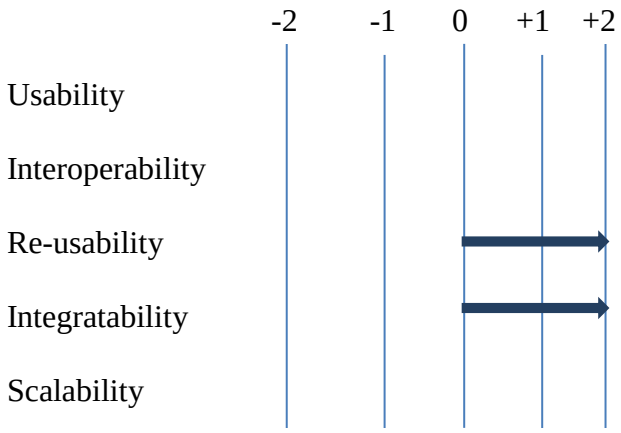
Though this pattern can handle multiple requests from the users, it may take some time for users to access the data in the back-end servers due to indirection layers in the front-end servers. Because every request has to pass through Look-up broker, as results it can lower the performance of our system.



6.1.3 Layers Pattern

As shown in the FRM below, Layers pattern has positive impact on re-usability of the system because the system decomposed into independent layers which can be reused by other layers within the system or with other external system (refer AD_2).

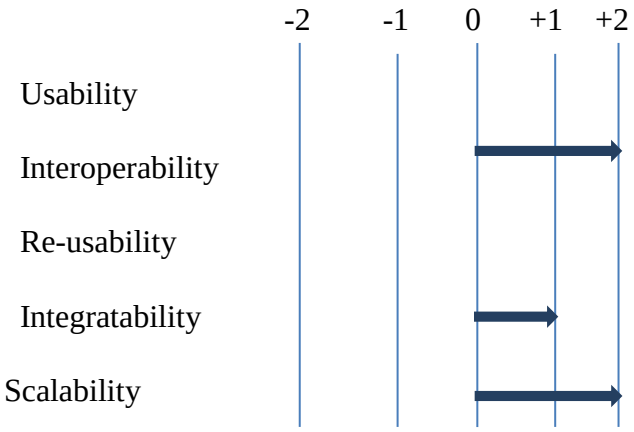
Also this pattern has strong positive impact on integratability because the use of layers decomposition allows the system to decompose into independent component that can be integrated to each other. For example, model component in the integration layer (refer Figure 20) used to integrate all the core functionalities of the system with any updates in the data access layer.



6.1.4 Broker (Trader system) pattern and Search adapter variant

The FRM below shows that, this pattern has strong positive impact to the interoperability of the system, because it provide an environment for multiple front-end servers to access the data in the repositories and external data sources through the search adapter(refer AD_4, 5, 6, 7). Also it shows that, this pattern has strong positive impact on the scalability of the system because it has registration mechanism to add new repository or external data sources (refer AD_4).

However the broker (trader) acts as a single point of accessing the data in the back-end servers and external data sources. Thus, if the broker fails then the whole system may fail. To solve this, we decide to use more than one broker and if one broker fails then the other broker will be responsible to provide data accessibility.

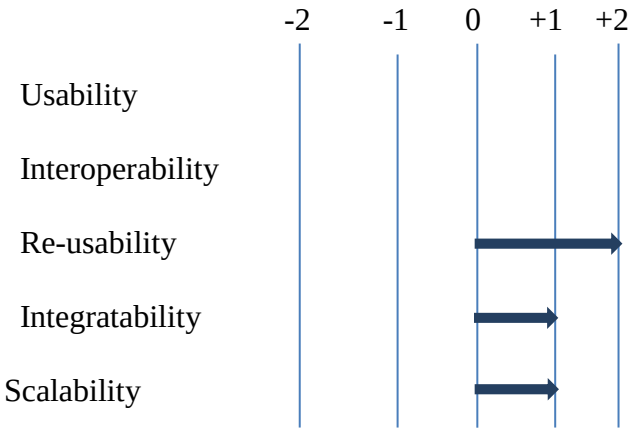


6.1.5 Shared repository Pattern

The FRM below shows that, the shared repository pattern has strong positive impact to re-usability force because it store data in the repository as an independent component which can be easily reused with multiple users at the same time(refer AD_6). For example clinical trials stored in the clinical trial repository can be re-used by other system such as RevMan/ ADDIS.

The integratability is good because the data in the repository are loosely coupled to each other as a results they can easily integrated to each other (refer AD_6). For example,the data stored by one reviewer can be integrated with any updates from the other reviewer. Also the FRM shows that, scalability of the system is good because through this pattern the system can handle millions of abstracts or clinical trials with multiple users of the system concurrently (refer section 5.3.2).

However, the solution from this pattern can be used as single point of failure because all the data are stored in a central repository. But, we assume that our system will implement the fault tolerant mechanism to replicate the repositories in order to maintain the accessibility of the data to users.



6.1.6 The complete architecture

The complete architecture is designed by combining all the patterns we decide to use in our system design (see figure 20) based on PDAP design approach.

The FRM below shows that, the designed architecture have strong positive impact on the usability of the system due to the use of Model –View Controller pattern and Dispatch View Pattern (refer AD_1,AD_3).Also, we use Façade pattern (refer AD_8) to provides easy and accessible interface for users to access the data repositories.

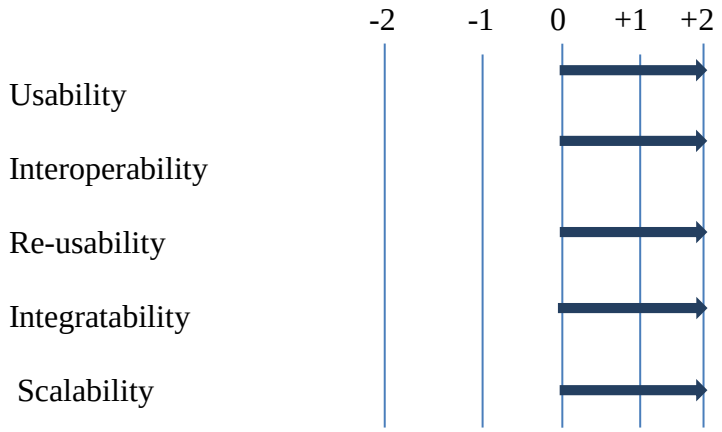
The designed architecture addresses well the interoperability (as shown from the FRM below), because the use of solution from Broker(trader) pattern provides an environment for multiple

front end servers to interact with external data sources for medical literature searches (refer AD_4).

Also, the FRM below shows that, the layer pattern and shared repository pattern address strongly the re-usability and integratability of the system (refer AD_2, AD_6). Though, the use of Model-View Controller hinders the reusability of view and controller as independent components. But we solve that design problem by using the layers pattern (refer AD_2) and dispatch View pattern to decompose our front-end servers (refer AD_3).

Also the use of solutions from look-up broker, broker(trader) and shared repository pattern and address strongly the scalability of the system because the system can be scalable with large number of users and millions of abstracts or clinical trials (refer AD_4, 5, 6).

Therefore, the designed architecture addresses well the usability, interoperability, re-usability, Integratability and scalability of the CRS.



6.2 Requirements Verification

In this section we describe the verification of our system requirements with respects to the patterns we use in designing our software architecture (refer chapter 3 and 4). The table13 below shows different patterns and we map them with every requirements of the system in order to show how these patterns satisfy our system requirements. This table shows the verification of functional requirements for searching, abstracts screening, full-text articles screening and processing of clinical trials. We use the “remark column” to describe how the identified patterns address the given requirements.

No.	Pattern Name
1	Look-up Broker

2	Layer Pattern
3	Broker(Trader system)
4	Shared repository
5	Model View Controller
7	Search Adapter variant
8	Dispatch View
9	Wrapper
10	Façade

6.2.1 Searching: Table 13 Verification of functional requirements

Requirement ID.	Patterns name	Remarks
FR1	8,5	The dispatch view provides the GUI for users to create their account in the system
FR2	4	The user can update their information in the user repository
FR3	4	The user can remove their account in the user repository
FR4		These functionalities have already automated. The user can formulate the research question and create the protocol on existing system such as RevMan/ADDIS
FR5		
FR6	3, 7	The system should interoperate well with external data sources in order to perform the search activities from them
FR7	7,3	The search results need to be processed by special program
FR8	2,4	The excluded abstracts or clinical trials are stored as use information in the shared repository.
FR9	4	The system uses the logging service as the built in functionality of the system.
FR10	4	The system use filters as built in functionality
FR11	2	The search results are processed by other external packages which are integrated with the main system.
FR12	2,1	The main system integrates with other external packages through the control layer in order to perform automatic processes.

6.2.2 Abstracts Screening

Requirements ID	Patterns used	Remarks
FR13	5,7	The system provides GUI for reviewer to perform screening activities abstract screening component and model component is used to control the process of this requests. See figure 20
FR14	4	The special programs are used to keep log of selected abstracts.
FR15	5,8,4	The dispatch view component used to handle multiple views to access the same data model in the shared repository.
FR16	4	The abstracts screened are stored in the shared memory which can be synchronized with new updates. The multiple reviewers can share the same database to store their data.

FR17	4	All data are stored in the shared repository.

6.2.3 Full-text screening

Requirement ID	Patterns	Remarks
FR18	5,8	The system provides GUI for reviewers to perform full-text screening activities.
FR19	2	The control layer provides the API for computer scientists to create algorithms for automating the full-text articles processes.
FR20	4	The reviewers read the identified articles manually.
FR21	5,8,4	The system provide the GUI to reviewers to perform full-text screening from the same data model.
FR22	5,8,4	The system can provide multiple views of the same search results to multiple reviewers
FR23	4	Excluded information can be stored in the shared repositories.
FR24	4,5	Data in the repository should be synchronized with new updates from different views.
FR25	4	System consists of different databases containing different clinical trials from different sources.

6.2.4 Process of Included Clinical Trials.

Requirement ID	Pattern used	Remark
FR26	8,5	System provides multiple views of data from the data model
FR27	2,1	The system consists of special program for processing the clinical trials.
FR28	5	The system should provide GUI for reviewers to create clinical trials manual
FR29	8,5,4	All views should be synchronized with the updates from the model
FR30	4,10	All data are stored in a shared memory with simple User interface for easy access
FR31	2,9	Due to heterogeneity the wrapper pattern used to provide the data in a common user interface between the that systems
FR32	4	The created bibliography can be stored in the shared repository
FR33		Not solved in our system design

7 Discussion

In this section we conclude our research by discussing the results of using software patterns in designing the architecture of a system for systematic review in order to improve the efficiency of systematic review. Thereafter we describe some gaps that need to be covered in the future research.

7.1 Conclusions

The current status for systematic review of clinical trials is a time consuming process due to manual work for searching and screening activities. This is because the clinical trials are scattered to various information systems that are not interoperated to each other. As a result, the process for systematic review is inefficient and error prone.

We designed the Clinical Review System (CRS) to support the processes for systematic review of clinical trials in an efficient way. First, we identified the quality attributes such as usability, interoperability, re-usability, integribility, and scalability as the key drivers to design such a system. Also we identified functional requirements such as literature searching, abstract screening and clinical trials export as the main use cases, based on a system context analysis. In order to meet those requirements, we used Pattern-Driven Architectural Partitioning (PDAP) to design the architecture of the CRS. We identified different patterns from architectural documents such as books or articles and decompose our system based on the solutions from the identified patterns. After combining the solutions from all identified patterns based on PDAP approach, we designed the complete software architecture for the CRS.

The designed architecture of CRS reduces the manual work of performing the systematic review. The CRS enables the reviewers to perform automatic literature searches from different sources of evidence from a single point of view. In addition, the CRS provides a GUI for reviewers to perform abstract screening and full-text screening in an efficient way.

Also, the designed architecture of CRS increases the re-usability of abstracts or clinical trials during the process of systematic review because the data stored in the shared repositories that can be accessed by multiple users at the same time. Moreover, the CRS provides an API for reviewers to perform all steps of systematic review in a single point of view. According to that, it

enables other existing systems to export the clinical trials stored in the repositories in order to perform other steps of systematic review.

The verification of our system showed that, the designed architecture addresses our system's requirements and key drivers strongly. Although there were some trade-offs, such as the complexity of the system and failure points, we identified other solutions or modified the existing solutions in order to balance our forces.

So we conclude that, the use software patterns in designed the architecture of CRS help us to save time during the architecting processes because we designed this architecture based on the solutions from existing system design. Though it took some time to identify the patterns that match with our design problems but also it helped us to design the complete architecture that satisfies its key drivers and functional requirements.

Therefore, software patterns improve the quality of designed architecture of CRS by enabling the satisfaction of its quality attribute and functional requirements. Due to that, the designed architecture of CRS increases efficiency of systematic review of clinical trials by reducing manual work and supporting literature searching and screening activities from a single point of view.

7.2 Future work

Although the designed architecture of CRS using software patterns increases the efficiency of systematic review there are still some gaps that need to be addressed in the future research.

First of all, there are some gaps in the study identification because every nation has its own name of clinical trials so it is difficult for CRS to trace clinical trials from different sources. Future research should address how clinical trials can be identified automatically and accurately.

In addition, data extraction is still a manual step, this process can be improved by providing automatic data extraction from the included clinical trials. Due to its API, the CRS can be used as a platform to develop automated data extraction algorithms.

Finally, although the reviewers are used to meet in order to agree or disagree on the selected abstracts or clinical trials, but it would be helpful to automate this process by implementing a collaboration mechanism in the CRS. This mechanism will enable reviewers to send instant messages or conduct an e-meeting in order to increase the efficiency of communication between reviewers.

References

1. CORE J2IEEE PATTERNS CATALOGY [Internet]. Available from: http://www.corej2eepatterns.com/Patterns2ndEd/ServiceToWorker.htm.

2. Alam R,Sturt J, Lall R, Winkley k. An updated meta-analysis to assess the effectiveness of psychological interventions delivered by psychological specialists and generalist clinicians on glycaemic control and on psychological . Patient Education and Counseling. 2009;75(1):25-36.

3. Boswell MV. Registering clinical trials. JAMA. 2005 Jul;8(3):249-50.

4. Buschmann F, Meunier R, Rohnert H, Sommerlad P, Stal M, editor. Pattern-oriented software architecture. John Wiley & Sons Inc,111 River Street, Hoboken, NJ07030, USA ed. West Sussex P0198SQ, England: John Wiley & Sons Ltd; 1996.

5. Carter MJ. Evidence-based medicine: An overview of key concepts. Ostomy Wound Manage,. 2010 Apr 1;56(4):68-85.

6. Derry S, LLoyd R, Moore RA, McQuay HJ. Topical capsaicin for chronic neuropathic pain in adults (review). 2009,University of Oxford, West Wing (Level 6), John Radcliffe Hospital, Oxford, Oxfordshire, UK, OX3 9DU.

7. Dickersin K, Manheimer E, Wieland S, Robinson KA, Lefebvre C, McDonald S. Development of the cochrane collaboration's CENTRAL register of controlled clinical trials. Eval Health Prof. 2002 Mar;25(1):38-64.

8. Dickersin K, Rennie D. Registering clinical trials. JAMA. 2003 Jul 23;290(4):516-23.

9. Gamma E, Helm R, Johnson R, vlissides J, editor. Design patterns - elements of reusable object-oriented software. 201 W.103rd Street,Indianapolis,IN46290: Addison-Wesley; 1994.

10. Gert van Valkenhoef, Tommi Tervonen, Tijs Zwinkels, Bert de Brock, Hans Hillege. ADDIS: A decision support system for evidence-based medicine(under reveiw). Under review,Erasmus University Rotterdam, Netherland Feb 2011.

11. Gert van Vanlkenhoef, Tommi Tervonen, Bert de Brock, Hans Hillege. Deficiencies in the transfer and availability of clinical evidence in drug development and regulation(under review). under review(International Journal of Medical Informatics),Faculty of Economics and Business, University of Groningen, Netherland Dec 2010.

12. Gillen JE, Tse T, Ide NC, McCray AT. Design, implementation and management of a web-based data entry system for ClinicalTrials.gov. Stud Health Technol Inform. 2004;107(Pt 2):1466-70.

13. Harrison N, Avgeriou P, Zdun U. Using patterns to capture architectural decisions. Software IEEE August 2007;24(4):38-45.

14. Harrison, N. ,Avgeriou,P. In: Pattern-driven architectural partitioning:Balancing functional and non-functional requirements. 1-5 July; San Jose, CA. Digital Telecommunications, 2007. ICDT '07. Second International Conference; 2007. p. 21-21.

15. Haynes RB, McKibbon KA, Wilczynski NL, Walter SD, Werre SR, Hedges Team. Optimal search strategies for retrieving scientifically strong studies of treatment from medline: Analytical survey. BMJ. 2005 May 21;330(7501):1179.

16. International Organization for standardization. ISO standard 9126:Software engineering product quality. . Geneve 2001(part1) , 2003(part 2 and part 3).

17. J.Paul RS, T.Prescott. The internet and clinical trials:Background , online resources, examples and issues. J med Internet Res. 2005;7(1).

18. J.Souza SM, N.Japkowiez. In: Evaluation data mining models: Pattern language. ; Proceeding of the 9th conference on pattern language of programs(PLOP'2002"),2002.

19. Dispatch view pattern [Internet]. [updated Jan 29 2006. Available from: http://www.corej2eepatterns.com/Patterns2ndEd/DispatcherView.htm.

20. Monica Kjeldstrom,Rasmus Moustgaad, Jacob Riis. Cochrane IMS budget request 2009-2012. Available from: http://ims.cochrane.org/projects/IMS-budget-request-2009-2012.pdf

21. Pai M, McCulloch M, Gorman JD, Pai N, Enanoria W, Kennedy G, et al. Systematic reviews and meta-analyses: An illustrated, step-by-step guide. Natl Med J India. 2004 Mar-Apr;17(2):86-95.

22. Shared repository patttern [Internet]. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.62.1879&rep=rep1&type=pdf.

23. Shalloway A TJR, editor. Design patterns explained: A new perspective on object-oriented design. Series editor: John M.Vlissides ed. Addison-Wesley; Dec 2002.

24. Starr M, Chalmers I, Clarke M, Oxman AD. The origins, evolution, and future of the cochrane database of systematic reviews. Int J Technol Assess Health Care. 2009 Jul;25 Suppl 1:182-95.

25. Twarek. W, Desforges.C, Robert.b, Krishnaswamy.R, editor. Pattern: Portal search custom design. U.S;ibm.com/redbooks/residencies.html: International Business Machines Corporation(IBM); 2004.