[Winter 2021] CS 687 Capstone Project Proposal
A Case Study of Software Reengineering using Emerging Cloud Computing: A Non-profit's National Math Competition Event Management System

Huixian Eunice Yuan
Advisor: Sam Chung, Ph.D.
Master of Science in Computer Science (MSCS)
School of Technology & Computing (STC)
City University of Seattle (CityU)
[yuanhuixian@cityuniversity.edu, chungsam@cityu.edu]

Abstract

Recently, there is some research about software reengineering, but few kinds of analysis use cloud computing technologies to demonstrate software reengineering for web application technologies. Cloud computing also will be an emerging computing paradigm that will affect information systems, such as event management systems. Therefore, this project will propose some best practices of doing software reengineering using AWS cloud computing services and web application technologies based on the case study of the KSEA National Math Competition event management system. We fill the knowledge gaps in software reengineering for web applications using cloud computing technologies with these best practices.

Keywords: Software Reengineering, Event Management System, Cloud Computing, AWS Cloud Computing Services, Web Application, Architectural Models.

1. INTRODUCTION

Problem Statement

This project aims to propose best practices of software reengineering a legacy web application to an emerging target system using AWS cloud computing services. We choose the KSEA National Math Competition event management system to do software reengineering as a case study.

Motivation

According to Chung et al. (2009), a legacy system could be visualized with a re-documented technique, 5W1H, to find the candidate services and some reusable business logic used in the target system. Chikofsky et al. (1990) defined reverse engineering and redocumentation. Nguyen (2011) also proposed definitions, approaches, processes, and risks of software reengineering. However, all of them did not use cloud computing technologies in their research. Besides, cloud computing is an emerging computing paradigm that will affect legacy information systems. Thus, it is essential and meaningful to demonstrate best practices about how to reengineer a legacy information system to an advanced target information system using clouding computing technologies.

Approach

The paper uses software reengineering methodology, AWS cloud computing services, and web application technologies to analyze and modernize a legacy KSEA National Math Competition (NMC) Event Management System (EMS) to develop an advanced target system and identify best practices of this process.

Conclusions

The paper aims to generate some best practices of software reengineering with AWS cloud computing services and web application technologies through a case study for the actual KSEA NMC EMS. These best practices describe the whole process of software reengineering with AWS cloud computing services and web application technologies based on a real case study and fill the knowledge gaps of this field.

2. BACKGROUND

Software Reengineering: According to Nguyen (2011), software reengineering is a complicated improvement or transformation of existing software. According to Manzella and Mutafelija (1992), software reengineering mainly includes reverse engineering and forward engineering. Chikofsky (1990) states that reverse engineering uses redocumentation and design recovery to analyze a system. Analyzing a system can get "the system's components and their interrelationships and create representations of the system in another form or at a higher level of abstraction." Design recovery can get high-level abstractions by code, domain knowledge, and design documentation.

5W1H Re-Doc: According to Chung et al. (2009), the legacy system can be visualized as a visual model in Unified Modeling Language (UML) to find the candidate services and some reusable business logic using the 5W1H re-documentation methodology. The 5W1H re-documentation methodology is "Who was involved in re-documentation, What happened for re-documentation, When did re-documentation take place, Where did re-documentation take place, Why did re-documentation happen, and How did re-documentation happen." Chung et al. (2009) used Kruchten's 4+1 views to define "where" (Kruchten, 1995).

4+1 views: In Kruchten's (1995) paper, the 4+1 views consist of "Use case View (UV), Design View (DV), ProcessView (PV), Implementation View (IV), and Deployment View (LV)." The 4+1 views stand for use requirements, class hierarchies, message exchange, source code, and relationships.

UML: According to GeeksforGeeks (2019) and uml.org, Unified Modeling Language (UML) is a universal modeling language to define a standard method to visualize a software system and describe the software system's behavior and structure.

Scrum: According to Deemer's (2012) paper, Scrum is a "development framework in which cross-functional teams develop products or projects in an iterative, incremental manner. It structures development in cycles of work called Sprints". A Scrum Team has a Product Owner, Scrum Master, and Team. A team will choose a project and make a target to finish. Each project has several sprints, and each has the same length and fixed content. Every day the team gathers to check the progress and adjust the next steps.

Information System: In Bourgeois (2019), an information system is a combination of "hardware, software, data, people, and processes." The "hardware, software, and data are technologies to collect, process, store, and distribute information," and the "people and processes" are responsible for separate the theory of information systems from other technologies.

KSEA NMC EMS: Both the Legacy and Target Information Systems is the KSEA National Math Competition (NMC) Event Management System (EMS). According to KSEA (2021), the Korean-American Scientists and Engineers Association (KSEA) is a non-profit association and focuses on engineering and science fields. The association aims to help Korean-American develop their careers in engineering and science fields. The NMC aims to improve students' mathematical and scientific abilities. The KSEA NMC EMS is a system to manage event registrations and report exam results to the KSEA headquarter, local chapters, and each participant.

AWS: According to Rungta (2020), Amazon Web Service (AWS) is a platform that can provide some flexible, spent-less, and scalable cloud computing solutions. In AWS (2020), Elastic Compute Cloud (EC2) is a web service that can "provide resizable compute capacity." Elastic Beanstalk is an application container that helps use web applications in services. Elastic Container Services (ECS) provides some API to help use Docker container applications and some functions. Relational Database Service (RDS) helps manage relational databases. Serverless Computing is to manage applications without services.

React and JavaScript: In Morris (2020), React.js is an open-source web front-end framework and JavaScript library to "build interactive elements on websites." JavaScript is a programming language to "create and control dynamic web content."

Laravel and PHP: In Heddings (2020), Laravel is an open-source server-side web framework to support model-view-controller (MVC) software architecture in PHP. According to Morris (2018), Hypertext Preprocessor (PHP) is a server-side scripting language used in web development.

Nginx: Nginx is also a web server that uses an "asynchronous, event-driven architecture to handle these massive amounts of connections."

MySQL: In Talend (2020), MySQL is a "relational database management system (RDBMS) based on structured query language (SQL)."

3. RELATED WORK

Recently, there is much research about software reengineering in different aspects. Many researchers focus their research on the theory and framework knowledge of software reengineering.

Manzella and Mutafelija (1992) fully described a development life cycle steps integrated with reverse engineering and forward engineering to generate a new reengineering life-cycle framework. Waheed (1992) stated reengineering software was an easy way to convert code from proprietary language to commercial language. This process could be done by "use programmable program transformation tools to partially translate the code to a standard commercial language." Therefore, Waheed (1992) provided some programmable tools and partial translation results from these tools to prove that reengineering existing code through partial translation can reduce costs and improve efficiency. Nguyen (2011) thought that software reengineering was a useful tool to improve the legacy system to a new target system. He summarized the process of software reengineering, including definitions, approaches, processes, and risks, which could be a basis for other researchers. However, Manzella and Mutafelija (1992), Waheed (1992), and Nguyen (2011) only were interested in the theoretical knowledge of software reengineering. However, Li (2011) focused on doing software reengineering using a scientific software, SeisSol. Li concluded that software reengineering could help a scientific software improve source code, software system, extensibility, and complexity by using a "domain-specific requirements model" and new reengineering ways.

According to Nguyen's (2011) paper, reverse engineering is one of the software reengineering steps. Some researchers not only pay attention to software reengineering but also have in-depth research on software reverse engineering.

Chikofsky and Cross (1990) provided the definitions of six terms for software reengineering: forward engineering, reverse engineering, re-documentation, design recovery, restructuring, and reengineering, which help rationalize and apply these terms to the underlying engineering processes. According to Chung et al.'s (2009) research, they came up with a re-documentation methodology, 5W1H Re-Doc, to visualize the legacy system to discover the candidate services and some reusable business logic used in the legacy system. Chikofsky and Cross focused on underlying software reengineering concepts. Chung et al. focused on service-oriented software engineering. Tonella (2005) focused on reverse engineering for source code. He introduced a code analysis framework and proposed several UML view-designed technologies with a running example to help reverse engineering for source code, including object, interaction, class diagrams, and so on. Fouad and Mohamed (2018) showed a method to help understand integrity constraints and improve database reverse engineering and database migration, maintenance, and resetting of systems using Object-Relational Database (ORDB) by generating a conceptual schema (CS) in an ORDB with integrity constraints. This method supplements the research on constraints in relational databases in reverse engineering.

Although there are many studies on software reengineering, a few researchers consider doing software reengineering using cloud computing technologies.

Ahmad and Babar (2014) provided a framework for utilizing software reengineering to restore the old source code's architecture based on the reengineering horseshoe model. The framework has four processes, including restoring the original architecture from source code and migration from legacy system to cloud, and so on. Zalazar, Gonnet, and Leone (2015) proposed a workflow of migrating applications to cloud computing based on different applications' characteristics and the characteristics and deployment models of cloud computing. They presented a workflow to show how the workflow works according to features and deployment models using a beverage distributor company. Sabiri and Benabbou (2017) came up with a meta-model that provided the basis for modernizing the old version of the application and the cloud service model's mobile selection. With this meta-model, companies analyzed a legacy system from business viewpoints, implementation and data viewpoints, and infrastructure viewpoints and modernized the legacy system with a suitable cloud service. Besides, Ellison, Calinescu, and Paige (2014) described the database reverse engineering and data migration problems in reengineering the database layer of legacy applications in the cloud.

Review Conclusions

These papers' main research issues focused on three parts: classical software reengineering, reverse engineering, and the use of cloud computing technology for software reengineering. Some findings from these references include:

These references almost focused on using different framework knowledge to do software reengineering or reverse engineering research. However, only a few research studies were related to use actual case studies to do software reengineering with cloud computing technologies. It is a feasible method to use cloud computing for software reengineering. This method needs to consider the type of cloud computing technologies, the type of functions reused in the system, the appropriate architectural model, etc.

4. APPROACH

This project will be a case study to propose best practices of how to do the software reengineering using cloud computing for a non-profit's national math competition to modernize the event management system as the legacy system to an advanced target system. In Table 1, three references in the first row come from Section 3 Related Work. We classify them in terms of reverse engineering only (Chung et al, 2009), reverse and forward software engineering i.e., software reengineering (Li, 2011), and software reengineering with cloud computing technologies (Ahmad & Babar, 2014). The first column shows comparison criteria. The other columns show how each reference used each criterion. Thus, Table 1 compares our approaches to the approaches from Section 3 to show differences between our approaches and others.

TABLE 1 DIFFERENCES BETWEEN OUR APPROACHES AND OTHERS

Criteria	(Chung et al., 2009)	(Li, 2011)	(Ahmad & Babar, 2014)	My Approach
Software Reengineering	Reverse	Reverse / Forward	Reverse / Forward	Reverse / Forward
Best Practices	5W1H Re-Doc	Document of Source Code, Redesign & Require	Framework for Migration of Legacy Systems to Cloud-enabled Software	5H1W Re-Doc Methodology / Forward Engineer with cloud computing services & web technologies
Cloud Computing Services	No	No	Cloud-based Architectures	AWS Cloud Computing Services
Web Application Technologies	No	No	No	Client-/Server-side Web Framework / Web Server / RDMS
Legacy/Target Systems	Web / Console Applications	SeisSol Scientific Software	No	KSEA NMS EMS
Architectural Models	UML / 4+1 View Model	No	No	UML / 4+1 View Model

Our Approaches to this process include five steps: First, analyze the given legacy system. The legacy system is the KSEA NMC EMS with some source code files. We can be familiar with the legacy system structure and discover some designs by analyzing the source code files.

Second, reverse engineer the legacy system by using the 5W1H re-documentation methodology. We use a Computer-Aided Software Engineering (CASE) tool to generate UML diagrams to identify the legacy system based on the 5W1H re-documentation methodology.

Thirdly, identify new cloud computing services and web technologies. The AWS cloud computing services and some web application technologies are chosen to do forward engineering based on the legacy system's UML diagrams.

Fourthly, forward engineer the generated architectural models using AWS cloud computing services and some web application technologies. The outcome of forwarding engineering will develop a new emerging EMS on AWS.

Lastly, propose the best practices of software reengineering. The best practices describe the whole process of software reengineering with AWS cloud computing services and web application technologies.

5. DATA COLLECTION

This paper uses the case study method of qualitative research to collect the necessary data. The case study is based on the legacy KSEA NMC EMS.

We collected the data below:

First, source code files. We collected source code files from the legacy KSEA NMC EMS: PHP files and SQL files, which are essential for software reengineering. We ran the source code files in the Visual Studio Code and found an NMC, and two database schemas, including a member database schema and an NMC database schema.

Second, legacy app model and target app model. We separately drew a legacy app model and a target app model by a CASE tool using Chung's (2009) 5W1H re-doc methodology based on the source code files, including some visual diagrams of the legacy system and the target system in different views. The target diagrams were based on the improved and upgraded legacy diagrams and only focused on the student registration object's use case diagram, sequence diagram, and class diagram. The CASE tool we used in this project is Sparx's Enterprise Architecture. We draw different diagrams for each view: a deployment diagram for deployment view, a component diagram for implementation view, a class diagram for static design view, a sequence diagram for dynamic design view, and a use case diagram for use case view. Figure 1 through figure 5 shows the diagrams of the legacy app model. Figure 6 to figure 8 shows the diagrams of the target app model.

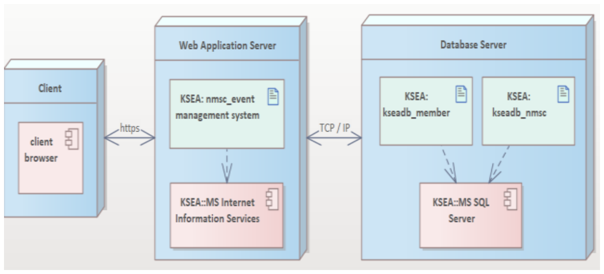


Figure 1. Legacy Deployment Diagram

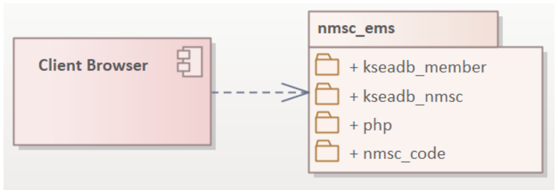


Figure 2. Legacy Component Diagram

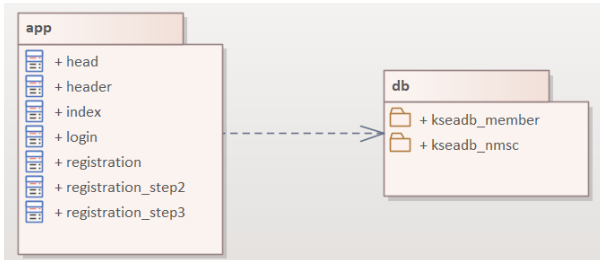


Figure 3. Legacy Class Diagram

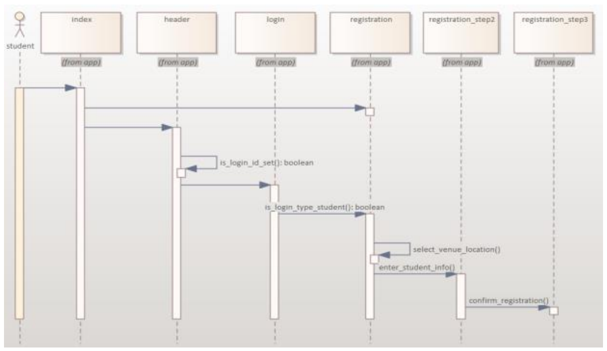


Figure 4. Legacy Sequence Diagram

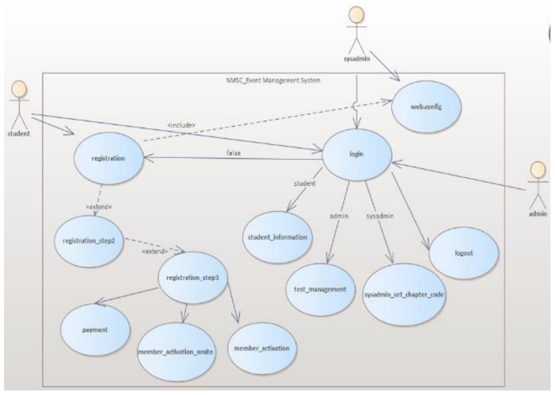


Figure 5. Legacy Use Case Diagram

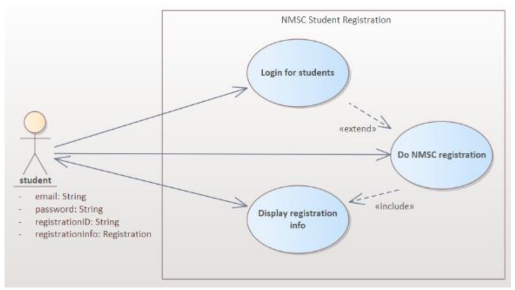


Figure 6. Target Use Case Diagram of The Student Registration Object

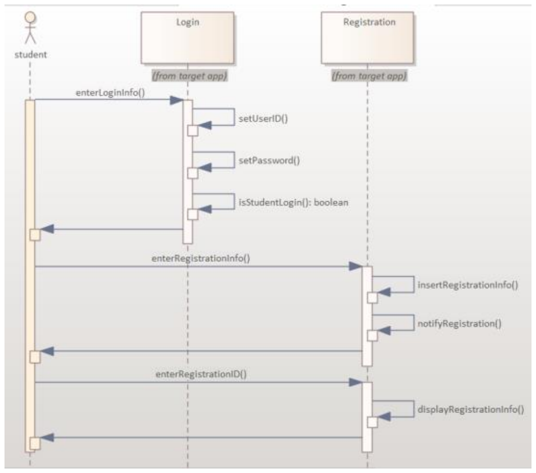
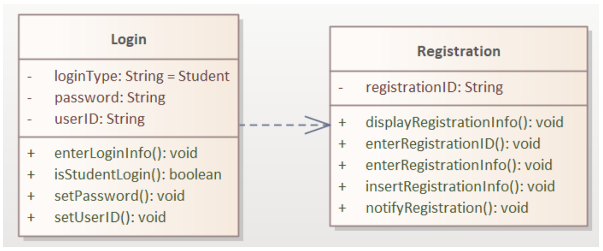


Figure 7. Target Sequence Diagram of The Student Registration Object



Login
- loginType: String = Student
- password: String
- userID: String
+ enterLoginInfo(): void
+ isStudentLogin(): boolean
+ setPassword(): void
+ setUserID(): void
Registration
- registrationID: String
+ displayRegistrationInfo(): void
+ enterRegistrationID(): void
+ enterRegistrationInfo(): void
+ insertRegistrationInfo(): void
+ notifyRegistration(): void

Figure 8. Target Class Diagram of The Student Registration Object

Third, legacy database model and target database model. We found the legacy database model in the legacy static design view and improved the model and drew a target database model. Both the legacy database model and the target database model include a member database diagram and an NMC database diagram. Figure 9 and figure 10 show two legacy database diagrams. Figure 11 and figure 12 show two target database diagrams.

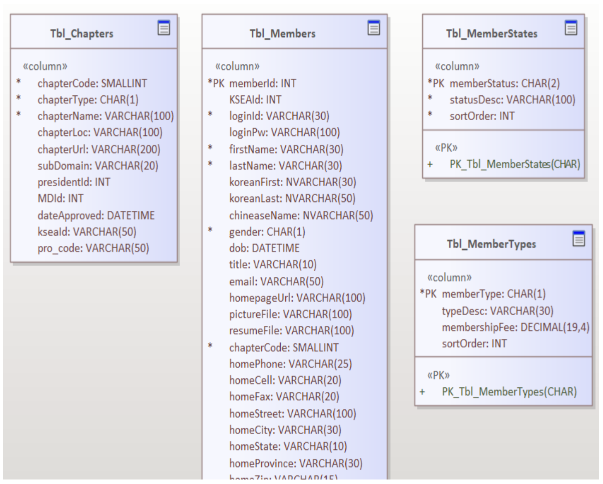


Figure 9. Legacy Member Database Diagram

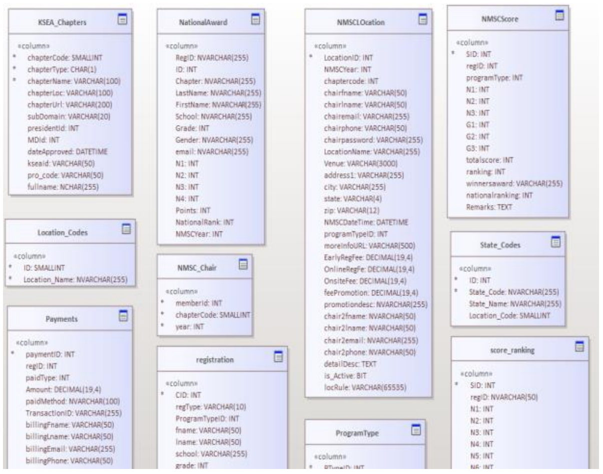
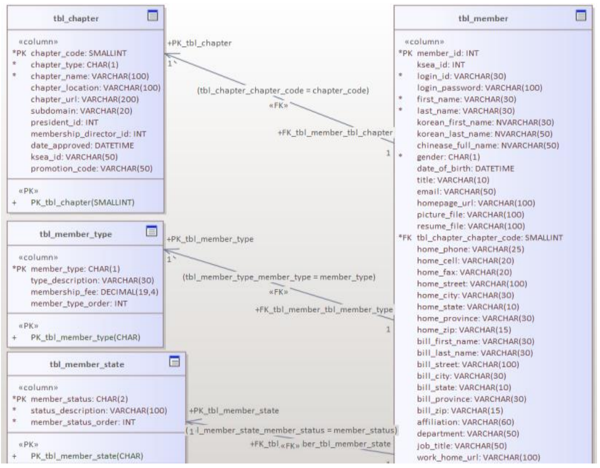
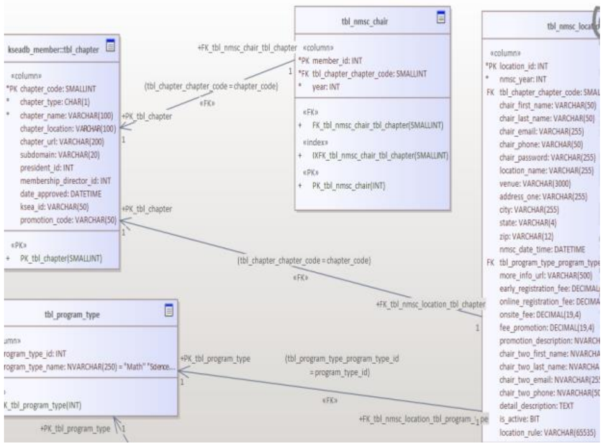


Figure 10. Legacy NMC Database Diagram



tbl_chapter
«column»
*PK chapter_code: SMALLINT
* chapter_type: CHAR(1)
* chapter_name: VARCHAR(100)
chapter_location: VARCHAR(100)
chapter_url: VARCHAR(200)
subdomain: VARCHAR(20)
president_id: INT
membership_director_id: INT
date_approved: DATETIME
ksea_id: VARCHAR(50)
promotion_code: VARCHAR(50)
«PK»
+ PK_tbl_chapter(SMALLINT)
tbl_member_type
«column»
*PK member_type: CHAR(1)
type_description: VARCHAR(30)
membership_fee: DECIMAL(19,4)
member_type_order: INT
«PK»
+ PK_tbl_member_type(CHAR)
tbl_member_state
«column»
*PK member_status: CHAR(2)
* status_description: VARCHAR(100)
* member_status_order: INT
«PK»
+ PK_tbl_member_state(CHAR)
tbl_member
«column»
*PK member_id: INT
ksea_id: INT
* login_id: VARCHAR(30)
login_password: VARCHAR(100)
* first_name: VARCHAR(30)
* last_name: VARCHAR(30)
korean_first_name: NVARCHAR(30)
korean_last_name: NVARCHAR(50)
chinease_full_name: NVARCHAR(50)
* gender: CHAR(1)
date_of_birth: DATETIME
title: VARCHAR(10)
email: VARCHAR(50)
homepage_url: VARCHAR(100)
picture_file: VARCHAR(100)
resume_file: VARCHAR(100)
*FK tbl_chapter_chapter_code: SMALLINT
home_phone: VARCHAR(25)
home_cell: VARCHAR(20)
home_fax: VARCHAR(20)
home_street: VARCHAR(100)
home_city: VARCHAR(30)
home_state: VARCHAR(10)
home_province: VARCHAR(30)
home_zip: VARCHAR(15)
bill_first_name: VARCHAR(30)
bill_last_name: VARCHAR(30)
bill_street: VARCHAR(100)
bill_city: VARCHAR(30)
bill_state: VARCHAR(10)
bill_province: VARCHAR(30)
bill_zip: VARCHAR(15)
affiliation: VARCHAR(60)
department: VARCHAR(50)
job_title: VARCHAR(50)
work_home_url: VARCHAR(100)
+PK_tbl_chapter
(tbl_chapter_chapter_code = chapter_code)
«FK»
+FK_tbl_member_tbl_chapter
+PK_tbl_member_type
(tbl_member_type_member_type = member_type)
+FK_tbl_member_tbl_member_type
+PK_tbl_member_state
(tbl_member_state_member_status = member_status)

Figure 11. Member Database Diagram of the Target System



kseadb_member::tbl_chapter
tbl_nmsc_chair
tbl_nmsc_location
tbl_program_type
+FK_tbl_nmsc_chair_tbl_chapter
(tbl_chapter_chapter_code = chapter_code)
+PK_tbl_chapter
+FK_tbl_nmsc_location_tbl_chapter
+PK_tbl_program_type
(tbl_program_type_program_type_id = program_type_id)
+FK_tbl_nmsc_location_tbl_program_type

Figure 12. NMC Database Diagram of the Target System

6. DATA ANALYSIS

We collected the source code files from the legacy system and drew the legacy and target system diagrams by 5W1H re-doc methodology in Section 5. The data analysis is shown in two parts:

1. Legacy and target NMC source code:

First, legacy NMC source code. According to the figure 1 through figure 5, the source code was divided into several functional elements by the divide and conquer strategy, such as login, registration, and logout element, etc. Each functional element had its attributes and operations. Besides, each functional element could communicate and could be grouped by its functions. However, there was not any pattern followed by the source code.

Second, target NMC source code. According to the figure 6 through figure 8, this target source code focused on a student object and a student registration class rather than system functions. The student registration class is one of the attributes and methods, which the student object had. The target source code was also designed as a Model layer and even will focus on the View and Controller layers later.

2. Legacy and target database:

First, the naming ways of table names and column names. In figure 11 and figure 12, all the table names were unique, singular, and lowercase words and used underscore to separate each word. The column names were named in the same way and avoided abbreviated and using numbers and reserve words. And most column names were no-longer than two terms. But in figure 9 and figure 10, the table names and column names did not use these ways to name.

Second, primary keys and constraints. All tables in both figure 11 and figure 12 identified their non-null primary keys and their constraints, and the column names and data types of primary keys are easy understand and unique. However, only several tables in figure 9 and figure 10 had primary keys. Besides, some column names of primary keys were hard to understand.

Third, foreign keys and relationships among tables. We identified all the foreign keys, their constraints, and relationships in the tables of figure 11 and 12. The column names of the foreign keys had the tables' names where the primary key column was located. We also deleted unnecessary tables that did not have relationship with other tables. But figure 9 and figure 10 did not identify any foreign keys, constraints, and relationships.

7. FINDINGS

According to Section 6, this section will be divided into two parts to explain the specific findings mainly.

1. Legacy and target NMC source code:

First, legacy NMC source code. According to Tutorialspoint (2021), we can identify that the legacy source code used the Structured Design design method to design but did not use any architecture pattern because the legacy source code's design method followed the feature of Structured Design.

Second, According to Tutorialspoint (2021), target NMC source code. Although the target source code only focused on the student object and the student registration class, we still can identify the target source code used Object-Oriented Design (OOD) to design and even design the whole system by the OOD. Besides, we can find that the target source code was developed by the MVC architecture pattern.

2. Legacy and target database:

First, the naming ways of table names and column names. Each database, table, and column satisfied the MySQL name convention general rules (Pandey, 2015): the database column names were singular, no space, and lowercase words; the column names were singular, lowercase and avoided number, space, abbreviated, and reserve words.

Second, primary keys and constraints. Each table's column primary keys are not null and uniquely identified the columns, which satisfied the database entity integrity rules and constraints (Watt & Eng, 2014).

Third, foreign keys and relationships among tables. Each table of the databases had its foreign keys and relationships, and each foreign key had its matching primary keys, which meant that each reference was valid. All of them indicate the databases satisfied the database referential integrity rules and constraints (Watt & Eng, 2014).

8. IMPLEMENTATION TECHNOLOGIES

The AWS cloud computing technologies used to implement the legacy system to a serverless web app are shown below:
First, AWS Amplify. AWS Amplify can be used to host the static web resources of the target web app.

Second, Amazon Cognito. Amazon Cognito can do user management and authentication for the target web app to allow users to register and login into the system and so on.

Third, Amazon API Gateway. API Gateway will be used as a RESTful interface to call Lambda function code.

Fourth, AWS Lambda. Lambda and API Gateway can form a public backend API, that is, a serverless backend, to send and receive data.

Fifth, Amazon Relational Database Service (RDS). RDS can manage and operate a relational database in AWS to store the app data.

9. CONCLUSION

With a CASE tool and the 5W1H re-doc methodology, the legacy KSEA NMC EMS was reverse engineered to some visual diagrams of different views. We used these visual diagrams to improve to be some visual diagrams of the target system. Based on the collected and analyzed database class diagram data, we conclude some best practices of making a good database. First, the web application design should consider using Object-Oriented Design and the MVC software architecture pattern to form a fixed frame and make the whole structure more logical. Second, the database's table and column names should follow the MySQL name conventions general rules. Third, each table should ensure the database entity integrity and referential integrity rules and constraints.

10. FUTURE WORK

Due to the complicated and huge KSEA NMC EMS, we only developed the legacy visual model and a small part of the target visual model. We also identified the implementation technologies, but without implementing the legacy NMC EMS to the new web app with AWS cloud computing technologies. Therefore, we plan to finish the whole target visual model and forward engineer a new EMS on AWS by the implementation technologies for future work.

11. REFERENCE

GeeksforGeeks. (2019, April 1). Unified Modeling Language (UML) | An Introduction. https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/

Jiang, J. Z. Architectural Blueprints–The "4+ 1" View Model of Software Architecture.

Deemer P., Benefield G., Larman C., and Vodde B. (2012). Scrum Primer. Retrieved from http://scrumprimer.org/scrumprimer20_small.pdf

KSEA (2021, February 3). 2021 KSEA Undergraduate and KSEA-KUSCO Graduate Scholarships Announcement. KSEA. https://ksea.org/us/

Bourgeois, D. (2019, August 12). Chapter 1: What Is an Information System? Information Systems for Business and Beyond (2019). https://opentextbook.site/informationsystems2019/chapter/chapter-1-what-is-an-information-system-information-systems-introduction/

AWS Products | AWS Cloud Service list | AWS. (2020). Amazon Web Services, Inc. https://www.amazonaws.cn/en/products/

Rungta, K. (2020, December 15). What is AWS? Amazon Cloud Services Tutorial. Guru99. https://www.guru99.com/what-is-aws.html

Morris, S. (2020, October 13). Tech 101: What is React JS? Skillcrush. https://skillcrush.com/blog/what-is-react-js/

Leslie, A. (2018). Attention Required! | Cloudflare. Hostingadvice.Com. https://www.hostingadvice.com/how-to/nginx-vs-apache/#performance

Morris, S. (2018, October 25). Everything You Need to Know About PHP. Skillcrush. https://skillcrush.com/blog/php/

Heddings, A. (2020, September 16). What Is Laravel, And How Do You Get Started with It? Cloudsavvyit. https://www.cloudsavvyit.com/1535/what-is-laravel-and-how-do-you-get-started-with-it/

What is MySQL? Everything You Need to Know | Talend. (2020, October 12). Talend Real-Time Open Source Data Integration Software. https://www.talend.com/resources/what-is-mysql/

Nguyen, P. V. (2011). The Study and Approach of Software Re-Engineering. arXiv preprint arXiv:1112.4016

Manzella, J., & Mutafelija, B. (1992, January). Concept of re-engineering life-cycle. In Proceedings of the Second International Conference on Systems Integration (pp. 566-567). IEEE Computer Society. 10.1109/ICSI.1992.217306

Waheed, M. A. (1992, November). A practical approach to re-engineering software. In Proceedings of the 1992 conference of the Centre for Advanced Studies on Collaborative research-Volume 1 (pp. 295-298).

Li, Y. (2011, May). Reengineering a scientific software and lessons learned. In Proceedings of the 4th international workshop on Software engineering for computational science and engineering (pp. 41-45). https://dl-acm-org.proxy.cityu.edu/doi/10.1145/1985782.1985789

Chikofsky, E. J., & Cross, J. H. (1990). Reverse engineering and design recovery: A taxonomy. IEEE Software, 7(1), 13-17. 10.1109/52.43044

Chung, S., Won, D., Baeg, S. H., & Park, S. (2009, December). Service-oriented reverse reengineering: 5W1H model-driven re-documentation and candidate services identification. 2009 IEEE International Conference on Service-Oriented Computing and Applications (SOCA). https://doi.org/10.1109/soca.2009.5410445

Tonella, P. (2005, May). Reverse engineering of object oriented code. In Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005. (pp. 724-725). IEEE. https://doi-org.proxy.cityu.edu/10.1145/1062455.1062637

Fouad, T., & Mohamed, B. (2018, January). Reverse Engineering of Object Relational Database. In Proceedings of the 2018 International Conference on Software Engineering and Information Management (pp. 73-76). https://doi-org.proxy.cityu.edu/10.1145/3178461.3178481

Ahmad, A., & Babar, M. A. (2014). A framework for architecture-driven migration of legacy systems to cloud-enabled software. In Proceedings of the WICSA 2014 Companion Volume (pp. 1-8). https://doi-org.proxy.cityu.edu/10.1145/2578128.2578232

Zalazar, A. S., Gonnet, S., & Leone, H. (2015). Migration of Legacy Systems to Cloud Computing. Electronic Journal of SADIO (EJS), 14, 42-55.

Ellison, M., Calinescu, R., & Paige, R. (2014, December). Re-Engineering the database layer of legacy applications for scalable cloud deployment. In 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing (pp. 976-979). IEEE. https://doi-org.proxy.cityu.edu/10.1109/UCC.2014.160

Sabiri, K., & Benabbou, F. (2017, March). A Legacy Application Meta-model for Modernization. In Proceedings of the 2nd International Conference on Big Data, Cloud and Applications (pp. 1-6). https://doi-org.proxy.cityu.edu/10.1145/3090354.3090385

Pandey, A. R. (2015, May 10). MySQL naming / coding conventions: tips on mySQL database. How To Tutorials. https://anandarajpandey.com/2015/05/10/mysql-naming-coding-conventions-tips-on-mysql-database/#:%7E:text=MySQL%20Name%20conventions%20general%20rules,not%20like%20project%2C%20james%2C%20e.t.c

Watt, A., & Eng, N. (2014, October 24). Chapter 9 Integrity Rules and Constraints – Database Design – 2nd Edition. Pressbooks. https://opentextbc.ca/dbdesign01/chapter/chapter-9-integrity-rules-and-constraints/

Tutorialspoint. (2021). Software Design Strategies - Tutorialspoint. https://www.tutorialspoint.com/software_engineering/software_design_strategies.htm