Dual-Step Phase Reverse Software Engineering Model for Legacy Software System

Ndigwe C.F.
Computer Science Dept. Chukwuemeka Odumegwu Ojukwu, University. Uli Campus, Anambra State

Okeke O.C.
Computer Science Dept. Chukwumemeka Odumegwu Ojukwu University.Uli Campus, Anambra State

ABSTRACT

Software reverse engineering is a unit of software engineering that specializes in using special tools for re-planning, re-engineering, re-designing and reuse of classic and deprecated applications or software systems. This paper proposes a Reverse Software Engineering Model (CRSE) for Legacy Software and deploy a use case scenario in validating the model in developing system. The research uses a hybrid of Object-Oriented Analysis and Design Methodology (OOADM) and the Agile Software Development - eXtreme Programming Methodology (XPM) as its methodology. The model and its analysis is presented in this paper. The Use case selected will be designed using the reverse Software engineering process model developed in the research.

General Terms

Reverse Software Engineering Model, Agile code refactoring, object-oriented analysis and design

Keywords

CSRE Model, Reverse software engineering, eXtreme programming, legacy software, classical application.

1. INTRODUCTION

The software engineers require different types of tools and documentation to perform maintenance of software and these tools, abstracted documents and artifacts (example requirements, design artifacts, architectures) in reverse engineering are required for re-planning, re-engineering, re-designing, reuse in the software systems. A well-known fact about technology today, is late for tomorrow, software working properly today may be deprecated in few months down the line. Reverse engineering often refers to moving from running binary into source code and then to design and back to code again. When it comes to the field of information technology, this profession requires a thorough study of the software needed to comprehend how these programs work. With the help of reverse engineering, it is possible to understand how the program works, what data it uses, where and what it sends, as well as what weaknesses it has and how it reacts to emergency situations. All this will help to further improve the software product (Shafagat, 2024). In modern software development old methods are replaced with the evolving newer methods, there is a transition into object-oriented programming which is more complex and structured, as well as forward engineering which are still the conventional norm in software development. There is a challenge of recovering classical useful applications using reverse software engineering. The Reverse Engineering techniques used in carrying out this task help to represent the system at higher levels of abstraction than code and the maintenance activities use the software reverse engineering techniques to represent the system at different levels of abstractions using the existing available source code and documents. Source code and available documents are used to extract different types of artifacts for maintenance tasks. In localization of very successful software strategies to illustrate the constructs, models, and methods of the framework reverse engineering rely on domain knowledge to identify intermediate artifacts first, which lead them towards their final goal (Thomas et al.,2024). Source code exist in many forms and many code are implemented in multiple languages or have different dialects or versions of compilers. There could be errors and difficult or impossible to compile cases or complete code is not available. Reverse engineering process is applied to recover the artifacts, which exist at domain, functional, structural and implementation level for the maintenance activities. The artifacts at implementation level are the files, the syntax and semantic of language and system components (program or module tree). The structural level represents how the system component are related and control each other, and at this level design is represented (example data flow, control flow and structure charts). The function level further abstract the system component or sub-components to reveal the relation and logic which perform certain tasks.

The domain level further more abstracts the functions/objects by replacing the algorithmic nature with concepts and specific to the application domain.

The artifacts are required to recover from the implementation, structural, functional and domain levels in varying levels of details for the maintenance activities. The artifacts exist in simple form to complex and require representing in different formats (example, UML diagrams). Some artifacts can be recovered at the same level but the artifacts like architectures require recovering many artifacts from different levels in varying details and abstracting it to form other artifacts. The reverse engineering is done at the implementation, structural, functional and domain level to abstract the artifacts and present it at higher levels of abstraction (Yumin, 2021).

In this paper some questions need answers like what are the factors on which the software reverse engineering process depends? Since Software engineers adapt different processes to perform the reverse engineering, what are the elements of software reverse engineering process do they use? The case study approach was selected to identify the factors and element of the reverse engineering process. The challenge this paper is to solve the problem of the process reverse engineering model currently in use which overlooks the challenges which include:

i) Current reverse engineering models are not tailored for the unique complexities of legacy systems, leading to inefficiencies in the reverse engineering process..
ii) Challenge of developing a good process model that have disassembler of classical application and Refactoring of source code to generate a useful and well design legacy

software that software engineers can use in building modern system.

iii) The problem of how to improve the process model to increase software Design Reuse (a key feature of software engineering) from legacy systems which were well designed but need to be redeployed in a new operational environment (example ***stand-alone app*** to full ***network app or online***).

The paper proposes a reverse Software Engineering Model for Legacy Software code named Chinwe Software Reverse engineering model (CSRE). These model intends to be a tailored reverse software engineering model specifically designed to address the intricacies of legacy systems. It can incorporate new processes that enable classical software without source code to be decompiled and the recovered code properly refactored and incorporate tools and techniques that enhance the reverse engineering process in the proposed model.

This proposed model will be important for use in the enhancing understanding of Legacy Systems. It will provide insight into the complexities of legacy systems, helping organizations to better understand and manage their existing software assets. It is expected to improve Reverse Engineering Practices by developing a tailored model, that will enhance current reverse engineering practices, making them more effective and applicable to diverse legacy systems. The proposed model will also offer clear strategies for integrating legacy systems with modern technologies, aiding organizations in their digital transformation efforts.

2. SOFTWARE ENGINEERING

Engineering is the practice of using science, mathematics and the engineering design process to solve technical problems, increase efficiency and productivity, and improve systems. Modern engineering comprises many subfields which include designing and improving infrastructure, machinery, vehicles, electronics, materials, and energy systems. Software is a set of instructions, data or programs used to operate computers and execute specific tasks. It is the opposite of hardware, which describes the physical aspects of a computer. Software consists of computer programs that instruct the execution of a task on the computer, it also includes design documents and specifications.

Software engineering on the other hand is the branch of computer science that deals with the design, development, testing, and maintenance of software applications. Software engineers apply engineering principles and knowledge of programming languages to build software solutions for end users. Reverse engineering is the process of analysing and understanding the design, structure, and functionality of a product or system by working backward from its final form. It is a powerful tool that allows us to understand, analyse, and innovate in the face of growing system complexity across industries.

Software engineering is the branch of computer science that deals with the design, development, testing, and maintenance of software applications. Software engineers apply engineering principles and knowledge of programming languages to build software solutions for end users. Software engineers design and develop computer games, business applications, operating systems, network control systems, and middleware and many other software (Michigan, 2024)

Software engineers develop applications using standardized process in almost any industry, including large and small businesses, government agencies, nonprofit organizations, healthcare facilities, and more. And as technology continues to evolve, the need for software developers in all areas of life continues to grow. Many companies are also shifting towards hiring software engineers who work from home, allowing for increased flexibility and more opportunities to enter the field. Software engineers build applications both over time they become obsolete and the technology used in building them get old to a level where they become imposable to maintain and yet very difficult to abandon due to its usefulness to the organization. These type of applications often need to be reverse engineered and rebuilt to old specifications that will make migration seamless.

2.1 Reverse Software Engineering

Reverse engineering (also known as backwards engineering or back engineering) is a process or method through which one attempts to understand through deductive reasoning how a previously made device, process, system, or piece of software accomplishes a task with very little (if any) insight into exactly how it does so. Depending on the system under consideration and the technologies employed, the knowledge gained during reverse engineering can help with repurposing obsolete objects, doing security analysis, or learning how something works (Yumin, 2021) .
Although the process is specific to the object on which it is being performed, all reverse engineering processes consist of three basic steps: information extraction, modeling, and review. Information extraction is the practice of gathering all relevant information for performing the operation. Modeling is the practice of combining the gathered information into an abstract model, which can be used as a guide for designing the new object or system. Review is the testing of the model to ensure the validity of the chosen abstract (Zhang, 2021).

Software Reverse Engineering is a process of recovering the design, requirement specifications, and functions of a product from an analysis of its code. This is the approach deployed in reverse software engineering challenge in contemporary projects (Thomas et al. ,2024). It builds a program knowledge base and generates information from this as a modern software engineering process involving refactoring. This article focuses on discussing reverse engineering in detail. Reverse engineering can extract design information from source code, but the abstraction level, the completeness of the documentation, the degree to which tools and a human analyst work together, and the directionality of the process are highly variable.

3. ANALYSIS OF THE EXISTING SYSTEM

This analysis presents the software reverse engineering process needs in an effort to design a new reverse Agile methodology aimed at in this dissertation. The existing system is the work of Fikri et al., (2020) clearly illustrated in the Architecture of Existing Reverse Engineering Process in figure 3.1. The model work by getting the source code first and conducting a Context parsing which generates intermediary representation format that result to component analysis. The Component Analyzing results to Component Specification and Component interrelationship which leads to Design Recovery. When the process reached design recovery, Domain Knowledge and Documentation are feed into the design recovery to produce the new System Information and System Requirement. The feedback goes to Design reconstruction which receives expert knowledge to produce the Design Model.

The fact that nowadays many source code from legacy applications may be well designed and poorly documented creates complicated situations when wanting to perform system repairs. Because it requires an expensive fee to replace all the source code on legacy applications, then the only sensible alternative is to maintain and develop existing source code. Much time is spent to understand a system before making improvements increasing the economic side because the development of software systems continues to grow, both in terms of size and also complexity. To help reduce the cost of understanding the program suggests that practicing with the reverse engineering technique increases the ability to understand the system rendered quickly and efficiently..

Reverse engineering is the process of analyzing the subject system for identifying system components and relating, and making system representations in other forms or on Higher levels of abstraction . In the existing system every method that aims to recover knowledge about software systems exists in supporting the execution of software engineering Tasks. Based on some previous literature that there has not been found the implementation of reverse engineering focusing on information architecture and also system usability. The existing literature only describes the implementation of reverse engineering to improve the existing system in a broader context. in an exhausting overview of the popular methods that can be applied to support reverse engineering. One of these is the method of design recovery: This method aims to restore the design abstraction from the source of available information. Part of the restored design is the domain model. Design abstractions are recreated by combining informal information, existing documentation, and source code.

Design recovery is a vast field. Therefore, most of the research focuses on subareas. Concept assignment issues try to find human-oriented concepts and connect them to locations within the source code. Often this is further divided into concept recovery and concept location. Concept location is at a lower level, while concept recovery has become a very active field of research in the reverse engineering community especially in the area of system design recovery.

The reverse engineering process was done with the first phase, the context parsing to analyze the source code contained in the system and then parsed the source code and changed the source code to a more structured representation such as the conversion of source code of Machine language to a low level language that is easier to understand. The second phase is to take the intermediate program structure to reveal component artifacts such as the current information architecture chart. Moreover, the third phase is the result of the extraction of design later in the analysis and concluded in the form of information at a higher level. In this phase also carried out the existing navigation improvements using the card sorting method. The card sorting method is done, which is hybrid card sorting.

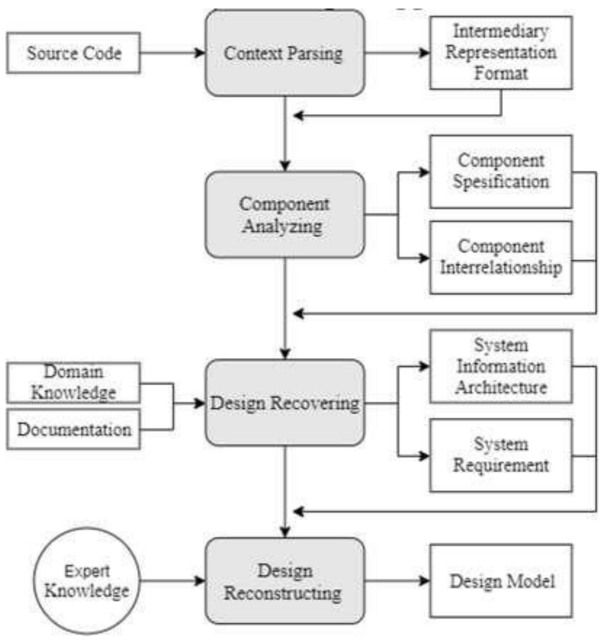

Fig 3.1: Architecture of Existing Reverse Engineering Process (Fikri et al., 2020)

Card Sorting is one of the methods of Information Architecture, where the respondent will be given a random collection of cards and then asked to classify the cards into a specific label. On open card sorting where respondents were asked to sort and classify the cards that are available according to the views that they think are most reasonable according to them. Then they will then describe each group created. This method is one of the generative methods done in the pre-design phase to help build the structure of information to be created. In the closed card sort, the respondent will be given a pile of cards along with the group category that has been specified. The respondent will then be asked to classify the cards into pre-defined categories according to their most sensible order. This method is an evaluative method done on post-design to evaluate and analyze the website created according to the previous research results. In addition, there is another category for the card sorting technique that is hybrid card sorting, wherein this technique is a combination of open method and closed card sorting. This study, hybrid card sorting techniques were used due to being more reproductive and also evaluative. It helps to generate ideas and also evaluations on how to re-engineer the information architecture of the website. Hybrid card Sorting is regarded as the most useful framework for designing and enhancing the information architecture of any website (Fikri et al., 2020).

In the existing system uses the term the structural planning of information space of the system consisting of three systems: 1) System organization, namely the system for organizing content to pages and managing the relationship between them, 2) Navigation System, which is the system to allow users to move from one page to another, and 3) the labeling system, namely the system for page naming and navigation functions. Information Architecture (IA) refers to the arrangement of digital information space to provide users with simple and intuitive access to content and functionality . IT on websites significantly affects usability and user experience during the navigation.

Nowadays, much information comes through the website, so the importance of a website in communicating the information in the right way becomes very important. The website provides the advantage of being globally accessible through an Internet-connected computer network, as well as from information available Online to quickly answer user questions. The website is an information service that is accessed by many Internet users because it can handle the demand of many users with good quality (reliable).

Information architecture involves several activities in order to remain maximally in making the right decision. First organizing and simplification of information, both design, third integration and aggregation of space or information systems, the four create ways for people to discover, understand, exchange, and manage information. Information becomes very important because the right decision does not rely on adequate information. The wrong decision makings occur mostly due to the poor quality of the information provided, such as erroneous data or incomplete data. A system that has large-scale information and continues to grow in a fast-paced time can cause the possibility of people to fail to find more valuable information.

Organizing information is one of the most influential factors affecting the way users think of and interact with the interface. Because of the Institute of Information, architecture is the art and science of organizing and labeling systems, intranets, online communities, and software to support usability. Usability is very related to the extent to which the user's mental model matches and predicts system actions. The mental model has been used in human-computer interactions and improving usability. Users may not always have an optimal mental model. Designing a system based on an imperfect user's mental model (expert knowledge) can impede user performance. So, it is crucial to know the user's representative mental model with regards to the software structure, as it allows a designer to create content according to the user's expectation, thus making the resulting design as intuitive as possible for users.

Reverse Engineering is the process of digging knowledge or useful information from an existing product for use in modelling a new one (Gao,2021). Reverse Software Engineering is usually used to design and modify a product based on an existing product. Reverse engineering techniques can be used in the process of evaluating product quality or in the process of redesigning products that are not available for development documentation. Initially, the concept of reverse engineering has been used only for hardware projects, but now it is also very much used in software, although very few has implemented reverse engineering on the information architecture of a system. Software Reverse Engineering is used as a tool to acquire lost knowledge, ideas, and design philosophies, thereby collecting best practices from already developed applications. In this research, reverse engineering will be applied to improve the existing model and the system architecture part, especially in the process navigation section. The reverse engineering process can be divided into four phases: Context Parsing, Component Analyzing, Design Recovering, and Design Reconstructing. However, this research will only end in the Design Recovering phase.

3.1 Weaknesses of Existing system

The existing system made a lot of assumptions which are weak on:1) **The stage of design recovering and design reconstruction**. It assumed that design methodology used in the classical software that will be reverse engineered will be the same with the design methodology of reconstruction which is the new system that need to be developed from the classic system. The reality is that software development methodology have changed over the years and will continue to improve and the design methodology and tools used when the classical software was developed could no longer be relevant at the time of reverse engineering the software. Though the knowledge extraction in the classic application may be very relevant but the tools of presentation in the design recovering may be different and may belong to a different methodology.

2) **Software Reverse Engineering process often does not begin from source code:** The model in the existing system process architecture shows that the beginning is source code but often that is not the case. In reality many classical software source code may not be available at the time the software is to be reverse engineered. Sometime the company or organization or developer team that developed the software may not be the team that need to reverse engineer it making the source code unavailable. However the running classical software are always around and will need to the decompiled to generate the source code hence the process need to properly begin with software decompilation.

3) **Post Code decompilation may not require parsing**: After decompilation and the code is available in source code format there is no need to pars the code for extraction of the logic of the source code or its knowledge base rather the system just need to be **refactored**. Refactoring is a state in Agile Software development process which can be applied at that point to generate a new design from the source code.

4.) The process have no redesign process: In the existing system the architecture did not include the redesign process either as a block or as a detail process diagram. The redesign if very vital to illustrate the step required to reintegrate the generated knowledge from the classical system into the new system which is the essence of the reverse engineering process.

3.2 Proposed Reverse Software Engineering Model

The proposed system is the Chinwe Reverse Software Engineering (CRSE) model which introduces some components which are different from the ones used in the existing reverse engineering process models and are Agile like in extreme programming. The component include the decompilation process which is included for classical application that have no source code. This is the process where the application in binary is first decompiled to generate the source code. In the existing system it is assumed that the source code is readily available which is always not true. However when the application is in the binary form the model uses decompilation to generate the source code before the source code is subjected to Context parsing. The new model offer two options where there is need for parsing and were parsing can be avoided. If there is need for context parsing, the source code is parsed to generate Intermediary Representation.

The Intermediary Representation Format is subjected to component analysis if the old Component Specification is available, the analysis will produce component interrelationship which is feedback to design recovery. The source code can also be simply send for refactoring which often required Domain Knowledge Extraction which is injected into the refactoring as useful tool for design recovery. Once the design Recovery is done the new System requirement and the System Information Architecture emerges which is readily feed into the Design Reconstruction to generate the new design for the new system that will be built.

Major components used in CRSE development include the process for the decompilation and the process for the refactoring of the source code to generate the new design. In figure 3.2 the architecture of the proposed system is presented which highlight all the processes analysed as components and how they interact with each other.

CRSE is an Agile Software Reverse Engineering model proposed in this thesis and has to be agile in nature, principle and philosophy. The methodology must also abide by the Agile Manifesto in its reverse conceptual form. CRSE is believed to be a dynamic process of building reverse software and dynamically specified since reverse software is expected to result to a new improved system with design features of the classic application. In figure 1 the proposed system considers the two main form of classical applications, the application that have source code and the one that the source code is not available. If the source code is available then there is no need to decompile the application, however if the application source code is not available the application itself need to be decompiled. These application whose source are not available are often in binary format or they may be in bytecode (for applications using virtual machine or run time to execute such as Java or dot Net). These application file in that format need to be decompiled to extract the source code. The source code can then be sent for context parsing to extract domain knowledge or the source code can be directly be refactored using Agile software method refactoring technique for design recovery. On the other hand the parsed code is represented in intermediate format before a detail component analysis is carried out on the system before it could be refactored or directly proceed to design recovery. In design recovery the new system information architecture and the new system requirement are integrated to the recovered design from the classical system to create a design reconstruction. During design reconstruction expert knowledge is injected into the system for the creation of a new model. The new model comes out as an improved system with the classical application that have being reverse engineered as the base design of the system. This have the benefit of inheriting all the merits of the classical system with improvement made during the design reconstruction and injection of current domain knowledge of modern application environment.

The CRSE model proposed in this thesis emphasizes **state** instead of **stage** in reverse software development cycle and because state is not only transient but also transitional, any flow is not only possible but adoptable to varying needs at varying times and dependent on availability of source code. **State** is the level the process is at the point of the reverse software model as presented in the architecture. It focuses more on activity to be accomplished than the steps taken to achieve that. **Stage** on the other hand emphasizes process in a rigid path within a state such as analysis involving component specification and component interrelationship. In a **stage**, the activity that must be done is not emphasized as much as the steps that must be followed in performing the activities. This has been the lot of many non-Agile methods and even some Agile methods. The reasoning has been which steps are best: analysis before design, design before analysis, design before implementation or testing after implementation or refactoring after implementation or before implementation and so on. However, in an attempt to fine tune each **stage** of the process before going to another, they have dug deep into some of the noble activities to be performed in reverse software development. Some of these deep activities get deep enough to distract developers from the primary measure of progress -Working Software. This is a major justification identified for the CRSE model. We also realize that these deep issues could also be important in getting software in another domain to be working in another domain once reverse engineered. CRSE believes that a **state** should transit to the activity which will get the reverse software working at a reasonable time span - complying to Agility.

3.2.1 CRSE Model Benefits

CRSE Process is an agile reverse software engineering process that is predicated on a set of agile principle and reverse software engineering steps and states that should be followed as stakeholders move from classical software to a reverse engineered software project. These offers :

i. Clear Road Map
The existing model does not clarify the state of action when an application is to be reverse engineered, it simply assumed that the application source code is available which is not correct bearing in mind that old software source are often in position of defunct companies which often are not available. The proposed system have introduced a clear action when the source code is available and when they are not available so the software developer can be clear about the action to be taken.

ii Integration of Agility
CRSE process is a perfect integration of steps model and state model to achieve the flexibility of agility without over simplification. CRSE process encompasses a set of activities and states that are inherent in Code Refactoring that leads to re-specification, verification and reconstruction of the classical software using agile principles. CRSE seem to provide a mechanism that overcome incompleteness, inconsistency, over simplification of process and analysis-design refactoring.

iii Object-Orientation is localized at Reconstruction stage
CRSE uses agility which implements its localized reverse software features, functions, and information content as components such as objects and classes constructed within the context of a system architecture. It handles customer requirements within the entire architecture using the data-method bonding if the classical software is procedural. The architecture therefore provides for the Unknown from the context of the known requirements and also from all the drivers required for the development of the system that will meet the customers' need today and make provision for the customers' possible need tomorrow. CRSE is simple but yet rich in guiding reverse software engineers in development through its process model.

iv. At each Stage a Verifiable Outcome can be produced
The proposed system is capable of producing set of output for each transition which can be used in the next iteration to compare with the requirement specification to make sure that the reverse software is moving towards the final target of production at the point of New Model production. It can produce state outputs which will be available for state testing during the model life cycle. This is tagged CRSE Reverse software process machine testing and it is a behavior model composed of a finite number of states transitions between those states, and activities, similar to a flow graph in which one can inspect the way logic runs when certain reverse software development conditions are met. CRSE machine is a finite-state machine yet the inverse is not true.

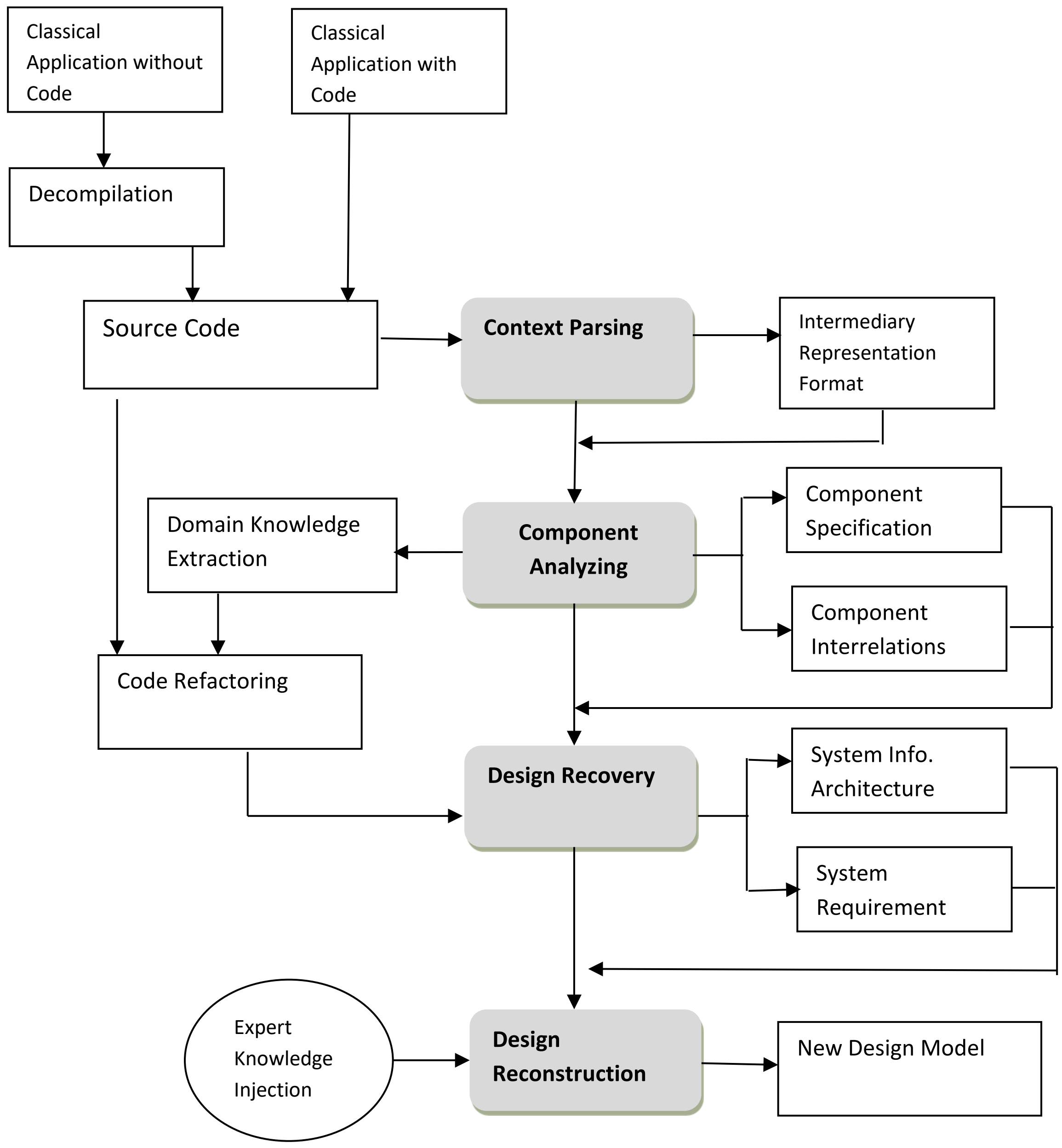

Fig 1: If necessary, the images can be extended both columns

v. Provision of Agile Refactoring

A serious justification of the proposed system is its ability to control the process in CRSE once it commences after the project has been initiated. It begins with the drafting of Agile-lists which itemizes the features and operations that the reverse software need to carry out. Agile-list (AL) card shown in table 3.1 is a special card that has two major divisions. The left part has 70% of the entire size while the right has 30%. The left part is used for itemizing the feature number (Fno), owner/user's features, Date the feature was specified (Date Sp), expected time of completion (ET), actual time of completion (AT), expected cost (EC), actual cost (AC), and percentage completion (% C) operations. The right is used in listing the Path Unknown (PU) features, unknown expected cost (UEC) and unknown expected time (UET) operations. The PU serves as balancing criteria useful for keeping the reverse software process along the Path throughout the development process. Customers decide what appears on the left of the AL card while the CRSE team marches it with their unknown corresponding features and operations.

4. OBJECT-MODELLING OF THE PROPOSED SYSTEM

Refactoring introduced in the proposed system need object modelling and this is presented in simple detail in this section. Refactoring is the process of restructuring code, while not changing its original functionality. The goal of refactoring is to improve internal code by making many small changes without altering the code's external behavior. Software reverse engineers using the proposed model refactor code to improve the design, structure and implementation of software. Refactoring improves code readability and reduces complexities. Refactoring can be done by binding data with functions to for classes and objects so that the benefit of object modelling can be derived after the new model is generated during design reconstruction. A Class represents a collection of similar objects. Objects are things of interest in the system being modeled. They can be a person, place, thing, or any other concept important to the system at hand. There are many ways to identify classes. One of the easiest to start with is noun extraction. Noun extraction identifies all of the Nouns in a problem statement and/or use-case scenario. The nouns extracted make excellent candidate classes.

Other ways to identify classes are to look for items that interact with the system, or things that are part of the system in code. We can ask for a customer of the system, and identify what the customer interacts with. Screens and reports represent interface classes, but for the sake of a CRC session, a single GUI class can represent these. If a class can't be named with less than three words, then it's probably not a class but a responsibility of another class.

A Responsibility is anything that the class knows or does. These responsibilities are things that the class has knowledge about itself, or things the class can do with the knowledge it has. Just like the 'Noun Extraction' technique for classes above a verb extraction can be used to identify responsibilities. Verb extraction identifies all of the verbs in a problem statement and/or use-case scenario. These are usually good indicators of actions that must be performed by the classes of the system. Other techniques included asking what the class knows, and what information must be stored about the class to make it unique. Ask what the class does, and then flesh out how the class might do it.

Collaboration occurs when a class needs information that it doesn't have. Classes know specific things about themselves. Very often to perform a task a class needs information that it doesn't have. Often it's necessary to get this information from another class, in the form of collaboration. Collaboration can also occur if a class needs to modify information that it doesn't have. One property of information that a class knows about itself is the ability to update the information. Often a class will want to update information that it doesn't have. When this happens, the class will often ask another class, in the form of a collaboration, to update the information for it. Generally for a collaboration to occur, one class is the initiator. In other words, there has to be a starting point for collaboration. Often times the initiating class is doing very little work beyond initialing the collaboration itself. However, one need to be careful that a class doesn't just 'pass the buck', just to have it passed again. For example, if class A collaborates with class B, just to have class B pass the work to class C, consider eliminating the collaboration with class B. The CRSE team performs the design exercise using the CRC cards, CRSE Abstraction Conceptualization (EAC) and an Agile Computer Aided Reverse software Engineering (ACASE) tool in their design work.

If a difficult design problem is encountered as part of the design of an Agile list, CRSE recommends the creation of an operational prototype of that portion of the design. It also recommends quick transition of that part of design to revalidation and reanalysis; that is why analysis and design in CRSE are recursive via Agile design revalidation and Lambda transition. The intent is to lower risk when true implementation starts and to validate the original estimates for the requirements containing the design problem.

A central notion in CRSE is that design is the fulcrum coordinating analysis, coding and maintenance it can occur both before and after coding commences. Refactoring means that design occurs continuously as the system is constructed. In fact, the coding and testing activity itself will provide the CRSE development team with insight on how to improve the design.

4.1 CRSE Tool

CRSE need a tool for its usage and implementation this tool is referred to in this thesis as CRSE Tool. It is based on the process activities such as project or task initiation, analysis, CRSE design, implementation and other project activities. The tool enable developer using the methodology to keep track of their project so that it can be monitored by management and client of the development project. It also enable CRSE teams to communicate and to have accurate record of project milestone so that costing and timing of various activities can be properly accounted for during the reverse engineering project life cycle.

In the tool the Stakeholders, Actions, and communication factors where paramount in CRSE effort to promote people-communicating-with-people rather than people-process interaction of classical methodologies.

4.2 Stakeholders

The stakeholders in CRSE include the project client, the development team, the user of the application, the owner of the classical application (reverse engineering a software may need permission of original owner) and other staff assigned to the project either by the development team leader or the project client management. The tool provides them with a facility to create users. This will be done for them by the tool administrator.

4.3 Actions

The actions provided in the CRSE Tool include decompilation tool, Backlog, Pipeline, All Project, Releases and Status.

Decompiler: This is software that is used to convert the binary or bytecode of the system to source code there are many tools however the one that needed to be used depends on the original type of the application and the programming language used in initially .

Backlog: This provide backlog information on projects ready for refactoring. These projects could be at various levels of implementation which include New or Initiation, Analysis, Development, Testing, Implementation, Completed or cancelled. Backlog also provides different backlogs for projects Tasks as well as Bugs.

Pipeline: Once a project is assigned to each group in the team. The work at hands which may include the listing of development classical application, source code or refactored method in a class or assembling of classes within a given interface.. Every team members has his pipeline which can be searched using the Ticket created for that pipeline. This will help in gathering information about the progress level of the project to help in the decision process about exchanging developers in-between project and for metric reasons. The progress of the project can be easily determined using these pipeline report

Ticket: The ticket information provides a simple way of estimating time of the project and possible cost based on business value rating and complexity rating by developers. It also provide an easy means to track team members based on the ticket title of their assigned job. The CRSE Tool is use throughout the reverse software engineering period of the project to evaluate the system to measure if the reverse engineering time and cost is worth the effort or if it is just better to build a new project from the scratch.

4.4 Use Case

The proposed Reverse Engineering model for software engineering projects, the CRSE an Agile Software Reverse Engineering model need to be tested. The design objective is to use a use case scenario to design the system using the CRSE model as a software reverse engineering design process. The CRSE is believed to be a dynamic process of building reverse software and dynamically specified since reverse software is expected to result to a new improved system with design features of the classic application. In this chapter we shall deploy a use case to implement our model. This use case is an online bookshop scenario, Csoft Nigeria Limited as the software development company and Ubookshop Limited as the system owner; (these are hypothetical companies). The book buying public is the system user. The existing Software classic is a system developed using Java but need to be reverse engineered to fit a PHP system in a new environment. We have chosen PHP, an open source object-oriented programming language as the Language of implementation and MySQL as the database engine for storing the information and data required for the system. Detailed procedure will also be explained.

4.4.1 Scenario Description: Classical Application without Code

Software engineering from CRSE perspective is more than concepts, tools, techniques, methods and processes. It is about people working and interacting well with classic software. In our scenario, Chinwe Ndigwe, who is the Reverse software engineer and director Csoft Nigeria Limited (Csoft) receives a letter from Jonah the Software developer and project secretary Csoft. Chukwudi Nda is the System analyst/programmer at Csoft is seated. Musa Umar Programmer I at CSoft is busy on the keyboard, while Gbenga Olukoya the marketing manager at CSoft just works into his desk with a client request demanding that a classic Bookshop software be made to adapt to their new work environment with Apache, PHP and MySqL as their new tools. It was further discovered that the application he has brought for Reverse engineering does not have a source code.

4.4.2.1 Project Use Case Scenario

Scenario: Worlu worked into the open office and announced that Ubookshop has indicated interest in building an online store to boost its sales to a wider market after a proposal is submitted to them. Mean while there is a classical Bookshop app that have been working but the source code is no longer available. A project team is immediately selected to meet Ubookshop for more clarification. A meeting holds between CSoft and Ubookshop and an agreement is reached to build an online store for Ubookshop. Management, the Csoft Reverse Software Engineering Team and the UBookshop Operator acting as owner of project. The Actions they take to initiate to project include Announcing UBookshop intent, send info to Csoft management, hold a reverse software Engineering meeting, select Csoft Project Team and Schedule Decompilation. These actions are performed by different actors the detail is derivable in the full design.

5. CONCLUSION

In conclusion a reverse software engineering process model have being proposed in this paper which can be used to guide the development of new application from a classic program. The developed CRSE model in this paper can guide software engineers in reverse engineering their classic programs using the process specified in the model..

6. RECOMMENDATION

In the previous chapter we have analysed and proposed a new Reverse Engineering model for software developers. A use case scenario was presented which can be used to test the system. We recommend that these use case should be used to test the system and new software development scenario where the code is present or cases where the source code is lacking but the application is available for decompilation to recover the source code. This software challenges can be developed using the process model proposed in this work, modelled and reverse engineered to produce modern system that preserve critical design in the classic system.

7. ACKNOWLEDGMENTS

Our thanks to Eke Bartholomew (Phd) the experts who reviewed the work and pointed corrections that led to the development of the CRSE model in this paper.

8. REFERENCES

[1] Shafagat, M. (2024) The essence and challenges of reverse engineering, Industrial Engineering and Strategic Management, pouyan press www.iesmj.com 1-7

[2] Thomas F., Tab Z., Willem V. I., Gertjan E., Bart C., Christian C., and Bjorn D. (2024) Tools and Models for Reverse softwareReverse Engineering Research. In Proceedings of the 2024 Workshop on Research on offensive and defensive techniques in the context of Man At The End (MATE) attacks (CheckMATE '24), Salt Lake City, UT, USA.

[3] Yumin S. (2021) Reverse Engineering Technology and its Application, World Scientific Research Journal, 7(8), 137-143

[4] Michigan Tech. (2024) What is Reverse Software Engineering, Michigan Technological University https://www.mtu.edu/cs/undergraduate/software/

[5] Zhang J. (2021) Application of Reverse Engineering Technology in the Field of Mechanical Manufacturing, Technology and Market, 28(2021) No.16, p.138-139.

[6] Fikri M, Kusumawardani S. S., and Ferdiana R. (2020) Reverse Engineering Website Navigation Using an Information Architecture Approach (Case Study: Kanal

[7] Pengetahuan Universitas Gadjah Mada), Journal of Physics: Conference Series 1577, Series IOP Publishing doi:10.1088/1742-6596/1577/1/012054

[8] Gao Yu, L I Wei-min, ZHAO Xu-dong, (2021): Current Situation and Prospects of Reverse Engineering Technology, Journal of Liaoning University of Technology (Natural Science Edition),41(2), 90-94,128.