Comparative Study between Clustering and Model Driven Reverse Engineering Approaches

El Beggar Omar, Bousetta Brahim, and Gadi Taoufiq

Abstract—A new or an improved system which replaces a previous one completely from scratch is very expensive, and represents a huge risk. Furthermore, the legacy software embeds a significant amount of knowledge over time that would be lost if it is entirely replaced. Reverse engineering is the best solution for obtaining improved systems without discarding the existing. In this paper, we will compare two approaches that aim recovering objects or classes from legacy systems.

Index Terms—Clustering, metamodel, model driven engineering, model transformation, reverse engineering.

I. INTRODUCTION

The older information systems ran on mainframe environment and they are often written in COBOL and store their data in files [1], [2]; they are usually large and complex and known as legacy systems. These legacy systems need to be maintained and evolved due to many factors, including anomalies correction, requirements change, amendment of business rules, etc [3]. But, there are many problems to maintain and evolve legacy systems like the difficulty to retrieve and understand the original system specifications and especially when it’s a lack of documentation supports related to those systems. The higher cost of maintaining and evolving legacy systems represent also a challenge to surmount.

Reverse Engineering is a discipline that aims to extract design information, functional specifications and, eventually, requirements from the source code, documents and any other available information that can be useful to maintain legacy systems or redevelop them [4]. Reverse engineering preserves the legacy knowledge of the systems and makes it possible to maintain software easily at a low cost. Recovering objects or classes and object oriented models from programs is one of the most important areas of research related to reverse engineering. Furthermore, COBOL applications present the most important target of reverse engineering projects. The choice of COBOL is not arbitrary due to the wide diffusion of this programming language and its existence all these years in spite of the presence of other languages more sophisticated. This paper present a comparative study between the reverse engineering processes based on clustering techniques and the Model Driven Reverse Engineering (MDRE) for recovering objects from legacy systems. The remainder of this paper is organized as follows: the Section II will be dedicated to experiment the first approach based on clustering techniques. In other hand, the Section III will be consecrated to apply the second approach based on model driven principles. The next Sections V and VI will be devoted respectively to discuss and evaluate the proposals and introduce the related works. Finally, in section 6 we conclude our work.

II. CLUSTERING APPROACH

The main purpose of this section is to identify the record fields that are related functionally through their occurrences in paragraphs. Clustering is a technique for finding related items in a given set. Firstly, we will give briefly an overview of the clustering approach. Next, we will present our performed cluster analysis experiment.

A. Clustering Approach Overview

Clustering consists of organizing data into groups (clusters) using a similarity distance or measure, where the similar data are in the same group. The clustering techniques can be applied in many fields such as segmentation of customers according to their purchasing habits, grouping homes in city quarters or identify new plant or animal species. There are many similarity measures used in clustering such as Jaccard distance, Euclidean distance and Manhattan distance, etc. Usually, the clustering techniques requires constructing a similarity matrix based on a given input data matrix and next an algorithm is applied to identify clusters. There are two main categories of clustering algorithms: Hierarchical algorithms and Partitional algorithms.

B. Cluster Analysis Experiment

In this subsection we will explain the clustering techniques that we have applied; firstly, our proposal starts by extracting data from source code and their occurrences in different program paragraphs in order to construct the input data matrix. Generally, the structure of COBOL programs is a hierarchical structure; it contains four main divisions (Identification, Environment, Data and Procedure). Each division may contain one or more sections. And a section may contain one or more paragraphs. It is important to select the paragraphs containing domain-related functionalities. A System Dependence Graph [5] can help to identify such paragraphs. Our cluster analysis is based on the hypothesis that record fields existing in the same paragraphs can be grouped. The chosen running example comprises four paragraphs within two records containing fields. For each field in record, we determine whether or not it is used in a
The chosen distance of similarity to identify clusters is the Euclidean distance. Many other distance measures are possible. The Euclidean distance is calculated as follows:

$$\text{distance}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad (1)$$

While, Table I shows the input data matrix that is used for the cluster analysis. The similarity matrix (see Table II) is deduced from the input data matrix by means of calculating the Euclidian distance between fields. For example, the distance between ID_CMD and NAME is calculated as follows:

$$|\text{NAME} - \text{ID}_\_\text{CMD}| = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} = \sqrt{3} \approx 1.73 \quad (2)$$

Calculating Euclidian distance allows constructing the similarity matrix that can be used as input to the clustering algorithm. In our proposal, we used a hierarchical clustering algorithm known as (Johnson’s algorithm) [6]. This algorithm starts by $N$ clusters each containing one field, and then proceeds by computing the similarity between clusters and merge the closest clusters in a single one until remaining only one cluster, and the algorithm terminates. The distance between clusters is calculated using the average distance between all elements in the 1st cluster, and all elements in the 2nd cluster.

<table>
<thead>
<tr>
<th>Field</th>
<th>PG1</th>
<th>PG2</th>
<th>PG3</th>
<th>PG4</th>
<th>PG5</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID_CUS</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NAME</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ID_CMD</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CMD_DATE</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CMD_CUS</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The objects c1 and c3 are associated via CMD_CUS as it is shown in Fig. 1. The same clustering analysis was applied on many programs to find the other objects and their relative associations.

The dashed cut line marks a possible segmentation. Consequently, the resulting objects issued from this clustering analysis correspond to the cluster c1 that regroups the customer’s data and c3 that specifies the command’s data. The objects c1 and c3 are associated via CMD_CUS as it is shown in Fig. 1. The same clustering analysis was applied on many programs to find the other objects and their relative associations.

III. MODEL DRIVEN REVERSE ENGINEERING

The Model Driven Engineering (MDE) considers models as being the most important element for software development, maintenance and evolution through model transformation [7]. It is also a code generative approach which produces code application from model using a different kind of automated model transformation. According to the Object Management Group (OMG) standard Model Driven Architecture (MDA) which is an instance of MDE since 2003, a model transformation is the process of converting one model into another model via a fine-grained rules that transform elements defined in a specific source metamodel into other elements of the target metamodel (see Fig. 4). MDA approach defines also models at three different levels of abstraction [8]: The computation independent viewpoint (represented by the Computation Independent Model or CIM); the platform independent viewpoint (represented by the Platform Independent Model or PIM); and the platform specific viewpoint (represented by the Platform Specific Model or PSM). However, the reverse engineering process aims to produce a second version of the original system by means of producing high-level models of
the studied systems [9]. The model provides a high-level representation of some knowledge embedded in code programs that can be reused in other context or simply generated in another form. Therefore, the reverse engineering process may use the MDE paradigms and principles and produce a new approach entitled Model Driven Reverse Engineering (MDRE).

A. MDRE Approach Overview

In this subsection, we will introduce our MDRE approach and the activities performed in each phase. Our MDRE approach consists of identifying objects from records descriptions via three phases: Extraction, Merge and finally Transformation and Refinement. In the first phase, we analyze the source code and we extract the data description which exists in the Section “File Description” in COBOL programs in order to create a PSM model conforms to the COBOL file description metamodel. The second phase of our MDRE approach consists of merging the entire PSM model extracted in the first phase and generating a common model that we called the Merge Model of File Descriptors (MMFD). The resulting MMFD contains all data structure used in the different source codes. The Transformation and refinement phase is dedicated to perform a reverse transformation from the PSM: MMFD to the Domain Class Diagram (DCD) using ATL language. Afterwards, we refine the DCD by applying a set of heuristics that extract information from the legacy data embedded in storage files such as associations, multiplicities, objects identifiers and equivalent types to fields group, etc.

B. PSM Metamodel

The PSM metamodel (see Fig. 2) specifies the structure of legacy data to extract and to transform in our MDRE approach. Thus, each file’s record contains fields of three kinds: Simple data item, Group data item and Array item. The group data item can include other data items or fillers (variables without name). Each data item has a level, a name, a type which exists in “PICTUREKIND” enumeration and finally a size. Some of those fields can represent record keys.

C. Merging PSM Models

The second phase in our MDRE approach consists of merging the entire PSM models related to file descriptors extracted from legacy programs in order to create a common model entitled the Merge Model of File Descriptors (MMFD) which regroups the whole files descriptors programs. The MMFD is also conforms to the previous metamodel. The MMFD will be the source of our model transformation to obtain the target PIM: domain class diagram.

D. Domain Class Diagram Metamodel

The domain class diagram is one of the leading diagrams in UML modeling. It allows dissecting the system by showing its different classes, their attributes and methods; it provides a static view of the object-oriented system. We note that The DCD metamodel shown in Fig. 3 is a reduced and simplified UML class diagram [10], [11] that respects the MOF specifications [12]. The meta-class “Class” which may inherit from other classes is composed of properties and operations. A property can be simply an attribute or an association end. The operation can contain parameters and it can return a value. The property, operation’s return and its parameters have a type that can be object class or a simple DATATYPE. A property can represent the class identifier; in this case its feature is ID must be true. Concerning the property features lower and upper, they represent the different kind of association multiplicities.

Thus, once we have defined the metamodels, the remaining step to complete automating of our reverse engineering process according to MDE principles is to define the mappings between models in order to specify how each different considered model is going to be generated during the process. We can refer to this mapping as model transformation [13]. To perform the model transformation, we used the ATLAS Transformation Language (ATL) [14], [15] that is a domain-specific language for specifying model-to-model transformations. It is a part of the AMMA
It is also considered as a hybrid language, i.e. it provides a mix of declarative and imperative constructs. Table III shows the main mapping rules written in a natural language before performing the model transformation in ATL.

**E. DCD Model Refinement**

To the best of our knowledge all previous works have done very little research on acquisition of information stored in files. Meanwhile, such data files contains vital information that can be retrieved implicitly [16]. Our proposal used also the data embedded in flat files in order to get out the association multiplicities, objects identifiers, equivalent types, etc. used to refine the DCD.

1) **Primary key rule**

Assume that we have different occurrences of Record R1 in data files. We determine the record key RK or the primary key by choosing the minimal field(s) where their values are unique. If they are candidate keys we exceed to other criteria: the type and the length of the field(s). We should choose an integer and a shorter (minimal length) one. “(3)” recapitulates what was explained above.

\[
R_{1_{kk}} = \begin{cases} 
\exists i \ F_i \in R1 \\
\forall j \neq k, V_j \land V_k \in Values(F_i) \Rightarrow V_j \neq V_k \\
\text{type}(F_i) = \text{integer} \\
\forall l \neq i, \text{length}(F_i) < \text{length}(F_i) 
\end{cases}
\]  

(3)

2) **Foreign key rule**

Suppose that we have two records R1 et R2 and RK1 is the record key of R1. To find foreign key in R2, we must consider only fields existing in R2 which its values are equal to RK1 values. The following equation is used to retrieve foreign key from the record R2 according to “(4)”.

\[
\begin{align*}
R_{2_{FK}} &= RK1 = R1_{kk} \\
\exists i \ F_i \in R2 \\
\text{type}(F_i) = \text{type}(RK1) \\
\forall j, V_j \in \text{Values}(F_i) \Rightarrow V_j \in \text{Values}(RK1) \\
\end{align*}
\]

(4)

3) **Equivalent types for the group item rule**

Concerning the group data item we analyze the subfields types by verifying if they are similar in this case the equivalent type of group item will be same as its subfields but its size will be the sum of the whole subfields sizes. Otherwise, if the subfields types are different, in this case the equivalent type of the group item will be always alphanumeric (PIC X) as it is explained in “(5)”:

\[
\begin{align*}
&Let G(S1, S2, ..., Sn) \in R \\
&\forall i, j \neq j \land \text{type}(S_i) = \text{type}(S_j) \Rightarrow \\
&\text{size}(G) = \sum_{i=1}^{n} \text{size}(S_i) \\
&\forall i, j \neq j \land \text{type}(S_i) \neq \text{type}(S_j) \Rightarrow \\
&\text{size}(G) = \text{PICX} \times \sum_{i=1}^{n} \text{size}(S_i) \\
\end{align*}
\]

(5)

4) **Association one-to-many rule**

After determining the foreign key R2FK of R2, we can deduce automatically the multiplicity association in the side of the record R2 is one-to-one. But the multiplicity in the other side related to R1 will be resolved by applying “(6)”.

\[
\begin{align*}
&FK = R2_{FK} \land \text{multiplicity}(R2 \rightarrow R1) = 1 \\
&\forall i, j \neq j \land V_i, V_j \in \text{Val}(FK), V_i = V_j \Rightarrow \\
&\text{multiplicity}(R1 \rightarrow R2) = 1.\star \\
&\forall i, j \neq j \land V_i, V_j \in \text{Val}(FK), V_i = V_j \land 3k \mid V_i, V_j \in \text{Val}(FK) \land V_i = \phi \Rightarrow \\
&\text{multiplicity}(R1 \rightarrow R2) = 0.\star \\
\end{align*}
\]

(6)
The Fig. 4 illustrates the different objects extracted from the running example by means of a model transformation that takes as source model the MMFD and generate the target model DCD that will be refined using the sub-mentioned rule.

IV. EVALUATION AND DISCUSSION

In this section, we will evaluate and discuss the previous approaches. Firstly, according to [9] a reverse engineering process aims to analyze a system for two goals: 1) To identify the system’s components and their interrelationships. 2) To create representations of the system in another form or at higher level of abstraction.

Preceding research on reverse engineering made great achievements concerning the first reverse engineering goal such our first approach based on clustering techniques. But there are very little researches in creating representation of the system in another form especially at higher level of abstraction [17] since the majority of researchers don’t integrate generally the metamodeling or metamodels in their reverse engineering process as higher abstract representation of a system aspect and don’t benefit of the MDE advantages. Therefore, as it was defined by Chikofsky, to attain the first reverse engineering goal, it’s judicious to use metamodels. Otherwise, evolving legacy system without benefit of a higher level of functionalities representation and structure presents risks of quality [1]. Atkinson and Kühne state that the main goal of MDE is to reduce the sensitivity of primary software artifacts to the inevitable changes that affect a software system [18]. The second proposed approach based on MDE falls into this area of research which includes techniques and concepts of MDE and which places the metamodels in the core of the different phases of the reverse engineering. However the first approach depends on many constraints related to the programs structures and their original implementation. The clustering techniques are based on correlation between fields by means of their occurrences in a legacy information system. In [15], the related work presented approaches to identify procedures and global variables in the legacy code, and to group them together in a system aspect and don’t benefit of the MDE advantages. Therefore, the second approach based on clustering is another criterion that may affect the clustering accuracy.

In order to test the accuracy of the two approaches supported by clustering analysis and model driven reverse engineering, a criterion was defined to check how close are the results of those approaches (named correct extracted classes) to the eventual classes produced by the human experts (named expected classes). Indeed, we have chosen a set of arbitrary programs written in COBOL for test and we have used three evaluation metrics: recall, precision and F-measure. In recognition systems and information retrieval, precision is the fraction of retrieved elements which corresponds a given criterion. And recall is the fraction of correct elements that are retrieved. Both precision and recall are based on an understanding and measure of relevance. In other side, the F-measure is a harmonic mean of precision and recall; it can be interpreted as a measure of overlapping between the true and estimated classes. These metrics are usually employed to evaluate knowledge extraction systems following the equations “(7), (8), and (9).” The Table IV shows the recall (Re), precision (Pr) and F-measure (F-m) for both approaches. The main purpose is to compare the results obtained by those approaches and choose the one that presents simultaneously high precision, recall and F-measure.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Expected classes</th>
<th>Correct extracted classes</th>
<th>Incorrect extracted classes</th>
<th>Missing classes</th>
<th>Re (%)</th>
<th>Pr (%)</th>
<th>F-m (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clustering</td>
<td>50</td>
<td>21</td>
<td>16</td>
<td>13</td>
<td>42.0</td>
<td>56.7</td>
<td>48.2</td>
</tr>
<tr>
<td>MDRE</td>
<td>50</td>
<td>34</td>
<td>9</td>
<td>7</td>
<td>68.0</td>
<td>79.0</td>
<td>73.1</td>
</tr>
</tbody>
</table>

Precision = \( \frac{\text{Correct ext classes}}{\text{Correct ext classes + Incorrect ext classes}} \) \hspace{1cm} (7)

Recall = \( \frac{\text{Correct extracted classes}}{\text{Expected classes}} \) \hspace{1cm} (8)

\[
F\text{-measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \hspace{1cm} (9)
\]

As can be seen from Table IV, the calculated recall, precision and F-measure concerning the MDRE are equal respectively to 68%, 79% and 73.1% which are better than the results obtained by the clustering approach.

Nevertheless, according to [19], [20] which affirm that over 50% of all reengineering projects currently fail due to the formalization and automation problems and the absence of standard that support reverse engineering processes. The automation of the model transformations together with the model driven development principles makes it possible to reuse the models involved in reverse engineering processes. The automation problem can also be solved due to the automated model transformation techniques proposed by the MDE. Furthermore, OMG proposes in 2003 the standard MDA to support the overall techniques and paradigms of MDE. Therefore, the second approach MDRE present better results and strength points than the first approach based on clustering.

V. RELATED WORKS

Recovering objects or classes and Object-Oriented models from legacy programs is a vast area of research and many approaches have been proposed to identify objects in code. A number of those works focused on the migrating procedural programs to Object-Oriented software [21], [22] using clustering analysis to identify candidate objects and methods, others ones attempted to find candidate objects using structure charts and data flow diagrams [23]. Reference [24] proposed an approach to identify procedures and global variables in the legacy code, and to group them together based on a set of attributes.

Regarding the MDRE approaches, [25] used the standard Knowledge Discovery Model (KDM) as the metamodel to represent all the different legacy software artefacts involved in a legacy information system. In [17], the related work presents a meta-model that unifies the conceptual view on programs with the classical structure-based reverse engineering meta-models and thereby enables the establishment of an explicit mapping between program
elements and the real-world concepts that they implement.

VI. CONCLUSION

This paper presents the capabilities of reverse engineering techniques and especially objects recovery using two approaches. The first approach is based on clustering analysis and the second one is based on model driven principles. An experiment and a qualitative comparative study between the two approaches are given in order to enable to practitioners and researchers from industry and academia with a vested interest in this area of research to discern the strengths and weaknesses of each approach. Nevertheless, as it is explained over this paper the MDRE is the best approach to support efficiently reverse engineering processes.

This work is a part of other ones that falls into our area of research about reengineering legacy systems that aims recovering knowledge hidden in those old systems by means of automated reverse engineering processes and modernize them through automated forward engineering processes [26]–[29].

REFERENCES


