Costless Expert Systems Development and Re-engineering

Manal Alsharidi, Abdelgaffar Hamed Ali
The Department of Information System, College of Computer Sciences and Information Technology,
King Faisal University, Hofuf, Saudi Arabia

***Abstract*—Symbolic AI is indispensable for the current LLM agents that are used for example to reason the context of the questions. An expert system is a symbolic AI that can explain the reasoning it reached to, which typically is a rule-based system has been attractive for different domains such as medicine, agriculture, and operations. On average, these systems involve hundreds of rules that are instable; moreover, they are coded at low levels of abstraction. Therefore, designing and reengineering an expert system is still costly and needs technical knowledge because of the manual process and maintaining of a low-level abstraction. On the other hand, model-driven architecture (MDA) has proven to be a successful technology that raised the abstraction level and formalized it to automate software development. It specifies business aspects in the platform-independent model (PIM) and implementation aspects in a platform-specific model (PSM). It then automates mapping between them using a standard mapping language called Query-View- Transform QVT. This paper argues that utilizing MDA principles such as the automation and abstractions represented by the descriptor PIM and PSM and mappings metamodels will not only overcome the instability of rules of expert systems, but also provides new insights for its usage. Therefore, this work proposes an MDA-compliant methodology that adopts a UML sequence diagram, a class diagram for the PIM descriptor, and a generic PSM) based on production rules. Moreover, a UML profile to support lacking features in the sequence model has been developed. However, the paper argues for a new kind of process-oriented expert system. Therefore, it not only allows domain experts to develop or participate in expert systems but also reduces the cost of developing new systems and re-engineering or maintenance of the critical and large-scale legacy expert systems.**

Keywords*—*Model-Driven-Architecture(MDA);Unified Modelling Language (UML); Platform-Independent Model (PIM); Platform-Specific Model (PSM); Query- View- Transform (QVT)

I. INTRODUCTION

Expert systems (ESs) are historically the most successful product of artificial intelligence (AI) [1]. It is widely used and designed to solve complicated problems that require reasoning about knowledge using mathematical logic. In fact, “based on Chatbot Agent—Google Bard” makes use of some reasoning based on logic to appear consistent and accurate. In AI, symbolic knowledge is represented as symbols that model concepts and relationships in the form of rules. It turns out that a variety of ESs that have been developed successfully and served stakeholders over a long period of time have become assets for many organizations. For example, an expert system was developed to provide clinical interpretations from thyroid hormone pathology tests decades ago. It had like 700 rules representing the knowledge-based approach, which provided 6,000 interpretations per year [2]. Also currently, the MD Anderson Cancer Center Expert System [3] helps oncologists make more informed treatment decisions. This system has a large knowledge base for oncology; it is expected to have thousands of rules.

However, the commonality among these systems and many others is the rapid change in the knowledge base because of the progress in the landscape of the field; for instance, new treatments, diagnostic techniques, and clinical guidelines that support the domain, as well as a technological change. Moreover, there is an essential business requirement for integrating these systems with others, such as electronic health records (HER), enterprise resource planning (ERP), and others, to increase their capabilities.

Although ES is an old field of study and there are many competing disciplines contributing to decision support problems, such as data mining, deep learning, and large language models (LLM), which are data-oriented approaches [4], ES conserves its unique properties of supporting problems that are rule-based and the capability of explaining reasoning. On the other hand, a hybrid model is typically used [5], whereby a weakness (i.e., learning capability from unstructured data) of one can be improved with the strength of the other (i.e., machine learning). In fact, rules are an intrinsic element of organizations, so decisions are driven by rules that support the production of services or products. Moreover, ES has attracted to some extent new domains, such as environmental data management analysis [6] and policy automation [7].

Given this situation and the fact that adopting code-based approaches such as Pyke, CLIPS, Prolog, Lisp, and other platforms for building ESs remains critical and costly, improvement is essential [8]. For instance, one reason for the interruption of some big systems (i.e., Garvan and IBM) is the high cost of maintenance and reengineering [9]. For example, frequent rule changes, driven by tech or business needs (i.e., adding some quality), require tedious hard coding, raising maintenance and re-engineering costs. On the other hand, there is a need to find new ways to make it easier for domain experts to develop or participate in ES. However, utilizing new engineering methodologies can provide a remedy for these problems.

However, the Object Management Group (OMG) has developed Model-Driven Architecture (MDA) as a methodology for automating software development, which is

standardized by OMG and supported by many tools, such as the Meta-Object Facility (MOF) design language [10], Object Constraint Language (OCL) [11], and XML Metadata Interchange (XMI) [11]. MDA encourages investment in metamodels using MOFs that are platform-independent, thereby paying back the low cost of development. A metamodel, or model of model, is a conceptual model of some design language that could have different implementations. In fact, having for different languages formally an equivalent representation (a phenomenon frequently referred to as syntactic sugar) is common (i.e., for language designers), such as Structured Query Language (SQL), relational algebra, tuple relational calculus, and query by example (QBE), which have the same underlying structure—the same metamodel [12]. In this case, this metamodel is known as abstract syntax, and its representation is called concrete syntax [13]. The strength of decoupling abstract syntax from concrete syntax allows much freedom in having different concrete syntaxes that re-use the same formal abstract syntax. Consequently, concrete syntax changes do not necessarily require abstract syntax to change. This flexibility allows you to re-use the same supporting software for a metamodel. For example, if we have the abstract syntax of SQL (a metamodel) itself and it is modeled in BNF (meta metamodel), it would be possible to send it a tool to render it into OCL, relational algebra, and probably first-order predictive calculus (i.e., specification of some constraints or assertions over a schema is needed). More importantly, MDA realizes the automation of mappings between these different metamodels using a standard mapping language called Query-View-Transform (QVT). Therefore, this separation of concerns, for instance, allows us to adopt a change without much cost incurred because of the high degree of sustainability. We argue that this trend is most suited to the instability of rule phenomena or the frequent changes in business requirements and technology in the context of ES.

A. Expert System

An ES is a computer program that manipulates facts and rules that constitutes a knowledge of some domain to solve complicated reasoning problems efficiently and effectively [14]. These problems require domain experts'intervention to capture the knowledge [15] which is limited by human capability of handling hundreds of factors at the same time.

Many applications in health, industry, or education fields are using ES [16], [17], [18]. In these cases, the utilization of ES is geared toward delivering exemplary performance in addressing intricate challenges within a particular domain. ESs are instrumental in offering explanation and incorporating symbolic reasoning methodologies during problem-solving processes. Consequently, diagnostic ESs continue to hold prominence as the most frequently employed applications in this regard [19]. More important, ES can serve different class of problems such as acting as a *classifier, predictor, and estimator* to serve different sort of application domains that require automation support for decisions.

Moreover, ES has two core components: a knowledge base (KB) and reasoning engine [20]. The propositional ES has a KB formalized using propositions logic; it models the real world in the form of predicates and rules that can be evaluated to check its truthiness with initiations of variables. The KB is like a warehouse that contains knowledge about a specific domain captured by human experts in a form of production rules. Typically, it is a result of an expensive process called in literature a knowledge-Acquisition that involves strong communications between domain experts in one knowledge area and ES developers. It is the critical part in engineering ESs because of the requirement for developers to transfer this rigorous nature of rules of a domain knowledge into symbolic abstraction using logic-based structure. The classical engineering methods in this context follow an informal approach where engineers build informal models to capture the requirements of the system [21]. While the reasoning engine is an interpreter that draws a conclusion from premises that are represented using like first order predicate calculus (FOPC). The well-established theory behind that is the mathematical logic and theorem proving [22]. An example of the theories behind reasoning are Mode's pones [23] that allows to draw a conclusion from premises; It says if P proposition is known and has the fact that P implies Q then we also know Q as a conclusion. Traditionally may software tools that are aczt as inference engine are used to support the execution of ESs: Shells like Drools [24], G2 [25], JADE [26] which is based on the theory of agents and the standard FIPA [27], and Oracle Intelligent Decision Management (OIDM) [28]. These inference engines or Shells are classified based on forward changing (goes from known premises to reach goals by applying rules) or backward changing (works from goal to find the necessary premises) types of reasoning. Because users and domain experts need to understand how these tools reach to some conclusion, Shells have typically adopted a third essential component the explanation facility that could be in different forms: sequence of rules, conditions under which a rule fires and the conclusion it draws, and high-level detailed specification for the reasoning step [29]. In addition, the user interface component is for inserting quires, inputs and converting the rule from the internal representation to be user-understandable form.

B. Model-driven Architecture

OMG has developed and standardized MDA with the aim of developing software without writing code. Models are first-class entities in MDA that enable the re-use of existing software assets, thereby reducing the complexity and cost of development. In MDA, a Platform-Independent Model (PIM) is used to specify the application concepts, while a Platform-Specific model (PSM) is used to specify the implementation issues independent of a technology. Then a standard mapping between PIM and PSM is specified using a Query- View-Transform (QVT), a standard mapping language that performs mappings between metamodels. MDA is an approach for building models, making the transformation of a source model into a target model [11]. To realize MDA, OMG has worked in the set of tools and standards that have been defined and standardized by OMG to support a good infrastructure. These OMG standards include UML, Meta- Object Facility (MOF), XML Metadata Interchange (XMI), Object-constraint language (OCL) and QVT. However, MDA has three types of abstraction: Computational Independent Models (CIM), PIM, and PSM. Each type is an abstraction technique for focusing on a particular part of concerns within a system and can be represented via one or more models.

A PIM is a conceptual model that is platform-independent and focuses on modeling domain concepts. PIM has a higher degree of independence from different platforms (e.g., .NET, CORBA and J2EE). In order to implement a PIM concept there should be a corresponding implementation abstraction involves a concept which can map it, typically the PSM abstraction. PSM is built from some technology's perspective but independent of it because MOF-based language is used. For example a developing database application requires to model the relational model using MOF; the example used in QVT standard document then by automating the transforming of the PIM instances into PSM instances, the main part of writing code is achieved. Thus, the PSM is considered a high-level APIs specification for a well-established platform such as database manager in. However, the main question here is how it would be able to relate a PIM concept with PSM concepts that will be automated. Also, what are the suitable models for representing problem space and solution space?

C. Query / View / Transformation

The OMG has defined a standard for model transformations in the MDA architecture which is QVT. It represents the intrinsic activity in MDA engineering of applications whereby it converts a source model to a target model. It is required that the source model and the target model must both be compliant with the MOF meta-model [30]. QVT defines three specific languages named: (1) Relations, (2) Core (3) Operational/Mapping. Relations and Core are declarative languages with two different levels of abstraction. The QVT Operational / Mapping is an imperative language, it provides common constructions in imperative languages (i.e. loops, conditions.).

Because the relation language allows a round trip mappings between metamodels, this paper uses relational QVT language, specifically MediniQVT [31] engine to test the developed rules of the proposed mappings. Relational QVT has two main clauses: (1) Check only and (2) Enforce.

The "check only" clause focuses on validating source and target models against pre- defined rules and constraints without modifying the target model and act as precondition. On the other hand, the "enforce" clause carries out the actual transformations on the target model based on predefined relationships made between PIM and PSM metamodels. For instance the following example is a part of a complete transformation example that maps UML conceptual model (PIM) for database application into a relational model (PSM) [30] :

Top relation PackageToSchema {

checkonly domain uml p: Package {name=pn}

enforce domain rdbms s: Schema {name=pn}}

The provided example shows a transformation rule or relation named "PackageToSchema" that transforms objects from the "uml", the source domain to "rdbms" target domain. It says that if it is true that an instance of package exists in the source, creates a corresponding instance of type schema in the target. A domain is a typed variable that can be matched with a model of specific type which consists of patterns (i.e. p: package {name=cn}. A pattern can be grasped as a set of variables and constraints that needs to be bound by elements from a model to satisfy it with a valid binding. A domain pattern is a blueprint for the objects and their properties that must be found, changed, or made in a candidate model in order to meet the relationship [30].

II. Related Works

The development of ESs using a model-based approach within the MDA process mapping is still an ongoing area of research, with limited studies focusing on this domain. Chungoorae et al. [14] suggested an approach based on MDA and ontology-driven system development to implement Interoperable Manufacturing Knowledge Systems (IMKS) for product lifecycle applications. It involves developing the PIM level using manufacturing core ontology and transforming it into a PSM in the XKS format. This approach lacks generalization for PSM and is limited to specific platforms. Moreover, it omits the mention of the mapping language used from PIM to PSM. Additionally, it primarily focuses on data-oriented ES type.

The BOM approach, within MDA, automates software generation using PRISMA architectural models, as described by Cabello et al. [15]. The approach utilized conceptual models, PIM, and CIM without specialized languages, resulting in generated program codes for C# .NET. This approach lacks generalization for PSM and is limited to specific platforms. Moreover, it omits the mention of the mapping language used from PIM to PSM. Furthermore, it does not specify the type of ES being utilized.

Yurin et al. [16] proposed using MDA to generate an ES for analyzing construction material damage. The implementation involves conceptual models, rule visual modeling language (RVML) for mapping PIM to PSM, and implemented in the form of a software prototype personal knowledge base designer (PKBD). This approach lacks generalization for PSM and is limited to specific platforms. Moreover, it utilizes an operational language that does not take maintenance into consideration. Additionally, it primarily focuses on data-oriented ES type.

Another research by Maylawati et al. [17] focused on UML diagrams (use case, class, and sequence) to describe ES components, including actor interaction and object relationships. Each use case requires a sequence diagram for normal and alternative processes, while the class diagram represents object interrelationships and adaptable attributes/methods for problem-solving. This approach involves the development of ES using UML diagrams in a general sense, without specifying the principles of MDA.

In study [18], the authors proposed developing decision-making modules for an intelligent system based on MDA principles. They start with the CIM, transforming spreadsheet data into a conceptual model using UML class diagrams. The PIM is then created as a rule-based module, formalizing decision tables with concepts from the CIM. RVML schemas represent these concepts. The PSM is developed depending on the knowledge representation language and converting decision tables into an RVML module. Program codes are generated

using PKBD for the specific platform. This approach lacks generalization for PSM and is limited to specific platforms. Moreover, it utilizes an operational language that does not take maintenance into consideration. Furthermore, it does not specify the type of ES being utilized.

Overall, the existing literature lacks comprehensive compliance with MDA principles. For instance, strong support for re-usability and platform independence such as having PSM that is special-case and PIM that is cluttered with PSM aspects. Also, the lack of maintenance considerations because of using operational mapping languages as well as in overall there is no support for it. Our proposed approach aims to generate a generalized PSM that can be adapted to different platforms as well as support maintenance. This is enabled by utilizing a relational QVT and developing reverse mapping rules. Moreover, this work recognizes a new type of process-oriented ES and enables both types of ES: process-oriented and data-oriented.

III. Research Methodology

This paper aims to automate the creation of rule-based ES through the utilization of MDA. MDA, as explained, intends to automate software development without writing code by raising the abstraction level, so the process is driven by high-level models and mappings. The proposed approach argues for process-oriented ES as well as data-oriented ES. However, MDA principles compliance is a target for this work as well as considering the maintenance support. Therefore, these reforms the classic process of developing ESs that use a code-based approach into a new methodology for the development that uses models as first-class entities. The following subsections deal with the following questions: What are the suitable models to represent the PIM and PSM metamodels? Is there any gaps (i.e. concepts) exist in the source or target design language? How are PIM concepts related to PSM concepts? How to abstract Shells (platforms) in a generic PSM? And finally how low-cost maintenance is supported?

The answer to these questions can be organized around main principles of the proposed approach: (1) Modeling Expert Business Rules in PIM metamodel, (2) Building a UML Profile, (3) Developing PSM of Production Rules (4) Building the mapping between PIM and PSM.

A. Model Expert Business Rules in PIM Metamodel

The expert system usually stemmed from business rules that represent the decision instrument which requires automation support. These rules classically captured manually in a form of conditional statements rendered as FOPL that probably augmented with the syntax of platform. They are two abstraction levels act as one level which makes it difficult for domain experts as well as developers to deal with these technical aspects. In contrast, the PIM is an alternative abstraction decuples the technical aspects of ES from application concepts therefore pertain to a conceptual model for problems under consideration typically developed independent of a platform so hides implementation details. Due to its abstract nature, it can capture essential features and requirements, which can help experts from different domains communicate by enabling shared understanding. MDA approach proposes MOF-based language such as any UML models to act as PIM metamodel [10]. The PIM can be metalevel 1 or metalevel 2. The concrete instances of a UML model that represent specific case of ES (i.e. some diseases diagnose system), while UML is at metalevel 2 because MOF can model it which is a metalevel 3 [32]. This flexibility allows different modeling representation capabilities that address the diversity of system. The business rules for ES in code-based approach are dependent on a clear understanding of the ES's requirements. This understanding may be achieved through close collaboration with domain experts, stakeholders, and end-users. But usually there is no corresponding explicit formal model such PIM metamodel that can capture these requirements.

More importantly this work argues for process-oriented that centers on the workflow or operational procedures of a given expert system, such as production processes, scheduling operations, and generally processes. In this type of systems, the rules are injected within a process, not like the classic ES that focuses on data so called data-oriented, process is dominating element. For example, in an academic system; a student can only register for 12 credit hours if his GPA is less than 2 and, while in manufacturing system; IF machine temperature reaches critical threshold, THEN stop the machine and call maintenance procedure. The former rule is a constraint augmented with an action – register, while the later expresses object properties constraints, the temperature of the machine. The distinction between them is useful for acquisition of knowledge process as well as it has impact in the design under the context of this work. Although, the model we have introduced is capable of accommodating either of these types, its emphasis is primarily on the latter (i.e., process-oriented). However, business rules, are set of guidelines or policies that dictate how an organization operates, are frequently embedded within processes to ensure and enforce consistency and integrity [33]. It is therefore a good candidate for PIM.

However, this paper proposes specifying business rules for ES in PIM, using both class and sequence diagrams. In which class diagram serves as a descriptor of facts or data with their schemas that is needed by the ES, while the sequence diagram is used for capturing the behavior of the system that usually is augmented with rules. Although there are alternatives to this design decision such as activity diagram, sequence diagram is less elaborative so more compact and easier to learn and communicate the problem of ES. Nevertheless, the relationship between UML sequence diagram and UML class diagram is that the sequence diagram involves objects and messages that are eventually comes from classes that are supposed to be fully specified by the UML class model. Therefore, the UML class diagram supplies the sequence diagram with schemes of data and their relationships.

1) PIM sequence diagram metamodel: A metamodel is a model of the model that is required here to capture the elements needed to support process-oriented ES instances; that is a UML PIM metamodel. We need to investigate the big enough and suitable UML sequence diagram metamodel to be ready for representing the instances of PIM that will be developed by domain experts and developers as well as guides

the mapping process later. A developer in this case typically develops models for ES at metalevel 1 where it models objects of specific ES such as a manufacturer production expert system or car faults diagnoses system. The intention of a sequence diagram in UML is to communicate the specific behavior by sketching the sequence of messages communicated between objects in a system so it's a dynamic view. Fig. 1 presents the necessary and big enough metamodel elements needed to model any business rules for experts.

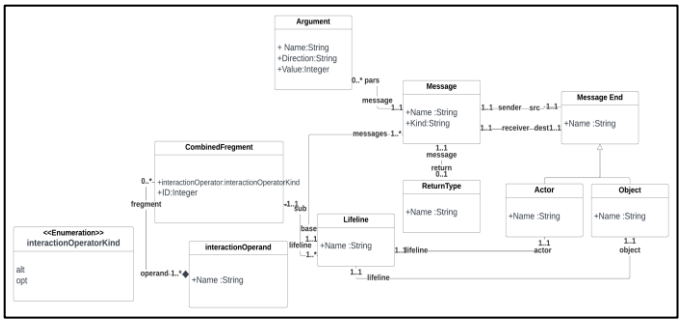

Fig. 1. PIM sequence diagram metamodel [34].

Fig. 1 is the abstract syntax of the sequence diagram that is part of the UML metamodel which fits the problem addressed by this work. The metamodel says a Message (with name and kind properties) can have zero or more Arguments (with name, direction, value properties). It can have a Return Type that specifies the type of value that is returned. In addition, the Message has the Message End indicates two ends of a message exchanged between two Lifelines (usually named element) where the first represents the source (base) and the second is the destination. A Message End is either an Object or an Actor. A lifeline can have more than one message. It indicates that a message has been successfully transmitted from the sender to the receiver, and that the receiver has finished processing the message.

The CombinedFragment is an essential component in diagrams that allows for the specification of complex control structures such as loops, parallelism, and conditions. Its behavior is determined by the chosen InteractionOperator, which dictates how the fragment behaves. For example, when using "alt" as the InteractionOperator, the CombinedFragment represents a choice of behavior where only one option is executed. Semantically an Operand selection within the "alt" fragment is based on guard expressions, which determine the conditions for executing each option. The use of the "else" guard expression represents a negation of all other guards within the CombinedFragment. If none of the options have guards that evaluate to true, none of them are executed, and the remaining part of the diagram continues. Note that an enumeration class called InterationOperKind is used to supply the kinds required for InteractionOperator. It is a useful feature can be utilized in expert systems in exceptions as well as normal conditional state.

2) PIM class diagram metamodel: The UML class model utilizes class diagram notation [35] to describe the data and their relationships using class, properties, inheritance (generalization-specialization) and association concepts. For example, in the journal system a Reviewer and Paper are classes (objects of metalevel 1), and a Review class has the association with Paper between them that represents the relationship a reviewer provides a review for a certain paper which also has an association with a Review class that represents the feedback. In the following, Fig. 2 illustrates the UML class diagram metamodel that is part from UML standard specifications developed by OMG to act as a descriptor for data and facts needed in ES with their integrity constraints.

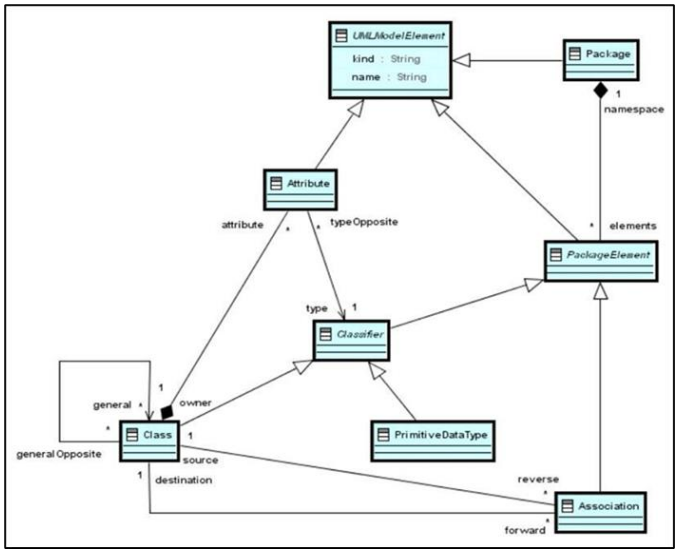

Fig. 2. PIM class diagram metamodel [30].

Fig. 2 shows part of the abstract syntax of the UML class diagram [20]. In which UMLModelElements are classes, interfaces, packages, and relationships. For example, a class diagram could include several UML model elements such as Classes, Attributes, and Associations, to model classes of objects with properties and the relationships between different objects in a system. A metaclass class in Fig. 2 is a central concept of the language and it is a kind of a classifier. A classifier in UML means a class that can be instantiated or has instances different from Abstract classes that do not have. Classes are defined by a set of attributes and methods that objects of that class will have. Because usually a class like Reviewer has attributes: ID, name, and Area of interest there is of metaclass Attribute to model this. Attributes are characteristics or properties of a class, and they define the data that objects of that class will have. In addition, Association, which describes the relationships between two or more classes. Associations define how objects of one class are related to objects of another class. Moreover, a Package is a grouping mechanism used to organize related elements, including classes, interfaces, and other packages. A PackageElement is an abstract class that is a kind of package.Therefore, a diagram usually exists in a package. Another critical UML element is the Classifier, which describes a set of objects that share common characteristics and behavior. Lastly, UML includes PrimitiveDataTypes, which are basic data types built into the modeling language or programming language used to represent

data. Examples of UML primitive data types are Boolean, integer, and string.

3) UML profile for PIM metamodel: A profile is a powerful lightweight extension mechanism that adds concept, syntax, and semantics to the metamodel [13]. It does not require substantial change to the metamodel so less costed approach because UML editors does not change because of extensions. A profile consists of stereotypes, tags, and metaclass classes of the elements that need to be extended, which are classifiers. The stereotype represents a new concept or syntax that is needed for extension while a tag adds some properties if needed to the stereotype. However, it is necessary to have an end of an extension that represents some metaclass class; a solid line notation designates this extension usually drawn between the two ends in UML standard. Profiling plays a pivotal role in filling the gaps in the metamodel. Because in this work there is a gap in the Sequence Diagram; it is not defined by UML Sequence metamodel which are the modeling elements: Not, OR, and Head that exist in the target (Shells). Therefore, there is to design a profile in order to allow developer/domain expert to use these concepts. Fig. 3 shows the UML profile diagram for the proposed approach.

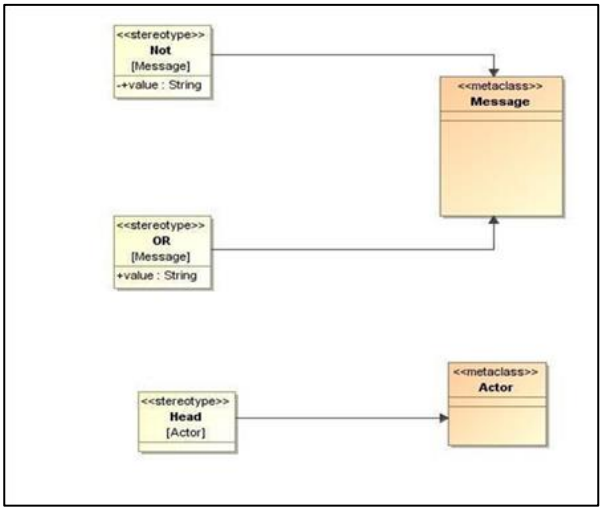

Fig. 3. PIM profile diagram metamodel.

In Fig. 3 a profile is designed for the PIM sequence diagram with three distinct stereotypes needed. These stereotypes serve to provide concepts that the PIM sequence model is lacking. It basically extends two metaclass classes: Message and Actor because semantically OR and NOT are related to messages in the level of abstraction and later to predicate as we will see in the mapping. The first stereotype of the 'Actor' element is named a 'head,' concept to allow designers of ES to utilize the concept of specifying the focal point which gets the benefit of the service provided by sequence diagram. A tool cannot easily determine without cost, so the profiling mechanism is a less costed solution. The second stereotype pertains to messages and introduces an 'OR' operator concept. This 'OR' is an indicative of multiple possible interactions that can exist between messages, a common practice in ESs (multiple rules with same head). The third stereotype extends the Message to introduce a 'NOT' operator concept. In this context, the 'NOT' signifies conditions or interactions that are explicitly negated or excluded (it is also common in ESs). Therefore, the utilization of these stereotypes within the profile diagram serves the purpose of specifying rules that involve disjunction, negation and as well discriminating the Head of the rule. We are ready now to look at how we design a generic PSM.

B. Develop PSM of Production Rules

A production rule traditionally used in different fields such complier, natural processing languages and logic which specifies how input stimuli are transformed into output responses (produce a symbol output from a symbol input). It consists of a set of rules, each consisting of a condition and an action or LHS and RHS. A condition specifies a set of constraints that must be satisfied by the input, whereas an action specifies a set of operations to be performed on the input [36]. To fire it means to replace the LHS with the RHS. To model production rules using PSM, it is necessary to identify the specific elements of FOPC [37] because it is what Shells of ES are based. Fig. 4 illustrates the PSM metamodel for the proposed ES.

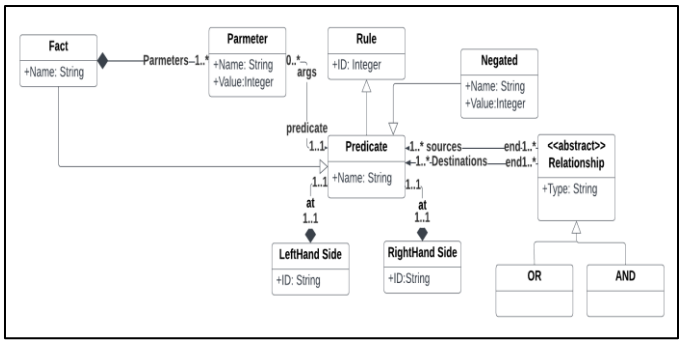

Fig. 4. Rule-based expert system PSM metamodel.

In this context, a typical production rule might be expressed in natural language, as in the example provided: "If the car won't start and there is no clicking sound when the key is turned, then the problem is likely a dead battery". To get the metamodel of FOPC as in Fig. 4, since after investigation instances of rules exist in this world, we need a metaclass called Rule in the PSM metamodel. A Rule can be identified by ID, so we need an attribute called ID of type integer. Since a Rule has ascendant that act as a set of conditions must be met for a rule to be applied or executed, we call it the right-hand side and the consequent or action will call it left-hand side. Therefore, it is necessary to define a metaclass classes called RightHandSide(RHS) and LeftHandSide (LHS) because there are many instances will be of this kinds for rule. Moreover, we observe that each of them consists of a basic building block known as predicates; so, we need to model a metaclass class called Predicate because there are many instances of it in a single rule. In addition, usually a Predicate involves parameters that are variables that should be bound during execution; we need a model it as a metaclass class called Parameter, which provides parameter's name and type. Now we can turn into the relationships between the PIM and PSM.

C. Build the Mappings between PIM and PSM

The aim of MDA eventually is to map the source (PIM metamodel) into a target (PSM metamodel) which acts as a part of writing the code in the development process. This model transformations should be done through the standard language QVT which is independent of both ends. In the context of rule-based ES, the expected output of this mapping process is a PSM instances that can be used to represent executable facts and rules for the target platform. It is typically a UML model instances or objects of metalevel 1 [32]. The PSM should be able to implement the PIM objects so translate them into specific constructs or patterns that are suitable for the target platform. The following subsections discuss the mapping rules of the proposed approach: (1) PIM Sequence Metamodel to PSM Metamodel, (2) PIM Class Metamodel to PSM Metamodel.

1) PIM sequence metamodel to PSM metamodel: Sequence diagram is utilized initially to facilitate the decision-making process by depicting the business process aspects of the ES in a high-level model that allows domain experts to communicate the problem easily. However, the goal is to convert business rules that act as the knowledge of expert in some domain into implementation using PSM concepts. As consequence of this, we need to find relationships between sequence diagram concepts and the PSM concepts which are production rules that are commonly represented using FOPC [37]. Table I shows these relationships of both metamodel concepts.

TABLE I. PIM SEQUENCE DIAGRAM TO PSM RULE-BASED ES MAPPING RULES

Rule	Transformation Rule	Source Model	Target Model
R1	MessageToLHS	Message: kind= ‘goal’	LeftHandSide
R2	MessageToPredicate	Message: kind= ‘normal’	RightHandSide
R3	AltToHead	Combinedfregment: interactionOperator='alt' and Message:kind= ‘goal’	LeftHandSide
R4	AltBodyToRHS	Combinedfregment: interactionOperator='alt' and Message:kind= ‘normal’	RightHandSide
R5	ArgumentToParameter	Argument	Parameter
R6	MessageEndToRelationship	MessageEnd	Relationship
R7	Negation	Message (has NOT)	Negated predicate

These transformation rules play a crucial role in converting from PIM sequence model to PSM of rule – based ES. For instance, the "MessageToLHS" rule maps a message with the 'goal' kind into the LeftHandSide format, while the "MessageToPredicate" maps a message with the 'normal' kind into the RightHandSide. The "AltToHead" rule transforms a Combinedfregment with interactionOperator='alt' and message with kind= ‘goal’ into the LeftHandSide. The "AltBodyToRHS" rule transforms a Combinedfregment with interactionOperator='alt' and message with kind= ‘normal’ into the RightHandSide. Similarly, the "ArgumentToParameter" rule converts an argument into a parameter, and the "MessageEndToRelationship" rule transforms a message end into a relationship. Lastly, the "Negation" rule is employed when a message includes 'NOT' to create a negated predicate in the target model. These rules are acting as separate artefacts so preserve the separation of concerns principle.

To automate the mapping process, mapping rules must be specified using QVT standard transformation language. The QVT mapping rules has the following structure:

transformation map (source: sequence, target: psm)

This transformation specification is like a procedure map or make a transformation from a source model represents as a sequence diagram (source) to a target model represents a PSM (target), it is telling the tool that the mapping direction.

The following formalizes the informal mapping rules specified in Table I for the specific mapping between PIM and PSM. Rules will be numbered (ascending order) for easier reference (All these rules are tested using (MediniQVT tool).

R1:

top relation MessageToLHS {

k, cn: String;

checkonly domain source m: sequance::message{kind = 'goal', name =k};

enforce domain target p:psm::predicate{at= a:psm::LHS{}, name = k};

where {ArguementToparmeter(m,p);

In R1, the source domain, involves a pattern that checks for messages in the sequence diagram with a specific *kind* = "goal" and a *name* that will bound using variable "k” based on instances of the source. If it’s true, then in the target domain, the rule enforces the creation of a *predicate* in the PSM; with a repository, with the name as "k" and an attribute "at" assigned to a parameter "a" of type "psm::LHS". Additionally, the rule includes a constraint that the relation "ArgumentToParameter" must be called after executing this rule which ensures transformation from argument to parameter. Because mapping to arguments is postcondition (executes after the first part above) and a complement, *where clause is added.*

R2:

relation MessageToPredicate {

pn: String;

checkonly domain source m: sequance::message{kind='normal', name=pn};

enforce domain target p : psm::predicate {at = a :psm::RHS{}, name=pn};

where { ArguementToparmeter(m,p);}}

The rule R2 directs a transformation from the sequence domain to the psm domain. It checks if there is a *message* with a specific *kind* (normal) and *name* = pn (bound to the current instances in the source) in the sequence diagram. if true, then in the target domain, it creates a *predicate* with a matching

name(pn value) and assigns to the attribute "*at*" a parameter "a" of type "psm::RHS". The rule also includes a constraint, postcondition that the rule R5 -"ArgumentToParameter" for parameter transformation must be executed afterwards.

R3:

```
top relation AltToHead{
cn,n:String; o: Integer;
checkonly domain source f:sequance::combinedfregment{
interactionOperator='alt',ID=o,lifelinee=k:sequance::LifeLine{name=n,
messages=m:sequance::message{name=cn,kind='goal'}}};
enforce domain target lt:psm::RHS{ID=o};
enforce domain target ltt:psm::LHS{ID=n};
enforce domain target p:psm::predicate{name=cn,at=a:psm::LHS{iD=n}};}
```

R3 is a rule that checks if the sequence model has a fragment with an interaction operator set to "alt" and an *ID* matching "o" (bound with current instances).Also, it verifies a *lifeline* with the *name* "n" and a *message* with the *name* "cn" and kind "goal" within that fragment. If true, then in the PSM model, the rule enforces creation for the right-hand side (RHS) with an *ID* matching "o", a left-hand side (LHS) with an *ID* matching "n", and a predicate with the *name* "cn" and *an* attribute referring to the left-hand side with ID "n." This rule does the initiation task where the rest of the rule will base.

R4:

```
top relation AltBodyToRHS{
i:Integer;cn,n:String;
checkonly domain source f:sequance::combinedfregment{
{iD=i,interactionOperator='alt',lifelinee=k:sequance::LifLine{name=n,
mesages=m:sequance::message{name=cn,kind='normal'}};
enforce domain target p:psm::predicate{name=cn,att=a:psm::RHS{iD=i}};
where { ArguementToparmeter (m,p);}}
```

R4 is a complement rule to R3 rule, asserts a *combined fragment* in the source sequence model with an interaction operator set to "alt" and an ID matching the given value of "i". It also asserts a lifeline with the *name* "n" and a *message* with the *name* "cn" and the *kind* "normal". If true, in the target domain, creates a *predicate* with the name "cn" and the attribute "att" assigned the right-hand side (RHS) object with the ID value for "i".Also, R5 is postcondition so needs to be executed afterwards as where clause exists.

R5:

```
relation ArguementToparmeter {
Cn ,n,q: String;
checkonly domain source m: sequence::message{kind = q,name = Cn,
pars = w:sequance::argument{name = n}};
enforce domain target p:psm::predicate{name = Cn,
args = k:psm::parmeter{name = n}};}
```

R5 relation asserts for a message in a sequence diagram with a specific *kind* and *name*, and pars attribute with *argument* that has *name= Cn, if turr in* a target domain, will be the creation of a *predicate* with the same name and parameter.

R6:

```
top relation MessageEndToRelationship{
checkonly domain source b:sequance::messageend{name = cn,
sender = e:sequance::message{ kind = 'normal' }};
enforce domain target w :psm::relationship{name ='AND',
srcP = ps:psm::predicate{}};
where {MessageToPredicate(e,ps);}}
```

R6 asserts if a *message end* in the source sequence diagram with a specific *name* and a *sender* that is a "normal" message. Accordingly, in the target domain, it enforces the creation of a relationship with the *name* 'AND' and a *source* predicate. This rule builds the relationship between predicates that usually is 'And' if not specified 'OR'. R2 is required as postcondition.

R7:

```
top relation negatedTopredicate {
Cn ,n: String;
checkonly domain source a:sequance::message{kind= 'negated',name = Cn};
enforce domain target o:psm::predicate{name=Cn +'not'};}
```

R7 rule checks for a message in the source sequence diagram with a specific kind "negated" and a name matching the variable "Cn". If true,in the target domain, it enforces the creation of a *predicate* with a name formed by appending "not" to the original name.

2) PIM Class Metamodel to PSM Metamodel: The class diagram is utilized initially to act as a descriptor for data and facts needed in ES. However, the goal is to convert the facts of experts in some domains into implementation using PSM concepts. As a consequence of this, we need to find relationships between class diagram concepts and the PSM concepts. Table I shows these relationships after a close investigation of both metamodel concepts. Table II shows transformation rules representing the mapping between PIM class diagram to PSM rule- based ES.

In Table II, the "ClassToFact" rule performs the transformation of classes into a fact in the target model. Similarly, the "AttributeToParameter" rule is employed to convert an attribute into a parameter. These rules play a vital role in the process of adapting and reshaping data within the

modeling contextualizing the need for facts that represent the essential part of knowledge.

TABLE II. PIM CLASS DIAGRAM TO PSM RULE -BASED EXPERT SYSTEM MAPPING RULES

Rule	Transformation Rule	Source Model	Target Model
R1	ClassToFact	Class	Fact
R2	AttributeToParameter	Attribute	Parameter

The QVT mapping rules starting with this statement:

transformation map (source: class, target:psm)

This transformation specifies the mapping between a source model represented as a class diagram (source) and a target model represented as a PSM (target).

The formal QVT mapping rules corresponds to Table II:

R1:

top relation ClasstoFact{

Cn , n : String;

checkonly domain source a:cla::classs{name = Cn};

enforce domain target o:psm::Fact{name = Cn};

where {AttributeToParameter(a,o);}}

R1 relation transforms a source model (class) to a target model(psm). It checks for a *class* in the source domain with a specific name "Cn". In the target domain, it enforces the creation of a *Fact* with the same name "Cn". Additionally, it includes a constraint R5 executes afterwards that ensures the mapping of attributes from the source class to *parameters* in the target for *Fact.*

R2:

relation AttributeToParameter{

Cn , n , v: String;

checkonly domain source a:cla::classs{name = Cn , attribute = ar:cla::Attribute{

name = n, value= v}};

enforce domain target o:psm::Fact{name = Cn, parmeters = w : psm::parmeter {

name = n, value= v}};

R2 relation is part of the "ClasstoFact" transformation. It checks for an attribute within the source class that matches the variable "n" and has a value matching the variable "v". In the target domain, it enforces the creation of a parameter within the Fact with the same name and value. This relation ensures the mapping of attributes to parameters during the transformation process.

D. Round- Trip Mapping

The extant issues encountered in the development of ES via a code-based approach has strong resolution through the contemporary application of MDA. This approach enables developers to focus on the high-level concepts of system design, reducing the complexity of development and facilitating the reuse of code and knowledge [12]. For instance, MDA offers cost-effective maintenance through the utilization of automation and the implementation of a round-trip mapping mechanism. This work argues for support of maintenance using relational QVT language (QVT-r is supported by EMF). It reflects the changes that happened to PSM such as having a new version of the software of shells. More important the changes in the PIM model (i.e., business rules change) will not change the PSM or mapping assets so can be re-used. This adds great value to the re-engineering effort required for ES. By structuring mappings in this manner, developers gain enhanced and ease mechanism to make change such as update to the rules. This approach optimizes the maintainability of the system, as it streamlines the process of rule manipulation and adaptation within the MDA framework. More importantly, there are legacy ESs serving organizations for a long time that can benefit from this model whereas in extreme cases models can be reverse engineered so can be changed and synchronized automatically with required changes. For instance, a rule can be developed for round- trip mapping for PSM – PIM Sequence Diagram where each represents different sort of changes:

RM1:

relation LHSToMessage{

k : String;

checkonly domain p :psm::predicate{ at = a:psm::LHS{}, name = k };

enforce domain target m: sequance::message{ kind = 'goal', name = k};

where {parmeterTOArguement (p,m);}}

The purpose of this rule is to enable the redirection of mapping from the left-hand side (LHS) back to the message, facilitating any necessary changes.

RM2:

relation PredicateToMessage{

pn: String;

checkonly domain target p: psm::predicate {at = a :psm::RHS{}, name = pn};

enforce domain target m: sequence:message {kind='normal',name = pn };

where { parmeterTOArguement (p,m);;}}

The purpose of this rule is to enable the redirection of mapping from the predicate back to the message. These rules establish a mechanism for reverse mapping that is not only applicable to the specific rules but also to other rules derived from the PIM Sequence Diagram to the PSM.

The round- trip mapping for PSM- PIM Class Diagram:

RM3:

top relation FactToClass{

Cn, n : String;

checkonly domain source o: psm::Fact{name = Cn};

enforce domain target a: cla::classs{name = Cn}

where {ParameterToParameter(o,a);}}

The purpose of this rule is to enable the redirection of mapping from fact back to the class.

RM4:

relation ParameterToAttribute{

Cn , n , v: String;

checkonly domain source o: psm::Fact{name = Cn, parmeters = w: psm::parmeter {name = n, value= v}};

enforce domain target a: cla::classs{name = Cn , attribute = ar:cla::Attribute{name = n, value= v}};}}

The purpose of this rule is to enable the redirection of mapping from parameter back to the attribute. Also, these rules establish a mechanism for reverse mapping that is not only applicable to the specific rules but also to other rules derived from the PIM Class Diagram to the PSM.

IV. Reengineering and New Insights

This section analyzes and discusses the feasibility, insights, and opportunities of using MDA at different scales of change in the scope of the ES. The reengineering of a system typically involves a radical change for the entire system to achieve some result, while maintenance tackles parts of a system to improve it by making corrective action or minor modifications [39]. In this work, we use the term reengineering in a broad context. For instance, maintenance could be applicable to any part of PIM, PSM, or mapping rules, while the process of making radical changes (from a code-based approach to a model-based approach) to the legacy ESs by using MDA is a re-engineering process. However, the capability of interoperability is also one of the main concerns of MDA, which is defined as the ability to seamlessly integrate different systems or components to exchange information and work together [11].

There are many reasons why current ESs need to change under the umbrella of maintenance or reengineering, for example, the need to interoperate or integrate with other systems. The basic assumption of the data or facts underlying ESs is to be provided in a static way for reasoning. Nowadays, this is not the case where data should be updated by dynamic systems such as in the medical field by EHR (i.e., supply patient data) or general business ERP (i.e., provide production information such as a master or detailed schedule). The data involved in such systems is not only current but also comprehensive. For instance, patients with new symptoms or a production machine show new odd behavior in one manufacture. Based on the application requirements, data needs to be pulled or pushed from these systems to the relative ES. It is obvious that manual pulling or pushing is not practical in this sense. This visibility is a sort of strong business requirement that must be achieved today. On the other hand, rules as shown are subject to change due to different reasons, such as progress or a shift in the landscape of the knowledge of a field (i.e., medicine), but we argue that our approach enables automated supplementation of rules because the transformation process in MDA is a separate and dynamic process with stable mapping rules. This will not only provide a dynamic way of running the ES but also provide new insights. One can imagine that GPT agents (like ChatGPT or Bard) or similar intelligent systems can utilize this feature. Indeed, these GPT-based AI tools use symbolic knowledge-based questions or queries to find the relevant information or identify the context of the question (using inferencing rules and knowledge). This view suggests that an integration mechanism allows data to be outsourced as well as rules, so ES can be provided as a dynamic service.

In addition, interfaces of an old legacy systems became obsolete and so there is tendency to be upgraded to new standards such as Jetpack Compose [40] developed by google for building native Android applications, SwiftUI [41] for IOs and mac applications, CSS frameworks than enable web-access for ESs, and many others taking into account in single system such as mobile you find different of GUI standards.

We now turn to the question of how MDA can support this strong demand for integration with these different systems. On the one hand, MDA has the abstraction of a PSM that is based on MOF to represent the technical aspects of a platform, one of which is GUI platforms or others such as APIs for specific platforms (i.e., EHR and ERP). So specifically, PSM for any of this need to be developed and a couple of transformations using like QVT [38]. More than a decade ago, typically the integration between systems followed standards such as service-based system technologies, web services, and WSDL [42], JOSON and RESTFul [43] or SOAP [44]. They contributed to interoperability between different tools and encourage more integration to be practiced. Cloud systems are basically complex, diverse systems that use these standards. More importantly, the literature is rich with some of these standards that exist as PSMs and can be re-used to reduce development costs. On the other hand, PIM is a business-level abstraction built independently of even PSM, so the portion of PIM related to interoperability, such as GUI or others discussed above, can be projected using the transformation capability of MDA. In this case, mapping rules only need to be changed if the PSM is already published (GUI, WSDL). However, there might be intermediate steps (pre-processing steps) needed, such as using the QVT view maintenance [30] capability to map PIM into more refined PIM or PSM into more refined PSM.

To conclude, MDA has rich architecture support for change, such as interoperability, that can allow even non-MDA systems to integrate in a manner that reduces the re-engineering cost. More importantly, this interoperability in this context provides new insights into using traditional expert systems, such as exposing expert systems as a service, and gains the power to of dealing with dynamic changes in facts or the instability of rules.

V. Evaluation using Case Study

An academic advising ES has been introduced to bridge the gap between students and advisors by shifting advising, complaining, evaluating, and suggestions from traditional ways to a more contemporary one [45]. The decisions need to be made by students during their academic journey such as course

enrolment, course withdrawal, postponing study, etc. In this paper, we take on a scenario of a rule-based ES for academic advising in the university system. The need for ES for academic advising is to take a decision for different actions involves uncertainty and a couple of factors need to be tested. For example, the decision to drop a course for low GPA students has different consequences which is not straightforward decision. Similar thing can be said for postponed study, drop a semester and so on.

The ES will build the work plan by identifying the student through some important points:

- Perquisite courses.
- Knowing the student's performance.
- The student’s weakness points.
- Domain skills.
- Skills that a student needs to improve.
- Student goals.
- Track the student pathway.

The ES works to facilitate the communication between students and the advisors by raising the student's performance giving some recommendations that help the low GPA student to develop specific skills for different semester actions such as course enrolment, course withdrawal, postpone study, etc.

A. Developing Process Model for ES

A student who is struggling with a low GPA might approach their academic advisor for assistance in considering the option of dropping a course for the current semester. The ES is responsible for determining when the low GPA student is eligible for dropping a course according to the following conditions:

- **Perquisites Course:** Perquisites course must be ‘Not Major’ category.
- **Skill Assessment:** Students skill must be a ‘Weak Skill’ in this course.

The advisory academic rules are:

- **Rule:DropCourse(SID,CID)=**IfGet_PreCourse(CList) AN Check_PreCouese_Category(CID)AND Check_Skill(Skill)
- In all other cases, the system does not allow the student to drop the course.

These rules consider the relevance of the PreCourse category and the strength of the student's skill set. If the conditions are met, the system facilitates the selection of appropriate PreCourse categories and skills, while disallowing the student from dropping the selected course. Conversely, if the conditions are not met, the system permits the student to drop the course if desired. Fig. 5 illustrates the sequence diagram outlining the process for dropping a course for a student with a low GPA.

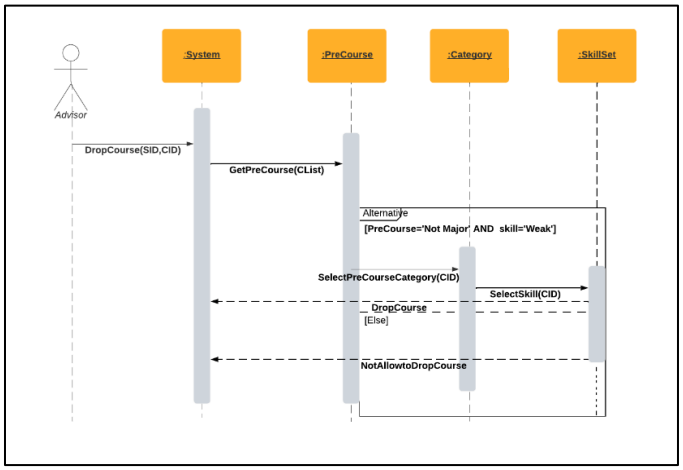

Fig. 5. Drop a course UML sequence diagram.

Fig. 5 presents the process involving the Advisor, System, and PreCourse, Category, and SkillSet objects, illustrating how t odecide on the drop of a course. The interaction commences as the advisor engages with the system, focusing on the rules associated with dropping a course. The first message, "If the PreCourse category is classified as 'Not Major' or the student possesses a 'Week' skill in a particular subject (skill='Weak'), then the system proceeds with the selection of the PreCourse category (CID) through the predicate SelectPreCourseCategory(CID) and the selection of the skill (CID) through the SelectSkill(CID).Following this, the system responds with the message "DropCourse" for dropping the course, provided the conditions specified in the previous message are met. However, in all other cases, the system responds with the message "NotAllowtoDropCourse".

As noted earlier, MDA develops rule-based ES by mapping transformation from PIM to PSM. The mapping process is performed using the MediniQVT tool, where the source file is the Ecore file [46] representing the Sequence diagram PIM metamodel, and the target file is the Ecore file representing the Production rule PSM metamodel. Also, an XMI file acts as an input containing instances of the sequence diagram metamodel for this case for example is utilized. Subsequently, the relational QVT mapping rules mentioned above are applied to create the production rules of ES. Table III shows the mapping rules used and Fig. 6 shows a sample of execution for final result of mappings.

B. Developing Data Model for ES

As mentioned, the UML class diagram is used as a descriptor to represent the data and facts of ES. Fig. 7 explains the UML class model by the using example of a case study of academic advising system in university.

As shown in the UML class diagram is that Student has a relationship with Course. In addition, the Course has a specific domain (such as Math, programming) consisting of skills needed as outcomes for the course(s) in this domain. Further, a course sometimes has Prerequisite course; the association between Course and Prerequisite course, which models this business rule. This will enable checking the integrity constraint that that Student must take the Prerequisite course and pass it

before registering in a new course. According to the types of courses, there are two types: 1- Taken Course, 2- Next Plan Course. Taken course refers to the taken courses in the semester, and next plan courses refers to the planned courses in the next semester. Nevertheless, Student must have a study plan to follow according to the program requirements. The academic advisor wishes to help students complete this study plan successfully with low risk by making the right decision at the right time which is the source of the calling the experience of the ES.

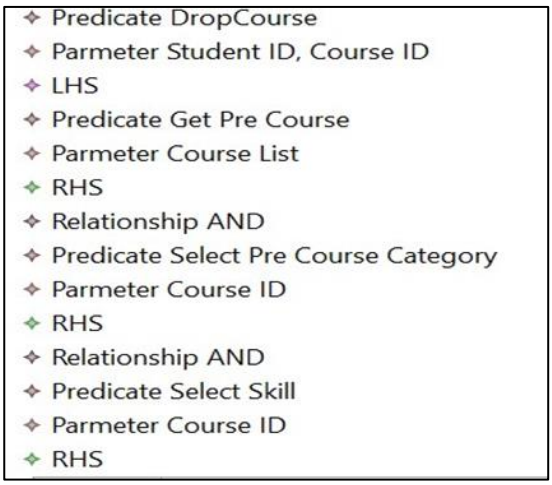

Fig. 6. PIM sequence diagram for drop a course target file.

Fig. 7. PIM Sequence Diagram to PSM Rule -based Expert System Mapping Results

No.	Predicate	Parameter	Predicate type	Justification
1	Drop Course	Student ID, Course ID	Left Hand Side	This predicate represents the LHS predicate of the drop a course rule.
2	Get Pre-Course	Course List	Right Hand Side	This predicate represents the RHS predicate of the drop a course rule. It has an AND relationship with next RHS predicate
3	Select Pre-Course Category	Course ID	Right Hand Side	This predicate represents the RHS predicate of the drop a course rule. It has an AND relationship with next RHS predicate
4	Select Course	Course ID	Right Hand Side	This predicate represents the RHS predicate of the drop a course rule.

As noted earlier, MDA develops rule-based ES by mapping transformation from PIM to PSM. Where the source file is the Ecore file representing the (Class diagram metamodel), and the target file is the Ecore file representing the (PSM metamodel). Additionally, an XMI file containing instances of the class diagram metamodel is utilized. Subsequently, the relational QVT mapping rules mentioned above are applied to create the facts of ES. Table IV shows the results of the mapping process, there are eight facts generated, each of which is associated with a specific parameter. These results provide a comprehensive description of the facts utilized by the domain experts in leveraging the ES effectively and Fig. 8 shows a sample of execution for final result of mappings.

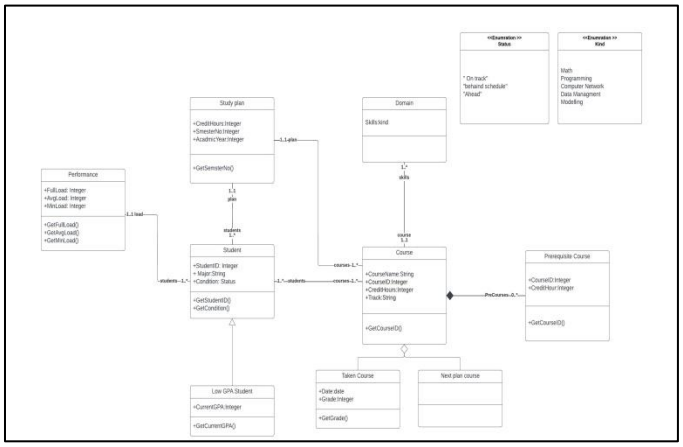

Fig. 8. Academic advising system UML class diagram.

TABLE III. PIM CLASS DIAGRAM TO PSM RULE TO BASED EXPERT SYSTEM MAPPING RESULTS

No.	Fact	Parameter
1	Performance	Full Load
2	Student	Student ID
3	Study Plan	Credit Hours
4	Course	Course Name
5	Perquisite Courses	Skills
6	Low GPA Student	Current GPA
7	Taken Course	Date
8	Next Plan Course	NA

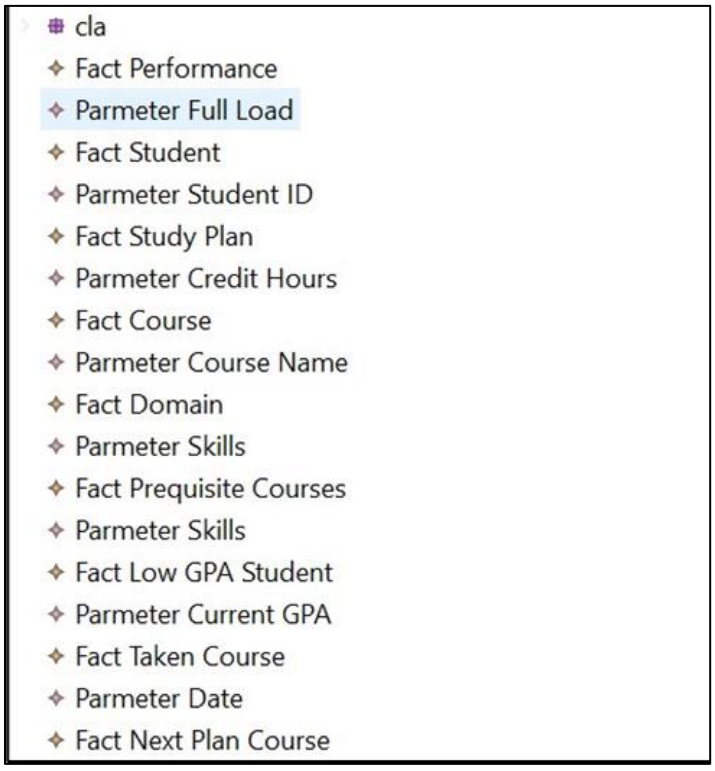

Fig. 9. PIM class diagram for advising low GPA students in university target file.

VI. Results and Discussion

The investment on the quality of software development became evident that will pay back the cost. Having the case that many legacy ESs contributing to different domains exist over a long period, such as in medicine ,health, and education [16],[17],[18], necessarily entails requirement changes such as in platform or business rules. More importantly, the discovery of the entrance of ESs into new domains (environmental management and cybersecurity) requires flexibility and less costly development methods. However, using MDA in this work provides these qualities. The PSM metamodel from production rules developed as a generic PSM and mapping rules can act as assets so they can be re-used with the development process of any kind of ES; therefore, the principles of reusability, platform independence, are achieved and hence reducing the cost. For instance, changing Prolog with the Pyke platform for any reason such as utilizing a forward chaining tool instead of backward changing tool, does not cause a change in the PIM, PSM, and properly minor change to mapping rules. Also, changing to a new version of a platform such as upgrading to acquire new features, the proposed approach does not require to change PIM or PSM and mappings.

Similarly, in more extreme maintenance cases where rules are updated or modified, only the PIM (model instances) needs an update, the rest will be re-used therefore coping with the rules instability. Moreover, the raising of abstraction afforded by MDA, such as in the PIM descriptor, allows domain experts to participate or write expert system, which bridges the gap between domain experts and developers.

On the other hand, the integration of an ES with other systems (i.e. HER) under reengineering process or maintenance, is an inexpensive approach because of the re-using utility provided by metamodeling and formalizations (using MOF) of the descriptors: PIM, PSM, and mapping rules. Thereby provides costless reengineering. Because different Shells have different features, the PSM developed is standard one and therefore comply with the principle of platform-independence so like portability can be achieved. It can be modified to incorporate additional features if is to put into practice, but it should be the commonality among all shell platforms. The XMI standard allows either PIM or PSM to be migrated to another tool so can be edited or manipulated.

The sequence diagram, in reality, reflects the nature of interactions involving the business rules of a desired ES. However, this study argues for a type of ES that is process-oriented, where a set of actions with a sequence that represents constraints such as pre-conditions and post-conditions need to be specified for the desired outcome. For example, the process of checking ripe and unripe fruits, the process of optimization such as in manufacturing (i.e., efficiency of steal production), control process, real-time recommendations process, and planning and scheduling processes.

VII. Conclusions

This work is about automating ESs from high-level models using the principles of MDA. ES is a long-sounding successful product of AI but lacks advanced methods of development and re-engineering, which leads to an increase in the cost of maintenance and development. Moreover, effective communication between developers and domain experts is a crucial yet challenging aspect of designing ESs. The inherent differences in technical knowledge and domain expertise often lead to communication gaps, hindering the accurate translation of domain knowledge into functional system components. However, MDA raises the abstraction level of the development of an application as well as provides a structured approach for automation so ES applications can leverage this feature. MDA decouples application concepts or domain of the problem that needs to be specified in the PIM metamodel from the technical aspects of implementation, which will be specified in the PSM metamodel; then mapping the first end (PIM) into the second (PSM) using the standard mapping language, QVT.

The proposed approach addresses some limitations in the literature, such as the lack of generic PSM and specific compliance to the MDA principles, as well as recognizes and supports a class of expert systems identified as process-oriented ES. A UML sequence diagram is used to model business aspects of this type of ES, and a class diagram is used to model facts by representing entities and their attributes. It is, therefore, establishing high-level specifications of business rules and processes. The generic PSM is developed based on pure production rules (FOPC), which makes it adaptable to different rule-based engines or Shells that implement PIM models of business aspects. Furthermore, we designed a UML profile diagram that extends the PIM sequence diagram, to support the lack of some features in the UML sequence model (OR and Not). Finally, in this tackle, we developed the necessary mapping rules (QVT) that act as a standard for the transformation of PIM sequence diagram metamodel into a rule-based PSM metamodel, generating the necessary rules and generating ES facts from UML class models as well as the developing round-trip mapping that supports the maintenance of ES.

To evaluate our proposed approach, which is design science research, a real case study of an academic advising system for low GPA students, was used for evaluation. QVT mapping rules that facilitate the transformation from the PIM to the PSM have been developed. In this process, we establish mapping rules that convert the PIM sequence diagram into a rule-based ES, generating the necessary ES rules. Additionally, we defined mapping rules that transform the PIM class diagram into a PSM rule-based ES, resulting in the creation of the required facts for ES. More importantly, utilizing the QVT Relational language that enables round-trip mapping thereby support potential changes (i.e. in rules, business requirement, platform) of PIM, PSM, mapping rules itself. A less costly maintenance therefore achieved because of the automation and the standardizing of round-trip mapping rules being developed. The results obtained from this case study provide practical evidence of the applicability and utility of our proposed approach in real-world scenarios.

Nevertheless, it is important to acknowledge the limitations of the current work. The connection between PSM and a platform is not tackled but since a generic PSM is developed the process is straightforward. Also, the consequences of the inclusion of OCL in UML models. In addition, although the

introduced model is adaptable to both process-oriented and data-oriented approaches, its primary focus lies in the process-oriented aspect. Also, the models lack the capability of using a relational or mathematical expression that can be needed in the PIM metamodel.

In future endeavors, our objective is to further advance the ES design approach by implementing and evaluating the proposed design on different domains of ES, ensuring its practical applicability and effectiveness, and supporting the lacking features in PIM. Also, incorporating the UML profile in the mapping process and resolving the limitation of tools (mapping engine) to recognize profiles.

REFERENCES

[1] H. Tan, "A brief history and technical review of the expert system research," IOP Conf. Ser. Mater. Sci. Eng., vol. 242, no. 1, 2017, doi: 10.1088/1757-899X/242/1/012111.

[2] R. Colomb, Deductive Databases and Their Applications. 1998.

[3] "The Oncology Expert Advisor," 2013. https://www.mdanderson.org/publications/annual-report/annual-report-2013/the-oncology-expert-advisor.html (accessed Dec. 08, 2023).

[4] B. T. Sayed, "Application of Expert Systems or Decision-Making Systems in the Field of Education," Inf. Technol. Ind., vol. 9, no. 1, pp. 1396–1405, 2021, doi: 10.17762/itii.v9i1.283.

[5] F. Lareyre, C. Adam, M. Carrier, and J. Raffort, "Automated segmentation of the human abdominal vascular system using a hybrid approach combining expert system and supervised deep learning," J. Clin. Med., vol. 10, no. 15, 2021, doi: 10.3390/jcm10153347.

[6] K. Fedra and L. Winkelbauer, "A hybrid expert system, GIS, and simulation modeling for environmental and technological risk management," Comput. Civ. Infrastruct. Eng., vol. 17, no. 2, pp. 131–146, 2002, doi: 10.1111/1467-8667.00261.

[7] A. M. Elsawi, S. Sahibuddin, and R. Ibrahim, "Model driven architecture a review of current literature," J. Theor. Appl. Inf. Technol., vol. 79, no. 1, pp. 122–127, 2015.

[8] S. Y. Choi and S. H. Kim, "Knowledge acquisition and representation for high-performance building design: A review for defining requirements for developing a design expert system," Sustain., vol. 13, no. 9, 2021, doi: 10.3390/su13094640.

[9] Y. Ran, X. Zhou, P. Lin, Y. Wen, and R. Deng, "A Survey of Predictive Maintenance: Systems, Purposes and Approaches," vol. XX, no. Xx, pp. 1–36, 2019, [Online]. Available: http://arxiv.org/abs/1912.07383.

[10] "OMG," 2014. https://www.omg.org/mda/ (accessed Feb. 02, 2022).

[11] "MDA," 2001. https://www.omg.org/mda/ (accessed Feb. 02, 2022).

[12] "ODM," Model Driven Eng. Ontol. Dev., no. September, pp. 215–233, 2009, doi: 10.1007/978-3-642-00282-3_8.

[13] "MOF," no. August, 2019, [Online]. Available: https://www.omg.org/spec/MOF/2.5.1/PDF.

[14] B. G. Buchanan and R. Q. Smith, "Fundamentals of expert system," Springer Ser. Mater. Sci., vol. 206, pp. 31–39, 1988, doi: 10.1007/978-3-662-44497-9_3.

[15] I. H. Sarker, A. I. Khan, Y. B. Abushark, and F. Alsolami, "Mobile expert system: Exploring context-aware machine learning rules for personalized decision-making in mobile applications," Symmetry (Basel)., vol. 13, no. 10, pp. 1–10, 2021, doi: 10.3390/sym13101975.

[16] N. Mayadevi, S. S. Vinodchandra, and S. Ushakumari, "A review on expert system applications in power plants," Int. J. Electr. Comput. Eng., vol. 4, no. 1, pp. 116–126, 2014, doi: 10.11591/ijece.v4i1.5025.

[17] S. S. A. Naser and M. H. Al-bayed, "Detecting Health Problems Related to Addiction of Video Game Playing Using an Expert System," J. Multidiscip. Res. Dev., vol. 2, no. 9, pp. 7–12, 2016.

[18] S. Khanna, A. Kaushik, and M. Barnela, "Expert Systems Advances in Education," Ncci, no. March, pp. 19–20, 2010, [Online]. Available: https://www.researchgate.net/profile/Akhil-Kaushik/publication/267862155.

[19] W. P. Wagner, "Trends in expert system development: A longitudinal content analysis of over thirty years of expert system case studies," Expert Syst. Appl., vol. 76, pp. 85–96, 2017, doi: 10.1016/j.eswa.2017.01.028.

[20] K. P. Tripathi, "A Review on Knowledge-based Expert System : Concept and Architecture," Artif. Intell. Tech. - Nov. Approaches Pract. Appl., vol. 4, no. 4, pp. 19–23, 2011.

[21] A. A. Mohammed, K. Ambak, A. M. Mosa, and D. Syamsunur, "Expert system in engineering transportation: A review," J. Eng. Sci. Technol., vol. 14, no. 1, pp. 229–252, 2019.

[22] C.-L. Chang and R. C. Tung lee, Symbolic Logic and Mechanical Theorem Proving. Academic Press, 1973.

[23] R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg, "DENDRAL: A case study of the first expert system for scientific hypothesis formation," Artif. Intell., vol. 61, no. 2, pp. 209–261, 1993, doi: 10.1016/0004-3702(93)90068-M.

[24] "Drools." https://docs.drools.org/8.44.0.Final/drools-docs/drools/introduction/index.html (accessed Dec. 08, 2023).

[25] gensym, "G2." http://dev.gensym.com/platforms/g2-standard/# (accessed Dec. 10, 2023).

[26] F. Bellifemine, A. Poggi, and G. Rimassa, "JADE a FIPA2000 compliant agent development environment," Proc. Int. Conf. Auton. Agents, pp. 216–217, 2001.

[27] P. Charlton, R. Cattoni, A. Potrich, and E. Mamdani, "Evaluating the FIPA standards and their role in achieving cooperation in multi-agent systems," 2002, doi: 10.1109/HICSS.2000.926996.

[28] "OIDM," 2023. https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Overview/Overview.htm (accessed Dec. 12, 2023).

[29] J. Giarratano and G. Riley, Expert Systems: Principles and Programming, Fourth Edition. Course Technology, 2004.

[30] "QVT," Transformation, no. January, pp. 1–230, 2008, [Online]. Available: http://www.omg.org/spec/QVT/1.0/PDF/.

[31] V. Nikulsins, "Transformations of software process models to adopt model-driven architecture," Proc. 2nd Int. Work. Model. Archit. Model. Theory-Driven Dev. MDA MTDD 2010, Conjunction with ENASE 2010, pp. 70–79, 2010, doi: 10.5220/0003044500700079.

[32] David S. Frankel, Model Driven Architecture： Applying MDA to Enterprise Computing, vol. 308. 2003.

[33] R. S. Aguilar-Savén, "Business process modelling: Review and framework," Int. J. Prod. Econ., vol. 90, no. 2, pp. 129–149, 2004, doi: 10.1016/S0925-5273(03)00102-6.

[34] "UML," Proc. - 2005 IEEE Symp. Vis. Lang. Human-Centric Comput., vol. 2005, no. December, p. 9, 2005, doi: 10.1109/VLHCC.2005.65.

[35] "UML," pp. 443–506, 2017, [Online]. Available: https://www.omg.org/spec/UML/2.5.1/PDF.

[36] K. Jetlund, E. Onstein, and L. Huang, "Adapted rules for UML modelling of geospatial information for model-driven implementation as OWL ontologies," ISPRS Int. J. Geo-Information, vol. 8, no. 9, 2019, doi: 10.3390/ijgi8090365.

[37] "PRR," OMG Specif., vol. 1.0, no. December, p. 74, 2009, [Online]. Available: http://www.omg.org/spec/PRR/1.0/.

[38] I. Essebaa and S. Chantit, "Toward an automatic approach to get PIM level from CIM level using QVT rules," SITA 2016 - 11th Int. Conf. Intell. Syst. Theor. Appl., no. PMarch, 2016, doi: 10.1109/SITA.2016.7772271.

[39] R. S. Pressman and B. Maxim, Software Engineering: A Practitioner's Approach. McGraw Hill, 2010.

[40] "Jetpack Compose basics," 2023. https://developer.android.com/codelabs/jetpack-compose-basics#0 (accessed Aug. 05, 2023).

[41] "SwiftUI," 2023. https://developer.apple.com/documentation/swiftui/ (accessed Jul. 09, 2023).

[42] "WSDL," 2007. https://www.w3.org/TR/wsdl/ (accessed Jul. 05, 2023).

[43] "RESTful," 2018. https://wiki.onap.org/display/DW/RESTful+API+Design+Specification (accessed Dec. 12, 2023).

[44] "SOAP," 2007. https://www.w3.org/TR/2007/REC-soap12-part0-20070427/ (accessed Dec. 12, 2023).

[45] R. M. Tawafak, G. Alfarsi, A. Romli, J. Jabbar, S. I. Malik, and A. Alsideiri, "A Review Paper on Student-Graduate Advisory Expert system," 2020 Int. Conf. Comput. Inf. Technol. ICCIT 2020, pp. 187–191, 2020, doi: 10.1109/ICCIT-144147971.2020.9213794.

[46] "EMF Tutorial - EclipseSource." https://eclipsesource.com/blogs/tutorials/emf-tutorial/ (accessed Oct. 18, 2022).