International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Transformation from CIM to PIM for Querying Multi-Paradigm Databases

Simmi Bagga[1], Dr. Anil Sharma[2]

Submitted: 22/10/2022 **Revised:** 21/12/2022 **Accepted**: 20/01/2023

Abstract*:* In model-driven engineering, the modeling process is prioritized above any other part of the software development cycle. This paradigm shifts the focus squarely onto the model for those working in design and analysis. Several standards, such as OMG's MDA method, have been provided in this respect. Model-Driven Architecture (MDA) is a software development paradigm that was standardized in 2003 by the Object Management Group (OMG). It is based on MDE concepts (Model-Driven Engineering). Through the power of abstraction, the MDA approach simplifies complex systems and the world around you. This core overview is expanded upon by the Object Management Group's conceptual framework, as well as by supplementary standards such as the Unified Modeling Language (UML), the Meta Object Facility (MOF), and the eXtensible Metadata Interchange (XMI) (XML Metadata Interchange). Once the foundational requirements of the system are laid down, transformation rules are applied to convert them into Platform Independent and further platform specific models.

Keywords: *MDA, CIM, PIM, PSM, UML*

1. Introduction

Within the realm of software engineering, a variety of different approaches to software development have been investigated and found to be successful. The procedural, object-oriented, component-oriented, and service-oriented paradigms are some examples of these methodologies. All of these methods set out to guarantee high-quality computer systems while also making their implementation on execution platforms easier. These systems for carrying out orders are constantly developing, diversifying, and increasing in complexity. One recommended strategy for decoupling business logic from platform characteristics is to extend the effectiveness of models beyond their traditional role as a technique of representation and documentation. Model-Driven Engineering, a relatively new engineering approach, depends on it heavily (MDE). Therefore, model-driven engineering is a software development process that places an emphasis on the models themselves, therefore preparing the conceptual foundation for code generation via the regular modification of the models. To that aim, we have been putting the MDA technique to work in order to create an application architecture with many layers: presentation, business, and data access.

In model-driven engineering, the modeling process is prioritized above any other part of the software development cycle. This paradigm shifts the focus squarely onto the model for those working in design and analysis. Several standards, such as OMG's MDA method, have been provided in this respect. Model-Driven Architecture (MDA) is a software development paradigm that was standardized in 2003 by the Object Management Group (OMG). Through the power of abstraction, the MDA approach simplifies complex systems and the world around you. This core overview is expanded upon by the Object Management Group's conceptual framework, as well as by supplementary standards such as the Unified Modeling Language (UML), the Meta Object Facility (MOF), and the eXtensible Metadata Interchange (XMI) (XML Metadata Interchange). Therefore, model-driven engineering is a software development process that places an emphasis on the models themselves, therefore preparing the conceptual foundation for code generation via the regular modification to the models. To that aim, we have been putting the MDA technique to work in order to create an application architecture with many layers: presentation, business, and data access.

2. Model Driven Architecture

An offshoot of MDE, "Model Driven Architecture" (MDA) was first offered by the Object Management Group in 2001. Portability, interoperability, and reusability are the three primary aims of MDA, all of which are realised via the practise of concern separation. The MDA is built on the idea of decoupling the design of a system from the specifics of how that system will be implemented on a certain platform. Platform-blind specification occurs when a system is specified without taking into account the platform on which it will run. After designing this Platform-blind specification, next steps include:

- Establish platforms.
- Determine the platform for your system.
- Transform a current system definition into a new specification that is optimised for the platform that has been selected. Modelling and model transformation approaches are mostly used. Conversions can be horizontal (PIM to PIM) or vertical (CIM to PIM) (PIM to PSM or PSM to PIM).

[1]*Research Scholar, Lovely Professional University, Phagwara, (Punjab), India*
[2]*Professor, Lovely Professional University, (School of Computer Application) Phagwara, (Punjab), India*
[1]*simmibagga12@gmail.com,* [2]*anil.19656@lpu.co.in*

2.1 CIM (Computational Independent Model)

Computational-Independent Model (CIM) is a framework for expressing requirements regardless of the underlying architecture. A CIM is a simple demonstration of a system that can be understood by ordinary people without detailed information on how it is implemented. It just shows the business logic of the system represented as models. CIM should be developed in such a way that it can further use in the development of PIM and PSM.CIM clears the functional and non-functional requirements of the system.

2.2 Platform Independent Model (PIM)

A Platform Independent Model (PIM) is a high-level abstraction model in the context of Model-Driven Architecture (MDA) that disregards implementation-technical issues. The transformation of PIM, PSM is developed by implementing it to specific platform. The advantages of this approach are to drive one or more PSM from single PIM. The main benefit of this approach stems from the possibility to derive different alternative PSMs from the same PIM depending on the target platform, and to partially automate the model transformation process and the realization of the distributed application on specific target platforms

2.3 Platform Specific Model (PSM)

A good example of a model that defines the technical method for resolving an issue is the Platform Specific Model (PSM). It's the vehicle via which news and analysis about PIM tools may be shared with the world.

3. Related Work

Since the standardization of MDA in 2003, it is widely being used in development of systems that range in a wide variety of tech domain. Below are some of the related researches from recent years.

Lachgar [4] suggests using MDA technique on mobile platforms to create a model that adheres to the n-tier design. In this work, the authors provide the meta-models required for constructing the display, business, and data access layers. It has been determined which model-to-model translation rules apply to which models.

These transformation rules were translated using the ATL model transformation language, which the authors chose. However, the ATL language transformation mechanism was not described in any detail.

Bezivin et al.[1] recommend using the MDA technique to web service-based systems. The authors were successful in generating both a meta-model for a web service platform and a meta-model using the UML language. Citations of the many rules were done using the language of ATL.

Using the QVT (Query / View / Transformation) model transformation language, Srai et al.[2] detail a process for creating a PSM model that conforms to the multi-tier architecture of e-learning platforms. This method was developed to set up a PSM model that is consistent with the hierarchical structure of modern e-learning platforms.

According to Srai et al. [3], the work to develop a PSM model for EJB systems has only recently reached this degree. Castro et al. [6] demonstrated a service transformation from CIM to PIM. The authors show the CIM level by modelling business processes with BPMN and identifying services from the beginning with a value model [5].

The authors extend the use case model and the activity diagram to reach the PIM level, which is achieved with the help of ATL. Using this approach, you can pinpoint the CIM services and enterprise processes that will steer your transition to PIM automation. The authors' PIM-level analysis is confined to use case diagrams and activity diagrams, without offering the structural viewpoint (class diagram) that would reveal this level's ultimate function.

To represent business perspective in CIM, Rodrguez et al. [7] describe a CIM-to-PIM transformation approach based on security criteria. BPMN is used by the authors to represent business processes in CIM, while QVT [12] is used to convert CIM into PIM class diagrams and use cases. This technique opens up exciting opportunities for transforming CIM into PIM-based security. This method makes advantage of safe information systems.

Model-driven engineering is used by Hahn et al. [9]. In ATL, the authors describe CIM using BPMN and PIM using SoaML models. This technique models services using SoaML, the most recent OMG standard, however it is not the end aim of PIM level, which employs structural diagrams (as class diagram). The representation of CIM and PIM as features and components is described by Zhang et al. [10]. This model's responsibilities link its features and components in a way that facilitates the transition of CIM and PIM. DSL is used for traceability by Grammel and Kastenholz [11]. Both methods provide conversion solutions for CIM to PIM but do not define CIM and PIM models. Gutierrez et al. [12] offer a method for transforming use case diagrams into activity diagrams that is based on MDA by making use of QVT transformations. Although the authors define use case functional requirements in CIM, their technique changes CIM to PIM utilising explicit principles.

Mazon et al. [13] develop a UML profile based on the modelling framework to determine CIM level. The authors built PIM on the conceptual modelling of a data warehouse developed by QVT. Only in data warehousing does this method transform CIM to PIM. As described by Kherraf et al. [8], a CIM-to-PIM methodology is a method

that "transitions" between the two models. The authors start with a business process model and use case diagram to model the business processes, and then end with a thorough activity diagram defining the system requirements.

As a result of this process, the requirements are mapped onto the various parts of the system. The diagram represents the first PIM parts. Finally, business archetypes transfer class diagram nodes to system components. Useful suggestions for making the change from CIM to PIM are provided by this technique. This approach uses a use case diagram at the CIM level, despite the fact that it depicts system functionality. Many CIM to PIM transition methods [14], [15], [16], [17] rely on BPMN to generate CIM levels, hence we are limited when translating CIM to PIM[18].

Post studying the usage of MDA is systems designing throughout the years; it was applied on a system that queries Multiparadigm databases. Multiparadigm databases are databases that support multiple data model paradigms, such as document, key-value, graph, and object-relational. These databases can handle a diverse range of data types and structures and provide flexibility in terms of querying and data manipulation.

4. CIM for Querying Multiparadigm Databases

CIM was made by gathering all the requirements that come under the spec of Multiparadigm database. As per the OMG standards, Activity Diagram, Sequence Diagram and Use Case Diagrams were constructed as part of it. These all diagram clears the functional and non-functional requirements of the system. CIM

should be developed in such a way that can be further used for developing the PIM and PSM. It simply demonstrates the requirements of the system without any information about the implementation. Various Diagram used to represent the CIM are:

- Use Case Diagram
- Activity Diagram
- Sequence Diagram

In this study main focus is on querying Multi-paradigm databases. In this modern scenario, no single database can fulfil the requirements of diverse variety of data so there is the need of time to use multi-paradigm databases. Below is the CIM for querying multiparadigm database.

4.1 Use Case Diagram

Use case diagram is the pictorial representation of requirements and entity with their association specifications. Use case diagrams indicate how a system's events and flows are represented. The use case diagram does not specify implementation details. A very much organized use case likewise depicts the pre-condition, post condition, and exceptions. The objective of use case diagram is to capture the dynamic aspect of a system. The following are some of the goals of use case diagrams:

- Used to gather the system's requirements
- Used to get an external perspective on a system
- Identify the system's internal and external influences.
- Demonstrate the relationship between the requirements and the actors.

2.4 Sequence Diagram

Sequence diagram is also known as interaction diagram that depicts the interaction behavior of the system. This helps to understand the overall functionality of the system. It explains the various objects involved in the system and the sequence of interaction among the objects needed to complete the functionality of the system. The sequence diagram is expressed as parallel vertical lines, different objects that exist simultaneously, and in the form of horizontal arrows, the messages exchanged between object according to order of occurrence.

2.5 Activity Diagram

The activity diagram expresses the flow from one activity to another i.e. the dynamic behavior of the system. Start and end of the user journey are shown by black circles in between which all possible processes are defined. We also have some conditional branches that make the system more dynamic

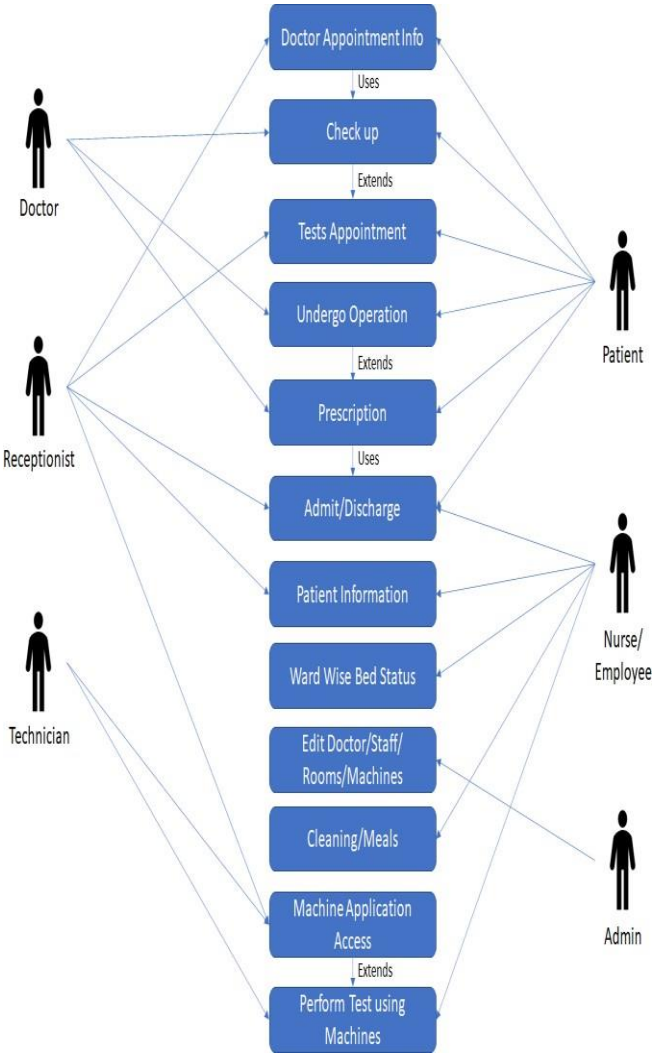

Fig 1: Use Case Diagram of Hospital System

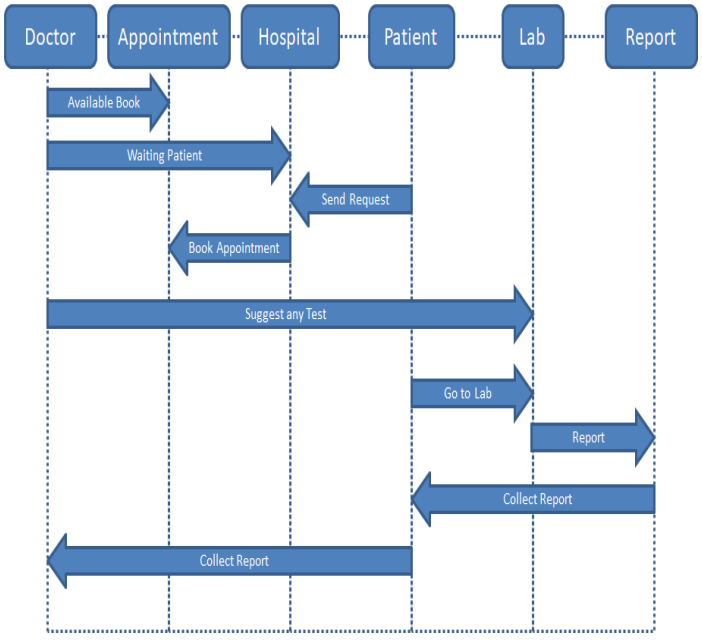

Fig 2: Sequence Diagram of Hospital System

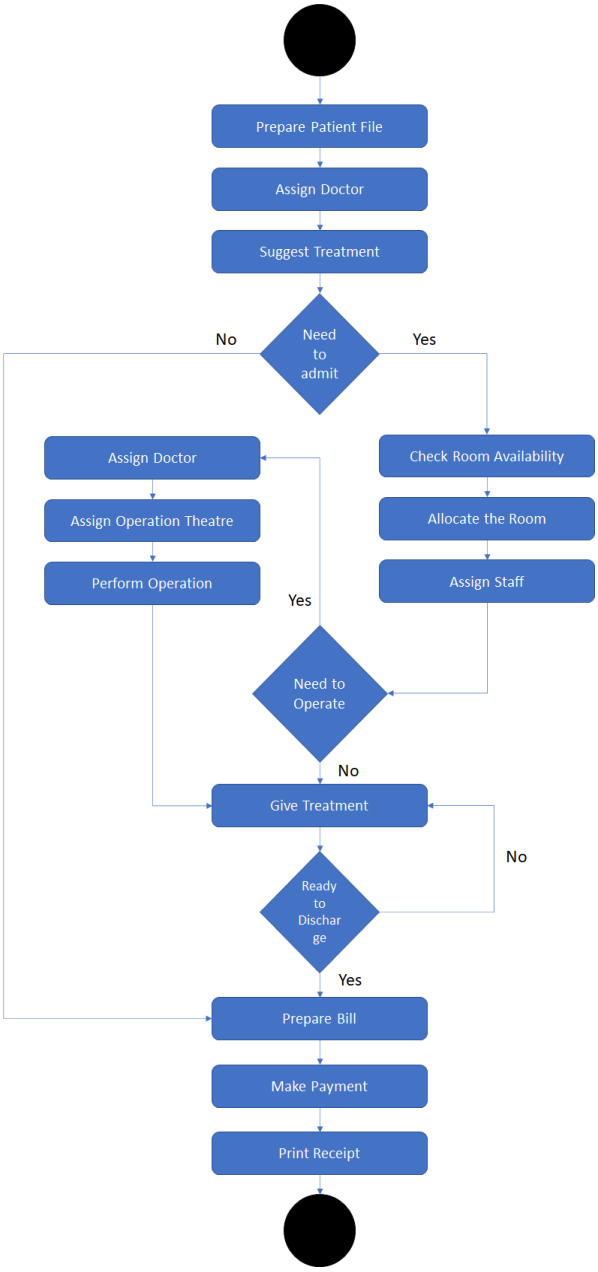

Fig 3: Activity Diagram of Hospital System

5. Transformation from CIM to PIM

Model Driven Architecture (MDA) consists of a collection of models, where every successive model is developed on the basis of its proceeding model using Model Transformation. Model Transformation is a process of converting a source model into a target model of the same system. Transformation may be done manually, semi-automatically or automatically. There are two varieties of model transformations: horizontal transformations and Vertical transformation. When the source and target models become at the same level of abstraction or at the same semantic level are called horizontal transformation and in vertical transformations, it is used when the considered models are recorded into two levels of abstraction or on two different semantic levels.

MDA provide ease to transform application to new platform because this approach separates the requirement logic, business logic and implementation details of an application. This separation is represented by number of models of MDA. The requirement

analysis is describes in the first level of MDA i.e. CIM(Computational Independent Model), the business logic is expressed by the PIM(Platform Independent Model) and PSM(Platform Specific Model describes the implementation detail in some particular platform. This separation of implementation detail from the logic helps to reuse PIM to create one or more PSM. This will reduce the overall cost of developing the application.

6. PIM for Querying Multi-Paradigm Databases

Combining the requirements gathered in the form of CIM with the transformation methodologies, the Platform Independent Model for Querying Multiparadigm Databases was created. It was created in the form of UML Activity Diagram where the four major units were End User, Front End System, Back End System and Databases. The system works when these units interact with each other.

As soon as the system is turned on (which can be through hitting a URL in browser or user logging in), Back End imports a library for connecting to the Integrated Databases and makes the connection. It is important to store each connection objects and reuse them throughout the Query Journey to avoid any extra latencies. The connection strings and database info can be hard coded or provided through a configuration file. After connecting to the Databases, the system needs to pull out the names of Tables/Collections and display them to the End User. This should be the first screen made visible to end user, From this step onwards, the end user drives the journey by providing the database name and table that needs to be queried. The selected data is then fetched and displayed to user. Along with it the columns/fields are also shown so that user can apply the required filters and project only the necessary fields. These filters and projections are first validated from the data previously obtained and passed to database only on successful validation.

This sums up the basic flow of the application. Another important feature when we talk about databases is the ability to get the data from multiple tables at once i.e. via Joins. In our application, after selecting the table to be queried, a list of eligible tables that can form a join are also displayed. The user can select and achieve the required join result and can further apply filters/projections to the joined result.

Features:

1. The system is scalable with respect to number/type of database in use.
2. Each database can provide features that it specializes in. The system just adds an extra layer on top of querying database and thus leaves the root functionalities unaffected. Thus, all the intrinsic features of Database will be inherited by the system.
3. The system is completely Platform Independent as it does not restrict developer to the use of any specific Programming Language / Database.

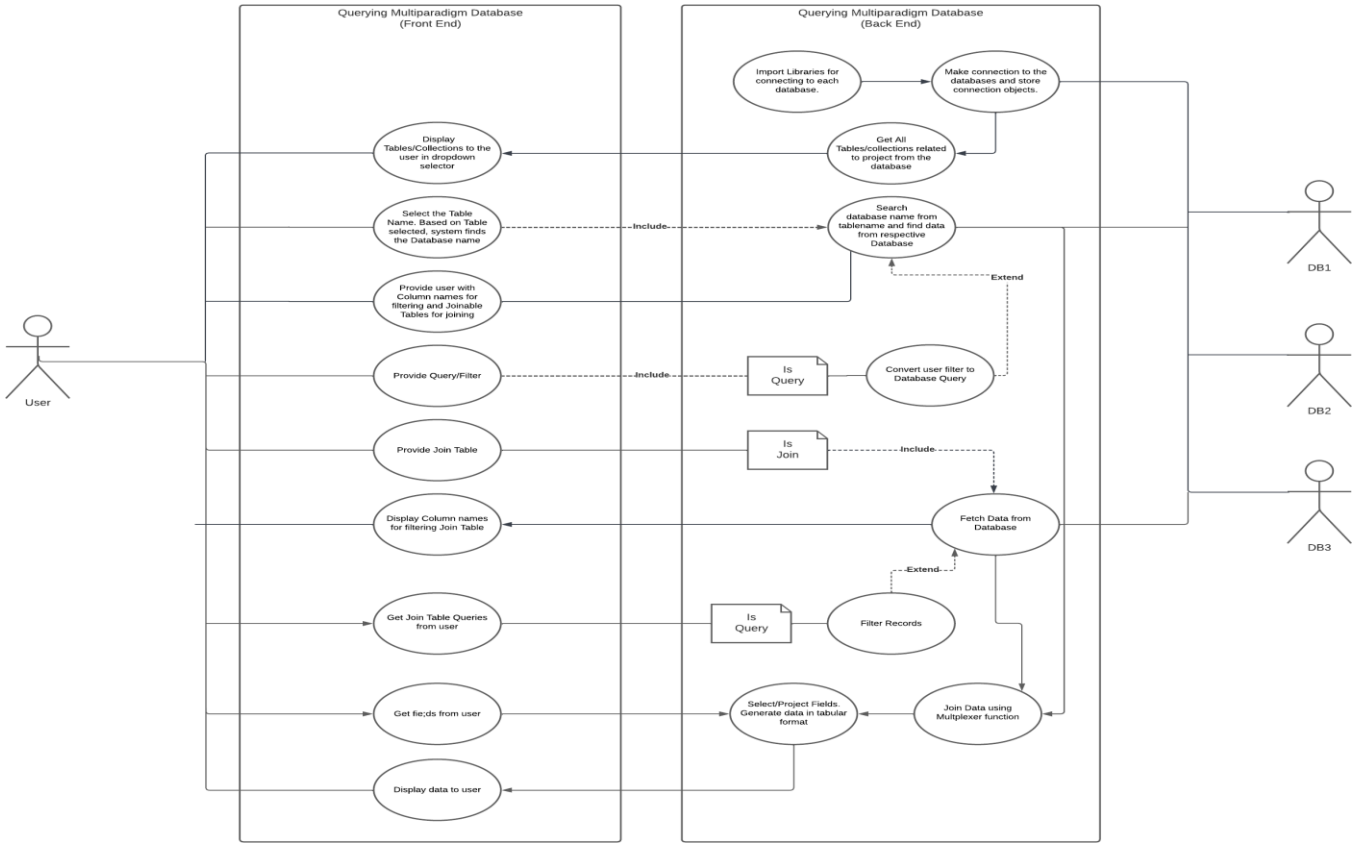

Fig 4: PIM for Querying Multi-Paradigm Databases

7. Conclusion

Platform Independent model is the key step in creating an Application System following MDA approach as it lays a model depicting how each functionality is implemented. Since there is no dependency on the underlying platform, there is plenty of scope for extending this model to any platform, which results in the formation of PSM (Platform Specific Model). MDA approach helped us solve a complex problem of Querying Multiparadigm databases. First we created a Computational Independent model where we focused more on requirement gathering and understanding the business logics involved. After that we construct the PIM as discussed in this research. Post PIM, the next step is to extend PIM by addition of the application development platforms which will result in formation of PSM. Thus the process of application development becomes very systematic and standardized which was the objective of OMG for bringing MDA.

References

[1] J. Bezivin, S. Hammoudi, D. Lopes, F. Jouault, "Applying MDA approach for web service platform," EDOC'04 preceedings of the 8th IEEE International Entreprise Distributed Object Computing Conference, pp. 58-70, 2004.

[2] A. Srai, F. Guerouate, N. Berbiche, H. Drissi, "Generated PSM Web Model for E-learning Platform Respecting n-tiers Architecture," International Journal of Emerging Technologies in Learning (iJET), vol. 12, no. 10, pp. 212-220, 2017.

[3] A. Srai, F. Guerouate, N. Berbiche, H. Drissi, "MDA Approach for EJB Model," 6th IEEE International Conference on Multimedia Computing and Systems (ICMCS'18). DOI: 10.1109/ICMCS.2018.8525924

[4] M. Lachgar, "Approche MDA pour automatiser la génération de code natif pour les applications mobiles multiplateformes," Thèse de Doctorat, 2017.

[5] J. Gordijn, and J. M. Akkermans, "Value based requirements engineering: exploring innovative e-commerce idea," Requirements Engineering Journal 8 (2), 2003, pp. 114–134.

[6] V. D. Castro, E. Marcos, and J. M. Vara, "Applying CIM-to-PIM model transformations for the service-oriented development of information systems," presented at 2nd Information and Software Technology, 2011, pp. 87–105.

[7] A. Rodríguez, I. García-Rodríguez de Guzmán, E. Fernández Medina, and M. Piattini, "Semi-formal transformation of secure business processes into analysis class and use case models: an MDA approach," presented at 9th Information and Software Technology 52, 2010, pp. 945–971.

[8] S. Kherraf, E. Lefebvre, and W. Suryn, "Transformation from CIM to PIM using patterns and archetypes," presented at 19th Australian Conference on Software Engineering, 2008,pp. 338-346.

[9] C. Hahn, P. Dmytro, and K. Fischer, "A model-driven approach to close the gap between business requirements and agent-based execution," presented at Proceedings of the 4th Workshop on Agent-based Technologies and applications for enterprise interoperability, Toronto, Canada, 2010, pp. 13–24.

[10] W. Zhang, H. Mei, H. Zhao, and J. Yang, "Transformation from CIM to PIM: a feature-oriented component-based approach," presented at MoDELS 2005, Montego Bay, Jamaica, 2005.

[11] B. Grammel, and S. Kastenholz, "A generic traceability framework for facet-based traceability data extraction in model-driven software development," presented at the 6th ECMFA

Traceability Workshop held in conjunction ECMFA 2010, Paris, France, 2010, pp. 7–14.

[12] J. J. Gutiérrez, C. Nebut, M. J. Escalona, M. Mejías, and I. M. Ramos, “Visualization of use cases through automatically generated activity diagrams,” presented at 11th international conference on Model Driven Engineering Languages and Systems, France, 2008, pp. 83-96 .

[13] J. Mazón, J. Pardillo, and J. Trujillo, “A model-driven goal-oriented requirement engineering approach for data warehouses,” presented at the Conference on Advances in Conceptual Modeling: Foundations and Applications, Auckland, New Zealand, 2007, pp. 255–264.

[14] Y. Rhazali, Y. Hadi and A. Mouloudi, "Disciplined approach for transformation CIM to PIM in MDA," Model-Driven Engineering and Software Development (MODELSWARD), 2015 3rd International Conference on, Angers, 2015, pp. 312-320.

[15] Y. Rhazali, Y. Hadi and A. Mouloudi, "Transformation approach CIM to PIM: from business processes models to state machine and package models," Open Source Software Computing (OSSCOM), 2015 International Conference on, Amman, 2015, pp. 1-6. doi: 10.1109/OSSCOM.2015.7372686

[16] Y. Rhazali, Y. Hadi and A. Mouloudi, (2016). Model Transformation with ATL into MDA from CIM to PIM Structured through MVC. Procedia Computer Science, 83, 1096-1101. doi:10.1016/j.procs.2016.04.229

[17] Y. Rhazali, Y. Hadi and A. Mouloudi, A Methodology of Model Transformation in MDA: from CIM to PIM, (2015) International Review on Computers and Software (IRECOS), 10(12), pp. 1186-1201.

[18] H. Wijekoon, V. Merunka, “Transformation of Class Hierarchies During Software Development in UML”, (2022) ICDS 2022 : The Sixteenth International Conference on Digital Society, pp. 23-27.

[19] D. Gaspar, M. Mabić, T. Krtalić, “Integrating Two Worlds: Relational and NoSQL”, (2017) Proceedings of the Central European Conference on Information and Intelligent Systems, pp. 11-18.

[20] Ken Ka-Yin Lee,Wai-Choi Tang, and Kup-Sze Choi. Alternatives to Relational Database: Comparison of NoSQL and XML approaches for Clinical Data Storage.Computer Methods and Programs in Biomedicine, 110(1):99–109, 2013.

[21] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach,Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E Gruber. Bigtable: A Distributed Storage System for Structured Data. ACM Transactions on Computer Systems (TOCS), 26(2):4, 2008.

[22] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: Amazon’s Highly Available Key-value Store. 41(6):205–220, 2007.

[23] Matthew Aslett. How will the Database Incumbents Respond to NoSQL and NewSQL. The 451 Group, pages 1–5, 2011.

[24] Katarina Grolinger, Wilson A Higashino, Abhinav Tiwari, and Miriam AM Capretz. Data Management in Cloud Environments: NoSQL and NewSQL Data Stores. Journal of Cloud Computing: Advances, Systems and Applications,2(1):22, 2013.

[25] Pramod J Sadalage and Martin Fowler. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Pearson Education, 2012.

[26] Christof Strauch, Ultra-Large Scale Sites, and Walter Kriha. Nosql databases. Lecture Notes, Stuttgart Media University, 2011.

[27] Peter Membrey, Eelco Plugge, and DUPTim Hawkins. The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing. Apress, 2010.