Revista Cubana de Ciencias Informáticas
Vol. 16, No. 2, Abril-Junio, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

Artículo original

Assessing the impact of an MDA-based approach to support the architecture design

Evaluación del impacto de una propuesta basada en MDA para el diseño arquitectónico

Nemury Silega Martínez [1*] https://orcid.org/0000-0002-8436-5650
Inelda Anabelle Matillo Alcívar [2] https://orcid.org/0000-0001-6810-9668
Gilberto Fernando Castro Aguilar [2,3] https://orcid.org/0000-0001-9050-8550
Katya M. Faggioni Colombo [2] https://orcid.org/0000-0001-6577-5761

[1] Southern Federal University. Taganrog, Russia. nsilega@sfedu.ru
[2] Universidad de Guayaquil – Facultad de Ciencias Matemáticas y Físicas, gilberto.castroa@ug.edu.ec, Inelda.martilloa@ug.edu.ec, katya.faggionic@ug.edu.ec
[3] Universidad Católica de Santiago de Guayaquil– Facultad de Ingeniería. Guayaquil, Ecuador. gilberto.castro@cu.ucsg.edu.ec

*Autor para la correspondencia. (silega@sefedu.ru)

RESUMEN

La fase de diseño de la arquitectura es crucial para el éxito de un proyecto de software. En esta fase se definen los componentes de alto nivel que compondrán el sistema, así como sus relaciones. Por lo general, el diseño de la arquitectura es desarrollado intuitivamente por los arquitectos. Por lo tanto, la calidad del diseño resultante dependerá de las habilidades de los arquitectos. Sin

embargo, es común encontrar errores generados durante esta fase. Los errores que no se detectan en esta fase se propagan a fases posteriores donde su solución demanda más tiempo y esfuerzo, provocando desviaciones en los objetivos del proyecto. Este artículo tiene como objetivo describir los resultados de un experimento para evaluar el impacto de una propuesta de apoyo al diseño de arquitectura. Este enfoque se basa en el paradigma de desarrollo dirigido por modelos (MDD). Reunimos evidencia cuantitativa que muestra el impacto favorable en la reducción de errores y el tiempo utilizado para el diseño arquitectónico.

Palabras clave: arquitectura dirigida por modelos (MDA); diseño arquitectónico; experimento; calidad de software.

ABSTRACT

The architecture design phase is crucial to the success of a software project. In this phase, the high-level components that will compose the system, as well as theirs relationships, are defined. Usually, the architecture design is intuitively developed by the architects. Therefore, the quality of the resulting design will depend on the architects' skills. However, it is common to find errors generated during this phase. The errors that are not detected in this phase are propagated to later phases where their solution demands more time and effort, causing deviations in the project objectives. This article aims to describe the results of an experiment to evaluate the impact of a proposal to support the architecture design. This approach is based on Model-Driven Development (MDD) paradigm. We gathered quantitative evidence showing the favorable impact in reducing errors and the time used for the architectural design.

Keywords: model-driven architecture (MDA); architecture design; experiment; software quality.

Recibido: 02/02/2022

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 2, Abril-Junio, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

Aceptado: 22/03/2022

Introduction

The software development complexity can be analyzed in terms of *essential complexity* and *arbitrary complexity* (Brooks). The essential complexity is inherent to the process (i.e. enterprise management processes, health management processes, etc.). This complexity is inevitable because it is not related to the software development process (Selic, 2008).

On the other hand, the arbitrary complexity is focused on the internal software development process. This complexity considers the technologies and methods used in a software development project as well as the developers' skills. Hence, reducing arbitrary complexity is a common concern of software managers. In that sense, it is critical to adopt the appropriate technologies and methods. For example, adopting a tool that automatically generates some part of the source code will reduce the complexity of the coding stage and, consequently, increase the productivity of the software developers. On the contrary, adopting a method with a high learning curve may reduce the productivity of the process. In general, a high essential complexity increases the need to reduce the arbitrary complexity.

During the architecture design phase, the high-level abstraction components to satisfy the clients' requirements are identified (Al-Jamimi & Ahmed, 2013). Hence, the decisions made in this phase have a considerable influence on the subsequent development phases. The characteristics of complex systems, for example, enterprise management systems, determine a high essential complexity for this phase. Therefore, it is suitable to adopt those technologies and methods that contribute to reducing the arbitrary complexity. Thus, the productivity of architects will be increased, and less time will be necessary to carry out this phase (Al-Jamimi & Ahmed, 2013).

Commonly, this phase is carried out intuitively, depending on architects' creativity (Al-Jamimi & Ahmed, 2013), but no formal guidance or adequate means to check the quality of the resulting design are adopted. These issues lead to frequent redesigns to fix those errors generated in the design phase but detected in subsequent phases. The redesign increases software development time, affects software quality, and increases project costs. Fixing post-implementation errors cost 100 times more than those detected at the design stage (Boehm, 1981; Moreno, Snoeck, Reijers, & Rodríguez, 2014; Sánchez, García, Ruiz, & Mendling, 2012).

The incorrect business process modeling and the correspondence between business processes and design commonly affect the quality of software systems Barjis (2008), especially those with high complexity. In that sense, business process models are the base artifact for architecture design. However, some researchers have demonstrated that business process models usually have errors (Mendling, 2009; Moreno et al., 2014). The incorrect validation of business process models lead to the propagating of their errors to the subsequent phases. This insufficiency increases the costs and efforts to fix those mistakes (Moreno et al., 2014; Sánchez et al., 2012; Sánchez, Ruiz, García, & Piattini, 2013; Wand & Weber, 2002).

On the other hand, Model Driven Development (MDD) paradigm is acknowledged as an alternative to reduce the arbitrary complexity of the software development process by means of increasing the abstraction level to improve the human communication and analysis (Mohagheghi et al., 2011).

Model Driven Architecture (MDA) is the most prominent MDD-based methodology. MDA organizes the software development in three abstraction levels: Computer Independent Models (CIMs), Platform Independent Models (PIMs), and Platform Specific Models (PSMs).

Taking into account the aforementioned benefits of MDA, we developed a MDA-based approach to support the architecture design phase. In that sense, this paper aims to describe the results of an experiment conducted to demonstrate the impact of the developed approach to enhance the architecture design of enterprise management systems. Our approach is based on MDA and ontologies, and a tool was developed to support the transformation and validation of models. The experiment yielded empirical evidence that corroborates the approach's applicability and impact.

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 2, Abril-Junio, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

The rest of the paper is structured as follows. Next section presents the methods adopted in this research. Then, the main components of our approach are introduced. In the Result section the results of the experiment are analyzed. Finally conclusions and future work are presented.

Methods or computational methodology

Model Driven Architecture (MDA)

MDA is a methodology promoted by the Object Management Group (OMG). It is a model-driven proposal because it defines models as the main artifacts for understanding, designing, developing, implementing, and maintaining systems (OMG, 2003).

MDA aims to develop systems with high flexibility in implementation, integration, maintaining, and testing. Portability, reusability and interoperability are the three main MDA principles. As we mentioned above, three types of models are defined in MDA: Computer Independent Models (CIMs), Independent Platform Models (PIMs) and Platform Specific Models (PSMs).

A CIM is a view of the business, regardless of system specifications. Bridging the gap between business experts and software developers is the main goal of this type of model. A PIM is a system view that does not include platform specifications. This type of model contributes to separating the logical design concerns from the platform-specific concerns. Whereas, a PSM is a view of the system that includes the details of a specific platform.

Model transformations is the other key component of MDA. A model transformation consists of generating a new model from others of the same system. Achieving the automatic model transformation is the aim of the scientific community to increase the productivity in the software development process and enhance the software quality.

Several studies have demonstrated the positive impact of MDA-based approaches to improve the productivity of the software development process and the quality of the resulting software (Bocanegra, Peña, & Ruiz Cortés, 2008), (Singh & Sood, 2010) (De Castro, Marcos, & Vara, 2010; Kharmoum, Ziti, Rhazali, & Fouzia, 2019; Melouk, Rhazali, & Youssef, 2020; Mora et al., 2008; Sánchez Vidales, Fermoso García, & joyanes Aguilar, 2008).

Ontologies

An ontology is a formal and explicit description of a discourse domain concepts (classes), the properties of each concept, attributes, and restrictions (Noy & McGuinness, 2001). The ontologies have been extensively exploited to represent and analyze knowledge in several domains (Bencharqui, Haidrar, & Anwar, 2022; Bouzidi, Nicola, Nader, & Chalal, 2019; Freitas, Canedo, & Jesus, 2018; Keet & Grütter, 2021; Nicola, Melchiori, & Villani, 2019; Silega & Noguera, 2021; Xinga, Zhonga, Luoa, Lic, & Wua, 2019; Yang, Cormicana, & Yub, 2019). The application of ontologies in the MDA context may enable the models consistency checking and validation. This new approach has been named Ontology Driven Architecture (ODA) (W3C, 2006). We found several MDA-based works (Kharmoum et al., 2019; Melouk et al., 2020) which include models to represent and validate processes through formal models (Laaz, Kharmoum, & Mbarki, 2020; Laaz, Wakil, Gotti, Gotti, & Mbarki, 2019; Li, Zhou, & Ye, 2019).

Some of the most important languages to represent ontologies are: Ontolingua, XML Schema, RDF (Resource Description Frame-work), RDF Schema (o RDF-S), and OWL (Xing & Ah-Hwee, 2010). OWL includes a set of operators to represent different types of relations, such as intersection, union, and negation. Since OWL is based on description logic, it is possible to employ reasoners to automatically check the models consistency. In addition, the tool Protégé supports the management of ontologies in OWL.

The benefits of adopting OWL ontologies have been extensively demonstrated (Pahl, Giesecke, & Hasselbring, 2009), (Bo & Li-juan, 2009), (Chengpu, Rob, & Xiaodong, 2010), (Chungoora & Young, 2008) y (Kruchten, 2004). Considering these advantages, we adopted an ontology-based approach to carry out the architecture design. We carried out an analysis of some important methodologies to develop ontologies (Kotis, Vouros, & Spiliotopoulos, 2020; Kumar, 2017). Finally, the methodology of Noy and McGuinness was adopted to develop our ontology. This methodology has been extensively adopted to guide the development of ontologies (Sattar, Surin, Ahmad, Ahmad, & Mahmood, 2020).

An approach based on MDA and ontologies to support the architecture design

In this section, we briefly introduced an approach to support the architecture design. A detailed description of this approach is presented in (Silega, Noguera, & Macias, 2016). Figure 1 depicts an overall view of the proposal. In Fig. 1, the rectangles represent the types of models, and the arrows represents the transformations. Three CIMs and two PIMS are included in the proposal. In addition, four models transformations are included as well. Previous works have described the aim of each model (Silega, Loureiro, & Noguera, 2014; Silega, Macías, Matos, & Febles, 2014).

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 2, Abril-Junio, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

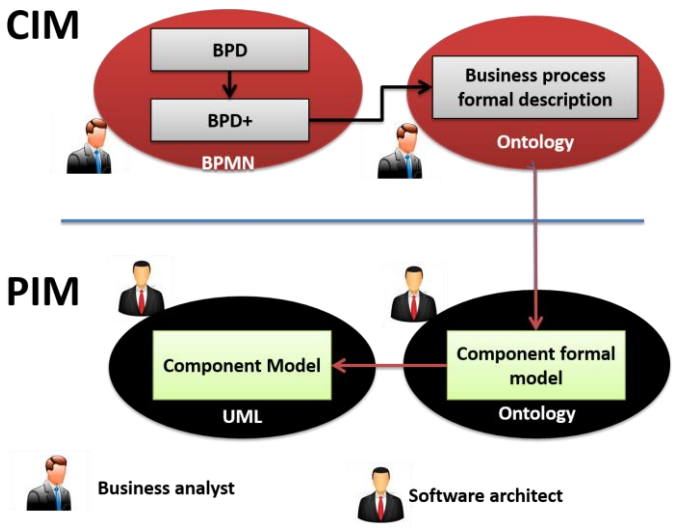

Fig. 1 – Overall view of the proposal: Models and transformations.

Computer Independent Models

BPD: This model allows creating business processes descriptions so that it can be easily understood by both business specialists and software developers. However, some particular domain concepts cannot be represented in this model. Therefore, the following CIM was included in this level.

BPD+: It is an intermediate model to extend the BPD with the particular-domain information that the BPD could not represent.

Ontological model: This ontology includes classes, properties, and restrictions to represent a business process. Some classes are related to BPMN concepts, such as Activity, Processs, FlowElement, Event, Gateway, and Object. The ontology also includes object properties to relate individuals.

Platform Independent Models

At this level, a high-level architecture view is provided. This view includes the components and their relationships. This representation aims to show how the system components are coordinated

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 2, Abril-Junio, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

to meet the business needs. The most significant concepts are Component, Service, and Functionality.

Ontological component model: The ontology includes classes and properties to represent the system architecture. Some relevant classes are Component, Service, and Functionality. A set of object properties were defined to relate the model concepts. For example, it was stated that a Service is provided by a Component and a Component implements some Functionalities. Furthermore, some classes to classify the components were included, for example, BusinessComponent, DomainComponent, and TechnologyComponent.

UML component model: Representing the component model by means of an ontology provides the benefits that were mentioned above. However, this type of model is not easy to understand for those who are non-experts with these technologies. Hence, it is suitable to include a model that is easy to understand by software developers. In that sense, the UML component model has been adopted. This is a well-known model and it is interpreted by a wide variety of modeling tools. As Fig. 1 shows, the PIM level is composed of an ontological model and an UML component model.

Results and Discussion

A tool to support the transformations

The tool that we developed to support the approach is described below. This tool was created as a Protégé plugin to automate the three transformations that the approach includes. These transformations are:

1. **BPD -> BPD+**: In this step, the BPD is extended with the specific domain information.
2. **BPD+ -> Ontological business process model**: With this transformation, an ontological representation of the processes is generated from the BPD. A set of transformation rules were defined to generate this model.

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 2, Abril-Junio, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

3. **Ontological business process model -> Ontological component model**: This is a CIM to PIM transformation. In this case, a component model is generated from the business process model. Both models are represented by means of ontologies.
4. **Ontological component model -> UML component model**: A UML component model is generated from the ontological component model in this transformation.

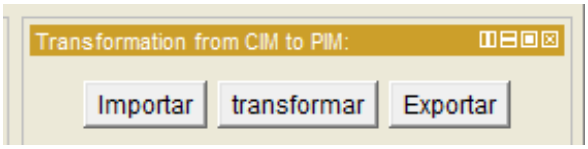

Fig. 2 – Plugin functionalities.

Considering that the main goal of this paper is to present the results of two experiments conducted to demonstrate the impact of our approach and the participants in this experiments speak English, we show a picture with the Spanish version of the plugin. Likewise, the models generated during the experiments are shown in the Spanish language. Figure 2 depicts the basic view of the plugin, which includes three functionalities. Functionality *Importar* supports the transformation one and two. Whilst the functionalities *Transformar* and *Exportar* support the transformations three and four, respectively. To illustrate how the tool works, we present the results of a study case. A BPD of the process *Liquidar pagos anticipados* is the first model (Figure 3). The functionality *Importar* opens the interface shown in Figure 4. In this window, the activities of the BPD will be listed. It is possible to add the domain-specific information for each activity. Fig. 4 shows the activities of the BPD that was depicted in Fig. 3. Figure 5 shows the activities generated in the ontology after executing the transformation.

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 2, Abril-Junio, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

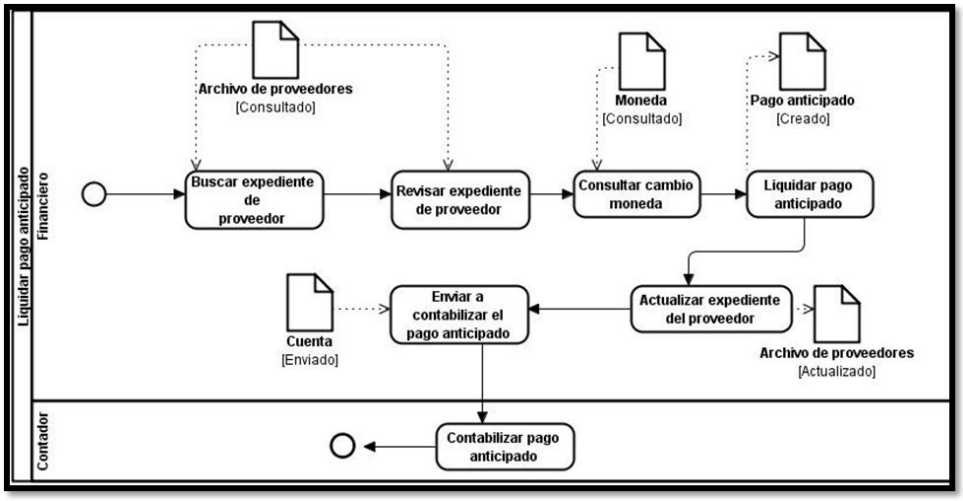

Fig. 3 – BPD of the process *Liquidar pago anticipado*.

Once the ontological business process model has been generated, it is possible to transform it into the component model (*Transformation 3*). To carry out this transformation, the functionality *Transformar* is executed. After executing this functionality, the ontological component model is automatically generated. Fig. 6 shows the instances of the classes Component and Functionality that was created in the ontology.

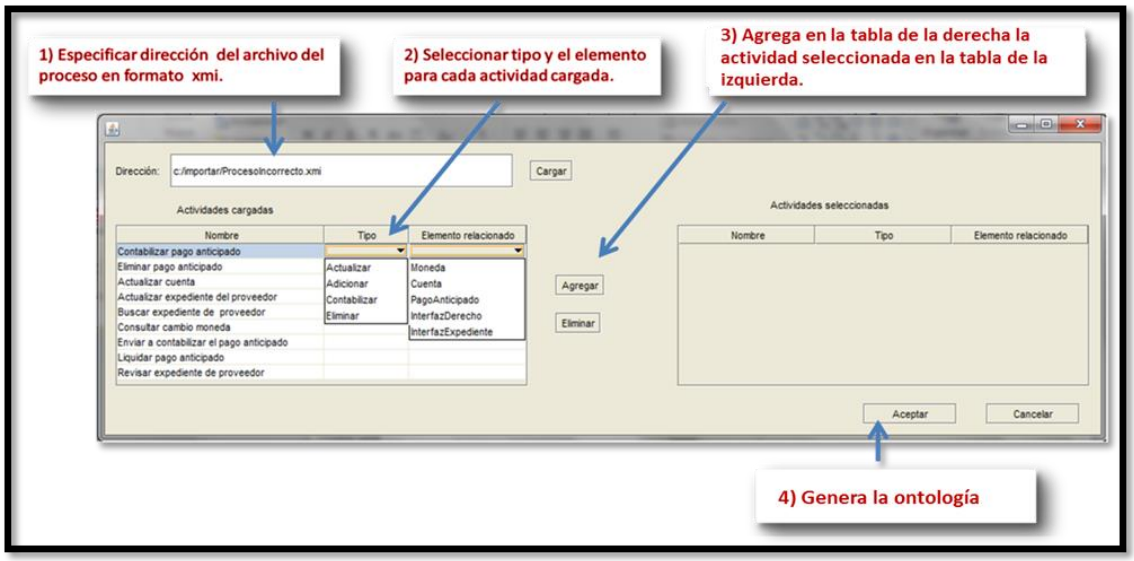

Fig. 4 – View of the interface to import a BPD and transform it into an ontology.

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 2, Abril-Junio, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

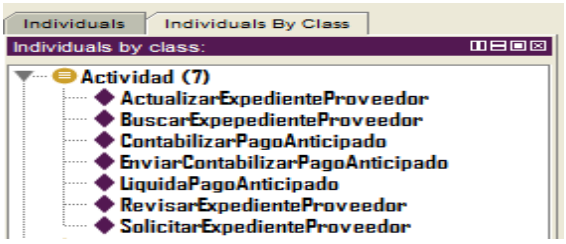

Fig. 5 – Generated instances of the class *Activity*.

Once the business process formal model has been generated, it is possible to transform it into a component model (*Transformation 3*). The functionality *Tranformar* of the plugin is employed to carry out this transformation. After executing this functionality, a component model is generated. Figure 6 depicts the generated instances of the classes Component and Functionality.

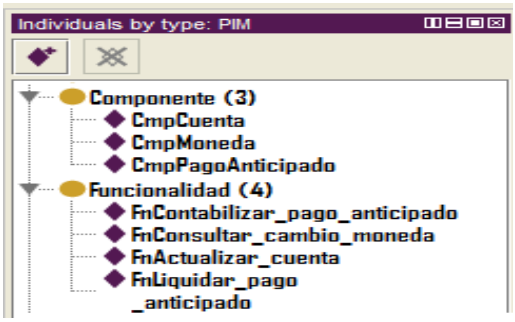

Fig. 6 – Generated instances of the classes Component and Functionality.

Once the ontological component model has been created, it is possible to transform it into a UML component model (Transformation 4). The functionality *Exportar* will automatically generate this new model. Figure 7 shows the generated component model. In this model, the unimplemented functionalities are highlighted with yellow color. Whilst the business components and domain components are highlighted with the color red and blue, respectively. Using different colors according to the characteristics of the model elements will facilitate its analysis.

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 2, Abril-Junio, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

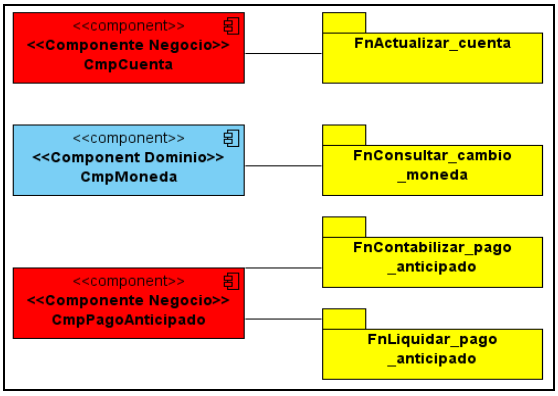

Fig. 7 – Generated UML component model.

Assessing the impact of our approach

To assess the impact of our approach, two (cuasi) experiments were conducted: (1) to evaluate the capacity to detect mistakes in the business process models and (2) to assess the capacity to avoid mistakes in the component models. The participants in these experiments were members of a project to develop an enterprise management system.

Assessing the capacity to detect mistakes in the business process models

The participants in this experiment were nine engineers and three students with experience modeling business processes. We provided the participants with a description of the process *Liquidar cobro anticipado*. Since we wanted to assess the capacity of the participants to detect mistakes, we included ten mistakes in this BPD. The participants were asked to review this BPD and to identify mistakes (if exist). Table 1 depicts the number of mistakes detected by each participant. In a second measurement, we reviewed the BPD applying our approach. At this time, all mistakes in the BPD were detected.

Table 1 – Number of mistakes detected for each participant.

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 2, Abril-Junio, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 137-156

Participant	1	2	3	4	5	6	7	8	9	10	11	12
Detected mistakes	3	3	2	3	10	10	4	4	6	5	3	1

Assessing the capacity to avoid mistakes in the component models

The second experiment was conducted with 11 participants who had at least 2-year experience as architects or programmers. The participants received the right description of the process *Liquidar pago anticipado* and were asked to elaborate a proposal of architectonic design based on the design practices defined in their projects. Table 2 shows the number of mistakes that each participant made. The minimum number o mistakes made by an architect was two, and the medium of errors was 4,27. The following are some of the most common mistakes:

1. The design does not cover the activities of the process. Hence, the design does not meet the user's needs.
2. Some components do not have a well-defined responsibility. This problem leads to unnecessary integrations and consequently to the system performance and reuse.
3. Violations of design rules that lead to high coupling and low cohesion. Hence, two of the most important design principles are not met (Larman, 1999).

Table 2 – Number of mistakes made for each participant.

Participant	1	2	3	4	5	6	7	8	9	10	11
Number of mistakes	7	5	2	2	5	6	3	3	5	7	3

We used our proposal to carry out the architectonic design for the second measurement. As a result, no mistakes were made, thus demonstrating the positive impact of our proposal. The use of ontologies and automated transformations are two of the main factors contributing to these results.The experiments also provided evidence that our proposal contributed to homogenizing the

design. Only two design proposals at the first measurement have high similarity. However, using our proposal, the resulting design was always the same.

Conclusions

This paper introduced an approach based on MDA and ontologies to support the architecture design. We briefly described a tool that includes the functionalities to support the creation and automated transformation of models. The use of this tool increases the productivity of the software development process and contributes to ensure the software quality. To demonstrate the impact of or proposal, two (cuasi) experiments were conducted. These experiments provided evidence that demonstrates the good performance of this approach to detect mistakes in business process models, reduce the number of mistakes during the architectonic design, and homogenize the design rationale. We are currently working to extend the proposal to cover other views of the software development process. Likewise, we are designing new experiments to provide new evidences that demonstrate the impact of our proposal on different variables of the software development process.

References

Al-Jamimi, H., & Ahmed, M. (2013). Transition from Analysis to Software Design: A Review and New Perspective. *International Journal of Soft Computing and Software Engineering (JSCSE), 3*(3), 169-176.

Barjis, J. (2008). The importance of business process modeling in software systems design. *Science of Computer Programming, 71*(1), 73-87.

Bencharqui, H., Haidrar, S., & Anwar, A. (2022). Ontology-based Requirements Specification Process. *E3S Web Conf., 351*.

Bo, D., & Li-juan, S. (2009). Ontology-Based Model for Software Resources Interoperability. *Information Technology Journal, 8*(6), 871-878.

Bocanegra, J., Peña, J., & Ruiz Cortés, A. (2008). *Una Aproximación MDA para Modelar Transacciones de Negocio a Nivel CIM.* Paper presented at the Jornadas de Ingeniería del Software y Bases de Datos, España.

Boehm, B. W. (1981). *Software Engineering Economics*: Prentice-Hall, Englewood Cliffs.

Bouzidi, R., Nicola, A. D., Nader, F., & Chalal, R. (2019). OntoGamif: A modular ontology for integrated gamification. *Appl. Ontology, 14*(3), 215-249. doi: 10.3233/AO-190212

Chengpu, L., Rob, P., & Xiaodong, L. (2010). ONTOLOGY-BASED QUALITY ATTRIBUTES PREDICTION IN COMPONENT-BASED DEVELOPMENT. *International Journal of Computer Science & Information Technology, 2*(5), 12-29.

Chungoora, N., & Young, R. I. M. (2008). *Ontology Mapping to Support Semantic Interoperability in Product Design and Manufacture.* Paper presented at the MDISIS 2008.

De Castro, V., Marcos, E., & Vara, J. M. (2010). Applying CIM-to-PIM model transformations for the service-oriented development of information systems. *Information and Software Technology, 53*(1), 87-105.

Freitas, S., Canedo, E., & Jesus, D. (2018). Calculating similarity of curriculum lattes. *IEEE Latin America Transactions, 16*(6), 1758-1764.

Keet, C. M., & Grütter, R. (2021). Toward a systematic conflict resolution framework for ontologies. *Journal of biomedical semantics, 12*(1). doi: 10.1186/s13326-021-00246-0

Kharmoum, N., Ziti, S., Rhazali, Y., & Fouzia, O. (2019). A Method of Model Transformation in MDA Approach from E3value Model to BPMN2 Diagrams in CIM Level. *IAENG International Journal of Computer Science, 46*(4).

Kotis, K. I., Vouros, G. A., & Spiliotopoulos, D. (2020). Ontology engineering methodologies for the evolution of living and reused ontologies: status, trends, findings and recommendations. *The Knowledge Engineering Review, 35*, e4. doi: 10.1017/S0269888920000065

Kruchten, P. (2004). *An Ontology of Architectural Design Decisions in Software-Intensive Systems*. Vancouver, B.C., Canada . University of British Columbia.

Kumar, M. K. (2017). *Creation of Dynamic Ontologies for Graphical Representation in User Interface using NeOn in Shodhganga.* Paper presented at the ETD2017 Symposium.

Laaz, N., Kharmoum, N., & Mbarki, S. (2020). Combining Domain Ontologies and BPMN Models at the CIM Level to generate IFML Models. *Procedia Computer Science, 170*, 851-856.

Laaz, N., Wakil, K., Gotti, Z., Gotti, S., & Mbarki, S. (2019). *Integrating Domain Ontologies in an MDA-Based Development Process of e-Health Management Systems at the CIM Level.* Paper presented at the International Conference on Advanced Intelligent Systems for Sustainable Development.

Larman, C. (1999). *UML Y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado* (1 ed.). México: Prentice Hall.

Li, Z., Zhou, X., & Ye, Z. (2019). A Formalization Model Transformation Approach on Workflow Automatic Execution from CIM Level to PIM Level. *International Journal of Software Engineering and Knowledge Engineering, 29*(09), 1179-1217.

Melouk, M., Rhazali, Y., & Youssef, H. (2020). An Approach for Transforming CIM to PIM up To PSM in MDA. *Procedia Computer Science, 170*, 869-874.

Mendling, J. (2009). Empirical Studies in Process Model Verification. In K. Jensen & W. M. P. van der Aalst (Eds.), *Transactions on Petri Nets and Other Models of Concurrency II* (Vol. 5460, pp. 208-224): Springer Berlin Heidelberg.

Mohagheghi, P., Gilani, W., Stefanescu, A., Fernandez, M. A., Nordmoen, B., & Fritzsche, M. (2011). Where does model-driven engineering help? Experiences from three industrial cases. *Software & Systems Modeling*.

Mora, B., García, F., Ruiz, F., Piattini, M., Boronat, A., Gómez, A., . . . Ramos, I. (2008). Software generic measurement framework based on MDA. *IEEE Latin America Transactions, VOL. 6,*(4).

Moreno, I., Snoeck, M., Reijers, H. A., & Rodríguez, A. (2014). A systematic literature review of studies on business process modeling quality. *Information and Software Technology*.

Nicola, A. D., Melchiori, M., & Villani, M. L. (2019). Creative design of emergency management scenarios driven by semantics: An application to smart cities. *Inf. Syst., 81*, 21-48. doi: 10.1016/j.is.2018.10.005

Noy , N. F., & McGuinness, D. L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. Stanford: Stanford Medical Informatics.

OMG. (2003). *MDA Guide Version 1.0.1*.

Pahl, C., Giesecke, S., & Hasselbring, W. (2009). Ontology-based modelling of architectural styles. *Information and Software Technology, 51*(12), 1739-1749.

Sánchez, L., García, F., Ruiz, F., & Mendling, J. (2012). Quality indicators for business process models from a gateway complexity perspective. *Information and Software Technology, 54*(11), 1159-1174.

Sánchez, L., Ruiz, F., García, F., & Piattini, M. (2013). Improving Quality of Business Process Models. In L. Maciaszek & K. Zhang (Eds.), *Evaluation of Novel Approaches to Software Engineering* (Vol. 275, pp. 130-144): Springer Berlin Heidelberg.

Sánchez Vidales, M. Á., Fermoso García, A., & joyanes Aguilar, L. (2008). Una recomendación basada en MDA, BPM y SOA para el desarrollo de software a partir de procesos del negocio en un contexto de Negocio Bajo Demanda.

Sattar, A., Surin, E. S. M., Ahmad, M. N., Ahmad, M., & Mahmood, A. K. (2020). Comparative analysis of methodologies for domain ontology development: A systematic review. *Int. J. Adv. Comput. Sci. Appl., 11*(5), 99-108.

Selic, B. (2008). Manifestaciones sobre MDA. *Novática*(192), 13-16.

Silega, N., Loureiro, T. T., & Noguera, M. (2014). Marco de trabajo dirigido por modelos y basado en ontologías para la descripción y validación semántica de procesos de negocio. *Latin America Transactions, IEEE (Revista IEEE America Latina), 12*(2), 292-299.

Silega, N., Macías, D., Matos, Y., & Febles, J. P. (2014). Framework basado en MDA y ontologías para la representación y validación de modelos de componentes. *Revista Cubana de Ciencias Informáticas (RCCI), 8*(2), 85-101.

Silega, N., & Noguera, M. (2021). Applying an MDA-based approach for enhancing the validation of business process models. *Procedia Computer Science, 184*, 761-766. doi: https://doi.org/10.1016/j.procs.2021.03.094

Silega, N., Noguera, M., & Macias, D. (2016). Ontology-based Transformation from CIM to PIM. *IEEE Latin America Transactions, 14*(9), 4156-4165.

Singh, Y., & Sood, M. (2010). The Impact of the Computational Independent Model for Enterprise Information System Development. *International Journal of Computer Applications, 11*(8).

W3C, E. P. T., Jeff Z. Pan, Daniel Oberle, Evan Wallace, Michael Uschold, Boeing, Elisa Kendall, . (2006, Editors' Draft Date: 2006/02/11). Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering. Retrieved 6 de febrero, 2011, from http://www.w3.org/2001/sw/BestPractices/SE/ODA/

Wand, Y., & Weber, R. (2002). Research commentary: information systems and conceptual modeling - a research agenda. *Information Systems Research, 13*(4), 363-376.

Xing, J., & Ah-Hwee, T. (2010). CRCTOL: A semantic-based domain ontology learning system. *Journal of the American Society for Information Science & Technology, 61*(1), 150-168.

Xinga, X., Zhonga, B., Luoa, H., Lic, H., & Wua, H. (2019). Ontology for safety risk identification in metro construction. *Computers in Industry, 109*, 14–30.

Yang, L., Cormicana, K., & Yub, M. (2019). Ontology-based systems engineering: A state-of-the-art review. *Computers in Industry, 111*, 148–171. doi: https://doi.org/10.1016/j.compind.2019.05.003.

Conflicto de Intereses

No Aplica

Contribuciones de los autores

Conceptualización: Nemury Silega Martínez

Curación de datos: Gilberto Fernando Castro Aguilar, Inelda Anabelle Matillo Alcívar y Katya M. Faggioni Colombo

Análisis formal: Nemury Silega Martínez

Adquisición de fondos: Nemury Silega Martínez

Investigación: Nemury Silega Martínez, Gilberto Fernando Castro Aguilar, Inelda Anabelle Matillo Alcívar y Katya M. Faggioni Colombo

Metodología: Nemury Silega Martínez y Katya M. Faggioni Colombo Inelda Anabelle Matillo Alcívar

Administración del proyecto: Nemury Silega Martínez

Recursos: Nemury Silega Martínez

Software: Nemury Silega Martínez y Inelda Anabelle Matillo Alcívar

Supervisión: Gilberto Fernando Castro Aguilar, Inelda Anabelle Matillo Alcívar y Katya M. Faggioni Colombo

Validación: Nemury Silega Martínez

Visualización: Gilberto Fernando Castro Aguilar, Inelda Anabelle Matillo Alcívar y Katya M. Faggioni Colombo

Redacción – borrador original: Nemury Silega Martínez y Inelda Anabelle Matillo Alcívar

Redacción – revisión y edición: Nemury Silega Martínez