Journal of Web Engineering, Vol. 16, No. 3&4 (2017) 212–227

APPLYING A MODEL-BASED METHODOLOGY TO DEVELOP WEB-BASED SYSTEMS OF SYSTEMS

M.A. BARCELONA
IT Area, ITAINNOVA; IWT2 Research Group, University of Seville
Zaragoza, Spain; Seville, Spain
mabarcelona@itainnova.es; miguel.barcelona@iwt2.org

L. GARCÍA-BORGOÑÓN
IT Area, ITAINNOVA; IWT2 Research Group, University of Seville
Zaragoza, Spain; Seville, Spain
laurag@itainnova.es; laura.garcia@iwt2.org

G. LÓPEZ-NICOLÁS
IT Area, ITAINNOVA; IWT2 Research Group, University of Seville
Zaragoza, Spain; Seville, Spain
glopez@itainnova.es; guillermo.lopez@iwt2.org

I. RAMOS
IWT2 Research Group, University of Seville
Seville, Spain
iramos@us.es

M.J. ESCALONA
IWT2 Research Group, University of Seville
Seville, Spain
mjescalona@us.es

Received June 8, 2016
Revised March 9, 2017

Systems of Systems (SoS) are emerging applications composed by subsystems that interacts in a distributed and heterogeneous environment. Web-based technologies are a current trend to achieve SoS user interaction. Model Driven Web Engineering (MDWE) is the application of Model Driven Engineering (MDE) into the Web development domain. This paper presents a MDWE methodology to include Web-based interaction into SoS development. It's composed of ten models and seven model transformations and it's fully implemented in a support tool for its usage in practice. Quality aspects covered through the traceability from the requirements to the final code are exposed. The feasibility of the approach is validated by its application into a real-world project. A preliminary analysis of potential benefits (reduction of effort, time, cost; improve of quality; design vs code ratio, etc) is done by comparison to other project as an initial hypothesis for a future planned experimentation research.

Keywords: Model-Based Web Engineering, Interaction Flow Modeling Language, System of Systems

1. Introduction

Current applications are large, complex, distributed, decentralized and based on the composition of heterogeneous and (semi)autonomous elements becoming, in fact, as systems of systems (SoS) [1]. This implies that a SoS is a composition: it consists of components that are themselves systems [2]. Web-based technologies are a current trend to achieve SoS user interaction.

Web Engineering is described as the use of scientific, engineering and management principles and systematic approaches with the aim of successfully developing, deploying and maintaining high quality Web-based systems and applications [3]. The lack of a standard notation, the high number of approaches, models and techniques and the lack of support tools for development are some of the research lines covered in the Web Engineering field in the latest years [4].

Model-Driven Engineering (MDE) focuses on abstract representations of knowledge and activities that govern a domain specific application in order to handle the complexity of systems development [5]. Models, metamodels and transformations are the key elements in MDE [6]. Model Driven Web Engineering (MDWE) is the application of MDE into the Web development domain, as Web Engineering is a specific domain in which MDE can be usefully applied [7].

In 2001 we set out to create a methodological approach, named *Cervantes*, that could be reused in the design and development of distributed systems, and it has been used in more than 15 real projects with companies in practice. In 2014 by using a Model-Driven Reverse Engineering (MDRE) process[8], the underlying metamodel was discovered [9]. In the usage of the methodology in practice we have evidenced the need to include Web-based interaction. According to the lessons learned in the usage of the methodology, and taking the emerging metamodel and the Web-based interaction gap, we have extended the model-based approach to more phases of the life cycle and composed a new methodology that has been validated by its application in a real-world project.

This paper presents a MDWE methodology to include Web-based interaction into SoS development. The feasibility of the approach is validated by its application into a real-world project. A preliminary analysis of potential benefits (reduction to effort, time, cost, improve of quality, etc) is done by comparison to other project as an initial hypothesis for a future planned experimentation research.

The remainder of this paper is structured as follows. Section 2 presents the related work in Web Engineering Approaches (WEA) and Model-Driven Web Engineering (MDWE) methodologies. The model-based methodology is detailed in Section 3. The industrial validation is exposed in Section 4. Finally, Section 5 shows preliminary lessons learned, conclusions and future work.

2. Related Work

This section exposes related work of MDWE methodologies and Web-based interaction approaches. The first group focuses on proposals in order to cover Web-based interaction. In the latest years, there has been a growing interest in order to create a framework of reference for Web Engineering. There have been a lot of initiatives in order to represent web and user interaction by using models, as follows. The Web Modelling Language (WebML) [10] for data-intensive Web applications. Web Application Extension for UML (WAE) [11], a UML

extension that describes Web application interfaces and client-server interactions. WebDSL [12], a domain-specific language (DSL) that defines entities, pages and business logic. A survey of Web methodologies is presented by Escalona and Aragón [4] and updated by Brambilla et al. [13] who cited initiatives regarding the application of model-driven techniques for web interaction, as TERESA [14], MARIA [15], MBUE[16], UsiXML [17] and UCP [18]. The Interaction Flow Modeling Language (IFML) supports the specification of the front end of applications independently of the technological details of their realization. It addresses the composition and context of the view, the commands, actions, effects of interaction and parameter binding [19]. The Object Management Group (OMG) [20] adopted the IFML [21] as a standard in July 2014.

Then we exposes related reviews and initiatives in the field of MDWE. Barry et al. [22] reviewed methods and techniques used for multimedia and Web development. They concluded that while developers do not want a cumbersome and expensive methodology, there was a clear perception that they would like to add structure to the development process. Mendes [23] performed a systematic review of Web Engineering Research in order to analyze how rigorous were claims in previous research, with the conclusion that more than 68% of selected studies were not. Mittal and Zeigler [24] proposed DEVS Unified Process (DUNIP), a model-based methodology for Web-Centric Development and Testing of SoS. Its uses Discrete Event System Specification (DEVS) formalism and proposes an XMLbased DEVS Modeling Language (DEVSML) for modeling that are deployed in a real-time infrastructure by using SOA-based agents. Their proposal is focused on the integration of heterogeneous SoS and on the assessing of interactions and interoperability in real-time by using simulations. Vallecillo et al. [25] proposed MDWEnet, with the objective to improve practices and tools for model-driven development of Web applications, in order to achieve interoperability among existing MDWE Methods. Meliá and Gómez [26] presented Web Software Architecture (WebSA) that provides a set of architectural models that may be used by the designer to specify a Web Application, taking the Model-Driven Architecture (MDA) paradigm. Escalona and Aragón [4] proposed Navigational Development Techniques (NDT), a model-based approach to deal with requirements in Web development. They proposed a metamodel and a support tool that implements transformations in a Web-based development.

As a conclusion, the great number of initiatives shows that Web Engineering is an important research line in software engineering. The lack of standard notation and the use of results of research in the MDE domain have been addressed in the latest years. Our approach follows a MDE approach including the recent IFML [21] standard to cover Web-based interaction modeling.

3. Model-based Methodology

This section details the methodology by: i) its main life cycle phases which may be applied according to waterfall, agile or any software development paradigm; ii) its models; iii) its transformations into models and text; iv) the implementation in a support tool, and; v) the quality assurance aspects covered.

3.1. *Overview*

A big picture of the methodology with its main phases, models and transformations is

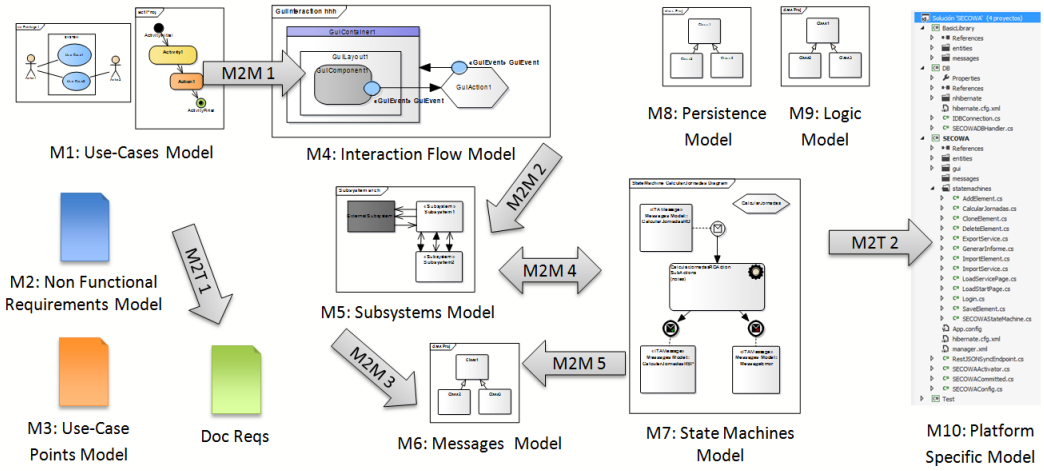

Fig. 1. Main phases, models and transformations of the methodology

shown in the Figure 1. The proposal is itself a methodological framework which may be combined with waterfall, agile or any software development paradigm. Anyway, it proposes a set of main phases that can be performed sequentially or iteratively. Its main stages are:

- Requirements: the purpose is to obtain a minimum understanding of the needs in order to achieve an effort estimation.
- Analysis: in this stage the user interaction is concreted by using mockups and the architecture of systems of systems and their services is defined.
- Design: the internal behavior of each subsystem according to the interface definition is designed as well as the persistence elements.
- Implementation: it includes coding and deployment of the solution according to *Cervantes* Runtime Framework [27].

3.2. *Models*

This section enumerates the list of models used in the proposal, according to the identifiers shown in the Figure 1:

- Use cases Model (M1): for functional requirements the Unified Modeling Language (UML) use cases model [28] is used. A step by step definition of the scenario may be concreted by using natural language or UML Activity diagrams [29] which are mandatory for alternative scenario description. Actions are defined with *tags* which represent user actions and are written by the prefix ##.
- Non Functional Requirements Model (M2): a template is used to elicit all aspects regarding security, performance, maintenance, portability, standardizations, technological restrictions among others. A list is modeled in natural language.

- Use Case Points Model (M3): at early stage an effort estimation is required based on a minimum understanding of requirements. For this purpose, use case points [30] technique is followed. The conversion from effort to cost is done based on historical data which are updated at the end of each project.
- Interaction Flow Model (M4): the user interaction is modeled by using IFML [21], an OMG [20] standard that addresses the composition and context of the view, the commands, actions, effects of interaction and parameter binding [19].
- Subsystems Model (M5): the systems of systems architecture is covered by a set of subsystems according to *Cervantes* metamodel [9].
- Messages Model (M6): each subsystem interacts to others following a service-oriented paradigm, based on Messages, which are a Platform Independent Model (PIM) view of interfaces.
- State Machines Model (M7): once a request is received, the Runtime Framework handles it by using State Machines, which represents the internal behavior of a subsystem, and are modeled again using *Cervantes* [9] metamodel.
- Persistence Model (M8): entities at memory and their conversion into persistent data may be modeled by using UML Class diagrams.
- Logic Model (M9): internal data structures and logic elements may be modeled in UML.
- Platform Specific Model (PSM, M10): all elements are at the end transformed into a concrete programming language, being C#.net, Java and C++ supported by the *Cervantes* Runtime Framework.

3.3. *Model-to-Model transformations*

Regarding the big picture exposed in the Figure 1, this section summarizes the Model-to-Model (M2M) transformations included in the methodological proposal as follows:

- M2M 1: from the use case models, according to actions described by using *tags*, it generates corresponding user actions and system events in the interaction flow model.
- M2M 2: from the interaction flow model actions, it creates the communication patterns between subsystems through Messages.
- M2M 3: for each communication included in the subsystems model, its Message is included in the Messages model.
- M2M 4: for all incoming Message in a subsystem, generates the State Machine responsible to handle it. When the State Machine is designed, if new internal Messages are created, the bidirectional model transformation creates the communication in the subsystem model.
- M2M 5: all new Messages included in the State Machines model are integrated in the Messages model.

3.4. *Model-to-Text transformations*

Moreover, according to the Figure 1, main Model-to-Text (M2T) transformations are shown in this section:

- M2T 1: all requirements, functional, not functional and the estimation is composed into an office document.
- M2T 2: all models are converted into a *Cervantes* Runtime Framework compliant PSM, currently in C#.NET. This M2T creates a Visual Studio Project where all subsystems, Messages, and the skeleton of State Machines are automatically generated. A basic Test Project is also included in order to invoke sequentially all interfaces according to the interaction flow defined, representing an emulator of a testing stimuli of all external subsystems.

3.5. *Tool support*

All theoretical aspects of the methodology are supported in an Integrated Development Environment (IDE) known as *Cervantes* Studio which runs as a set of editors and plugins over Enterprise Architect (EA), as a way to achieve a practical usage of the framework by industry. Model editors are performed according to the generation of UML Profiles of IFML and *Cervantes* metamodel, as it's shown in Figures 2 and 3 respectively.

All M2M and M2T transformations are developed as EA plugins where some of them are transparent to the user and others are performed by contextual menu. Figure 4 shows an overview of several model editors and the usage of M2T in practice.

3.6. *Quality assurance*

The model-based methodology proposed covers some problems regarding quality aspects in the Web Engineering domain. In a general view, the main benefit is to guarantee a practical traceability from the requirements to the final code, by using model transformations. In particular:

- From requirements, all user action is traced and automatically included in the interaction flow. This ensures that all user requirements are covered in the development.
- In case any user interaction is modeled and not aligned with requirements, the misunderstanding is detected and platform specific code is not generated until corrections are performed. The Figure 5 shows an example of an action included in the interaction model without any requirement, which is marked in red.
- From the user interaction flow, all actions are derived in the equivalent interfaces in a service-oriented architecture. This guarantees that all user actions are managed by the appropriate subsystem.
- According to the *Cervantes* Runtime Framework architecture, each message has the logic to be manage by its corresponding State Machine.

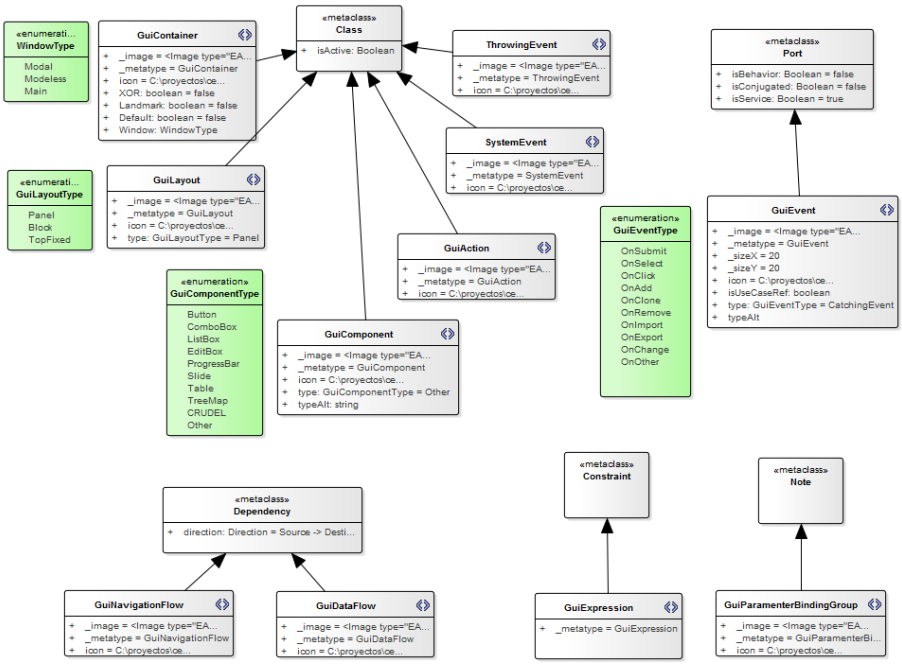

Fig. 2. IFML Profile

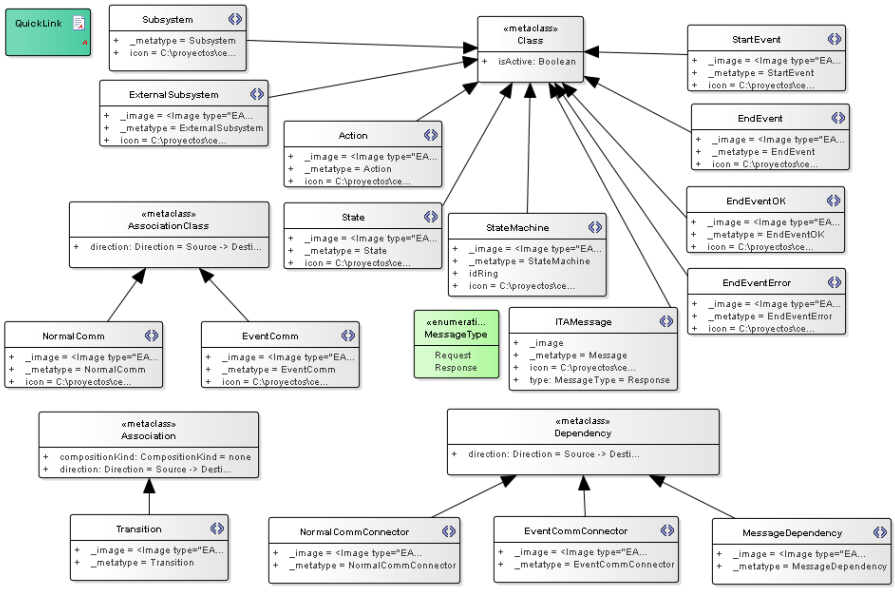

Fig. 3. *Cervantes* UML Profile

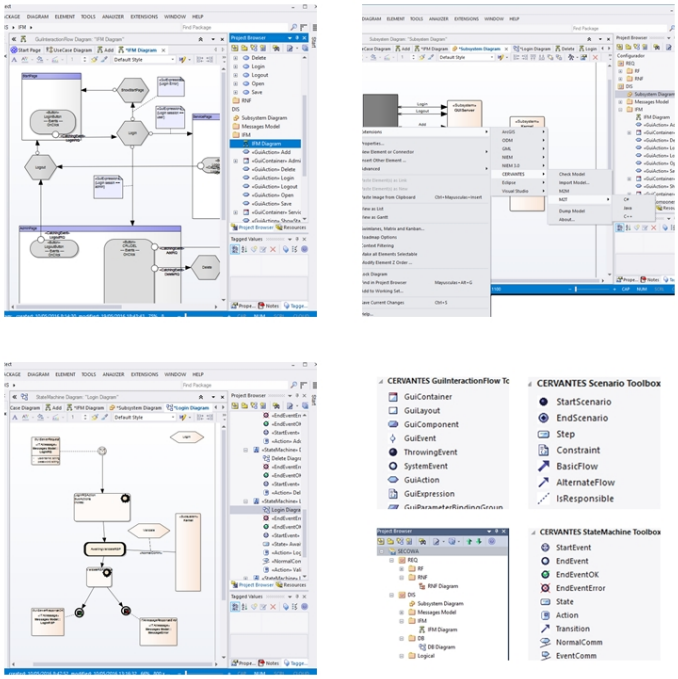

Fig. 4. *Cervantes* Studio IDE screenshots

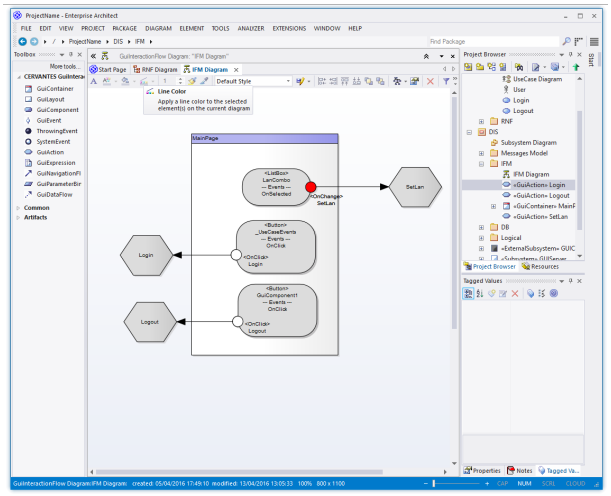

Fig. 5. Lack of traceability between a functional requirement in the Interaction Flow Model

- In case the development has to cover multiple specific platforms, the model is reused and the automated code generation achieved thought M2T transformation ensures the same common structure and behavior. This M2T transformation improves quality of source code generated because it's done once and same code is created in all projects.
- The automated generation of emulators of interfaces for testing reduces the risks associated with SoS integration where not all devices or hardware elements are available before deployment into production environment.
- There are some quality aspects covered not only by the methodological approach but also by the usage of the *Cervantes* Runtime Framework, as: i) all SoS have the same architecture (Subsystems that interchange Messages that are processed by State Machines) so that corrective or evolutionary maintenance may be performed even by people who did not participate in the development, and; ii) usage based logs and the usage based monitor allows to have a clear view of performance and usage at runtime, so that improvements in code or deployment can be done in those parts of the SoS that are being used in practice.

4. Industrial Validation

The methodological approach has been validated by a real-world project in order to build a Web-based tool [31] that simulates the resources required to perform a selective collection of waste (SECOWA). Although the first version of the system is not a pure SoS architecture, its planned evolution includes integration with smart city sensors, transport management systems and external services, that will be iteratively added in future increments. This way, the methodology was followed at a very early stage of the development, as a way to ensure that the approach is not only valid to SoS but also to Web-based developments.

This way, all models included Figure 1 have been created by using the IDE which are described as follows:

- Use cases Model (M1): functional requirements are detailed by use cases and a use case scenario in Figure 6. Two actors (Administrator and User) may perform CRUD operations and the configuration of a service which is described by scenarios.
- Non Functional Requirements Model (M2): described in natural language and included in the doc file auto-generated by M2T 1.
- Use Case Points Model (M3): size and effort estimation were done by using use case points and historical data, as it's described in Figure 6.
- Interaction Flow Model (M4): the user interaction is shown in Figure 7 by using the IFML editor included in the tool. This model allows the traceability between user actions from requirements to messages among subsystems. The model shows several actions, events and interaction components that are interrelated among them.
- Subsystems Model (M5): the SoS architecture shown in Figure 8 is detailed according to *Cervantes* metamodel. In this release there is a single subsystem that interacts with the external client stimuli.

- Messages Model (M6): all messages between client and web server are modeled in Figure 8 according to a PIM. They are the translation from user actions and interoperability between subsystems following a service-oriented approach.
- State Machines Model (M7): for each request a State Machine is modeled, 12 in total, according to *Cervantes* metamodel. An example of different State Machine models is shown in Figure 9. A State Machine starts with the reception of a Request Message, and by using Actions a Response is sent back to the origin. In case more SoS interactions are required, the model includes a step by step action following the Request-Response message pattern.
- Persistence Model (M8): a set of entities handled at memory are modeled to be converted into persistent data in runtime.
- Logic Model (M9): internal data structures as complex datatypes in Messages are modeled in UML Class diagrams and will be generated as PSM classes.
- Platform Specific Model (M10): all models are transformed into C#.net according to *Cervantes* Runtime Framework, including entities, State Machines and Graphical User Interface elements as it's shown in Figure 10. As a result of the case study a complete Web-based solution was developed by using the approach exposed in this paper. A screen shot of the final version is shown in Figure 11.

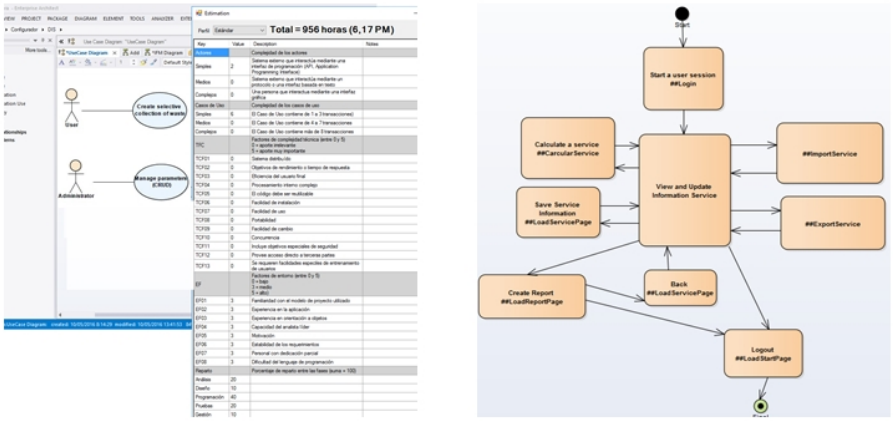

Fig. 6. Industrial Validation Use Cases Model, Use Case Points Model and Use Case Scenario

5. Preliminary Lessons Learned, Conclusions and Future Work

This paper presents a MDWE methodology to include Web-based interaction into SoS development. Taking existing *Cervantes* methodology for SoS development, the purpose of this study is to add Web-based interaction in order to cover all development phases, from the requirements to the code generation and maintenance, by using a MDE approach.

The proposal is based on ten models (Use cases Model, Non Functional Requirements Model, Use Case Points Model, Interaction Flow Model, Subsystems Model, Messages Model,

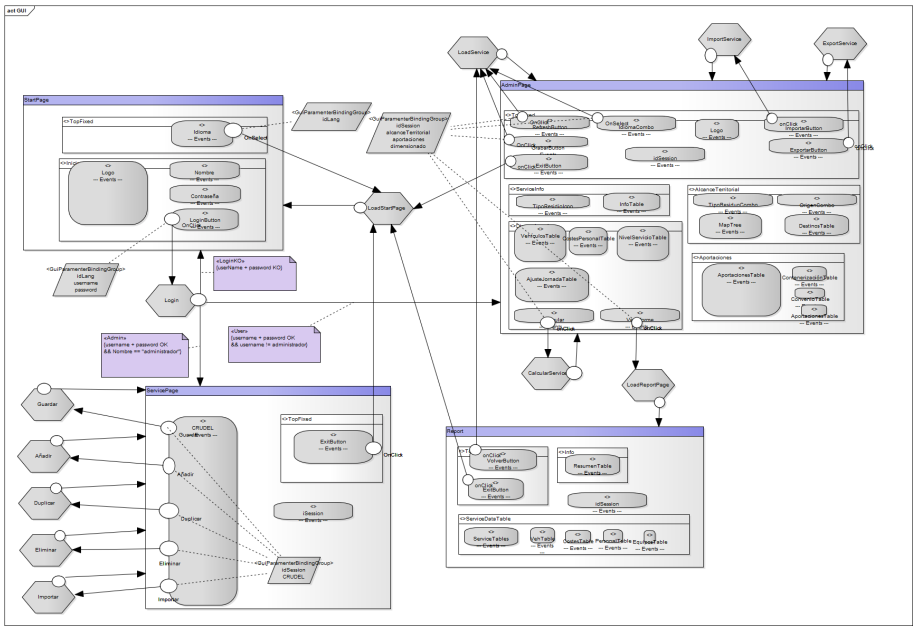

Fig. 7. Industrial Validation Interaction Flow Model

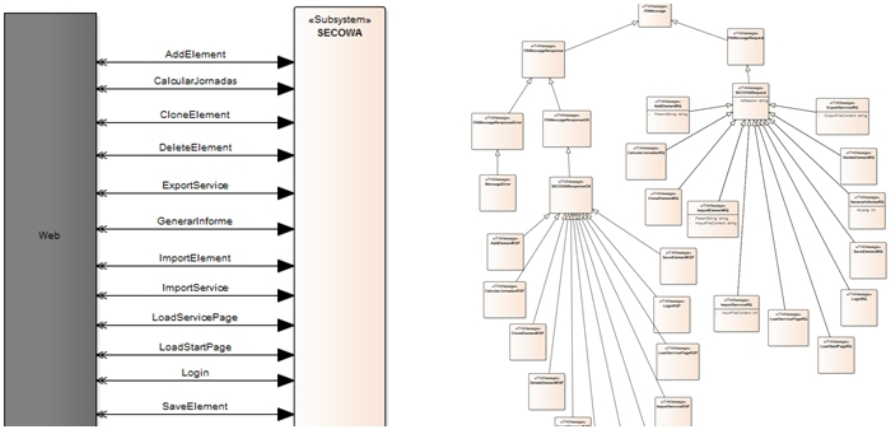

Fig. 8. Industrial Validation *Cervantes* Subsystems and Messages Model

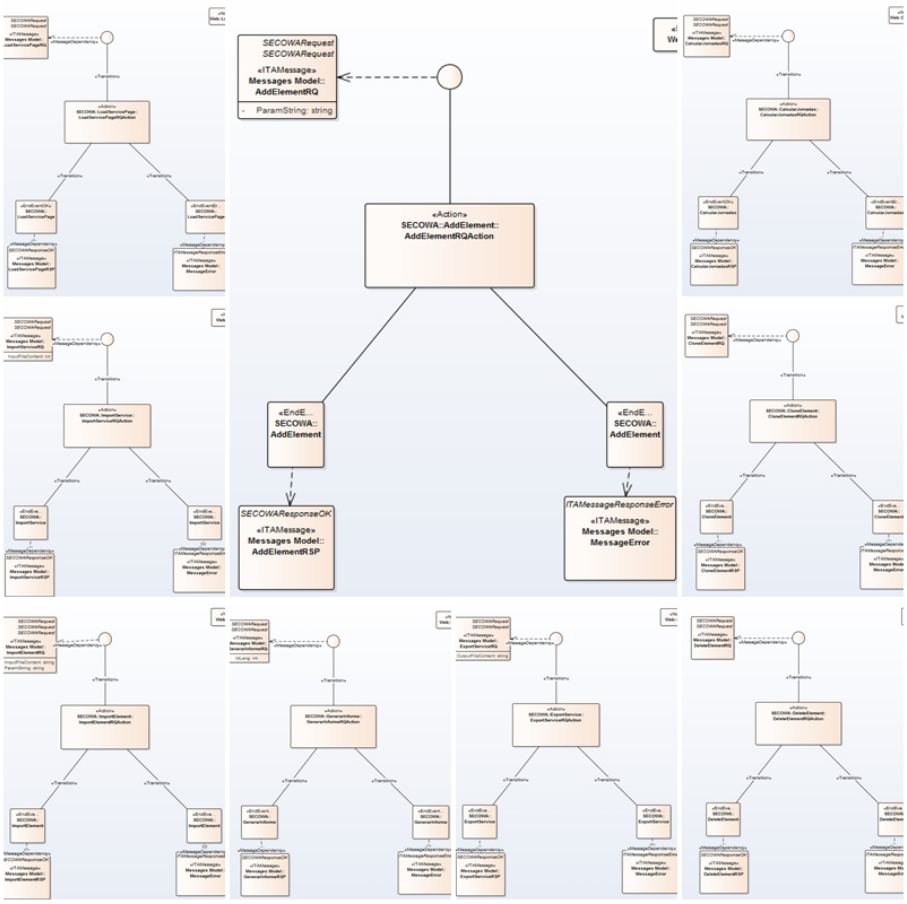

Fig. 9. Industrial Validation *Cervantes* State Machine Model

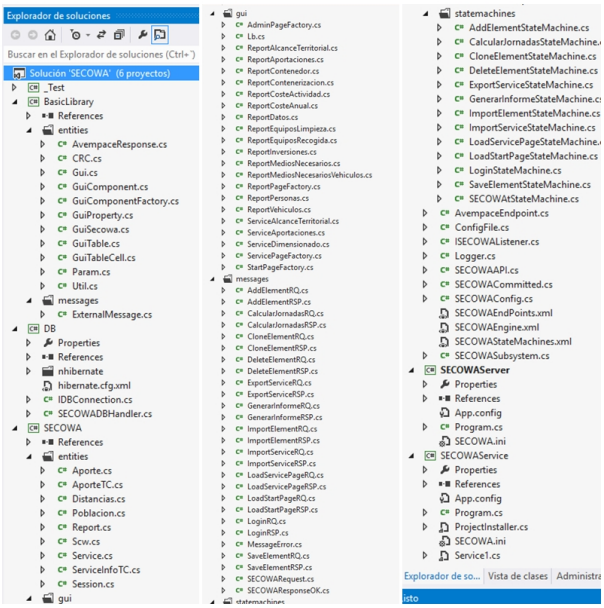

Fig. 10. Industrial Validation Platform Specific Model (C#.NET)

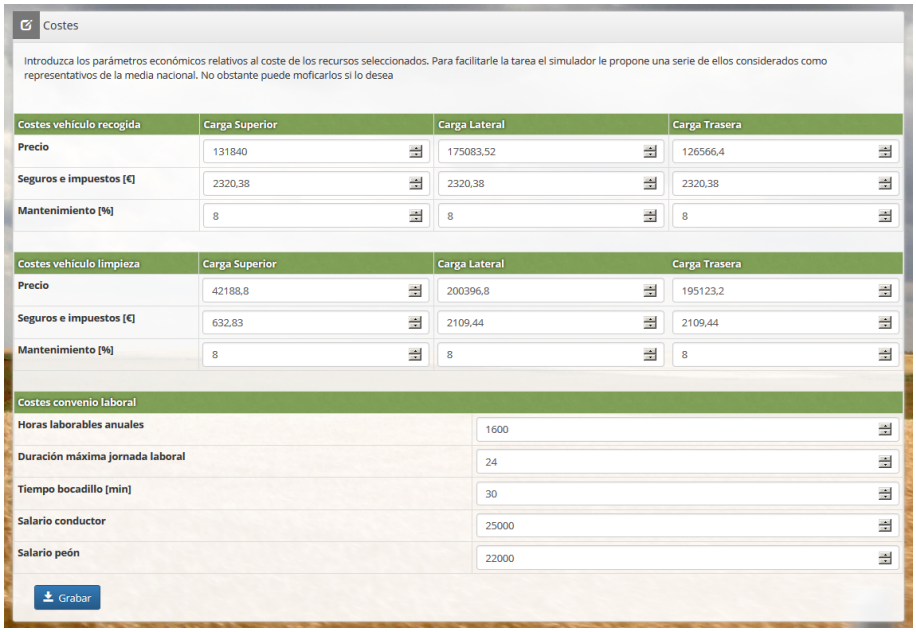

Fig. 11. Industrial Validation Screen shot

State Machines Model, Persistence Model, Logic Model and Platform Specific Model) which are created by using UML and two metamodels (IFML and *Cervantes*). It includes five M2M and two M2T. All theoretical elements are supported by a IDE as a way to be used in practice by industry.

Once the industrial validation project has finished we have compared a set of factors to a previous and similar development performed one year before. The purpose of this comparison is just to perform an early assessment of the potential benefits from the adoption of this MDWE methodology against our traditional development method. Figures are taken from the analytic systems (person-hours per activity in each project, number of defects from issue tracking systems, etc) when comparison may be performed, or by an estimation of values when both projects present differences to achieve a comparison. We are aware that conclusions cannot be quantitatively validated because there may be other aspects that affect results, but we believe it's interesting as an initial hypothesis for a future planned experimentation research. Main results of this analysis are shown in Table 1.

Table 1. Early assessment of potential benefits

Factor	Hypothesis	Value
Reduction to time to market	Expected reduction between 10%-20%	25%
Reduction to development effort	Expected reduction between 10%-20%	12%
Reduction to development cost	Expected reduction between 10%-20%	16%
Improvement of quality	Expected reduction of defects detected after release between 5%-10%	4%
Think vs Coding Ratio	Expected duplication of time invested on thinking (analysis, design) vs coding	Think time from 18% to 74% of person-hours
Reduction of cost on evolutionary maintenance	Expected reduction between 25%-50%	60%
Documentation effort	Expected reduction between 25%-50%	93%

As preliminary conclusions, this methodology allows us to focus more on the problem (understand requirements) and on the solution design than just coding. This way the total effort, time and cost of the development is reduced, specially when we add the maintenance (corrective and evolutionary) stage. The quality should be improved but we don't have enough evidences due to the nature of the projects that were not a pure SoS architecture so that the complexity and inter-dependency among subsystems were not present. Focus on PIM level is another benefit, so that PSM is achieved though automatic generators in multiple environments and programming languages. The usage of IFML allow us to extend the proposal to other user interaction technologies (not only Web-based but also mobile or desktop), but at an early stage we have identified a potential learning curve and a barrier in its adoption, specially as a way to interact to end user when capturing user interaction requirements (maybe mockups and other more visual languages would be better).

As future work we plan to run some experiments in order to obtain quantitative evidences of the benefits and the barriers for its adoption in practice. Another element to be extended

is testing, by the inclusion of a requirements metamodel and the usage of Unified Testing Profile (UTP) for test-case generation in all phases of the life cycle. We also plan to extend M2M by generating from subsystems model the corresponding IFML emulator so that we can obtain by M2T the basic runtime test-cases for all Messages.

Acknowledgements

We would thank MEDEN GROUP for its collaboration in the validation of the proposal. Additionally, this research has been supported by: the MeGUS project (TIN2013-46928-C3-3-R), the SoftPLM Network (TIN2015-71938-REDT) of the Spanish Ministry of Economy, Industry and Competitiveness as well as the Aragón Region European Social Fund.

References

1. M. Jamshidi, *System of systems engineering: innovations for the twenty-first century.* John Wiley & Sons, 2011, vol. 58.
2. T. Samad and T. Parisini, "Systems of systems," *The Impact of Control Technology*, pp. 175–183, 2011.
3. S. Murugesan, Y. Deshpande, S. Hansen, and A. Ginige, "Web engineering: A new discipline for development of web-based systems," in *Web Engineering.* Springer, 2001, pp. 3–13.
4. M. J. Escalona and G. Aragón, "Ndt. a model-driven approach for web requirements," *Software Engineering, IEEE Transactions on*, vol. 34, no. 3, pp. 377–390, 2008.
5. D. C. Schmidt, "Model-driven engineering," *COMPUTER-IEEE COMPUTER SOCIETY-*, vol. 39, no. 2, p. 25, 2006.
6. D. Cetinkaya and A. Verbraeck, "Metamodeling and model transformations in modeling and simulation," 2011.
7. N. Moreno, J. R. Romero, and A. Vallecillo, "An overview of model-driven web engineering and the mda," in *Web Engineering: Modelling and Implementing Web Applications.* Springer, 2008, pp. 353–382.
8. M. Brambilla, J. Cabot, and M. Wimmer, "Model-driven software engineering in practice," *Synthesis Lectures on Software Engineering*, vol. 1, no. 1, pp. 1–182, 2012.
9. L. García-Borgoñón, M. Barcelona, J. Calvo, I. Ramos, and M. J. Escalona, "Cervantes: A model-based approach for service-oriented systems development," 2014.
10. S. Ceri, M. Matera, F. Rizzo, and V. Demaldé, "Designing data-intensive web applications for content accessibility using web marts," *Communications of the ACM*, vol. 50, no. 4, pp. 55–61, 2007.
11. J. Conallen, *Building Web applications with UML.* Addison-Wesley Longman Publishing Co., Inc., 2002.
12. D. M. Groenewegen, Z. Hemel, L. C. Kats, and E. Visser, "Webdsl: a domain-specific language for dynamic web applications," in *Companion to the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications.* ACM, 2008, pp. 779–780.
13. M. Brambilla, A. Mauri, and E. Umuhoza, "Extending the interaction flow modeling language (ifml) for model driven development of mobile applications front end," in *Mobile Web Information Systems.* Springer, 2014, pp. 176–191.
14. S. Berti, F. Correani, G. Mori, F. Paternò, and C. Santoro, "Teresa: a transformation-based environment for designing and developing multi-device interfaces," in *CHI'04 extended abstracts on Human factors in computing systems.* ACM, 2004, pp. 793–794.
15. F. Paterno, C. Santoro, and L. D. Spano, "Maria: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 16, no. 4, p. 19, 2009.
16. G. Meixner, M. Seissler, and K. Breiner, "Model-driven useware engineering," in *Model-Driven*

Development of Advanced User Interfaces. Springer, 2011, pp. 1–26.
17. J. Vanderdonckt, "A mda-compliant environment for developing user interfaces of information systems," in *Advanced Information Systems Engineering*. Springer, 2005, pp. 16–31.
18. D. Raneburger, R. Popp, S. Kavaldjian, H. Kaindl, and J. Falb, *Optimized GUI generation for small screens*. Springer, 2011.
19. M. Brambilla and P. Fraternali, *Interaction Flow Modeling Language: Model-Driven UI Engineering of Web and Mobile Apps with IFML*. Morgan Kaufmann, 2014.
20. OMG, "Object management group, http://www.omg.org, accessed 2016-05-10."
21. OMG, "Interaction Flow Modeling Language (IFML), http://www.ifml.org, Accessed 2016-05-10."
22. C. Barry and M. Lang, "A survey of multimedia and web development techniques and methodology usage," 2001.
23. E. Mendes, "A systematic review of web engineering research," in *Empirical Software Engineering, 2005. 2005 International Symposium on*. IEEE, 2005, pp. 10–pp.
24. S. Mittal and B. P. Zeigler, "Devs unified process for web-centric development and testing of system of systems," DTIC Document, Tech. Rep., 2008.
25. A. Vallecillo, N. Koch, C. Cachero Castro, S. Comai, P. Fraternali, I. Garrigós Fernández, J. Gómez Ortega, G. Kappel, A. Knapp, M. Matera *et al.*, "Mdwenet: A practical approach to achieving interoperability of model-driven web engineering methods," 2007.
26. S. Meliá and J. Gomez, "The websa approach: applying model driven engineering to web applications," *Journal of Web Engineering*, vol. 5, no. 2, pp. 121–149, 2006.
27. M. Barcelona, L. García-Borgoñon, J. Calvo, and M. Escalona, "Cervantes: Un framework para el diseo y desarrollo de sistemas distribuidos," *Actas de las Jornadas de Ingeniera del Software y Bases de Datos (JISBD)*, 2014.
28. D. Rosenberg, "Use case driven object modeling with uml," 1999.
29. M. Dumas and A. H. Ter Hofstede, "Uml activity diagrams as a workflow specification language," in *UML 2001The Unified Modeling Language. Modeling Languages, Concepts, and Tools*. Springer, 2001, pp. 76–90.
30. R. K. Clemmons, "Project estimation with use case points," *The Journal of Defense Software Engineering*, pp. 18–22, 2006.
31. ITAINNOVA and MEDENGROUP, "Secowa tool," 2015. [Online]. Available: http://www.ita.es/secowa/