





PROTOTYPES AS STARTING POINT IN MDE: PROOF OF CONCEPT

Alejandro Sánchez-Villarín¹^a, Alejandro Santos-Montaña¹^b, Nora Koch²^c and
David Lizcano Casas³^d

¹ *Department of Computer Languages and Systems. University of Seville, Spain*
{alejandrosvillarín, alejandrosmontano}@us.es

² *IWT2 Group. University of Seville, Spain*
nora.koch@iwt2.org

³ *Madrid Open University, Spain*
david.lizcano@udima.es

Keywords: Software Prototyping, User-centered Design, Model-driven Engineering.

Abstract: Prototyping is a technique frequently used in the early stages of software development, but often prototypes are discarded or the provided information manually transferred into further steps of the software production. An open issue in industry and research is the automated reuse of valuable information included in these prototypes. We propose to build a tool that transforms prototypes into models. It is based on model-driven engineering concepts. In this paper, we present a first proof of concept for such a tool. For the validation, we built a mockup based prototype and a plugin for the transformations. The prototype and the generated analysis models were presented to potential end-users of software development companies. They confirmed the need for tool support for reusing information provided by prototypes and considered transforming prototypes into NDT suite models the most appropriate solution for them.

1 INTRODUCTION


Prototypes are nowadays very often part of the software development. They are used among others by requirement engineers to check if the solution planned will satisfy the expectations of the clients; in an ideal situation showing these prototypes also to end-users. There is a lot of literature that supports the fact that prototypes are an excellent tool to communicate with users (Escalona et al 2008) that companies tend to use them in an effective way (Rivero et al 2014) also for other purposes like validation and testing of concepts, processes, technologies, etc.


However, there are also some problems with using prototypes. They are usually developed together with


clients at the beginning of the life cycle, are validated and then generally fall into disuse or are discarded (Sánchez-Villarín et al 2019). For this reason, there is a tendency not to invest many resources in the so-called throw-away prototypes. Many times this decision results to be a failure because later in the development if the final product does not represent what the client expects, prototypes have to be rebuilt, spending more resources and delaying the project (Budde et al 1992).

In this context, the idea for the SocietySoft project was born. This project is based on the affirmation that dedicating resources to prototypes to improve communication with users is a great investment to guarantee the quality of the final product, and that it can be considered a profitable aspect if prototypes can

^a  <https://orcid.org/0000-0002-7740-5327>

^b  <https://orcid.org/0000-0002-0749-6411>

^c  <https://orcid.org/0000-0001-6404-0405>

^d  <https://orcid.org/0000-0001-7928-5237>

somehow be reused in the production lifetime cycle (Hehn, & Uebernickel 2018). The SocietySoft project works on, using model-guided engineering, to generate automatically analysis models like requirements and user interface models, from the prototypes. In that way gaining time in the first steps of the software development.

The project started with a systematic study of the literature (SLR) (Sanchez et al. 2019). The main conclusion of the SLR is a current lack on suitable approaches and tools that perform transformations while maintaining traceability, and subsequently obtain requirements or other elements of analysis, as well as focus on the user and the company.

The lack of work regarding tools that allow the reuse of prototypes documented in the SLR produces directly a new research question. Is the development of such a tool interesting? Within the scope of the SocietySoft project we developed a draft version of a prototype reuse tool and, currently, we are trying to prove its value in both the research and the enterprise environment.

Proof of concept and software validation in real environments are important and usually difficult to implement very often due to time to market issues. In the academic area, validations are made by students or researchers, where the conditions of an industrial environment are almost impossible to be simulated (Salman et al 2015). Thus, the functionality provided by the resulting software tools developed in the academic area is very often far away from the solutions required by the industry (Escalona et al 2007).

In order to evaluate our tool in an enterprise environment, a set of proof of concepts were executed. In this paper, we present the results of the first proof of concepts performed in two companies.

The paper is structured as follow. Section 2 presents an overview of the SocietySoft project. Section 3 presents the objectives of a proof of concepts and its implementation. Section 4 presents the validation performed by two companies. Section 5 presents the lessons learned in the proof of concepts and Section 6 gives an overview of some related work. Finally, Section 7 concludes with the next steps to be carried out within the scope of the project.

2 THE SOCIETYSOFT PROJECT

Information and communication technologies (ICT) have changed the way in which society at all levels manages its life. The SocietySoft project focuses on the ICT sector and more specifically aims to offer solutions that come from the academic world to strengthen software consulting companies through innovation mechanisms that allow them to compete in the new market niches opened by the Digital Society.

For companies to be competitive in the development of these types of products, it is necessary that they provide substantial improvements for their customers. The development of quality software reducing costs is a constant search and projection of software companies.

When it comes to ICT systems or solutions aimed at society, that is, systems aimed at providing solutions to groups of people, the situation is largely measured by the capacity that the software offers when it comes to interacting and being accessible and friendly to that society. However, classically, in software development for society, the focus is on technology, forgetting that it must be people-oriented software.

As society evolves towards a Digital Society, technologies and solutions must evolve with that society. When facing a software project, there are two different teams: the technical teams (engineers, programmers, etc.) and the functional teams (users, clients, etc.). Until now, most of the solutions developed by companies were oriented towards technical teams, to facilitate their work and help them “become aware” and “blend in” with the needs of the functional teams. But, due to the fact that these functional teams are increasingly digital, it is necessary for companies to begin to think of them as active and participatory teams that should have a say in their own developments. This of course is not an easy task and requires solutions that guide companies in the sector towards this new change of perspective.

The essential idea of the SocietySoft project is the development of a technology that arises from the need that we have detected both at a research and business level and which consists of the development of a tool for designing transferable prototypes.

For this, two core technologies are selected:

- Mockups for building navigable prototypes as the starting point. The project team has already used prototyping as an efficient technique to facilitate communication with customers and users (Huber et al. 2020) in previous projects (Torrecillas et al. 2015).
- Model-driven engineering (MDE) for the development process (Topcu et al 2016) (Abrahao et al 2017). The NDT-Suite tool was developed by members of the project team and successfully applied in several industrial projects. For the systematization of processes and tasks in software construction (Escalona et al 2008), NDT suite allows among others the automatic transformation of requirements models into analysis and testing models needed in further stages of the development life cycle of a software product.

In short, within the scope of the SocietySoft project, a tool prototype is developed, that provides an interface for creating navigable prototypes based on mockups. These screens will be stored in a structured way as an instance of a specific metamodel. This instance will be transformed, through model-guided engineering, into another model that will be the basis for the generation of a system requirements catalogue.

Figure 1 shows a high-level model of the architecture of the proposed tool solution. The functional team can develop its mockup using a tool (in the example used of the proof of concept, this tool was Microsoft PowerPoint, A in Figure). When the functional team creates the mockup, in fact, they are creating a model as an instance (B1) of an abstract interface metamodel (B). In the SocietySoft a set of Query-View-Transformations (QVT) was defined. These transformations allow generating a System Requirements Metamodel (C) from the abstract Interface Metamodel. Thus, the tool implements these transformations in a transformation engine (C1). With this environment, the model generated by the functional team (B1) can automatic generate a set of Systems Requirements Models (D1), that is, an instance of the System Requirements Metamodel (D). The result of these transformations is presented as analysis models (use cases, activity diagrams and

class diagrams) that can be interpreted by NDT-Suite, the tool of our methodology NDT (E).

Thus, the automatic transformation is supported in the plugin based on:

- Metamodels that are carried out according to the principles defined by the IFML standard (B and D).
- Transformations described in QVT (C) are implemented in a transformation engine (C1) that allows their automatic execution.

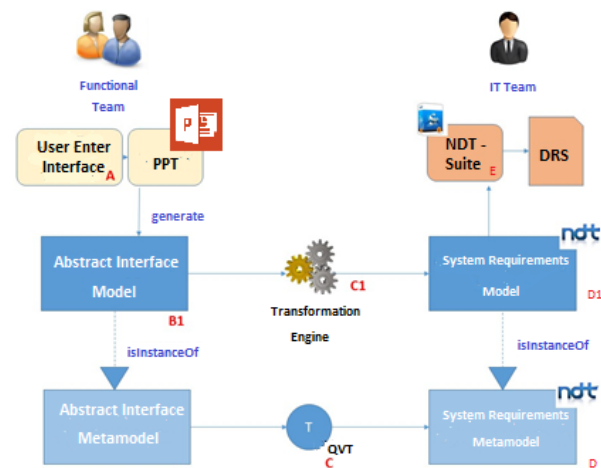


Figure 1: Abstract SocietySoft Tool Architecture

In the solution proposed above, the knowledge about the new software to be built is captured in these prototypes for being “reused” automatically to generate the system requirements models. Thus, the cost of defining and validating requirements is reduced, but also the generation of errors in the early stages of the development process can be minimized.

3 PROOF OF CONCEPT

The theoretical solution proposed by the SocietySoft project is promising and responds to concrete needs observed in enterprises by the team in previous projects (Escalona et al. 2007). A practical solution, i.e. the tool implemented to support the theoretical approach requires a set of decisions regarding the user interface, selection of technologies, plugins for existing tools, etc. Taking such decisions is not an easy task and implies significant risks for the acceptance of the final product by the end users.

Therefore, the validation of software products in the industrial environment where they will be used in the production software in the future is an important aspect of a software project. When the validation is performed as an initial step it is also known as proof of concept or feasibility study.

A proof of concept consists of the construction of a model or demo in order to evaluate the feasibility of a functional requirement, the development of a software component, the acquisition of a product, or a combination of all of them. However, the decision of performing a proof of concept depends on the complexity, the risks and the costs of the project and of the practicability of the proof of concept itself.

3.1 Aim and Preconditions

The proof of concept is frequently used in industry as a mechanism to value the capacity of a solution before to develop it. It consists in a short validation by a set of suitable stakeholders of an idea sketched in a demo or a very simplified prototype that The expectation is to obtain critical feedback on the viability of the project (Dingsøy et al. 2018).

The following aspects must be considered when the realisation of a proof of concept is decided:

- Risks and complexity of the project should be high. If the risk is low, the resources and the time to build the proof are usually not profitable.
- The proof should be evaluated according to clearly defined objectives. If possible including what to measure, how to measure it and how the evaluation should be carried out.
- A specific period has to be allocated in order to have enough time to obtain the required feedback and results.
- The proof must be an integral part of the project implementation strategy, having been duly planned and documented, including the time and effort necessary for its analysis, design, and implementation.

In our case, the proof of concept was necessary due to a high risk of user acceptance of the final software product of SocietySoft, which requires an investment of a large amount of time and resources.

We defined a proof of concept process adapting the pragmatic guide provided by Zuci Systems (Vivek, 2020) to the context of our project.

3.2. Implementation

The proof of concepts for the SocietySoft tool aimed to determinate the feasibility of a transformation tool for navigable prototypes designed in PowerPoint into user interface prototypes of the tool Enterprise Architect, as shown in Figure 1. Last ones should be appropriated to interact with the tool NDT-Suite. The NDT plugin supports a complete model-driven approach, especially requirements engineering in the development of software products, mainly web applications.

The proof of concepts follows a series of steps depicted in the activity diagram shown in Figure 2.

Our first step and the more time consuming was to build a draft version of the plugin for the transformations. It was developed and installed into Enterprise Architect in the NDT-Suite.

Another activity was the design of a very simple navigable prototype using the tool Microsoft PowerPoint. This prototype comprised different elements. The goal is to check whether the tool can transform a variety of elements like buttons and text boxes.

In addition to the navigable prototype and the plugin, we selected a first set of appropriate companies for the proof of concepts. This paper presents only the preliminary execution of the proof of concept. The reader is referred to the next section for a brief description of the two companies and the feedback obtained.

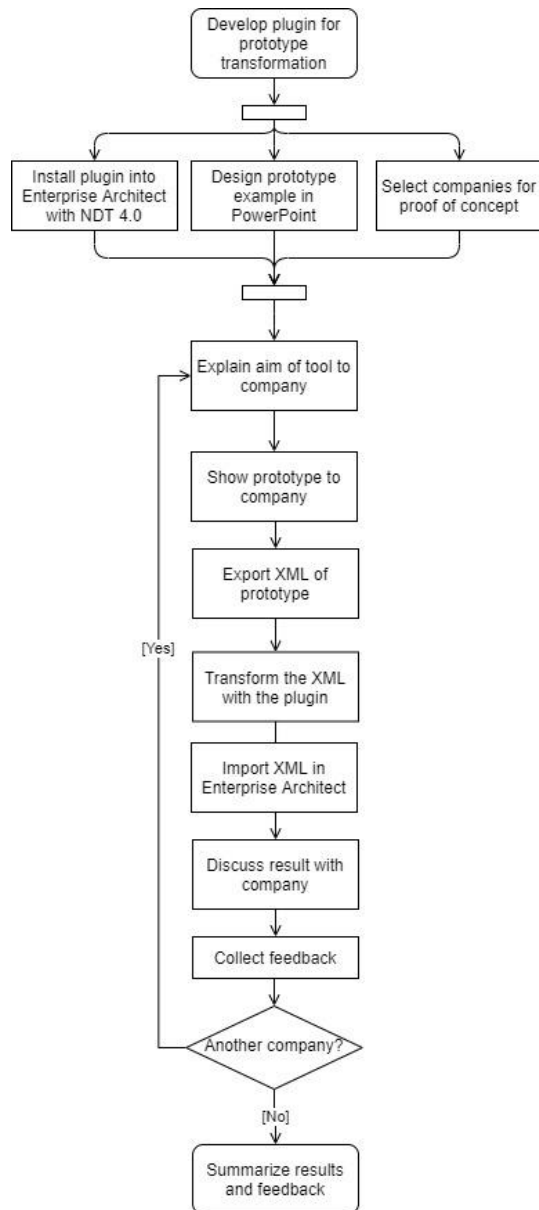


Figure 2: Process for the proof of concept

The simple navigable prototype consists of a couple of slides; two of them are described in detail in the following. The first one represents a form of terms and conditions, consisting of a text box with the terms and conditions, and two buttons, one to accept it and another to reject it (see Figure 3). The second one is a basic creation form including input boxes to write a date, an email, and a text as well as a button to send these data (see Figure 4).

The next step of the proof is to perform the transformation to obtain the navigable prototype in NDT. This transformation has been made using an automatic conversion between XMIs. PowerPoint has an XMI which includes information on the number of slides, the number of elements in each slide, their shapes, colours, position, etc.

The XMI is imported into Enterprise Architect and will be converted to specific XMI format needed for the navigable prototypes in NDT.

The code of the transformation has been written using the language Kotlin, which includes libraries to easily interact with PowerPoint, like Apache Poi.

Finally, the navigable prototype in Enterprise Architect is generated from the imported XMI by the plugin installed in the Enterprise Architect.

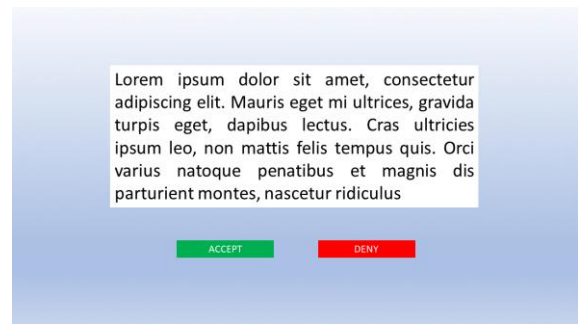


Figure 3: Prototype term and conditions in PowerPoint

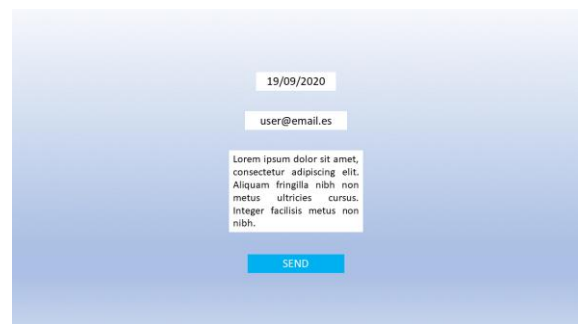


Figure 4: Prototype basic form in PowerPoint

Figure 5 and Figure 6 show the two pages of the NDT-Suite prototype that were generated automatically based on the navigable prototypes designed in Microsoft PowerPoint. The transformation produced an almost identical prototype in which the different interface elements like buttons, input fields and text boxes were

generated as depicted in Figure 5 and Figure 6. The internal representation corresponds to the XMI of the NDT metamodel.

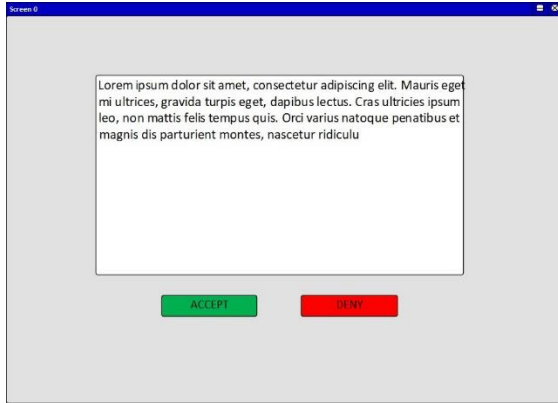


Figure 5: Prototype term and conditions in EA



Figure 6: Prototype basic form in EA

4 EXECUTION OF THE PROOF OF CONCEPTS

The last step of the proof of concept is the validation by the selected companies, in our case, Everis and G7Innovation. Their feedback will be useful for planning the next versions of our tool solution.

4.1 Everis

Everis is an NTT DATA Company, dedicated to consulting and outsourcing in all sectors. Everis has 24500 professionals across Europe, USA, and Latin America.

For the proof of concept, we arranged with the Everis team a meeting using the platform Microsoft Teams. In that meeting, we made a presentation in which the most important aspects of the proof of concepts were presented, as well as the tool developed for the proof of concept.

The main conclusions of their validation are:

- Interest of Everis in the developed tool as it is closely related to the development process used in the company.
- Check whether PowerPoint could be replaced by another alternative prototyping tool that offers identification of model structures.
- Align the export to the XMI versions 1.1 and 2.1, both used in Everis. This will avoid future problems of compatibility.
- A communication channel will be maintaining between Everis and the SocietySoft team to update them with new versions of the tool.
- Possibility of physically attending the Everis Zaragoza offices to make a presentation of the product.

4.2 G7Innovation

G7Innovation is a technology-based company, based in Seville (Spain) with a focus in the development of ICT solutions applied to clinical and health management and characterised by a high level of innovation. The multidisciplinary group of computer engineers, healthcare personnel and experts in business management of G7Innovation, has extensive experience in the health sector.

For the validation of our approach, the same presentation prepared for Everis was used in the Skype meeting with G7Innovation showing the most relevant and essential aspects for the proof of concepts.

In the meeting all the points of the presentation and the conclusions obtained with Everis were raised. As a result, G7Innovation agreed in almost all points with Everis, stressing the need to search for other software instead of PowerPoint for prototyping. They explained how they currently perform prototyping, mentioning that the tool for navigable prototypes we

propose in the proof of concept would be highly beneficial for its company.

5 LESSONS LEARNED

Thanks to the completion of these proof of concepts, the junior members of the project team could better understand the concept of proof of concept. They could appreciate its relevance for a project that aims to develop a tool or extend the functionality of a tool in order to analyse its feasibility and impact.

In addition, the possibility of validating the tool prototype with real companies allowed us to better understand the acceptance of such a tool and the improvements required by real potential users.

Specifically, we received a suggestion for the use of an online tool for designing mockups, which is more appropriate for the companies. We will analyse these prototyping alternatives in detail and report accordingly in future works.

6 RELATED WORK

Many works report on the need for a proof of concept before a software development project is started. Some of them report on how they performed such a proof of concept and on who was involved in such proof. Our focus was on efforts using storyboards (Rasheed et al, 2019), dashboards (Croon et al, 2015) or mockups (Rivero et Conte, 2013; Rivero et al. 2014 & 2019) due to similarities to our project.

For example, Croon et al (2015) proved the idea of a dashboard based tool to identify patients in need of follow-up. They used a rapid prototyping methodology. The prototype was evaluated by 12 students and 15 general practitioners.

The technique WebDUE (Web Design Usability Evaluation) and the Mockup DUE tool proposed by Rivero & Conte (2013) to evaluate the usability of web applications was instead only evaluated by a group of students in the academic field. However, their work is interesting from the point of view of the use of mockups for inspection and annotation in the web design.

Mockups is as well the technique selected as a starting point by Rivero et al. (2011, 2014) for the software development process. The Mockup Model-driven (MockupDD) approach generates user interface models based on model transformations and metamodels. Their aim – similar to ours – is to avoid the loss of information included in the mockups.

Rasheed et al. (2019) propose an interesting approach for merging the concepts of storyboarding and metamodel. Their aim is to automate the creation of evolutionary prototypes. An online booking application is used as a case study to validate their approach showing that their metamodel is capable of generating both simple as well as complex storyboards. A proof of concept in an industrial environment is neither reported nor planned neither in Rasheed et al (2019) nor Rivero et al (2011, 2014).

7 CONCLUSIONS AND FUTURE WORKS

This paper presents a preliminary work in the context of the SocietySoft project. SocietySoft proposes that if a suitable tool for reusing navigable prototypes developed in the early stages of a software development project could help companies to make a bigger investment in their development and validation.

In order to evaluate the suitability of a tool based on the model-driven paradigm and its acceptance in the industrial environment, a proof of concept was executed in two companies. This paper describes how the proofs of concept were planned, carried out and the results obtained. These results indicate that there is indeed a need for tool support for reusing information provided by mockups and that transforming them into models of NDT suite is the right kind of tool.

We concluded that the proof of concept was a success, regarding the interest and acceptance in real environments of the tool for transforming prototypes into models. Our investment of time and resources in building this very simple prototype and the draft version of the plugin were affordable. In addition, the companies guaranteed future support, meetings for the control of the tool, and the test in the internal

development area of the companies once a stable version is available.

However, this proof of concept is preliminary. An important future work is to try to improve it. Obviously, feedback of two companies may not be enough. We plan further proof of concepts in other companies to evaluate the suitability of our tool.

A next step then will be the construction of the complete SocietySoft tool comprising the prototype design and the plugin for importing the prototype and transforming it into models. In this sense, the next concrete steps would be the development and testing of the transformations to obtain the elements of analysis, such as system requirements. Therefore, first, we will check and eventually change the prototyping tool from PowerPoint to an online tool with community support such as draw.io. This tool has to fulfil our requirements of exporting in XML format in a way that allows us to obtain appropriate transformations for our NDT plugin. In addition, we have to refactor the code of the plugin developed to C# in order to obtain greater efficiency.

For our beta version of the tool, we plan an experimentation and validation following the best practices of Juristo & Moreno (2001). They propose the following phases for the lifecycle of the validation: goal definition, design of the experiment, execution of the experiment, and analysis of the results.

With all this, we will be able to get a complete tool, which allows maintaining traceability between the prototype and the model elements, with the cost reduction and time savings that this implies.

ACKNOWLEDGEMENTS

This paper was supported by the project “SocietySoft-Transfer of tools, policies, and principles for creating quality software for the digital society” (AT17 5904 USE) of the Andalusian Regional Government’s Department of Economy, Knowledge, Business, and Universities (Spain) and NICO project (PID2019-105455GB-C31) of the Ministry of Science, Innovation and University of the Spanish Government.

Furthermore, the authors would also like to thank the anonymous reviewers for their valuable

comments and suggestions to improve the quality of the paper.

REFERENCES

- Abrahão, S., Bordeleau, F., Cheng, B., Kokaly, S., Paige, R. F., Störrle, H., Whittle, J. (2017). User Experience for Model-Driven Engineering: Challenges and Future Directions. In 2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS), 229-236. IEEE.
- Alfonso Hoyos, J.P., Restrepo-Calle, F. (2018) Fast prototyping of web-based information systems using a restricted natural language specification. *Communications in Computer and Information Science*, 866, 183-207.
- Anjum R., Azam, F., Anwar, M.W., Amjad, A. (2019). A meta-model to automatically generate evolutionary prototypes from software requirements. *ACM International Conference Proceeding Series*, 131-136.
- Budde, R., Kautz, K., Kuhlenskamp, K., & Züllighoven, H. (1992). What is prototyping?. *Information Technology & People*.
- De Croon, R., Klerkx, J., and Duval, E. (2015). Design and Evaluation of an Interactive Proof-of-Concept Dashboard for General Practitioners, 2015 International Conference on Healthcare Informatics, Dallas, TX, 150-159, doi: 10.1109/ICHI.2015.25.
- Dingsøyr, T., Moe, N. B., Fægri, T. E., Seim, E. A. (2018). Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation. *Empirical Software Engineering*, 23(1), 490-520.
- Escalona, M. J., Gutierrez, J. J., Villadiego, D., León, A., & Torres, J. (2007). Practical experiences in web engineering. In *Advances in Information Systems Development*, 421-433. Springer, Boston, MA.
- Escalona, M. J., Gutierrez, J. J., Villadiego, D., León, A., Torres, J. (2007) Practical Experiences in Web Engineering, 421–433. Springer US.
- Escalona, M.J., Aragón, G. (2008). NDT: A Model-Driven Approach for Web Requirements. *IEEE Trans. on Software Engineering* 34 (3) 377-390
- Escalona, M.J., Gutierrez, J.J., Mejías, M., Aragón, G., Ramos, I., Torres, J., Domínguez, F.J. (2011). An overview on test generation from functional requirements. *Journal of Systems and Software*, 84 (8), 1379-1393.
- García-García, J.A., Alba Ortega, M., García-Borgoñón, L., Escalona, M.J. (2012). NDT-Suite: A model-based

- suite for the application of NDT. 7387. 469-472. 10.1007/978-3-642-31753-8_46.
- Hehn, J., Uebernickel, F. (2018, August). The use of design thinking for requirements engineering an ongoing case study in the field of innovative software-intensive systems. In *2018 IEEE 26th International Requirements Engineering Conference (RE)* (pp. 400-405). IEEE.
- Huber, T. L., Winkler, M. A., Dibbern, J., Brown, C. V. (2020). The use of prototypes to bridge knowledge boundaries in agile software development. *Information systems journal*, 30(2), 270-294.
- IFML. Interaction Flow Modeling Language. <http://www.ifml.org> (last visit 10.09.2020).
- Iyengar, P., Wessels, S., Noyer, A., Pulvermueller, E., Westerkamp, C. (2016). A novel approach towards model-driven reliability analysis of Simulink models. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*.
- Juristo, N., Moreno, A.M. Basics of software engineering experimentation. Kluwer, 2001.
- Kamalrudin, M., Grundy, J. (2011) Generating essential user interface prototypes to validate requirements. 26th IEEE/ACM International Conference on Automated Software Engineering, ASE 2011, Proceedings. 6100126, 564-567.
- McMillan, C., Hariri, N., Poshyvanyk, D., Cleland-Huang, J., and Mobasher, B. (2012). Recommending source code for use in rapid software prototypes. In *Proceedings of the 34th International Conference on Software Engineering*, pages 848–858. IEEE Press.
- QVT. MOF QUERY/VIEW/TRANSFORMATION SPECIFICATION VERSION 1.3. <http://www.omg.org/spec/QVT/About-QVT/> (last visit 10.09.2020)
- Rasheed, Y., Azam, F., Anwar, M.W., Tufail, H. (2019). A model-driven approach for creating storyboards of web based user interfaces. In: *Proceedings of the 2019 7th International Conference on Computer and Communications Management*, 169–173.
- Rivero, J. M., Grigera, J., Rossi, G., Luna, E. R., Montero, F., Gaedke, M. (2014). Mockup-driven development: providing agile support for model-driven web engineering. *Information and Software Technology*, 56(6), 670-687.
- Rivero J.M., Grigera J., Rossi G., Robles Luna E., Koch N. (2012). Towards Agile Model-Driven Web Engineering. In Nurcan S. (Eds) *IS Olympics: Information Systems in a Diverse World*. CAiSE 2011. LNCS, 107. Springer, Berlin, Heidelberg.
- Rivero, L., Conte, T. (2013). Using an empirical study to evaluate the feasibility of a new usability inspection technique for paper based prototypes of web applications. *Journal Software Engineering Research and Development* 1, 2.
- Salman, I., Tosun, A., Juristo, N. (2015). Are students representatives of professionals in software engineering experiments? In *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015*, Florence, Italy, Volume 1, 666–676.
- Sánchez-Villarín, A., Santos-Montaño, A., Enríquez, J.G. (2019). Automatic Reuse of Prototypes in Software Engineering: A Survey of Available Tools. *15th Int. Conference on Web Information Systems and Technologies (WEBIST 2019)*, 144-150. 10.5220/0008352901440150.
- Topçu, O., Durak, U., Oğuztüzün, H., & Yılmaz, L. (2016). Model Driven Engineering. In *Distributed Simulation*, 23-38. Springer International Publishing.
- Torrecilla-Salinas, C. J., Sedeño, J., Escalona, M. J., Mejías, M. (2015). Estimating, planning and managing Agile Web development projects under a value-based perspective. *Information and Software Technology*, 61, 124-144.
- Vivek, J. (2020). The Ultimate Guide to Proof of Concept for Software Development, Zuci System. Medium. <https://medium.com/@zucisystems/the-ultimate-guide-to-proof-of-concept-for-software-development-2020-859aec3a82d3> (last visit 5.10.2020).