MockupDD: Facilitating Agile Support for Model-Driven Web Engineering

José Matías Rivero[1,2] and Gustavo Rossi[1,2]

[1] LIFIA, Facultad de Informática, UNLP, La Plata, Argentina
{mrivero,gustavo}@lifia.info.unlp.edu.ar
[2] Conicet

Abstract. Model-Driven Web Engineering methodologies provide a more productive way of building Web Applications using high-level models and generating final implementations from them. However, they follow a waterfall-like development process, forcing to specify a different set of models sequentially to obtain a first runnable prototype of the Web Application. On the other hand, agile methodologies pursue an iterative process based on the delivery of application prototypes in short periods of time using manual coding, which results less productive and more error-prone in comparison to model-based approaches. In this work we propose a hybrid agile and Model-Driven approach called MockupDD that intends to blend the best of MDWE and agile development processes.

1 History

Model-Driven Web Engineering (MDWE) approaches like WebML [1], UWE [2] or OOHDM [3] have become mature solutions for developing Web Applications. These methodologies intend to apply Model-Driven Development (MDD) concepts to the Web Applications field, capturing high-level concepts relative to Web development (domain objects, pages, hyperlinks, rich interaction functionality, etc.) into models, letting developers automatically generate runnable applications from them. While standard MDWE processes improve productivity by describing Web Applications with such languages, they tend to leave User Interface (UI) aspects to the end of the development cycle [4]. The classical MDWE process starts building a content model describing the different types of objects that will be managed by the application and how they relate. Then, a hypertext model specifying the navigational structure of the Web Application and how the aforementioned objects will be shown and manipulated is defined. Finally, a presentation model is constructed detailing how the pages structured in the hypertext model should look in detail and refining interaction aspects.

On a different track, agile methodologies have shown a quick and massive adoption over the last years. These methodologies, instead of following a set of linear steps or high-level languages to define Web Applications, rely on direct coding to generate deliverable versions of the product being built to promote early and constant interaction with customers or end-users. The purpose of this strategy is to assert that the software being built complies with end-users requirements by constantly delivering

Q.Z. Sheng and J. Kjeldskov (Eds.): ICWE 2013 Workshops, LNCS 8295, pp. 325–329, 2013.

prototypes developed in short periods of time. Agile approaches argue that software specifications must emerge naturally, enhancing former prototypes along the development until the final application is obtained, and the same applies for good practices and patterns that help the developer teams day by day to build high-quality software products[1].

Both approaches show advantages and weak points. While MDD and agile approaches are usually seen as contradictory or incompatible, in this work, we propose to combine both in order to maximize their pros and reduce their disadvantages as much as possible. In order accomplish this task, we chose to use user interface prototypes (usually referred as *mockups*) as a starting requirements artifacts that end-users or customers can understand [5] and then introduce them as valuable model specifications in the process. Since user interface modeling and prototyping represents a very studied field (for instance, considering Canonical Abstract Prototypes [6] or UsiXML [7]) and also have been applied into agile processes [8], we argue that integrating them in a novel MDD process will provide a better requirements understanding and a more quick a less error-prone model-based development process.

2 Problem

While MDWE methodologies facilitate software specification portability, abstraction and productivity, they fail in providing *agile* interaction with customers because concrete software results are obtained too late, since they follow a waterfall-like modeling process with linearly structured steps. Modelers must define a set of models sequentially after reaching a final prototype of the application that can be shown to end-users or customers. Moreover, detailed requirements that cannot be fulfilled natively by the MDWE language concepts have to be coded manually (which leads to breaking the MDD abstraction and its inherent advantages) or force developers to extend the MDWE language and code generators, which implies additional time overheads to obtain a running prototype of the application as quick as possible.

On the other hand, agile methodologies are heavily based on implementation through direct coding. Thus, they have more freedom to code detailed business-related requirements ad-hoc and also they can adapt or *mock* they implementations more easily to speedily show running versions of the application to end-users or customers and assess that requirements have been correctly captured and implemented. However, the use of direct coding implies more proneness to human errors, forces developers to manually maintain an uniform coding style and also implies writing again and again repetitive and common functionalities (like, for instance, classical CRUD operations) that can be easily generated automatically or semi-automatically as in MDWE.

[1] Principles behind the Agile Manifesto – `http://agilemanifesto.org/principles.html`

3 Solution

We propose an hybrid model-based agile methodology – called Mockup-Driven Development (*MockupDD*) – aiming to extract the best of both approaches, i.e. a process driven by the active participation of users and customers, and a classical approach following the phases of analysis, design and implementation assisted with the use of models in all stages. Our approach starts by the requirement analysis defining a set of user interface mockups to agree upon the application's functionality. After being built with active end-user or customer participation, mockups are translated to an abstract User Interface model that can be directly derived to specific MDWE presentation models or technology-dependent UI prototypes [9].

After this stage, we propose to enrich mockups by a *tagging* process. In this step, mockups (now linked to presentation models) are enriched with navigation, data, data manipulation, business logic and interaction specs. Again, end-users or customers can actively participate in the most of the process (excluding technical specifications) since they understand the underlying concepts in the foundational models (mockups): widgets, pages, etc. This also facilitates a better traceability of the requirements being modeled, since they are associated to specs that were defined directly or with high participation and assessment of end-users or customers.

Following the MDD principles, MockupDD relies on artifacts generation from models – in this case, UI models expressed by mockups plus specs applied over them. Thus, it provides both code and models generation (for more popular MDWE approaches) as the final step of the process. In addition, after a tagging session, a functional prototypical version of the application can be run using a *demo sandbox* tooling provided by the methodology, without requiring any compilation or deployment. The application of the MockupDD approach within the well-known Scrum agile process is depicted in Fig. 1.

4 Current Approaches and Related Work

One of the key fields to which MockupDD is related to is UI modeling and prototyping. This is an extensively studied field. Currently, a lot of UI mocking tools has been defined like Balsamiq, Pencil, among dozens of many others including DENIM [10], in which several levels of UI sketching are provided in a *top-down* incremental way. Also, well-known UI modeling proposals like UsiXML exist [7], in companion of extensive tool support. While the former are oriented to build quick-and-dirty and disposable prototypes for requirements gathering purposes, the latter provide a modeling language and environment to formally define user interfaces and generate running implementation from them. However, MockupDD does not intend to provide *yet another* UI prototyping or modeling environment, but to use enriched mockups (that are good for requirements gathering and facilitate customer-developer interaction) as a foundation to *generate* models for existing MDWE and Model-Based User Interface (MBUI) approaches like UsiXML.

From the modeling point of view, user interfaces were used in numerous approaches as a basis for requirements or software specifications. For instance, Panach et al. propose gathering interaction requirements from UI sketches and then creating structural task models from them [11]. In [5] and [12], UI models are used to specify the structure and dynamics of the interface using formal or informal storyboard-oriented specifications. The Interaction Flow Modeling Langauge[2] (IFML) recently approved as a standard by the OMG, uses visual models (more technical-oriented than mockups) to assemble UI descriptions and specify detailed actions over them.

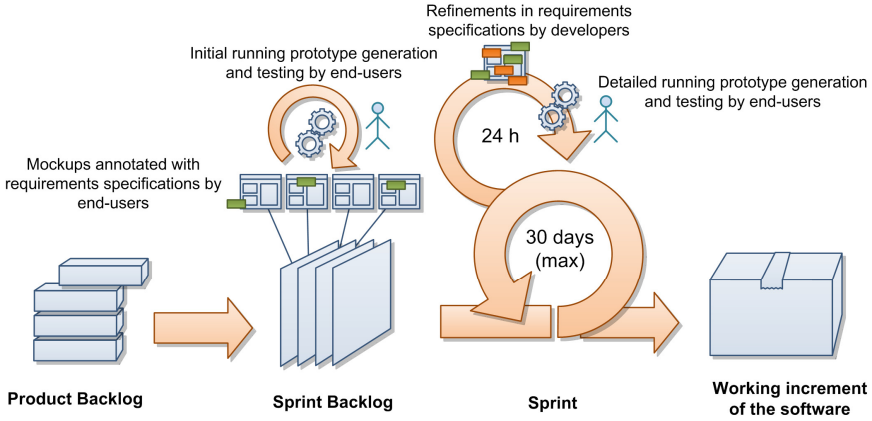

Fig. 1. MockupDD Scrum process adaptation

5 Research Methodology

We already built several tools to test the methodology. First, we implemented a Mockup Processing Engine [4] that is able to take mockups built with traditional mockup tools and abstract them into a common UI model to be further used in the modeling process. We also built a *tagging environment* for such processed mockups.

Using these tools, we already conducted a quantitative experiment in which we compared MockupDD performance vs. traditional modeling using WebML in terms of completion, speed and model quality. We are currently conducting a second experiment in which we are comparing detailed modeling using a refined MockupDD annotation set oriented to data models vs. data modeling using mainstream tools. As a result of the implementations and experiments, we expect to show that MockupDD is able to improve the modeling process both quantitative and qualitative in comparison to traditional existing modeling and agile *pure* code-based methods.

[2] IFML: The Interaction Flow Modeling Language - `http://www.ifml.org/`

6 Agenda and Further Work

After obtaining the final results of our experiments, we are aiming to improve MockupDD process to make it more complete and friendlier both for developers and end-users or customers. We are planning to extend the set of specifications that the methodology currently provides to cover other well-known Web Application fields like data validation, RIA behavior, etc.

Since MockupDD is in essence an MDD methodology, the problem of coping with detailed requirements is an important issue to tackle. Because it is founded on existing artifacts (User Interface mockups), we are planning to provide custom APIs to extend aspects modeled over the UI using direct coding in a non-intrusive fashion, considering also code reuse among different mockups and specifications.

References

1. Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): a modeling language for designing Web sites. Computer Networks 33, 137–157 (2000)
2. Koch, N., Knapp, A., Zhang, G., Baumeister, H.: UML-Based Web Engineering. Springer, London (2008)
3. Rossi, G., Pastor, O., Schwabe, D., Olsina, L.: Modeling and Implementing Web Applications using OOHDM. In: Rossi, G., Pastor, O., Schwabe, D., Olsina, L. (eds.) Web Engineering: Modelling and Implementing Web Applications, pp. 109–155. Springer, London (2008)
4. Rivero, J.M., Rossi, G., Grigera, J., Luna, E.R., Navarro, A.: From interface mockups to web application models. In: Bouguettaya, A., Hauswirth, M., Liu, L. (eds.) WISE 2011. LNCS, vol. 6997, pp. 257–264. Springer, Heidelberg (2011)
5. Mukasa, K.S., Kaindl, H.: An Integration of Requirements and User Interface Specifications. In: 6th IEEE International Requirements Engineering Conference, pp. 327–328. IEEE Computer Society, Barcelona (2008)
6. Constantine, L.L.: Canonical Abstract Prototypes for Abstract Visual and Interaction Design. In: Jorge, J.A., Jardim Nunes, N., Falcão e Cunha, J. (eds.) DSV-IS 2003. LNCS, vol. 2844, pp. 1–15. Springer, Heidelberg (2003)
7. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., López-Jaquero, V.: USIXML: A Language Supporting Multi-path Development of User Interfaces. In: Bastide, R., Palanque, P., Roth, J. (eds.) EHCI-DSVIS 2004. LNCS, vol. 3425, pp. 200–220. Springer, Heidelberg (2005)
8. Ferreira, J., Noble, J., Biddle, R.: Agile Development Iterations and UI Design. In: AGILE 2007 Conference, pp. 50–58. IEEE Computer Society, Washington, DC (2007)
9. Rivero, J.M., Rossi, G., Grigera, J., Burella, J., Luna, E.R., Gordillo, S.: From mockups to user interface models: An extensible model driven approach. In: Daniel, F., Facca, F.M. (eds.) ICWE 2010. LNCS, vol. 6385, pp. 13–24. Springer, Heidelberg (2010)
10. Lin, J., Newman, M.W., Hong, J.I., Landay, J.A.: DENIM: finding a tighter fit between tools and practice for Web site design, pp. 510–517 (2000)
11. Panach, J.I., España, S., Pederiva, I., Pastor, O.: Capturing Interaction Requirements in a Model Transformation Technology Based on MDA. J. UCS 14, 1480–1495 (2008)
12. Luna, E.R., Garrigós, I., Grigera, J., Winckler, M.: Capture and Evolution of Web Requirements Using WebSpec. In: Benatallah, B., Casati, F., Kappel, G., Rossi, G. (eds.) ICWE 2010. LNCS, vol. 6189, pp. 173–188. Springer, Heidelberg (2010)