M. Riebisch, M. Tropmann-Frick (Hrsg.): Modellierung 2022,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2022 145

Supporting Requirements Engineering and Development with Event Modeling - an Overview

Sebastian Copei [1], Clemens Emme [2], Maximilian Freiherr von Künßberg [3], Adam Malik [4], Natascha Nolte [5], Ulrich Norbisrath [6], Albert Zündorf [7]

Abstract: Workflow based systems are frequently modeled using BPMN diagrams for requirements engineering and UML diagrams for analysis and design. While such diagrams are great input for the software development, non-ICT people like domain experts, customers, end users or business people usually do not understand their content. Therefore, non-ICT people are frequently not able to contribute to these diagrams nor able to identify missing or incorrect parts. This paper proposes the very informal event storming approach as a means to involve users into the requirements engineering activities. We then refine the event storming results via Event Modeling by adding elaborated user interface scenarios. These event models then serve as input for integrated data modeling steps.

Keywords: Requirements Engineering; Design; Event Storming; Wireframes; Data Modeling

1 Introduction

This paper focuses on workflow based systems like Enterprise Resource Planning (ERP), management, or administrative systems. Generally, a software project consists of requirements engineering phases, design phases, development phases, and deployment and operation phases. For workflow based systems, there exists a large body of knowledge, methods, languages, and tools for requirements engineering especially in the area of Business Process Models and Notations (BPMN) [OPM11, Sc00]. Similarly, there is a large body of knowledge, methods, languages, and tools for software architecture, design, and modeling – notably the Unified Modeling Notation [Bo96], the Unified Modeling Process [JBR99], and Design Pattern [Ga95]. In spite of these achievements, there are still a lot of open problems to be solved:

Requirements engineering often uses formal notations like BPMN diagrams to specify workflows precisely. Such notations are good for the communication towards the development

[1] Software Engineering, Kassel University, Germany, sco@uni-kassel.de
[2] Software Engineering, Kassel University, Germany, clemens@uni-kassel.de
[3] Software Engineering, Kassel University, Germany, maximilian-kuenssberg@uni-kassel.de
[4] Adam Malik Consulting, Germany, am@adam-malik.de
[5] Software Engineering, Kassel University, Germany, natascha@uni-kassel.de
[6] Distributed Systems, University of Tartu, Estonia, ulino@ut.ee
[7] Software Engineering, Kassel University, Germany, zuendorf@uni-kassel.de

doi:10.18420/modellierung2022-007

team, but formal notations like BPMN are usually hard to understand for non-ICT people like problem owners, domain experts, customers, end users, or business persons.

The same problems also persists in analysis and design: formal notations like UML diagrams are hard to understand and use effectively for non-ICT people. Therefore, it is hard for customers and end users to contribute to analysis and software design.

Frequently, requirement, analysis, and design documents are not well integrated nor aligned. Implementing, matching, and verifying of such requirements is hard and again contributing to the problem of non-ICT people feeling or actively seeking to be excluded from the software analysis and design process.

Due to our experience, non-ICT people are easily involved into the development when it comes to user experience (UX) e.g. via wireframes or mockups. Unfortunately, UX activities are often considered as a minor and late activity that is done after analysis and design. This causes user feedback to be quite late. In addition, UX activities are often not well integrated with the requirements elicitation activities. Again this is a difficulty for the contribution of non-ICT people.

To address some of these problems, we propose a slightly adapted software development process: iterate through 1. do some event storming (analog), 2. model events, wireframes, and read models for some steps and discuss these with users, 3. derive data modeling from the wireframes for the developers, 4. implement the modeled steps, 5. deploy. The following sections explain these steps.

2 Related Work: Event Storming

Event Storming has been proposed by Alberto Brandolini [Br13, Br21b]. Figure 1 shows an example of the result of an Event Storming workshop, which he teaches through workshops [Br21a] or presents in social media (eg. Youtube) [Br19]. Brandolini proposes to invite a group of relevant domain experts, customers and end users from all parts of your customer's enterprise. Each participant contributes to the workshop, by using sticky notes and a board marker on a large empty wall paper. By doing so, the participants create a contributive model containing of domain events and actions triggering such events.

Brandolini states that with the help of some moderation such an Event Storming workshop achieves a high involvement of non-ICT people. He claims that you should come up with the identification of all relevant workflows within a one or two day workshop. In addition, the grouping of people in front of the Event Storming Board already reveals some organisational structures and candidates for bounded contexts [EE04].

A great contributor to the facilitating effect is that Event Storming uses just pencil and paper and no formal notation, thus everybody can participate and gets easily involved. The

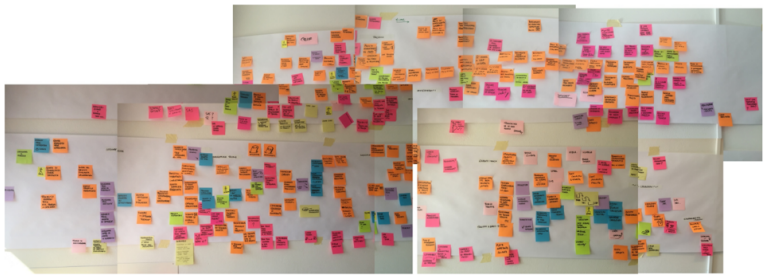


Fig. 1: Event Storming Big Picture ala Brandolini from [Br21b]

"Story Telling" part also fits very natural to the people. This is an experience that we share [Zü05, Zü19, NJZ13], and that attracted us to Event Storming.

The notation of BPMN diagrams is slightly related to event storming diagrams. BPMN diagrams provide decision points where workflows may branch into different directions. This helps to address all possibilities within a single diagram. Event Storming focuses on representative example workflows. If there are decisions that result in different behavior, we need to provide alternative scenarios for each case. The advantage of the alternative scenarios is that they outline in which situation which behaviour is chosen. This has the risk that some special case scenarios might not be covered. However, in our experiences domain experts are really good in pointing us to missing cases and help coming up with scenarios for these special cases. Thus, we believe that event storming scenarios are more likely to cover all relevant cases as BPMN diagrams that have been created by requirements experts. Still, we might summarize alternative Event Storming scenarios into a common BPMN diagram in order to have a condensed input for the analysis and design phases.

3 Related Work: Event Modeling

Once we have done some requirements elicitation e.g. with Event Storming, we need to become a little bit more formal in order to do analysis and design. However, we still want the non-ICT people to be involved and we want analysis and design integrated with the requirements steps. Therefore, the idea is to add wireframes to the Event Storming that show scenario steps from the user interface perspective. This refinement step is called *Event Modeling*. Thus, Event Modeling integrates UX design into Event Storming.

On writing this paper we learned that Adam Dymitruk has coined the phrase *Event Modeling* in 2018 [Dy18]. Actually, Dymitruk's ideas are quite close to ours. He also proposes to use wireframes and read models to refine Event Storming stories. Adding details to them allow for better user understanding and more preciseness and are a good starting point to the design and implementation phase. However, Dymitruk uses Event Modeling still in a very informal way on a whiteboard. In addition, he uses explicit command notes and read

model notes that we omit as they provide little extra information. Finally, we use a formal notation for the description of an event model. This formal notation allows us to generate different views on the system, e.g. a clickable mockup prototype, see below.

4 Event Modeling with FulibWorkflows

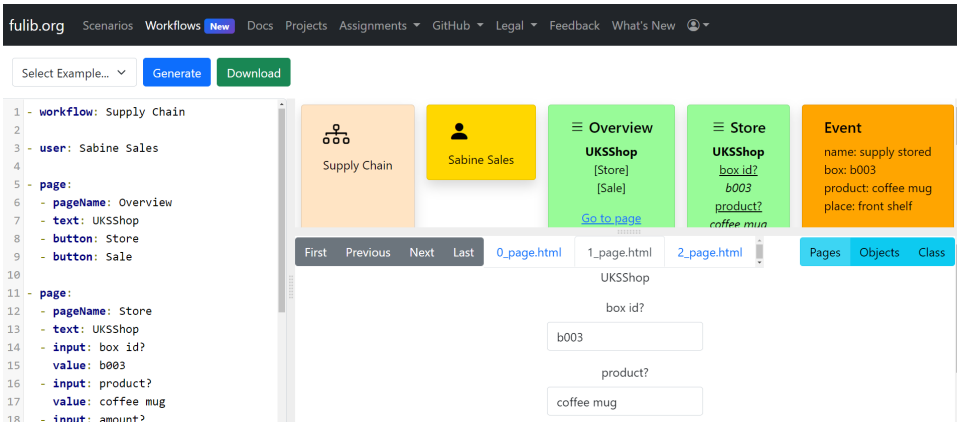


Fig. 2: The fulibWorkflows Tool

Figure 2 shows a screenshot of our *fulibWorkflows tool* with an example for a small event model for a University Kassel (UKS) web shop. The tool is available at our main tool site fulib.org [KFvK22]. In Figure 2 we have just started to formalize some domain events and added some wireframes. The left panel is an editor for our textual event modeling notation. In the upper right panel you see a graphical representation of the domain events in an event storming like event board. The lower right panel shows the mockups of a simple click prototype that is derived from our event model.

The fulibWorkflows tool uses a very simple Yaml based notation as input, cf. Listing 1, 2, and 3. We call this our *Event Modeling Notation*. Although, our event modeling notation tries to be as simple as possible, we do not expect that non-ICT people will be able to read and write event models in this notation. Thus, writing event model descriptions shall be done by requirements engineers or other software people. However, from the event modeling notation, we can derive the event storming board view containing events and simple wireframes as simple clickable mockups. These additional views allow us to integrate non-ICT people in our event modeling activities.

Workflows and Events

Listing 1 shows our Event Modeling notation for workflows and events. Lines 1 and 10 introduce the workflows “Supply” and “Selling”, respectively. Lines 2 to 4 introduce a

"supply ordered" event with the additional fields "product" and "amount". These additional fields are used to provide details on the information that is provided by an event and it will be input for our data modeling step. Usually, these additional fields are derived from input forms, see below. Other events are shown at lines 5, 11 and 14.

```
1 - workflow: Supply
2 - event: supply ordered
3   product: coffee mugs
4   amount: 50
5 - event: supply stored
6   box: b003
7   product: coffee mug
8   amount: 50
9   place: front shelf
10 - workflow: Selling
11 - event: prod offered
12   product: coffee mugs
13   price: Euro 1
14 - event: prod ordered
15   orderId: O_001
16   product: coffe mugs
17   amount: 1
18   customer: Alice
19   address: Wonderland 1
20 # ...
```

Listing 1: Events

```
1 - workflow: Supply
2 # ...
3 - user: Sabine Sales
4 - page:
5   - pageName: Overview
6   - text: UKSShop
7   - button: Store
8   - button: Sale
9     targetPage: Prices
10  - button: "..."
11 - page:
12  - pageName: Store Box
13  - text: UKSShop
14  - input: box id?
15    value: b003
16  - input: product?
17    value: coffee mug
18  - input: amount?
19    value: 50
20  - input: place?
21    value: shelf1
22  - button: add
23 # ...
```

Listing 2: Pages

```
1 # ...
2 - data: Box b003
3   product: mug
4   amount: 50
5   place: shelf1
6 - data: Box b002
7   product: mug
8   amount: 10
9   place: shelf2
10 - data: Box b001
11   product: tshirt
12   amount: 42
13   place: shelf1
14 - data: Product mug
15   boxes: [b002, b003]
16 - data: Product tshirt
17   boxes: [b001]
18 - data: Shelf shelf1
19   boxes: [b003]
20 - data: Shelf shelf2
21   boxes: [b002]
22 # ...
```

Listing 3: Data

We render the described events into a board of workflows. Ideally, the resulting event board resembles the analog board created in our event storming workshop. As explained in the industrial expert interview [MFvK22], participants in such an event storming workshop often keep a good spatial memory of the analog event board with its workflows and their steps. For the discussion of details, it helps tremendously to point to the physical location of the overall board where the corresponding events are shown. The former participants will immediately know what part of the overall workflow you are referring to and if they are not experts in that area themselves they will at least point you to the people that work in that area and that may provide you with detailed information to answer your detailed questions. Enabling even more communication, we stick to the ubiquitous language that we have developed during the event storming workshops.

Pages

Listing 2 focuses on the "supply stored" event of our example. The "user" entry of line 3 specifies that the following activities are executed by "Sabine Sales". This is rendered

by a yellow note in our event board, cf. Figure 3. Now we have added some very simple wireframes to our event model using “page” entries cf. lines 4 to 10 and lines 11 to 22 of Listing 2, producing the green notes in Figure 3.

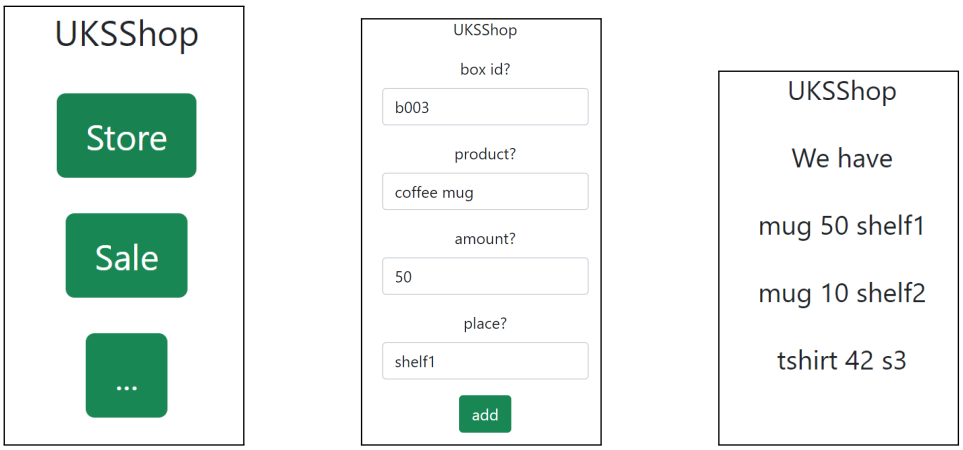
Fig. 3: Store Supply Pages

These two green pages of Figure 3 try to explain the user interaction that results in the orange “supply stored” event. The first page object describes the “Overview” page of our employee app with a [Store] button and a [Sale] button. A [...] button indicates that we expect more buttons to be added later. The [Store] button shall lead to the “Store Box” page. Per default, buttons just switch to the next page. Alternatively, line 9 of Listing 2 tells that the [Sale] button of the “Overview” page switches to a “Prices” page that is not shown in our Figures. The “Store Box” page is a form with four input fields containing the needed information for storing a new box. In our Yaml notation, you may specify example values for the input fields.

While the event board shows page details only as some ASCII art wireframe, the lower right compartment of fulibWorkflows shows an HTML based mockup of the described page. Figure 4 shows how a user might click through a scenario of 3 steps.

Figure 4a shows the mockup for the “Overview” page. If we click on the [Store] button, the lower right compartment of fulibWorkflows will switch to a mockup of our “Store Box” page. Depending on the scenario, you may first show an empty form and on another click switch to a filled form, cf. Figure 4b.

Walking users through such a little click dummy also facilitates users evaluating mockups. This possibility is also triggered by our expert interview[MFvK22]. When the users click through the mockups following the current scenario, the user will soon point you to missing information, missing inputs, missing alternatives and to superfluous or repeated inputs. In our example, Sabine Sales would probably ask for a feedback page that shows that adding the coffee mugs was successful. Thus we may add an “Inventory” page as shown in Figure 4c to our scenario. From the addition of another page and the potential discussion about the design of said page event more pages or changes to the existing workflow can emerge. Overall our experience shows that clickable mockups result in good involvement even of non-ICT people.



(a) "Overview" page (b) "Store Box" page (c) "Inventory" page

Fig. 4: Store Supply Mockups

Instead of our event modeling notation, one may use any other User eXperience (UX) approach, wireframe tool, or mockup library for the same purpose. However, due to our experience it is very important to link the UX steps to the corresponding parts of the event storming board. This reference helps the non-ICT people to recall what workflow parts are addressed and helps them to contribute to the discussion.

5 Data Model

While our detailed mockups and usage scenarios require quite some work, the resulting forms and "Inventory" pages do not only allow for excellent user involvement but they also provide a very good input for the data modeling and design phases. For example, the "Store Box" page shown in Figure 4b contains input fields for a box id, a product name, the amount of the product that the box contains and the shelf where the box is stored. These input fields will result in an event handler method with corresponding parameters and eventually in an object and class within our system model. Actually, the "Inventory" page in Figure 4c shows that there may be multiple boxes with the same product and that there are multiple products and shelves.

To allow the integration of data modeling into Event Modeling our Yaml notation uses e.g. "- data: Box b003" notes to describe objects (i.e. an object of type Box with id b003) and their attributes, cf. Listing 3. In our event board we render data notes in dark green color, cf. Figure 5.

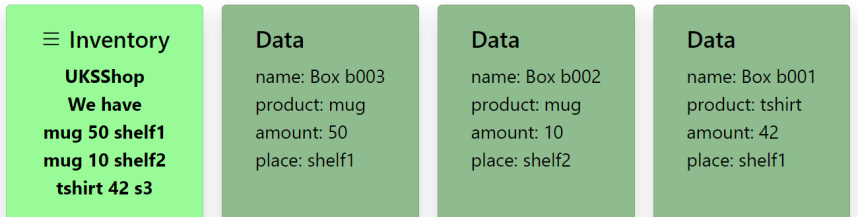


Fig. 5: FulibWorkflows Event Board with Data Notes

6 Conclusions and Future Work

As discussed, BPMN diagrams and UML diagrams are great means for providing the software development steps with concise information. However, these diagrams are hard to grasp for non-ICT people. Therefore, we propose to complement classical requirements engineering and software engineering diagrams with informal techniques like event storming. Similarly, we exploit wireframes and GUI mockups for better involvement of non-ICT people. Event modeling integrates wireframes and mockups into the event storming board. This again helps the involvement of non-ICT people. Detailed event models then serve as input for the data modeling steps. Again, our approach integrates the data modeling activities into the event storming notation in order to support consistency during iterative development.

We can even derive class diagrams for some of the core data models, we can employ code generation like using the FULib code generator [ZKE21] (or any other capable UML tool) to derive a Java implementation. However, the automatically derived class diagram is usually only a first conceptual model and thus data modeling remains an explicit engineering activity. In addition, the common class diagram tends to grow larger with the number of workflows that are incorporated. An idea for tackling the complexity of our data models is to split the whole system into a number of microservices, cf. [Co22]. [Co22] outlines that the workflows defined by Event Storming and refined through Event Modeling are a good start for splitting up monolithic models into a number of smaller micromodels.

Our approach exploits the experiences and insights of our co-author Adam Malik who uses many of our ideas in industrial projects. We have used our event modeling notation with great success in two courses at Kassel University [Zü21b, Zü21a]. We plan further evaluation of our approach within scientific as well as within industrial projects.

With fulibWorkflows [KFvK22], we have developed a first simple tool support for our event modeling approach. We plan to continue the development of the fulibWorkflows tool in order to support more enhanced wireframes and mockups. In addition, we need to develop mechanisms to connect analog event storming boards with the content of fulibWorkflows descriptions.

Bibliography

[Bo96] Booch, Grady; Jacobson, Ivar; Rumbaugh, James et al.: The unified modeling language. Unix Review, 14(13):5, 1996.

[Br13] Brandolini, Alberto: Introducing event storming. blog, Ziobrando's Lair, 18, 2013.

[Br19] Brandolini, Alberto: , 100,000 Orange Stickies Later. https://www.youtube.com/watch?v=fGm62ra_mQ8, 2019.

[Br21a] Brandolini, Alberto: , Event Storming. https://www.eventstorming.com/, 2021.

[Br21b] Brandolini, Alberto: , Introducing EventStorming. https://leanpub.com/introducing_eventstorming, 2021.

[Co22] Copei, Sebastian; Eickhoff, Christoph; Malik, Adam; Nolte, Natascha; Norbisrath, Ulrich; Sorgalla, Jonas; Weber, Jens H.; Zündorf, Albert: From Monolithic Models to Agile Micromodels. In: MODELSWARD. 2022.

[Dy18] Dymitruk, Adam: , Event Modeling: What is it. https://eventmodeling.org/posts/what-is-event-modeling/, 2018.

[EE04] Evans, Eric; Evans, Eric J: Domain-driven design: tackling complexity in the heart of software. Addison-Wesley Professional, 2004.

[Ga95] Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John; Patterns, Design: Elements of Reusable Object-Oriented Software. Design Patterns. massachusetts: Addison-Wesley Publishing Company, 1995.

[JBR99] Jacobson, Ivar; Booch, Grady; Rumbaugh, James: The unified process. Ieee Software, 16(3):96, 1999.

[KFvK22] Kunz, Adrian; Freiherr von Künßberg, Maximilian: , fulib.org. https://fulib.org/workflows, 2022.

[MFvK22] Malik, Adam; Freiherr von Künßberg, Maximilian: , Expert interview on using Event Storming for Requirements Engineering in an industrial context (In German). https://www.youtube.com/watch?v=2hYlh_xENDY, 2022.

[NJZ13] Norbisrath, Ulrich; Jubeh, Ruben; Zündorf, Albert: Story driven modeling. CreateSpace Independent Publ. Platform, 2013.

[OPM11] Omg, OMG; Parida, R; Mahapatra, S: Business process model and notation (bpmn) version 2.0. Object Management Group, 1(4), 2011.

[Sc00] Scheer, August-Wilhelm: ARIS—business process modeling. Springer Science & Business Media, 2000.

[ZKE21] Zündorf, Albert; Kunz, Adrian; Eickhoff, Christoph: Addressing Industrial Needs with the Fulib Modeling Library. In: MODELSWARD. pp. 155–162, 2021.

[Zü05] Zündorf, Albert: Story driven modeling: a practical guide to model driven software development. In: Proceedings of the 27th international conference on Software engineering. pp. 714–715, 2005.

[Zü19] Zündorf, Albert; Copei, Sebastian; Diethelm, Ira; Draude, Claude; Kunz, Adrian; Norbisrath, Ulrich: Explaining Business Process Software with Fulib-Scenarios. In: 2019 34th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW). IEEE Computer Society, pp. 33–36, 2019.

[Zü21a] Zündorf, Albert: , Microservices Course WT 21-22, Kassel University. https://www.youtube.com/playlist?list=PLohPa1TMsVqpdFQ8a2fPw1goIwwTMRolJ, 2021.

[Zü21b] Zündorf, Albert: , Programming and Modeling Course WT 21-22, Kassel University. https://www.youtube.com/playlist?list=PLohPa1TMsVqoveiQzR6nPqkV9OaJWzTCI, 2021.