Timisto: A Technique to Extract Usage Sequences from Storyboards

Joël Vogt[1,2] **Kris Luyten**[2] **Mieke Haesen**[2] **Karin Coninx**[2] **Andreas Meier**[1]

[1]Department of Informatics
University of Fribourg, 1700 Fribourg, Switzerland
{joel.vogt, andreas.meier}@unifr.ch

[2]Hasselt University - transnationale Universiteit Limburg - IBBT Expertise Centre for Digital Media
Wetenschapspark 2, 3590 Diepenbeek, Belgium
{kris.luyten, mieke.haesen, karin.coninx}@uhaselt.be

ABSTRACT
Storyboarding is a technique that is often used for the conception of new interactive systems. A storyboard illustrates graphically how a system is used by its users and what a typical context of usage is. Although the informal notation of a storyboard stimulates creativity, and makes them easy to understand for everyone, it is more difficult to integrate in further steps in the engineering process. We present an approach, "Time In Storyboards" (Timisto), to extract valuable information on how various interactions with the system are positioned in time with respect to each other. Timisto does not interfere with the creative process of storyboarding, but maximizes the structured information about time that can be deduced from a storyboard.

Author Keywords
Design Methods;Analysis Methods;

ACM Classification Keywords
H.5.2 User Interfaces (D.2.2, H.1.2, I.3.6): User-centered design; I.2.4 Knowledge Representation Formalisms and Methods (F.4.1): Temporal logic

INTRODUCTION
Engineering an interactive software system is essentially a creative activity that needs input from both technical and non-technical people. Current notations and tools often follow a strict *separation of concern* strategy in which members of such a team use the notations and tools they are accustomed with [3]. Given the wide diversity of notations and tools, synchronizing the various efforts is cumbersome [7]. Informal design artifacts are very accessible to non-technical people and often only require pen and paper [4, 13]. Storyboarding is such an informal technique that is frequently used for the design of interactive systems [14] and that will be the basic notation for the contributions described in this paper. The focus in this paper is on the extraction of both a comprehensible and formal specification of time based on informal storyboard drawings. With this piece of work we strengthen the link between the valuable informal artifacts that help us to understand the requirements and the users point of view and the engineering artifacts that are used to construct the interactive software system. As such this piece of work contributes to the engineering process to create interactive systems, more specifically helps to connect informal and formal artifacts.

We use McCloud's works on comics [9, 10] as a reference framework for analyzing storyboards. Like comics, storyboards are a visual form of storytelling. Images of imaginary or real things such as places, people or ideas are used to illustrate and convey ideas. Furthermore, drawing comics or storyboards is inherently spatial. The physical space that is used by sequences of drawings often visualizes the progression of the story in time. The sequence of images, their size, and the distance between images help the reader to move through time by moving through space. Panels have a special role in comics. They are snapshots that explicitly show moments of the story. The story evolves between the panels and is developed further in the reader's imagination. Figure 1 presents a storyboard that describes an interactive setup for an exhibition.

Unlike textual descriptions of scenarios or engineering models, understanding storyboards comes naturally to people because images do not require specific knowledge to decode [9]. A limitation of storyboards for further usage in an engineering process is that they are informal and subjective and often are still a subject to discussion. There is no explicit model that is agreed upon, and the information in storyboards is therefore not accessible to computers. Further comments or graphical annotations may be added to improve understanding among members of the design team [4, 16]. Without formal semantics, the content remains difficult if not impossible to process by a computer [12]. Providing tool support that would allow to automatically infer new information from the knowledge base or transform model to other modeling languages is therefore complex and not possible [4, 8, 12, 13]. There is already a lot of temporal information available in storyboards that can be extracted. Although this information is often incomplete or even subject to

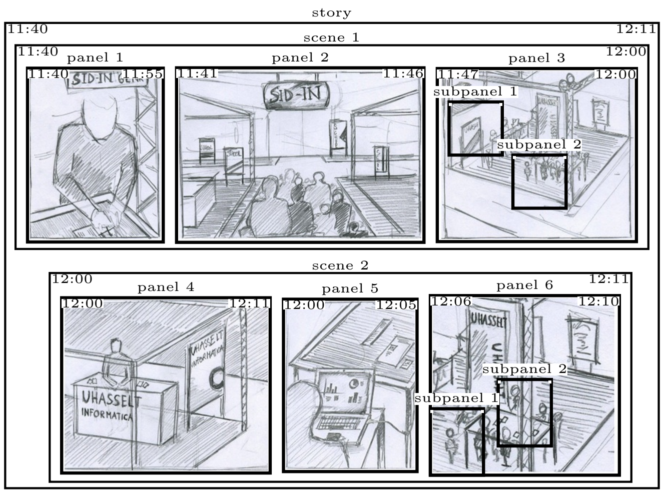


Figure 1: A storyboard presenting an interactive multi-touch system for exhibitions or fairs in seven panels.

change, most of the expected behavior of the software can be deduced from a storyboard. Notice we make the assumption a storyboard is sufficiently detailed to get a detailed overview of how a system works.

Storyboarding is a technique that is used in various domains to provide initial ideas and requirements. For example storyboards are a common tool in media production [6] and animations [15]. However, the applications form these domains are not designed to be integrated within an engineering process. We found the essential bit to connect storyboards with other software engineering artifacts, is the ability to extract temporal information from storyboards in a machine-understandable way. Haesen et al explored the technique of storyboarding as part of the engineering process [4, 8], but to our knowledge there is no well-defined way to translate system behavior as described in a storyboard in such a way it can be adopted by other software engineering artifacts such as a process model. We argue that the behavior of interactive software equals the set of ordering that can be found in a storyboard.

We present Timisto, an approach to extract useful temporal information from a storyboard. Our approach does not restrict designers' creativity since we do not interfere with the storyboarding activity itself, but rather helps designers to make precise statements about the time in a storyboard. After the storyboard has been created, our tool asks to annotate it with absolute time stamps. We use absolute time stamps for two reasons: first it fits with the storyboarding concept in drawing a concrete example so we handle the annotations accordingly. Second, using relative timings requires translating this information to more precise timing information which can be cumbersome. Third, using absolute timings avoids disagreement and misunderstandings, since relative timing annotations might be interpreted differently by the team

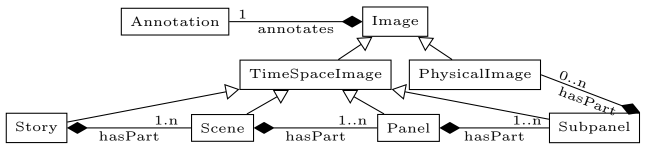


Figure 2: A storyboard contains images to visualize time and physical items. The passage of time is visualized by the structure of the storyboard and the size of images

members. Notice that our approach translates these timings to relative timings afterward and encodes them using Allen's temporal interval algebra [1]. The temporal information is used to map the content of the storyboard onto a timeline that visualizes the passage of time in storyboard. This visual depiction of the timeline helps designers and users to analyze how an interactive system behaves over time according to the storyboard.

STORYBOARD LANGUAGE

Our interpretation of storyboards is strongly inspired by McCloud's work on comics [9, 10]. McCloud defines comics as a "sequential art" or more verbosely as "juxtaposed pictorial and other images in deliberate sequence" [9]. By defining a storyboard language we want to clarify the way a storyboard provides information by means of images and structure. This combination of images and structure is very powerful to specify temporal information in storyboards.

A storyboard can be compared to a comic and basically presents images that are shown in a sequential order. These images are called *panels*. The meaning of the term panel will be explained below, but first we will give an example of a storyboard and its panels. The storyboard in Figure 1 presents an interactive multi-touch system for exhibitions or fairs in seven panels. On the one hand, the system presented by the storyboard provides information to visitors of the booth at the exhibition. On the other hand, it allows visitors to enter their personal data in order to be contacted by the exhibitor afterwards. Panel 1 zooms in to the multi-touch system and its users, panels 2, 3 and 6 provide an overview of the booth at the exhibition and panels 4 and 5 visualize the representative of the booth behind a desk, collecting information of the visitors that was entered into the multi-touch system. Besides the aforementioned structure that is clearly visualized by the panels in a storyboard, other information can also be inferred from a storyboard. To make use of this inferred information for specifying temporal information in tool support, we propose to annotate basic elements of a storyboard. These annotations concern *scenes* (a group of related panels), *panels* (images in a storyboard) and *subpanels*. The relationship between them is shown in the storyboard model depicted in Figure 2. The characteristics of the storyboard's structural elements can be summarized as follows:

Scene : A storyboard is drawn as one or more scenes by physically grouping panels of a storyboard that are related to each other in time or in space. In some comics, the author tries to fit all panels of one scene to a page in order to improve the readability of a comic. However, authors of comics can also present two different scenes in successive panels, just to emphasize significant distances between time or space [11].

Panel : A panel is a window into a moment of the story, that shows what is happening during that time. The size of a panel often refers to the time the panel takes. Inside a panel, an actor performs an action during that moment that implies the advancement of time. An action is visually depicted as "motion" or "sounds". Although it is difficult to visually present motions or sounds in still images, there are several techniques to realize this [9]. Panels also steer the reader's interpretation of the story with the angle through which the reader views the story, the level of detail and the placement of objects within the panel.

Subpanel : According to McCloud, a panel with more than one action implies more than one moment. A subpanel captures a specific action inside a panel, thus mostly shows one or more users performing an action. For each action that occurs in a panel a separate subpanel is used.

TIME AND SPACE IN STORYBOARDS

The way time is visualized in a storyboard is not precise enough when concrete information about the passage of time is needed. Truong et al. [16] found that explicit depictions of time can affect a reader's understanding of a story. They argue that "time passing was a significant element needed to understand particular storyboards" [16].

The Timisto Approach

The Time In Storyboards (Timisto) approach builds on the findings of Truong et al. to add an additional layer of precise temporal information on existing storyboards: (1) we extract temporal relations from a storyboard based on structure and content in combination with annotations for more precision and (2) when these temporal relations are incorrect we provide feedback to the user.

For example, panel 1 in Figure 1 can be annotated with specific timestamps that show it lasts from 11:40AM to 11:55AM. During that time the exhibitor checks the multi-touch system and observes the newly arrived visitors. Panel 2 starts at 11:41AM, when the group of visitors arrives at the exhibition, and ends at 11:46AM. In panel 3, booth visitors interact with the multi-touch table: one group from 11:47AM to 11:55AM and a second group from 11:53AM until 12:00PM. These interactions represent two sub panels of panel 3. Panel 3 therefore lasts from 11:47AM to 12:00PM. Timestamps are provided by users and designers to discuss the progression of the story. These values are estimates and the precise timing is not important. What is important however are the temporal relationships between elements in the storyboard that users implicitly describe through the precise time stamps. For example, in panel 1 the exhibitor observes a group of visitors, who just arrived in panel 2. After having arrived, visitors gather around the multi-touch table in panel 3. This means that the action of observing visitors occurs when new visitors arrive. Furthermore, visitors must first arrive before they can access the multi-touch table. With the precise timestamps, such temporal relationships between different parts of the storyboard can be automatically inferred.

Allen's Temporal Interval Algebra

We make use of *Allen's temporal interval algebra* [1] to describe temporal relationships in storyboards. This algebra has thirteen disjoint relationships.The five basic relationships are $before$, *equals*, *overlaps*, *meets* and *during*. The relationships *starts* and $finishes$ are two special cases of *during*. Each relationship has an inverse relationship (except for *equals*). The temporal relationships between the first three panels of Figure 1 that were informally introduced for the example in the pervious section, are as follows: "Exhibitor observes" (Panel 1) *overlaps* "Visitors arriving" (Panel 2). "Visitors arrive" (Panel 2) *before* "Gathering around the multi-touch table" (Panel 3). Allen's temporal relationships are transitive: "Exhibitor observes visitors" *before* "They gather around the multi-touch table". Allen's temporal interval algebra is suitable for specifying time in storyboards, because it is a generic algebra and usuable within most application domains. Furhtermore, Allen's temporal algebra works with relative time.

Temporal Domain Ontology for Storyboards

We developed our ontology specifically to accommodate the extra information we encode with respect to the Web Ontology Language (OWL)-Time ontology, being the storyboard structure. Since both use Allen's temporal interval algebra as the foundation for time specification, full equivalence is guaranteed with OWL-Time ontology. Before explaining how the actual extraction of temporal information is done, we define *five* interval types that need to be considered for storyboards:

- **Temporal interval:** A temporal interval defines the time span of each interval with $from$ and *to*, i.e. the time beginning and the end of the interval i, where $from(i) \leq to(i)$. If an interval contains other intervals, its $from$ value is set to that of its first ancestor and the *to* value to that of its last ancestor. The *hasPart* relationships, respectively *partOf* refer to the the physical nesting of their annotations on the storyboard. The relationships *hasPart* and *partOf* are strictly between intervals of lower, respectively higher granularity on the storyboard. Furthermore, each interval is unique and it is assumed that all intervals are known.

- **Story Interval:** A story interval represents the time of the entire story. It consists of the sequence of scenes that lead up to one or more actor achieving their goal and ending the story. A story interval states *how long* a story took.

- **Scene Interval:** A scene interval represents the time of a scene. In the storyboard, a scene is most likely a group of panels on a page or a very large panel. Figure 1 contains two scenes. The first scene informs the reader that the main setting is an exhibition and the actors are an exhibitor and visitors. It shows a newly arrived group of visitors who gather around a multi-touch table. The second scene shows the exhibitor using a remote monitoring tool to analyze how the multi-touch table is used.

- **Task Interval:** A task interval is the time during which one or more subjects perform one or more actions to reach a goal. A task interval is associated with one or more panels. A task interval describes *during* what time someone did something at a location. *Where* is described by a location ontology. In the first panel of Figure 1, the main task of the exhibitor is to watch newly arrived visitors. His goal is provide information to visitors if they have questions.

- **Action Interval:** An action describes the time of an image that visualizes motion or sound that advance the story in time. An action states *when* a subject does a task. The type of action is described by another ontology. *Who* is described for example in a persona. Again in the first panel of Figure 1, the action of the exhibitor is watching newly arrived visitors to be able to assist if needed.

VISUALIZING TIME IN A STORYBOARD

In this section, we discuss the Timisto application that was developed to provide tool support for the Timisto approach and explain how the precise time information can be used to visualize the passage of time in the storyboard as a timeline.

Specifying the Time with the Timisto Application

Our tool allows users to draw annotations on digital images of storyboards and specify time of the annotation. The annotations delimit the structure and layout of the storyboard and specify the kind of structural element, i.e. scene, panel or subpanel, of each annotation, as shown in Figure 2.

The time of an annotation specifies the duration of that image, which is stored as a temporal interval. The Timisto application keeps a list that associates structural elements of the storyboard language to interval types. The storyboard is represented as a story interval. An annotation of a scene is represented as a scene interval, an annotation of a panel as a task interval and an annotation of a subpanel as an action interval. This assignment can be made automatically, since the structure of the storyboard represents the structure of time.

Function createTimeLine(interval,addLane)

```
levels ← {{}, {}, {}, {}}, level ← 0;
begin
    if addLane then
        lane ← {};
        add(levels [level ],lane)  // Add propagated lane;
    else
        lane ← last(levels [level ]) // Use existing lane;
    end
    descendants ← {};
    cType ← descendantTypeConstraint (interval);
    if cType ≠ null then
        descendants ← ∀i ∈ cType: hasPart(interval,i);
    end
    sort(descendants);
    foreach descendant ∈ descendants do
        level ++;
        createTimeLine(descendant,addLane);
        add(lane,descendant);
        conflict ← ∀i ∈ descendants : overlaps(descendant
        , i) ∨ contains(descendant , i) ∨ equals(descendant
        , i);
        if |conflict| > 0 then
            lane ← {};
            add(levels [level − 1],lane);
            addLane ← True    // Propagate new lane;
        else
            addLane ← False    // Set existing lane;
        end
        level- -;
    end
end
```

Time Extraction and Visualization Algorithm

In this section we present the algorithm we designed for extracting temporal relations from a storyboard and, simultaneously, creating a graphical overview of these temporal relationships. For the sake of reproducibility, we provide an in depth description of the algorithm alongside an example of execution of the algorithm. In the spirit of RepliCHI[1] and Executable Papers[2], we will provide a publicly accessible executable demo of this tool through the SHARE environment [2].

Based on the precise time information, the Timisto application can render a timeline of the storyboard to visualize the passage of time. It will split the storyboard in subparts according to the temporal interval types linked to annotations and present a graphical timeline with these subparts of the storyboard ordered on top of the timeline. We believe this is an important feature of our approach, since during storyboarding a visual and detailed representation of the temporal relationships within a storyboard also informs the creators of the storyboard. It allows for additional adjustments and to detect ambiguities with respect to time during the storyboarding phase.

The createTimeLine function receives an temporal interval, interval and a boolean value, addLane, that states if a

[1]http://replichi.org/
[2]http://www.executablepapers.com/

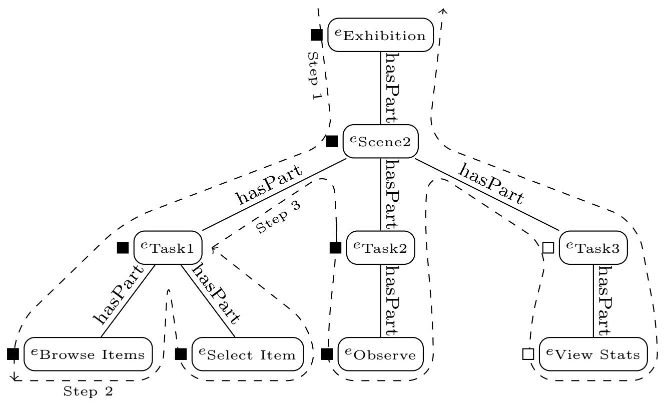
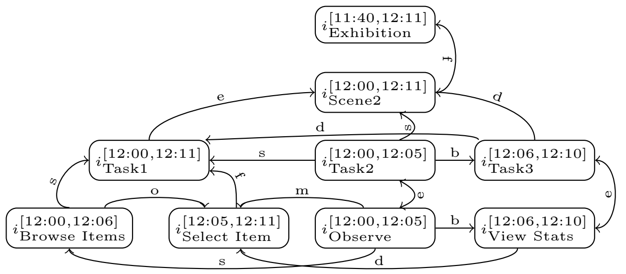
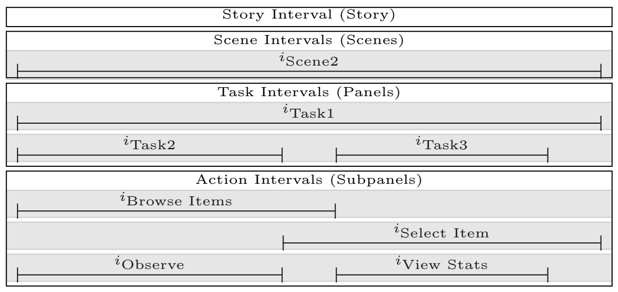


Figure 3: Interval hierarchy for scene 2 in Figure 1. The dashed line shows how createTimeLine places intervals on the timeline. The square left of an interval shows the value of addLane. Full square: a new lane is added. Empty square: existing lane is used

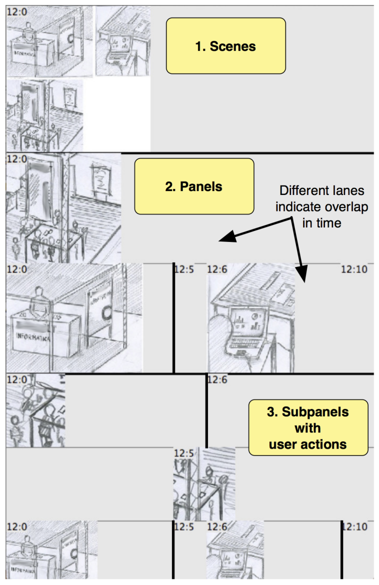
new lane for its direct descendants has to be created or an existing lane can be used. A lane is one horizontal layer that can be seen in Figure 4c, so it is a graphical division that represents a timeline for a specific level of detail. To create a timeline, the first argument is the topmost temporal, i.e. the story interval, and addLane is set to true to add a new lane to each level. createTimeLine first prepares a lane in the timeline on which direct descendants of interval will be placed. Next, it processes the direct descendants of interval in chronological order. For each direct descendant, createTimeLine is called, with addLane as argument. descendant is then added to the previously prepared lane. The algorithm then verifies if the descendant right of interval, which will be processed in the next iteration, overlaps with the current descendant. If there is no overlap, that interval can be placed on the same lane as descendant and addLane is set to false. However if the next direct descendent overlaps with descendant, subsequent direct descendants will be placed on a new lane below. Because the time of an interval depends on its first, respectively last descendent, a new lane is also propagated to lower levels by setting addLane to true. The recursive implementation of createTimeLine means that a temporal interval is only added to the timeline after all of its descendants are added. The first temporal interval to be added is therefore the first action interval that starts the storyboard. The timeline is finally rendered by substituting temporal intervals with their annotations. Consider timeline in Figure 4c. It was rendered by annotating the structure of the second scene in Figure 1 and specifying the time in each image. This information is stored as a temporal interval hierarchy, shown in Figure 3. createTimeLine was called with the values interval $= i_{Exhibition}$ and addLane = true as arguments. The dashed line with arrow in Figure 3 shows the order in which createTimeLine traverses the interval hierarchy to place each interval in the right order and level on the timeline.



(a) Temporal relationships between intervals



(b) Timeline visualizes temporal relationships



1. Scenes
2. Panels
Different lanes indicate overlap in time
3. Subpanels with user actions
12:0
12:5
12:6
12:10

(c) Replacing intervals with images

Figure 4: Visualizing the time of scene 2 as a timeline

- **Step 1** createTimeLine is called with interval $= i_{Exhibition}$, and addLane = true, causing a new lane to be added to each level except for the story level. $i_{Exhibition}$ will not be placed on the timeline, because it is not called $i_{Exhibition}$. createTimeLine first sorts the direct descendants of interval, before increasing level by one and calling itself recursively for each descendant. For i_{Scene2}, it adds a lane to the task level, and for i_{Task1} to the action level.

- **Step 2** An interval is added by the createTimeLine for its parent interval. $i_{BrowseItems}$ is added first, then $i_{SelectItem}$. Because createTimeLine for i_{Task1} detects an overlap between its two descendants, it adds $i_{SelectItem}$ onto a new lane. createTimeLine for i_{Task1} exits and is added to the task level.

- **Step 3** However, it overlaps with its sibling i_{Task2}. A new lane is therefore added to the task level. Furthermore, addLane is set to true, to add a lane for the descendant of i_{Task2}, $i_{Observe}$. $i_{Observe}$ is added to the new lane of the action level, and then i_{Task2} is added to the new lane of the task level. Because i_{Task3} does not intersect with i_{Task2}, it will also added to the same level. addLane is set to false, causing the first descendant of i_{Task3} to be added to the same lane as $i_{Observe}$.

Once the timeline contains all intervals (Figure 4), they are replaced with their corresponding annotated image, as shown in Figure 4c.

CONCLUSION

In this paper we introduced Timisto, an essential part to enable correct and consistent integration of storyboards with other engineering artifacts. A storyboard contains a lot of important time-related information on the usage of an interactive system, which was not accessible up until now. Timisto extracts temporal information from a storyboard as a set of temporal relationships that reflect the temporal behavior that should also be represented by the other software models leading to an interactive system, such as process models. Our algorithm uses simple straight-forward annotations on top of a storyboard and generates temporal relations that adhere Allen's temporal interval algebra, a widely accepted algebra for specifying relative time. Our approach has several benefits: it provides an accessible way for non-engineers to describe the time-related aspects of an interactive system. It provides a visual overview of the temporal information in a storyboard. Figure 4 presents such a timeline and shows both the hierarchy (containment in time), overlaps and sequences are easy to read from this visualization. Temporal relationships *meets* and $before$ that correspond to sequential events are placed in chronological order. Events that occur in parallel, related by *overlaps*, *equal* and *during* (with the special cases *starts* and $finishes$), are placed on separate lanes. Especially encompassing user interface description languages such as UsiXML [5] can take advantage of our work to connect with the early, informal stages in the engineering cycle. We are currently planning an integration with UsiXML by integration our algorithm with the StoryBoardML language devised by Luyten et al. [8].
The focus in this paper is on the extraction of both a comprehensible and formal specification of time based on informal storyboard drawings. With this piece of work we strengthen the link between the valuable informal artifacts that help us to understand the requirements and the users point of view and the engineering artifacts that are used to construct the interactive software system.

REFERENCES

1. Allen, J. F. Maintaining knowledge about temporal intervals. *Communications of the ACM 26*, 11 (1983), 832–843.
2. Gorp, P. V., and Mazanek, S. Share: a web portal for creating and sharing executable research papers. *Procedia CS 4* (2011), 589–597.
3. Haesen, M., Coninx, K., Van den Bergh, J., and Luyten, K. Muicser: A process framework for multi-disciplinary user-centred software engineering processes. In *EICS*, Springer (2008), 150–165.
4. Haesen, M., Van den Bergh, J., Meskens, J., Luyten, K., Degrandsart, S., Demeyer, S., and Coninx, K. Using storyboards to integrate models and informal design knowledge. *MDDAUI* (2011), 87–106.
5. Jean Vanderdonckt, J., Beuvens, F., Melchior, J., and Tesoriero, R., Eds. *UsiXML*. W3C Working Group Submission, 2010. http://www.lilab.be/W3C/.
6. Jones, M. Getting started with celtx: Scriptwriting and pre-production. *Screen Education*, 46 (2007), 196.
7. Lindland, O. I., Sindre, G., and Solvberg, A. Understanding quality in conceptual modeling. *Software, IEEE 11*, 2 (1994), 42–49.
8. Luyten, K., Haesen, M., Ostrowski, D., Coninx, K., Degrandsart, S., and Demeyer, S. D. On stories, models and notations: Storyboard creation as an entry point for model-based interface development with usixml. In *USIXML*, ACM (2010).
9. McCloud, S. *Understanding comics : the invisible art.* HarperPerennial, New York, 1994.
10. McCloud, S. Reinventing comics: How imagination and technology are revolutionizing an art form. *Perennial, New York* (2000), 118–122.
11. McCloud, S. *Making comics : storytelling secrets of comics, manga and graphic novels.* Harper, New York, 2006.
12. Meyer, B. On formalism in specifications. *Software, IEEE 2*, 1 (1 1985), 6–26.
13. Ozenc, F. K., Kim, M., Zimmerman, J., Oney, S., and Myers, B. How to support designers in getting hold of the immaterial material of software. In *CHI '10*, ACM (2010), 2513–2522.
14. Rogers, Y., Sharp, H., and Preece, J. *Interaction Design: Beyond Human-Computer Interaction.* John Wiley and Sons Ltd, 2011.
15. Smith, J., Osborn, J., and Team, A. C. Adobe creative suite 5 design premium digital classroom. *Beograd: Mikro knjiga* (2010).
16. Truong, K. N., Hayes, G. R., and Abowd, G. D. Storyboarding: an empirical determination of best practices and effective guidelines. In *DIS '06*, ACM (New York, NY, USA, 2006), 12–21.