# Supporting the HCI Aspect of Agile Software Development by Tool Support for UI-Pattern Transformations

Peter Forbrig(✉) and Marc Saurin

University of Rostock, Albert-Einstein-Str. 22, 18051 Rostock, Germany
peter.forbrig@uni-rostock.de

**Abstract.** Continuous changing requirements of software are the result of continuously changing reality. This reality can be considered as the context of the software. Agile development methods allow quick adaptations to changing requirements. Initially, agile development methods were focused on the development of the application core only. Recently, process models were discussed that integrate HCI aspects. This paper will discuss ideas to integrate user evaluations into the development process. User interfaces are structured as UI-pattern instances. Tool support is provided that allows the specification of pattern instances as XAML specifications. Additionally, the tool allows the replacement of one pattern instance by another one. In this way, different versions of the same user interface can be generated rapidly without much effort. These different versions can be evaluated with the help of users. Based on these usability tests final decisions for the software design can be made. New requirements can be captured additionally. This will be based on feedback of the users.

**Keywords:** UI-Patterns · Pattern instance transformation · Agile software development · Human-Centered Design

## 1 Introduction

Our society changes continuously. Therefore, software solutions have to be adapted during usage. However, even during the development requirements are not stable. Developers have to react on dynamical changes. This is the reason for the need of agile approaches. Classical development methods often fail.

Unfortunately, software engineers often focus on the development of the application core only. Aspects of user interface design and HCI methods are not in the focus of their work. That is the reason why process models of agile methods like SCRUM do not contain HCI activities. Recently, there are several approaches like [2, 5–8, 15].

Agile development methods very much support the communication between developers and customers. However, users should be involved as well.

The Human-Centered approach is accompanied with a phase where design decisions are evaluated. To support design decisions tool support would be helpful that generated different alternatives. Based on the user evaluation the best design can be selected and further developed. We will present a tool that allows developers the development of


C. Bogdan et al. (Eds.): HCSE 2016/HESSD 2016, LNCS 9856, pp. 17–29, 2016.
DOI: 10.1007/978-3-319-44902-9_2

different user interfaces based on pattern transformations. Additionally, we will discuss how this tool fits into agile development methods. We will discuss this aspect on a specific process model for SCRUM.

The rest of the paper will be structured in the following way. First, we will discuss the idea of UI patterns and the corresponding tool support. Afterwards, the integration of the Human-Centered Design into SCRUM will be discussed. Additionally, it will be shown how the developed tool can be used within the development process. Finally, there will be a summary and an outlook.

## 2 UI-Patterns and Tool Support

### 2.1 UI-Patterns

The success story of patterns in computer science started with the well-known book by the "Gang of Four" [3]. Later, this idea was adapted to different subdomains. In the meantime, there exist patterns about workflows, tasks, ontologies, and a lot of other aspects.

UI-Patterns have been proven to be very useful for designing interactive software systems. Resource for that are e.g. [11, 16, 17], to mention only a few of them. Most of existing libraries are for human browsing only. The application has to be performed manually. However, there exist tools that allow the application of UI patterns.

The term pattern is sometimes used a little bit vague. In the final user interface, one cannot see neither any pattern nor any pattern instance. One can only see the result of the application of a pattern instance. Let us assume the following application process of UI patterns:

1. **Identification:** A subset S' of user interface elements S is identified that can be transformed by a pattern. $S' \subseteq S$
2. **Selection:** A pattern P is selected that can be applied to S'.
3. **Adaptation:** The pattern P is adapted to the current context of use M'.
   As a result, a pattern instance I is delivered. $A(P,S') = I$
4. **Integration:** The instance I is integrated. It replaces M' in M.
   $I(S', S) = S^*$ (Pattern instance I is applied to subset S' of S and delivers a new set of user elements S* – a new user interface)

Using this terminology a user interface presents the result of the application of pattern instances. Pattern instances are the result of adaptations of patterns to the current context of use. They are applied to existing elements and deliver new user interface elements. However, to be short sometimes the structure of user interfaces is presented by the names of the corresponding patterns only.

It was already mentioned that there exist tools supporting the application of UI patterns. However, about the transformation of user interfaces by pattern is yet not much reported even there exist papers about such transformations for a relatively long time. Already in 2004 in [4] it was reported about the opportunity to transform user interfaces that were constructed based on patterns. The paper discusses ideas, which

pattern applications should be replaced by other ones in case the application should run on mobile devices. They call it pattern mapping. We will recall only three mapping rules of their Table 12.4. The enumeration comes from their pattern catalog.

P.1 Bread crumbs is replaced by

P.1 s – Shorter bread crumb trail; and
P.15 – Drop-down 'History' menu

P.2 Temporary horizontal menus replaced by

P.2 s – Shorter menu; and
P.3 – Link to full-page display of menu options ordered vertically.

We adapt this idea for mapping or transformations of pattern applications on the same platform. It can be considered as refactoring like in [10]. With tool support, different versions of a user interface can be generated quickly. A horizontal menu could e.g. be transformed to a vertical one. Such kind of tool support is discussed in the next section.

## 2.2 Tool Support of UI-Patterns Using XAML

Within a Master Thesis [12] a tool was developed for constructing and transforming user interfaces based on UI-patterns. The tool is based on Visual Studio using the technology of VSIX extensions and XAML specifications of user interfaces. Figure 1 shows the XAML representation of the Split Pane Pattern in its horizontal version.

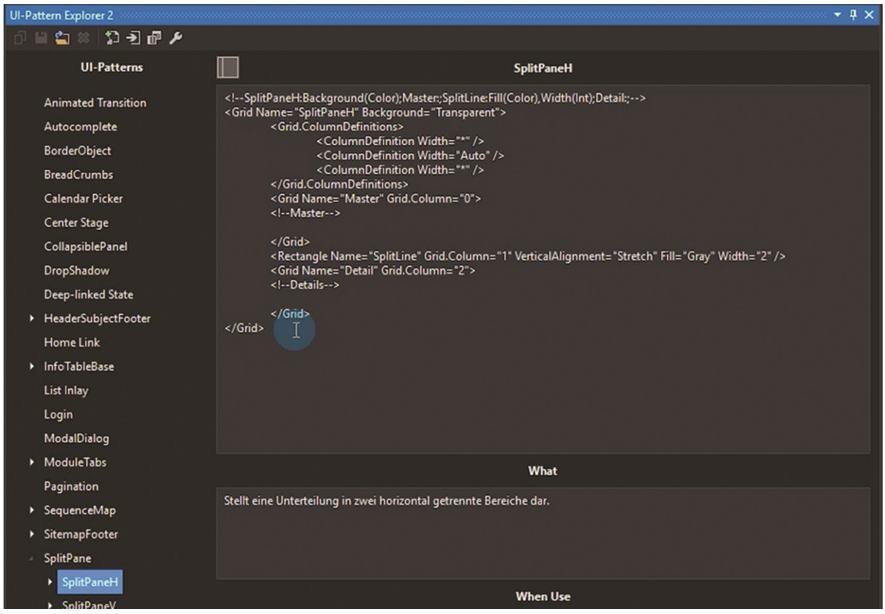


**Fig. 1.** Pattern specification in UI-Explorer 2.

The UI-Explorer 2 tool uses representation files for patterns. Currently XAML is used. It fits well to the provided features of WPF (Windows Presentation Foundation). However, any other XML-based specification language like HTML or ASPX could be used as well. The Grid-tag was used to represent patterns. Some attribute can be set initially- They can be changed later. In the example above the master and detail part are still empty. They can be filled later. Figure 2 demonstrates the application of the pattern.

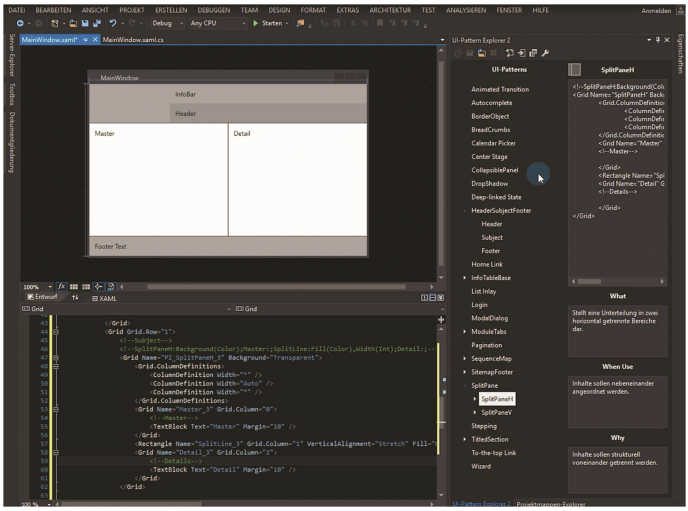


**Fig. 2.** Application of the horizontal Split Pane Pattern.

Let us have a look at replacing the application SplitPaneH by SplitPaneV (split pane vertical). UI Explorer 2 supports this kind of transformation and delivers the result below.

Currently pattern applications can be transformed at one location only. For the future it is planned to allow the replacement of all instances of a pattern application by another one. It will also be possible to allow to replace a sequence of pattern applications by another sequence of pattern application. This would include the replacement of one pattern application by a sequence of applications as well as the replacement of a sequence of pattern applications by one pattern application (Fig. 3).

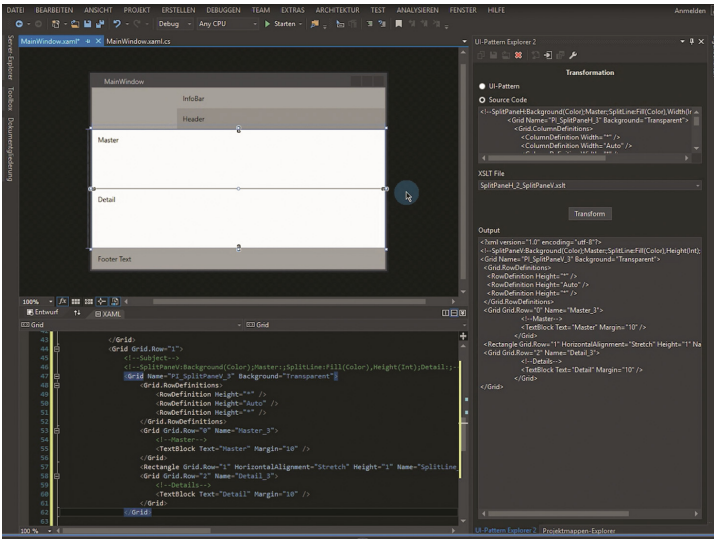


**Fig. 3.** Replacement of the application of the SplitPaneH by SplitPaneV.

Formally, this can be described by the following notation:

| | | |
|---|---|---|
| P1(X) | => P2(X) | Pattern P1 is replaced by P2 |
| P1(X, P2(Y)) | => P3(X, Y) | Patterns P1 and P2 are replaced by P3 |
| P3(X, Y) | => P1(X, P2(Y)) | Pattern P3 is replaced by P1 and P2 |
| P1(X, P2(Y)) | => P3(X, P4(Y)) | Patterns P1 and P2 are replaced by P3 and P4 |

## 2.3 Case Study

To get an impression of the applicability of the tool, a case study was performed. The websites of Lufthansa, Eurowings, and Norwegian were analyzed und their structure according to UI-Patterns applications were analyzed. The resulting structure is shown in Fig. 4.

An already refined and transformed version of the user interface is presented as Fig. 5. The horizontal version of Master and Detail was replaced by a vertical one.

A further transformation yields to the result of Fig. 6. The navigation in the calendar is replaced by a new pattern application. This structure corresponds to the structure of the webpage of Norwegian (Fig. 7).

The case study had shown that the approach worked for examples of real applications. The tool was able to handle transformations of different levels of abstractions (Fig. 8).


**Fig. 4.** Structure of the web-page of Eurowings.

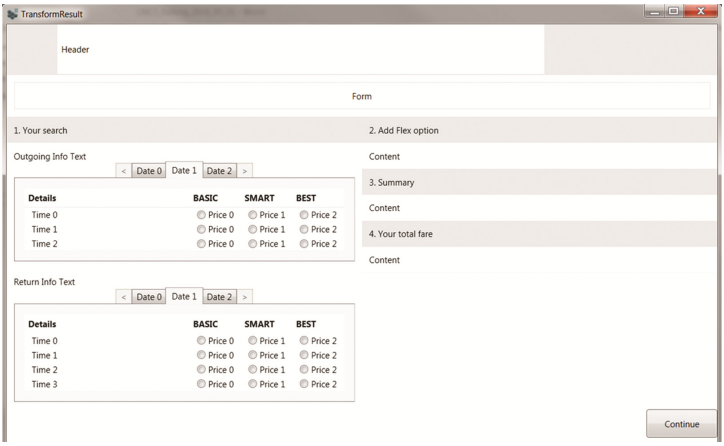


**Fig. 5.** Refined and transformed part of the user interface of the web page of Eurowings.

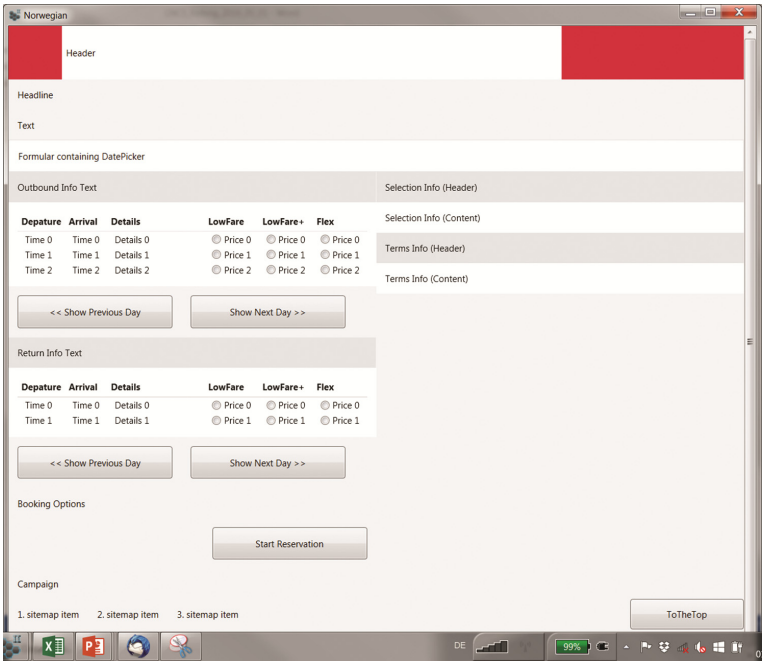


**Fig. 6.** Further transformed page that corresponds to the structure of that of Norwegian.

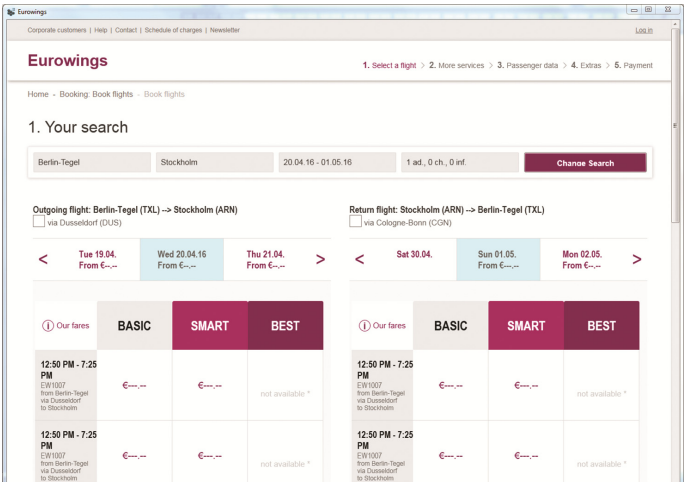


**Fig. 7.** Part of the detailed user interface of Eurowings.

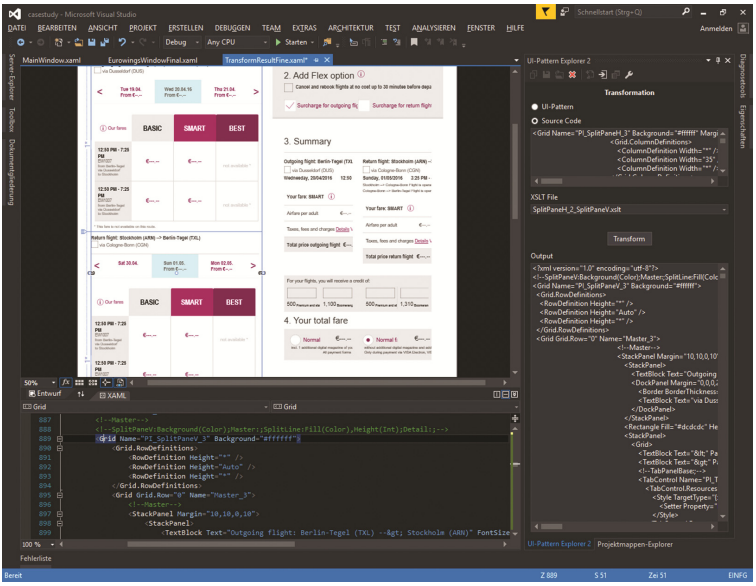

**Fig. 8.** Transformed detailed user interface of Eurowings within the Pattern Explorer 2 tool.

It is not astonishing, that based on the structure similarities even the detailed webpage of Norwegian could be generated (Fig. 9).

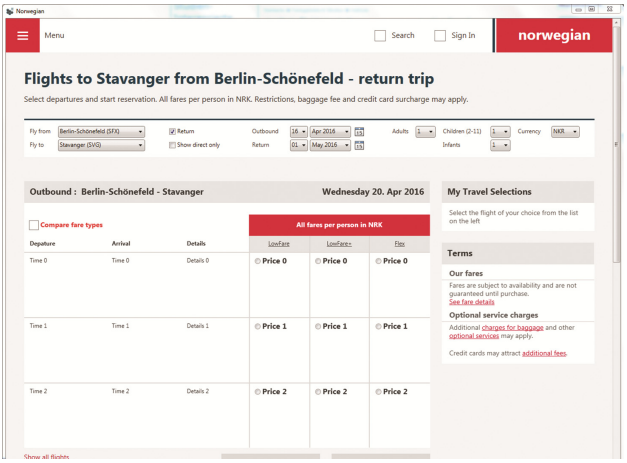


**Fig. 9.** Detailed webpage that is very similar to that of Norwegian.

Currently, the structure of the user interface has to be modeled by hand. That means that the creation of the result of the pattern instance applications has to be done by hand.

However, in the future it is planned to integrate the approach into a model-based tool chain that allows the generation of user interfaces. Parts of this tool chain can even be model-driven. A combination with the approach of Yigitbas et al. [18] seems to be promising.

## 3 Agile Development and Continuous Human-Centered Design

After introducing the developed tool a little bit, we will focus on the development process and in which way the tool could be used. Additionally, we are interested to combine the better of two worlds - the best of Human-Centered Design and Agile Development. The first principle of the Agile Manifesto [1] is: "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software". According to this principle, customers are most important. This might be perfect from the business perspective because the customer has to pay the bill. However, from the quality aspect it is important to get the users involved as well.

User-Centered Design and nowadays Human-Centered Design are in the same way popular within the community of usability and user experience experts as agile methods for software engineers that focus on the application core. HCD focusses especially on the context of use and the evaluation of design decisions. That seems to be the major reasons for its popularity. In this context, user requirements are considered to be more important than functional requirements coming from the customer. Finally, the users will really get what they need to get their working tasks supported. ISO 9241-210 is a standard for the HCD process that consists of a planning phase and four phases that are performed in an iterative way.

In the first phase, stakeholders and their context of use are identified by analysts. Typical application scenarios are specified. Additionally, tasks that have to be supported are analyzed. Users and their roles are identified. The roles are related to tasks. However, tasks are also related to objects that are changed by performing the task or that are used as tools. Additionally, the context of use of the software under development is specified. This can be the location, the surrounding persons or objects and in some cases available services.

User requirements are specified based on this analysis. They contain besides functional and non-functional requirements additionally the goals of the users and their profiles.

First design solutions are produced afterwards. They have to fulfill the identified requirements. Such design solutions focus mainly on first ideas of user interfaces. This can be mock-ups or running prototypes.

In the last phase of the HCD process, developed design solutions are evaluated. Very often, the design solutions do not meet the requirements. They are not the wanted result. Therefore, new considerations have to be made. In the worst case, one has to start with the first phase again. The context of use has to be analyzed again. However, if the general analysis of the context of use seems to be correct but some requirements were specified in the wrong way, one has only to rewrite some requirements or has to identify some

new ones. If only some design solutions did not meet the requirements, one has to look for an alternative design. The optimal case is of course if the requirements of the users are met immediately. In this case, the development process comes to an end and the implementation of the application core can be performed.

Most of the time there will be several cycles until the design fulfills the analyzed user requirements. A visual impression of the HCD process model is given by Fig. 10.

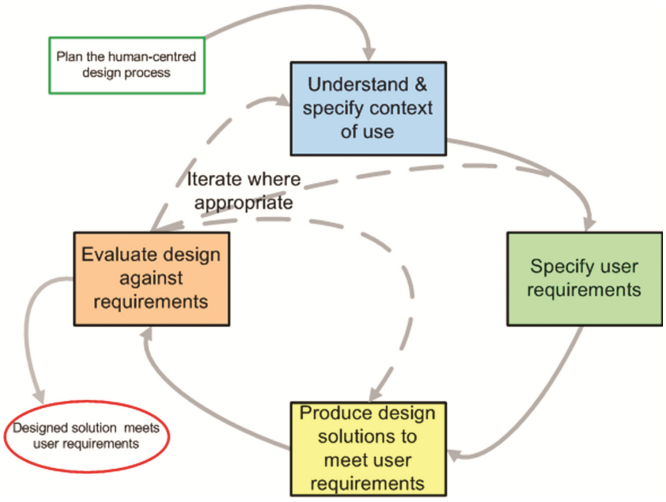


**Fig. 10.** The design process from ISO 9241-210–Human-centered design process (from https://thestandardinteractiondesignprocess.wordpress.com/).

Even that Fig. 10 provides a good overview of the main ideas of the HCD process, it does not provide hints how the idea of HCD can be integrated into the agile development process. However, the agile development process neglects the problems of HCD as well. Indeed, it would be perfect to have an integrated process model considering both aspects, the development of the application core and the development of the user interface. Additionally, a common understanding of the role of the users would be perfect.

A joined process model of both approaches was published by Paelke et al. [7]. They called it Agile UCD-Process. (User-Centered Design was the predecessor of HCD.). The process model suggests to have a common initial phase for developers and HCI specialists. Afterwards there are activities of both groups. Unfortunately, it is not quite clear in which order these activities are performed. Additionally, the requirements elicitation is a little bit too much uncoupled from the software development process. A stronger coupling was suggested by Paul et al. [9]. It additionally provides the names of models that have to be specified in the corresponding phase of the software development like user or task model.

Two interleaving processes for developers and HCI specialists are suggested by Sy [15]. She suggests that at the beginning, there has to be a common plan and some user data have to be gathered. Afterwards, developers start in the first development cycle with implementations that are not much related to the user interface. This could be e.g. certain services of the application that are not related to user interface aspects. In parallel HCI specialists provide certain design solutions for cycle two and gather customer data for cycle three.

In cycle two developers implement the design solutions from cycle one and in parallel their code from cycle one is tested by HCI specialists. Additionally, they design for the next cycle and analyze for the cycle after the next cycle. This is the general development pattern. In some way, interaction designers work two cycles ahead to developers in analyzing customer data and one cycle ahead in developing design solutions.

A similar approach by separating the activities of analysts and developers was presented in [2] for the SCRUM approach. The development cycle of analysts is executed in parallel to the cycle of the developers. It runs at least one cycle ahead.

The suggested process model starts with an initial phase of all project members to get a common understanding. Later it is intended that the HCD process is executed in parallel to the development of the software. The HCD process should always be executed on cycle ahead of the development process. This can be reached by in such a way that developers start with configuration of the software development tools and with some features not related to the user interface.

Following Sy [15], both cycles have always the same length. This is also the way, companies we interviewed, work at the moment. However, this number of observed companies is very small and the companies are not representative. We also recognized, that they most of the time do not evaluate alternative designs. Most of the time there is one design solution only and this solution goes into the final software system.

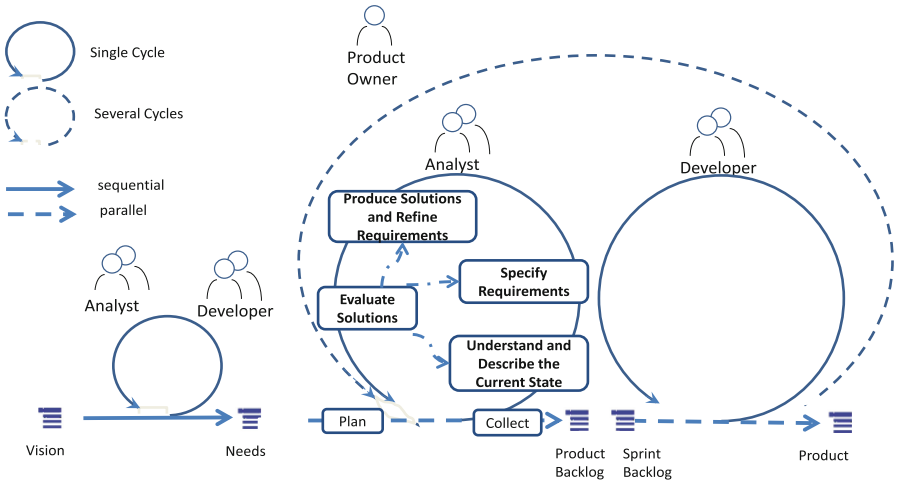


**Fig. 11.** Human-Centered Design Process for SCRUM.

Indeed, applying HCD methods is sometimes long lasting. The usage of questionnaires and interviews could sometimes not be possible within one sprint. In this way, the HCD process could last two, three, or more sprints. A synchronization of these activities might be the challenge for the future. A precise analysis of the requirements and an intelligent planning would be necessary for these cases. It has to be observed in the future, how companies behave, whether they pick up this idea or have activities of the same length.

There is also the question, when to stop with the development. The distinction between development and maintenance might not be useful anymore. Maintenance, should also be done in an agile way and fits to the process of Fig. 11. Continuous Software Engineering might be a solution for that. It can be characterized as combination of Software Engineering, Human-Centered Design, and Business Administration. An overview of integrating SE and UE can be found in [14].

## 4 Summary and Outlook

In this paper, we discussed the advantages of following a pattern-based approach in designing user interfaces. It allows the transformation of existing user interfaces based on the exchange of one pattern instance by another one. In a case study based on the websites Eurowings, Lufthansa, and Norwegian it was shown that pattern-based representations and transformations on different level of abstraction were possible.

It was shown that the structure of the website from Norwegian differs to the structure of the website of Eurowings by some pattern transformations only. The results of the transformations on an abstract and on a detailed refined level were presented. In this way, different version of a user interface can be generated easily without many efforts. Participatory design can be supported is supported very well by the application of the UI-Explorer 2 tool. Evaluations of different alternatives can be performed in an early stage of development. This can be done with abstract or already very detailed specifications.

Currently, the pattern-based creation of user interfaces has to be done manually. However, the model-based or model-driven development of such user interface specification was already shown (e.g. [18]) and should be combined with the UI-Explorer 2 in the future.

It is suggested to apply the UI-Explorer 2 tool in an agile development process that respects Human-Centered Design. Suggestion for a development process model for SRCUM were discussed. It was discussed, how such a process model could looks like, and whether sprints of the HCD process should last exactly one sprint or whether they can last for two or three sprints because of the needed time.

## References

1. Agile Manifesto. http://agilemanifesto.org/. Accessed 4 June 2015
2. Forbrig, P., Herczeg, M.: Managing the Agile process of human-centred design and software development. In: Beckmann, C., Gross T. (eds.) INTERACT 2015 Adjunct Proceedings, pp. 223–232 (2015)

3. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, Reading (1994)
4. Javahery, H., Seffah, A., Engelberg, D. and Sinnig, D.: Migrating user interfaces across platforms using HCI patterns. In: [13], pp. 241–259 (2004)
5. Kuusinen, K.: Task allocation between UX specialists and developers in agile software development projects. In: Abascal, J., Barbosa, S., Fetter, M., Gross, T., Palanque, P., Winckler, M. (eds.) INTERACT 2015. LNCS, vol. 9298, pp. 27–44. Springer, Heidelberg (2015)
6. Memmel, T., Gundelsweiler, F., Reiterer, H.: Agile human-centered software engineering. In: Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI…but not as we know it (BCS-HCI 2007), vol. 1, pp. 167–175. British Computer Society, Swinton (2007)
7. Paelke, V., Nebe, K.: Integrating agile methods for mixed reality design space exploration. In: Proceedings of the 7th ACM Conference on Designing Interactive Systems (DIS 2008), pp. 240–249. ACM, New York. http://doi.acm.org/10.1145/1394445.1394471
8. Paul, M., Roenspieß, A., Mentler, T., Herczeg, M.: The usability engineering repository (UsER). In: Hasselbring, W., Ehmke, N.C. (eds.) Software Engineering 2014 - Fachtagung des GI-Fachbereichs Softwaretechnik, 25.-28. Februar 2014, Kiel. Gesellschaft für Informatik e.V. (GI), pp. 113–118 (2014)
9. Paul, M.: Systemgestützte Integration des Usability-Engineerings in den Software-Entwicklungsprozess, Ph.D. thesis, University of Lübeck (2015)
10. David Ricardo Do Vale Pereira, Uirá Kulesza: Refactoring a web academic information system using design patterns. SugarLoafPLoP 2010, pp. 17:1–17:14 (2010)
11. Richard, J., Robert, J.-M., Malo, S., Migneault, J.: Giving UI developers the power of UI design patterns. In: Smith, M.J., Salvendy, G. (eds.) HCII 2011, Part I. LNCS, vol. 6771, pp. 40–47. Springer, Heidelberg (2011)
12. Saurin, M: Integration der Werkzeugunterstützung für die Anwendung von UI-Patterns in der agilen Softwareentwicklung. Master Thesis, University of Rostock 2016 (2016)
13. Seffah, A., Javahery, H.: Multiple User Interfaces - Cross-Platform Applications and Context-Aware Interfaces. John Wiley & Sons, Ltd. (2004). ISBN: 0-470-85444-8
14. Sohaib, O., Khan, K.: Integrating usability engineering and agile software development: a literature review. In: Proceedings of the International Conference on Computer design and Applications (ICCDA), vol. 2, pp. 32–38 (2010)
15. Sy, D.: Adapting usability investigations for agile user-centered design. J. Usability Stud. **2**(3), 112–132 (2007)
16. Tidwell, J.: Designing Interfaces. http://designinginterfaces.com/patterns/
17. Welie, M.: Patterns in interactive design. http://www.welie.com/patterns
18. Yigitbas, E., Mohrmann, B., Sauer, S.: Model-driven UI Development Integrating HCI Patterns. LMIS@EICS 2015, pp. 42–46 (2015)