Abstract. The Agile Method known as Scrum has been widely adopted in many companies due to its ability to handle scope changes and produce tangible deliverables at a fast rate. This article presents some limitations of Agile Methods and the results of using Scrum in a specific project. It was found that good practice in development of human computer interfaces have been left aside in favor of reducing sprint duration. The developers almost always "listen to what customers say instead of watch what customers do." This behavior has a tendency to negatively impact usability. This article proposes an Expert System to help developers design Human-Computer Interfaces.


1. Introduction

The use of Agile Methods in software development process is getting more and more supporters. They have emerged as an innovative development idea at the end of the 20th century in response to increasingly tight deadlines, the need for frequent requirements changes and problems of communication with customers. Agile Methods addresses these issues as they are based on flexibility, communication skill, self-organization, capacity to respond to changes and ability to offer new products in short periods of time [13].

Agile Software Engineering combines a philosophy and a set of guidelines for development. The philosophy encourages customer satisfaction and incremental software delivery. The guidelines emphasize on deliverables themselves as well as active and continuous communication between developers and customers [10].

To apply the Agile Methods is to develop a plan promoting adaptive evolutionary and iterative delivery, adding other values and practices that support agility and rapid response to change [4].

The Agile Software Engineering is a reasonable alternative to conventional Software Engineering for certain categories of software and design of certain types of software. It has been demonstrated that it delivers successful systems in very short time [10].

Several Agile Process models have been proposed satisfying to a greater or lesser extent the Agile Development Manifesto [10]. Among its main values one can find [2]:

- The highest priority is to satisfy the customer through early and continuous delivery of valuable software;
- Welcome changing requirements, even late in development. Agile Processes harness change for the customer's competitive advantage;
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale;
- Business people and developers must work together daily throughout the project;
- Build projects around motivated individuals. Give them the environment and support their needs and trust them to get the job done;
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation;
- Working software is the primary measure of progress;
- Agile processes promote sustainable development. Sponsors, developers, and
users should be able to maintain a steady pace indefinitely;
• Continuous attention to technical excellence and good design enhances agility;
• Simplicity – the art of maximizing the amount of work not done – is essential;
• The best architectures, requirements, and designs emerge from self-organizing teams;
• At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Certainly any developer agrees with many of the principles established in the Agile Manifesto, which can be proven in practice.

However one of the weaknesses of agile methods, as stated by Jakob Nielsen [7], is the quality of Human-Computer Interfaces (HCI).

A lot of work is being done to establish specific life cycles, methodologies and guidelines for HCI design. The lack of them in the Agile development framework is a concern.

One of the criticisms to the methods is that they are proposed by developers and primarily address the implementation in detriment of system design.

As a result, interaction and usability design and evaluation are left aside during coding [10].

This article presents some limitations of Agile methods, the results of Scrum implementation in a specific project and a proposal for an Expert System to help developers design Human-Computer Interfaces.

2. Limitations of Agile Methods

Some important points highlighted by Nielsen [7] for Agile Development, are:
• Location of members of developer and customer teams can cause several issues that do not exist when they are collocated;
• Assumption that customers are always available to schedule meetings, participate, solve questions and make decisions together with the development team, which is not always feasible in practice;
• Minimum documentation may appear too weak for another developer, making it difficult to reuse;
• Ignore a primary usability guideline: see what customers do, rather than hear what they say;

• Many issues arise during the implementation of interaction detailed design. Developers may not solve these issues in order to improve project usability;
• The product development is divided into smaller steps that are accomplished one at a time. This approach threatens to undermine the user conceptual component, as coined by Norman [8]. The user does not have the experience of integrating different features and therefore how to work with them.

Agile Methods focus on quick software development without detailing how its interface are specified.

The author had a chance to see these facts during an implementation following Scrum in a private company IT department.

3. Case study using Scrum

With the advent of Agile Methods, the IT department of the chosen company began a new process to manage and implement software using Scrum. Since then [1]:
• Productivity increased;
• Time to achieve results decreased;
• Effectiveness in serving the demands of deadlines improved;
• Customer intimacy improved;
• Requirements change management gets dramatically improved.

The application, a media manager, interface was developed by a web designer. However, as Ahumada mentions [1], there were no usability assessments. Prototypes, one of the most common practices in interface development, have been left aside in favor of reducing sprint duration. The task of analyzing color usage was the only task included in the specification.

Thus, during the implementation of this methodology, it was concluded that Scrum is a methodology focussing on project planning and monitoring with very little Software Engineering emphasis.

Not considering the Interface specification seemed to be a step back on employing "best practices on the development of interactive systems" [3].

Some tips from Jakob Nielsen [7] to minimize the negative impacts cited above were recommended for deployment:
• Perform usability activities, such as user testing, in a few days. One fruitful
approach is to plan for testing before you know exactly what will be available for testing. Weekly tests are completely feasible and give you a surefire way to integrate several user feedback rounds within even the shortest sprint.

- A 3-day course on how to perform a complete round of user testing by actually testing the team on its own project. You can do this type of quick testing in a day. And, it takes less than a day to both prepare the test and analyze its findings.
- Most successful teams have adopted a parallel track approach, where the user experience work is continuously done one step ahead of the implementation work. So, by the time such teams start to develop a feature, the initial user experience work on it has just completed;
- Finally, it’s needed foundational user research that goes beyond feature development. Ideally, organizations should conduct this work before a development project even starts. Also, bigger companies should house basic knowledge about user work flows, people, and usability guidelines outside individual projects so it can be reused for years across many projects.

Based on Nielsen last two suggestions, an idea of proposing an Expert System to support the construction of interfaces focusing on Agile Methodologies arose. The primary objective is to offer system designers practical results for the user interface design.

The purpose of the tool is to provide the knowledge of good design and development of interactive systems, as well as guidelines to support these steps.

4. The Expert System to support Agile Methods

The Expert Systems use knowledge stored at its knowledge base to solve problems. They are designed to be used in problems that require a considerable amount of human knowledge and specialization [12]. The importance of Expert Systems is to incorporate information of value, ability to make decisions and to estimate the sensitivity of their decisions [11].

The proposed Expert System aims to assist in the construction of interfaces during Agile Software Development. It should offer practical results for the design of the user interface, provide design best practices of interactive systems development, as well as policies, guidelines and formalisms to support these steps.

The system will establish heuristics for designing individual features depending on users and the system being built and should make reference to heuristics of usability. Nielsen in his book Usability Engineering [5] proposes ten basic usability heuristics [6] to be addressed by the expert system:

- Visibility of system status. The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
- Match between system and the real world. The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
- User control and freedom. Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
- Consistency and standards. Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
- Error prevention. Even better than good error messages is a careful design that prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
- Recognition rather than recall. Minimize the user’s memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
- Flexibility and efficiency of use. Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
- Aesthetic and minimalist design. Dialogues should not contain information
that is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

• Help users recognize, diagnose, and recover from errors. Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

• Help and documentation. Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

In general, it is necessary for the Agile Development designer to determine what aspects are the priorities for the specific project because it's virtually impossible to handle all aspects in a satisfactory manner.

The designer of HCI should question themselves in relation to how the user will interact with the application to perform their tasks or solve problems. It's necessary to establish a mechanism by which the Expert System identifies and characterizes a user's profile. Traditional categories [9] are:

• Related to the specific role within the organization, such as users of the operational, tactical or strategic levels;

• Related to the degree of expertise with computers being beginner and experienced extreme values of this measure;

• Related to the level of knowledge of the field of the application. Applications for novice users should show clear information on the state of interaction with tips for recovery. For an expert user it should offer sophisticated features that enable him/her to optimize the application;

• Related to the frequency of use of the application, e.g.: occasional or frequent;

• Related to the familiarity with a style of graphical interface like Windows, Macintosh, Motif, etc.

The purpose of the user analysis is to generate a profile of key users for each interface being designed. From the results of these modeling the Expert System will be able to identify the most important aspects to be considered in the design of the interface.

The architecture of the Expert System will include:

• Knowledge Base: the formalism used in the representation of knowledge is the production rules. This syntax is appropriate for the type of knowledge necessary to express the restrictions or conditions under which the design of the interface should be modeled;

• Interface: will consist of a series of questions to be used to determine relevant characteristics of the interface. According to [3] it’s necessary to conduct a user, tasks, and functional analysis, as well as identify interface components, dialogue and interaction space. Through various characterizations it’s possible to select the appropriate rules for the design of the interface being specified.

• Database: contains the knowledge of the design of the interface under specification as <Attribute, Value> which can be implemented such as the following predicate: user_class(IHC_design_nbr_1, skilled_user).

• Inference Engine: uses forward chaining to select the rules to be applied to the database. According to this strategy, the inference engine examines the antecedents of all rules and selects the one, which has every antecedent (that is the <condition> part) satisfied by the database facts.

5. Expert System construction

The development of an Expert System is a process that depends greatly on available resources and how these resources are organized and managed [12]. The development process is composed of a series of iterations as shown in Fig 1 [12]:

1. Planning: identify the domain, select the team and tool;
2. Knowledge Acquisition: identify, conceptualize and formalize;
3. Implementation: represent knowledge in the tool. Implement the interface suggested by the Expert System, generate the documentation.
The construction of the proposed expert system is a process that depends on the Knowledge Engineer, experience of specific knowledge on HCI design and modeling and experience with handling uncertainties.

6. Conclusion

The proposed expert system can also be expanded with scenarios that comprise a knowledge base to be consulted for the reuse of HCI designs as well as for learning good principles, rules and guidelines for HCI development. It’s known that HCI development tools cannot guarantee the quality of the product being developed. But they give the designer information necessary to make right decisions during the development process, especially when this process is part of Agile Methods so popular these days.

7. References