Dangers of Prototyping

Adrian Kosmaczewski

2008-08-15

Frederick P. Brooks Jr. has written about prototypes, saying that they are not only useful but strictly fundamental pieces of the overall software process, as in many other engineering activities. He gives the example of a pilot chemical plant, prepared to process 10'000 units per day instead of the 2 million units a day that the final plant would have to handle, in order to demonstrate the feasibility and uncover some unforeseen problems.

He summarizes his opinion in the famous phrase "plan to throw one away" (Brooks, 1995, page 116), underlining the problem of change management: managing change, right from the beginning of the project, instead of ignoring or avoiding it, is particularly important in software projects, since it presents a solid mindset for all stakeholders in order to avoid scope creep, schedule and staffing problems.

One of the proposed solutions to manage change (the "only constant thing") in a software project is the use of prototypes, which consist of UI mockups, showing basic interactions between the components and screens, allowing users to see in real time how the final software will look like, and serving as part of the feasibility study before creating the final product.

Several authors have highlighted different problems related to prototypes:

We consider three dangers to be most serious.

- Assumptions may be hidden (…)
- Feedback may be too expensive (…)
- Inappropriate human-computer interaction issues are highlighted

(Atwood et al, 1995, page 180)

One of the risk of creating a User Interface Prototype is accidentally raising unrealistic expectations about future progress on the project.

(McConnell, 1998, page 117)

Firstly, there are still plenty of users and managers around who think that the software is not much deeper than the user interface, and so if they see a nearly finished interface, they think the project is nearly done. (…) Secondly, engineers often get carried away about making a lovely user interface, and spend far too much time on it. (…) Lastly, and perhaps most importantly, is the fact that almost no code that is written as a 'temporary throw-away' actually gets thrown away. It usually gets used in the product in one way or another.

(Whitehead, 2001, page 254)

- Needs cooperation of management, developers, and users
- Managers may view prototyping as wasteful
- Managers and/or customers and/or marketing may view prototype as final product
- Programmers may lose discipline
- Prototype can be overworked (reason for prototype is forgotten)
- Prototyping tool may influence design
- Possibility of overpromising with prototype

(Hartson, 2001, page 5)

In my personal experience, I must say that I have not worked in many teams using prototyping extensively. Many times, when I have came up with the idea of doing a prototype, my managers (and even my coworkers) would dismiss it with the following criteria:

"We don't have time - money - staff - (add your own preferred variable here)" "OK, but do the prototype in such a way that we could reuse it later" "We do not lose time in prototypes in this company; we do real work here" "We do not have a budget for prototypes in the project" "We know exactly what the customer wants" "Oh, we've tried that once, but the customers never like what they see anyway, so why bother" "Our client does not want prototypes, period" "Our CEO does not want prototypes, period" "I do not want prototypes, period" "Our policy does not allow prototypes"

However, I must add that in just one case (a successful project I worked in three years ago), the analysts team managed to fit in the project budget the capability to create a set of user interface prototypes using Microsoft Visio. This tool (or any other diagramming tool) has many advantages over the traditional "Visual Basic"-based prototyping:

- Any user with Visio (be it analysts, developers, end users with basic Microsoft Office knowledge) can contribute to the prototypes, suggest changes and play with the mockups;
- No code is needed, which avoids developers to reuse it later;
- The output can be inserted (typically as images) in other documents, e-mails, etc.

This technique had a tremendously positive impact in the project:

- A great deal of feedback from the client was related to the mock-ups;
- The design documentation had an excellent complement of information, in the form of screenshots, coupled with the textual indication of every

possible action on the user interface;

- The developers could use these indications to reduce the uncertainty and deliver a product that matched the final decisions exactly;
- Since the prototypes were not “reusable” (they were after all only drawings on Visio) the developers could not be tempted to reuse the code in the final product.

In my opinion, this approach was one of the key factors of success of the project, but not the only one: the team members and the customer agreed that the project was critical and complex, and that a prototype could help everyone to understand it better. There was a specific mindset in the whole project, at both sides of the equation, which made prototyping a good option, and a success factor. Without this mindset, I do not think that prototyping will fit in any project.

References

Atwood, M. E.; Burns, B.; Girgensohn, A.; Lee, A.; Turner, T.; Zimmermann, B.; “Prototyping Considered Dangerous”, Interact ’95 Conference Proceedings, Pp. 179-184, NYNEX Science and Technology [Internet] http://www.fxpal.com/people/andreasg/interact95-2.pdf (Accessed June 24th, 2007)

Hartson, “Rapid Prototyping and Its Role in Development and Evaluation of User Interaction”, CS 5714 Fall 2001 - Usability Engineering, Virginia Tech [Internet] http://courses.cs.vt.edu/~cs5714/fall2001/notes/pdf/05_RP.pdf (Accessed June 24th, 2007)

Brooks Jr., F. P.; “The Mythical Man-Month - Essays on Software Engineering, Anniversary Edition”, 1995, Addison Wesley, ISBN 0-201-83595-9

McConnell, Steve, “Software Project Survival Guide”, Microsoft Press, 1998, ISBN 1-57231-621-7

Whitehead, R.; “Leading a Software Development Team - A Developer’s Guide to Successfully Leading People & Projects”, Addison-Wesley, 2001, ISBN 0-201-67526-9