Continuous Prototyping: Unified Application Delivery from Early Design to Code

Lukas Alperowitz[1], Andrea Marie Weintraud[2], Stefan Christoph Kofler[3] and Bernd Bruegge[4]

Abstract: Developing for devices like smartphones, tablets or smartwatches is more than just "shipping code". Especially in mobile development there is a strong focus on user interface design and user experience. In order to explore the design space, development teams and designers need early feedback from users testing the designs.

Continuous Delivery (CD) is a well-established technique for the delivery of software. In this paper we describe Continuous Prototyping which extends CD to cover the delivery of early artifacts like user interface mockups that usually do not benefit from an automated delivery process. Continuous Prototyping enables stakeholders to receive all artifacts through a unified delivery channel in fast cycles, from the first mockup to the finished product.

We developed PROTOTYPER as a tool to demonstrate the technical feasibility of Continuous Prototyping. PROTOTYPER allows developers and designers to deliver mockups, mobile applications as well as a mixture of both using the same deployment pipeline.

1 Introduction

Developing for devices like smartphones or smartwatches is more than just "shipping code". Especially in mobile application development user expectations regarding the user interface design and user experience are high. Therefore, development teams do not only consist of software developers. User interface and experience designers play an essential role in all phases of the evolution of a mobile application. In early project phases they provide interactive mockups to help the development team to define and refine the requirements. By providing design elements or mockups for a certain functionality, they contribute to new features in later phases of a project [Ru09]. These mockups help the software team to define the overall interaction of the user with the application [Be09]. Current mobile applications are leveraging multi-modal interaction techniques such as touch or speech input. They also often incorporate heterogeneous IoT devices like sensors and wearables. A mockup for such an application can be complex already in an early project stage.

[1] Technical University of Munich, Germany, alperowi@in.tum.de
[2] Technical University of Munich, Germany, weintraa@in.tum.de
[3] Technical University of Munich, Germany, koflers@in.tum.de
[4] Technical University of Munich, Germany, bruegge@in.tum.de

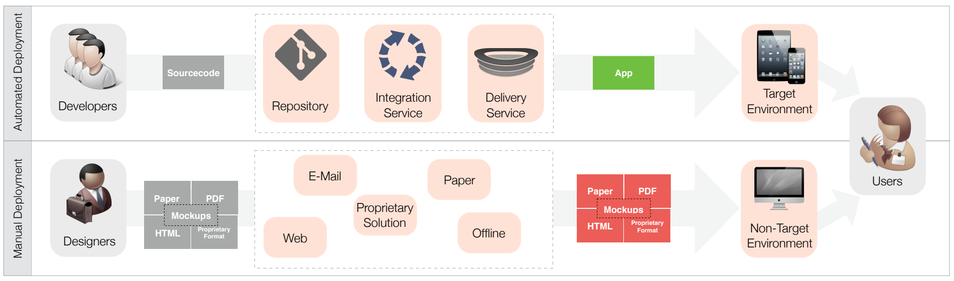


Figure 1: Current Practice: Software is delivered using a deployment pipeline, mockups manually and using various channels.

Although mockups are an important part of mobile application development, their delivery is given little attention. While Continuous Delivery (CD) is a well-established technique for the delivery of software, mockups are delivered manually using various channels ranging from e-mail, offline media to web-based solutions and come in various file-formats. Figure 1 illustrates this discrepancy.

We think the delivery of mockups should be as easy and automated as the delivery of software. We believe there is a need for a unified delivery concept for application development which covers all phases from the creation of mockups to actual software.

This paper introduces the concept of *Continuous Prototyping* to bridge the gap between the delivery of mockups and software. Continuous Prototyping allows designers and developers to deliver mockups as well as software using the same, unified deployment pipeline.

The remainder of this paper is structured as follows. In Section 2 we revisit the current research in the field of user interface prototyping and CD. We then define the term Continuous Prototyping in Section 3. In Section 4 we describe how we evaluated Continuous Prototyping using a design-demonstration called PROTOTYPER. We finally provide an outlook on our next steps in Section 5.

2 Background

2.1 Prototyping

There are several definitions of prototypes in software engineering and human-computer interaction. Bruegge and Dutoit refer to prototypes as simplified versions of a system, similarly to Guida et al. who state that ”[a] prototype is a dynamic model of the software system under construction” [BD10]. Prototypes are demonstrating parts of the final system in a simplified manner. They are often used for the exploration and elicitation of requirements.

Several purposes and benefits of prototypes are mentioned in literature: a prototype tests and demonstrates the feasibility of the functionality of a system while assessing possible risks; it provides a common basis for discussion between developers, users and other stakeholders; and it is useful to explore a project's requirements which are often not completely known upfront [Ru09][Ur92][Bu92][Be09][GLZ99].

In the domain of mobile applications, the term *mockup* is frequently used in place of or in relation to prototype. Within this paper the term 'mockup' shall refer to prototypes of applications that are not written in code.

Mockups can be analyzed along different dimensions: representation, precision, interactivity and evolution [BLM12]. The representation depends on the format, which might be on paper, created with a design tool or with an online service. According to the representation, the mockup can then vary in detail and can be interactive or non-interactive, evolutionary or revolutionary. Furthermore, mockups can be executable. We define a mockup to be executable if it can be executed on the target environment of a system to be developed.

There is a variety of tools to create mockups of different form, precision, interactivity and executability. Several design programs and online tools, such as *Illustrator*[1], *Sketch*[2], *Balsamiq*[3] and *Marvelapp*[4] are capable of creating interactive mockups. Some services offer the possibility to download and execute a mockup on a mobile phone using proprietary solutions. However, as shown in Figure 1, none of these services comprehends the delivery process as a whole of both mockups and software being executable in the target environment. The term target environment refers to the hardware and software environment the product in development will be deployed to when released to its end users.

2.2 Continuous Delivery

CD is a software delivery concept which fits, because of its iterative nature, well into agile software projects [HF10a]. It implies that the software being developed is always in a state where it can easily be delivered to the customer or end user – but also that software is in fact frequently and regularly released. With the practices Humble and Farley describe in their book "Continuous Delivery", they claim that "[s]oftware releases can – and should – be a low-risk, frequent, cheap, rapid, and predictable process" [HF10a]. In order to achieve this, the process of building, deploying, testing and releasing software should be largely automated and tracked [Na15][HF10a]. CD is based on appropriate tool support for building a deployment pipeline. A typical deployment pipeline consists of a version control system which keeps artifacts like source code or media items, an integration service which automates the build and test process and a delivery service which covers aspects related to deployment.

[1] `http://www.adobe.com/de/products/illustrator.html`
[2] `https://www.sketchapp.com`
[3] `https://www.balsamiq.com`
[4] `https://www.marvelapp.com`

2.3 Our Contribution

Although there are several connections in literature between prototyping on the one hand and agile methods on the other, there does not seem to be an explicit link between CD and prototyping. While CD is well established for the delivery of software built from code, the automated delivery of mockups is not covered yet. Indeed, during our literature research we could not find a source explicitly concerned with the automated delivery of mockups. We also found no source that analyzed the role of the delivery process during the transition from mockups to actual software. In the next section we discuss how Continuous Prototyping solves this by allowing development teams to use the same delivery process for mockups as they use for software.

3 Continuous Prototyping

Continuous Prototyping incorporates iterative prototyping as an approach to software development. Additionally, it adopts CD for the delivery of these prototypes, such as mockups. Continuous Prototyping provides an automated and repeatable delivery process for all kinds of product increments from mockups to applications. We believe that the benefits of CD claimed by Humble and Farley should be applied to the delivery of mockups: shorter development cycles, faster collection of user feedback and in the end higher quality of software products [HF10b].

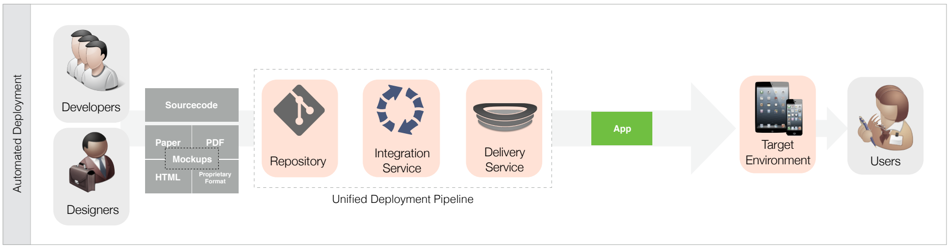


Figure 2: Solution: Continuous Prototyping - Deployment of software as well as interactive mockups using a unified deployment pipeline.

The foundation of Continuous Prototyping is a unified deployment pipeline, shown in Figure 2. Using this pipeline, artifacts like interactive mockups and actual software can be delivered to the target environment and accessed by users through the same channel. We believe that developers, designers and users gain the following benefits if they adopt Continuous Prototyping in their software projects right from the beginning:

Developers apply a certain set of workflows to structure their development activities like using a branching model in the version control system or a structured build promotion process for the delivery to different groups of users. Adopting Continuous Prototyping allows them to apply the same workflows for mockups as they are already using for apps. For example they could include the same user feedback system or usage analytics framework, starting from the delivery of the first mockup.

Designers can deliver executable mockups using the unified deployment pipeline. In collaboration with the development team they can prepare combined deliveries which include mockups and already implemented parts. The unified delivery pipeline allows *Designers* to apply concepts such as A/B testing to both mockups and mobile apps.

Testers and Users can access each new iteration, regardless of whether it is an early mockup or already implemented software, using the same delivery service such as an internal app-store. Starting from the first mockup, they can provide feedback using the same workflow for each release.

In the next section we present PROTOTYPER, a tool we developed to demonstrate the technical feasability of Continuous Prototyping.

4 Design Demonstration

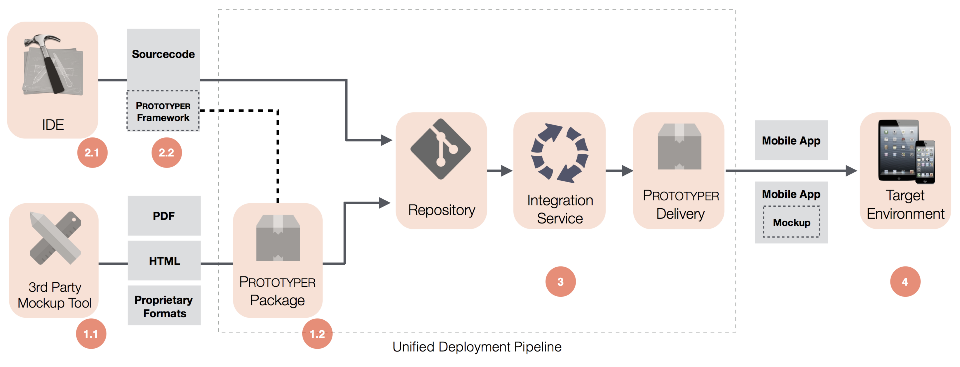


Figure 3: PROTOTYPER'S deployment pipeline architecture

PROTOTYPER is a software solution we developed in order to allow developers and designers to automate the delivery of both mockups and software. In this section we describe the architecture of the PROTOTYPER solution.

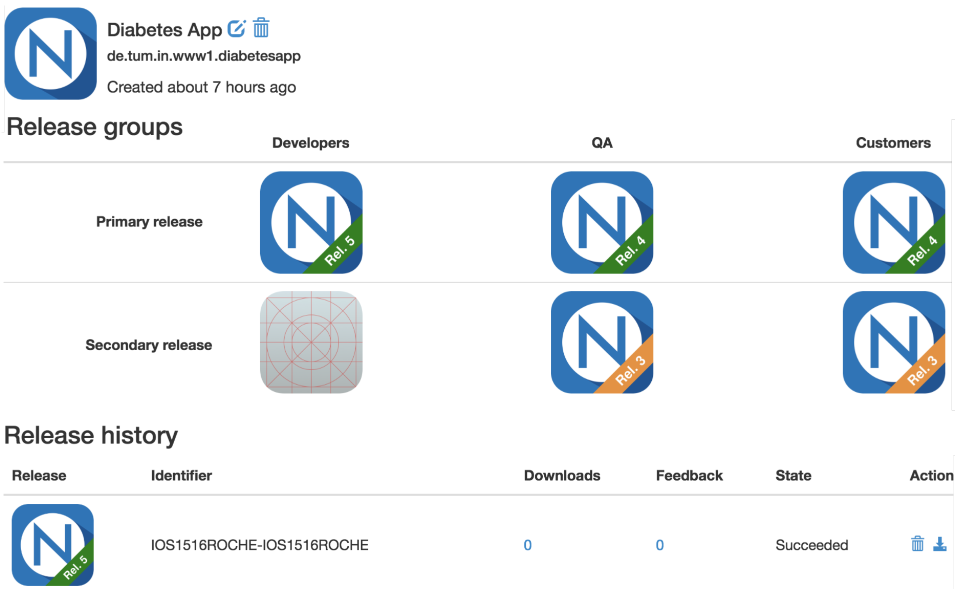
Figure 3 shows PROTOTYPER's overall architecture. PROTOTYPER allows software teams to deliver a user-interface mockup – created with a 3rd party mockup tool of their choice – as an executable mobile application. In order to achieve this, PROTOTYPER is integrated into a deployment pipeline, which in our example consists of a version control service, continuous integration service and a delivery service.

Within this pipeline, PROTOTYPER can operate in three modes:

PROTOTYPER PACKAGER *Designers* can use PROTOTYPER PACKAGER to transform mockups created using mockup tools into mobile apps. Figure 3 shows the components involved in this workflow. A developer or designer creates a mockup using a 3rd party tool (1.1). He uses PROTOTYPER PACKAGER to create an executable app out of the mockup

(1.2): Prototyper Packager first downloads e.g. a web-based representation of the mockup from the 3rd party tool. In a next step Prototyper Packager adds functionality for in-app user feedback and analytics to the downloaded mockup. Finally it creates a native mobile app matching the target environment of the project, such as an iOS app for the Apple iOS ecosystem. A developer or designer can then use the same deployment pipeline (3) that the software team uses for actual mobile apps to deploy the packaged mockup (4).

Prototyper Framework comes into place if a software team has already written parts of the application in code and creates a mockup for a new or redesigned feature that needs to be evaluated. Using Prototyper Framework, developers can create a package consisting of the mockup and a framework matching the target environment of the project. A developer can easily integrate this package into the existing software project using the IDE of the target environment (2.1). Using Prototyper Framework, he can now add a new part to the existing application which shows the new feature demonstrated in the mockup (2.2). After this step both parts, i.e. the parts of the application which are written in code and the mockup, are delivered using the unified deployment pipeline (3) and can be tested by a group of users (4).


Diabetes App
de.tum.in.www1.diabetesapp
Created about 7 hours ago

Release groups

	Developers	QA	Customers
Primary release	Rel. 5	Rel. 4	Rel. 4
Secondary release		Rel. 3	Rel. 3

Release history

Release	Identifier	Downloads	Feedback	State	Action
Rel. 5	IOS1516ROCHE-IOS1516ROCHE	0	0	Succeeded	

Figure 4: Prototyper's user interface for release group management (Excerpt). Mockup releases are orange, native apps green.

Prototyper Delivery is the third part of our Prototyper solution and the web-based app delivery solution developed. Prototyper Delivery allows the delivery of all kinds of applications: Pure mockups, applications mixed with mocked parts and applications developed only in code. An excerpt of Prototyper Delivery's user interface is presented in Figure 4 and presents shows functionality: Developers and Designers can define release groups, e.g. to target a Release to the QA-department or to a group of beta testers. Sometimes a Designer wants to deliver two different variants of a mockup to a group of testers in order to get feedback which one to choose. Using PROTOTYPER PACKAGER he can deliver two Releases at the same time to a release group. A tester can then install both simultaneously on his mobile device and compare them in the target environment.

5 Conclusion

In this paper we introduced the concept of Continuous Prototyping. Continuous Prototyping allows the automated delivery of user interface mockups as well as actual applications in fast cycles and using a unified deployment pipeline. We described the concept and showed PROTOTYPER, a design demonstration implementing the core principles of Continuous Prototyping in the domain of mobile application development.

The PROTOTYPER solution consists of three components: PROTOTYPER PACKAGER allows designers and developers to package mockups into executable mobile applications. PROTOTYPER FRAMEWORK allows teams to combine mockups and applications developed in code. With PROTOTYPER DELIVERY we implemented a uniform delivery pipeline for both, mockups as well as mobile apps.

As a next step we want to explore the impact of Continuous Prototyping on the communication between designers, developers and users. For instance, we will investigate how the more frequent and rapid delivery of mockups influences the quality of the developed software product. With regard to the PROTOTYPER solution, we will add an analytics component to improve the area of user feedback by e.g. automatically collecting contextual data or recording user interaction steps for a more complete picture of the usage context. We also plan to support additional target platforms like web-based environments. Finally we will evaluate PROTOTYPER in an industrial setting.

We believe that applying the concept of Continuous Prototyping will have a lasting benefit on the collaboration between designers, developers and users.

References

[BD10] Brügge, Bernd; Dutoit, Allen H.: Object Oriented Software Engineering Using UML, Patterns, and Java. Prentice Hall, 2010.

[Be09] Berenbach, B.; Paulish, D.; Kazmeier, J.; Rudorfer, A.: Software & Systems Requirements Engineering: In Practice. McGraw-Hill Education, 2009.

[BLM12] Beaudouin-Lafon, Michel; Mackay, Wendy E.: The Human-Computer Interaction Handbook. CRC Press, chapter Prototyping Tools and Techniques, pp. 1081–1104, 2012.

[Bu92] Budde, Reinhard; Kautz, Karlheinz; Kuhlenkamp, Karin; Züllighoven, Heinz: Prototyping. Springer-Verlag Berlin Heidelberg, 1992.

[GLZ99] Guida, Giovanni; Lamperti, Gianfranco; Zanella, Marina: Software Prototyping in Data and Knowledge Engineering. Springer Netherlands, chapter The Prototyping Approach to Software Development, pp. 1–32, 1999.

[HF10a] Humble, Jez; Farley, David: Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Pearson Education, 2010.

[HF10b] Humble, Jez; Farley, David: Continuous delivery: reliable software releases through build, test, and deployment automation. Pearson Education, 2010.

[Na15] Narayan, S.: Agile IT Organization Design: For Digital Transformation and Continuous Delivery. Pearson Education, 2015.

[Ru09] Rupp, Chris: Requirements-Engineering und -Management. Carl Hanser Verlag München, 2009.

[Ur92] Urban, Joseph E.: Software Prototyping and Requirements Engineering. Technical report, Arizona State University, 1992.