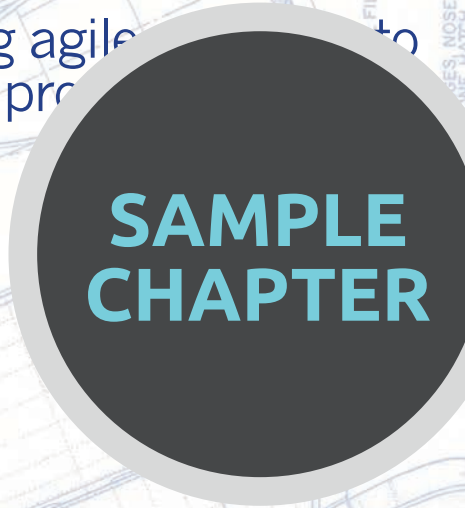




Co-Pilot

using agile
land IT pro



**SAMPLE
CHAPTER**

COLORING NOTES:
AIRCRAFT PORTRAYED
TIRE AIRCRAFT HIGH POLISHED NATURAL
SURFACES SILVER
LIGHT BLUE NAVY BLUE PIN STRIPE
DOTTED LINE INCLUDES BEECH BIRD STRIPE
FUSELAGE, WING DESIGN AND
LICENSE NUMBERS AND VERTICAL
DESIGN, ENGINE COWL TRIM.
BLACK

STA. 96

FUEL
FILL

LANDING FLAPS
ELECTRICALLY
OPERATED

FABRIC COVERED
AILERONS AND
FLAPS, DOPED
SILVER

AND ROUND

FLYING
LINES
THREE
ALL
ONLY
ONE

PRE WAR MACELLE
OUTLINE
COSTUME

POWER, TWO PR... & WHITNEY R-985 WASP, JF. ENGINE
450 h.p. max., DIRECT DRIVE

SPECTIC

ES
ACTABLY
NG LIG

K RUBER
BOOT
EN EMP
CH RUBI

ED

HCR
del i

183-4
WAR M

ART
PAUL R. I
SCALE: ORIGIN

DETAILED & INKE
SCALE: ORIGIN

FILL

SCALE: ORIGIN

Impressum

Managing Editor:

Till Bay
Comerge AG
Bubenbergstrasse 11
8045 Zurich
Switzerland
Tel: +41 43 501 38 38
Fax: +41 43 501 38 39
Email: till.bay@comerge.net

Associate Editors:

Bernd Schoeller
Benno Baumgartner
Beat Herlig
Dennis Rietmann
Michela Pedroni

Layout and Artwork:

Eva Jutzeler

all rights reserved
© Comerge AG

Co-Pilot

using agile methods to
land IT projects smoothly

Introduction

There are many books on agile methods and software development best practices available today. So why write another book on the topic?

There are a number of reasons: First of all, we do not believe in taking a book and following it word for word. We rather trust in action and daily practice. The book you are holding in your hands describes a set of process patterns that have worked well for us in the past. By constantly remixing and improving them, we ensure that they continue to fit the needs of our evolving company. In fact, the Co-Pilot includes a framework for adapting the process to changes, thus driving it further as we write. Hence, this book does not describe a specific menu to cook in the software kitchen, but it describes a list of ingredients and spices that can be mixed and matched.

The second reason for turning our practices into written form is: the ideas described in the Co-Pilot may be reused by other companies in other environments applying a similar mix and match approach. The way of working defines and significantly influences a company's culture, but it can only become culture if you are convinced of the practices. We hope that the processes and methods described in the Co-Pilot will entice you to choose some and invent your own flavor of an agile software development process to be implemented at your company. The name of this book — Co-Pilot — reflects this idea. It serves as guidance and inspiration and provides inputs, but you remain the pilot in charge.

The processes and ideas that we describe have roots in the history of our company, Comerge. Comerge grew in its first ten months from four employees to over twenty. During this rapid evolution, we were confronted with various managerial and technological challenges.

Coming to a slower growth pace in recent months (we are now almost three years old) motivated us to look back and identify the ideas that helped us master these challenges while still satisfying the needs of our customers. Many of these ideas have their roots in our experience as employees or co-workers of inspiring proponents of agile software methods: Benno Baumgartner and Till Bay have worked at Erich Gamma's lab, five of us worked for Bertrand Meyer at the Chair of Software Engineering at ETH Zurich, and others have managed outsourcing projects with programmers located in Eastern Europe. The Co-Pilot collects these experiences and the best practices of our past and present.

We hope that you enjoy the read and that it inspires you to put some of it into action at your company.

We would like to thank the following people for making this book possible: Daniela Bomatter, Julian Tschannen, Geraldine von Roten, and Urs Doenni for their reviews of the drafts, Eva Jutzeler for her fantastic layout and drawings, and all our customers and employees for their constant support and encouragement. You are great and together we make the shipping of high quality software happen and safely land agile software projects.

November 2010

Till Bay, Bernd Schoeller, Benno Baumgartner, Beat Herlig, Dennis Rietmann, and Michela Pedroni

Table of contents

Projects

Our values	6
How to read	8
Process roles	12
How to plan a project	14
How to iterate	17
How to work with issues	19
How to triage issues	26
How to create a release	THE SAMPLE CHAPTER 29

Meetings

How to hold a meeting	34
How to hold a scrum meeting	36
How to participate in a core meeting	38
How to hold a retrospective meeting	40

Tests

How to test	44
How to unit test	46
How to smoke test	48
How to test a release	50
How to do acceptance testing	52

56

How to work with customers

57

How to do customer support

61

Glossary

Customers

How to create a release

A release is a piece of software that customers receive. Because customers use it to verify that the project is well on track, it must be of high quality. Being able to release at any given moment in a project is the number one success factor of an agile team. To do so, it is essential to automatize the building and deployment process.

Testing and fixing days

Deploy release candidate

The most important thing when creating a release is to test the system under end-user conditions. This means that a build is deployed on the system (or at least an exact copy of the system) on which it will go live. In an ideal world, the release candidate will become the release without any changes.

Write test plan

In parallel to the deployment of the release candidate, the teams write the test plan. Usually, one team member per team takes over this task. It is important to gather input about the test items from all stakeholders that worked on the release candidate. The product manager notifies the testers in advance to ensure that they are available during the testing days. She may also invite customers on site to participate in this activity. At the end of this chapter, you can find a test plan template.

Test release

During the testing days, the deployed release candidate is tested according to the test plan. This may take several days. No code changes are allowed during



A test plan listing the test items and which testers have tested them.

this time, unless a blocking issue prevents testing. The testers describe every failed test in an issue report and file it against the team responsible for the code causing the test to fail. See “How to test” for details about release testing.

Triage issues meeting

The team leader organizes a meeting and the team triages the new issues. The goal of the meeting is to let the team leader decide, if the tested release candidate has an acceptable quality. If so, he signs off the build. Signing off a build means that the team leader accepts all parts of the build, not just his own component. By refusing to sign off a build, a team leader can veto a release.

If all team leaders sign off, the release takes place. Otherwise, the teams need to fix the identified bugs, create a new release candidate, verify the fixes, and decide again if the quality is acceptable. This may, potentially, take forever. But usually this only happens once, sometimes twice.

It is important that the product manager defines what acceptable quality means for the current release. This highly depends on the kind of release. The acceptable quality of a milestone differs from that of a product release. The product manager may, for example, define that the release must not contain any open major issues. Usually, these quality requirements are directly derived from the contract made with the customer. Acceptable quality never means flawless. The decision what to fix and what not to fix is a very hard problem and depends on many factors. A rule of thumb is that the higher the chance that the fix for an issue will introduce a new bug the less likely it is that the issue will be fixed during the fixing days.

Fixing days

During the fixing days, the team members fix all issues which have been planned for the current release during the “triage issues” step. Each fix needs an issue report. This allows to verify the fix later on. To fix an issue, the team leader needs to give his approval. At least one review is required for each code change. The developer of the fix can choose any team member for the review. Fixes are not committed, but sent to the reviewer as a patch file. Only after the issues are reviewed by the reviewer, they can be committed.

Verify fixes

All issues fixed during the fixing days are verified in the newly deployed release candidate. Verifying means, that a team member other than the one who fixed the issue tries to reproduce the issue. If he cannot reproduce it, the issue is verified. Otherwise, the issue has to be reopened and fixed again.

Release

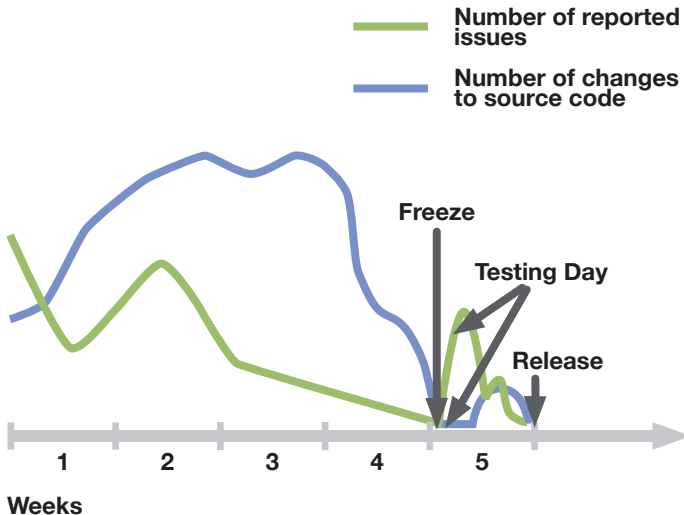
Once the release candidate is signed off by all team leaders, the release takes place. In a perfect world, the release candidate becomes the release without any changes. In reality, the deployment usually requires configuration steps, such as copying the release candidate onto another machine or disabling access control to make a web site publicly available. After this has been done, the product manager may inform all stakeholders about the release. A document containing what is new and noteworthy can support this.

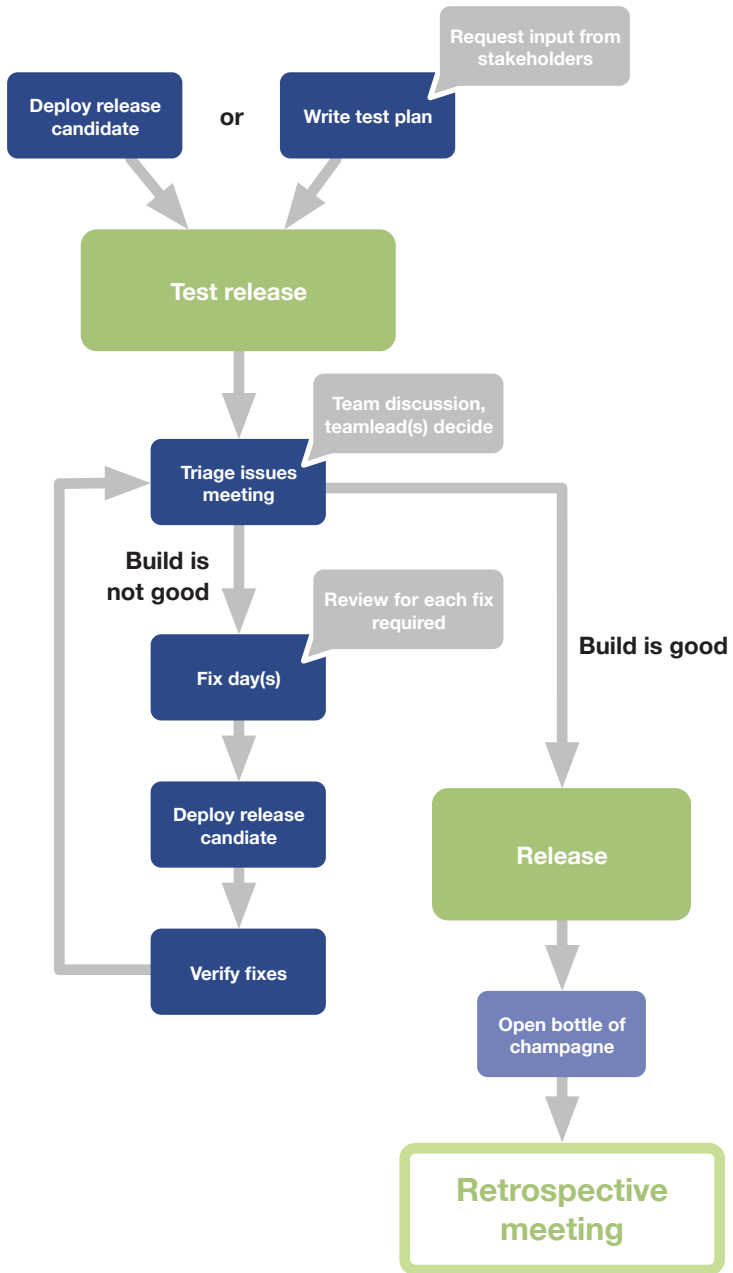
Open bottle of champagne

After the release it is important to throw a little party, have a beer together, and give the team a couple of days to relax. This is called the decompression phase.

Retrospective meeting

Hold a retrospective meeting, see “How to hold a retrospective meeting” for details.







Comerge AG
Bubenbergstrasse 11
8045 Zurich
Switzerland

Tel: +41 43 501 38 38
Fax: +41 43 501 38 39
E-Mail: info@comerge.net

<http://www.comerge.net>