COMPUTER LAW & SECURITY REVIEW 000 (2018) 1–8

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/CLSR

Computer Law & Security Review

Agile programming – Introduction and current legal challenges

Thomas Hoeren[a,*], *Stefan Pinelli*[b]

[a] *Institute for Information, Telecommunication and Media Law, University of Münster, Münster, Germany*
[b] *Attorney at Law, Volkswagen AG, Wolfsburg, Germany*

ARTICLE INFO

Article history:
Available online xxx

Keywords:
Agile programming
New distribution of roles
Product vision
Sprint process
Liability

ABSTRACT

Within the realms of software development, customers must specify the requirements of their new software before the start of the project. Today, this leads to considerable delays with respect to the start of the project. In addition, the integration of new requirements into a system already developed in parts is becoming increasingly time-consuming and cost-intensive. Yet the specifically necessitated functions of a software are often only revealed through the process of development. By means of agile programming, changes in the requirements of a software product can be handled flexibly in shorter development cycles. In the following, the framework of agile software development projects as it applies under German law is described and current legal problems of such projects – in particular, the issue of contract type and the new building contract law – are considered. The unplanned project design appears contrary to the legal approach. The article shows, however, that agile software products development provides customers with dynamic and quickly scalable products and that customers can leave the project after individual project steps. The new development of building contract law, which focuses on subunits and approvals, is also very much in line with the above-mentioned programming.

1. Introduction

Many software development contracts traditionally follow the typical waterfall model[1] in its sequential approach based on long development cycles in which the individual sub-project steps viz. "requirements analysis, design, programming and testing" are completed one after another. However, this creates many legal and other challenges. These are caused by the fact that the model assumes that each project phase can be completed before the start of a new phase. Sequential processing leads to new process flows being started when changes in customer requirements arise. On the part of the customers, process failure is therefore regarded (only) as a breach of contract.

This is where the logic of agile programming comes in. Agile programming refers to the model of a software development process, in which progress is unpredictable and can always be threatened by changes or disruptions. Accordingly, the model features a more flexible process and builds in failure as a risk.

The following table sets out the main differences between programming following the traditional waterfall-model and

* Corresponding author at: Institute for Information, Telecommunication and Media Law, University of Münster, Leonardo-Campus 9, D - 48149 Münster, Germany.
E-mail address: hoeren@uni-muenster.de (T. Hoeren).

[1] M. Schmidl, IT-Recht von A-Z, 2. Auflage 2014, p. 285 ("Wasserfallmodell"); also K. Borkert, in: Taeger, Tagungsband DSRI-Herbstakademie 2013: Law as a Service (LaaS), p. 927.

https://doi.org/10.1016/j.clsr.2018.04.004
0267-3649/

Please cite this article as: T. Hoeren, S. Pinelli, Agile programming – Introduction and current legal challenges, Computer Law & Security Review: The International Journal of Technology Law and Practice (2018), https://doi.org/10.1016/j.clsr.2018.04.004

agile programming with reference to the Scrum-model. In the following, the single elements will be discussed in detail.

Traditional programming (waterfall-model)	Agile programming (scrum-method)
Long delivery cycles	Small development units
Complete and consistent software solution is delivered	Constant delivery of small, independent parts of software
Long-term phases (requirements analysis, design, programming, testing)	Short-term sprints
Strict flow chart	Flexible work flow
Traditional roles: customer and contractor	New roles: product owner, development team, scrum master
Targets predefined by customer	Close interaction and ongoing adjustments
Progress is measured by reference to the targets	Progress measuring units have to be defined

In the course of the development of agile programming methods, another trend named "DevOps" (Development and Operations) has emerged. While agile programming is a specific way to create software, DevOps aims at changing entire business structures. Traditionally, separated departments (software development and IT Operations) form a common team to program software more suitable to business operations and to accelerate the entire development process. Often, agile programming methods are used in DevOps projects. The specific legal issues related to DevOps, especially in the case of different business undertaking a common project, are complex; however, they go beyond the scope of this paper. Therefore, it will focus on agile programming methods and point out the possibilities and challenges from a German legal point of view.

2. What is agile programming?

2.1. *The agile manifesto*

In 2001, several software developers published the Manifesto about agile programming.[2] The twelve principles include in particular:

- Customer satisfaction is best achieved through timely and continuous delivery of valuable software.
- Working software is the first goal of development and should be delivered regularly within short time frames.
- Changing requirements are welcome, even late in the development process.

[2] http://agilemanifesto.org/iso/de/manifesto.html (last visited Feb. 16, 2018); M. Cohn, Succeeding with Agile: Software Development Using Scrum, 2010. For studies of different methods of agile programming see the overview given by F. Koch, in: Der IT-Rechts-Berater [ITRB] 2010, pp. 114, 115 f.

2.2. *Different forms of collaboration in traditional and agile projects*

In agile projects, developers and business teams interact very closely. To that effect, the project is divided into small development units (called repetitions or sprints).[3] Each sub-project consists of an independent development of design, coding and testing within two to four weeks. Every result of a self-contained partial solution must be usable for itself. The aim should be to develop first the basic functionalities, which are most important from the customer's point of view, while comfort functionalities can be developed later on. This is intended to ensure that the customer receives a (partial) product, which can be used at an early stage. Thus, the client or a third party can further develop the result of an iteration based on the source code.

Under German law, the meaning and purpose of a specific software development contract has to be considered in terms of the general aim of such a contract. In particular, the purpose of the contract and its implementation in an executable product call for clarification. Waterfall-projects often focus on the supplementary question of the project's failure and its legal consequences. Accordingly, the law governing contracts to produce a work in the meaning of section *631 ff. of the German Civil Code* (BGB) is not suitable for IT projects. It is based on a framework of success that has been defined from the outset and is checked and confirmed, if possible, at the end of the project. *Nicklisch* has already pointed out that, under German law, IT projects are complex long-term contracts that are not covered by the grid of traditional contract law.[4] Traditionally, the functional specification and definition of the contract purpose and its implementation, as a continuous daily process, are neglected.[5] This is where agile programming comes into play, in which the key figures of the project, the structure of deadlines and the elements of project documentation are worked out more clearly.[6]

2.3. *The jurists' tendency to define a contract type*

It is wrong to assume that agile projects correspond with a certain type of contract. Rather, depending on the purpose of the contract, possible flexibility must be combined with sharp contours in terms of acceptance and pricing in general. Not all activities need to be carried out in an agile way; one might also

[3] Also K. Borkert (supra note 2), pp. 927, 930; M.-J. Buchholz, in: ZD-Aktuell 2013, 03170.

[4] F. Nicklisch, in: Richterliche Rechtsfortbildung, Festschrift der Juristischen Fakultät zur 600-Jahr-Feier der Ruprecht-Karls-Universität Heidelberg 1986, Technologierecht und Rechtsfortbildung, pp. 231, 237; F. Nicklisch, Komplexe Langzeitverträge für neue Technologien und neue Projekte, Heidelberger Kolloquium Technologie und Recht 2001, 2002; concerning the complex long-term contracts F. Nicklisch, in: Neue Juristische Wochenschrift [NJW] 1985, pp. 2361 ff.; C. Zahrnt, in: Computer und Recht [CR] 1992, pp. 84 ff.

[5] The organization of the project without prejudging the outcome is inherent in the system; see M. Witzel, in: CR 2017, pp. 557 ff.; P. Hoppen, in: CR 2015, pp. 747 ff.

[6] J. Schneider, in: ITRB 2010, pp. 18, 20; C. Frank, in: CR 2011, p. 138.

Please cite this article as: T. Hoeren, S. Pinelli, Agile programming – Introduction and current legal challenges, Computer Law & Security Review: The International Journal of Technology Law and Practice (2018), https://doi.org/10.1016/j.clsr.2018.04.004