COMPUTER LAW & SECURITY REVIEW 34 (2018) 1131–1138

ELSEVIER

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:*

*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/© 2018 Thomas Hoeren and Stefan Pinelli. Published by Elsevier Ltd. All rights reserved.

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.

combine customizing with agile development. Therefore, it is important, before the start of the project, to determine which subproject should be carried out agilely and which should not. Accordingly, it is advisable not to enter into the discussion about service contracts or contracts for work and labour from the outset. This is so, even if this kind of thinking in such templates belongs to the jurist's genetic disposition. Those classifications play a major role in the later course of the project.[7] It may also be a good idea then to combine both grids flexibly for the form of the contract, such as a service contract framework, so that some services may be conceived based on a contract for work and labour. As it will be shown below, the 'Definition of Done' offers the opportunity to combine a service contract as framework with reduced contracts to produce a work at the level of the single sprints. The differentiation is only important if the parties use standard contracts that are subject to general standard terms and conditions in accordance with *section 307 (2) no. 1 BGB*. On the other hand, there is almost unlimited contractual freedom for individual contracts.

### 2.4. *Clarity*

It is questionable under German law whether the parties should provide such a definition on a regular basis. At the very least, they have to decide whether it is a service contract or a contract to produce a work. In its decision of July 23rd 2009 (silo plant case), the German Federal Supreme Court of Justice decided in favour of a contract to produce a work, because the predominant planning phase was evidence of a contract to produce work.[8] This also prevents the application of *section 651 BGB*.[9] Thus, the extensive planning work involved in an agile project speaks in favour of applying a warranty for defects based on a contract to produce a work and, above all, considering acceptance as an important element.[10] In this case, something needs to be said about tests and reviews. A corresponding agreement must also include compliance with all programming standards, ensuring a complete documentation, which is necessary. It is a mistake to think that agile projects can do without any documentation at all. The Higher Regional Court of Frankfurt has rightly pointed out that a documentation of the system architecture must generally be available early in the project lifecycle when the result of the project has become so fixed that a correction of the system architecture is no longer anticipated.[11] If the parties dispute the fulfilment of such definitions, a dispute resolution mechanism is appropriate. The project is finished when all aspects of the backlog (so-called 'backlog items') are developed and finished, according to the 'Definition of Done'. The list of tasks at the beginning is quite different from the list at the end of the project.

## 3. New distribution of roles in the agile project

### 3.1. *The fundamental roles determine the agile programming, especially in the scrum model*

#### 3.1.1. *Product owner*

The "owner" of the product ('product owner') is the customer's key representative, even if he is not identical with the client.[12] He is responsible for successfully communicating the vision of the customer and his requirements to the development team.[13] He also decides whether a backlog entry was completed during a sprint. He creates, organizes and manages the product backlog and permanently prioritizes/revises it during the project, according to the customer's requirements. He also participates in all meetings of the development team during each sprint. He is appointed by the customer and has access to all relevant stakeholders of the customer.[14] It is important that his qualification for dealing with Scrum Projects is clear from the beginning.[15] The owner needs sufficient freedom regarding time and professional matters to perform his duties and is obliged to react promptly to questions from the development team.[16]

#### 3.1.2. *The development team*

The next step is to define the role of the development team.[17] It is responsible for the current development work and the delivery of the increments within each sprint. The team must have the necessary competence to work with Scrum Projects and has to be open to members with experience in coding or testing.[18] Sometimes the customer's technicians do not have the necessary technical skills for an efficient collaboration. Furthermore, their "interference" is a major legal challenge in

[7] This is also a vulnerability of the decision of the LG Wiesbaden [Regional Court Wiesbaden] Nov. 30, 2016, MMR 2017, pp. 561 f. The court saw the negotiated draft contract as an indication of the classification as a contract to produce a work (in the meaning of section 631 BGB). The OLG Frankfurt [Higher Regional Court of Frankfurt] (see case cited infra note 12) rightly emphasizes that the classification of the contractual relationship may be leaved undetermined, since the claim is due based on a clear payment agreement. In this case, it does not matter, if service contract law (section 611 ff. BGB) is applied or contract law of producing a work (section 631 ff. BGB).

[8] Bundesgerichtshof (BGH) [Federal Court Justice] July 23, 2009, NJW 2009, pp. 2877 ff.

[9] BGH [Federal Court of Justice] July 23, 2009, NJW 2009, pp. 2877 ff.; C. Heinemann, in: Taeger, Tagungsband DSRI-Herbstakademie 2016: Smart World – Smart Law?, pp. 633, 646.

[10] A. Fuchs/C. Meierhöfer/J. Morsbach/L. Pahlow, in: MMR 2012, pp. 427 ff.

[11] OLG Frankfurt [Higher Regional Court Frankfurt] Aug. 17, 2017, CR 2017, pp. 646 f.

[12] Borkert takes this as an indication of the existence of a service contract in the meaning of sections 611 ff. BGB, see Borkert (supra note 2), pp. 927, 933.

[13] Important is that neither the product owner nor the scrum master issue an instruction, see A. Friedl/D. Heise, in: Neue Zeitschrift für Arbeitsrecht [NZA] 2015, pp. 129, 133.

[14] A. Friedl/D. Heise, in: NZA 2015, pp. 129, 133.

[15] Doubts with regard to the qualification especially of inexperienced users in: A. Bussche/T. Schelinski, in: Leupold/Glossner, Münchener Anwaltshandbuch IT Recht, 3. Auflage 2013, part 1 para. 95.

[16] See A. Hengstler, in: ITRB 2012, pp. 113, 115.

[17] It should be noted, that the influence of the principal on the composition of the team may result in sanctions according to the Temporary Employment Act (Arbeitnehmerüberlassungsgesetz, AÜG).

[18] A. Friedl/D. Heise, in: NZA 2015, pp. 129, 131.

view of making a clear assignment of risks and responsibility between customer and developer. Therefore, development teams are often mixed together to provide all the necessary capabilities to deliver the increment. In any case, the contract must specify how the customer is involved in the composition of the development team. It is essential that the agile set of rules and the procedures contained are communicated to the whole team so that these rules flow into the development work on a daily basis. If the parties cannot commit themselves to a team within a fixed period, it has to be possible to terminate the contract without liability risks. It also has to be regulated that each member of the development team is fully involved in the project during the term of the project and cannot be dismissed without the prior written consent of the customer (however in respect of labour law restrictions).

### 3.1.3. *Scrum master*

The Program Master or Scrum Master has the actual key role in the project; his tasks are similar to those of a coach. He ensures that the development team and the owner of the product cooperate according to the project plan.[19] He is not a product coordinator, but only supports the owner of the product and the development team.[20] Contractually, it is possible to select the Scrum Master from the development staff or to involve a third party independent of the customer or the developer.[21] Key tasks of the Scrum Master have to be regulated in the contract. The independence of the Scrum Master as well as his or her availability during the project duration also have to be guaranteed.

## 4. The product vision

As a starting question, only the product vision is described. The corresponding statement describes the overall purposes and objectives of the project. A corresponding document should therefore at least be included in the development contract at all times. This is not to be confused with the product backlog, namely a description of the requirements to be developed. Structured according to importance, individual requirements are prioritized here.[22]

The individual characteristics of the program to be developed are described and its economic significance is estimated, as well as the time required. Then a priority is set for each task. Often user stories as the following are created: "As a user, I want the feature X, so I have the advantage Y." The aim is to keep the description short. The priorities must be described as clearly as possible and are reviewed by the product owner regularly. An estimated 5–10% of a sprint should be used to redefine collectively the product backlog. Because product backlog items are often different in complexity, measuring units have to be found. Normally, sprints are always the same length and therefore equally expensive. However, the number of backlog entries can vary depending on the effort involved. Experience has shown that development teams (assuming continuity of people) are also getting faster and more efficient and therefore are able to do more work later than at the beginning. It is advisable to develop and discuss the product backlog in an initial workshop between the product owner and the development team. The initial product backlog should then be used by the development team to provide the product owner with an estimate of the effort required (see below). Estimates are made in accordance with the contract with reasonable care and based on fair assumptions. When the cost estimate is completed, the product owner should be obliged to prioritise each task based on effort and business value. The parties should be obliged to redefine the product backlog regularly during a sprint in a workshop, if necessary. The product owner is also free to change the product backlog at any time. Exceptions are unilateral changes in the cost estimate or priority areas during a sprint.

It is questionable whether the application of *sections 313–315 BGB* is helpful. Too much is based on the one-sided provision of services by one of the parties and/or according to fairness here. Thus, in a contract according to *section 313 BGB*, too little is communicated, but rather inflicted unilaterally. Legal practitioners should be as flexible as possible in order to prevent an agile project from being strangled by legal constraints from the outset. However, it should be noted, as shown below, that the pricing modalities should be regulated more precisely, especially in relation to the purchase price.

For the customer it may seem invidious that he has to make out a blank cheque for the project costs. Therefore, a system of objectives must be included in the contract. Pricing models include a fixed price per user story.[23] There is a conflict between the need for clear descriptions of performance and flexibility in agile projects. It is recommended to link price payments to the achievement of certain software functionalities, which can be determined by means of function-point analysis, for example.[24] It is important to define clearly the initial requirement and to follow it up with a checkpoint phase, in which decisions are made on the further implementation of the overall project after initial implementation. Only then can fixed prices and acceptances be determined in a contractually binding manner.[25] In case of complex projects, a specification phase can also be provided after the project application phase and before the offer phase, which leads to a preliminary specification and thus also to more realistic fixed prices.[26] The planning of a feasibility study, followed by a contract phase is similar; the feasibility study (so-called Sprint Zero) can be usefully located in the service contract law, while all other solutions are designed as a contract to produce a work.

[19] A. Friedl/D. Heise, in: NZA 2015, pp. 129, 130 f.

[20] K. Schwaber/J. Sutherland, Der Scrum Guide, Nov. 2017, p. 9, http://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-German.pdf (last visited Feb. 16, 2018).

[21] Usually, the Scrum Master is part of external staff and contractually connected with the client.

[22] K. Schwaber/J. Sutherland (supra note 21), p. 16.

[23] The fixed price therefore plays a central role in agile projects, see A. Opelt/B. Gloger/W. Pfarl/R. Mittermayr, Der agile Festpreis, 2012, p. 31.

[24] Concerning the Function-Point-analysis in accordance to ISO/IEC 20926 see Borkert (supra note 2), p. 943.

[25] C. Heinemann (supra note 10), p. 640.

[26] C. Heinemann (supra note 10), p. 641; see also J. Bergsmann, Requirements Engineering für die agile Softwareentwicklung, 2014, pp. 237, 246.

A product description that contains a detailed description of the functions and design of the finished product and shows how the finished product corresponds to the product vision is recommended. The product vision should be open to comments and change requests from the customer and should be linked to a dispute resolution mechanism. The legal warranty covers the product's freedom from defects and its compatibility with the product description. Warranty and liability are limited in time; the use of open source software and virus protection should also be clarified.

## 5. The sprint process

The parties determine the duration of sprints, usually as a very short period of time (two to four weeks). This duration should not be changed, even if the progress of the project is behind schedule. Unfinished products should be reassessed and prioritized in a product backlog.

Each sprint typically has three types of meeting. The sprint starts with a planning meeting between product owner, development teams and the Scrum Master. The product owner will explain to the development team which points are of high priority in the current sprint and which goals and business contexts are connected with the individual points. The development team will then determine which particularly important goals will be developed during the sprint. After this meeting, a sprint backlog is created by the development team, which breaks down the overall task into individual tasks and determines the effort required for each individual task. Thereafter, short meetings of the development team to describe which work has been completed by which member of the development team and which work is currently in progress are held daily (so-called daily scrums). In this connection, emerging obstacles to finish the work are discussed as well. At the same time, the cost estimates are updated. A sprint chart, which shows the cost estimate for each module in the context of the overall cost estimate, is recommended. Finally, there is a Sprint Review Meeting. Here it is important to record the achieved goals. In this respect, a 'Definition of Done' is required (see below).

Contractually, the progress of the sprint process should be determined, so that the development teams are also obliged to determine how many central elements can be developed in the current sprint. The customer is obliged to acknowledge the agreed tasks in the sprint. It is also determined how the Sprint Backlog is updated, depending on planning meetings. The development team undertakes to implement the planned improvements in the next sprint. The sprint process continues until the project is finished or the contract is terminated. Commentators dispute whether the outcome of the sprinting process should be regarded as binding. In our opinion, we are dealing with elements of a contract to produce a work if backlog entries for a sprint including the 'Definition of Done' have been defined. This then leads to the problem that unachieved but agreed backlog entries would normally have to be processed in the next sprint without additional remuneration, which is difficult for the teams in practice.

## 6. Definition of done, prices and warranty

Acceptance is of central importance in the law of contracts to produce a work, including:

- specification of the accepted work,
- the loss of the right to new production,
- the transfer from performance claims to warranty claims,
- the due date of payment (*section 641 BGB*),
- the transfer of the risk of compensation (*sections 644, 645 BGB*),
- the commencement of the limitation period for warranty claims (*section 634a [2] BGB*),
- the exclusion of known, not reserved defects (*section 640 [2] BGB*).

Acceptance "of the work produced in accordance with the contract" means physical acceptance usually by transfer of ownership, combined with a declaration by the customer that he recognises the work as the performance in matter of the main object in accordance with the contract.[27] In extension to delivery within the meaning of *section 438 (2) BGB*, acceptance requires the express or tacit approval of the service as essentially in accordance with the contract.[28] In the past, there was a dispute as to whether software could be accepted at all.[29] Today, however, this is generally accepted.[30] In case of computer services, the approval requires the possibility of examining the software, i.e. its complete and proper delivery. The prerequisite for this is approval by the customer, since only then can the customer check the "matching".[31] A work record with the remark "System in order" is not sufficient.[32] Acceptance should be regulated as clearly as possible in the contract, in particular in order to clarify disputes regarding the commencement of warranty and compensation obligations.[33]

The implied acceptance is permissible and includes conduct carried by the client's will, with which the client expresses that he considers the "building" to be essentially in accordance with the contract.[34] If the parties agree on a formal acceptance, this must always take place. Only in very narrow exceptional cases, the implied cancellation of the formal acceptance is to be assumed. An implied cancellation of the formal acceptance requirement shall not be considered if the client sends a comprehensive request for the remedy of defects to the contractor in good time after reference to the

[27] H. Sprau, in: Palandt, 76. Auflage 2017, section 640 para. 3.

[28] OLG Hamm [Higher Regional Court Hamm] Dec. 12, 1988, NJW 1989, pp. 1041 f.; Compare A. Feuerborn, in: CR 1991, pp. 1 ff. and A. Feuerborn/T. Hoeren, in: CR 1991, pp. 513 ff.

[29] With doubts OLG Celle [Higher Regional Court Celle] Feb. 26, 1986, CR 1988, pp. 303 ff., because suitability could only be determined when used.

[30] See OLG Hamburg [Higher Regional Court Hamburg] Aug. 9, 1985, CR 1986, pp. 83 ff.

[31] Compare BGH [Federal Court of Justice] Jan. 24, 1990, NJW 1990, pp. 1290 ff.

[32] OLG Düsseldorf [Higher Regional Court Düsseldorf] Sept. 28, 2001, CR 2002, pp. 324 f.

[33] J. Bergsmann (supra note 27), p. 242.

[34] BGH [Federal Court of Justice] Nov. 3, 1992, NJW 1993, pp. 1063 ff.; BGH [Federal Court of Justice] Nov. 15, 1973, NJW 1974, pp. 95 f.

object, with the remark that nothing stands in the way of a formal acceptance once the remedy of these defects has occurred.[35] It is therefore erroneous to assume that the mere production of software applications is already making up for the acceptance.

A conclusive acceptance is for example in

- the beginning of the intended use,[36]
- the unconditional payment of the work wages,[37]
- the retention of the amount for notified defects in the course of the final conversation.[38]

The implied acceptance may be because the customer does not give notice of defects after completion of the service, receipt of the completed work and expiry of an inspection period of six months.[39] If the service is only partially performed in accordance with the contract, a conclusive acceptance is not to be considered.[40] Nor is there any implied acceptance in a notice of termination.[41] Likewise, the use of the software "under pressure" cannot be regarded as tacit acceptance.[42] Advance payments also do not constitute an implied acceptance. Accordingly, the unconditional payment of a (partial) invoice for an additional service does not contain any statement by the customer to the effect that he wishes to set aside the existence of all or part of the outstanding accounts receivable at the same time. Advance payments are down payments relating to the remuneration entitlement for the entire work. After termination of the contract, the contractor has to settle his services definitively. This obligation arises from the agreement on the provisional payments and applies irrespective of whether it is expressly provided in the contract.[43]

It is also conceivable to divide the acceptance into partial approvals. Partial acceptance can be agreed if the parts of the total service that can be valued independently are concerned. A software developer may only demand acceptance in parts based on a corresponding agreement.[44] This must be unequivocal.[45]

The reform of the building contract law as of January 1, 2018, which also benefits agile software development projects, is particularly important here. One aim of the new regulations is to accelerate the acceptance procedure with regard to a rapid inflow of remuneration to the contractor and thus create legal certainty with regard to remuneration in accordance with *section 640 (1) BGB* .[46] Thus, the section was amended in order to facilitate conclusive acceptance. Another aim is to increase legal certainty. To reach this goal, the German legislator created *section 648a BGB*, which allows a party to terminate the contract under certain circumstances without notice. Until the amendment, this right only existed as judicial law. Beside these two changes, various specific regulations on building contract law were added, which have no effect on agile programming.

As mentioned, regulations on acceptance, especially deemed acceptance, were modified. The formerly valid deemed acceptance in accordance with section 640 (1) *sentence 3 BGB f. v.* presupposed non-acceptance within a certain period of time despite an obligation to do so. According to the new law, deemed acceptance occurs if the customer does not refuse acceptance within the set period of time stating a defect. In contrast to *section 640 BGB f. v.*, the customer is now obliged to react actively to a demand for acceptance in order to avoid the occurrence of deemed acceptance, combined with a reversal of the burden of proof. Otherwise, acceptance shall be deemed to have been granted, even if there are actually substantial defects.

The new version of the law amends two essential details. Firstly, completion of the work becomes a requirement for setting a reasonable deadline for acceptance, in order to avoid misuse of the acceptance function by an early tender of the work.[47] Secondly, the developer from *section 640 (2) BGB n. v.* can effect acceptance himself if, after completion of the work, he sets a deadline and the customer does not refuse acceptance stating "at least one defect". The amendment to the wording of at least one "defect" did not result until the last stages of the Bundestag deliberations as a proposal from the corresponding subject committees.[48] This is very helpful for an agile project as the customer is now requested to report any project defects as soon as possible and to discuss them with the project team. The purchaser can prevent a unilateral initiation of acceptance by refusing acceptance, stating any defect, even if it is insignificant, whereby the actual existence of the defect is initially completely irrelevant for acceptance *(section 640 [2] sentence 1 BGB n. v.)*.[49]

[35] LG Frankenthal [Regional Court Frankenthal] Dec. 17, 2013, Baurecht [BauR] 2014, p. 740.
[36] BGH [Federal Court of Justice] Sept. 20, 1984, NJW 1985, pp. 731 f.
[37] BGH [Federal Court of Justice] Nov. 24, 1969, NJW 1970, pp. 421 ff.; OLG Köln [Higher Regional Court Cologne] Apr. 1, 1992, BauR 1992, pp. 514 (515).
[38] OLG Koblenz [Higher Regional Court Coblenz] July 29, 1993, Neue Juristische Wochenschrift Rechtsprechungs-Report [NJW-RR] 1994, pp. 786 f.
[39] BGH [Federal Court of Justice] Sept. 26, 2013, NJW 2013, pp. 3513 ff. The Higher Regional Court of Frankfurt has explicitly left open the question, whether there is an implied acceptance because of the fact that the project is progressing iteratively with the previous programming performance, see OLG Frankfurt [Higher Regional Court Frankfurt] Aug. 17, 2017, CR 2017, pp. 646 f.
[40] OLG Hamm [Higher Regional Court Hamm] June 23, 1996, NJW-RR 1996, pp. 86 ff.
[41] BGH [Federal Court of Justice] Dec. 19, 2002, NJW 2003, pp. 1450 f.
[42] OLG Dresden [Higher Regional Court Dresden] Jan. 11, 2012, Immobilien- & Baurecht [IBR] 2014, p. 132.
[43] OLG Dresden [Higher Regional Court Dresden] Jan. 11, 2012, IBR 2014, p. 132.
[44] BGH [Federal Court of Justice] Feb. 10, 1994, NJW 1994, pp. 1276 ff.
[45] BGH [Federal Court of Justice] May 11, 2006, Neue Zeitschrift für Baurecht und Vergaberecht [NZBau] 2006, pp. 519 f.
[46] Regierungsentwurf [Cabinet Draft], Bundestag Drucksachen [BT] 18/8486, p. 48.
[47] W. Langen, in: NZBau 2015, pp. 658 (659).
[48] The Cabinet Draft (see supra note 47) still talks about "deficiencies" (plural).
[49] R. Kimpel, in: NZBau 2016, pp. 734, 734.

## 7. Termination

Agile development has the advantage that the customer is not tied to long delivery cycles and receives flexible products that are scalable at short notice. The production components are functional after the sprints. However, the customer should be able to leave the project after the completion of each project step. For this purpose, measurable criteria of termination must be defined and it must be regulated how to deal with them in the course of the project.[50] It is questionable whether this right should also be given to the developer, as he might simply stop working in the middle of the project. The insolvency risk should also be regulated. Furthermore, termination management is required. The fate of already made payments should also be contractually determined. Finally, it is worth looking at a dispute resolution mechanism, as it is usually not worthwhile to drive such cases to a state court. However, the questions that arise in this context are so complex that it is not possible to discuss them in depth within the framework of this article.

In the context of termination, the previous termination option under *section 649 BGB* will be extended to include the possibility of termination for good cause in accordance with *section 648a BGB*. The regulation repeats in large parts *section 314 BGB*. Termination may be effected without notice and termination for good cause already applies if a contracting party cannot reasonably be expected to continue the contractual relationship until the work has been completed, taking into account all circumstances of the individual case.[51] Thus, the extraordinary termination option based on judicial law is to be recognized, creating legal certainty as a practical matter. The draft refrains from a special act of termination for insolvency as regulated in *sections 8, 9 VOB/B*: Where an insolvent supplier proves to be unreliable and inefficient, in individual cases such as in the "protective shielding procedure" according to *section 270 German Insolvency Code (InsO)*, there is, by way of exception, no good cause for termination.[52] Ultimately, each software development contract is regarded as a long-term contract that approximates a continuing obligation.

A real novelty, especially for agile projects, is the partial termination clause of *section 648a (2) BGB n. v.*, which refers to a "part of the service that can be demarcated according to the contract". In contrast to *section 8 (3) VOB/B*, the hurdle is not based on the "self-contained part of a service", so that a clear distinction can be made between services rendered and those yet to be performed.[53] With regard to partial termination, it would seem reasonable to assess definable benefits, according to the criteria laid down in the contract.[54] For the IT sector, this makes it necessary to divide projects more clearly into self-contained units. Only when parties make an effort to define self-contained partial acceptance steps, they can make partial termination provisions, which is in the interest of both parties. As far as the judiciary is concerned, the need to consider the failure of the overall project when considering IT projects, is also eliminated.

Another significant new element in this context is the obligation to cooperate in determining the performance level, resulting from *section 648a (4) BGB n. v.* If one of the parties violates this obligation to cooperate by refusing it or not having done so by a reasonably set date, the burden of proof, according to *section 648a (4) sentence 2 BGB n. v.* will pass to this party, unless it is not responsible for its absence and informs the other party immediately.[55] This is intended to provide an incentive to assist in the quantitative assessment of services rendered up to the date of termination, in order to prevent subsequent disputes over the exact scope of the termination.[56]

*Section 648a (4) BGB n. v.* outlines the first steps for transition and termination management. However, the question how those contractual duties can ever be enforced remains. In addition, an exception to the exception is again provided if the absence on the date was not attributable to the absent party[57] and the corresponding circumstance had been communicated to the other party immediately.[58] The wording here is not very precise. *Section 648a (4) BGB n. v.* mentions two circumstances for the reversal of the burden of proof, namely the refusal of participation or absence at a date determined by the other contracting party for the assessment of performance. The exception to the exception shall only apply if the contracting party remains "absent" because of a circumstance for which it is not responsible. According to sense and purpose, the exception in sentence 3 only refers to the agreed and appropriately set deadline. The refusal to set a date for an appointment shall not be equated with a refusal of the assessment. After all, the other contracting party still has the possibility of setting a reasonable deadline for the assessment. Besides refusing the assessment of the performance status, a rejection of the assessment is possible. A contracting party is deprived of the possibility to assess the performance level (e.g. by a prohibition to enter the property). In addition to the reversal of the burden of proof, this results in a claim for damages due to positive breach of contract of subsequent cooperation obligations, in accordance with *sections 241, 280 et seqq BGB*.[59]

Ultimately, this results in the transfer of VOB/B structures into general contract law to produce a work. However, *section 648a BGB n. v.* goes beyond the VOB/B provisions, as the latter only applies to determine the "condition of parts of the service", if these parts of the service are withdrawn by the further execution of the test and determination. The VOB/B is

[50] J. Bergsmann (supra note 27), p. 236.

[51] W. Langen, in: NZBau 2015, pp. 658, 660.

[52] W. Langen, in: NZBau 2015, pp. 658, 660.

[53] Regierungsentwurf [Cabinet Draft], Bundestag Drucksachen [BT] 18/8486, p. 51.

[54] W. Langen, in: NZBau 2015, pp. 658, 561.

[55] H. Reiter, in: Gsell et al., beck-online.GROSSKOMMENTAR BGB [BeckOGK BGB], Oct. 2017, section 649 para. 222.

[56] Regierungsentwurf [Cabinet Draft], Bundestag Drucksachen [BT] 18/8486, p. 51.

[57] The attribution requirement in sentence 3 is based on the general principles of section 276 BGB. The circumstance is attributable if it was caused intentionally or negligently.

[58] Immediately in accordance with section 121 (1) sentence 1 BGB means "without culpable delay". The determination is carried out in individual cases.

[59] See legal literature on the VOB/B: B. Gartz, in: Nicklisch et al., VOB/B, 4. Auflage 2016, section 4 para. 176.

concerned, for example, with the assessment of fire protection devices that cannot be inspected in shafts and ceiling areas, since they are later to be sealed.[60] In essence, the assessment of performance conditions is not necessarily linked to the question of defects; essentially it only refers to the "condition" and is thus purely objective.[61] It may additionally cover a statement of freedom from defects, but this is not a necessity. Within the literature on the VOB, there is agreement that the mere condition assessment does not entail a contractual partial acceptance and can therefore not be linked to its general effects.[62]

Because of assessments, the burden of presentation and proof lies with those who wish to rely on a different, deviating level of performance. In *section 648a (4) BGB n. v.* it is added that the burden of proof does not only pass after the moment of joint assessment, but even as soon as the assessment in question is refused or the appointment is missed.[63]

## 8. Conclusion

To a certain extent, agile programming is a counterpoint to legally planned project design. The motto for a well-designed agile project can only be to include as few legal provisions as possible and as many as necessary. The trend towards agile projects is favoured by the new law on building contracts, as it calls for projects to be planned and awarded in closed subunits and accordingly to be subject to partial acceptance.

## Acknowledgement

The authors would like to thank Mrs. Carola Kaiser (Volkswagen) for her valuable practical advice and the ITM students Mrs. Elisa Röhr, Mrs. Carola Kaiser and Mr. Moritz von Saß for the revision of the footnotes.

[60] See the example of B. Gartz (supra note 60), section 4 para. 166.

[61] See the example of B. Gartz (supra note 60), section 4 para. 167.

[62] D. Merkens, in: Kapellmann/Messerschmidt, VOB/B, 6. Auflage 2018, section 4 para. 226; R. Leinemann, in: Leinemann, VOB/B, 6. Auflage 2016, section 4 para. 193; M. Bschorr, in: Franke et al., VOB/B, 6. Auflage 2017, section 4 para. 335.

[63] This particular legal consequence is controversial concerning the VOB/B. See R. Leinemann (supra note 63), section 4 para. 194; W. Junghenn, in: Ganten/Jansen/Voit, VOB/B, 3. Auflage 2013, section 4 Abs. 10 para. 8; in contrast B. Gartz (supra note 60), section 4 para. 175 referring to the statutes on distribution of evidence in the individual case.