www.jusletter.ch

Wolfgang Straub

Verträge für agil geführte Projekte

Les exigences auxquelles doivent satisfaire les projets informatiques ne peuvent généralement pas être définitivement fixées à l'avance. Ces dernières années, des méthodes dites « agiles », tels que Scrum, ont dès lors fortement gagné en importance. Des modèles traditionnels de contrat ne s'adaptent cependant pas directement aux projets développés sur la base d'une méthode agile. La contribution donne un aperçu sur la nature juridique et sur les contenus possibles de « contrats agiles ». Elle sera complétée dans une contribution ultérieure par une liste de contrôle. (nse)

Catégories d'articles: Contributions
Domaines juridiques: Informatique et droit; Droit des marchés publics

Proposition de citation: Wolfgang Straub, Verträge für agil geführte Projekte, in : Jusletter 21 décembre 2015

Inhaltsübersicht

I. Phasenmodelle und agile Vorgehensmethoden

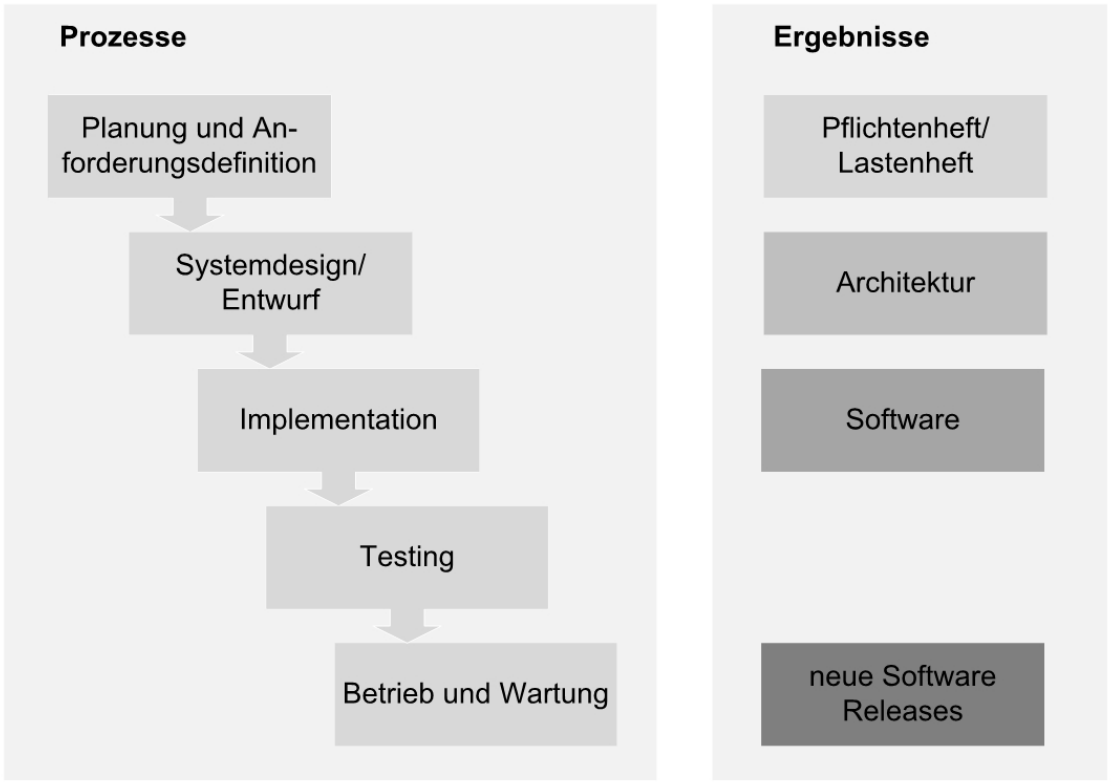
[Rz 1] In den letzten Jahrzehnten wurden zahlreiche Vorgehensmodelle zur Herstellung von Software und IT Systemen entwickelt.[1] Strukturierte Vorgehensmethoden dienen vor allem dem *Risikomanagement* und der *Qualitätssicherung* von Entwicklungsprojekten.

[Rz 2] Gemäss dem «*Wasserfallmodell*» von Royce[2] folgen in einem Entwicklungsprojekt verschiedene Phasen auf einander: Zunächst werden die Anforderungen definiert und eine Grobplanung vorgenommen. In einem nächsten Schritt wird das Systemdesign entwickelt. Anschliessend erfolgt die eigentliche Programmierungsphase. Nach erfolgreichen Tests wird die Software in Betrieb genommen, gewartet und gegebenenfalls weiterentwickelt.

---

1 Siehe dazu auch den Überblick bei Reichert Jörg, Vertragsfreiheit und agile Softwareentwicklung: vertragstypologische Einordnung agiler Softwareentwicklungsverträge, Saarbrücken 2013, S. 15ff; Witzel Michaela, Projektmanagement, in: Schneider Jochen/von Westphalen Friedrich (Hrsg.), Software-Erstellungsverträge: Projektgestaltung, Vertragstypen, Rechtsschutz, 2. A., Köln 2014, S. 783–848.

2 Siehe dazu Royce Winston W., Managing the Development of Large Software Systems, in: Proceedings of IEEE Western Electronic Show and Convention, Los Angeles 1970, S. 1–9 (ein Reprint ist auch online verfügbar unter http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf; Alles Websites zuletzt besucht am 16. Dezember 2015), S. 1ff. Dort wird die Methode allerdings noch nicht als «*Wasserfall*» bezeichnet. Royce empfiehlt als Modifikationen bereits Iterationen und einen Einbezug des Kunden in den Entwicklungsprozess.



Grafische Übersicht 1: Wasserfallmodell

[Rz 3] Weiterentwicklungen des Wasserfallmodells wie das Spiralmodell[3] sehen *Iterationen* vor: Die Software bzw. das IT System wird hier in mehreren, nacheinander durchlaufenen Zyklen geplant, entwickelt, getestet und verbessert.

[Rz 4] Im «*V-Modell*» wird zusätzlich eine Korrelation von den Entwurfs- und Umsetzungsphasen zu den Test- und Abnahmephasen hergestellt. Das von Boehm[4] vorgeschlagene Modell wurde inzwischen ebenfalls weiterentwickelt. Daraus ist unter anderem das V-Modell XT entstanden, welches von den deutschen Bundesbehörden als Entwicklungsstandard für Systementwicklungsprojekte verwendet wird.[5] Die Ergebnisse der Realisierung, Systemintegration, Ablieferung und Abnahme werden gegenüber den Anforderungsdefinitionen und der Systemspezifikation validiert und verifiziert.[6]

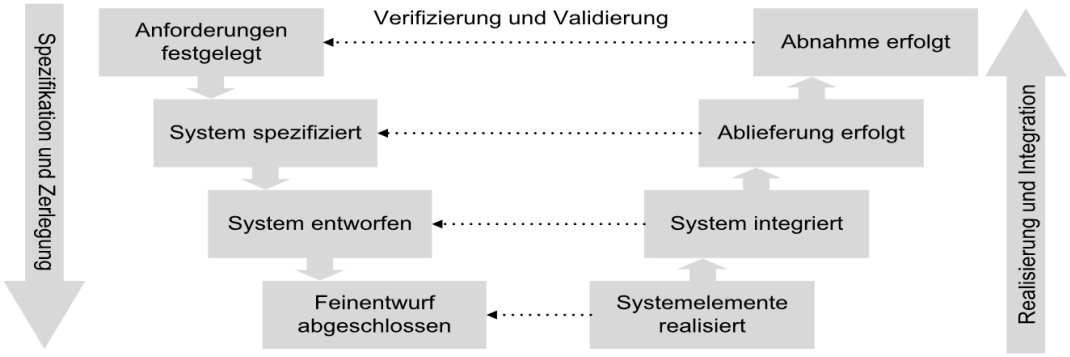
---

3 So werden etwa im Spiralmodell von Boehm vier Phasen mehrmals durchlaufen: 1. Festlegung der Ziele, Lösungsalternativen und Rahmenbedingungen; 2. Beurteilung der Lösungsalternativen und Risikoanalyse; 3. Entwicklung und Validierung; 4. Planung des nächsten Zyklus; siehe dazu im Einzelnen Boehm Barry W., A Spiral Model of Software Development and Enhancement, Computer 5/1988, S. 61–72, S. 61ff.

4 Siehe Boehm Barry, Guidelines for Verifying and Validation Software Requirements and Design Specifications, in: Samet Jonathan (Hrsg.), Proceedings of Euro IFIP 79, Amsterdam 1979, S. 711–719, zit. Boehm, Guidelines for Verifying and Validation, S. 711ff.

5 Siehe dazu www.v-modell-xt.de.

6 *Validation* stellt sicher, dass das Spezifizierte auch implementiert wurde, *Verifikation*, dass die Implementierung korrekt funktioniert.



Grafische Übersicht 2: V-Modell XT

[Rz 5] Auch das *ITIL* Release Management und das im schweizerischen Bundesumfeld verwendete *HERMES* Modell beruhen letztlich auf dem Wasserfallmodell.[7] HERMES unterscheidet grundsätzlich die Phasen Initialisierung, Konzeption, Realisierung und Einführung.[8] Das ITIL Release Management gliedert sich in die Phasen Planung, Realisierung, Test und Produktivsetzung.[9]

[Rz 6] Eine *Anwendung von Phasenmodellen* ist im Allgemeinen dann sinnvoll, wenn sich Anforderungen, Leistungen und Abläufe in der Planungsphase bereits relativ präzise beschreiben lassen. Bei komplexen Vorhaben lassen sich aber kaum je alle Anforderungen zum voraus definieren. In lang dauernden Projekten entstehen zudem aufgrund von technischen, organisatorischen – und mitunter auch rechtlichen – Entwicklungen während der Implementierungsphase oft neue Anforderungen.

[Rz 7] In den letzten Jahren haben *agile Methoden* wie Scrum, Extreme Programming (XP)[10], Feature Driven Development (FDD)[11], Dynamic Systems Development Method (DSDM)[12], Crystal Methods[13], Kanban Software Development[14] und Lean Software Development[15] stark an Bedeutung gewonnen.[16] Es scheint, dass in der Schweiz etwa gleich viele Projekte nach agilen Metho-

7 Siehe zur Einordnung von Phasenmodellen in den Beschaffungsablauf auch Schreiber Josef, Beschaffung von Informatikmitteln: Submissionsverfahren, Pflichtenheft, Evaluation, 5. A., Bern 2015, S. 20ff.

8 Siehe dazu im Einzelnen Informatiksteuerungsorgan des Bundes ISB (Hrsg.), HERMES 5.1 Referenzhandbuch, 2. A., Bern 2015, online verfügbar unter http://www.isb.admin.ch/themen/methoden/01661/01662/index.html?lang=de, S. 5ff.

9 Siehe dazu Hoppen Peter/Victor Frank, ITIL – Die IT Infrastructure Library – Möglichkeiten, Nutzen und Anwendungsfälle in IT-Verträgen, CR 2008, S. 199–204, S. 201ff.

10 Siehe dazu Beck Kent, Extreme programming explained; embrace change, Boston 2000, S. 8ff.

11 Siehe dazu Palmer Stephen R./Felsing John M., A Practical Guide to the Feature-Driven Development, Upper Saddle River, New Jersey 2002, S. 35ff.

12 Siehe http://www.dsdm.org.

13 Siehe dazu Cockburn Alistair, Crystal clear: a human-powered methodology for small teams, Upper Saddle River, New Jersey 2005, S. 17ff.

14 Siehe dazu Anderson David J., Kanban: Evolutionäres Change Management für IT-Organisationen, Heidelberg 2011, S. 15ff; Leopold Klaus/Kaltenecker Siegfried, *Kanban in der IT:* Eine Kultur der kontinuierlichen Verbesserung schaffen, München 2013, S. 7ff.

15 Siehe dazu Poppendieck Mary/Poppendieck Tom, Lean Software Development: An Agile Toolkit, Boston 2003, S. 4ff.

16 Siehe dazu auch den Überblick über die einzelnen Methoden und ihre vertragsrechtlichen Implikationen bei Reichert (Fn. 1), S. 33ff und S. 206ff.

den[17] realisiert werden wie nach Phasenmodellen[18]. Die Gestaltung von Verträgen für solche Projekte wurde in der schweizerischen Rechtsliteratur bisher allerdings noch nicht sehr oft thematisiert.[19] Soweit ersichtlich gibt es in der Schweiz auch noch keine entsprechende Rechtsprechung.[20]

[Rz 8] Es gibt keine verbindliche Definition von Agilität. Unter «agil» werden in der Praxis teilweise recht *unterschiedliche Vorgehensmethoden* verstanden.[21] Währenddem z.B. der Fokus von Extreme Programming (XP) auf der Vorgehensweise bei der Programmierung liegt[22], stellt Scrum ein Management Framework dar, das Rollen, Abläufe und Verantwortlichkeiten für Projekte festlegt.[23] Nachfolgend wird vor allem auf *Scrum*[24] Bezug genommen, da diese Methode weitaus am weitesten verbreitet[25] ist.[26]

---

17 Gemäss SwissQ/Universität Zürich (Hrsg.), Software Development 2015: Agile, Requirements, Testing, online verfügbar unter http://www.swissq.it, S. 8 machten rund 450 befragte Personen zur vorwiegenden Projektvorgehensart in ihrem Unternehmen folgende Angaben: Agile: 41%; Wasserfall: 26%, Iterativ: 26%; Keine: 7%. Eine zwei Jahre früher publizierte Studie von SwissQ/Universität St. Gallen (Hrsg.), Agile 2013, Trends & Benchmarks Report Schweiz, online verfügbar unter http://www.swissq.it, S. 7 und 11, ergab folgendes Bild zu den angewandten Vorgehensmodellen: Agil: 49%; Wasserfall 53%; Iterativ 20%, RUP und HERMES: je 10%. Nur 7.4% der Unternehmen führten sämtliche Projekte agil. Ein direkter Vergleich der beiden Studien ist allerdings nicht möglich, da die Fragestellung nicht identisch war.

18 Im Gegensatz zu agilen Methoden werden Vorgehensweisen wie das Wasserfallmodell oder das V-Modell, welche vom sukzessiven Durchlaufen verschiedener Phasen ausgehen, oft als «traditionell» oder als «*ingenieurmässig*» bezeichnet. Nach dem hier zugrunde gelegten Verständnis sind aber auch agile Methoden ingenieurmässig, indem sie den Entwicklungsprozess systematisch strukturieren. Daher wird hier für nicht agile Vorgehensweisen der Begriff «*Phasenmodelle*» verwendet.

19 Siehe zu agilen Verträgen im schweizerischen Recht Urs Egli, Agile Softwareprojekte: Rechtliche Qualifikation und vertragliche Umsetzung, in: Jusletter 31. August 2015; Fröhlich-Bleuler Gianni, Softwareverträge, System-, Software-Lizenz und Software-Pflegevertrag, 2. A. Bern 2014, S. 104–109, Rz. 343ff; Sury Ursula, Informatikrecht, Bern 2013, S. 44.

20 Eine Übersicht über internationale Gerichtsverfahren und Literaturmeinungen zu agilen Projekten findet sich bei Reichert (Fn. 1), S. 58ff.

21 Siehe dazu auch Fröhlich-Bleuler (Fn. 19), Rz. 343ff.

22 Extreme Programming sieht insbesondere die ständige Verfügbarkeit eines entscheidbefugten Kundenvertreters und das paarweise Entwickeln durch je zwei Programmierer vor, welche sich beim Schreiben und Überprüfen des Programmcodes abwechseln. Siehe zu Projektverlauf und Arbeitstechniken von Extreme Programming im Einzelnen Beck, S. 131ff. Siehe zur rechtlichen Einordnung Reichert (Fn. 1), S. 39ff, S. 209 und S. 254; Schneider Jochen, Beratung bei Projekt-Verträgen bis einschliesslich Pflichtenheft, in: Schneider Jochen/von Westphalen Friedrich (Hrsg.), Software-Erstellungsverträge: Projektgestaltung, Vertragstypen, Rechtsschutz, 2. A., Köln 2014, S. 193–318, Rz. C 128ff.

23 Siehe dazu Arbogast Tom/Larman Craig/Vodde Bas, Practices for Scaling Lean & Agile Development, Amsterdam 2010, S. 499ff; Opelt Andreas/Gloger Boris/Pfarl Wolfgang/Mittermayr Ralf, Der agile Festpreis: Leitfaden für wirklich erfolgreiche IT-Projekt-Verträge, 2. A., München 2014, S. 6ff; Hengstler Arndt, Gestaltung der Leistungs- und Vertragsbeziehung bei Scrum-Projekten, ITRB 2012, S. 113–116, S. 113ff.

24 «*Scrum*» bedeutet «Gedränge» und stammt aus dem Rugby. Dort bezeichnet der Begriff eine Methode, das Spiel neu zu starten. Siehe zu den Inhalten der Scrum Methodik im Einzelnen Schwaber Ken/Sutherland Jeff, The Scrum Guide, Version 1.2, 2013, online verfügbar unter http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf#zoom=100 (deutsche Übersetzung: http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-DE.pdf), S. 3ff; Scrum Alliance (Hrsg.), Scrum: a description, Version 2012.12.13 (online verfügbar unter http://agileatlas.org/images/uploads/corescrum.pdf, deutsche Übersetzung von Canditt Sabine/Gaertner Markus/Schliep Andreas: http://agileatlas.org/images/uploads/AgileAtlas-DE.pdf), S. 2ff.

25 Gemäss SwissQ/Universität St. Gallen, Agile 2013 Studie (Fn. 17), S. 5, wurden in der Schweiz über 85% der agil geführten Projekte nach Scrum realisiert. Die Studie SwissQ/Universität Zürich, Software Development 2015 (Fn. 17) schlüsselt noch weiter auf: Danach nutzten 87.4% der agilen Projekte Elemente von Scrum, aber 39.3% verwendeten Scrum in Kombination mit einem anderen Vorgehensmodell.

26 Siehe zur rechtlichen Bedeutung von agilen Projektmethoden (unter Berücksichtigung des deutschem Rechts) Reichert (Fn. 1), S. 35ff und S. 206; Beardwood John P./Shour Michael, Risk Management and Agile Software Development: Optimizing Contractual Design, CR 2010, S. 161–170; Opelt/Gloger/Pfarl/Mittermayr (Fn. 23), S. 6ff; Jähnichen Stefan, Formen Agilen Programmierens, in: Redeker Helmut/Hoppen Peter (Hrsg.), DGRI Jahrbuch 2011, Köln 2012, S. 119–125, S. 119ff; Auer-Reinsdorff Astrid, Feststellung der versprochenen Leistung

[Rz 9] In der Praxis werden allerdings zunehmend *hybride Methoden* verwendet, welche Elemente mehrere agiler Vorgehensweisen mit einander kombinieren oder agile Elemente innerhalb eines Phasenmodells verwenden. Beispielsweise enthält HERMES 5 ein eigenes Modul «Entwicklung Agil».[27] Die nachfolgenden Ausführungen sind auf solche Projekte *mutatis mutandis* anzuwenden.

II. Charakteristika agiler Methoden

[Rz 10] Agile Methoden wollen den Softwareentwicklungsprozess (inklusive Projekt- und Produktmanagement) *unmittelbarer, effizienter und flexibler* gestalten. Auch sie enthalten vor Projektstart eine Vorbereitungs- oder Explorationsphase. Diese ist je nach Methodik unterschiedlich stark ausgeprägt.[28] Es wird aber grundsätzlich auf das Erstellen abschliessender Spezifikationen vor Beginn der Realisierung verzichtet.

[Rz 11] Den einzelnen Anforderungen werden im Hinblick auf die Realisierung unterschiedlichen *Prioritäten* zugeordnet, welche im Lauf des Projekts geändert werden können. Die Umsetzung erfolgt schrittweise nach einer iterativen Vorgehensweise.

[Rz 12] Es soll möglichst in raschen Zyklen funktionsfähige Software (*Product Increments*) ausgeliefert und getestet werden. Dadurch kann bereits sehr früh konkreter Nutzen generiert werden. Zudem kann so rascher auf Probleme oder Anforderungsänderungen reagiert werden (*fail fast – learn fast – improve fast!*).

[Rz 13] Agile Methoden orientieren sich inhaltlich an den *Werten*, die im *Agile Manifesto*[29] festgehalten sind. Agilität kann aber auch als eine Verbindung von Team Leadership Philosophie, Projekt Management Prozess und Engineering Best Practices charakterisiert werden.[30]

[Rz 14] Das *Agile Manifesto* umfasst vier Grundmaximen:[31]

- «*Wir erschliessen bessere Wege, Software zu entwickeln, indem wir es selbst tun und anderen dabei helfen. Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:*

---

beim Einsatz agiler Projektmethoden, ITRB 2010, S. 93–95; Frank Christian, Bewegliche Vertragsgestaltung für agiles Programmieren, CR 2011, S. 138–144; Schneider Jochen, «Neue» IT-Projektmethoden und «altes» Vertragsrecht, ITRB 2010, S. 18–23; Koch Frank A., Agile Softwareentwicklung – Dokumentation, Qualitätssicherung und Kundenmitwirkung, ITRB 2010, S. 114–119, S. 114ff; Kremer Sascha, Gestaltung von Verträgen für die agile Softwareerstellung, ITRB 2010, S. 283–289, S. 283ff; Söbbing Thomas, Agile Projekte in der IT-rechtlichen Praxis, ITRB 2014, S. 214–219; Fuchs Anke/Meierhöfer Christine/Morsbach Jochen/Pahlow Louis, Agile Programmierung – Neue Herausforderungen für das Softwarevertragsrecht? MMR 2012, S. 427–433.

27 Siehe dazu Informatiksteuerungsorgan des Bundes, HERMES Referenzhandbuch (Fn. 8), S. 28 und S. 165ff sowie Informatiksteuerungsorgan des Bundes ISB (Hrsg.), HERMES und Agilität (verfügbar unter http://www.isb.admin.ch/themen/methoden/01661/01667/index.html?lang=de&down-load=NHzLpZeg7t,lnp6I0NTU042l2Z6ln1acy4Zn4Z2qZpnO2Yuq2Z6gpJCEeYR6g2ym162epYbg2c_JjKbNoKSn6A--&t=.pdf#sprungmarke0_21), S. 4ff. Inwieweit agile Vorgehensmethoden auch in die Realisierungsphase des V-Modells sinnvoll integriert werden können, wird kontrovers diskutiert. Siehe dazu Reichert (Fn. 1), S. 26f, mit weiteren Hinweisen. Umgekehrt ist auch denkbar, dass bloss die Anforderungen mit agilen Methoden konkretisiert werden (z.B. Erarbeitung einer Feinspezifikation im Rahmen des Requirements Engineering) und die Realisierung anschliessend nach einem Phasenmodell erfolgt. Siehe dazu auch Egli (Fn. 19), Rz. 27; Schneider (Fn. 1), Rz. C 154.

28 Siehe dazu im Einzelnen Reichert (Fn. 1), S. 33ff.

29 http://agilemanifesto.org. Historisch gesehen sind die meisten agilen Methoden allerdings älter als das Agile Manifesto. Siehe zu dessen Entstehungsgeschichte Reichert (Fn. 1), S. 32.

30 Siehe dazu Informatiksteuerungsorgan des Bundes, Studie HERMES und Agilität (Fn. 27), S. 6.

31 Freie Übertragung des englischen Originaltexts. Siehe auch http://agilemanifesto.org/iso/de. Siehe zu den einzelnen Maximen Opelt/Gloger/Pfarl/Mittermayr (Fn. 23), S. 6.

- *Individuen und Interaktionen mehr als Prozesse und Werkzeuge*
- *Funktionierende Software mehr als umfassende Dokumentation*
- *Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung*
- *Reagieren auf Veränderung mehr als das Befolgen eines Plans*
- *Das heisst, obwohl wir die Werte auf der rechten Seite wichtig finden, schätzen wir die Werte auf der linken Seite höher ein.*»

[Rz 15] Die *Grundmaximen* werden allerdings oft überinterpretiert. Sie bedeuten keineswegs, dass Verträge, Spezifikationen oder Dokumentationen bei agilen Entwicklungen überflüssig sind. Tatsächlich müssen in Verträgen für agile Projekte viele Themen sogar eingehender geregelt werden.[32] Verträge, Spezifikationen und Dokumentationen sollen aber nicht Selbstzweck werden und die Ressourcen sollen vor allem in die Entwicklung funktionsfähiger Software investiert werden.

[Rz 16] Ausgehend von den Werten des Agile Manifesto wurden 12 *Prinzipien* für agile Entwicklungen definiert.[33]Die daraus abgeleiteten *Agile Praktiken* machen im Sinn von «Best Practices» konkrete Vorgaben für bestimmte Aspekte. Sie betreffen beispielsweise die Aufwandschätzung[34], die Durchführung von Retrospektiven[35] oder das Testen[36].

[Rz 17] Agile Methoden eignen sich vor allem für Softwareentwicklungsprojekte[37], welche sequentiell realisiert werden können[38] und bei denen die Leistungsbezügerin zu einer aktiven Mit-

---

32 Siehe zu den Elementen agiler Vertragsgestaltung Rz. 84ff sowie Reichert (Fn. 1), S. 136f, S. 205 und S. 247ff.

33 Die 12 Prinzipien lauten wie folgt (http://agilemanifesto.org/iso/de/principles.html):

- Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufrieden zu stellen.
- Heisse Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.
- Liefere funktionierende Software regelmässig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne.
- Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten.
- Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen.
- Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist im Gespräch von Angesicht zu Angesicht.
- Funktionierende Software ist das wichtigste Fortschrittsmass.
- Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Benutzer sollten ein gleichmässiges Tempo auf unbegrenzte Zeit halten können.
- Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität.
- Einfachheit – die Kunst, die Menge nicht getaner Arbeit zu maximieren – ist essenziell.
- Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.

In regelmässigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an.

34 Siehe zu Aufwandschätzungen Rz. 32 sowie Cohn Mike, Agile Estimation and Planning, Upper Saddle River, New Jersey 2005 (Fn. 34), S. 35ff und S. 49ff.

35 Siehe zu Retrospektiven Rz. 34.

36 Siehe dazu etwa http://www.it-agile.de/wissen/praktiken/agiles-testen.

37 Siehe dazu auch Egli (Fn. 19), Rz. 27ff. und Schneider (Fn. 1), Rz. C 118f. Agile Methoden können über die Softwareentwicklung hinaus auch in anderen Bereichen verwendet werden. Siehe dazu auch die Erhebungen von SwissQ/Universität Zürich, Software Development 2015 (Fn. 17), S. 25.

38 In der Praxis haben sich in den letzten Jahren indessen verschiedene Methoden zur Skalierung agiler Entwicklungen mit mehreren parallelen Entwicklungsteams etabliert, insbesondere Scaled Agile Framework (SAFe, http://www.scaledagileframework.com), Large Scale Scrum (LeSS, http://less.works/) und NEXUS Scaled Professional Scrum (https://www.scrum.org/Ressources/What-is-Scaled-Scrum).

wirkung bereit ist. Besonders *erfolgreich*[39] scheinen kleinere Projekte.[40] Diese haben aufgrund der geringeren Komplexität ohnehin höhere Erfolgschancen.[41] Allerdings erschwert die unterschiedliche Spezifikationstiefe bei Projektstart einen Vergleich zwischen agil geführten und nach Phasenmodellen realisierten Projekten: Letztere werden nur dann als vollständig erfolgreich betrachtet, wenn die gesamte in den Spezifikationen definierte oder durch Changes einvernehmlich modifizierte Funktionalität innerhalb des Zeit- und Budgetrahmens umgesetzt wurde. Demgegenüber ist der *Project Scope*[42] eines agilen Projektes viel offener formuliert, so dass schwieriger überprüft werden kann, inwieweit die effektiv realisierte Funktionalität diesen abdeckt.

[Rz 18] Für die *Initialisierung eines agilen Projekts*[43] ist eine *Vision* des zu realisierenden Produkts erforderlich.[44] Gestützt darauf wird in einer Planungsphase eine erste Liste der gewünschten Funktionen zusammengestellt. Diese wird im *Product Backlog*[45] in der Form von *User Stories*[46]

---

39 Gemäss SwissQ/Universität St. Gallen, Agile 2013 Studie (Fn. 17), S. 6ff, hatten 2013 bereits 27.9% der befragten Unternehmen mehrjährige Erfahrung mit agilen Methoden. 36.3% der agilen Projekte wurden mit der erwarteten Funktionalität und innerhalb des Zeit- und Kostenrahmens abgeschlossen und nur rund 4% der Projekte wurden vorzeitig abgebrochen. Nach SwissQ/Universität Zürich, Software Development 2015 (Fn. 17), S. 16, haben sich die Zahlen inzwischen noch verbessert: Es wurden 58.9% der agilen Projekte erfolgreich abgeschlossen und nur 1.1% wurden abgebrochen. Dennoch war in beiden Studien rund die Hälfte der Befragten mit der Umsetzung von agilen Methoden in der eigenen Organisation nicht oder nur teilweise zufrieden (insbesondere längere Projektdauern als erwartet und komplizierte Umsetzung). Agile Projekte scheinen am ehesten an Inkompatibilitäten zwischen Unternehmensphilosophie und agilen Werten sowie an mangelnder Unterstützung durch das Management der Leistungsbezügerin zu scheitern.

40 Siehe dazu auch Standish Group, Chaos Manifesto 2013, http://www.standishgroup.com, S. 25. Danach scheiterten 6% der agil geführten kleineren Projekte – gegenüber 8% der nach dem Wasserfallmodell realisierten. Hingegen wurden 46% der agil geführten Projekte – gegenüber 49% der nach dem Wasserfallmodell realisierten – als vollständig erfolgreich betrachtet (mit der erwarteten Funktionalität und innerhalb des Zeit- und Kostenrahmens). Siehe zu den kritischen Erfolgsfaktoren für kleinere agile Projekte Standish Group, Chaos Manifesto (Fn. 40) 2013, S. 26f. Siehe zum Projekterfolg nach Vorgehensmodell auch SwissQ/Universität Zürich, Software Development 2015 (Fn. 17), S. 16.

41 Standish Group, Chaos Manifesto 2013 (Fn. 40), S. 4, bezeichnet als «kleine Projekte» solche mit einem Entwicklungsaufwand unter 1 Mio. USD und als «gross» solche mit einem Aufwand von mehr als 10 Mio. USD. Gemäss dieser Studie scheiterten 2012 rund 38% der grossen aber nur 4% der kleinen Projekte. Lediglich 10% der grossen, aber 76% der kleinen Projekte wurden als erfolgreich betrachtet. Die übrigen Projekte wurden zwar erfolgreich abgeschlossen, waren aber mit Zeit- und/oder Kostenüberschreitungen verbunden. Die Methodik der Chaos Studien der Standish Group wurde teilweise kritisiert, weil etwa Kriterien wie die Zufriedenheit der Benutzer und Mehrwert durch Abweichungen von Spezifikationen fehlten. Siehe Eveleens Laurenz J./Verhoef Chris, The rise and fall of the Chaos report figures, IEEE Software, 1/2010, S. 30–36; Glass Robert L., The Standish Report: Does It Really Describe a Software Crisis? Communications of the ACM, 8/2006, S. 15–16. Andere Studien kamen teilweise auf abweichende Werte. Siehe dazu auch El Emam Khaled/Koru A. Günes, A Replicated Survey of IT Software Project Failures, IEEE Software 5/2008, S. 84–90; Reichert (Fn. 1), S. 20. Aufgrund der Anzahl der untersuchten Projekte und der langen Zeitreihe (rund 90'000 Projekte seit 1985) gibt die Chaos Studie der Standish Group aber auf jeden Fall wertvolle Anhaltspunkte zu den Erfolgsfaktoren von IT Projekten. Die Studie von SwissQ/Universität Zürich, Software Development 2015 (Fn. 17), S. 16f, enthält bezogen auf die Schweiz detailliertes Zahlenmaterial zum Projekterfolg nach Komplexität, Grösse und Branche.

42 Siehe zum *Project Scope* Rz. 36.

43 Siehe zur Initialisierung agiler Projekte im Rahmen von HERMES auch Informatiksteuerungsorgan des Bundes, HERMES Referenzhandbuch (Fn. 8), S. 166; Informatiksteuerungsorgan des Bundes, Studie HERMES und Agilität (Fn. 27), S. 12.

44 Siehe zur *Product Vision* Rz. 36. Gemäss SwissQ/Universität Zürich, Software Development 2015 (Fn. 17), S. 27 und 34, verwenden allerdings nur 22.4% der agil geführten Projekte im Rahmen des Requirements Engineering eine Vision. 82.9% arbeiteten aber mit einem Product Backlog, 80% mit User Stories und 58.1% mit Epics. Rund 11–15% des Gesamtprojektaufwandes bzw. 16–20% des Entwicklungsaufwandes der untersuchten Projekte aller Kategorien entfielen auf das Requirements Engineering.

45 Siehe zum *Product Backlog* Rz. 37.

46 Eine *User Story* ist eine in Alltagssprache formulierte funktionale Anforderung an die zu entwickelnde Software. Die Beschreibung umfasst in der Regel nicht mehr als zwei Sätze. Sie kann durch Grafiken visualisiert werden. Zusätzlich können in der User Story Testfälle beschrieben werden (z.B. Beschreibungen eines gewünschten Ergebnisses als «Gutfall» bzw. eines nicht gewünschten Ergebnisses als «Schlechtfall»).

niedergelegt. Anschliessend wird die Priorität der gewünschten Funktionen bestimmt und auf Sprints[47] verteilt. Das Ergebnis jedes Sprints sollte ein Stück funktionsfähige Software sein. In der Regel müssen aber zuerst konzeptionelle Arbeiten geleistet werden (z.B. Festlegung der Architektur, Definition der Schnittstellen, Design der Benutzeroberfläche, Aufbau der Entwicklungsumgebung).

[Rz 19] Die Leistungsbezügerin[48] muss die Entwicklung während des ganzen Projektes mitverfolgen, bewerten und steuern. Agile Methoden setzen hohe fachliche Kompetenz, praktisch permanente zeitliche Verfügbarkeit und rasche Entscheidungswege[49] bei allen Beteiligten voraus. Aus rechtlicher Sicht hat die Leistungsbezügerin bei agil geführten Projektenweit gehende *Mitwirkungspflichten*, deren Verletzung gravierende Konsequenzen haben kann. Da agil geführte Projekte ohne entsprechende Mitwirkung der Leistungsbezügerin gar nicht realisiert werden können, handelt es sich nach der hier vertretenen Auffassung dabei in der Regel um rechtlich verbindliche Pflichten und nicht nur um Obliegenheiten, deren Verletzung lediglich zu Gläubigerverzug und sonstigen Rechtsnachteilen führen kann.[50] Dadurch wird auch die Abgrenzung der Risikosphären[51] durchlässiger. Die Verantwortung für die technische Umsetzung der Anforderungen bleibt aber grundsätzlich bei der Leistungserbringerin.[52]

III. Rollen

[Rz 20] Weil agile Projekte weniger spezifikationsorientiert sind, haben Organisation[53] und Prozesse eine grosse Bedeutung. Ein Scrum Team umfasst folgende *Rollen*:[54]

[Rz 21] Der *Product Owner* ist auf Seiten der Leistungsbezügerin für die strategische Produktentwicklung zuständig. Seine Verantwortung beinhaltet die Konzeption der Produktvision sowie

47 Sprints dauern typischerweise jeweils 1–2 oder 3–4 Wochen.

48 Nach agilen Methoden geführte Projekte können insbesondere auftrags- oder werkvertragsrechtlicher Natur sein. Siehe dazu Rz. 75ff. Daher ist nachfolgend als Oberbegriff jeweils von «*Leistungserbringerinnen*» und «*Leistungsbezügerinnen*» die Rede. Von den Leistungsbezügerinnen (Bestellerinnen, Auftraggeberinnen) sind die *End User* zu unterscheiden (z.B. Mitarbeitende der Leistungsbezügerin oder mit ihr rechtlich oder wirtschaftlich verbundener Organisationen).

49 Gemäss Standish Group, Chaos Manifesto (Fn. 40) 2013, S. 6, fallen pro 1'000 USD Projektkosten durchschnittlich 1.5 Entscheidungen an. Der Executive Sponsor bzw. Product Owner müsse an rund 20% dieser Entscheidungen mitwirken.

50 Siehe dazu Egli (Fn. 19), Rz. 31ff und 48; Fröhlich-Bleuler (Fn. 19), Rz. 352; Reichert (Fn. 1), S. 134; Schneider (Fn. 1), Rz. C 127. Allerdings wird die Durchsetzbarkeit solcher Pflichten durch allfällige Rechte der Leistungsbezügerin zur vorzeitigen Projektbeendigung begrenzt. Siehe dazu auch Rz. 59 und Rz. 95.

51 Siehe zur Risikoverteilung Rz. 92.

52 Fröhlich-Bleuler (Fn. 19), Rz. 366, geht davon aus, dass die Realisierungsverantwortung für das Gesamtprojekt bei Verwendung einer agilen Methodik nicht bei der Leistungserbringerin liegt. Siehe demgegenüber Reichert (Fn. 1), S. 226. Nach der hier vertretenen Auffassung liegt die Gesamtverantwortung nur dann bei der Leistungsbezügerin, wenn diese die Mitarbeitenden der Leistungserbringerin voll in ihre eigene Arbeitsorganisation eingliedert und ihnen entsprechende Weisungen erteilt. Die Verantwortung für das Anforderungsmanagement (was ist in welcher Priorität zu realisieren?) bleibt aber in jedem Fall bei der Leistungsbezügerin.

53 Gemäss SwissQ/Universität St. Gallen, Agile 2013 Studie (Fn. 17), S. 14, verwendeten die befragten Unternehmen unter anderem folgende Führungsstile in agilen Projekten: Selbstorganisation in Scrum Teams: 64%; klassische Projektorganisation: 33.6%; klassische Linienorganisation: 10.2%.

54 Siehe zu den Rollen Schwaber/Sutherland (Fn. 24) S. 4ff. Siehe zur Besetzung der einzelnen Rollen in Projekten auch die statistische Untersuchung von SwissQ/Universität Zürich, Software Development 2015 (Fn. 17), S. 24.

die Festlegung und Priorisierung Produkteigenschaften im Rahmen des Product Backlogs.[55] Ihm obliegt die Kontrolle über Funktionalität, Kosten und Auslieferungszeitpunkte. Er entscheidet – gestützt auf die *Definition of Done (DoD)*[56] – auch darüber, ob die am Ende jedes Sprints gelieferte Funktionalität akzeptabel ist.

[Rz 22] Ein *Entwicklungsteam* besteht in der Regel aus 3–11 Mitgliedern und ist meist interdisziplinär zusammengesetzt. Es organisiert sich selbst. Das Team ist für die Entwicklung der Funktionalität in der vom Product Owner gewünschten Reihenfolge verantwortlich und trägt gemeinsame Verantwortung für die Einhaltung der vereinbarten Qualitätsstandards. Es muss auch den Realisierungsaufwand der Backlogeinträge abschätzen. Dies erfolgt in der Regel im Rahmen von *Estimation Meetings*.[57] Das Entwicklungsteam hat die für einen Sprint ausgewählten Backlogeinträge in technische Arbeitsschritte (*Tasks*) herunterzubrechen. Die Bearbeitung eines Tasks soll in der Regel nicht länger als einen Tag dauern.

[Rz 23] Der *Scrum Master* ist für die Umsetzung der Methodik verantwortlich. Er hat gegenüber dem Entwicklungsteam eine Führungsfunktion, ist aber weder ein fachlicher noch ein Linien-Vorgesetzter:[58] Er soll einzelnen Teammitgliedern keine konkrete Arbeitsanweisungen geben und hat sie nicht zu beurteilen oder zu disziplinieren. Der Scrum Master führt das Team in die Scrum Regeln ein und überprüft deren Einhaltung. Er moderiert die Meetings und kümmert sich um jede Störung des Scrum Prozesses. Zudem führt er den *Impediment Backlog*[59]

[Rz 24] Als Entscheidinstanz kann ein *Lenkungsausschuss* (*Steering Group*)[60] aus Vertretern beider Parteien definiert werden.

[Rz 25] Bereits bei Vertragsschluss kann von beiden Parteien gemeinsam ein unabhängiger *IT Gutachter* bestimmt werden, der im Konfliktfall konsultiert werden kann.

[Rz 26] Auch die *End User* spielen bei agilen Methoden eine wichtige Rolle, da die Funktionalität im Dialog mit ihnen entwickelt werden soll.

[Rz 27] Eventuell sind für ein Projekt noch *weitere Rollen* wie Manager und Sponsoren erforderlich.

[Rz 28] Da der Erfolg oder Misserfolg von nach agilen Methoden geführten Projekten noch in viel höherem Mass als bei klassischen IT Projekten von der Qualität der Teamarbeit abhängt, sollte die Leistungsbezügerin ein *Ablehnungsrecht* für Mitglieder des Entwicklerteams haben.[61] Einem Missbrauch dieses Rechts, welcher eine Realisierung faktisch verhindern könnte, kann durch entsprechende vertragliche Regelungen vorgebeugt werden. Die Ablehnung kann z.B. alternativ vom Vorliegen eines wichtigen Grundes – über welchen gegebenenfalls im Rahmen eines Konflikteska-

55 Siehe dazu auch Reichert (Fn. 1), S. 206. Egli (Fn. 19), Rz. 74, ist demgegenüber der Ansicht, dass die Funktion des Product Owners eher durch einen Vertreter der Leistungserbringerin zu übernehmen sei, da andernfalls das Entwicklungsrisiko für sie zu hoch sei.

56 Siehe zur *Definition of Done* Rz. 33.

57 Siehe zu *Estimation Meetings* Rz. 32.

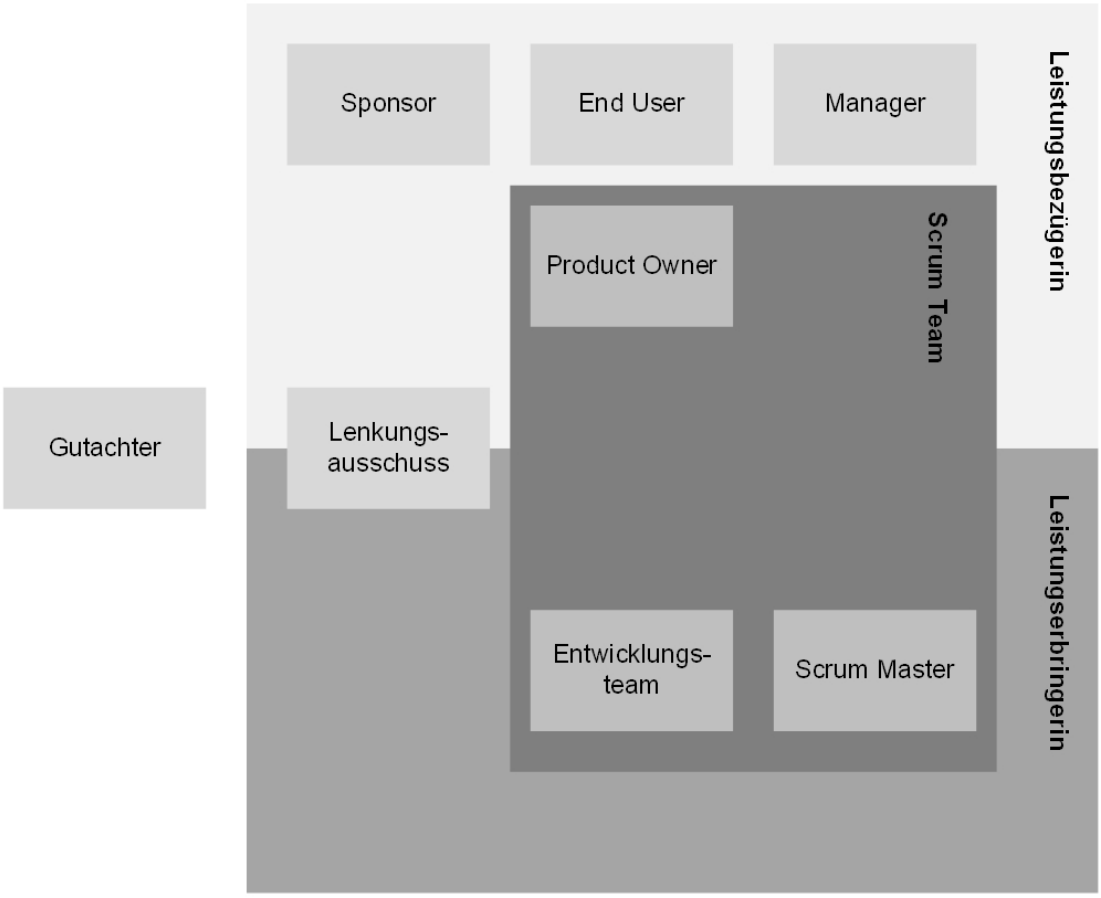
58 Gemäss SwissQ/Universität St. Gallen, Agile 2013 Studie (Fn. 17), S. 14, setzten 22.6% der befragten Unternehmen den Scrum Master auch als Projektleiter ein. 27.7% der Unternehmen wiesen diese Funktion eher dem Product Owner zu. Nach SwissQ/Universität Zürich, Software Development 2015 (Fn. 17) wurde die Rolle des Scrum Master in 61.4% der agil geführten Projekte dediziert innerhalb eines Teams besetzt, in 24.7% der Projekte war ein Scrum Master zuständig für mehrere Teams und in 10.2% wurde die Rolle gar nicht besetzt.

59 Siehe zum *Impediment Backlog* Rz. 42.

60 Mitunter werden statt eines einzigen Lenkungsausschusses auch eine *Scope Steering Group* und eine *Executive Steering Group* vorgesehen. Siehe dazu auch Opelt/Gloger/Pfarl/Mittermayr (Fn. 23), S. 58.

61 Siehe dazu Fröhlich-Bleuler (Fn. 19), Rz. 356.

lationsverfahrens entschieden wird – oder der Bezahlung einer Einarbeitungspauschale abhängig gemacht werden.



Grafische Übersicht 3: Übersicht zu den Rollen bei agilen Entwicklungen

IV. Ereignisse

[Rz 29] Zu den wichtigsten Prozessen agil geführter Projekte gehören regelmässige *Meetings*:[62]

[Rz 30] *Sprint Planning*:[63] Im Rahmen des Sprint Planning wird die Arbeit für den kommenden Sprint geplant. Grundlagen der Planung bilden der *Product Backlog*[64], die bereits vorhandenen

62 Siehe zu den *Meetings* Schwaber/Sutherland (Fn. 24), S. 7ff. Siehe zu den in der Praxis verwendeten Prozessen in agilen Projekten auch die statistische Untersuchung von SwissQ/Universität Zürich, Software Development 2015 (Fn. 17), S. 26f.

63 Sprint Plannings sollten grundsätzlich nicht mehr als 8 Stunden dauern. Sie können unterteilt werden in ein «*Sprint Planning 1*», in welchem die Anforderungen an den Sprint definiert werden und ein «*Sprint Planning 2*», in welchem das Design und die eigentliche Planung des Sprints festgelegt werden. Siehe dazu auch Opelt/Gloger/Pfarl/Mittermayr (Fn. 23), S. 16f.

64 Siehe zum *Product Backlog* Rz. 37.

Product Increments und die Kapazität des Entwicklungsteams, insbesondere dessen Entwicklungsgeschwindigkeit (*Velocity*[65]).

[Rz 31] *Daily Scrum*: In diesem Meeting, welches täglich durchgeführt wird, aber nicht mehr als 15 Minuten dauern sollte, werden die Aktivitäten der Mitglieder des Entwicklungsteams synchronisiert und die Planung für den nächsten Tag erarbeitet.

[Rz 32] *Estimation Meeting*: Der Product Owner und die Teammitglieder aktualisieren periodisch den Product Backlog. Dabei werden der Aufwand für die bereits vorhandenen User Stories geschätzt[66] und gegebenenfalls neue User Stories aufgenommen. Gleichzeitig wird die Realisierungspriorität der Backlog Items aktualisiert.

[Rz 33] Am Ende jedes Sprints steht ein *Sprint Review*, welcher nicht mehr als 4 Stunden dauern sollte. Das Entwicklungsteam präsentiert seine Ergebnisse und prüft, ob das zu Beginn gesteckte Ziel erreicht wurde. In Sprint Reviews sollten auch die End User einbezogen werden, welche die fertige Funktionalität nun testen und benutzen können. Daraus kann sich wichtiges Feedback für die weitere Produktgestaltung ergeben. Es ist aber Aufgabe des Product Owners, die entwickelte Funktionalität zu begutachten und anhand der im *Sprint Planning* festgelegten Bedingungen zu entscheiden, ob sie abgenommen werden kann oder nicht. Wenn ein Product Increment noch nicht der *Definition of Done*[67] entspricht, kann dessen Abnahme verweigert werden. Die entsprechenden User Stories gehen dann meist in den Product Backlog zurück und werden vom Product Owner neu priorisiert.

[Rz 34] Zwischen dem Sprint Review und dem nächsten Sprint Planning findet eine *Sprint Retrospektive* statt. Darin wird analysiert, welche Arbeitsprozesse verbessert werden sollten. Die Analyseresultate sowie potenzielle Realisierungshindernisse werden im *Impediment Backlog* festgehalten.

V. Artefakte

[Rz 35] Im Rahmen von agilen Projekten werden verschiedene *Artefakte* (Arbeitsergebnisse) erstellt:[68]

[Rz 36] Die *Product Vision* steht am Anfang eines agilen Projekts. Sie umfasst eine kurze Beschreibung des *Project Scopes*. Dieser beinhaltet die wesentlichen *Ziele* und *Themen*[69] des Projekts. Sie werden anschliessend in *Epics* heruntergebrochen. Ein Epic umfasst wiederum mehrere *User Stories*. Durch dieses Verfahren findet eine Konkretisierung von einer hohen Abstraktionsebene zu einer technisch umsetzbaren Beschreibung statt.

---

65 Siehe zur Bedeutung der *Velocity* Rz. 57.

66 Zur Schätzung des Aufwandes gibt es für Scrum differenzierte Methoden (z.B. *Planning Poker*). Die einzelnen Teammitglieder sollen rasch und unabhängig von einander Schätzungen vornehmen und sich in einem interaktiven Prozess auf gemeinsame Bewertungen verständigen. Ausgangspunkt ist die Bewertung von Grösse, Aufwand und Entwicklungsrisiken der einzelnen Produktfunktionen mit *Story Points*. Siehe dazu im Einzelnen Cohn (Fn. 34), S. 35ff und S. 49ff; Opelt/Gloger/Pfarl/Mittermayr (Fn. 23), S. 18f und 50ff.

67 Siehe zur Begriff und Bedeutung der *Definition of Done* Rz. 40.

68 Siehe dazu Schwaber/Sutherland (Fn. 24), S. 13ff.

69 Ein *Thema* beinhaltet eine Gruppe von Anforderungen aus betrieblicher Sicht. Die einzelnen Anforderungen werden auf einem hohen Abstraktionsniveau (in der Regel in einem einzigen Absatz) beschrieben.

[Rz 37] Der *Product Backlog* ist eine Liste, welche die funktionalen und nichtfunktionalen Anforderungen[70], die in der Software enthalten sein sollen, in der Reihenfolge ihrer Realisierungspriorität wiedergibt. Darin werden auch alle späteren Änderungsanforderungen aufgenommen. Im Product Backlog sollten zusätzlich Schätzungen des Realisierungsaufwandes und des Nutzwerts der einzelnen *Product Backlog Items* enthalten sein. Er versteht sich aber nicht als eine vollständige Spezifikation. Es sollte aber zwischen verbindlichen Anforderungen und «*nice to have*» Funktionen unterschieden werden, welche nicht zwingend realisiert werden müssen.[71] Die Realisierung erfolgt entsprechend der festgelegten Priorisierung im Rahmen einzelner Sprints.

[Rz 38] Der *Sprint Backlog* dient als Übersicht der innerhalb eines Sprints zu erledigenden Aufgaben. Er entspricht im Detaillierungsgrad etwa einer Feinspezifikation und wird oft in Tabellenform geführt. Die in einem Sprint umzusetzenden User Stories müssen der *Definition of Ready*[72] entsprechen und werden in technische Arbeitsschritte (*Tasks*) heruntergebrochen. Zudem wird in der Regel ein *Sprint Goal* definiert, das die Hauptziele des Sprints beschreibt. Ein *Burndown Chart* kann zur Visualisierung der bereits geleisteten und noch verbleibenden Arbeit dienen.

[Rz 39] Ergebnis eines Sprints sollte ein *Product Increment* sein, d.h. ein Stück lauffähige Software inklusive der erforderlichen Dokumentation[73]. Mehrere Product Increments – oder in grösseren Projekten auch *System Increments* – können jeweils zu einem *Release* zusammengefasst werden.[74]

[Rz 40] Die *Definition of Done* legt fest, unter welchen Bedingungen eine Arbeit als fertig bezeichnet werden kann. Sie enthält insbesondere Qualitätskriterien, nichtfunktionale Anforderungen und Einschränkungen. Es müssen aber nicht alle Elemente der Definition of Done für jede User Story relevant sein.[75] Umgekehrt könnten für einzelne User Stories zusätzlich spezifische Akzep-

---

70 Funktionale Anforderungen werden in der Regel in Form von *User Stories* beschrieben. Der Product Backlog kann zusätzlich auch andere *Product Backlog Items* enthalten (z.B. Beschreibungen nichtfunktionaler Anforderungen wie Sicherheitsanforderungen, Portierbarkeit, Skalierbarkeit). Siehe zur Unterscheidung von funktionalen und nichtfunktionalen Anforderungen im Rahmen des Requirements Engineering auch Egli (Fn. 19), Rz. 11ff; Schneider (Fn. 1), Rz. C 145.

71 Siehe dazu auch Reichert (Fn. 1), S. 139 und S. 206.

72 Die *Definition of Ready* bestimmt, wann ein Backlog Item genügend präzise beschrieben ist, um in ein Sprint Planning aufgenommen werden zu können. Siehe dazu auch Schwaber/Sutherland (Fn. 24), S. 14. Innerhalb des Product Backlogs sollten höher priorisierte Einträge generell einen grösseren Detaillierungsgrad aufweisen. Die Product Backlog Einträge, mit denen sich das Entwicklungsteam im kommenden Sprint beschäftigen soll, müssen so präzise beschreiben werden, dass sie innerhalb des Sprints auf «*done*» gebracht werden können. Nur Product Backlog Einträge, welche diesen Anforderungen entsprechen, werden als «*ready*» angesehen. Siehe dazu auch http://guide.agilealliance.org/guide/definition-of-ready.html.

73 Die Mitlieferung der für die Nutzung der Software erforderlichen Dokumentationen gehört grundsätzlich zu den Leistungspflichten der Entwickler. Siehe dazu auch BGE 124 III 456 E. 4b/aa sowie den Entscheid des Handelsgerichts des Kantons Zürich vom 06. September 1996. Entsprechend den Maximen des *Agile Manifesto* wird die Dokumentation in agil geführten Projekten oft auf ein Minimum beschränkt (z.B. Kommentierung im Quellcode und Online Hilfsfunktion für die Benutzenden). Siehe dazu auch Fröhlich-Bleuler (Fn. 19), Rz. 357; Reichert (Fn. 1), S. 247ff; Redeker Helmut, Realisierung (Kauf-/Werkvertrag), in: Schneider Jochen/von Westphalen Friedrich (Hrsg.), Software-Erstellungsverträge: Projektgestaltung, Vertragstypen, Rechtsschutz, 2. A., Köln 2014, S. 319–452, Rz. D 111f. Hingegen folgt aus der Anwendung einer agilen Methodik nicht ohne weiteres ein Verzicht auf jegliche Dokumentation. Siehe demgegenüber Egli (Fn. 19), Rz. 23.

74 Siehe zu System Increments und Releases in grösseren Projekten auch http://www.scaledagileframework.com/release/.

75 Bei der Definition of Done geht es grundsätzlich um allgemeine Qualitätskriterien (z.B. Konformität mit Architekturbeschreibung und Schnittstellendefinitionen, Einhaltung der Programmierrichtlinien und Software-Qualitätskriterien, Aufrechterhaltung der Funktionalität vorheriger Product Increments, Antwortzeiten gemäss Anforderungen des Produktivsystems, Lauffähigkeit der Software auf der Testumgebung, Testen der Funktionalität bis zur System- und Plattformebene, Release Notes). Die Definition of Done kann durch *Criteria of Satisfaction* ergänzt werden, welche z.B. durch Prämien honoriert werden. Gemäss der Studie SwissQ/Universität Zürich, Software Development 2015 (Fn. 17), S. 27, werden allerdings nur rund in der Hälfte der agil geführten Projekte eine Definition of Done und Akzeptanzkriterien für User Stories verwendet. Gemäss dieser Studie, S. 44, betrugt

tanzkriterien definiert werden. Die Definition of Done wird zu Beginn des Projektes erstellt und im Laufe der Entwicklung angepasst. Sie muss zwischen den Parteien ausgehandelt werden. Mit zunehmender Erfahrung des Scrum Teams steigen in der Regel auch die Anforderungen an die Qualität.

[Rz 41] Der *Release Plan* dient in grösseren Entwicklungsprojekten der Planung und Einführung von Releases – welche typischerweise 3–12 Product Increments umfassen.

[Rz 42] Der *Impediment Backlog* ist eine Liste aller Hindernisse, die im Rahmen der Entwicklung erkannt wurden.

[Rz 43] In der Regel gibt es gemeinsame *Inspektionspunkte* am Beginn und am Ende eines Sprints sowie nach Ablieferung eines Release. Diese können vertraglich als Entscheidungspunkte in der Form von Meilensteinen ausgestaltet werden, z.B. für Projektänderungen oder einen Projektstop (*Exit Points*). Beide Parteien müssen zu diesen Zeitpunkten für Reviews und Entscheidungen zur Verfügung stehen.

VI. Vergütungsmodelle

[Rz 44] Es gibt unterschiedliche *Vergütungsmodelle* für agile Projekte. Das gewählte Modell sollte insbesondere der Risikoverteilung zwischen den Parteien Rechnung tragen.[76] In der Praxis[77] haben sich neben herkömmlichen Formen wie Vergütung nach Aufwand oder Festpreis auch spezifische Ansätze für agil geführte Projekte herausgebildet. Nachfolgend wird ein Überblick über gängige Vergütungsmodelle gegeben.

[Rz 45] «*Time & Material*»: Vergütung aller Leistungen – oder nur der Initialisierungsphase – nach Aufwand. Das Kostenrisiko liegt hier ganz bei der Leistungsbezügerin. Es kann allenfalls durch Kostendächer limitiert werden. Dieser Ansatz entspricht in der Regel einer rein auftragsrechtlichen Konzeption. Vergütungen nach Aufwand sind aber auch im Rahmen von werkvertraglichen Leistungen möglich (z.B. für einzelne Sprints). Vertraglich ist insbesondere zu regeln, ob eine Vergütung auch geschuldet ist, wenn ein Product Increment nicht der Definition of Done entspricht, bzw. wer Nachbesserungsarbeiten bezahlt.

[Rz 46] Eine Variante stellt das Vergütungsmodell «*Proviant und Prämie*» dar: Die Leistungsbezügerin bezahlt Leistungen nach Aufwand zu einem relativ niedrigen Stundensatz. Zusätzlich erhält die Leistungserbringerin aber eine Prämie, wenn bestimmte zum voraus definierte Kriterien (z.B. *Criteria of Satisfaction*) erfüllt werden.[78]

---

der Testaufwand bei den untersuchten Projekten aller Kategorien rund 16–20% des Gesamtprojektaufwandes bzw. 21–30% des Entwicklungsaufwandes.

76 Siehe dazu auch Fröhlich-Bleuler (Fn. 19), Rz. 358. Egli (Fn. 19), Rz. 57ff unterscheidet die Projektrisiken grundsätzlich nach Entwicklungsrisiko, Terminrisiko und Kostenrisiko. Aus einer werkvertraglichen Tragung des Entwicklungsrisikos durch die Leistungserbringerin folgt nicht unbedingt auch eine Übernahme des Kostenrisikos. Vielmehr sieht Art. 374 OR – sofern vertraglich nichts anderes vereinbart wurde – eine Berücksichtigung des Aufwandes bei der Bemessung der Vergütung vor. Siehe zur Bewertung unterschiedlicher *Angebotsmodelle* auch Opelt/Gloger/Pfarl/Mittermayr (Fn. 23), S. 15ff.

77 Gemäss SwissQ/Universität St. Gallen, Agile 2013 Studie (Fn. 17), S. 11, verwendeten die befragten Unternehmen folgende Vergütungselemente für agile Projekte: «Time & Material»: 33.6%; Festpreis: 26.3%; «Money for Nothing, Changes for Free»: 6.6%.

78 Siehe dazu auch Fröhlich-Bleuler (Fn. 19), Rz. 358.

[Rz 47] Auch agil geführte Projekte können zu einem *Festpreis* realisiert werden, wenn der Scope definiert ist. Die Leistungserbringerin trägt in diesem Fall aber ein hohes Kostenrisiko. Um den Bedürfnissen beider Parteien Rechnung zu tragen, hat sich in der Praxis ein spezifisches Modell zur Kostenschätzung und Preisbestimmung herausgebildet, das als «*Agiler Festpreis*»[79] bezeichnet wird:

[Rz 48] In einem ersten Schritt wird der *Vertragsgegenstand grob beschrieben* (*Project Scope*[80]). Zudem wird der rechtliche Rahmen vereinbart.

[Rz 49] Anschliessend wird ein für das Projekt *repräsentatives Epic ausgewählt* und bis zur Ebene der User Stories detailliert spezifiziert. Das Epic sollte eine ausreichende Anzahl an User Stories mit unterschiedlichem Funktionsumfang enthalten, die als *Reference User Stories* gelten können.

[Rz 50] In einem Workshop schätzen die Parteien dann anhand der Reference User Stories und der anderen Epics mit Hilfe von *Story Points* gemeinsam die Aufwände für das gesamte Projekt.[81] Zudem werden die Implementierungsrisiken und der Nutzen geschätzt. Daraus ergibt sich ein *indikativer Festpreisrahmen*. Dieser ist noch nicht bindend.

[Rz 51] Im nächsten Schritt wird die *Checkpoint Phase* festgelegt (typischerweise 2–5 Sprints). Diese dient als Testphase für die Zusammenarbeit. Anhand der während der Checkpoint Phase realisierten Story Points kann die Entwicklungsgeschwindigkeit des Teams (*Velocity*) geschätzt werden.[82] Dabei sind einerseits Anlaufschwierigkeiten mit zu berücksichtigen, welche die Produktivität bremsen. Andererseits ist zu bedenken, dass das Tempo dauerhaft gehalten werden soll. Die Checkpointphase kann nach Aufwand oder zu einem zum voraus festgelegten Betrag vergütet werden. Mitunter wird dafür auch ein reduzierter Ansatz angewendet oder ein Teil der Vergütung wird erst im Fall einer erfolgreichen Realisierung weiterer Phasen ausbezahlt.

[Rz 52] Am Ende der Checkpoint Phase überprüfen die Parteien die Definition des Scope und die anfangs gemachten Annahmen. Sie entscheiden gestützt darauf, ob sie das Gesamtprojekt wie geplant umsetzen wollen. Aus der – nun bereits genauer geschätzten – Anzahl Story Points, der Velocity und dem Preis pro Story Point ergibt sich ein Zeit- und Kostenrahmen, welcher *vertraglich verbindlich fixiert wird*. Spätestens in diesem Zeitpunkt ist auch die Verteilung der Risiken[83] zu klären und es ist festzulegen, wie Mehr- oder Minderaufwand – z.B. nach einem bestimmten Verteilschlüssel (*Riskshare*[84]) – getragen werden.

[Rz 53] Anders als klassische Festpreisprojekte darf ein Projekt mit agilem Festpreis *beendet* werden, wenn die Leistungsbezügerin den erwarteten Nutzen durch die bereits erfolgten Lieferungen als erfüllt erklärt.[85] Dies kann bereits der Fall sein, bevor die gesamte Funktionalität umgesetzt wurde. Als Ausgleich für den entgangenen Auftragswert kann der Leistungserbringerin ein be-

79 Siehe dazu im Einzelnen Opelt/Gloger/Pfarl/Mittermayr (Fn. 23), S. 43ff; Egli (Fn. 19), Rz. 71.

80 Siehe zum Begriff des *Project Scope* Rz. 36.

81 Mit *Story Points* können Aufwand und Risiken der Umsetzung einzelner User Stories bewertet werden. Siehe zur Bedeutung von Story Points auch Cohn (Fn. 34), S. 36f und S. 69ff.

82 Siehe zur Bedeutung der *Velocity* auch Cohn (Fn. 34), S. 38f.

83 Siehe zur Risiko- und Chancenverteilung auch Rz. 92.

84 Der *Riskshare* ist ein Verteilschlüssel, wie Mehr- oder Minderaufwand von den Parteien zu tragen ist. Siehe auch Opelt/Gloger/Pfarl/Mittermayr (Fn. 23), S. 47.

85 Bei der Projektbeendigung ist allerdings auch an Wartung und Support der bereits entwickelten Software zu denken. Siehe dazu auch Rz. 58.

stimmter Prozentsatz des Restumfanges ausbezahlt werden oder es kann ihr ein neuer Auftrag im entsprechenden Umfang erteilt werden.

[Rz 54] Eine weitere Variante besteht in der Vereinbarung von *Stückpreisen*: Alle Aufwände werden in Story Points[86] geschätzt. Pro umgesetzten Story Point wird jeweils ein Fixbetrag bezahlt.

[Rz 55] In der Praxis haben sich zudem verschiedene Prinzipien zur Risiko- und Aufwandsverteilung für agil geführte Projekte herausgebildet. Durch die Vereinbarung solcher Prinzipien kann das gewählte Vergütungsmodell ergänzt werden:

[Rz 56] «*Money for nothing*»: Das Projekt kann durch die Leistungsbezügerin jederzeit beendet werden, z.B. weil bereits genügend Funktionalität umgesetzt wurde oder Kosten und Nutzen bei einer Fortsetzung des Projekts für sie nicht mehr in einem vernünftigen Verhältnis stehen würden. Für diesen Fall kann vertraglich vorgesehen werden, dass das ungenutzte Restbudget in einem zum voraus definierten Verhältnis zwischen beiden Parteien aufgeteilt wird, z.B. gemäss einem Riskshare[87] von 80:20.

[Rz 57] «*Change for free*»: Einzelne Features können solange mit ihrer Umsetzung noch nicht begonnen wurde[88] auch im Fall eines Festpreises ohne Zusatzkosten gegen andere mit gleichwertiger Komplexität ausgetauscht werden. Mehraufwand ausserhalb des Project Scope geht grundsätzlich zulasten der Leistungsbezügerin. Zusatzaufwand innerhalb des Project Scope sollen primär durch Reduktion der Komplexität oder Verzicht auf bereits definierte User Stories ausgeglichen werden. Ist dies nicht möglich, so werden die Mehrkosten zwischen den Parteien nach einem zum voraus definierten *Riskshare* aufgeteilt. Dieses Modell kann zusätzlich durch eine Toleranz (*Safety Charge)* ergänzt werden, innerhalb derer die Leistungsbezügerin bereit ist, Zusatzaufwand zu tragen. Ob ein Change innerhalb oder ausserhalb des Scopes liegt, ist gegebenenfalls im Rahmen eines vertraglich definierten Konflikteskalationsverfahrens zu bestimmen.

[Rz 58] «*Geld zurück mit Ausstieg*»: Die Leistungsbezügerin hat das Recht, Abnahme und Bezahlung einer zum voraus bestimmten Anzahl von Product Increments (in der Regel 1–3) ohne Angabe von Gründen zu verweigern. Die Leistungserbringerin übernimmt damit das auf eine bestimmte Anzahl Sprints limitiertes Risiko, Software zu liefern, die der Leistungsempfängerin gefällt. Dieses Vertragsmodell setzt in besonders hohem Mass eine kooperative Partnerschaft voraus. Die Leistungsbezügerin kann das Projekt ganz beenden oder durch Dritte weiterführen lassen. Allerdings ist in komplexen Projekten mitunter nur die ursprüngliche Entwicklerin in der Lage, Weiterentwicklungen, Wartung und Support effizient zu erbringen. Vertraglich sollte insbesondere geregelt werden, wem die Rechte an nicht vergüteten Product Increments gehören.

[Rz 59] «*Geld zurück mit Nachbesserungen*»: Die Leistungsbezügerin darf die Abnahme eines Product Increments verweigern, ohne dass sie das Vorhandensein eines eigentlichen Abnahmeverweigerungsgrundes belegen muss.[89] Wenn von der Leistungsbezügerin geltend gemachte Män-

86 Siehe zur Bedeutung von *Story Points* auch Rz. 32 und 50.

87 Siehe zum Begriff des *Riskshare* Rz. 52.

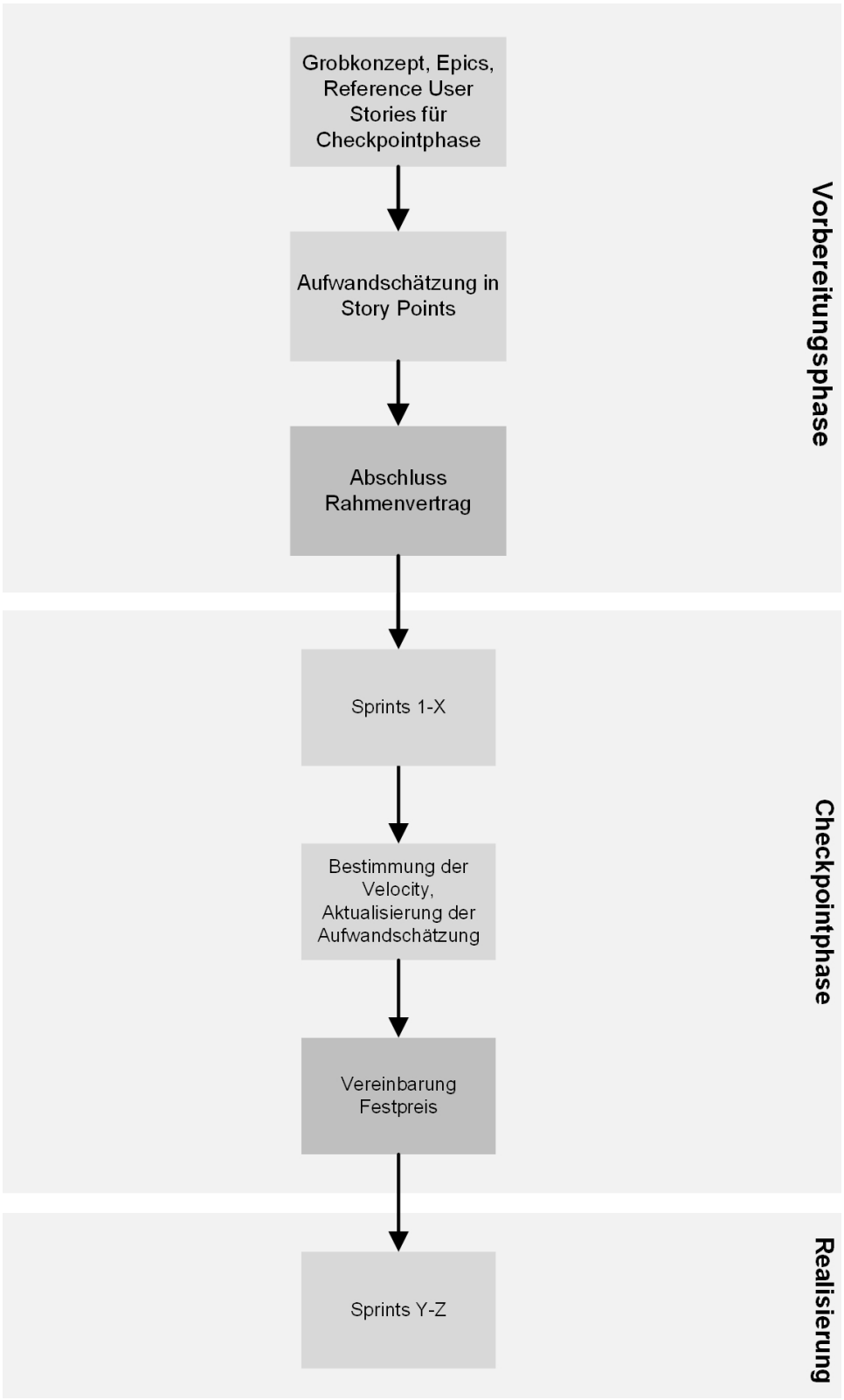
88 Während eines Sprints sollte es grundsätzlich keine Changes geben – dies ist auch ein Grund, weshalb Sprints kurz sind. Änderungen von User Stories für spätere Sprints erfolgen in der Regel nicht über ein herkömmliches Change Management Verfahren, sondern über Anpassungen des Product Backlog (z.B. im Rahmen des nächsten Estimation Meetings).

89 Bei werkvertraglichen Leistungen ist zwischen gewöhnlichen Mängeln und Abnahmeverweigerungsgründen im Sinn von Art. 368 Abs. 1 OR zu unterscheiden. Die Abgrenzung bereitet in der Praxis immer wieder Schwierigkeiten. Die Beweislast für das Vorliegen eines Abnahmeverweigerungsgrundes kann für die Leistungsbezügerin durch das Modell «Geld zurück mit Nachbesserungen» vermieden werden.

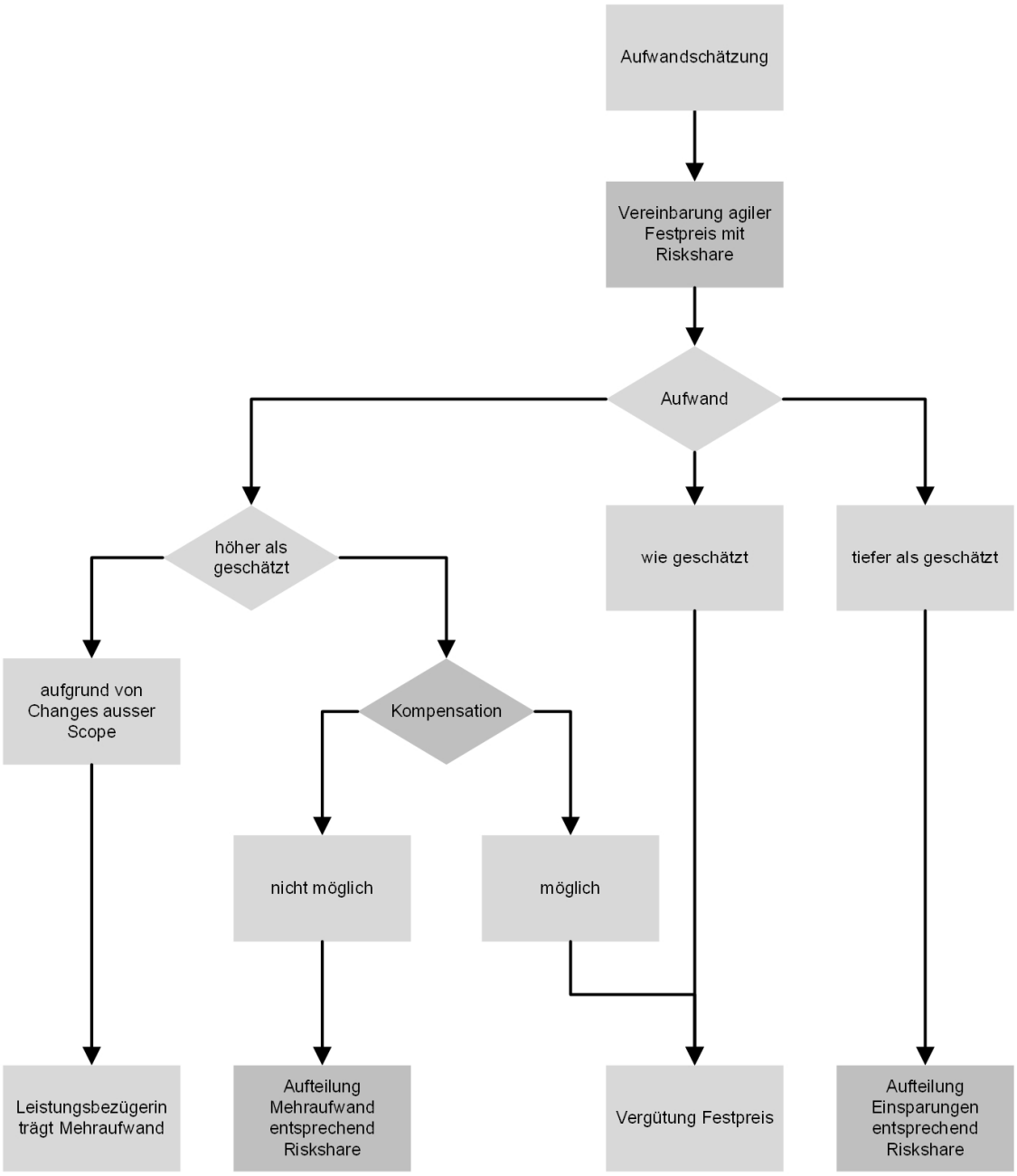
gel in einem der nächsten Sprints erfolgreich behoben werden, so hat die Leistungserbringerin Anspruch auf die ursprüngliche Vergütung. Ab einer bestimmten Anzahl von Abnahmeverweigerungen (z.B. 3 Verweigerungen) steht beiden Parteien ein Recht zur sofortigen Vertragsbeendigung zu.

[Rz 60] «*Ordnung von Termin, Kosten, Qualität und Scope*»: Im Gegensatz zu herkömmlichen «Festpreis»-Projekten, in denen ein vorgängig definierter Scope zu einem fixen Preis, zu einem festgelegten Zeitpunkt und in einer bestimmten Qualität umgesetzt werden soll, wird in diesem Modell von den Parteien die Priorisierung der Projekt-Eckpunkte «Termin», «Kosten», «Qualität» und «Scope» gemeinsam festgelegt.[90] Auf bestimmte Entscheidungspunkte hin (z.B. Ende eines Sprints oder Lieferung eines Release) werden die Projekt-Eckpunkte beurteilt. Ist das oberste Ziel erreicht, wird entschieden, ob das Projekt abgeschlossen wird oder ob noch weitere Ziele umgesetzt werden sollen.

90 Dieses Vertragsmodell geht vom «*magischen Viereck der Zielkonflikte*» aus, wonach Zeit, Kosten, Funktionalität und Qualität nicht gleichzeitig optimiert werden können.



Grafische Übersicht 4: Checkpointphase



Grafische Übersicht 5: Tragung von Mehr- oder Minderaufwand bei Projekten mit agilem Festpreis

VII. Qualifikation von Softwareentwicklungsverträgen

[Rz 61] Die *vertragstypologische Einordnung* von Softwareentwicklungsverträgen ist grundsätzlich nicht von der Projektmethodik abhängig, sondern von der Definition des Leistungsgegenstandes und dem Parteiwillen:

[Rz 62] Die *Herstellung von Individualsoftware nach gemeinsam festgelegten Spezifikationen* fällt nach der hier vertretenen Auffassung generell unter das Werkvertragsrecht, da in diesem Fall ein konkretes Arbeitsergebnis zu erbringen ist.[91] Wurde hingegen bloss ein Tätigwerden auf Anweisung der Leistungsbezügerin ohne Ergebnisverantwortung vereinbart, so ist Auftragsrecht darauf anwendbar.[92] Etwas verkürzt gesagt kommt es darauf an, ob ein *Werk* oder ein *Wirken* geschuldet ist bzw. ob das Entwicklungsrisiko von der Leistungserbringerin oder von der Leistungsbezügerin getragen werden soll. Die Abgrenzung zwischen Auftrag und Werkvertrag kann bei Entwicklungsprojekten mit offenen Zielvorgaben allerdings schwierig sein.[93]

[Rz 63] Auch die Erstellung von *Konzepten, Plänen und Studien* kann je nach dem konkreten Vertragsinhalt unter Werkvertragsrecht oder unter Auftragsrecht fallen.[94] Handelt es sich bei solchen Dokumenten eher um schriftlich festgehaltene Beratungsergebnisse, spricht dies für die Anwendung von Auftragsrecht. Stellen sie hingegen eine selbständige Teiletappe für ein Projekt dar, drängt sich eher Werkvertragsrecht auf.[95] Mitunter bildet die Erarbeitung von Spezifikationen ein eigenständiges Projekt (mit Auftrags- oder Werkvertragscharakter), gestützt auf dessen Ergebnisse ein Werkvertrag für die Realisierung geschlossen wird.

[Rz 64] *Planungs-, Beratungs- und Projektmanagementleistungen* fallen grundsätzlich unter Auftragsrecht.[96] Erfolgen sie im Rahmen eines *umfassenden Werkvertrages*, gilt grundsätzlich einheitlich Werkvertragsrecht. Allerdings können nach der Rechtsprechung des Bundesgerichtes zum Architektenvertrag einzelne Komponenten innerhalb eines komplexen Vertragsverhältnisses unter Umständen ihren Auftragscharakter bewahren.[97]

[Rz 65] Softwareentwicklungen können auch im Rahmen von gemischten Verträgen oder von *Innominatkontrakten* erfolgen, auf welche werk- oder auftragsrechtliche Bestimmungen höchstens analog anwendbar sind (z.B. Individualisierung von Software, welche der Leistungsbezügerin nur als *Software-as-a-Service* zur Verfügung gestellt werden soll).

---

91 Siehe dazu BGE 4A.265/2008 vom 26. August 2008, E. 2.2, BGE 4C.393/2006 vom 27. April 2007 E. 3.1, zurückhaltend BGE 124 III 456, E. 4b/bb; Gauch Peter, Der Werkvertrag, 5. A., Zürich 2011, Rz. 334ff, mit weiteren Hinweisen; Fröhlich-Bleuler (Fn. 19), Rz. 392; Heusler Bernhard/Mathys Roland, IT-Vertragsrecht: Praxisorientierte Vertragsgestaltung in der Informationstechnologie, Zürich 2004, S. 24ff; Widmer Ursula, Risikofolgeverteilung bei Informatikprojekten: Haftung für Softwaremängel bei Planung und Realisierung von Informationssystemen, unter besonderer Berücksichtigung der Rechtsnatur des Entwicklungsvertrags für Informationssysteme, Zürich 1990, zugl. Diss. Bern 1990, S. 46ff; Egli (Fn. 19), Rz. 41ff und 58ff; Zindel/Pulver, Basler Kommentar, N. 19 vor Art. 363–379 OR. Slongo Wagen Doris, Der Softwareherstellungsvertrag, Zürich 1991, zugl. Diss. Zürich 1991, S. 155f; Rauber Georg, IT-Projektvertrag, in: Jörg Florian S./Arter Oliver (Hrsg.), Internet-Recht und IT-Verträge, Bern 2005, S. 221–260, S. 229ff; und Bühler Theodor, Zürcher Kommentar, N. 104 zu Art. 363 OR betrachten Softwareherstellungsverträge hingegen grundsätzlich als Innominatkontrakte *sui generis*.

92 Siehe dazu auch Fröhlich-Bleuler (Fn. 19), Rz. 405f; Heusler/Mathys, S. 26ff; Egli (Fn. 19), Rz. 51.

93 Als weitere Indizien für die Abgrenzung zwischen Werkvertrag und Auftrag kommen allenfalls der Grad des Vertrauensverhältnisses zwischen den Parteien und die Personenbezogenheit der Leistung in Betracht. Siehe dazu Gauch (Fn. 91), Rz. 22. Allerdings lassen sich auch diese Kriterien in Bezug auf IT Verträge nur beschränkt objektivieren.

94 Siehe dazu Egli (Fn. 19), Rz. 44; Slongo Wagen (Fn. 91), S. 115; Schaub Rudolf P., Der Engineeringvertrag: Rechtsnatur und Haftung, Zürich 1979, zugl. Diss. Bern 1979, S. 80ff; differenzierend Gurovits András, EDV-Beratungsverträge, Zürich 1993, zugl. Diss. Zürich 1993, S. 67ff.

95 Das Bundesgericht hat bei Gutachten darauf abgestellt, ob diese objektiv überprüfbar sind oder ob sie weitgehend auf subjektiven Wertungen beruhen. Siehe BGE 127 III 328 E. 2, wo es um eine Preisschätzung ging sowie BGE 4A.51/2007 vom 11. September 2007, E. 4.4, wo ein Optimierungskonzept im Finanzbereich zu beurteilen war. Siehe zur rechtlichen Qualifikation erfolgsbezogener IT Beraterverträge auch Morscher Lukas, Leistungsbeschrieb, Gewährleistung und Haftung in IT-Verträgen, in: Jörg Florian S./Arter Oliver (Hrsg.), IT-Verträge, 10. Tagungsband, Bern 2007, S. 86.

96 Siehe dazu Gurovits (Fn. 94), S. 85ff; Fröhlich-Bleuler (Fn. 19), Rz. 492ff; Egli (Fn. 19), Rz. 51.

97 Siehe dazu BGE 109 II 466 E. 3.

[Rz 66] Phasenmodelle unterscheiden grundsätzlich zwischen einer *Spezifikations- und einer Realisierungsphase*. Am Ende der Realisierungsphase erfolgt typischerweise ein Abnahmeverfahren, in welchem die Konformität des Arbeitsergebnisses mit den Spezifikationen verglichen wird. Danach beginnt in der Regel die Betriebs- und Wartungsphase. Bei grösseren Projekten wird dieses Grundmodell allerdings kaum je in Reinform angewendet, sondern meist durch weitere Elemente ergänzt:

[Rz 67] Die Spezifikationsphase kann weiter in die Erarbeitung von *Grob- und Feinspezifikationen* unterteilt werden.

[Rz 68] Unter Umständen erfolgt zusätzlich ein *Proof of Concept* oder die Entwicklung eines *Prototyps*, bevor die Realisierung freigegeben wird.

[Rz 69] Grössere Projekte werden in der Regel in *Teilprojekte* untergliedert, welche jeweils mit Teilabnahmen abschliessen.

[Rz 70] Meist werden zusätzliche *Meilensteine* vereinbart, bis zu welchen, Zwischenresultate geliefert werden müssen.

[Rz 71] Auch bei der Verwendung eines Phasenmodells müssen die Spezifikationen während der Realisierung in der Regel noch verfeinert und ergänzt werden. Eigentliche Änderungen sind grundsätzlich nur im Rahmen des *Change Managements* möglich und meist mit Zusatzaufwand verbunden. Die Abgrenzung zwischen blossen Präzisierungen und Änderungen der Spezifikationen erweist sich in der Praxis allerdings immer wieder als schwierig.

[Rz 72] Unter Umständen erfolgt für das gesamte Projekt oder während der Implementierungsphase eine *externe Qualitätssicherung*.

[Rz 73] *Abnahmeverfahren* werden oft in Teil- und Gesamtabnahmen oder in *Factory Acceptance Tests*, *Site Acceptance Tests* und *User Acceptance Tests* untergliedert.

[Rz 74] Nach einer erfolgreichen Abnahme erfolgen oft *weitere Leistungen* (z.B. Datenmigration aus früheren Systemen, Einführung und Schulung, Betriebs- und Wartungsleistungen).

VIII. Qualifikation von Verträgen für agile Projekte

[Rz 75] Vertraglich können agile Projekte auf unterschiedliche Weise ausgestaltet werden:[98]

[Rz 76] Nach der hier vertretenen Auffassung stellen Verträge über agil geführte Projekte nicht automatisch *Dauerschuldverhältnisse* dar.[99] Wenn gleichartige Sprints über eine bestimmte Zeitdauer hinweg vereinbart werden, so bildet dies aber immerhin ein Indiz für einen Dauercharakter – umso wichtiger ist eine präzise vertragliche Regelung der Beendigungsmodalitäten.

---

98 Siehe dazu auch Fröhlich-Bleuler (Fn. 19), Rz. 346ff; Hengstler (Fn. 23), S. 113ff. Gemäss SwissQ/Universität St. Gallen, Agile 2013 Studie (Fn. 17), S. 11, verwendeten die befragten Unternehmen insbesondere folgende Vertragselemente für agile Projekte: Vergütung nach «Time & Material»: 33.6%; Festvertrag pro Feature: 5.1%; Festvertrag pro Sprint: 4.4%. 41.6% der befragten Unternehmen an, agile Projekte gestützt auf interne Projektvereinbarungen realisiert zu haben – es waren bei dieser Frage aber Mehrfachnennungen möglich. Tatsächlich dürften agile Methoden besonders oft bei firmeninternen Entwicklungen zur Anwendung kommen. Siehe dazu auch Egli (Fn. 19), Rz. 26.

99 Siehe dazu auch Reichert (Fn. 1), S. 270ff.

[Rz 77] Die gesamte Realisierung kann *auftragsrechtlich* erfolgen. Auftragsrecht dürfte vor allem dann zur Anwendung kommen, wenn eine Ergebnisverantwortung der Leistungserbringerin ausdrücklich ausgeschlossen wird.[100]

[Rz 78] Nach der hier vertretenen Auffassung ist aber grundsätzlich auch eine *werkvertragliche Realisierung* eines agil geführten Gesamtprojekts möglich. Werkverträge setzen nicht voraus, dass das zu erstellende Werk bereits zum voraus detailliert beschrieben wird, sondern lediglich, dass sein Inhalt bestimmbar ist, so dass am Ende der Realisierung die Vertragskonformität im Rahmen eines Abnahmeverfahrens überprüft werden kann.[101] Auch bei nach Phasenmodellen geführten Projekten kann die Spezifizierung einer Software oder eines IT Systems im Rahmen des Requirement Engineering einen wesentlichen Teil der werkvertraglichen Leistung ausmachen. Die Konkretisierung der Leistungsinhalte für agile Projekte erfolgt einerseits durch Auslegung und Lückenfüllung vorhandener vertraglicher Vereinbarungen (z.B. Project Scope, Product Vision, initialer Product Backlog), andererseits im Rahmen der vertraglich definierten Prozesse (insbesondere Fortschreibung des Product Backlog). Aus der Anwendbarkeit von Werkvertragsrecht folgt indessen nicht automatisch ein Festpreis. Falls ein fester Zeit- und Kostenrahmen vereinbart wird, trägt die Leistungserbringerin hohe Projektrisiken. Sie liefert sich aber nicht einem einseitigen Leistungsbestimmungsrecht der Leistungsbezügerin aus. Wenn diese überzogene Anforderungen stellt, hat die Leistungserbringerin sie allerdings abzumahnen.

[Rz 79] Denkbar ist auch eine auftragsrechtliche – oder allenfalls auch werkvertragliche[102] – Ausgestaltung der *Initialisierungsphase* (z.B. bis zum Ende einer Checkpointphase) mit anschliessender werkvertraglicher Realisierung.

[Rz 80] Es ist zudem möglich, nur für einzelne Product Increments oder Releases *werkvertragliche Einzelverträge* zu schliessen. Es fragt sich, ob in diesem Fall bezüglich der *Rahmenbedingungen* auch werkvertragliche Bestimmungen zur Anwendung kommen (z.B. bezüglich der Gewährleistungsfristen und der Auflösung). Nach der hier vertretenen Auffassung ist dies durchaus möglich, im Rahmen der Vertragsfreiheit allerdings nicht zwingend.[103]

---

100 Siehe dazu auch Reichert (Fn. 1), S. 270, zurückhaltend Egli (Fn. 19), Rz. 69. Denkbar ist zudem die Realisierung im Rahmen einer Personalstellung; siehe dazu auch Egli (Fn. 19), Rz. 50ff.

101 Siehe dazu Gauch (Fn. 91), Rz. 382, wonach der Werkinhalt genügend bestimmbar ist, wenn es in den Grundzügen vertraglich umschrieben ist und die noch offenen Einzelheiten der Werkausführung sich nach den Umständen, nach einem Leistungsziel (funktionale Leistungsbeschreibung) oder nach späterer Übereinkunft (Art. 2 OR) richten. Agile Methoden legen zudem spezifische Verfahren fest, wie die Leistungsbeschreibung schrittweise gemeinsam konkretisiert werden kann – die Offenheit der Spezifikation wird hier durch die Definition eines Prozesses zur weiteren Leistungsbestimmung ergänzt. Egli (Fn. 19), Rz. 43ff und 74f, geht davon aus, dass auch Verträge über agile Softwareprojekte grundsätzlich werkvertraglicher Natur sind. Unter dem deutschen Recht generell zurückhaltend zu einer werkvertraglichen Qualifikation Schneider (Fn. 1), Rz. C 154; Lejeune Mathias, Internationale Softwareerstellungsverträge, in: Schneider Jochen/von Westphalen Friedrich (Hrsg.), Software-Erstellungsverträge: Projektgestaltung, Vertragstypen, Rechtsschutz, 2. A., Köln 2014, S. 575–600, Rz. F 19. Reichert (Fn. 1), S. 215, ist der Auffassung, dass für Projekte auf Basis von Extreme Programming, DSDM oder Crystal Clear mit nur grober oder unvollständiger Planung keine Werkverträge in Betracht kommen, dass sie bei anderen agilen Methoden aber möglich sind. Eine werkvertragliche Realisierung setzt zudem eine gewisse unternehmerische Freiheit voraus, die nicht gewährleistet ist, falls die Leistungserbringerin ausschliesslich nach den Weisungen der Leistungsbezügerin arbeiten muss.

102 Die Erarbeitung von Planungsunterlagen kann grundsätzlich auch werkvertraglich ausgestaltet werden. Siehe dazu Rz. 63 sowie Reichert (Fn. 1), S. 256. Allerdings hat die Leistungsbezügerin auch in diesem Fall wesentliche Mitwirkungspflichten, da Backlog, Releaseplanung etc. nicht alleine von der Leistungserbringerin erarbeitet werden können.

103 Fröhlich-Bleuler (Fn. 19), Rz. 346, geht davon aus, dass die Rahmenbedingungen in agilen Verträgen grundsätzlich auftragsrechtlichen Charakter haben, da der Vertragsgegenstand zu Beginn des Projekts noch nicht genau bestimmt ist. In Rahmenverträgen können z.B. lediglich Mindestanforderungen an das Produkt (insbesondere Mindestfunktionalität, nichtfunktionale Anforderungen, Beschreibung des Einsatzgebietes und der Rahmenbedingun-

[Rz 81] Ausnahmsweise kommt auch die Begründung eines *Gesellschaftsverhältnisses* in Betracht, wenn die zusammen entwickelte Software gemeinsam am Markt verwertet werden soll.[104] In der Regel fehlt es aber am Erfordernis der Verfolgung eines gemeinsamen Zwecks.

[Rz 82] Verträge über agile Entwicklungen können auch den Charakter von *gemischten Verträgen* haben, z.B. Werkverträge mit auftragsrechtlichen Komponenten.[105] Denkbar ist zudem eine Qualifikation als *Innominatkontrakte*, auf welche werkvertragliche oder auftragsrechtliche Bestimmungen nur analog anwendbar sind.[106]

IX. Vertragsgestaltung für agile Modelle

[Rz 83] Die Rahmenbedingungen der Zusammenarbeit sollten zum voraus vertraglich festgelegt werden. Oft werden sie in einem *Rahmenvertrag* definiert. Darin kann bezüglich der Rollen, Prozesse und Ergebnisse ergänzend auch auf externe Dokumente verwiesen werden (z.B. den Scrum Guide).[107] Ein pauschaler Verweis auf die Projektmethodik genügt aber nicht, da es auch innerhalb von nach Scrum geführten Projekten viele Gestaltungsmöglichkeiten gibt. Zu den *Rahmenbedingungen* gehören insbesondere folgende Elemente:

[Rz 84] *Prozesse*: Es sind die Projektphasen (Initialisierungsphase, eventuell Checkpointphase, Realisierungsphase), die Art der Zusammenarbeit (Erstellung, Anpassung und Priorisierung der Backlog Items, Meetings, Abnahmen etc.) und das Konfliktlösungsverfahren zu beschreiben.[108]

[Rz 85] *Projektorganisation*: Die Rollen[109] von Personen und Gremien sowie die Auswechslungsmodalitäten für Schlüsselpersonen sind zu definieren.

[Rz 86] *Mitwirkungspflichten*:[110] Dazu gehören insbesondere die Mitwirkung bei der Erstellung der Backlogs, die Priorisierung der einzelnen User Stories, die Abnahme von Product Increments, die Teilnahme an Planungsmeetings und Reviews sowie gegebenenfalls die Analyse und Reorganisation betrieblicher Abläufe, die Bereitstellung von Umsystemen und die Koordination allfälliger Drittleistungserbringer.

[Rz 87] *Zeitrahmen*: Dieser kann z.B. bloss ungefähr skizziert oder aber zeitlich begrenzt werden. Es kann auch ein Release Plan mit Entscheidpunkten für das weitere Vorgehen festgelegt werden.

---

gen des Einsatzes) sowie die Prozesse zur Realisierung beschrieben werden. Siehe dazu auch Fröhlich-Bleuler, Rz. 351; Egli (Fn. 19), Rz. 52ff betrachtet solche Konstruktionen aufgrund des administrativen und rechtlichen Mehraufwandes als nicht praktikabel.

104 Siehe dazu auch Egli (Fn. 19), Rz. 39f; Reichert (Fn. 1), S. 332f.

105 Siehe dazu auch Reichert (Fn. 1), S. 258.

106 Siehe dazu auch Egli (Fn. 19), Rz. 54f; Reichert (Fn. 1), S. 341f.

107 Wenn die Parteien auf nicht von ihnen selbst redigierte Dokumente verweisen, ist sicherzustellen, dass sie deren Inhalte tatsächlich verstanden haben. Bei periodisch aktualisierten Dokumenten (z.B. Normen) ist zu klären, ob der Verweis statisch (Stand im Zeitpunkt des Vertragsschlusses) oder dynamisch (jeweils aktuelle Version) zu verstehen ist. Für dynamische Verweise sind möglichst auch die Folgen einer Aktualisierung zu regeln (z.B. indem die Auswirkungen im Rahmen eines vertraglich definierten Change Management Verfahrens geklärt werden).

108 Siehe zu spezifischen Scope Governance und Scope Eskalationsprozessen in einem Scrum Umfeld auch Opelt/Gloger/Pfarl/Mittermayr (Fn. 23), S. 60f.

109 Siehe zu den einzelnen Rollen Rz. 20ff.

110 Siehe zu den Mitwirkungspflichten auch Rz. 19.

[Rz 88] *Arbeitsergebnisse*: Dazu gehören alle im Rahmen des Projekts zu erstellenden Artefakte.[111] Im Vertrag sollten insbesondere auch die Mindestanforderungen an die Dokumentation festgelegt werden.[112]

[Rz 89] *Rechte an Arbeitsergebnissen*: Eine gemeinsame Entwicklung kann zu einer Miturheberschaft an der Software führen.[113] Dies hat zur Folge, dass die Parteien grundsätzlich nur gemeinschaftlich darüber verfügen können, was in den meisten Fällen nicht beabsichtigt sein dürfte. Die Berechtigung und die Nutzungsrechte an der Software – aber auch an Dokumentationen, Product Vision, Product Backlog etc. – sollten daher vertraglich präzise geregelt werden.

[Rz 90] *Abnahmeverfahren*: Das Akzeptieren eines Product Increments am Ende eines Sprints hat abnahmerechtliche Wirkung. Vertraglich sollte das Verhältnis zu weiteren Einzelabnahmen und zu allfälligen Gesamtabnahmen (z.B. Integrationsabnahmen von Releases) geregelt werden.[114] Soweit nichts anderes vereinbart wurde, ist jeder Abnahmegegenstand an den für ihn relevanten Elementen der Definition of Done[115] zu messen.

[Rz 91] *Gewährleistung*: Die Gewährleistungsfristen beginnen grundsätzlich mit der Lieferung bzw. der Abnahme von Software zu laufen. In der Regel wird im Rahmen von Product Increments jedoch die gesamte funktionsfähige Software – inklusive der vorgängig entwickelten Teile – neu ausgeliefert. Dies spricht dafür, dass die entsprechenden Gewährleistungsfristen für den gesamten Inhalt der Lieferung jeweils neu beginnen. Die Frage sollte aber auf jeden Fall vertraglich geklärt werden. Im Vertrag kann z.B. auch vorgesehen werden, dass die Gewährleistungsfristen erst ab der Abnahme von Releases zu laufen beginnen, welche mehrere Product Increments zusammenfassen.[116] Vertraglich sollten zudem die Gewährleistungsmodalitäten geregelt werden (z.B. Nachbesserung im Rahmen der nächsten Sprints).

[Rz 92] *Risiko- und Chancenverteilung*: Sie sollte an der Analyse folgender Fragen anknüpfen: Welche wesentlichen Chancen und Risiken sind für die Parteien mit dem Vorhaben verbunden? Wer kann die Risiken und Chancen am besten beeinflussen (*«who is the cheapest risk avoider/risk insurer»)*? Wie können diese vertraglich optimiert werden?[117] Die Risikosphären der Parteien können z.B. durch eine Verantwortlichkeitsmatrix mit subsidiären Auffangregeln abgegrenzt werden. In Bezug auf Mehr- oder Minderaufwand der Realisierung kann ein Riskshare definiert werden.[118]

[Rz 93] *Vergütung*: Es kann entweder ein herkömmliches Modell (z.B. Vergütung nach Aufwand, Aufwand mit Kostendach oder Festpreis) oder ein spezifisch auf agile Projekte zugeschnittenes gewählt werden.[119] Im Vertrag sind auch die Zahlungsmodalitäten zu regeln (z.B. Zahlungen nach Abnahme von Sprints oder Releases, Garantierückbehalt, Prämien).

---

111 Siehe zu den einzelne Artefakten Rz. 36ff.

112 Siehe dazu auch Fröhlich-Bleuler (Fn. 19), Rz. 357; Egli (Fn. 19), Rz. 74.

113 Siehe dazu auch Fröhlich-Bleuler (Fn. 19), Rz. 353; Egli (Fn. 19), Rz. 40 und 74.

114 Siehe dazu Fröhlich-Bleuler (Fn. 19), Rz. 354; Egli (Fn. 19), Rz. 74.

115 Siehe zur *Definition of Done* Rz. 33.

116 Siehe dazu Fröhlich-Bleuler (Fn. 19), Rz. 354. Nach Egli (Fn. 19), Rz. 74, sollen Gewährleistungsfristen grundsätzlich erst ab Gesamtabnahme zu laufen beginnen.

117 Im Wesentlichen bestehen folgende Möglichkeiten: Risiken reduzieren, Auswirkungen von Risiken begrenzen, Auswirkungen auf Gegenpartei überwälzen, Chancen erhöhen, Auswirkungen des Eintritts von Chancen verbessern, sich Partizipation daran sichern.

118 Siehe zum *Riskshare* Rz. 52.

119 Siehe zu den Vergütungsmodellen im Einzelnen Rz. 44ff.

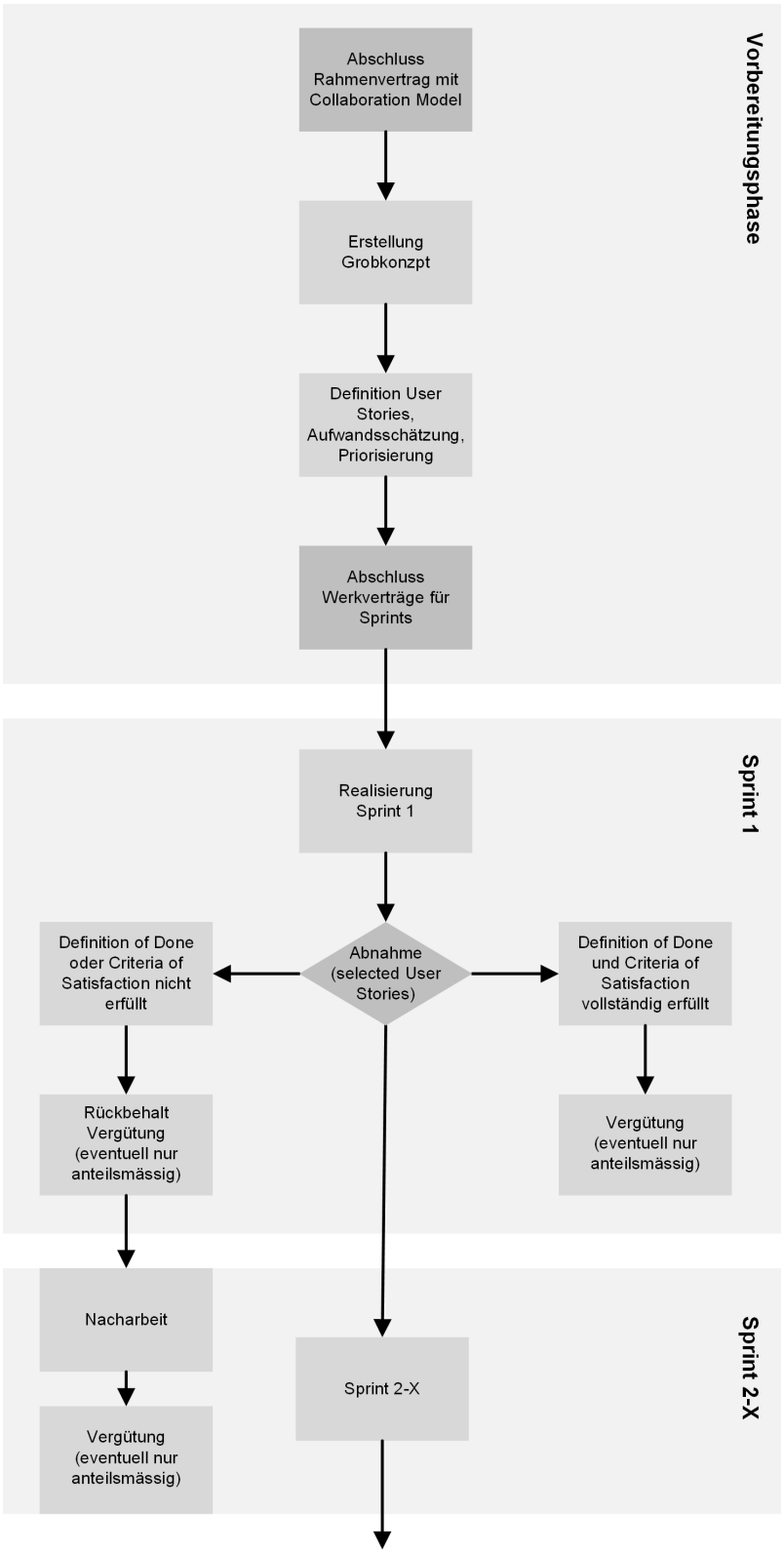
[Rz 94] *Vertragsanpassungsmöglichkeiten*: Änderungen sind in Verträgen über agil geführte Projekte grundsätzlich in gleicher Weise wie in anderen Verträgen möglich. Sie können auch die Projektmethodik betreffen.[120] Währenddem Optimierungen innerhalb des methodischen Rahmens (z.B. Änderung der Sprintdauern) leicht möglich sind, erfordert eine Umstellung von der agilen Entwicklung auf ein Phasenmodell aber eine umfassende Überprüfung der vertraglichen Regeln.

[Rz 95] *Beendigungsmodalitäten*: Es ist einerseits die ordentliche Vertragsbeendigung zu regeln (z.B. wenn alle Product Backlog Items realisiert wurden und der Definition of Done entsprechen oder wenn ein zum voraus definierter Zeitrahmen für die Realisierung endet). Andererseits ist zu bestimmen, unter welchen Voraussetzungen und zu welchen Bedingungen eine vorzeitige Beendigung möglich ist (z.B. jederzeitige Beendigungsmöglichkeit der Leistungsbezügerin[121] gegen Vergütung entsprechend dem vereinbarten Riskshare, freie Auflösung bei Exit Point nach Checkpointphase sowie bei wiederholtem Scheitern von Abnahmen, schwerwiegenden Vertragsverletzungen, Insolvenz etc.).

---

120 Siehe dazu Reichert (Fn. 1), S. 135f.

121 Siehe dazu Rz. 53 sowie auch Fröhlich-Bleuler (Fn. 19), Rz. 359; Egli (Fn. 19), Rz. 74. Bei einer rein auftragsrechtlichen Konzeption ist zudem das jederzeitige Auflösungsrecht gemäss Art. 404 OR zu beachten. Siehe dazu Weber, Basler Kommentar, N. 9ff zu Art. 404 OR, mit weiteren Hinweisen.



Grafische Übersicht 6: Beispiel Abnahme und Gewährleistung in agilem Projekt

X. Vergaberechtliche Aspekte agiler Projekte

[Rz 96] Das geltende Vergaberecht ist nicht auf agile Entwicklungen zugeschnitten. Soweit die WTO Schwellenwerte überschritten werden, müssen öffentliche Ausschreibungen durchgeführt werden. So soll das wirtschaftlich günstigste Angebot für die Deckung eines bestimmten Bedarfs gefunden werden. Die Beurteilung des massgeblichen Preis-Leistungsverhältnisses der Angebote setzt eine präzise Beschreibung der zu erbringenden Leistungen voraus. Lassen sich agile Methoden mit öffentlichen Ausschreibungsverfahren überhaupt in Einklang bringen? Eine vertiefte Auseinandersetzung mit den vergaberechtlichen Aspekten agiler Entwicklungsprojekte würde den Rahmen des vorliegenden Beitrages bei weitem sprengen.[122] Daher können hier nur einige Hinweise im Sinn von «*pistes de réflexion*» gegeben werden. M.E. sind grundsätzlich verschiedene *Vorgehensvarianten* denkbar.

[Rz 97] *Vergaberechtlicher Dialog*[123]: Komplexe Leistungsgegenstände können in einem Dialogverfahren mit ausgewählten Anbietern näher konkretisiert werden.[124] Im Rahmen eines Dialogs kann z.B. ein initialer Product Backlog[125] erstellt und von den Anbietern mit Storypoints[126] bewertet werden. Am Ende der Dialogphase reichen die Anbieter gestützt darauf ihr finales Angebot für die Realisierung ein. Allenfalls könnte auch eine parallele Checkpointphase[127] mit mehreren Anbietern als Dialog ausgestaltet werden. So liessen sich zusätzlich Arbeitsqualität und Velocity[128] der Anbieter bewerten – ein ähnliches Vorgehen wird mitunter für die Erstellung von *Proofs of Concept* gewählt. M.E. ist eine Checkpointphase als Teil eines Dialogverfahrens zwingend zu vergüten. Die Höhe der Vergütung kann aber zum voraus pauschal festgelegt werden.

[Rz 98] *Festpreise* oder *Kostendächer* machen nur Sinn, wenn auch die entsprechenden Leistungsinhalte – zumindest funktional – bereits genügend präzise umschrieben werden, so dass im Rahmen des Vergabeverfahrens das Preis-Leistungsverhältnis der einzelnen Angebote ermittelt wer-

---

122 Siehe dazu auch die Entwürfe des Branchenverbandes SwissICT zu einem Leitfaden für agile Beschaffungen, http://www.swissict.ch/expertenwissen/tools/agile-beschaffung/leitfaden. Eine Auswertung der Simap-Datenbank für den Zeitraum von September 2010 bis November 2015 ergab rund 30 Ausschreibungen, welche die Begriffe «agil» oder «Scrum» enthielten. Nicht alle bezogen sich auf Entwicklungsprojekte – einzelne Ausschreibungen betrafen lediglich ergänzende Leistungen wie Testing und Schulung. Es ist allerdings wahrscheinlich, dass während dieses Zeitraumes mehr Projekte zumindest teilweise agil geführt wurden. Rund 80% der Ausschreibungen erfolgten im offenen, die übrigen im selektiven Verfahren. Eine verwendete zudem das Element des vergaberechtlichen Dialogs. Verschiedene Ausschreibungen bezogen sich auf Rahmenvereinbarungen oder Stundenpakete, andere auf konkrete Projekte. Aufgrund der geringen Datenbasis lassen sich indessen noch keine Trends feststellen.

123 Das Dialogverfahren ist heute nur auf Bundesebene explizit geregelt (Art. 26a VöB). Es handelt sich um eine Form von Verhandlungen. Auf kantonaler Ebene sind Dialoge m.E. aber nur dann ausgeschlossen, wenn das kantonale Recht jegliche Art von Verhandlungen – nicht nur reine Preisverhandlungen – verbietet. Im künftigen Recht soll der Dialog sowohl auf Bundes- wie auch auf Kantonsebene verankert werden. Siehe dazu auch Art. 28 VE BöB bzw. Art. 28 E-IVöB; siehe zum vergaberechtlichen Dialog im Einzelnen Leuthold Alexis, Verhandlungen und der neue «Dialog»: Spielräume bei Bund und den Kantonen, in: Zufferey Jean-Baptiste/Stöckli Hubert (Hrsg.), Aktuelles Vergaberecht 2010, Zürich 2010, S. 277–310; Beyeler Martin, Die revidierte VöB – ein Kurzkommentar, BR/DC 2010, S. 110 ff; Fetz Marco, Dialog im Vergaberecht, BR/DC Sonderheft 2006 S. 59 ff; Galli Peter/Moser André/Lang Elisabeth/Steiner Marc, Praxis des öffentlichen Beschaffungsrechts – eine systematische Darstellung der Rechtsprechung des Bundes, der Kantone und der Europäischen Union, 3. Aufl. Zürich 2013, S. 308-312; Koordinationskonferenz der Bau- und Liegenschaftsorgane der öffentlichen Bauherren (KBOB), Leitfaden öffentliche Beschaffungen mit Dialog, Bern 2014; Ramer Erich, Günstigst statt billigst: Anstösse aus Planersicht, in: Zufferey Jean-Baptiste/Stöckli Hubert (Hrsg.), Aktuelles Vergaberecht 2012, Zürich 2012, S. 299–323.

124 Leider ist das Instrument der *Innovationspartnerschaft* entsprechend Art. 31 der EU RL 2014/24 im VE BöB und im E-IVöB noch nicht enthalten. Dieses könnte zusätzlichen Handlungsspielraum für agile Projekte bieten.

125 Siehe zum *Product Backlog* Rz. 37.

126 Siehe zur Bedeutung von *Story Points* Rz. 32 und 50.

127 Siehe zum Inhalt einer *Checkpointphase* Rz. 51ff.

128 Siehe zum Begriff der *Velocity* Rz. 30.

den kann. Dies ist kaum möglich, wenn bloss eine vage Beschreibung des Project Scope vorliegt. In der Praxis werden agile Ansätze aber oft mit anderen Vorgehensmodellen verbunden (z.B. mit HERMES).[129] Auf diese Weise können vor der Ausschreibung eines agil zu realisierenden Projekts bereits relativ detaillierte Spezifikationen erarbeitet werden. Allerdings lässt sich die Flexibilität, welche agile Methoden anstreben, so nur teilweise nutzen. Wenn bereits ein detaillierter Product Backlog vorliegt, könnte eine Bewertung von Aufwand und Risiken der Realisierung durch Storypoints (verbunden mit einem Stückpreis pro Storypoint) erfolgen.[130] Zusätzlich zum Preis könnte allenfalls auch ein von den Anbietern zu offerierender Riskshare[131] als Zuschlagskriterium berücksichtigt werden.

[Rz 99] *Mehrstufige Realisierung*: Die externe Erarbeitung von Spezifikationen kann als eigenständiger Auftrag vergeben und die Realisierung erst nach deren Vorliegen ausgeschrieben werden. Ein solches Vorgehen wird mitunter auch bei der Verwendung von Phasenmodellen gewählt. Für agile Projekte könnten Initialisierung und Checkpointphase als Grundauftrag zu einem Festpreis vergeben werden. Für eine folgende Realisierung könnte z.B. eine bestimmte Anzahl von Stunden als Option mit ausgeschrieben werden.[132] Dafür können zwar Stundensätze bewertet werden, im Zeitpunkt der Ausschreibung bleibt aber offen, ob das vorgegebene Stundenpaket für die Realisierung tatsächlich ausreichen wird. Zudem lässt sich vor der Durchführung der Checkpointphase die Entwicklungsgeschwindigkeit der Anbieter kaum abschätzen, so dass das Preis-Leistungsverhältnis einer solchen Option nicht genau bewertet werden kann. Diese Argumente sprechen dafür, eine Checkpointphase besser im Rahmen eines vergaberechtlichen Dialogverfahrens durchzuführen.

[Rz 100] Verwendung von *Rahmenvereinbarungen*[133]: Wenn eine zum Voraus nicht genau bestimmbare Anzahl gleichartiger Leistungen (z.B. Sprints oder Releases) bezogen werden soll, könnte dafür allenfalls auch eine Rahmenvereinbarung ausgeschrieben werden. Der Abruf könnte dann auch im Rahmen eines Wettbewerbs unter mehreren Vertragspartnern erfolgen (z.B. durch *Mini-Tender*[134]). Rahmenvereinbarungen sind lediglich eine formales Instrument und schaffen keinen Freiraum von den Grundsätzen des Vergaberechts. Nach der hier vertretenen Auffassung müssen sie in persönlicher, zeitlicher, sachlicher und finanzieller Hinsicht begrenzt sein. Insbesondere müssen der bzw. die Vertragspartner (und die Auswahlkriterien unter mehreren Vertragspartnern), der Leistungsgegenstand, die Maximaldauer, die finanziellen Konditionen und der maximale Kostenrahmen zum voraus festgelegt werden. Die obigen Überlegungen zur Be-

129 Siehe zu hybriden Vorgehensweisen Rz. 9.

130 Siehe zu Stückpreisen für Storypoints Rz. 54.

131 Siehe zur Bedeutung des *Riskshare* Rz. 52.

132 Siehe zur Zulässigkeit von inhaltlich weitgehend unbestimmten Optionen allerdings zurückhaltend BVwGer B-562/2015 (ORMA), Zwischenentscheid vom 21. April 2015, E. 5.8.

133 Rahmenvereinbarungen sind in der Praxis zwar verbreitet, bisher aber weder auf Bundes- noch auf Konkordatsebene geregelt. Siehe dazu auch Art. 29 VE BöB bzw. Art. 29 E-IVöB sowie auch Art. 33 der EU RL 2014/24; Rohner Beatrice/Rizvi Salim, Rahmenvereinbarungen im öffentlichen Beschaffungsrecht, in: SZW 2015, S. 36 ff, S. 39f; Galli/Moser/Lang/Steiner (Fn. 123), Rz. 275ff; Beyeler Martin, Der Geltungsanspruch des Vergaberechts, Probleme und Lösungsansätze im Anwendungsbereich und im Verhältnis zum Vertragsrecht, Zürich 2012, Rz. 2927ff; Scherler Stefan, Die Rahmenvereinbarungen – Les accords-cadres, BR/DC 2004 S. 163f; Straub Wolfgang, Beschaffung komplexer Leistungen zwischen Vertragsfreiheit und Beschaffungsrecht, AJP 2005 S. 1330–1340, S. 1331ff online auch verfügbar unter www.it-recht.ch.

134 Siehe zu *Mini-Tendern* auch Rohner/Rizvi (Fn. 133), S. 39f.

stimmung des Leistungsgegenstandes gelten daher grundsätzlich auch für Rahmenvereinbarungen.[135] Dieser ist gegebenenfalls im Rahmen eines Dialogverfahrens näher zu definieren.

Dr. Wolfgang Straub, LL.M., ist Rechtsanwalt in Bern und Lehrbeauftragter am CAS Programm ICT Beschaffungen der Universität Bern.
Für wertvolle Anregungen danke ich Martin Beyeler, Robert Blaser, Julia Bhend, Gianni Fröhlich-Bleuler, Adrian Hässig, Stephan Kronbichler, Christian Laux, Reto Maduz, Stephan Rothenbühler und Matthias Stürmer.

135 Kann nur das Profil der Mitarbeitenden, nicht aber der Inhalt des von ihnen zu realisierenden Projekts bestimmt werden, kommen allenfalls auch *Personalverleihverträge* in Betracht. In diesem Fall liegt aber die gesamte Führungs- und Realisierungsverantwortung bei der Leistungsbezügerin. Siehe dazu auch die Weisungen des Bundesrates vom 19. August 2015 zum Abschluss von Personalverleihverträgen in der Bundesverwaltung, BBl 2015, S. 6309ff.