Sonderdruck aus OBJEKTspektrum 02/2017

Refinement@Scale

Wann und in welcher Form es sich lohnt – ein Erfahrungsbericht

In Scrum ist es eine Empfehlung, im Skalierungsframework Nexus ein fester Bestandteil: das Refinement. Es dient dazu, vage Ideen herunterzubrechen in Sprint-taugliche User-Storys und Abhängigkeiten zwischen den Teams offenzulegen. Trotz des offenkundigen Nutzens ist das Refinement jedoch nicht immer ein Selbstläufer und auch nicht einfach „yet another meeting“. Der Artikel beschreibt die Erfahrungen aus einem realen Projekt und die Lehren, die sich daraus ziehen lassen.

Der aktuelle Scrum Guide versteht unter Refinement (in früheren Versionen als „Backlog Grooming“ bezeichnet) einen kontinuierlichen Vorgang, kein einmaliges Event. Die Beschreibung gibt an, was beim Refinement passiert: „Als Verfeinerung [Refinement] des Product Backlog wird der Vorgang angesehen, in dem Details zu Einträgen hinzugefügt, Schätzungen erstellt oder die Reihenfolge der Einträge im Product Backlog bestimmt werden“ (vgl. [Suth16]). Offen bleibt die äußere Form, in der das Team dabei vorgeht, und wie weit das Team in die Zukunft schaut. Dabei bezieht sich der Scrum Guide primär auf die Arbeit eines Scrum-Teams an einem Product Backlog.
Mit dem Nexus Guide (vgl. [Sch15]), der Beschreibung des Skalierungsframeworks Nexus, hat der Scrum Guide eine Erweiterung erfahren für den Fall, in dem rund drei bis neun Scrum-Teams am gleichen Product Backlog arbeiten. Scrum zu skalieren, bedeutet für den Product Owner, sich zahlreichen Abhängigkeiten gegenüberzusehen, teilweise zwischen den einzelnen Teams, teilweise auch von externen Einheiten.
Damit reicht es nicht mehr aus, wenn ein Team relativ exakt schätzen kann, wie viele der Anforderungen es im nächsten Sprint umsetzen wird. Denn wenn es dabei beispielsweise auf die Zulieferung eines anderen Teams angewiesen ist, sich jedoch nicht auf einen Liefertermin verlassen kann, dann torpediert diese Unsicherheit die gesamte nachfolgende Planung. Grundsätzlich ist die Sprint-Planung jedoch ein viel zu entscheidender Teil in Scrum, als solche Unwägbarkeiten einfach in Kauf nehmen zu können. Daher ist es kaum überraschend, dass der Nexus Guide das Refinement als wichtigen Bestandteil der Backlog-Pflege und -Planung fest in die Gruppe der erforderlichen Events aufgenommen hat.
Allerdings bleibt auch der Nexus Guide offen hinsichtlich der Form des Refinements. Er gibt an, dass die Anzahl, Frequenz, Dauer und erforderlichen Teilnehmer des Refinements auf den Product Backlog inhärenten Abhängigkeiten basieren.

Refinement – der Nutzen

Lapidar gesagt ist das Refinement das oder die Meetings, in dem aus einem „Das brauche ich“ des Product Owner ein „mit dem Wissen, das wir jetzt haben, können wir an die Umsetzung gehen“ des Entwickler-Teams wird. Das impliziert ein gemeinsames und grundlegendes Verständnis der Anforderungen auf beiden Seiten. Bei der Skalierung kommt ein weiterer wichtiger Aspekt hinzu: die Entlastung des Product Owner. Wenn der Product Owner für bis zu neun Teams jede Anforderung bis ins letzte Detail beschreiben muss, dann bleibt ihm oder ihr kaum noch freie Kapazität für andere wichtige Aufgaben wie die Kommunikation mit den Stakeholdern.
So klar der Nutzen des Refinements sein mag, so sehr hängt seine Ausgestaltung von der individuellen Projektsituation ab. Das erlaubt es kaum, nach einer simplen Vorgabe hinsichtlich der Häufigkeit und der Form der Meetings zu arbeiten. Ob das Refinement zielführend ist, hängt jedoch nicht zuletzt davon ab, wie oft Refinement-Meetings stattfinden, wann, mit welchem Teilnehmerkreis und wie darin vorgegangen wird. Das lässt sich deutlich an einem Projektbeispiel illustrieren.

Die Ausgangslage in einem konkreten Beispiel

In unserem Beispielprojekt arbeiten mehrere Teams an einer E-Commerce-Anwendung mit einem gemeinsamen Product Backlog. Die einzelnen Teams sind nicht räumlich getrennt, insgesamt verteilen sich die Teams jedoch auf diverse unterschiedliche Standorte. Einem Chief Product Owner stehen mehrere Product Owners zur Seite, einzelne fachliche Aufgaben werden von eigenen Produkt- beziehungsweise Projektmanagern betreut. Die Entwicklerteams arbeiten nach den Scrum-Prinzipien und haben sich für eine Sprint-Dauer von zwei Wochen entschieden.
Als wir zu dem Projekt hinzukamen, fanden die Refinements jeweils zum Sprint-Wechsel statt, also am gleichen Tag wie das Sprint Planning. Mike Cohn (vgl. [Coh15]) empfiehlt, einige Tage Abstand zwischen beiden Meetings zu lassen. Sinn dieser Empfehlung ist es, dem Product Owner Raum zu geben, im Refinement als offen identifizierte Fragen bis zum eigentlichen Planning zu beantworten. Erfahrungsgemäß bedarf es etwas Zeit, die erforderlichen Informationen zu beschaffen. Aus logistischen Gründen ist es jedoch nicht immer möglich, dieser Empfehlung zu folgen.
Wie sich schnell zeigte, gab es einige Punkte, die den Erfolg der Refinements deutlich schmälerten. So nahmen zwar Entwickler aus den verschiedenen Teams teil, aber häufig nicht diejenigen, die anschließend mit der Umsetzung betraut waren. Der Fokus lag mehr auf der reinen Schätzung als auf der detaillierten Planung der nahen Zukunft, etwa den folgenden

drei Sprints. Bei den Storys wurde häufig nicht danach unterschieden, ob sie in den nächsten Sprint wandern sollten oder ob ihre Umsetzung möglicherweise erst Monate später anstand. Die nachfolgenden Sprints blieben unklar. Die Planung war schon allein deshalb nicht zuverlässig möglich, weil sich die im Refinement anwesenden Entwickler zwar untereinander auf eine Schätzung der durchschnittlichen Team-Geschwindigkeit einigten. Doch dieser Durchschnittswert passte häufig nicht zu den realen Werten der umsetzenden Teams.
Zudem lag der Fokus nicht primär auf der inhaltlichen Prüfung der Anforderungen, daher blieben Abhängigkeiten oft verborgen. Sie wurden erst während der Umsetzung entdeckt. Im Ergebnis wurde die angestrebte Planungssicherheit genauso wenig erreicht wie das gemeinsame Verständnis.
Wir begannen daher damit, das Refinement in einem „Inspect & Adapt"-Prozess zu ändern.

Schritt 1: Das Cross Team und das Local Refinement

Am Anfang etablierten wir zwei Typen von Refinement-Meetings (**siehe Abbildung 1**):

- das *Cross Team Refinement* als – modifiziertem – Nachfolger des bisherigen Refinements und
- die *Local Refinements* als Basisplanung der einzelnen Teams.

Dabei sollten die Ergebnisse der lokalen Refinements spätestens eine Woche vor Beginn eines neuen Sprints an den Chief Product Owner gehen beziehungsweise an den Stellvertreter, wenn die Teilnahme am Cross Team Refinement an einen der unterstützenden Product Owners delegiert wurde. Diese Vorgabe spiegelt die Cohn-Empfehlung wider. Die Verbindung zwischen beiden Refinement-Typen formiert einen Kreislauf, weil die Ergebnisse wechselseitig die Grundlage der weiteren Verfeinerung bilden.
Für das Cross Team Refinement setzen wir zwei Stunden an, wichtig ist der direkte Kontakt der Teilnehmenden, sprich, deren persönliche Anwesenheit. Damit der logistische Aufwand vertretbar bleibt, entsendet jedes Entwicklerteam einen Repräsentanten. Es bietet sich an, das Cross Team Refinement auf einen Termin zu setzen, an dem die Team-Repräsentanten ohnehin vor Ort sind. Das Cross Team Refinement dient vor allem zwei Zielen, zunächst einer groben Schätzung des möglichen Scopes für die nächsten drei Sprints. Außerdem sollen darin Abhängigkeiten identifiziert und möglichst aufgelöst werden. Dieses Meeting ist auch der richtige Ort, um über Sprint-Ziele zu diskutieren. Das gilt sowohl für das Sprint-Ziel jedes einzelnen Teams als auch für das übergeordnete Ziel des Sprints, das die Einzelergebnisse integriert.
Damit alleine bliebe die Gefahr, zu wenig auf das Detailverständnis eingegangen zu sein und in der Folge eine zu grobe und wenig zuverlässige Schätzung zu erhalten. Deshalb ergänzen die lokalen Refinements die Planung. Sie finden während eines Sprints statt und bilden den Zeitpunkt, zu dem Features zerlegt werden, in einzelne, Sprint-taugliche User-Storys.
Was die Teilnehmenden anbelangt, gibt es einen signifikanten Unterschied zwischen dem Cross Team und dem lokalen Refinement. Bei Ersterem reicht der eine Repräsentant des jeweiligen Entwicklerteams vollkommen aus. Dagegen ist es sehr wichtig, dass alle Mitglieder des Entwicklungsteams beim lokalen Refinement präsent sind, damit alle über die Ergebnisse die gleiche Kenntnis haben – ohne Umwege über die stille Post. Nach unserer Erfahrung lohnt sich der Mehraufwand, weil dann während der Umsetzung viel seltener ad hoc Klärungen nötig sind, die sonst regelmäßig zu Verzögerungen führen. Dringend empfohlen ist die Teilnahme für Mitglieder des Fachbereichs beziehungsweise Stakeholder. Wenn es, wie in unserem Fall, mehrere Product Owners gibt, sollte immer derjenige mit der fachlichen Verantwortung teilnehmen.
Für die lokalen Refinements halten wir uns an den Vorschlag aus dem Scrum Guide, maximal 10 Prozent der Teamkapazität dafür zu beanspruchen. Jedes Team teilt sich diese 10 Prozent selbst ein, typischerweise bestehen sie jedoch aus ein bis zwei Team Sessions, dazu addiert sich der Austausch zwischen einzelnen Teammitgliedern und dem Fachbereich zu Fragen, die während der Sessions auftauchen. Der Prozess endet, wenn die Ergebnisse aus den lokalen Refinements im nächsten Cross Team Refinement besprochen werden. Dabei arbeiten die Teilnehmenden Änderungen und Unterschiede in die Planung ein und verfeinern die einzelnen Sprints.

Anpassungsbedarf nach Schritt 1

Beim „Inspect & Adapt" des beschriebenen Refinement-Prozesses zeigte sich an unterschiedlichen Stellen Anpassungs-

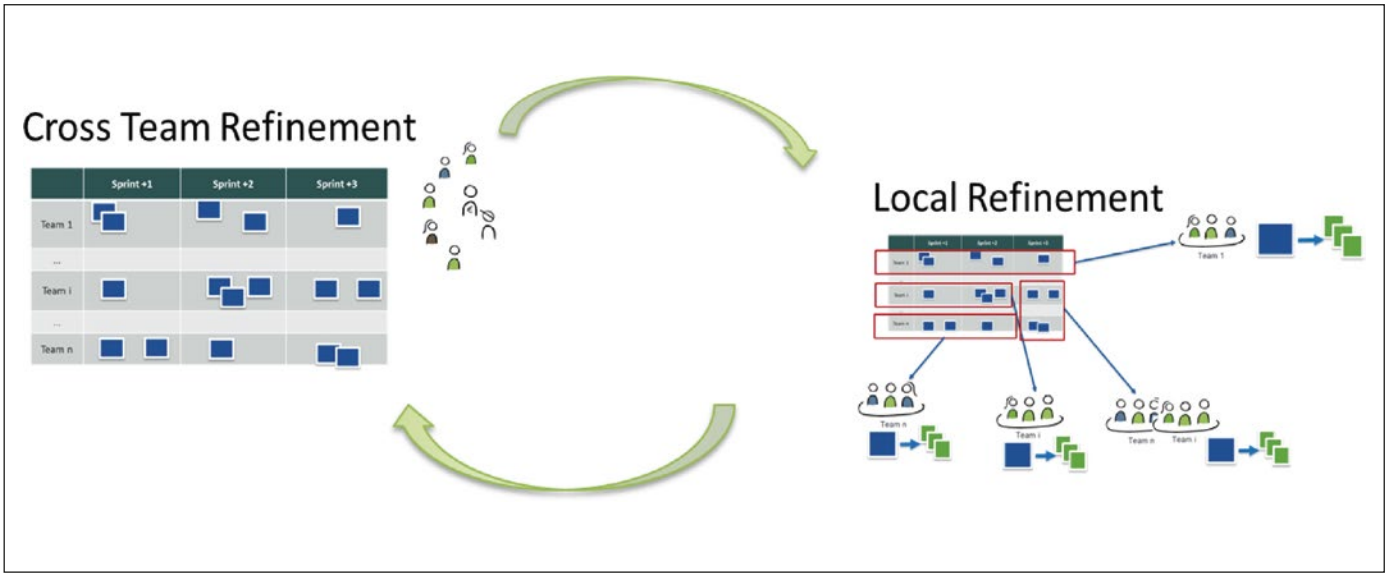


Abb. 1: Refinement Cycle V1: Die übergreifenden Refinements werden ergänzt durch lokale Refinements jedes einzelnen Teams

bedarf. So führte die anfängliche Forderung zur Verwirrung, nur der nächste Sprint müsse komplett „ready“ sein und die nachfolgenden lediglich zu einem bestimmten Prozentsatz.
Um Diskussionen darüber zu vermeiden, wann ein Sprint „zu 70 Prozent fertig“ ist, einigten wir uns auf die Vorgabe, immer die folgenden drei Sprints „ready“ zu haben, soweit das der Kenntnisstand ermöglicht. Wir halten die Festlegung auf die drei nächsten Sprints für einen tragfähigen Kompromiss zwischen Planungsstabilität und Flexibilität.
Als nicht zufriedenstellend erwies sich dagegen die Anfangspraxis, jedem Team selbst den Zeitpunkt zu überlassen, wann es die Rückmeldungen aus den lokalen Meetings an den Product Owner gibt. Auch zwei mögliche Termine mit dem Product Owner führten zu Verschleppungen, oftmals wurden Abhängigkeiten erst im Cross Team Refinement thematisiert, wo es dann nicht mehr gelang, sie aufzulösen.

Der neue Refinement-Cycle

Wir entschieden deshalb, die zwei frei auswählbaren Termine mit dem Product Owner zu ersetzen. An ihrer Stelle etablierten wir ein weiteres, festes Refinement, das „Online Cross Team Refinement“ (**siehe Abbildung 2**).
Dieses Meeting dauert jeweils 45 Minuten und findet zwei Mal pro Woche via Telefon statt. Jedes Team nominiert dazu Repräsentanten. Während die einzelnen Teams ihren Scope für die nächsten Sprints vorstellen, kristallisiert sich klar heraus, wo Abhängigkeiten entstehen. Das erlaubt, sie noch vor dem nächsten Cross Team Refinement – und ebenfalls vor der nächsten Planung – zu berücksichtigen und mögliche Lösungen zu erarbeiten. Sinn und Zweck dieser festen Termine ist es, die Informationen schnell und in konsolidierter Form weiterzugeben.

Was wir erreicht haben

Der neue Refinement-Prozess hat deutliche Vorteile gebracht: Mittlerweile sind regelmäßig zwei Sprints im Voraus „ready“ und der dritte zumindest vorbereitet.
Schon alleine die zeitliche Beschränkung des Online Cross Team Refinement entlastet den Product Owner, nachdem dieser Austausch mit den Teams zuvor häufig mehrere Stunden in Anspruch genommen hatte – ohne ein Mehr an Informationen zu bringen.
Die Entwicklerteams sind viel stärker in die Planung involviert, der verbesserte Austausch mit den Fachbereichen hat bereits dazu geführt, die Zahl der gescheiterten Storys pro Sprint zu reduzieren. Denn je besser das Entwicklerteam den fachlichen Hintergrund einer Anforderung versteht, umso geringer wird die Gefahr, an der Anforderung vorbei zu entwickeln.

Offene Herausforderungen

Dennoch bleiben nach wie vor einige Herausforderungen ungelöst. Beispielsweise bedürfen die lokalen Refinements einer langfristigen Begleitung. Sonst besteht die Gefahr, dass sie das jeweilige Entwicklerteam ohne Beteiligung des Fachbereichs abhält. Dementsprechend fraglich ist dann, ob das fachliche Verständnis der Anforderungen tiefgreifend genug ist, um eine valide Planung zu erlauben. Tatsächlich tendierten einige Teams zu solchen „isolierten“ Meetings.
Eine weitere Schwierigkeit besteht in der Tatsache, dass es zwar, wie beschrieben, gelungen war, die Zahl der gescheiterten Storys zu reduzieren. Dennoch blieben regelmäßig Storys übrig, die am Ende des Sprints nicht fertig waren und dann im nächsten Sprint als Verzögerung wirkten. Einen ähnlichen Effekt erzeugen kurzfristige Änderungswünsche von der Fachseite aus.
Nicht wirklich zufriedenstellend sind auch bislang die Werkzeuge, die für das Cross Team Refinement zur Verfügung stehen. Die üblichen Ticketing-Systeme eignen sich sehr gut dafür, die User-Storys eines Teams darzustellen oder die gesamten, integrierten Storys. Für eine digitale Übersicht der Tasks aller Teams in den nächsten Sprints sind sie weniger geeignet. Vor allem lassen sie sich kaum einsetzen, um Abhängigkeiten zu visualisieren.
Beim Cross Team Refinement selbst bietet

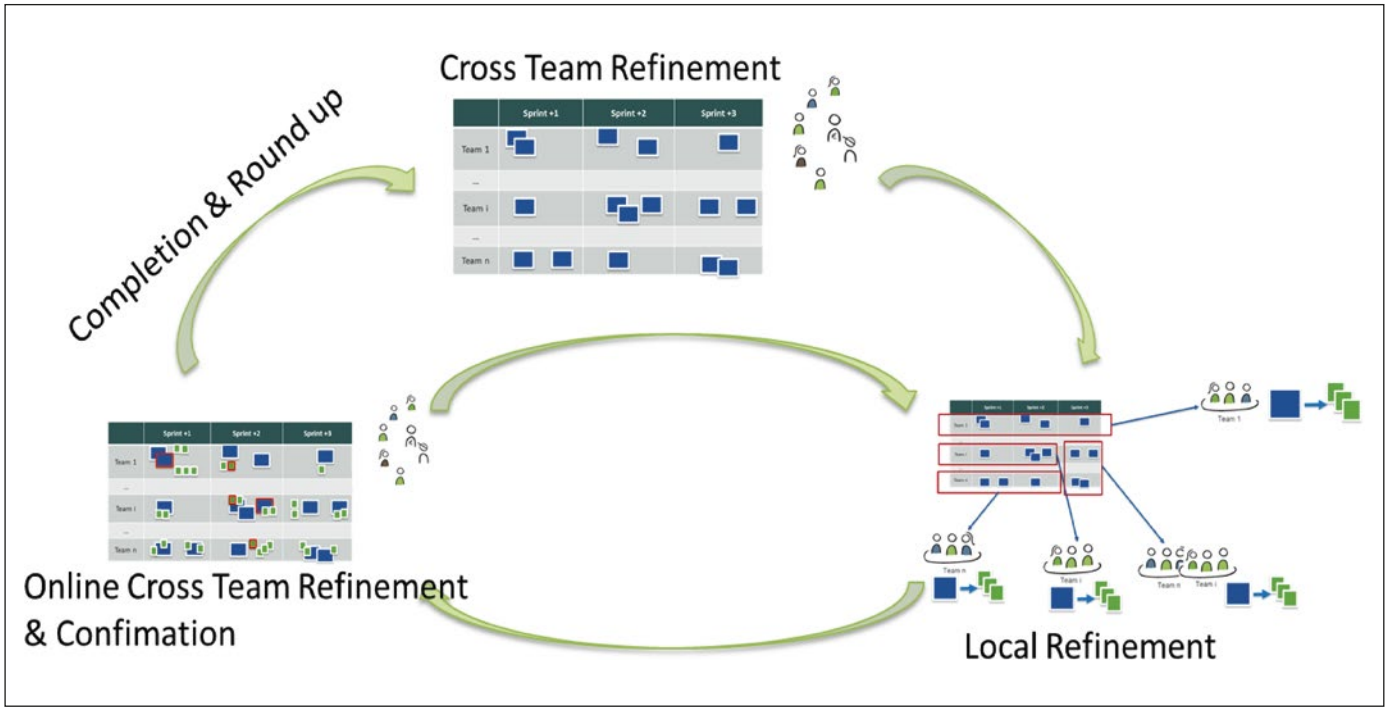


Abb. 2: Neuer Refinement-Cycle mit Cross Team Refinement: Das online abgehaltene Cross Team Refinement als Konsolidierungsschritt für die Ergebnisse der lokalen Refinements

es sich an, mit Karten und einem physischen Nexus-Board zu arbeiten. Es versteht sich jedoch, dass diese analoge Darstellung für ein Meeting am Telefon nicht passt. Wir suchen deshalb derzeit nach Möglichkeiten, die digitale Illustration zu verbessern.

Fazit

Ein typisches Ziel der Skalierung ist es, die Time-to-Market zu reduzieren. Im Zuge der Skalierung treten jedoch schnell Abhängigkeiten auf. Der beschriebene Prozess bietet eine solide Herangehensweise, um diese Abhängigkeiten zu managen. Eine – auch bei uns diskutierte – Alternative besteht darin, dass jedes Team für genau einen Bereich der Anwendung zuständig ist und die Storys entsprechend geschnitten werden. Damit vermeidet man die Abhängigkeiten zwischen den Teams und der oben beschriebene, aufwendige Refinement-Prozess wird überflüssig. Tatsächlich?

Im Zuge der Diskussion kam die Frage auf, wer in diesem Fall die Storys schneidet. Denn es ist nicht so, dass es innerhalb des beschriebenen Vorgehens keine Abhängigkeiten gibt, was sich spätestens beim Schneiden der Storys zeigt. Die Abhängigkeiten entstehen trotzdem, ihre Auflösung verlagert sich lediglich eine Ebene nach oben – also hin zum Product Owner. Dessen Entlastung war jedoch eines unserer Kernziele, als wir den Prozess eingeführt haben, und dieses Ziel würde so ad absurdum geführt. Außerdem ist fraglich, ob es überhaupt möglich ist, wirklich jede Story im Vorfeld so gut zu schneiden. Erfahrungen aus dem Wasserfall-Modell belegen das Gegenteil.

Letztendlich wurde auch die Kritik laut, ein solches Vorgehen erschwere das wertgetriebene Arbeiten. Wertgetrieben bedeutet für die Teams, sich denjenigen Themen zu widmen, die am meisten Wert bringen. Und das sind eben nicht zwangsläufig Features für genau die Komponente, die innerhalb des vertikalen Schnitts in der Verantwortung des Teams liegt.

Gehen wir also davon aus, dass wertgetriebenes Arbeiten und eine kurze Time-to-Market die Ziele der Skalierung sind. Dann wird, zumindest gelegentlich, mehr als ein Team ein und dasselbe Feature umsetzen. In diesem Fall halten wir den beschriebenen Refinement-Prozess für empfehlenswert. Denn dann sind unerkannte Abhängigkeiten ein massives Risiko für die Planung und ein solides Refinement eine gute Abhilfe. ||

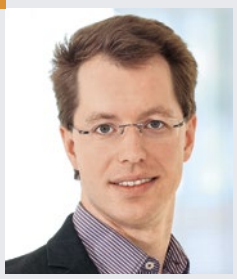
Literatur & Links

[Coh15] M. Cohn, Product Backlog Refinement (Grooming), Blog-Beitrag, 2015, siehe:
https://www.mountaingoatsoftware.com/blog/product-backlog-refinement-grooming

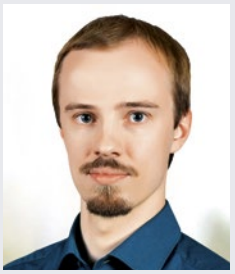
[Sch15] K. Schwaber, Nexus™ Guide, scrum.org, 2015, siehe:
https://www.scrum.org/Resources/Nexus-Guide

[Suth16] J. Sutherland, K. Schwaber, Der Scrum Guide, scrum.org, 2016, siehe:
http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-German.pdf

Die Autoren



Jan Baumann
(jan.baumann@andrena.de)
studierte Informatik und ist für die andrena objects ag aktiv als Softwareentwickler, Trainer für agile Softwaretechniken (Stichwort ASE) und Coach. Er ist Verfechter eines „T"-förmigen Skillsets und besonders interessiert an den Scrum-Rollen sowie kollegialem Coaching und Architektur.



Hartmut Senska
(hartmut.senska@andrena.de)
verfügt über mehrjährige Erfahrung im agilen Umfeld, seit 2014 arbeitet er als Agile Coach und Scrum Master für die andrena objects ag. Im Moment begleitet er einen Kunden dabei, Projekte agil zu skalieren und deren Erfolg durch kontinuierliche Professionalisierung sicherzustellen.