Scrum

Der Pradigmenwechsel im Projekt- und Produktmanagement – Eine Einführung

Boris Gloger

Einführung

Scrum[1] ist heute der De-facto-Standard in der *agilen*[2] Softwareentwicklung. Es hat sich in den letzten Jahren aus einer (agilen) Projektmanagementmethode zu einem neuen Verständnis darüber entwickelt, wie man dysfunktional arbeitende Teams, Abteilungen, ganze Organisationseinheiten und Firmen agil und lean managt. Meist wird Scrum von Firmen zunächst auf Team- oder Projektebene als Projektmanagementmethode eingesetzt. Dabei bleibt es für einige Firmen; andere gestalten im Laufe der Zeit ihre gesamte Organisation mit Scrum.

Kurze Geschichte von Scrum

In den USA begannen in den 1990er-Jahren Alistair Cockburn, James Coplin, Jim Highsmith, Ken Schwaber, Kent Beck u. a. die agile Softwareentwicklung zu kreieren.[3] Sie erkannten, dass kleine *Entwicklungsteams* mit Teammitgliedern, die im Wesentlichen alle Skills haben, effektiver und schneller bei höherer Qualität Softwareapplikationen liefern als große Teams [2, 4, 13]. Zunächst verbreitete sich *eXtreme Programming* (XP) [1], dann startete Ken Schwaber mit dem Certified ScrumMaster Programm ab 2002 eine weltweite Bewegung. Er bot die Möglichkeit, zu lernen, wie agile Softwareentwicklung funktioniert. Da Scrum im Gegensatz zu XP das Management und das Business anspricht, setzte es sich als De-facto-Standard der agilen Softwareentwicklung durch.[4]

Andere agile Softwareentwicklungsmethoden

Scrum, im Klima der Bewegung zur agilen Softwareentwicklung entstanden, war zu Anfang nur eine unter vielen agilen Softwareentwicklungsmethoden. Zu diesen Methoden gehörten: das Extreme Programming, Kent Beck; Crystal, Alistair Cockburn; Feature Driven Development, Jeff de Luca und einige andere unbedeutendere Ansätze.

Scrum hat sich dabei immer als reine Management-*Rahmenmethode* verstanden.[5] Daher sagt Scrum nichts darüber aus, wie Software entwickelt werden soll. Methoden wie XP oder FDD haben im Gegensatz dazu klare Vorgaben zur Entwicklung gemacht. Daher kann Scrum mit jeder gängigen Softwareentwicklungsmethode, also auch mit anderen agilen Methoden kombiniert werden.

[1] Der Name Scrum kommt aus dem Artikel *The New New Product Development Game*, in dem Nonaka und Takeushi das Neuentwickeln von Projekten mit Rugby vergleichen und für das Zusammenarbeiten von cross-funktionalen Teams das Bild des *Scrum*, eine Freistoßsituation im Rugby, verwenden [11].

[2] *Agile Softwareentwicklung* ist der Oberbegriff für den Einsatz von Agilität in der Softwareentwicklung. [...] Agile Softwareentwicklung versucht mit geringem bürokratischen Aufwand und wenigen Regeln auszukommen. Siehe dazu auch: http://de.wikipedia.org/wiki/Agile_Softwareentwicklung.

[3] Siehe dazu das Agile Manifesto (www.manifesto.org).

[4] Weltweit gibt es mehr als 60.000 Certified ScrumMaster, mehr als 100.000 Menschen sind in Scrum ausgebildet, mehr als 1000 Firmen nutzen Scrum.

[5] Ken Schwaber stellt es selbst immer als Management Framwork dar [14].

DOI 10.1007/s00287-010-0426-6

Boris Gloger
Schwarzwaldstrasse 139, 76532 Baden-Baden
E-Mail: boris.gloger@borisgloger.com

*Vorschläge an Prof. Dr. Frank Puppe <puppe@informatik.uni-wuerzburg.de> oder Prof. Dr. Dieter Steinbauer <dieter.steinbauer@schufa.de>

Alle „Aktuellen Schlagwörter" seit 1988 finden Sie unter: www.ai-wuerzburg.de/as

Zusammenfassung
Scrum ist der De-facto-Standard in der agilen Softwareentwicklung. Mit Scrum werden Prinzipien aus dem Knowledgemanagement und dem Toyota Production System in das Managen von Softwareentwicklung eingeführt. Der Artikel stellt den Scrum-Flow und die Rollen kurz vor.

Ein Paradigmenwechsel

Scrum entwickelte aus den Ideen des Knowledgemanagements[6] die gleichen Vorgehensprinzipien wie das Toyota Production System (TPS)[7]:

- kleine *Self Directed Work Teams* oder *cross-funcional* Teams, in denen die Teammitglieder alle verschiedenen Aufgaben durchführen können,
- der PDCA-Cycle (siehe Abschn. „Der Deming Cycle und Scrum“) als kontinuierlicher Verbesserungsprozess,
- der *one-piece-flow* (nur ein *Teil* ist in Arbeit),
- das ständige Ausmerzen von *Waste*, in Scrum *Impediments* und
- das Pull-Prinzip statt des Push-Prinzips.

Mit *dem kontinuierlichen Liefern* von fertiger, nutzbarer Software am Ende einer Entwicklungs-Iteration, dem *Sprint*, bricht Scrum gänzlich mit traditionellen Projektmanagementansätzen.

Scrum erhöht die Produktivität

Von den Anwendern von Scrum wird berichtet, dass die Anwendung dieser Prinzipien zu erheblichen Produktivitätssteigerungen bei der Softwareentwicklung führt.[8] Sie berichten von höherer Zufriedenheit der Mitarbeiter, höherer Codequalität, wesentlich verbesserter Transparenz zum Stand der Produktentwicklung und vielem mehr. Exemplarisch kann man das bei der Firma Salesforce.com, San Francisco, sehen. Steve Green berichtet von 568% Verbesserung der Produktivität (Anzahl von entwickelter Funktionalität) und z. B. einer 38%igen Steigerung der Lieferungen von Funktionalität pro Entwickler innerhalb eines Jahres.[9] Gründe für diese Produktivitätssteigerungen sind:

- Cross-funktionale Teams arbeiten gemeinsam an dem Produkt, ihre Kenntnisse werden sofort miteinander ausgetauscht. Abstimmungen finden sofort statt.
- Eine klare Fokussierung der Teammitglieder auf *eine* Aufgabe, *ein Projekt*.
- Scrum-Teams brauchen keine aufwändige Verwaltung und Kontrolle, d. h. der sonst vielfach übliche Overhead kann weitestgehend entfallen.
- Es gibt klare Verantwortlichkeiten. Konflikte und Probleme werden so früh wir möglich erkannt, besprochen und entschieden.
- Wenn sich etwas im Hinblick auf das Ergebnis nicht bewährt, dann fällt das sehr schnell auf, d. h. Fehler können schnell und kostengünstig beseitigt werden.

Das sind nur einige Gründe. Allgemein kann man sagen, dass Scrum dysfunktionale Strukturen aufdeckt und dass der ScrumMaster dafür sorgt, dass diese sofort beseitigt werden.

Die Prinzipien angewendet

Kleine, selbstorganisierte Teams

Ein *Scrum-Team* besteht im Idealfall aus sieben Personen, dem *ScrumMaster*, dem *Product Owner*, und den Personen des *Entwicklungsteams*. Diese sind hocheffektiv, weil sie nach dem klassischen Modell der *self-directed work teams* [7], den autonomen Arbeitsgruppen, gebildet sind: Alle Mitglieder des *Entwicklungsteams* sind in der Lage, mehrere Arbeiten im Arbeitsprozess durchzuführen. Sie organisieren ihre Aufgaben vollständig selbst. Der ScrumMaster ist *nicht* Teil des Entwicklungsteams. Er organisiert die Rahmenbedingungen um das Scrum-Team herum. Der Product Owner steuert das Team aus der fachlichen Sicht. Er entscheidet, was wann vom Team umgesetzt werden soll, macht aber keine Vorgaben, wie das Produkt erstellt werden soll.

[6] [11] enthält bereits die Ideen zu cross-funktionalen, autonomen Teams und zu einem Entwicklungsverfahren, in dem alle gemeinsam gleichzeitig am Produkt arbeiten.

[7] Siehe zum TPS [10].

[8] Dazu gibt es immer noch wenige verlässliche empirische Daten. Was es aber gibt, ist die Aussage der Menschen, die erfolgreich Scrum eingeführt und gelebt haben: Sie alle erklären, dass Scrum der richtige Weg zu Produkivitätssteigerung und mehr Transparenz in der Entwicklung von Produkten ist.

[9] Aus meiner eignen beruflichen Praxis kann ich diese Aussagen bestätigen. Die Präsentation in [9] liefert hier gutes Datenmaterial.

Die Teile und Verantwortlichkeiten des Scrum-Teams. Es gibt drei Teile, die das *Scrum-Team* bilden: *ScrumMaster*, *Product Owner* und das *Team*. Die Umwelt des Scrum-Teams sind die drei Teile: *Manager*, *Customer* und *User*.

Das Scrum-Team. Der *ScrumMaster* als *Rolle* managt die Grenzen des Scrum-Teams.[10] Er *beschützt* es vor äußeren Einflüssen. Er sorgt dafür, dass der Scrum-Prozess von allen eingehalten wird, implementiert Scrum und arbeitet mit dem Management an produktivitätssteigernden Verbesserungen.

Der ScrumMaster als *Person* ist eine Führungskraft ohne disziplinarische Verantwortung. Er sorgt für die Selbstorganisation des Teams, in dem es die formale Autorität, alle notwendigen Ressourcen und alle notwendigen Informationen bekommt. Er sorgt dafür, dass sich das Team für das zu Liefernde *verantwortlich* fühlt. [7, S. 16]

Der *Product Owner* als *Rolle* erstellt die *Product Vision*[11] und das priorisierte und geschätzte Product Backlog, die Liste der Funktionalitäten, die zu erarbeiten sind. Er stellt die Profitabilität der Produktentwicklung sicher, indem er streng auf den Return-on-Investement[12] achtet.

Als *Person* ist er Visionär. Er führt die Produktentwicklung strategisch und beantwortet dem *Entwicklungsteam* fachliche Fragen.

Das *Team* als *Rolle* ist verantwortlich für die Lieferung des Produkts, die technische Umsetzung, die Qualität des Gelieferten und die Einschätzung, was es tatsächlich liefern kann.

Als *Personen*, als Teammitglieder, organisieren sie sich so, dass sie alles Versprochene liefern können. Dazu führen sie alle Arbeiten gemeinsam aus und erhöhen ständig ihre Produktivität.

Die Scrum-Teamumgebung: drei Nebenrollen. Die Umgebung des *Scrum-Teams* kennt drei weitere Rollen: Der *Manager* als *Rolle* setzt die organisatorischen Rahmenbedingungen.[13] Als *Person* kann das der Head of Development sein, der Regeln darüber aufstellt, wie zu entwickeln ist.

Der *Customer* als *Rolle* bezeichnet den Auftraggeber.

Der *User* als *Rolle* gibt das Feedback zur erstellten Funktionalität. Als *Person* nutzt er das Produkt.

Der Deming Cycle und Scrum

Im Kern von Scrum, als Mindset, um professionell Produkte zu entwickeln, steht die kontinuierliche Verbesserung. Daher ist der *Scrum-Flow* als Implementierung einer Folge von Meetings zu sehen, um den von Dr. W. Edwards Deming eingeführten Plan-Do-Check-Act-Zyklus (PDCA)[14] zu implementieren (siehe Abb. 1).

Der *Sprint* wird vom *strategischen Planungsprozess* umrahmt. Der Product Owner konkretisiert kontinuierlich die *Product Vision*, aktualisiert und re-priorisiert das *Product Backlog* (die Liste der Funktionalitäten, die zu erarbeiten sind) und arbeitet *mit* dem *Entwicklungsteam* an der *Sprint-übergreifenden* Release-Planung. Dazu dienen ihm einige Meetings: das *Estimation Meeting* und das *Business Value Estimation Meeting*.

Der One-Piece-Flow

Das *Entwicklungsteam* erarbeitet während des Sprints die zu liefernde Funktionalität konsequent in der priorisierten Reihenfolge. Im Idealfall arbeiten alle Teammitglieder dabei immer an genau einer Funktionalität *(One-Piece-Flow)*. Das Ziel ist es, eine Funktionalität nach der anderen zu liefern.

Maintenance-Aufgaben, z. B. Fehlerbehebung während des Sprints, werden sofort erledigt. Ungeplante, neue Funktionalitäten werden ins Product Backlog gegeben und im nächsten Sprint erledigt.

Impediment-Bekämpfung

Probleme, *Impediments*, im Toyota Production System *Waste*, die beim Erstellen einer Funktionalität auftreten, werden vom ScrumMaster möglichst sofort behoben.

Das Pull-Prinzip

Das Toyota Production System führt konsequent das Pull-Prinzip ein. Dieses Prinzip führte zum vollständigen Paradigmenwechsel in der Automo-

[10] Zur Idee des Managers als *boundary manager* siehe [7].

[11] In Scrum bezeichnet man mit *Product Vision* die zugrunde liegende Produktidee.

[12] Der Begriff Return-on-Investment (deutsch Kapitalverzinsung oder Kapitalrendite, kurz ROI) bezeichnet ein Modell zur Messung der Rendite des eingesetzten Kapitals. Siehe dazu: http://de.wikipedia.org/wiki/Return_on_Investment.

[13] Zur Rolle des Managers in einer Organisation wie Pixar siehe [5].

[14] http://en.wikipedia.org/wiki/PDCA.

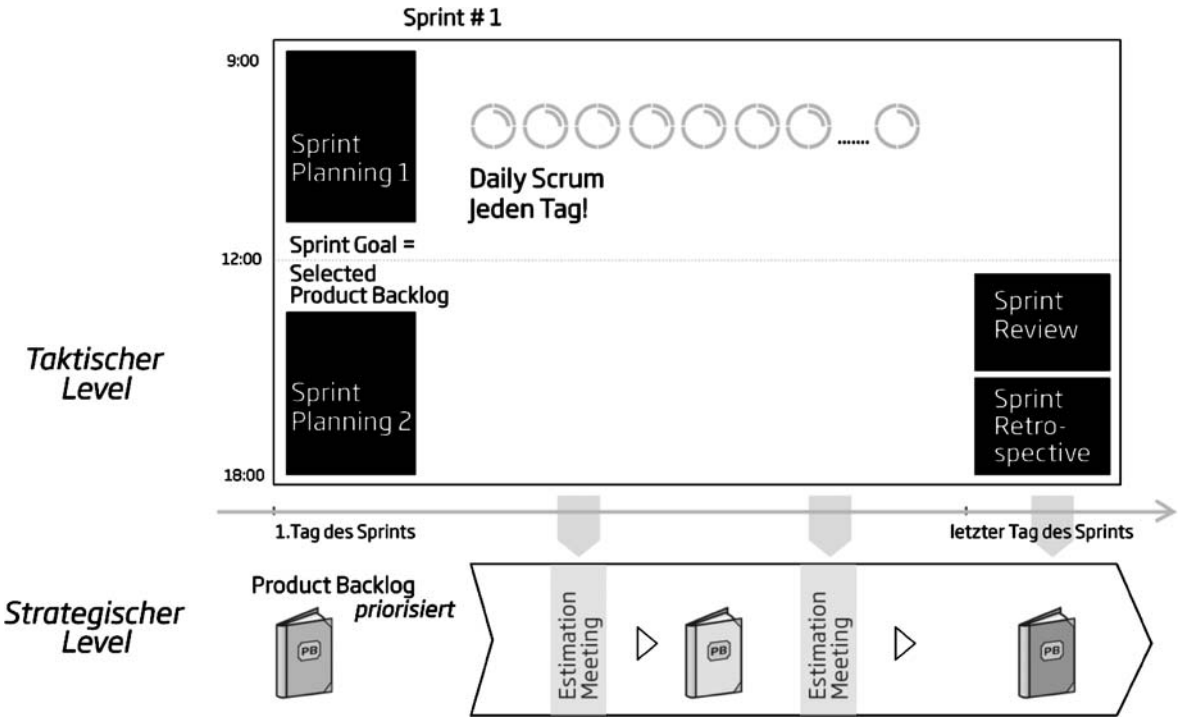

Abb. 1 Der Scrum Flow – strategische und taktische Ebene

bilindustrie und Logistikindustrie. Nicht mehr die Produktionskapazität steuert den Ausstoß der Produktion, sondern einzig der tatsächliche Bedarf an einem Produktteil.

Dieses Prinzip wird von Scrum beim Managen von Projekten *gelebt*:

1. Das Produkt Backlog wird vom Product Owner basierend auf den Markterfordernissen priorisiert. Technische Machbarkeit spielt bei der Priorisierung der Funktionalitäten eine untergeordnete Rolle.
2. Das *Entwicklungsteam* entscheidet im Sprint Planning Meeting 1, wie viel Funktionalität es liefern wird.

Done

Das entscheiden Prinzip ist: *Am Ende eines Sprints hat das Entwicklungsteam potenziell nutzbare Funktionalität zu liefern.* Das heißt keine weiteren Arbeiten sind notwendig, um diese Funktionalität an den End-User zu übergeben.

Diese Vorgabe muss an die jeweiligen Entwicklungsbedingungen angepasst werden. Deshalb wird zwischen dem *Entwicklungsteam* und dem *Product Owner* der *Level of Done* vereinbart. Der ScrumMaster arbeitet mit dem Scrum-Team daran, diesen kontinuierlich zu erhöhen. Im Idealfall wird am Ende des Sprints an den End-User ausgeliefert.[15]

Scrum in der Umsetzung

Scrum in Unternehmen kann nur gelingen, wenn auf zwei Ebenen, der *strategischen* und der *taktischen* gearbeitet wird. Um den *PDCA*-Zyklus auf beiden Ebenen effektiv durchzuführen, bedarf es einiger Meetings und Artefakte.

Die Meetings auf strategischem Level.
Im *Estimation Meeting* wird vom Team gemeinsam mit dem Product Owner ein erstes Grundverständnis über die zu liefernde Funktionalität erarbeitet und die Größenordnung eingeschätzt.[16]

Im *Business Value Estimation Meeting* werden die Product Backlogs unterschiedlicher Abteilungen gegeneinander priorisiert. Auf diese Weise kann jedes Team an den jeweils wichtigsten Inhalten arbeiten.

[15] Es gibt mittlerweile Scrum-Teams, die nicht nur am Ende des Sprints fertige Software an den Product Owner oder ihre internen Kunden ausliefern, sonderen diese Teams liefern kontinuierlich, d. h. während des Sprints, die neu entstandene Funktionalität an Kunden und End-User aus.

[16] In der agilen Softwareentwicklung werden Funktionsumfänge, nicht Aufwände geschätzt. Siehe dazu [8] und [3].

Im *ScrumMasters Weekly* besprechen die ScrumMaster aller Teams wichtige Änderungen und identifizieren Impediments auf organisatorischer Ebene und vereinbaren die Lösungsmaßnahmen.

In den *Knowledge Domain Meetings* treffen sich die Spezialisten der Teams, zum Beispiel die Datenbankentwickler, um fachliche Richtlinien zu erstellen.

Die Meetings – Taktischer Level.
Im *Sprint Planning Meeting 1* erstellt das *Entwicklungsteam* eine *Analyse* der zu liefernden Funktionalität. Basierend auf der genauen Kenntnis, wie die Funktionalität sein soll, wird vom *Scrum-Team* die Auswahl getroffen, was geliefert wird. Im *Sprint Planning Meeting 2* entsteht eine klare Vorstellung darüber, wie die jeweilige Funktionalität zu implementieren ist.

Ein tägliches Meeting, *Daily Scrum*, erzeugt die notwendige Sichtbarkeit, um aus der individuellen Leistung des Einzelnen eine *öffentliche* Leistung des Teams zu machen.[17] Die Visualisierung der Leistung des Teams erfolgt mithilfe eines *Taskboards* oder *Scrumboards*.[18] In diesem Meeting planen die Teammitglieder ihre täglichen Aktivitäten und sprechen sich untereinander ab.

Arbeiten mehrere *Scrum-Teams* daran, *ein* Produkt zu liefern, treffen sich die Teammitglieder aus den *Entwicklungsteams täglich* zum *Scrum of Scrums* (SoS). Hier werden die technischen Abhängigkeiten der Teams bekannt gegeben und dann von den *Entwicklungsteams* aufgelöst.

Die Product Owner bilden das *Product Owner Team*. Sie treffen sich täglich im *Product Owner Daily Scrum* (PODS), um die Koordination auf der Großprojektebene durchzuführen.

Im *Sprint Review* lassen die Teammitglieder den End-*User* mit der erstellten Funktionalität experimentieren. Der Product Owner und der ScrumMaster nehmen jede dabei entstehende Idee in das Product Backlog auf.

Im *Sprint Retrospective* Meeting berät das *Entwicklungsteam* gemeinsam mit dem ScrumMaster, wie man die Produktivität durch das Ausräumen von Impediments erhöhen kann.[19]

[17] Zur Veränderung in der Art und Weise, wie Software in der agilen Softwareentwicklung *gemacht* wird, hin zur einer Typisierung der agilen Softwareentwicklung als „kollektiv", „körperlich", „implizit" und „öffentlich", findet sich in [12].
[18] Die Darstellung, wie man mit dem Taskboard arbeitet, findet sich in [8].
[19] Eine gute Darstellung des gesamten Ablaufs findet sich in [8] und in [6].

Scrum in großen Umfeldern

In den letzten Jahren wurden Praktiken entwickelt, um Scrum auch in großen Projekten mit mehr als 100, ja 1000 Personen zum Steuern von Abteilungen und sogar Unternehmen einzusetzen.

In diesen Umfeldern werden synchronisierte Sprint Planning Meetings mit bis zu 15 Teams gleichzeitig, SoS mit bis zu 60 Personen, virtualisierte Daily Scrums und road-show-artige Sprint Reviews durchgeführt.

Organisationen setzen Scrum ein, wenn Teams an unterschiedlichen Standorten gemeinsam an einem Produkt arbeiten. Die Arbeiten von Jeff Sutherland zeigen, dass mit Scrum die Chance besteht, dass die Produktivität der Softwareentwicklung linear mit der Anzahl der Scrum-Teams wachsen kann [15]. Das wird erreicht, wenn ein tiefes Verständnis aller Beteiligten in der Organisation für die Wirkweise von Scrum besteht und der Paradigmenwechsel in der Durchführung von Entwicklungsprojekten bereits real gelebt wird.

Schluss

Prinzipien und Erkenntnisse des Knowledge Managements und des Toyota Production Systems sind mit Scrum in das Managen von Softwareprodukt-Entwicklungsprojekten eingeflossen. Viele Organisationen erleben mit Scrum erhöhte Produktivität ihrer Softwareentwicklungsteams. Der Paradigmenwechsel hat in diesen Firmen begonnen, und sie erleben eine gesteigerte Produktivität bei höherer Zufriedenheit ihrer Mitarbeiter. Damit ist Scrum aus der modernen Softwareentwicklung nicht mehr wegzudenken.

Literatur

1. Beck K, Andres C (2004) Extreme Programming Explained: Embrace Change, 2nd edn. Addison-Wesley Professional, Boston
2. Cockburn A (2006) Agile Software Development: The Cooperative Game, 2nd edn. Addison-Wesley Professional, Boston
3. Cohn M (2006) Agile estimating and planning. Prentice Hall Professional Technical Reference, Upper Saddle River, NJ
4. Coplien JO (1994) Borland software craftsmanship: A new look at process, quality and productivity. Software Production Research Department AT&T Bell Laboratories, Orlando, FL
5. Cutmull E (2008) How pixar fosters collective creativity. Harvard Bus Rev 9:60–72
6. Derby E, Larsen D (2006) Agile Retrospectives: Making Good Teams Great. Pragmatic Bookshelf, Raleigh, NC, Dallas, TX
7. Fisher K (2000) Leading Self-Directing Work Teams. McGraw-Hill, New York
8. Gloger B (2008) Scrum. Produkte zuverlässig und schnell entwickeln. Hanser Fachbuch, München
9. Green S (2009) A Year of Living Dangerously. How Salesforce.com delivered Extraordinatory Results through a "Big Bang" Enterprise Agile Revolution. Salesforce.com, Scrum Gathering, Stockholm

10. Liker J (2003) The Toyota Way. McGraw-Hill, New York Chicago San Francisco
11. Takeushi H, Nonaka I (1986) The new new product development game. Harvard Bus Rev
12. Schmidt R (2008) Praktiken des Programmierens – zur Morphologie von Wissensarbeit in der Softwareentwicklung. Z Soziol 37(4):282–300
13. Schwaber K (2004) Agile project management with Scrum. Microsoft Press, Redmond, WA
14. Schwaber K (2007) The enterprise and Scrum. Microsoft Press, Redmond, WA
15. Sutherland J, Downey S, Granvik B (2009) Shock Therapy: A Bootstrap for Hyper-Productive Scrum. agile, 2009 Agile Conference, pp 69–73