Agil verlangt schnelle Reaktionen auf Veränderungen und das bedeutet, Probleme so früh wie möglich sichtbar zu machen. Leider möchten neue oder durchschnittliche Teams häufig gar nicht, dass Probleme sichtbar werden. Insbesondere möchten Sie nicht die Arbeit einstellen, Probleme beheben und Kritik riskieren. Das japanische Toyota-Management besichtigte nach sechs Monaten das erste NUMMI (New United Motor Manufacturing, Inc) Toyota-Werk in Amerika und stellte fest, dass Angestellte Angst hatten, das *Andon*-Cord (*Andon* = Lampe) zu ziehen – die Reißleine, die dafür sorgt, dass eine Warnlampe leuchtet und einen Countdown startet, der das Fließband stoppt). Die Arbeiter hatten das *Andon*-Cord zur Beseitigung von Hindernissen nicht ausreichend genutzt. Das Management zog das *Andon*-Cord, um das Fließband sofort anzuhalten und den Arbeitern zu vermitteln, dass das größte Hindernis ihre Widerwilligkeit sei, das Band zu stoppen. Nur durch ein Anhalten des Fließbandes werden Probleme aufgedeckt und schnell behoben. „Kein Problem ist ein Problem" lautet das japanische Management-Mantra (*MIT Sloan Management Review 51 [Sho10]*, S. 63–68).

Das Team muss sich an den *Product Owner* wenden, wenn die Dinge nicht gut laufen. Nicht nur das, das *Entwicklungsteam* muss sich auch mit ihm darüber einigen, wie man schnell große Probleme angeht, die Auswirkungen auf das Erreichen des *Sprint-Ziels* haben.

Daher gilt:

Wenn Sie sich noch oben im Burn-down befinden, versuchen Sie eine Technik, die routinemäßig von Piloten angewandt wird. Wenn schlimme Dinge passieren, führen Sie die *Sofortmaßnahmen* durch, die speziell für dieses Problem entwickelt wurden.

Verzögern Sie die Durchführung nicht dadurch, dass Sie versuchen herauszufinden, was falsch läuft oder was zu tun ist. In einem Kampfflugzeug könnten Sie in kürzerer Zeit tot sein, als es dauert, das Problem zu ergründen. Der *¶19 Scrum Master* ist dafür verantwortlich, sicherzustellen, dass das Team umgehend die Scrum-*Sofortmaßnahmen* durchführt, besonders wenn die Dinge bis zur Mitte des *Sprints* vom Kurs abkommen. Das erfordert eine sorgfältige Koordination mit dem *Product Owner.* Die Kaizen-Denkweise verlangt den Einsatz dieses Musters auch, wenn der *Product Owner* nicht zur Verfügung steht. Sehr gute Teams handeln ohne Erlaubnis und bitten später um Entschuldigung (siehe *¶95 Vertrauensgemeinschaft* auf Seite 424).

Scrum-Sofortmaßnahmen: (Tun Sie nur so viel wie nötig)

1. Ändern Sie die Arbeitsweise des Teams. Tun Sie etwas anderes.
2. Nehmen Sie Hilfe in Anspruch, normalerweise indem Sie das Backlog an jemand anderen übergeben.
3. Reduzieren Sie den Umfang.
4. Brechen Sie den *Sprint* ab und planen Sie neu.
5. Informieren Sie das Management, inwieweit der Notfall Auswirkungen auf den Release-Termin hat.

Sobald Teams auf Schwierigkeiten stoßen, möchten Sie häufig den Umfang reduzieren. Sehr gute Teams wählen stattdessen eine andere Strategie, um das *Sprint-Ziel* zu erreichen. In der Fußballsaison 2005/6 musste John Terry, Kapitän und Innenverteidiger beim FC Chelsea, in einem Spiel gegen den FC Reading als Torwart einspringen, nachdem Petr Čech einen Schädelbruch erlitten hatte und der Ersatztorwart Carlo Cudicini kurz vor Spielende bewusstlos vom Platz getragen wurde. Terry hatte zwei gute Paraden und Chelsea gewann 1:0. Auch bei der Software können neue Verfahren, die die Verschwendung beseitigen, die Leistung vervielfachen, und gleichzeitig den Aufwand drastisch verringern.

Wenn mehrere Teams an denselben Produkten arbeiten, kann ein Team das Backlog oft an ein anderes Team weitergeben, das mehr Spielraum hat. Das Unternehmen PatientKeeper, ein Pionier der agilen Entwicklung im medizinischen Bereich, automatisierte diese Strategie (*Proceedings of Agile Development Conference (ADC'05) [Sut05]*). Wenn ein Team zurücklag, konnte es *¶73 Sprint-Backlog-Einträge* an ein anderes Team übergeben. Wenn dieses zweite Team sie nicht nehmen konnte, reichte es sie an ein drittes Team weiter. Wenn dieses Team auch nicht in der Lage war, daran zu arbeiten, trafen sich die drei Teams, um zu entscheiden, was zu tun war. Das verteilte die Belastung durch das Backlog automatisch auf mehrere Teams, sodass sie es gemeinsam fertigstellen konnten.

Es ist besser, frühzeitig den Umfang zu reduzieren, damit das Team geplante Arbeit fertigstellen kann, als auf einen Misserfolg zuzusteuern. Statt überrascht zu werden, kann die Organisation überprüfen und sich an die Situation anpassen. Siehe *¶74 Teams, die früh fertig werden, beschleunigen schneller.*

Den *Sprint* abzubrechen (die Verbindung unterbrechen) kann die beste Option sein, besonders wenn das Team wiederholt nicht ausliefern kann. Nur der *Product Owner* kann entscheiden, ob der *Sprint* abgebrochen wird: So schlimm es auch sein mag, der *Product Owner* kann auch zu dem Schluss kommen, dass sich das Geschäft möglicherweise nicht lohnt, oder dass der Abbruch des *Sprints* langfristige, negative Konsequenzen für den Markt oder das Geschäft haben könnte.

Nachdem der *Sprint* beendet ist, hält das Team normalerweise ein kurzes *¶24 Sprint Planning* für einen verkürzten *Sprint* ab (um in der Kadenz zu bleiben wie bei *¶47 Organisatorischer Sprint-Takt*; siehe auch *¶77 Follow the Moon*), und so das *Sprint-Ziel* möglichst zu erreichen und so viel Wert wie möglich auszuliefern. Alternativ kann das Team auch eine ausgedehntere *¶36 Sprint-Retrospektive* abhalten, um Probleme in der Umgebung oder bei der Scrum-Implementierung des Teams zu sondieren und zu beheben. Dann kann es neu planen und zum nächsten *Sprint* übergehen. Aber noch einmal, wertvoll wird ein *Sprint*-Abbruch dadurch, dass man fundamentale Hindernisse öffentlich sichtbar macht, die das Team davon abhalten, seinen Job zu erledigen. Ein sichtbares Problem ist eines, das das Team lösen kann.

Die Beendigung eines *Sprints* sendet eine starke Botschaft durch die Organisation, dass etwas falsch läuft, und sie erhöht die Fähigkeit, Hindernisse zu beseitigen, die die Ursache für den Misserfolg sind. Eine spielerische Scrum-Tradition (oder zumindest eine Metapher) ist die „Zeremonie zum Sprint-Abbruch", die angeblich in den Lobbys von Unternehmenszentralen abgehalten wird, wo die Mitglieder des *Entwicklungsteams* sich versammeln, sich auf den Rücken legen, schreien und mit Armen

und Beinen in der Luft rudern, um Dampf abzulassen. Die Absicht dahinter ist, zu verdeutlichen, dass der *Product Owner* die Verpflichtung des Teams aufgehoben hat.

Das *¶7 Scrum-Team* wendet dieses Muster an, und es ist besonders nützlich für Hochleistungsteams. Für Teams, die ernsthaft Kaizen praktizieren (siehe Kaizen und Kaikaku auf Seite 92), ist Scrum ein Extremsport, und sie beginnen einen *Sprint* mit einigem Risiko, um schneller zu sein. Ihr größtes Risiko sind emergente Anforderungen oder unerwartete technische Probleme. Alle anderen Ursachen für Misserfolge hat das Team bereits bearbeitet. Das Team kann dieses Muster jeden dritten oder vierten *Sprint* nutzen, besonders dann, wenn es neue Technologien implementiert und den derzeitigen Stand der Dinge vorantreiben will. Sehr gute Teams werden sich jedoch von den meisten Notfällen erholen und die *Sprint-Ziele* erreichen. Wenn sie die Verbindung unterbrechen (den *Sprint* abbrechen), dann werden sie für ihren Prozess Poka Yoke nutzen (*Toyota Kata: Managing People for Improvement, Adaptiveness and Superior Results [Rot10]*, damit dasselbe Problem nicht noch einmal auftritt.

> **Poka Yoke** *(ポカヨケ)* ist ein japanischer Begriff, der „versehentliche Fehler vermeiden“ oder „Fehlersicherung“ bedeutet. Ein Poka Yoke ist jeder Mechanismus in einem Lean-Produktionsprozess, der einem Geräteführer hilft, einen Fehler (Poka) zu vermeiden (Yokeru). Sein Zweck ist es, Produktfehler zu eliminieren, durch Vorsorge, Korrektur oder das Lenken der Aufmerksamkeit auf menschliche Fehler, sobald sie auftreten.[25]

Probleme sichtbar zu machen, ist Teil der Kaizen-Denkweise; siehe Kaizen und Kaikaku.

Das Team lernt, schnell und diszipliniert auf Veränderungen zu reagieren und Schwierigkeiten zu überwinden. Sobald die Dinge nicht gut laufen, denken Teams in vielen Organisationen nicht mehr klar und sind frustriert und demotiviert. Sie verstehen die Ursache ihrer Probleme und deren Lösung nicht. Die *Sofortmaßnahmen* anzuwenden, wird das Team darin schulen, sich auf den Erfolg zu fokussieren und Hindernisse systematisch aus dem Weg zu räumen. Großartige Teams werden sich selbst mit ihrer Fähigkeit überraschen, Widerstände zu überwinden und dabei immer stärker werden. Es steigert die Chancen, erfolgreich ein *¶85 Regelmäßiges Produktinkrement* auszuliefern, sowohl kurz- als auch langfristig. Wenn es *Sofortmaßnahmen* nutzt, hat das Team das Gefühl, alles was möglich ist, zu tun, um wieder in die Spur zu kommen, angetrieben von beiden Arten von Stolz (siehe *¶118 Team Pride* auf Seite 426 und *¶38 Product Pride*).

Mit dem Muster *Illegitimus Non Interruptus* können Sie *Sofortmaßnahmen* noch disziplinierter umsetzen, um die Transparenz bei unkontrollierten Anforderungen zu erhöhen.

Siehe auch *¶97 Keine kleinen Ausrutscher* auf Seite 424.

[25] „Poka Yoke“, Wikipedia, https//en.wikipedia.org/wiki/Poka-yoke, 19. Mai 2018 (aufgerufen am 6. Juni 2018).

Dieses Muster ist für den Einsatz in hochdisziplinierten Teams gedacht. Wenn ein Team es zu oft anwendet (z. B. öfter als einmal alle vier *Sprints*) und seinen Wert, die Qualität und die Auslieferungsquote nicht verbessert, dann sollte das Team darüber nachdenken, ob etwas grundsätzlich falsch läuft in der Umgebung oder in der Nutzung von Scrum im Team. Normalerweise ist es für junge Teams besser, ihr Bestes zu geben, um auszuliefern, den *Sprint* bis zu Ende abzuhalten und dann im *Sprint* zu versagen. Abseits der Hitze des Gefechts der *Sprint-Retrospektive* kann das Team die Ursachen für den Misserfolg erforschen und Kaizen planen. Einige Prozessverbesserungen können dem Team helfen, zukünftig in entsprechenden Situationen einen Ausweg mithilfe der *Sofortmaßnahmen* zu suchen.

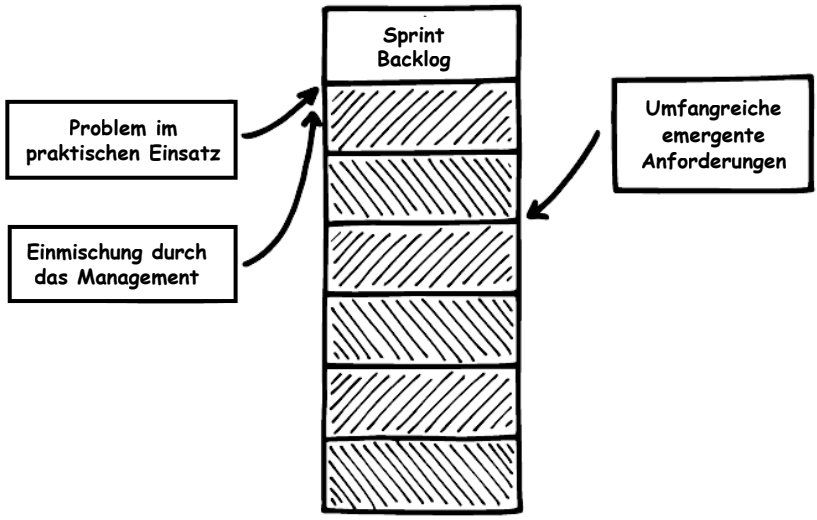
¶33 Illegitimus Non Interruptus

Bewertungssterne: *

Kühe grasen auf freiem Land auf der Halbinsel Gower.

... das *¶7 Scrum-Team* dient vielen Stakeholdern, die alle um die Aufmerksamkeit des Teams konkurrieren. Das Team bekommt Anfragen und Forderungen aus dem Management, von Kunden, vom Vertrieb und vom Marketing. Darüber hinaus kann die laufende Arbeit unerwartete Mängel beim Produkt selbst aufdecken, die bearbeitet werden müssen. Die Häufigkeit und die Bedeutung solcher Anfragen variieren mit der Zeit, und gelegentlich sind ihr Umfang und ihre Dringlichkeit überwältigend.

Sich ändernde Prioritäten und Probleme im praktischen Einsatz unterbrechen häufig die Arbeit des *Scrum-Teams* während eines *¶46 Sprints*. Vertriebs- und Marketinganforderungen in Kombination mit einer Einmischung des Managements können chronische Funktionsstörungen im Team, wiederholtes Scheitern von *Sprints*, Nichteinhalten von Release-Terminen und sogar ein Versagen des Unternehmens zur Folge haben.



In vielerlei Hinsicht ist das *Scrum-Team* eine Gemeinschaftsressource, die die Bedürfnisse vieler Stakeholder erfüllt. Die Tragik der Allmende ist ein Dilemma, das dadurch entsteht, dass mehrere Einzelpersonen unabhängig voneinander handeln und rational ihre eigenen Interessen verfolgen, wodurch letztlich eine geteilte, begrenzte Ressource erschöpft wird, selbst wenn klar ist, dass das langfristig niemand wollen kann. Der amerikanische Ökologe und Philosoph Garrett Hardin beschrieb dieses Dilemma zuerst in dem wegweisenden Artikel „The Tragedy of the Commons", der erstmals 1968 in der Zeitschrift *Science* erschien.[26]

Das *Scrum-Team* ist eine wichtige Ressource für das Erstellen einer neuen und die Instandhaltung der alten Software. Damit ist es eine zentrale Ressource für die Lösung von Problemen, die sowohl während der Entwicklung als auch während der Anwendung des Produktes auftreten, für die technische Kommunikation mit den Kunden, für Marketingpräsentationen und für spezielle Projekte, die den Bedürfnissen aller in der Organisation gerecht werden. Siehe ¶*102 Work Flows Inward* auf Seite 424.

Oft bekommt das *Scrum-Team* aufgrund einer unzureichenden Product-Ownership konkurrierende Prioritäten. Einige Teams wurden sogar schon erpresst, um an Features zu arbeiten, die sich nicht im ¶*54 Product Backlog* befanden.

In fast allen Fällen ist es wünschenswert, dass das *Scrum-Team* die eigenen Produkte auch selbst anwendet. Wenn sie einen Fehler produzieren, der dann in den Markt kommt, müssen sie ihn so schnell wie möglich reparieren. Die Einrichtung spezieller Wartungsteams zur Fehlerbeseitigung kann dazu führen, dass das *Scrum-Team* nicht so motiviert ist, auf latente Fehler zu achten.

[26] „The Tragedy of the Commons", Wikipedia, https://en.wikipedia.org/wiki/Tragedy_of_the_commons, Juni 2018 (aufgerufen am 6. Juni 2018).

Aus diesen und vielen anderen Gründen wird das *Scrum-Team* immer wieder unterbrochen, sodass die Produktion gestört wird.

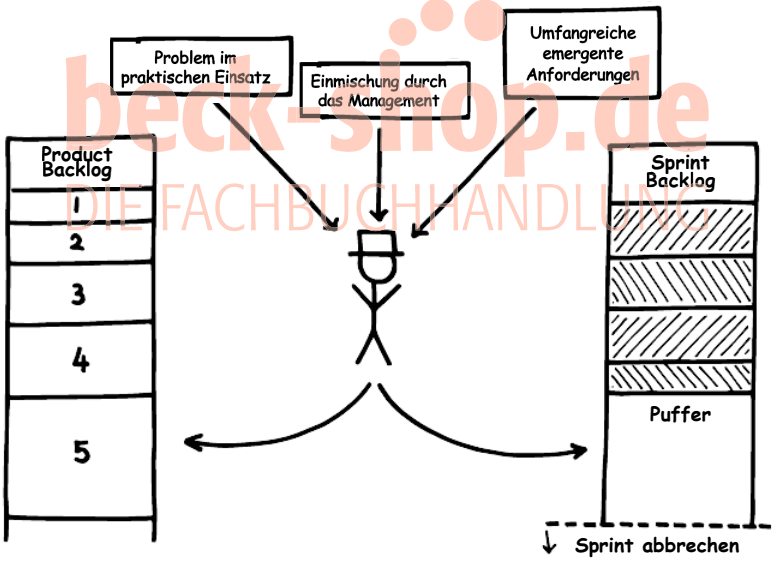
Daher gilt:

Räumen Sie eine bestimmte Zeit für Unterbrechungen ein, und genehmigen Sie keine Arbeit, die diese Zeit überschreitet. Sollte das dennoch der Fall sein, brechen Sie den *Sprint* ab.

Um zu vermeiden, dass die Produktion gestört wird, stellen Sie drei einfache Regeln auf, die dazu führen, dass die Organisation sich selbst organisiert.

Diese Strategie wird dem Team helfen, während des *Sprints* neu zu planen, und so die Chancen erhöhen, das vollständige ¶*85 Regelmäßige Produktinkrement* auszuliefern.

1. Das Team schafft einen Puffer für unerwartete Items, basierend auf historischen Daten. Nehmen wir beispielsweise an, dass ein Drittel der Arbeit des Teams ungeplant ist, da sie unerwartet im *Sprint* auftritt. Falls die Team-Velocity durchschnittlich 60 Punkte erreicht (Anmerkungen zur Velocity auf Seite 287), reserviert das Team 20 Punkte für den Unterbrechungspuffer.



2. Alle nicht trivialen Anfragen müssen zur Sichtung an den ¶*11 Product Owner* gehen. (Rechtschreibfehler auf Webseiten und Kompilierungsfehler sind Beispiele für triviale Fehler, deren Behebung so eindeutig ist, dass es nichts bringt, wenn die Unternehmensseite sich damit beschäftigt. Entwickler können in einem kurzen, befristeten Zeitraum auch nicht triviale Fehler bearbeiten, bevor sie diese an den *Product Owner* weiterleiten.) Der *Product Owner* wird einigen Items eine niedrige Priorität zuweisen, falls es keinen wahrnehmbaren Wert in Bezug auf den Business Plan gibt. Der *Product Owner* wird viele weitere Items in

nachfolgende *Sprints* verschieben, selbst wenn sie einen sofortigen Wert bringen. Einige Items sind wichtig, und das Team muss sie im laufenden *Sprint* fertigstellen, also packt der *Product Owner* sie in den Puffer für Unterbrechungen.

3. Falls der Puffer überläuft, also wenn der *Product Owner* einen Punkt mehr als 20 in den *Sprint* nimmt, muss das *Scrum-Team* automatisch abbrechen, der *Sprint* neu geplant werden und der *Product Owner* das Management informieren, dass die Termine sich verschieben.

Es ist wichtig, dass das Management mit diesen Regeln einverstanden ist und sie unterstützt. Der *Product Owner* muss immer für das Team und andere Stakeholder zur Verfügung stehen. Wenn der *Product Owner* abwesend ist, sollte das Team einen eigenen einsetzen, der diese Rolle temporär übernimmt.

Der *Product Owner* balanciert die Puffergröße aus, um ein Gleichgewicht zwischen kurzfristiger Befriedigung der Kundenwünsche und zukünftiger Umsatzgenerierung herzustellen. Häufig besitzt der *Product Owner* Kennzahlen einer dritten Partei über die Kundenzufriedenheit, die er mithilfe der Puffergröße nach oben oder unten anpassen kann.

Die Strategie ist unabhängig von dem Fokus auf die Reparatur aller Fehler, die im *Sprint* durch bearbeitete Backlog-Einträge auftreten (siehe *¶80 Good Housekeeping*). Sie ist ebenfalls unabhängig von *¶55 Product-Backlog-Einträgen*, die durch den *Product Owner* als Teil des *¶24 Sprint Plannings* für den *Sprint* vorgesehen wurden, um technische Schulden zu reduzieren. Eine niedrige Fehlertoleranz steigert im Allgemeinen die Velocity, aber den Puffer zu erweitern, reduziert sie normalerweise um mindestens 50 Prozent. Der *Product Owner* muss den gesunden Menschenverstand einsetzen, um diese Kräfte auszubalancieren. Siehe *¶81 Whack the Mole*.

Diese Regeln werden immer dazu führen, dass einzelne Personen sich selbst organisieren, um zu vermeiden, den *Sprint* zu torpedieren, da niemand als direkte Ursache dafür wahrgenommen werden möchte.

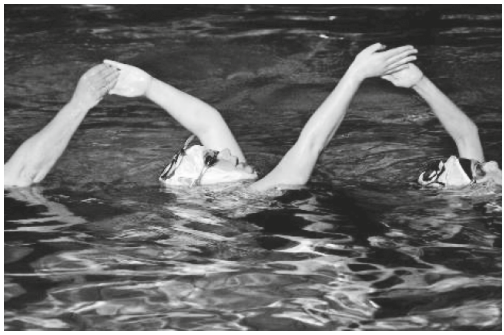
Besser ist es, wenn der Puffer niemals voll ist, und es so dem Team möglich ist, früh fertigzustellen und im Backlog weiter voranzukommen und/oder Hindernisse zu beseitigen. Das ist wichtig, denn *¶74 Teams, die früh fertig werden, beschleunigen schneller*. Falls das Team *¶66 Yesterday's Weather* nutzt, um die Größe des Puffers zu bestimmen, und sich der Puffer fast nie füllt, wird die Puffergröße darüber hinaus kontinuierlich abnehmen und das störende Problem verschwindet.

Entgegen der Intuition verursacht das keine großen Probleme, die verdeckt oder ungelöst bleiben. Der *Product Owner* wird alle wichtigen Items in das *Product Backlog* packen. Es hilft dem Team, seine Velocity und den Output zukünftiger *Sprints* zu steigern und bietet normalerweise mehr als genug Zeit, wichtige Items zu bearbeiten, oft sogar mit knappen Kapazitäten.

Ein Team zeigt ein hohes Maß an *¶38 Product Pride*, wenn es in der Arbeit zum Wohle der Produktqualität und des Rufs innehält. Weitere damit verbundene Muster umfassen: *Product Owner, Product Backlog, Teams, die früh fertig werden, beschleunigen schneller, Work Flows Inward* auf Seite 424 und *¶98 Completion Headroom* auf Seite 424.

¶34 Scrum of Scrums

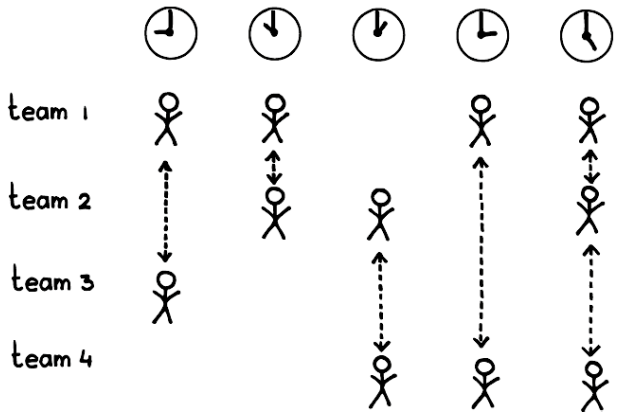
Bewertungssterne: *



... ein *Scrum-Team* arbeitet mit mehreren *¶14 Entwicklungsteams* an einem einzigen Produkt. Die *Entwicklungsteams* müssen Abhängigkeiten und die gemeinsame Arbeit koordinieren. Unaufgelöste Abhängigkeiten innerhalb einzelner Teams sind eine gemeinsame Herausforderung für alle Teams.

Wenn mehrere Teams unabhängig voneinander arbeiten, neigen sie dazu, sich kurzsichtig auf ihre eigenen Belange zu fokussieren und gemeinsame Ziele aus dem Blick zu verlieren.

Organisationen fallen vielleicht in einen Command-and-Control-Stil zurück, in der falschen Annahme, dass Agilität nur im Rahmen eines einzigen Teams funktioniert. Doch die Komplexität nimmt in diesem Fall nicht ab, sondern zu. Durch hierarchische Kontrolle kommt es vermehrt zu Verzögerungen, und die Reaktionsfähigkeit der Teams und der umfassenderen Organisation auf Geschäfts- und Technologieänderungen wird reduziert.



Das *Scrum-Team* könnte das Problem in kleinere Portionen herunterbrechen, sodass jedes *Entwicklungsteam* separat an einem Teil des Liefergegenstandes arbeiten