Bei einem plangesteuerten Ansatz findet die Iteration innerhalb solcher Aktivitäten statt, bei denen formale Dokumente zur Kommunikation zwischen den Prozessphasen benutzt werden. Zum Beispiel entwickeln sich die Anforderungen stetig weiter, bis schlussendlich eine Anforderungsspezifikation erstellt wird. Diese wird dann zur Eingabe des Entwurfs- und Implementierungsprozesses. Bei einem agilen Ansatz findet die Iteration in allen Aktivitäten statt. Daher werden die Anforderungen und der Entwurf zusammen statt getrennt entwickelt.

Ein plangesteuerter Softwareprozess kann inkrementelle Entwicklung und Auslieferung unterstützen. Es ist durchaus möglich, die Zuteilung von Anforderungen sowie die Entwurfs- und Entwicklungsphase als eine Folge von Inkrementen zu planen. Ein agiler Prozess ist nicht zwangsläufig codezentriert und er könnte einige Entwurfsdokumente erzeugen. Wie im folgenden Abschnitt noch erläutert wird, könnte das agile Entwicklerteam entscheiden, einen sogenannten „Spike" für die Dokumentation einzulegen: Anstatt eine neue Version des Systems zu produzieren wird an der Systemdokumentation gearbeitet.

Tatsächlich beinhalten die meisten Softwareprojekte sowohl Vorgehensweisen von plangesteuerten als auch von agilen Ansätzen. Um über die Balance zwischen einem plangesteuerten und einem agilen Ansatz zu entscheiden, müssen eine Reihe von technischen, personellen und organisatorischen Fragen beantwortet werden:

1. Ist es wichtig, Spezifikation und Entwurf sehr detailliert auszuarbeiten, bevor man zur Implementierung übergeht? Falls ja, dann braucht man wahrscheinlich einen plangesteuerten Ansatz.

2. Ist eine inkrementelle Auslieferungsstrategie realistisch, bei der man die Software an die Kunden liefert und schnelle Rückmeldungen erhält? Falls ja, sollte der Einsatz von agilen Methoden in Betracht gezogen werden.

3. Wie groß ist das System, das entwickelt wird? Agile Methoden sind am effektivsten, wenn das System mit einem kleinen, in einem Raum untergebrachten Team, das sich informell austauschen kann, entwickelt wird. Dies ist eventuell für große Systeme, die größere Entwicklerteams benötigen, nicht möglich, sodass hier ein plangesteuerter Ansatz benutzt werden sollte.

4. Welche Art von System wird entwickelt? Systeme, bei denen vor der Implementierung ein hoher Analysebedarf besteht (z. B. Echtzeitsysteme mit komplexen Anforderungen an die zeitliche Synchronisation), benötigen gewöhnlich einen ziemlich detaillierten Entwurf, um diese Analysen auszuführen. Ein plangesteuerter Ansatz könnte unter diesen Umständen am besten sein.

5. Wie ist die erwartete Lebenszeit des Systems? Langlebige System erfordern möglicherweise mehr Entwurfsdokumentation, um dem Supportteam die ursprünglichen Absichten der Systementwickler zu vermitteln. Anhänger der agilen Methoden argumentieren zu Recht, dass Dokumentation oft nicht aktuell gehalten wird und für lang dauernde Systemwartung nicht von Nutzen ist.

6. Welche Technologien sind verfügbar, um die Systementwicklung zu unterstützen? Agile Methoden beruhen häufig auf guten Werkzeugen zur Beobachtung eines sich weiterentwickelnden Entwurfs. Wenn Sie ein System mithilfe einer IDE entwickeln, die keine guten Werkzeuge für Programmvisualisierung und -analyse bietet, dann ist vermutlich mehr Entwurfsdokumentation nötig.

7. Wie ist das Entwicklerteam organisiert? Falls das Entwicklerteam dezentral aufgestellt ist oder Teile der Entwicklung ausgelagert sind, dann muss man eventuell Entwurfsdokumente erstellen, um die Kommunikation zwischen den einzelnen Entwicklerteams zu gewährleisten. Es muss eventuell im Voraus geplant werden, welche Dokumente dies sein sollen.

8. Gibt es kulturelle Aspekte, die die Systementwicklung beeinflussen können? Traditionelle Unternehmen haben eine Kultur von planungsbasierter Entwicklung, wie dies der Standard im Ingenieurwesen ist. Dies erfordert gewöhnlich ausführliche Entwurfsdokumentation statt des informellen Wissens, das in agilen Prozessen benutzt wird.

9. Wie gut sind die Entwickler und Programmierer im Team? Es wird manchmal angeführt, dass für agile Methoden höheres fachliches Können als für planungsbasierte Ansätze, bei denen Programmierer einfach einen detaillierten Entwurf in Code übersetzen, erforderlich ist. Falls das Team eher geringer qualifiziert ist, muss man die besten Leute für den Entwurf abstellen, die anderen sind für die Programmierung verantwortlich.

10. Unterliegt das System externen Vorschriften? Falls ein System von einer externen Regulierungsbehörde genehmigt werden muss (z. B. wird Software, die kritisch für den Betrieb eines Flugzeugs ist, in den USA von der Federal Aviation Authority (FAA) genehmigt), dann wird man wahrscheinlich genötigt sein, ausführliche Dokumentation als Teil der Systemsicherheit zu erstellen.

Tatsächlich ist die Frage, ob ein Projekt als plangesteuert oder agil bezeichnet werden kann, nicht sehr wichtig. Letztendlich ist die Hauptsorge von Käufern eines Softwaresystems, ob sie ein ausführbares Softwaresystem bekommen, das ihre Bedürfnisse befriedigt und für den individuellen Einsatz oder die Organisation nützlich ist oder nicht. In der Praxis haben viele Unternehmen, die behaupten, agile Methoden benutzt zu haben, ein paar agile Verfahrenweisen angenommen und diese in ihre plangesteuerten Prozesse integriert.

3.3 Extreme Programming

Extreme Programming (XP) ist wahrscheinlich die bekannteste und am häufigsten verwendete agile Methode. Die Bezeichnung wurde von Beck geprägt (Beck, 2000), da dieser Ansatz dadurch entwickelt wurde, bekannte gute Praktiken wie iterative Entwicklung „ins Extreme" zu steigern. Zum Beispiel könnten in XP mehrere neue Versionen eines Systems an einem Tag von verschiedenen Programmieren entwickelt, integriert und getestet werden.

Beim Extreme Programming werden Anforderungen in Form von **Szenarios** ausgedrückt, die User-Storys genannt werden. Diese Szenarios werden direkt in Form einer Abfolge von Aufgaben implementiert. Die Programmierer arbeiten paarweise zusammen, um Tests für die einzelnen Aufgaben zu entwickeln, bevor sie Code schreiben. Alle Tests müssen erfolgreich ausgeführt werden, wenn neuer Code in das System integriert wird. Zwischen den einzelnen **Releases** des Systems vergeht nur wenig Zeit. ▶ Abbildung 3.3 veranschaulicht den XP-Prozess für ein Inkrement eines zu entwickelnden Systems.