# Agile Softwareentwicklung in der Wissenschaft
## Beleuchtung von Fallstudien

**— Seminararbeit —**

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

Vorgelegt von: Schacht, Fabian
E-Mail-Adresse: 1schacht@informatik.uni-hamburg.de
Matrikelnummer: 5093623
Studiengang: Informatik

Betreuer: Julian Kunkel

Hamburg, den 31.08.2015

# Inhaltsverzeichnis

# Agile Softwareentwicklung in der Wissenschaft
## Beleuchtung von Fallstudien

***Fabian Schacht,*** *Universität Hamburg*

**Diese Arbeit gliedert sich in drei Hauptteile. In jedem Teil wird mindestens eine Fallstudie beleuchtet. Im ersten Teil geht es um die CSE Domäne und ob diese von agilen Entwicklungsmethoden profitieren könnte. [1] Im zweiten Teil geht es um die Nachteile und Grenzen agiler Prozesse im allgemeinen und wie sich diese über die Jahre verändert haben. [2] Im dritten Teil wird eine Fallstudie vorgestellt, die Beispiele aus der Praxis auswertet, um festzustellen in wie weit agile Methoden in der wissenschaftlichen Softwareentwicklung genutzt werden. [3]**

## 1 Einleitung

### Die Computational Science and Engineering (CSE) Domäne

Als Repräsentant für Softwareentwicklung in der Wissenschaft insbesondere um die Schnittstelle zwischen den Ingenieurswissenschaften, der Mathematik und der Informatik zu beleuchten wird die CSE-Domäne herangezogen.

Das Hauptziel von CSE ist es, Aufgabenstellungen mit Hilfe von Computerprogrammen und spezieller Computerhardware zu modellieren und zu simulieren. Des weiteren wird ein besonderes Augenmerk auf die Theorie zur numerischen Lösung partieller Differentialgleichungen, auf Datenanalyse und auf parallele Hochleistungsrechner gelegt.

Als Motivation wird gezeigt, warum agiles Vorgehen in Bereich CSE sinnvoll sein könnte. Im Hauptteil geht es um die Anwendbarkeit agiles Vorgehens im Bereich CSE, um die Grenzen und Nachteile des agilen Vorgehens im allgemeinen und wann es sinnvoll ist agile Vorgehensweisen zu benutzen.

Zwei agile Rahmenwerke, Scrum und XP, werden betrachtet um zu zeigen, ob diese sich für die Anwendbarkeit der CSE Domäne eignen.

Zum Abschluss wird noch eine Studie vorgestellt, die die Umsetzung agiler Methoden in der wissenschaftlichen Softwareentwicklung recherchiert hat.

## 2 CSD vs. CSE

### Kommerzielle Software Development Methoden im Gegensatz zu den Anforderungen der CSE Domäne

Betrachtet man kommerzielle Software Development Projekte und Methoden, und vergleicht diese mit Projekten in der CSE Domäne, so findet man Problemstellungen, die nur eingeschränkt von traditionellen SD Methoden gelöst werden.

So sind kommerzielle Projekte umfangreich, gewinnorientiert und versuchen eine möglichst große User Base anzusprechen. Sie haben eine erwartete lange Lebensdauer, während dieser sie weiter entwickelt werden und Wartung benötigen. Entwickelt werden diese oft von einer Gruppe, oder sogar mehreren Gruppen, von erfahrenen Entwicklern die in den SD Methoden bewandert sind.

In der CSD Domäne hingegen finden sich neben großen auch viele kleine Projekte, deren User Base klein und deren Lebenserwartung kurz ist, da sie oft von nur einem oder wenigen unerfahrenen Entwicklern für nur einen bestimmten Nutzen entwickelt werden. Da, aus verschiedenen Gründen, die Entwickler nicht selten die Wissenschaftler selbst sind und wenig Erfahrung mit der Entwicklung von Software haben ist die Gefahr groß, dass sie sich im Detail oder in der nötigen Verwaltung verlieren.

[1] Siehe: Is Scrum and XP suitable for Computational Science and Engineering (CSE) Development? (2012) [Blo12]
[2] Siehe: Limitations of Agile Processes 2002 [TFR02] , 2009 [Col09] , 2014/2015 [MVS15]
[3] Siehe: What Do We Know about Scientific Software Development's Agile Practices? [SHPL12]

# 3 CSE Development und Agile Development

Aus oben genannten Gründen folgt die Hypothese, dass für die CSE Domäne eine andere Herangehensweise nützlicher sein könnte als traditionelle SD Methoden. Im Weiteren wird erörtert, ob sich agile Methoden besser eignen.

## 3.1 Welche Relevanz hat das Agile Manifest für die CSE Domäne

Agile Methoden entstanden in der Mitte der 90er Jahre und 2001 wurde das „Agile Manifest“ als ein Meilenstein entwickelt. Laut Blom sind alle 12 Prinzipien relevant oder extra relevant für die CSE Domäne woraus er folgert, das agile Softwareentwicklung gut anwendbar ist für die CSE Domäne.

(Siehe Tabelle: Agil Manifesto)

**Tabelle 1:** *Agile Manifesto* [Blo12]

| | |
|---|---|
| • | Customer Satisfaction |
| •• | Embrace Change |
| • | Frequent Deliveries |
| • | Work Together |
| •• | Motivated Individuals |
| • | Face-to-face Conversation |
| •• | Working Software |
| • | Sustainable Development |
| • | Technical Excellence |
| • | Simplicity |
| • | Self-organizing Teams |
| • | Self-reflection |

less relevant: (no dot) | relevant: • | extra relevant: ••

Als extra relevant sieht er „Embrace Change“, „Motivated Individuals“ und „Working Software“.

Embrace Change gilt als Hauptgesichtspunkt von agilem Vorgehen und als Motivation um agile Methoden einzusetzen somit ist es immer extra relevant. Für die CSE Domäne ist es noch aus einem zweiten Gesichtspunkt extra relevant, da „CSE development is more exploratory than most non-CSE development and being prepared to change the software to cope with changing requirements is hence extra relevant.“ [Blo12]

Motivierte Mitwirkende sind zwar immer wünschenswert, aber Blom liefert keine Erklärung dazu, warum dies für die CSE Domäne extra relevant ist, sondern widerspricht sich eher in dem er sagt, „[...] motivation is generally in place to start with since the CSE developers are producing software for their own use.“ [Blo12]

Face-to-face Conversation gibt es immer seltener in der heutigen Zeit und obwohl die direkte Kommunikation einen sehr hohen Stellenwert im agilen Vorgehen hat und laut Blom sogar viele Entwickler in der CSE Domäne alleine arbeiten, sieht Blom hier kein Problem. Heutige Methoden der Kommunikation wie Videokonferenzen oder ähnliches erzielen hier nicht die gewünschten Ergebnisse.

Laut Blom sind viele Programmierer im Bereich CSE unerfahren. Dies steht meines Erachtens aber im Widerspruch zu den relevanten, oder extra relevanten, Punkten Technical Excellence, sowie Simplicity und Working Software– die meiner Meinung nach Technical Excellence voraussetzen.

Zusammenfassend ist laut Blom jeder Punkt relevant oder extra relevant - oder zumindest wünschenswert. Somit zieht Blom das Fazit, dass das agile Vorgehen eine Bereicherung für die CSE Domäne sein sollte.

Der von Blom herausgestellte Fakt, dass CSE Entwicklung näher an der Forschung ist als nicht-CSE Entwicklung, sowie die damit verbundene Tatsache von stetigen Anforderungsänderungen sind ein guter Indikator dafür, dass agile Methoden eine Bereicherung darstellen. Dennoch zeigen die oben erwähnten Einschränkungen oder Widersprüche, dass wahrscheinlich nicht alle Methoden in der Praxis umgesetzt werden können, was im Einzelfall dazu führen kann, dass das agile Vorgehen nicht mehr sinnvoll ist.

# 4 Grenzen der agilen Prozesse

Um für eine Domäne zu entscheiden, ob es sinnvoll ist, agile Methoden und Prozesse einzusetzen, sollte man einen Blick auf die Grenzen des agilen Vorgehens im allgemeinen werfen und wie sich diese im laufe der Jahre verändert haben, oder auch nicht.

## 4.1 Kritikpunkte - Stand 2002

Nachdem 2001 das Agile Manifest veröffentlicht wurde und agile Methoden schon einige Jahre benutzt wurden, zeichneten sich immer deutlicher deren Grenzen ab.

**Eingeschränkte Unterstützung für die Vergabe von Unteraufträgen ►** Outsourcing von Entwicklungsaufgaben ist dadurch eingeschränkt, dass oft keine genauen Anforderungen und Auslieferungsgegenstände definiert werden können. Es ist trotzdem

möglich Verträge aufzusetzen, in die ein fest definierter und einen variabler Teil eingearbeitet sind. [4]

**Eingeschränkte Unterstützung für große Entwicklerteams ►** Agile Prozesse und deren Methoden sind auf kleine oder mittlere Teamgrößen abgestimmt. Bei größeren Teams verlieren viele Methoden die gewünschte Intention.

Traditional software engineering practices that emphasize documentation, change control and architecture-centric development are more applicable here. This is not to say that agile practices are not applicable in such environments. There may be opportunities for teams to use agile practices, but the degree of agility possible may be less than that found in smaller projects.[TFR02]

**Eingeschränkte Unterstützung für verteilte Entwicklungsumgebungen ►** Persönliche Kommunikation ist fester Bestandteil agiler Methoden und sollte ständig stattfinden können. In Fällen, wo der Kunde und das Team oder sogar einzelne Teammitglieder räumlich voneinander getrennt sind, kann dies zu ernsten Problemen führen. In such cases, one can at least approximate face-to-face communication using technologies such as videoconferencing, but these technologies are expensive and not as effective as one would hope.[TFR02]

**Eingeschränkte Unterstützung für das schaffen von wieder benutzbaren Artefakten ►** Agile Methoden sind darauf ausgerichtet Software in kurzer Zeit zu produzieren für ein spezifisches Problem. Dabei steht die Entwicklung von wiederverwertbaren Artefakten nicht im Vordergrund, auch wenn diese sinnvoll wäre.

**Eingeschränkte Unterstützung für die Entwicklung sicherheitskritischer Software ►** Fehler in sicherheitskritischer Software kann zu Verletzung von Personen oder ökonomischen Schäden führen. Die von agilen Methoden allein zur Verfügung gestellten Mechanismen zur Qualitätskontrolle können nicht sicher stellen, dass das Endprodukt sicher ist.

Formal specification, rigorous test coverage, and other formal analysis and evaluation techniques included in software engineering approaches provide better, but also more expensive, mechanisms to tackle the development of safety- or business-critical software. [TFR02]

[4] Vergl. hierzu [TFR02]

Trotzdem können Entwickler sicherheitskritischer Software von agilen Methoden profitieren. Zum Beispiel können Unit-Tests im Test Driven Development Ansatz sicher stellen, das Vorgaben definiert und geprüft werden können, bevor Code geschrieben wird und das diese Vorgaben auch während der Entwicklung jeder Zeit erneut geprüft werden können.

Therefore, it can be assumed that agile and formal software development are not incompatible, but can be combined when needed: Formal techniques may be used in an agile way to handle critical pieces of the software to increase quality and confidence. [TFR02]

**Eingeschränkte Unterstützung für die Entwicklung komplexer Software ►** The assumption that code refactoring removes the need to design for change may not hold for large complex systems in particular. In such software, there may be critical architectural aspects that are difficult to change because of the critical role they play in the core services offered by the system. In such cases, the cost of changing these aspects can be very high and therefore it pays to make extra efforts to anticipate such changes early. [TFR02]

Andere Ansätze wie Model Driven Architecture sind hier besser geeignet.

## 4.2 Kritikpunkte - Stand 2009

**Agil entworfen für Stars und von Stars ►** Agile has been designed by experienced, smart, and high-achieving people. [Col09]

Und so wurden diese Methoden auch in erster Linie für erfahrene und hoch produktive Teams geschrieben. Somit ist davon auszugehen, dass nicht jede Gruppe von Individuen erfahren und motiviert genug ist ein selbst-organisiertes effektives Team zu bilden. Gerade in einer wissenschaftlichen Domäne, in der selten ausgebildete Softwareingenieure Teil des Entwicklungsprozesses sind, kann dies zu enttäuschenden Ergebnissen führen.

**Agil passt nicht zur Unternehmenskultur ►** Enabling agile behavior requires a great dose of individual and team freedom. It translates into cross-functional, constantly adapting work, and switching roles as needed. It also entails adjusting processes continuously to reflect the current situation. More than anything, it means that processes are secondary to people.[Col09]

Traditionelle Strukturen können so eingefahren sein, dass eine hinreichende Änderung, die es möglich

macht agile Methoden einzusetzen, nicht zu erreichen ist. Dennoch ist es möglich auch unter solchen Strukturen agil zu arbeiten, wenn es möglich ist einzelne kleine Gruppen aus diesen Strukturen herauszulösen und ihnen die Freiheiten zu geben, die nötig sind um agiles Vorgehen umzusetzen. Als Beispiel hierfür kann eine Entwicklungsabteilung in einem traditionell organisierten unternehmen gelten.

**Teamleistung oder Eigenleistung ►** Agile Methoden sind auf Teamwork zugeschnitten. In einer Umgebung, die individuelle Leistung belohnt führt dies zu unerwünschtem Verhalten.

If we rely exclusively on individual accountability, we tend to generate selfish behavior that can affect teamwork. If we rely exclusively on team assessment, we overlook that individuals perform differently in a given team, creating opportunities for underperforming team members to get away with it and lessening incentives to perform in a superior way. [Col09]

Beide Aspekte müssen also in Betracht gezogen werden. Eine Möglichkeiten hierzu ist ein zweistufiges Belohnungssystem, was auf höherer Ebene das Team belohnt und diese innerhalb des Teams auf die einzelnen Personen, je nach individueller Leistung verteilt wird.

**Fast kein Fokus auf Methodik ►** Software development methodologies include several processes, such as analysis, architecture, implementation, project management, configuration management, and so on. However, most agile methodologies, such as Scrum for example, do not define processes. In particular, most agile methodologies do not define any project management processes. [Col09]

Egal, ob man ein agiles Rahmenwerk benutzt oder nicht, müssen Prozesse wie Budget, Risiken oder auch Änderungsmanagement in einer geeigneten Form adressiert werden können.

**Eingeschränkte Unterstützung für große Entwicklerteams ►** Sieben Jahre später scheint dies immer noch ein ungelöstes Problem im Agilen Rahmenwerk zu sein.

Agile Teams sind in der Größe nach oben und nach unten begrenzt. Als optimale Größen gelten Teams mit 3,5,7 oder 9 Mitgliedern. Das sich diese Größen bewährt haben, hat verschiedene Gründe, die in der Regel schlecht mit noch größeren Teams zu vereinbaren sind.

Das Team muss in der Lage sein sich selbst zu organisieren, jedes Teammitglied sollte einen Gesamtüberblick haben, die Teammitglieder und wichtige Stakeholder arbeiten täglich am gleichen Ort miteinander und stehen so jeder Zeit auch für spontane Kommunikation zur Verfügung. Je größer ein Team wird, desto schwieriger ist es all dies sinnvoll umzusetzen.

**Eingeschränkte Unterstützung für verteilte Entwicklungsumgebungen ►** Auch dieser Punkt wird sieben Jahre später immer noch bemängelt.

## 4.3 Kritikpunkte - Stand 2015

**Nichtverfügbarkeit des Kunden ►** The value "Continuous collaboration with the Customer" [agilmanifest2015] considers the customer's constant presence as essential to explain and detail the requirements, validate and test the application. However, in practice what is observed is that the customer's continuous presence is an exception, not the rule. [MVS15]

Dies kann zu Verzögerungen im Projektablauf führen, zu nicht oder falsch getroffenen Entscheidungen die im Schlimmsten Fall einen Projektabbruch mit sich ziehen.

**Teilweise Unterstützung für große Entwicklerteams ►** 2013 wurden die Large-Scale-Srum Framework Regeln aufgestellt. LeSS ist Scrum, welches auf mehrere Teams erweitert wurde. Ob dies sinnvoll ist, ist immer noch Diskussionsgegenstand, aber es zeigt, dass es in diesem Bereich Forschung und Ergebnisse gibt.

**Teilweise Unterstützung für verteilte Entwicklungsumgebungen ►** Da die Praxis zeigt, dass die meisten Entwicklerteams mehr oder weniger verteilt sind werden immer mehr Anpassungen an den Rahmenwerken vorgenommen um diesen Fakt zu bewältigen. Dabei bleibt weiterhin die Frage der sinnvollen Kommunikation zwischen allen Beteiligten die größte Herausforderung.

**Unzureichende Erfassung von Anforderungen ►** Requirements Engineering (RE) provides the appropriate mechanism to understand what the customer wants, analyzing the needs, verifying the feasibility, negotiating solutions, specifying the unambiguous and managing their changes. Despite the importance of RE in the success of the development of the software and minimization of project risks, this activity is seen in agile methods as bureaucratic, which makes the process less agile. [...] The quality of the

SRS (Software Requirements Specification) directly affects the success of software development. [MVS15]

Nicht in jedem Projekt kann man auf RE verzichten, nur um agile Methoden wie empfohlen einzusetzen. Ein Blick auf die Praxis zeigt aber, das eine Mischung von traditionellen und agilen Methoden nicht zwangsläufig zu schlechteren Ergebnissen führt.

**Mitarbeiterwechsel in Teams** ► Turnover Team: software development companies have one of the largest professional turnover rates [Tec]. So knowledge of business rules, data model, architecture and other important information should not be exclusively in people's minds. It is necessary to document and share the knowledge that adds value to someone else. This prevents people to interrupt their activities to pass on knowledge to others. [MVS15]

Vor allem bei Projekten mit langer Laufzeit kann dieser Fakt zu ernsten Problemen führen und führt immer zu einem Verlust von Team-Geschwindigkeit. Das agile Vorgehen verbietet nicht sämtliche Dokumentation. Insofern kann erhöhter Dokumentationsbedarf, da in so einem Fall nötig, mit in Betracht gezogen werden.

**Softwarewartung** ► Software Maintenance: Before proceeding with changes in software, it is necessary to understand the current functionality so that you can assess the impact and make the appropriate changes in the source code. A poor quality documentation compromises the execution of those activities; [MVS15]

**Zusammenfassung** ► Zusammenfassend gibt einige Problem- bzw. Kritikpunkte am agilen Vorgehen, die sich über die Jahre nicht verändert haben.

## 4.4 Nachteile des agilen Modells

**Erforderlichen Aufwand schätzen** ► In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle. [Cer15]

In der Wissenschaft und bei explorativer Softwareentwicklung ist dieser Fakt nicht zwingend ein Nachteil, solange sich alle Beteiligten der Konsequenzen bewusst sind. Zudem ist es immer möglich eine grobe Planung bei Bedarf weiter zu verfeinern, bis für alle Beteiligten ein vertretbarer Stand erreicht ist.

**Design und Dokumentation** ► There is lack of emphasis on necessary designing and documentation. [Cer15]

Dies kann auch im wissenschaftlichen Bereich zu Fehlentwicklungen führen. Proffesionelle Unterstützung kann hierbei von Nöten sein.

**Mangel an Focus** ► The project can easily get taken off track if the customer representative is not clear what final outcome that they want. [Cer15]

Wie bei allen explorativen oder wissenschaftlichen Projekten muss ein Scheitern in Betracht gezogen werden. Allerdings liefert auch ein gescheitertes Projekt ein Ergebnis ab, nämlich das ein falscher Weg beschritten wurde.

**Mangel an Fachkenntnissen** ► Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources. [Cer15]

Wenn man den Anspruch hat das agile Modell in allen Einzelheiten umzusetzen braucht man eine Mischung aus erfahrenen Leuten, die den unerfahrenen zur Seite stehen und kommt selten ohne externe Hilfe aus. Die Praxis zeigt aber das auch schon einzeln eingesetzte agile Methoden eine Bereicherung für ein Projekt darstellen.

## 4.5 Anwendbarkeit des agilen Modells

**Änderungen begrüßen** ► When new changes are needed to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced. [Cer15]

Änderungen und Unschärfen sind ein Hauptmotivationspunkt um agile Methoden einzusetzen. Bei Projekten mit hoher Änderungswahrscheinlichkeit sind diese sogar zu bevorzugen.

**Neue Funktionen** ► To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it. Changes can be discussed and features can be newly effected or removed based on feedback. [Cer15]

Das iterative und inkrementelle Vorgehen macht roll backs und Umstrukturierungen erst möglich.

**Schneller Start** ► Unlike the waterfall model in agile model very limited planning is required to get started with the project. [Cer15]

So ist es auch möglich ein agiles, exploratives Projekt zu starten um mit den schnell erzielten Ergebnissen den eigentlichen Planungsprozess zu starten.

**Optionen freihalten ►** Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill. [Cer15]

Dieser Fakt ist besonders nützlich in dem Fall, wo genaue Rahmenbedingungen, Ergebnisse oder ähnliches, noch nicht feststehen, aber das Projekt auf den Weg gebracht werden soll.

**Weniger Dokumentation ist akzeptierbar ►** The agile methodology is not setting a lot of focus on design and documentation, these can't be critical in the project. [Cer15]

Obwohl agiles Vorgehen Dokumentation und Design nicht verbietet, kann es in diesem Fall sein das man nur ausgewählte agile Methoden benutzen kann.

# 5 SCRUM

Scrum ist ein Rahmenwerk für iterative, inkrementelle agile Softwareentwicklung. Entwickelt wurde Scrum Anfang der neunziger Jahre. Scrum wurde Anfang des einundzwanzigsten Jahrhunderts immer populärer und gilt als eines der wichtigsten Rahmenwerke für agiles Vorgehen. Somit kann Scrum als Beispiel dienen um zu verifizieren, dass sich ein solches agiles Rahmenwerk auch für die CSE Domäne eignet.

## 5.1 Wie unterstützen die Scrum Techniken und Methoden die CSE Entwicklung

Laut Blom sind alle Scrum Techniken und Methoden, bis auf die Rolle des Product Owners relevant für die CSE Domäne (Siehe Tabelle: SCRUM). Ob diese Einschätzung richtig ist werde ich in einem tiefer gehend Einblick erörtern.

## 5.2 Product Owner

Bolm hält die Rolle des Product Owners für weniger relevant für die CSE Domäne, da „In CSE there might not be an external customer at all and this is hence deemed as less relevant. In projects without an external product owner, the team itself needs to act as the product owner, for instance when prioritizing features or testing the software. This requires a change of viewpoint from developer to customer, which has been difficult in some cases.“ [Blo12]

**Tabelle 2:** *SCRUM [Blo12]*

| | Product Owner |
|---|---|
| (•) | Team |
| • | Scrum Master |
| •• | Sprints |
| • | Backlogs |
| • | Burndown Chart |
| • | Sprint Planning Meeting |
| • | Sprint Review |

less relevant: (no dot) | relevant: • | extra relevant: ••

Diese Einschätzung ist meines Erachtens nach falsch, da der Product Owner eine wichtige Rolle darstellt und dessen Funktion nicht, wie Blom auch erahnt, von einem Teammitglied übernommen werden sollte.

Der ProductOwner (bzw. der Kunde selbst) ist einzige Schnittstelle im Zusammenspiel von Kunde und Team. Der ProductOwner gibt der Software Entwicklung eine Richtung. Hierfür ist es wichtig, das der ProductOwner nicht Teil des Entwicklerteams ist.

A common mistake is to have the product owner role filled by someone from the development team [Rot14]

Rothman ist der Meinung wenn das Team nur eigenverantwortlich arbeitet, bricht das agile Ecosystem zusammen. Das Vorgehen bleibt iterativ und inkremental, aber ist nicht mehr agil!

When the business is unaccountable, the agile ecosystem breaks down. [...] It's iterative and incremental, but it's not even close to Scrum. It's not agile. [Rot14]

## 5.3 Team

It is argued that Scrum works best for teams of 5-7 people, but most of the techniques can be used for single developers as well.[Blo12]

Die Techniken kann man zwar anwenden aber der „Spirit of Scrum“ wird damit verraten. Scrum lebt vom Team und Kommunikation und Austausch sind zentrale Elemente von Scrum. Als Alleinentwickler benutzt man also agile Methoden, arbeitet aber nicht agil. „but for larger CSE projects, teams would be as relevant as for non-CSE projects.“[Blo12] Wobei man beachten sollte, dass die empfohlenen Teamgrößen

3,5,7 oder 9 Mitglieder sind, also immer eine ungerade Anzahl an Mitgliedern haben, damit Entscheidungen nie in einer Patt-Situation enden können.

### 5.4 Scrum Master

Any team running Scrum would need a scrum master and CSE teams would be no different.[Blo12]

Diese Aussage ist richtig, führt aber in sofern zu Komplikationen, da Blom davon ausgeht das viele Entwickler in der CSE Domäne entweder alleine arbeiten oder es wenig Ressourcen gibt um externe Mitarbeiter in das Projekt mit einzubeziehen. Another common pitfall is for a scrum master to act as a contributor [Ber14]

Scum postuliert, dass auch der Scrum Master nicht Teil des Entwicklerteams sein sollte, verbietet dies allerdings nicht. Allerdings sollte die Rolle des Scrum Masters zuerst ausgefüllt werden und die Person sollte nicht an Kritischen Komponenten des Systems arbeiten.

### 5.5 Sprints

Sprints sind laut Blom extra relevant. Sprits eignen sich auch hervorragend für explorative Projekte, bei denen die genauen Spezifikationen zu Beginn noch nicht feststehen.

Working in sprints, i.e. in iterations where a small part of the system is finished in each sprint, would suit a CSE developer well since the exact specification of the system might not be available at first. The more uncertainty there is, the more difficult it is to specify a complete requirements list directly.[Blo12]

Sprints bzw inkrementelle Entwicklung sind der Grundstein für die Möglichkeit Änderungen oder auch neue Funktionen umzusetzen.

## 6 Extreme Programming (XP)

Extreme Programming (XP) wird oft auch als „Scrum Kompakt“ bezeichnet. Es gibt also viele Gemeinsamkeiten zwischen Scrum und XP. Dabei arbeiten alle Aspekte von XP zusammen als ein ganzes. Sobald auch nur ein Aspekt vernachlässigt wird, kann dies den gesamten Prozess beeinträchtigen. Ein sehr ineinandergreifendes System ist nur so stark wie das schwächste Glied.

**Tabelle 3:** *Extreme Programming XP[Blo12]*

| | |
|---|---|
| • | Pair Programming |
| • | TDD |
| •• | Incremental Design |
| • | Continuous Integration |
| • | Collective Code Ownership |
| • | Informative Workspace |
| • | Coding Standard |
| • | Sustainable Pace |

less relevant: (no dot) | relevant: • | extra relevant: ••

### 6.1 Wie unterstützen die XP Techniken und Methoden die CSE Entwicklung

All key points of XP are relevant, except Incremental Development that was graded as extra relevant. XP would in other words suit CSE development well.[Blo12]

Laut Blom ist jeder Punkt relevant für die CSE Domäne. Das das inkrementelle Design extra relevant ist wurde weiter oben im Teil „Sprints“ erörtert.

Allerdings gilt das pair programming als ëxotischeMMethode. Leute sind schwer davon zu überzeugen und somit wird es selten genutzt.

In practical examples pair programming is seldom used. [SHPL12]

Schaut man sich die Praxis an, funktioniert XP auch ohne das pair proogramming ständig eingesetzt wird. Somit würde ich es als eine wünschenswerte Methode bezeichnen

## 7 Praxisbeispiele

### Studie: What Do We Know about Scientific Software Development's Agile Practices?

In der Studie “What Do We Know about Scientific Software Development's Agile Practices?” [SHPL12] wurde versucht herauszufinden, in wie weit agile Methoden in der Softwareentwicklung von Forschern und für Forscher benutzt werden.

Hierzu wurde eine Literatur Recherche mehrerer Literaturdatenbanken, ein webbasierter Fragebogen und eine Fallstudie ausgewertet.

### 7.1 Literatur-Recherche

Obwohl 2012 über 100 Veröffentlichungen gefunden wurden, die über Softwareentwicklung in der Wissen-

schaft berichten, konnten nur 5 identifiziert werden, die über die mögliche Benutzung agiler Methoden in diesen Projekten berichten.

In jedem der Projekte wurden agile Methoden genutzt. Als Fazit ziehen die Autoren: „Thus, the literature review indicated that projects using agile practices better handle testing related activities. The review also supports the assumption that projects using agile practices are better at handling requirements activities, but the findings aren't as substantial as for testing.

## 7.2 Fallstudien-Methode

**Charakteristiken der Fallstudienobjekte** ► Als Fallstudienobjekte wurden drei große Softwareprojekte aus der Wissenschaft ausgewählt. Alles waren Langzeitprojektdauer mit einer Laufzeit über 10 bzw. über 30 Jahren, mit 10 bis 50 involvierten Personen.

Eines der Projekte benutzte explizit agile Methoden (Scrum). Nur in diesem Projekt wurde eine Vielzahl der möglichen agilen Methoden gefunden. Die von Scrum empfohlenen waren zum größten Teil auch umgesetzt.

## 7.3 Internetfragebogen

Der Internetfragebogen hatte fast 2000 Antworten von Forschern aus aller Welt. Allerdings zielte keine Frage auf agile Methoden ab. Somit ist das einzige Fazit, was man aus dieser Umfrage in Relation zu Softwareentwicklungsmethoden ziehen kann, dass das Niveau des Verständnisses dieser Methoden stark unter den Forschern variiert.

The results for RQ8 [How familiar are scientists with standard concepts of software engineering?] indicate that there is a great deal of variation in the level of understanding of standard software engineering concepts by scientists. The level of importance that scientists assign to a standard software engineering concepts is mostly consistent with their understanding of this concept. We found, however, that in particular for the concepts 'software testing' and 'software verification' scientists assign on average a higher level of importance to these concepts than they judge their level of understanding of these concepts. [HPL+09]

## 7.4 Fazit der Studie

Das Fazit der Studie ist das agile Methoden sehr wohl von Projekten genutzt werden, die wissenschaftliche Software entwickeln.

Allerdings sind für die meisten keine positiven Beweise für ihre Anwendung gefunden worden. Den einzigen Negativbeweis wurde für die exotische Methode „pair programming“ gefunden.

Die am öftesten benutzten Methoden waren: timeboxed sprints, short daily meetings, self-organizing team, release planning, user stories, dedicated open work space for team, project velocity is measured, customer is always available, integrate often, collective ownership and refactor whenever and wherever possible.

Für etwa die Hälfte der möglichen agilen Methoden ist das Bild allerdings unklar. Es wurden sowohl Projekte gefunden, die diese Methoden nutzen als auch solche, die sie nicht nutzen. Die Anzahl der Projekte war allerdings zu klein um daraus Schlüsse ziehen zu können.

# 8 Fazit

Agile Methoden kommen in immer mehr Projekten im wissenschaftlichen Bereich zur Anwendung, auch in solchen, die kein agiles Rahmenwerk nutzen.

Agil bei vielen Projekttypen und schadet zumindest nicht [Pec14]

Als besonders nützlich für den wissenschaftlichen Bereich haben sich die Methoden erwiesen, die eine explorative Entwicklung unterstützen und im Bereich Software Tests und Software Verifikation eingesetzt werden können.

Das in keinem der untersuchten Projekte alle Methoden, wie empfohlen, umgesetzt wurden, diese Projekte aber dadurch auch nicht scheitern oder nachhaltig behindert werden, zeigt, dass oft ein Mittelweg für die wissenschaftliche Softwareentwicklung der richtige ist und das Agil kein Rahmenwerk ist, dass alle vorherrschenden Probleme lösen kann.

Mit dem richtigen Projektumfeld passt agile Entwicklung gut zu der CSE Domäne bzw. der wissenschaftlichen Softwareentwicklung.

Agile Methoden werden am besten genutzt, wenn man flexibel auf Änderungen und neue Funktionen reagieren muss und sich Optionen freihalten möchte.

Agiles Vorgehen im allgemeinen ist weniger geeignet für Anfänger ohne professionelle Unterstützung.

# 9 Bibliography

## Literatur

[Ber14] Steve Berczuk. Mission possible: Scrummaster and technical contributor. http://www.agileconnection.com/, 2014.

[Blo12] Martin Blom. Is scrum and xp suitable for cse development? *Procedia Computer Science*, 2012. International Conference on Computational Science, ICCS 2010.

[Cer15] ISTQB Exam Certification. What is agile model - advantages, disadvantages and when to use it? http://istqbexamcertification.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/, 2015.

[Col09] Bruno Collet. Limitations of agile software development. http://www.brunocollet.com/blog, 2009.

[HPL+09] Jo Erskine Hannay, Dietmar Pfahl, Hans Petter Langtangen, Carolyn MacLeod, Janice Singer, and Greg Wilson. How do scientists develop and use scientific software? *ResearchGate*, 2009.

[MVS15] Juliana Medeiros, Alexandre Vasconcelos, and Carla Silva. Integration of agile practices: An approach to improve the quality of software specifications, 2015.

[Pec14] Jörg Pechau. Projektmanagement. Department Informatik, Uni Hamburg, 2014.

[Rot14] Johanna Rothman. When you have no product owner at all. http://www.jrothman.com/, 2014.

[SHPL12] Magnus Thorstein Sletholt, Jo Erskine Hannay, Dietmar Pfahl, and Hans Petter Langtangen. What do we know about scientific software development's agile practices? *Computing in Science and Engineering*, pages 24–36, 2012.

[Tec] Techrepublic. Techrepublic: Tech companies have highest turnover rate. http://www.techrepublic.com/blog/career-management/tech-companies-have-highest-turnover-rate/.

[TFR02] Dan Turk, Robert France, and Bernhard Rumpe. Limitations of agile software processes. *Third International Conference on Extreme Programming and Flexible Processes in Software Engineering*, pages 43–46, 2002.

## Tabellenverzeichnis