Jens Gallenbacher (Hrsg.): INFOS 2015 - Informatik allgemeinbildend begreifen,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 219

Ganzjähriger Projektunterricht mit agilem Framework

Ulrich Kiesmüller[1], Petra Kastl[2] und Ralf Romeike[3]

Abstract: In diesem Beitrag werden zwei jeweils achtmonatige Unterrichtsprojekte zweier 10. Klassen eines bayerischen Gymnasiums vorgestellt. Über die gesamte Zeit entwickelten Gruppen von je fünf bis neun Schülerinnen und Schülern mit der Programmiersprache Java ihr eigenes Softwareprojekt und erarbeiteten sich dabei informatische Konzepte der objektorientierten Programmierung und Modellierung. Zur Unterstützung wurden geeignete agile Praktiken ausgewählt und jeweils zeitverzögert durch weitere ergänzt. Die vorgenommene Anpassung des agilen Modells an den Kontext, die praktische Umsetzung und Beobachtungen werden im vorliegenden Beitrag beschrieben. Sie werden kontrastiert zu den Erfahrungen aus den Vorjahren, in denen nach dem Wasserfallmodell vorgegangen wurde. Abschließend werden wesentliche Erkenntnisse und Erfahrungen, die in die Weiterentwicklung des agilen Modells fließen, zusammengestellt.

Keywords: Einsatz agiler Methoden der Softwareentwicklung im Informatikunterricht, Projektunterricht

1 Objektorientierung und Softwareentwicklung im Unterricht

Grundlagen der objektorientierten Modellierung und Programmierung sind für die naturwissenschaftlich-technologische Ausbildungsrichtung an bayerischen Gymnasien im Lehrplan der 10. Jahrgangsstufe [IS03] verankert. Als Abschluss ist dort ein kleines Softwareprojekt vorgesehen, um den Lernenden zu vermitteln, dass man umfangreiche Aufgaben nur mit sorgfältig geplanter Teamarbeit, strukturiertem Vorgehen und basierend auf fachlichem Wissen lösen kann. Hierbei geben der bayerische Lehrplan und die gängigen Schulbücher dem Wasserfallmodell den Vorzug. Über diese vom Lehrplan geforderten Ziele hinaus sind mir als Lehrkraft auch die Berücksichtigung von Aspekten wie selbstständiges Arbeiten, Kreativität und kritische Reflexion [HNR07] wichtig. Außerdem soll das Projekt Sozialkompetenzen wie Kommunikation und die Fähigkeit zur Entscheidungsfindung in einer Projektgruppe sowie einen konstruktiven Umgang mit Konflikten [Gu08] fördern. Überdies möchte ich den Lernenden mit der Projektarbeit ein modernes, zeitgemäßes Bild vom Beruf eines Informatikers vermitteln [GR13].

In der praktischen Umsetzung erlebte ich wiederholt, dass Schülerinnen und Schüler am Ende der 10. Jahrgangsstufe in Projekten zwar sehr motiviert sind, aber in den zehn Unterrichtsstunden, die der Lehrplan vorsieht, keine brauchbaren oder gar spannenden Produkte gemeinsam planen, entwickeln, vorstellen und reflektieren können. Dafür ist die

[1] Simon-Marius-Gymnasium Gunzenhausen, Simon-Marius-Straße 3, 91710 Gunzenhausen, kiesmueller@simon-marius-gymnasium.de
[2] Friedrich-Alexander-Universität Erlangen-Nürnberg, Didaktik der Informatik, petra.kastl@fau.de
[3] Friedrich-Alexander-Universität Erlangen-Nürnberg, Didaktik der Informatik, ralf.romeike@fau.de

Zeit zu knapp bemessen. Deshalb habe ich in den letzten Jahren die Projektphase erheblich ausgedehnt und weitere Lehrplaninhalte integriert. Für den dazu notwendigen fachlichen Input wurde die Projektarbeit der Schülerinnen und Schüler regelmäßig meist zu Beginn jeder Doppelstunde unterbrochen. Problematisch bei diesem Vorgehen war, dass die Lernenden zu Beginn des Projekts weder über die fachlichen, planerischen und sozialen Fähigkeiten und Fertigkeiten verfügten, die ein lineares Prozessmodell wie das Wasserfallmodell voraussetzt, noch die Schritte einer strukturierten Vorgehensweise in der Softwareentwicklung kannten. Deshalb waren die Jugendlichen immer stark auf Unterstützung angewiesen. Für die Lehrkraft bedeutete es einen enormen Betreuungsaufwand, wobei die Unterstützung noch anspruchsvoller wurde, wenn jede Projektgruppe ihr eigenes Thema wählen konnten. Entsprechend langsam war der beobachtbare Projektfortschritt. Andererseits erhöhte ein eigenes Thema die Motivation und das Durchhaltevermögen der Lernenden insbesondere in der langen Planungsphase. Und so waren die Rückmeldungen der Lernenden am Ende des Schuljahres zwar großteils positiv und sie waren stolz auf ihr Produkt. Das lange Warten auf Unterstützung wurde aber regelmäßig von den meisten Projektbeteiligten bemängelt.

Im Einsatz agiler Methoden sah ich die Möglichkeit, es den Lernenden durch iteratives Vorgehen zu ermöglichen, mit vorhandenen fachlichen, sozialen sowie organisatorischen Fähigkeiten und Fertigkeiten, unterstützt durch ausgewählte agile Praktiken und Artefakte, selbstorganisiert loszulegen und ihre Kompetenzen in jeder Iteration auszubauen. Die Rolle der Lehrkraft wandelte sich hierbei vom „Fragenbeantworter“ und „Fehlersucher“ hin zu 50% Coach und 50% Beobachter (siehe Kapitel 5). Sehr gut konnte ich den fachlichen Lernfortschritt und die positive Entwicklung sozialer und organisatorischer Fähigkeiten bei den Schülerinnen und Schülern beobachten. Von zu langen Phasen des Wartens auf Unterstützung war nicht mehr die Rede. Obwohl ich kaum noch helfen musste, gaben die Lernenden in der Rückmeldung an, dass sie sich gut betreut fühlten.

2 Rahmenbedingungen

In diesem Artikel werden Erfahrungen aus zwei 10. Klassen eines naturwissenschaftlich-technologischen Gymnasiums (NTG) in Bayern vorgestellt. Die Klassen hatten beide ca. 25 Schülerinnen und Schüler und wurden jeweils in Doppelstunden unterrichtet, die im selben Computerraum mit 20 Einzelrechnern stattfanden. Die Größe des Raumes bot jeder Gruppe Platz für ihr Project Board sowie die davor stattfindenden Diskussionen und Planungen, ohne dass sich die Schülergruppen dabei gegenseitig störten. Die Vorkenntnisse im Bereich der Objektorientierung aus der 6. Jahrgangsstufe und und der Algorithmik aus der 7. Jahrgangsstufe waren bei den meisten Schülerinnen und Schülern gering. Allerdings verfügten beide Klassen über einige Projekterfahrung, weil sie in den vorangegangenen Jahren mehrere Projekte erfolgreich durchgeführt hatten. In der 10. Jahrgangsstufe gilt es wie oben angeführt die Grundlagen der objektorientierten Modellierung und Programmierung unter Verwendung von Java zu vermitteln. Als Entwicklungsumgebung wurde dabei BlueJ [KB09] verwendet. Als weitere Hilfsmittel standen den Lernenden das Buch *Java ist*

auch eine Insel [Ul14] zur Verfügung sowie die Java-Klasse ZEICHENFENSTER[4], die es erlaubt, einfache geometrische Objekte in einigen wenigen Farben graphisch darzustellen.

3 Einsatz und Anpassung eines agilen Frameworks

3.1 Agile Praktiken in der Vorbereitung des Projekts

Vor dem Projektstart werden die grundlegenden Voraussetzungen bezüglich der objektorientierten Modellierung und Programmierung sowie des Einsatzes geeigneter Werkzeuge vermittelt. Parallel dazu werden erste Elemente des agilen Modells für Projekte im Informatikunterricht (AMoPCE) [RG12] angepasst und in den Unterrichtsverlauf integriert. In diesem Kapitel beschreibe ich das Vorgehen und dahinter stehende Intentionen exemplarisch. Nach einem Theorie-Input zur Erstellung von Klassendiagrammen erhalten die Schülerinnen und Schüler eine Aufgabe, die sie in Kleingruppen kooperativ planen und modellieren. Da sie später im Projekt mit der Klasse ZEICHENFENSTER arbeiten werden, bieten sich Themen wie das Erstellen eines Szenenhintergrunds oder von Figuren für ein Spiel an, die aus einfachen geometrischen Objekten zusammengesetzt sind. Die Planung erfolgt in einem **Stand-Up-Meeting** vor dem **Project Board** der Gruppe, das hier nur eine freie Planungsfläche ist. Die Lernenden üben dabei im Stand-Up-Meeting Diskussionen effektiv und konzentriert zu führen, Absprachen zu treffen, Probleme zu identifizieren und sich auf einen gemeinsamen Plan zu einigen. Ihr Project Board ist für sie von Beginn an ein zentraler Arbeits- und Planungsbereich. In der Rolle eines geeignet angelegten Kunden, der **Kundengespräche** mit den einzelnen Gruppen führt, integrierte ich wesentliche Aspekte einer Anforderungsermittlung in die Planung. Aufgabe der Teams ist es, durch gezielte Fragen Kundenwünsche zu präzisieren, dem Kunden ihren geplanten Entwurf zu „verkaufen“ und die Ergebnisse in der Modellierung umzusetzen. Bei der anschließenden arbeitsteilig durchgeführten Implementierung wird in das Werkzeug BlueJ eingeführt, die Klasse ZEICHENFENSTER vorgestellt und zum ersten Mal Quelltext zusammengeführt. Im weiteren Verlauf werden Grundlagen der objektorientierten Programmierung und der algorithmischen Grundstrukturen in mehreren Kleinstprojekten vermittelt. Die Arbeitsaufträge werden in Form von **User Stories** (also Funktionalitäten aus Kundensicht) gestellt. Zu Beginn sind diese sehr präzise formuliert und durch Teilaufträge in Form von **Tasks** (also zu erledigende Aufgaben aus Entwicklersicht) ergänzt. Gegen Ende werden User Stories auch gezielt offener formuliert („Ich möchte, dass sich ein Ball quer über den Bildschirm bewegt.“, „Ich möchte, dass sich ein Kreis in einem Kreisring auf dem Bildschirm bewegt.“), um Spielraum für eigene Interpretationen zu schaffen und um die Lernenden die Tasks selbständig identifizieren und formulieren zu lassen. So führte ich sie schrittweise an den Perspektivwechsel von Kunden- zu Entwicklersicht heran. Die Bearbeitung der Aufträge erfolgte großteils in Gruppen und die Planung fand weiterhin in Stand-Up-Meetings vor dem Project Board statt. Für die Implementierung bilden die Schülerinnen und Schüler Paare und nehmen abwechselnd die Rolle des *Drivers*[5] und des *Navigators*[6]

[4] erstellt von M. Kölling, B. Quig und Ch. Heidrich

[5] bedient die Tastatur, schreibt den Code und denkt dabei laut

[6] behält das große Ganze im Auge, schlägt Alternativen vor und spricht Fehler an

ein. Aus didaktischer Sicht ist dieses Pair-Programming für mich interessant, weil Lernende dadurch passiv voneinander lernen und üben, ihre Codiertätigkeiten in Worte zu fassen und verschiedene Codierstile zu diskutieren. Themen und Aufgabenstellungen der Kleinstprojekte wurden so gewählt, dass User Stories, Tasks und erstellter Code für die Großprojekte angepasst und dort integriert werden konnten. Die in dieser Phase gebildeten Gruppen hatten stets wechselnde Zusammensetzungen, damit sich keine eingespielten Rollenverteilungen herausbildeten und die Lernenden unterschiedliche Herausforderungen kooperativen Arbeitens zu bewältigen lernten. Die Schülerinnen und Schüler konnten Kundengesprächstermine mit mir vereinbaren, wenn sie Unterstützung benötigten.

3.2 Das Projekt

Um den Lernenden spielerisch eine Idee von agiler Projektarbeit zu vermitteln und ihnen Mut zu machen, wird vor dem Einstieg ins Projekt ein „Warm-Up-Spiel“[7] durchgeführt. Hierbei erfahren die Schülerinnen und Schüler die Bedeutung von gemeinsamen Absprachen, Vorteile kurzer, iterativer Entwicklungsphasen (flexible Reaktion auf Änderungen, stete Verbesserung der eigenen Performanz, Sinn einer Reflexion) und die motivierende Wirkung von Zielsetzungen, die in kurzer Zeit erreichbar sind. Stimmen aus dem Kreis der Lernenden nach der Durchführung waren unter anderem: „Noch nie habe ich mich für etwas so reingehängt.“ „Hätte nicht gedacht, dass wir uns so steigern können.“ „Schade, dass wir nicht noch weiter gemacht haben, da wäre noch mehr drin gewesen.“ Für die sich bis zum Schuljahresende erstreckende Projektarbeit dürfen sich die Lernenden eigene Themen wählen. Die Gruppen bilden sich dann entsprechend der fachlichen Interessen.

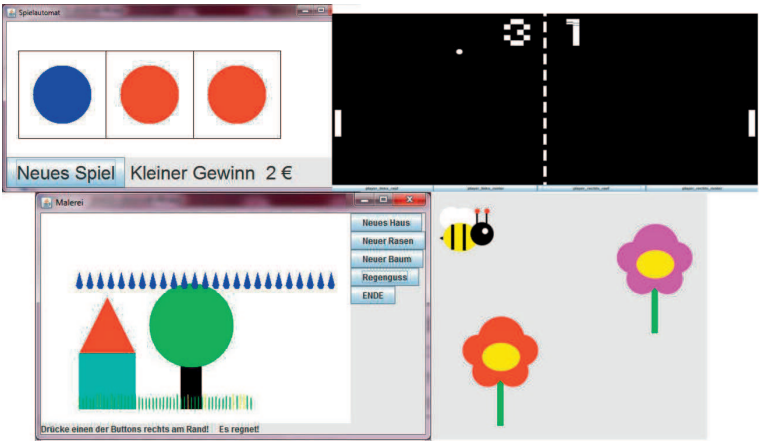

Abb. 1: Anreize zur Themenwahl: Spielautomat, Arcade-Game, bewegte Grafiken

Durch Präsentation einiger, in der zur Verfügung stehenden Zeit umsetzbarer, Beispiele (vgl. Abbildung 1) wird ihnen eine Entscheidungshilfe geboten. Nach der folgenden geheimen Abstimmung der Lernenden bezüglich eines Wunschthemengebiets ließen sich die

[7] http://borisgloger.com/scrum/materialien/tools/

Klassen jeweils problemlos in drei etwa gleich große Interessensgruppen teilen. Die themengebundene Zusammensetzung der Projektgruppen führt dazu, dass in vielen Fällen nicht nur „eingespielte Teams“ zusammenarbeiten und sich die Schülerinnen und Schüler mit für sie „neuen“ Gesprächs- und Teampartnern zurechtfinden müssen.
Die Teams treffen sich zu einer ersten Besprechung in einem Stand-Up-Meeting an ihrem Project Board, das ihnen bis Schuljahresende exklusiv zur Verfügung steht. Die Lehrkraft stößt zu jedem Team, verfolgt die Gespräche einige Zeit, schlüpft dann in die Rolle des Kunden, um die Wünsche zu kanalisieren und verdeutlicht als Berater erste erreichbare Etappenziele. Dieses Vorgehen wiederholt sich im weiteren Projektverlauf, wobei die Kontrolle über das Projekt sobald wie möglich vollständig der Gruppe überlassen wird. Anschließend greift die Lehrkraft nur noch dann ein, wenn die Schülerinnen und Schüler sich deutlich zu hohe Anforderungen stellen oder völlig falsche Wege bei der Implementierung beschreiten wollen. Die Produktfunktionalitäten, die Ziel der jeweiligen Etappe sind, werden als User Stories formuliert und in Reihenfolge ihrer Priorität mit den Namen der Bearbeitenden versehen und an das Project Board (siehe Abbildung 2) geheftet.

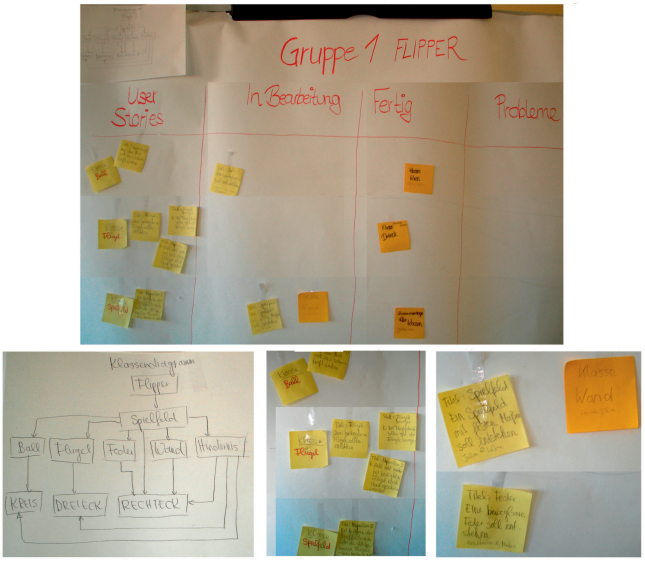

Abb. 2: Project Board „Flipper“ – Klassendiagramm – User Stories/Tasks geplant/in Bearbeitung

Dabei ergibt sich die Priorität entweder logisch aus den Funktionalitäten oder die Teammitglieder einigen sich darüber, was ihnen wichtiger und was weniger wichtig ist. Das Project Board ist ab jetzt ein Board im engeren Sinn der Softwareentwicklung, mit drei Spalten, für User Stories und dazu geplanten Tasks links, für Tasks in Bearbeitung in der Mitte und für fertige Tasks und abgeschlossene User Stories rechts. Sie bieten allen Beteiligten eine ständig aktuelle Übersicht über den Stand des Projektes. Dies gilt sowohl für die Lernenden, die sich bei auftretenden Fragen bezüglich des Verlaufs oder nächster Schritte gerne auch spontan am Board informieren, als auch für die Lehrkraft, die jederzeit einen Überblick hat, wo sich die einzelnen Gruppen aktuell im Projekt befinden und sogar, was die einzelnen Programming-Pairs gerade implementieren. Die Teams halten diese

Informationen immer aktuell. In der rechten Spalte unserer Project Boards gab es eine separate Zelle für **Probleme**. Hier heften die Schülerinnen und Schüler Zettel mit offenen Fragen und Problemen an, die sie nach der Iteration selbstständig oder mit dem Lehrer zusammen klären möchten. Bis zum Ende der Etappe ist der Ablauf nun so, dass die Teams in einem Stand-Up-Meeting in etwa die nächsten 20 Minuten planen. Die Teammitglieder wählen dazu eine oder mehrere User Stories unter Beachtung der Priorität aus, mit deren Umsetzung das Team ca. 20 Minuten beschäftigt ist und formulieren dazu Tasks. Pro Doppelstunde finden so in der Regel zwei Stand-Up-Meetings statt, auf jeden Fall aber eines zu Stundenbeginn. Die geplanten Tasks werden arbeitsteilig und im Pair-Programming implementiert. Das Project Board halten die Schülerinnen und Schüler dabei aktuell. Einen Task, den ein Pair in Bearbeitung nimmt, versieht es vor dem Umhängen mit Namen, damit die Gruppenmitglieder sich bei Fragen gezielt an das Pair wenden können. Auch die Lehrkraft kann, nach Studieren der in Bearbeitung befindlichen Tasks, die entsprechenden Schülerinnen und Schüler direkt ansprechen und sich den Arbeitsstand zeigen lassen. Treten während der arbeitsteiligen Phase dringend durch das ganze Team zu klärende Fragen oder Probleme auf, rufen die Schülerinnen und Schüler ein spontanes Stand-Up-Meeting aus. Gegen Ende der Doppelstunde wird der Code eines Teams zusammengeführt, getestet und falls nötig korrigiert. Ziel ist es, nach jeder Doppelstunde eine getestete und lauffähige Programmversion zu haben. Die nächste Doppelstunde beginnt dann wieder mit einem Stand-Up-Meeting, in dem die Teams erledigte Aufgaben der letzten Stunde rekapitulieren, eventuelle Probleme ansprechen und die nächste Arbeitsphase planen.
In diesen Ablauf wurde der Lerninhalt der 10. Jahrgangsstufe integriert. In den bisherigen Durchläufen zeigte sich, dass die einzelnen Teams trotz unterschiedlicher Themen und individuellem Arbeitstempo oft nahezu zeitgleich an bestimmte Problemstellungen gelangen, z. B. „wir bräuchten etwas, um viele gleichartige Dinge auf einmal anzusprechen“. In diesen Fällen bietet sich eine zentrale Theorie-Input-Phase durch die Lehrkraft an – im genannten Fall zum Thema „(eindimensionale) Felder“. Kommt eine Gruppe deutlich früher zu einer Problemstellung, wird ihr individuell weiter geholfen, um möglichst raschen Projektfortschritt zu gewährleisten. Themen, auf welche die Lernenden nicht von selbst stoßen, können in Kundengesprächen gezielt angeschnitten werden. Beispielsweise wird ein Kundenwunsch nach einer GUI, also einfachen Buttons und Labels mit Begeisterung aufgenommen und mit Eifer angegangen. Hierzu vermittelt die Lehrkraft die Theorie und stellt auch entsprechende Codefragmente zur Verfügung.

4 Beobachtungen und Erfahrungen

Wie in der Einleitung erwähnt, veränderte sich meine Rolle hin zum Coach und Beobachter, da die Schülerinnen und Schüler von Anfang an ihre Projektarbeit selbstständig organisierten. Zu Beginn war dies nur mit Abstrichen „zielorientiert und gründlich“, aber sie behalfen sich, wenn nötig, mit selbst einberufenen spontanen Stand-Up-Meetings und lernten rasch, worauf es bei einer Planung ankommt. Ich griff in der Anfangsphase häufiger als Kunde und/oder Berater unterstützend ein. Später konnte ich als Beobachter durch die starke Betonung der interaktiven Elemente bei den agilen Methoden beispielsweise auch soziale Kompetenzen und deren Entwicklung sehen. Defizite im Bereich der Kommunikationsfähigkeit einzelner Lernender hätte ich früher mit großer Wahrscheinlichkeit nicht

bemerkt, da ich zwar ständig bei einzelnen Teams war, jedoch jeweils nur für kurze Zeit und nur um Fragen zu beantworten. Jetzt konnte ich solche Defizite bemerken und die Entwicklung der Lernenden verfolgen. Dadurch wird die Bewertung und Einschätzung der Einzelleistungen auf einer wesentlich solideren Basis möglich und eine Rückmeldung an die Lernenden über die rein fachlichen Kriterien hinaus möglich.

Einige Tätigkeiten und Prinzipien der Softwareentwicklung, die sonst nur mit Mühe vermittelt werden können, werden von den Schülerinnen und Schülern bei der agilen, iterativen Vorgehensweise als sinnvoll und hilfreich erkannt – denn sie bedürfen keiner zusätzlichen Motivation durch die Lehrkraft. Klassendiagramme beispielsweise erstellen die Teams relativ früh freiwillig und erweitern sie, weil sie ihnen bei der Planung späterer Iterationen, bei der Implementierung sowie beim Testen helfen. Wie wichtig exakte Absprachen im Schnittstellenbereich wie z. B. Namenskonventionen für *Getter* und *Setter* sind, erkennen sie durch die arbeitsteilige Vorgehensweise im Team rasch. Sonst treten beim Testen nach der Zusammenführung Fehler auf oder es muss nachgefragt werden. „Wie heißt dein Getter“ ist zwar schnell durch den Raum gerufen, aber effektiver und ungestörter arbeiten lässt es sich mit verbindlichen Absprachen. Mit zunehmender Programmkomplexität müssen die Lernenden immer häufiger „fremden“ Code lesen und verstehen, um ihn erweitern zu können. Je besser der Code kommentiert ist, umso leichter und schneller geht das. Auch Codefragmente, die Schülerinnen und Schüler sich selbst z. B. mit Hilfe des Openbooks [Ul14] hart erarbeitet haben und die sie gerne voller Stolz weiter geben, führen bei guter Kommentierung zu weniger Nachfragen. Tests am Ende jeder Doppelstunde werden gerne durchgeführt, weil man sehen will, was die anderen Teammitglieder implementiert haben und weil man wieder ein fehlerfreies, lauffähiges Produkt haben möchte. Andere agile Praktiken werden angenommen, aber bis zu ihrer endgültigen Beherrschung benötigen die Lernenden einige Zeit. Die Erstellung kompakter User Stories, die nur aus wenigen einzelnen Tasks bestehen, sowie die Formulierung konkreter, innerhalb einer Arbeitsphase zu bewältigender Tasks, gestaltet sich für die Schülerinnen und Schüler anfangs extrem schwierig. Allerdings helfen theoretische Erläuterungen und Praxisbeispiele hier wenig; erfahrungsgemäßwird diese Technik am Besten durch das Sammeln eigener Erfahrung erlernt: Als zeitversetzt nach einigen Wochen das Planning Poker als Möglichkeit zur Aufwandsabschätzung vorgestellt wurde, hatten die Lernenden bereits ein gutes Gespür für die passende Größe von User Stories und Tasks entwickelt. Die Technik der Aufwandsabschätzung hatte für sie keinen Mehrwert und wurde deshalb verworfen. Ähnliches zeigt sich bezüglich des Planens einer Arbeitsphase. Unabhängig davon, dass mit 20 Minuten ein für Neulinge im Bereich des Programmierens überschaubares Zeitfenster gewählt wurde, gelingt vielen Schülerinnen und Schülern zunächst keine saubere Planung. Stattdessen berufen sie bis zu fünf weitere Stand-Up-Meetings ein, um noch nicht bedachte Probleme zu klären. Nach ein, zwei Doppelstunden jedoch gelingt es den Lernenden problemloser die Arbeitsphase sauber zu planen. Wenn einzelne Teammitglieder vor anderen ihre Aufgaben umgesetzt haben, es aber keine weiteren in der verbleibenden Zeit umsetzbaren Tasks gibt, verbessern die Betroffenen z. B. die Inline-Dokumentation oder testen einzelne Klassen oder das gesamte Projekt.

Etwas schwierig ist es, einen regelmäßigen Wechsel der Rollen beim Pair-Programming zu erreichen. In der Regel übernimmt die besser programmierende Person die Rolle des *Dri-*

vers und die schwächere die Rolle des *Navigators*, obwohl die Lernenden den Sinn eines Rollenwechsels einsehen. Um den Wechsel zu motivieren, wurde von mir nach etwa sechs Wochen der **Truck Factor** eingeführt und ausgerufen. In der Softwaretechnik wird vom Truck Factor gesprochen, wenn es darum geht, dass jeder Mitarbeitende bei spontanem Ausfall eines Teammitglieds vollständig über dessen Aufgaben und Arbeitsstand informiert ist und somit jederzeit die Bearbeitung übernehmen kann. Auch bei schulischen Projekten ist dieser Aspekt nicht von der Hand zu weisen. Immer wieder erkranken Lernende und ihre jeweiligen Partner können dann nicht weiterarbeiten, was zu Verzögerungen in der Projektentwicklung führt. Um die Anforderung zu prüfen, ob sich wirklich jeweils beide Partner eines Programmierteams auf dem gleichen Informationsstand befinden, kann man als Lehrkraft zu einem nicht genauer angekündigten Zeitpunkt den Truck Factor „ausrufen“. Dann wird das angesprochene Paar an zwei verschiedene Rechner gesetzt, die jeweiligen Dateien werden kopiert und beide Partner müssen getrennt voneinander weiterarbeiten. Die Lehrkraft kann sich hierbei von jedem einzelnen erläutern lassen, was seine aktuell zu bewältigenden Aufgaben sind. Anfangs standen die Schülerinnen und Schüler dieser Unterrichtsmethode skeptisch gegenüber, entwickelten aber sehr schnell den Ehrgeiz zu zeigen, dass auch nach Trennung eines Paars beide Beteiligten ohne Nachfragen weiterarbeiten können.

5 Kritischer Vergleich mit klassischen Methoden

Abschließend werden nun die Erfahrungen und Beobachtungen der drei Durchgänge mit projektbasiertem Unterricht kontrastiert mit denjenigen der Vorjahre, in denen ohne Großprojekt arbeitsgleich oder mit vorgegebenem bzw. individuellem Thema, aber nur in Zweiergruppen gearbeitet wurde.

5.1 Sicht der Lehrkraft

Arbeitsgleiches Vorgehen besitzt für die Lehrkraft den Vorteil langfristig möglicher Vorplanung. Der ständig gleiche Unterrichtsverlauf kann aber zur Abstumpfung hinsichtlich der Bedürfnisse und Probleme der Lernenden führen. Außerdem können permanente Wiederholungen demotivierende Effekte hervorrufen. Egal ob diese Methode mit Großgruppen oder mit Programming Pairs durchgeführt wird, ist der Betreuungsaufwand für die Lehrkraft nicht allzu hoch – zumindest ist er kalkulierbar. Individuelle Projekte in Zweiergruppen hingegen fordern die Lehrkraft sowohl in sportlicher als auch informatischer Sicht. Es ist nicht voraussehbar, welches Team zu welchem Zeitpunkt welche Problemstelle erreichen wird. Die Lehrkraft betreibt oft „Turnschuhdidaktik“, indem sie rastlos von einer Gruppe zur nächsten eilt und sich in kürzester Zeit in den jeweiligen Programmcode einlesen und Problemlösungen mit den Lernenden erarbeiten muss. Zusätzlich muss sie noch dafür sorgen, dass alle Theorieinhalte vermittelt werden, was oft dazu führt, dass die Schülerinnen und Schüler „auf Vorrat“ lernen müssen, da sie zum Zeitpunkt der Besprechung gewisser Inhalte diese in keiner Weise in ihrem Projekt benötigen und einsetzen können. Mit Hilfe agiler Methoden lassen sich diese Probleme vermeiden. Die individuelle Projektgestaltung führt zu einer interessanten und abwechslungsreichen, aber auf

Grund der geringen Themenzahl nicht ausufernden Arbeit für die Lehrkraft. Rasch sichtbare Ergebnisse und Anerkennung der Leistungen über die Gruppe hinaus stärken das Selbstbewusstsein der Lernenden und ihr Vertrauen in die eigenen Fähigkeiten. Sie entwickeln den Ehrgeiz, selbstständig Lösungen für ihre Aufgaben zu finden und zeigen dabei ein wesentlich höheres Durchhaltevermögen und deutlich mehr Leistungsbereitschaft als ohne agile Methoden. Auch der Betreuungsaufwand hält sich in Grenzen und ist durch die Verwendung der Project Boards gut planbar. Mit den selbst gewählten Projektthemen erreichen die Lernenden sehr schnell komplexe Fragestellungen wie zum Beispiel: „Wie kann ich eigene Farben definieren?“. Erstens ist es für die Lehrkraft zeitlich nicht möglich, all diese Problemstellungen gleich nach deren Auftreten für die Teams zu lösen. Zweitens wird auch bei Softwareentwicklungs-Großprojekten nicht jede Codezeile direkt erstellt, sondern es werden vielmehr bereits bekannte Teillösungen angepasst und integriert. Diese Technik, mit der die Lernenden bereits im Zusammenhang mit der Java-Klasse ZEICHENFENSTER bei der Erstellung kurzer Programme in Kontakt kamen, wird nun weiter vertieft. Je nach den technischen Gegebenheiten erhalten die Lernenden die Erlaubnis die Online-Version des Openbooks *Java ist auch eine Insel* von Christian Ullenboom [Ull14] zu verwenden oder es wird ihnen die entsprechende Offline-Version zur Verfügung gestellt. Sie dürfen das Buch nicht nur als Informationsquelle verwenden, sondern sich auch bei den Code-Beispielen bedienen, die sie für ihre eigenen Projekte entsprechend anpassen müssen. Die Theorieeinheiten sind zwar hinsichtlich ihres Zeitpunktes nicht langfristig zu terminieren, lassen sich aber gut über das Schuljahr verteilen, so dass auch kleine schriftliche Leistungserhebungen durchgeführt werden können. Die Beurteilung der individuellen Leistungen gestaltet sich, unterstützt durch die Eintragungen an den Project Boards und den Beobachtungen bei Kundengesprächen oder Stand-Up-Meetings, einfacher als bei den zu Beginn des Abschnitts beschriebenen Vorgehensweisen.

5.2 Sicht der Schülerinnen und Schüler

Bei der Projektarbeit in Zweiergruppen bei „klassischem Vorgehen“ kritisieren die Lernenden am meisten die langen Wartezeiten auf Betreuung durch die Lehrkraft. Sie empfinden einerseits die individuell wähl- und gestaltbaren Projektideen als sehr motivierend und insgesamt positiv, sind aber andererseits in Bezug auf den aus oben genannten Gründen sehr langsamen Projektfortschritt unzufrieden. Insbesondere ist es für die Schülerinnen und Schüler frustrierend, dass sie häufig nicht nur wochenlang kaum einen Fortschritt bei ihrer Software sehen können, sondern es bis zum Schuljahresende nicht schaffen, ein funktionsfähiges Produkt erstellt zu haben. Außerdem entstand bei einigen Lernenden der Eindruck, dass ihre individuellen Leistungsbewertung im Verhältnis zu denen ihrer Teampartner nicht gerechtfertigt sei. Auch aus Sicht der Lernenden führt der Einsatz der agilen Methoden dazu, die oben genannten negativen Aspekte zu vermeiden. Sie fühlen sich rundum betreut, sehen selbst an den nach jeder Iteration lauffähigen Programmversionen einen permanenten Projektfortschritt und empfinden die Bewertung ihrer individuellen Leistung durchweg als nachvollziehbar und korrekt. Außerdem meldeten viele Lernende zurück, dass sie durch diese Methode auch „etwas fürs Leben“ gelernt haben.

6 Fazit und Ausblick

Der Einsatz eines agilen Frameworks bei der Gestaltung des Informatikunterrichts mittels eines ganzjährigen Projekts stellt sich aus Sicht aller Beteiligten als positiv dar. Sowohl die Lehrkraft als auch die Lernenden arbeiteten hochmotiviert, die Schülerinnen und Schüler hatten viel Spaß– sicher mitbegründet durch häufige Erfolgserlebnisse – und erlernten vieles, was in den letzten Jahren unerreichbar schien oder gar nicht in Erwägung gezogen wurde. So erschlossen sich fast alle Beteiligten, wie man selbst Farben mit Hilfe der Hexadezimalcodierung definieren kann, einige lernten, wie man mehrere geometrische Grundbausteine zu grafischen Gesamtobjekten kombinieren kann, die dann zum Beispiel mit Farbverläufen gefüllt werden können. Es entstanden Stadtgrafikprojekte, bei denen sich die Sonne auf einer bogenförmigen Bahn über den Himmel bewegt und nachts dann die Lichter in den Fenstern der Häuser angehen. Die Spielautomatenprogramme wurden soweit ausgebaut, dass wirklich ein Hebel wie bei den „One-Arm-Bandits“ betätigt werden musste, um die Walzen zu starten, die während ihres Laufs wechselnde Symbole (verschiedene Früchte und nicht lediglich Kreise) zeigten. Bei Gewinn fielen dann auch Münzen aus dem Geldauswurfschacht. Für eine authentische Darstellung der Arcade-Games wurde für die Punkteanzeige eine 4x8-Punkt-Matrix eingesetzt.

Die Gestaltung des agilen Frameworks wird in weiteren Durchgängen weiter ausgebaut. Die Hilfsprogramme und Materialien für die Theorieeinheiten werden gesammelt und in den nächsten Jahren Informatiklehrkräften als „Werkzeugkasten“ zugänglich gemacht.

Literaturverzeichnis

[GR13] Göttel, T.; Romeike, R.: Agiler Projektunterricht in der Schulinformatik. In (Breier, N.; Stechert, P.; Wilke, T., Hrsg.): 15. GI Fachtagung Informatik und Schule, Praxisband. Kiel Computer Science 2013/3. Department of Computer Science, CAU Kiel, S. 151–158, 2013.

[Gu08] Gudjons, H.: Handlungsorientiert lehren und lernen: Schüleraktivierung, Selbsttätigkeit, Projektarbeit. 2008.

[HNR07] Hartmann, W.; Näf, M.; Reichert, R.: Informatikunterricht planen und durchführen. Physica-Verlag, 2007.

[IS03] ISB München: , Lehrplan Informatik 10 (NTG), 2003. http://www.isb-gym8-lehrplan.de/contentserv/3.1.neu/g8.de/index.php?StoryID=26435.

[KB09] Kölling, M.; Barnes, D. J.: Java lernen mit BlueJ: Eine Einführung in die objektorientierte Programmierung. Pearson Studium, 2009.

[RG12] Romeike, R.; Göttel, T.: Agile Projects in High School Computing Education: Emphasizing a Learners' Perspective. In: Proceedings of the 7th Workshop in Primary and Secondary Computing Education. WiPSCE '12, ACM, New York, NY, USA, S. 48–57, 2012.

[Ul14] Ullenboom, Ch.: , Java ist auch eine Insel: Das umfassende Handbuch (Galileo Computing), 2014. http://openbook.galileocomputing.de/javainsel11.