Landeshauptstadt
München
Referat für
Bildung und Sport

Projekt: AP - Final

Abgabe

Gruppe 2

Autoren	Kevin Kraus & Abassin Saleh
SCRUM MASTER	

Erklärung:

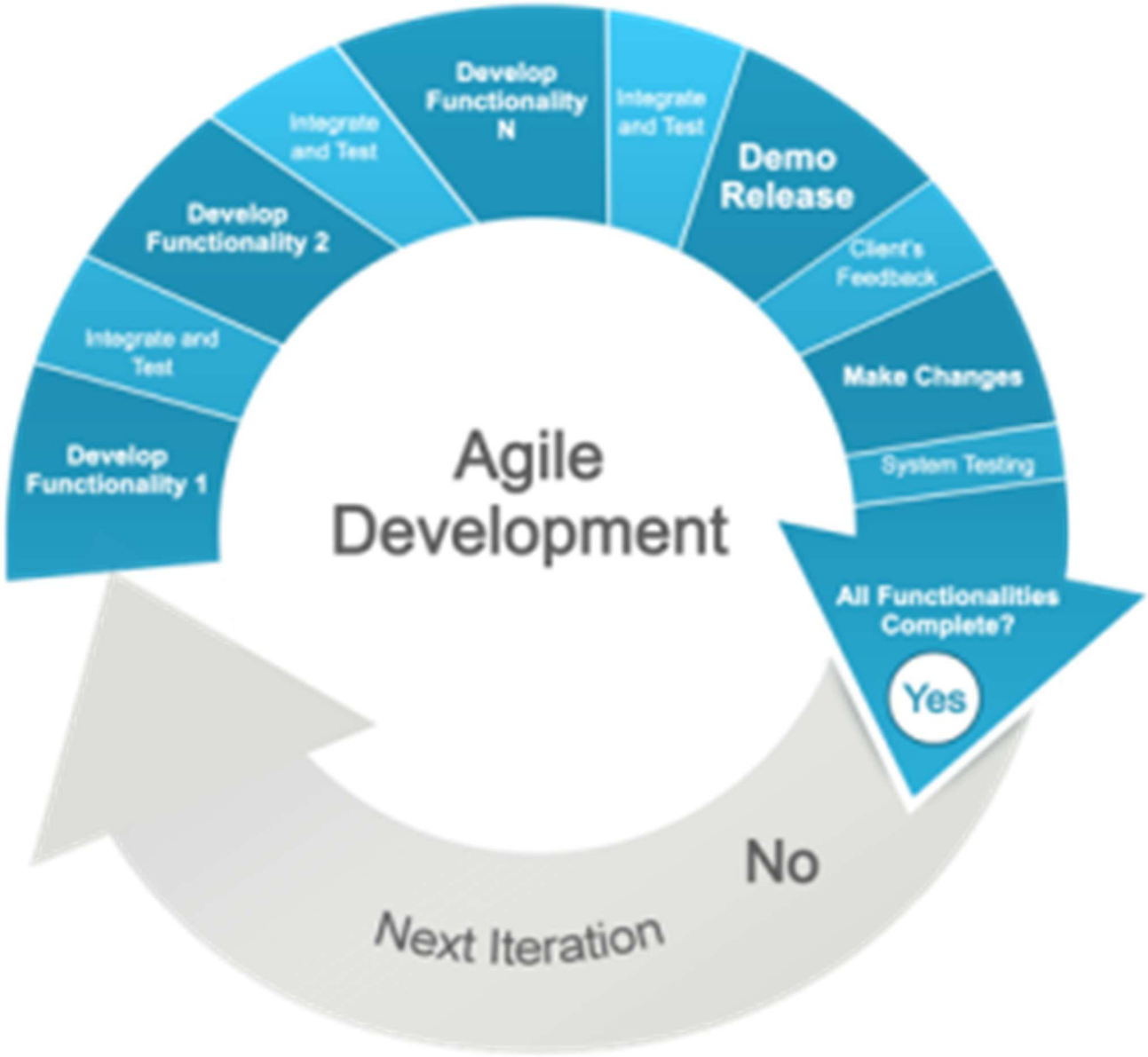
Wir versichern hiermit, dass wir unsere Projektarbeit mit dem Thema: **AP - Final** selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben. Wir versichern zudem, dass der elektronische Teil der Abgabe (JAR-Files, etc.) entweder via selbstständige Kopie auf das vorgesehene Klassenlaufwerk **oder** als USB-Stick –Kopie auf das Klassenlaufwerk durch die Lehrkraft erfolgt ist **oder** das Abgabe-ZIP mit allen Teilen im vorgesehenen MS-Teams Ordner hochgeladen wurde.

Unterschriften:

_____ _____

Anlagen: - *Scrum Protokoll*
- *umfangreiches Fazit mit kurzer Projektumschreibung (Idee, genauer Ablauf, u.v.m.)*

Städtische Berufsschule
für Informationstechnik



Autor Dietmar Reich,
(basierend auf Projekt Weinverwaltung v. M. Niedermair)
Berufsschule für Informationstechnik
http://www.bsinfo.eu/

Projektbeschreibung 1.00
Variante *JavaFX,Swing,[optional:Web/C++/C#/modern OOP-Language]*

Version 2021-03-08 (Revision: 008)

Inhaltsverzeichnis

Projektbeschreibung

1. Zielsetzung

Mit dem Projekt „AP - Final“ schließen Sie Ihre Berufsschulausbildung im Fach AP ab. Darin soll eine komplette GUI Anwendung in Java Swing (oder wahlweise einer alternativen modernen objektorientierten Programmiersprache) mit umfangreichem User-Management (User / Admin – Rechte, hinterlegter Adresse, Kontaktmöglichkeiten via Fax / Email / Skype / etc.) und passender Datenbankanbindung programmiert werden. Das „Produkt“ innerhalb der GUI-Anwendung ist dabei von vorneherein als leerer Container zu sehen und als Basisanforderung **nicht** definiert!
Das Gelingen und somit auch eine (sehr) gute Note steht und fällt mit dem erfolgreichen Umsetzen von CRUD (= *Create-Read-Update-Delete*). Dies kann mit dem geforderten User-Management vollständig abgebildet werden.
Die GUI soll dabei nach einem Login-Prozess in einem „Admin“ und einem „User“ Mode hochgefahren werden können, worauf unterschiedliche Oberflächen zur Administration oder Anwendung der Nutzer (= „nettes Bild mit Aufschrift – TODO / CONTENT“) angezeigt werden. Die Konfiguration der Benutzerkonten ist hierbei für ALLE verpflichtend umzusetzen.
Die Anwender-Sicht, d.h. ein möglicher CONTENT für die Anwendung kann mit der jeweiligen Lehrkraft „live“ besprochen und festgehalten werden.
Als Vorgehensmodell für das Projekt werden wir die agile Softwareentwicklungsmethode *Scrum*[1] verwenden (allgemeine Infos, siehe Anhang sowie *Projektablauf mit Scrum*).

Folgende Bereiche sind davon betroffen:

- Datenmodell für das Software-Projekt
- Erstellen einer **Mini-**Dokumentation (Details siehe unter Punkt 5)
- SQL-Befehle
- Erstellen von entsprechenden Oberflächen mit Swing (bzw. JavaFX) oder wahlweise alternative Frontend – (Web-) Sprachen wie beschrieben.
- Anwendung des „Design Pattern“ MVC. (Sind alternative, individuellen Frameworks bedingte Design Pattern gewünscht - bitte kurze Absprache mit Lehrkraft)
- Sourcecode – Dokumentation (z.B. Doxygen, o.ä. bzw. sauber „händisch“ mit Kurzbeschreibung) …

2. Datenmodell

Kreieren Sie ein Datenmodell, mit dem sich Ihr zu entwickelndes Programm abbilden lässt. Dabei sind folgende **Mindestvoraussetzungen** für alle Teams gleichermaßen verpflichtend:

- Das Datenmodell muss auf einer relationalen Datenbank umgesetzt werden können (*SQL-DB*)
- Das Datenmodell besteht mindestens aus 5 Tabellen. (Tipp: u.a. Tabelle *Kontakttyp*, o.ä.)
- Das Datenmodell liegt (zwingend!) in der **dritten** Normalform vor
- Das Datenmodell muss komplette Beziehungen und Kardinalitäten beinhalten (1:1, 1:n, n:m)
- Das Datenbankmodell besitzt referentielle Integrität

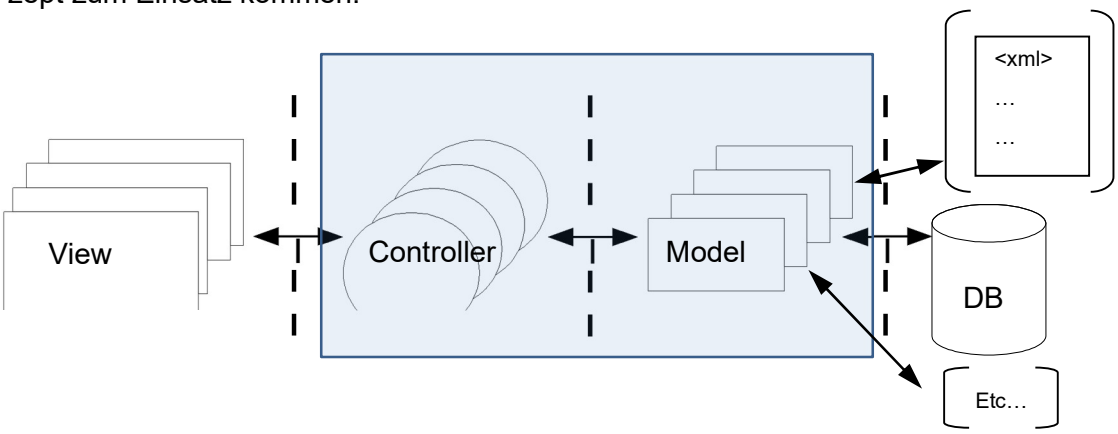
[optional / als Unterstützung für den Frontend-Entwickler]
- Die **Datenbank** könnte sinnvollerweise zwei geeignete *Views* sowie enthalten sowie bereits möglichst viele **Logikaufgaben** bereitstellen (Genutzt werden könnten z.B. *Stored Procedure*, *Stored Functions* oder ggf. *Trigger* soweit bekannt, sinnvoll und nützlich).

Hinweis: Entwickeln Sie das Datenmodell mit einem geeigneten Tool (z.B. *mySQL Workbench*), so dass Sie daraus die DDL erstellen können. Als Datenbank wird *mySQL* verwendet.

[1] siehe Anhang („*Scrum in einer Seite erklärt*“)

3. Programmmodell

Die Programmlogik muss in einzelne Bereiche getrennt werden. Hierbei soll das MVC-Konzept zum Einsatz kommen.



- Eingaben/Anzeigen im GUI-Fenster rufen Informationen von einem entsprechenden Kontroller ab. Dieser löst die entsprechende Anforderung mit eigenen Algorithmen sowie Methodenaufrufen in eine Modelklasse, die sich um die Datenablage (wo auch immer) kümmert.

  Während in unserem Fall eine mySQL Datenbank zur Datenablage dienen wird, wäre es durchaus denkbar diese durch eines Tages durch ein XML-File ersetzt werden soll. Daher ist es zwingend notwendig, dass die Model-Klassen Aufrufe in der Controller Klasse immer gleich aussehen, egal was „dahinter" geschieht…
- Die eleganteste Möglichkeit diese Unabhängigkeit zu erreichen erfolgt über die Realisierung von entsprechenden Schnittstellen (im Beispiel Java sog. Interfaces). Die Kommunikation bzw. die Aufrufe erfolgen dann mit einer konkreten Implementierung gegenüber den Schnittstellen. (Diese Umsetzung ist nicht zwingend notwendig, wird aber durch die Lehrkräfte gerne unterstützt)

Ein paar Tipp's zur Realisierung:

- Erstellen Sie ggf. für jede Grundtabelle eine eigene View, in der die Daten angezeigt, geändert und entfernt werden können.
- Erstellen Sie ggf. für bestimmte Teile Ihres Programms komplexere Views, um diese komfortabel zu bearbeiten / anzeigen zu können. Hier kann der „zeitlich unterforderte Datenbank-Schreiber" für die GUI bzw. Frontend Schreiber massiv Zeit einsparen…

[optional :]

• Sollte Ihr Programm sich als **zu wenig** aufwändig entpuppen, hier ein paar Vorschläge um es abzurunden…:

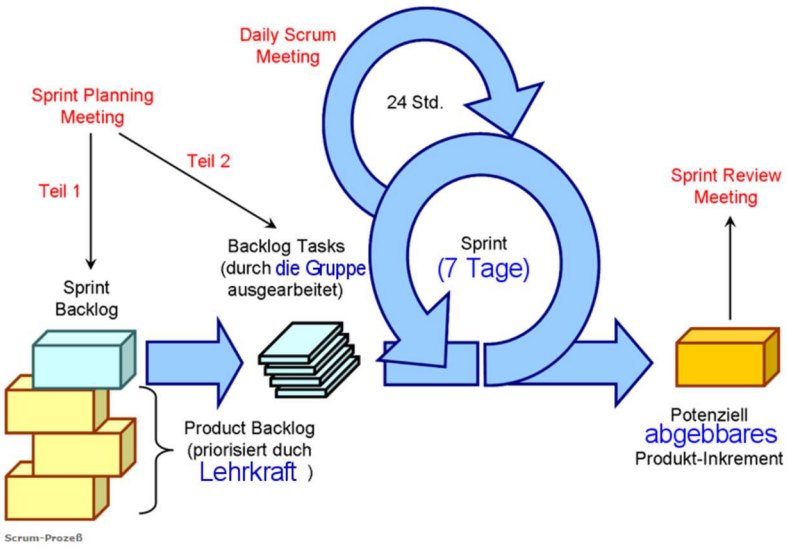
- Ausdruck von entsprechenden Datensätzen durch Generierung von DinA4 Seiten im PDF-Format. (Verwenden Sie dazu Bibliotheken wie iText[2])
- Weitere GUI-Dialoge mit passenden „anderen Sichten" auf den Datenbestand. (Denken Sie sich in den Nutzer hinein – was könnte ihn noch interessieren?)
- Kontext-Hilfe. Durch Drücken der „F1" – Taste könnte eine (HTML-/ PDF- / etc…) Hilfe aufgehen mit dem entsprechenden Hilfseintrag.
- Multi-Language Support. Überlegen Sie sich „String-Tabellen" die entsprechend eine Deutsch / Englisch – Umschaltung Ihrer GUI-Applikation genügen.
- u.v.m… ;-)

[2] http://www.lowagie.com/iText/

4. Projektablauf mit Scrum

Das Scrum-Modell wird für unsere Zwecke angepasst und wie folgt verwendet:

- Die Lehrkraft ist der *Product Owner*,
- Das *Team* besteht aus den zwei Gruppenmitgliedern (Schüler)
- Ein Schüler aus jedem *Team* stellt den *Scrum Master*
- Die Anforderungen (*Requirements*) werden in einer Liste (*Product Backlog*) gepflegt (!), erweitert und priorisiert (ggf. Word/ Ascii –Text / etc. – Datei!)
- **Am Anfang jeder Blockwoche** (i.d.R. Montag-Morgen**)** findet ein *Sprint Planning Meeting* statt. Hierbei wird der *Sprint Backlog* geplant sowie die *Backlog Tasks* auf die Teammitglieder angepasst und verteilt. Einmal festgelegt, dürfen während des gesamten *Sprints* keine weiteren Anforderungen zu diesem *Sprint* dazukommen, um die Fertigstellung nicht zu gefährden!
- Die Teams halten zum Ende jeden Tages ein *Daily Scrum Meeting* ab, welches in einem Dokument protokolliert wird. Dies muss nicht besonders aufwändig sein. Jedoch muss täglich mindestens ein kurzes Fortschrittsupdate von jedem Teammitglied beigesteuert werden!
- **Am Ende jeder Blockwoche** findet ein *Sprint Review Meeting* statt. Hierbei wird der aktuell funktionierende Stand dem *Product Owner* (=Lehrkraft) am Rechner präsentiert. Stellen Sie sich darauf ein, dass diese Vorstellung hin und wieder allen Gruppen vorgeführt wird (Fortschrittsrückkopplung für alle anderen Gruppen)
- Das *Product Backlog* befindet sich ständig im Fluss. Dies muss vom Scrum Master bei entsprechenden Änderungen dokumentiert werden (ggf. History Einträge, etc.)
- Die *Requirements* sowie Priorisierungen können durch den *Product Owner* (nach Notwendigkeit) für jeden „nächsten“ *Sprint* angepasst werden.



© scrum-master.de , modifiziert für BS-Info Projekt

Tipps:

- Durch das Pflegen des *Product Backlog* und diversen *Daily Scrum Meeting*- Protokollen ist die Projektdokumentation zum Ende des Projekts leicht erstellbar und in einem zeitlich überschaubaren Rahmen zu bewerkstelligen. Wird dies vernachlässigt wird es am Ende nur unnötig hektisch und unvollständig was zu Abzügen in der Benotung führt!

- Das letzte „*Review Meeting*“ entspricht gleichzeitig der Projektvorstellung vor dem Product-Owner sowie der **ganzen Klasse**. Zu dem Zeitpunkt muss auch die Projektdokumentation vorliegen, die als Präsentationsgrundlage dienen soll (KEINE WEITERE POWERPOINT o.ä.)

5. Abgabe

Rahmenbedingungen

• Projekt für maximal **zwei** Personen[3]! (Frontend u. Datenbank/ Scrum Master)

• Zeitraum: **8. März 2021** bis **16. April 2021**
➔ Abgabe in der Schule im vorbereiteten Verzeichnis auf dem Klassenlaufwerk!

• Auf kurzfristigen Datenverlust / etc. kann keine Rücksicht genommen werden! Bitte nutzen Sie eine der zahllosen Datenspeicherungs- und Kooperationsplattformen wie Dropbox, G-Drive, MyDrive, Skydrive, xyz-Cloud u.v.m… sinnvollerweise gepaart mit einem Versionsverwaltungstool wie z.B. Subversion, Git, etc...

Was muss nach Fertigstellung abgegeben werden:

• Kompletter Sourcecode (Java 7/8 und UTF-8 Kodierung oder alternative Programmiersprache) mit allen notwendigen Dateien (Eclipse - Projekt). Es dürfen **KEINE ABSOLUTEN PFADE** vorkommen!

Dateiname: **ap_final_2021_*frontend*_projekt.zip**[4] und **ap_final_2021_*db*.zip**

Im Sourcecode muss erkennbar sein, wer für welchen Teil zuständig ist.

• Lauffähiges Java / Windows - Programm oder lauffähige Webseite, welches die mySQL-Datenbank (Version ≥5) über localhost einrichtet.

– Erstellung der notwendigen Datenbank (Name: **gruppeX**).
Hinweis: Die Gruppennummer **X** wird bei Projektstart / Schülerzuteilung vergeben.

– Einrichten eines entsprechenden DB-Users (Name **gruppeXadmin**) mit vollen Rechten auf die Datenbank **gruppeX**.

– Erstellung aller notwendigen Tabellen, Indizes, …

– (ggf.) Einspielen von notwendigen Daten (z.B. vorgegebene, programmnotwendige DB-Inhalte!)

```
java  -jar  gruppeXinit.jar
Admin:  root
PW:     ****
->  Datenbank  'gruppeX'  angelegt
->  User  'gruppeXadmin'  angelegt
->  Tabellen,  ... erstellt
->  Daten  eingespielt
```

BEISPIEL JAVA:

• `gruppeXinit` ist ein lauffähiges Java-Programm in einem JAR-Archiv. Im JAR-Archiv müssen alle notwendigen Klassen, Dateien, Bibliotheken etc. enthalten sein. Der Aufruf erfolgt über

```
java  -jar  gruppeXinit.jar
```

aus einem beliebigen Verzeichnis.

ALLE ANDEREN SPRACHEN:

• **`gruppeXinit.exe`** ist das Installationstool z.B. in C++/C#/…

• **`install.php`** ist die **einmalig** auszuführende Installationsseite im Web-Browser.

[3] Nach Absprache mit Lehrkraft ggf. variierbar
[4] Ein Import über *Eclipse → File → Import → Existing Projects into Workspace* muss einwandfrei funktionieren!

• **Mini-Dokumentation** besteht aus den folgenden Punkten:
- 1.Seite: Projektabgabeseite (Autoren, Kurzbeschreibung, Datenbankmodell als Grafik exportiert) – wird gestellt als „.docx – Datei."
- 2. bis n-te Seite: Scrum-Meeting Proktokolle (Start = Ziel der Woche / Fazit am Ende)
- n.-te – Seite: Abschlussseite mit Projekt-Fazit, Datum und Unterschriften.

⇨ Dateiname: **gruppeX_2021_doku.pdf**

• Eine ZIP-Datei mit folgenden Namen[5] und Inhalt
(*BEISPIEL FÜR ABGABE IN JAVA***)**

<klasse>_<name1>_<name2>_<name3>_gruppeX_2021.zip

z. B. **3FA083_huber_meier_mueller_ gruppeC_2021.zip** die folgende Dateien enthält:

→ gruppeX_*eclipse*_projekt.zip (Sourcecode / Eclipse-Workspace Verzeichnis – View)
→ gruppeX_db.zip (Sourcecode DB – Installationsprogramm)
→ gruppeX_db_init.jar (Installationsprogramm für DB / localhost)
→ gruppeX.jar (Auszuführendes GUI-Programm / Windows)
→ gruppeX_2021_doku.pdf (Projekt-Dokumentation)

Was geht in die Benotung ein?

• Sind alle formalen Kriterien erfüllt?
– Abgabetermin in der Schule eingehalten
– Alle notwendigen Dateien mit korrekten Namen und Format vorhanden?

• Funktionsumfang

• Funktionsfähigkeit[6]

• Sourcecode
Codequalität:
- Wurden entsprechende Klassen und Interfaces verwendet?
- Sind Design Pattern verwendet worden?
- Ist der Code übersichtlich und verständlich?
- Ist der Code fehlerfrei oder erzeugt Exceptions (NullPointerException, ArrayIndexOutOfBounds- Exception, …)

Dokumentation im Sourcecode
- Wurde ausreichend im Code dokumentiert?
- Wurde eine sinnvolle Dokumentierungsstruktur verwendet (z.B. DoxyGen,…)

• Dokumentation

• Kreativität und Ideenreichtum[7]

Wie wird benotet?

• Die Dokumentation wird nach formalen Kriterien, Umfang und Richtigkeit bewertet**.**
• Der Sourcecode wird nach Fehlerfreiheit, Funktionalität und Umfang bewertet**.**
• Das DB-Modell wird auf Anforderungskatalog, Richtigkeit (3. NF, ref. Integrität) sowie Sinnhaftigkeit der gewählten Entitäten bewertet**.**

⇨ Die Projektnote wird als Schulaufgabe gewertet (Attestpflicht!)

[5] In Kleinbuchstaben mit US-ASCII - Kodierung
[6] Compilerfehler oder NullPointerException sind **KO**-Kriterien!
[7] Lieber weniger Funktionen, die sauber und ohne Fehler laufen, als sehr viele Funktionen, die nicht 100% funktionieren!

Was passiert, wenn …

1. kein Projekt bzw. ein unvollständiges Projekt abgegeben wird?

→ Die fehlenden Teile werden mit der Note 6 bewertet.

2. mein USB-Stick bzw. meine Festplatte kurz vor der Abgabe nicht mehr funktionieren?

→ Für die Datensicherung sind Sie selbst verantwortlich. Es gilt 1.

3. ich vor der Abgabe krank werde?

→ Wird termingerecht eine AU abgegeben, bekommen Sie die Tage, in denen Sie nachweislich krank waren, als Verlängerung angerechnet. Ansonsten gilt 1.

4. sich die Datenbank beim Test nach der Abgabe nicht einrichten lässt oder das Programm nicht startet?

   → Bei der Bewertung werden entsprechende Abzüge gemacht, je nachdem, wie schwerwiegend der Fehler ist.

Testen Sie Ihr Programm nicht nur auf dem Rechner, auf dem Sie es entwickeln. Sie sind zu zweit (dritt,…), daher sollten jeweils unabhängige Tests durchgeführt werden!

6. Anhang

Das Vorgehensmodell Scrum – auf einer Seite erklärt[8]

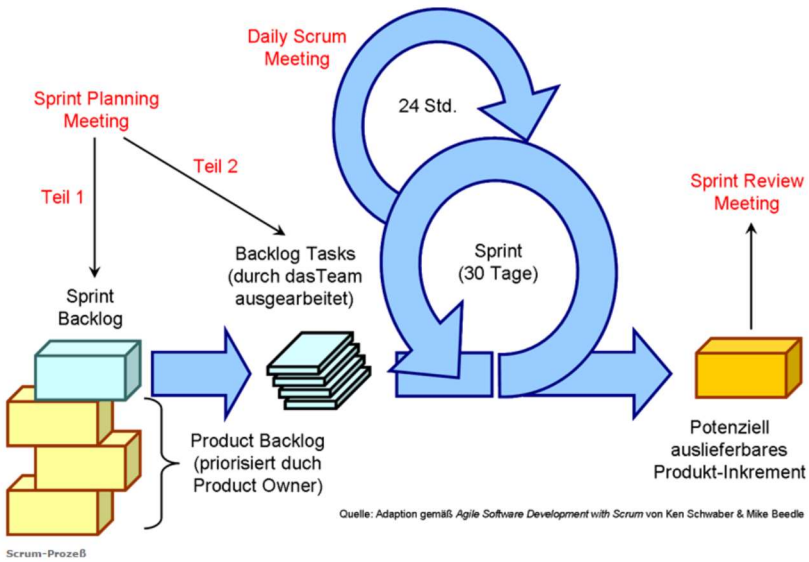
Scrum kennt drei Rollen für direkt am Prozeß Beteiligte: **Product Owner** (stellt fachliche Anforderungen und priorisiert sie), **ScrumMaster** (managt den Prozeß und beseitigt Hindernisse) und **Team** (entwickelt das Produkt). Daneben gibt es als Beobachter und Ratgeber noch die **Stakeholders**.

Die Anforderungen (**Requirements**) werden in einer Liste (**Product Backlog**) gepflegt, erweitert und priorisiert. Das Product Backlog ist ständig im Fluß. Um ein sinnvolles Arbeiten zu ermöglichen, wird monatlich vom Team in Kooperation mit dem Product Owner ein definiertes Arbeitspaket dem oberen, höher priorisierten Ende des Product Backlogs entnommen und komplett in Funktionalität umgesetzt (inkl. Test und notwendiger Dokumentation). Dieses Arbeitspaket, das **Increment**, wird während der laufenden Iteration, des sog. **Sprint**s, nicht durch Zusatzanforderungen modifiziert, um seine Fertigstellung nicht zu gefährden. Alle anderen Teile des Product Backlogs können vom Product Owner in Vorbereitung für den nachfolgenden Sprint verändert bzw. neu priorisiert werden.

Das Arbeitspaket wird in kleinere Arbeitspakete (**Tasks**) heruntergebrochen und mit jeweils zuständigem Bearbeiter und täglich aktualisiertem Restaufwand in einer weiteren Liste, dem **Sprint Backlog**, festgehalten. Während des Sprints arbeitet das Team konzentriert und ohne Störungen von außen daran, die Tasks aus dem Sprint Backlog in ein **Increment of Potentially Shippable Functionality**, also einen vollständig fertigen und potentiell produktiv einsetzbaren Anwendungsteil, umzusetzen. Das Team gleicht sich in einem täglichen, streng auf 15 Minuten begrenzten Informations-Meeting, dem **Daily Scrum Meeting**, ab, damit jeder weiß, woran der andere zuletzt gearbeitet hat, was er als nächstes vor hat und welche Probleme es evtl. gibt.

Am Ende des Sprints präsentiert das Team dem Product Owner, den Stakeholders u.a. interessierten Teilnehmern in einem sog. **Sprint Review Meeting** live am System die implementierte Funktionalität. Halbfertiges oder gar Powerpoint-Folien sind während des Reviews verboten. Das Feedback der Zuseher und die neuen Anforderungen des Product Owners für den kommenden Sprint fließen dann wieder in das nächste Sprint Planning Meeting ein, und der Prozeß beginnt von neuem.

Der ScrumMaster sorgt während des gesamten Prozesses dafür, daß Regeln eingehalten werden und der Status aller Tasks im Sprint Backlog von den jeweils zuständigen Team-Mitgliedern täglich aktualisiert wird. Er macht den Projektfortschritt transparent durch einen geeigneten Reporting-Mechanismus: die Veröffentlichung sog. **Burndown Charts**, welche den Fortschritt für den aktuellen Sprint bzw. für das gesamte Projekt jeweils in Form einer Kurve visualisieren. Eingezeichnete Trendlinien erlauben es, mögliche Probleme und Verzögerungen einfach (und rechtzeitig!) zu erkennen.



Im Kern basiert Scrum also auf einer inkrementellen Vorgehensweise, der Organisation von Entwicklungsabschnitten und Meetings in vordefinierten Zeitabschnitten (**Time-Boxes**) und der Erkenntnis, daß ein funktionierendes Produkt wichtiger ist als eine dreihundertseitige Spezifikation.

[8] Quelle: http://scrum-master.de/