Schulungskonzept zu Git und GitLab – Gamification zum besseren Lernen

Patrick Jüptner[1], Manuela Dalibor[2], Ute Trautwein-Bruns[1] und Bernhard Rumpe[2]

[1]Universitätsbibliothek, RWTH Aachen University;
[2]Software Engineering, RWTH Aachen University

Im Zuge der Digitalisierung wird der Umgang mit textbasierten Datenformaten aber auch die Nutzung von Codeschnipseln zur Datenerhebung und Analyse zu einer grundlegenden Datenkompetenz für Wissenschaftlerinnen und Wissenschaftler, auch in Fachbereichen, die bisher als weniger IT-affin galten. Git und GitLab sind gut etablierte Werkzeuge zur Entwicklung, Dokumentation, Zusammenarbeit und Veröffentlichung von Softwareprojekten und werden zunehmend auch im Kontext des Forschungsdatenmanagements eingesetzt, da sie sich gut zur Versionsverwaltung von textbasierten Forschungsdaten und zur Unterstützung der Zusammenarbeit und des „Data Sharings“ eignen.

Um auch Nicht-Informatikern den Einstieg in diese Tools zu ermöglichen und die Motivation während des Lernprozesses aufrecht zu halten, setzen wir auf das Konzept der Gamification. In einer spielerischen Umgebung werden die Grundlagen der Versionsverwaltung, essentielle Git Befehle und der Umgang mit GitLab trainiert, was die Motivation der Lernenden steigert und sie zur Mitarbeit anspornt. Aus diesem Ansatz sind zwei Konzepte entwickelt worden, die über einen spielerischen Weg den Umgang und die nötigen Kompetenzen zur Verwendung von Git und GitLab vermitteln.

1 Einleitung

Im Bereich des Forschungsdatenmanagements wird der Umgang mit textbasierten Datenformaten und Codeschnipseln zu einer grundlegenden Datenkompetenz für Wissenschaftlerinnen und Wissenschaftler. Die Zusammenarbeit und das Teilen solcher Daten sind dabei eines der Kernthemen. Git und GitLab sind bereits etablierte Werkzeuge zur Entwicklung, Dokumentation, Zusammenarbeit und Veröffentlichung von Softwareprojekten [1], welche in zunehmendem Maße auch im Kontext des Forschungsdatenmanagements eingesetzt werden [2], da sie sich gut zur Versionsverwaltung von textbasierten Forschungsdaten, wie z.B. Tabellen im csv-Format, und zur Unterstützung der Zusammenarbeit und des „Data Sharings“ eignen. Weil dies auch weniger IT-affine Fachbereiche betrifft, sind Schulungskonzepte erforderlich, die auf einfache Art und Weise den Einstieg in diese Tools ermöglichen.

Publiziert in: Vincent Heuveline, Nina Bisheh (Hg.): E-Science-Tage 2021. Share Your Research Data. Heidelberg: heiBOOKS, 2022.
DOI: https://doi.org/10.11588/heibooks.979.c13715 (CC BY 4.0)

Aus diesem Grund und um die Motivation im Lernprozess aufrecht zu halten, haben wir uns für das Konzept der Gamification entschieden [3]. Kern von Gamification ist die Anwendung von Spielkonzepten bzw. spieltypischen Elementen in einem spielfremden Kontext. Mit der „Git Scavenger Hunt“ und „WordGuess“ wurden zwei Spielkonzepte entwickelt, in denen in spielerischer Umgebung die Grundlagen der Versionsverwaltung, essentielle Git Befehle und der Umgang mit GitLab geübt werden. Dies soll die Motivation des Lernenden steigern und zur Mitarbeit anspornen [4]. Darüberhinaus liegt im WordGuess-Konzept der Fokus auf GitLab als Projektmanagement-Tool. Für den konzeptionellen Rahmen setzen wir auf Scrum als Projektmanagement-Methode, welche hauptsächlich in der agilen Softwareentwicklung eingesetzt wird [5]. Die Rollen und Phasen des Scrum-Modells werden auf die Spielumgebung transferiert und steuern den Spielablauf. So erhalten die Lernenden als Add-On einen oftmals ersten Einblick in ein Projektmanagement-Modell und können sich spielerisch mit diesem vertraut machen.

Die dargestellten Konzepte sind Teil eines Schulungsprogramms zu Git und GitLab, das sich aus einem Einstiegskurs zu Git und Gitlab (elearning), dem Training in den Lernspielen und einem Präsenzworkshop zusammensetzt. Der Moodle-Kurs „Einstieg in die Versionsverwaltung mit Git und GitLab“ führt in die Thematik der Versionsverwaltung mit Git und des Projektmanagements mit Gitlab ein und bietet Tutorials, um die entsprechende Arbeitsumgebung zu installieren und erste Grundbefehle kennenzulernen. Der Präsenzworkshop schließlich dient neben der vertiefenden Übung der Prozesse dem interaktiven Austausch mit Kollegen und der Diskussion von Anwendungsbeispielen.

Initiert wurde die Entwicklung der Lernspiele aufgrund der großen Nachfrage nach Git/GitLab-Schulungen an der RWTH Aachen University auch für individuelle Arbeitsgruppen, dem Wegfall von Präsenzkursen während der Corona-Pandemie und der Beobachtung, dass das Niveau der Gitkompetenzen im Workshop stark variierten und komplette Git-Neulinge davon profitieren würden, ein wenig Training der Befehle und Prozesse vorab zu haben.

2 Git Scavenger Hunt - eine Schnitzeljagd mit Git

2.1 Kursaufbau

In Git Scavenger Hunt gilt es für den Spieler sieben Level mit ansteigender Schwierigkeit zu lösen, um die Schnitzeljagd erfolgreich abzuschließen. Das vorrangige Lernziel dieses Konzeptes ist es, das Verständnis dafür zu erhöhen, was ein Git Repository ist, wie in diesem navigiert wird und wie auf frühere Versionen zugegriffen werden kann.

Git Scavanger Hunt ist eine Einzelspieler-Erfahrung über sieben Level, d.h. jeder Teilnehmende löst die Schnitzeljagd für sich auf dem eigenen Rechner. Die komplette Git Scavanger Hunt-Spielumgebung befindet sich in einem öffentlichen GitLab-Repository (`https://git.rwth-aachen.de/patrick.jueptner/git-hunt-lvl1-7`). Dieses wird zu Beginn von jedem Spieler lokal auf den eigenen Rechner geklont. Das komplette Spiel

findet in diesem lokalen Klon über die Kommandozeile (Terminal) statt. Dies ermöglicht ein direktes reagieren auf Meldungen von Git und fördert so das grundsätzliche Verständnis für die durchgeführten Operationen und Befehle [6]. Nachfolgend werden sowohl das Repository als auch die weiteren Bestandteile des Schulungkonzeptes beschrieben.

Den Einstieg in den Kurs bildet ein RWTHmoodle Raum (Abb. 1) [7], in dem Anleitungen und Tutorials bereitstehen, welche sämtliche benötigen Befehle und Konzepte umfassen, die zur Lösung der Schnitzeljagd - vollständig in der Umgebung der Kommandozeile - benötigt werden. Dies sind neben den verwendeten Git-Befehlen auch die grundlegenden Kommandos für den Umgang mit der Kommandozeile, wie zum Anzeigen von Ordner-/Verzeichnisinhalten, dem Wechseln von Verzeichnissen oder dem Erstellen von neuen Verzeichnissen. Außerdem wird eine Möglichkeit zum Anzeigen von Inhalten einer Textdatei direkt in der Konsole gezeigt. Da Git, z.B. bei einem Merge nach der „Recursive Strategy“, automatisch einen Texteditor für die Eingabe einer Commit-Message startet, wird auch der grundlegende Umgang mit dem Texteditor vim [8] erklärt.

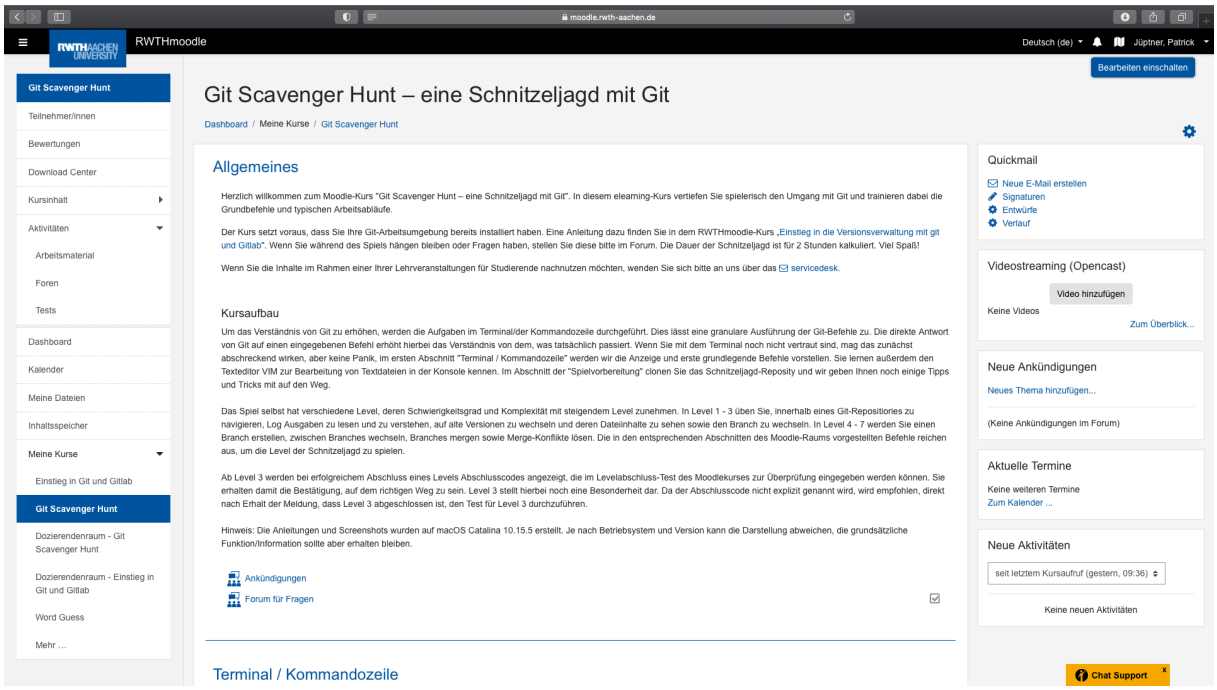

Abbildung 1: Startseite des Moodle Raums Git Scavenger Hunt.

Der Moodle Raum verfügt über ein Forum für Fragen und den Austausch der Teilnehmenden untereinander. Hier können die Spieler Fragen zum Spiel oder den verwendeten Befehlen stellen und sich auch gegenseitig beantworten. Die Kursbetreuer beobachten und moderieren das Forum und beantworten die Fragen nach Bedarf.

Für die Level 3 bis 7 bietet der Moodle Raum die Möglichkeit, Lösungscodes einzugeben, um den korrekten Abschluss eines Levels zu überprüfen und sich nicht in einer Sackgasse zu verlaufen. Die Lösungscodes erhalten die Spieler jeweils zum Abschluss eines Levels. Über den Levelabschluss-Test kann zudem überprüft werden, welche Teilnehmer wie weit

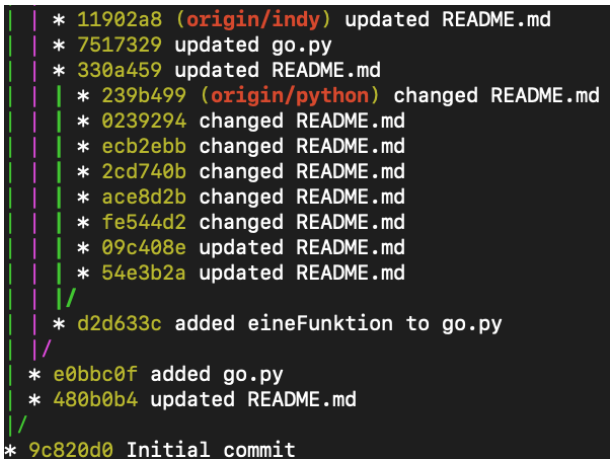

Abbildung 2: Ausschnitt des Git Scavenger Hunt Repositories mittels git log –graph –oneline –all.

im Spiel gekommen sind, um gegebenenfalls individuell auf spezielle Probleme in einem bestimmten Level zu reagieren. Um die Schnitzeljagd lösen zu können, müssen die Git Befehle log, checkout, merge und branch angewendet werden.

Über den Befehl *git log* läßt sich die Commit-Historie eines Repositories anzeigen. In seiner einfachsten Form listet der Befehl alle Commits, inklusive Commit-Message, Commit-ID, Identität der Person, die den Commit durchgeführt hat, und weiteren Informationen auf. Damit lassen sich die Änderungen der Dateien in einem Repository verfolgen.

Mittels *git checkout* wird zwischen Commits und Branches gewechselt. Dies ist im Spiel notwendig, um die benötigten Hinweise zu finden und die gestellten Aufgaben zu lösen. Der Befehl ermöglicht es dem Nutzer, durch das Repository zu navigieren und durch den Wechsel in ältere Commits auf ältere Versionen von Dateien zuzugreifen. Die Übung mit dem Befehl verdeutlicht die Funktionsweise der Versionierung innerhalb von Git und wie der Zugriff und die Weiterarbeit mit älteren Dateiversionen abläuft.

Der Befehl *git merge* wird verwendet, um zwei Entwicklungszweige (Branches) in Git zusammenzuführen. In der Softwareentwicklung ist es überlich, dass neue Features in Softwareprojekten in einem eigenen Branch entwickelt und getestet werden, bevor diese mit dem Features dann über einen Merge in den Master-Branch einfließen.

git branch ist dafür zuständig, einen neuen Branch zu erstellen. Dies geschieht aus dem aktuellen Commit heraus, in dem sich der Anwender befindet, oder über die Angabe einer Commit-ID, von welcher aus der neue Branch erstellt werden soll. Auf dem kurzen Ausschnitt des Repsitories in Abb. 2 sind einige Branches und die Commits, aus denen sie erstellt wurden, zu erkennen.

2.2 Lernziele

Mit der Git Scavenger Hunt lernen die Teilnehmenden, grundlegende Kommandos auf der Kommandozeile kennen und können durch Ordnersysteme auf ihrem Rechner navigieren. Die Teilnehmende verstehen wie ein Git-Repository aufgebaut ist und können mit den Git Befehlen log, checkout, branch und merge umgehen. Sie können die Log-Ausgaben von Git lesen und verstehen diese. Darüberhinaus können sie Branches erstellen und wechseln sowie Merge-Konflikte lösen.

2.3 Spielablauf

Der Spielablauf ist für jedes Level der Schnitzeljagd identisch (Abb. 3). Jedes Level startet in einem Commit.
Beim Start des Spiels ist dies der aktuellste Commit im Master Branch, der beim Klonen eines Git Repositories standardmäßig aktiv ist. Für die weiteren Level erhält der Spieler jeweils einen Hinweis, in welchem Commit das nächste Level startet. Ist der Commit gefunden, gilt es stets die README.md Datei zu lesen. Üblicherweise enthält die README-Datei Informationen über das Projekt oder die Software, zu der die Datei gehört. Sie informiert über wichtige Details und sollte vom Benutzer grundsätzlich gelesen werden. In diesem Spiel enthält die README-Datei die Hinweise bzw. die Aufgabenstellung zum Level. Diese Aufgaben reichen vom Finden von Informationen in Commit-Nachrichten bis hin zur Durchführung von Merge-Operationen auf Branches des Repositories. Die Aufgaben sind im Schwierigkeitsgrad ansteigend. Im Falle der Merge-Aufgaben muss z.B. erst ein „Automatic Merge“ durchgeführt werden, den Git selbstständig vollziehen kann. Im späteren Verlauf müssen Merge Konflikte gelöst werden und diese mit einer entsprechenden Commit-Message ins Repository aufgenommen werden. Die korrekte Lösung der Aufgaben ergibt sich dabei entweder direkt aus dem Dateiinhalt oder aus Textausgaben auf dem Bildschirm oder sie wird in einigen Leveln über ein Skript geprüft, welches der Spieler ausführen muss. Sollte ein Skript zur korrekten Überprüfung der Lösung zur Anwendung kommen, wird der Spieler in der Aufgabenstellung schon darauf hingewiesen. Wurde die Aufgabe erfolgreich gelöst, erhält der Spieler in den Leveln 3 bis 7 einen Levelsabschlusscode zur Eingabe im Moodle Raum.
Dieser Code kann eine bestimmte Zeichenfolge sein, oder aber Teil einer Commit-Nachricht (z.B. Datum), zu der der Spieler in der Schnitzeljagd geführt wurde. Außerdem erhält der Spieler einen Hinweis, in welchem Commit das nächste Level startet. Dieser Hinweis ist dabei oftmals kein direkter Verweis auf den gesuchten Commit, sondern über ein Rätsel zu entschlüsseln. Hierbei kann und muss auf Google oder eine andere Suchmaschine der Wahl bzw. das Internet im allgemeinen zurückgegriffen werden, um an die benötigen Informationen zu gelangen und so den Weg zum richtigen Commit zu finden.

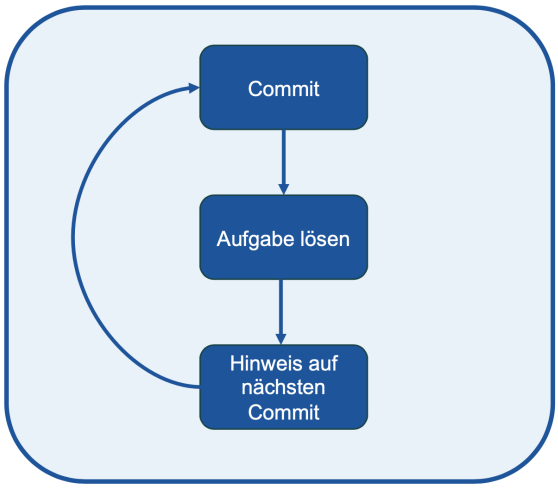

Abbildung 3: Allgemeiner Ablauf der Git Scavanger Hunt.

2.4 Beispiellevel

Bei Abschluss der vorausgehenden Levels hat der Spieler den Hinweis erhalten, dass es im Branch Caesar weitergeht. Der erste Schritt besteht für den Spieler darin, in den Branch Caesar zu wechseln.

Befehle sind im folgenden Fett dargestellt, die folgende Bildschrimausgabe ist nicht-fett.

```
> git checkout Caesar
Your branch is up to date with 'origin/Caesar'.
```

Im nächsten Schritt ist die Date README.md zu lesen, sie enthält die Aufgabenstellung und Hinweise.

```
> cat README.md
Hallo, ich bin Caesar und du hast den richtigen Branch
↪  ausgecheckt!
In diesem Branch werde ich eines meiner Meisterwerke
↪  schreiben, jedenfalls Teile davon.

Es wird der Gallische Krieg heißen und sehr
↪  wahrscheinlich mehrere Bücher umfassen.
In diesem Branch (Caesar) habe ich mein Werk mit Buch
↪  1 begonnen.
Buch 2 habe ich in einem eigenen Branch geschrieben.

Deine Aufgabe:
merge den Branch buch_2, in welchem ich Buch 2
↪  vollendet habe, in diesen Branch.
```

```
Den Befehl dazu solltest du bereits kennen. Eine
 ↪  Kleinigkeit wird diesmal aber anders sein, da
 ↪  diesmal kein automatic/fast forward Merge
 ↪  durchgeführt werden kann, sondern
nur ein Merge nach der recursive Strategy.
Der Merge muss als Commit in das Repository
 ↪  aufgenommen werden und erfordert auch eine Commit
 ↪  Message.
Aber keine Sorge, das meiste davon erledigt Git für
 ↪  dich von allein. Ich denke, dass du das hin
 ↪  bekommst!
Nach dem Merge musst du erneut das Skript checker.sh
 ↪  aufrufen, diesmal aber ohne Parameter. Der Befehl
 ↪  lautet:

./checker.sh

Hast du alles richtig gemacht, erhälst du den nächsten
 ↪  Hinweis!
```

Als nächstes muss der Merge durchgeführt werden.

```
> git merge buch_2
Merge made by the 'recursive' strategy.
Buch2.txt | 9 +++++++++
1 file changed, 9 insertions(+)
create mode 100644 Buch2.txt
```

Nach Ausführung des merge Befehls, muss eine Commit Message über den Texteditor eingegebn werden, bevor der gezeigte Merge durchgeführt wird. Die Überprüfung findet über das Skript checker.sh statt.

```
>./checker.sh

Herzlichen Glückwunsch! Der Merge hat geklappt!
Damit hast du Level 5 abgeschlossen!
Der Levelabschlusscode für Level 5 lautet: 9gt2b7
Gib diesen Code im Moodle Raum ein!
Weiter geht es in einem anderen Branch.
Finde heraus welcher Branch gesucht ist und checke ihn
 ↪  aus.
Der Hinweis:
Trfhpug vfg qre Anpuanzr rvarf Znaarf, qre fvpu vz
 ↪  Cevamvc nhpu zvg Mrvpuraxbqvrehat orfpunrsgvtg
 ↪  ung. Re jheqr 1791 trobera haq vfg 1872 trfgbeora.
```

```
Qvr iba vuz haq rvarz
Xbyyrtra reshaqrar Mrvpuraxbqvrehat jheqr nhpu Ynaq
↪ Yvar Pbqr tranaag. Purpxr qra Oenapu nhf, qrffra
↪ Anzr qrz Anpuanzra qre trfhpugra Crefba ragfcevpug!
```

Der Level ist erfolgreich beendet worden, der Levelabschlusscode und der Hinweis auf den nächsten Level sind dem Spieler angezeigt worden. In diesem Fall ist der Hinweis ein verschlüsselter Text, der dekodiert werden muss. Da das Spiel derzeit im Caesar Branch ist, ist der Code mittels Caesar Verschlüsselung kodiert [9]. Zur Dekodierung muss noch der Verschiebungsfaktor gefunden werden. Diesen findet der Spieler in der README.md Datei in einem früheren Commit des Branches. Hat der Spieler den Text dekodiert, kann er in den genannten Branch wechseln und das Spiel dort fortsetzen.

3 WordGuess – ein kollaboratives Spiel mit Git und GitLab

3.1 Kursaufbau

In WordGuess wird auf spielerische Art und Weise die Arbeit mit Git und GitLab trainiert und gleichzeitig ein Fokus auf das Projektmanagement entsprechend dem Scrum-Modell gesetzt. WordGuess ist dabei als kooperative Mehrspielererfahrung ausgelegt, in der die Teilnehmenden in Gruppen von idealerweise fünf Personen gemeinsam in einem GitLab-Repository als Spielumgebung spielen. Im Spielverlauf werden die Phasen und Rollen von Scrum auf die Spielphasen übertragen und mit einem Issue-Board in GitLab abgebildet, um so den typischen Ablauf einer per Scrum geführten Projektentwicklung mit GitLab nachzustellen. Ziel des Spiels ist es, eine Spielrunde entsprechend eines Sprints erfolgreich abzuschließen, indem ein Suchwort erraten wird. Die Spielidee orientiert sich dabei an JUST ONE, dem Spiel des Jahres 2019. Die Herausforderung besteht nun aber darin, dass die Spielenden nicht zusammen an einem Spieltisch sitzen, sondern verteilt an Ihren Schreibtischen. Die notwendige Kommunikation zum Spiel findet deshalb ausschließlich über Kommentare in GitLab-Issues statt. Aber auch die typischen Arbeitsabläufe im Zusammenspiel von Git und GitLab bei der gemeinsamen Bearbeitung von Dateien werden im Spiel trainiert, da es im Spielverlauf erforderlich ist, gemeinsam an einer Textdatei, dem „Sprint Backlog“ zu arbeiten. Dazu müssen die Spielenden das GitLab-Repository lokal auf den Rechner klonen, die Datei verändern und wieder an das Remote-Repository in GitLab übertragen. Das GitLab-Wiki wird zur Projektdokumentation eingesetzt. Außerdem finden die Teilnehmenden dort auch die kondensierten Informationen zum Spielablauf.

Den Einstieg in den Kurs bildet wieder ein RWTHmoodle Raum [10]. Dort stehen den Teilnehmenden alle Tutorials und Anleitungen bereit, die für das Spiel benötigt werden. Der Fokus dieses Kurses liegt auf GitLab und der Vermittlung von Basiswissen zum Projektmanagement. Die benötigten Git-Kenntnisse werden bereits mit der Git-Schnitzeljagd abgedeckt, so dass sie in diesem Kurs lediglich mit einem Quiz aufgefrischt werden, das gleichzeitig dem Selbsttest der Teilnehmenden dient, ob die Git-Kenntnisse für das Spiel

ausreichen. Die Einteilung in Spielgruppen erfolgt durch Selbstzuordnung in der Gruppenverwaltung von Moodle. Zur initiale Rollenverteilung dient ein Gruppenforum. Neben diesem beinhaltet der Moodle-Raum auch ein Standardforum für generelle Fragen zu den Kursinhalten.

3.2 Lernziele

Nach Absolvieren des WordGuess-Kurses verstehen die Teilnehmenden, wie GitLab die Zusammenarbeit an (textbasierten) Forschungsdaten unterstützt und können Projektmanagementkonzepte mit GitLab umsetzen. Sie kennen mit dem GitLab-Wiki eine Möglichkeit der Dokumentation von Forschungsprojekten und Entwicklungsprozessen und können Wikiseiten erstellen und bearbeiten. Die Teilnehmenden verstehen die verschiedenen Ebenen von Git/GitLab (workspace-local Repository–Remote Repository) und haben den grundsätzlichen Git/GitLab-Workflow mit „add-commit-push“ vertieft. Mit den Befehlen „push“ und „pull“ können sie zwischen lokalem und remote Repository kommunizieren. Sie können Branches erstellen und wechseln sowie Merge-Konflikte auflösen.

Abbildung 4: Ablauf einer WordGuess Runde.

3.3 Umsetzung der Spielidee

In WordGuess nehmen die Spielenden die verschiedenen Rollen nach dem Scrum-Modell ein. Es gibt einen Product Owner, einen Scrum Master, alle übrigen bilden das Entwicklerteam. Der Product Owner trägt die Produktverantwortung. In WordGuess entspricht er dem Rater. Wenn er den Ratebegriff errät, hat das Team ein funktionsfähiges Produkt

Tabelle 1: Scrum Begriffe und ihre Entsprechung in WordGuess.

Scrum	WordGuess
Sprint	Spielrunde
Product Owner	Rater
Scrum Master	Organisiert/moderiert, gibt Suchbegriff vor
Entwicklerteam	Stellen und priorisieren Hinweise
Backlog	Textdatei mit Suchbegriff und Hinweisworten
Scrum Phasen	Scoped Labels in GitLab-Issues
Planning	Suchbegriff bestimmen, Hinweise zusammenstellen
Planning Poker	Max. Anzahl an zu gebenden Hinweisen
Daily Scrum	Hinweis geben - Suchwort raten
Sprint Review	Projektdokumentation erstellen
Sprint Retrospektive	Neue Rollenverteilung

entwickelt. Der Scrum Master ist für die Überwachung und Einhaltung der Scrum-Regeln verantwortlich. Er bestimmt das Suchwort und moderiert den gesamten Prozess. Das Entwicklerteam hat die Aufgabe, die Sprints inhaltlich zu planen und die Pläne dann auszuführen. Sie stellen die Hinweisworte zusammen und diskutieren gemeinsam mit dem Scrum Master die Gestaltung des Sprints. Ziel des Spiels ist es, dass der Product Owner anhand von Hinweisen ein vom Scrum Master präsentiertes Suchwort errät.

Gespielt wird über mehrere Runden, wobei eine Runde einem Sprint in der Scrum-Terminologie entspricht. Ein Sprint teilt sich in die Phasen Planning, Daily Scrum, Sprint Review und Sprint Retrospektive auf, wobei sich das Daily Scrum solange wiederholt, bis der Ratebegriff erraten wurde oder kein weiterer Hinweis im Sprint-Backlog vorgesehen ist. Eine Übersicht über den Ablauf und die Phasen eines Sprints gibt Abb. 4. Eine tabellarische Übersicht der Scrum-Begriffe und ihrer Entsprechung im WordGuess-Spiel ist in Tabelle 1 aufgeführt.

Projektmanagement und Kommunikation werden in GitLab als Spielumgebung mit den Features Issues, Labels, Boards und Milestones umgesetzt:

Issues: Jede Spielrunde (Sprint) verfügt über die GitLab-Issues „Sprint_x“ und „Sprint_x_Entwicklung“ (x=Rundennummer). Diese werden zu Beginn des Sprints vom Scrum Master erstellt und mit den entsprechenden Labels versehen. In diesen Issues findet über Kommentare die gesamte Kommunikation innerhalb des Spiels statt. Der Scrum Master weist die Issues den restlichen Spieler zu, so dass diese eine Benachrichtigung erhalten, sobald im Issue eine Änderung erfolgt ist.

Labels: Die Scrum-Phasen werden über Scoped Labels abgebildet. Diese haben den Vorteil, dass zu jedem Zeitpunkt nur ein Scoped Label des gleichen Typs einem Issue zugeordnet werden kann und die Scrum Phasen so klar voneinander getrennt sind. Zur Kennzeichnung der aktiven Issues und ob ein Sprint erfolgreich war oder nicht, sind im Spiel weitere normale, unscoped Labels vorhanden.

Boards: Zum Planen, Organisieren und Visualisieren des Projektverlaufs können Issue-Boards verwenden werden.
Das Scrum-Board besteht neben der „Open“- und „Closed“-Listen aus Listen für die verschiedenen Scrum-Phasen-Labels. Im Spiel verschiebt der Scrum-Master die Issues entsprechend der Scrum-Phasen über das Board.

Milestones: Milestones werden genutzt, um dem Projekt eine zeitliche Struktur zu geben. In WordGuess wird für jeden Sprint ein Milestone erstellt und dieser den entsprechenden Issues zugewiesen. Wie lange ein Sprint in WordGuess dauert, stimmen die Spielenden im „Planning Poker“ ab.

3.4 Exemplarischer Ablauf eines Sprints

In der Spielvorbereitung muss der erste Scrum-Master die Spielumgebung bereitstellen. Dazu importiert er die über den Moodle-Raum bereitgestellte Exportdatei des WordGuess-Projekts in GitLab. Er legt das Scrum-Board an und lädt seine Mitspieler als Mitglieder zu seinem Projekt ein. Das Spielteam klont das GitLab-Repository und ist damit startklar. Im Folgenden wird exemplarisch für eine Spielrunde (siehe Abb.4) der Sprint 1 ausgeführt. Alle weiteren Sprints verlaufen analog.

Planning: Ein Sprint startet immer mit der Planning-Phase. In dieser Phase erstellt der Scrum Master die zum Sprint gehörenden Issues „Sprint_1“ und „Sprint_1_Entwicklung“. Im Issue „Sprint_1“ vergibt er das Label *In Progress*, um zu kennzeichnen, dass der Sprint begonnen hat, und weist das Issue seinen Mitspielern zu. Das Issue „Sprint_1_Entwicklung“ weist er nur dem Entwicklerteam zu. Auf dem Scrum-Board verschiebt er nun die Issues in die Liste des *Planning*-Labels, um zu kennzeichnen, dass die Planning-Phase läuft.

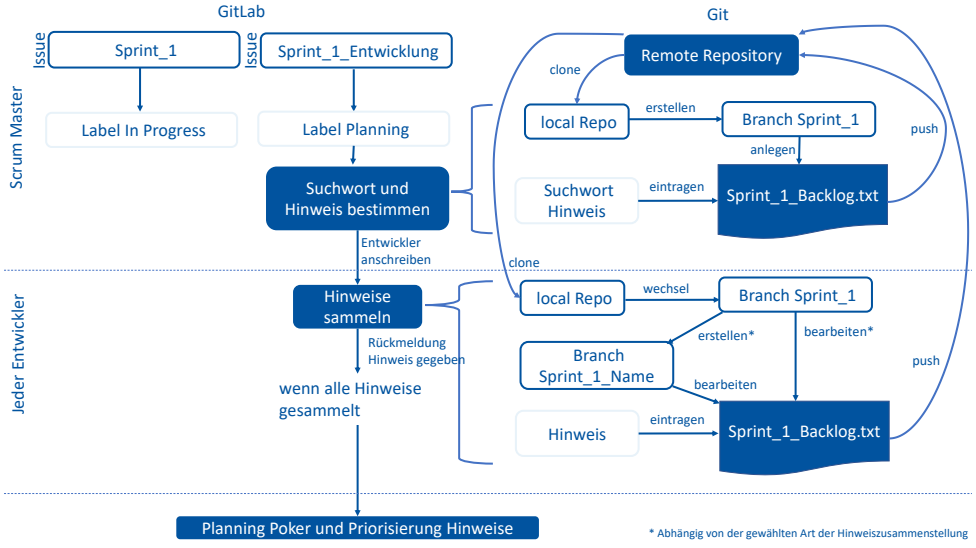

Abbildung 5: Ablauf der Planning Phase nach Erstellung der zugehörigen Issues.

Der weitere Ablauf der Planning-Phase findet sowohl in der Git-Umgebung, als auch im GitLab statt. Ein Überblick der Planning-Phase ist in Abb.5 dargestellt. Der Scrum Master erstellt einen neuen Branch „Sprint_1“ und legt die Datei „Sprint_1_Backlog.txt“ an, in die er das Suchwort sowie ein erstes Hinweiswort einträgt. Er pusht zurück ins Gitlab und fordert das Entwicklerteam auf, Hinweisworte zu ergänzen. Das Entwicklerteam aktualisiert ihr lokales Repository, wechselt in Branch „Sprint_1“ und hat nun zwei Möglichkeiten:

Variante A: Jeder fügt sein Hinweiswort in die Datei „Sprint_1_Backlog.txt“ ein und pusht ins Gitlab. Es werden Merge-Konflikte entstehen, da die Teammitglieder nicht aufeinander warten werden. Dies führt dazu, dass eventuell das lokale Gitlab Repository erneut aktualisiert werden muss, wenn der Stand im Gitlab von einem anderen Teammitglied geändert wurde.

Variante B: Jedes Teammitglied erstellt aus „Sprint_1“ einen weiteren Branch, z.B. „Sprint_1_Name“, fügt dort in der Datei „Sprint_1_Backlog.txt“ ein Hinweiswort hinzu und pusht ins Gitlab. Dabei sollten keine Merge-Konflikte entstehen. In dieser Variante mergt der Scrum Master alle erstellten Unter-Branches in seinen „Sprint_1“ und löst die Merge-Konflikte. In beiden Varianten sind die Merge-Konflikte so zu lösen, dass kein Hinweiswort verloren geht. Die Variante B ist etwas aufwendiger, bietet aber die Möglichkeit den Umgang mit Branches und Merges in Git zu üben.

Nun vergibt der Scrum Master das Label „Planning Poker“ in Issue „Sprint_1_Entwicklung“ und diskutiert über Kommentare mit seinem Entwicklerteam die Anzahl und Reihenfolge der Hinweise sowie die Sprintdauer. Er aktualisiert das „Sprint_1_Backlog.txt“, legt einen Milestone „Sprint_1“ entsprechend der festgelegten Sprintdauer an und ordnet beide Issues diesem Milestone zu. Schließlich entfernt er das Label „Planning Poker“.

Daily Scrum: Der Scrum Master verschiebt die Issues auf dem Scrum-Board in die Liste des *Daily Srcum*-Labels. Da es sich, wie alle Labels der Scrum-Phasen, um ein Scoped Label handelt, sorgt dies automatisch dafür, dass das Label ”Planningäus dem Issue entfernt wird. Weiter schreibt der Scrum Master den Product Owner an, dass ein Suchwort bestimmt wurde und verrät ihm den ersten Hinweis. Der Ablauf der Daily Scrum-Phase ist in Abb.6 dargestellt.

Der Product Owner versucht anhand des Hinweises das Suchwort zu erraten und erhält Feedback. Hat der Product Owner falsch geraten und es sind noch weitere Hinweisworte vorgesehen, startet das Daily Scrum erneut. Hat er richtig geraten, ist das Sprintziel erreicht. Der Sprint ist erfolgreich und das Sprint Review folgt. Sind alle Hinweise gegeben oder die Zeit abgelaufen ohne dass der Product Owner den Suchbegriff erraten konnte, endet der Sprint „erfolglos“ und das Sprint Review folgt ebenso.

Sprint Review: Der Scrum Master verschiebt die Issues auf dem Scrum-Board in die Liste des *Sprint Review*-Labels. Er entfernt das Label „In Progress“ in Issue „Sprint_1“ und vergibt das Label „Sprint erfolgreich“ oder „Sprint fehlgeschlagen“. Er verrät das Suchwort, falls es nicht geraten wurde.

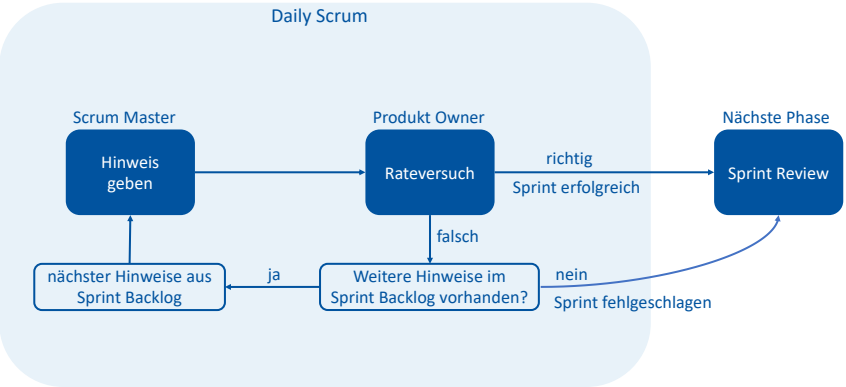

Abbildung 6: Ablauf der Phase Daily Scrum.

Er teilt dem Entwicklerteam über einen Kommentar in Issue „Sprint_1_Entwicklung“ den Ausgang des Sprints mit und schließt das Issue.

Im Sprint Review wird im GitLab-Wiki die Projektdokumentation zum aktuellen Sprint entsprechend einer vorgegebenen Template verfasst. Dies kann durch den Scrum Master erfolgen, oder ein Entwickler meldet sich freiwillig oder wird bestimmt. Nach Erstellung der Dokumentation erfolgt eine Rückmeldung an die Mitspieler und Änderungen können eingebracht werden.

Sprint Retrospektive: Der Scrum Master verschiebt die Issues auf dem Scrum-Board in die Liste des *Sprint Retrospektive*-Labels. In der „Sprint Retrospektive“ werden die Rollen für den nächsten Sprint vergeben. Dazu übergibt der Scrum Master seinen Posten über einen Kommentar in Issue „Sprint_1“ an eine/-n Mitspieler/-in. Der oder die Auserwählte bestätigt dies im Kommentar. Der Scrum Master entfernt das Label *Sprint Retrospektive*, schließt das Issue „Sprint_1“ und wird zum neuen Product Owner im nächsten Sprint. Alle weiteren Teilnehmenden sind automatisch Teil des neuen Entwicklerteams.

4 Fazit/Ausblick

Die Git Scavenger Hunt wird seit Januar 2021 im Kursprogramm für Promovierende und den akademischen Mittelbau an der RWTH Aachen University angeboten [7]. WordGuess wird im Mai 2021 folgen [10]. Zudem wurden die Kurse mit Studierendengruppen getestet und bereits in erste Lehrveranstaltungen integriert.

Das Feedback aus ersten Spielrunden in beiden Konzepten ist überaus positiv. Beispielsweise antworteten einige Studierende der Informatik, dass sie der spielerische Ansatz motiviert habe, die Schnitzeljagd bis zum Ende durchzuspielen, obwohl bereits vorab Git-

Kenntnisse vorhanden waren. An WordGuess wurden nach einem Test mit 16 Studierenden, welche sich gegenseitig Hinweise gaben und in den verschiedenen Rollen abwechselten, noch Verfeinerungen an dem Spiel durchgeführt. Zum einen wurden die Scrum Rollen als Lehrergänzung hinzugefügt, zum anderen die Gruppengröße limitiert.

Beide Kurse wurden weitestgehend im Sinne von Open Educational Ressources (OER), also nachnutzbar für Jedermann, erstellt. Limitierender Faktor ist das RWTHmoodle als Lernplattform, das den Zugang auf Personen mit RWTH-ID einschränkt. Um dies zu relativieren werden die Kurssicherrungsdateien der Moodle-Räume über RWTH Publications zum Download bereit gestellt, so dass sie in anderen Moodle-Instanzen wiederhergestellt und ggf. angepasst werden können. Die GitLab-Repositorien selbst sind öffentlich und enthalten ausreichend Informationen, um die Spielideen auch ohne begleitenden Moodle-Raum umsetzen zu können. Beide Lernspiele sind geeignet, um Data Literacy Kompetenzen an Studierende zu vermitteln. Für Dozierende der RWTH Aachen University werden deshalb die Kursinhalte in einem separaten Moodle-Raum bereitgestellt, von dem aus sie abgeholt und in eigene Vorlesungs- und Übungskonzept eingepflegt werden können.

Aufgrund des großen Interesses und dem generellen Bedarfs an Schulungsangeboten zu Git und GitLab, planen wir auch zukünftig die Konzepte weiterzuentwickeln und zu erweitern. Ideen für weitere Level der Schnitzeljagd liegen bereits vo.[1]

Literaturverzeichnis

[1] Hethey, Jonathan M. “GitLab Repository Management.” Packt Publishing Ltd (2013).

[2] Ayer, Vidya and Herrmann, Fabian and Peil, Vitali and Pietsch, Christian and Rempel, Andreas and Schirrwagen, Jochen and Vompras, Johanna and Wiljes, Cord. “Automatische Qualitätskontrolle von Forschungsdaten durch kontinuierliche Integration mit GitLab CI.” 2019.

[3] Sailer, Michael and Hense, Jan and Mandl, J and Klevers, Markus. “Psychological perspectives on motivation through gamification.” Interaction Design and Architecture Journal, Nr. 19 (2014): 28-37.

[4] Sailer, Michael. “Die Wirkung von Gamification auf Motivation und Leistung.” Springer (2016).

[5] Schwaber, Ken and Sutherland, Jeff. “The scrum guide.” Scrum Alliance, Nr. 21 (2011).

[6] Umali, Rick. “Learn Git in a Month of Lunches.” Manning Publications Co. (2015).

[7] Jüptner, Patrick und Dalibor, Manuela und Trautwein-Bruns, Ute. “Git Scavenger Hunt - Eine Schnitzeljagd mit Git”. DOI: https://doi.org/10.18154/RWTH-2021-03720.

[1] Gefördert durch die Deutsche Forschungsgemeinschaft (DFG) im Rahmen der Exzellenzstrategie des Bundes und der Länder - EXC 2023 Internet of Production

[8] Neil, Drew. “Practical Vim: Edit Text at the Speed of Thought.” Pragmatic Bookshelf. (2015).

[9] Beutelspacher, Albrecht. “Cäsar oder Aller Anfang ist leicht!” Kryptologie (2020): 1-25.

[10] Jüptner, Patrick und Dalibor, Manuela und Trautwein-Bruns, Ute. “WordGuess - ein kollaboratives Spiel mit Git und GitLab”. DOI: https://doi.org/10.18154/RWTH-2021-04607.