

SquidWord

SWT 2020

Konrad Hanff, Jan-Eric Hellenberg, Tobias Kantusch, Felix Roth, Ole Schlüter



Motivation



Nachdem wir in SWA erfolgreich das Spiel “Volldampf” in GameDesign und Grafik entwickelt haben, war es Zeit für eine weiter Komponente: Story. Natürlich braucht es Charaktere, die etwas zu erzählen haben und eine tiefe Hintergrundgeschichte, um die Immersion zu erhöhen. Aber wie schreibt man so eine Geschichte auf? Natürlich in einem RichTextEditor.



Initialer Stand des Projekts

SquidWord

Save as... Save Load Undo Redo Document New

bold italic underlined footnote heading1 heading2 heading3

Mein wunderbarer Text

Lore ipsum dolor sit amet, consetetur sadipscing elitr, **sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.**

At vero eos et accusam et justo duo dolores et ea rebum.
Stet clita kasd gubergren, no sea takimata sanctus est

Lore ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Manage Structures



Initialer Stand des Projekts

SquidWord

Save as... Save Load Undo Redo Document New

bold italic underlined footnote heading1 heading2 heading3

Mein wunderbarer Text

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, **sed diam nonumy eirmod tempor** invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.
At vero eos et accusam et justo duo dolores et ea rebum.
Stet clita kasd gubergren, no sea takimata sanctus est
Lore ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Manage Structures

Structure Editor

bold italic underlined footnote heading1 heading2 heading3

bold italic underlined struckOut extendable

Font Reset font

Color Reset color

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam at fringilla est, eget suscipit ex. Interdum et malesuada fames ac ante ipsum primis in faucibus.

+ - Up Down

Als wir das Legacy Projekt zum ersten Mal öffneten, konnten wir schon einige Grundfunktionalitäten erkennen. Ein Text- sowie Struktureneditor waren bereits vorhanden und ermöglichten uns so einen guten Start in das Projekt. Hier fiel uns jedoch relativ schnell auf, dass sich das System an einigen Stellen nicht so verhielt wie wir es erwarten würden.



Funktionale Anforderungen

Funktionen, die den Einstieg vereinfachen

- Help Window

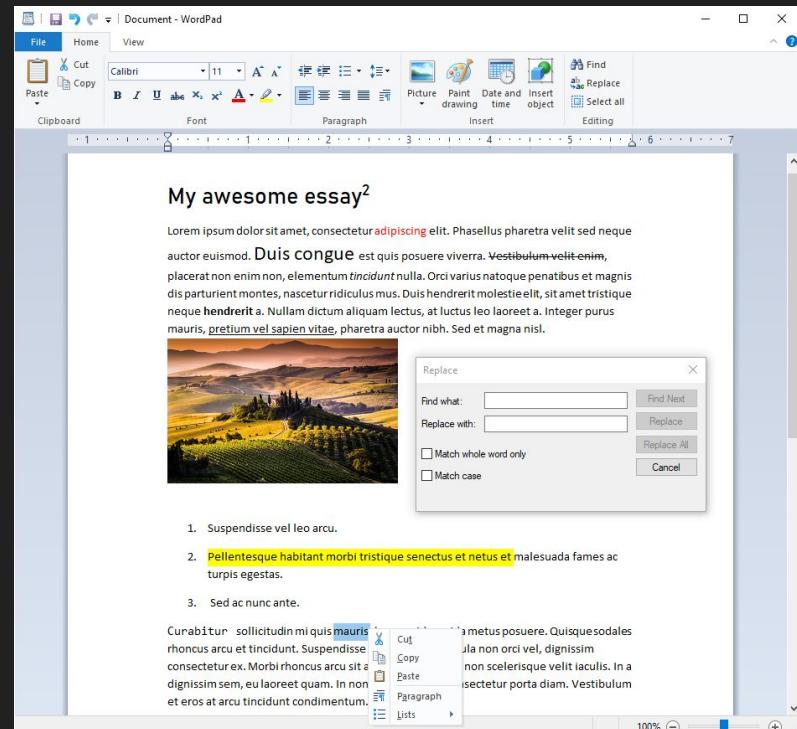
Funktionen für Power User

- Shortcuts

Zusätzliche Textinformationen

- Wortzahl, Zeilennummer, Cursor Position

API



Da wir Anfangs nicht direkt alle Funktionen entdeckt haben war es uns als Funktionale Anforderung wichtig, dass wir Funktionen bieten, die neuen Nutzern einen einfachen Einstieg ermöglichen. Dort haben wir ein Hilfefenster implementiert, welches die Funktionen des Editors erläutert.

Auch wollten wir aber den Workflow von erfahrenen Nutzern erleichtern und so Shortcuts einfügen um Strukturen einfach anwenden zu können. Zudem sollten verschiedene Statistiken über den Text sichtbar sein.

Ein weiterer wichtiger Punkt für Entwickler soll eine API sein. Es soll also die Möglichkeit geben mittels eines DocumentReadStreams den Text schrittweise programmatisch auslesen zu können. Dies spiegelt auch unsere Motivation mit dem Spiel dar, da dies das einfache programmatische Verarbeiten von Text ermöglicht.



Nicht funktionale Anforderungen

Product:

- Import/Export einer Datei mit 10.000 Zeichen dauert maximal 1s
- Beim Import/Export einer Datei geht kein Inhalt verloren
- *SquidWord verhält sich wie ein ähnlicher Rich-Text-Editor*

Organizational:

- Kein Teammitglied wird abgehängen
- Arbeitszeit einer Person beträgt nicht mehr als 40% der Gesamtarbeitszeit pro Sprint

Grundsätzlich wollten wir den Editor so gestalten, dass er sich ähnlich zu bekannten Editoren verhält und so einen einfachen Umstieg ermöglichen.

Als nicht funktionale Anforderungen war uns außerdem die Verlässlichkeit des Systems wichtig. Es soll nicht passieren, dass bei einem Import einer Datei Inhalt verloren geht. Ferner soll das Im-/Exportieren von 10.000 Zeichen in 1s durchgehen. Diese Metrik erschien uns als sinnvoll, da wir diese einerseits testen und andererseits so sicherstellen können, dass der Nutzer stets ein angenehmes Erlebnis hat.

Sehr wichtig waren uns aber auch organisatorische Anforderungen. Wir wollten sicherstellen, dass sich jedes Teammitglied wohlfühlt und so zu jedem Zeitpunkt versteht, wie der Code funktioniert und was die Überlegungen waren. Auch sollte sich kein Teammitglied so fühlen, als würde es die ganze Arbeit erledigen. Wir haben uns so darauf festgelegt, dass kein Teammitglied mehr als 50% der Gesamtarbeitszeit in einem Sprint haben soll.



User Stories: Features

Fußzeile

Autovervollständigung

Hilfsfunktion

Wortzahl

Zeilennummern

Shortcuts

Rechtsklickmenü

Import von
Dateien

Hyperlinks

Listen

Datei speichern

Markdown
Export/Import

Bilder

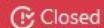
Zeilenabstand

Alignment

Lesestream

Hier haben wir die Funktionen, die in User-Stories vorgeschlagen wurden aufgelistet. In grün auf der linken Seite sind die bereits bearbeiteten Stories, während auf der rechten Seite die noch nicht bearbeiteten Stories dargestellt sind.

Shortcut support #61

[Edit](#)[New issue](#)

j-hellenberg opened this issue on May 3 · 0 comments

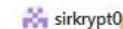


j-hellenberg commented on May 3 · edited by frcroth

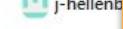
Member



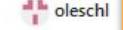
Assignees



sirkrypt0



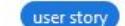
j-hellenberg



oleschl



Labels



user story



valid story



Projects



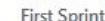
RichTextEditing



Done



Milestone



First Sprint



Linked pull requests

Successfully merging a pull request may close this issue.



Origin/us61

As an user with an preference for using the keyboard as much as possible, I want to be able to use shortcuts to assign structures in my text, in order to reduce the number of necessary mouse clicks and to increase my productivity.

(Note: This is an duplicate of issue #12, which was closed (most likely rejected back then) and I am unable to reopen it at this point).

Acceptance criteria:

- The first ten structures are assigned a shortcut from Ctrl+1 to Ctrl+0 (0=10 in this case).
- When hitting the shortcut in the Rich Text Editor, the same action is taken as if the corresponding structure was selected in the list of available structures
- If the structures ordering is changed, the shortcuts update accordingly

Estimated effort: 8



1



frcroth

added the [user story](#) label on May 4



sirkrypt0

assigned sirkrypt0, j-hellenberg and oleschl on May 4



sirkrypt0

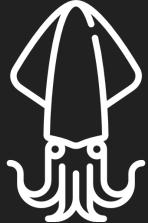
added this to the [Next Meeting](#) milestone on May 4

Aus Acceptance criteria
ergeben sich
Acceptance tests

Code Review findet in
Pull Request statt

User Story ist fertig
geschätzt und bereit
ausgewählt zu werden

Wir haben für User Stories das Github Feature der Issues verwendet. Dafür haben wir zunächst in der die Story als “As a ... , I want ... , in order to ... ” formuliert. Dann haben wir Akzeptanzkriterien formuliert und die Story gemeinsam im Team geschätzt. Auf den Kundentreffen wurden die Stories ausgewählt, dadurch wurden sie einem Milestone (jeder Sprint ein Milestone) zugewiesen. Danach haben wir die Stories auf Entwickler aufgeteilt und Feature-branches erstellt, die nach der Namensgebung “US61” funktionieren.

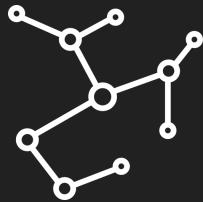


Demo



Backup-Video

1. Help Browser
2. Import Datei
 - a. Lustiger Text über Schienen/Züge
3. Text ändern
 - a. Auf Footer hinweisen
4. Styling
 - a. Über Menü
 - b. Über Shortcut
5. Neue Code Struktur mit Monospace Font + Farbe anlegen
 - a. Struktur anwenden
6. Speichern des Texts
7. Laden des Gespeicherten

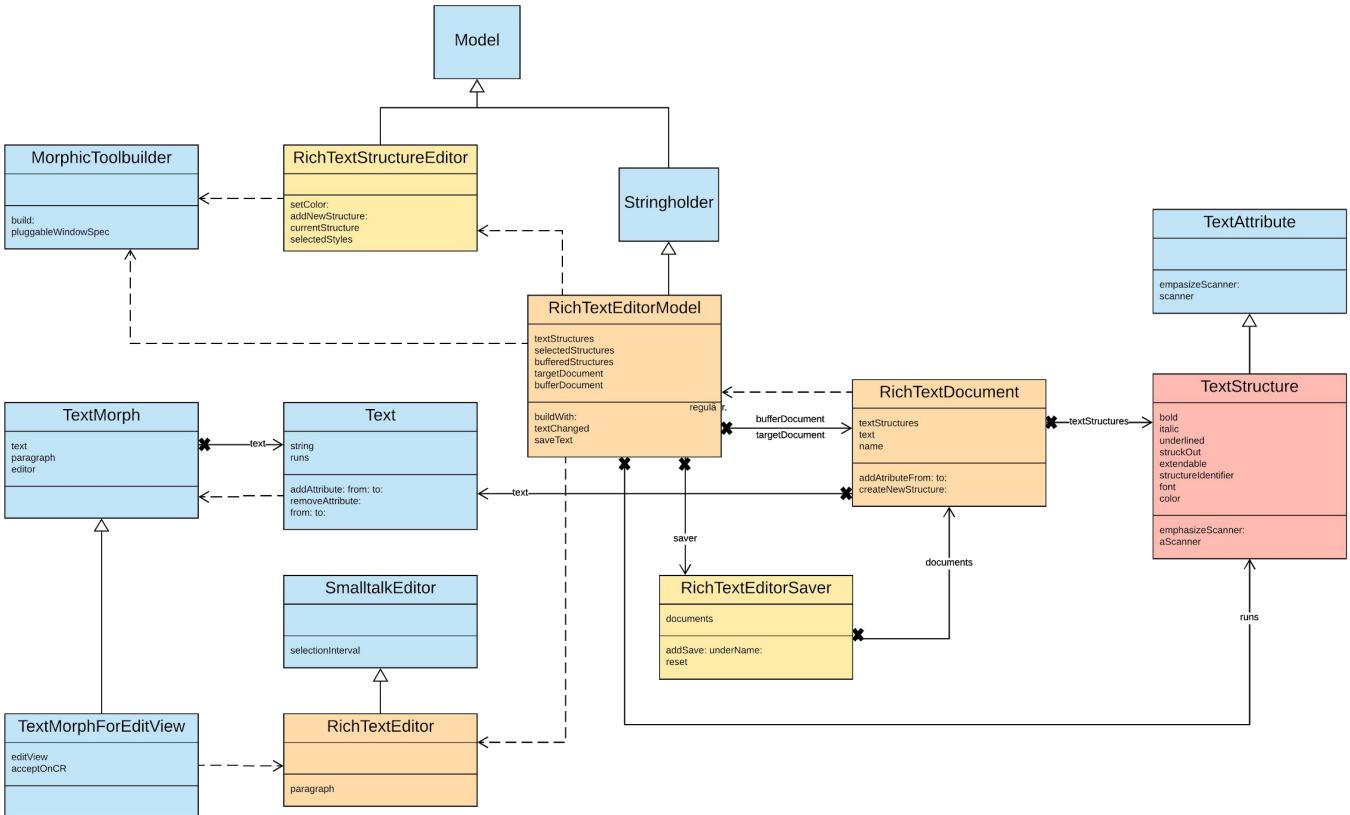


Architektur

Architektur: gegeben



unveränderte Klasse Veränderte Klasse Squeak





Shortcuts



Shortcuts

As an user with a preference for using the keyboard as much as possible, I want to be able to use shortcuts to assign structures in my text, in order to reduce the number of necessary mouse clicks and to increase my productivity.

(Note: This is a duplicate of issue [#12](#), which was closed (most likely rejected back then) and I am unable to reopen it at this point).

Acceptance criteria:

- The first ten structures are assigned a shortcut from Ctrl+1 to Ctrl+0 (0=10 in this case).
- When hitting the shortcut in the Rich Text Editor, the same action is taken as if the corresponding structure was selected in the list of available structures
- If the structures ordering is changed, the shortcuts update accordingly

Estimated effort: 8

Exemplarische User story: Da wir auf einem bestehenden System aufbauen, mussten wir gucken, wie man neue Features am besten einbaut. Der TextEditor kann bereits Shortcuts wie z.B. `ctrl+a`. Daher konnten wir uns daran orientieren. Anhand der User Story wird außerdem erklärt, wie die Komponenten des Systems zusammenarbeiten.



Shortcuts

- Gibt es bereits in Squeak (z.B CTRL + 1)
- Auch im TextEditor gibt es die Möglichkeit für Shortcuts

initializeShiftCmdKeyShortcuts

```
super initializeShiftCmdKeyShortcuts.  
self structureKeymap do: [:each |  
    shiftCmdActions at: each asciiValue + 1  
    put: #toggleStructureSelectionAt:]
```

SquidWord

File Document Help

bold
italic
underlined
footnote
heading1
heading2
heading3
list

Manage Structures

bold

italic

underlined

footnote

heading1

heading2

heading3

list

Manage Structures

You are using SquidWord · 100 words · Line 1 Col 2 · unsaved document

SquidWord

File Document Help

bold
italic
underlined
footnote
heading1
heading2
heading3
list

Manage Structures

bold

italic

underlined

footnote

heading1

heading2

heading3

list

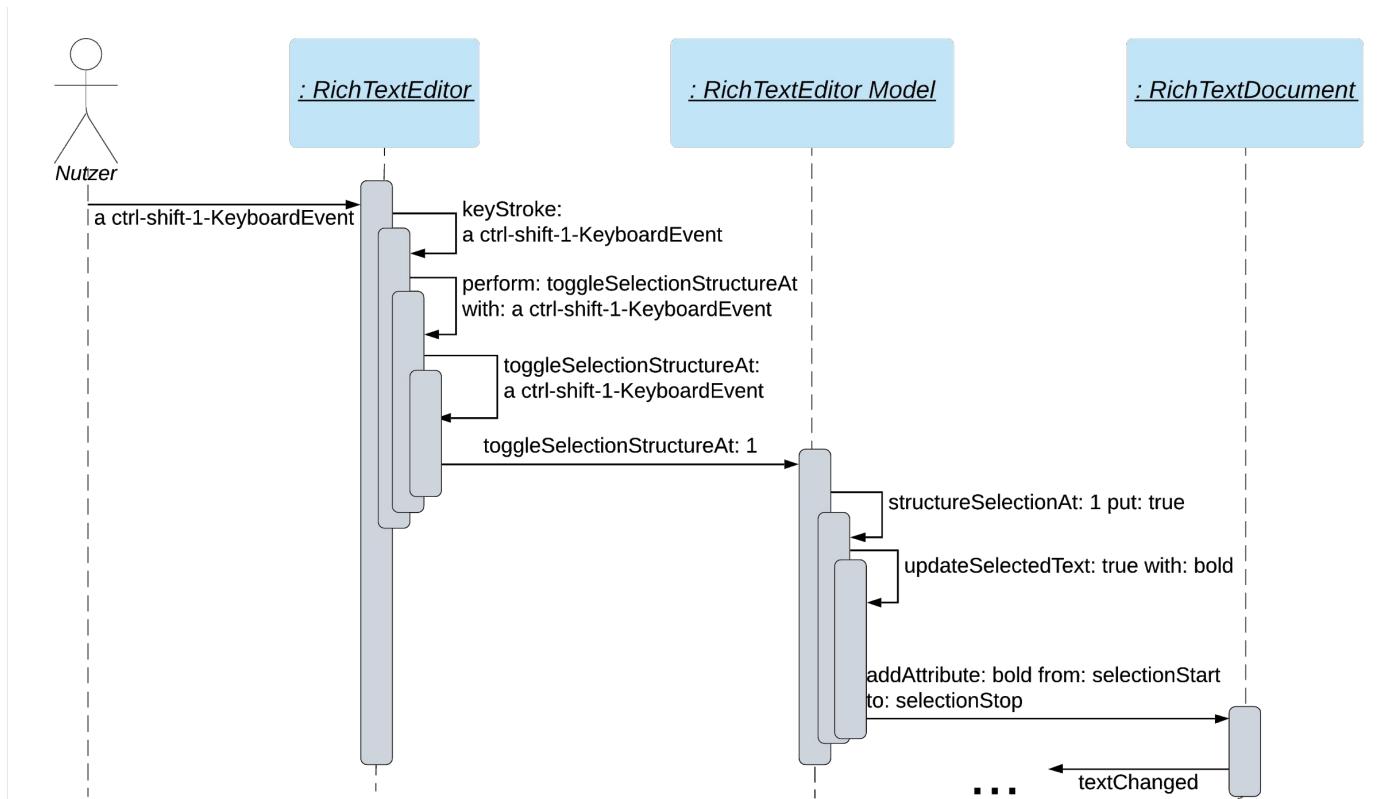
Manage Structures

You are using SquidWord · 100 words · Line 1 Col 2 · unsaved document



Strg + Shift + 1

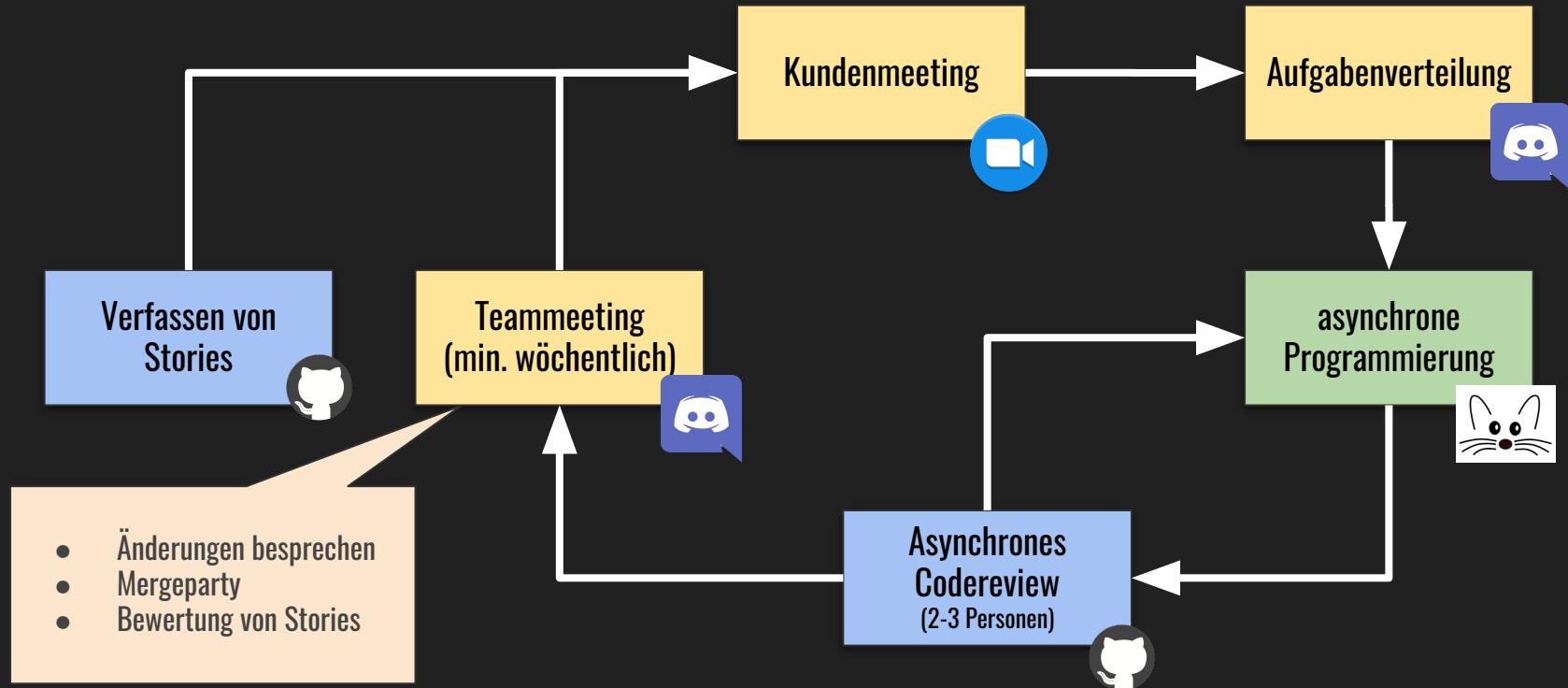
↔ Sequenzdiagramm





Entwicklungsprozess

Unser Entwicklungsprozess

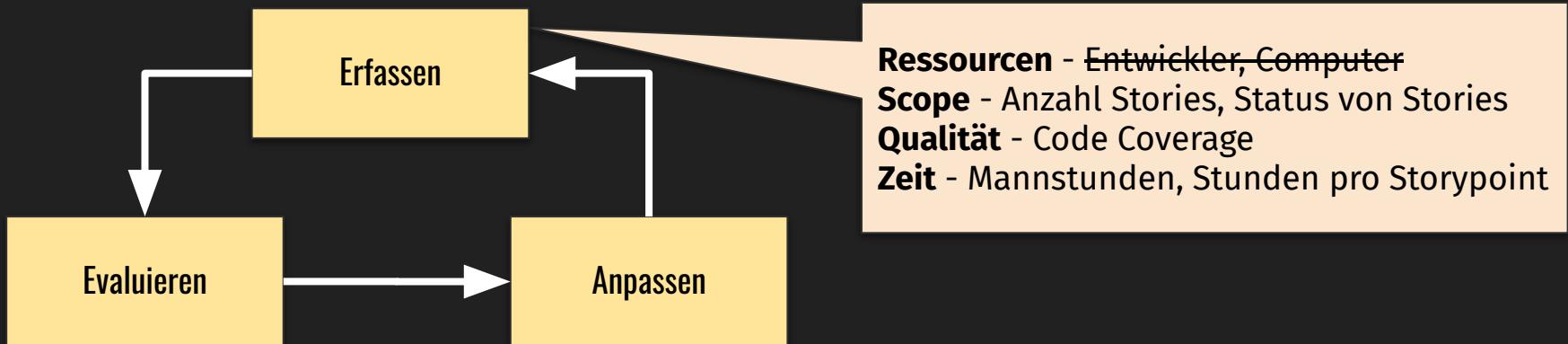


Unserem Entwicklungsprozess lag XP zu Grunde. Wir haben 2 wöchige Sprints gemacht, wobei in jeder Woche mindestens ein Teammeeting stattfand, an dem wir uns über unseren Entwicklungsstand, Probleme und weiteres ausgetauscht haben. Die Entwicklung selbst verlief einzeln, asynchron. Es wurde die Code Review Funktion bei Pull Requests von Github extensiv genutzt, hier haben dann meist 2-3 andere Personen die Änderungen betrachtet und Verbesserungsvorschläge gemacht.

User Stories wurden unabhängig als Github Issue verfasst, wie oben beschrieben.



Resources, Scope, Quality, Time



In der Praxis gibt es mehrere Anforderungen an den Entwicklungsprozess, welche schwer in Einklang zu bringen sind. So soll möglichst viel nutzbare Funktionalität für den Kunden produziert werden und die dabei geschriebene Software von möglichst hoher Qualität sein, gleichzeitig aber auch nicht zu viel Zeit aufgewendet werden, während die Ressourcen für das Projekt (Entwickler, Hardware, Geld) weitgehend konstant bleiben.

Wenn nun festgestellt wird, dass die Aufgaben nicht in der gesetzten Zeit fertiggestellt werden, gibt es mehrere Möglichkeiten, dem entgegenzuwirken:

- Erhöhter Zeitaufwand (ungünstig, führt zu Überarbeitung und eventuell managelnder Zeit für andere Projekte)
- Geringere Qualität (ungünstig, wird früher oder später zu Problemen führen, die bloß noch mehr Zeit in Anspruch nehmen)
- Geringerer Output (unbequem gegenüber dem Kunden, aber häufig die beste Option)

Der Sinn und Zweck von *Ressources Scope Quality Time* ist es nun, relevante Metriken über den Entwicklungsprozess zu erheben, um unerwünschte Trends (bspw. ein Teammitglied arbeitet wesentlich mehr als die anderen, die Testabdeckung sinkt [sinkende Qualität], oder Metaarbeit nimmt zu viel Raum ein [ineffizienter Prozess]) frühzeitig zu erkennen, um ihnen entgegen wirken zu können.



Resources, Scope, Quality, Time

Beginn der Woche	2020-04-20	2020-04-27	2020-05-04	2020-05-11	2020-05-18	2020-05-25	2020-06-01	2020-06-08	2020-06-15	2020-06-22
			Auto completion, Shortcut support	Line numbers, Wordcount, Lists, Default Values, Help Import, Import M: Unsaved warning				Line spacing, Alignment		
geschriebene User Stories	0	2	0	8	0	0	6	3	2	0
verteilte Storypoints (alle 2 Wochen)	0	0	11	11	20	20	21,5	21,5	20	20
Gesamtzeit für User Stories	0	0	7	6	10,5	9	8	10	4,5	0
Gesamtzeit für andere Programmierung	0	3	2	6	10	0,5	2	8	13	0
Gesamtzeit Meetings	0	1	4,5	4	7,5	4	7,5	4	14	1
Gesamtzeit Meta	0	3	1	4	6	2,5	5	8	10	0
Gesamtzeit	0	7	14,5	20	34	16	22,5	30	41,5	1
Stunden/Storypoint	0	0	0,6363636364	0,5454545455	0,525	0,45	0,3720930233	0,4651162791	0,225	0
Übertrag fehlgeschlagener US in nächste Woche	0	0	0	0	0	0	0	0	5	5
Anzahl Commits auf dev	0	0	0	2	7	21	44	52	112	112
Anzahl Pull Requests	0	0	0	0	1	3	10	10	23	23
Coverage auf dev	88	88	88	88	87	87	87	89	92	93
Anzahl vorhandene Tests	67	67	67	67	67	69	78	79	103	103
Anzahl durchlaufender Tests	67	67	67	67	67	69	74	75	103	103
Maximaler Anteil an Gesamtarbeitszeit	0,2857142857	0,3448275862		0,5	0,3529411765	0,3125	0,2666666667	0,2666666667	0,3614457831	1

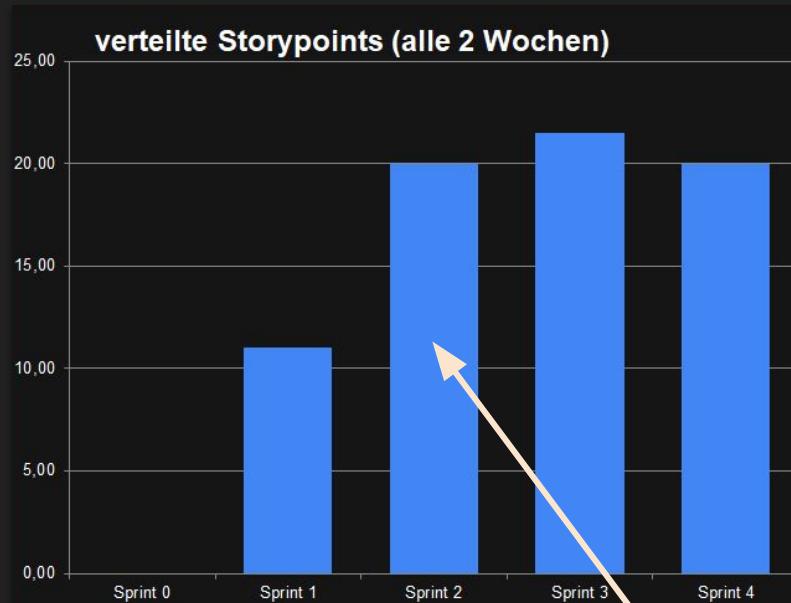
Tracking auf Google Docs (ohne Zeit pro Person)

Aus den Metriken konnten Schlüsse gezogen werden, dies wird in den nächsten Folien dargestellt.

Alle Daten sind auf dem Stand vom 25.6.2020 (dies war der Stichtag, bis zu dem sie noch in den Vortrag einbezogen wurden).



RSQT - Scope



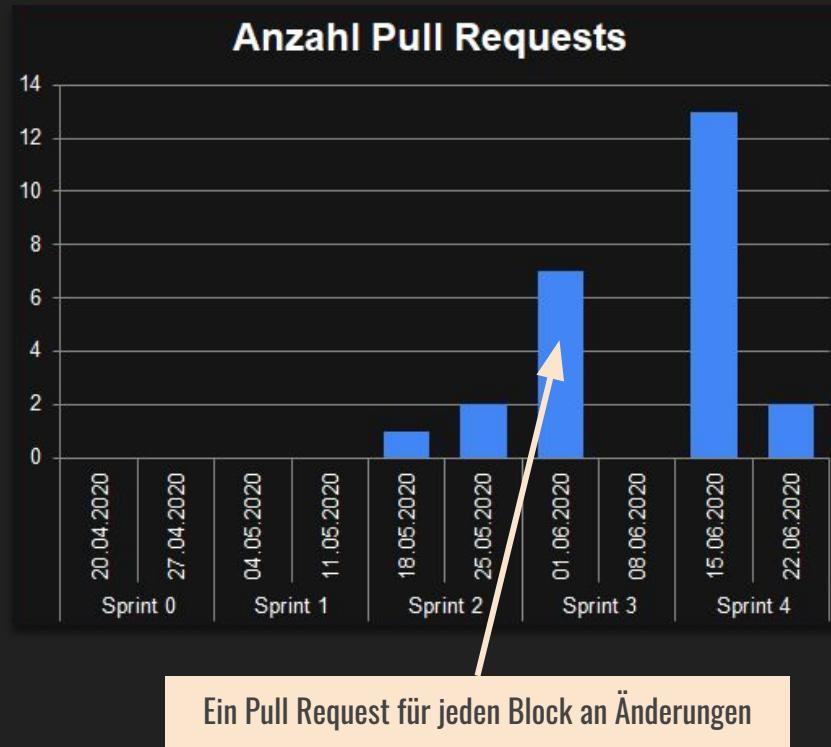
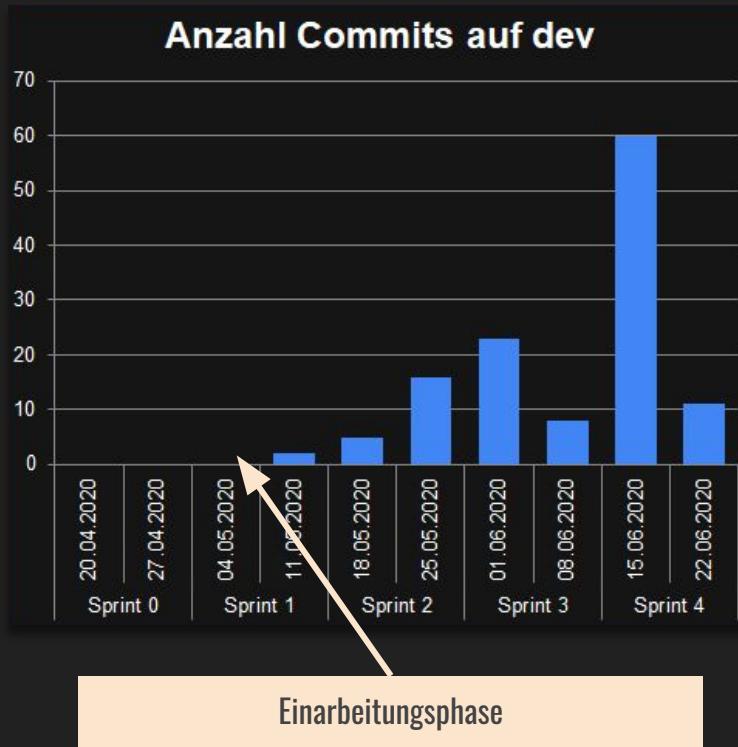
Anpassung Storypointbudget

Auf der linken Seite ist zu sehen, dass das Storypointbudget vom ersten auf den zweiten Sprint massiv anstieg. Dies liegt darin begründet, dass im ersten Sprint noch Zeit für Reverse Engineering aufgewendet werden musste.

Auf der rechten Seite ist zu sehen, dass die Anzahl an verfassten User Stories über die Zeit abnahm. Das liegt darin begründet, dass bereits viele User Stories angesammelt wurden, die teilweise auch schwieriger als die anfänglichen waren, wodurch kein großer Bedarf nach neuen Stories bestand.



RSQT - Output

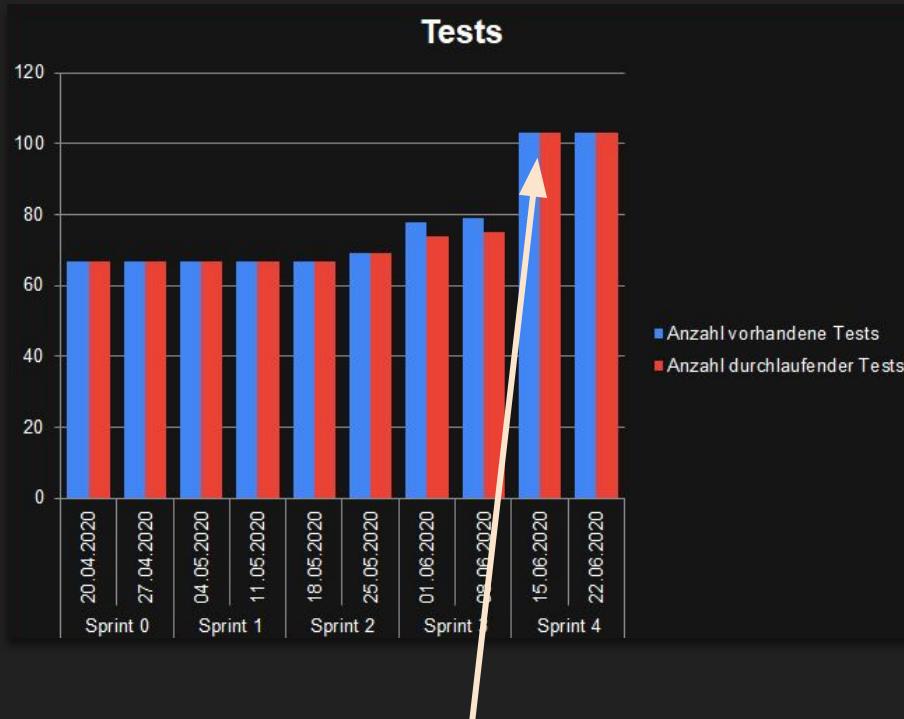


Auf der linken Seite ist die Anzahl der Commits zu sehen, die zu diesem Zeitpunkt auf dem *dev* Branch, unser “master”, waren. Auch hier sieht man gut das Reverse Engineering am Anfang. Die große Säule ist durch das Mergen von verschiedenen Featurebranches entstanden, in denen viele kleine Commits verwendet wurden.

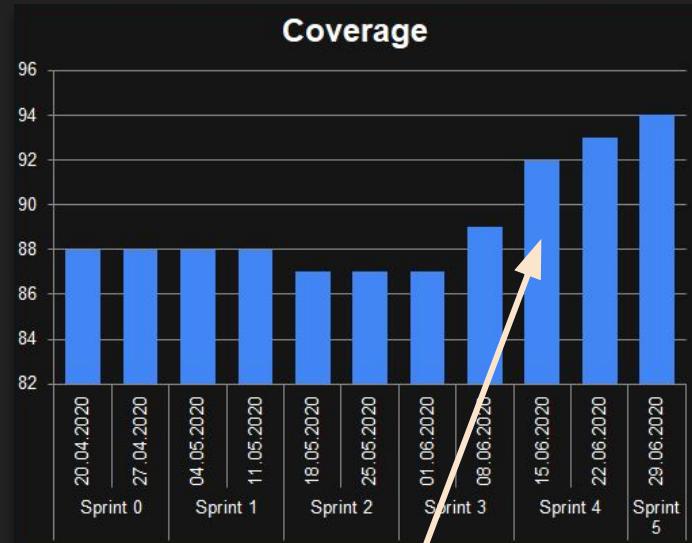
Auf der rechten Seite sind die Anzahl der gestellten Pull Requests zu sehen. Pull Requests wurden üblicherweise erst gestellt nachdem ein Feature oder Refactoring fertig entwickelt wurde und bereit zum Review war. Auch hier sind damit die Ende der Sprints zu erkennen, an denen die aktiven Branches dessen, gemerged werden sollten. Darauf gehen wir in der Reflexion nochmal näher ein.



RSQT - Output



Anforderung: 100% erfolgreiche Tests auf Dev-Branch



Unitests für noch ungetestete (Legacy-) Funktionalität

Auf der linken Seite ist die Anzahl der durchlaufenden mit der Anzahl an vorhandenen Tests verglichen. Wie auffällt wurden zunächst kaum Tests geschrieben. In dieser Zeit hatten wir auch noch nicht CI richtig konfiguriert, wodurch dies nicht sehr ins Auge fiel. Wir haben allerdings bald bemerkt, dass Tests notwendig sind und verlangt, dass fertige Features auch Tests dafür enthalten.

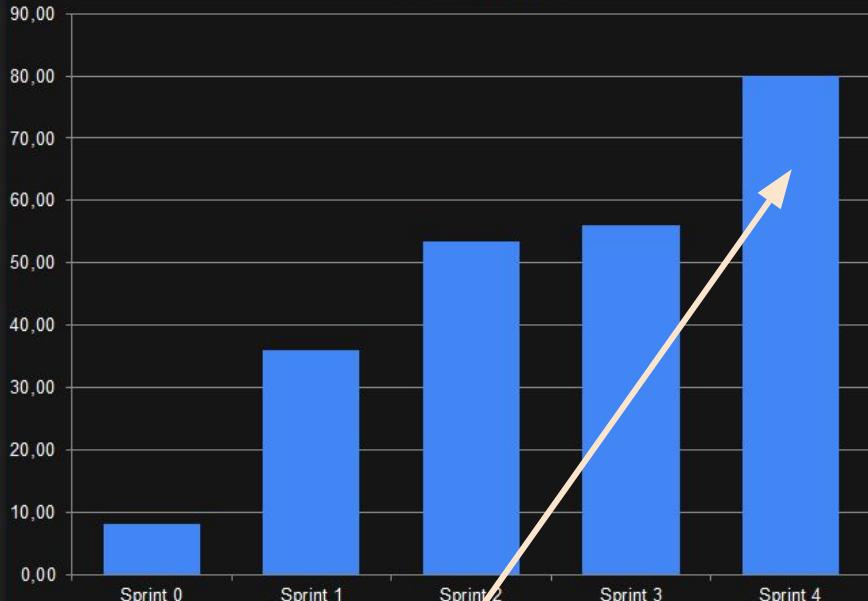
Zwischenzeitlich wurde die Funktionalität von Kernaspekten des Editors angepasst, worauf hin einige Tests nicht mehr funktionierten. Nach einem Kundentreffen, bei dem wir darauf hingewiesen wurden, haben wir auf unserem *dev* Branch keine fehlschlagenden Tests mehr erlaubt.

Im späteren Verlauf wurde Coverage für uns eine relevante Metrik, weshalb wir auch für bereits vorhandes, aber ungetestetes Verhalten anfingen Unitests zu schreiben.



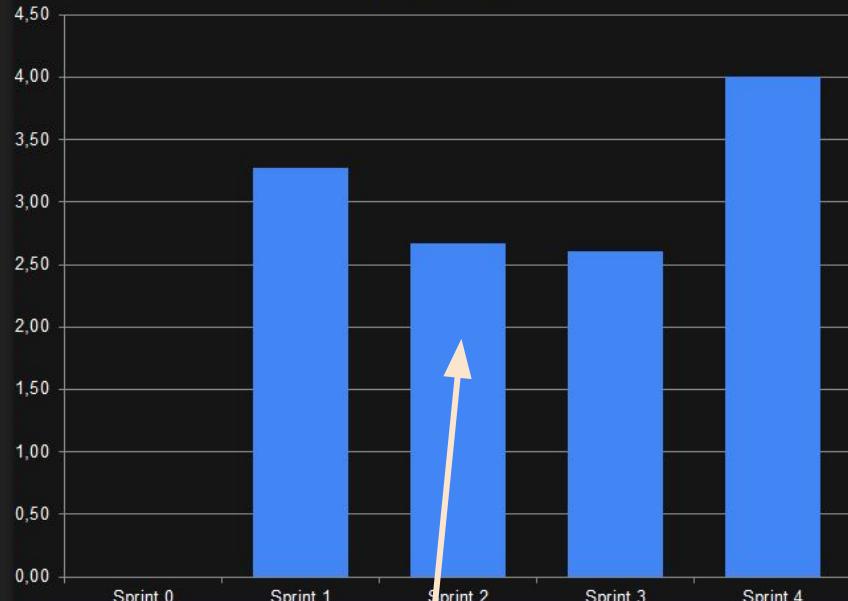
RSQT - Arbeitszeit

Gesamtzeit



Großes Refactoring und Vortragsvorbereitung

Stunden/Storypoint



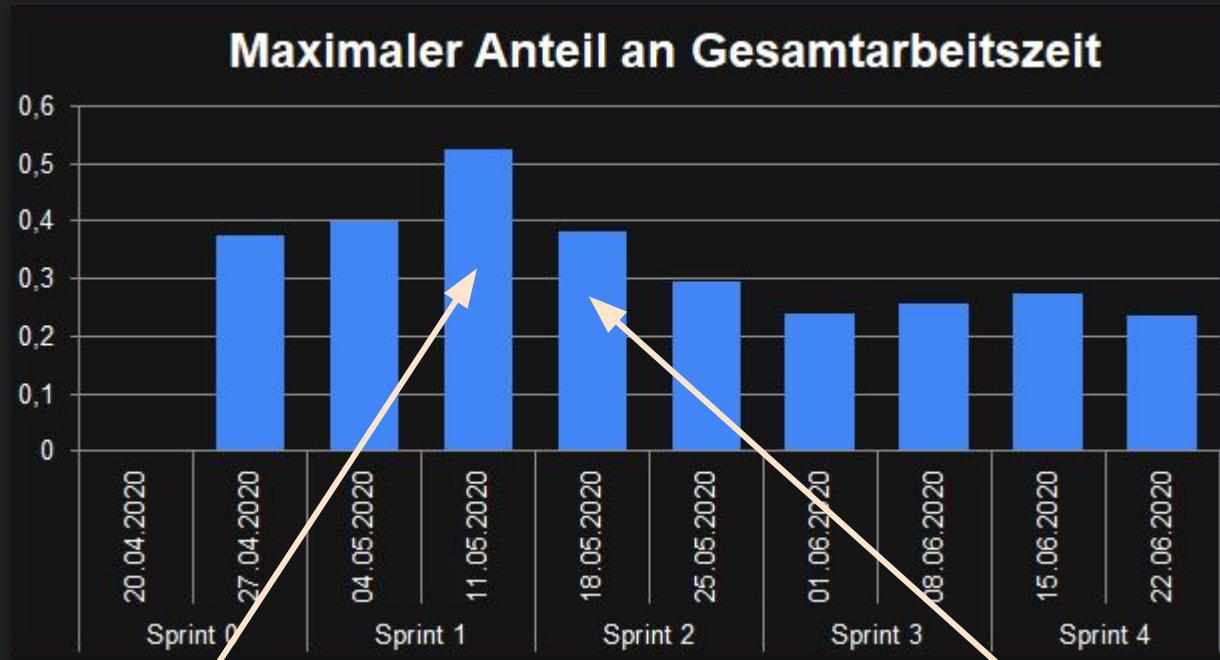
Einarbeitung in das System und Budgetanpassung

Im linken Diagramm ist die Gesamtarbeitszeit, die wir auch weiter aufgeschlüsselt gemessen haben, pro Sprint verzeichnet. Die Arbeitszeit von Sprint 1,2 und 3 war relativ ähnlich, während Sprint 4 wesentlich aufwendiger war. Wie später gezeigt lag das jedoch vor allem an der Vortragsvorbereitung. Da wir hier alle zusammen an den Vorträgen gearbeitet haben, sind aus z.B. 4 Stunden Vortragsvorbereitung 20 Mannstunden geworden.

Im rechten Diagramm ist zu erkennen, dass das Verhältnis von Storypoint zu Zeit im Verlauf des Projekts recht konstant war.



RSQT - Anteilmäßige Arbeitszeit



! Eine Person hat mehr als 50% der Gesamtarbeitszeit !



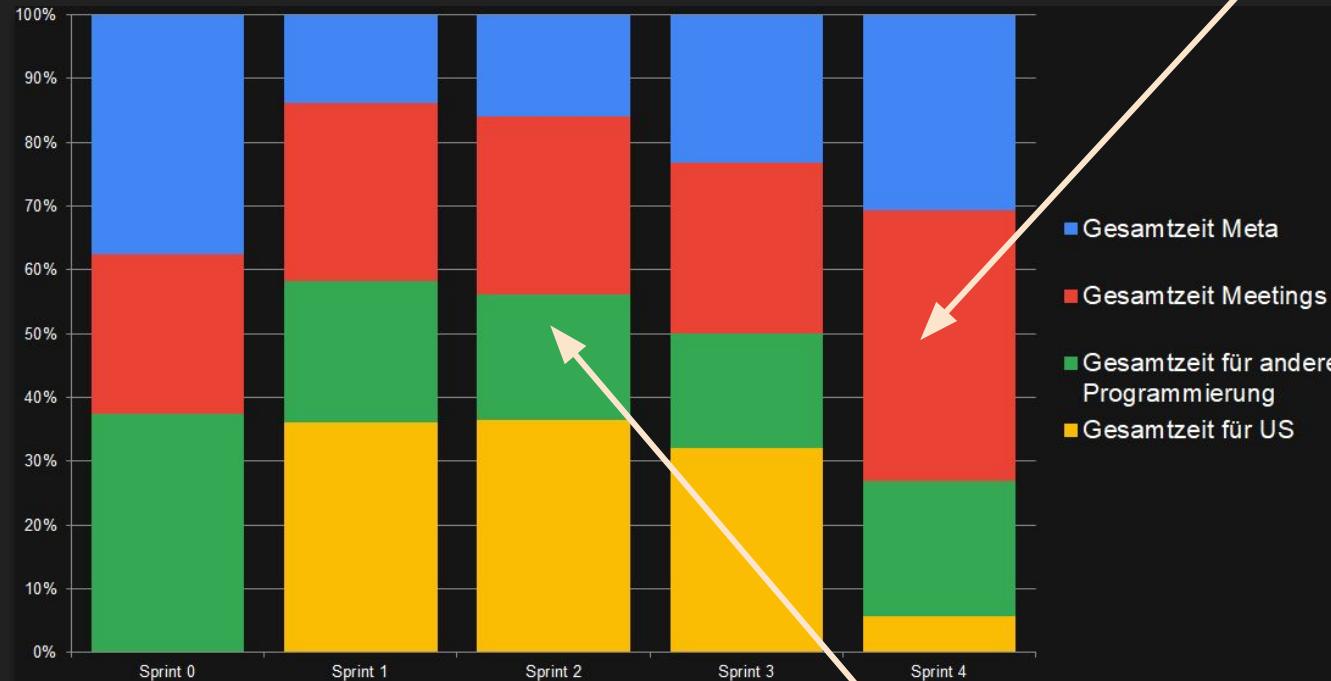
Besprechung im Team | gleichmässiger Aufteilen

Zu Beginn stellten wir fest, dass die Arbeit im Team noch recht ungleich verteilt war und einzelne Personen einen recht hohen Anteil der Gesamtarbeitszeit beisteuerten. Dem haben wir versucht entgegen zu wirken und die Arbeit gleichmäßiger zu verteilen, indem die Stories anders verteilt wurden und Refactorings nicht vornehmlich von bereits belasteten Personen durchgeführt werden sollten. Dies hat dazu geführt, dass in den letzten Wochen der maximale Anteil eines Teammitgliedes in unserem 5er-Team relativ konstant bei nur noch akzeptablen ca. $\frac{1}{4}$ lag.



RSQT - Arbeitszeitverteilung

Großes Refactoring und Vortragsvorbereitung



Learning: Anhaltend hoher Zeitaufwand, der nicht direkt mit Programmierung der User Stories zusammenhängt

In dieser Grafik ist die Arbeitszeit in verschiedene Kategorien aufgeteilt:

- Meta beschreibt Stories schreiben, Pull Requests, Metriken erfassen und ähnliches.
- Andere Programmierung beinhaltet neben Refactoring und Schreiben von zusätzlichen Tests für bereits bestehende Funktionalität auch Reverse Engineering.
- Meetings sind alle synchronen Aktivitäten mit mehreren Personen, also insbesondere Team-Meeting, Kundentreffen und Vortragsvorbereitung.

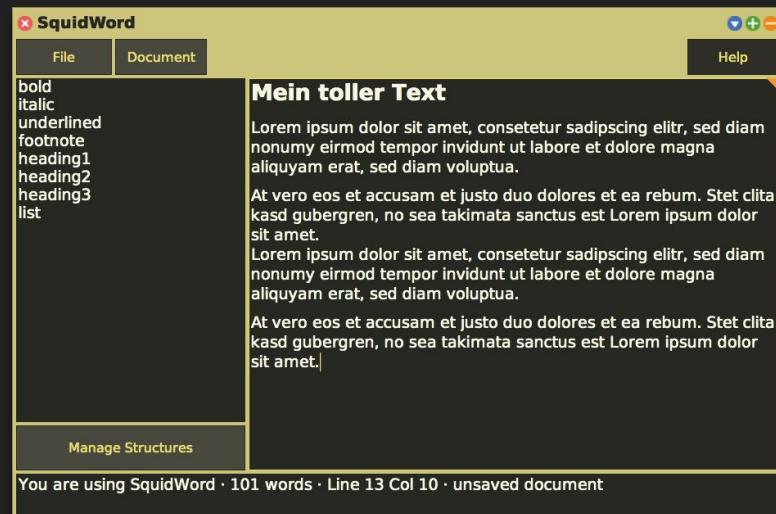
Das Tracken der anteiligen Arbeitszeit für einzelne Arbeitsbereiche brachte die Erkenntnis, dass nur gut $\frac{1}{3}$ der Zeit für tatsächliche Programmierung aufgewendet werden kann und der Rest der Zeit u.a. für Organisation, Einarbeitung in das System bzw. die Squeak-Umgebung und Sicherstellung der Code-Qualität aufgewendet werden muss.

Dies ist bei Abschätzung des Zeitaufwandes für neue User Stories zu berücksichtigen.

Acceptance Testing

67 Acceptance Tests bereits vorhanden

- testUS18SelectionStyleIsAppliedToSelection
- testUS18TextStyleIsSelectedInList
- testUS32StructureIsRendered
- testUS44OldButtonsHaveBeenRemoved
- testUS44SelectedStructureIsAssigned
- testUS44SelectedStylesChangeAccordingToSelectedText
- testUS44StructureListExists



User Interaktion erfolgt bisher
ausschließlich durch die UI

Acceptance Testing

testUS<IssueNumber><NameInCamelCase>

frcroth commented 20 days ago • edited

Member

As a user who edits texts with the tool, I want to be able to load files from my computer's file system into SquidWord in order to edit them in my favorite environment, which allows me to use structures.

This is a sub-issue of #5

Acceptance criteria:

- there is an option to select a file from the user's computer
- when a file is selected the file is opened in SquidWord as raw text

Estimated Effort: 7

testUS84LoadFileButtonExists

```
self assert:  
    (self menuItemWithLabelExists:  
        'Import from File')
```

testUS84LoadFileFromFilenamesystem

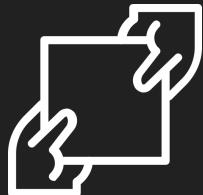
```
self editorModel loadFileFrom:  
    self exampleFilePath.  
  
self assert: self fileContent  
    equals: self modelText string
```



Acceptance Testing - Fazit

- Testet lediglich die Anforderungen der User Story, jedoch nicht darüber hinaus
 - Teilweise gab es Bugs, die nicht durch die Acceptance Tests abgedeckt wurden
- Keine vollständige Code Coverage (Vorgänger Gruppe hatte 88%)
- Um den Code besser zu testen eignen sich zusätzliche Unit-Tests

⊕ Weitere Praktiken



Collective Code Ownership

- Durchgesetzt durch Storyverteilung



Small Releases

- Abgewandelter “Git Flow”
- 2 wöchige Releases
- Benachrichtigungen durch Github “Watch for Releases”



Planning Game

- Estimation mit Telegram Umfrage
- Ausgeglichene Verteilung auf Personen
- Evaluation durch Metriken

⊕ Weitere Praktiken



Coding Standards

Styleguide

In this document the group of 2020 has formulated the idioms and general styling decisions which should be adhered to while coding on the RichTextEditor project.

Methods

Style methods like:

```
methodName: aParameter
"Here is a comment stating some useful information"

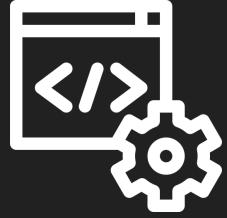
self doSomething.
^ returnValue
```

Naming

Indicate the type of parameter with the parameter name, use *a* and *an* if applicable. Use `camelCase` when naming methods.

When using multiple parameters, make sure to name the method in a way that indicates which kind of parameter should follow.

Cascades



Refactoring

Nach einiger Zeit stellten wir fest, dass die Art und Weise, wie die Textstrukturen in dem System umgesetzt waren, nur schlechte Erweiterbarkeit bietet und zudem nur mäßig gut für die Umsetzung neuer Funktionen wie bspw. Links und Listen geeignet war. Ferner wurde hier Teilweise der Code aus bereits vorhandenen Text Attributen wie bei `emphasizeScanner`: `aScanner` zu sehen lediglich kopiert.

Deshalb beschlossen wir, die Klassen umzubauen, und die einzelnen Aspekte statt durch Instanzvariablen über ein Composite-Pattern darzustellen.

Da dies jedoch in einem Schritt durchgeführt wurde, war ein einzelner riesiger und recht unübersichtlicher Pull-Request die Folge.

Dies brachte uns zu der Erkenntnis, dass ein so großes Refactoring besser in kleinere zusammenhängende Teile mit jeweils einem Merge-Request aufgesplittet werden sollte, sodass sich die Arbeit besser aufteilen lässt und das reviewen der Änderungen leichter und übersichtlicher wird.



Refactoring

TextStructure

- identifier
- bold
- italic
- underlined
- struckOut
- extendable
- color
- font
- styleSet

initialize

```
super initialize.  
self  
    unsetBold;  
    unsetItalic;  
    unsetUnderlined;  
    unsetStruckOut;  
    unsetExtendable;  
    color: nil;  
    font: nil
```

Erweiterbarkeit?



Refactoring

TextStructure

- identifier
- bold
- italic
- underlined
- struckOut
- extendable
- color
- font
- styleSet

Erweiterbarkeit?

updateStyles: aSet

```
(aSet includes: #bold)
    ifTrue: [self setBold]
    ifFalse: [self unsetBold].

(aSet includes: #italic)
    ifTrue: [self setItalic]
    ifFalse: [self unsetItalic].

(aSet includes: #underlined)
    ifTrue: [self setUnderlined]
    ifFalse: [self unsetUnderlined].

(aSet includes: #struckOut)
    ifTrue: [self setStruckOut]
    ifFalse: [self unsetStruckOut].

(aSet includes: #extendable)
    ifTrue: [self setExtendable]
    ifFalse: [self unsetExtendable].
```

self styleSet: aSet



Refactoring

TextStructure

- identifier
- bold
- italic
- underlined
- struckOut
- extendable
- color
- font
- styleSet

```
emphasizeScanner: aScanner
    aScanner addEmphasis: self emphasisCode.
    self color ifNotNil: [
        aScanner textColor: self color].
    self font ifNotNil: [
        aScanner setActualFont: self font]
```

Duplikat?



Refactoring

TextStructure

- identifier
- bold
- italic
- underlined
- struckOut
- extendable
- color
- font
- styleSet



Erweiterbarkeit



**Kombination aus
existierenden Strukturen**



**Code identisch zu
existierenden Strukturen**



Refactoring

TextStructure

- identifier
- bold
- italic
- underlined
- struckOut
- extendable
- color
- font
- styleSet

Kombination aus
existierenden Strukturen



Composite Pattern



Refactoring

TextStructure

- identifier
- attributes
 - ~~bold~~
 - ~~italic~~
 - ~~underlined~~
 - ~~struckOut~~
 - ~~extendable~~
 - ~~color~~
 - ~~font~~
 - ~~styleSet~~

TextStructure>>add:aTextAttribute

```
(self attributes includes: aTextAttribute)  
    ifTrue: [self remove: aTextAttribute].  
self attributes add: aTextAttribute
```



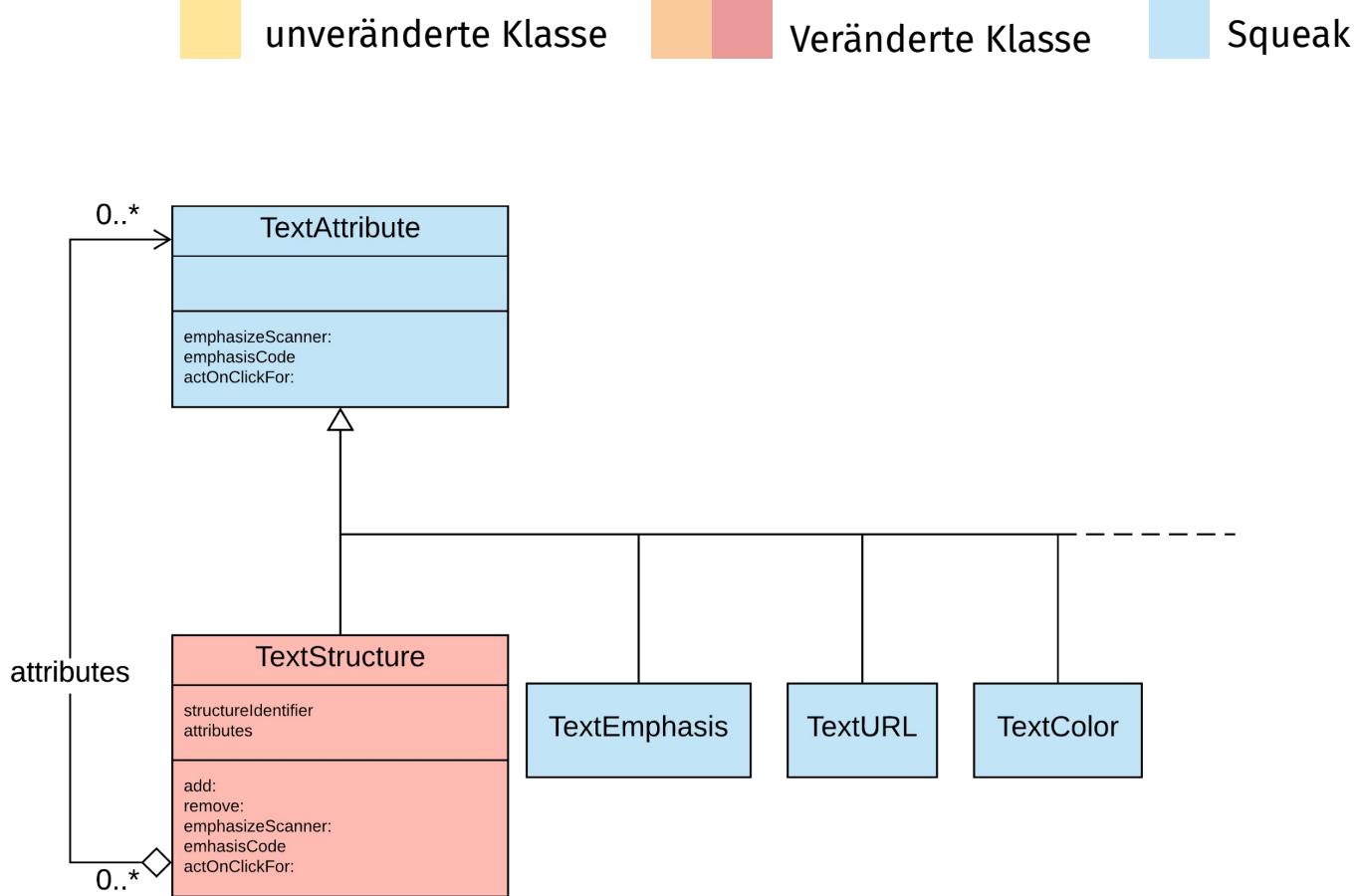
Refactoring

TextStructure

- identifier
- attributes
- ~~bold~~
- ~~italic~~
- ~~underlined~~
- ~~struckOut~~
- ~~extendable~~
- ~~color~~
- ~~font~~
- ~~styleSet~~

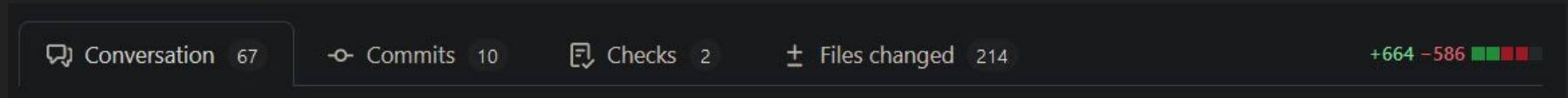
```
TextStructure>>emphasizeScanner:aScanner
self attributes do: [:each |
    each emphasizeScanner: aScanner]
```

Architektur: angepasst





Refactoring - Fazit



60+ Kommentare

200+ Files changed

+664 -586 Zeilen

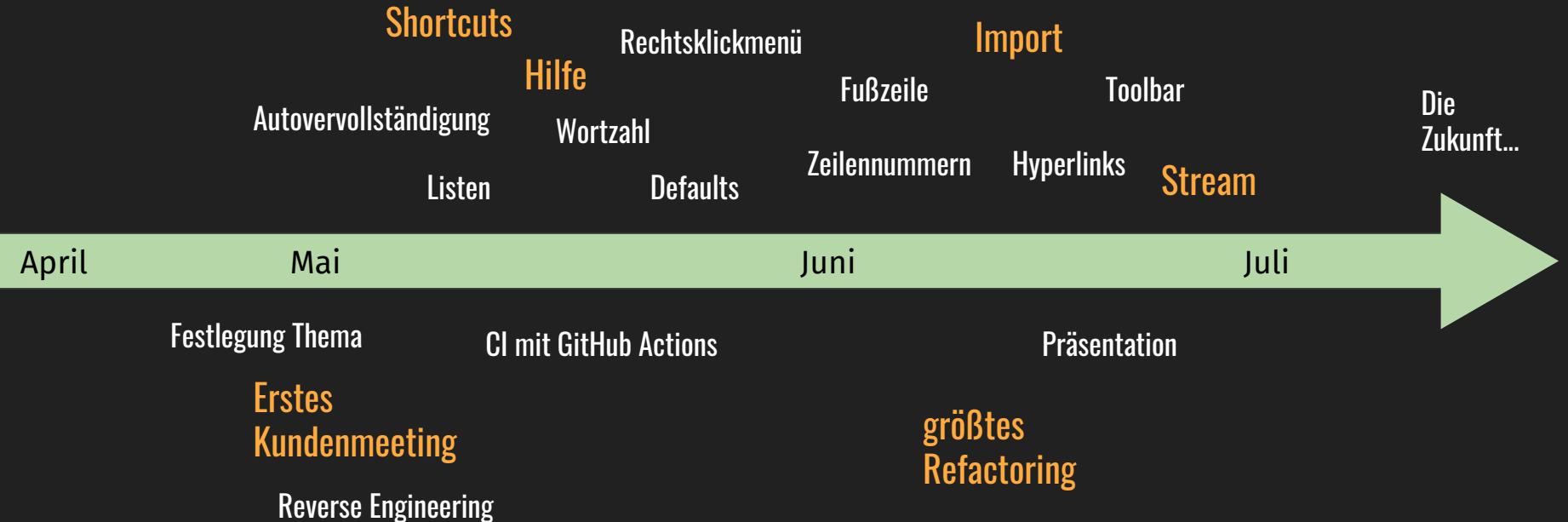
“Took me 1h just reading through all changes, lol”
- Anonymer Tintenfisch



Fazit



Zeitstrahl



Auf dem Zeitstrahl betrachtet können die Userstories hier nach Zeit eingeordnet werden. Außerdem werden weitere Aufgaben erfasst, die bereits vorher erwähnt wurden, wie die CI und das Reverse Engineering.



Zukunft



Features

- + Import/ Export in verschiedene Formate
- + API Funktionen weiter ausbauen
- + Textfeatures: Alignment, Listen, ...



Weiteres

- + Dokumentation (GitHub Wiki oder Readme) erweitern

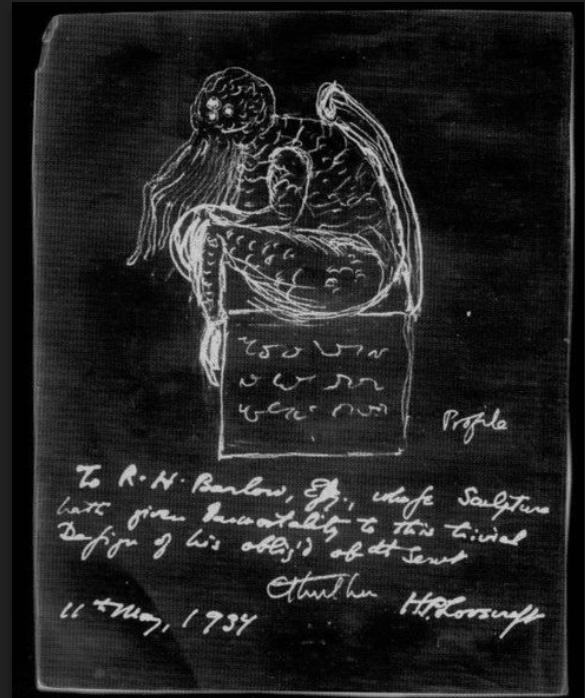
Bisher gibt es für das Projekt noch keine Wiki und das Readme wurde nicht stark erweitert. Wir möchten das ändern um unseren Nachfolgern den Einstieg zu erleichtern.



Reflexion

Was kann verbessert werden?

- Chaotischer Merge-Prozess vor Kundenmeeting
 - **Problem:** Zeit für Review und Tests wurde unterschätzt
 - **Lösung:** Features früher fertigstellen
- Pair Programming wurde nicht eingesetzt
 - **Gefahr:** Qualität des Codes leidet
 - **Lösung:** Ausgiebige Code Reviews
- Refactoring in zu großen Schritten
 - **Problem:** Hoher Aufwand beim Reviewen
 - **Lösung:** Refactoring eher langsamer durchführen

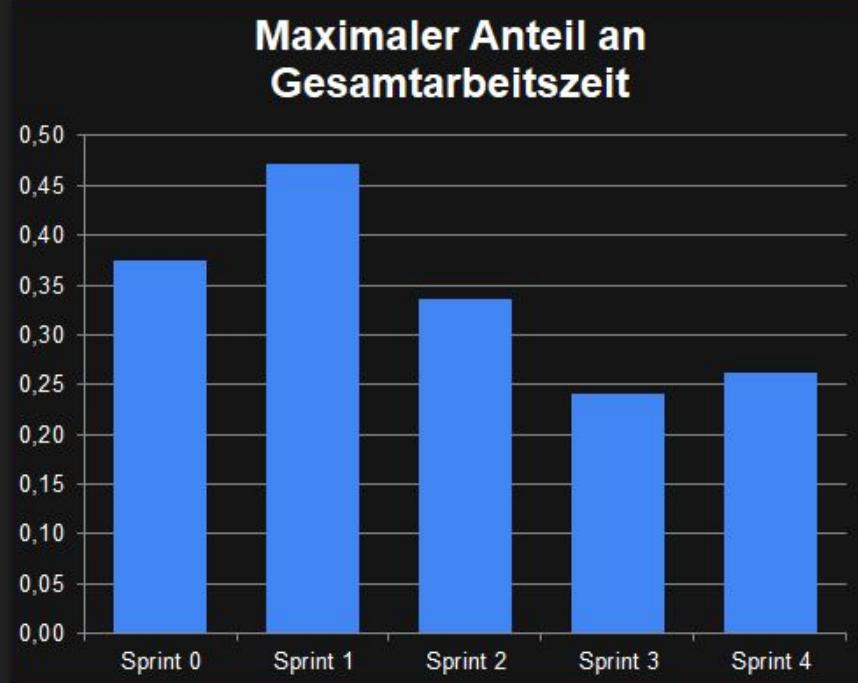




Reflexion

Was lief gut?

- Gut anpassbares Projekt
 - Kurze Einarbeitungszeit
 - Schnelle Featureeinarbeitung
- wöchentliche Treffen, offene Diskussion von Entscheidungen
- Features fast immer vollständig fertiggestellt
- Storypunkt-Zeit-Verhältnis relativ konstant
- Meist gut ausgeglichene Arbeitslast





Bewertung

Das Projekt eignet sich für die Weiterentwicklung in SWT 2021



Ende



Anhang



Bildquellen

- Target, Lightbulb, List, Stop watch icons made by Good Ware
- Gears icon made by Becris
- Speedometer, Additional, Goal, Question mark, Tools, Exclamation mark, Truck, Video, Code, Thumbs up, Thumbs down, Key, Rail tunnel, Medal, Podium, Requirements, Find, Squid, Refactoring, Future, Finger icons made by Freepik
- Left-right icon made by Becris
- Cross, Tiles, Share icons made by Pixel perfect
- Train, Acceptance, Flag icons made by Smashicons
- Transformation icon made by geotatah
- Puzzle, Innovation (Rocket) icon made by Icongeek26
- Wagon icon made by photo3idea_studio
- Hammer icon made by Nhor Phai
- Start line, Time icons made by Surang
- Shortcut icon made by Swifticons

(All from www.flaticon.com)