



# Animations-Editor

Jessica Ziegler, Tim Kuffner, David Matuschek, Lukas Hüller

Group 15 | Joana Bergsieck  
SWT | Summer Term 2020  
Software Architecture Group

# Fahrplan

**Anforderungen an das Softwaresystem**  
"Ich als Entwickler fand es super blöd, die für mein SWA-Spiel benötigten Animationen in einem externen Tool erstellen zu müssen. Das soll auch in Squeak gehen."

- Präsentation der Software
- funktionale Anforderungen
- nicht-funktionale Anforderungen

**Architektur des Softwaresystems**

- Gesamtarchitektur
- Architektur des AnimationsEditors
- Arbeit mit den bestehenden Systemen

**Disclaimer:** Nicht alle Aspekte der nachfolgenden Folien sind bereits im Code umgesetzt.

**Entwicklungsprozess**

- Ablauf eines Sprints
- angewendete Praktiken
  - Sustainable Development
  - Planning Game
  - Prototyping

**Metriken und Projektverlauf**

- zeitlicher Projektverlauf
- Stand des Projektes

**Zusammenfassende Bewertung**

- Reflexion des Projektes
- Reflexion der Teamarbeit

**SWT 1 -  
AnimationsEditor**  
Gruppe 15

Chart 2

The image is a composite of two screenshots. The left side shows a presentation slide with a red background. The title is "Anforderungen an das Softwaresystem". Below it is a quote: "Ich als Entwickler fand es super blöd, die für mein SWA-Spiel benötigten Animationen in einem externen Tool erstellen zu müssen. Das soll auch in Squeak gehen." A list of requirements follows: Präsentation der Software, funktionale Anforderungen, nicht-funktionale Anforderungen. The right side shows the Unity Editor's Scene View, Project, and Inspector panes. The Scene View displays a 3D scene with a red bus and a camera. The Project pane lists assets like BlueLight, PurpleLight, Camera, FPSCounter, Sponza, DirectionalLight, SettingsGUI, AnimationPlayer, and Scene Root. The Inspector pane is focused on a Camera component, showing settings for Keep Aspect, Cull Mask, Environment, H Offset, V Offset, Doppler Tracking, Projection, Current, Fov, Near, Far, and Spatial.

# Anforderungen an das Softwaresystem

"Ich als Entwickler fand es super blöd, die für mein SWA-Spiel benötigten Animationen in einem externen Tool erstellen zu müssen. Das soll auch in Squeak gehen."

- Präsentation der Software
- funktionale Anforderungen
- nicht-funktionale Anforderungen

Length (s): 15  
Step (s): 0.1

https://i.ytimg.com/vi/gSzJmG6YoDg/maxresdefault.jpg

Scene Import

+ Filter nodes

- BlueLight
- Particles
- PurpleLight
- Particles
- Camera
- FPSCounter
- Sponza
- DirectionalLight
- SettingsGUI
- AnimationPlayer
- Camera
- Camera2
- Camera3
- Camera4
- Scene Root

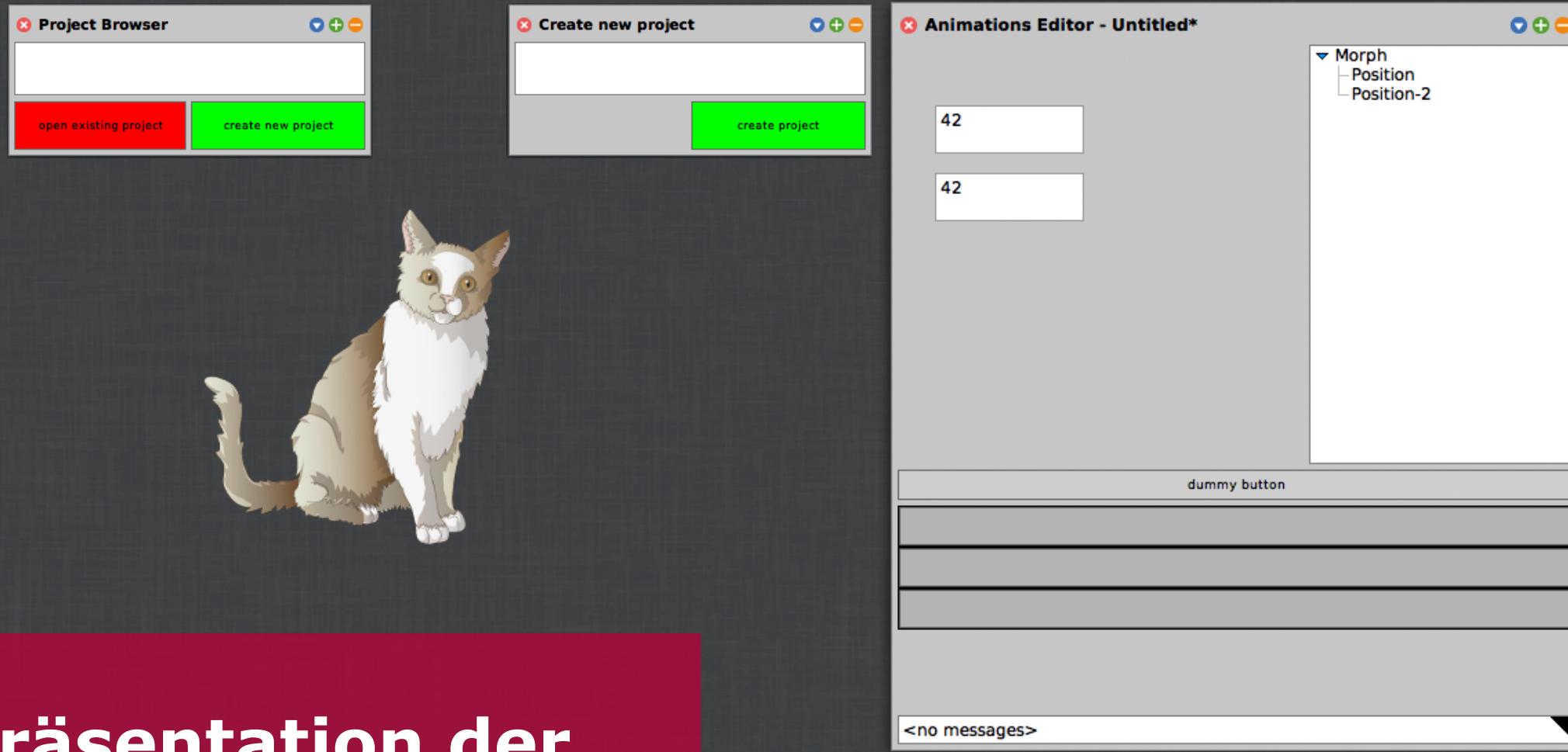
Inspector Node

Camera

Keep Aspect	Keep Height
Cull Mask	sponzatres
Environment	0
H Offset	0
V Offset	0
Doppler Tracking	Disabled
Projection	Perspective
Current	<input checked="" type="checkbox"/> On
Fov	70
Near	0.05
Far	100

Spatial

Transform



# Präsentation der Software

# Funktionale Anforderungen

Grundidee: AnimationsEditor bietet **UI** für das Package **Animations-Core**

**Kundenwünsche:** Der AnimationsEditor ermöglicht ...

- ... das **Öffnen** eines Morphs im AnimationsEditor
- ... das **Konfigurieren** von **verschiedenen Animationen**
- ... das **Abspielen** von Animationen (Play & Stop)
- ... die **Visualisierung** des **Animationsablaufs** (Timeline)
- ... die Anzeige von **Eigenschaftsspuren** in der Timeline
- ... die **Generierung** eines **Animationsskripts**, das in Squeak eingebunden werden kann

## Animations Editor - Details

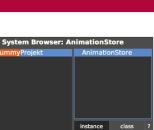
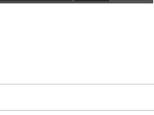
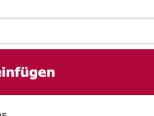
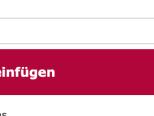
- Animations make games lively
- Designing animations might require many iterations, when defining the animations solely through code the feedback loop is rather long, even in Squeak
- A live animations editor could:
  - Provide live feedback on the animation by playing the animation in a loop while editing the animation
  - Provide a direct manipulation interface for defining the animation instead of writing code
  - Generates animation scripts and can be used to edit existing animation scripts

**SWT 1 -  
AnimationsEditor**

Gruppe 15

Chart 5

# Übersicht aller User Stories

<p><b>Erstellen &amp; Benennen einer Animation</b></p> <p>Ich als Nutzer würde gerne Animationen mit Namen versehen können, damit ich sie wiedererkennen kann.</p> <ol style="list-style-type: none"> <li>1. Popup: Auswahl neue Animation/Animationen</li> <li>2. Popup: Projekt benennen</li> <li>3. Animationen im Editor anzeigen</li> </ol> 	<p><b>Öffnen des Editors</b></p> <p>Ich als Nutzer möchte den AnimationEditor über ein spezielles Befehl aufrufen und damit öffnen können.</p> <ol style="list-style-type: none"> <li>1. System Browser: AnimationStore</li> </ol> 	<p><b>Animationen speichern</b></p> <p>Ich als Nutzer muss Animationen speichern können, damit ich sie immer wieder abspielen kann.</p> <ol style="list-style-type: none"> <li>1. Position eines Morphs auslesen</li> <li>2. Bewegung erkennen</li> <li>3. Neue Position im Editor festlegen</li> </ol> 	<p><b>GUI: Eigenschaften-Seitenleiste implementieren</b></p> <p>Ich als Nutzer braucht eine Stelle, an der ich Einstellungen bearbeiten kann. Die Seitenleiste dient als Platzhalter für Eigenschaften. Hier will ich die zu editierende Eigenschaft per Klick aus einer Dropdown auswählen können.</p> <ol style="list-style-type: none"> <li>1. GUI-Elemente integrieren: Buttons + Fortschrittsbalken</li> <li>2. Abspielen einer Animation mit Play inkl. Fortschrittsanzeige</li> <li>3. Anhalten einer laufenden Animation mit Stop</li> </ol> 
<p><b>Animation Store in dem Projekt automatisch erstellen lassen</b></p> <p>Der Nutzer kann ein Projekt anlegen und sich damit innerhalb seines Paketes automatisch eine entsprechende Klasse hinzufügen lassen, welche die Animationen für dieses Projekt verwaltet.</p> <ol style="list-style-type: none"> <li>1. AnimationStore automatisch im Ziel Projekt erstellen</li> <li>2. Animationen per Namen laden</li> <li>3. Animationen einem Morph zuweisen</li> </ol> 	<p><b>Play/Stop Button + Animation Zeitbalken</b></p> <p>Ich als Nutzer will Play und Stop/Pause Knöpfe. Durch diesen Klick soll der Zeitbalken loslaufen bzw. anhalten.</p> <ol style="list-style-type: none"> <li>1. GUI-Elemente integrieren: Buttons + Fortschrittsbalken</li> <li>2. Abspielen einer Animation mit Play inkl. Fortschrittsanzeige</li> <li>3. Anhalten einer laufenden Animation mit Stop</li> </ol> 	<p><b>neue Position des Morphs per Drag&amp;Drop setzen</b></p> <p>Blocked by: Keyframe hinzufügen Ich als Nutzer braucht die Möglichkeit eine Positionsänderung als Animation darzustellen, damit ich die Animationen mit Bewegungen versehen kann.</p> <ol style="list-style-type: none"> <li>1. Position eines Morphs auslesen</li> <li>2. Bewegung erkennen</li> <li>3. Neue Position im Editor festlegen</li> </ol> 	<p><b>GUI-Refactoring</b></p> <p>Ich als Nutzer möchte die GUI entsprechend meinen neuen Vorstellungen angepasst haben, um die Animationen in meinem SWA-Projekt noch einfacher bearbeiten zu können.</p> <ol style="list-style-type: none"> <li>1. Bereiche             <ul style="list-style-type: none"> <li>Eigenschaftsfenster</li> <li>Timeline</li> <li>Symbolbild</li> </ul> </li> <li>2. Setzen der Eigenschaften             <ul style="list-style-type: none"> <li>o Baumstruktur</li> <li>o Schaltflächen</li> <li>o Animation Starten</li> <li>o Keyframe hinzufügen</li> <li>o Timeline</li> <li>o zeigt alle Spuren an</li> </ul> </li> </ol> 
<p><b>Beginn/Ende d. Animation festlegen</b></p> <p>Ich als Nutzer muss den (zeitlichen) Anfang und das Ende der gesamten Animation festlegen können, damit sie zeitlich korrekt ist.</p> <ol style="list-style-type: none"> <li>1. Beginn einer Animation festlegen</li> <li>2. Ende einer Animation festlegen</li> <li>3. Gesamtdauer festlegen</li> </ol> 	<p><b>Hinzufügen einer Eigenschaftsspur</b></p> <p>Ich als Nutzer möchte in die Timeline sog. Eigenschaftsspuren hinzufügen können, damit ich verschiedene Eigenschaften des animierten Objekts voneinander unabhängig animieren kann.</p> <ol style="list-style-type: none"> <li>1. Container-Element für Eigenschaftsspuren erstellen</li> <li>2. GUI-Element Eigenschaftsspur erstellen (zunächst nur position)</li> <li>3. Eigenschaftsspur mit zu animierenden Objekt verknüpfen</li> </ol> 	<p><b>Animationen im Code einfügen</b></p> <p>Ich als Nutzer will gerne Animationen über einen Codeblock in meinem Projekt einfügen können, damit ich sie leicht in meinem SWA-Spiel abspielen kann.</p> <ol style="list-style-type: none"> <li>1. Animation über Code-Snippet aufrufbar machen</li> <li>2. Code ausführen → Animation abspielen</li> </ol> 	<p><b>Timeline anzeigen</b></p> <p>Ich als Nutzer braucht eine Art Timeline, damit ich die Zustände (halb-sekündige Intervalle) zwischen den Zuständen eines Objektes sehen kann.</p> <p>Timeline besteht aus</p> <ul style="list-style-type: none"> <li>Timeline-Rahmen</li> <li>Wrapper für die Spuren</li> <li>im Idealfall schafft es die Spuren zusammen</li> <li>Spur-Platzhalter</li> <li>Platzhalter für beliebige Spuren</li> </ul> 
<p><b>Key-Frames in Timeline einfügen</b></p> <p>Ich als Nutzer muss sog. Key-Frames in die Timeline einfügen können, damit ich bestimmte Zeitpunkte fixieren kann.</p> <ol style="list-style-type: none"> <li>1. Key Frames auf der Timeline hinzufügen</li> </ol> 	<p><b>Eigenschaften der Animation bearbeiten</b></p> <p>Ich als Nutzer benötige in dem Einstellungsfenster mehrere Funktionalitäten, um die Animation anpassen zu können.</p> <ol style="list-style-type: none"> <li>1. Gewünschte Einstellungen als Eingabeelemente in die GUI integrieren</li> <li>2. Animation entsprechend Eingabe in GUI aktualisieren</li> </ol> 	<p><b>Animationen live abspielen</b></p> <p>Ich als Nutzer brauche die Fähigkeit, Animationen beim Erstellen auch live abspielen zu können. Damit sehe ich, ob die Animation meinen Vorstellungen entspricht.</p> <ol style="list-style-type: none"> <li>1. Beim Abspielen ändern sich Live die Eigenschaften der Objekte</li> <li>2. Die Position des Morphs wird zwischen zwei Keyframes berechnet</li> <li>3. Beim Ändern eines Keyframes ändert sich auch die Position des Objektes</li> </ol> 	<p><b>Chart 6</b></p>

**SWT 1 - AnimationsEditor**  
Gruppe 15

# Funktionale Anforderungen

## User Story: Erstellen & Benennen einer Animation

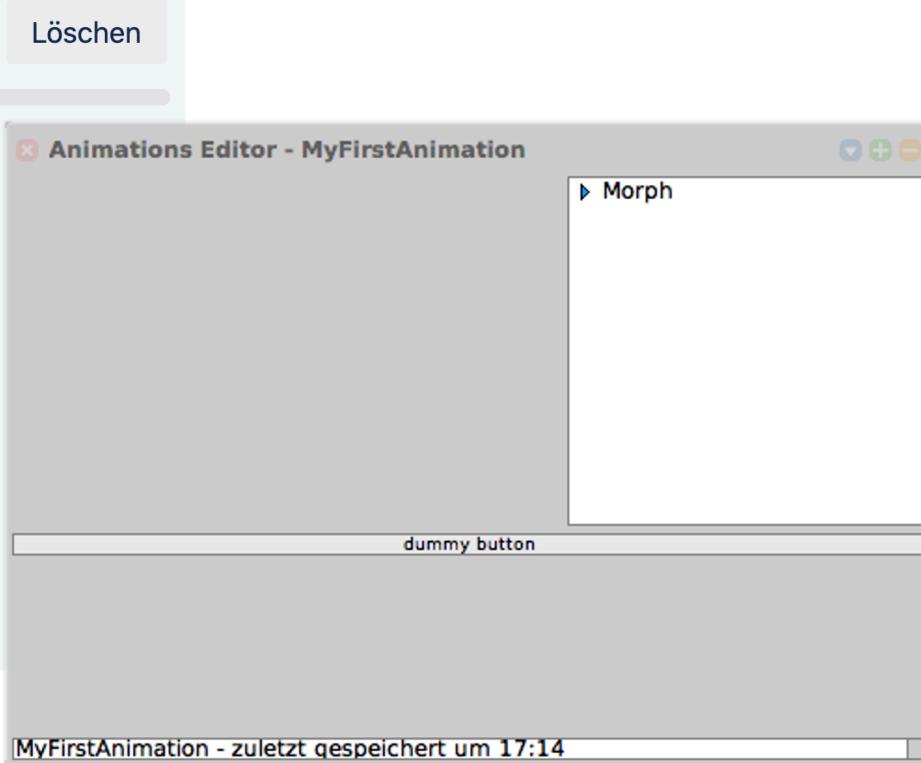
Ich als **Nutzer** würde gerne Animationen mit einem Namen versehen können, damit ich sie wieder erkennen kann.

Einzelaufgaben:

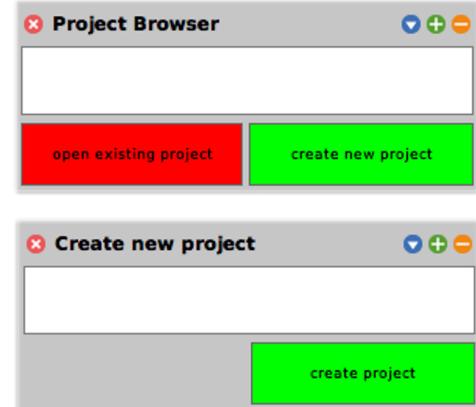
0%

- Popup 1: Auswahl zw. neuer Animation erstellen oder vorhandenem Projekt öffnen
- Popup 2: Benennen des Projektes
- AnimationsEditor öffnen
- Animationsname wird oben im Animations-Editor-Fenster angezeigt
- Neu: Fenster schließen sich automatisch
- Code Refactoring
- Tests schreiben

Löschen



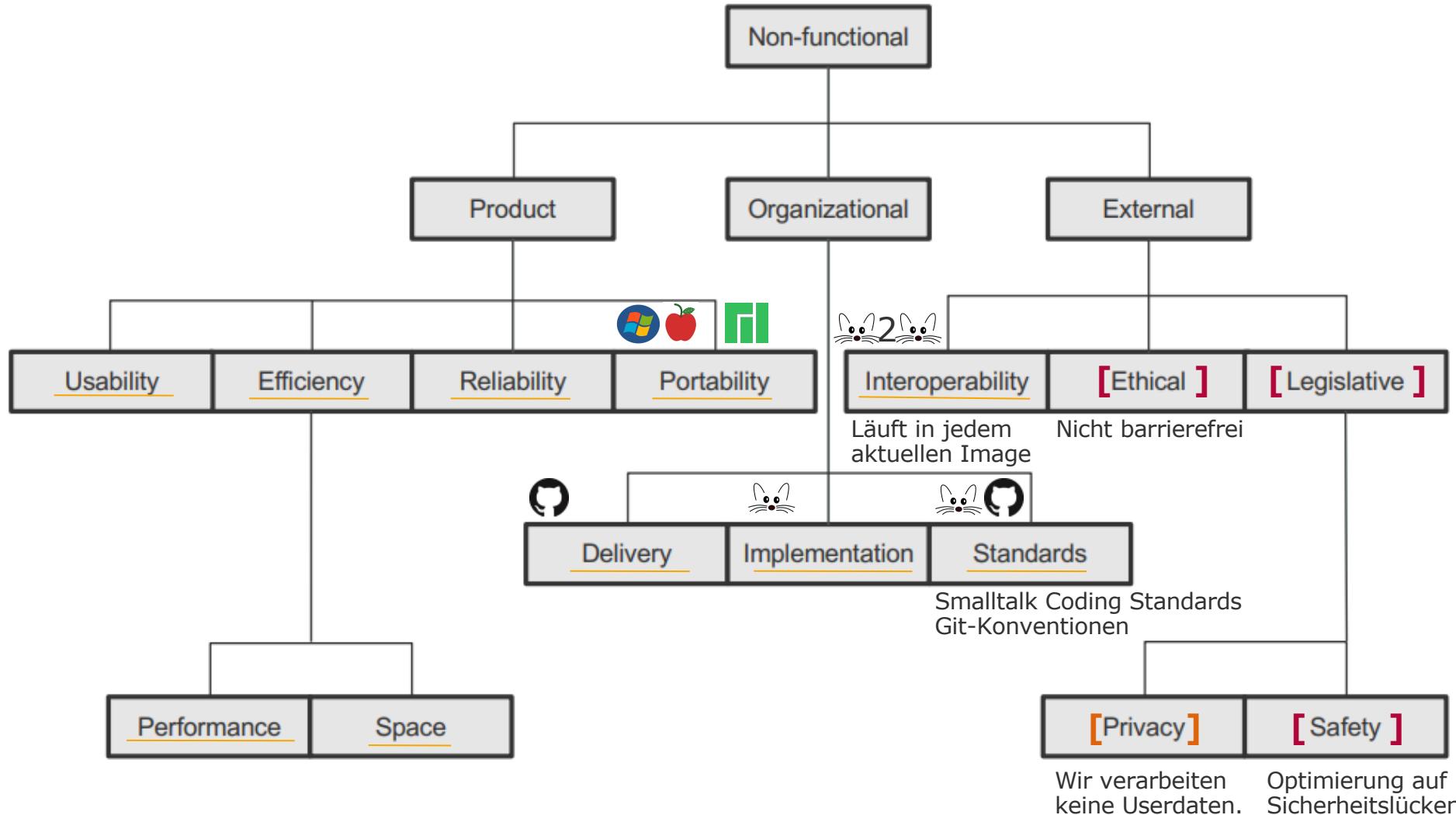
The screenshot shows the 'Animations Editor - MyFirstAnimation' window. It contains a single morph named 'Morph'. Below the morph is a 'dummy button'. At the bottom of the window, the text 'MyFirstAnimation - zuletzt gespeichert um 17:14' is visible.



**SWT 1 -  
AnimationsEditor**  
Gruppe 15

Chart 7

# Nicht-funktionale Anforderungen (Überblick)



[ ] - out of scope

[ ] - betrifft uns nicht

**SWT 1 - AnimationsEditor**

Gruppe 15

Chart 8

# Nicht-funktionale Anforderungen

## Implementation, Delivery, Efficiency

⌚ **Implementierung:** Das Produkt ist in Squeak/Smalltalk implementiert.

⌚ **Auslieferung:** Das Produkt wird auf GitHub ausgeliefert.  
[\(https://github.com/hpi-swa-teaching/AnimationsEditor/\)](https://github.com/hpi-swa-teaching/AnimationsEditor/)

### ⌚ Effizienz

- **Performance:** Das Produkt hat eine in Squeak übliche Reaktionszeit. (< 1 Sekunde)
- **Speicherbedarf:** Das Produkt beansprucht nicht mehr Speicher als in Squeak üblich. (RAM is the limit)

➤ Testen

SWT 1 -  
AnimationsEditor  
Gruppe 15

# Nicht-funktionale Anforderungen

## Interoperability, Portability



# Interoperabilität & Portabilität

- Das Produkt ist in jedem aktuellen Squeak-Image einsetzbar.
  - Der Funktionsumfang ist auf allen Plattformen (= Betriebssystemen) gleich.  
  
➤ Wir verwenden das durch den Lehrstuhl bereitgestellte Image *SWT2020*.  
➤ Wir nehmen nur Änderungen im Package *Animations-Core* vor und verwenden keine externen Tools.  
➤ Wir entwickeln und testen auf 3 verschiedenen Betriebssystemen.

# **SWT 1 - AnimationsEditor**

# Nicht-funktionale Anforderungen

## Standards

### Standards

- Das Produkt entspricht den Smalltalk Coding Standards.
  - Die UI sieht "squeak-typisch" aus.
  - Der übliche Git Deployment Prozess wird eingehalten.
- 
- Jedes neue Feature wird auf einem Feature-Branch entwickelt.
  - Neuer Code muss vor Pull Request ein Refactoring durchlaufen.
  - Im Refactoring überprüfen wir den Code auf Coding Standards.
  - Jeder Pull-Request muss vom Team erfolgreich reviewed werden.
  - Auf dem Master liegt nur einsatzbereiter Code.
  - Am Ende des Projekts wird der Linter eingesetzt.

**SWT 1 -  
AnimationsEditor**  
Gruppe 15

# Nicht-funktionale Anforderungen

## Usability, Reliability

### Usability, Zuverlässigkeit (& Robustheit)

- Die UI ist intuitiv benutzbar.
  - Der AnimationsEditor stürzt nicht unerwartet ab und produziert keine unerwarteten Fehler.
- 
- Sammeln von Kundenfeedback im Kundentreffen
  - Manuelles & automatisiertes Testen
  - Später: Die Kundin im Kundentreffen live unvorbereitet testen lassen.

**SWT 1 -  
AnimationsEditor**  
Gruppe 15

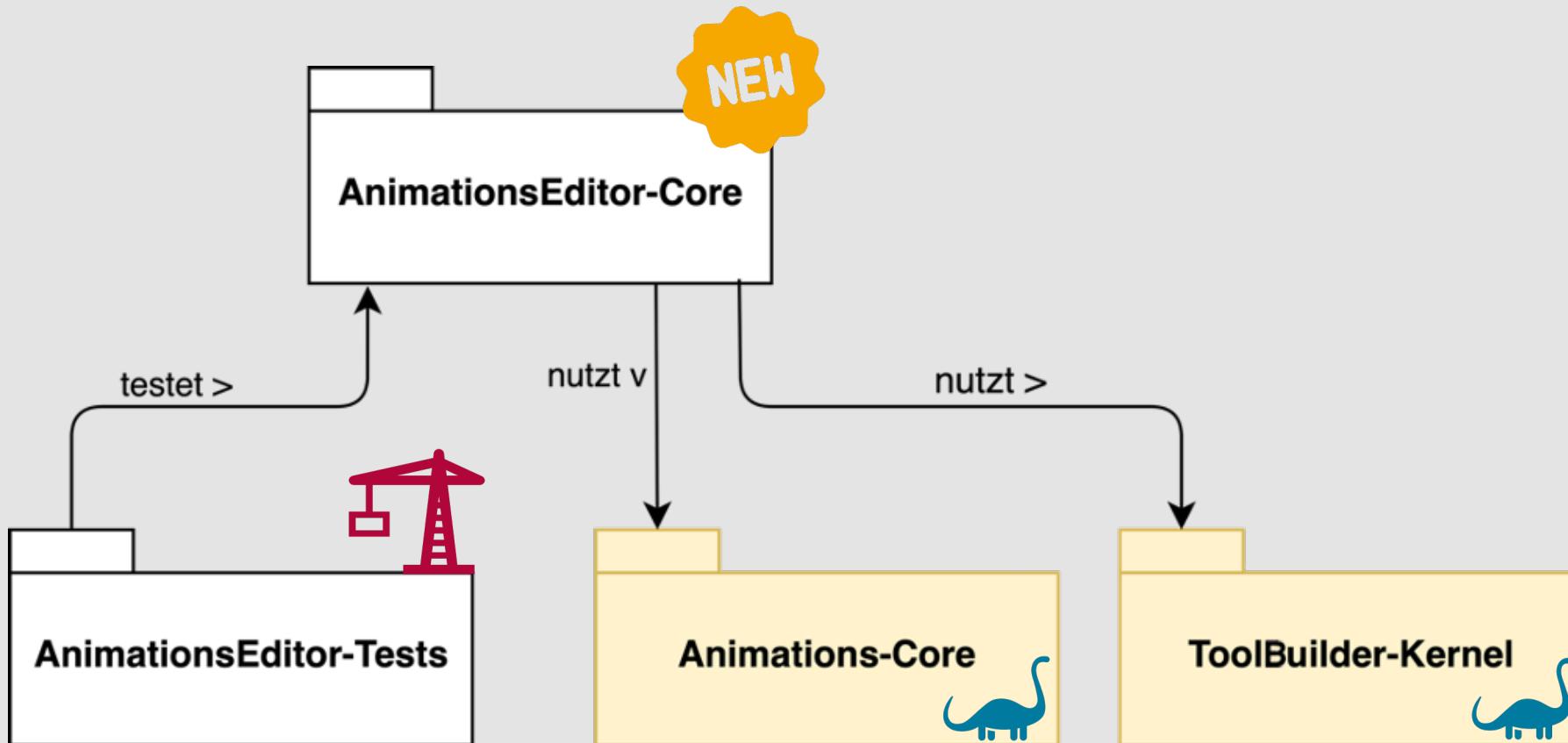
# Architektur des Softwaresystems

- Gesamtarchitektur
- Architektur des AnimationsEditors
- Arbeit mit den bestehenden Systemen

*Disclaimer: Nicht alle Aspekte der nachfolgenden Folien sind bereits im Code umgesetzt.*



# Gesamtarchitektur

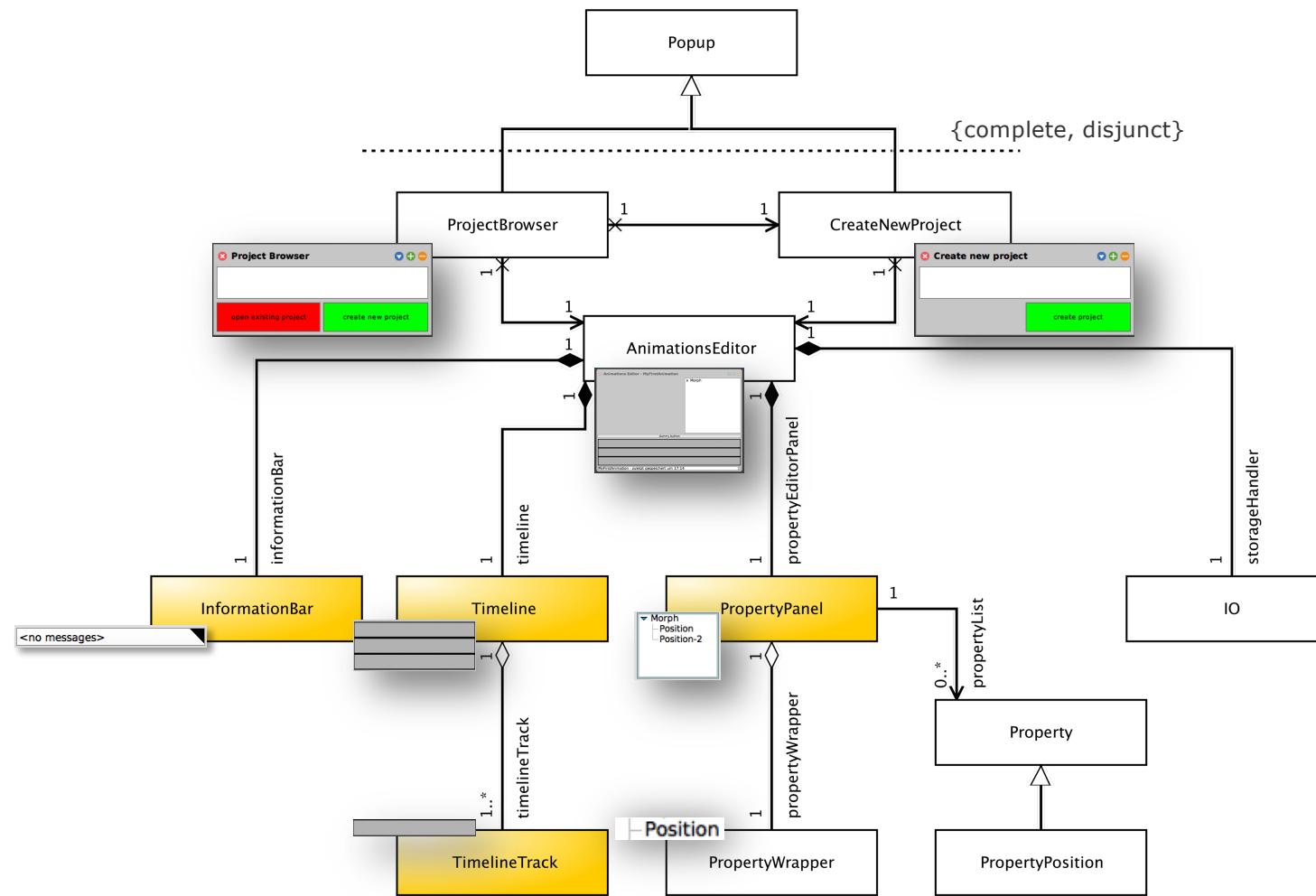


**SWT 1 -  
AnimationsEditor**

Gruppe 15

Chart 14

# Architektur des AnimationsEditors

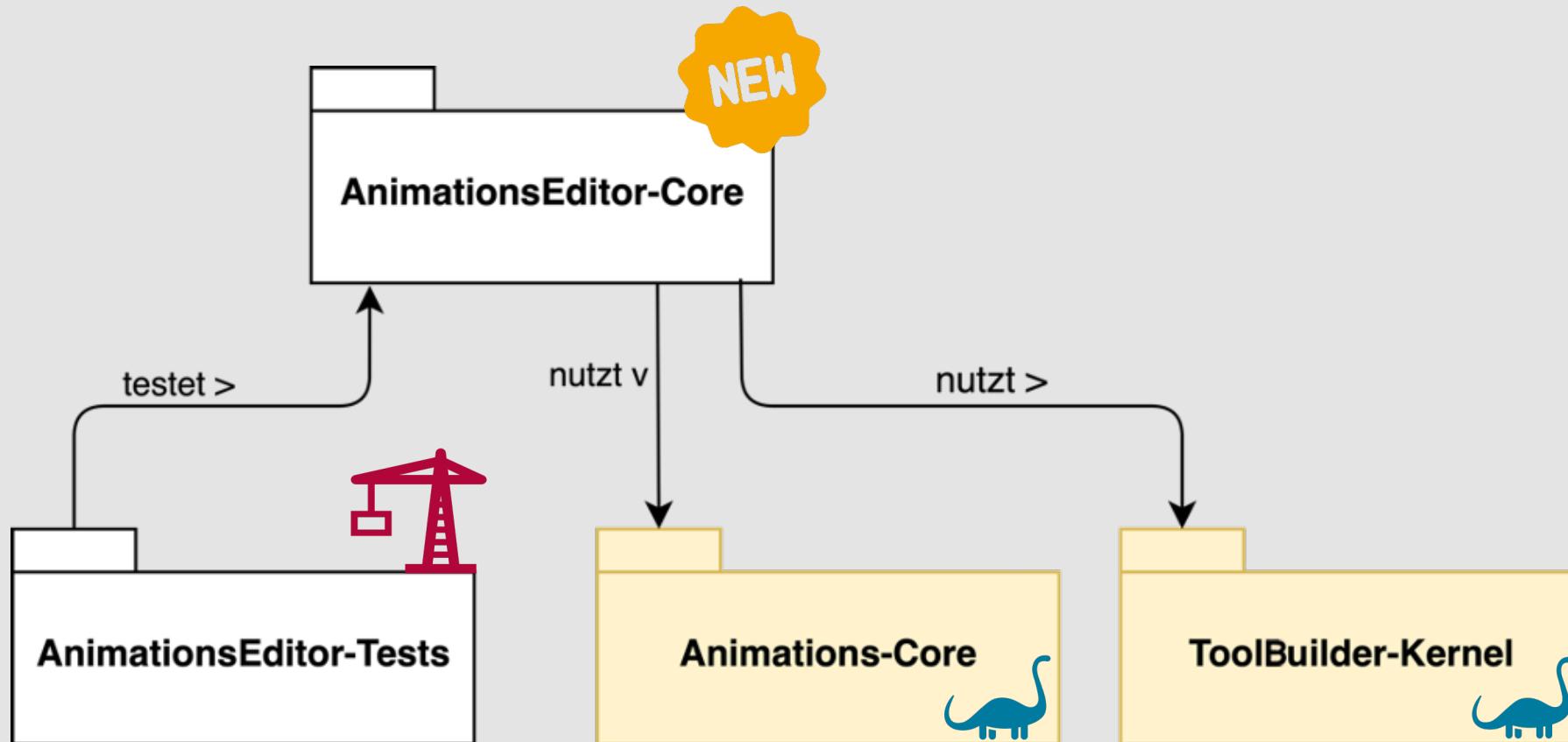


**SWT 1 -  
AnimationsEditor**

Gruppe 15

Chart 15

# Gesamtarchitektur



**SWT 1 -  
AnimationEditor**

Gruppe 15

Chart **16**

# Arbeit mit dem bestehenden System **Erfahrungen mit ToolBuilder**

- **zeitaufwändige Einarbeitung**
  - wenig Dokumentation, jede Komponente mit Eigenheiten, keine Code-Beispiele, (vorher) keine Erfahrungswerte dazu im Team
  - Demo nachbauen, Klassendefinitionen und Verwendungen im Image anschauen, Ausprobieren und exploratives Testen kostet Zeit
- für **statische UI** gut geeignet
- nicht ausgelegt für **dynamische UI-Gestaltung zur Laufzeit**
  - keine Unterstützung von Abfragen, Einfügen, Löschen von Sub-UI-Elementen zur Laufzeit
  - Unsere (Not-)Lösung: dynamisches Einfügen durch
    - Ersetzen der Submorph-Collection in leerem Panel im Hauptfenster
    - Update-Methode des PanelSpec

**SWT 1 -  
AnimationsEditor**  
Gruppe 15

Chart 17

# Arbeit mit dem bestehenden System Erfahrungen mit ToolBuilder

**Unsere Lösung:** Ersetzen der Submorph-Collection eines leeren Panels

**buildWith:** aBuilder

```
self toolBuilder: aBuilder.  
self buildMainWindowWith: aBuilder.  
self collectAllEmptyPanels.  
self collectPropertyTreeMorph.  
self initializePropertyEditorPanel.
```

^ mainWindow.



**initializePropertyEditorPanel**

```
self propertyEditorPanel: (self emptyPanels at: 2).  
self propertyEditorPanel getChildrenSelector: #propertyEditorPanelChildren.  
self propertyTreeMorph getChildrenSelector: #propertyList.
```

**SWT 1 -  
AnimationsEditor**

Gruppe 15

Chart 18

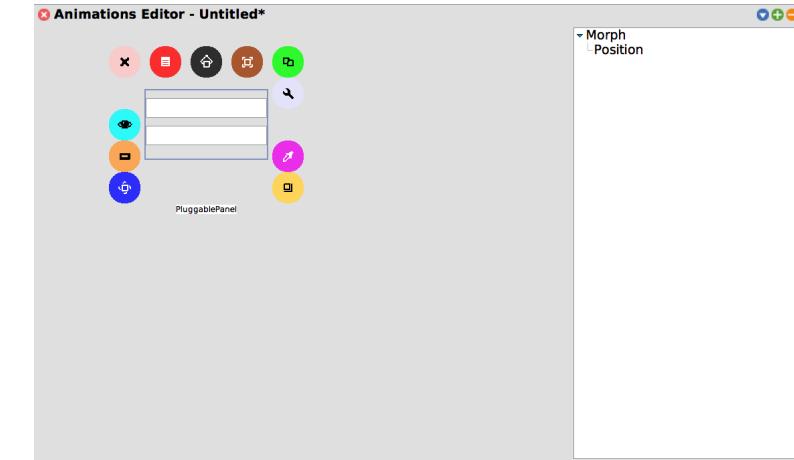
# Arbeit mit dem bestehenden System Erfahrungen mit ToolBuilder

**Unsere Lösung:** Ersetzen der Submorph-Collection eines leeren Panels

```
selectedTreeNode: aNode
selectedTreeNode := aNode.
```

```
aNode class = AnimationsEditorProperty
ifFalse: [self insertMorph: (aNode buildUIWith: toolBuilder)
    Into: self propertyEditorPanel].
```

```
insertMorph: aMorph Into: aPluggableSpecMorph
aPluggableSpecMorph children removeAll.
aPluggableSpecMorph children add: aMorph.
aPluggableSpecMorph update: aPluggableSpecMorph getChildrenSelector.
aMorph bounds: aPluggableSpecMorph bounds.
```



**SWT 1 -  
AnimationsEditor**  
Gruppe 15

Chart 19

# Arbeit mit dem bestehenden System **Erfahrungen mit AnimationsCore**

**Dokumentation** aus dem GitHub-Repository war **gut zum Einarbeiten**

- Gute Beispiele für den Start
- Ausführliche Dokumentation
- Tests für Funktionen haben zum Verständnis beigetragen



**Einarbeitung** verlief relativ **schnell** mittels Pair Programming

Problem: **Speichern** der Animation **kann Image zum Absturz bringen**

- wenn zu animierender Morph bereits in Welt geöffnet  
→ String Objekt kann nicht erzeugt werden → Image stürzt ab
- Lösung
  1. Entfernen des Morphs vor dem Speichern sicherstellen
  2. zu speichernde Animation erst bei Gebrauch erzeugen und danach verwerfen

**Animations** 

- 
1. [How to Install](#)
  2. [Simple Example](#)
  3. [Basic Animation Concept](#)
  4. [Variant Animations](#)
  5. [Property Animations](#)
  6. [How to Register Animations](#)
  7. [Graphics Animations](#)
  8. [Composite Animations](#)

**SWT 1 -  
AnimationsEditor**

Gruppe 15

Chart **20**

# Entwicklungsprozess

- Ablauf eines Sprints
- angewendete Praktiken
  - Sustainable Development
  - Planning Game
  - Prototyping

# Ablauf eines Sprints

Zyklusdauer: 2 Wochen

- Sprint-Rückblick
- Planung des neuen Sprints

- Vorbereitung des Kundentreffens
- Review + Feedback zum letzten Sprint

## Tool-Legende

	PowerPoint		Squeak
	Zoom		Github
	Trello		
	Discord		Screensharing



- Nachbesprechung des Kundentreffens
  - Festlegung der Programming Pairs
  - Vergabe der Arbeitspakete
  - Zeitliche Planung Treffen/Deadlines
- 
- Feature-Branch erstellen
  - Coding
  - Refactoring
  - Pull-Request
  - Peer-Review des Pull-Requests

**Koordination im Team** hat auch remote gut funktioniert

- *Trello* als Story-Board mit *Agile Guy* als Timetracking-Tool
- (Zeitlich dringende Absprachen in Telegram-Gruppe)
- Team-Termine als Kalender-Invites

**Remote-Zusammenarbeit im Team** funktionierte gut

- Pair Programming via Screensharing in Discord
- Synchronisierung der PP-Pairs durch
  - Peer-Review der Pull-Requests (GitHub)
  - Briefing im Discord-Weekly (< 5 Minuten)

**SWT 1 -  
AnimationsEditor**  
Gruppe 15

# Verwendete Praktiken im Überblick

## Xtreme Programming

Exactly X hours per week	On-site customer	Small releases	Test First Programming	Simple Design	Pair Programming
Planning Game	Sustainable Development 	Collective Code Ownership	CI + Ten Minute Build	Refactoring	Coding Standards

**SWT 1 - AnimationsEditor**  
Gruppe 15

Chart **24**

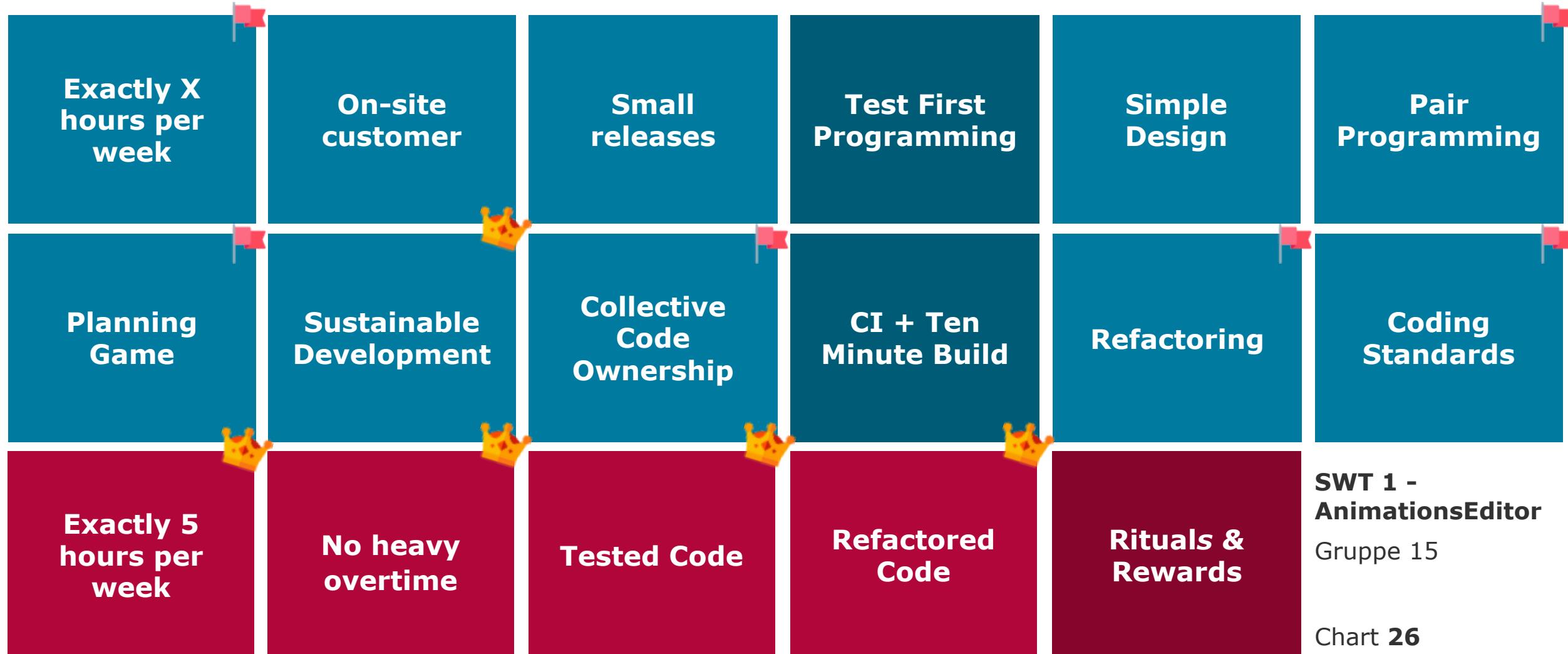
# Praktiken im Überblick

## Sustainable Development

Exactly X hours per week	On-site customer	Small releases	Test First Programming	Simple Design	Pair Programming
Planning Game	Sustainable Development 	Collective Code Ownership	CI + Ten Minute Build	Refactoring	Coding Standards
Exactly 5 hours per week    	No heavy overtime	Tested Code	Refactored Code	Rituals & Rewards   	SWT 1 - AnimationsEditor Gruppe 15 Chart 25

# Praktiken im Überblick

## Sustainable Development



# Praktik 1

# Sustainable Development

Berechnung der Arbeitszeit pro Woche:

Exactly 5  
hours per  
week



Jedes Pair Programming-Pair arbeitet genau **5 Stunden/Woche**



**SWT 1 -  
AnimationsEditor**  
Gruppe 15

Chart **27**

# Praktik 1

# Sustainable Development

Berechnung der Arbeitszeit pro Woche:

5h-Woche

$$\frac{(8 \text{ h} - 2 \text{ Vorlesungen} * 1.5 \text{ h}) * 4 \text{ Teammitglieder} * 2 \text{ Wochen Sprintlänge}}{2 \text{ (aufgrund PairProg.)}} = 20 \text{ Gummibären}$$



Jedes Pair Programming-Pair arbeitet genau **5 Stunden/Woche**

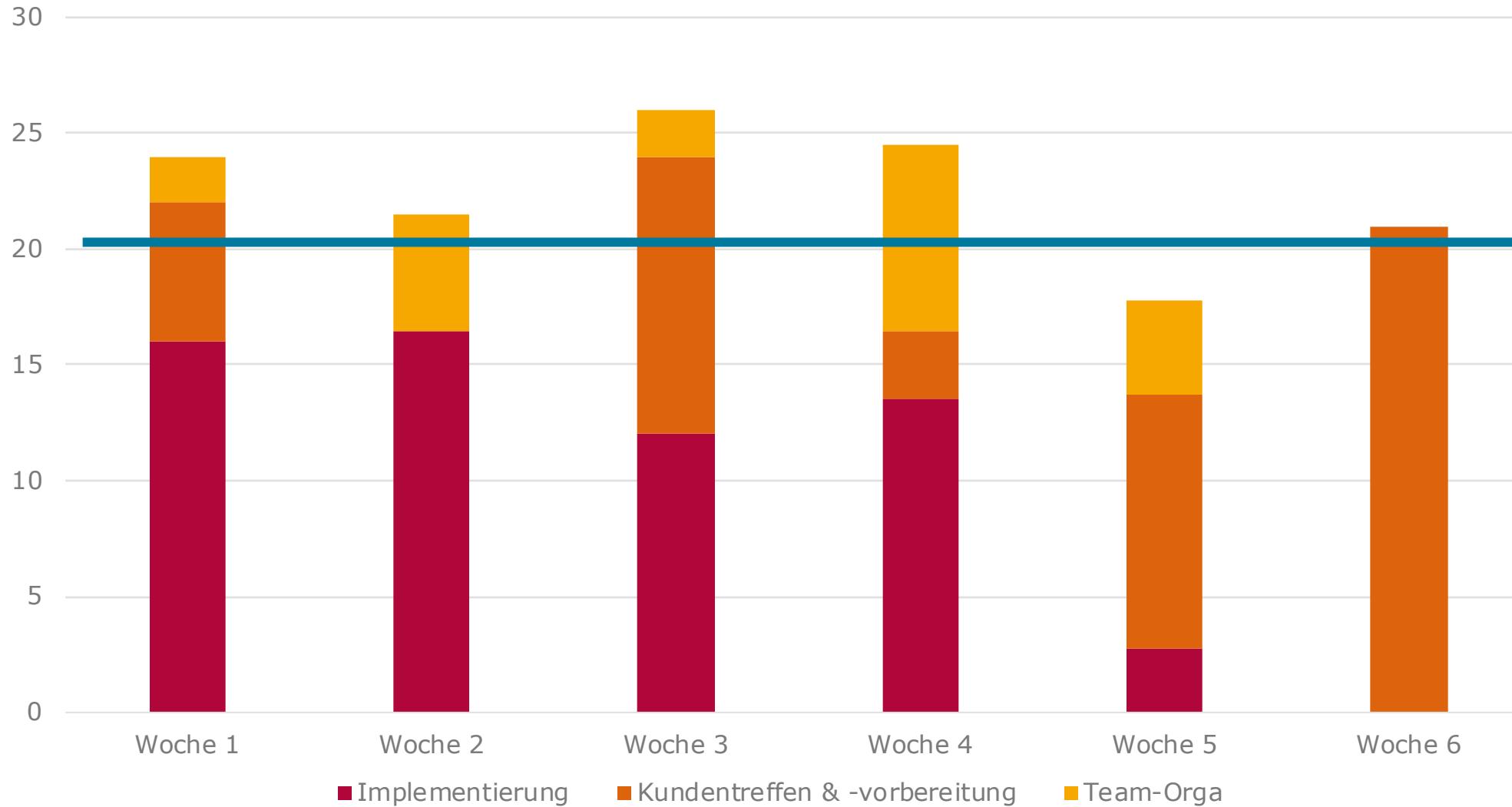


SWT 1 -  
**AnimationsEditor**  
Gruppe 15

Chart **28**

# Umsetzung der 5h-Woche

## Arbeitszeit pro Woche (kumuliert)



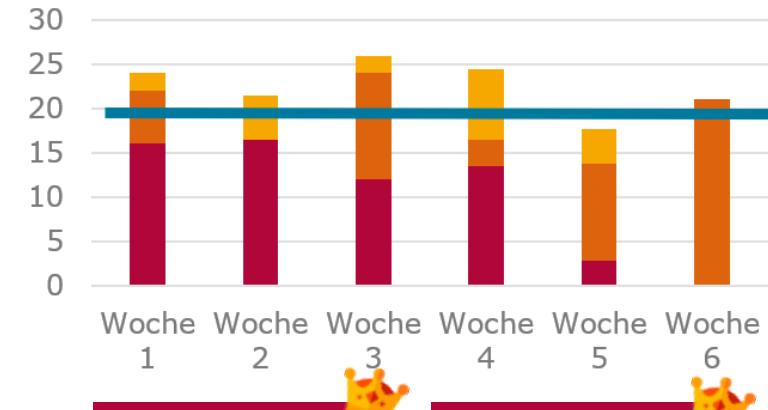
**SWT 1 -  
AnimationsEditor**

Gruppe 15

Chart **29**

# Umsetzung der 5h-Woche Lessons Learned

- Vorteile
  - beschränkte **Arbeitszeit** (*Parkinson'sches Gesetz*)
  - bessere **Planbarkeit**
  - beschränkter **Concorde-Effekt**
  - **störungsfreies & konzentriertes** Arbeiten
  
- Nachteile
  - **Orga-Overhead** verringert Arbeitszeit
  - **Features** können nicht immer **fertiggestellt** werden
  - der **Zeitdruck** ist ein ständiger Begleiter



No heavy  
overtime

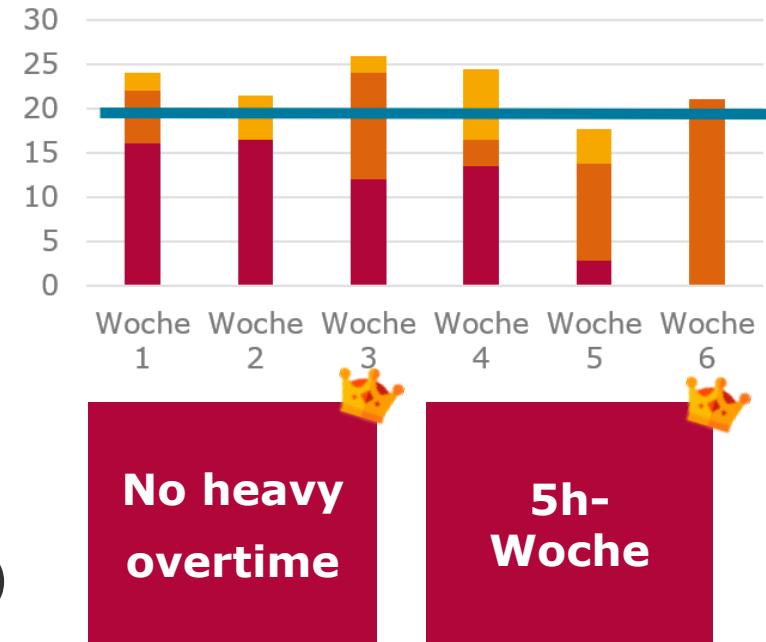
Exactly 5  
hours  
per week

SWT 1 -  
AnimationsEditor  
Gruppe 15

Chart 30

# Umsetzung der 5h-Woche Lessons Learned

- Ein **Hard Cut nach 5h** hat definitiv **Vorteile**:
  - **bessere Planbarkeit** in der Woche (Squeak-Life-Balance)
  - beschränkte Arbeitszeit für Projekt und Projektziele (*Parkinson'sches Gesetz*)
  - produktive, störungsfreie und konzentrierte Nutzung der Arbeitszeit
  - Wenn etwas nicht funktioniert, schaut man nächste Woche mit **frischem Blick** darauf. (beschränkter Concorde-Effekt)
- Trotzdem sind 5h pro Woche nicht viel:
  - **Overhead** muss möglichst gering gehalten werden
  - **Features** können auf den letzten Metern **nicht fertiggestellt** werden
  - Geringe Arbeitszeit sorgt für **Druck** im Sprint (positiv und negativ)

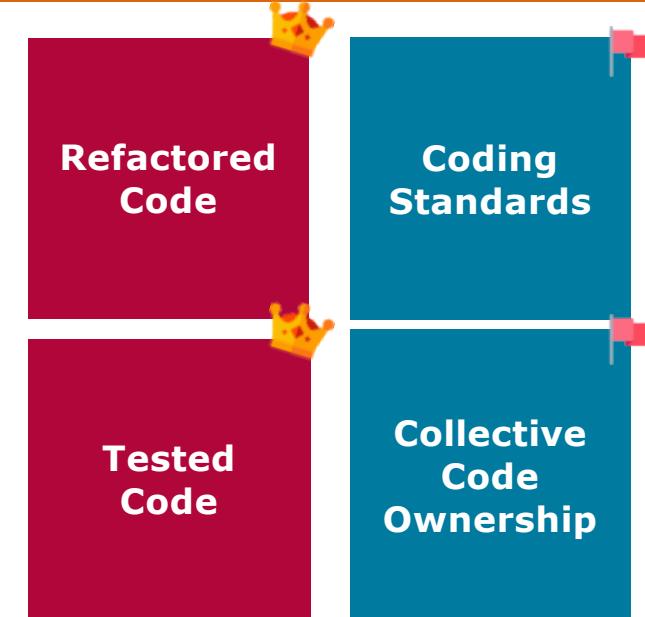


SWT 1 -  
AnimationsEditor  
Gruppe 15  
Chart 31

# Umsetzung Sustainable Development Lessons Learned

## ■ Kontinuierliches Refactoring

- sorgt für vorzeigbaren Code am Ende einer Arbeitsphase
- Einhaltung von Coding Standards
- Macht Code verständlich (Collective Code Ownership)



## ■ Testen

- Bisher existieren wenige Tests
- Im letzten Drittel wird CI und Testabdeckung ein wichtiges Thema

## ■ Peer Review

*anderes Partnerteam als Reviewer bei den Pull Requests einstellen*

- spart Zeit, da jeder einzeln Code Review betreibt
- Trägt zu Collective Code Ownership bei

**SWT 1 -  
AnimationsEditor**  
Gruppe 15

Chart 32

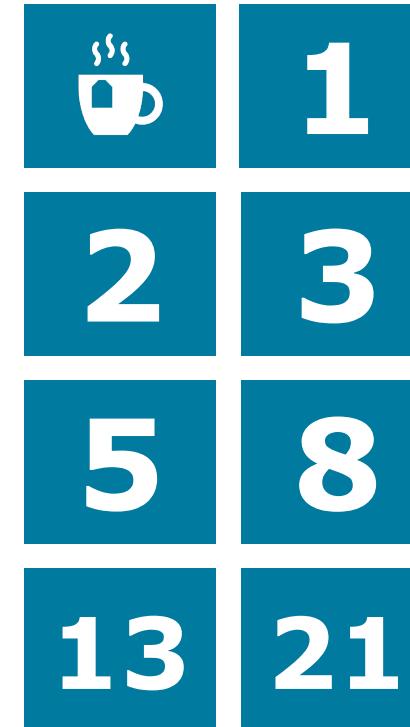
# Praktik 2

## Planning Game

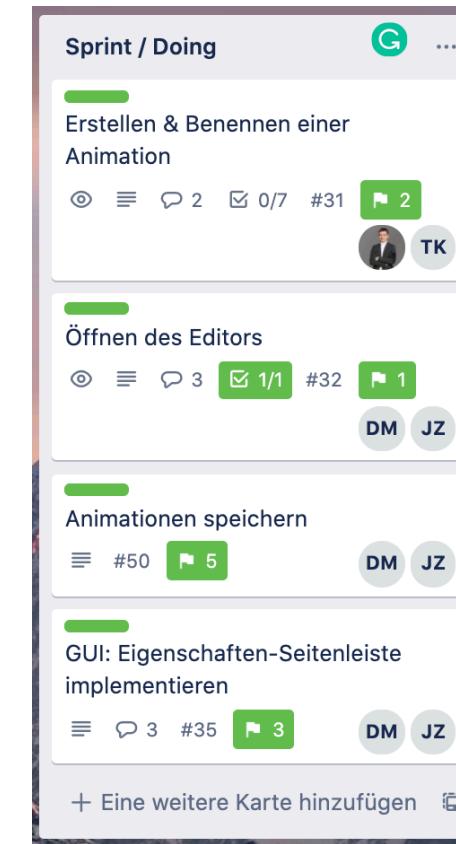
*User Stories  
definieren*



*Aufwand  
schätzen*



*Sprint-Backlog  
erstellen*



**Planning  
Game**

**SWT 1 -  
AnimationsEditor**  
Gruppe 15

**Chart 33**

# Planning Game Lessons Learned

- Gutes Mittel, um **Inhalt und Umfang** einer User Story zu **präzisieren**
- User Stories **nicht vor dem Kunden** schätzen
- **Zeitaufwandsschätzungen werden** im Laufe des Projekts **realistischer**
  - Nicht alle User Stories zu Beginn schätzen
  - Evtl. später neu schätzen/aufteilen

	1
2	3
5	8
13	21

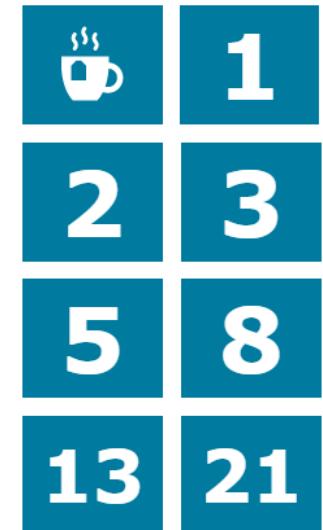
**Beispiel:** Aufteilung der US *Animation live abspielen*

- vorher: 21h
- nachher: 10h für Position, 11h für Rotation & Farbe

**SWT 1 -**  
**AnimationsEditor**  
Gruppe 15

# Planning Game Lessons Learned

- Gutes Mittel, um **Inhalt und Umfang** einer User Story zu **präzisieren**
  - offenbart unterschiedliche Vorstellungen der Teammitglieder
  
- User Stories **nicht vor dem Kunden schätzen**
  - **Befangenheit**, User Stories werden tendenziell niedriger eingeschätzt
  - "**Ad hoc Schätzungen**" sind tendenziell falsch



**SWT 1 -  
AnimationsEditor**  
Gruppe 15

Chart 35

# Planning Game Lessons Learned

- **Zeitaufwandsschätzungen werden im Laufe des Projekts **realistischer****
  - zu Beginn: fehlende Erfahrung mit Frameworks, Unterschätzung der Details
  - späteres evtl. Neuschätzen bzw. Aufteilen notwendig

	1
2	3
5	8
13	21

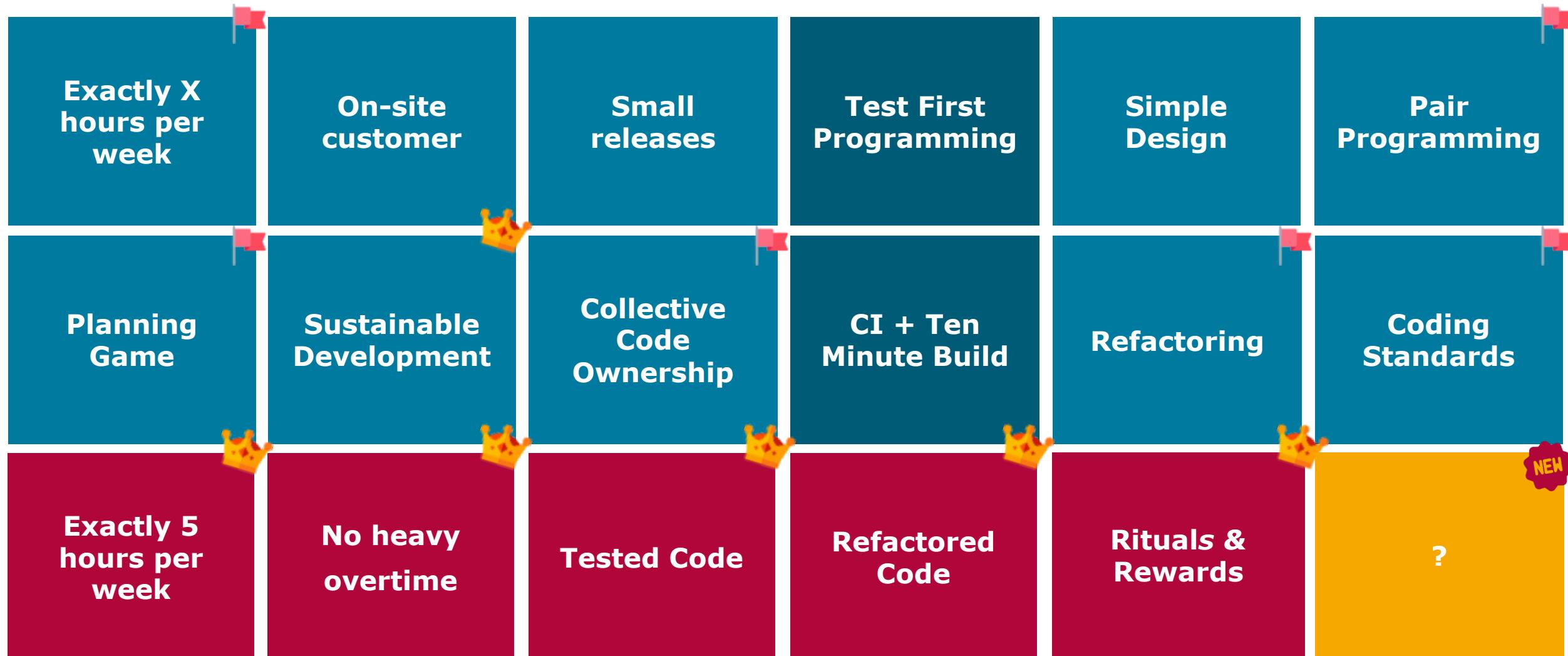
**Beispiel:** Aufteilung der US Animation *live abspielen*

- vorher: 21h
- nachher: 10h für Position, 11h für Rotation & Farbe

**SWT 1 -  
AnimationsEditor**  
Gruppe 15

Chart **36**

# Praktiken Anpassungen im Prozess

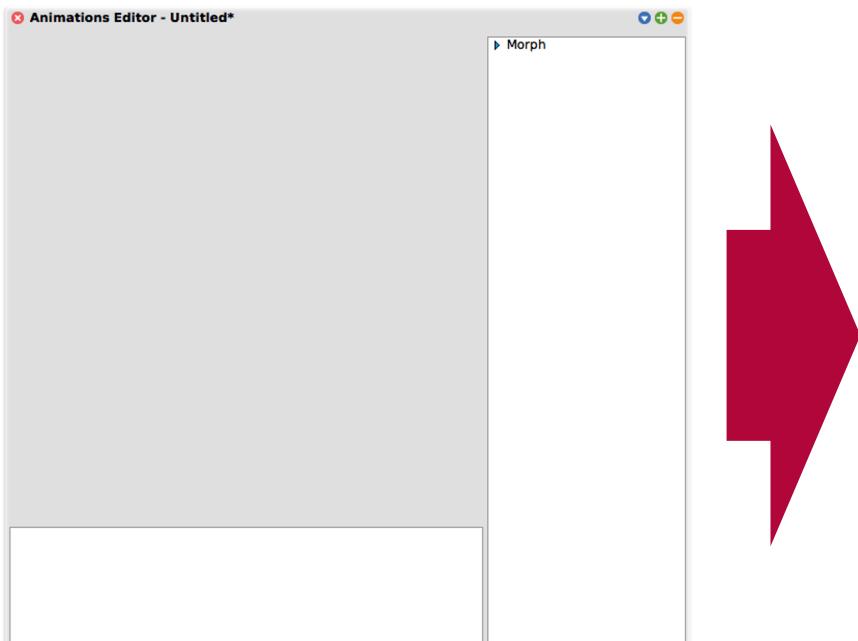


# Missverständnisse im Requirements Engineering

## Das GUI-Refactoring

Aufgrund eines Missverständnisses mit dem Kunden über den Aufbau der GUI, musste nach dem ersten Sprint die entstandene GUI umgebaut werden.

### Stand vor dem Refactoring



### Entwürfe für neues GUI-Design

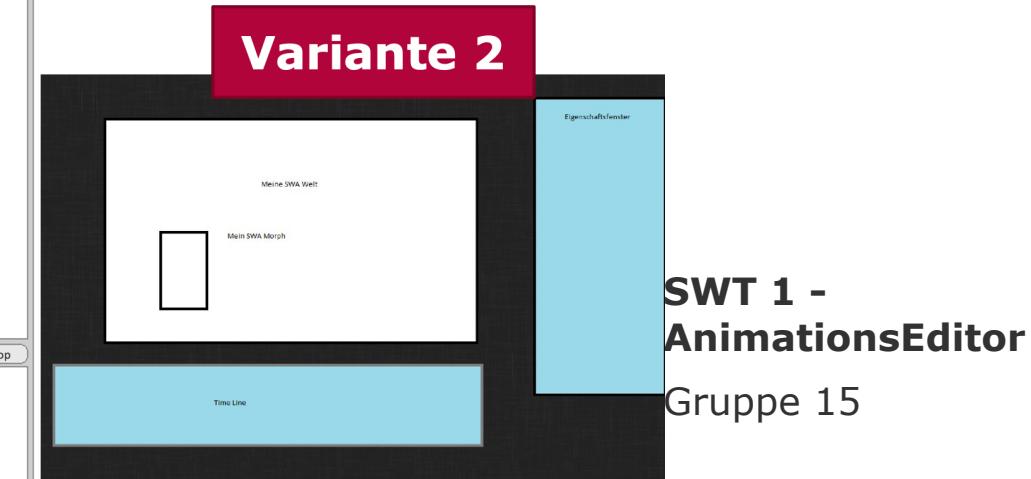
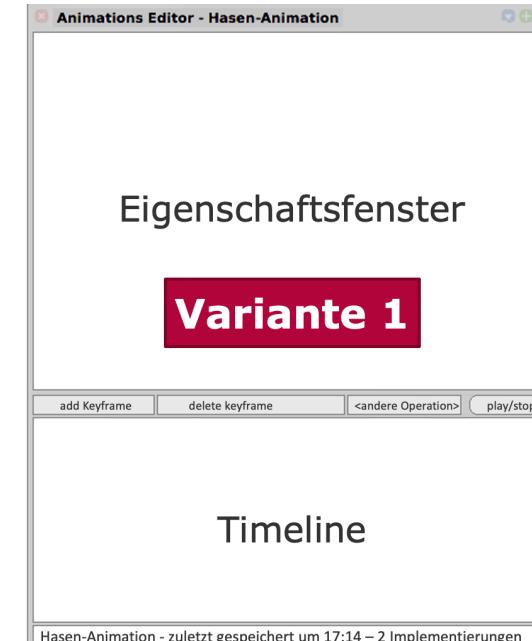


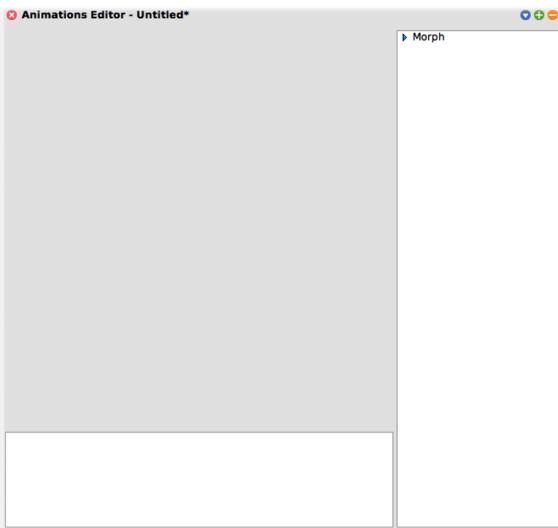
Chart 38

# Missverständnisse im Requirements Engineering

## Das GUI-Refactoring

Aufgrund eines Missverständnisses mit dem Kunden über den Aufbau der GUI, musste nach dem ersten Sprint die entstandene GUI umgebaut werden.

**Stand vor dem Refactoring**

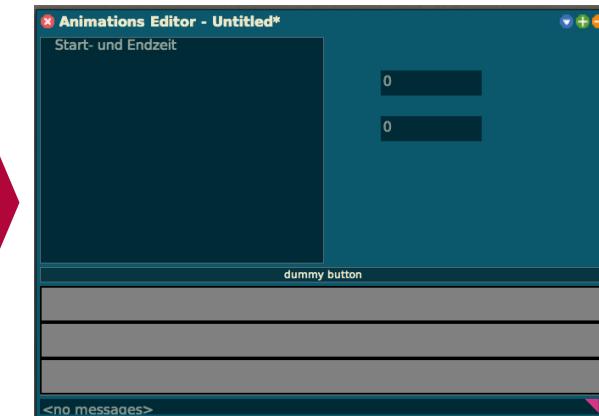


**Entwürfe für neues GUI-Design**

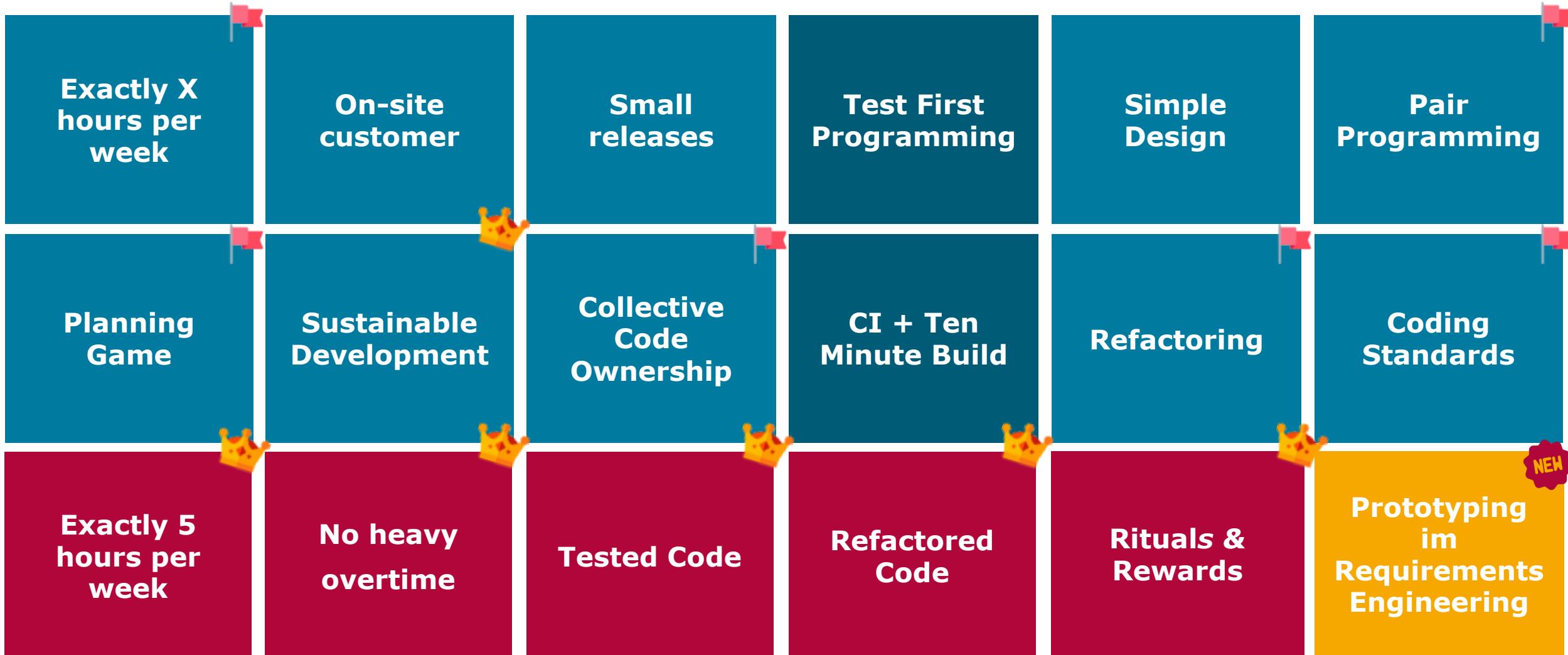


**Learning: zu jeder GUI-User Story gehört eine Skizze (Prototyp)!**

**neue GUI**



# Praktiken Prototyping



# Praktik: Prototyping Lessons Learned

**Problem:** Unstimmigkeiten zw. Kunde und Entwicklern entstehen durch fehlende Visualisierung im Requirements Engineering

Prototyping  
im  
Requirements  
Engineering

NEW

**Worst Case:** Unterschiedliches Verständnis fällt erst nach Fertigstellung auf

**begrenztes Zeitkontingent:** Zeit für Behebung fehlt für Umsetzung neuer Features

**Prototyping ist schnelle und einfache Möglichkeit,  
Missverständnisse zu vermeiden.**

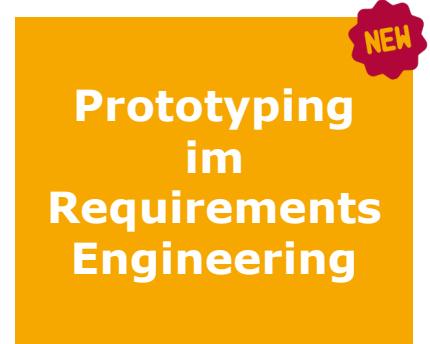
**SWT 1 -  
AnimationsEditor**

Gruppe 15

Chart 41

# Praktik: Prototyping Lessons Learned

- Durch das fehlende Prototyping sind Unstimmigkeiten zwischen der Kundin und dem Team entstanden
- Es ist sinnvoll zu Beginn einen Prototypen anzulegen, um die Kommunikation über die Gestaltung zu vereinfachen
- Missverständnisse können immer auftreten, können aber fatal sein.
  - Im Worst Case sitzt man x Stunden an einem Feature und die Kundin wollte es ganz anders.
  - Die Zeit, um das zu beheben, kann nicht für neue Features investiert werden.
  - Überstunden sind zu vermeiden, also fehlen am Ende produktive Arbeitsstunden.
- Daher sollten Missverständnisse von vornherein verhindert werden.



**SWT 1 -  
AnimationsEditor**  
Gruppe 15

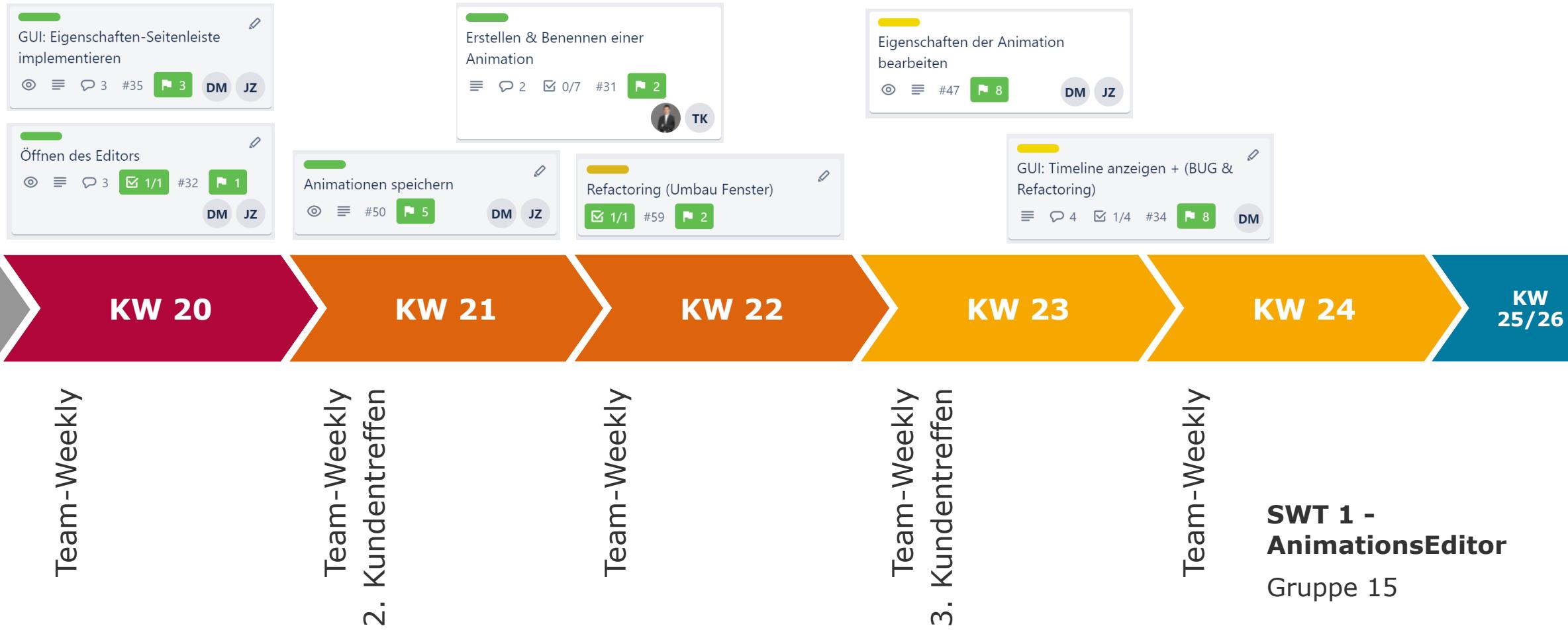
Chart 42

	A	B	C	D	E	F
1	Thema   Woche 1 (08.05.-14.05.)	Driver1	Observer1	Driver2	Observer2	
2	Planung, Tracking, Team-Orga	0,5	0,5	0,5	0,5	
3	Kundengespräche incl. Vorbereitung	1,5	1,5	1,5	1,5	
4	Coding incl. Recherche	3,66	3,66	4,33	4,33	
5	Tests schreiben					
6	Summe	5,7 h	5,7 h	6,3 h	6,3 h	
7						
8	Thema   Woche 2 (15.05-21.05.)	Driver1	Observer1	Driver2	Observer2	
9	Planung, Tracking, Team-Orga	1,25	1,25	1,25	1,25	
10	Kundengespräche incl. Vorbereitung					
11	Coding incl. Recherche	5,25	5,25	2,5	2,5	
12	Tests schreiben			0,5	0,5	
13	Summe	6,5 h	6,5 h	4,3 h	4,3 h	
14						
15	Thema   Woche 3 (22.05-28.05.)	Driver1	Observer1	Driver2	Observer2	
16	Planung, Tracking, Team-Orga	0,5	0,5	0,5	0,5	
17	Kundengespräche incl. Vorbereitung	3	3	3	3	
18	Coding incl. Recherche	2	2	4	4	
19	Tests schreiben					
20	Summe	5,5 h	5,5 h	7,5 h	7,5 h	
21						
22	Thema   Woche 4 (29.05-04.06.)	Driver1	Observer1	Driver2	Observer2	
23	Planung, Tracking, Team-Orga	2	2	2	2	
24	Kundengespräche incl. Vorbereitung		1,5		1,5	
25	Coding incl. Recherche	2,75	1	5,75	4	
26	Tests schreiben					
27	Summe	4,8 h	4,5 h	7,8 h	7,5 h	
28						
29	Thema   Woche 5 (05.06-12.06.)	Driver1	Observer1	Driver2	Observer2	
30	Planung, Tracking, Team-Orga	1	1	1	1	
31	Kundengespräche incl. Vorbereitung	4	6	1		
32	Coding incl. Recherche	0,25	0,25	0,25	2	
33	Tests schreiben			WWW-Zwischenklausur		
34	Summe	5,3 h	7,3 h	2,3 h	3, h	

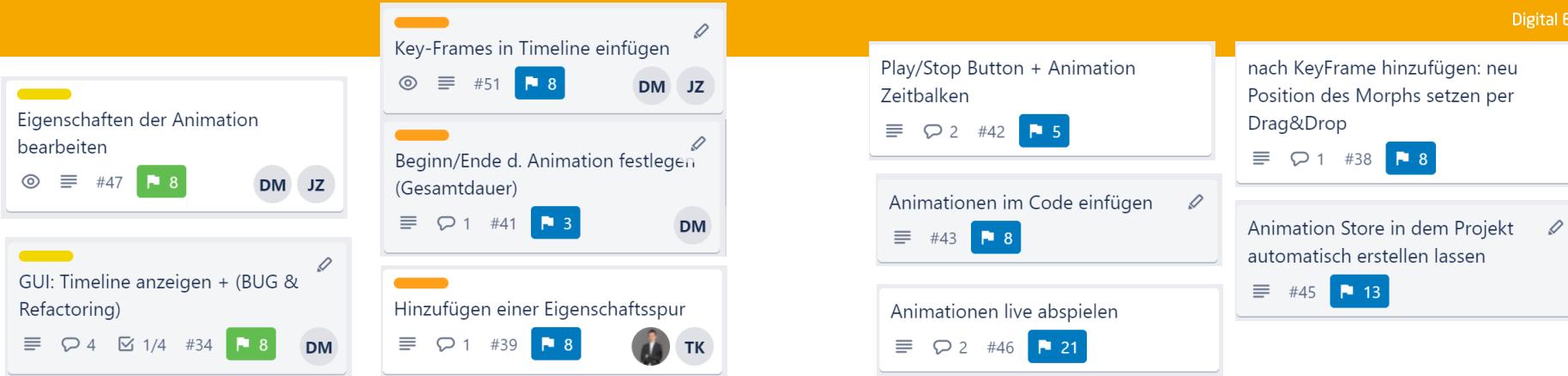
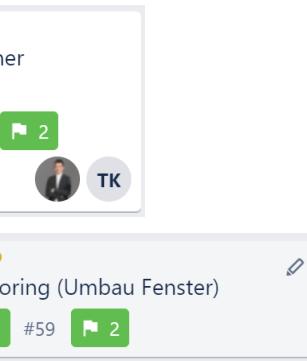
# Metriken und Projektverlauf

- zeitlicher Projektverlauf
  - Stand des Projektes
  - Metriken zum Projektverlauf

# zeitlicher Projektverlauf



# zeitlicher Projektverlauf



KW 22

KW 23

KW 24

KW 25/26

KW 27/28

KW 29/30

Team-Weekly  
3. Kundentreffen

Team-Weekly

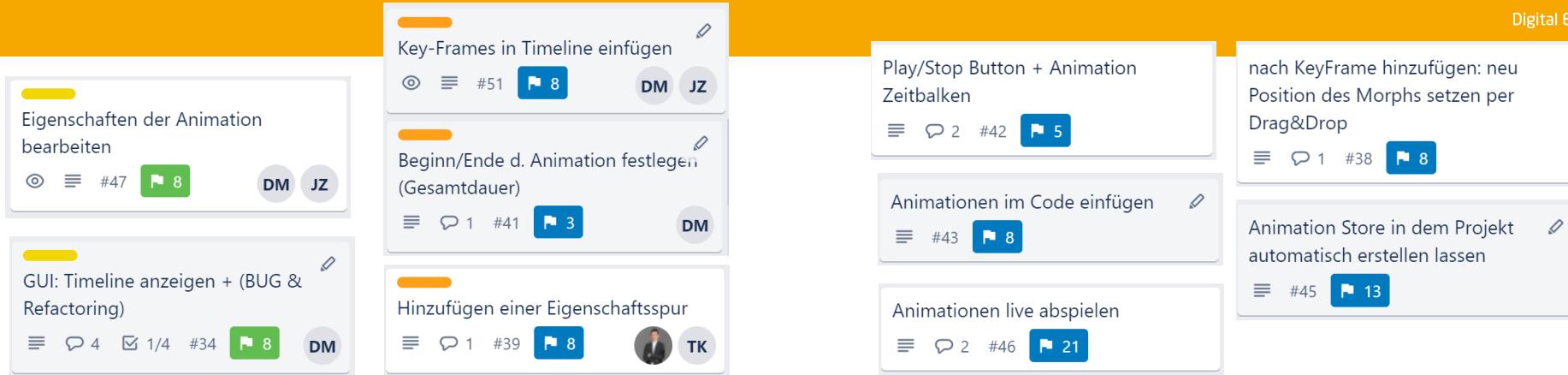
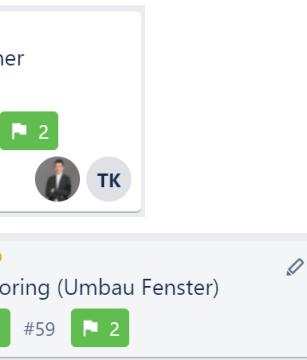
4. Kundentreffen  
Projektpräsentation

Team-Weekly

SWT-Klausur  
**SWT 1 -  
AnimationsEditor**  
Gruppe 15

Chart 45

# zeitlicher Projektverlauf



KW 22

KW 23

KW 24

KW 25/26

KW 27/28

KW 29/30

## Das Projekt...

- ... integrierte sich gut ins Studium.
- ... umfasste einen machbareren Workload
- ... war zeitlich gut planbar
- ... konnte auch remote recht gut durchgeführt werden
- ... verhielt sich zeitlich äquivalent zu Projekten anderer Module

WT-Klausur  
Abgabe  
**SWT 1 -  
AnimationsEditor**  
Gruppe 15

# Wie integrierte sich das Projekt ins Studium?

## Das Projekt...

- ... integrierte sich gut ins Studium.
- ... umfasste einen machbaren Workload
- ... war zeitlich gut planbar
- ... konnte auch remote recht gut durchgeführt werden
- ... verhielt sich zeitlich äquivalent zu Projekten anderer Module

Leider waren wir mit 5 Wochenstunden für die Programmierung recht eingeschränkt. Gern hätten wir mehr Zeit investiert, um größere Fortschritte zu erzielen.

**SWT 1 -  
AnimationsEditor**  
Gruppe 15

# Stand des Projektes

- vergleichsweise geringe Codebase
- detaillierte Klassenarchitektur
- wenig lange Methoden  
(vor allem durch GUI-Initialisierungen)
- Tests sind bisher nicht aussagekräftig

	<b>7,5</b>	<b>Pakete (NOP)</b>	<b>1+1</b>
	<b>10,3</b>	<b>Klassen (NOC)</b>	<b>9 + 6</b>
<b>5,9</b>	<b>Methoden (NOM)</b>	<b>114 + 41</b>	
<b>Codezeilen (LOC)</b>		<b>615 + 309</b>	

Code Tests

**Function Coverage:**  
**100%**

**Anzahl Tests:**  
**39**

Not Covered Code (100% Code Coverage)

AnimationsEditorPropertyWrapper setItem:method:parent:

brows sende impler versio inherit hierar vars source

setItem: anObject model: aModel parent: itemParent  
self parent: itemParent.  
self setItem: anObject model: aModel.

JIZ 6/22/2020 08:33 · accessing · 3 implementors · in change

39 run in 0:00:00:00.190

**SWT 1 -**  
**AnimationsEditor**  
Gruppe 15

Chart 48

# Metriken zum Projektverlauf

## Arbeitszeit pro Story

User Story	Sprint	Geschätzter Aufwand	Benötigter Aufwand	Legende
Öffnen des Editors	1	1	3	gute Schätzung
GUI-Seitenleiste	1	3	4	< 175 %
Animation speichern	1	3	5	> 175 %
Animation erstellen & benennen	1	2	9	
GUI-Refactoring	2	2	2	
Timeline anzeigen	2	8	8	
Animationseigenschaften bearbeiten	2	8	15	
Beginn & Ende der Animation festlegen	3	3	3	

**SWT 1 -  
AnimationsEditor**  
Gruppe 15

Chart 49



## Zusammenfassende Bewertung

- Reflexion des Projektes
- Reflexion der Teamarbeit

### Manchmal frustrierend ist, dass ...

- bestehende Tools wie Godot deutlich umfangreicher sind.
- der bisherige Projektfortschritt für uns nicht zufriedenstellend ist (→ motivationssenkend).
- das Projekt für ein 4-köpfiges Team zu groß wirkt.

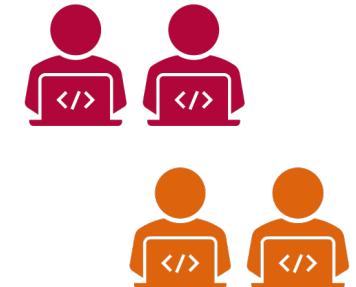
### Uns gefällt am Projekt besonders ...

- das Arbeiten mit GitHub/im Git-Workflow
- die regelmäßigen Kunden-Check-Ins.
- die regelmäßigen, gut vorbereiteten Tutor-Check-Ins.
- Sustainable Development: gutes Zeit-, Aufgaben- und Teammanagement sind in unserem Projekt essentiell (Lifeskill)

SWT 1 -  
AnimationsEditor  
Gruppe 15

### Das lief gut:

- die rein virtuelle **Kommunikation** und **Organisation**
- das (Remote) Pair Programming (**Teamfeeling** trotz Distanz)
- die produktive **Arbeitsmentalität**, gleichmäßige **Arbeitsverteilung** und das **Teamwork**

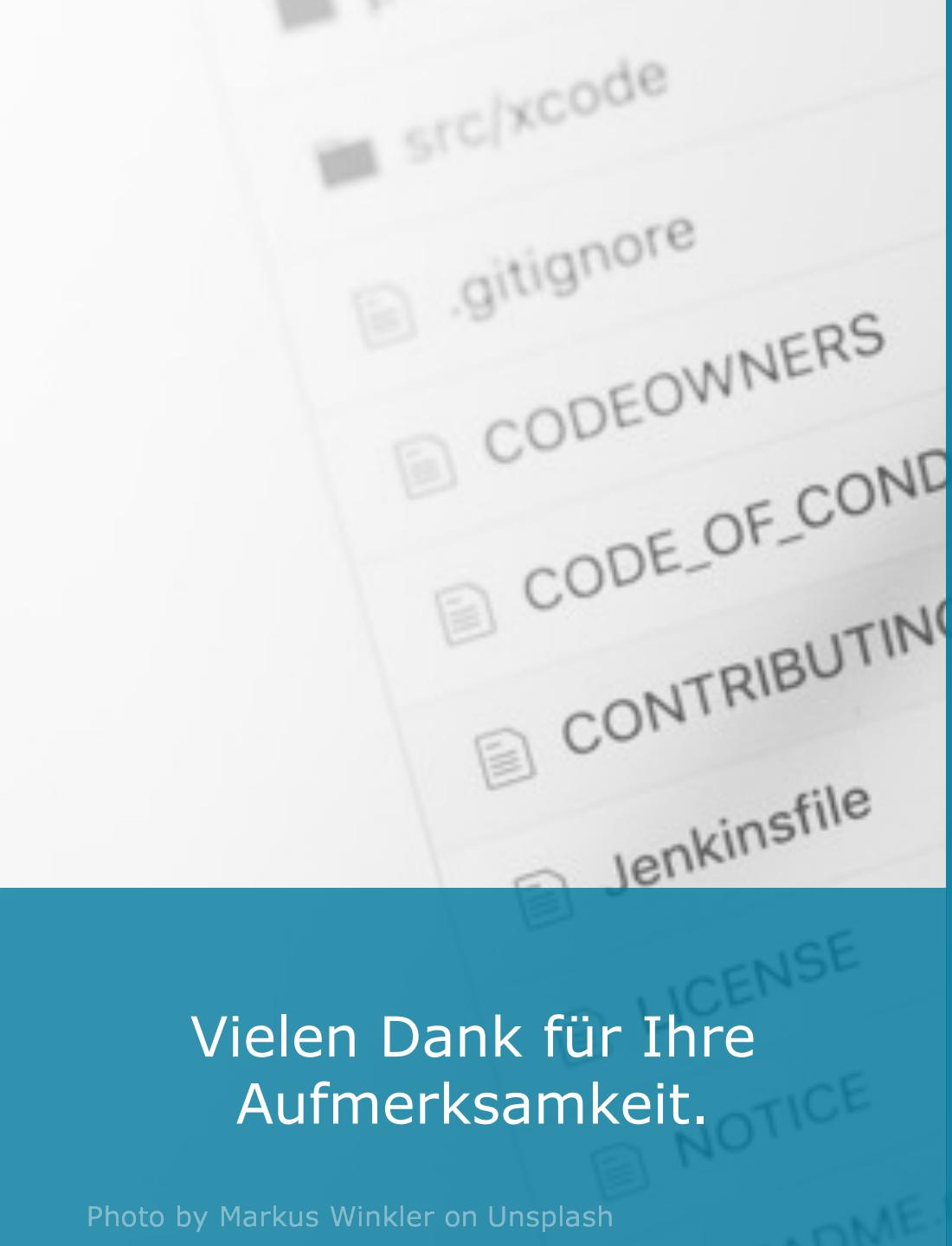


### Das machen wir in zukünftigen Projekten anders:

- Sustainable Development durch **bessere Zeitschätzungen** und **geringeren Overhead** konsequenter umsetzen.
- **Organisatorische Team-Aufgaben** (von Anfang an) **klar aufteilen**:  
Je eine verantwortliche Person für GitHub, Kundin/Tutorin-Kommunikation, Trello/Timetracking, Termine.
- als **Story-Board** GitHub Project Boards anstatt Trello nutzen  
(direkte Issue & Code-Integration, SSoT = GitHub)

**SWT 1 -  
AnimationsEditor**  
Gruppe 15

Chart **52**



Vielen Dank für Ihre  
Aufmerksamkeit.

## Quellen

Bilder und Icons

<https://www.unsplash.com/>  
<https://www.flaticon.com/>

*Diese Präsentation wurde mit freundlicher  
Unterstützung der zur Verfügung gestellten  
Vorjahresvorträge erstellt.*

# Erstellen & Benennen einer Animation

Ich als Nutzer würde gerne Animationen mit einem Namen versehen können, damit ich sie wiedererkennen kann

1. Popup: Auswahl neue Animation/Animation öffnen
2. Popup: Projekt benennen
3. Animationsname im Editor anzeigen



# Öffnen des Editors

Ich als Nutzer möchte den AnimationEditor über einen einfachen Befehl aufrufen und damit öffnen können.



# Animationen speichern

Ich als Nutzer muss Animationen speichern können, damit ich sie immer wieder abspielen kann



# GUI: Eigenschaften-Seitenleiste implementieren

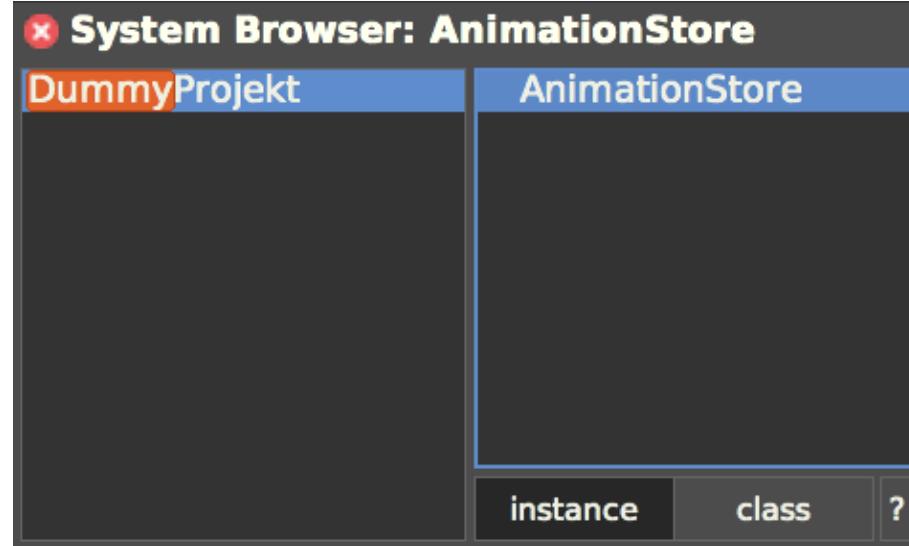
Ich als Nutzer bräuchte eine Stelle, an der ich Einstellungen bearbeiten kann. Die Seitenleiste dient als Platzhalter für Eigenschaften, wie zum Beispiel die Farbe von animierten Objekten. Hier will ich die zu editierende Eigenschaft per Klick aus einer Dropdown auswählen können.



# Animation Store in dem Projekt automatisch erstellen lassen

Der Nutzer kann ein Projekt anlegen und sich damit innerhalb seines Packages automatisch die "AnimationStore" Klasse hinzufügen lassen, welche die Animationen für dieses Projekt verwaltet.

1. *AnimationStore* automatisch im Ziel Projekt erstellen
2. Animationen per Namen laden
3. Animationen einem Morph zuweisen



13

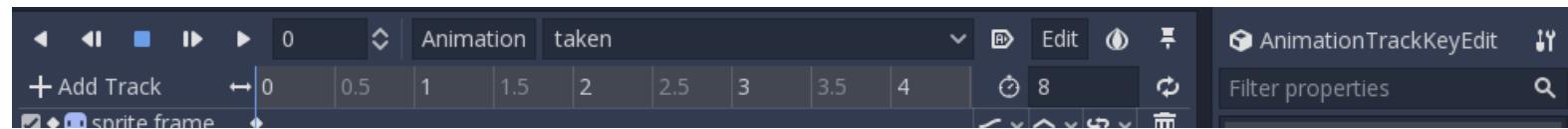
SWT 1 -  
**AnimationsEditor**  
Gruppe 15

Chart **58**

# Beginn/Ende d. Animation festlegen

Ich als *Nutzer* muss den (zeitlichen) Anfang und das Ende der gesamten Animation festlegen können, damit sie zeitlich korrekt ist.

1. Beginn einer Animation festlegen
2. Ende einer Animation festlegen
3. Gesamtdauer der Animation berechnen



<https://godotengine.org/storage/app/uploads/public/5b1/961/771/5b1961771e56d290557236.png>

**SWT 1 -  
AnimationsEditor**

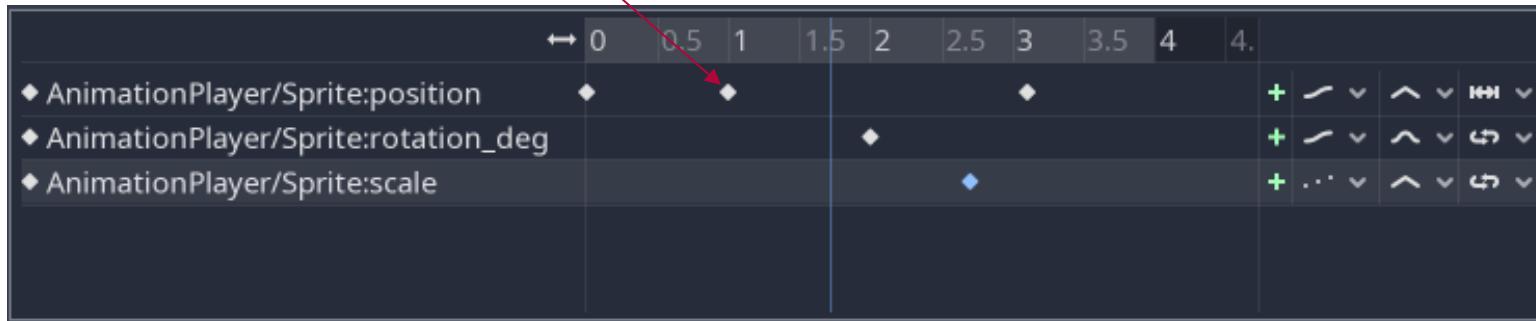
Gruppe 15

Chart **59**

# Key-Frames in Timeline einfügen

Ich als Nutzer muss sog. Key-Frames in die Timeline einfügen können, damit ich bestimmte Zeitpunkte fixieren kann.

## 1. Key Frames auf der Timeline hinzufügen



[https://docs.godotengine.org/en/3.0/\\_images/animation\\_timeline.png](https://docs.godotengine.org/en/3.0/_images/animation_timeline.png)



**SWT 1 -  
AnimationsEditor**  
Gruppe 15

Chart **60**

# Play/Stop Button + Animation Zeitbalken

Ich als Nutzer will Play und Stop/Pause Knöpfe. Durch diesen Klick soll der Zeitbalken loslaufen bzw. anhalten.

1. GUI-Elemente integrieren: Buttons + Fortschrittbalken
2. Abspielen einer Animation mit Play inkl. Fortschrittsanzeige
3. Anhalten einer laufenden Animation mit Stop



Symbolbild

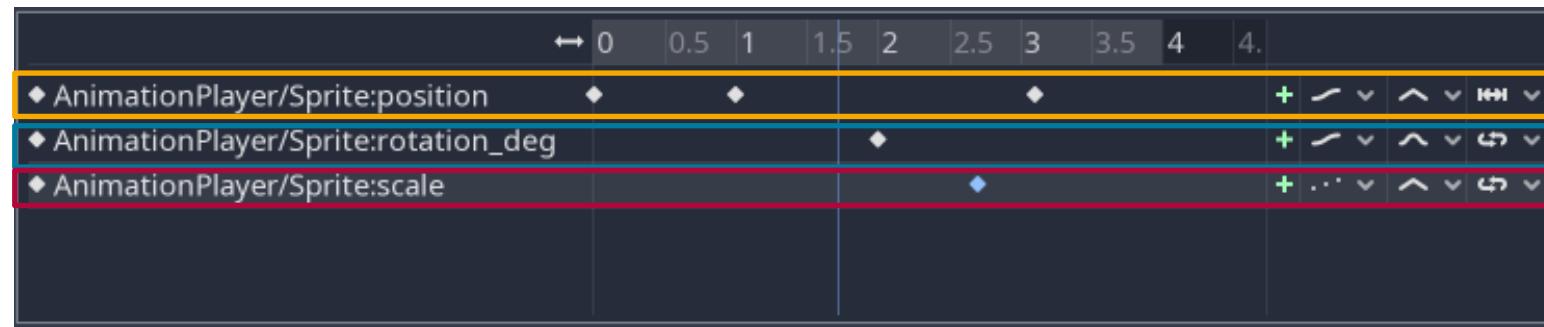
**SWT 1 -  
AnimationsEditor**  
Gruppe 15

Chart **61**

# Hinzufügen einer Eigenschaftsspur

Ich als Nutzer möchte in die Timeline sog. *Eigenschaftsspuren* hinzufügen können, damit ich verschiedene Eigenschaften des animierten Objekts voneinander unabhängig animieren kann.

1. Container-Element für Eigenschaftsspuren erstellen
2. GUI-Element Eigenschaftsspur erstellen (zunächst nur *position*)
3. Eigenschaftsspur mit zu animierenden Objekt verknüpfen

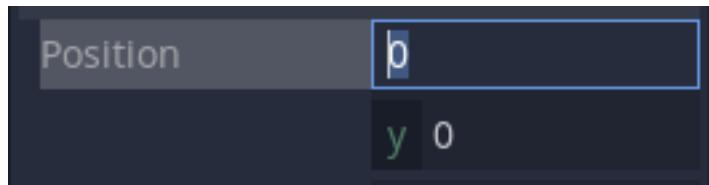


**SWT 1 -  
AnimationsEditor**  
Gruppe 15

# Eigenschaften der Animation bearbeiten

Ich als Nutzer benötige in dem Einstellungsfenster bestimmte Funktionalitäten, um die Animation anpassen zu können.

1. Gewünschte Einstellungen als Eingabeelemente in GUI integrieren
2. Animation entsprechend Eingabe in GUI ändern



8

13

# neue Position des Morphs per Drag&Drop setzen

Blocked by: KeyFrame hinzufügen

Ich als *Nutzer* brauche die Möglichkeit eine Positionsänderung als Animation darzustellen, damit ich die Animationen mit Bewegungen versehen kann.

1. Position eines Morphs auslesen
2. Bewegung erkennen
3. Neue Position in Keyframe speichern



Pos: 100@100



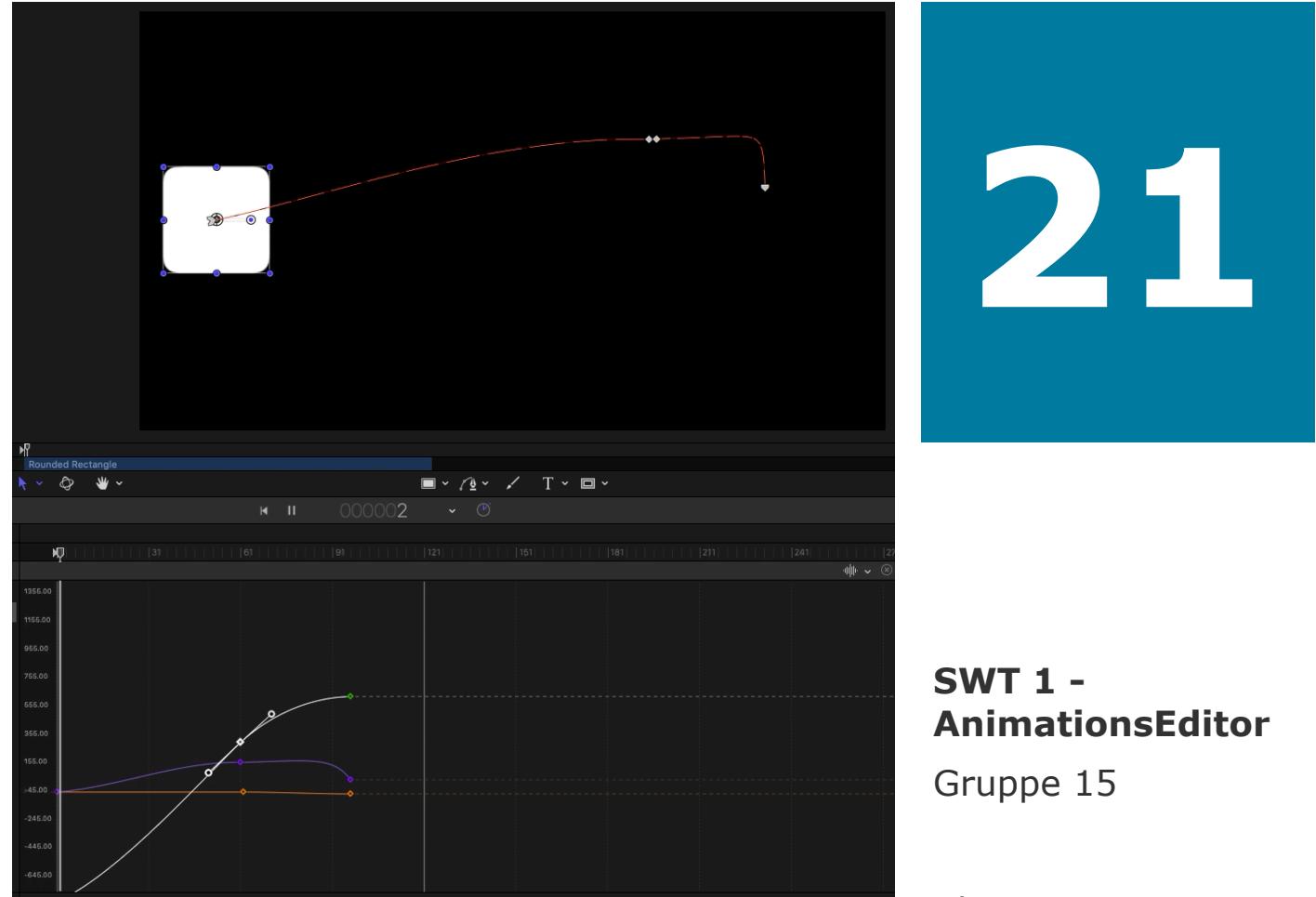
**SWT 1 -  
AnimationsEditor**  
Gruppe 15

Chart **64**

# Animationen live abspielen

Ich als Nutzer brauche die Funktionalität, Animationen beim Erstellen auch live abspielen zu können. Damit sehe ich, ob die Animation meinen Vorstellungen entspricht.

1. Beim Abspielen ändern sich Live die Eigenschaften der Objekte
2. Die Position des Morphs wird zwischen zwei Keyframes berechnet
3. Beim Ändern eines Keyframes ändert sich auch die Position des Objektes

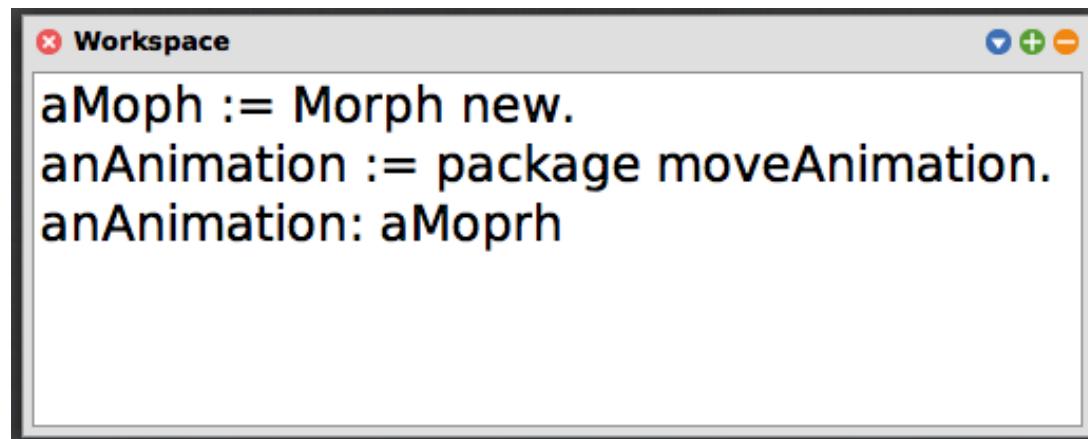


**SWT 1 -  
AnimationsEditor**  
Gruppe 15

# Animationen im Code einfügen

Ich als Nutzer will gerne Animationen über einen Codeblock in meinem Projekt einfügen können, damit ich sie leicht in meinem SWA-Spiel abspielen kann.

1. Animation über Code-Snippet aufrufbar machen
2. Code ausführen → Animation abspielen



```
aMoph := Morph new.  
anAnimation := package moveAnimation.  
anAnimation: aMoprh
```

SWT 1 -  
**AnimationsEditor**  
Gruppe 15

Chart **66**

# GUI-Refactoring

Ich als Nutzer möchte die GUI entsprechend meiner skizzierten Vorstellungen angepasst haben, um die Animationen in meinem SWA-Projekt noch einfacher bearbeiten zu können.

3 Bereiche

- Eigenschaftsfenster
  - Setzen der Eigenschaften (Baumstruktur)
- Schaltflächen
  - Animation Starten / Stoppen
  - Keyframe hinzufügen / löschen
- Timeline
  - zeigt alle Spuren an



# Timeline anzeigen

Ich als Nutzer bräuchte eine Art Timeline, damit ich die Abstände (halb-sekündige Intervalle) zwischen den Zuständen des Objektes sehen kann.

Timeline besteht aus

- Timeline-Rahmen
  - Wrapper für die Spuren
  - *im Idealfall scrollbar*
- Spur-Platzhalter
  - Platzhalter für beliebige Spuren

