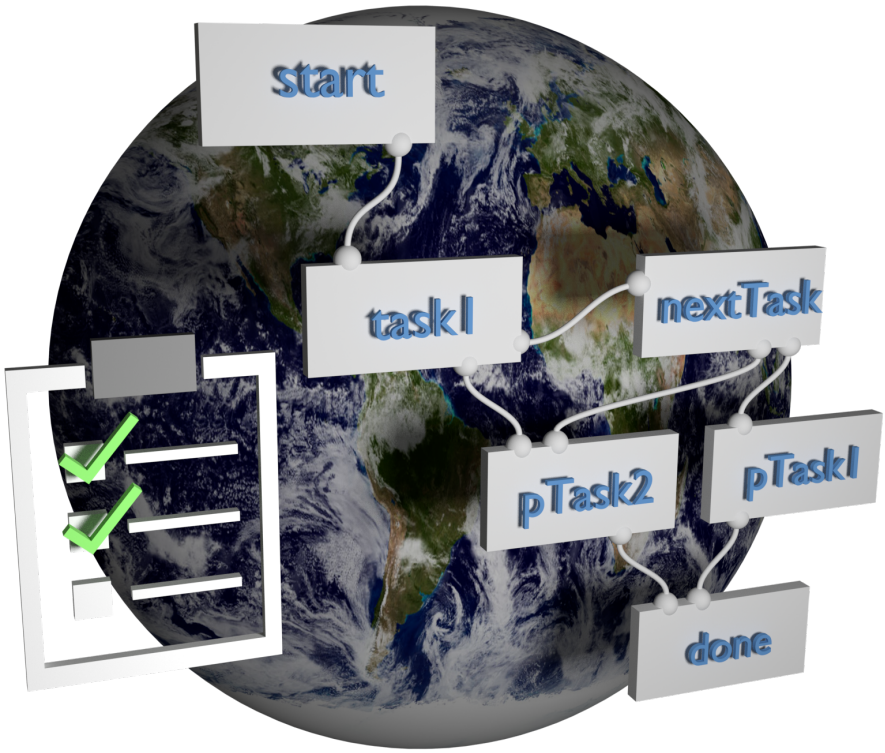
Implementierungsbericht

Workflow System für eine virtuelle Forschungsumgebung für Geodaten

Denis Gensh, Marcel Herm, David Krah,
Eduard Kukuy, Daniel Milbaier, Richard Rudolph



8. Februar 2018
Karlsruher Institut für Technologie, SCC

Inhaltsverzeichnis

1 Einleitung

Für größere Projekte ist eine lückenlose Dokumentation unerlässlich. Hierfür wird dieses Dokument vor allem auch auf Änderungen zwischen Entwurfsdokument und tatsächlicher Implementierung eingehen. Dabei wird auch auf die Unterschiede zwischen geplanten Funktionalitäten und Implementierten Funktionen eingegangen. Zusätzlich werden die einzelnen Funktionen genauer erläutert und der Ablauf der Implementierung erklärt.

2 Planung

2.1 Einleitung

2.2 Aufgabenverteilung

Grob wurde das Projekt in zwei große Blöcke je 3 Leute aufgeteilt, in Client und Server. Innerhalb dieser Blöcke wurde wieder etwas unterteilt, beim Server waren dies die Unterblöcke Datenbank, Cron Utils, Cron Receiver, Cron Scheduler und REST API. Clientseitig wurden die Aufgaben in Editor, Einstellungen, Fehlermanagement und Workflowmanagement unterteilt. Die zugewiesenen Aufgaben wurden in kleinen Schritten erledigt und bei Abschluss wurde ein weiterer kleiner Abschnitt zugewiesen, um eine dynamische Zuweisung zu ermöglichen. So konnten im späteren Verlauf Leute vom Client zum Server wechseln, nachdem der Client eher als geplant fertiggestellt wurde. Als Client und Server hinreichend fertig waren, um verbunden zu werden, wurde dies von je einem aus dem Client Bereich und einem aus dem Server Bereich vollzogen. Am Ende hat das gesammte Team sich ums Testen und um kleinere Bugfixes etc. gekümmert.

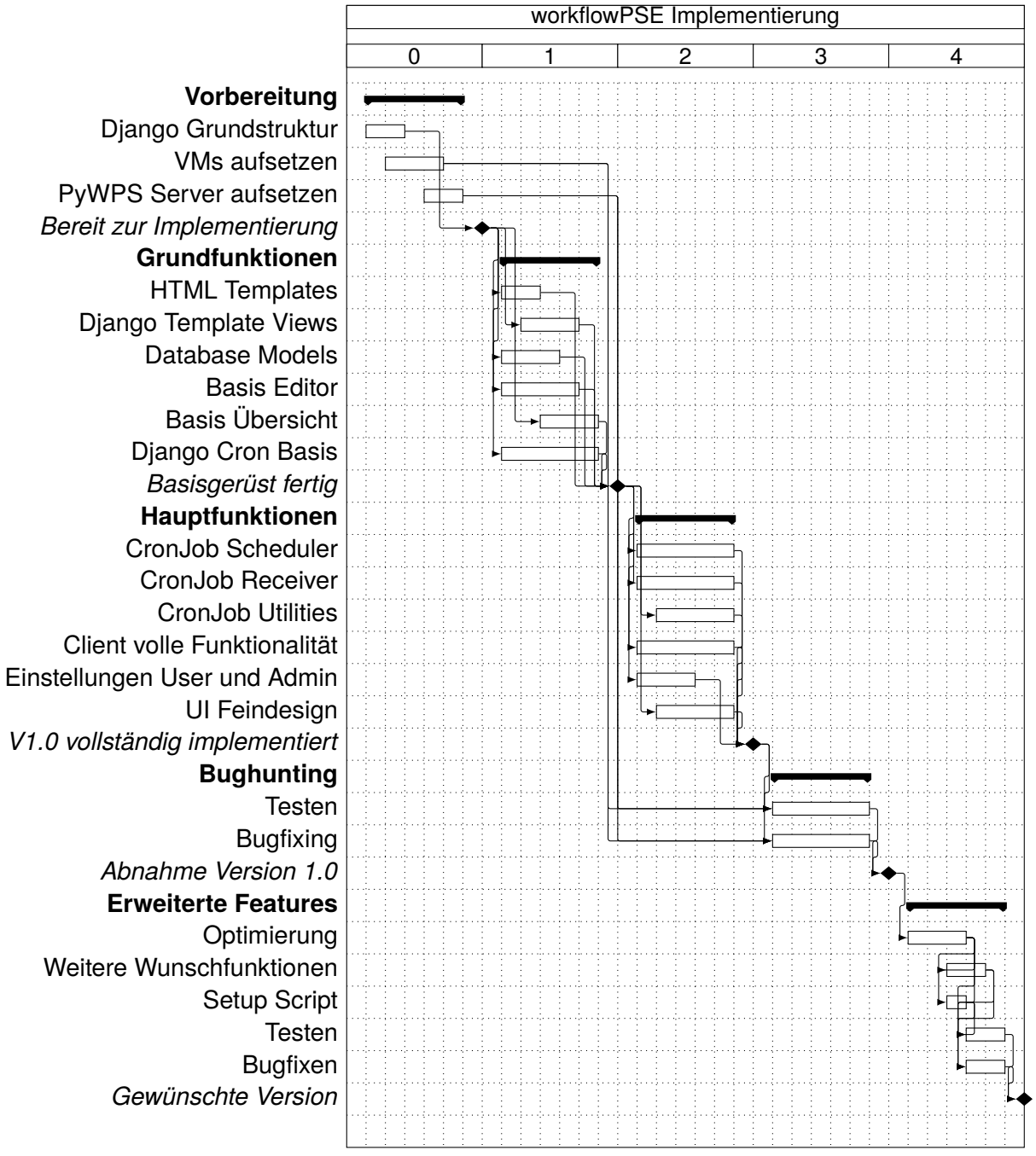
2.3 Zeitplan

Zeitlich war ein Ablauf innerhalb von 4 Wochen in den Blöcken Grundfunktionen, Hauptfunktionen, Bughunting und Erweiterte Features geplant. Nach einem schnellen Start mit dem Abschluss der Grundfunktionen vor der geplanten Zeit, stellte sich der Plan jedoch als sehr ambitioniert heraus. Für die eigentliche Implementierung wurden 3 Wochen anstatt von einer Woche benötigt und somit musste der Block Erweiterte Features herausgenommen werden. Das Testen wurde instinktiv in die Implementierung eingebunden, sodass hier kein klarer eigenständiger Block zu erkennen war.

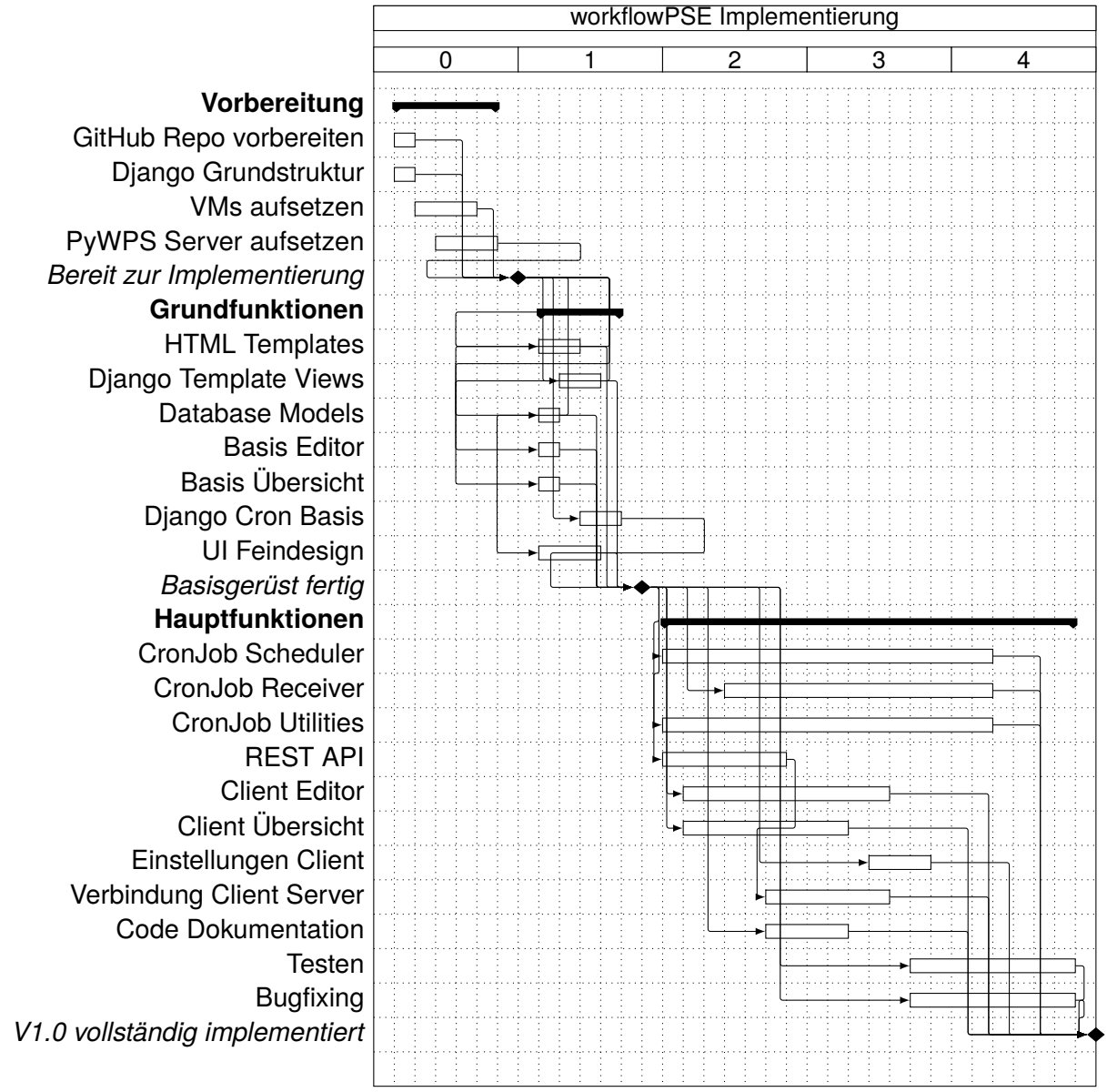
2.4 Verzögerungen

Da jeder der Teammitglieder die Entwicklung zeitgleich mit dem normalen Arbeitsaufwand der Uni vereinbaren musste, zögerte sich die Implementierung länger als zunächst gedacht heraus.

2.5 Gantt wie geplant



2.6 Gantt wie erfolgt



Aus den beiden Gantt-Diagrammen kann man leicht den oben beschriebenen Zeitplan und die Änderungen, die sich innerhalb der Implementierung ergeben haben, herauslesen.

3 Implementierte Features

3.1 Einleitung

In den meisten Projekten muss während der Entwicklung aufgrund unvorhergesehener Gegebenheiten so manches umstrukturiert werden.
Daher findet man in den folgenden Auflistungen die Kriterien die wir aus dem Pflichtenheft umgesetzt haben, sowie einige Features die wir für nützlich befanden und hinzugefügt haben, jedoch auch Punkte die als zu komplex, aufwendig oder wenig nützlich erachtet und somit verworfen wurden.

3.2 Musskriterien

- Verwalten von **WPS** Workflows
  - Erstellen, Bearbeiten, Speichern, Laden, Anzeigen von Workflows
  - Auflistung von Workflows mit Ausführungsstatus
  - Wiederherstellung der letzten Sitzung
  - Fehlermanagement
    - Editorprüfung auf Kompatibilität von Tasks innerhalb eines Workflows
  - Workflows und **Task**s auf Syntax und Kompatibilität prüfen
  - Erstellen und Bearbeiten in grafischem **Drag and Drop** Editor
  - Dynamisches einbinden von neuen **WPS** Tasks in den Editor
- Der Login-Status des Nutzers wird erfasst
- Nutzerfreundlichkeit
  - Einfache, intuitive Benutzung des Editors
- Open-Source
- Technische Dokumentation (Benutzeranleitung)

3.3 Wunschkriterien

- Logging aller Aktivitäten
- Detailansicht einzelner Workflows
- Ein extra Interface für Administratoren
- Erweiterte Konfigurationseinstellungen
- Installations-/Einrichtungsassistent

3.4 Zusätzlich

- Auflistung der Resultate
- Ausgabe von Fehlern
- Sortieren der Prozesse nach Server
- Beim Hinzufügen von neuen Servern wird automatisch validiert und falls nötig eine Fehlermeldung angezeigt
- Falls der Nutzer aktiv einen Workflow bearbeitet wird vom Client jede Sekunde eine Anfrage an den Server geschickt der daraufhin wieder die Status der Tasks überprüft und Änderungen zurück gibt
- Der Admin kann die Prozesse über einen Button in den Einstellungen aktualisieren, was ansonsten alle 5 Minuten geschieht

3.5 Nicht implementierte Kriterien

- Der Nutzer hat die Möglichkeit WPS konforme Dateien als Workflows hochzuladen
- Integrierte Ausführung der Workflows auf dem gleichen Server nur nach erfolgter Prüfung von bestimmten Kriterien, z.B. Serverlast
- Import und Export von Workflows

4 Unittests

In diesem Abschnitt werden alle Unittests des Systems aufgelistet und kurz beschrieben. Das Django Framework verlangt, dass die Bezeichnungen aller Testmethoden mit dem Präfix „test“ anfangen.
Da alle im Weiteren beschriebene Methoden Testmethoden sind, haben die alle den Rückgabetyp void und erwarten keine Eingabeparameter.

4.1 ParserTestCase

Innherhalb dieser Testklasse werden Methoden getestet, die für die Analyse der WPS Server Responses verantwortlich sind. Dabei werden zwei statische xml Testdateien verwendet, die mittels WPS getCapabilities und WPS describeProcesses Requests als Response von einem test Server zurückgegenen werden.
In der Methoden Beschreibungen wird der Einfachheit halber oft das Wort „Information“ benutzt. Genau Liste der Attributen die für das Erzeugen eines Objekts notwendig sind, finden Sie im Entwurfsdokument (Abschnitt 3.1.3 Models).

Name	Beschreibung
test_parse_service_provider_info	Testet, ob die Information über einen WPS Provider richtig aus einer xml Datei ausgelesen wird.
test_parse_service_provider_info_fail	Prüft die Ausnahmebehandlung beim Parsen der Information über einen service provider.
test_parse_wps_server_info	Testet, ob die Information über einen WPS Server richtig aus einer xml Datei ausgelesen wird.
test_parse_wps_server_info_fail	Prüft die Ausnahmebehandlung beim Parsen der Information über einen Server.
test_parse_wps_process	Es wird getestet, ob die Information über einen WPS Prozess richtig aus einer xml Datei ausgelesen wird.
test_parse_wps_process_fail	Prüft die Ausnahmebehandlung beim Parsen der Information über einen WPS Prozess.
test_parse_process_input_literal	Testet, ob die Information über einen Input mit Datentyp „LiteralData“ korrekt ausgelesen wird.
test_parse_process_output_literal	Es wird überprüft, ob die Information über einen Output mit Datentyp „LiteralOutput“ korrekt ausgelesen wird.

Name	Beschreibung
test_parse_process_input_complex	Es wird überprüft, ob die Information über einen Input mit Datentyp „ComplexData“ korrekt ausgelesen wird.
test_parse_process_output_complex	Es wird überprüft, ob die Information über einen Output mit Datentyp „ComplexOutput“ korrekt ausgelesen wird.
test_parse_process_input_bounding_box	Es wird überprüft, ob die Information über einen Input mit Datentyp „BoundingBoxData“ korrekt ausgelesen wird.
test_parse_process_output_bounding_box	Es wird überprüft, ob die Information über einen Output mit Datentyp „BoundingBoxOutput“ korrekt ausgelesen wird.
test_parse_process_input_fail	Prüft die Ausnahmebehandlung beim Parsen der Information über einen Input.
test_parse_process_output_fail	Prüft die Ausnahmebehandlung beim Parsen der Information über einen Output.

4.2 DatabaseTestCase

In dieser Testklasse werden Methoden getestet, die für die Arbeit mit einer relationalen Datenbank verantwortlich sind. Methoden mit dem Präfix „search“ werden verwendet, um Duplikate in der Datenbank zu eliminieren.

Name	Beschreibung
setUp	(Überschrieben) Es werden die Testdaten in die Datenbank geschrieben.
test_searc_provider_in_database	Sucht einen Service Provider in der Datenbank nach den Namen und der Webseite.
test_search_provider_in_empty_database	Prüft, ob im Falle der Abwesenheit des Providers in der Datenbank, eine Ausnahme geworfen wird.
test_search_server_in_database	Sucht einen Server in der Datenbank nach den Titel.
test_search_server_in_empty_database	Prüft, ob im Falle der Abwesenheit des Servers in der Datenbank, eine Ausnahme geworfen wird.
test_search_process_in_database	Sucht einen WPS Prozess in der Datenbank nach den Titel und Titel des dazugehörigen Servers.
test_search_process_in_empty_database	Prüft, ob im Falle der Abwesenheit des WPS Prozesses in der Datenbank, eine Ausnahme geworfen wird.
test_search_input_output_in_database	Sucht einen Input/Output in der Datenbank nach seinen Identifikator (String) und den dazugehörigen WPS Prozess.
test_search_input_output_in_empty_database	Prüft, ob im Falle der Abwesenheit des Inputs/Outputs in der Datenbank, eine Ausnahme geworfen wird.
test_overwrite_server	Testet, ob die Information über Server korrekt überschrieben wird.
test_overwrite_process	Testet, ob die Information über WPS Prozess korrekt überschrieben wird.
test_overwrite_input_output	Testet, ob die Information über Input/Output korrekt überschrieben wird.

4.3 CronTestCase

Innerhalb dieser Testklasse werden Methoden des Moduls cron.py getestet, durch einen Cron daemon in regelmäßigen Abständen ausgeführt werden.

Name	Beschreibung
setUp	(Überschrieben) Es werden die Testdaten in die Datenbank geschrieben.
tearDown	(Überschrieben) Es werden alle Datensätze aus der Datenbank gelöscht
test_send_task	Es wird ein Test Task ausgewählt und überprüft, dass man einen Response von Server bekommt und die status URL in die Datenbank geschrieben wird.
test_execution	Führt einen Task aus und überprüft, ob in der Datenbank ein richtiges Ergebnis steht.
test_update_wps_processes	Testet, ob die WPS Prozesse von einem im voraus gespeicherten WPS Server korrekt gesammelt und in der Datenbank gespeichert werden.
test_update_wps_processes_with_empty_database	Testet, ob bei einer leeren Datenbank (also, wenn es kein Server gespeichert ist) nichts gemacht wird.

5 Änderungen zum Entwurf

5.1 Änderungen am Client

Die Architektur des Clients nach der Implementierung entspricht der geplanten Architektur aus dem Entwurfsdokument. Auch die Anzahl der Klassen weicht nur leicht vom Entwurfsdokument ab. Die größte Abweichung liegt in der Komplexität der Klassen. Einige Methoden und Attribute sind neu dazugekommen, die nur schwer im Voraus zu planen waren.

5.1.1 Neue Komponenten

ProcessDialogComponent

Eine Detailansicht des ausgewählten Prozesses.

Attribute:

Name	Datentyp	Beschreibung
process	Process	Der angezeigte Prozess

ResultDialogComponent

Eine Auflistung der Resultate der aktuellen Ausführung.

Attribute:

Name	Datentyp	Beschreibung
workflow	Workflow	Ausgeführter Workflow
results	Artefact<'output'>[]	Liste aller Resultate

5.1.2 Erweiterte Komponenten

EditorComponent

Hinzugefügte Attribute:

Name	Datentyp	Beschreibung
movement	MovementData	Metadaten zur Mausbewegung

snapshots	Workflow[]	Änderungsverlauf des Workflows
taskComponents	QueryList<TaskComponent>	Liste der Tasks im aktuellen Workflow
workflowChanged	EventEmitter<Workflow>()	Wird bei jeder Änderung des Workflows ausgeführt
running	boolean	Gibt an, ob sich der aktuelle Workflow in der Ausführung befindet

Hinzugefügte Methoden:

Name	Rückgabetyp	Parameter	Beschreibung
clickEdge	void	edge: Edge	Wird bei klick auf eine Edge aufgerufen
scrollToMiddle	void	-	Zentriert das Sichtfeld des Editors
changeArtefact	void	-	Wird aufgerufen, wenn ein Artefact geändert wurde
snapshot	void	-	Erstellt ein Snapshot
undo	void	-	Ziegt den letzten Snapshot an

ProcessComponent

Hinzugefügte Methoden:

Name	Rückgabetyp	Parameter	Beschreibung
openDialog	void	-	Öffnet die Detailansicht

ProcessListComponent

Hinzugefügte Methoden:

Name	Rückgabetyp	Parameter	Beschreibung
processByWPS	void	id	Gibt alle Prozesse eines bestimmten WPS Servers zurück

TaskComponent

Hinzugefügte Attribute:

Name	Datentyp	Beschreibung
parameterDrag	EventEmitter	Hilfsfunktion für Drag and Drop
parameterDrop	EventEmitter	Hilfsfunktion für Drag and Drop
taskRemove	EventEmitter	Wird beim Löschen des Tasks aufgerufen
changeArtefact	EventEmitter	Wird beim Ändern eines Artefacts aufgerufen
running	boolean	Gibt an, ob sich der Task in der Ausführung befindet

Hinzugefügte Methoden:

Name	Rückgabetyp	Parameter	Beschreibung
hostContextmenu	void	-	Öffnet das Infomenü
openDetail	void	-	Öffnet die Detailansicht
addArtefact	void	ProcessParameter	Fügt ein Artefact dem Task hinzu
removeArtefact	void	ProcessParameter	Löscht das Artefact
hasArtefact	boolean	ProcessParameter	Gibt an, ob ein Artefact definiert ist

5.2 Änderungen am Server

Serverseitig wurde das System um das Modul utils.py erweitert, das grundsätzlich Funktionen für die Arbeit mit WPS Responses und der Datenbank gedacht ist. Funktionen sind leicht erweiterbar und können an anderen Stellen des Systems benutzt werden.

5.2.1 Änderungen an REST API

Method	Änderung	Beschreibung
GET /user	Hinzugefügt	Gibt den aktuellen Benutzer zurück
POST /process	Entfernt	Prozesse werden jetzt ausschließlich über das Hinzufügen von WPS Servern erstellt
PATCH /process/:id	Entfernt	Prozesse können aus Inkonsistenzgründen nicht mehr direkt geändert werden
DELETE /process/:id	Entfernt	Prozesse können nicht direkt entfernt werden
POST /login	Hinzugefügt	Loggt den Benutzer ein

5.2.2 edu.kit.scc.pseworkflow.cron

Hier wird nicht django-cron sondern django-crontab verwendet, da dies wesentlich bequemer zu implementieren war und unseren Ansprüchen besser entsprochen hat.

Methoden:

Name	Rückgabetyp	Parameter	Beschreibung
schedule	void	-	Hauptmethode der schedule Funktion. Hier wird der Workflow entsprechend der Reihenfolge der Tasks ausgeführt. Die XML Dateien werden erstellt und dem Servern zur Ausführung geschickt
xml_generator	void	string	Generiert die XML Datei für die auszuführede Tasks in dem übergebenen Pfad
create_data_doc	lxml.etree._Element/int	task	Hilfsmethode für xml_generator
send_task	void	task_id, xml_dir	Hilfsmethode für schedule. Sended Task an den WPS Server
get_execute_url	string	task	Hilfsmethode für send_task. Holt die Ausführungs URL aus der Datenbank.
receiver	void	-	Hauptmethode der receiver Funktion. Hier werden die Ausführungsstatus der laufende Tasks überprüft, und eventuelle Ergebnisse in die Datenbank geschrieben.
parse_execute_response	task	int	Hilfsmethode für receiver. Prüft die Antwort XML vom WPS Server des Tasks
parse_output	void	output, task	Hilfsmethode für parse_execute_response

parse_response_literaldata	void	artefact, data_elem	Hilfsmethode für parse_output
parse_response_bbox	void	artefact, data_elem	Hilfsmethode für parse_output
parse_response_complexdata	void	artefact, data_elem	Hilfsmethode für parse_output
calculate_percent_done	int	workflow	Berechnet den Prozentsatz fertig des Workflows
task_failed_handling	void	task, err_msg	Erzeugt ein ERROR Artefact, falls der Task fehlgeschlagen ist um den Client zu informieren
update_wps_processes	void	-	Hauptmethode um WPS Prozesse der vorhandenen WPS Server zu aktualisieren

5.2.3 edu.kit.scc.pseworkflow.models

Workflow

Attribute:

Name	Änderung	Datentyp	Beschreibung
shared	Hinzugefügt	boolean	Zeigt an ob der Workflow öffentlich ist oder privat ist

Status (Enum)

Name	Änderung	Beschreibung
NONE	Hinzugefügt	Der Task ist idle, bzw der Workflow wird vom Nutzer bearbeitet
READY	Geändert	In diesem Zustand wartet ein Task auf den Input eines noch nicht beendeten Tasks
WAITING	Geändert	Der Task ist bereit ausgeführt zu werden

WPS

Attribute:

Name	Änderung	Ersetzt	Rückgabetyp	Beschreibung
service_provider	umbenannt	wps_provider	WPSProvider	Der Besitzer des WPS Servers

InputOutput

Die Klasse InputOutput ist nicht mehr abstakt und übernimmt die Rolle der Klassen Input und Output, welche ursprünglich von InputOutput erben sollten. Die Klassen Input und Output wurden gestrichen.
Attribute:

Name	Änderung	Rückgabetyp	Beschreibung
format	Hinzugefügt	string	Das Format eines Inputs oder Outputs

Artefact

Die Klasse Artefact ist nicht mehr abstakt und übernimmt die Rolle der Klassen InputArtefact und OutputArtefact, welche ursprünglich von der abstrakten Klassse Artefact erben sollten. Die Klassen InputArtefact und OutputArtefact wurden gestrichen.

Attribute:

Name	Änderung	Rückgabetyp	Beschreibung
parameter	Hinzugefügt	InputOutput	InputOutput Parameter des Tasks

6 Statistiken

6.1 Zahlen

- Lines of Code: ca. 15000
- Commits: ca. 370

6.2 Graphen und Statistiken

Abschließend sind folgend ein paar Graphen aus GitHub und gitstats abgebildet, die die Verteilung von Commits und Zeilen Code pro Autor darstellen. Die Graphen werden durch die hohe Anzahl an Zeilen Einzelner, die viel automatisch generiert haben, leicht verfälscht. Insgesamt kann man erkennen, dass die Arbeitsaufteilung ungefähr gleich ist.

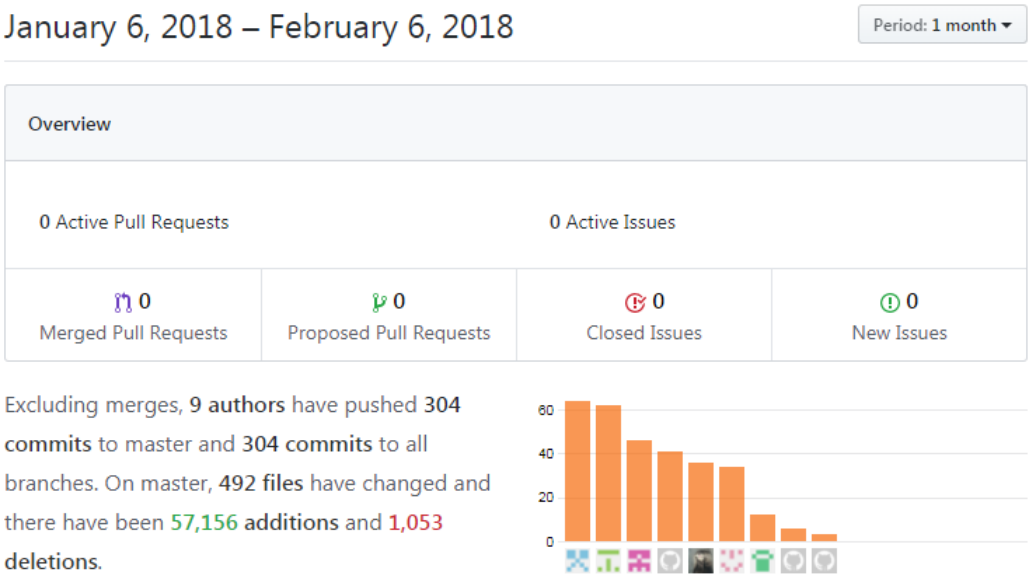


Abbildung 6.1: GitHub Insights - Pulse

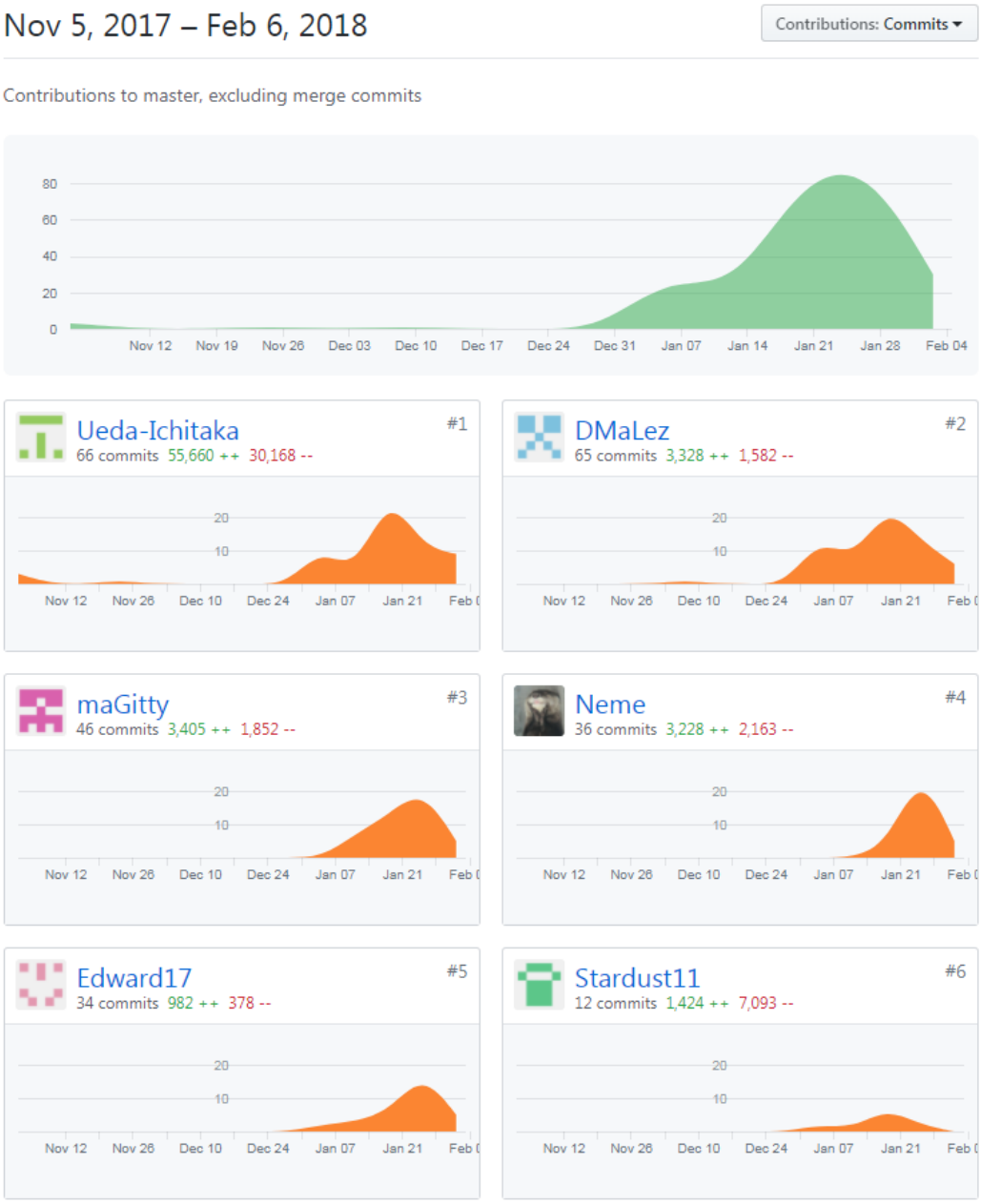


Abbildung 6.2: GitHub Insights - Contributors

Author	Commits (%)	+ lines	- lines	First commit	Last commit	Age	Active days	# by commits
Daniel Milbaier	101 (28.37%)	111746	83152	2017-12-11	2018-02-06	56 days, 21:12:33	17	1
Denis Gensh	80 (22.47%)	3238	1526	2017-12-12	2018-02-05	55 days, 1:14:11	18	2
Ueda-Ichitaka	79 (22.19%)	73810	30218	2017-11-05	2018-02-05	91 days, 21:47:01	23	3
Marcel Herm	50 (14.04%)	3270	1722	2018-01-16	2018-02-05	19 days, 18:49:56	16	4
Eduard Kukuy	30 (8.43%)	828	369	2018-01-18	2018-02-06	18 days, 10:01:24	11	5
Stardust11	13 (3.65%)	1424	7093	2018-01-09	2018-02-01	23 days, 1:34:51	7	6

Abbildung 6.3: GitStats - List of Authors

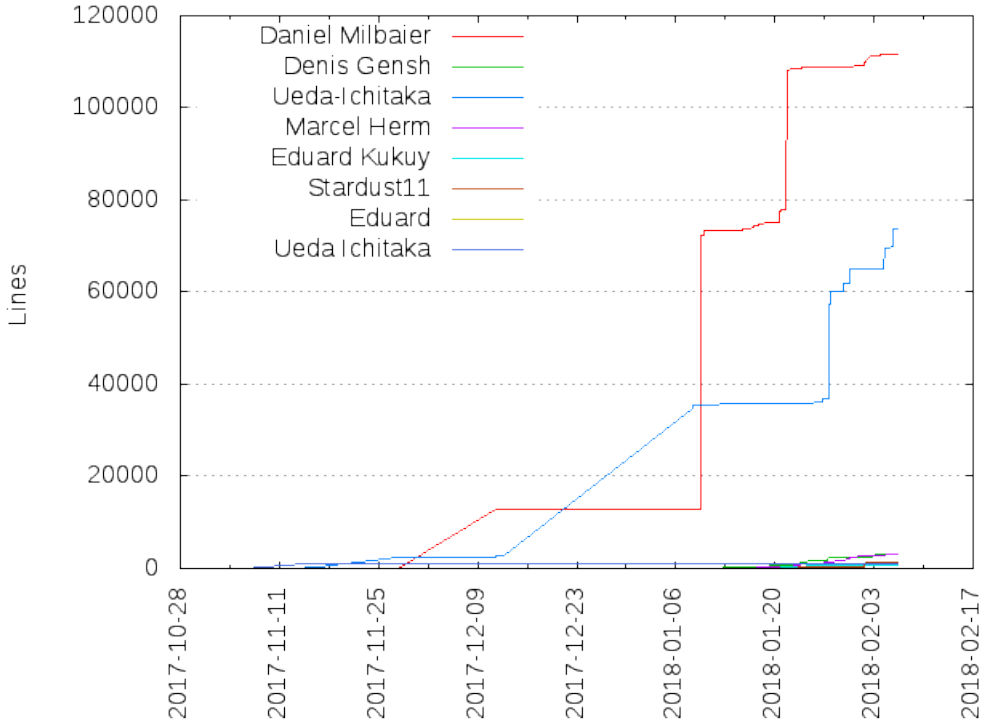


Abbildung 6.4: GitStats - Lines of Code per Author

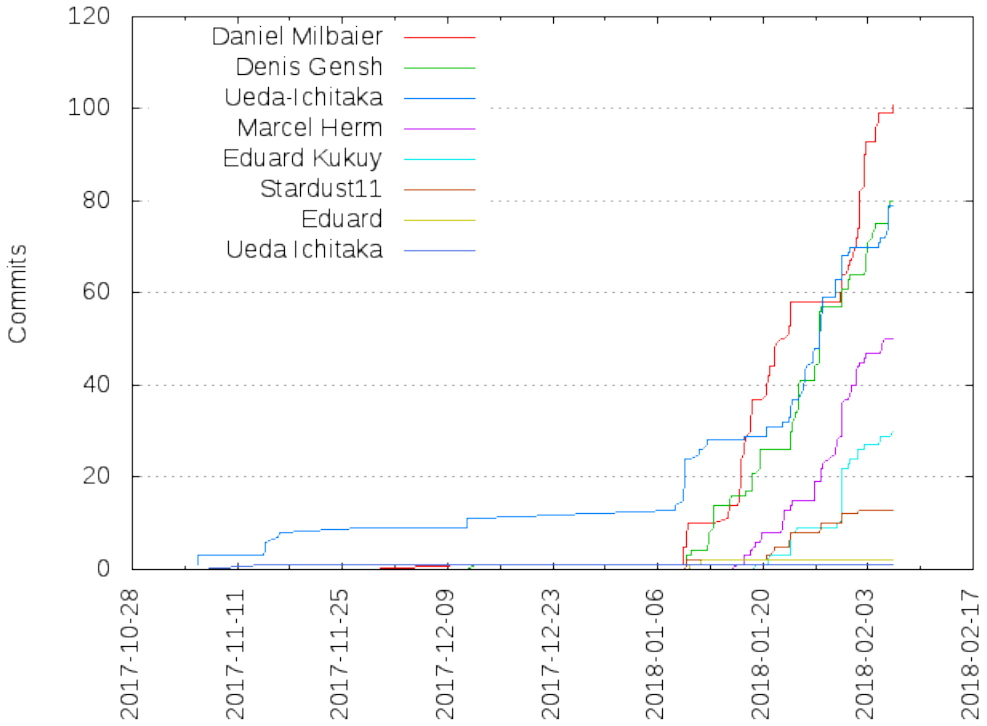


Abbildung 6.5: GitStats - Commits per Author

Glossar

Drag and Drop Methode zur Bedienung grafischer Benutzeroberflächen von Rechnern durch das Bewegen grafischer Elemente mittels eines Zeigegerätes.

Task Ein Schritt in einem Workflow.

Unittest Ein Modultest wird angewendet, um die funktionalen Einzelteile von Computerprogrammen auf korrekte Funktionalität zu prüfen.

WPS Der Web Processing Service (WPS) ist ein Protokoll, welches über das Internet eine räumliche Analyse von Geodaten durchführt.