



TECHNISCHE UNIVERSITÄT
CHEMNITZ

ENTWICKLERHANDBUCH



IntelliPhoto

Gruppe 3

8. Februar 2020

Inhaltsverzeichnis

1	Einleitung	2
1.1	Grundsätzliches	2
1.2	Anforderungsbeschreibung	2
1.3	Vorgehen	3
2	Requirements Engineering	3
2.1	Motivation	3
2.2	Volere-Snow-Cards	4
3	Design	4
3.1	Responsibility-Driven Design	5
3.2	UML-Diagramme	5
3.2.1	Klassen-Diagramm	6
3.2.2	Use-Case Diagramm	6
3.2.3	Sequenz Diagramme	8
3.2.4	Zustandsdiagramm	9
4	Implementierung	11
4.1	Programmiersprache und Libraries	11
5	Testen	12
5.1	Test Resultate	12
5.2	Bekannte Fehler und Probleme	12
6	Wartung	13
7	Installation	13
8	Schlusswort	14
9	Anhang	14
9.1	Volere-Snow-Cards	14
9.2	CRC-Karten	19

1 Einleitung

1.1 Grundsätzliches

IntelliPhoto ist ein Bild Editor der von unserer Gruppe im Rahmen des Software Engineering Praktikum im Wintersemester 2019/20 entworfen und programmiert wurde. Ziel des Praktikums war es sich mit den Schritten der heutigen Software Entwicklung auseinander zu setzen und erste Erfahrungen in diesem Bereich zu sammeln. Betreut wurde das Projekt von Prof. Janet Sigmund und Shadi Saleh.

Im folgenden sind die einzelnen Schritte des Softwarelebenszyklus für IntelliPhoto dokumentiert. Orientiert wird sich dabei an der das Praktikum begleitenden Vorlesung Software Engineering an der TU Chemnitz. Definitionen werden wenn nicht anders vermerkt dieser Vorlesung entnommen.

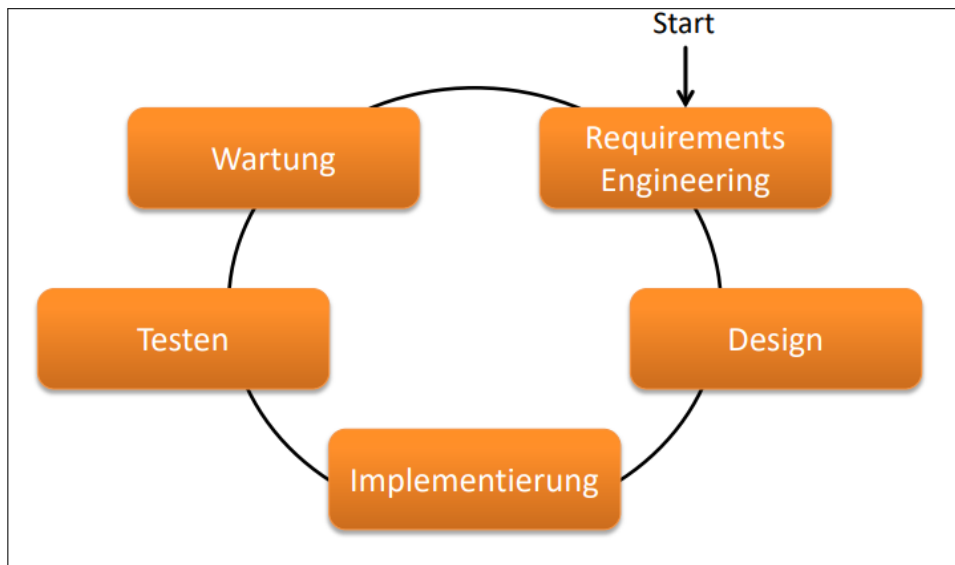


Abbildung 1: Softwarelebenszyklus nach Vorlesung

Es ist sinnvoll sich an diesen Zyklus zu halten, weil dadurch Probleme frühzeitig erkannt und behoben werden können. Die Kosten ein Problem zu beheben stiegen mit Fortschritt des Projekts.

1.2 Anforderungsbeschreibung

IntelliPhoto liegt die folgende Anforderungsbeschreibung, in der die Grundfunktionen des Editors spezifiziert werden, zu Grunde. Im Laufe des Projektes kamen weitere Wünsche und Anforderungen von "Kunden" hinzu. Diese werden an entsprechender Stelle vorgestellt.

Die neue Bildbearbeitungssoftware IntelliPhoto ist ein interaktives Tool zum Anzeigen und Bearbeiten von Bildern. Jedes Bild wird durch ein zweidimensionales Array von Bytes repräsentiert, wobei jeder Byte-Wert für einen Farbwert des Bildpunktes steht. Der Benutzer soll in der Lage sein die Bilddimensionen abzufragen. Es sollen zwei verschiedene Arten von Bildern repräsentiert werden können: **RasterImage** und **ShapedImage**, wobei letzteres eine Spezialform vom RasterImage ist. Ein ShapedImage besitzt eine nicht-rechteckige Form (Polygon), wobei die Bytes im Array angeben, ob die jeweiligen Punkte transparent oder opak dargestellt werden sollen. Darüber hinaus soll die Software einfache Manipulationen von Bildern erlauben. So soll das Drehen, als auch das Vergrößern und Verkleinern von Bildern, das Setzen neuer Farbwerte im Bild und das Zusammenfügen zweier Bilder zu einem neuen Bild innerhalb von 0.2 Sekunden möglich sein.

Abbildung 2: Anforderungsbeschreibung IntelliPhoto

Diese Beschreibung bildet den Kern des Projekts und wird im folgenden immer als Anforderungsbeschreibung referenziert. Auf Sie wird, insbesondere im Design und Requirements Engineering Abschnitt, oft verwiesen.

1.3 Vorgehen

Wie schon in 1.1 Grundsätzliches erwähnt wurde sich bei der Entwicklung von IntelliPhoto an der Vorlesung orientiert. Da neue Inhalte im Projekt erst angewendet wurden nachdem Sie in der Vorlesung erläutert wurden ergab sich für unsere Gruppe ein lineares voranschreiten durch den Softwarelebenszyklus. Aus diesem Grund können wir rückwirkend feststellen dass wir in erster Linie dem Wasserfallmodell zur Softwareentwicklung gefolgt sind.

2 Requirements Engineering

2.1 Motivation

Requirements beschreiben Bedingungen oder Eigenschaften, die ein System benötigt um Probleme zu lösen, Ziele zu erfüllen oder ein einen Vertrag/Standard zu genügen. Sie werden an ein Programm gestellt und spezifizieren dessen Eigenschaften und Funktionalitäten. Bestimmt werden Sie von den Stakeholdern, dieser Begriff bezeichnet die Gruppen die Interesse am Projekt hegen.

Requirements Engineering bezeichnet den Vorgang, diese Requirements zu finden und zu kategorisieren. Es ist ein wichtiger Schritt beim Software-Entwurf da eine eindeutige und überprüfbare Anforderungsbeschreibung entsteht. Es ist somit möglich Widersprüche zu identifizieren und eine vom Kunden eventuell in Fachsprache verfasste Beschreibung verständlich zu gestalten.

2.2 Volere-Snow-Cards

Requirements Engineering gliedert sich in vier Schritte: Anforderungsermittlung, Anforderungsanalyse, Anforderungsbeschreibung und Anforderungsrevision. Bei der Anforderungsermittlung werden Stakeholder- und Szenario basierend die Requirements identifiziert. In der folgenden Anforderungsanalyse werden diese in Funktionale, *was* soll das System leisten, und nicht-Funktionale, *wie* solle das System etwas leisten, Requirements unterteilt, sowie deren Erfüllbarkeit überprüft und Prioritäten zugeordnet. Die Anforderungsbeschreibung fasst obige Ergebnisse in genormter Form zusammen, in unserem Projekt nutzen wir dazu das Format der Volere-Snow-Card. Diese Beschreibung wird in der Anforderungsrevision erneut auf Probleme und Realisierbarkeit geprüft.

Im folgenden finden sich unsere Ergebnisse dieser Schritte als Volere-Snow-Cards. Da diese im Original jedoch kostenpflichtig sind (50\$), haben wir ein eigenes Template erstellt in Anlehnung an das in der Vorlesung gezeigte und erklärte:

Req-Id:	Req-Type:	Events/UCs:
Description:		
Rationale:		
Originator:		
Fit Criterion:		
Costumer Satisfaction:	CostumerDissatisfaction:	Priority:
Supporting Material:	Conflicts:	
History:		

Abbildung 3: Volere-Snow-Card Template nach Vorlesung

Dabei liegt *Costumer Satisfaction*, *Costumer Dissatisfaction* und *Priority* eine Skala von 0-9 wobei 0 kein und 9 hohes Gewicht hat.

Die konkreten Karten finden sie im Anhang unter 9.1.

3 Design

Generelles Ziel ist es eine Struktur zu finden die einfach zu verstehen und zu implementieren ist, Möglichkeiten zur Erweiterung bietet und wenn korrekt implementiert die Realisierung aller Anforderungen garantiert. Die Idee ist dabei das Programm in einzelne miteinander interagierende aber von einander unabhängige Objekte aufzuteilen und diesen Funktionen zuzuweisen. Dies nennt man Objekt-Orientiertes Dekomposition und wurde von uns beim designen des Programms verwendet. Es sei angemerkt dass dies ein iterativer Prozess ist, sich also Ergebnisse auch nach der Design Phase ändern können.

Im folgenden finden sich also unsere finalen Ergebnisse zum Zeitpunkt der Abgabe.

3.1 Responsibility-Driven Design

Eine Technik ein Objekt-Orientierten Designs zu realisieren ist mittels Responsibility-Driven Design (RRD). Das RRD gliedert sich in eine initiale Exploration, bei der Klassen, Verantwortlichkeiten und Kollaborationen identifiziert werden, und eine detaillierte Analyse, bei der Ergebnisse konkretisiert werden und nach Klassenhierarchien gesucht wird um Klassen eine vollständige Signatur zu geben.

Die Ergebnisse der ersten Phase können mit CRC-Karten (stellvertretend für Candidates-Responsibility-Collaboration-Karten) einheitlich visualisiert werden. Das Design dieser Karten nach Vorlesung ist wie folgt:

Klasse	
Funktion	Kollaborationen
Funktion	Kollaborationen
Funktion	Kollaborationen

Abbildung 4: Volere-Snow-Card Template nach Vorlesung

Die Karten für unseren Fall finden Sie unter dem Punkt 9.2 im Anhang.

Die Ergebnisse nach der detaillierten Analyse können auf verschiedene Weise dargestellt werden. Die Informationen die für eine vollständige Signatur benötigt werden können etwa auf der Rückseite der jeweiligen CRC-Karte vermerkt werden¹. Klassenhierarchien können zum Beispiel mit Venn Diagrammen abgebildet werden. Eine Möglichkeit die beiden Informationen in einem gemeinsamen Diagramm vereint darstellt, ist das im nächsten Absatz für unser Projekt vorgestellte UML-Klassendiagramm.

3.2 UML-Diagramme

UML (Unified Modeling Language) repräsentiert den internationalen Industriestandard für grafische Modellierungssprache. Sie ermöglicht komplexe Designs in Form von Diagrammen zu visualisieren. Der große Vorteil an UML ist das es hinreichend gut dokumentiert und standardisiert ist, dass heißt dass Diagramme ohne gesonderte Erklärung der Bedeutung ihrer Symbole

¹siehe <https://de.wikipedia.org/wiki/Class-Responsibility-Collaboration-Karten>

verständlich sind sofern Sie dem UM-Standard genügen. Sollte der Betrachter mit diesem nicht vertraut sein ist es ein einfaches für ihn die Bedeutung heraus zu finden. Deshalb wird im folgenden auf Erklärung der Diagramme verzichtet da diese dem Standard genügen.

3.2.1 Klassen-Diagramm

Das vermeintlich wichtigste UML-Diagramm ist das Klassendiagramm, da in diesem kompakt alle Klassen mit ihrer Signatur und alle Interaktionen abgebildet werden können. Unserer UML-Diagramm für IntelliPhoto basiert auf der detaillierten Analyse, dem zweiten Schritt des vorhergegangenen RDD.

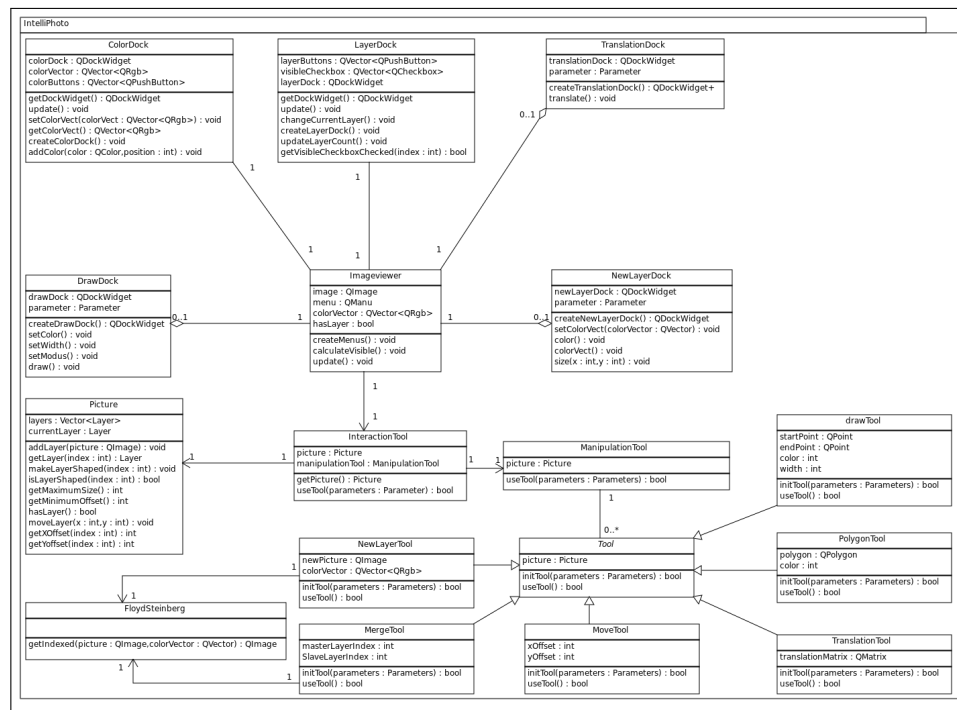


Abbildung 5: UML-Klassendiagramm

Für bessere Auflösung ist eine 'Klassendiagramm.svg'-Datei beigelegt.

3.2.2 Use-Case Diagramm

Auch können mit UML Use-Case Diagramme erstellt werden. Ein Use-Case ist eine generische Beschreibung einer gesamten Transaktion, welche mehrere Aktoren involviert. Ein Use-Case Diagramm präsentiert eine Menge von Use-Cases (Ellipsen) und deren externe Aktoren, die mit dem System interagieren. Das Anwendungsverhalten verschiedener Nutzer Gruppen ist uns wie

folgt gegeben worden:

Da Sie damit beauftragt wurden die neue Bildbearbeitungssoftware IntelliPhoto zu implementieren, führten Sie eine Umfeldanalyse durch. In dieser haben Sie wertvolle Informationen über verschiedene Nutzergruppen sammeln können. So erfuhren Sie, dass Casual User und Einsteiger die Software hauptsächlich für kurze Aufgaben wie das Zusammenschneiden von Bildern, das Ändern der Bildauflösungen und dem Drehen von Bildern benutzen wollen. Außerdem möchten die Casual User die Software dazu benutzen, um bestimmte Regionen in einem Bild zu retuschieren. Eine weitere Nutzergruppe, die freiberuflichen Fotografen, hingegen möchten neben der Bildretusche auch eine Reihe an Korrekturwerkzeugen, wie der Helligkeit/Kontrast, Farbton/Sättigung und den Gradationskurven, als auch Auswahlwerkzeuge und verschiedene Pinsel haben. Die letzte Gruppe von potentiellen Benutzern, die 3D Künstler, wünschen sich eine Schnittstelle für den Import von gängigen 3D-Dateien. Auch soll es für sie möglich sein, einfache geometrische 3D-Objekte direkt im Bild zu erzeugen. Jede Nutzergruppe gab an, dass sie sich eine Ebenendarstellung in der Software vorstellen können und benutzen würden.

Abbildung 6: Ergebnisse für Anwendung Intelli Photo

Durch verarbeiten dieser Informationen kommen wir zu folgenden exemplarischen zwei Use-Case Diagrammen für IntelliPhoto:

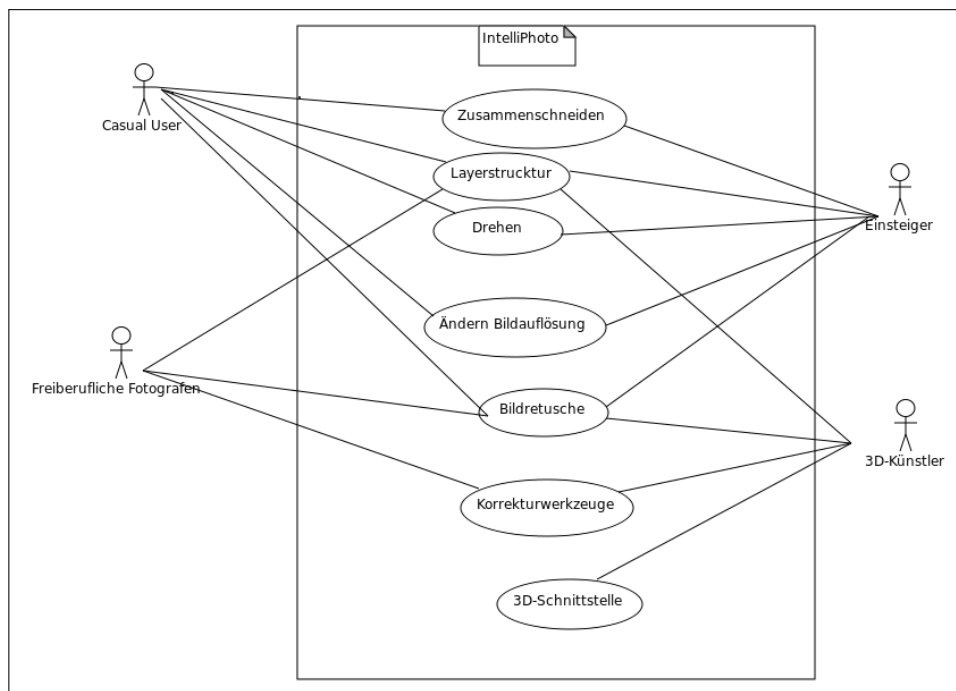


Abbildung 7: Generelles Use-Case-Diagramm

Die ist ein sehr generelles Diagramm welches das Nutzer Verhalten aller Gruppen dokumentiert. Die Anforderungen an 'Korrekturwerkzeuge' und '3D-Schnittstelle' lassen sich näher spezifizieren:

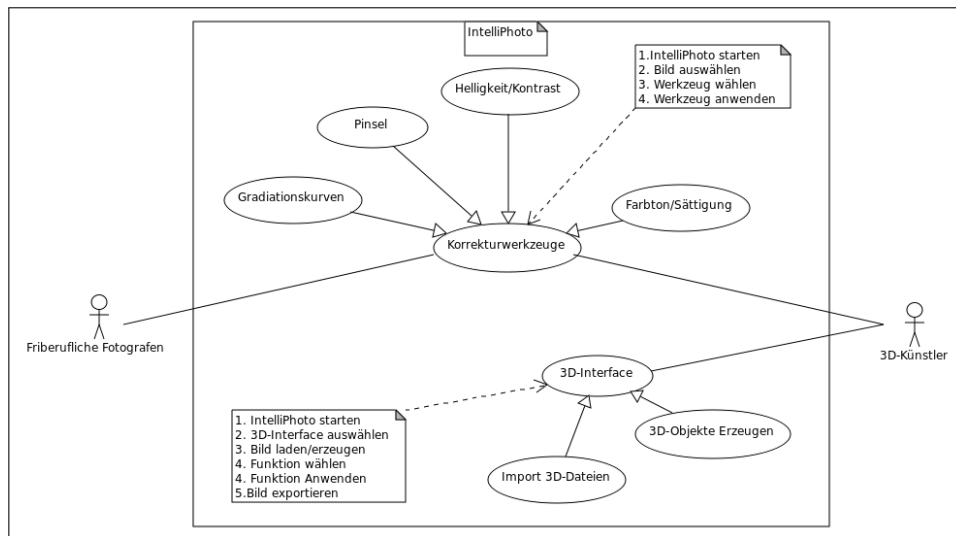


Abbildung 8: Detailliertes Use-Case-Diagramm

Auch diese Diagramme sind zu besserer Sichtbarkeit als UseCaseGeneral.svg und UseCaseSpezial.svg beigefügt.

3.2.3 Sequenz Diagramme

Ein Szenario ist eine Instanz von einem Use Case, das ein typisches Beispiel einer Ausführung zeigt. Szenarien können durch UML Sequenzdiagramme dargestellt werden. Ein Sequenzdiagramm beschreibt ein Szenario durch das Zeigen von Interaktionen zwischen einer Menge von Objekten in einer zeitlichen Abfolge. Objekte (keine Klassen!) werden als vertikale Balken gezeichnet. Events oder Nachrichtensendungen werden als horizontale (oder schräge) Pfeile vom Sender zum Empfänger gezeichnet. Im folgenden zeigen wir zwei Sequenz Diagramme für IntelliPhoto. Im ersten ist das Diagramm für das Szenario ein Bild Shaped zu machen, im zweiten für das Szenario ein neues Layer zu erstellen dargestellt.

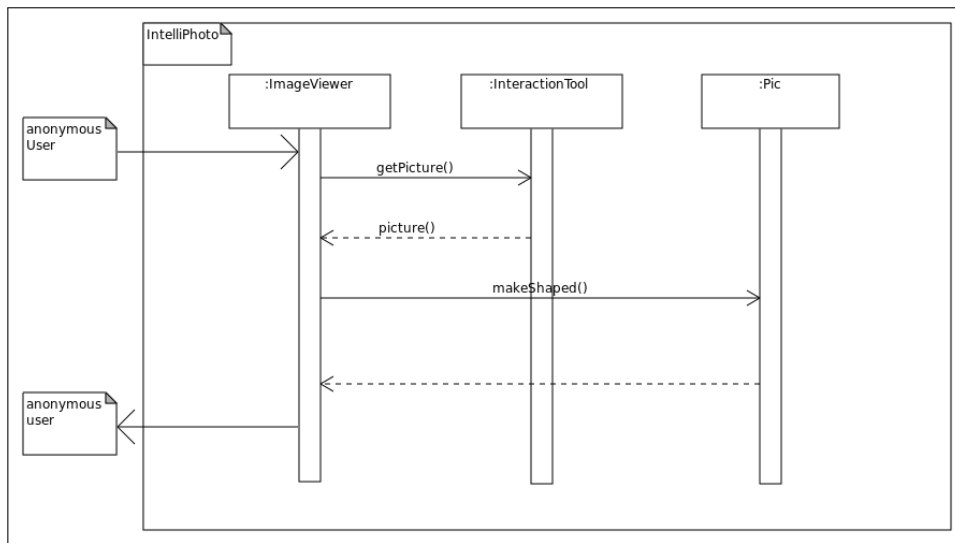


Abbildung 9: Sequenz Diagramm Shaped Image

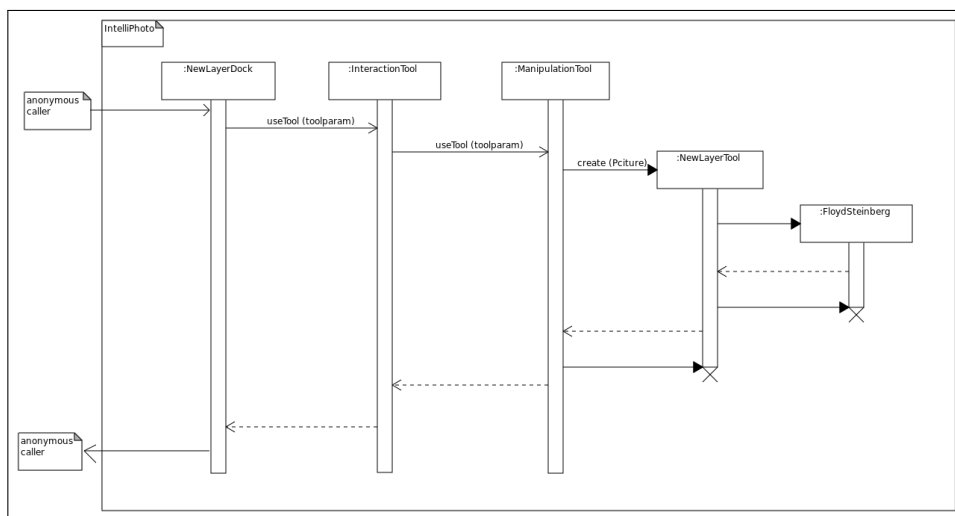


Abbildung 10: Sequenz Diagramm neues Layer

Auch diese Diagramme sind zur besseren Sichtbarkeit als NewLayerSD.svg und MakeShapedSD.svg beigefügt.

3.2.4 Zustandsdiagramm

Ein Zustandsdiagramm beschreibt die zeitliche Evolution eines Objektes von einer gegebenen Klasse in Abhängigkeit von Interaktionen mit anderen Objekten innerhalb und außerhalb des Systems.

Im folgenden ist exemplarisch das Zustandsdiagramm für ein Objekt der Klasse Picture und ein Objekt der Klasse DrawDock dargestellt.

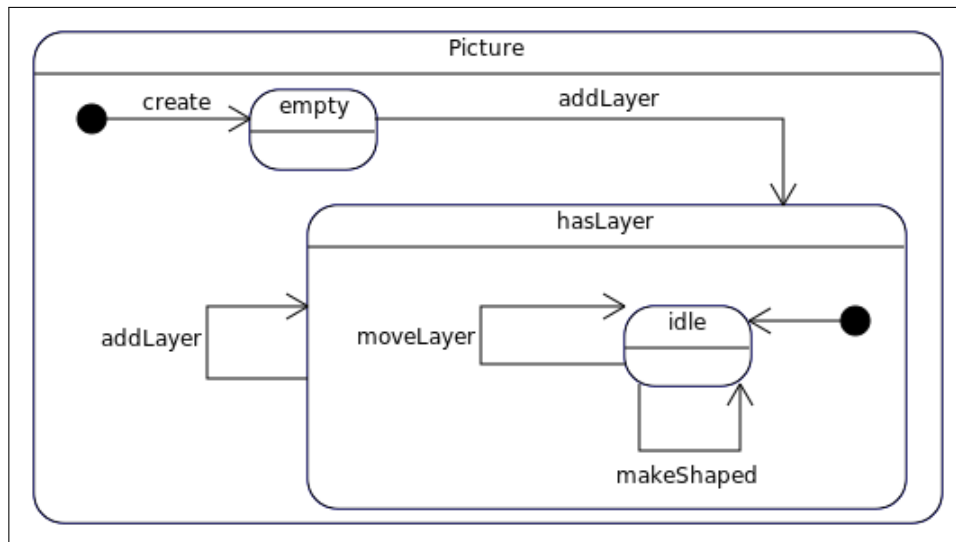


Abbildung 11: Zustands Diagramm Picture

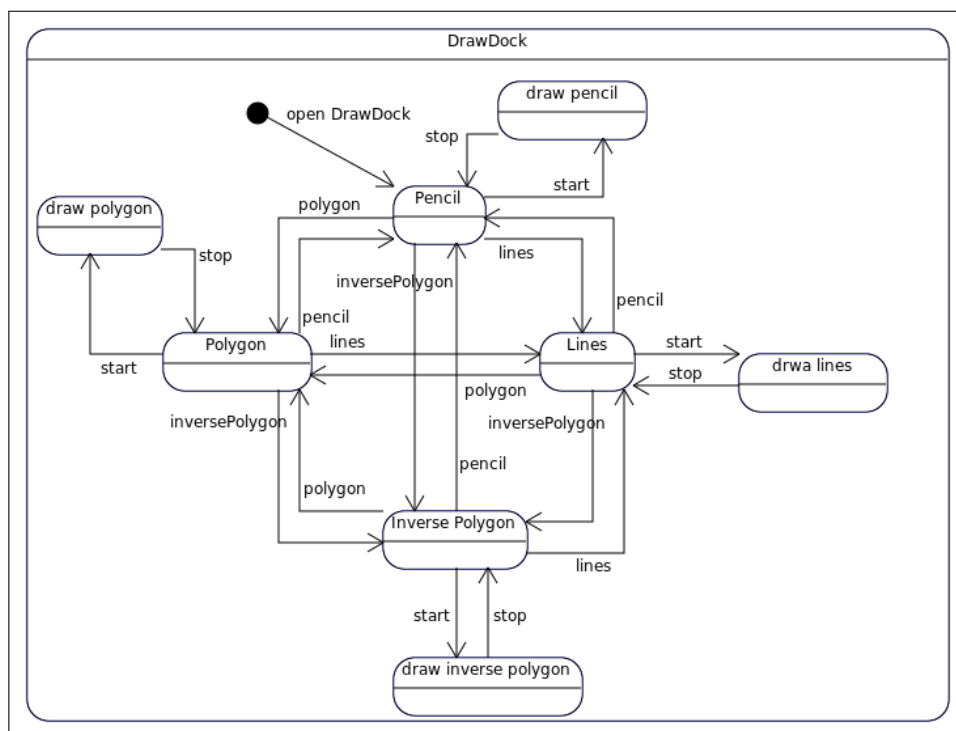


Abbildung 12: Zustands Diagramm DrawDock

Auch diese Diagramme sind zur besseren Sichtbarkeit als DrawDockZD.svg und PictureZD.svg beigelegt.

4 Implementierung

4.1 Programmiersprache und Libraries

Wir haben uns bei der Implementation von IntelliPhoto für die Programmiersprache C++ entschieden. C++ bietet die Möglichkeit zum Objekt orientierten Programmieren was ein für uns nach der Designphase (mit Objekt orientierte Dekomposition) ein essentielles Programmierparadigma war. C++ bietet weiterhin die Vorteile dass es Plattform unabhängig ist, über eine starke Community, im Sinne von vielen Libraries und Foren, verfügt und eine sehr performante Sprache ist. Letzteres war insbesondere im Anbetracht der Zeit-Constraints in der Anforderungsbeschreibung ein wichtiger Grund für die Entscheidung. Das einzige Problem dass C++ mit sich bringt ist dass es keinen Garbage Collector besitzt, es kann also durch unaufmerksames Programmieren schnell viel Speicher von der Anwendung benötigt werden.

Die Kern Library unseres Programms ist Qt5. Qt ist ein open source Anwendungsframework und GUI-Toolkit zu plattformübergreifenden Entwicklung von Programmen. Grund für diese Entscheidung war zunächst die gute Kompatibilität mit C++, so ist Qt in dieser Sprache implementiert und setzt selbst einen Objekt orientierten Ansatz um. Zum anderen wird Qt stark in real Welt Anwendungen genutzt, etwa beim KDE-Plasma ². Die für uns letztlich wichtigsten Faktoren für die Entscheidung waren die Existenz einer Qt eigene Wikipedia ³ und die mit 'Qt Creator' frei verfügbare IDE speziell für die Entwicklung mit Qt. Beides war für uns als Neulinge in der GUI-Programmierung sehr hilfreich und hat dazu beigetragen, dass wir das Projekt umsetzen konnten.

Zu Unser Implementierung muss man leider sagen dass diese teilweise sehr abstrakt ist. Verschärft wird dies dadurch dass wir aus Zeitgründen und aufgrund von Abstimmungsproblemen nur wenig Kommentare im Code haben. Geplant war ursprünglich eine Code Dokumentation mit Doxygen, diese fiel aufgrund des ersten Punktes aber aus.

²[https://en.wikipedia.org/wiki/Qt_\(software\)#Qt_in_use](https://en.wikipedia.org/wiki/Qt_(software)#Qt_in_use)

³<https://wiki.qt.io/Main>

5 Testen

5.1 Test Resultate

Im folgenden finden sich unsere Testresultate. Wir haben uns dafür entschieden manuelle Tests auszuführen sowie Zeitmessungen im Code durchzuführen. Bei den Tests gefundene Fehler haben wir nach Möglichkeit korrigiert, die verbliebenen uns bekannten sind unten aufgeführt.

Die Zeitmessung wiederum testen die Kernfunktionen von IntelliPhoto und messen deren Zeit. Dies ist wichtig um zu überprüfen ob die zeitliche Schranke eingehalten wird. Getestet haben wir auf einem Lenovo Thinkpad L570, DualCore, 8 Gb Ram.

Die Resultate sind im folgenden tabellarisch festgehalten:

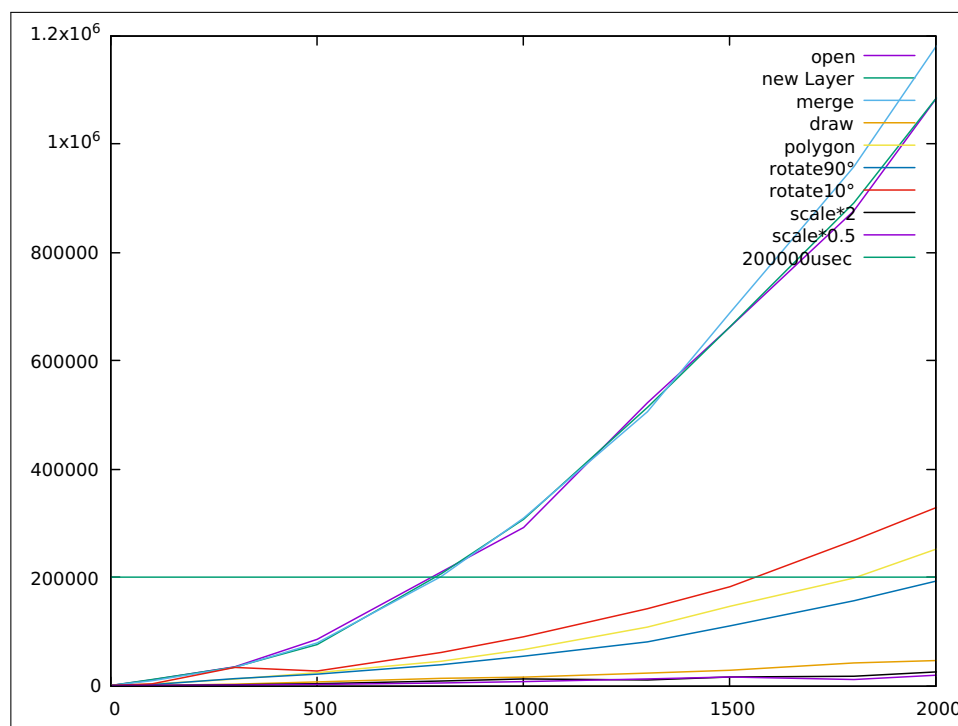


Abbildung 13: Testzeiten

An der y-Achse ist die Zeit in Mikro-Sekunden abgetragen auf der x-Achse die Größe eines Bildes (gleich für x/y).

5.2 Bekannte Fehler und Probleme

Die folgenden Bugs und anderer Probleme sind bekannt und werden bei Abgabe des Projektes noch nicht behoben sein:

- Absturz nach Abbruch des Öffnen: Ist noch kein Layer geöffnet und man klickt 'Open' und beendet den sich öffnenden File-Browser ohne ein Bild zu laden, so stürzt das Programm ab.
- Ein wiederholtes drehen eines Layers führt zu einem sich wellenden Rand, dieser verkleinert das Bild. Außerdem wird das Layer in immer größere transparente Umgebungen eingebettet
- Eine schnelles hintereinander ausführen von Translationen führt dazu dass das alte Layer nicht mehr gelöscht wird
- Wird im Draw-Dock eine Draw-Funktion außer Pencil aktiviert und das Dock anschließend geschlossen, so ist beim erneuten öffnen des Draw-Docks zwar Pencil ausgewählt, die vorherige aktivierte Draw-Funktion ist aber weiter aktiv

6 Wartung

Da IntelliPhoto im Rahmen eines Vorlesung begleitenden Praktikums erstellt wurde, also keine Veröffentlichung des Programms und damit reale Nutzung geplant sind, sowie die Verantwortlichkeiten der Gruppenmitglieder mit Abgabe des Projektes enden, ist es nicht geplant Wartungstätigkeiten auszuführen. Dies soll jedoch keines Falls bedeuten diese wäre in einem anderen Szenario nicht notwendig. Uns ist klar, dass eine Wartung des Programmes allein wegen der bereits bekannten Fehler vonnöten wäre, die im Rahmen des Projektes nicht behoben werden konnten. Außerdem hat das Programm mit hoher Wahrscheinlichkeit eine Vielzahl, trotz testen, noch unbekannter Fehler, die erst in der extensiven Nutzung erkannt und gewartet werden müssten.

Eine wichtige Erweiterung von IntelliPhoto die im Zusammenhang mit Wartungsarbeiten ausgeführt werden sollte, ist das Entwerfen und Implementieren eines eigenen Bildformats für IntelliPhoto, welches in der Lage ist die Sperrpixel zu speichern durch die ShapedImages realisiert werden. Dadurch würde das oben angesprochene Problem des Unterbrechen eines Projektes mit ShapedImages behoben werden.

7 Installation

Der Code wurde auf Linux Systemen entwickelt und ist mit Sicherheit auf diesen lauffähig. Er ist aber wahrscheinlich auch auf anderen Betriebssystemen lauffähig. Nachdem Sie das Programm erhalten haben, entweder über den Kontakt Professur Softwaretechnik der Tu Chemnitz, Softwareengineering Praktikum WS 19/20 Gruppe 3 oder jeremias.piljug@s2017.tu-chemnitz.de führen Sie unter einem Linux System einfach die in der READ-

ME beschreibenen Schritte aus.

Bei Fragen oder Problemen wenden sie sich an einen der obigen Kontakte.

8 Schlusswort

IntelliPhoto ist das erste größere Software Projekt an dem jedes unserer Gruppenmitglieder gearbeitet hat. Uns ist bekannt dass das Resultat viel zu wünschen übrig lässt und vieles nicht so funktioniert wie es sollte. Dennoch haben wir, gerade aus der Menge unserer Fehler, eine Menge gelernt und sind recht zufrieden mit dem Ergebnis, für ein erstes Projekt.

Viel Spaß mit dem Programm,

Pascal Schröter, Jeremias Piljug, Dongze Yang und Xiangyu Tong
Gruppe 3

9 Anhang

9.1 Volere-Snow-Cards

Req-Id: 1	Req-Type: 1	Events/UCs: 1
Description: Setzen neuer Farbwerte im Bild		
Rationale: Essentiell für Bildbearbeitungs Programm		
Originator: Benutzer jeglicher Art, Auftraggeber		
Fit Criterion: Erfüllt wenn beliebige im rgba-Farbraum darstellbare Farbe ausgewählt werden und damit mindestens ein Pixel gefärbt werden kann		
Costumer Satisfaction: 9	CostumerDissatisfaction: 8	Priority: 8
Supporting Material: Anforderungsbeschreibung	Conflicts: keine	
History: Erstellt: 30.10.19; Bearbeitet: Schröter 21.12.19;		

Req-Id: 2	Req-Type: 1	Events/UCs: 2
Description: Drehen eines Bildes		
Rationale: Essentiell für Bildbearbeitungs Programm		
Originator: Benutzer jeglicher Art, Auftraggeber		
Fit Criterion: Wenn sich die Drehung eines beliebigen Bildes um einen beliebigen Winkel realisieren lässt		
Costumer Satisfaction: 9	CostumerDissatisfaction: 6	Priority: 7
Supporting Material: Anforderungs- beschreibung		Conflicts: keine
History: Erstellt: 30.10.19; Bearbeitet: Schröter 29.12.19;		

Req-Id: 3	Req-Type: 1	Events/UCs: 3
Description: Zusammenfügen zweier Bilder zu einem neuen Bild		
Rationale: Essentiell für Bildbearbeitungsprogramm mit Layer Konzept um verschiedene einzeln bearbeitete Bilder zu vereinigen		
Originator: Benutzer jeglicher Art, Auftraggeber		
Fit Criterion: Erfüllt wenn sich zwei beliebiger Bilder zu einem neuen Bild vereinigen lassen und im neuen Bild die sichtbaren Anteile beider Bilder wieder sichtbar sind		
Costumer Satisfaction: 9	CostumerDissatisfaction: 5	Priority: 6
Supporting Material: Anforderungsbeschreibung		Conflicts: keine
History: Erstellt: 30.10.19; Bearbeitet: Piljug 04.01.20;		

Req-Id: 4	Req-Type: 1	Events/UCs: 4
Description: Erstellen von Bildern in Form eines beliebigen Polygons, so genannte ShapedImages		
Rationale: Alleinstellungsmerkmal IntelliPhoto		
Originator: Auftraggeber		
Fit Criterion: Erfüllt wenn sich beliebige Polygon förmige Bilder erstellen lassen. Die Bereiche die ehemals zum Bild gehört haben sollen nicht sichtbar und beschreibbar sein		
Costumer Satisfaction: 9	CostumerDissatisfaction: 9	Priority: 9
Supporting Material: Anforderungsbeschreibung		Conflicts: 1 - Färben der Bereiche außerhalb nicht erlauben
History: Erstellt: 30.10.19; Bearbeitet: Schröter 15.01.20;		

Req-Id: 5	Req-Type: 2	Events/UCs: 1,2,3
Description: Zugehörige Events und Vergrößern/Verkleinern in <0.2 Sekunden		
Rationale: Für Nutzer flüssiges Programm		
Originator: Benutzer jeglicher Art, Auftraggeber		
Fit Criterion: Erfüllt wenn die Zeit zwischen stellen der Anforderung im Programm und erfüllen der Anforderung höchstens 0.2 Sekunden beträgt		
Costumer Satisfaction: 3	CostumerDissatisfaction: 9	Priority: 8
Supporting Material: Anforderungsbeschreibung		Conflicts: 7 - mögliche Leistungsminderung durch GUI
History: Erstellt: 30.10.19; Bearbeitet: Schröter 20.01.20;		

Req-Id: 6	Req-Type: 2	Events/UCs: 1,4
Description: Bilder repräsentiert als 1 Byte Array		
Rationale: Alleinstellungsmerkmal IntelliPhoto		
Originator: Auftraggeber		
Fit Criterion: Für den Farbwert jedes Pixel des Bildes nur 1 Byte		
Costumer Satisfaction: 9	CostumerDissatisfaction: 9	Priority: 9
Supporting Material: Anforderungsbeschreibung		Conflicts: 1 - Potentiell nicht jede Farbe möglich
History: Erstellt: 30.10.19; Bearbeitet: Schröter 03.12.19;		

Req-Id: 7	Req-Type: 2	Events/UCs: 1,2,3,4
Description: Interaktives Tool = GUI		
Rationale: Auf graphischer Ebene nutzbar zu sein ist wichtig um verständlich zu machen. Außerdem müssen Bilder angezeigt werden		
Originator: Benutzer jeglicher Art, Auftraggeber		
Fit Criterion: Alle Interaktionen mit der Anwendung passieren in der GUI		
Costumer Satisfaction: 7	CostumerDissatisfaction: 9	Priority: 9
Supporting Material: Anforderungsbeschreibung		Conflicts: 4 - Schwierig darstellbar
History: Erstellt: 30.10.19; Bearbeitet: Piljug 22.11.19;		

Req-Id: 8	Req-Type: 2	Events/UCs: 7
Description: Plattformübergreifende Nutzbarkeit		
Rationale: Wichtig für Kundenreichweite		
Originator: Benutzer des jeweiligen Betriebssystems, Auftraggeber		
Fit Criterion: Auf mehr als einem Betriebssystem nutzbar		
Costumer Satisfaction: 6	CostumerDissatisfaction: 1	Priority: 1
Supporting Material: Anforderungsbeschreibung		Conflicts: 7 - Fraglich ob universell
History: Erstellt: 30.10.19; Bearbeitet: Piljug 30.01.20;		

9.2 CRC-Karten

interaktives Tool	
Interpretieren Nutzer Input	
Funktionen aufrufen	Färber,Dreher,Skalierer Transformator,Erzeuger Zusammenfüger

Heißt in unserer Implementation *InteractionTool*.

GUI	
Bilder anzeigen	Bild
Nutzer Input aufnehmen	

Heißt in unserer Implementation *Imageviewer*.

Färber	
Bildbereiche färben	Bild,Pixel

Heißt in unserer Implementation *drawTool*.

Skalierer	
Bilder skalieren	Bild

Heißt in unserer Implementation *TranslationTool*.

Dreher	
Bilder drehen	Bild

Heißt in unserer Implementation *TranslationTool*.

Bild	
Farbwerte speichern	Pixel
Dimensionen speichern	

Heißt in unserer Implementation *Picture*.

Farbwert	
Einen Rgba-Farbwert speichern	
Einen Sperrbereich speichern	

Wird über Eigenschaft von *QImage* verwaltet, dies ist eine Qt Klasse.

Transformator	
Images zu ShapedImages konvertieren	Bild,Färber,Farbwert

Wird bei uns mit Sperrpixeln und *DrawTool* umgesetzt.

Erzeuger	
Neues Bild erzeugen	Bild,Pixel,Transformator

Heißt in unserer Implementation *NewLayer*.

Zusammenfüger	
Bilder zusammenfügen	Bild,Pixel,Transformator Erzeuger

Heißt in unserer Implementation *MergeTool*.