FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Entwicklung einer graphischen Benutzeroberfläche zur übersichtlichen Darstellung komplexer, mechanischer Anlagen samt Berechnung anlagenspezifischer Schadensfrequenzen

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur/in

im Rahmen des Studiums

Wirtschaftsingenieurwesen-Informatik

eingereicht von

Michael Buchheit
Matrikelnummer 9825157

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer/in: Univ.-Prof. Dr. techn. Dipl.-Ing. Johann Wassermann

Wien, 25.11.2015

Buchheit Michael
(Unterschrift Verfasser/in)

(Unterschrift Betreuer/in)

FAKULTÄT FÜR INFORMATIK
Faculty of Informatics

Bestätigung über die Abfassung der Diplomarbeit

Name der Betreuerin/des Betreuers: ...JOHANN WASSERMANN...

Name der/des Studierenden: ....MICHAEL BUCHHEIT.....

Die Betreuerin/Der Betreuer und die Verfasserin/der Verfasser der Diplomarbeit bestätigen mit ihrer Unterschrift, dass Sie die Richtlinien der Fakultät für Informatik zur Verfassung einer Diplomarbeit sowie zur Erstellung des Diplomarbeitsposters (siehe: http://www.informatik.tuwien.ac.at/dekanat/abschluss-master) zur Kenntnis genommen haben und darüber informiert wurden, dass deren Nicht-Einhaltung die Abweisung der Diplomarbeit bzw. des Posters zur Folge haben kann.
Die Betreuerin/Der Betreuer und die Verfasserin/der Verfasser nehmen weiters zur Kenntnis, dass eine Sperre der Diplomarbeit nur in besonders begründeten Fällen und nur im Ausmaß von max. einem Jahr genehmigt werden kann (Die Sperre muss im eigenen Interesse der/des Studierenden liegen und ist vor einer etwaigen vertraglichen Geheimhaltungsverpflichtung durch den/die Studiendekan/in genehmigen zu lassen).

.....
Unterschrift der Betreuerin/des Betreuers

Buchheit Michael
.....
Unterschrift der/des Studierenden

TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

Ich habe zur Kenntnis genommen, dass ich zur Drucklegung meiner Arbeit unter der Bezeichnung

Diplomarbeit

nur mit Bewilligung der Prüfungskommission berechtigt bin.

Ich erkläre weiters Eides statt, dass ich meine Diplomarbeit nach den anerkannten Grundsätzen für wissenschaftliche Abhandlungen selbstständig ausgeführt habe und alle verwendeten Hilfsmittel, insbesondere die zugrunde gelegte Literatur, genannt habe.

Weiters erkläre ich, dass ich dieses Diplomarbeitsthema bisher weder im In- noch Ausland (einer Beurteilerin/einem Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe und dass diese Arbeit mit der vom Begutachter beurteilten Arbeit übereinstimmt.

Wien, im November 2015

Buchheit Michael

Dipl.-Ing. Michael Buchheit

Abstract

As part of a project financed by external fundings (FFG and Simonsfeld AG) at the Institute of Mechanics and Mechatronics, Department of Measurement and Actuator, at the Technical University of Vienna a graphical user interface (GUI) was created, which enables the user to create models of machines (in particular transmissions), to perform specifications of assemblies, bearings and sensors, and to calculate important parameters such as rotational speed and meshing frequencies. With the use of the program generated models respectively model data are stored in form of semi-structured XML-data records. The, with the program generated, model data records will be threaded in by a further, also at the institute developed, program, which focus on condition monitoring of machines, to localize damaged components of monitored machines. For the development of the program were, in addition to a good knowledge of a programming language, knowledge in Software Engineering, UML and XML necessary. Furthermore, for the communication with the people involved, the design and development of the software, detailed knowledge in the field of mechanical engineering was required. This concerned, among other things, Knowledge of the standardized representation of transmission parts, the calculation of the bearing characteristics and the shaft speeds. The following describes the development from the ideation for the project to the implementation and testing of the developed software. The document is suitable for both information scientists as well as mechanical engineers.

Kurzfassung

Im Rahmen eines von Drittmitteln finanzierten Projektes (FFG und Simonsfeld AG) am Institut für Mechanik und Mechatronik, Abteilung Messtechnik und Aktorik, an der Technischen Universität Wien wurde eine graphische Benutzeroberfläche (GUI) geschaffen werden, die es dem Nutzer ermöglicht Maschinen (im Speziellen Getriebe) zu modellieren, Spezifikationen von Baugruppen, Lagern, Sensoren durchzuführen und wichtige Parameter, wie zum Beispiel Drehzahlen und Zahneingriffsfrequenzen, zu berechnen. Die durch das Programm erstellten Modelle bzw. deren Modelldaten werden in Form von semi-strukturierten XML-Datensätzen gespeichert. Die, mit dem Programm erstellten, Modelldatensätze werden schließlich von einem weiteren, an diesem Institut, entwickelten Softwarepaket, dessen Fokus auf der Zustandsüberwachung von Maschinen liegt, eingelesen und für die Lokalisierung von Schäden an Komponenten der zu überwachenden Anlagen eingesetzt. Für die Entwicklung des Programms waren neben guten Kenntnissen in einer Programmiersprache auch Kenntnisse in Softwareengineering, UML und XML notwendig. Des Weiteren war für die Kommunikation mit den beteiligten Personen, dem Entwurf und der Entwicklung der Software Detailwissen aus dem Bereich Maschinenbau erforderlich. Dies betraf u.a. Wissen über die normgerechte Darstellung von Getriebeteilen, die Berechnung der Lagerkennwerte und der Wellendrehzahlen. Im Folgenden wird die Entwicklung von der Ideenfindung für das Projekt bis zur Implementierung und dem Testen des erzeugten Softwareproduktes beschrieben. Die Arbeit ist sowohl an interessierte Informatiker|innen als auch Maschinenbauer|innen gerichtet.

Inhalt

1 Vorwort

1.1 Inhalt der Arbeit

Im Rahmen eines von Drittmitteln finanzierten Projektes (FFG und Simonsfeld AG) am Institut für Mechanik und Mechatronik, der Abteilung Messtechnik und Aktorik, an der Technischen Universität Wien, sollte eine graphische Benutzeroberfläche (GUI) entwickelt werden, die es dem Nutzer ermöglicht Maschinen (im Speziellen Getriebe) zu modellieren, Spezifikationen von Baugruppen, Lagern, Sensoren durchzuführen und wichtige Parameter wie zum Beispiel Drehzahlen und Zahneingriffsfrequenzen zu berechnen. Das Programm dient als wichtige Schnittstelle zu einem weiteren, an diesem Institut entwickelten Softwarepaket, dessen Fokus auf der Zustandsüberwachung von Maschinen liegt.

1.2 Ausbildung und berufliche Erfahrung des Autors

1993 - 1998	HTL für Maschinenbau in Eisenstadt
1998 - 2004	Diplomstudium Wirtschaftsingenieurwesen Maschinenbau an der Technischen Universität Wien
2005 - 2010	Konstrukteur bei Bombardier in Wien
Seit 2010	Masterstudium Wirtschaftsingenieurwesen-Informatik
Seit 2012	Doktorat der technischen Wissenschaften an der Technischen Universität Wien
Seit 2012	Projektassistent am Institut für Mechanik und Mechatronik

1.3 Danksagung

Ich möchte an dieser Stelle meinen Betreuern Professor Dr. Johann Wassermann und Dr. Erwin Quintus für ihre Unterstützung danken.

2 Beschreibung des Vorgehens

Wie bereits in Vorwort erwähnt, bestand die zentrale Aufgabe dieser Arbeit ein System beziehungsweise eine Anwendersoftware zur Modellierung von Getrieben zu entwickeln. Im Folgenden soll die Entwicklung von der Ideenfindung für das Projekt bis zur Implementierung und dem Testen des erzeugten Softwareproduktes beschrieben werden.

3 Idee/Ziel des Projektes

Für Wartung und Instandhaltung von Maschinen werden heute verschiedene Strategien eingesetzt. Zustandsüberwachung stellt hierbei die technisch anspruchsvollste Strategie des Portfo-

lios dar. Die Maschine wird hierfür mit einer Reihe von Sensoren zur Messung der im Betrieb auftretenden Schwingungen ausgestattet. Die gemessenen Schwingungen werden über einen bestimmten Zeitraum aufgezeichnet. Zur Auswertung der aufgezeichneten Daten werden spezielle Analysealgorithmen eingesetzt. Die Analyse kann je nach Ausführung einfach (Aussagen betreffend diverser statistischer Kennwerte) bis komplex (Berechnung des Cepstrums, Hüllkurvenanalysen, Zeit-Frequenzanalysen) sein. Im Rahmen eines Projektes sollte am Institut für Mechanik und Mechatronik, Abteilung Messtechnik und Aktorik ein Softwarepaket entwickelt werden, mit dessen Hilfe die Qualität der Zustandsüberwachung von Maschinen deutlich verbessert werden kann. Parallel zur eigentlichen Analysesoftware musste ein System entwickelt werden, das das Modellieren von Getrieben in einfacher Art und Weise ermöglicht und wichtige maschinenspezifische Informationen, wie etwa Lagerschadensfrequenzen berechnet und für die folgende Analyse zur Verfügung stellt. Im Rahmen dieser Diplomarbeit wurde das Modelliertool „GetriebeStudio“ entwickelt; es befindet sich seit Anfang 2014 beim Projektpartner erfolgreich im Einsatz.

4 Stakeholder

Als „Stakeholder“ werden Personen bezeichnet, welche ein berechtigtes Interesse am Verlauf eines Projekts und beziehungsweise oder dessen Ergebnissen haben. Man unterscheidet hierbei (unternehmens-)interne und externe Stakeholder. Zu den internen Interessenten zählen unter anderem die Eigentümer, das Führungspersonal und die Mitarbeiter des Unternehmens. Zu den wichtigsten externen Stakeholdern sind die Kunden, Lieferanten, Gläubiger, der Staat und die Gesellschaft zu zählen.

Für dieses Projekt konnten im Wesentlichen folgende Stakeholder identifiziert und zur Mitarbeit bzw. zum Mitwirken im Projekt gewonnen werden: Projektleiter, Projektmitarbeiter, Financiers und zukünftige Nutzer des Systems.

Der Projektleiter hatte die Aufgabe, die Anforderungen der Financiers und Nutzer zu ermitteln und in einem Pflichtenheft zu dokumentieren.

Es stellte sich heraus, dass der Kooperationspartner ein System zur Zustandsüberwachung seiner Anlagen bereits im Einsatz hatte, dieses jedoch nicht den Ansprüchen genügte. Viele Fehler (z.B.: Lagerschäden, Risse an Zahnrädern, etc.) konnten erst entdeckt werden als es bereits zu spät war und die Anlage zur Reparatur unplanmäßig ausgeschaltet werden musste. Aus Sicht des Financiers ist als wichtigste Anforderung an das Gesamtprojekt eine frühe Erkennung eines möglichen Schadens zu nennen.

Die Nutzer können in drei Gruppen eingeteilt werden:

1. Experten, welche für die Modellierung der Anlagen zuständig sind (Nutzer des Modellierungstools GetriebeStudio).
2. Experten für die Fehlerfrüherkennung (Nutzer der Zustandsüberwachungssoftware SPECTIVE[1]).

[1] SPECTIVE – Software for Powerful Envelope analysis and Calculation of Traces for Immediate Visualization of Emergencies [40]

3. Sonstige Nutzer, welche sich etwa über den allgemeinen Zustand einer Anlage möglichst schnell einen Überblick verschaffen möchten (Nutzer des Tools HealthMonitor).

Die Anforderungen der Nutzer des Teilsystems GetriebeStudio werden im Folgenden dargestellt.

5 Anforderungen an das Teilsystem GetriebeStudio

Die Anforderungen an das System sind vielschichtig und sollen im Folgenden beschrieben werden.

Funktionale „Muss“-Anforderungen:

1. Es soll mittels einer grafischen Benutzeroberfläche möglich sein, Modelle von gängigen Getrieben in vereinfachter Darstellung zu erzeugen.
2. Die Modelle sollen in geeigneter Form manipulierbar sein. Dies schließt das Erzeugen, Löschen, Verknüpfen, Anpassen bzw. Spezifizieren der Bauteile ein. Zu den verwendeten Bauteilen zählen: Wellen, Lager, Zahnräder, etc.
3. Laden und Speichern der Modelldaten.
4. Berechnung der für das Analyseprogramm (nicht Teil dieser Arbeit) nötigen Parameter und Bereitstellung der Daten in einer XML-Datei.
5. Beim Anlegen der Lager (Zuweisen des Herstellers, Art, Bezeichnung und Schadensfrequenzen) sollen diese in einem Lagerkatalog (XML-Datenblatt) abgespeichert werden.
6. Bevor ein Lager neu angelegt und abgespeichert wird, muss kontrolliert werden, ob das Lager nicht bereits angelegt worden ist.
7. Das Zuweisen der Lagerdaten zu den Lagern soll mittels einer Suchmaschine möglich sein. Hierzu gibt man die genaue Lagerbezeichnung des Herstellers ein. Lager, die eine ähnliche Bezeichnung haben, sollen ebenfalls angezeigt werden (Ergänzung, siehe Kapitel 12).
8. Die Verknüpfungen zwischen den Teilen und zwischen Baugruppen und Bauteilen sollen, zweckmäßig etwa in Form einer Tabelle angezeigt werden. Nicht verknüpfte (lose) Teile und andere Verknüpfungsfehler sollen, zur Erleichterung der Fehlersuche, ebenfalls angegeben werden.
9. Die Auswahl bzw. Anpassung der (primären) Farbe zur Darstellung der Bauteile und Baugruppen soll mit Hilfe einer einfachen Farbpalette erfolgen.
10. Bei der manuellen Eingabe von Daten sollen diese auf Korrektheit geprüft werden. Wird beispielsweise eine Zahl vom Typ Double erwartet, so darf diese Eingabe keine Buchstaben enthalten.

Funktionale „Kann“-Anforderungen:

1. Zusätzlich zur Möglichkeit Baugruppen zu erzeugen, soll es auch möglich sein Standardbaugruppen zu generieren und zu spezifizieren. Beispielhaft könnte ein Planeten-

getriebe, das aus vielen Einzelteilen modelliert wird, in einer Standardbaugruppe zusammengefasst werden (wurde im Verlauf des Projektes umgesetzt).

Nicht funktionale Anforderungen:

1. Die Größe der modellierten Getriebe ist überschaubar (max. 100 Teile).

Anforderungen an die Dokumentation:

Die Dokumentation umfasst alle nötigen Unterlagen für Installation und Inbetriebnahme. Der Anwender soll den Umgang mit dem Softwarepaket z.B. anhand eines Tutorial erlernen. Ein Hilfemenü soll dem Anwender den Umgang mit dem Programm erleichtern.

6 Getriebe

6.1 Allgemein

Für die Anforderungsanalyse sowie der Entwicklung des Modelliertools „GetriebeStudio“ war es zunächst von großer Bedeutung, Wissen über die Funktion, Kinematik und Darstellung der zu modellierenden Getriebe, deren Formen und Komponenten zu sammeln.

Es soll hier geklärt werden, was man unter einem Getriebe versteht und wozu sie dienen. Hierzu gibt es eine Vielzahl von Definitionen, von denen die folgenden zwei exemplarisch herausgegriffen wurden.

Laut **Duden** [1] wird der Begriff Getriebe wie folgt erläutert:

Getriebe sind: *„Vorrichtungen in Maschinen oder Ähnlichem, die Bewegungen übertragen und die Maschine oder Ähnlichem funktionstüchtig macht.“*

Technisch konkreter ist hierzu die **Norm VDI 2127**, in der Getriebe wie folgt definiert werden:

„Getriebe dienen zur Übertragung und Umformung (Übersetzung) von Bewegung, Energie und/oder Kräften.“

Der Einsatzbereich für Getriebe ist vielseitig. Bekannte Beispiele sind u.a.:

- **Fahrradgetriebe:** Die Pedalkraft wird in eine Drehbewegung gewandelt. Abhängig von der (Getriebe-)Übersetzung gelangt dabei mehr oder weniger Antriebskraft an das Hinterrad.

- **Fahrzeuggetriebe**, ein moderner PKW besitzt u.a.:
  - Lenkgetriebe: Wandelt die Lenkraddrehbewegung in eine translatorische Bewegung, welche ihrerseits die Räder in die gewünschte Richtung schwenkt.
  - Antriebsgetriebe: Wandelt das Motordrehmoment und -drehzahl.
  - Differenzial: Sorgt dafür, dass die Antriebsräder bei Kurvenfahrt unterschiedlich schnell drehen.

6.2 Übertragungs-/Umformprinzipien

Für die Übertragung und/oder Umformung von Bewegung, Energie und Kräften können folgende Prinzipien unterschieden werden: **Mechanische**, **hydraulische, pneumatische** und **elektromagnetische** Prinzip [2].

6.2.1 Mechanisches Prinzip

Die Übertragung bzw. Umformung erfolgt über feste Körper. Je nachdem wie die Kraft mechanisch übertragen wird, unterscheidet man formschlüssige und kraftschlüssige Getriebe. Meist werden Zahnräder zur Übertragung der wirkenden Kräfte bzw. Momente eingesetzt. Weitere wichtige Übertragungselemente sind Reibräder, Ketten und Riemen.

6.2.1.1 Getriebeformen[2]

Zahnradgetriebe (nach [3], Seite 643): „Zahnradgetriebe bestehen aus einem oder mehreren Zahnradpaaren, die vollständig oder teilweise von einem Gehäuse umschlossen sind."

Ein maximales Übertragungsverhältnis[3] von 1:8 bzw. Untersetzung von 8:1 pro Stufe bzw. Zahnradpaar [3] sollte nicht überschritten werden. Für höhere Übersetzungen ist die Stufenbauweise zu empfehlen.

Merkmale: Kompakte Bauweise, hoher Wirkungsgrad, starre Kraftübertragung, bei hoher Drehzahl unerwünschte Schwingungen (kann durch bessere Verzahnungsqualität reduziert werden).

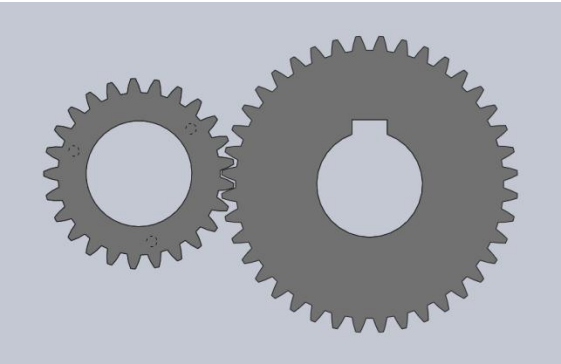

Abbildung 1: Zahnradpaar im Eingriff (erstellt mit SolidWorks; geradverzahnte Stirnräder, Zahnanzahl 25/40)

Reibradgetriebe: Ähnlich Zahnradgetriebe; nutzt Reibräder statt Zahnräder zur Übertragung der Kräfte. Im Gegensatz zur formschlüssigen Bauweise eines Zahnradgetriebes, werden die Kräfte zwischen den Reibrädern durch Kraftschluss übertragen. Das Drehmoment wird hierbei durch Reibung zwischen den Rädern übertragen.

Stufengetriebe: Die Übertragung bzw. Umformung erfolgt in mehreren Stufen. Man findet sie häufig in Kombination mit anderen Getriebeformen. Die Gesamtübersetzung ergibt sich aus dem Produkt der jeweiligen Stufenübersetzung.

[2] Die nachfolgende Aufzählung der Getriebeformen ist nicht taxativ.

[3] Verhältnis der Eingangsdrehzahl zur Ausgangsdrehzahl einer Stufe bzw. des gesamten Getriebes

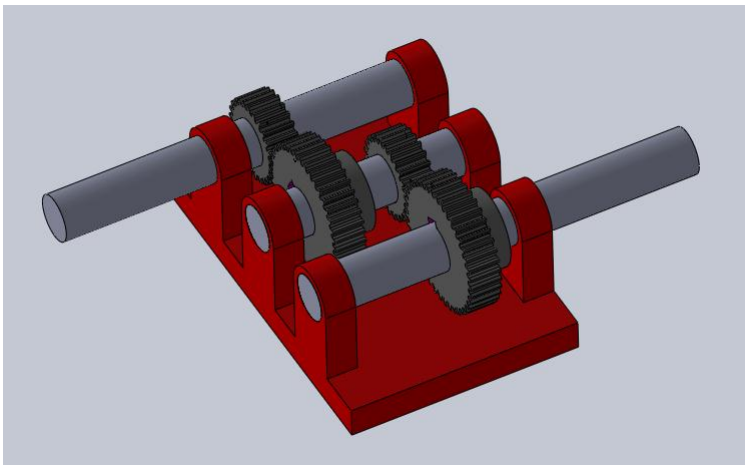

Abbildung 2: Modell eines zweistufigen (Zahnrad-)Getriebes (erstellt mit SolidWorks, 2013)

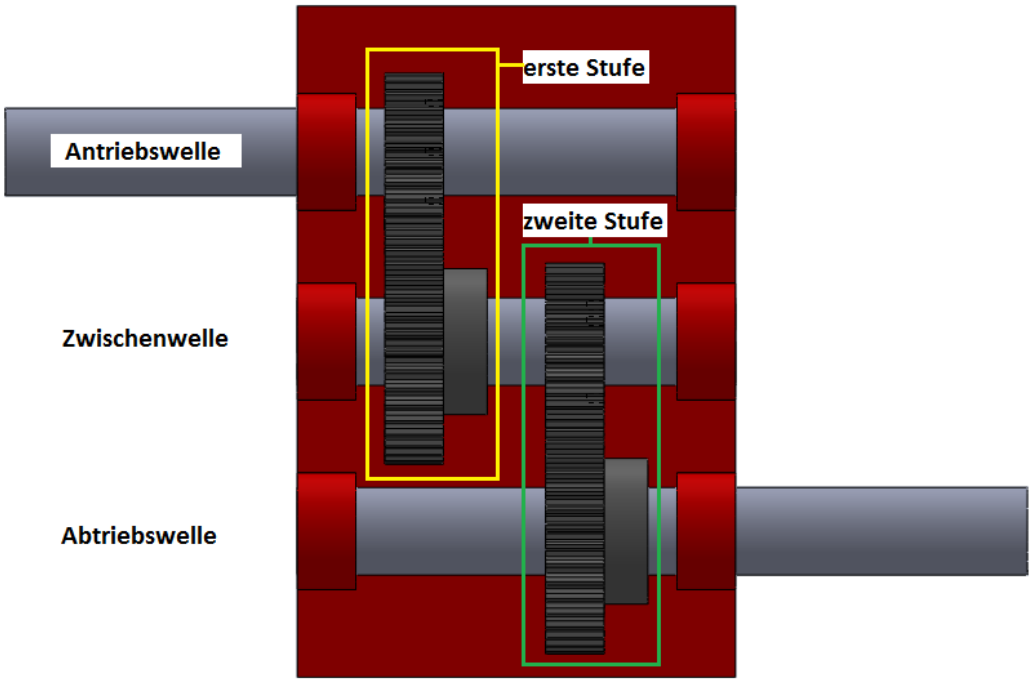

Abbildung 3: Zweistufiges Getriebemodell - Aufsicht (erstellt mit SolidWorks, 2013)

Umlaufrädergetriebe/Planetengetriebe: Die Kraftübertragung erfolgt mittels Zahn- oder Reibrädern. Ein einfaches Planetengetriebe (vgl. Abbildung 4 und Abbildung 6) besteht aus folgenden Bauteilen: einem Sonnenrad; einem oder mehreren Planeten; einem Planetenträger, der den oder die Planeten in einem konstanten Abstand um das Sonnenrad führt und einem Hohlrad. Planetengetriebe werden u.a. in Automatikgetrieben von Fahrzeugen eingesetzt. Neben den hier näher beschriebenen Formen sind noch weitere bekannt. In Abbildung 5 wird schematisch ein Planetengetriebe mit zwei Zentralrädern dargestellt.

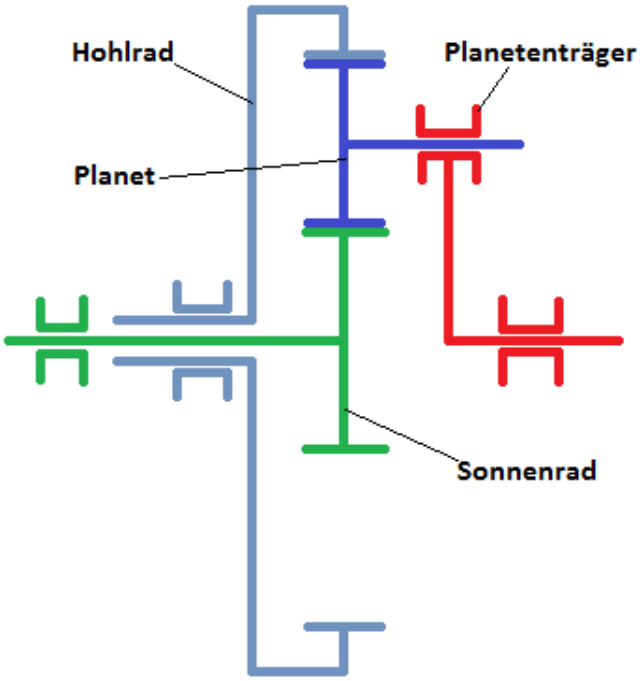

Abbildung 4: Schema eines einfachen Planetengetriebes

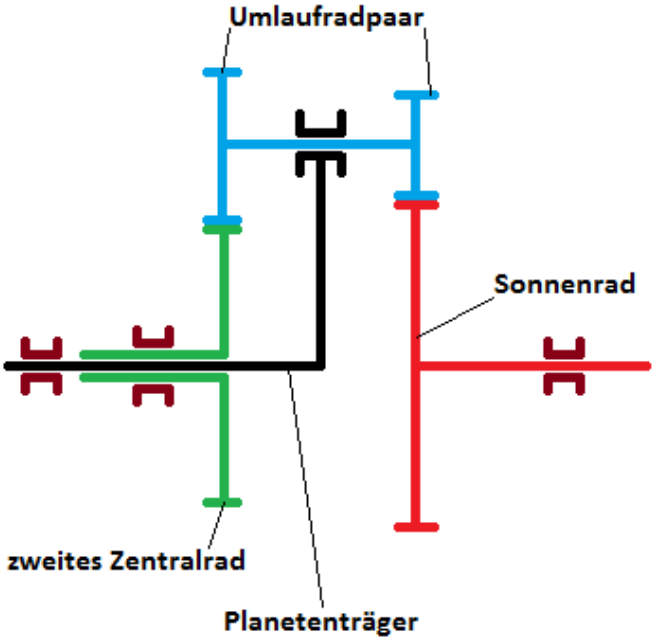

Abbildung 5: Schema eines Umlaufgetriebes mit zwei Zentralrädern

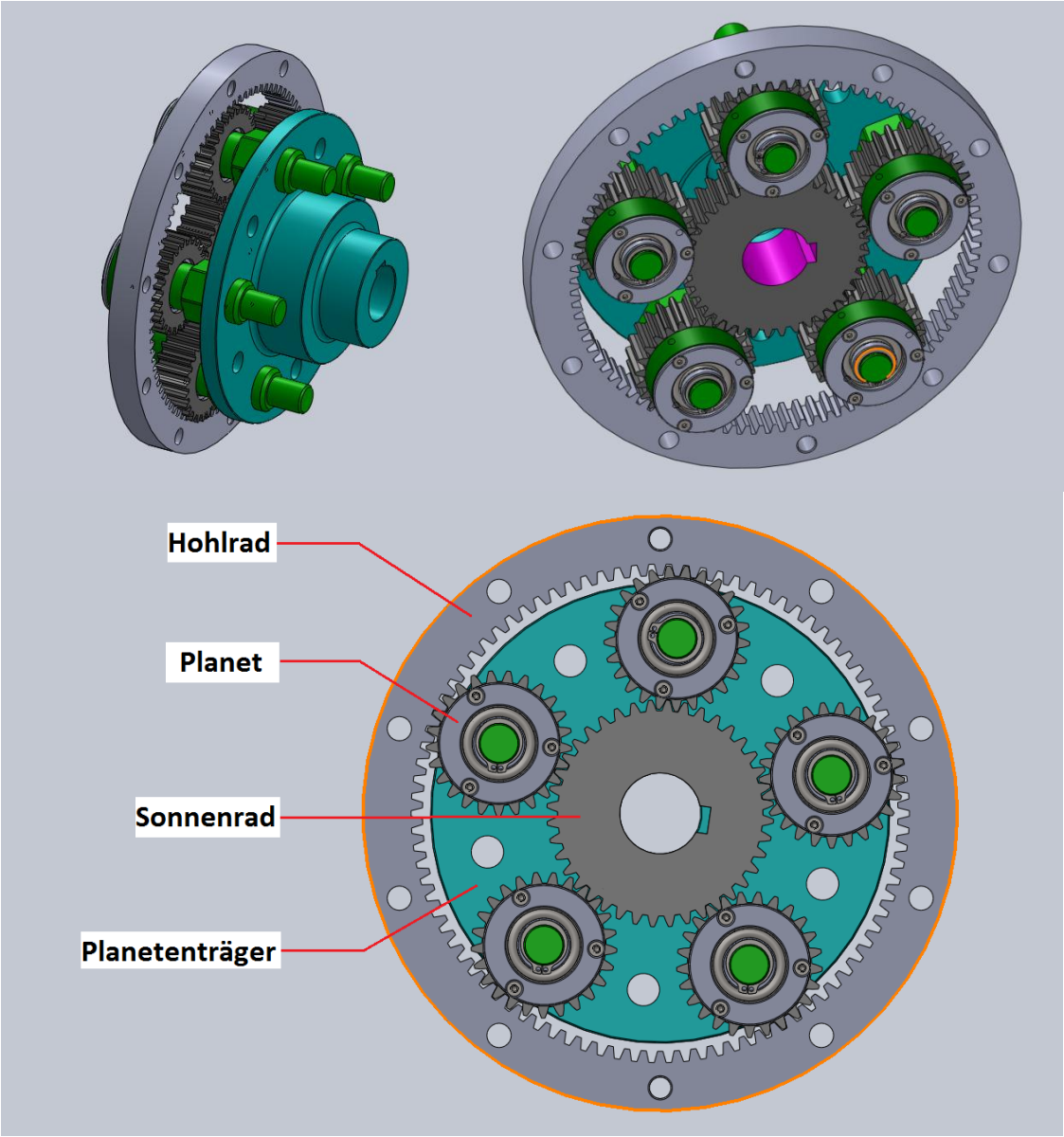

Abbildung 6: Planetengetriebestufe (erstellt mit SolidWorks, 2013)

Folgende sechs (Schalt-)Varianten werden hierbei unterschieden:

Zweiwellenbetrieb

1. Sonnenrad treibend (getrieben), Planetenträger getrieben (treibend), Hohlrad fix
2. Sonnenrad treibend (getrieben), Planetenträger fix, Hohlrad getrieben (treibend)
3. Sonnenrad fix, Planetenträger treibend (getrieben), Hohlrad getrieben (treibend)

Dreiwellenbetrieb

4. Sonnenrad treibend, Planetenträger treibend, Hohlrad getrieben
5. Sonnenrad treibend, Planetenträger getrieben, Hohlrad treibend
6. Sonnenrad getrieben, Planetenträger treibend, Hohlrad treibend

Je nach Variante ergeben sich bei sonst gleichen Parametern (Zahnanzahl bei Zahnrädern bzw. Außendurchmesser bei Reibrädern von Sonnenrad, Planeten und Hohlrad) unterschiedliche Übersetzungen.

Harmonic-Drive-Getriebe: Auch „Keilgleitgetriebe“ genannt, besteht aus zumindest folgenden drei Elementen (beschrieben von innen nach außen) [4]:

1. Im Zentrum des Getriebes ist eine elliptische Scheibe, auch „Wave-Generator“ genannt, angeordnet.
2. Um diese herum befindet sich eine verformbare Buchse mit Außenverzahnung, auch „Flexspline“ genannt. Zwischen Flexspline und Wave-Generator befinden sich Wälzkörper, auf denen der Flexspline am Wave-Generator abrollt.
3. Ein Hohlrad („Circular Spline“), wie es von Planetengetrieben bekannt ist, umschließt den Flexspline und steht mit diesem im Zahneingriff. Die Innenverzahnung des Hohlrades hat mehr Zähne als die Außenverzahnung des Flexspline.

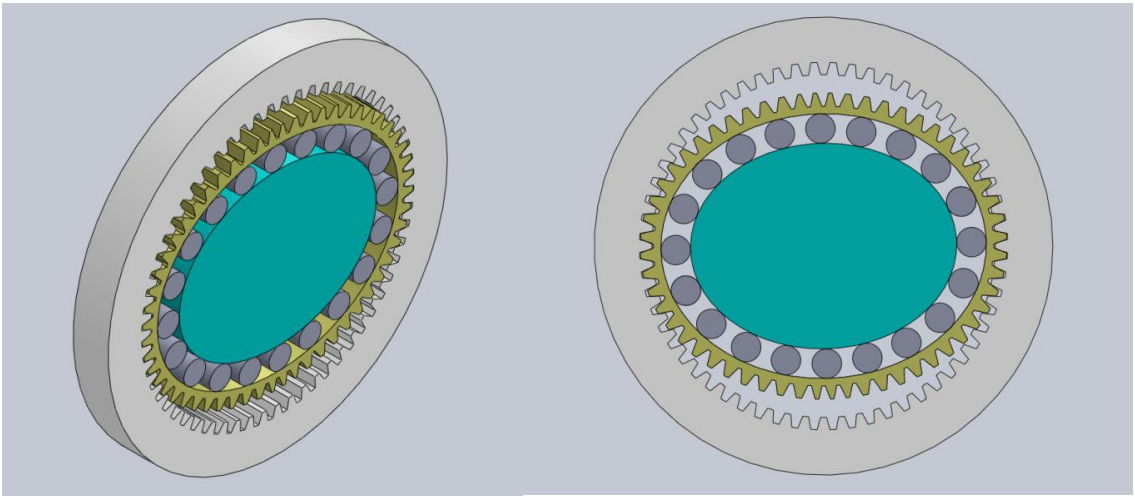

Abbildung 7: **Harmonic Drive (erstellt mit SolidWorks, 2013)**
Bezeichnung der Komponenten (von innen nach außen):
Wave Generator, Wälzkörper, Flex Spline und Circular Spline

Funktionsweise: Der Wave-Generator verformt den um ihn angeordneten Flexspline. Dieser greift über dessen Außenverzahnung im Bereich der Hauptachse der Ellipse in die Innenverzahnung des Circular Spline. Eine Animation des Funktionsprinzips sowie weitere Informationen sind unter [4] und [5] verfügbar.

Diese Getriebeform überzeugt durch ihr hohes Übersetzungsverhältnis (von 30:1 bis über 300:1) und geringen Wartungsaufwand. Der Wirkungsgrad beträgt bis zu 85 %. An- und Abtrieb erfolgt analog zum Planetengetriebe.

Schneckengetriebe: Besteht aus zumindest einer Schnecke (schraubenförmiges Zahnrad) und einem, in diese greifenden, schrägverzahnten Zahnrad, auch Schneckenrad genannt. Die Achsen der Schneckenwelle und die des Schneckenrades stehen normal zueinander. Das Übersetzungsverhältnis berechnet sich als Quotient aus der Zahnanzahl des Schneckenrades zur Gangzahl der Schnecke. Das Schneckengetriebe kann im GetriebeStudio durch zwei ineinandergrei-

fende Zahnräder näherungsweise dargestellt werden. Die Gangzahl der Schnecke entspricht hierbei der Zahnanzahl des Zahnrades [3].

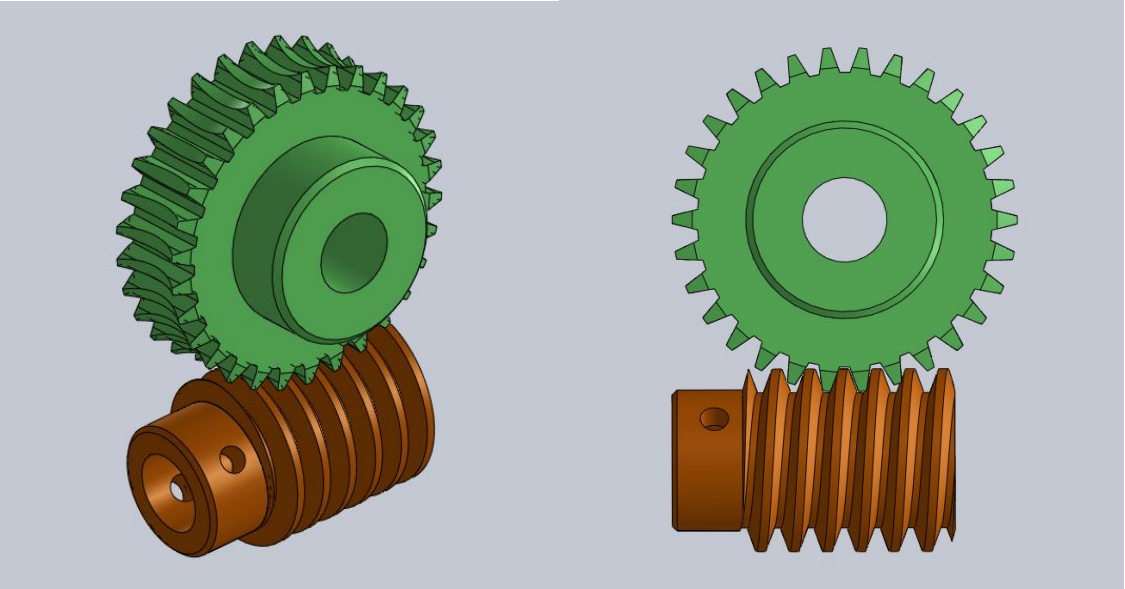

Abbildung 8: Schneckenrad (oben) und Schnecke (unten) im Eingriff (SolidWorks, 2013)

Weitere Getriebeformen [2]**:**

Verteilergetriebe: Die Getriebeform ähnelt einem Stufengetriebe mit einer Antriebswelle und mehreren Abtriebswellen.

Extrudergetriebe: Extruder (für Spritzgussmaschinen) und Getriebe bilden eine Einheit.

Kassettengetriebe: Sie bestehen aus zwei Hauptwellen und sind schaltbare Getriebe. Jede Hauptwelle trägt der Anzahl der zu schaltenden Gänge entsprechend viele Zahnräder. Durch den konstruktiv bedingten geringen Montageaufwand, werden Kassettengetriebe vor allem im Motorsport eingesetzt.

Kegelradgetriebe: Im Vergleich zu einem Stirnradgetriebe sind die Zahnräder eines Kegelradgetriebes nicht zylindrisch sondern kegelig ausgeführt (vgl. Abbildung 9). Die im Eingriff befindlichen Zahnradachsen stehen in einem bestimmten Winkel zueinander. Die Achsen der Zahnräder können einander schneiden oder sind windschief ausgeführt (Hypoidgetriebe). Ein Kegelradgetriebe wird im GetriebeStudio durch zwei ineinandergreifende Stirnzahnräder näherungsweise dargestellt.

Abbildung 9: Kegelradpaar mit Schrägverzahnung [6]

Kegelringgetriebe: Ein Kegelringgetriebe besteht aus mindestens zwei Wellen mit kegeligem Bereich sowie einem Ring, durch den ein Kraftschluss zwischen den beiden kegelig ausgeführten Bereichen der Wellen zustande kommt. Das Übersetzungsverhältnis des Kegelradringgetriebes kann durch Verschieben des Ringes variiert werden.

Koppelgetriebe: Sie dienen der Wandlung von rein rotatorischer Bewegungen in einen sich periodisch wiederholenden Bewegungsablauf mit translatorischen und/oder rotatorischen Bewegungsanteilen und umgekehrt. In Abbildung 10 ist der Aufbau eines einfachen mit vier Gelenken ausgestatteten Koppelgetriebes dargestellt. Es besteht aus einem Gestell und zwei mit diesem gelenkig verbunden Bauteilen. Die beiden Teile werden über eine sogenannte Koppel gelenkig miteinander verbunden. Koppelgetriebe werden unter anderem bei der Erdölförderung eingesetzt.

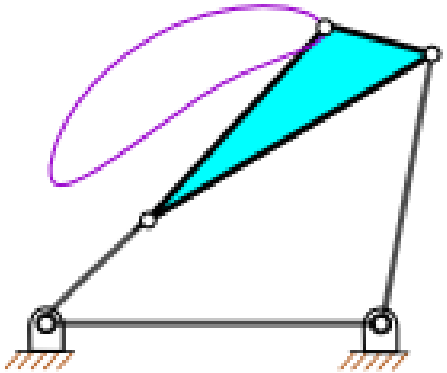

Abbildung 10: Viergelenk-Koppelgetriebe [7]

Kurvengetriebe: Sie dienen der Verwirklichung von vorgeschriebenen Bewegungsabläufen mittels vorgegebener Steuerkurve [8]. Diese werden unter anderem im Automobilbau zur Steuerung des Ventiltriebes bei der Nockenwelle (Steuerkurve) eingesetzt.

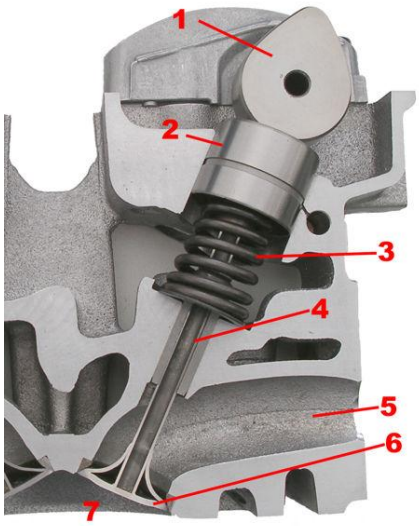

Abbildung 11: **Ventilantrieb [9] mit Nocke (1), Stößel (2), Ventilfeder (3), Ventilführung (4), Kanal(5), Ventil (6) und Brennraum (7)**

Flächendruckgetriebe (Kreisschubgetriebe): Diese Getriebeform zeichnet sich durch die charakteristische Bewegung eines ihrer Bauteile aus. Es führt eine sogenannte Kreisschubbewegung aus, bei der jeder Punkt dieses Bauteils sich auf einer kreisförmigen Bahn bewegt. Abbildung 12 zeigt exemplarisch ein Flächendruckgetriebe mit seinen wichtigsten Komponenten. Der Bewegungsablauf ist in Abbildung 13 dargestellt. Flächendruckgetriebe werden u.a. heute in der Textil-, Pharma- und Lebensmittelindustrie eingesetzt [10].

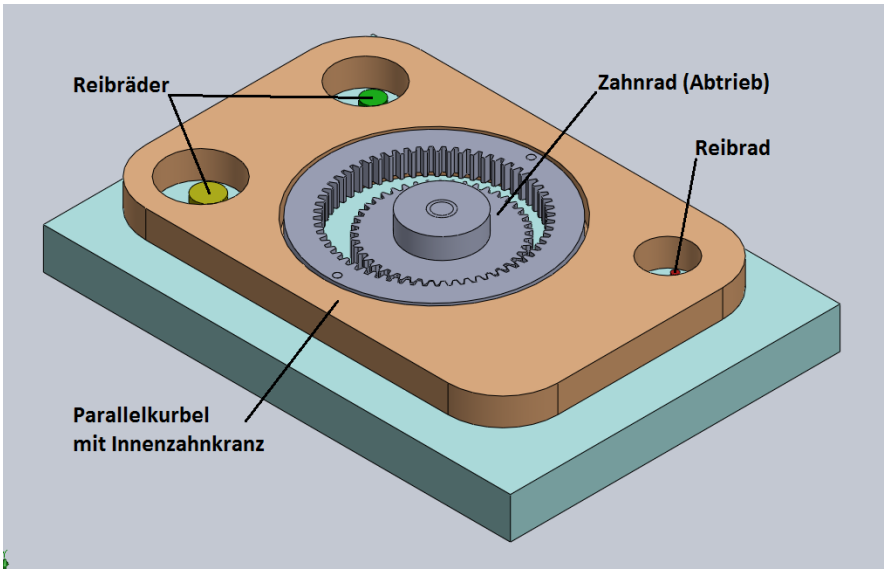

Abbildung 12: Flächendruckgetriebe (erstellt mit SolidWorks, 2014)

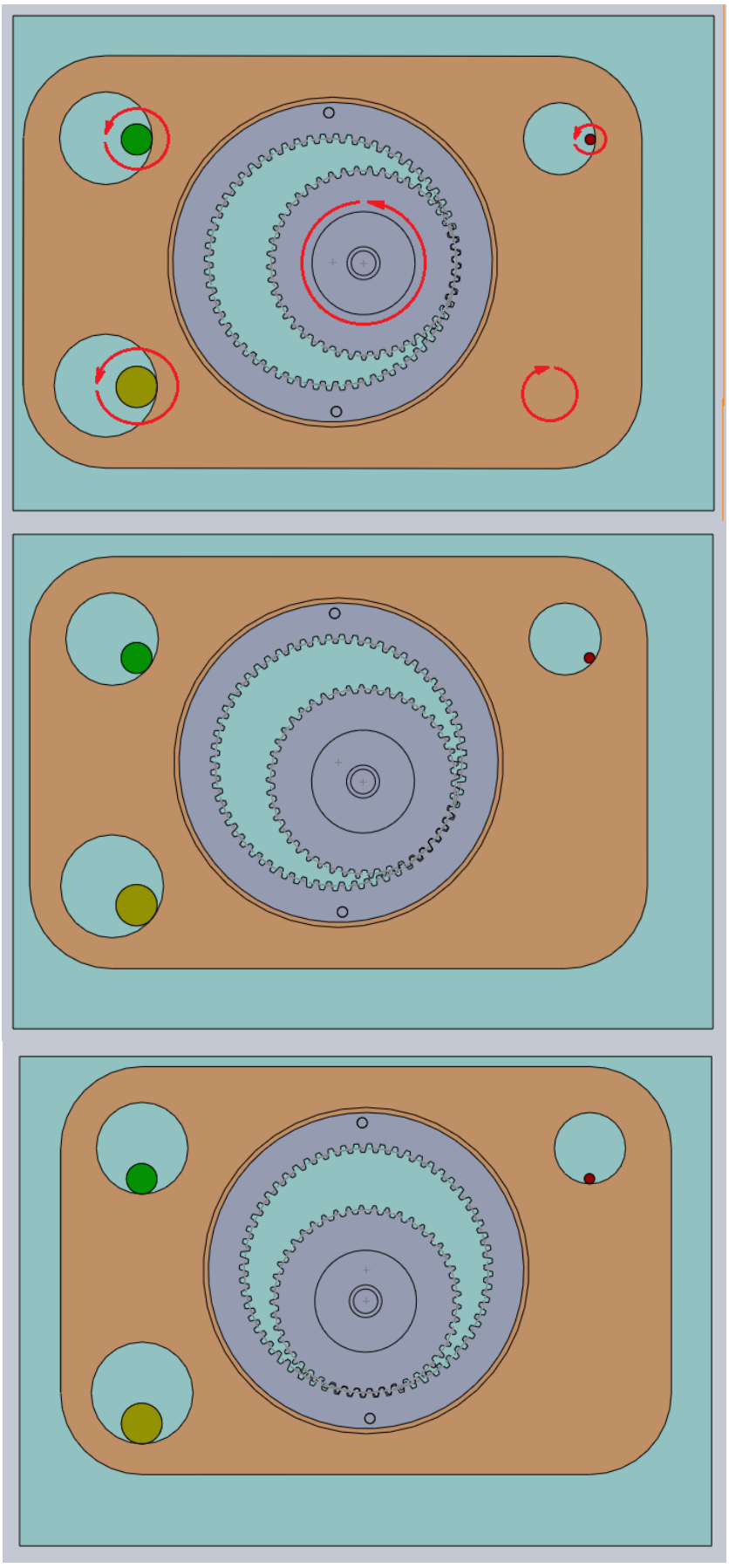

Abbildung 13: Flächendruckgetriebe – Bewegungsablauf (erstellt mit SolidWorks, 2014)

Malteserkreuzgetriebe: Ein spezielles Kurvengetriebe, bei dem eine kontinuierliche, rotatorische Bewegung in eine intermittierende (lat. Intermittere = unterbrechen/aussetzen), rotatorische Bewegung transformiert wird. Ihren Namen verdankt dieses Getriebe der Form des intermittierend drehenden Bauteils (siehe Abbildung 14), das einem Malteserkreuz ähnelt. Es sind Ausführungen mit vier oder mehreren Schlitzen möglich. In Abbildung 15 ist der Bewegungsablauf eines Maltesergetriebes zu erkennen. Zu den technischen Anwendungen zählen Filmprojektoren und Uhren.

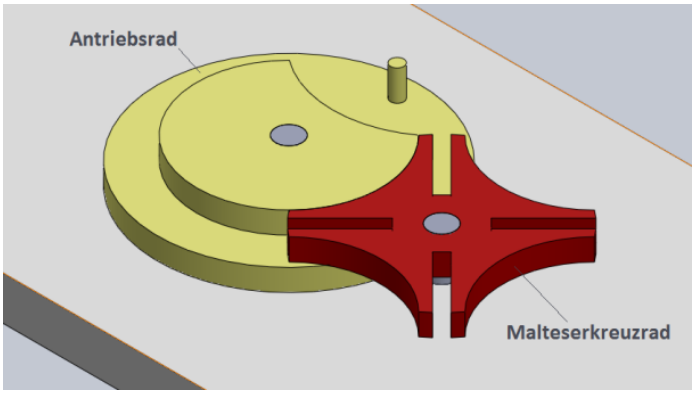

Abbildung 14: Malteserkreuzgetriebe (erstellt mit SolidWorks, 2014)

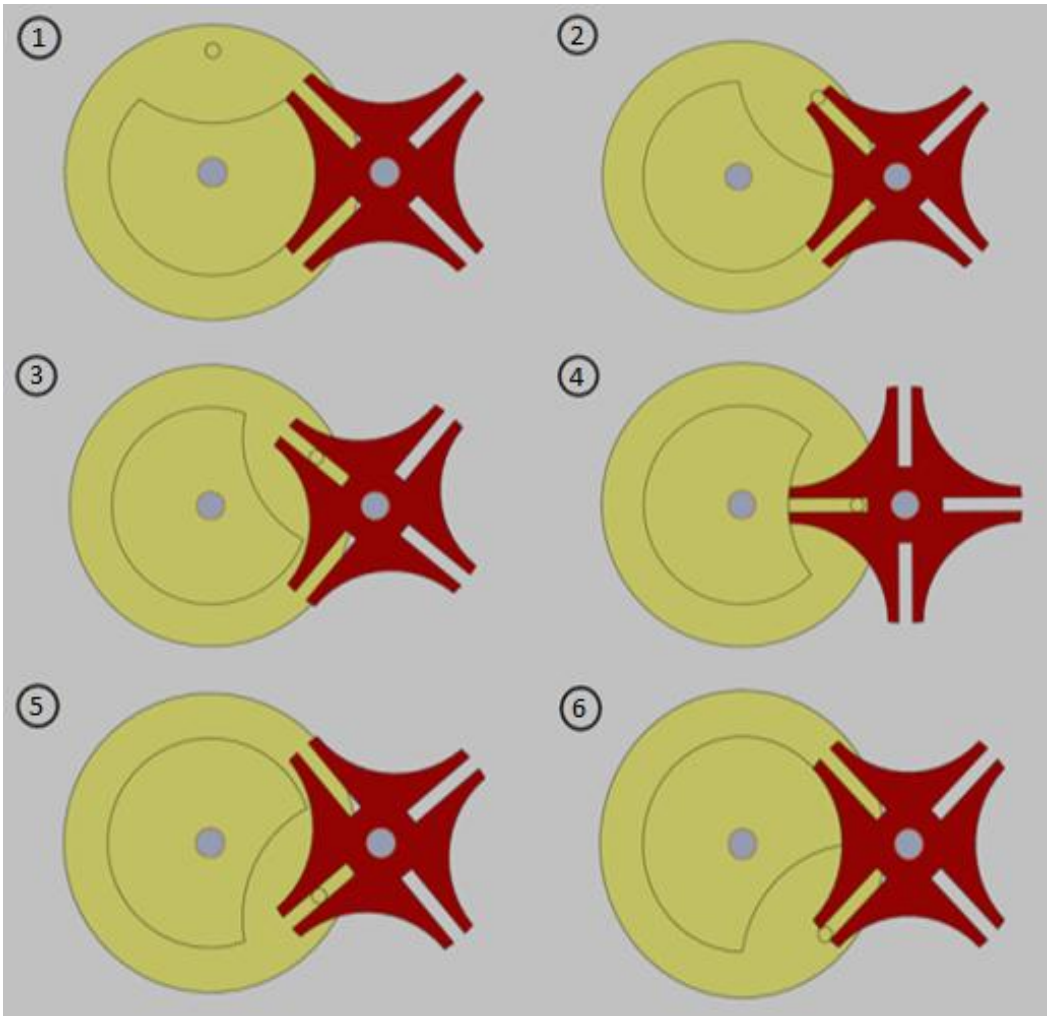

Abbildung 15: Maltesergetriebe – Ablauf (erstellt mit SolidWorks, 2014)

Schieberadgetriebe: Die Übersetzung kann durch verschieben der Zahnräder verändert werden. Der Zahneingriff eines Zahnradpaares wird hierbei durch das Einschieben eines der beiden Zahnräder hergestellt und durch das Ausrücken wieder gelöst.

Schraubenradgetriebe: Besteht aus einer Spindel, einer Spindelmutter und einem Gestell. Durch das Verdrehen der Spindel verschiebt sich die Spindelmutter relativ zum Gestell. Schraubgetriebe werden unter anderem für Spindelpressen, Schraubstöcke und Wagenheber eingesetzt.

6.2.2 Hydraulisches und pneumatisches Prinzip

Zum Übertragen der Energie und/oder Bewegung wird neben mechanischen Bauteilen ein Übertragungsmedium in Form einer Flüssigkeit (Hydraulik) oder Gas (Pneumatik) eingesetzt [2]. Im Allgemeinen werden folgende Formen unterschieden:

Hydrostatische Getriebe: Der für die Übertragung der Bewegung bzw. Kräfte nötige Druck wird statisch aufgebracht. Als Beispiele hierfür seien die hydraulische Bremse eines Fahrzeuges, die Kolbenpumpe und Hydraulikzylinder zu nennen.

Hydrodynamische Getriebe: Im Gegensatz zu den hydrostatischen Getrieben wird der nötige Druck zur Übertragung der Bewegung bzw. Kräfte und Momente dynamisch erzeugt. Strömungskupplung und Drehmomentwandler sind bekannte Beispiele hydrodynamischer Getriebe.

Pneumatisches Getriebe: Bekannte Beispiele hierfür sind Pneumatikzylinder, Bremskraftverstärker und Kompressoren.

6.2.3 Elektromagnetisches Prinzip

Mechanische Energie oder Bewegung wird in elektrische Energie gewandelt und an anderer Stelle wieder in mechanische Energie oder Bewegung umgesetzt. Als Beispiel soll hier die elektrische Welle genannt werden [2].

Elektrische Welle: Ist die Nachbildung einer mechanischen Welle deren Aufgabe darin besteht Drehmoment und/oder Bewegung von einer Seite der Welle auf die andere zu übertragen. Der Vorteil einer elektrischen Welle besteht darin, dass die beiden Enden der Welle nicht starr miteinander verkoppelt sind.

6.3 Einschränkung der Diplomarbeit

Im Rahmen der Diplomarbeit war aus Zeit- und Ressourcengründen eine Beschränkung auf Stufen- und Planetengetriebe, sowie deren Mischformen erforderlich. Ein weiterer Ausbau ist jedoch denkbar. Im Folgenden sollen kurz die für die darzustellenden Getriebemodelle benötigten Komponenten aufgezeigt werden. Die Funktion der einzelnen Bauteile sei als bekannt vorausgesetzt, weshalb in dieser Arbeit nicht näher darauf eingegangen wird.

6.4 Getriebekomponenten

6.4.1 Stufengetriebe

Zu den wichtigsten Bauteilen eines Stufengetriebes zählen Wellen, Lager und Komponenten zum Übertragen des Drehmomentes von einer Welle auf die nächste Welle einer Stufe. Hierzu zählen u.a. Zahnräder und Kupplungen. Abbildung 16 zeigt beispielhaft ein am Institut für Mechanik und Mechatronik entwickeltes zweistufiges Versuchsgetriebe mit seinen wichtigsten Komponenten.

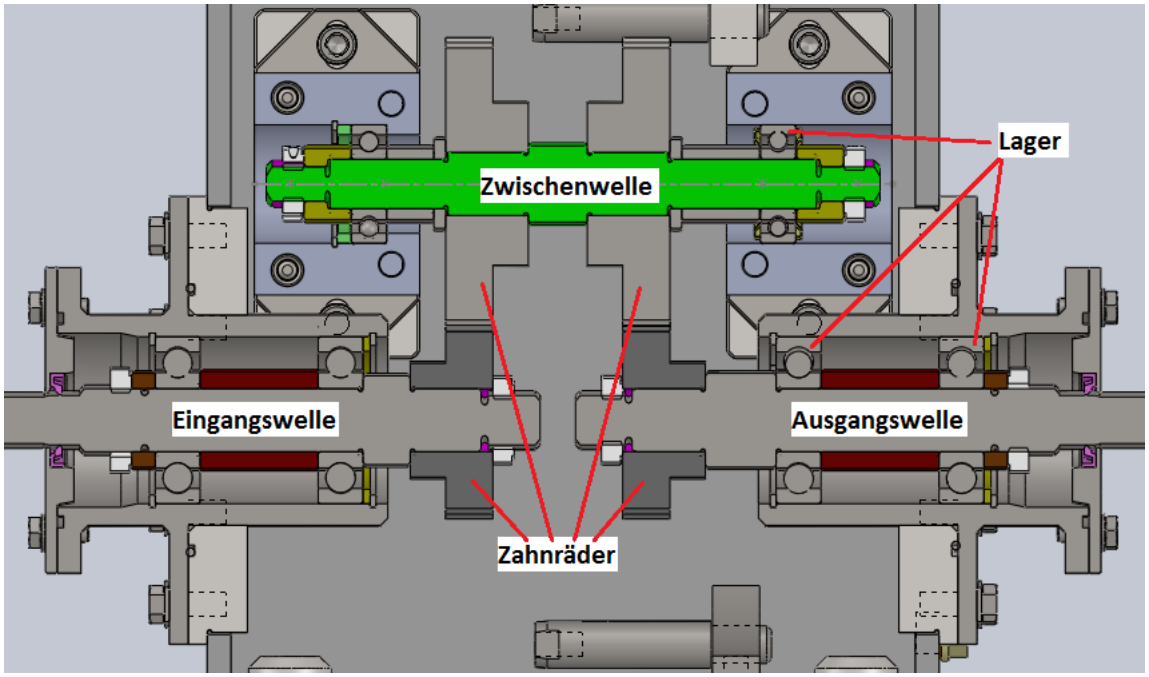

Abbildung 16: Zweistufiges Getriebe für Versuchsbetrieb (Gesamtübersetzung 1:1, erstellt mit SolidWorks, 2013)

6.4.2 Planetengetriebe

Zu den wichtigsten Komponenten eines einfachen, wie in Abbildung 4 schematisch dargestellten, Planetengetriebes zählen das Sonnenrad und die Planeten (außenverzahnte Zahnräder oder Reibräder), das Hohlrad (innenverzahnter Zahnkranz oder Reibring), der Planetenträger, Wellen und Lager. Abbildung 17 zeigt ein einstufiges Planetengetriebe mit seinen wichtigsten Komponenten. Für die normgerechte vereinfachte 2D-Darstellung von Lagern und Zahnrädern sei auf [3], [11] und [12] verwiesen. In Abbildung 18 werden wichtige, mit „GetriebeStudio“ dargestellte Getriebekomponenten angezeigt.

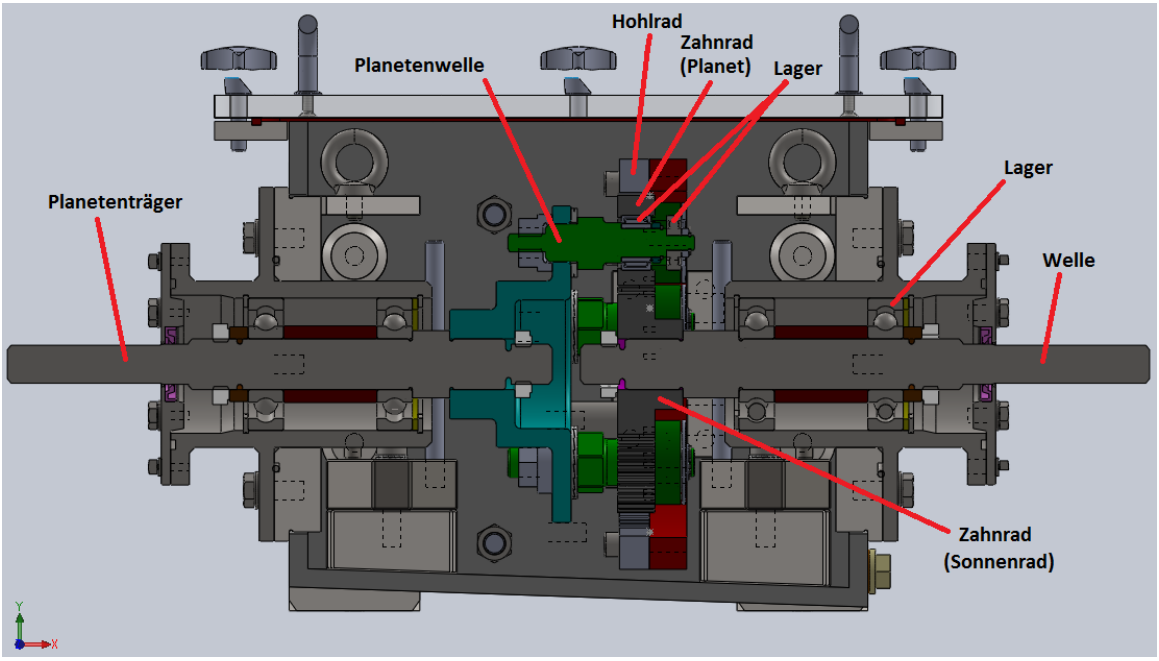

Abbildung 17: Komponenten eines einstufigen Planetengetriebes mit fixem Hohlrad (SolidWorks, Schnittansicht, 2013)

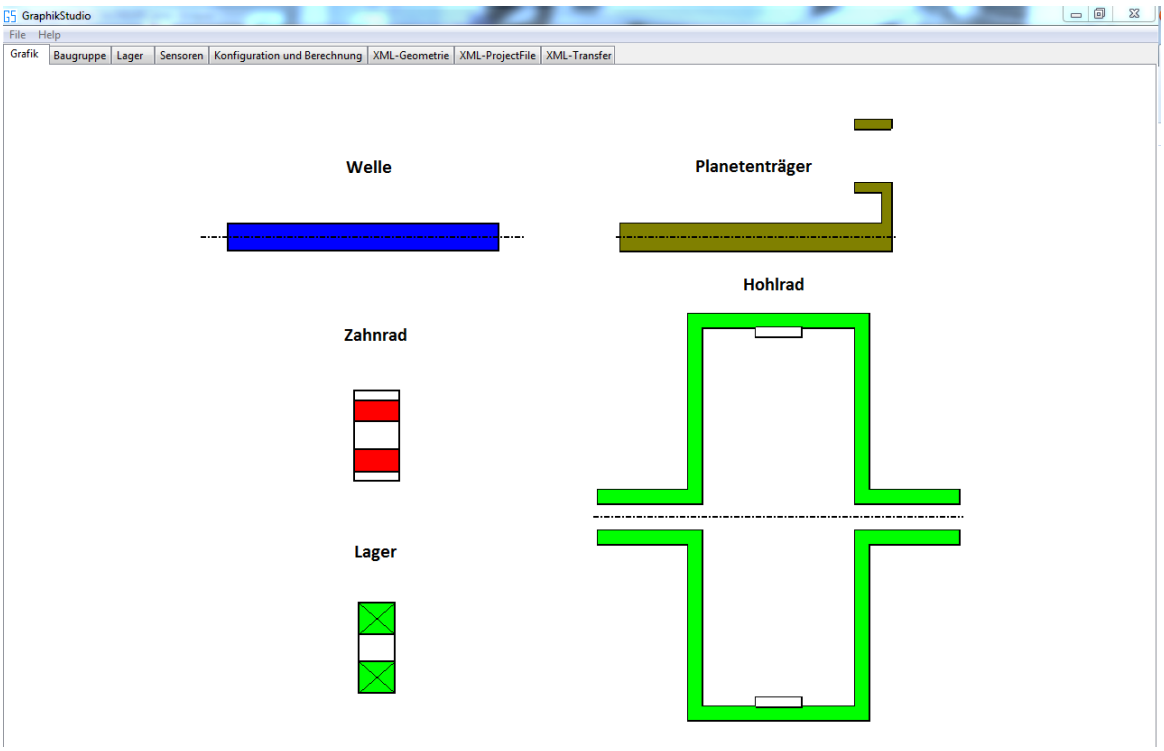

Abbildung 18: Wichtige, mit "GetriebeStudio" dargestellte Getriebekomponenten

7 Architektur

Das Programm ist Teil eines Systems, das auf einer Fat-Client-Architektur (siehe [13]) aufbaut. Der Austausch zwischen den einzelnen Programmen erfolgt hierbei über XML-Dokumente. Das ausgetestete Programm wird in eine ausführbare Datei (executable Jar) exportiert. Alle erforderlichen Ordner, Dokumente und sonstige wichtige Objekte sind in einem Ordner des Systems integriert. Mit Hilfe einer Settings-Datei (Text-file) können wichtige Parameter wie etwa Speicherpfade verändert werden. Für die Verwendung wird die Installation eines passenden Java Runtime Environment (JRE) - Softwarepaketes vorausgesetzt.

8 Schnittstellen

Die Daten werden mittels XML-Dokumente zwischen den einzelnen Teilen des Systems ausgetauscht. Die zulässige Form der Dokumente wurde in Form von XML-Schemas (XSD, Grundlagen siehe hierzu [14]) festgehalten.

9 Design

Im Folgenden soll das Design des Systems beschrieben werden. Hierfür wurden die wichtigsten Anwendungsfälle und Abläufe in geeigneter Form (UML + Prosa) festgehalten. Die entwickelten UML-Diagramme stellen wichtige Dokumente für die spätere Implementierung des Codes, der Wartung des Programmes und der Dokumentation für Entwickler und Wartungspersonal dar.

Als Beschreibungssprache wurde **UML** (Unified Modelling Language, Grundlagen siehe [15]) verwendet. Diese wird schon seit den 1990er Jahren in der Softwareentwicklung zur Beschreibung von Softwaresystemen eingesetzt. Es handelt sich hierbei um eine graphische Modellierungssprache mit der einerseits die Struktur und andererseits das gewünschte Verhalten des Systems dargestellt werden kann. Insgesamt stehen 14 verschiedene Diagramme zur Verfügung. Für die Beschreibung des hier zu entwickelnden Softwaresystems wurden ein UML-Anwendungsfalldiagramm, eine textuelle Beschreibung eines jeden Anwendungsfalles sowie mehrere UML-Aktivitätsdiagramme erstellt.

Zum Erstellen der Diagramme diente das freie, open-source UML-Werkzeug UMLet. UMLet bietet für jede UML-Diagrammart einen Satz von vordefinierten Bausteinen, welche mittels Drag&Drop zu einem Diagramm zusammengefügt werden können. UMLet ist unter [16] frei verfügbar.

Nachfolgend sollen die vom System abzudeckenden Anwendungsfälle und Aktivitäten vorgestellt werden.

9.1 Anwendungsfälle

Für das Verständnis des nachfolgenden Diagrammes soll zunächst geklärt werden, was man im Allgemeinen unter einem Anwendungsfall versteht. Hierzu folgende Definitionen:

„Ein Anwendungsfall beschreibt die Funktionalität eines Systems aus Sicht des Benutzers. Ein Anwendungsfall wird stets durch einen Akteur initiiert und führt gewöhnlich zu einem für die Akteure wahrnehmbaren Ergebnis“, ([15]).

„Ein Anwendungsfall (use case) beschreibt eine (durch einen Akteur angestoßene) Reihe von erkennbaren Aktionen, die ein System ausführt und die zu einem erkennbaren Ergebnis mit Wert für den Handelnden (Akteur) führt“, ([17]).

Die Anwendungsfälle werden hierbei, wie es der UML-Standard vorschreibt, als Ellipsen dagestellt (vgl. Abbildung 19).

Die Benennung eines Anwendungsfalles erfolgt meist nach dem, vom Akteur, verfolgten Ziel. Anwendungsfälle werden innerhalb, Akteure wie etwa Nutzer und andere Systeme werden außerhalb der Systemgrenzen angezeigt. Anwendungsfälle können einander erweitern oder auch inkludieren. Dies kann, wie in Abbildung 19, durch einen strichlierten Pfeil mit zugehörigem Text dargestellt werden. Auf die Darstellung und Beschreibung sonstiger, das Anwendungsfalldiagramm betreffender, jedoch für die Arbeit irrelevanter Konzepte wird hier verzichtet.

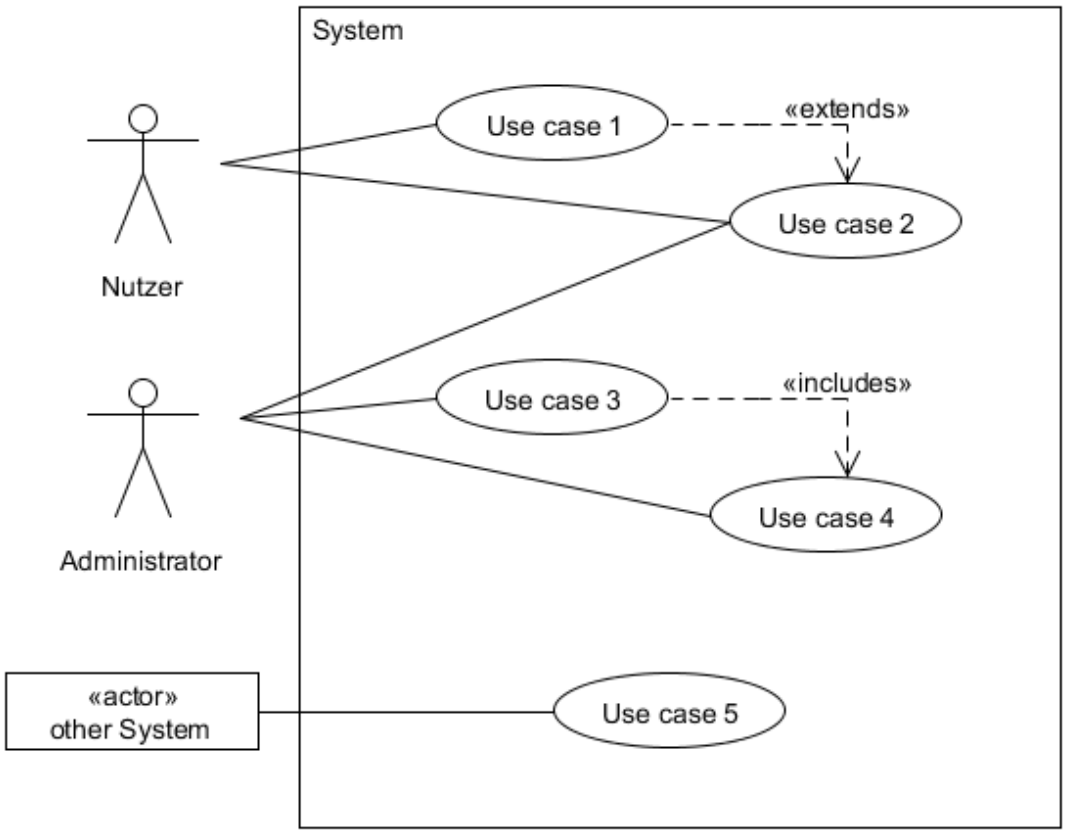

Abbildung 19: Anwendungsfalldiagramm (erstellt mit UMLet am 28.07.2014)

Das folgende Diagramm (Abbildung 20) bietet einen Überblick über die vom System „GetriebeStudio“ abzudeckenden Anwendungsfälle. Zur Modellierung wurde die UML-Syntax verwendet. Die genaue Beschreibung eines jeden Anwendungsfalles erfolgt aus Gründen der Übersicht und des Platzes separat (siehe Kapitel 9.1.1).

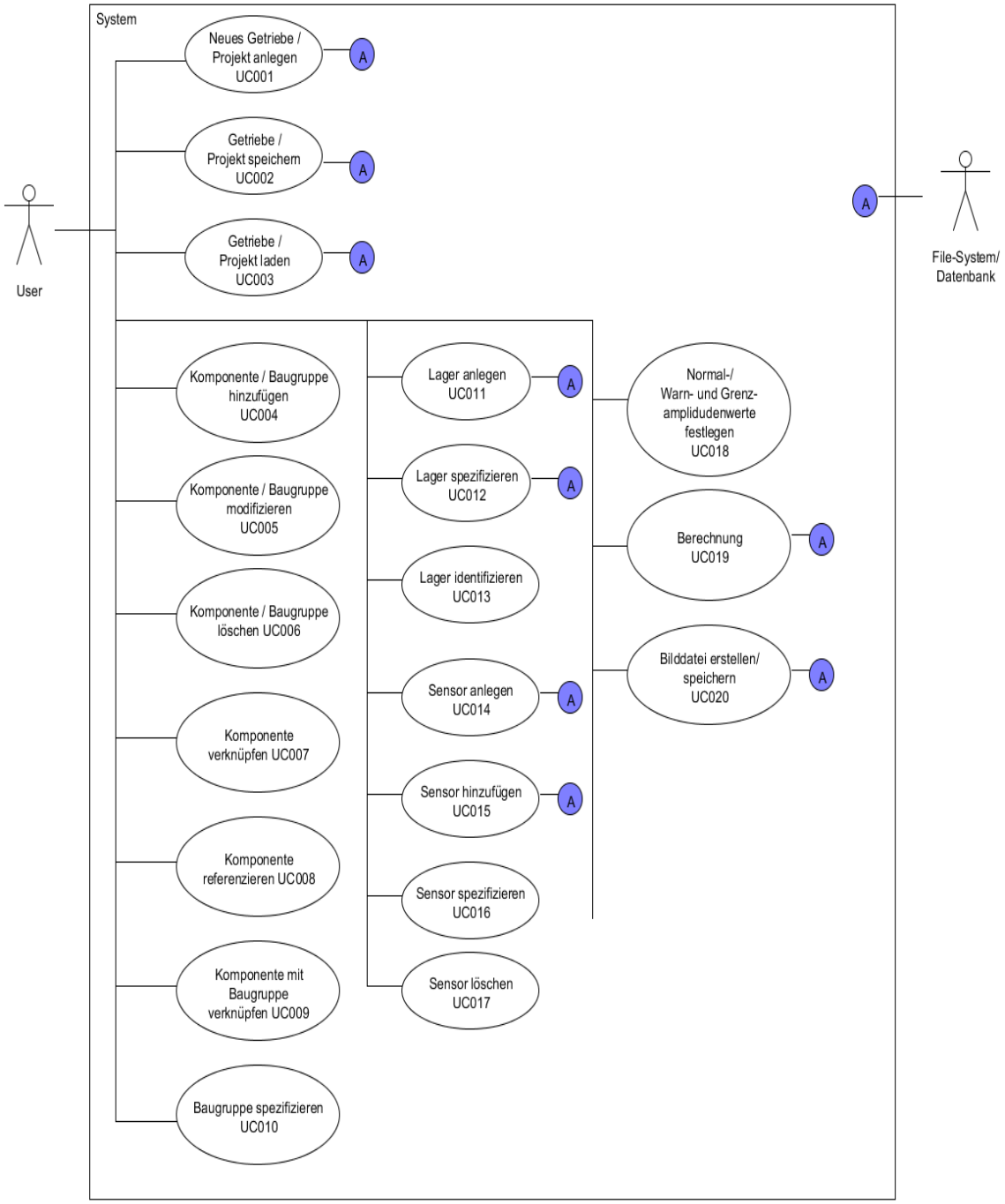

Abbildung 20: "GetriebeStudio"-Anwendungsfalldiagramm (Stand 25.4.2013)

9.1.1 Beschreibung der Anwendungsfälle

UC001 - Neues Getriebe/Projekt anlegen

Ein Getriebe wird mit all den spezifizierten Daten in einem Projekt angelegt. Zu den Projektdaten zählen u.a. die Daten der geometrischen Figuren wie zum Beispiel Abmessungen von Wälzlagern oder Zahnrädern sowie die Spezifikationen von Baugruppen, Sensoren und Lagern.

UC002 – Getriebe/Projekt speichern

Das modellierte und spezifizierte Getriebe soll mit all seinen Daten in geeigneter Form gespeichert werden (Filesystem). Als Speicher- und Austauschformat dient XML. Dies hat unter anderem den Vorteil, dass die Daten für Mensch und Maschine (u.a. Browser und Editoren) gleichermaßen lesbar sind. Für eine spätere kommerzielle Nutzung wäre es sinnvoll die Daten persistent und verschlüsselt in einer Datenbank zu verwalten (dies ist aber nicht Teil der Diplomarbeit).

Die Daten werden in geometrie-, projekt- und transferdatenbezogene Informationen unterteilt, in getrennten XML-Dateien gespeichert und in einem gemeinsamen (Projekt-)Ordner hinterlegt. Die Transferdaten dienen dem Austausch zwischen dem System GetriebeStudio und dem Zustandsüberwachungssystem SPECTIVE.

UC003 – Getriebe/Projekt laden

Die gespeicherten Projekte (XML) sollen für eine spätere Weiterbearbeitung wieder geladen werden können.

UC004 – Komponente/Baugruppe hinzufügen

Ein Getriebe besteht aus Komponenten und Baugruppen. Zu den Komponenten zählen Lager, Planetenlager, Zahnräder, Hohlräder, Planetenträger und Wellen. Zu den Baugruppen zählen Planetengetriebe, Differenziale etc. wobei in der ersten Version nur Planetengetriebe umgesetzt werden. Wichtig ist, dass beim Design und der späteren Implementierung auf die einfache Erweiterbarkeit des Programmes geachtet wird.

UC005 – Komponente/Baugruppe modifizieren

Die dargestellten und per Mausklick auswählbaren Komponenten und Baugruppen haben Eigenschaften, die über eine Eigenschaftsliste modifiziert werden können. Die Objekteigenschaften werden nach der Modifikation aktualisiert.

UC006 Komponente/Baugruppe löschen

Nicht benötigte bzw. versehentlich angelegte Baugruppen und Komponenten sollen aus dem Projekt entfernbar sein.

UC007 – Komponenten verknüpfen

Die einzelnen Komponenten lassen sich unter Einhalten bestimmter Regeln miteinander verknüpfen. Beispiel: Eine Welle kann mit einer beliebigen Anzahl von Lagern und Zahnrädern verknüpft sein.

UC008 – Komponente referenzieren

Für die Berechnung ist es erforderlich, bestimmte Eingangsgrößen wie etwa die Drehzahl der Referenzwelle festzulegen. Weitere Eingangsgrößen sind die Anzahl der Messpunkte pro Umdrehung und die Abtastfrequenz.

UC009 – Komponenten mit Baugruppe verknüpfen

Eine Baugruppe besteht aus verschiedenen Komponenten. Beispiel: Ein Planetengetriebe enthält normalerweise ein Sonnenrad, Planetenräder, ein Hohlrad, einen Planetenträger, Lager, Planetenlager und Wellen. Für eine spätere Spezifizierung der Teile der Baugruppe müssen diese zuvor mit der Baugruppe verknüpft sein. Beispiel für eine Spezifizierung: Ein bestimmtes Zahnrad ist ein Sonnenrad, ein anderes ein Planet, das Hohlrad sei fix, usw.

UC010 – Baugruppe spezifizieren

Die in der Baugruppe enthaltenen Komponenten sollen baugruppenabhängig (z.B. durch Angabe der Art der Baugruppe) spezifiziert werden.

Beispiel: Ein Planetengetriebe enthält mehrere Wellen. Diese können getrieben, treibend oder fixiert (nicht drehbar) sein.

UC011 – Lager anlegen

Die modellierten Lager müssen, damit sie in der Berechnung berücksichtigt werden können, mit einem Lager der Lagerdatenbank verknüpft sein. Dazu ist es erforderlich, die Lager in der Lagerdatenbank mit ihren Schadensfrequenzen (Außenring, Innenring, Wälzkörper und Käfig) bei Bedarf anlegen zu können.

Alternativ ist die Modifikation des auf XML basierten Datendokumentes über jeden Editor möglich. Ein bereits angelegter Datensatz soll natürlich nicht zweimal angelegt werden können.

UC012 – Lager spezifizieren

Die modellierten Lager müssen, damit sie in der Berechnung berücksichtigt werden können, mit einem Lager der Lagerdatenbank verknüpft sein. Die modellierten Lager sollen über eine Tabelle angezeigt und z.B. über einen Button modifizierbar sein. Die Daten real existierender Lager, etwa Bezeichnung und normierte Schadensfrequenzen, sollen in geeigneter Form abgespeichert werden. Ergänzung nach Absprache mit den Key-Usern (siehe auch Kapitel 12): Die Lagersuche soll mit Hilfe einer Suchmaschine einfach und bequem durchführbar sein. Die gefundenen Lagerdaten werden etwa mittels Button oder Doppelklick auf das Suchergebnis der zuvor gewählten und modellierten Lager zugewiesen.

UC013 – Lager identifizieren

Zur leichteren Spezifikation soll es eine Möglichkeit geben, die zu spezifizierenden Lager zu identifizieren z.B. per Button in der Spezifikation und/oder durch Selektion in der graphischen Darstellung.

UC014 – Sensor anlegen

Neben den Lagern sind auch die für das spätere Messen der Schwingungen (Zustandsüberwachung) benötigten Sensoren anzulegen. Hierzu sind die Sensorart (Beschleunigungssensor, Drehzahlsensor, etc.), die Bezeichnung, und falls notwendig die Empfindlichkeit anzugeben.

Ein bereits angelegter Datensatz soll natürlich nicht zweimal angelegt werden können.

UC015 – Sensor hinzufügen

Die benötigten Sensoren sollen aus den zuvor angelegten Sensoren ausgewählt und der Liste der projektspezifischen Sensoren hinzugefügt werden. Der Inhalt dieser Liste soll in einer Tabelle angezeigt werden.

UC016 – Sensor spezifizieren

Die hinzugefügten Sensoren sollen je nach Bedarf weiter spezifizierbar sein. Im Falle eines Drehzahlsensors soll zusätzlich die Möglichkeit bestehen, das Attribut „bezogen“ auf *true* oder *false* zu setzen. Wobei *true* bedeutet, dass das gemessene Sensorsignal bzw. die sich daraus ergebende Wellendrehzahl als Referenz für die Analyse durch SPECTIVE herangezogen wird.

UC017 – Sensor löschen

Die spezifizierten Sensoren sollen, falls sie doch nicht benötigt werden, aus der Sensorspezifikation löschbar sein.

UC018 – Normal-, Warn- und Grenzamplitudenwerte festlegen (Task revidiert[4])

Für die Zustandsüberwachung sind neben der Erfassung der Messdaten auch die Angabe von Warn- und Grenzwerten der Amplituden von erheblicher Bedeutung. Dadurch wird ein automatischer Vergleich der gemessenen Werte mit den Warn- und Grenzwerten erst möglich.

Der Warnwert bezieht sich auf jene Amplitude, bei der, wenn die zugeordnete Amplitude diesen Wert erreicht oder überschreitet, eine Warnung ausgegeben wird.

Der Grenzwert bezieht sich auf jene Amplitude, ab der ein Schaden mit großer Wahrscheinlichkeit eingetreten ist.

Für jedes Lager bzw. für deren Komponenten (Innenring, Außenring, Käfig und Wälzkörper) sind die Warn- und Grenzwerte für die ersten fünf Harmonischen und die drei oberen und unteren Seitenbänder der ersten Harmonischen anzugeben.

Für Zahneingriffe sind die Warn- und Grenzwerte für die ersten drei Harmonischen und die zu dem jeweiligen Zahnrad gehörigen, drei oberen und unteren Seitenbänder anzugeben.

UC019 – Berechnung

Vorbereitungsmaßnahmen:

- Getriebe modelliert,
- gewünschte Bauteile referenziert,
- Baugruppen, Lager und Sensoren spezifiziert,
- Grenzamplitudenwerte bestimmt,
- Anlagennummer angegeben und
- Parameter (Referenzdrehzahl, Anzahl der Messpunkte pro Umdrehung, Sample-Frequenz) gesetzt.

Zu berechnen:

- Drehzahlen aller Bauteile,
- Zahneingriffsfrequenzen und
- Lagerschadensfrequenzen.

[4] Grund hierfür war der große Aufwand für die Eingabe der benötigten Werte. Für ein Lager müssten, für jede Komponente, Leistungsklasse, die ersten fünf Harmonischen sowie die drei oberen und unteren Seitenbänder, die Warn- und Grenzwerte eingetragen werden. Dadurch ergäben sich über hundert Einträge pro Lager. Die Warn- und Grenzwerte werden nun über SPECTIVE direkt ermittelt.

Zu erstellende XML-Datei

- Transfer-Dokument (Schnittstelle zu SPECTIVE-Zustandsüberwachungssystem)

UC020 – Bilddatei erstellen/speichern (Task revidiert[5])

Vom modellierten System soll zum Beispiel für die spätere Zustandsüberwachung oder für die Präsentationen eine Möglichkeit zur Erstellung von Screenshots bestehen. Das dabei erstellte Bild soll nach Wunsch des Benutzers an einem beliebigen Ort im gewünschten Format speicherbar sein.

9.2. Aktivitätsdiagramme

Mittels Aktivitätsdiagramm werden schwerpunktmäßig prozedurale Abläufe dargestellt. Aktivitätsdiagramme spezifizieren den Kontroll- und Datenfluss zwischen verschiedenen Arbeitsschritten, den Aktionen, die zur Realisierung einer Aktivität notwendig sind [15].

Eine Aktivität besteht aus meist mehreren Aktionen und Subaktivitäten sowie als offene Pfeile dargestellte Kanten, die den Kontrollfluss und/oder Objektfluss zwischen den Aktivitäten und von und zu Kontrollknoten darstellen. Mittels Kontrollknoten können Konzepte wie Verzweigungen und Parallelitäten in Aktivitäten abgebildet werden. Für die Beschreibung der Aktivitäten werden ausschließlich die in Abbildung 21 dargestellten Elemente verwendet.

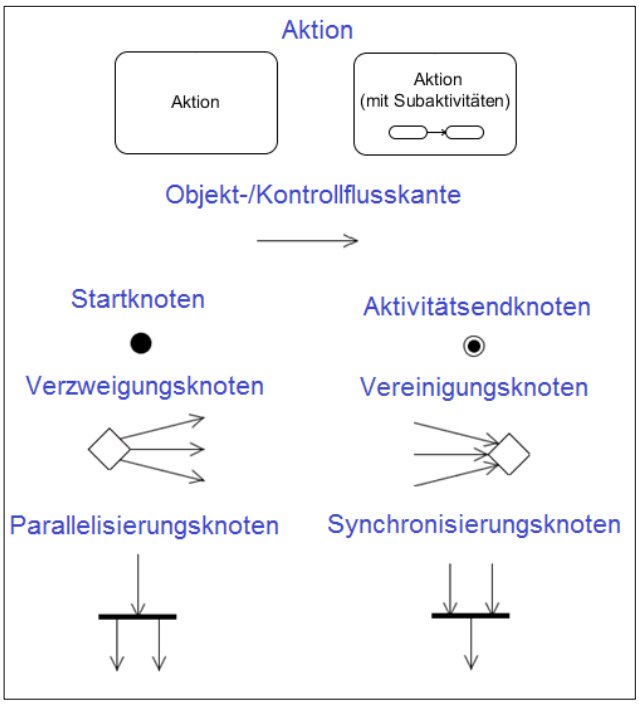

Abbildung 21: Aktivitätsdiagrammelemente

[5] Kann auch über andere, bereits vorhandene Tools durchgeführt werden.

In Abbildung 22 ist zum leichteren Einstieg die Aktivität „Radfahren“ dargestellt. Die Grundlagen sowie weitere Konzepte wie zum Beispiel Partitionierung oder Fehlerbehandlung werden in [15] genauestens beschrieben. Aktivitätsdiagramme stellen ein wichtiges Hilfsmittel für die Implementierung und das Testen des Systems dar. Sie ermöglichen es, die Arbeit in kleinere Einheiten (Aktionen, Subaktivitäten) zu zerlegen. Jeder Anwendungsfall wird von mindestens einem Aktivitätsdiagramm gedeckt.

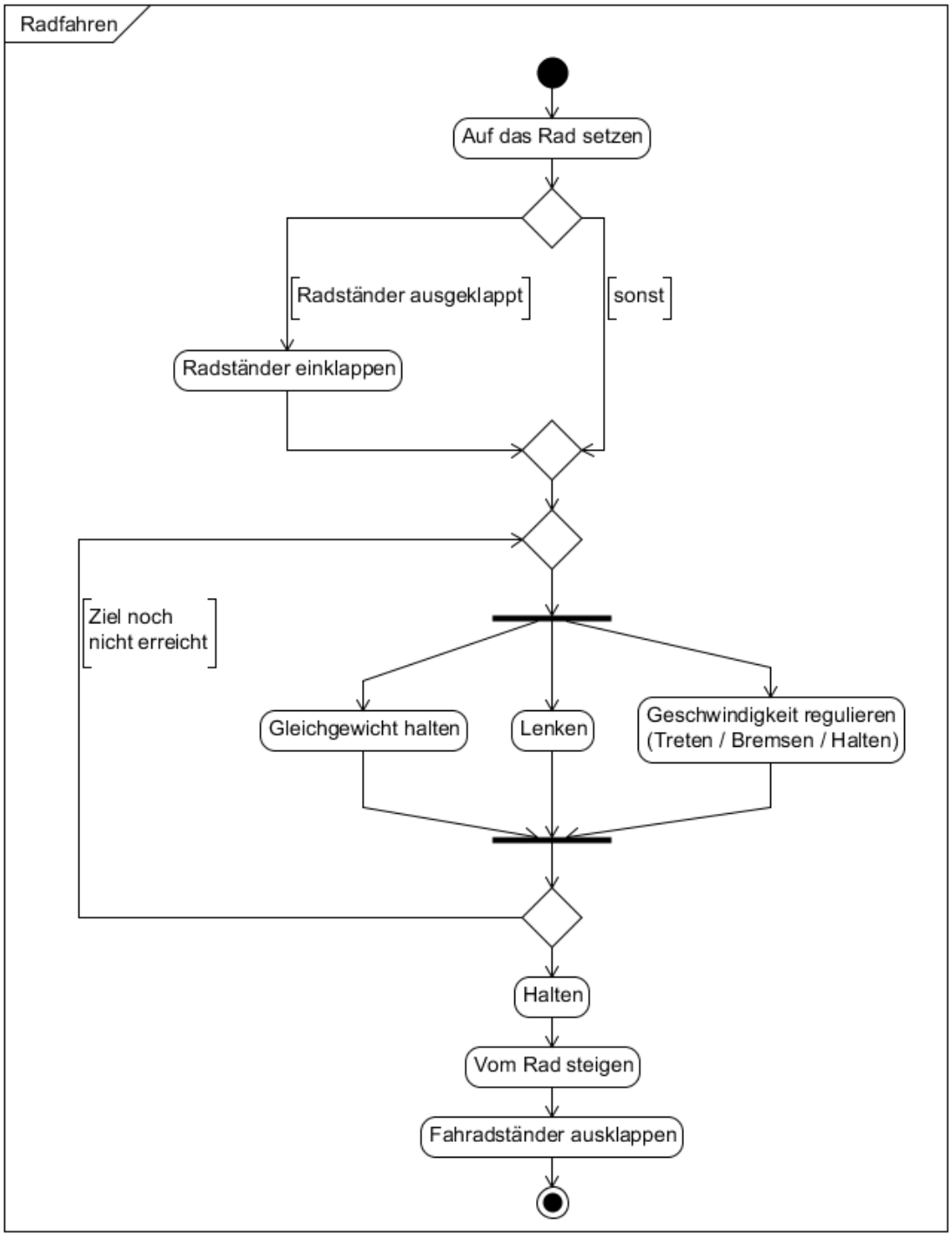

Abbildung 22: Aktivitätsdiagramm Beispiel „Radfahren“

9.2.1 Getriebe/Projekt neu anlegen

Das Anlegen eines neuen Getriebes/Projektes soll für den Anwender möglichst intuitiv erfolgen. In Abbildung 23 ist der gedachte Ablauf zum Anlegen eines neuen Projektes bzw. Getriebes ersichtlich. Es sei hier darauf hingewiesen, dass bei dem hier dargestellten Ablauf kontrolliert wird, ob ein entsprechendes Getriebe mit dem gewählten Namen und Pfad bereits existiert. Um das unabsichtliche Überschreiben bestehender Dateien zu vermeiden, wird deshalb nach dem Betätigen des Bestätigungsbuttons kontrolliert, ob das Projekt bereits besteht. In diesem Falle wird der Benutzer über das Problem informiert und gefragt, ob er die Daten überschreiben möchte.

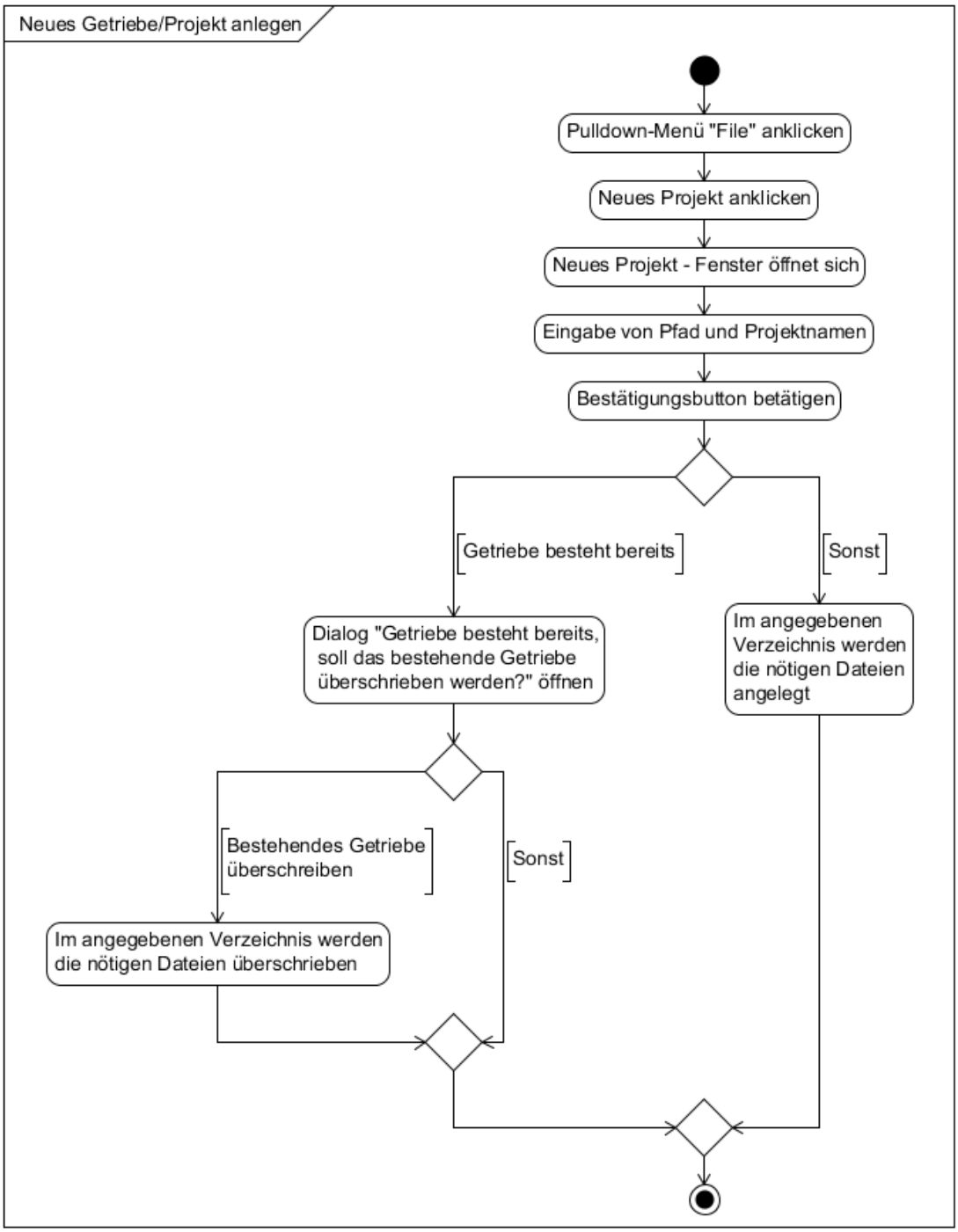

Abbildung 23: UML-Aktivitätsdiagramm „Neues Getriebe/Projekt anlegen“

9.2.2 Getriebe/Projekt speichern

Die Abbildungen 24 und 25 zeigen den prinzipiellen Ablauf des Speicherns der modellierten Getriebe bzw. Projekte und deren Daten. Beim Speichern des Projektes wird danach unterschieden, ob die modellierten Daten unter dem aktuell geöffneten Projekt (Save) oder unter einem neuen Projekt (Save as) gespeichert werden sollen. Im Falle eines neu anzulegenden Projektes wird zunächst untersucht, ob das Projekt nicht unter dem entsprechend angegebenen Pfad bereits besteht. In diesem Fall wird, analog zur Aktivität „neues Getriebe/Projekt anlegen“, der Benutzer mittels Dialogfeld befragt, ob er das bestehende Projekt überschreiben möchte.

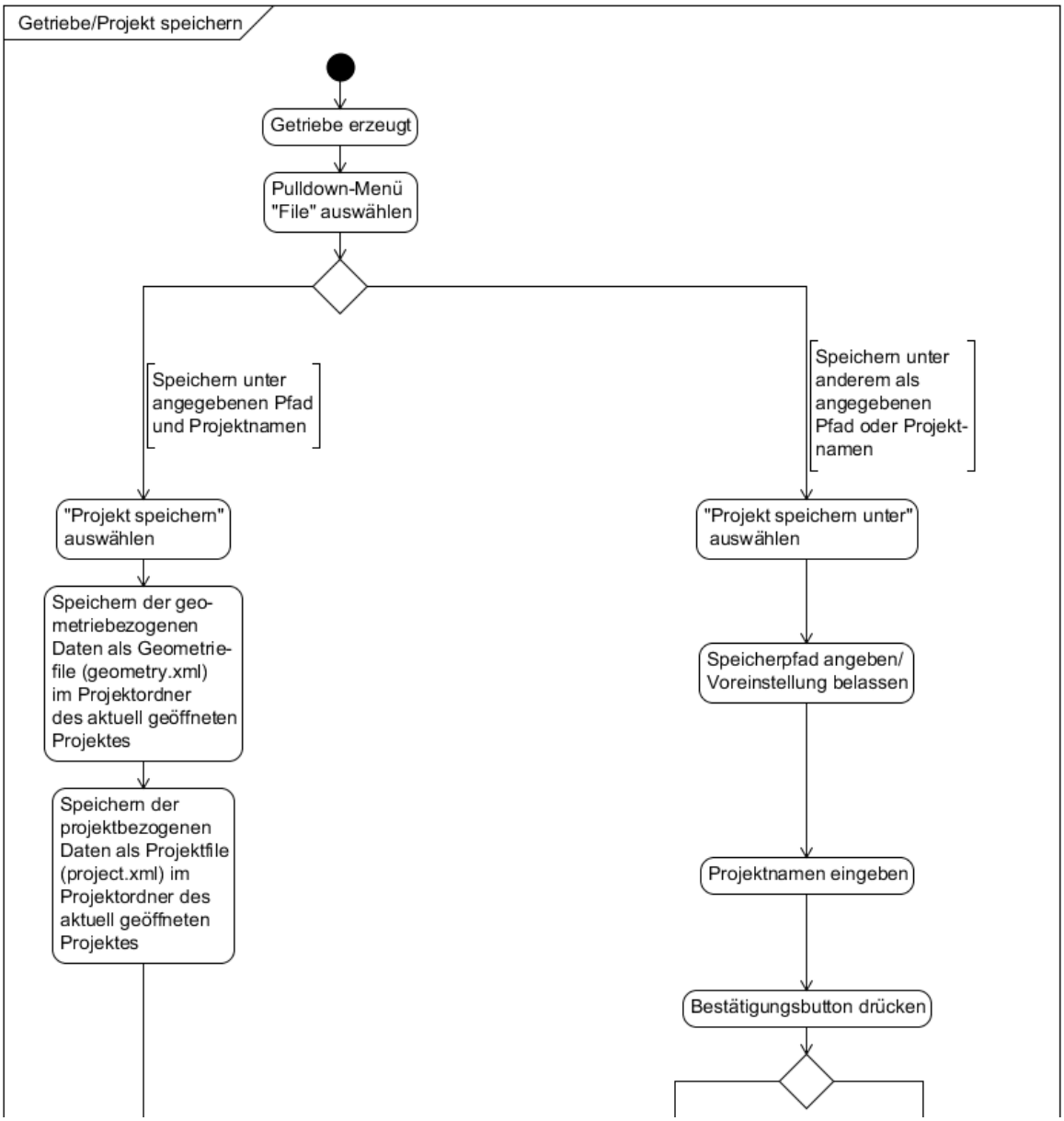

Abbildung 24: UML-Aktivitätsdiagramm „Getriebe/Projekt speichern“ (erster Teil)

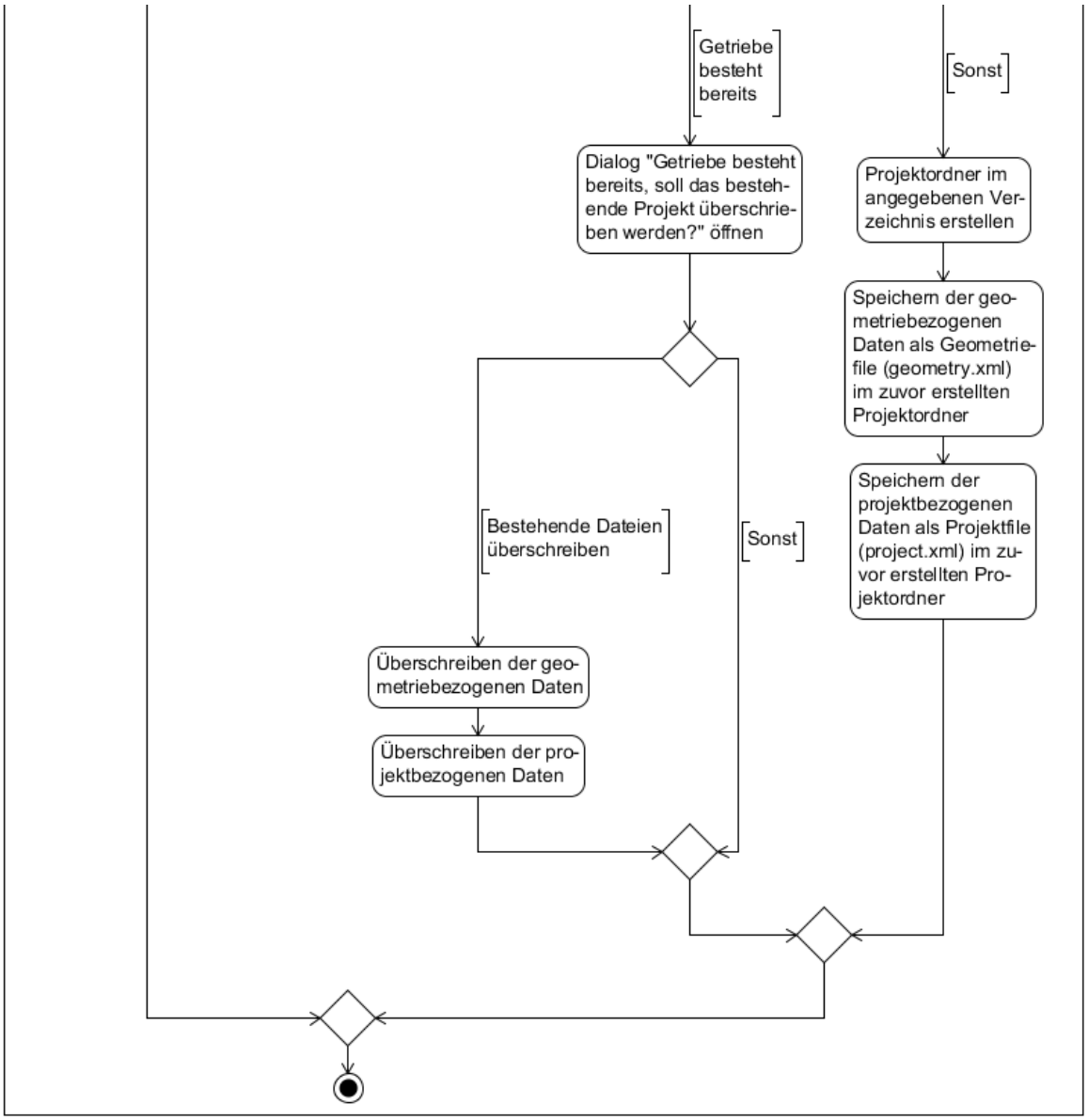

Abbildung 25: UML-Aktivitätsdiagramm „Getriebe/Projekt speichern“ (zweiter Teil)

9.2.3 Getriebe/Projekt laden

Abbildung 26 zeigt den prinzipiellen Ablauf zum Laden eines bestehenden Getriebedatensatzes in das GetriebeStudio. Als Alternative könnten die Projektdaten auch mittels Datei-Explorer geladen werden. Dies wurde jedoch aus Zeitgründen nicht umgesetzt.

9.2.4 Komponente oder Baugruppe hinzufügen

Das Hinzufügen von Komponenten (Bauteilen und Baugruppen) erfolgt durch die Auswahl aus einem Baukasten. Der Ablauf zum Hinzufügen einer Komponente ist in Abbildung 27 dargestellt. Nach der Wahl der hinzuzufügenden Komponente, wird diese zentrisch im Graphikbereich des ersten Reiters des Hauptfensters angezeigt. Steht nur ein Bildschirm zur Verfügung, wird das Fenster zur Auswahl verfügbarer Bauteile geschlossen. Sind mehrere Bildschirme

verfügbar, wird das Fenster zunächst im zweiten Bildschirm geöffnet und wird erst geschlossen, wenn der Nutzer dieses Fenster nicht mehr benötigt. Die Komponente wird automatisch vorselektiert, alle anderen Komponenten deselektiert, wodurch diese mittels Drag&Drop-Funktionalität rasch an die gewünschte Position verschoben werden kann.

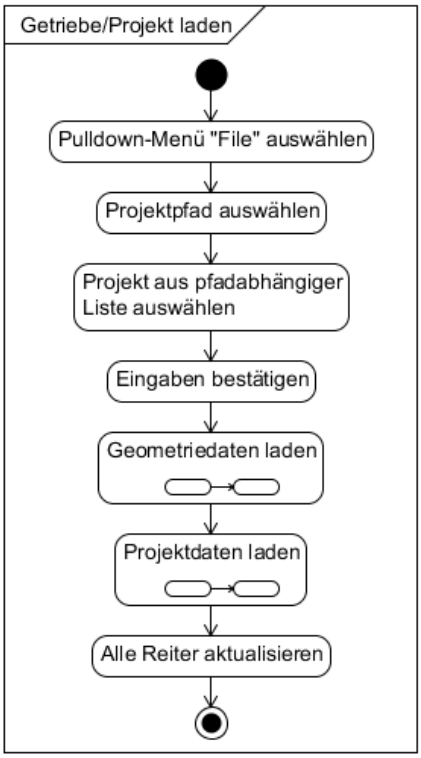

Abbildung 26: UML-Aktivitätsdiagramm „Getriebe/Projekt laden“

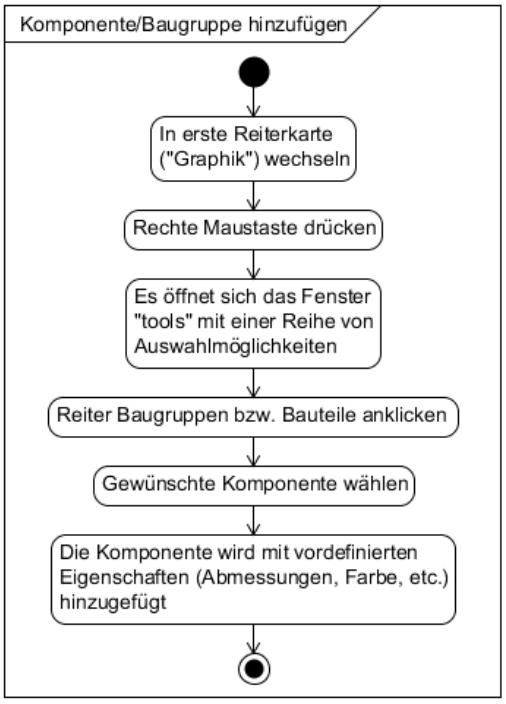

Abbildung 27: UML- Aktivitätsdiagramm „Komponente oder Baugruppe hinzufügen“

9.2.5 Komponeten oder Baugruppe modifizieren

Der Ablauf zum Modifizieren von Komponeten bzw. Baugruppen ist in Abbildung 28 dargestellt. Hierbei ist zu beachten, dass Komponenten grundlegende gemeinsame Eigenschaften wie etwa Namen oder Koordinaten besitzen. Je nach Komponententyp werden weitere Angaben betreffend der Eigenschaften wie etwa Abmessungen benötigt. Abhängig von der Art der zu modifizierenden Komponente werden daher unterschiedliche Eigenschaften zur Modifikation angeboten. Die Grundfarbe wird nicht als Zahlenwert oder RGB angezeigt sondern in Form eines passend gefärbten Rechtecks. Zur Modifikation der Farbe wird diese aus einer Palette durch Selektion per Doppelklick gewählt.

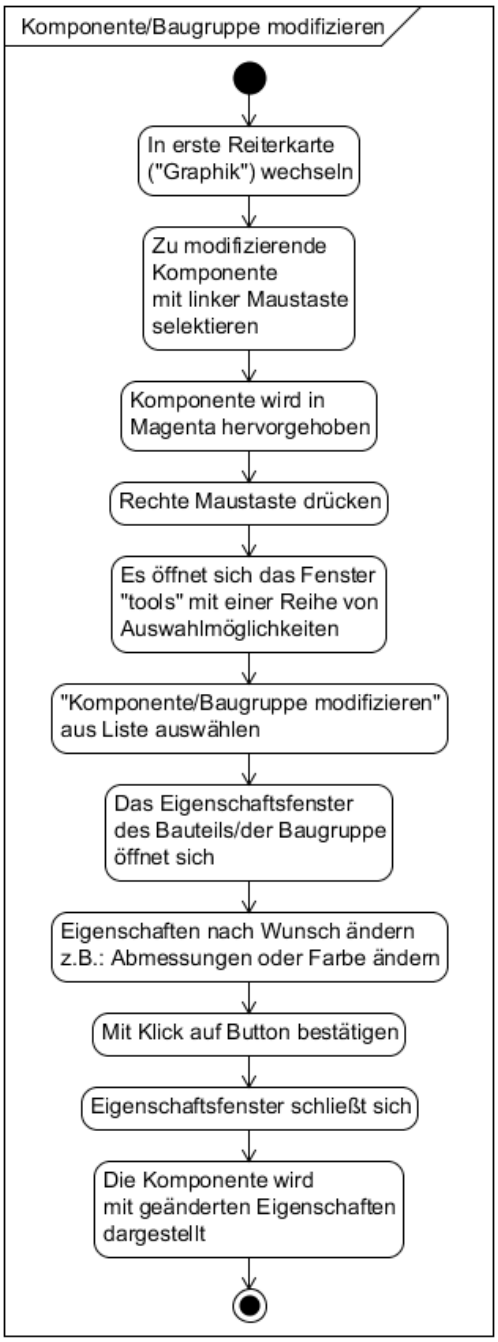

Abbildung 28: UML-Aktivitätsdiagramm „Komponente/Baugruppe modifizieren“

9.2.6 Komponente oder Baugruppe löschen

Falls Komponenten nicht mehr benötigt werden oder versehentlich hinzugefügt wurden, können diese nach erfolgter Selektion und nach Auswahl der Funktion „Komponete/ Baugruppe löschen“ im Fenster „tools“ gelöscht werden (die Verknüpfungen zu anderen Bauteilen und Baugruppen, die Verknüpfung zum Projekt und sonstige gesetzte Verknüpfungen (u.a. gesetzte Referenzwelle) werden entfernt). Der prinzipelle Ablauf zum Löschen einer oder mehrerer Komponenten und/oder Baugruppen ist in Abbildung 29 dargestellt.

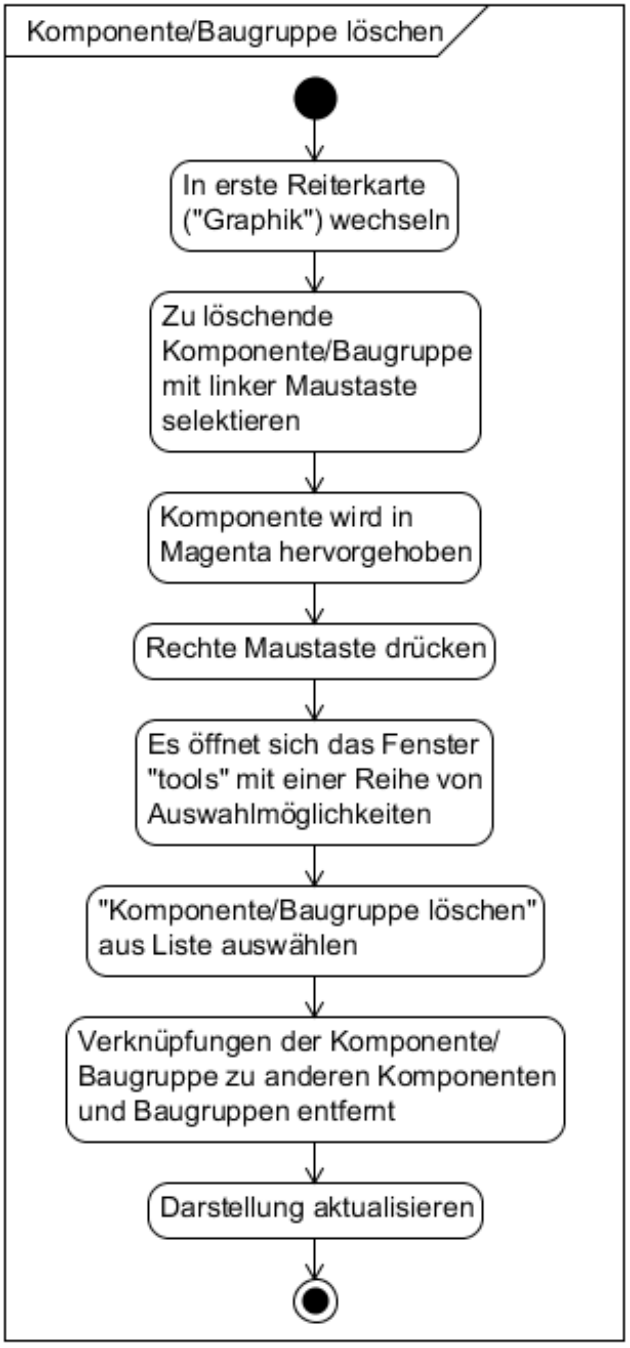

Abbildung 29: UML-Aktivitätsdiagramm „Komponente oder Baugruppe löschen“

9.2.7 Komponenten miteinander verknüpfen

Nach dem Erstellen der benötigten Bauteile müssen diese miteinander zu einem Getriebe kombiniert werden. Dies wird durch das Verknüpfen der einzelnen Bauteile erreicht. Hierbei wird zunächst unterschieden, welche Bauteile miteinander verknüpft werden sollen. Falls eine Verknüpfung möglich ist, werden die Bauteile bzw. Objekte miteinander verknüpft. Hierzu besitzen alle Bauteile und Baugruppen eine oder mehrere Tabellen in denen die verknüpften Teile angeführt sind. Die verknüpften Teile werden im Anschluss an die eigentliche Verknüpfung in ihrer Ausrichtung zueinander und bezüglich Geometrie angepasst.

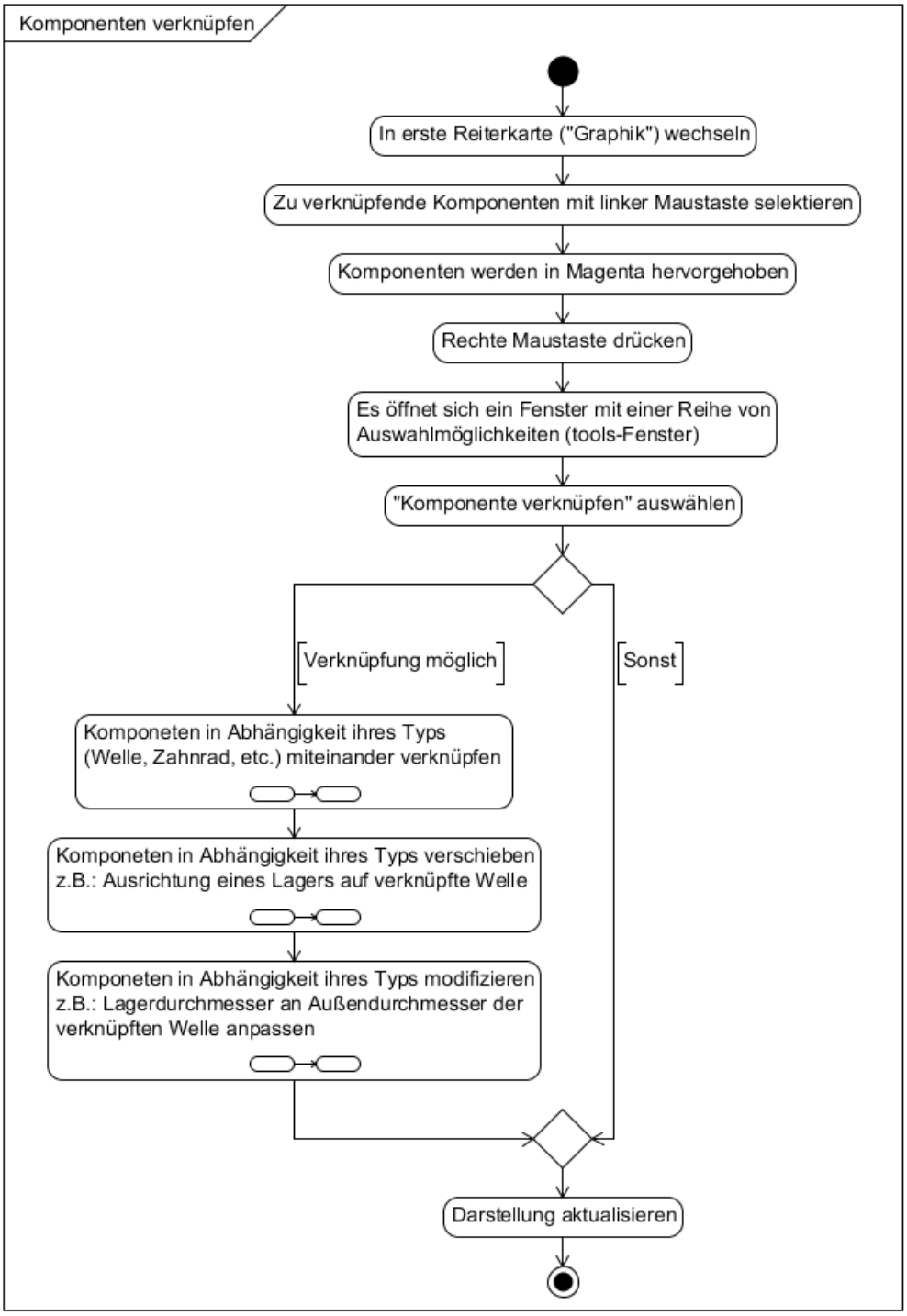

Abbildung 30: UML-Aktivitätsdiagramm „Komponenten verknüpfen“

9.2.8 Komponenten referenzieren

Für die Berechnung der Drehzahl der Bauteile und sonstigen Parameter ist die Angabe einer Referenzwelle erforderlich. Die spätere Berechnung der Drehzahlen der modellierten, verknüpften Bauteile erfolgt ausgehend von der Drehzahl der referenzierten Welle, deren Drehzahl zwingend bekannt sein muss. Alternativ zum, in Abbildung 31, dargestellten Ablauf, kann die Auswahl der Referenzwelle auch über die Reiterkarte „Konfiguration und Berechnung“ erfolgen. Die Referenzwelle wird hierbei aus einer Reihe von Einträgen verfügbarer Wellen, Planetenträger und Hohlräder ausgewählt.

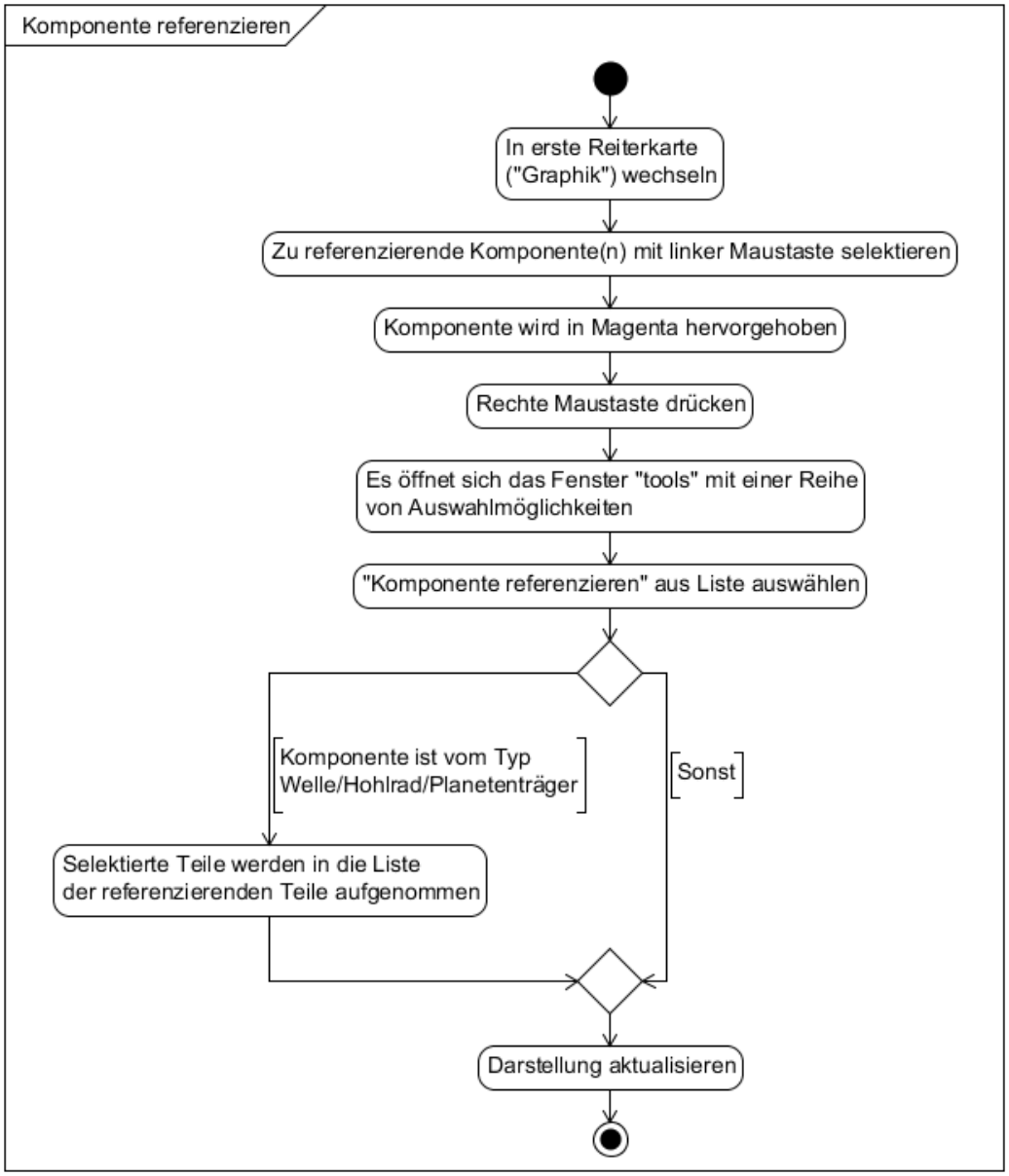

Abbildung 31: UML-Aktivitätsdiagramm „Komponenten referenzieren“

9.2.9 Referenzierte Komponenten anzeigen

Neben der Auswahl einer Referenzwelle (siehe Kapitel 9.2.8), soll es auch möglich sein, die referenzierte Komponente, z.B. eine Welle, im Graphikbereich bei Bedarf hervorzuheben. Abbildung 32 zeigt den hierzu vorgegebenen Ablauf.

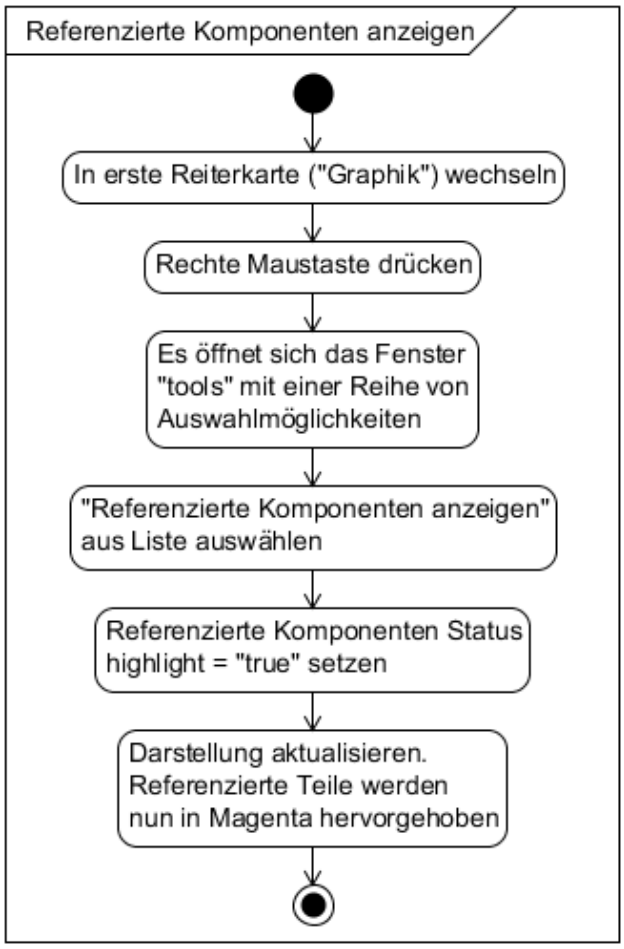

Abbildung 32: UML-Aktivitätsdiagramm „referenzierte Komponenten anzeigen“

9.2.10 Komponenten mit Baugruppe verknüpfen

Neben dem Verknüpfen von Bauteilen mit anderen Bauteilen ist es bei Bedarf erforderlich Bauteile mit Baugruppen zu verknüpfen. Der Vorgang ähnelt dem Verknüpfen von Bauteilen mit anderen Bauteilen. Abbildung 33 zeigt den hierzu als Aktivitätsdiagramm visulisierten Ablauf.

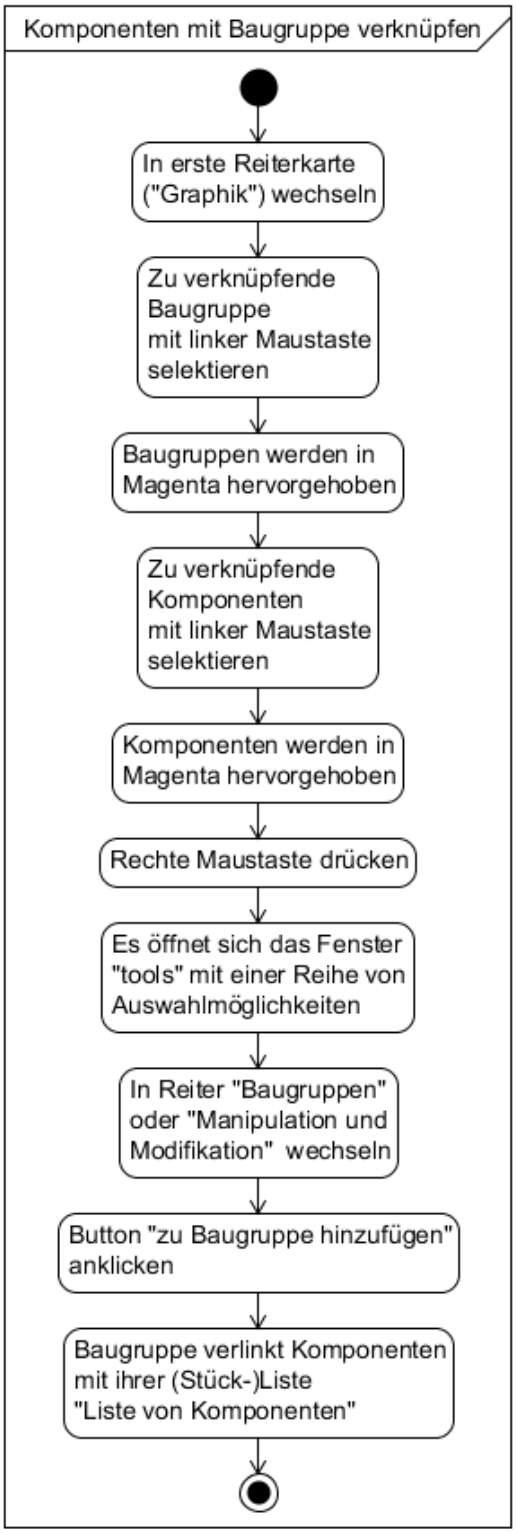

Abbildung 33: UML-Aktivitätsdiagramm „Komponenten mit Baugruppe verknüpfen“

9.2.11 Baugruppen spezifizieren

Nachdem eine Baugruppe angelegt und alle nötigen Bauteile mit dieser verknüpft wurden, muss die Baugruppe noch genauer spezifiziert werden. Hierzu wird ein weiterer Reiter („Baugruppe“) hinzugefügt. Beim Wechsel in den entsprechenden Reiter werden alle bis dahin modellierten Baugruppen mit ihren Teilen angezeigt. Sind für ein Bauteil einer Baugruppe verschiedene Spezifikationsmöglichkeiten vorhanden, können diese über eine Auswahlliste (Combo) dem Bauteil zugewiesen werden. Der gedachte prinzipielle Ablauf ist in Abbildung 34 dargestellt.

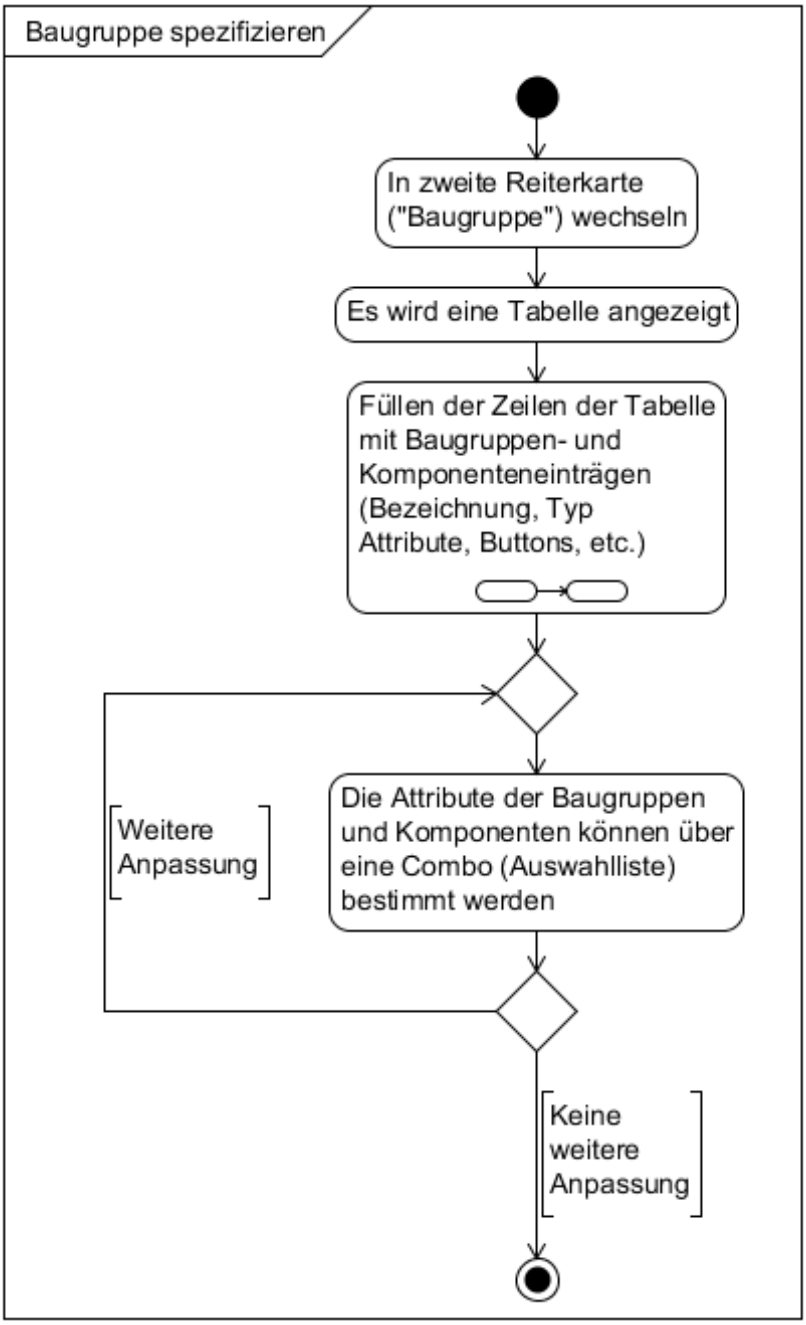

Abbildung 34: UML-Aktivitätsdiagramm „Baugruppe spezifizieren“

9.2.12 Lager anlegen

Neben dem Hinzufügen von Lagern zum Getriebe (vgl. Kapitel 9.2.4) ist es für die nachfolgende Berechnung der Schadensfrequenzen der Lager bzw. derer Komponenten erforderlich, diese mit einem Datensatz eines realen Lagers zu verknüpfen (vgl. Kapitel 9.2.13). Die hierzu nötigen Datensätze wurden zunächst mittels eines selbstentwickelten Parsers und einem bereits vorhandenem Textdokument häufig genutzter Lager, dass der Projektpartner Windkraft Simonsfeld AG freundlicherweise zur Verfügung stellte, in die gewünschte Form (XML) gebracht. Der hierbei entstandene Lagerkatalog enthält bereits ca. 300 Lagerdatensätze (vgl. Abbildung 36). Weitere Datensätze können, wie in Abbildung 35 dargestellt, dem Lagerkatalog bequem hinzugefügt werden.

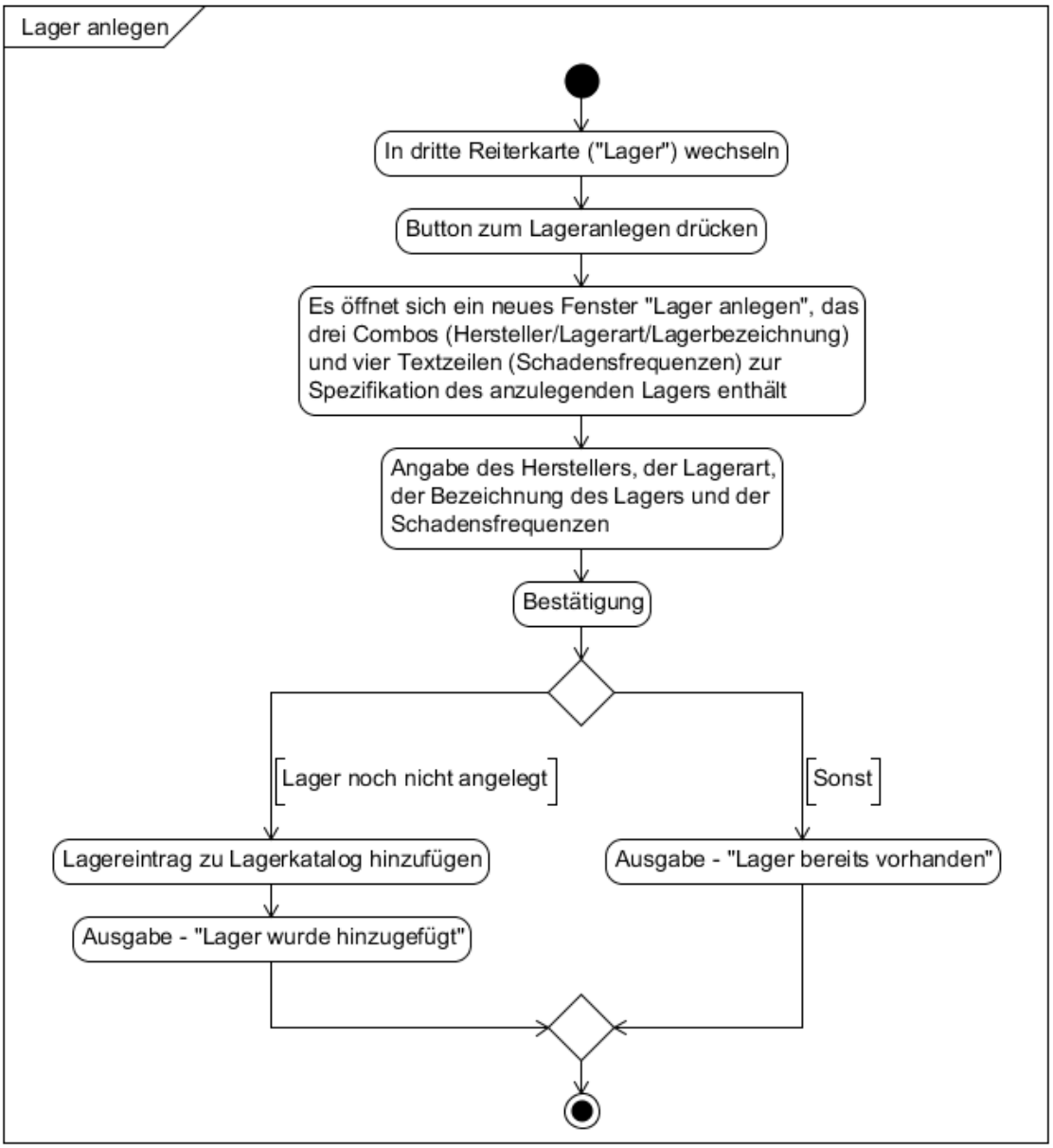

Abbildung 35: UML-Aktivitätsdiagramm „Lager anlegen“

Beim Anlegen eines neuen Lagers muss der Hersteller (z.B.: SKF, INA, FAG), die Lagerart (Rillenkugellager, Zylinderlager, etc.) und die Bezeichnung des Lagers ausgewählt werden. Weiters sind die normierten Lagerschadensfrequenzen anzugeben. Für die genaue Bestimmung dieser Frequenzen sind die geometrischen Abmessungen von Innen- und Außenring, des Käfigs und der Wälzkörper notwendig. Meist werden vom Hersteller nur die Hauptabmessungen (Außendurchmesser, Bohrung und Breite des Lagers) angegeben. Die Lagerhersteller stellen jedoch häufig eine Möglichkeit zur Bestimmung dieser Frequenzen bereit. SKF bietet unter [18] ein Wälzlagerberechnungsprogramm an, das neben der Lebensdauer eines Lagers auch dessen Schadensfrequenzen ermitteln kann. Die Frequenzen von FAG- und INA-Wälzlagern können unter [19] berechnet werden. Im Laufe der Entwicklung des Programmes „GetriebeStudio“ wurde der Lagerkatalog ständig mit zusätzlichen Lagern erweitert. Die hierfür benötigten normierten Schadensfrequenzen[6] wurden dabei über die zuvor erwähnten Kalkulationsprogramme der Lagerhersteller ermittelt.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <Waelzlagerkatalog>
  - <Hersteller Name="FAG">
    - <Lagerart Name="unbekannte Lagerart">
      - <Lager Name="230/600B">
          <Schadfrequenz Name="Innenring">16.455</Schadfrequenz>
          <Schadfrequenz Name="Außenring">13.667</Schadfrequenz>
          <Schadfrequenz Name="Wälzkörper">10.646</Schadfrequenz>
          <Schadfrequenz Name="Käfig">0.455</Schadfrequenz>
        </Lager>
      - <Lager Name="230/630B">
          <Schadfrequenz Name="Innenring">16.39</Schadfrequenz>
          <Schadfrequenz Name="Außenring">13.61</Schadfrequenz>
          <Schadfrequenz Name="Wälzkörper">10.56</Schadfrequenz>
          <Schadfrequenz Name="Käfig">0.45</Schadfrequenz>
        </Lager>
      - <Lager Name="23244K">
          <Schadfrequenz Name="Innenring">11.37</Schadfrequenz>
          <Schadfrequenz Name="Außenring">8.63</Schadfrequenz>
          <Schadfrequenz Name="Wälzkörper">6.98</Schadfrequenz>
          <Schadfrequenz Name="Käfig">0.43</Schadfrequenz>
        </Lager>
```

...

Abbildung 36: Auszug aus Wälzlagerkatalog.xml

[6] Für die Berechnung der normierten Lagerschadensfrequenzen wurde eine Lagerinnenringdrehzahl von 60 Umdrehungen pro Minute gewählt.

9.2.13 Lager spezifizieren

Wie bereits in Kapitel 9.2.12 erwähnt, wird eine Möglichkeit zum Spezifizieren (Zuweisen von Lagerdatensätzen) der modellierten Lager benötigt. Der hierfür entwickelte Ablauf ist in Abbildung 37 dargestellt.

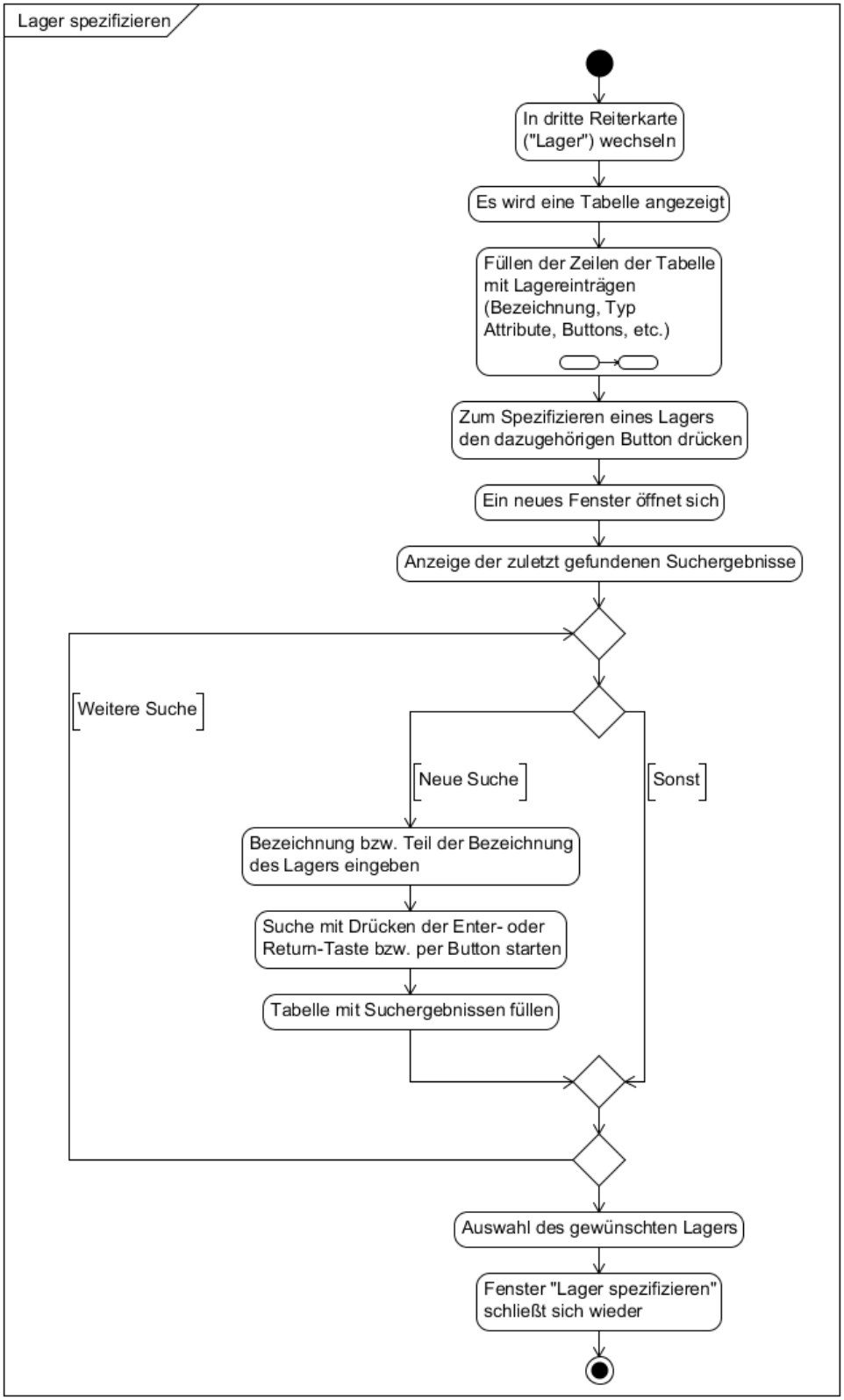

Abbildung 37: UML-Aktivitätsdiagramm „Lager spezifizieren“

9.2.14 Lager identifizieren

Da die Darstellung und Spezifikation der Lager in verschiedenen Reitern erfolgt, benötigt man eine Möglichkeit zum Erkennen, welcher Eintrag (dritter Reiter) zu welchem Lager (erster Reiter) gehört. In Abbildung 38 sind die verschiedenen Möglichkeiten in Form eines Aktivitätsdiagramms dargestellt.

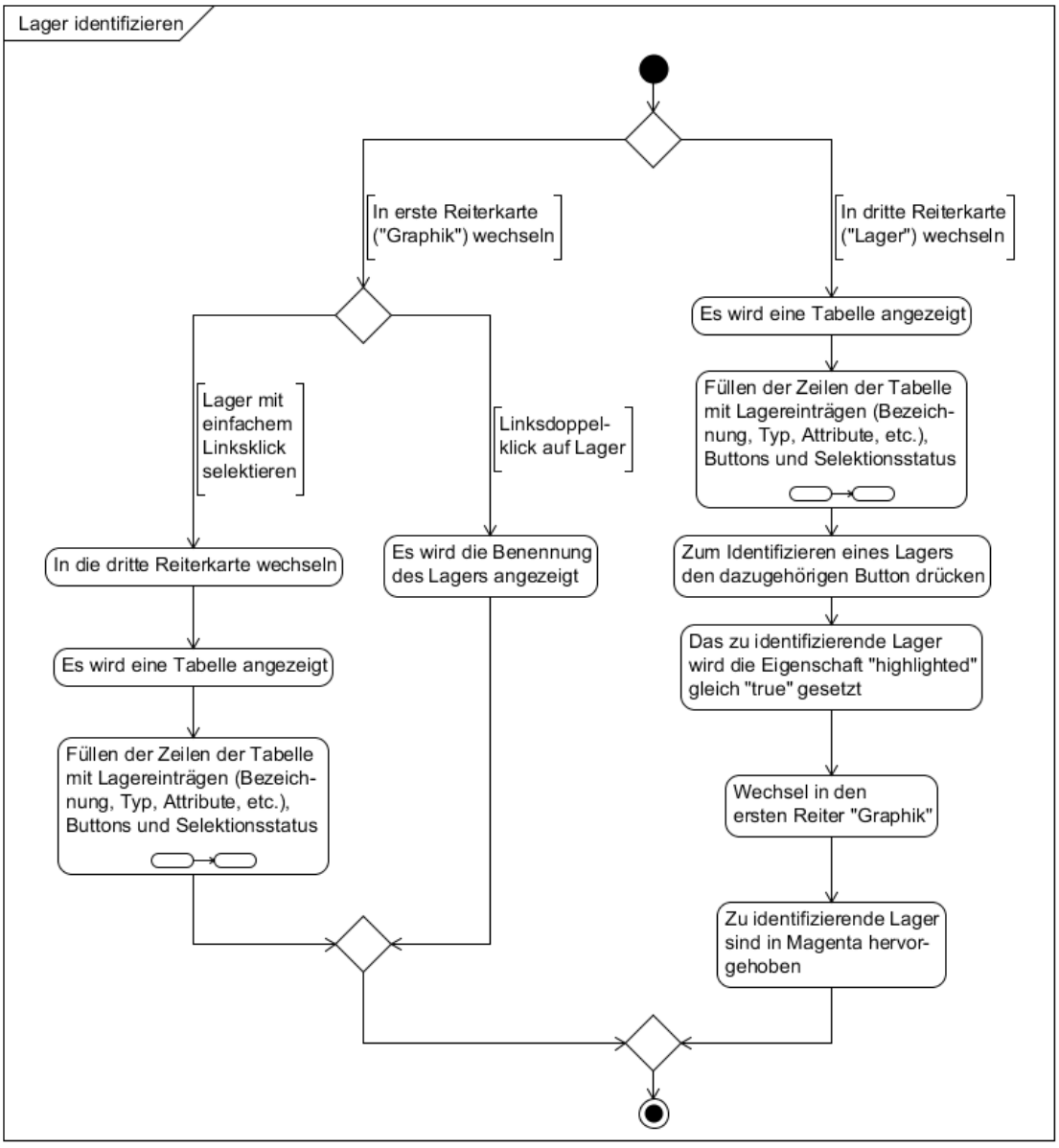

Abbildung 38: UML-Aktivitätsdiagramm „Lager identifizieren“

9.2.15 Sensor anlegen

Das Anlegen eines Sensors erfolgt analog zum Anlegen eines Wälzlagers. Der hierfür spezifizierte Ablauf ist in Abbildung 39 dargestellt.

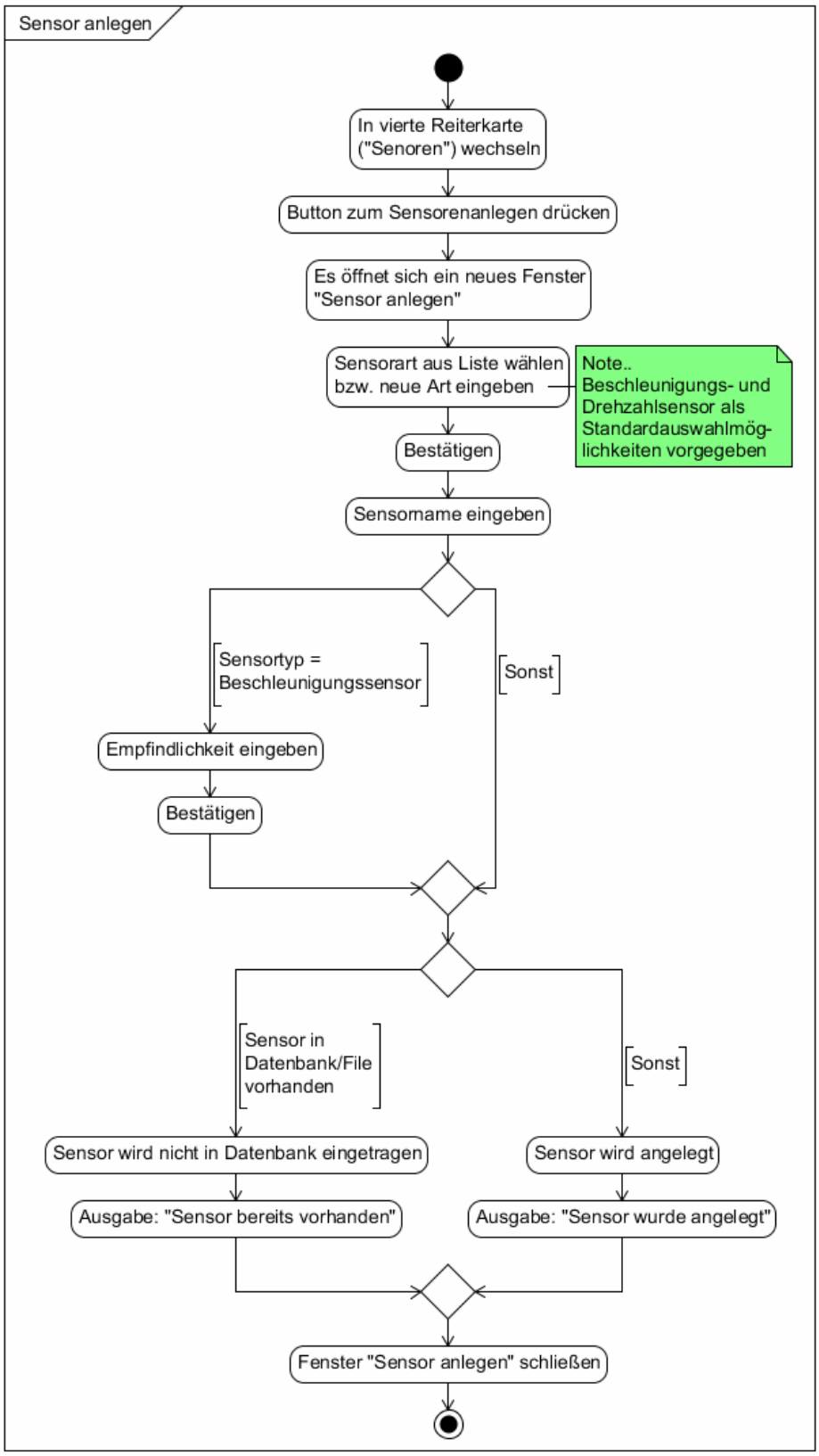

Abbildung 39: UML-Aktivitätsdiagramm „Sensor anlegen“

9.2.16 Sensor hinzufügen

Nachdem der gewünschte Sensor im Sensorkatalog angelegt ist (XML-Dokument, enthält alle bisher angelegten Sensoren) können diese, wie in Abbildung 40 dargestellt, zum Getriebe bzw. zur Tabelle in der vierten Reiterkarte (Liste der Sensoren) hinzugefügt werden.

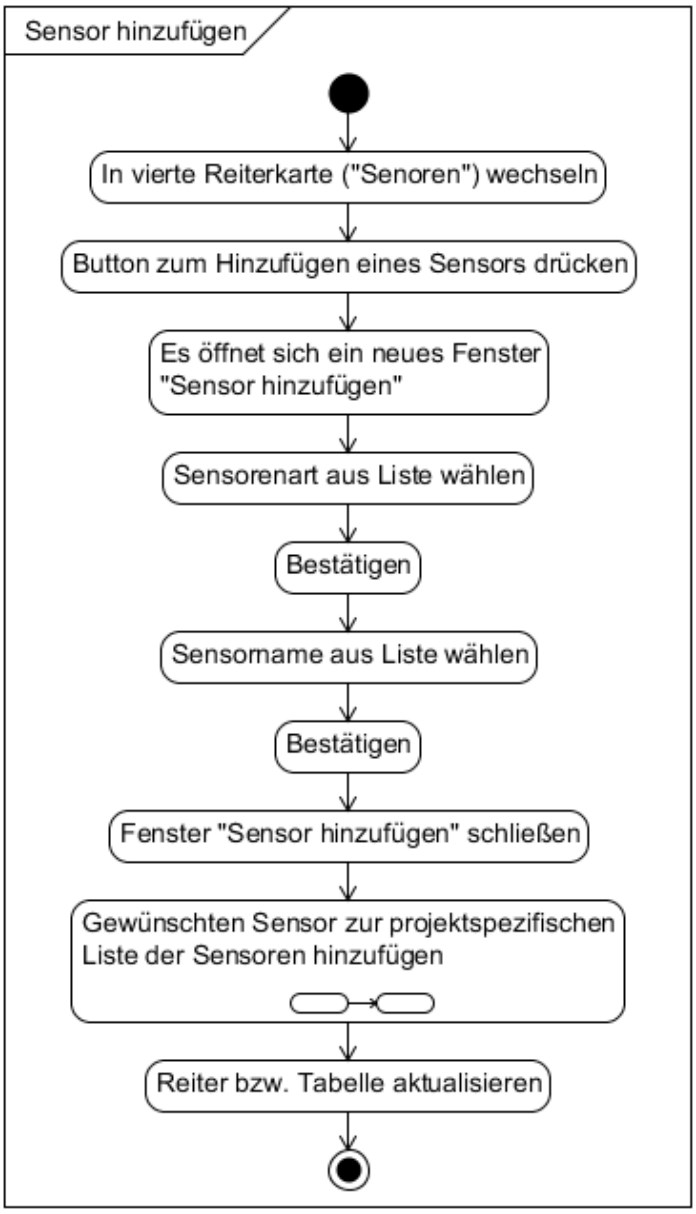

Abbildung 40: UML-Aktivitätsdiagramm „Sensor hinzufügen“

9.2.17 Sensor spezifizieren

Das Spezifizieren von Sensoren erfolgt a) beim Anlegen der Sensoren und b) nach dem Hinzufügen zur Liste der verwendeten Sensoren (Tabelle). Wobei Fall b derzeit nur für Drehzahlsensoren relevant ist. Bei diesen kann das Attribut „bezogen“ auf „False“ oder „True“ gesetzt werden. Das Setzen des „bezogen“-Attributes bewirkt, dass SPECTIVE den gewählten Drehzahlsensor bzw. dessen Drehzahlsignal für die Berechnung der Drehzahl der Referenzwelle heranzieht. Alles andere wird beim Anlegen der Sensoren spezifiziert.

9.2.18 Sensor entfernen

Falls ein Sensor versehentlich zur Tabelle der spezifizierten Sensoren hinzugefügt wurde oder nicht mehr benötigt wird, kann dieser, wie in Abbildung 41 dargestellt, aus der Tabelle entfernt werden.

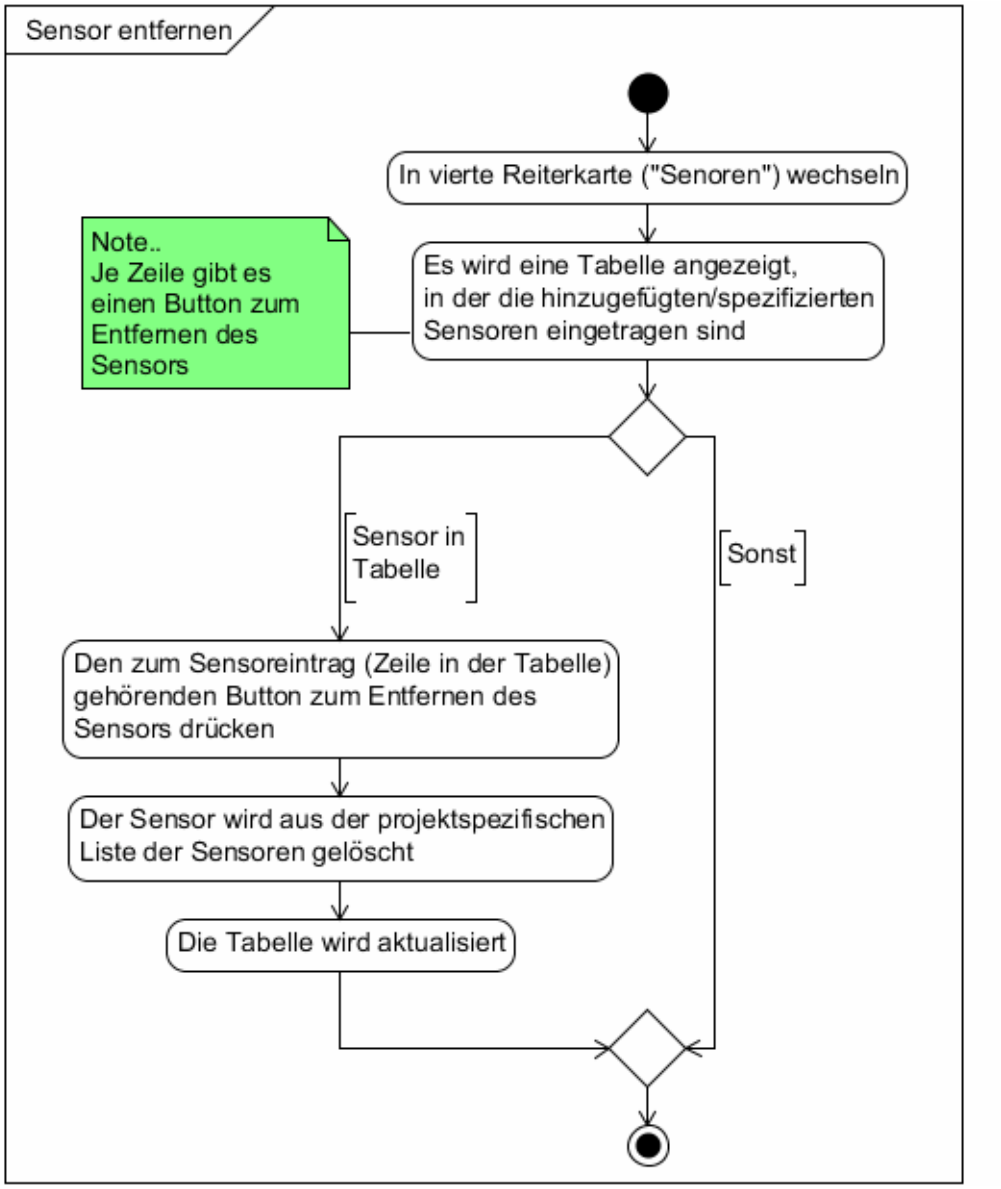

Abbildung 41: UML-Aktivitätsdiagramm „Sensor entfernen“

9.2.19 Drehzahl- und Parameterberechnung

Die Berechnung der Drehzahl der Bauteile und wichtiger Parameter, wie etwa die Lagerschadensfrequenzen, erfordert, dass zuerst ein entsprechendes Getriebe modelliert und eventuell vorhandene Baugruppen, Lager und Sensoren, etc. spezifiziert wurden. Ist dies der Fall, kann die Berechnung nach Eingabe der Referenzwelle, Drehzahl der Referenzwelle etc. im Reiter „Konfiguration und Berechnung“ durch Bestätigung der Daten mittels Button gestartet werden. Im Anschluss werden die gewünschten Informationen berechnet und in einem XML-Dokument (Transferdokument) gespeichert (Ablauf siehe Abbildung 42).

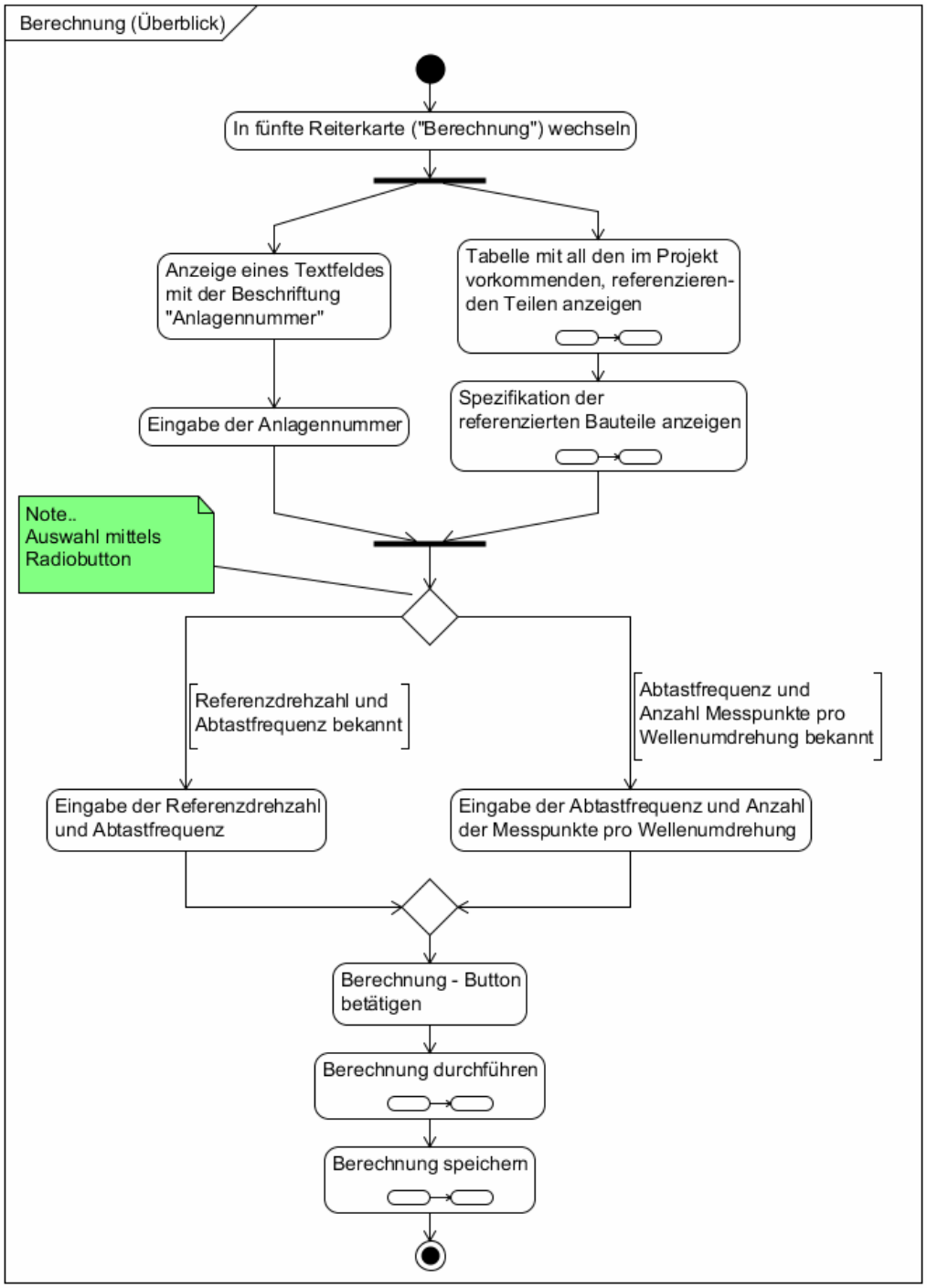

Abbildung 42: UML-Aktivitätsdiagramm „Berechnung (Überblick)“

9.2.19.1 Berechnung durchführen

Die eigentliche Berechnung besteht aus vier Teilen (vgl. Abbildung 43):

1. *Berechnung der Drehzahl der Komponenten*: Jedem Bauteil des Getriebes ist eine Drehfrequenz zuzuordnen.
2. *Berechnung der Lagerschadensfrequenzen*: Ein Wälzlager besteht aus vier Komponenten: Innenring, Außenring, Käfig und Wälzkörper. Jeder Bauteil erzeugt im Betrieb Schadensfrequenzen.
3. *Berechnung der Zahneingriffsfrequenzen*: Zu jedem im Eingriff befindlichen Zahnradpaar ist eine Eingriffsfrequenz zu bestimmen.

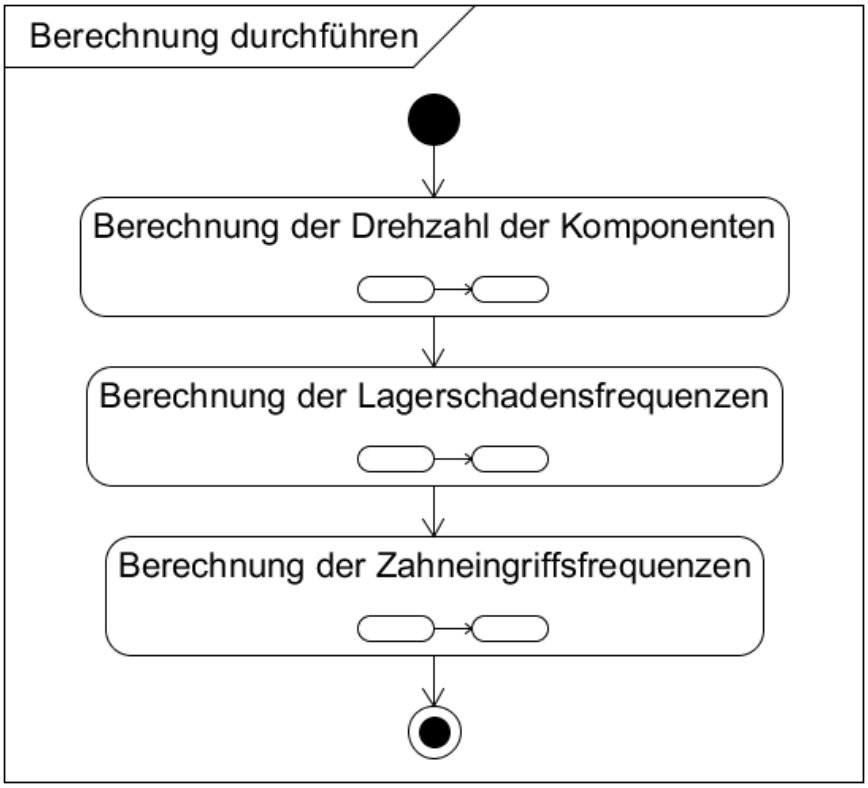

Abbildung 43: UML-Aktivitätsdiagramm „Berechnung durchführen“

Die Subaktivitäten werden in den folgenden Kapiteln 9.2.19.1.1 bis 9.2.19.1.3 näher beschrieben.

9.2.19.1.1 Berechnung der Drehzahl der Komponenten

Die Berechnung der Drehzahl der modellierten und spezifizierten Bauteile und Baugruppen startet bei einer Referenzwelle (Drehzahl der Welle muss bekannt sein).

Die in Abbildung 44 dargestellte Aktivität berücksichtigt bereits, dass es theoretisch mehrere Referenzwellen geben kann. Z.B. bei einem Differenzial oder Planetengetriebe mit mehreren angetriebenen Wellen ist die Angabe von weiteren Referenzwellen und deren Referenzdrehzahl erforderlich.

Die weiteren Abläufe zur Berechnung der Drehzahl der Komponenten sind in den Abbildungen 45 und 46 dargestellt.

Für die Berechnung der Drehzahl der Komponenten einer Baugruppe werden Grundkenntnisse der Kinematik vorausgesetzt. Für bestimmte Baugruppen, wie etwa eine einfache Planetenstufe, können die Zusammenhänge leicht hergeleitet werden (siehe Anhang „Drehzahlberechnung eines Planetengetriebes“) bzw. man entnimmt sie entsprechender Literatur, z.B.: [20] und [21]. Die Herleitung der, für die Berechnung der Drehzahlen der Komponenten eines Planetengetriebes, nötigen Formeln sind im Anhang dieser Arbeit angeführt.

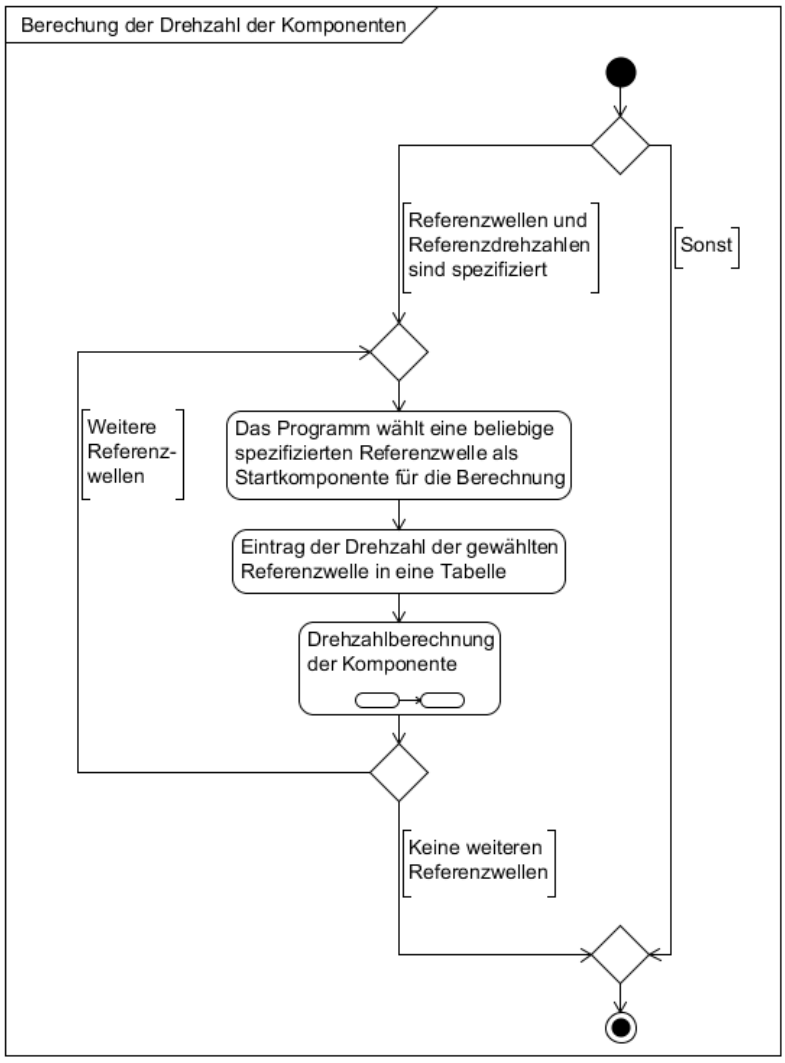

Abbildung 44: UML-Aktivitätsdiagramm „Berechnung der Drehzahl der Komponenten“

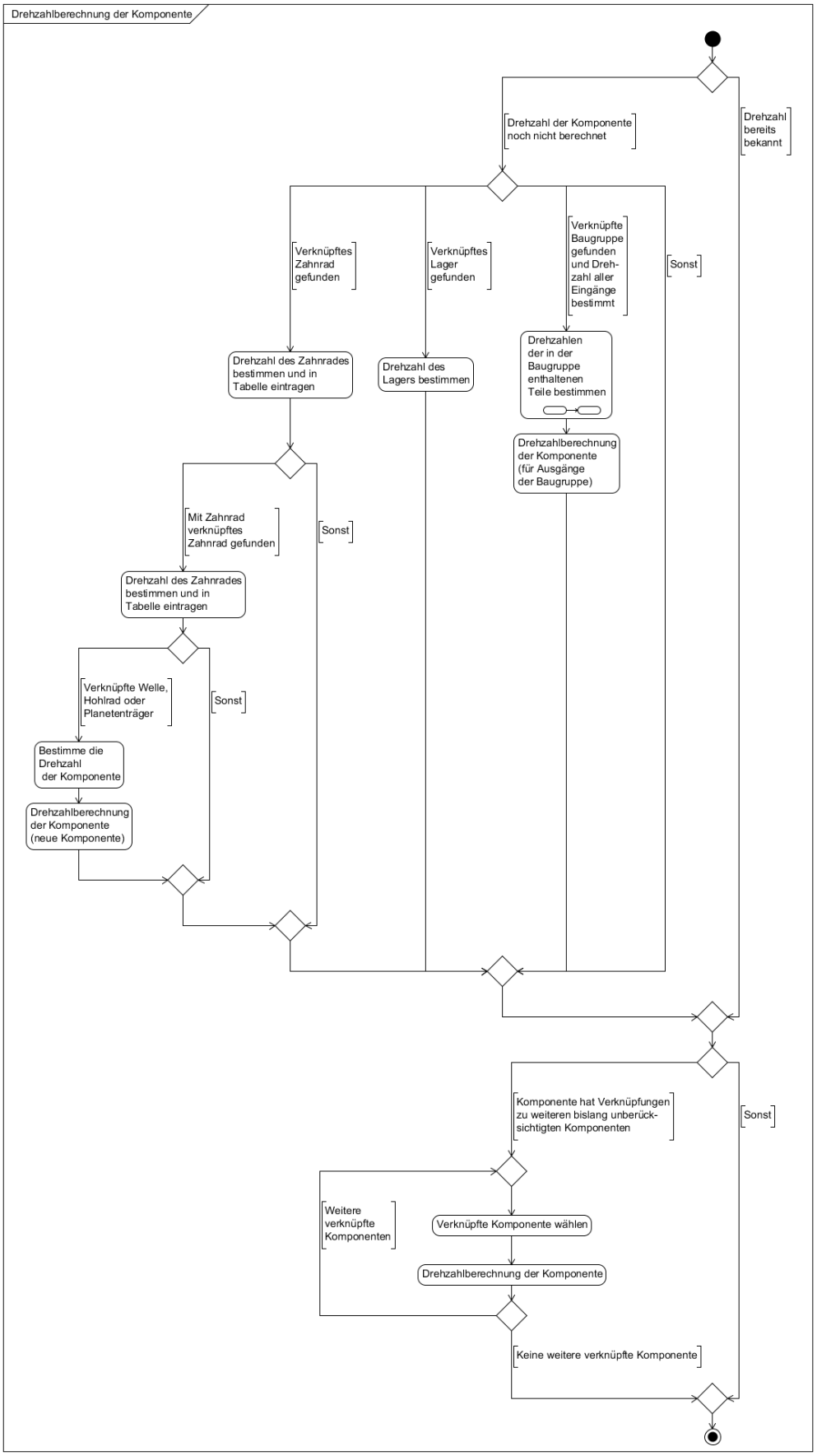

Abbildung 45: UML-Aktivitätsdiagramm „Drehzahlberechnung der Komponente“

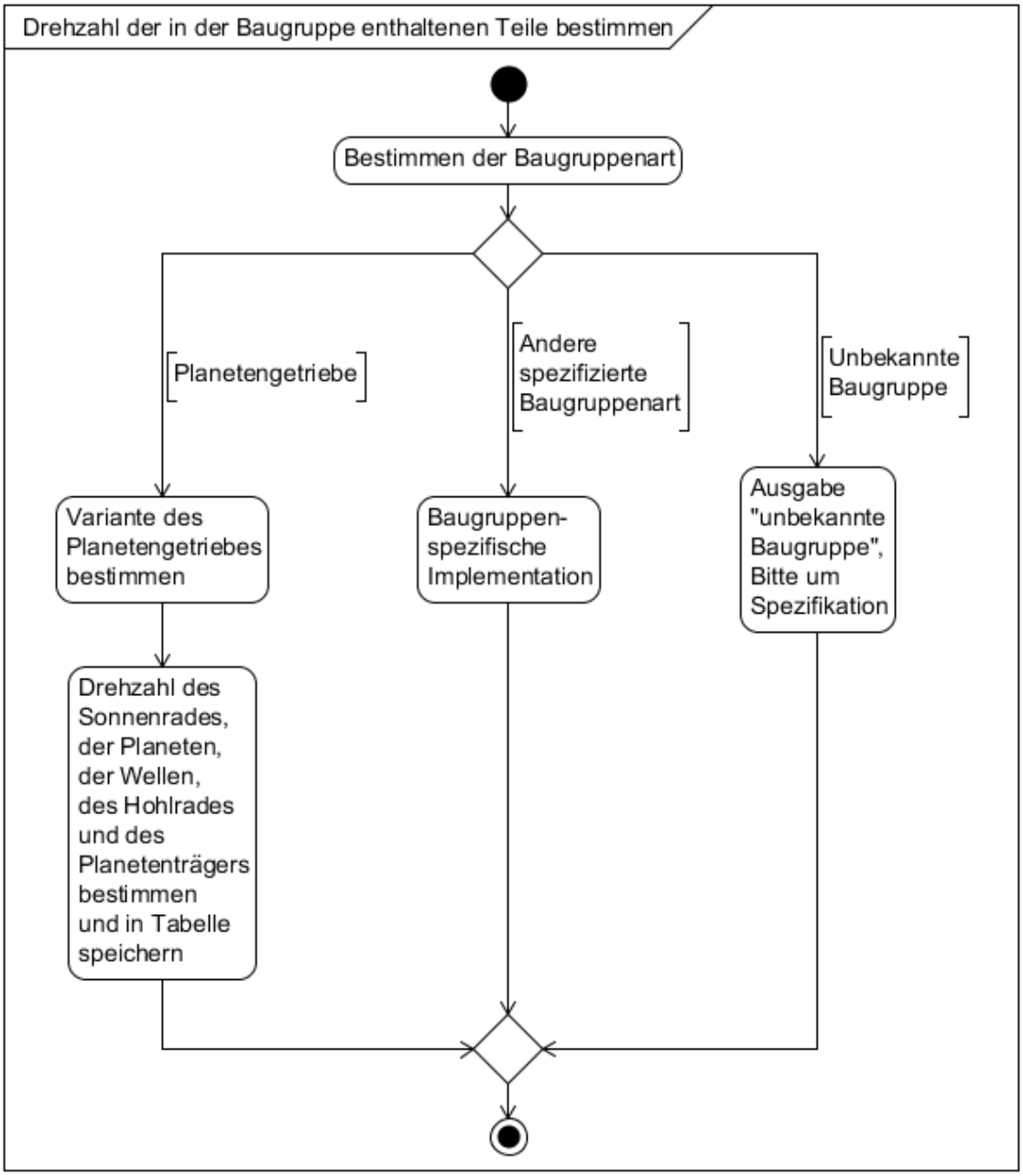

Abbildung 46: UML-Aktivitätsdiagramm „Drehzahl der in der Baugruppe enthaltenen Komponenten bestimmen“

9.2.19.1.2 Berechnung der Lagerschadensfrequenzen

Die Berechnung der Lagerschadensfrequenzen erfordert, dass zuerst die Drehzahl der Lager und die normierten Frequenzen bekannt sind. In den Abbildungen 47 und 48 ist der grundsätzliche Ablauf zur Berechnung der Frequenzen dargestellt. Die Bestimmung der Frequenz des Außenrings, des Käfigs und der Wälzkörper erfolgt analog zu dem, in Abbildung 48, dargestellten Ablauf.

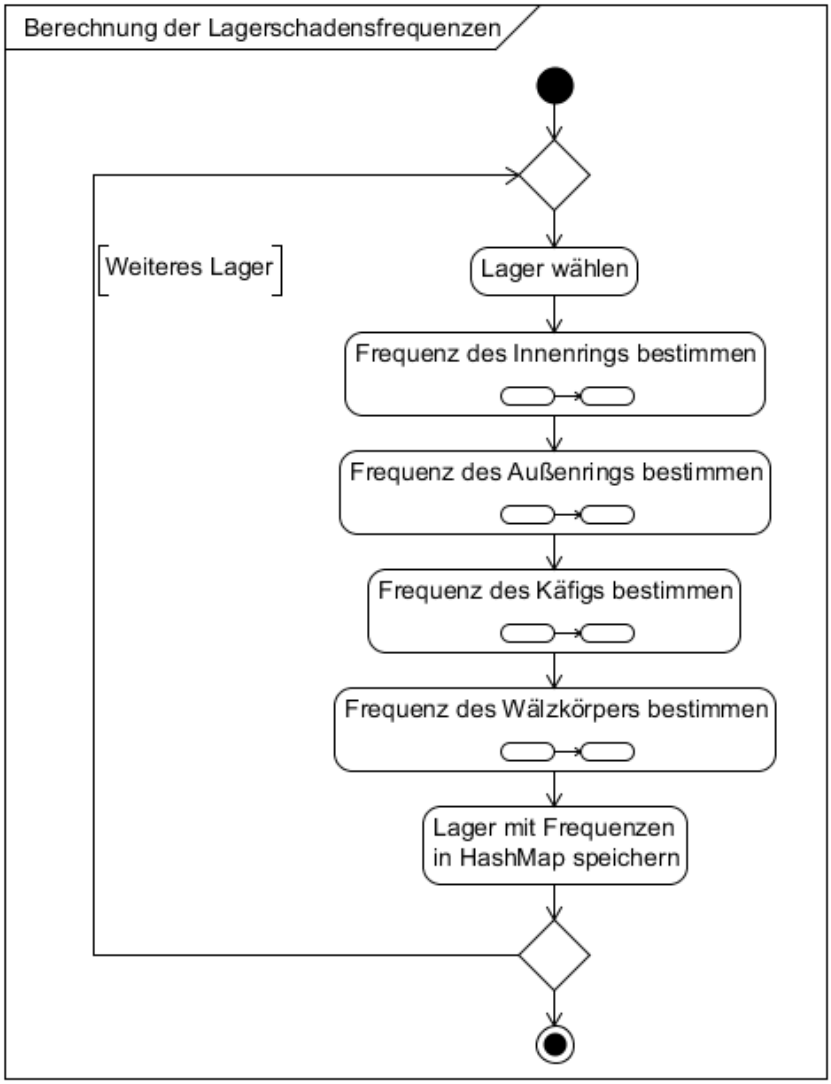

Abbildung 47: UML-Aktivitätsdiagramm „Berechnung der Lagerschadensfrequenzen“

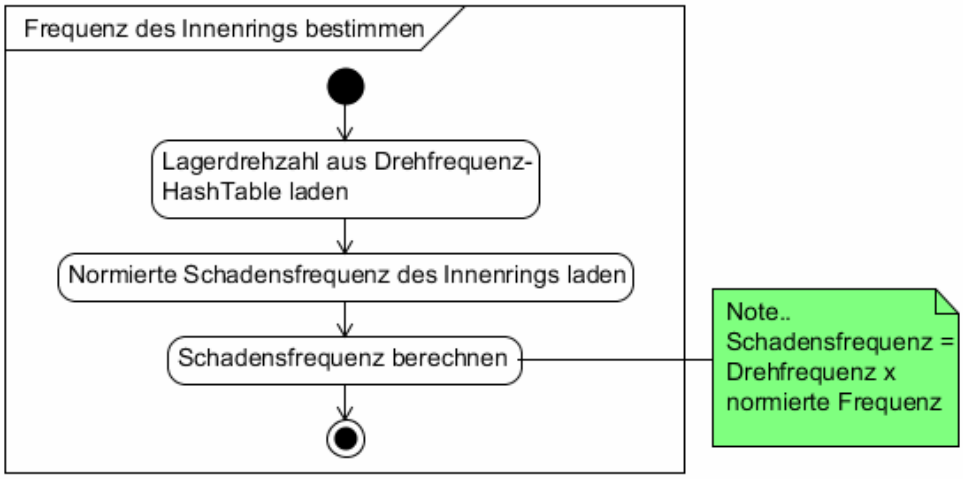

Abbildung 48: UML- Aktivitätsdiagramm „Frequenz des Innenrings bestimmen“

9.2.19.1.3 Berechnung der Zahneingriffsfrequenzen

Abbildung 49 zeigt den grundsätzlichen Ablauf zur Berechnung der Zahneingriffsfrequenzen der Zahnradpaare eines Stufengetriebes.

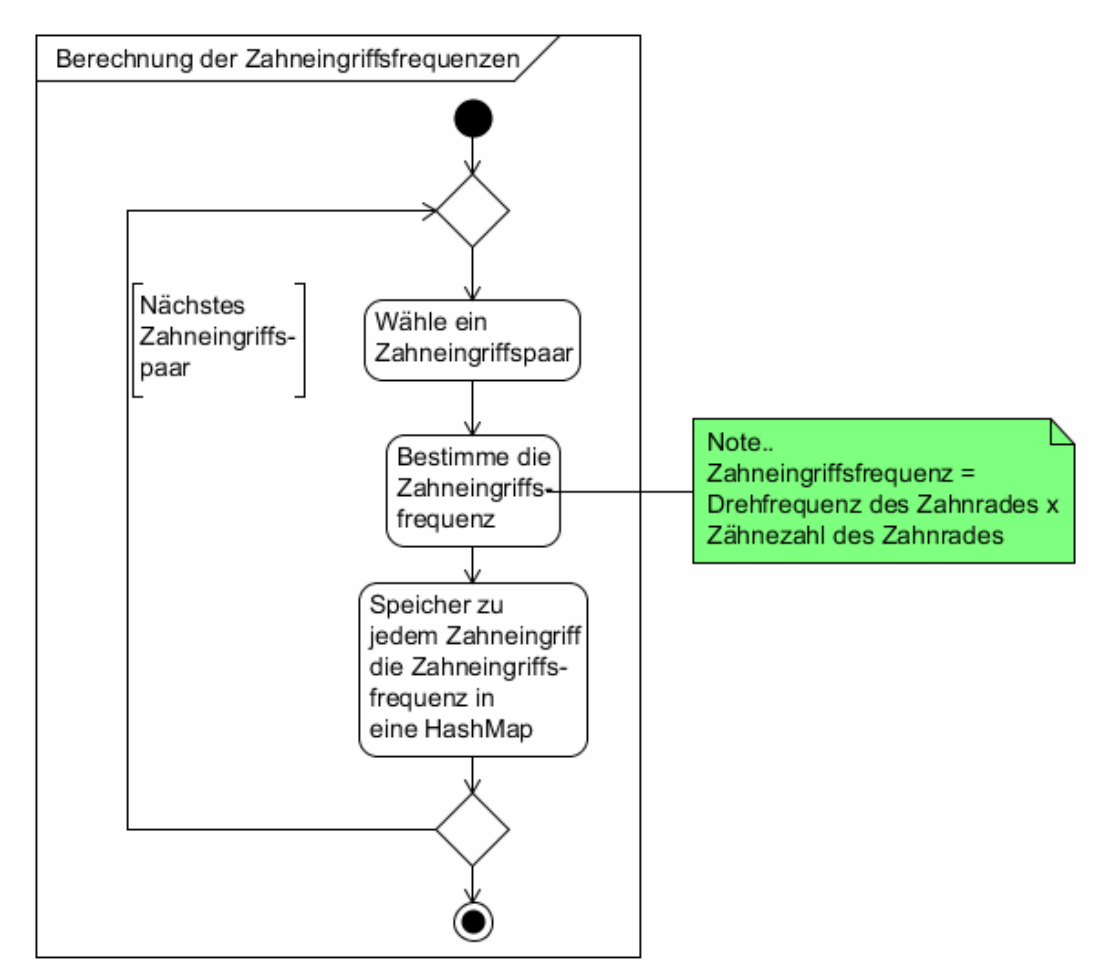

Abbildung 49: UML- Aktivitätsdiagramm „Berechnung der Zahneingriffsfrequenzen“

9.2.19.2 Berechnung speichern

Nachdem die Berechnung abgeschlossen ist, werden die Ergebnisse in Form eines XML-Dokumentes abgespeichert. Wegen der besonderen Bedeutung dieses Dokuments als Schnittstelle zwischen den beiden Programmen „GetriebeStudio“ und „SPECTIVE“ wurde ein entsprechendes Schema in XSD-Format ausgearbeitet.

9.2.20 Bilddatei erstellen/speichern (Task revidiert)

Ursprünglich sollte vom modellierten Getriebe eine Bilddatei erstellt werden können. Das erstellte Bild wäre für die spätere Verwendung im Tool „HealthMonitor“ eingesetzt worden. Stattdessen nutzt „HealthMonitor“ die mit „GetriebeStudio“ erstellten Geometriedaten zur Darstellung des Getriebemodells. Dies ermöglicht neben der Darstellung des Getriebemodells eine einfache Anzeige des Gesundheitszustandes der Komponenten (die Farbe der Komponente signalisiert deren Gesundheitszustand). Abbildung 50 zeigt einen Screenshot eines, mit dem „GetriebeStudio“, modellierten und im „HealthMonitor“ geöffneten Getriebes.

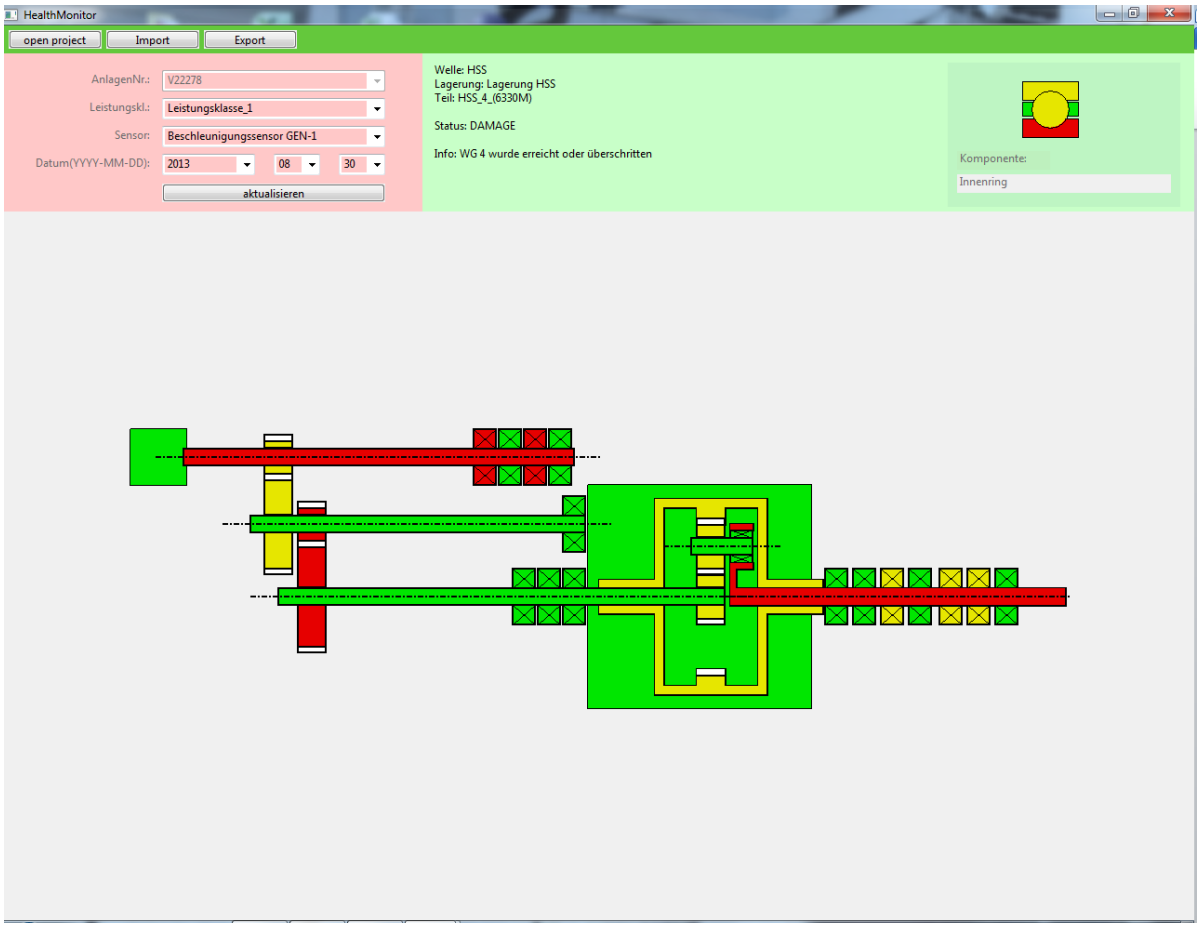

Abbildung 50: Screenshot aus „HealthMonitor“ (dargestelltes Getriebemodell mit „GetriebeStudio“ modelliert)

10 Implementierung

Graphical User Interface (GUI)

Für den Aufbau und die Gestaltung der Fenster wurde das Java-SWT-Softwarepaket verwendet. Dieses bietet ein umfangreiches Sortiment an Bausteinen wie zum Beispiel Fenster, Tabellen, Schalter, etc. und „Listenern“[7], durch welche Bausteinen, wie etwa Schalter, erst eine Funktion zugewiesen werden kann. AWT oder Swing sind wichtige Alternativen zu SWT und haben einige Vor- und Nachteile. Im Wesentlichen bietet SWT mehr Möglichkeiten zur Gestaltung (look and feel, Browser).

Beim Starten der Anwendung wird das Hauptfenster (siehe Abbildung 51) geöffnet. Dieses enthält ein Menü und eine Sammlung von Reitern. Über das File-Menü (siehe Abbildung 52) lassen sich wichtige Funktionen wie das Anlegen, Öffnen oder Speichern eines Projektes sowie das Speichern von Standardbaugruppen ausführen.

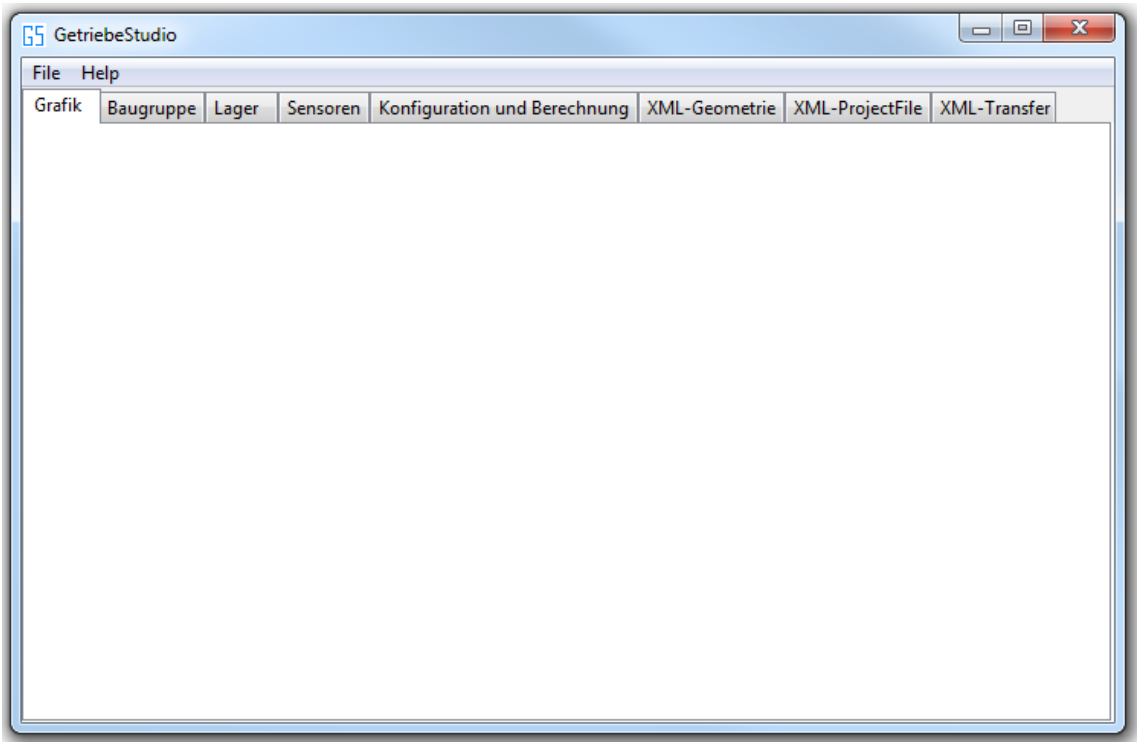

Abbildung 51: GetriebeStudio GUI

[7] Listener dienen der Ereignissteuerung eines Programmes

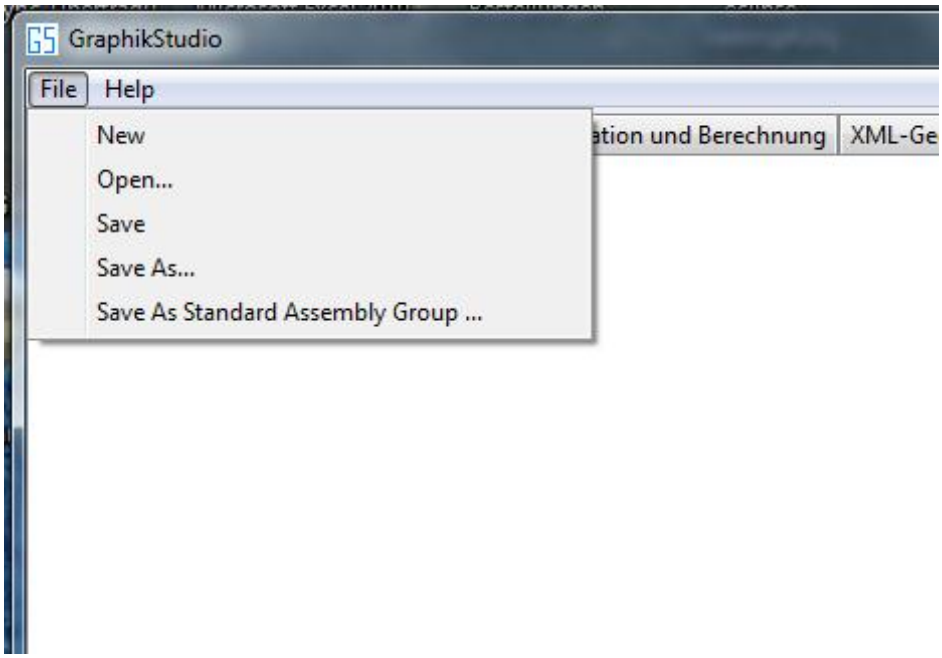

Abbildung 52: File-Menü

Aufbau von Getriebemodellen

Zum Gestalten eines Getriebes bzw. einer Baugruppe wird ein neues Projekt angelegt oder ein bestehendes ausgewählt. Die Bauteile und Baugruppen können über ein zusätzliches Fenster (siehe Abbildung 53), das mittels Rechtsklick auf die erste Reiterkarte geöffnet wird, ausgewählt werden.

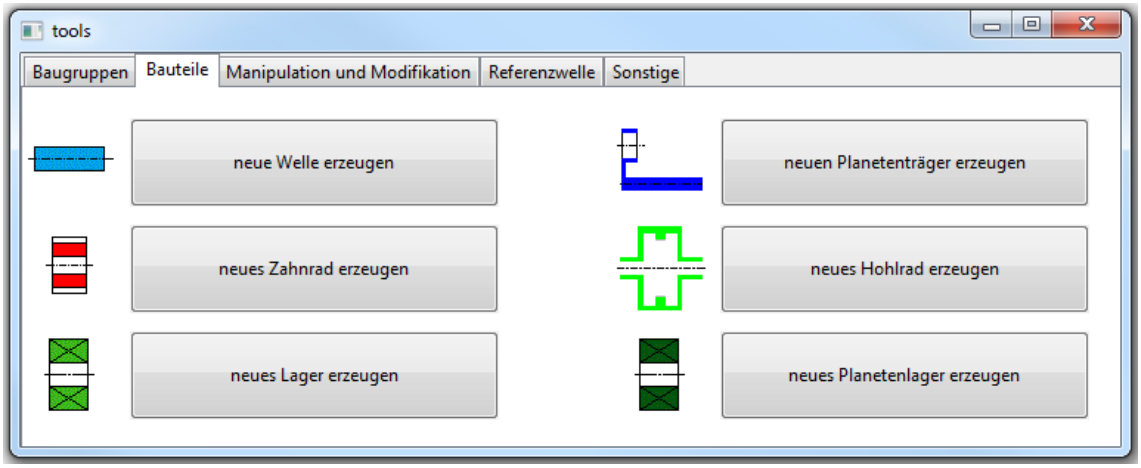

Abbildung 53: „tools“-Fenster (Reiter Bauteile)

Das Fenster „tools“ ist ähnlich aufgebaut wie das Hauptfenster. Es enthält eine Reihe von Reiterkarten in denen die Tools angeordnet sind. Möchte man zum Beispiel eine Welle hinzufügen, klickt man auf die Reiterkarte „Bauteile“ und den Button „neue Welle erzeugen“.

Neben dem Hinzufügen von Bauteilen enthält „tools“ eine Reihe weiterer Funktionen (siehe Abbildung 54). Zu diesen zählen: Laden von Baugruppen, Spezifizieren der Bauteile und Bau-

gruppen, Löschen von Baugruppen, und viele weitere nützliche Features. Neue Funktionen können einfach mittels eines neuen Items im jeweiligen Reiter hinzugefügt werden. Das Hinzufügen neuer Reiterkarten ist ebenfalls möglich. Ein und dieselbe Funktion kann auf mehreren Reitern zur Verfügung gestellt werden. Dadurch muss man nicht immer den Reiter wechseln. So wurde u.a. die Funktion „Baugruppe/Bauteil modifizieren“ in die Reiter „Baugruppen“ und „Manipulation und Modifikation“ integriert.

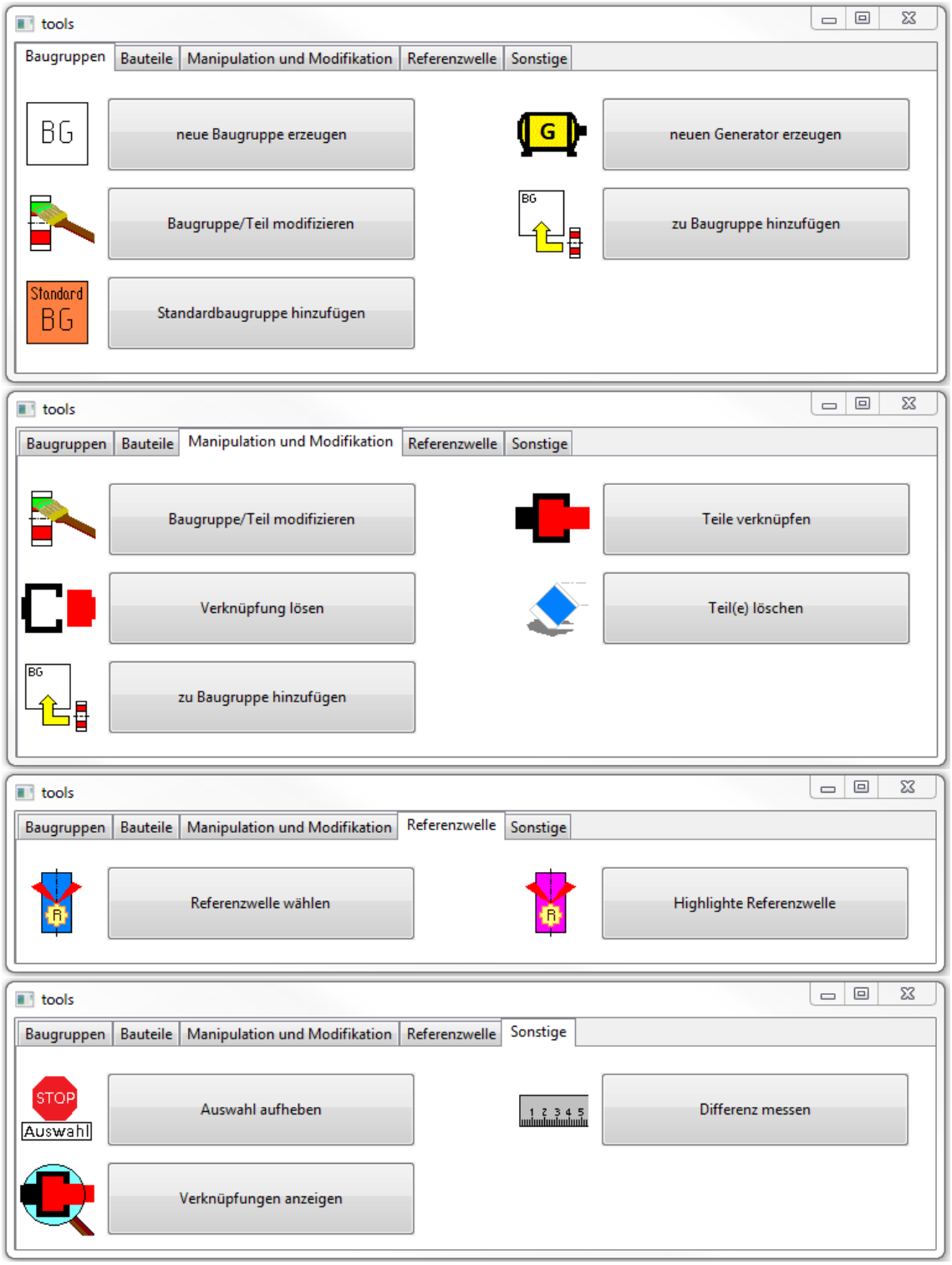

Abbildung 54: „tools“-Fenster (weitere Reiter)

Verknüpfen von Bauteilen mit anderen Bauteilen

Das Zusammenfügen der zuvor erstellten Bauteile erfolgt durch Verknüpfungen. Diese werden durch, den Bauteilobjekten zugeordnete, Hashtabellen realisiert. Der zu verknüpfende Teil wird in eine dieser Tabellen eingetragen. Durch das Verknüpfen der Bauteile konnten zusätzliche Funktionen, die den Umgang mit dem Programm erleichtern sollen, wie das Ausrichten der Lager bzw. der Zahnräder auf die zugehörige Welle realisiert werden. Der Benutzer kann zwei Bauteile wie folgt miteinander verknüpfen:

1. Selektiere einen Bauteil (oder eine Baugruppe).
2. Selektiere einen zweiten Bauteil (oder eine Baugruppe).
3. „tools“-Fenster (siehe Abbildung 54) mit Maus-Rechtsklick auf die Oberfläche öffnen.
4. Reiter „Manipulation und Modifikation“ wählen.
5. Den Button mit Beschriftung „Teil verknüpfen“ anklicken.
6. Durch Betätigung des zugehörigen Buttons, wird im ersten Objekt die Verknüpfung zum zweiten Objekt und umgekehrt in die zugehörige Hashtabelle eingetragen.

Die Abbildungen 55 bis 57 zeigen am Beispiel zweier Bauteile den zuvor beschriebenen Ablauf.

Voraussetzung für eine erfolgreiche Verknüpfung ist jedoch, dass die Paarung dieser Teile zulässig ist. So können u.a. Lagerobjekte etwa mit Wellen, jedoch nicht mit anderen Lagern verknüpft werden. Die Umsetzung erfolgte in einen ersten Schritt im Programmcode. Als Alternative würde sich unter anderem der Einsatz einer Rule-Engine wie etwa „Drools“ anbieten. Die Regeln (Rules) und die Datenbasis (Facts) können hierbei getrennt voneinander entwickelt und gespeichert werden. Ein wesentlicher Vorteil dieser Lösung wäre die bessere Wartbarkeit des Codes.

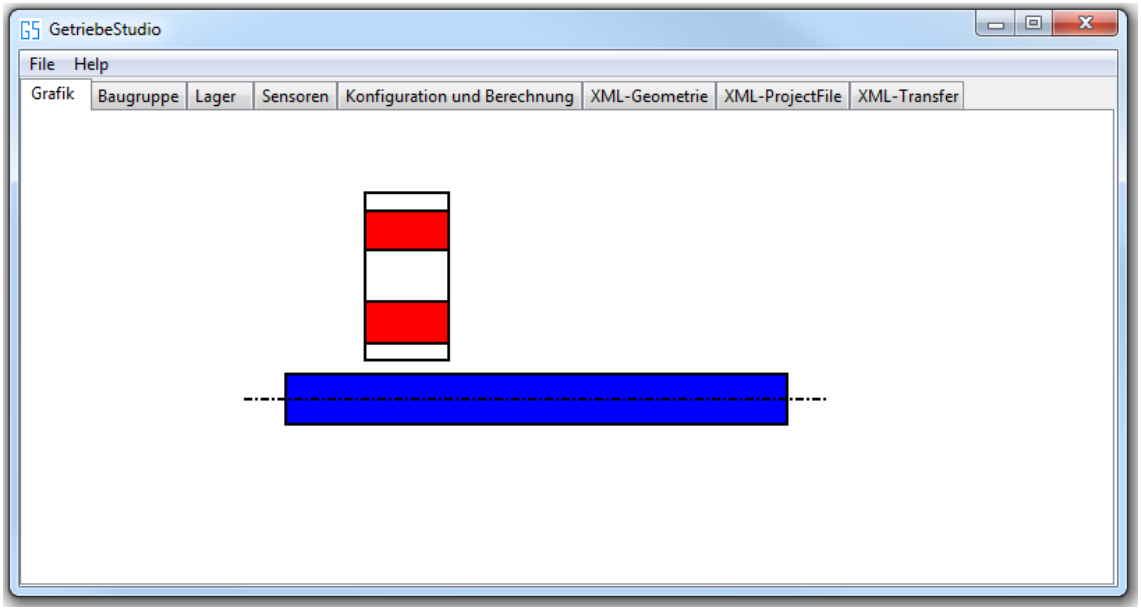

Abbildung 55: Zahnrad (rot) und Welle (blau) sind dem Projekt hinzugefügt

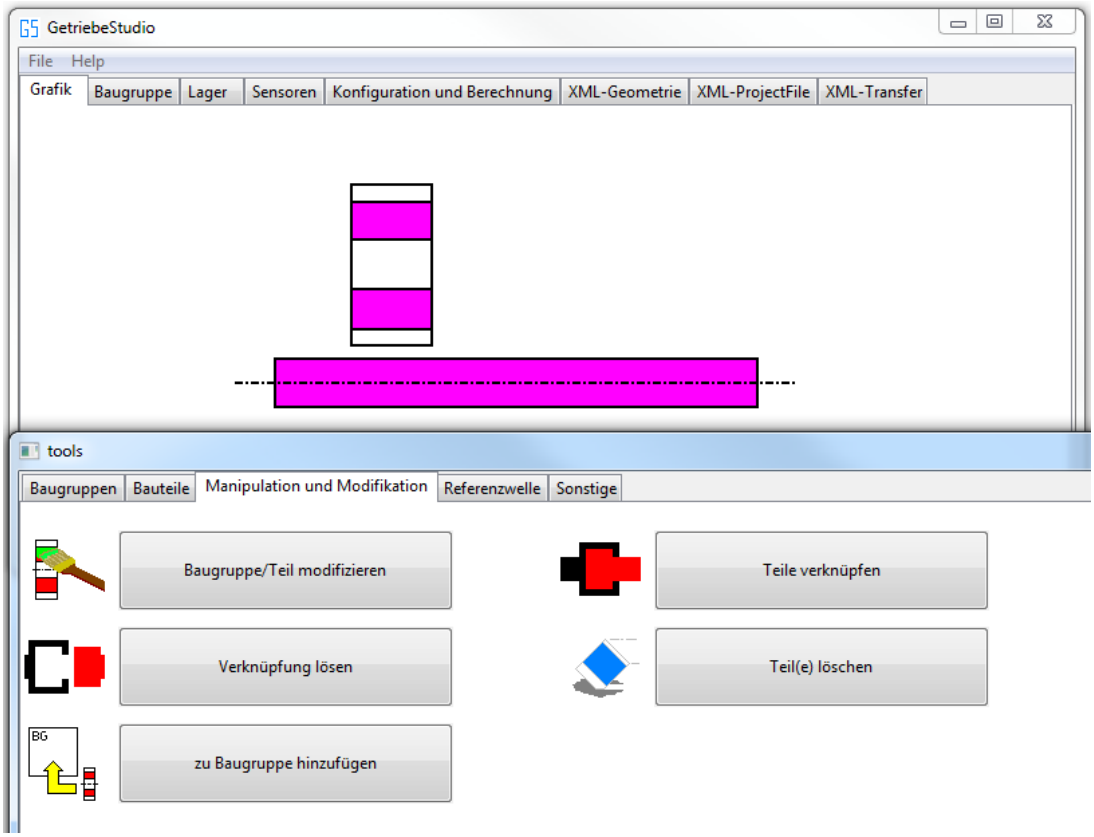

Abbildung 56: Selektierte Bauteile (Zahnrad und Welle werden in Magenta hervorgehoben) und „tools“-Fenster

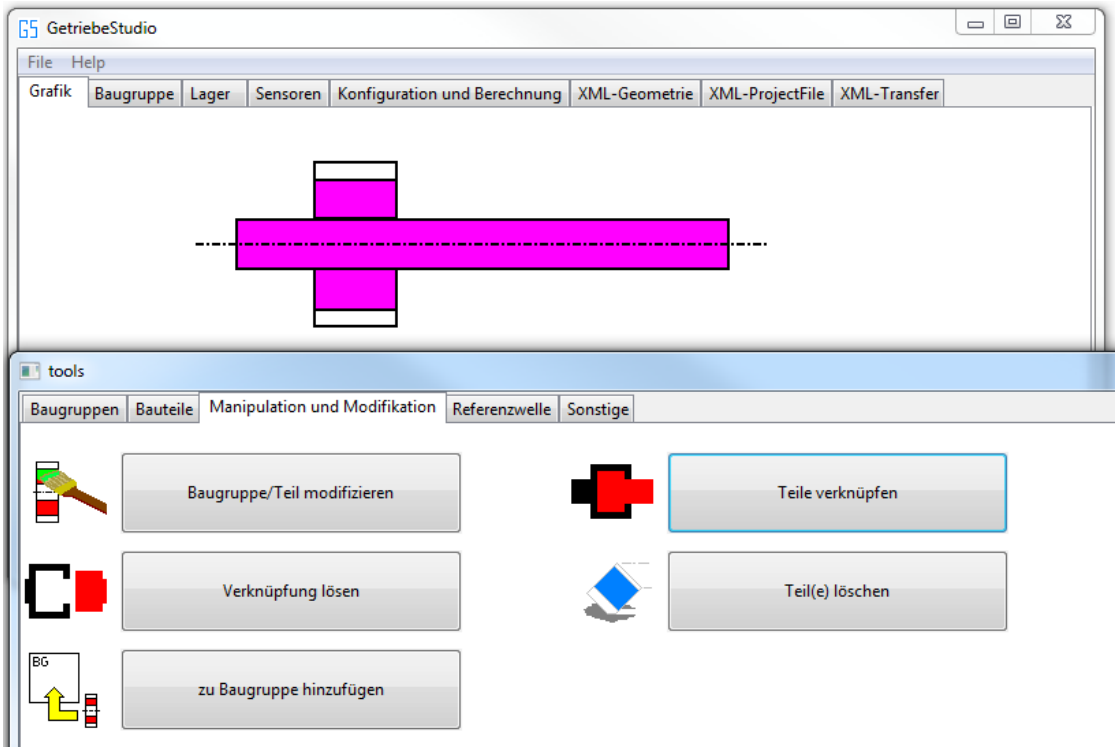

Abbildung 57: Verknüpfung der beiden Bauteile (Zahnrad wird auf Welle zentriert)

Verknüpfen von Bauteilen mit Baugruppen

Das Verknüpfen von Bauteilen mit einer Baugruppe erfolgt analog zum Verknüpfen von Bauteilen mit anderen Bauteilen. Die Baugruppe und die zu verknüpfenden Bauteile werden zunächst mittels Klick auf die linke Maustaste selektiert. Danach wird das „tools“-Fenster mit einem Klick auf die rechte Maustaste geöffnet. Durch das Anklicken des Buttons „zu Baugruppe hinzufügen“ (siehe Abbildung 57) startet der eigentliche Verknüpfungsvorgang. Die selektierten Teile werden der „Stückliste“ (Hashmap) der Komponenten der Baugruppe hinzugefügt.

Selektion und Deselektion von Bauteilen/Baugruppen

Mittels Selektion eines Bauteiles oder einer Baugruppe kann dieses bzw. können diese manipuliert bzw. modifiziert werden. Die gewählten Bauteile und -gruppen werden in einer Hashmap gespeichert, so dass das Programm auf diese schnell und gezielt zugreifen kann. Alle selektierten Komponenten werden in Magenta hervorgehoben, wodurch man diese von den nicht selektierten Bauteilen unterscheiden kann. Möchte man den Teil wieder deselektieren, so klickt man erneut auf die dargestellte Komponente, wodurch das Teil bzw. die Baugruppe aus der Hashmap der selektierten Komponenten entfernt wird. Die dargestellte Komponente wird sodann in ihrer spezifizierten Farbe angezeigt. Durch Doppelklick neben die modellierten Teile und Baugruppen, werden alle bisher selektierten Komponenten deselektiert.

Selektion der Teile einer Baugruppe

Da Baugruppen meist aus einer größeren Anzahl von Teilen bestehen, ist es für die Manipulation der Baugruppe und deren Teile zweckmäßig diese möglichst schnell selektieren und deselektieren zu können.

Zur Selektion der Baugruppe und aller mit der Baugruppe verknüpften Teile wurde folgender Ablauf umgesetzt:

1. Wechsel, falls noch nicht erfolgt, in den ersten Reiter (siehe Abbildung 58).
2. Selektieren der Baugruppe im Grafikbereich (erster Reiter) mit einem Klick auf die linke Maustaste. Die Baugruppe, ohne die verknüpften Teile, wird selektiert und dementsprechend farblich hervorgehoben (siehe Abbildung 59).
3. Durch wiederholtes Anklicken der Baugruppe werden die Baugruppe und die mit ihr verknüpften Teile selektiert (siehe Abbildung 60). Bei leeren Baugruppen (Baugruppen ohne verknüpfte Teile) wird dieser Punkt übersprungen. Das Selektieren bzw. Deselektieren einzelner Bauteile ist weiterhin jederzeit möglich.
4. Wird die Baugruppe ein weiteres Mal angeklickt, so werden die Baugruppe und die mit ihr verknüpften Teile deselektiert (siehe Abbildung 61).

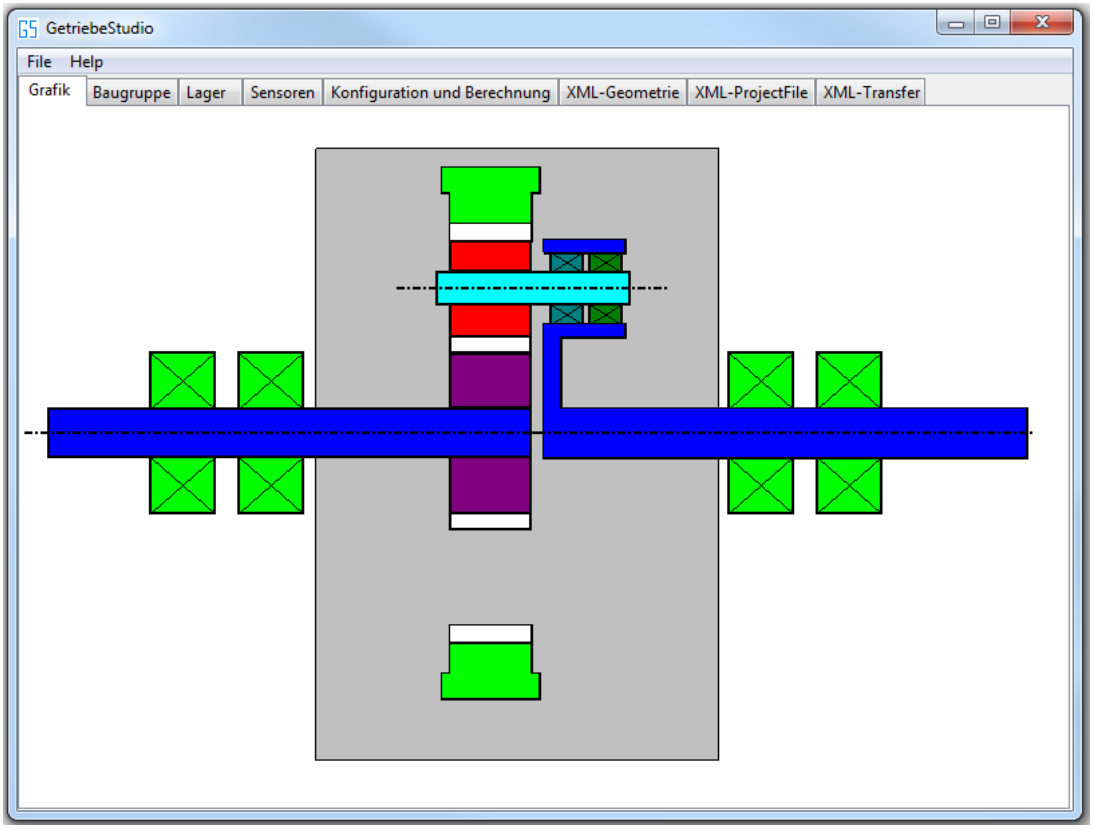

Abbildung 58: Selektion einer Baugruppe 1 (Baugruppe, dargestellt als graues Rechteck, und zugehörige Bauteile wurden miteinander verknüpft)

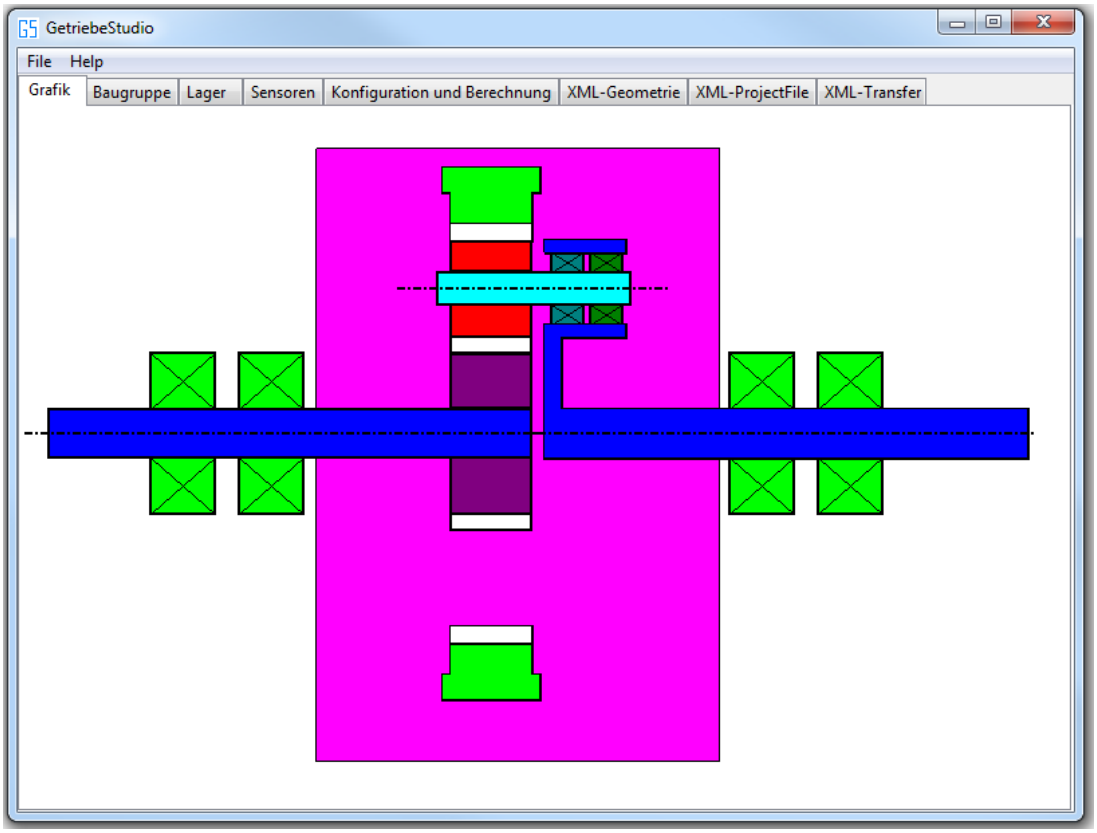

Abbildung 59: Selektion einer Baugruppe 2 (Baugruppe in Magenta hervorgehoben)

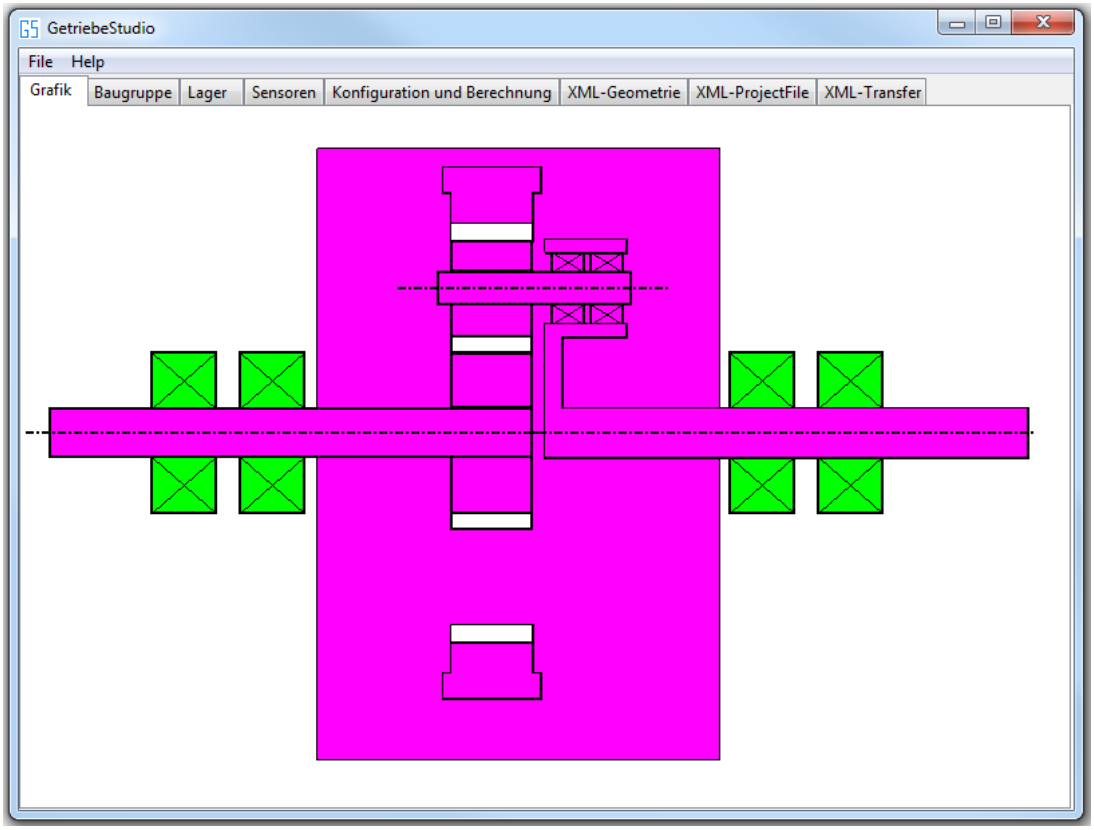

Abbildung 60: Selektion einer Baugruppe 3 (Baugruppe inklusive verknüpfte Bauteile in Magenta hervorgehoben)

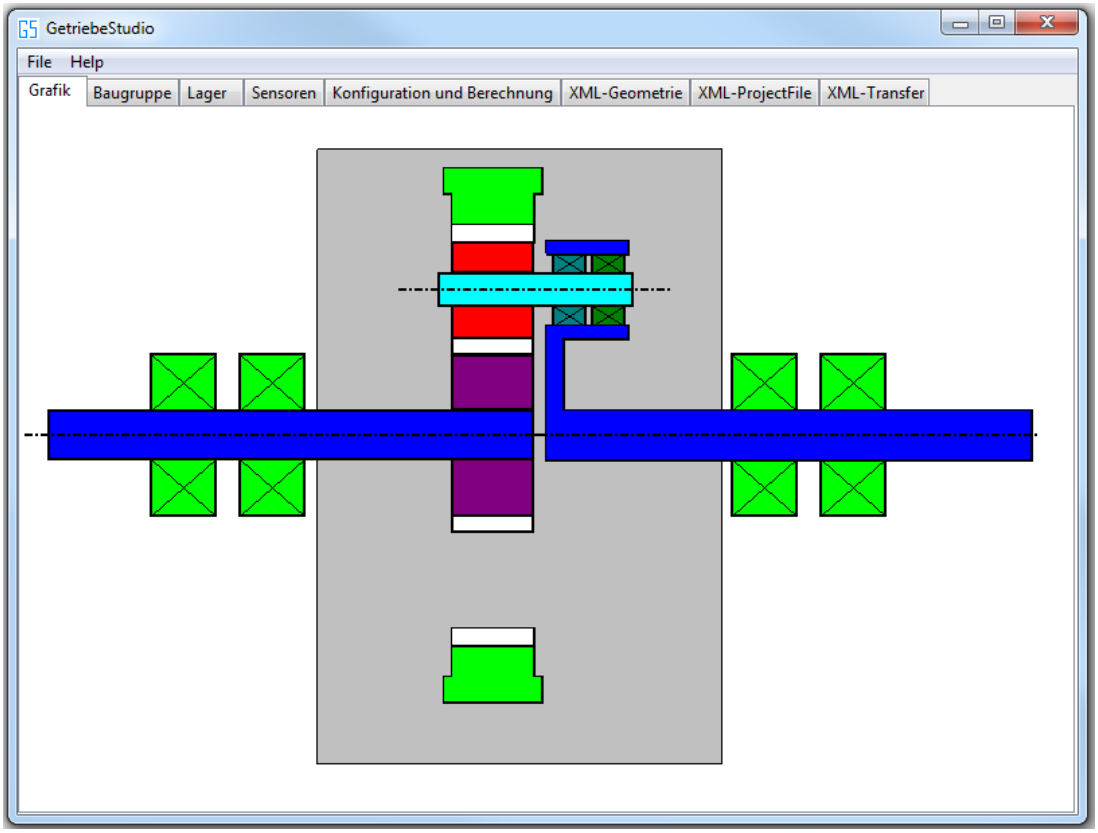

Abbildung 61: Selektion einer Baugruppe 4 (Baugruppe und verknüpfte Teile deselektiert)

Kontrolle gesetzter Verknüpfungen

Da Bedienfehler bei der Verknüpfung von Bauteilen nicht auszuschließen sind, muss eine Möglichkeit bestehen, die vorhandenen Verknüpfungen in geeigneter Art und Weise dem Benutzer des Programmes anzuzeigen.

Möchte man die Verknüpfungen eines Teiles zu einem weiteren Bauteil oder Baugruppe anzeigen lassen, so kann man wie folgt vorgehen:

1. Man wechselt in den Graphik-Bereich.
2. Durch Doppelklick auf den zu untersuchenden Teil werden die verknüpften Bauteile und Baugruppen in einem kleinem Textfeld in der Nähe des Bauteils angezeigt (siehe Abbildung 62).
3. Durch erneutes Doppelklicken auf einen anderen Teil werden die Verknüpfungen dieses Teiles angezeigt.
4. Die Textanzeige verschwindet durch Doppelklicken in einen Bereich der Modelliroberfläche in dem sich kein Teil befindet.

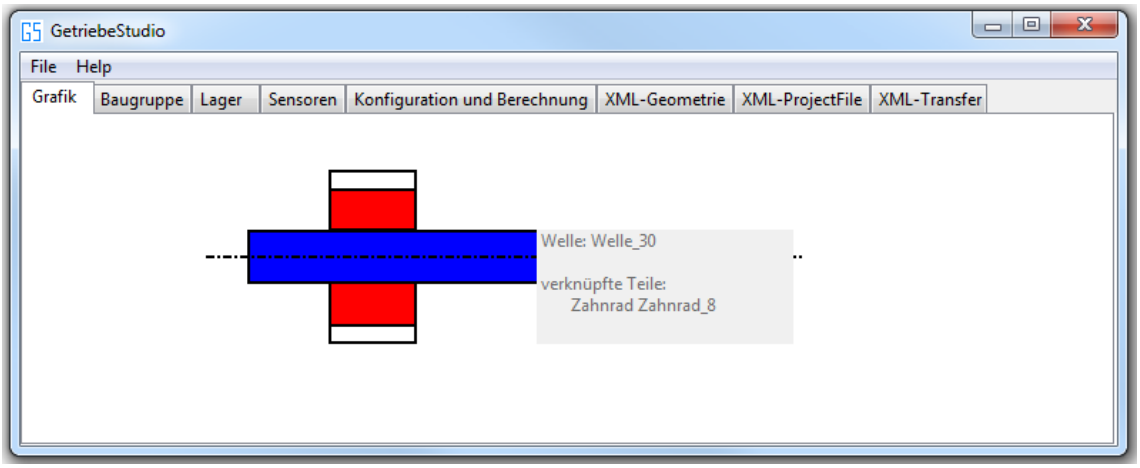

Abbildung 62: Anzeige der Welle und all ihrer (möglichen) Verknüpfungen

Alternativ kann auch die Funktion „Verknüpfungen anzeigen“ aus dem Tool-Baukasten (siehe Reiter „Sonstige“, in Abbildung 54) genutzt werden. Diese erstellt eine Übersichtstabelle aller Teile und Baugruppen und deren Verknüpfungen (siehe Abbildung 63). Durch vorherige Selektion eines Bauteils oder einer Baugruppe werden in einem eigenen Reiter die objektspezifischen Verknüpfungen angezeigt.

Verknüpfungen entfernen

Möchte man eine Verknüpfung zwischen zwei Bauteilen lösen oder die Verknüpfung zu einer Baugruppe entfernen, so sollte man wie folgt vorgehen:

1. Selektion (Linksklick) der Bauteile/Baugruppen deren Verknüpfungen zueinander gelöst werden sollen.
2. „tools“-Fenster öffnen (Rechtsklick).
3. Im sich öffnenden „tools“-Fenster die Reiterkarte „Sonstige“ selektieren.
4. Den Button „Verknüpfungen anzeigen“ wählen.
5. Im Reiter „Selektierte Teile“ können mit Hilfe des Buttons „Verknüpfung lösen“ die zugehörigen Verknüpfungen gelöst werden (diese Funktionalität wurde auch in den Reitern Bauteile und Baugruppen integriert).
6. Die Verknüpfungen werden paarweise entfernt.

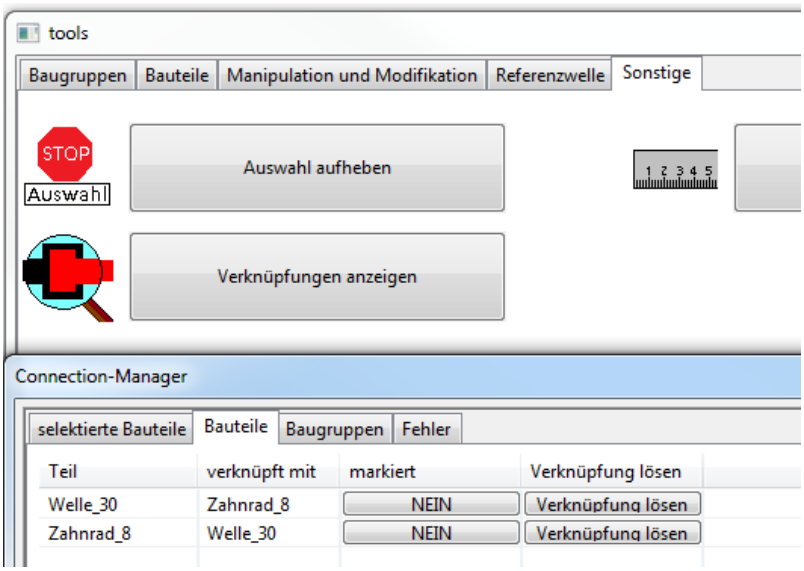

Abbildung 63: Anzeige der Verknüpfungen aller modellierten Bauteile

Möchte man hingegen alle Verknüpfungen eines Bauteiles entfernen, geht man wie folgt vor:

1. Selektion eines oder mehrerer Bauteile/Baugruppen deren Verknüpfungen gelöst werden sollen.
2. „tools“ öffnen (Rechtsklick).
3. Im „tools“-Fenster „Verknüpfungen lösen“ wählen.

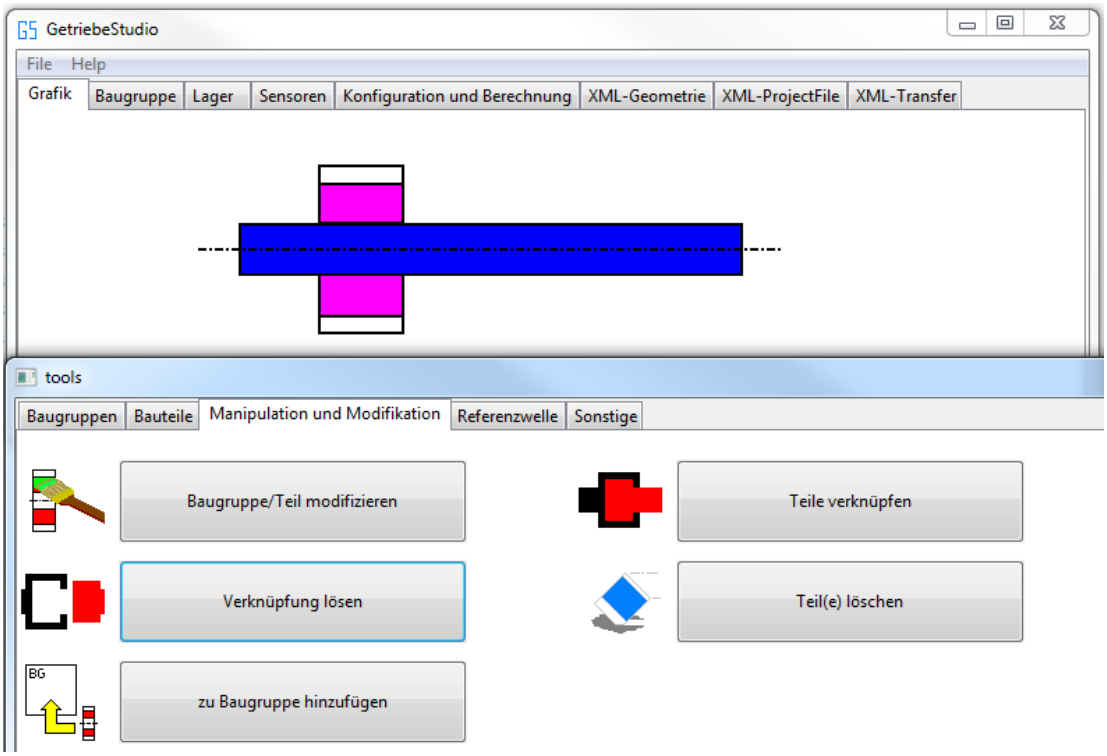

Abbildung 64: Lösen aller Verknüpfungen eines Bauteiles

Löschen von Bauteilen/Baugruppen

Für das Löschen von Baugruppen und Bauteilen muss der Benutzer wie folgt vorgehen:

1. Selektion eines oder mehrerer Bauteile/Baugruppen die gelöscht werden sollen.
2. „tools“ öffnen (Rechtsklick).
3. Den Reiter „Manipulation und Modifikation“ selektieren.
4. Den Button „Bauteil/Baugruppe löschen“ wählen.

Folgende Veränderungen an den Objekten werden dabei vorgenommen:

1. Verknüpfungen zu anderen Bauteilen und/oder zur Baugruppe werden gelöst.
2. Entfernung der Objekte aus der Liste der projektbezogenen Bauteile und Baugruppen (siehe Klasse Projekt; nur jene Teile und Baugruppen, die dem Projekt bekannt sind, werden angezeigt bzw. können gespeichert werden).
3. Aktualisieren der Darstellung in allen Reitern.

Anzeige von Teilen/Baugruppen ohne Verknüpfungen

Manchmal kann es vorkommen, dass bei der Modellierung vergessen wurde, eine Verknüpfung eines Bauteils zu einem Weiteren zu setzten. Ist dieser „lose“, hat also keine Verknüpfung zu anderen Teilen oder einer Baugruppe, kann dies wie folgt angezeigt werden (siehe auch Abbildung 65):

1. „tools“ öffnen (Rechtsklick).
2. In den Reiter „Sonstige“ wechseln.
3. Den Button „Bauteil/Baugruppe löschen“ anklicken.
4. In den Reiter Fehlermeldungen wechseln.

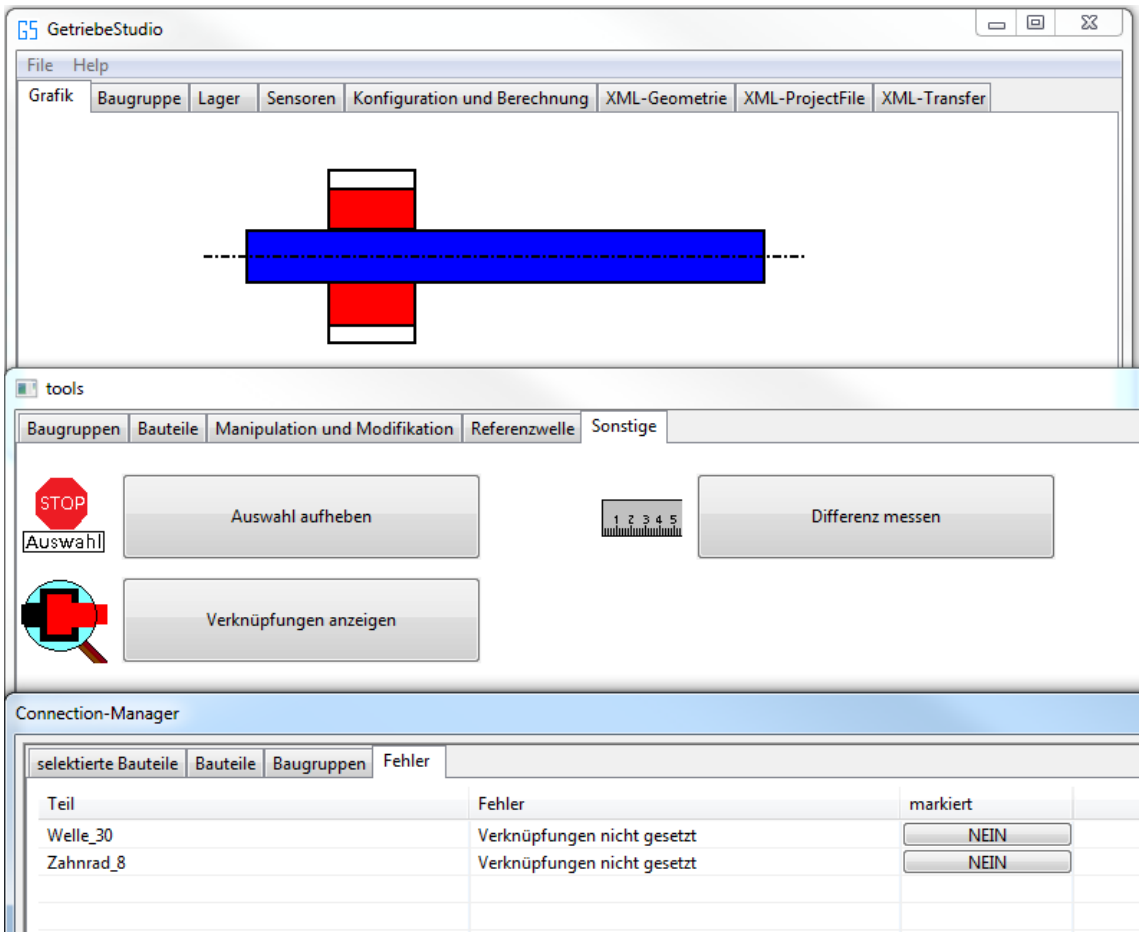

Abbildung 65: Bauteile ohne Verknüpfungen

Umgang mit fehlerhaften Verknüpfungen

Das Setzen von fehlerhaften Verknüpfungen wird vom Programm nach Möglichkeit vermieden. So können zwei Teile nur einmal miteinander verknüpft sein, da die Verknüpfungen in Hashtabellen eingetragen werden. Zudem gibt es bauteil- bzw. baugruppenspezifische Regeln, die eingehalten werden müssen. Unter anderem darf ein Lager nur mit einer weiteren Komponen-

te (Welle, Planetenträger oder Hohlrad) sowie einer Baugruppe verknüpft sein. Planetenlager dürfen zudem mit Zahnrädern verknüpft werden.

Manipulation von Teilen und Baugruppen

Für das Verschieben von Bauteilen und Baugruppen müssen diese zunächst selektiert werden. Die selektierten Komponenten können mittels der Pfeiltasten des Keyboards oder durch Gedrückthalten der linken Maustaste und Ziehen der Maus an die gewünschte Position verschoben werden.

Spezifikation von Baugruppen

Nachdem die Baugruppen hinzugefügt und die nötigen Bauteile mit der Baugruppe verknüpft wurden, erfolgt die Spezifikation der Baugruppen.

Durch Spezifizieren von Baugruppen ist es möglich, im Programm zu definieren um welche Art von Baugruppe es sich handelt, welche Bauteile als Ein- bzw. Ausgänge definiert sind und welche Funktion bestimmte Bauteile in den Baugruppen haben. Dies ermöglicht eine spätere Berechnung u.a. der Drehzahlen der Bauteile.

Die Spezifikation der Baugruppen erfolgt über den dafür vorgesehenen Reiter „Baugruppe“ (siehe Abbildung 66).

Eine im Reiter enthaltene Tabelle wird mit den baugruppenspezifischen Daten versorgt. Die zugeordneten Bauteile werden jeweils unterhalb der Baugruppe dargestellt.

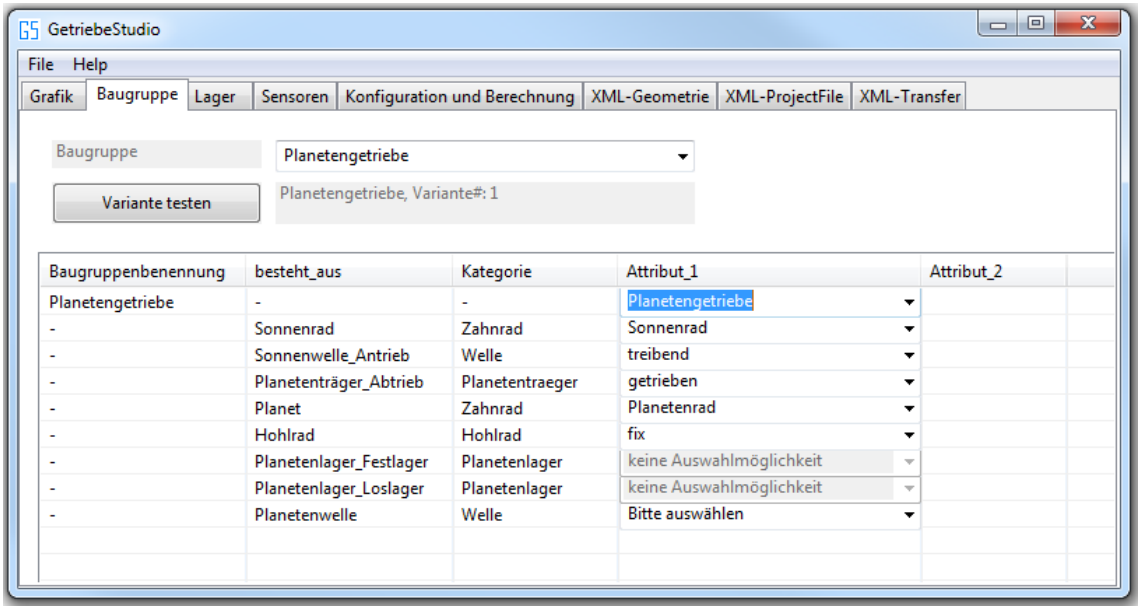

Abbildung 66: Spezifikation der Baugruppen

Zur Überprüfung der Plausibilität der Baugruppe (handelt es sich um eine zulässige Spezifikation) gibt es zusätzlich die Möglichkeit anhand eines Textfeldes oberhalb der Tabelle, diese auf Korrektheit zu testen. Hierzu gibt man die genaue Bezeichnung in das Textfeld ein und drückt den zugehörigen Button zur Bestätigung der Eingabe. Ist die Baugruppenspezifikation bekannt, wird dies unterhalb des Textfeldes angegeben. Handelt es sich um eine unbekannte Spezifikation wird dies ebenfalls unterhalb des Textfeldes angezeigt.

Lagerdatenbank

Die Lagerdaten werden in Form eines erstellten XML-Files abgespeichert. Der erstellte „Lagerkatalog“ enthält alle aktuell verfügbaren Wälzlager, deren Hersteller, zugehörende Lagerart (z.B. Rillenkugellager) und die normierten Schadensfrequenzen des Innen- und Außenrings, des Käfigs und der Wälzkörper. Die korrekte normierte Schadensfrequenz kann auf rechnerischem Wege manuell bestimmt werden. Lagerhersteller bieten auf ihren Webseiten die Möglichkeit, für die von ihnen gefertigten Lager, die Schadensfrequenzen zu bestimmen.

Möchte man einen neuen Lagerdatensatz dem Instanz-Dokument hinzufügen, so kann man dies über einen passenden Editor, oder komfortabler, über die implementierte Eingabemaske (wird durch Drücken des Buttons „neues Lager anlegen“ im Reiter „Lager“ geöffnet, siehe Abbildungen 67 und 68) durchführen. Diese bietet zudem den Vorteil, die Datenbank nach bestehenden Einträgen zu durchsuchen. Wird ein entsprechender Eintrag gefunden, so wird das Dokument nicht verändert. Möchte man diesen Datensatz trotzdem abspeichern, so müsste man ihn unter einen anderen Namen abspeichern. Das Überschreiben der Datensätze wird vom Programm derzeit nicht unterstützt.

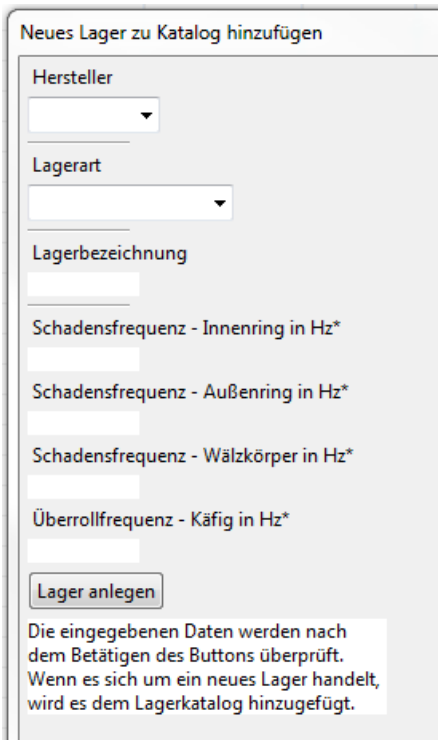

Abbildung 67: Eingabemaske zum Anlegen eines neuen Lagerdatensatzes

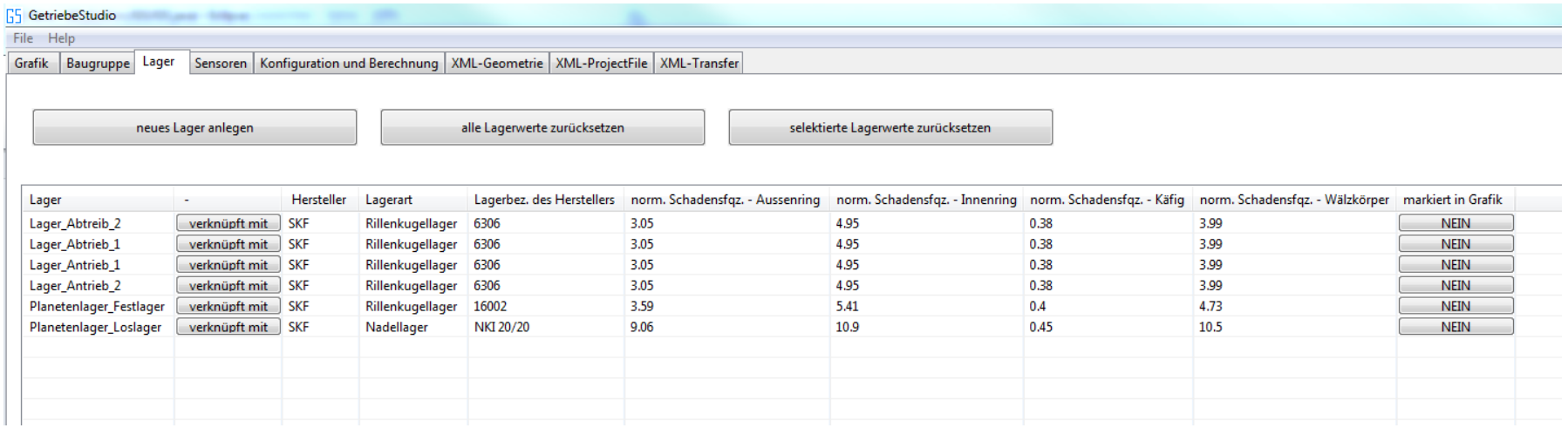

Abbildung 68: Reiter "Lager"

Verknüpfen von Lagern mit Lagerdaten aus der Datenbank

Dem Projekt hinzugefügte modellierte Lager müssen für eine erfolgreiche Berechnung der Schadensfrequenzen mit einem Lager der Lagerdatenbank verknüpft werden. Dazu wechselt man zunächst in den Reiter „Lager“ (siehe Abbildung 68). Dieser zeigt alle dem Projekt hinzugefügten und graphisch angezeigten Lager in einer Tabelle an. Die Tabelle bietet die Möglichkeit per Rechts-Klick auf den Button „verknüpfen mit“ dem jeweiligen modellierten Lager ein Lager aus der Lagerdatenbank zuzuordnen. Bei Betätigung eines dieser Buttons öffnet sich ein Fenster („Loogle - die Lagersuchmaschine“, siehe Abbildung 69). Durch Angabe der genauen Bezeichnung, beziehungsweise eines Teiles der Bezeichnung, werden alle passenden Einträge in einer Tabelle ausgegeben. Hierzu wird mittels Parser im Lagerkatalog nach passenden Einträgen gesucht. Hat man den gewünschten Eintrag gefunden, kann dieser per Klick auf den, der Zeile zugeordneten, Button übernommen werden (siehe Abbildung 70).

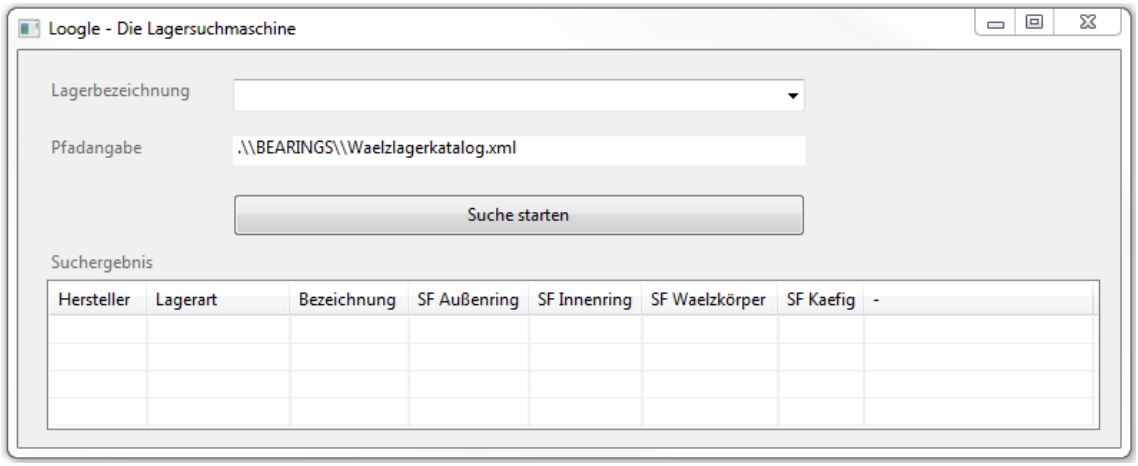

Abbildung 69: Loogle - Die Lagersuchmaschine

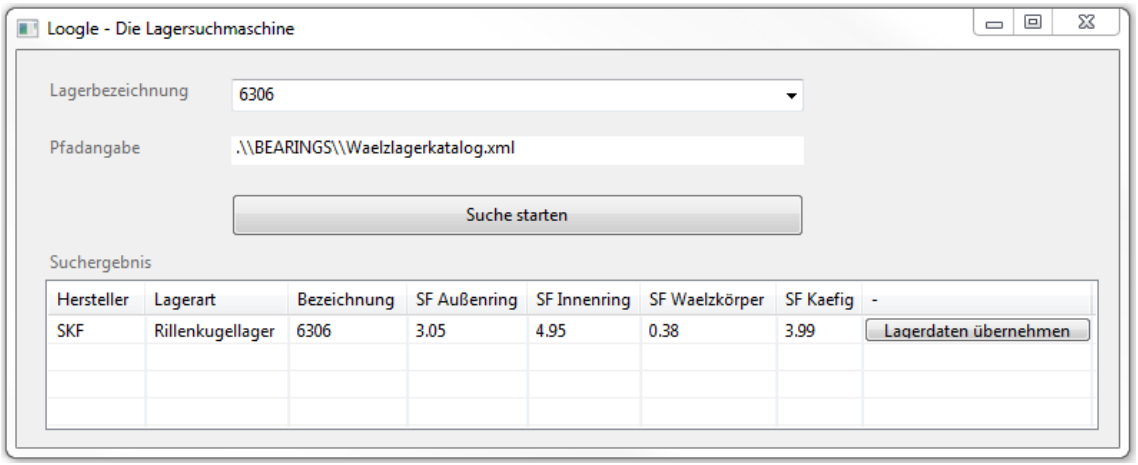

Abbildung 70: Suchergebnis/Zuweisung der Lagerdaten zu einem modellierten Lager

Modifikation von Bauteilen und Baugruppen

Die Modifikation der Bauteile und Baugruppen erfolgt nach dem folgenden Schema:

1. Selektion des zu modifizierenden Bauteiles bzw. der Baugruppe (siehe Abbildung 71).
2. Öffnen des „tools“-Fensters (Rechtsklick auf Oberfläche des ersten Reiters, siehe Abbildungen 53 und 54).
3. Im „tools“-Fenster den Reiter „Manipulation und Modifikation“ auswählen (siehe Abbildung 71).
4. Button „Baugruppe/Teil modifizieren“ anklicken.
5. Im sich öffnenden Fenster nötige Modifikationen durchführen[8] und per Klick auf den Bestätigungsbutton Vorgang abschließen[9] (siehe Abbildungen 72 bis 75).

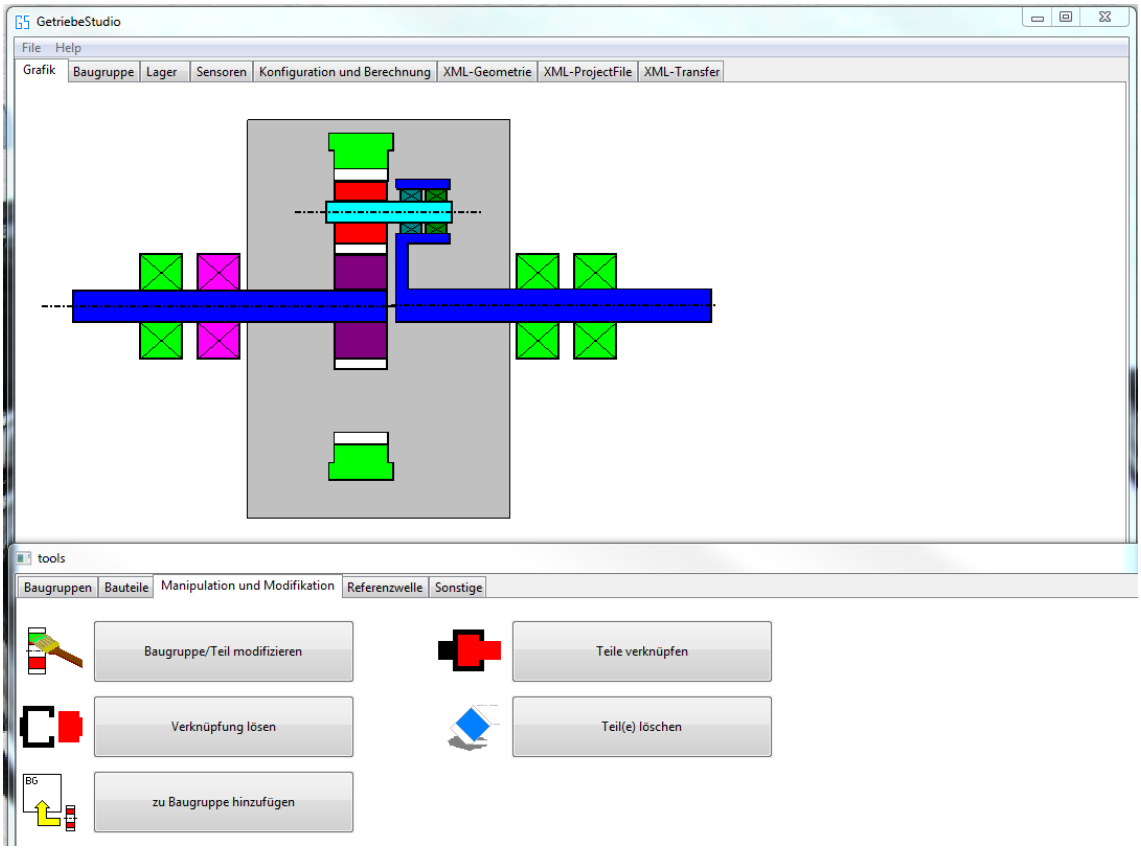

Abbildung 71: Modifikation eines Bauteiles - Selektion eines Lagers, Auswahl der Funktion "Baugruppe/Teil modifizieren"

[8] Das Programm überprüft die Korrektheit der Eingaben. Korrekte Eingaben werden grün, Fehlerhafte hingegen rot hinterlegt

[9] Die Bestätigung der Änderungen ist nur möglich, wenn die Eingaben korrekt sind. Andernfalls ist der Bestätigungsbutten nicht selektierbar

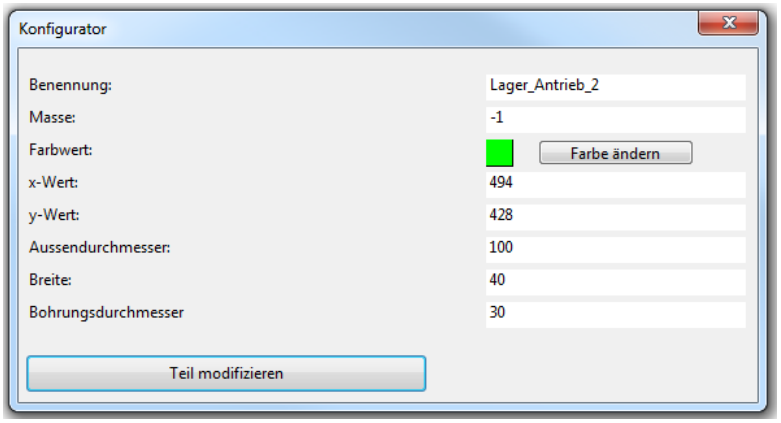

Abbildung 72: Fenster zum Modifizieren der Eigenschaften der Baugruppe bzw. des Bauteils

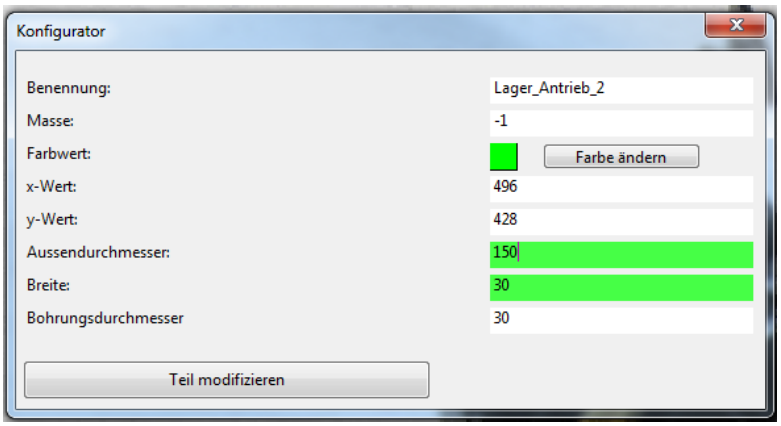

Abbildung 73: Änderung des Außendurchmessers und der Breite des ausgewählten Lagers (zulässige Änderungen werden grün hervorgehoben, unzulässige Änderungen werden rot hinterlegt)

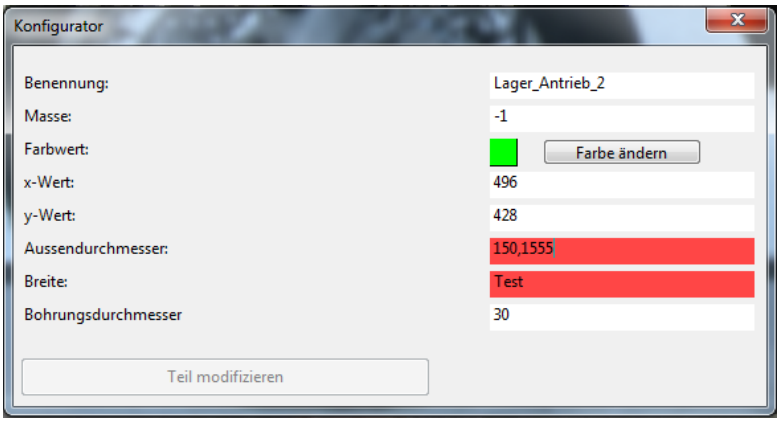

Abbildung 74: Unzulässiger Modifikationsversuch (Button "Teil modifizieren" nicht selektierbar)

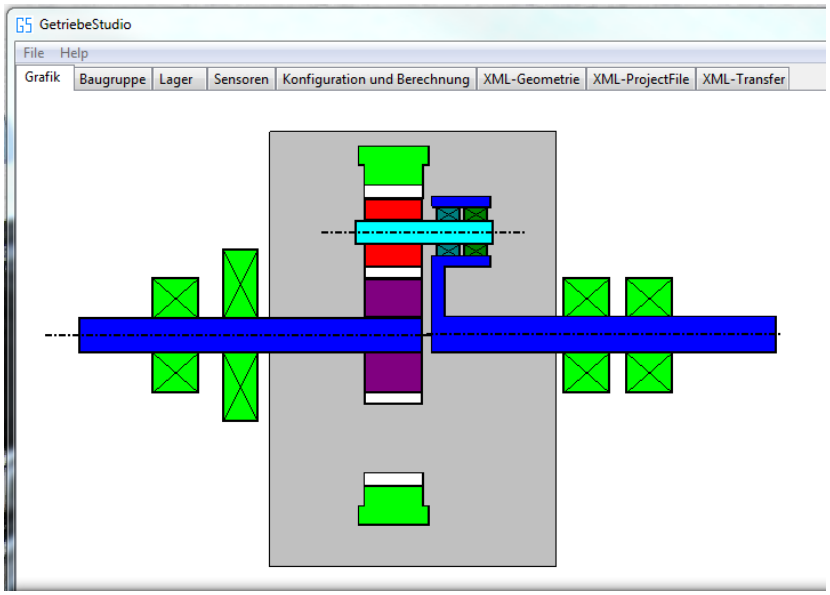

Abbildung 75: Getriebe nach erfolgter Modifikation

Farbauswahl

Beim Erzeugen neuer Teile und Baugruppen ist ein Farbwert bereits vorgegeben. Die Farbe soll nun durch Selektion einer Farbe aus einer Palette geändert werden können. Dazu wurde wie folgt vorgegangen:

Der Farbwert (Integer; 0 bis 29 möglich) wird analog zu den anderen Eigenschaftswerten in einem Textfeld gespeichert, dem Benutzer jedoch nicht angezeigt, sondern als, dem Farbwert entsprechend, eingefärbtes Rechteck dargestellt (siehe Abbildung 76). Daneben befindet sich ein Button, bei dessen Betätigung eine Farbpalette angezeigt wird. Diese beinhaltet 30 verschiedene Systemfarben. Durch Links-Doppelklick auf eine der Farben wird diese selektiert und der neue Farbwert ermittelt. Die gewählte Farbe wird im Modifikationsfenster aktualisiert. Durch Betätigung der OK-Taste werden die neuen Werte übernommen und das Getriebeteil bzw. die Baugruppe in der neuen Farbe angezeigt.

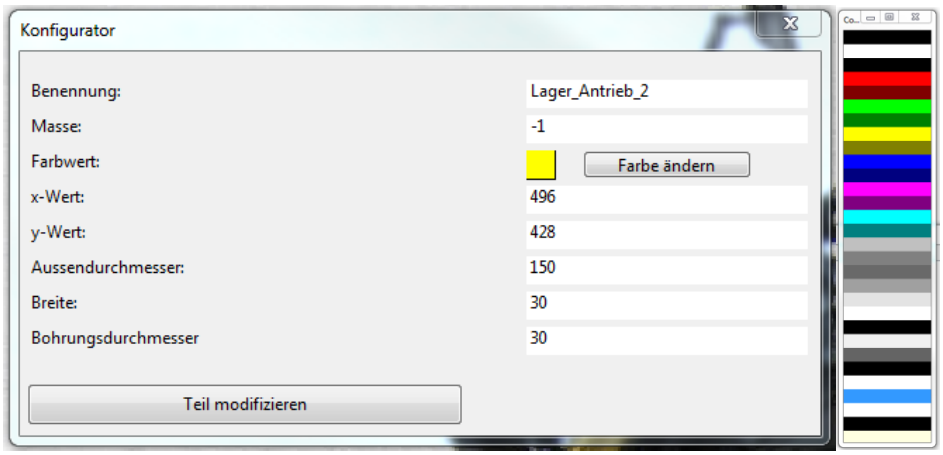

Abbildung 76: Modifikation der Farbe des ausgewählten Lagers (rechts: Farbpalette)

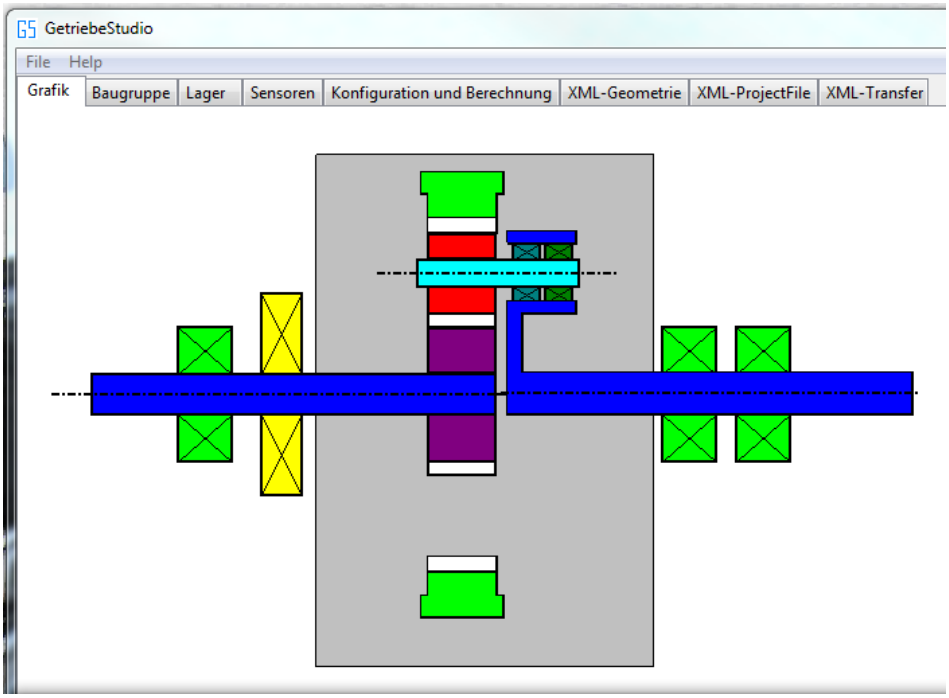

Abbildung 77: Darstellung des Getriebes nach Änderung der Farbe des zweiten Lagers von links

Berechnung der notwendigen Parameter für das Analyseprogramm:

Das Programm GetriebeStudio stellt eine wichtige Schnittstelle zu anderen Programmen, wie beispielsweise SPECTIVE[10], dar. Die Kommunikation zwischen den Programmen erfolgt mittels eines Transferdokuments auf XML-Basis. Die benötigten Parameter werden mit Hilfe des GetriebeStudios modelliert, spezifiziert und in bestimmten Fällen, wie etwa der Drehzahl der Bauteile, der Schadensfrequenzen von Lagern und der Zahneingriffsfrequenzen berechnet.

Berechnung der Drehzahl

Hierzu wurde folgender Ansatz entwickelt:

Unter den verknüpften Teilen wird eine Referenzwelle (Welle, Planetenträger oder Hohlrad) selektiert. In weiterer Folge soll es auch möglich sein, mehrere Wellen, Planetenträger und/oder Hohlräder als Referenzwelle zu wählen. Zu dieser Welle wird eine Drehzahl spezifiziert (Reiter „Konfiguration und Berechnung“). Die Referenzwelle und ihre Drehzahl werden in eine Drehzahltabelle (Hashtabelle) gespeichert. Die Berechnung startet bei eben dieser Referenzwelle. Nun wird überprüft, ob die Welle mit anderen Bauteilen verbunden ist. Eine Welle kann unter anderem mit Lagern und Zahnrädern verbunden sein. Werden diese Bauteile gefunden, wird deren Drehzahl berechnet und in die Drehzahltabelle eingetragen. Dies geschieht solange, bis alle verknüpften Bauteile eine Drehzahl zugeordnet bekommen haben. Für die Berechnung der Drehzahlen einer Baugruppe ist es von besonderer Wichtigkeit zuvor anzuge-

[10] Es handelt sich hierbei um ein Analysetool zur Zustandsüberwachung von Maschinen.

ben, um welche Art von Baugruppe es sich handelt, z.B. ein Planetengetriebe, welche Teile der Baugruppe Eingänge („getrieben“) und welche Ausgänge („treibend“) sind und welche Funktion bestimmte Teile im Getriebe haben. Zum Beispiel kann ein Zahnrad eines Planetengetriebes ein Sonnenrad oder Planet sein. Eine falsche oder fehlende Spezifikation führt zum Abbruch der Berechnung und Anzeige des Problems. Die Berechnung der Drehzahlen der Komponenten einer Baugruppe beginnt erst, wenn alle Eingangsdrehzahlen bekannt sind. Sind diese bekannt, werden nach einem festgelegten Schema die Drehzahlen der Komponenten berechnet.

Berechnung - Stufengetriebe

Das Drehzahlverhältnis zweier über Zahnräder miteinander verbundener Wellen verhält sich umgekehrt proportional zum Verhältnis der Anzahl ihrer Zähne bzw. dem Verhältnis ihrer Teilkreisdurchmesser (vgl. Verzahnungsgesetz in [3] auf Seite 677).

Berechnung - Planetengetriebe

Die Formeln für die Berechnung der Winkelgeschwindigkeit des Sonnenrades ω_S, des Hohlrades ω_H, der Planeten ω_P und des Planetenträgeres ω_T sind in Abhängigkeit des Radius des Teilkreises des Sonnenrades r und des Radius des Teilkreises des Hohlrades R nachfolgend angeführt (vgl. [22]):

$$\omega_S r + \omega_H R = \omega_T (R + r)$$

$$\omega_P = \frac{\omega_H R - \omega_S r}{R - r} \quad .$$

Mit Hilfe des Zusammenhangs zwischen (Teilkreis-)Radius, Modul und Zahnanzahl eines Zahnrades (vgl. Kapitel 20, Formel 21.1 sowie Profilverschiebung und Schrägungswinkel in [3]) sowie des Zusammenhangs zwischen Winkelgeschwindigkeit und Drehzahl, konnten aus obigen Formeln die folgenden Zusammenhänge zwischen der Drehzahl des Sonnenrads n_S, des Hohlrades n_H, der Planeten n_P und des Planetenträgeres n_T in Abhängigkeit der Zahnanzahl des Hohlrades z_H und des Sonnenrades z_S abgeleitet werden:

$$n_S z_S + n_H z_H = n_P (z_H + z_S)$$

$$n_P = \frac{n_H z_H - n_S z_S}{z_H - z_S} \quad .$$

Obige Zusammenhänge gelten für gerad- und schrägverzahnte Zahnräder mit oder ohne Profilverschiebung.[11] Die hergeleiteten Formeln wurden mittels bekannter Getriebedaten auf Korrektheit überprüft.

[11] Im Transferdokument wird die Relativdrehzahl zwischen Planeten und Planetenträger als Drehzahl des Planeten festgehalten, daher $n_P - n_T$ berechnet und gespeichert

Speichern des Projekts

Die modellierten getriebe- und projektspezifischen Daten werden im Projektordner in zwei Files gespeichert (Geometry.xml, Project.xml). Es werden hierfür die beiden Funktionen „Save“ und „Save as“ im Menü angeboten. Die berechneten Daten werden nach erfolgreicher Berechnung in das Transfer.xml – File des Projektes gespeichert.

Anzeige der erstellten Dokumente

Zur Kontrolle von durchgeführten Änderungen können die gespeicherten Dokumente direkt über die dafür vorgesehenen Reiter mittels Browser angezeigt werden.

Scrollen und Zoomen

Durch Scrollen und Zoomen ist es möglich auch große Modelle graphisch vernünftig darzustellen.

Scrollen: Hierbei wird das dargestellte Modell in x- und y-Richtung (vertikal und horizontal) verschoben. Der Maßstab wird nicht verändert. Der Benutzer geht hierbei wie folgt vor:

1. Wechseln in den Graphikreiter (einmal in den Graphikbereich klicken).
2. Zum Starten des Scroll-Vorgangs die mittlere Maustaste drücken.
3. Verschieben der Maus in die Richtung, in die der dargestellte Bildschirmbereich verschoben werden soll. Die Anzeige der Verschiebung des dargestellten Bildschirmbereichs erfolgt dabei in Echtzeit.
4. Zum Beenden des Scroll-Vorgangs mittlere Maustaste drücken.

Zoomen: Der Maßstab des dargestellten Zeichenbereiches kann wie folgt verändert werden.

1. Wechseln in den Graphikreiter (einmal in den Graphikbereich klicken).
2. Drehen des Mausrädchens nach oben zum Vergrößern bzw. nach unten zum Verkleinern.
3. Maßstab vergrößert/verkleinert sich.
4. Die Koordinaten der dargestellten Objekte werden neu berechnet und die Größe der Objekte an den neuen Maßstab angepasst.
5. Die Grafik wird aktualisiert.

Die hierfür nötigen Transformationen für das Zoomen werden im Folgenden beschrieben:

Als Zentrum (Z) der Vergrößerung beziehungsweise Verkleinerung wurde die Mitte der Zeichnungsfläche gewählt. Der Koordinatenursprung (U) der dargestellten Objekte befindet sich in der linken, oberen Ecke der Zeichenfläche. Soll nun ein Punkt (P) eines Objektes (dessen Koordinaten x und y bekannt sind und die sich auf den nicht verschobenen (kein Scrollen), nicht gezoomten Ausgangszustand beziehen) maßstabsgerecht angezeigt werden, kann folgender mathematische Zusammenhang genutzt werden:

zf …	Zoom-Faktor	mit	zf = 1	im Ausgangszustand,
			zf > 1	bei Vergrößerung und
			0 < zf < 1	bei Verkleinerung

ZU … Vektor zwischen Zentrum Z und Ursprung U

ZP … Vektor zwischen Zentrum Z und Punkt P vor der Vergrößerung

UP … Vektor zwischen Ursprung U und Punkt P

P´ … Lage des Punktes P nach der Vergrößerung, ohne Korrektur

P* … Lage des Punktes P nach der Vergrößerung, mit Korrektur

$ZP = ZU + UP$

$ZP_{(zf = 1)} = ZU + UP_{(zf = 1)}$

$ZP_{(zf)} = zf\ ZP_{(zf = 1)} = zf\ UP_{(zf = 1)} + (zf - 1)\ ZU$

In Abbildung 78 ist dieser Zusammenhang graphisch dargestellt. Betrachtet man einen beliebigen Punkt P, so verschiebt sich dieser entsprechend dem gewählten Vergrößerungsfaktor (zf), auf seine neue Lage P*. Dies kann auch erreicht werden, indem man den Punkt P zunächst auf P‘ verschiebt und weiter auf P*. P, P‘ und P* bilden zusammen mit U, Z, und P ähnliche Dreiecke (zu jedem Winkel des einen Dreiecks gibt es einen gleich großen im Anderen) wodurch auch die Affinitätsgesetze gelten, die nachfolgend zur Herleitung des oben erwähnten Zusammenhangs genutzt werden:

$ZP^* = zf\ ZP$

$PP^* = ZP^* - ZP = zf\ ZP - ZP = (zf - 1)\ ZP$

$UP' = zf\ UP$

$PP' = UP' - UP = zf\ UP - UP = (zf - 1)\ UP$

$P'P^* = -(PP' - PP^*) = (zf - 1)\ ZU$

$ZP_{(zf)} = ZP^* = UP' + P'P^* = zf\ UP + (zf - 1)\ ZU$

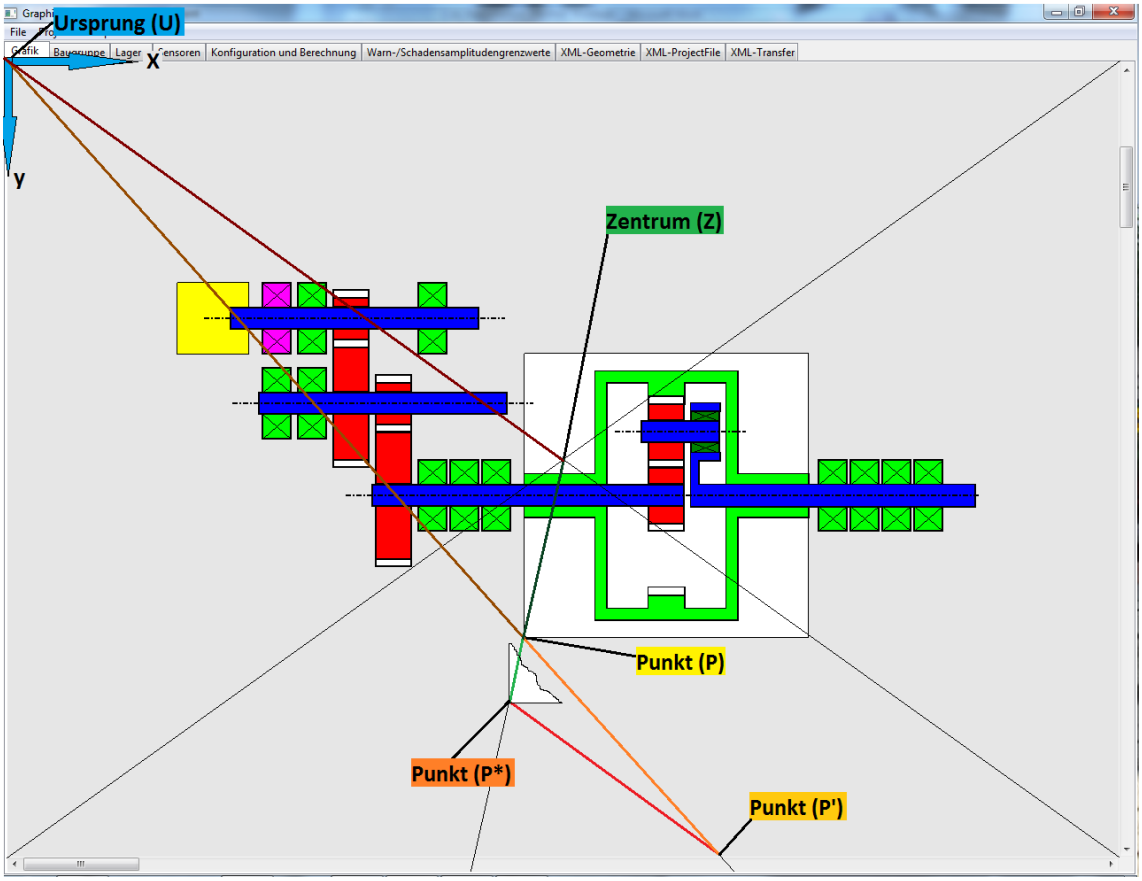

Abbildung 78: Zoom - Koordinatentransformation

10.1 Tools zum Erzeugen von GUI-Oberflächen

Zur Reduktion der Entwicklungszeit empfiehlt es sich geeignete Tools zur Modellierung der benötigten Oberflächen einzusetzen. Als Beispiel hierfür ist das Tool „WindowBuilder“ zu nennen, das als Plug-in für Eclipse angeboten wird (siehe auch [23]). Die Modellierung erfolgt hierbei graphisch. Die Bausteine können per „Drag and Drop“-Funktion hinzugefügt werden. Der benötigte Source-Code wird automatisch erstellt. In Abbildung 79 wurde beispielhaft eine SWT-Shell (Window) mit einem Menü sowie einigen Reiterkarten modelliert. Hierzu werden die benötigten Strukturen im mittig platzierten Toolkasten selektiert und im rechten Bereich an passender Stelle (Möglichkeiten werden angezeigt) abgelegt. Im unteren Teil des Fensters können diverse Eigenschaften komfortabel angepasst werden. Der zugehörige Quellcode wurde automatisch generiert und ist in Abbildung 80 ausschnittsweise dargestellt.

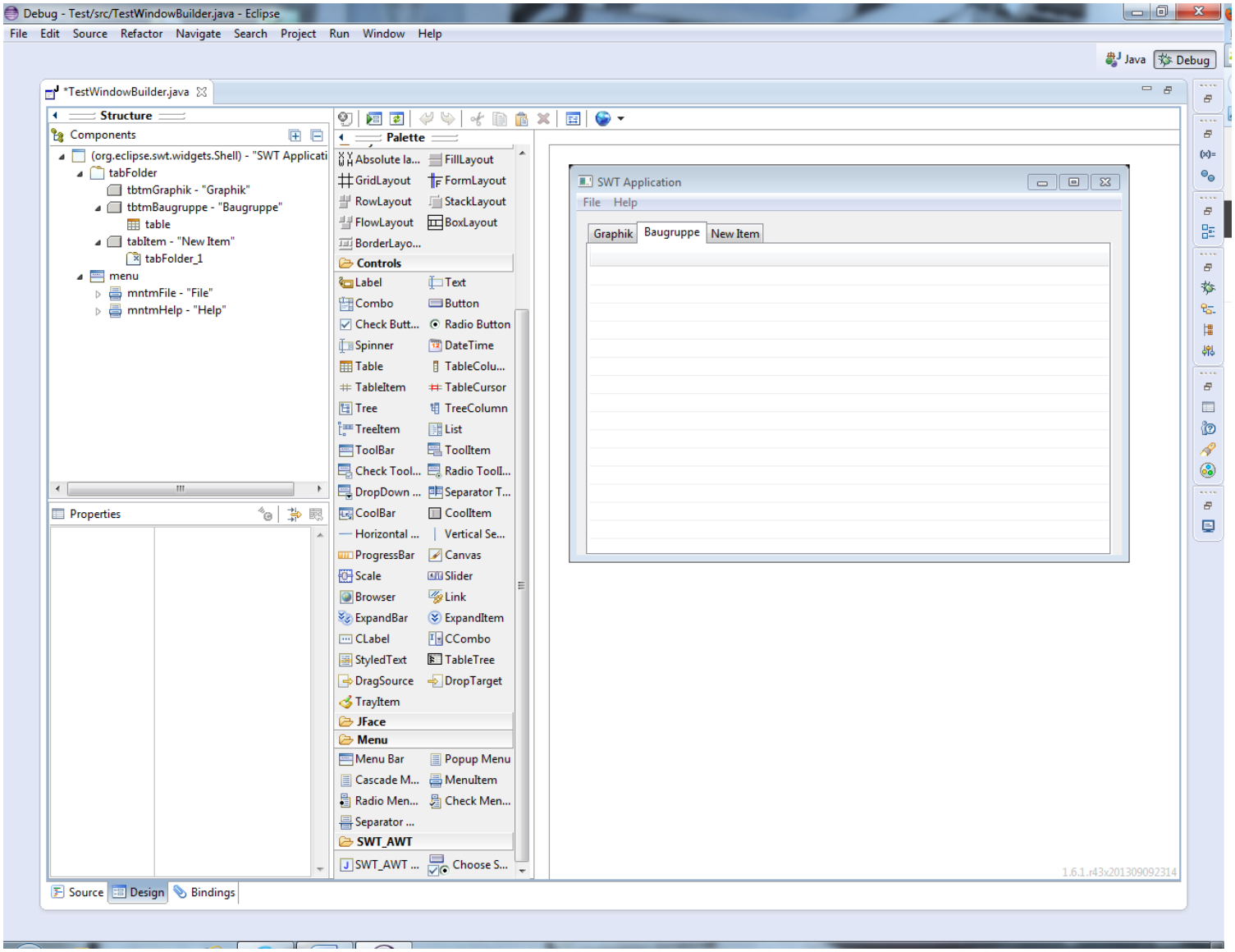

Abbildung 79: WindowBuilder Designansicht

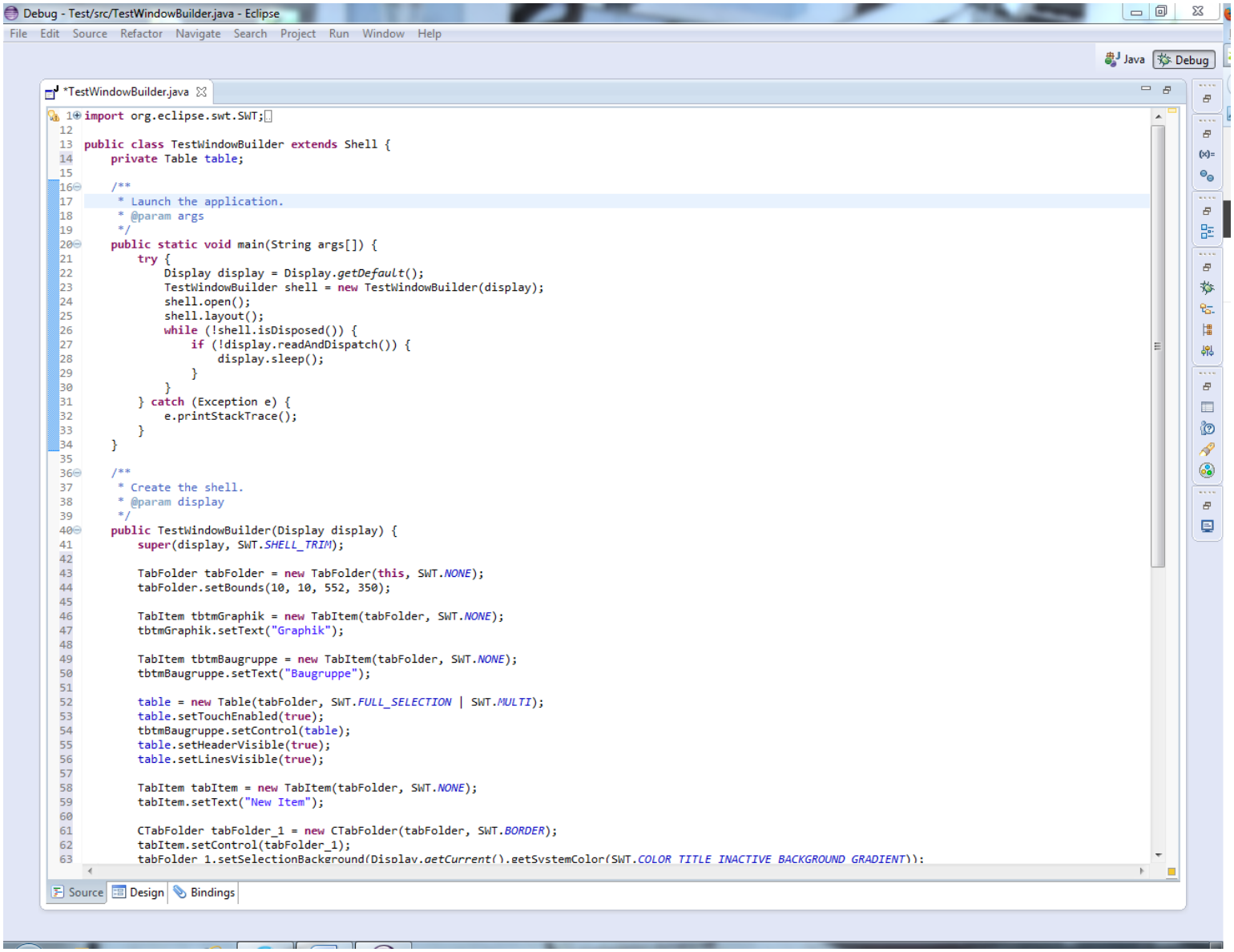

Abbildung 80: WindowBuilder Sourcecode-Ansicht

11 Software-Testen

Das Testen der Software stellt einen wichtigen Teil der Entwicklung einer Software dar. Im Folgenden werden daher wichtige Aspekte des Testens von Software kurz erläutert. Darüber hinausgehend können die Werke [17] und [24] zur Vertiefung der Kenntnisse des Testens von Software empfohlen werden.

11.1 Software-Testen-Einführung

Im Folgenden sollen wichtige Aspekte des Testens von Software im Allgemeinen und GUIs im Speziellen erläutert werden, wobei die folgenden Fragen beantwortet werden sollen:

- *Warum testet man Software?*
- *Welcher Aufwand ist mit dem Testen verbunden?*
- *Was soll getestet werden?*
- *Welche Modelle können als Testleitfaden herangezogen werden?*
- *Wann sollte im Projektverlauf mit dem Testen begonnen werden?*
- *Wer sollte testen?*
- *Womit wird getestet? Welche Werkzeuge stehen zur Verfügung? Welche Methoden gibt es?*

11.1.1 Testzweck („Warum testet man Software?“)

Zweck des Testens ist die Entdeckung von Fehlern in der Software. Als Fehler bezeichnet man hierbei das Abweichen der Software vom erwarteten Verhalten. Werden Fehler in der Software nicht entdeckt und entfernt, können diese zu erheblichen Folgekosten führen. So verursachte 2012 ein fehlerhafter Algorithmus, der den Kauf und Verkauf von Aktien veranlasste, einem Börsenhandelsunternehmen in nur 45 Minuten einen Schaden 440 Millionen Dollar [25].

11.1.2 Testaufwand

Nach [17] werden für das Testen von Software in Abhängigkeit der Größe und Komplexität des Systems etwa 25 bis 50 Prozent der Gesamtentwicklungszeit und –kosten aufgewendet.

11.1.3 Testkriterien („Was soll getestet werden?“)

Getestet werden sowohl die Funktionalität als auch nicht funktionale Qualitätskriterien wie beispielsweise Zuverlässigkeit, Benutzbarkeit (Erlernbarkeit, Anwendbarkeit), Performance und Sicherheit. Die gewünschte Funktionalität sowie nicht funktionale Abnahmekriterien werden im Pflichtenheft in Form eines Anforderungskataloges festgehalten (siehe auch Kapitel 5). Neben dem Pflichtenheft gibt es weitere Quellen wie etwa Richtlinien (nicht verbindlich) und Verordnungen (verbindlich), die vom Programm eingehalten werden sollen oder müssen ([17], [24]).

11.1.4 Testprozessmodelle

Prozessmodelle stellen einen Leitfaden zum Testen von Systemen dar. Eines der bekanntesten dieser Modelle ist das sognannte V-Modell (siehe [24], Seite 33; [26], Seite 375ff), das hier kurz erläutert werden soll. Das V-Modell (benannt nach seiner V-förmigen Darstellung, siehe Abbildung 81) wurde 1979 von Barry Boehm erstmals vorgestellt und dient als Grundlage weiterer aktueller Modelle. Das V-Modell ist sequentiell aufgebaut, das bedeutet, erst wenn eine Phase abgeschlossen ist, kann die Nächste beginnen. Das Prozessmodell startet mit der Anforderungsdefinition und endet mit dem Abnahmetest. Die V-Form des Modells soll andeuten, dass jeder konstruktiven Phase eine Phase des Prüfens gegenübersteht. Boehm möchte damit unterstreichen, dass als Basis jedes Tests bzw. jeder Testphase, die in der korrelierten, konstruktiven Phase erstellte Spezifikation (Anforderungen, Anwendungsfälle, Aktivitätsdiagramme, Code, etc.) dienen soll. Die verschiedenen Testphasen werden im Weiteren auch als Teststufen bezeichnet.

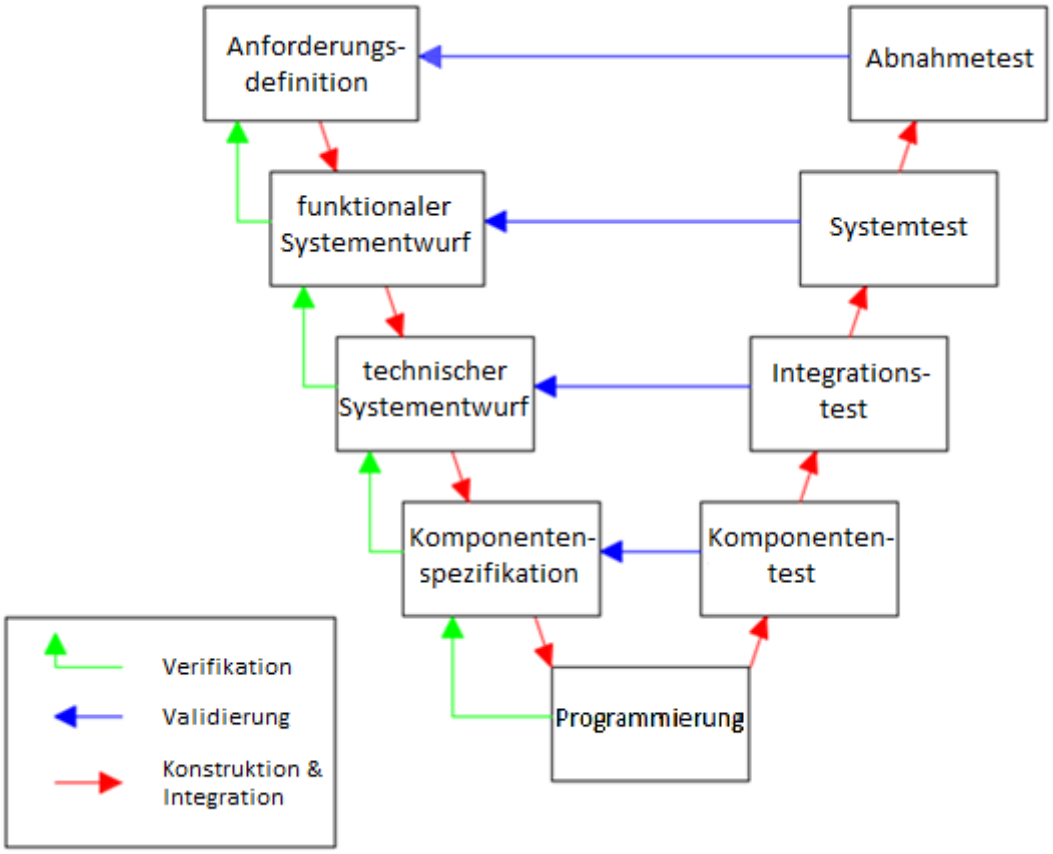

Abbildung 81: Das allgemeine V-Modell (nach [24], Seite 33)

11.1.5 Testbeginn

Wann sollte im Projektverlauf mit dem Testen begonnen werden? Hier gilt: Je früher ein Fehler entdeckt wird, desto geringer sind die Fehlerbehebungskosten. Bei klassischen Testansätzen (Wasserfallmodell, V-Modell) wird mit dem Testen (Validierung) erst nach der Codierung und Kompilierung der Software begonnen. Moderne Ansätze, wie z.B.: Test Driven Development, beginnen hingegen bereits in frühen Phasen des Entwicklungsprojekts.

11.1.6 Tester (Wer sollte testen?)

Die Software sollte nach Möglichkeit nicht von derselben Person getestet werden, die sie entwickelt. Nach Möglichkeit sollte ein dafür ausgebildeter und geprüfter Tester das Testen der Software übernehmen. Hierfür sprechen folgende Vorteile:

- Objektivität des Testers und
- Fach- und Methodenwissen.

Dem gegenüber stehen folgende Nachteile:

- Der Tester muss sich in die zu testende Software einarbeiten und
- der zusätzliche Abstimmungsaufwand.

11.1.7 Womit wird getestet? Welche Werkzeuge stehen zur Verfügung? Welche Methoden gibt es?

Zum Testen der Software werden im Allgemeinen spezielle Testtools eingesetzt. Abschnitt 11.4 Testwerkzeuge gibt eine Übersicht über die verschiedenen, verfügbaren Testwerkzeuge.

11.2 Testfälle

Tests werden in Form von Testfällen genau beschrieben bzw. spezifiziert. Konkret sollen folgende Fragen geklärt werden:

- Was ist ein Testfall?
- Welche Quellen für Testfälle gibt es?
- Wie können sie unterschieden werden?
- Welche Testfälle sind von besonderer Bedeutung?

11.2.1 Begriff Testfall

Nach [27] ist ein Testfall die Beschreibung eines elementaren, funktionalen Softwaretests, der zur Überprüfung einer, in einer Spezifikation, zugesicherten Eigenschaft eines Testobjektes dient. Eine Testfallbeschreibung besteht im Allgemeinen aus (nach [27]):

- Der Angabe der Vorbedingungen (Zustand des Systems vor Ausführung),
- der Bezeichnung und Spezifikation des Testobjektes,
- den Eingabedaten und Handlungen für die Durchführung des Tests,
- den erwarteten Ergebnissen und Nachbedingungen nach Durchführung des Tests und
- den Prüfanweisungen.

11.2.2 Testfallquellen

In Anlehnung an das, in Kapitel 11.1.4, beschriebene V-Modell beziehen sich die zu erstellenden Testfälle auf die in der frühen Phase des Projektes ermittelten Anforderungen bzw. den davon abgeleiteten Dokumenten. Zu diesen zählen u.a.:

- **User-Stories:** Sind von Nutzern in Alltagssprache formulierte Anforderungen an die zu entwickelnde Software (nach [10]) und dienen als wichtiges Kommunikationsmittel zwischen den zukünftigen Nutzern, den Entwicklern und Testern. Sie offerieren u.a. bereits in frühen Phasen der Entwicklung Abnahmetests zu erstellen (nach [17], Seite 408).

- **Anforderungskatalog** (vgl. Kapitel 5)

- **Anwendungsfälle** (vgl. Kapitel 9.1)

- **Aktivitätsdiagramme** (vgl. Kapitel 9.2)

Neben den genannten Dokumenten wird in der Praxis des Testens auch auf die Intuition und Erfahrung von Experten gesetzt (siehe [17], Seite 166ff).

11.2.3 Testfallarten

Im Allgemeinen können zwei Arten von Testfällen unterschieden werden:

- **Positivtest:** Es wird das Verhalten des Testobjekts infolge gültiger Vorbedingungen überprüft [27].

  Beispiel – Taschenrechner:
  Zwei Zahlen werden addiert. Der Test gilt als bestanden, wenn die Summe der beiden Zahlen dem erwarteten Wert entspricht.

- **Negativtest:** Prüft das Verhalten des Testobjektes aufgrund ungültiger Vorbedingungen und/oder Eingaben [27].

  Beispiel – Taschenrechner:
  Es soll eine Division durch Null ausgeführt werden. Diese würde bei der Ausführung einen Laufzeit-Fehler auslösen. Ein Test soll die korrekte Fehlerbehandlung testen.

11.2.4 Testprioritäten

Das Setzen von Prioritäten ist beim Testen von Software sehr wichtig. Da nicht alles Erdenkliche und darüber Hinausgehende aus Zeit und Ressourcengründen getestet werden kann (siehe „Psychologie des Testens“ in [24]), muss rechtzeitig festgelegt werden, welche Komponenten gründlicher getestet werden als Andere.

Von besonderer Bedeutung für das hier zu entwickelnde Modellier- und Managementtool GetriebeStudio sind u.a. das Speichern und Laden der Modelle sowie die korrekte Berechnung der Wellendrehzahlen, Zahneingriffs- und Lagerfrequenzen.

11.3 Testarten

Tests lassen sich anhand einer Reihe von Merkmalen zu unterschiedlichen Kategorien bzw. Testarten zuordnen. Man unterscheidet im Allgemeinen folgende Testarten:

- Gliederung nach Teststufen,
- statischer/dynamischer Test,
- funktionaler/nicht-funktionaler Test und
- manuelles Testen/automatisiertes Testen/Regressionstest.

Im Folgenden werden die einzelnen Testarten näher erläutert.

11.3.1 Gliederung nach Teststufen

Die Gliederung erfolgt in Anlehnung an das, im Kapitel 11.1.4, beschriebene V-Modell. Dieses unterscheidet folgende Teststufen:

Unittest

Die Tests beginnen auf der untersten Stufe mit dem Prüfen separat testbarer Komponenten in Form von Klassen, Modulen, Objekten, etc. Als Testbasis dient der vorhandene kompilierte Code.

Integration & Systemtest

Die nächsthöhere Stufe beschäftigt sich mit Komponenten- und Systemintegrationstests sowie den Gesamtsystemtests. Die Dokumentation der Architektur und des Entwurfes dienen hierbei als Testbasis für die Integrationstests. Die Anforderungsspezifikation (Requirement Specification) und Anwendungsfälle (Usecases) werden als Basis für die Gesamtsystemtests herangezogen.

Akzeptanztest

Das Gesamtsystem wird schließlich vom Kunden oder einem Benutzer des Systems getestet. Erst hier wird deutlich, ob das System den Erwartungen des Kunden genügt.

Nähere Erklärung der Teststufen siehe Kapitel 7.3 in [26].

11.3.2 Statische und dynamische Tests

Im Folgenden werden Tests danach unterschieden, ob das Testobjekt zur Ausführung kommt (dynamischer Test) oder nicht (statischer Test).

11.3.2.1 Statischer Test

Bei einem statischen Test wird das Testobjekt nicht ausgeführt. Man benötigt daher keinen ausführbaren Code um das Objekt zu testen. Als Testobjekte eignen sich neben Code auch andere Dokumente wie etwa Anwendungsfall- und Zustandsgraphen zur Analyse. Mit Hilfe von statischen Tests können daher Fehler und Abweichungen von Spezifikationen, Standards und Richtlinien früh entdeckt und Korrekturen veranlasst werden. Eine oder mehrere Personen analysieren das Testobjekt mit Hilfe verschiedener statischer Methoden bzw. mit Hilfe spezieller Werkzeuge, die im Folgenden kurz erläutert werden sollen (nach [17], Seite 81ff).

Strukturierte Gruppenprüfung (Review)[12]

Es handelt sich hierbei um eine nicht werkzeugunterstütze Art der Analyse durch eine Gruppe von Personen. Die einzelnen Gruppenmitglieder haben dabei verschiedene Rollen (Manager, Moderator, Autor, Protokollant und Gutachter) zu erfüllen. Generell durchläuft ein Review sechs Arbeitsschritte: Planung, Vorbesprechung, Vorbereitung, Reviewsitzung, Überarbeitung und Nachbereitung. Je nach Reviewart können die einzelnen Arbeitsschritte unterschiedlich stark ausgeprägt sein. In der eigentlichen Reviewsitzung werden dem Autor Fragen zu dem von ihm erstellten Analyseobjekt gestellt. Fehler und Mängel werden in einem Protokoll festgehalten. Reviews sind grundsätzlich auf alle möglichen Dokumente anwendbar (z.B. Anforderungsspezifikation, Verträge, Programmtexte, etc.) und bildet im Allgemeinen den Abschluss einer Phase im V-Modell.

Man unterscheidet folgende Reviewarten:

- **Walkthrough:** Review mit Schwerpunkt Sitzung.
- **Inspektion:** Formellste Form aller Reviews. Der Ablauf folgt bestimmten Regeln. Die Gutachter nutzen u.a. Checklisten zur Bestimmung von Unklarheiten, Fehlern und Defekten.
- **Technisches Review:** Mehrere Gutachter überprüfen individuell, ob das Prüfobjekt allen Spezifikationen, Standards und Richtlinien etc. genügt.
- **Informelles Review:** Abgeschwächte Form eines Reviews in stark vereinfachter Form. Auf eine Sitzung wird oft verzichtet. Die Rückmeldung der Gutachter erfolgt meist schriftlich.

[12] nach [17], Seite 81ff

Statische Analyse[13]

Im Gegensatz zu Reviews ist die statische Analyse rechengestützt. Das Spektrum der Analyse reicht von einer einfachen Rechtschreibprüfung hin bis zur Prüfung der formalen Struktur. Die statische Analyse wird u.a. für die Analyse von technischen Anforderungen, der Softwarearchitektur und des Softwareentwurfs eingesetzt.

Die statische Analyse wird oft in der Vorbereitungsphase eines Review eingesetzt. Der Einsatz statischer Tests kann eine Vielzahl von Fehlern bereits in einer frühen Phase der Entwicklung aufdecken. Viele Fehler sind jedoch erst zur Laufzeit, daher mittels dynamischer Tests, zu beobachten.

11.3.2.2 Dynamischer Test

Im Unterschied zu statischen Tests wird bei einem dynamischen Test das Testobjekt zur Ausführung gebracht. Man benötigt daher ein ablauffähiges Programm. In den unteren Teststufen (Komponenten- und Integrationstests) wird ein sogenannter Testrahmen benötigt, in dem das zu prüfende Testobjekt, z.B. eine Komponente, eingebettet und getestet werden kann. Des Weiteren werden auch Testdaten sowie Stubs (simulierte Schnittstellen zu anderen Programmen) und Treibern (Versorgen das Testobjekt mit Eingabedaten) für die Durchführung eines Tests benötigt (nach [17], Seite 109ff).

11.3.3 Funktionale und nicht-funktionale Tests

Das Prüfen der Funktionalität stellt einen wichtigen Aspekt des Testens dar. Daneben müssen aber auch Eigenschaften wie Performance, Benutzbarkeit oder etwa Sicherheit überprüft werden. Dementsprechend können Tests in *funktionale* und *nicht-funktionale Tests* gegliedert werden.

11.3.3.1 Funktionale Tests

Funktionale Tests können weiter in White- und Blackbox-Test unterteilt werden.

Als Blackbox bezeichnet man hierbei einen Teil der Software, deren genauer innerer Aufbau dem Tester nicht bekannt ist. Bei einem Blackbox-Test wird die Blackbox mit Eingabedaten versorgt und die erzeugten Ausgangsdaten mit dem erwarteten Output verglichen.

Bei einem Whitebox-Test ist hingegen der innere Aufbau der Box genau bekannt. Ziel des Testens ist eine möglichst hohe Codeabdeckung. Folgende Codeabdeckungsmaße werden unterschieden (vgl. Kapitel 7.4 in [26] sowie Kapitel 5.2 in [17]):

[13] nach [17], Seite 98ff

- **Anweisungsabdeckung:** Die Testfälle sind so zu wählen, dass alle Knoten des Kontrollflusses des zu testenden Objektes zumindest einmal durchlaufen werden (C0-Abdeckung).

- **Zweigabdeckung:** Die Testfälle werden so gewählt, dass alle Kanten des Kontrollflusses zumindest einmal durchlaufen werden (C1-Abdeckung).

- **Entscheidungsabdeckung:** Die Testfälle sind so auszuwählen, dass alle atomaren Entscheidungen der Entscheidungsstruktur zumindest einmal wahr und einmal falsch sind (C2-Abdeckung). Zusätzlich kann auch gefordert sein, dass alle Teilentscheidungen zumindest einmal richtig und einmal falsch sind (C3-Abdeckung).

- **Pfadabdeckung:** Die Testfälle sind so zu wählen, dass jeder mögliche Pfad zumindest einmal durchlaufen wird (C4-Abdeckung). Eine hundertprozentige Pfadabdeckung ist oft nicht erreichbar. So können Schleifen ohne definierte Schleifenzahl zu einer sehr hohen Anzahl an Pfaden führen, welche wiederum von einer großen Anzahl von Testfällen abgedeckt werden müssten, um eine vollständige Pfadüberdeckung sicherzustellen.

Die Anweisungsüberdeckung (C0-Abdeckung) ist das einfachste, aber auch schwächste, Abdeckungsmaß. Die Pfadabdeckung (C4-Abdeckung) hingegen ist das aufwendigste, aber auch strengste, aller vier Abdeckungsmaße. Um die Anzahl der möglichen Werte für die Eingabeparameter möglichst überschaubar zu halten, werden verschiedene funktionale Methoden wie etwa die *Äquivalenzklassenmethode*, die *Grenzwertanalyse* oder die *Klassifikationsbaummethode* eingesetzt. Diese und weitere Methoden werden in [26] (Kapitel 7.4.2) genauer erläutert.

11.3.3.2 Nichtfunktionale Tests

Mit Hilfe nichtfunktionaler Tests soll überprüft werden, ob das zu testende System die, an dieses gestellten, Qualitätsanforderungen (Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit, Übertragbarkeit und Sicherheit) erfüllt. Zu den wichtigsten nichtfunktionalen Tests zählen *Leistungsfähigkeits-* und *Benutzbarkeitstests*, die im Folgenden kurz erläutert werden.

- **Leistungsfähigkeitstest** (nach Kapitel 3.7.2 in [17] und Kapitel 7.5 in [26]): Dient der Überprüfung der Leistungs- und Skalierfähigkeit. Als Qualitätskriterium werden zum Beispiel die Antwortzeit auf eine Anfrage, die Anzahl durchführbarer Operationen pro Zeiteinheit und/oder der dabei benötigte Speicherbedarf erhoben.
  Zu den wichtigsten Leistungsfähigkeitstests zählen:

  - Performancetest: Prüft die Leistungsfähigkeit des Testobjektes für bestimmte Anwendungsfälle (Usecases) bei gleichbleibender oder zunehmender Last.

- Lasttest: Spezieller Performancetest bei dem das System an seine Grenze der Belastbarkeit gebracht wird. Durch diesen Test sollen Systemengstellen („Bottlenecks“) wie etwa ein zu kleiner Hauptspeicher oder die Verwendung ungeeigneter Algorithmen aufgezeigt werden.

- Stresstest: Zeigt auf, wie sich das System bei Überlastung verhält.

Um das Testergebnis nicht zu verfälschen, muss die Testumgebung exakt der Produktivumgebung gleichen.

- **Benutzbarkeitsevaluierung** (nach Kapitel 12 in [26]): Erstellte Konzepte, Designs, Prototypen und schließlich das erstellte System werden bezüglich ihrer Benutzbarkeit evaluiert.

  Hierbei werden zwei grundlegend verschiedene Herangehensweisen unterschieden:

  - *Empirische Methoden* werden eingesetzt, um Fehler in der Praxis aufzudecken und binden grundsätzlich die zukünftigen Benutzer des Produktes bzw. einen repräsentativen Vertreter dieser Gruppe in die Evaluierung des zu prüfenden Objektes mit ein. Als Beispiele werden in [26] (Kapitel 12) *Usability-Tests mit Benutzern* und *Feldtests* angeführt und näher erläutert.

  - Bei *nichtempirischen Methoden* hingegen versuchen Experten, die über das nötige Wissen, über die Domäne und die Benutzer sowie theoretische Grundlagen, Richtlinien und Modelle verfügen, Fehler vorherzusehen. Der Experte versetzt sich hierbei in die Rolle des Benutzers. Als Beispiele für nichtempirische Methoden werden in [26] (Kapitel 12) die *Heuristische Evaluierung*, der *Cognitive Walkthrough* und die *Task-Analyse* genannt und ausführlich beschrieben.

11.3.4 Manuelles Testen, automatisiertes Testen und Regressionstests

Softwaretests können manuell oder mit Unterstützung entsprechender Werkzeuge automatisiert durchgeführt werden. Beide Methoden haben Vor- und Nachteile. Manuelles Testen hat gegenüber automatisiertem Testen den Vorteil in der Vorbereitungsphase weniger Aufwand zu verursachen. Zudem entstehen keine zusätzlichen Kosten durch Lizenzen sowie Schulungs- und Einarbeitungsaufwand für benötigte Tools. Automatisiertes Testen wird meist eingesetzt, wenn ein Test öfter wiederholt werden muss. Als weitere Vorteile sind eine kurze Durchführdauer und die Möglichkeit einer automatisierten Berichterstattung anzuführen. Viele Tests, wie etwa Lasttests, sind meist nur als automatisierte Tests durchführbar. Ein weiteres Einsatzgebiet für automatisierte Tests bilden sogenannte Regressionstests. Nach [17] (Seite 76 bis 258) wird ein Regressionstest als „erneuter Test eines bereits getesteten Programmes bzw.

einer Teilfunktionalität nach deren Modifikation“ bezeichnet. Regressionstests sollen, durch die Modifikation des Systems eingebaute oder freigelegte, Fehlerzustände aufdecken.

11.4 Testwerkzeuge

Für das Management, Steuerung, Erstellen und Ausführen von Tests steht eine Vielzahl von Testwerkzeugen zur Verfügung (z.B. [28]). Im Folgenden sollen die einzelnen Werkzeugtypen kurz erläutert werden.

11.4.1 Werkzeuge für Management und Steuerung von Tests

Testmanagementtools erlauben nach [24] (siehe Kapitel 7, Seite 209) die Erfassung, Katalogisierung, Verwaltung und Priorisierung von Testfällen. Im Weiteren ermöglichen Testmanagementtools eine bequeme Überwachung des Status der Testfälle. Darüber hinausgehend bieten viele dieser Tools eine Reihe von Funktionen zur Unterstützung des Projektmanagements an.

11.4.2 Werkzeuge für die Spezifikation von Tests

Zu den wichtigsten Werkzeugen dieser Kategorie zählen sogenannte **Testdatengeneratoren** (siehe [17], Kapitel 7, Seite 212). Diese erlauben es, die für die Testfälle nötigen Testdaten zu generieren. Testgeneratoren lassen sich nach [29] in Abhängigkeit der betrachteten Testbasis unterscheiden. Als Testbasis dienen hierbei Datenbanken, der Quellcode des Testobjektes, die zu analysierenden Schnittstellen eines Testobjektes oder Spezifikationen. In [17] wird darauf hingewiesen, dass diese Werkzeuge Tests nach bestimmten Regeln systematisch erzeugen können, jedoch nicht unterscheiden, ob diese „gut oder schlecht, wichtig oder irrelevant“ sind. Zudem müssen meist die Sollreaktionen manuell hinzugefügt werden.

11.4.3 Werkzeuge für statische Tests

Für statische Tests (siehe Kapitel 11.3.2.1) stehen eine Reihe unterschiedlicher Werkzeuge zur Verfügung. Nach [17] (Kapitel 7, Seite 213) können diese wie folgt eingeteilt werden:

- **Werkzeuge zur Reviewunterstützung:** Für Planung, Durchführung und Auswertung von Reviews.

- **Statische Analysatoren:** Dienen u.a. zum Identifizieren von komplexen, fehleranfälligen (Source-)Codeabschnitten, sowie dem frühzeitigen Erkennen von Unstimmigkeiten und Fehlern im Quellcode wie z.B. Verletzung von Programmierrichtlinien.

- **Model Checker:** Spezielle Analysatoren für Spezifikationen, die die Struktur eines Modells überprüfen und fehlende Zustände, fehlende Zustandsübergänge und Inkonsistenzen im zu prüfenden Modell aufdecken.

11.4.4 Werkzeuge für dynamische Tests

Für dynamische Tests (vgl. Kapitel 11.3.2.2) kommen nach [17] (siehe Kapitel 7, Seite 215) folgende Testwerkzeuge zum Einsatz:

- **Debugger:** Tool zum Lokalisieren, Analysieren und Entfernen von Fehlerzuständen. Mit Hilfe des Debuggers kann ein Programm schrittweise ausgeführt und bei Bedarf angehalten werden. Zudem erlauben Debugger den Wert von Variablen zu setzen und die aktuellen Werte auszulesen.

- **Testtreiber, Platzhalter und Testrahmen:** Bezeichnet eine Gruppe von Programmen und Werkzeugen, welche das Verhalten, von über Schnittstellen mit dem Testobjekt verbundenen Komponenten und Systemen, simulieren, das Testobjekt mit Testdaten versorgt und dessen Antworten bzw. Reaktionen entgegennimmt.

- **Simulatoren:** Bilden die geplante Einsatz- oder Produktivumgebung nach und werden für Systemtests eingesetzt in denen die Produktivumgebung bzw. die Zielumgebung nicht für den Test zur Verfügung steht oder der hierfür nötige Aufwand zu groß wäre.

- **Testroboter:** Diese, auch *Capture-and-Replay-Tools* genannten, Werkzeuge arbeiten nach dem Prinzip eines Videorekorders und dienen dem Aufzeichnen und Wiederabspielen der Interaktion zwischen Nutzern und GUIs. Die während der Aufzeichnung aufgenommenen Bedienschritte (z.B.: Tastatur- und Mausklicks) werden in Form eines (Test-)Skriptes festgehalten und können für nachfolgende Tests angepasst werden.

11.4.5 Werkzeuge für nicht funktionale Tests

Für die, in Kapitel 11.3.3.2 erwähnten, nicht funktionalen Tests steht eine Vielzahl unterschiedlicher Werkzeuge zu Verfügung. Zu diesen zählen die folgenden Test-Werkzeuge (nach [17], Kapitel 7, Seite 220):

- **Last- und Performancetestwerkzeuge:** Generieren synthetische Last, z.B.: Datenbankabfragen, Benutzertransaktionen und Netzwerkverkehr; messen und protokolieren die Antwortzeit.

- **Monitore:** Soft- oder Hardwarewerkzeug zur Überwachung, Aufzeichnung, Analyse und Verifizierung des Betriebs des Testobjekts. Werden u.a. für Last- und Performancetests benötigt.

- **Werkzeuge zur Prüfung von Zugriffs- und Datensicherheit:** Dienen der Überprüfung eines Systems auf Sicherheitslücken, durch deren Ausnutzung unbefugte Personen auf das System zugreifen könnten. Zu dieser Kategorie von Werkzeugen zählen auch Virenscanner und Firewalls, da die von den Werkzeugen generierten Protokolle Hinweise auf Sicherheitsmängel liefern.

- **Werkzeuge für das Assessment der Datenqualität:** Werden für Systemtests und hier insbesondere in Datenkonversions- oder Migrationsprojekten zur Überprüfung der Datenbestände eingesetzt.

Darüber hinaus kommen vor allem **Checklisten** zur Evaluierung nichtfunktionaler Qualitätseigenschaften zum Einsatz. Diese werden insbesondere dann eingesetzt, wenn ein dynamischer Test zu teuer, zu aufwendig, nicht durchführbar oder nicht sinnvoll ist [30].

11.4.6 Werkzeuge für GUI-Tests

Für das Testen von GUIs wird aktuell eine Vielzahl verschiedener Werkzeug eingesetzt. Die derzeit verfügbaren Werkzeuge lassen sich u.a. nach ihrer Arbeitsweise, der Testumgebung und deren Einsatzspektrum unterscheiden.

Im Folgenden soll ein kurzer Überblick über derzeit am Markt befindliche Produkte zum Testen von auf SWT basierenden Softwareprojekten gegeben werden.

SWTBot: Es handelt sich hierbei um ein Open-Source, auf Java basierendes Tool zum Testen von SWT-Applikationen. Es bietet unter anderem einen Test-Recorder und -Generator zum einfachen Aufzeichnen von GUI-Aktionen. Der Recorder zeichnet die Interaktion des Benutzers mit dem GUI auf und erzeugt dabei den für die Tests benötigten Code. Der Tester kann diesen Code in seinem Testfall einbauen. Beim Durchführen der Tests werden die aufgezeichneten und in die Tests integrierten Abläufe automatisch wiederholt. Mittels „Assertions“ (Behauptungen beziehungsweise Aussagen über das erwartete Ergebnis) werden die erwarteten Ergebnisse des Tests spezifiziert. Treffen alle Assertions eines Tests zu, ist dieser bestanden (nach [31]).

WindowTester: Dieses Testwerkzeug stellt eine von Google entwickelte Alternative zu SWTBot dar. WindowTester steht derzeit nur für Java 1.5 und 1.6 sowie Eclipse 3.4 bis 3.6 zur Verfügung. Weitere Informationen sind unter [32] verfügbar.

QF-Test: Es handelt sich hierbei um eine kommerzielle Testumgebung zum Testen von Swing, SWT und Web-Applikationen (statisch und dynamisch) und ermöglicht unter anderem System-, Last- und Regressionstests. Dieses Testautomatisierungstool arbeitet nach dem „Capture and Replay“-Prinzip und erlaubt die erstellten Sequenzen in Form von Bibliotheken wiederzuverwenden [33]. Weitere Informationen sind unter [34] einsehbar.

GUIdancer/Jubula: In beiden Fällen handelt es sich um Eclipse-basierte Werkzeuge zum Erzeugen von automatisierten Tests für GUIs. GUIdancer erweitert Jubula um wichtige Funktionen wie etwa Code Coverage und Reporting. Im Unterschied zu anderen GUI-Testtools, erlauben GUIdancer und Jubula Tests bereits vor der eigentlichen Implementierung des Codes zu erstellen und damit die Umsetzung von Test-Driven-Development. Die Tests werden dabei mittels Drag&Drop aus einer Bibliothek atomarer Aktionen erstellt (nach [35]). Recording bleibt jedoch weiterhin möglich (nach [36]).

Beide Tools werden ständig weiterentwickelt und sind als Open-Source-Produkte verfügbar. Weitere Informationen können unter [37] nachgelesen werden.

TPTP: Bei diesem Tool handelt es sich um einen automatisierten GUI-Recorder, der in seiner Arbeitsweise SWTBot ähnelt. TPTP ist auf die Plattform Eclipse Helios und Galileo beschränkt. Weiters wird nur Java 5 unterstützt (Details siehe [38]).

Im Anhang befindet sich eine Liste weiterer, gängiger GUI-Testtools.

11.5 Testautomatisierung

Unter Testautomatisierung wird in der Regel die vollautomatische Durchführung und Verifikation von Testfällen verstanden (nach Kapitel 7.6 in [26]). Ferner gilt die automatisierte Erstellung von Testfällen als typische automatisierte Aktivität bei Softwaretests. Im Folgenden werden die verschiedenen Einsatzmöglichkeiten der Testautomatisierung (Kapitel 11.5.1) betrachtet und deren Einsatz im Projekt (11.5.2) bewertet.

11.5.1 Einsatzmöglichkeiten der Testautomatisierung

Zu den wichtigsten Einsatzgebieten der Testautomatisierung zählen (nach Kapitel 7.6 in [26]):

- **Automatisierte Komponententests** (nach Kapitel 7.6.1 in [26])**:** Mit Hilfe moderner Unit-Test-Frameworks können Komponententests direkt in der Entwicklungsumgebung definiert und parallel zur Entwicklung ausgeführt und ausgewertet werden. Häufig werden parallel dazu Code-Coverage-Frameworks (siehe auch Kapitel 11.3.3.1) eingesetzt.

- **Automatisierte GUI-Tests:** Das Simulieren eines oder mehrerer Benutzer eines Systems über dessen graphische Benutzerschnittstelle zählt zu den klassischen Testautomatisierungsgebieten. Folgende Verfahren kommen in der Praxis zum Einsatz:

  - ***Capture & Replay* – Verfahren** (nach Kapitel 7.6.2 in [26])**:** Wie in Kapitel 11.4.4 angemerkt, werden die Benutzerinteraktionen zunächst mit Hilfe eines Recorders aufgezeichnet. Die Aufzeichnungen stehen für die nachfolgenden Tests in Form von Testskripts zu Verfügung.

    Bekannte GUI-Testwerkzeuge:

    - SWTBot (siehe Kapitel 11.4.6)
    - WindowTester (siehe Kapitel 11.4.6)
    - Selenium[14]

[14] Für Web-Anwendungen, Homepage: http://www.seleniumhq.org/

Vorteile:

- Einfaches Verfahren
- Leicht Erlernbar
- Für primitive Regressionstests/Lasttests geeignet

Nachteile:

- Schlechte Wartbarkeit der Tests
- Erst anwendbar, wenn das Testobjekt zur Verfügung steht

- ***Skripted* – Verfahren:** Die benötigten Testskripten werden von Grund auf Schritt für Schritt entwickelt. Zum Erstellen der Skripten stehen höhere Programmiersprachen wie zum Beispiel C++ oder Java sowie spezialisierte GUI-Testwerkzeuge zu Verfügung.

  Bekannte GUI-Testwerkzeuge:

  - GUIdancer (siehe Kapitel 11.4.6)
  - Jubula (siehe Kapitel 11.4.6)

  Vorteile:

  - Das Testobjekt ist für die Erstellung der Tests nicht zwingend erforderlich
  - Gut wartbare Tests

  Nachteile:

  - Im Vergleich zu „Capture & Replay"-Verfahren längere Vorbereitungsphase für das Erstellen der Tests

- **Regressionstests** (nach Kapitel 7.6.23 in [26], siehe auch Kapitel 11.3.4)**:** Diese prüfen, ob das Systemverhalten nach einer Systemänderung noch gewährleitet ist. Das korrekte Systemverhalten wird durch die vorhergehende Version bestimmt. Durch die oftmalige Ausführung von Regressionstests über den Entwicklungsprozess sind diese wirtschaftlich für die Testautomatisierung geeignet.

11.6 Testen des GetriebeStudios

Im Folgenden wurde auch über den Einsatz von speziellen GUI-Testwerkzeugen zur Automatisierung der Tests nachgedacht. Als wesentliche Vorteile sind hierbei u.a. die kurze Testdurchführungsdauer, die Wiederholbarkeit der Tests und die automatisierte Dokumentation zu nennen. Dem gegenüber stehen jedoch folgende Nachteile: Schulungsaufwand für das Testtool, Testvorbereitungsdauer, Testwartungsaufwand, etc. (vgl. auch Kapitel 11.3.4).

Im Rahmen der Diplomarbeit wurden folgende GUI-Testtools bezüglich ihrer Einsatzfähigkeit getestet:

- SWTbot,
- WindowTester,
- Jubula und
- Abbot.

Die Testprogramme wurden zunächst, wie in der jeweiligen Dokumentation beschrieben, installiert. Bevor mit den eigentlichen Tests begonnen werden konnte, wurde die vorhandene Dokumentation gelesen und das Erlernte mit Hilfe von Tutorials überprüft. Die getesteten Werkzeuge waren einfach zu bedienen und ermöglichten es, schnell und komfortabel Testfälle zu erstellen. Schließlich sollte das ablauffähige „GetriebeStudio“ mit Hilfe der Testprogramme getestet werden. Es zeigt sich, dass die Tools erhebliche Probleme haben das Programm zu Testen. Ein Vergleich mit den in den Tutorials verwendeten Programmcodes deutete darauf hin, dass die Testtools mit den neu erstellten Listener-Klassen nicht funktionieren. Der Programmcode wäre, um mit diesen Tools getestet werden zu können, komplett umzustrukturieren gewesen. Da dies den zeitlichen Rahmen der Diplomarbeit gesprengt hätte, wurde von der Verwendung dieser Tools abgesehen. Das Testen erfolgte daher rein manuell. Die erzeugten Klassen wurden mittels Unit-Tests geprüft. Für das Testen des Systems wurden verschiedene Projektdatensätze (XML-Dokumente) modellierter Getriebe erzeugt. Die Datensätze mussten korrekt eingelesen und verarbeitet werden. Danach wurde die Funktionalität des Programmes getestet. Das Programm wurde schließlich von einer Reihe von Key-Usern (von Kollegen und vom Projektpartner) getestet. Anfang 2014 wurde das Programm nach einigen Verbesserungen in der Produktivumgebung (Server am Betriebsgelände unseres Projektpartners) installiert, getestet und zu Verwendung freigegeben.

12 Wartung, geplante Erweiterungen und Änderungen

Zu den geplanten Erweiterungen und Änderungsmaßnahmen zählen:

- Bereitstellen neuer Bauteilmodelle: Kupplungen, Riemen und sonstiger wichtiger Bauteile.
- Bereitstellen neuer Getriebearten: z.B. Differential.
- Möglichkeit der Auswahl mehrerer Referenzwellen, statt wie bisher einer einzelnen.
- Möglichkeit zur Dokumentation: Hilfsgeometrie und Kommentare erzeugen und modifizieren; Fotos und/oder Graphiken hinzufügen.

Nach Rücksprache mit einem Key-User wurden folgende Verbesserungen durchgeführt:

Bezüglich Zuweisung von Lagerdaten/Lagersuchmaschine

- Das Suchergebnis wird nun lexikalisch aufsteigend nach der Bezeichnung des Lagers geordnet.
- Das Textfeld für die Eingabe der Lagerbezeichnung wurde durch eine Combo ersetzt (siehe Abbildung 82). Diese speichert die letzten zehn Suchbegriffe. Der letzte Suchbegriff wird beim Aufrufen der Funktion vorgewählt und das Suchergebnis angezeigt. Beim Schließen werden die Suchbegriffe in einem Textdokument persistent gespeichert, sodass beim nächsten Aufruf dieser Funktion, diese wieder in die Combo geladen werden und dem User zur bequemen Auswahl zur Verfügung stehen.
- Das Starten der Suche wird nun auch nach jeder Selektion aus der Combo bzw. durch Bestätigen der Eingabe der Lagerbezeichnung in den Eingabebereich der Combo ausgeführt.
- Die Zuordnung der Lagerdaten kann nun auch per Doppelklick auf die Zeile mit den entsprechenden Lagerdaten erfolgen (siehe Abbildung 82).

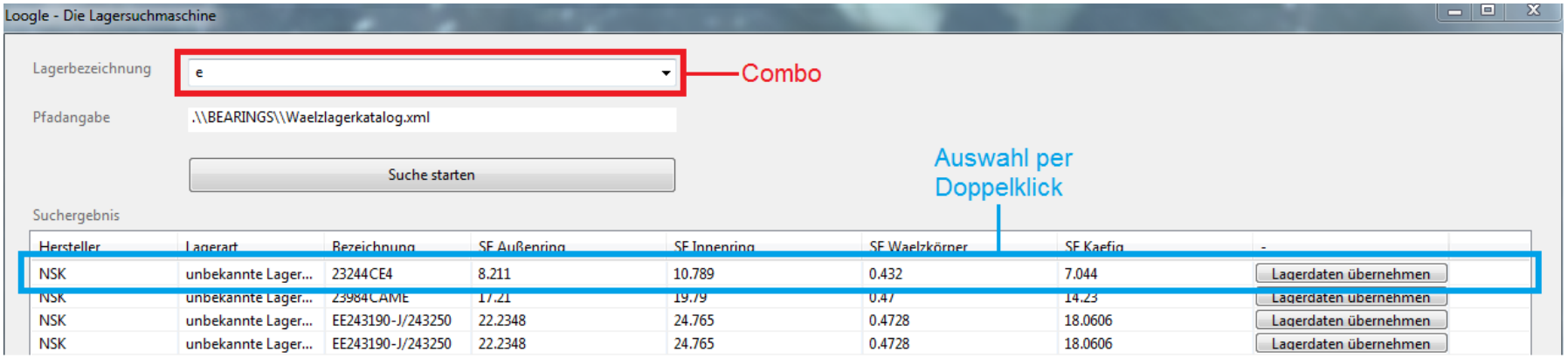

Abbildung 82: Lagersuchmaschine nach dem Einpflegen der Verbesserungsvorschläge

13 Zusammenfassung

Für die Entwicklung des Modellierwerkzeuges „GetriebeStudio“ waren neben guten Kenntnissen in einer Programmiersprache auch Kenntnisse in Softwareengineering, UML und XML, notwendig. Des Weiteren war für die Kommunikation mit den beteiligten Personen, dem Entwurf und der Entwicklung der Software Detailwissen aus dem Bereich Maschinenbau erforderlich. Dies betraf u.a. Wissen über die normgerechte Darstellung von Getriebeteilen, die Berechnung der Lagerkennwerte und der Wellendrehzahlen.

Für die Beschreibung der Anforderungen und den Entwurf der Abläufe sowie deren Dokumentation wurde die Modelliersprache UML (siehe [15]) eingesetzt. Als Modellierwerkzeug diente das UML-Tool UMLet (siehe [16]).

Der Austausch von Daten mittels XML bot eine interessante Alternative zum reinen Text beziehungsweise tabellenartigen Dokumenten.

Als Entwicklungsumgebung wurde Eclipse (siehe [39]) eingesetzt. Diese Software bietet neben dem reinen Codieren (Syntaxfehler werden angezeigt, Soforthilfe wird angeboten, Sourcecode wird automatisch kompiliert), Unterstützung unter anderem beim Debuggen, Testen und Warten des Programmes.

Aus Zeit- und Budgetgründen wurde auf den Einsatz von speziellen Testtools zum Testen der GUI verzichtet. Die Java-Klassen wurden mittels Unit-Test getestet und das Gesamtsystem bzw. Teile davon nach jedem größeren Entwicklungsschritt zuerst vom Entwickler und parallel dazu von einer weiteren Person, die nicht Entwickler ist, getestet. Die Software wurde nach jedem größeren Entwicklungsschritt einem ausgewählten Personenkreis präsentiert und zu Versuchszwecken zur Verfügung gestellt. Änderungswünsche wurden mit den betroffenen Personen diskutiert und nach Möglichkeit eingepflegt.

Die Entwicklungsdokumentation erfolgte hauptsächlich durch UML-Diagramme und als Kommentar im Programmcode. Ein Tutorial wird demnächst den Anwendern zur Verfügung gestellt, ein Wiki ist derzeit nicht geplant.

Anhang

Drehzahlberechnung eines Planetengetriebes

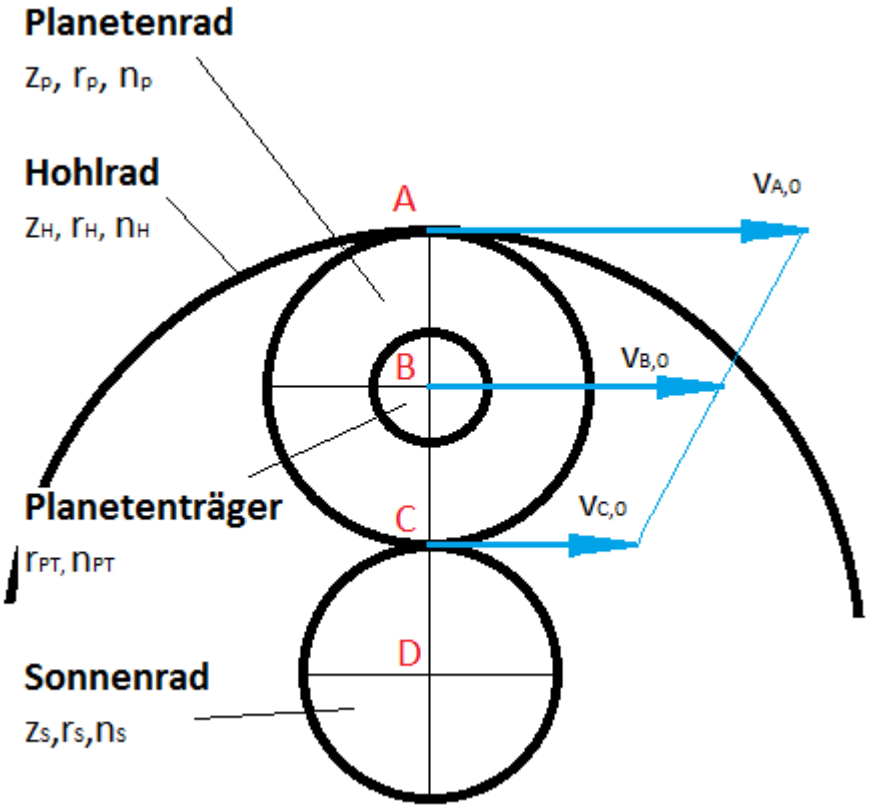

Abbildung 83: Schematische Darstellung eines Planetengetriebes

Variablen:

z … Zähnezahl des Zahnrades

r … Teilkreisradius des Zahnrades

n … Drehzahl des Zahnrades in U/min

Geg.: Planetengetriebe

Ges.: Zusammenhang zwischen der Drehzahl des Hohlrades, des Sonnenrades, des Planetenträgers und des Planeten gegenüber dem Inertialsystem 0.

Mögliche Fälle:

1. Allgemeiner Fall (zur Ableitung der Fälle zwei bis sieben)
2. Hohlrad fix
3. Planetenträger fix
4. Sonnenrad fix
5. Hohlrad getrieben (Planetenträger und Sonnenrad treibend)
6. Planetenträger getrieben (Hohlrad und Sonnenrad treibend)
7. Sonnenrad getrieben (Hohlrad und Planetenträger treibend)

Im Folgenden soll hier nur der allgemeine Fall betrachtet werden. Die Fälle zwei bis sieben lassen sich aus diesem jedoch einfach ableiten.

Allgemeiner Fall

Alle Teile drehen sich im, oder gegen, den Uhrzeigersinn um das Inertialsystem 0 (Punkt D). Für die weitere Berechnung wird angenommen, dass die Zahnräder und der Planetenträger sich im Uhrzeigersinn drehen. Die Punkte A, B und C bewegen sich mit der Geschwindigkeit $v_{A,0}$, $v_{B,0}$ und $v_{C,0}$ gegenüber dem ruhenden Inertialsystem.

I) $v_{A,0} = \omega_H * \overline{AD} = \frac{n_H * \pi}{30} * \overline{AD}$

$v_{A,0} = \frac{n_H * \pi}{30} * (\overline{AC} + \overline{CD})$

$v_{A,0} = \frac{n_H * \pi}{30} * (r_S + 2 * r_P) = v_H$

v_H … Absolutgeschwindigkeit des Hohlrades im Punkt A gegenüber dem Inertialsystem 0

II) $v_{C,0} = V_S = r_S * \omega_S = r_S * (n_P * \pi/30)$

III) $v_{B,0} = (v_{A,0} + v_{C,0})/2 = v_{PT} = v_P$

v_{PT} … Absolutgeschwindigkeit des Planetenträgers im Punkt B gegenüber dem Inertialsystem 0

v_P … Absolutgeschwindigkeit des Planeten im Punkt B gegenüber dem Inertialsystem 0

IV) $v_P = v_S + \overline{BC} * \omega_P = v_S + r_p * \omega_p = v_S + r_p * (n_P * \pi/30)$

v_S … Absolutgeschwindigkeit des Sonnenrades im Punkt C gegenüber dem Inertialsystem 0

V) $v_P = v_H - \overline{BC} * \omega_P = v_H - r_p * \omega_p = v_H - r_p * (n_P * \pi/30)$

VI) $v_{PT} = \overline{BD} * \omega_{PT} = r_{PT} * \omega_{PT} = r_{PT} * \frac{n\mathrm{PT} * \pi}{30}$

$v_{PT} = (r_P + r_S) * \frac{n\mathrm{PT} * \pi}{30}$

VII) $v_{A,0} = v_{C,0} + 2 * r_p * \omega_P$

aus I und II in VII folgt der gesuchte allgemeine Zusammenhang zwischen den Drehzahlen der Komponenten des Planetengetriebes:

$n_p = (n_H * (r_S + 2r_p) - n_S * r_S)/(2 * r_p)$

GUI-Test-Tools

Open-source-GUI-Tools

[vgl. http://en.wikipedia.org/wiki/List_of_GUI_testing_tools, 23.01.2014; http://blog.dreamcss.com/tools/gui-testing-tools/, 28.01.2014]

Tool Name Firma Homepage	Beschreibung	Einsatzbereich	Anforderungen an das Testsys-tem	Anforderungen an das zu testende System
Abbot Java GUI Test Framework	Automatisierte Eventgenerierung und Validierung von GUI Komponenten Ermöglicht: Record & Playback, verwendet Scripting als auch kompilierten Code Freeware	Zum Testen von Java AWT, Java SWT und Java Swing basie-renden GUIs	-	-
AutoHotkey	Free, Open Source Makroerzeugungs- und Automatisierungstool	z.B.: Remapping des Keyboards, automati-sieren repetitiver Auf-gaben, etc.	Microsoft Windows	-
Autotest.net	GUI Testplattform	-	-	-
CubicTest	Grafisches Eclipse Plug-In zum Schreiben von Selenium und Watir-Tests	Functional Web Testing	Eclipse	Selenium and Watir
Dogtail **Red Hat** **Fedora hosted.org**	Open source GUI Testing Tool, bietet easy-to-use framework, verwendet AT-SPI	Automatisieren von GUI-Tests	Verwendet Pyton, Linux	-
Dojo Objective Harness **Dojo Founda-tion**	Open source modular JavaScript Library, Rapid development of cross-platforms	Zum Testen von JavaScript/ Ajax-basierende An-wendungen und Web-seiten	-	Web, für verschiedene Browser geeignet
FitNesse Fitnesse.org	Java-basierendes Testframework und Collaborationtool Tests sind Data-driven	Akzeptanz-/ Integrationstest	-	Java, C++, .NET, Perl, Smalltalk, Ruby, etc.
Frankenstein Openqa.org	Tool zum Testen von Java Swing-Applikationen	Funktionale Tests	JDK 1.5 (oder dar-über), Apache Ant	-
GTT – GUI Testing Tool	GUI Testing Tool	Zum Testen von Java Swing basierten Appli-kationen	-	-
loadUI	Freeware	Lasttest Tool	-	-

Jemmy jemmy.java.net	GUI Testing Tool, Java Library	-	Netbeans, Swing, SWT, JavaFX	Java Anwendung
Linux Desktop Testing Project	Open source Testing Tools, verwendet Libraries zum Triggern von GUI-Widgets, basiert auf Assistive technology	GNOME-Anwendungen, Mozilla, OpenOffice.org, Swing-basierende Anwendungen, etc.	Linux	-
Maveryx maveryx.com	Testing Tool für Software Quality Assurence Teams, für traditionelle und agile Entwicklungsumgebung geeignet, nutzt intelligente Objekterkennung	Geeignet zum Testen in frühen Entwicklungsstadien, benötigt keine UI-Maps oder Test-Recording, unterstützt Data- und Keyword-driven Testing	Java Eclipse plugin	-
QAliber qaliber.net	GUI Test Automatisierungstool	Für agile Projekte geeignet, Web und Desktop Anwendungen	Microsoft Windows, C-sharp und VB.NET	Web, unterstützt nur Internet Explorer mit einer sichtbaren Seite
Tellurium automated testing framework	Automatisiertes Testframework auf Selenium basierend	-	-	-
Selenium Seleniumhq.org	Portables Software Testing Framework für Web Applikationen, Record & Playback Tool	Web Applikationen	Geeignet für Windows, Linux und Macintosh	Web, für verschiedene Browser geeignet
Sikuli sikuli.org	Automatisierungstool, verwendet Screenshots und Annotation zum Testen von GUIs Besteht aus mehreren Projekten: *SikuliScript*, *SikuliFramework* und *SikuliSlides*	Automatisieren und Testen von GUIs ohne Einschränkung auf bestimmte Plattformen Besonders geeignet für rapid Prototyping	-	-
SWTBot eclipse.org	GUI Testing Tool	Automatisieren von GUI-Tests	SWT-, Eclipse- und GEF-basierende Applikationen	Funktioniert auf allen Plattformen, auf denen SWT läuft
Robot Framework robotframework.org	Generisches Testautomatisierungswerkzeug für Akzeptanztests, key-driven Testing Framework	Akzeptanztests	Python, Java, .Net sind am besten geeignet. Andere Sprachen sind ebenfalls möglich	-
Watir watir.com	Ruby-basierendes GUI Testing Tool für Web-Applikationen. Besteht aus mehreren Projekten: *Watir-classic* (simuliert Browser, Object Linking and Embedding Capabilities), *Watir Web Driver* (auf Selenium basierend) und *Watirspec*	-	Für gängige Browser geeignet	Web
WatiN watin.org	Entwickelt zum Testen von .NET-Applikationen. Als Sprache dient C#	-	-	Internet-Explorer 6 bis 9 oder FireFox 2 und 3

Ranorex test automation toolset	Testautomatisierungstool für Desktop-, Web- und mobile Anwendungen Nutzt objektbasierten Capture & Replay-Editor Keine Freeware	Geeignet für HTML5, Firefox, .NET, Adobe, Google Chrome, Internet Explorer, Safari, Java, C#, Android, etc. Unterstützt Data-Driven (CSV, Excel, SQL) und Keyword-Driven Testautomatisierung	-	-
RIATest cogitek.com	GUI-Testautomatisierungstool für Webapplikationen Bietet: Recorder zum Aufzeichnen der Useraktivitäten und Erstellen von Scripts. Komponenten Inspektor für GUI-Elemente Keine Freeware	Zum Testen von Adobe Flex, Sencha ExtJS, HTML4, HTML5, JavaScript, jQuery und Windows 8 Store Apps basierenden Anwendungen geeignet	Windows und OS X	Flex
Samurai web testing framework	Linux-Umgebung vorkonfiguriert als Web Penetrationstestumgebung	Penetrationstests	-	-
Sahi	Tool zum Automatisieren von Web Applikationstests	JavaScript basierte Tests	-	-
SWAT – Simple web automation Toolkit	GUI Testing Tool zum Automatisieren von Webapplikationstests auf mehreren Browsern. Das Tool verfügt über einen Editor mit UI-Record-Funktionalität sowie SQL-Tools	-	-	-
Xnee xnee.wordpress.com	Speziell für X Window entwickeltes GUI Testing Tool	Zum Testen und Demonstrieren von X11 Applikationen	Unix	X Window

Proprietary GUI testing tools

[vgl. http://en.wikipedia.org/wiki/List_of_GUI_testing_tools, 23.01.2014]

Tool Name Firm Homepage	Beschreibung	Einsatzzweck	Anforderungen an das Testsys-tem	Anforderungen an das zu testen-de System
AutoIt AutoIt autoitscript. Com	Freeware automation language to create automation scripts for Microsoft Windows	Automatisierung von Windows GUI-Tests und Scripting	Microsoft Windows	Microsoft Windows
eggPlant **TestPlant Ltd** **testplant.com**	Blackbox GUI Automatisierungstool Keine Freeware	Zum Testen einer beliebigen Technolo-gie (C++, Flash, HTML5, etc.) auf einer beliebigen Plattform (Mainframe bis mobile Geräte)	Microsoft Windows, Linux, OS X	Microsoft Windows, Linux, OS X, iOS, Android, Blackberry, Win Embedded, Win CE VNC-Server benötigt
Froglogic's Squish froglogic.com	Objektbasierende Cross-Plattform, Cross-GUI-Technology Testing Tool Verwendet nicht Bilder-kennung und Bildver-gleich Record & Playback-Fähigkeit Unterstützt mehrere Scriptsprachen: Python, JavaScript, Ruby, Perl und Tcl Keine Freeware	Unterstützt Testauto–matisierung von auf folgenden Technolo-gien basierenden Anwendungen: Java, Swing, AWT, SWT, Web, Eclipse, RCP, WPF, Qt, QML, Flex, MFC, WinForms, etc.	-	Betriebssysteme: iOS, Android, Windows, Linux, Unix, OS X, WinCE, etc.
GUIdancer Bredex GmbH	Eclipse basierendes automatisiertes GUI Testing Tool Standalone und Eclipse plugin erhältlich Keine Freeware	Zum Testen von SWING-, SWT-, RCP-, GEF-, HTML-, .NET- und iOS- Anwendun-gen	Java (Swing, RCP/SWT, GEF), HTML, .NET and iOS GUI-test	Java (Swing, RCP/SWT, GEF), HTML, .NET and iOS GUI-test
HP QuickTest Pro-fessional (Nachfolger von HP WinRunner) Hewlett-Packard	Dient dem Automatisie-ren von Softwaretests; nutzt VBScript als Script-Sprache; erkennt Objekte der Anwendung und kann deren Eigen-schaften auslesen sowie gewünschte Operatio-nen (Mouse-Click, Key-board-Events) durch-führen	Für funktionale Tests und Regressionstests geeignet	Microsoft Windows	Microsoft Windows

IBM Rational Functional Tester **IBM**	Verwendet Record & Playback Mechanismus zum Erstellen von Skripten für Java oder .NET Applikationen Keine Freeware	Führt automatisierte Regressionstests aus; geeignet für .Net, Java, SAP, Ajax, etc.	-	-
IcuTest **site.icu-project.org**	Freies GUI Unit Testing Tool für Windows Presentation Foundation, kein Record & Playback System, vergleicht aktuell erstellte Snap-Shots mit Hinterlegten	-	Test Runner (z.B.: NUnit, MSTest, etc.) nötig; Programmiersprache C#	WPF
Infragistics infragistics.com	Test Automatisierungstool Keine Freeware	-	*HP Quick Test Professional, IBM Rational Functional Tester* oder *CodedUI for Microsoft Test Manager* werden vorausgesetzt	-
iMacros imacros.net	Macro Recorder Tool speziell für Web Browser Verwendet Bilderkennungstechnologie Keine Freeware	Web Scripting, Web Scraping, Internet Server Monitoring und Web Testing geeignet für HTML, Adobe Flash, Adobe Flex, Silverlight und Java-Applikationen	-	Unterstütze Browser: iMacros Browser, Internet Explorer, Firefox, Google Chrome (bisher nicht vollständig implementiert)
Microsoft Windows SDK Microsoft	Software Entwicklungsumgebung zum Entwickeln von Anwendungen für Microsoft Windows und .NET-Framework. Entwicklern als auch Testern werden verschiedene Werkzeuge z.B. zum Inspizieren der Eigenschaften in GUIs dargestellter Widgets zur Verfügung gestellt	Entwicklungsumgebung zum Entwickeln von Anwendungen für Microsoft Windows-Framework und .NET-Framework *UI Automation Verify* bietet Testern ein Framework zum manuellen und automatischen Testen von GUIs	Windows	-
Jubula Bredex GmbH Eclipse.org/ jubula	Automatisiertes, funktionales GUI Testen Freeware	Akzeptanztests (Blackbox-Test), agile Prozesse und frühes Testen Unterstützt Swing, SWT, RCP, GEF, HTML, .NET und iOS-Anwendungen	Windows, Linux/Unix, Mac	Windows, Linux/Unix, Mac

Microsoft Visual Studio Microsoft	Software Entwicklungsumgebung zum Entwickeln von Windows Forms, WPF, Web Applikationen und Services, Consol- und Graphical User Interfaces Auf Tester zugeschnittene Version verfügbar	Ab Professional Paket sind Unittest für GUI-Anwendungen verfügbar Im Premiumpaket ist zusätzlich Code-Coverage enthalten Das Ultimatepaket enthält weiters Lasttest-Funktionalität	-	-
QF-Test **Quality First Software** **qfs.de**	GUI-Testtool für Java Swing, Eclipse SWT und Web Anwendungen Capture & Replay-Fähigkeit Keine Freeware	Automatisiertes Testen (Cross-Plattform und Cross-Browser) von Java und Web Anwendungen mit GUI-Interface, Lasttests	Windows, Linux/Unix	Unterstützt nur 32-Bit Browser
Ranorex ranorex.com	Software für automatisierte Tests von Desktop-, Web- und mobilen Anwendungen Verwendet C# und VB.NET als Script Sprache Keine Freeware	Akzeptanztests, auto–matisierte Tests, Blackbox Tests, Funktionale Tests, GUI Tests, Web Tests, Test Mobiler Anwendungen, Data- und Keyword-Driven Tests, etc.	-	-
RIATest	GUI- Testautomatisierungstool für Flex, html, jQuery, ExtJS und Windows 8 Store Apps	-	Windows und OS X	Flex
SilkTest Micro Focus International	GUI Testing Tool Erkennt Windows und Controls Nutzt Bilderkennung und -vergleich Record & Playback Fähigkeit	Geeignet für automatisierte Funktions-, Regressions-, Unit- und Akzeptanztests Unterstützt .NET, Java Swing und SWT, DOM, IE, Firefox, SAP Windows GUI	Für verschiedene Entwicklungsumgebungen, Scriptsprachen (Java, VB.NET, C#, 4Test) und Browser geeignet.	Windows und Linux
Soatest Parasoft	Test- und Analysetool zum Testen von APIs und API-getriebenen Anwendungen	Zum Testen von Services, ESBs, Datenbanken, Mainframes, Web UIs, ERPs, etc. Geeignet für Unit-, Integration-, Regressions-, System-, Sicherheits- und Lasttests, Simulation und Mocking, Code Coverage, etc.	-	-

Telerik Test Studio Telerik telerik.com/teststudio	Testtool für automatisierte Tests Unabhängig vom Browser Record & Playback-Fähigkeit Nutzt Bilderkennung und -vergleich Keine Freeware	Unterstützt funktionales-, explorativen-, manuelles Testen, Last- und Leistungstests, Testen von mobilen Anwendungen (iOS, Android) HTML-, Ajax-, WPF- und Silverlight-Tests, Testen in Visual Studio	-	-
Test Automation FX **testautomationfx.com**	Windows GUI Test Automatisierungs-Framework für Visual Studio Record & Playback-Fähigkeit Test geschrieben/ aufgezeichnet in einfachem .NET Code (C# oder VB.Net)	-	Entwicklungsumgebung Visual Studio 2008/2010/ 2012/2013	-
TestComplete Smartbear smartbear.com	Tool zum automatisierten Testen von Software Tests können aufgezeichnet oder manuell erstellt werden (Scripting oder Verwendung von Keyword Operationen) Keine Freeware	Für Desktop, Web und mobile Anwendungen geeignet Unterstützt Web, Windows, WPF, HTML5, Flash, Flex, Silverlight, .Net, VCL, Java, C++, Delphi, Qt, Visual Basic, etc.	Nur für Microsoft Windows geeignet Unterstützte Browser: Microsoft Internet Explorer, Mozilla Firefox, Google Chrome, Opera, Apple Safari für Windows	Windows, Android und iOS
Testing Anywhere Automation Anywhere Automation anywhere.com/ Testing/	Testautomatisierungssoftware zum Aufzeichen, Debuggen, Planen und Ausführen von Tests Verfügt über einen Windows/Web-Recorder zum Aufzeichnen der User-Eingaben sowie Objekt- und Bilderkennung Keine Freeware	Geeignet für eine Vielzahl von Anwendungen basierend auf Java, Silverlight, .NET, mainframe, C++, etc. Unterstützt u.a.: Last-, Regressions-, Blackbox-, Keyword-driven-, Smoke-, Data-driven, GUI-Tests, etc.	Microsoft Windows	-

TestPartner **Micro Focus** **microfocus.com**	Funktionales Automatisieren und Testen von GUIs Speziell für Visual Basic for Applications (VBA) entwickelt Ergänzend zur codeorientierten Entwicklung von Tests kann auf eine Storyboard-basierende Testumgebung zugegriffen werden Record & Playback-Fähigkeit zum Aufzeichnen von User-Eingaben Keine Freeware	Unterstützte Technologien: Web, .NET, Java, SAP, Oracel, etc.	Entwicklungsumgebung VBA	-
Test Studio Telerik tele-rik.com/teststudio	Softwaretesttool für Windows-basierende Desktop-, Web- und mobile Anwendungen Als stand-alone oder als Visual Studio-Plugin verfügbar Keine Freeware	Geeignet für: Funktionales Testen von Web und Desktop-Anwendungen, Last-, Leistungs- und Regressionstests sowie dem Testen von mobilen Anwendungen (iOS, AppStore) Unterstütze Technologien: HTML5, AJAX, Silverlight, ASP.NET, WPF, MVC, etc.	Windows	Für verschiedene Browser geeignet (IE, Firefox, Chrome, Safari)
TOSCA Testsuite **TRICENTIS Technology & Consulting**	Testing Tool für automatisierte Regressionstests und funktionale Tests Für Business-Spezialisten/Nicht-Techniker geeignet Keine Freeware	Für manuelle/ automatisierte GUI/None GUI-Tests geeignet Last- und Stresstest sind möglich Unterstützte Technologien: Delphi, .Net, WPF, Java Swing/SWT/AWT, Visual Basic, Gupta, PowerBuilder, SAP, Siebel, Microsoft Outlook, Microsoft Excel, Flash, SOAP, ODBC	-	Unterstützte Browser: Internet Explorer, Firefox, Chrome
Twist Thoughtworks thoughtworks.com/products/twist-agile-testing	Tool für Testautomatisierung und funktionales Testen Nutzt Java oder Groovy zur Testimplementierung	Unterstützt Behaviour-Driven und Test-Driven Development (BDD & TDD) für funktionales Testen von Applikationen, agiles Testen	Optional: Selenium, Sahi für Web-Anwendungen. SWTBot für Eclipse SWT Applikationen. Frankenstein für Java Swing Anwendungen	-

Testautomatisierungstools

[vgl. http://en.wikipedia.org/wiki/List_of_GUI_testing_tools, 23.01.2014

Tool Name	Produced by	Latest version
AutoIt	Jonathan Bennett & AutoIt Team	3.3.8.1
Cucumber	Open Source	1.0.2
eggPlant	TestPlant	2012
EiffelStudio AutoTest	Eiffel Software	7.1, Jun 2012
FitNesse	Open Source	v20130530
HP QuickTest Professional	HP Software Division	11.5
IBM Rational Functional Tester	IBM Rational	8.3.0
LabVIEW	National Instruments	2013
Maveryx	Maveryx	1.3.1
Oracle Application Testing Suite	Oracle Corporation	12.1
QF-Test	Quality First Software GmbH	3.5.4
Ranorex	Ranorex GmbH	4.1
Rational robot	IBM Rational	2003
Robot Framework	Open Source	2.8.2
Robotium	Open Source	4.3
Selenium	Open source	2.39.0
Sikuli	Open Source	1.0.1
SilkTest	Borland	14.0
Soatest	Parasoft	9.5
TestComplete	SmartBear Software	9.1
Testing Anywhere	Automation Anywhere	8.0
TestPartner	Micro Focus	6.3
Test Studio	Telerik	2013.R1 SP1
Time Partition Testing (TPT)	PikeTec GmbH	4.2
TOSCA Testsuite	TRICENTIS Technology & Consulting	8.0.1
Visual Studio Test Professional	Microsoft	2012
Watir	Open Source	3.0

Literaturverzeichnis

[1] „duden.de,“ [Online], Available: http://www.duden.de/rechtschreibung/Getriebe, [Zugriff am 26. November 2013].

[2] „Wikipedia,“ [Online], Available: http://de.wikipedia.org/wiki/Getriebe, [Zugriff am 28. Juli 2014].

[3] W. Matek, D. Muhs, H. Witte und M. Becker, „Roloff/Matek Maschinenelemente,“ in *18., überarbeitete Auflage*, Braunschweig/Wiesbaden, Vieweg & Sohn Verlagsgesellschaft mBH, 2007.

[4] „http://de.wikipedia.org,“ Wikipedia, 5 Mai 2014. [Online], Available: http://de.wikipedia.org/wiki/Harmonic-Drive-Getriebe, [Zugriff am 25. Juli 2014].

[5] „harmonicdrive.de,“ Harmonic Drive AG, [Online], Available: http://harmonicdrive.de/startseite/, [Zugriff am 25. Juli 2014].

[6] „de.wikipedia.org,“ [Online], Available: http://de.wikipedia.org/wiki/Kegelradgetriebe#/media/File:Gear-kegelzahnrad.svg, [Zugriff am 21. April 2015].

[7] „freenet,“ [Online], Available: http://lexikon.freenet.de/Datei:Linkage_path.png, [Zugriff am 21. April 2015].

[8] L. Hagedorn, W. Thonfeld und A. Rankers, Kurvengetriebe, Springer Berlin Heidelberg, 2009.

[9] „de.wiki.org,“ [Online], Available: http://de.wikipedia.org/wiki/Ventilsteuerung, [Zugriff am 21. April 2015].

[10] „Wikipedia,“ [Online], Available: http://de.wikipedia.org/wiki/Kreisschubgetriebe, [Zugriff am 28. Juli 2014].

[11] K. Hoffmann, E. Krenn, Stanker und Gehard, „Fördertechnik,“ in *Fördertechnik, Band1, Bauelemente, ihre Konstruktion und Berechnung*, Wien München, R. Oldenburgverlag, 1997.

[12] A. Fischherz und R. Domayer, „Konstruktionslehre,“ in *Die Konstruktionsgerechte Zeichnung, 1. Teil Allgemeine Zeichengrundlagen*, Wien, Pichler Verlagsbuchhandel, 1993.

[13] „Fat Client,“ wikipedia, 24 3 2015. [Online], Available: http://de.wikipedia.org/wiki/Fat_Client, [Zugriff am 27. Mai 2015].

[14] „w3schools.com," [Online], Available: http://www.w3schools.com/schema/schema_intro.asp, [Zugriff am 10. Mai 2013].

[15] M. Hitz, G. Kappel, E. Kapsammer und W. Reschitzegger, in *UML@Work*, Heidelheim, dpunkt.verlag GmBH, 2005.

[16] „umlet.com," [Online], Available: http://www.umlet.com/, [Zugriff am 5. April 2013].

[17] T. L. Andreas Spillner, Basiswissen Softwaretest, dpunkt.verlag, 2010.

[18] „webtools3.skf.com," SKF, [Online], Available: http://webtools3.skf.com/BearingCalc/selectProduct.action, [Zugriff am 28. Mai 2013].

[19] „medias.schaeffler.de," Schaeffler, [Online], Available: http://medias.schaeffler.de/medias/de!hp.ec.br.pr2/;ayLcJzY-SdAe?mode=calc, [Zugriff am 28. Mai 2013].

[20] B. Schlecht, in *Maschinenelemente 2*, Pearson Studium, 2010.

[21] „maschinendiagnose.de," GfM Gesellschaft für Maschinendiagnose mbH, [Online], Available: http://www.maschinendiagnose.de/kompendium-getriebekinematik.html, [Zugriff am 20. März 2013].

[22] „systemdesign.ch," [Online], Available: http://www.systemdesign.ch/index.php?title=Massenmittelpunkt,_Kinematik, [Zugriff am 4. August 2014].

[23] „eclipse.org," [Online], Available: http://www.eclipse.org/windowbuilder/, [Zugriff am 16. Juni 2013].

[24] A. Spillner, T. Roßner, M. Winter und T. Linz, Praxiswissen Softwaretest, Heidelberg: dpunkt.verlag, 2014.

[25] „inside_it.ch," 16 Jänner 2013. [Online], Available: http://www.inside-it.ch/articles/31318, [Zugriff am 22. August 2014].

[26] T. Gechenig, M. Bernhart, R. Breiteneder und K. Kappel, in *Softwaretechnik - Mit Fallbeispielen aus realen Entwicklungsprojekten*, Pearson Studium, 2010, p. 375ff.

[27] „de.wikipedia.org," [Online], Available: http://de.wikipedia.org/wiki/Testfall, [Zugriff am 3. Jänner 2015].

[28] „imbus.de," imbus Akademie, [Online], Available: https://www.testtoolreview.de/de/, [Zugriff am 22. Mai 2015].

[29] M. Fewster und D. Graham, „Software Automation," in *Software Automation - Effective use of test execusion tools*, Addision-Wesley, 1999.

[30] „qse.ifs.tuwien.ac.at," [Online], Available: qse.ifs.tuwien.ac.at/course/sktiptum/download/10P_Nonfunkt_wid_20040204.pdf, [Zugriff am 1. März 2015].

[31] „Eclipse SWTbot," [Online], Available: http://eclipse.org/swtbot/, [Zugriff am 13. September 2012].

[32] „developers.google.com," [Online], Available: https://developers.google.com/java-dev-tools/wintesters/html/index, [Zugriff am 5. Jänner 2015].

[33] „qfs.de," [Online], Available: www.qfs.de/de/qftest, [Zugriff am 4. Jänner 2015].

[34] „qfs.de," [Online], Available: www.qfs.de, [Zugriff am 4. August 2013].

[35] „Wikipedia - GUIdancer," [Online], Available: de.wikipedia.org/wiki/GUIdancer, [Zugriff am 4. Jänner 2015].

[36] „Wikipedia - Testing," [Online], Available: https://wiki.eclipse.org/Eclipse/Testing, [Zugriff am 5. Jänner 2015].

[37] „bredex.de," [Online], Available: http://www.bredex.de/index.php/home_en.html, [Zugriff am 28. Juli 2013].

[38] „eclipse.org," [Online], Available: http://www.eclipse.org/tptp/test/documents/userguides/Intro-Auto-GUI.html, [Zugriff am 16. Juli 2013].

[39] „eclipse.org - Homepage," [Online], Available: http://www.eclipse.org/, [Zugriff am 3. September 2012].

[40] E. Quintus, in *Entwicklung eines hochsensitiven Zustandsüberwachungssystems für den Antriebsstrang von Windkraftanlagen*, Wien, TU Wien, 2014.

[41] „de.wikipedia.org," [Online], Available: http://de.wikipedia.org/wiki/User-Story, [Zugriff am 28. Dezember 2014].

Abbildungsverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
BDD	Behaviour Driven Development
DOM	Document Object Model
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
FFG	Österreichische Forschungsförderungsgesellschaft
GUI	Graphical User Interface
HTML	Hypertext Markup Language
JRE	Java Runtime Environment
ODBC	Open Database Connectivity
RGB	Red Green Blue
SAX	Simple API for XML
SPECTIVE	Software for Powerful Envelope analysis and Calculation of Traces for Immediate Visualization of Emergencies
SOAP	Simple Object Access Protocol
TDD	Test Driven Development
UI	User Interface
UML	Unified Modeling Language
VBA	Visual Basic for Applications
VDI	Verein Deutscher Ingenieure
VNC	Virtual Network Computing
WPF	Windows Presentation Foundation
XML	Extensible Markup Language
XSD	XML Schema Definition