Institut für Softwaretechnologie

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Refactoring und Erweiterung eines Programms zur Untersuchung des Leistungsverhaltens im Eisenbahnbetrieb

Matthias Gürtler

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Stefan Wagner
Zweitprüfer:	Prof. Dr.-Ing. Ullrich Martin
Betreuer:	Dr.-Ing. Xiaojun Li und Dr. Ivan Bogicevic
Beginn am:	1. April 2018
Beendet am:	1. Oktober 2018

Kurzfassung

In der rechnergestützten Leistungsuntersuchung des Eisenbahnbetriebs werden in der Engpassanalyse potentielle Engpässe der Infrastruktur gesucht, die bei höherer Belastung zu Behinderungen im Betrieb führen. Zur Ermittlung dieser Engpässe wurde durch das Institut für Eisenbahn- und Verkehrswesen ein Programm zur Untersuchung der Leistungsfähigkeit, kurz PULEIV, entwickelt. Diese Arbeit befasst sich mit der Anpassung und Erweiterung dieser Anwendung mit einem neuen Ansatz zur Bestimmung der Engpässe. Außerdem werden Defizite in der Softwarequalität analysiert und Lösungsmöglichkeiten zur Behebung dieser Defizite erarbeitet und teilweise umgesetzt. Das umfasst unter anderem die Auswertung und Dokumentationen von unkommentiertem Programmcode und bislang undokumentierten Schnittstellen zu anderen Anwendungen.

Abstract

The bottleneck analysis as part of the computer-aided performance analysis of railway operations deals with the search for potential infrastructure bottlenecks, which lead to disruptions in operation at higher loads. In order to identify these bottlenecks, the Institute of Railways and Transport has developed a software program called PULEIV to investigate performance. This thesis deals with the adaptation and extension of this application with a new approach to identify bottlenecks. In addition, deficits of software quality are analyzed and possible solutions for eliminating these deficits are developed and partially implemented. This includes the evaluation and documentation of uncommented program code and previously undocumented interfaces to other applications.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1. Einleitung

Motivation

In Zeiten, in denen wieder mehr Verkehr von der Straße auf die Schiene verlagert werden soll, ist es wichtig, dass die vorhandene Infrastruktur optimal ausgenutzt wird, um den Mehrverkehr bei geringen Mehrkosten abwickeln zu können. Zu diesem Zweck wird in der Verkehrsplanung die Leistungsfähigkeit der existierenden und projektierten Infrastruktur untersucht. Insbesondere wird in der Leistungsuntersuchung geprüft, wie groß die erträgliche Belastung der Infrastruktur sein kann, bevor es zu schwerwiegenden Engpässen kommt. In diversen Arbeiten des Instituts für Eisenbahn- und Verkehrswesen der Universität Stuttgart wurden Ansätze zur rechnergestützten Leistungsuntersuchung entwickelt und teilweise in eine Softwarelösung integriert: PULEIV. Ein neuer Ansatz zur Erkennung und Analyse von Engpässen soll im Rahmen dieser Arbeit in diese Bestandssoftware eingepflegt werden.

Software ist im Gegensatz zu Maschinen zwar ein imaterielles Gut, es unterliegt aber auch manchen Grundsätzen, die auf materielle Güter zutreffen. So endet die Arbeit an einer Software in den seltensten Fällen mit der Auslieferung, sondern es erfolgen auch danach noch Korrekturen, Anpassungen und Erweiterungen. Ein hohes Maß an Wartbarkeit erleichtert diese Arbeiten. Dazu gehört unter anderem die einfache Analysierbarkeit des Bestandscodes.

Die Anwendung PULEIV wurde und wird über die Jahre hinweg von vielen verschiedenen Entwicklern entworfen, bearbeitet, erweitert und angepasst. Deshalb ist auch bei PULEIV die Wartbarkeit des Codes sehr wichtig. Durch die vielen Anpassungen hat die Software diesbezüglich einige Defizite, die es auszugleichen gilt.

Problemstellung

Das ursprüngliche Ziel dieser Bachelorarbeit war, die softwaregestützte Umsetzung eines vorhandenen methodischen Ansatzes zur Engpassanalyse zu spezifizieren und in einem neuen Software-Modul in PULEIV zu implementieren. Dabei sollte zunächst der Ablauf des Ansatzes mit Hilfe des Modellierungswerkzeugs Enterprise Architect strukturiert werden. Danach war die Implementierung des Software-Modul zur Engpassanalyse mit der Programmierungssprache C# in PULEIV vorgesehen, um als Ergebnis die identifizierten Engpässe und deren Ursachen grafisch dargestellt zu können.

Im Zuge der Arbeiten an dieser Erweiterung wurde eine Umstrukturierung der Softwarebestandteile vorgeschlagen. Außerdem können die bei der Einarbeitung in den Sourcecode der Bestandssoftware aufgefallenen Defizite hinsichtlich der Softwarequalität durch Refactoring-Maßnahmen langfristig ausgeglichen werden. Deshalb sind nun die Strukturierung und die Verbesserung der Bestandssoftware neben der Erweiterung durch den neuen Algorithmus zwei weitere Schwerpunkte der Arbeit. Die vorliegende Arbeit besteht aus mehreren Abschnitten:

- Die theoretischen Grundlagen der Leistungsuntersuchung und insbesondere der Engpassanalyse werden in Kapitel 2 erläutert.
- Hintergründe zum Thema Softwarequalität, Wartbarkeit und Refactoring werden in Kapitel 3 betrachet.
- Kapitel 4 befasst sich mit der Bestandssoftware und bietet einen Überblick über die bestehende Funktionalität und den Aufbau der Anwendung.
- Kapitel 5 beschreibt die Softwarearchitektur von PULEIV und die für die Restrukturierung der Teilprojekte notwendigen Anpassungen der Software.
- In Kapitel 6 stehen die Defizite hinsichtlich der Softwarequalität und mögliche Maßnahmen zur Verbesserung der Wartbarkeit im Vordergrund. Es wird auch auf die vorgenommene Durchführung solcher Korrekturen und auf die Details und Probleme bei der Einrichtung einer Pipeline zur kontinuierlichen Integration (CI) eingegangen.
- Kapitel 7 enthält die Konzeption und die praktische Umsetzung des neuen Vier-Phasen-Ansatz in PULEIV.
- Kapitel 8 bietet einen Überblick über die Ergebnisse und einen Ausblick auf Möglichkeiten zur weiteren Verbesserung der Software.
- Die XML-Schnittstelle zum Simulationsprogramm RailSys ist im Anhang A erläutert.
- Weitere Codeausschnitte und Informationen zur Umsetzung des Vier-Phasen-Ansatzes befinden sich in Anhang B

2. Theoretische Grundlagen

Dieses Kapitel erläutert die dem Programm PULEIV zugrundeliegenden theoretischen Aspekte zur Leistungsuntersuchung im Schienenverkehr. Dabei wird auch auf verschiedene Begriffe und Kenngrößen eingegangen und Größen für den neuen Algorithmus werden eingeführt.

2.1. Was ist eine Leistungsuntersuchung?

In der Leistungsuntersuchung im Schienenverkehr werden Betriebsprogramme in Bezug auf gegebene Infrastrukturen betrachtet. Anhand der dabei berechneten Ergebnisse soll die Leistungsfähigkeit, Störanfälligkeit und Betriebsqualität des Zusammenspiels von Betriebsprogramm und Infrastruktur bewertet werden.

Pachl [1] unterscheidet in der Leistungsuntersuchung zwischen der analytischen Methode und der simulativen Methode. Bei der analytischen Methode werden anhand einer mathematischen Analyse Erwartungswerte für diverse Kenngrößen ermittelt. Dabei werden nicht einzelne Fahrpläne zugrunde gelegt, sondern stochastisch verteilte Störungen auf ein festes Betriebsprogramm angewendet.

Bei der simulativen Methode hingegen werden wiederholt Simulationen von Szenarien durchgeführt und die Ergebnisse durch den Vergleich zwischen den Simulationsdaten und den Plandaten ermittelt. Pachl unterteilt weiter in asynchrone und synchrone Simulationsverfahren. Die asynchrone Simulation betrachtet zunächst die einzelnen Fahrten als Ganzes. Kollisionen werden nach und nach durch Verschiebungen ausgeräumt. Im Gegensatz dazu führt die synchrone Simulation einen Zeitschritt für alle Fahrten aus, bevor der nächste Zeitschritt simuliert wird. Dieses Verfahren ermöglicht eine realitätsnähere Abbildung des Betriebsablaufs und wird deshalb in der Engpassananalyse zugrunde gelegt.

2.1.1. Wozu rechnerunterstützte Untersuchung?

Die Leistungsuntersuchung kann sowohl händisch, als auch (teil-)automatisiert am Rechner erfolgen. Mit steigender Komplexität der betrachteten Infrastrukturen und Betriebsprogramme nimmt der Aufwand der Untersuchung stark zu. Durch den Einsatz von Computern im Untersuchungsprozess kann der Aufwand für die Beteiligten vereinfacht werden. Zudem wird die Untersuchung beschleunigt und kann leichter unter gleichen Bedingungen wiederholt werden.

Neben der Vereinfachung bietet die rechnerunterstützte Untersuchung auch neue Möglichkeiten:

Einfaches Einspielen von Daten: Computerprogramme bieten standardisierte Schnittstellen zum Einspielen der Untersuchungsdaten. Das erhöht die Kompatibilität und Austauschbarkeit der Daten zwischen verschiedenen Anwendern.

Automatische Variation: Mittels Simulationswerkzeugen können aus Betriebsprogrammen viele verschiedene Varianten abgeleitet werden, die jeweils einzelne Facetten des Betriebsprogramms verstärken oder abschwächen. So kann beispielsweise die Zug- bzw. Taktfolge verdichtet werden.

Finden von Mustern: Durch die elektronische Erhebung der Ergebnisse ist es einfacher, Muster zu erkennen, die in der selben, aber auch in verschiedenen Untersuchungen auftreten.

Optimierung: Auf Basis der häufig auftretenden Muster können Optimierungsvorschläge erarbeitet werden, die situationsbezogen zur Verfügung stehen.

2.2. Begriffe und Kenngrößen in der Leistungsuntersuchung

2.2.1. Infrastrukturmodelle

Verschiedene Methoden in der Leistungsuntersuchung erfordern verschiedene (Daten-) Modelle für die Eisenbahninfrastruktur. Diese unterscheiden sich in der Granularität und folglich auch im Informationsgehalt.

- **Makroskopische Modelle** definieren die Infrastruktur grobgranular, denn es werden Betriebsstellen und Bahnhöfe als Ganzes betrachtet. Sie werden jeweils als Knoten aufgefasst und die Verbindungsstrecken als gerichtete Kanten zwischen zwei Knoten.
- **Mesoskopische Modelle** verfeinern die Betrachtung der Betriebsstellen und Bahnhöfe. Die Gleisanlagen eines Knotens werden in verschiedene funktionale Gruppen unterteilt: Fahrstraßenknoten und Gleisgruppen. Gleisgruppen bestehen aus mehreren Gleisen zwischen zwei Fahrstraßenknoten. Diese wiederum enthalten die Weichenbereiche, die verschiedene Gleise miteinander verbinden.
- Bei **mikroskopischen Modellen** werden die Infrastrukturbestandteile noch feiner gegliedert. So werden die Weichenbereiche z.B. in kleinere Abschnitte unterteilt oder Gleisabschnitte anhand von Signalen und Zugschlussstellen getrennt.

Der dieser Arbeit zugrundeliegende Ansatz zur detaillierten Engpassanalyse [2] erfordert ein eigenes mikroskopisches Modell, das auf zwei Ebenen basiert.

Fahrwegkomponenten sind gerichtete Elemente, die mögliche Belegungen richtungsbezogen in einem Graph modellieren. Die Definition nach Li lautet:

> „Eine Fahrwegkomponente (FK) ist das kleinste gerichtete Belegungselement auf einem Fahrweg, das gesondert aufgelöst werden kann (z. B. Fahrstraße oder Abschnitt einer Fahrstraße).“[2]

In Abbildung 2.1 ist eine beispielhafte Fahrwegkomponente blau hervorgehoben. Sie wird durch Signale (rote Marker) oder Zugschlussstellen (blaue Marker) begrenzt. Eine Fahrwegkomponente hat im Modell mehrere Eigenschaften:

- Startknoten
- Endknoten
- vorherige Fahrwegkomponenten
- nachfolgende Fahrwegkomponenten
- abhängige Fahrwegkomponenten

Knoten zur Unterteilung in verschiedene Fahrwegkomponenten können Signale und Zugschlussstellen [1] sein. Durch die Verkettung mit den vorherigen und nachfolgenden Fahrwegkomponenten lässt sich der Fahrweg eines Zuges durch die Infrastruktur abbilden. Über die abhängigen Fahrwegkomponenten lassen sich mehrere Komponenten zu Fahrstraßen zuordnen. Diese dürfen erst dann befahren werden, wenn alle ihre Komponenten unbelegt sind.

An einer Stelle der Infrastruktur können aber mehrere – auch gegenläufige – Fahrwegkomponenten vorhanden sein. Ein Engpass ist dann nicht nur auf eine der Fahrwegkomponenten beschränkt, sondern bezieht sich auf alle Fahrwegkomponenten an dieser Stelle. **Basisstrukturen** sind deshalb im Gegensatz zu den Fahrwegkomponenten ungerichtete Belegungselemente. Eine Basisstruktur ist also ein durch Signale und Zugschlussstellen voneinander abgegrenzter Bereich der Infrastruktur, in dem sich zu jedem Zeitpunkt maximal ein Zug aufhalten kann[2]. In Abbildung 2.1 ist eine Basisstruktur als Beispiel grün dargestellt.

Zusammengenommen ergeben diese beiden Ebenen das Zwei-Ebenen-Modell, auf dem die Berechnungen der Engpassanalyse basieren.

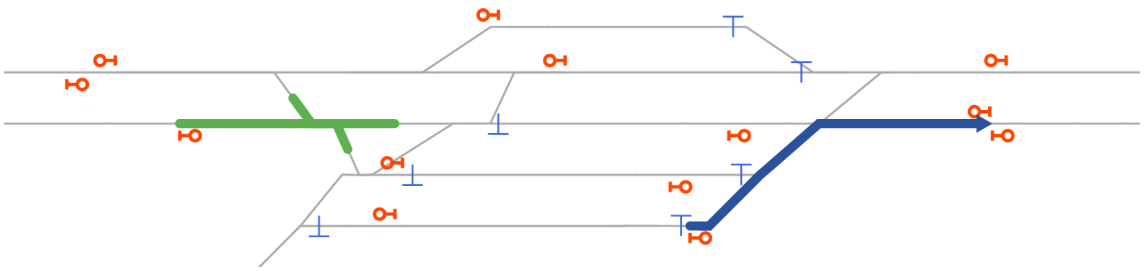

Abbildung 2.1.: Infrastruktur mit hervorgehobener Fahrwegkomponente (blau) und Basisstruktur (grün)

[1] Eine Zugschlussstelle ist eine Position, über die ein Zug komplett gefahren sein muss, bevor der dahinterliegende Abschnitt wieder freigegeben werden kann.

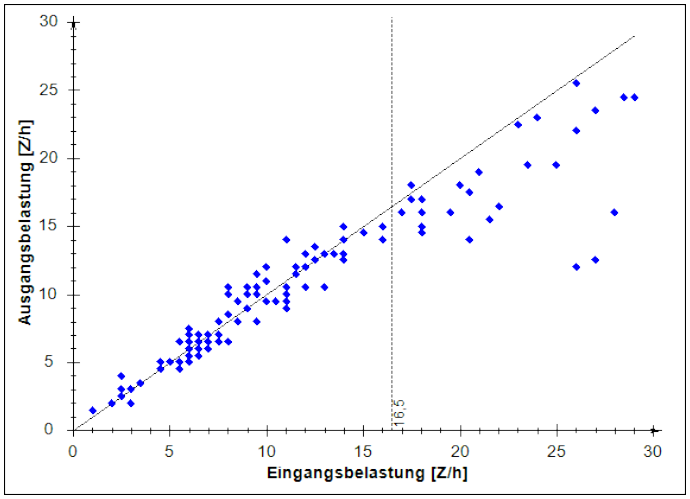

Abbildung 2.2.: Beispiel zur maximalen Leistungsfähigkeit (Quelle: PULEIV)

2.2.2. Kenngrößen der simulativen Methode

Zur Ermittlung der Engpässe werden verschiedene Kenngrößen benötigt, die aus den Simulationsergebnissen gewonnen werden können. Nach DB-Richtlinie 405 [3] werden Verzögerungerungen, die durch andere Züge verursacht werden, als **Wartezeit** aufgefasst. Der Quotient aus der Summe aller Wartezeiten und der untersuchten Gesamtzeit ist der **Behinderungsgrad**.

Maximale Leistungsfähigkeit

Unter der **maximalen Leistungsfähigkeit** versteht man die Belastung der Infrastruktur, ab der sich die Ausgangsbelastung [2] nicht mehr parallel zur Eingangsbelastung [3] verhält, sondern stagniert, geringer anwächst oder sogar wieder sinkt. Die maximale Leistungsfähigkeit wird deshalb in der Literatur auch als durchsatzbezogene Leistungsfähigkeit bezeichnet.

Der Wert kann über eine Annäherung anhand der Standardabweichung der Datenpunkte ermittelt werden. Sobald mehrere Datenpunkte den Toleranzbereich der Standardabweichung unterschreiten, ist von einer solchen Abweichung der Ausgangsbelastung auszugehen. Abbildung 2.2 veranschaulicht das Verhalten der beiden Belastungen an einem Beispiel. Im linken Sektor liegen die Datenpunkte alle entlang der Geraden mit Steigung 1. Im rechten Sektor liegen die Punkte teilweise weit unterhalb dieser Geraden. Die rechnerisch bestimmte maximale Leistungsfähigkeit liegt hier bei der durch die Trennlinie dargestellten Belastung von 16,5 Zügen pro Stunde.

[2] Die Ausgangsbelastung ist die Anzahl der Züge pro Stunde, die den betrachteten Infrastrukturabschnitt verlassen bzw. an ihrem Endziel innerhalb dieses Infrastrukturabschnitts ankommen.

[3] Die Eingangsbelastung ist die Anzahl der Züge pro Stunde, die in den betrachteten Infrastrukturabschnitt einfahren bzw. innerhalb dieses Infrastrukturabschnitts ihre Fahrt beginnen.

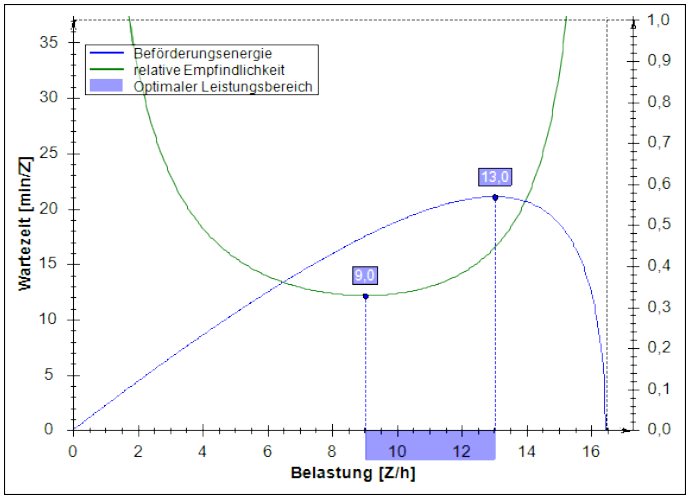

Abbildung 2.3.: Zusammenhang zwischen den Kenngrößen und dem Optimalen Leistungsbereich (Quelle: PULEIV)

Wartezeitfunktion, relative Empfindichkeit und Beförderungsenergie

Es lässt sich eine **Wartezeitfunktion** aufstellen, die die auftretenden Wartezeiten in Abhängigkeit von der Belastung der Infrastruktur darstellt. Detaillierte Modelle zur Annäherung dieser Wartezeitfunktion wurden von Christine Schmidt [4] und Zifu Chu [5] entwickelt.

Die **relative Empfindlichkeit** EMPF_{rel} in Abhängigkeit von der `Belastung` lässt sich anhand der `maximalen Leistungsfähigkeit` berechnen:

$$\text{EMPF}_{rel} = \frac{1}{\text{Belastung}} + \frac{b}{\text{max. LF}} * (1 - \frac{\text{Belastung}}{\text{max. LF}})^{-1}$$

wobei b ein Parameter der Wartezeitfunktion ist.

Die **Beförderungsenergie** EQ wird als das Produkt aus der Anzahl der Züge und dem Durchschnitt der Beförderungsgeschwindigkeit aufgefasst:

$$EQ = \text{Anzahl der Züge} * \text{durchschnittl. Beförderungsgeschw.}$$

Der Funktionsgraph der Beförderungsenergie beschreibt einen Parabelbogen, da jeder der beiden Faktoren einen Tiefpunkt bewirkt. Bei geringer Anzahl der Züge im System nähert sich das Produkt 0 an. Durch eine zu große Zugzahl wird aufgrund der Wartezeiten die durchschnittliche Beförderungsgeschwindigkeit reduziert. Auch dann nähert sich das Produkt 0 an.

Optimaler Leistungsbereich

Der Optimale Leistungsbereich ist der Auslastungsbereich zwischen minimaler Empfindlichkeit, also höchster Robustheit, und maximaler Beförderungsenergie, also höchstem Durchsatz. Der Zusammenhang zwischen den Kenngrößen ist in Abbildung 2.3 dargestellt. Der Optimale Leistungsbereich wird vom Tiefpunkt der relativen Empfindlichkeit (grün) und dem Hochpunkt der Beförderungsenergie (blau) begrenzt. Für die Leistungsuntersuchung spielt dieser Bereich eine große Rolle.

2.2.3. Engpässe

Es gibt keine allgemeingültige Definition für Engpässe in Bezug auf Schieneninfrastruktur. Engpass-Definitionen aus anderen Fachrichtungen verwenden die Begriffe Engpass und Flaschenhals synonym. Übertragen auf den Schienenverkehr ist ein Engpass entsprechend eine Komponente mit geringer Kapazität und hoher Auslastung. Die DB Netz AG spricht in der Richtlinie 405 [3] von Engpässen als den „maßgebenden Netzelementen für das Leistungsverhalten, [dessen] Nutzungsgrad der Nennleistung im mangelhaften Bereich der Qualität liegt“. Die Betrachtung der Auslastung alleine ist allerdings kein eindeutiges Kriterium für Engpässe, da auch Gleise mit abgestellten Zügen eine hohe Auslastung haben, aber nicht zwangsläufig zu Behinderungen anderer Zugfahrten führen. Andererseits können auch Abschnitte mit niedriger Auslastung Behinderungen an anderer Stelle bewirken.

Aufbauend auf diese Überlegung wurde in der Arbeit ”Methoden zur Engpassanalyse bei der Infrastrukturbemessung im Schienenverkehr”[6] eine erweiterte Definition aufgestellt:

> „Ein Infrastrukturabschnitt ist dann ein Engpass, wenn andere Fahrten wegen der Belegung auf diesem Infrastrukturabschnitt so stark beeinträchtigt werden, dass der Betrieb auf benachbarten Abschnitten behindert und damit die Betriebsqualität negativ beeinflusst wird, d.h. dieser Infrastrukturabschnitt betriebsbehindernd wirkt.“[6]

In der Engpassanalyse gilt es, mehrere Aspekte dieser Engpässe zu beleuchten:

- Lokalisierung der (möglicherweise relevanten) Engpässe
- Verdichtungsgrad, bis zu dem potentielle Engpässe keine schädigenden Folgen haben
- mögliche Gegenmaßnahmen zur Vermeidung von Engpassfolgen

2.2.4. Begriffe aus der Engpassanalyse

Zur Ermittlung von Engpässen wird die zu untersuchende Infrastruktur im Zusammenspiel mit einem Betriebsprogramm betrachtet. Unter Betriebsprogramm versteht man die geplanten Zugfahrten mit Relationen, Zuggattungen, Zuggeschwindigkeiten und Frequenzen. In verschiedenen **Verdichtungsstufen** wird dieses Betriebsprogramm verdichtet,

d.h. die Frequenzen schrittweise erhöht. Da Engpässe nicht bei jeder Belastung, sondern teilweise auch erst bei sehr hoher Verdichtung auftreten, werden einige Größen auf Basis der verschiedenen Verdichtungsstufen ermittelt.

Engpassrelevanz

Zur Ermittlung potentieller Engpässe wurde die Engpassrelevanz als Kenngröße eingeführt:

> „Die Engpassrelevanz beschreibt die Wahrscheinlichkeit, dass ein Infrastrukturabschnitt als Engpass unter bestimmten Bedingungen (Struktur des Betriebsprogramms) in Erscheinung tritt und verdeutlicht somit das Engpasspotential innerhalb eines Untersuchungsraums bei Anwendung eines Betriebsprogramms."[6]

Je sensibler der entsprechende Bereich der Infrastruktur auf ansteigende Belastungen reagiert, desto höher ist die Relevanz eines Engpasses. Die Betrachtung erfolgt dabei über alle Verdichtungsstufen hinweg.

Engpasssignifikanz

Im Gegensatz zur Relevanz bezieht sich die Engpasssignifikanz auf eine konkrete Verdichtungsstufe des Betriebsprogramms. Sie gibt an, ob der betrachtete potentielle Engpass bei dieser Verdichtungsstufe tatsächlich zu Behinderungen führt oder nicht. Bei sehr hohen Verdichtungen können auch nicht als relevant eingestufte Bereiche eine Engpasssignifikanz aufweisen.

Nicht erfüllbare Belegungswünsche

Für die Engpassanalyse mit dem Zwei-Ebenen-Modell wurde eine neue Kenngröße definiert, die die auftretenden Behinderungen an einer Basisstruktur beschreibt:

> „Die Nicht erfüllbaren Belegungswünsche (NEB) eines Belegungselements entsprechen der Summe der behinderungsbedingten Wartezeiten aller Züge, die dieses Belegungselement anfordern können und werden in einer geeigneten zeitlichen Einheit gemessen."[7]

Engpassempfindlichkeit

Die Engpassenpfindlichkeit ist die Wachstumsrate des Behinderungsgrades in Abhängigkeit vom Belegungsgrad einer Basisstruktur. Zur Ermittlung der Engpassempfindlichkeit wird anhand von Datenpunkten (x_i, y_i) mit dem Belegungsgrad x_i und dem Behinderungsgrad y_i eine lineare Funktion modelliert, deren Steigung der Engpassempfindlichkeit entspricht. Sie ist somit eine Eigenschaft, die sich auf das gesamte Betriebsprogramm und nicht nur einzelnen Verdichtungsstufen bezieht.

Maßgebender Engpass

In einer Infrastruktur kann es zugleich mehrere Engpässe geben, die einen Einfluss auf die gesamte Betriebsqualität haben. Im Rahmen der Engpassanalyse soll unter anderem festgestellt werden, welcher der Engpässe die schwerwiegendsten Auswirkungen verursacht. Dieser **maßgebende Engpass** ist ein Engpass, der bei Steigerung der Eingangsbelastung nicht mehr Züge bedienen kann, also gesättigt ist. Maßnahmen an diesem Engpass haben einen erhöhten positiven Effekt auf den gesamten Untersuchungsraum.

2.3. Vier-Phasen-Ansatz

Der neue Ansatz nach Li [2] zur Untersuchung einzelner Basisstrukturen basiert auf der Entwicklung der Nicht erfüllbaren Belegungswünsche (NEB) in Abhängigkeit von den Verdichtungsstufen. Der Verlauf der NEBs ist in Abbildung 2.4 dargestellt. Dabei sind vier Phasen erkennbar, die durch den Anfang und das Ende des Optimalen Leistungsbereiches sowie durch die maximale Leistungsfähigkeit unterteilt sind:

- **Phase 1** enthält alle Verdichtungsstufen unterhalb des Optimalen Leistungsbereichs. In ihr ist die Belastung so gering, dass durch die Basisstruktur keine großen Einschränkungen bewirkt werden.
- **Phase 2** umfasst den Optimalen Leistungsbereich. In diesem Abschnitt ist der Zuwachs der NEB ein wenig höher als bei der ersten Phase und verhält sich ähnlich wie die Wartezeitfunktion.
- Die **Phase 3** schließt oberhalb des Optimalen Leistungsbereiches an und enthält die Verdichtungsstufen bis zur maximale Leistungsfähigkeit. Der Anstieg der NEB ist in ihr wesentlich größer als in den ersten beiden Phasen.
- Alle Verdichtungsstufen ab der durchsatzbezogenen Leistungsfähigkeit sind Teil der **Phase 4**. Wenn die Baisstruktur ein maßgeblicher Engpass ist, nehmen ihre Nicht erfüllbaren Belegungswünsche nicht mehr weiter zu, da ihre maximale Kapazität erreicht ist. Basisstrukturen, die noch Reserven aufweisen, haben noch einen Anstieg der NEBs zu verzeichnen.

In diesem Ansatz wird die Anzahl der Nicht erfüllbaren Belegungswünsche der jeweiligen Verdichtungsstufen als Maß für die Engpasssignifikanz genommen. Die auf das Betriebsprogramm bezogene Engpassrelevanz hingegen wird über eine neue Größe bestimmt. Es wird nicht die absolute Zahl der NEBs betrachtet, sondern ihr Anstieg innerhalb der vier Phasen:

> „Definition: Die **NEB-Zuwachsrate** (**NZR**) eines Belegungselements bezeichnet die Änderung der Nicht erfüllbaren Belegungswünsche in Abhängigkeit von den Verdichtungsstufen (Belastungen) des gesamten Untersuchungsraums.“[2]

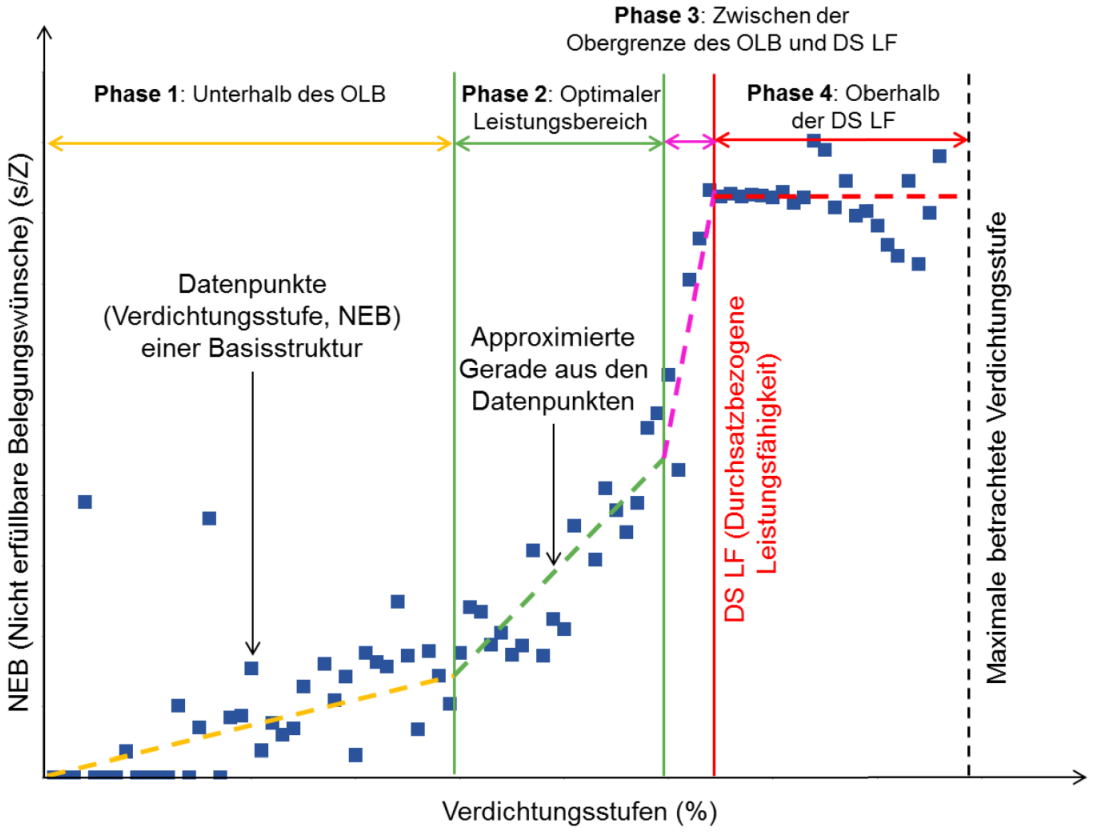

Abbildung 2.4.: Einteilung der vier Phasen in der Entwicklung der Nicht erfüllbaren Belegungswünsche (Quelle: [2])

Sie wird analog zur Engpassempfindlichkeit von Fahrwegkomponenten durch das Modellieren einer Funktion mit den Datenpunkten (*Verdichtungsstufe*, *NEB*) ermittelt. Anhand der ermittelten Kenngrößen können die Belegungselemente dann bewertet werden.

Zusammenfassung

In diesem Kapitel wurden die Grundlagen der rechnergestützten Leistungsuntersuchung und Engpassanalyse und die dabei benötigten Kenngrößen erläutert. Die Belastung einer Infrastruktur wird in der simulativen Analyse anhand vieler verschiedener Verdichtungsstufen bestimmt. Einen Überblick über die diesbezüglichen Funktionen in PULEIV bietet Kapitel 4. Die programmseitige Umsetzung des Vier-Phasen-Ansatzes ist in Kapitel 7 beschrieben.

3. Hintergründe zur Softwarequalität

In diesem Kapitel werden die Grundlagen der Softwarequalität betrachtet. Diese spielt in der Softwareentwicklung eine große Rolle, da sie sowohl durch den Entwicklungsprozess als auch durch das Produkt selber beinflusst wird. Sie setzt sich aus vielen verschiedenen Merkmalen zusammen, die sich kategorisieren und – nach Ludewig – in einen Qualitätenbaum einordnen lassen. [8]

Auch industrielle Normen kategorisieren die Qualitätsmerkmale. Die ISO-Norm 25010 [9] unterteilt in folgende Bereiche:

- Funktionale Software
- Verlässliche Software
- Performance
- Sicherheit
- Kompatibilität
- Usability
- Wartbarkeit
- Portierbarkeit

Hinsichtlich der Wartbarkeit listet die ISO-Norm verschiedene Kriterien auf:

- **Modularer Aufbau:** Die Anwendung ist nicht monolithisch aufgebaut, sondern besteht aus mehreren Modulen, die miteinander über Schnittstellen verknüpft sind.
- **Wiederverwendbare Komponenten:** Einzelne Bestandteile der Anwendung lassen sich ohne umfangreiche Anpassungen für andere Anwendungen verwenden.
- **Analysierbarkeit / Codeverständnis:** Der Code ist so geschrieben und dokumentiert, dass es unbeteiligten Entwicklern möglich ist, sich in den Quellcode einzuarbeiten.
- **Modifizierbarkeit:** Es ist mit akzeptablem Aufwand möglich, die Anwendung zu verändert oder zu erweitern.
- **Testbarkeit:** Die Bestandteile der Anwendung lassen sich testen.

Im Folgenden werden insbesondere die Aspekte zur Förderung des Codeverständnisses erläutert.

3.1. Dokumentation

Die Dokumentation eines Programms erfüllt verschiedene Zwecke. Eine der wichtigsten Aufgaben ist der dauerhafte Erhalt von Wissen. Da Mitarbeiter aus der Entwicklung ausscheiden können, ist es nicht ausreichend, wenn es Experten mit einer Wissensbasis gibt. Deshalb ist es auch wichtig, dass die Dokumentation gleich ab Beginn der Entwicklung parallel mitgeführt wird. Ludewig unterscheidet dabei zwischen [8]:

- **Integrierter Dokumentation:** Alle Informationen, die im Programmcode verfügbar sind. Das umfasst Zeilen- und Methodenkommentare sowie sprechende Bezeichner von Methoden und Variablen
- **Separater Dokumentation:** Dokumente, die nicht im Programmcode eingebettet sind. Darunter fallen unter anderem Spezifikationsdokumente und Handbücher.

Letztere wird häufig vernachlässigt und veraltet schnell, wenn sie nicht konsequent bei Programmänderungen mitgeführt wird. Sie sollte vor allem die Bestandteile enthalten, die durch die integrierte Dokumentation nicht abgebildet werden können, wie z.B. die Systemarchitektur. Auch digital in die Anwendung eingebundene Hilfe-Dokumente müssen aktuell gehalten werden.

3.1.1. Integrierte Dokumentation

Da die separate Dokumentation für das Verständnis des tatsächlichen Programmcodes meist nicht reicht, ist es wichtig, dass die Form und Funktion des Progammcodes leicht verständlich sind. Dazu ist es förderlich, wenn bei der Entwicklung für das Programm festgeschriebene Richtlinien eingehalten werden. Diese regeln z.B. die Wahl der Bezeichner, Anweisungen zur Kommentierung, quantitative Begrenzungen für Klassen, Methoden und Lines of Code. Aus der integrierten Dokumentation lassen sich auch durch Werkzeuge wie JavaDoc[10] und Doxygen[11] externe Dokumente extrahieren.

3.2. Codemerkmale

Auch die Struktur des Codes ist ein wichtiges Merkmal zur Unterstützung des Verständnisses. Richard Fairley [12] empfiehlt beispielsweise, dass:

- Entwickler Code nicht auf Kosten der Nachvollziehbarkeit überoptimieren sollen.
- Programmanweisungen nicht zu viele Verschachtelungsebenen haben sollten.
- Bezeichner nicht mehrfach für verschiedene Zwecke verwendet werden.
- Methoden nicht zu viele Parameter haben.

Diese Auflistung lässt sich durch viele weitere Empfehlungen anderer Autoren ergänzen. Negative Beispiele werden in der Literatur auch häufig als „code smells“ bezeichnet. Um diese nachträglich zu beseitigen, wird der Code einem sogenannten Refactoring unterzogen.

3.3. Refactoring

Der Begriff Refactoring wird von Martin Fowler folgendermaßen definiert:

> „A change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior.“[13]

Als Refactoring bezeichnet man also Änderungen am Code, die den Code aufwerten, ohne seinen Funktionsumfang zu ändern. Diese Anpassungen erfolgen zumeist in kleinen Schritten, um die Auswirkungen regelmäßig auf Fehlerfreiheit prüfen zu können.

Mögliche Maßnahmen zur Behebung typischer Schwächen werden in der Literatur oft beschrieben. Fowler unterscheidet dabei:

- Änderungen innerhalb von Methoden zur Reduzierung von redundantem Code und Komplexität
- Änderungen von Zuständigkeiten durch Verlagerung von Methoden
- Anpassungen der Datenstruktur zur Zusammenfassung zusammengehöriger Werte
- Vereinfachungen der Kontrollstruktur
- Anpassung der Vererbungshierarchie um Methoden und Variablen neu einzuordnen

Entsprechende Schwächen können in der Praxis durch den Einsatz von Refactoring-Tools oder Funktionen der Entwicklungsumgebungen teilautomatisiert behoben werden. Wichtig ist die entsprechende Anpassung der Dokumentation nach der Durchführung dieser Maßnahmen.

3.4. Kontinuierliche Integration

In der Softwareentwicklung ist neben dem Schreiben des Codes das erfolgreiche Kompilieren und Testen der Software essenziell. Im herkömmlichen Entwicklungsprozess (Abbildung 3.1) erfolgt dies alles manuell durch den Tester und ist sehr zeitaufwendig. Deshalb werden häufig viele Änderungen vollzogen, bevor die aufwendigen Tests für alle Änderungen zusammen durchgeführt werden. Auftretende Probleme sind dann nur schwer auf einzelne Änderungen zurückzuführen.

Dem gegenüber steht das Konzept der kontinuierlichen Integration (engl. Continous Integration CI), das Kent Beck 1998 als Teil des Extreme Programming [14] beschrieb. Bei der CI werden Änderungen in kurzen Intervallen zur Gesamtanwendung hinzugefügt und getestet. Dadurch bleiben die Auswirkungen der einzelnen Änderungen beherrschbar und Probleme lassen sich einfacher auf die individuellen Änderungen zurückführen.

Um dieses regelmäßige Kompilieren und Testen personalsparend umzusetzen, wird häufig eine CI-Pipeline eingesetzt. Dabei werden bestimmte Schritte automatisch durch einen Server ausgeführt, nachdem der Entwickler die Änderungen auf dem Server abgelegt hat.

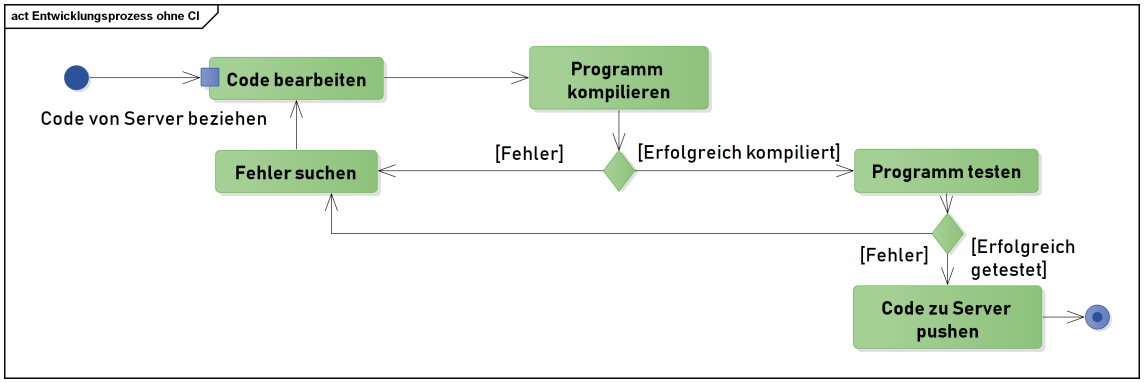

Abbildung 3.1.: Entwicklungsprozess ohne CI

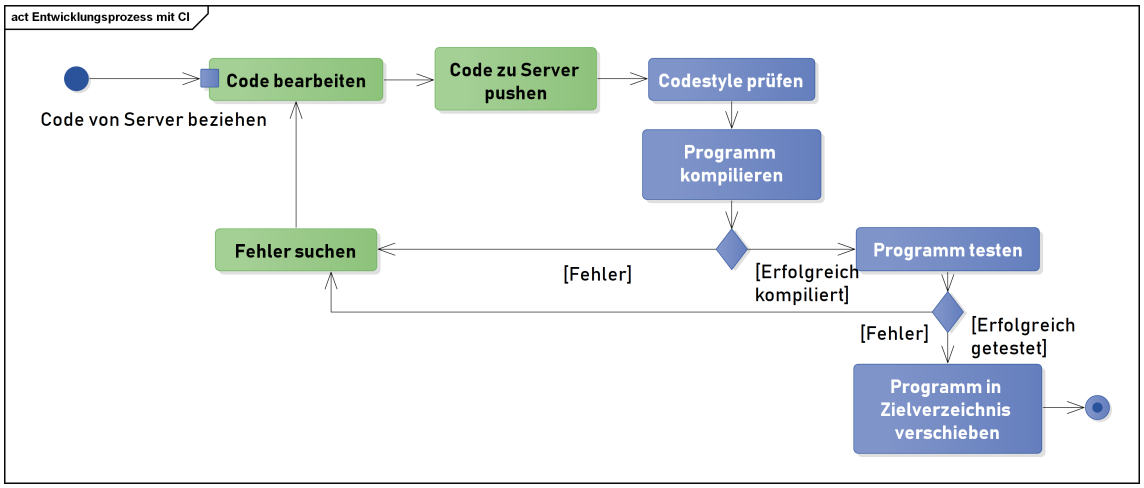

Abbildung 3.2.: Entwicklungsprozess mit CI und automatisierter Pipeline

Abbildung 3.2 stellt einen solchen Entwicklungsprozess mit den Schritten einer vereinfachten CI-Pipeline dar. Dabei werden die blau gefärbten Schritte automatisiert in der Pipeline ausgeführt. Der Entwickler muss weniger manuelle (grüne) Schritte selbst durchführen.

Nach Fowler [15] sind für die kontinuierliche Integration einige Grundsätze notwendig. Dazu zählt eine Codeverwaltung in einem gemeinsamen Repository, die nach regelmäßigen Commits die Anwendung automatisiert kompiliert und anschließend testet.

Zusammenfassung

Die in diesem Kapitel erläuterten Merkmale der Softwarequalität werden in Kapitel 6 in Bezug auf PULEIV betrachtet und mögliche Maßnahmen zur Verbesserung erläutert bzw durchgeführt. Auch die Umsetzung einer CI-Pipeline zur Vermeidung von Qualitätsdefiziten im Entwicklungsprozess von PULEIV wird betrachtet.

4. Einführung in die Software PULEIV

PULEIV ist ein **P**rogramm zur **U**ntersuchung des **Lei**stungs**v**erhaltens von Eisenbahninfrastrukturen und wird seit 2006 durch die VWI Stuttgart GmbH im Auftrag der DB Netz AG entwickelt. Im Rahmen diverser Projekte wurde die Funktionalität der Anwendung immer wieder erweitert, sodass nun ein größerer Funktionsumfang vorliegt:

- Generierung von Verdichtungsstufen auf Basis eines grundlegenden Fahrplans
- Auswertung von Simulationsergebnissen aus verschiedenen Simulationswerkzeugen
- Berechnung des optimalen Leistungsbereichs (vgl. Kapitel 2.2.2)
- Ermittlung von Engpässen in der Infrastruktur (vgl. Kapitel 2.2.3)
- Berechnung von Verspätungskoeffizienten
- Optimierung von Betriebsprogrammen

Die Anwendung wird unter anderem durch das Institut für Eisenbahn- und Verkehrswesen zur Untersuchung von Infrastrukturen eingesetzt.

4.1. Grundlagen

Bei PULEIV wird ein komponentenbasierter Ansatz verfolgt. Die Komponenten sind zu verschiedenen thematisch zusammenhängenden Teilprojekten gruppiert:

- Teilprojekt **Optimaler Leistungsbereich:** Dieses Teilprojekt bietet die Grundfunktionen rund um die Ermittlung des Optimalen Leistungsbereichs.
  - **Basisfahrplan:** Er enthält das grundlegende Betriebsprogramm, auf das aufgebaut wird.
  - **Fahrplanverdichtung:** Aufbauend auf den Basisfahrplan werden hier verdichtete Fahrpläne angelegt.
  - **Simulationsergebnisse:** Nach der Simulation in einer externen Anwendung werden hier die Simulationsergebnisse eingelesen.
  - **Optimaler Leistungsbereich:** Anhand der Simulationsergebnisse werden relevante Kenngrößen im Leistungsverhalten ermittelt.
  - **Transportimpulsdifferenz:** In dieser Komponente ist die Berechnung der Transportimpulsdifferenz nach Oetting [16] möglich.

  - **Engpassanalyse:** Die Engpassanalyse stellt die Wartezeiten und Belastungen in Bezug auf Betriebsstellen und Strecken dar.
  - **Knotenkapazität:** Diese Komponente visualisiert die Belastungen und Engpässe der einzelnen Belegungselemente.
- Teilprojekt **Detailanalyse:** Mit der Detailanalyse können einzelne Aspekte der Leistungsuntersuchung direkt untersucht werden:
  - **Simulationsergebnisse:** Nach der Simulation in einer externen Anwendung werden hier die Simulationsergebnisse eingelesen.
  - **Transportimpulsdifferenz:** In dieser Komponente ist die Berechnung der Transportimpulsdifferenz nach Oetting [16] möglich.
  - **Engpassanalyse:** Die Engpassanalyse stellt die Wartezeiten und Belastungen in Bezug auf Betriebsstellen und Strecken dar.
- Teilprojekt **Verspätungskoeffizienten:** In diesem Teilprojekt befindet sich nur die gleichnamige Komponente
  - **Verspätungskoeffizienten:** Diese Komponente analysiert das Verhältnis zwischen Eingangs- und Ausgangsverspätungen in der Infrastruktur.
- Teilprojekt **Optimierung Betriebsprogramm:** In diesem Teilprojekt werden die Auswirkungen von Störungen auf die Betriebsprogramme betrachtet.
  - **Eingangsfahrplan:** Er enthält das zu optimierende Betriebsprogramm.
  - **Trassenbündel:** In dieser Komponente werden die simulierten Störeinflüsse auf die Fahrplantrassen festgelegt.
  - **Optimierung Zeitreserven:** Die Fahrplandaten werden mit Pufferzeiten versehen, um Störeinflüsse auszugleichen.
  - **Fahrplananalyse:** Diese Komponente ermöglicht die Untersuchung der Verspätungskoeffizienten mehrerer Fahrpläne.
- **Vergleich:** Mit der Vergleichskomponente können verschiedene Teilprojekte anhand der jeweiligen optimalen Leistungsbereiche verglichen werden.

Abbildung 4.1 zeigt das Anwendungsfenster von PULEIV. Es ist in zwei Bereiche unterteilt, den Projektbaum auf der linken Seite und den Hauptbereich auf der rechten Seite, in dem die Oberfläche der gerade im Projektbaum selektierten Komponente angezeigt wird.

Der Projektbaum enthält alle Teilprojekte und deren untergeordnete Komponenten, die im aktuellen Projekt existieren. Ein solches Projekt kann entweder beim Programmstart bzw. über das Menü neu generiert werden oder aus einer gespeicherten Datei geladen werden. Ebenso können Projekte abgespeichert werden. Nähere Informationen zur Speicherung gibt es in Kapitel 5.4. Teilprojekte können nur als Ganzes erzeugt oder gelöscht werden, das Erstellen, Duplizieren oder Entfernen von einzelnen Komponenten ist nicht möglich.

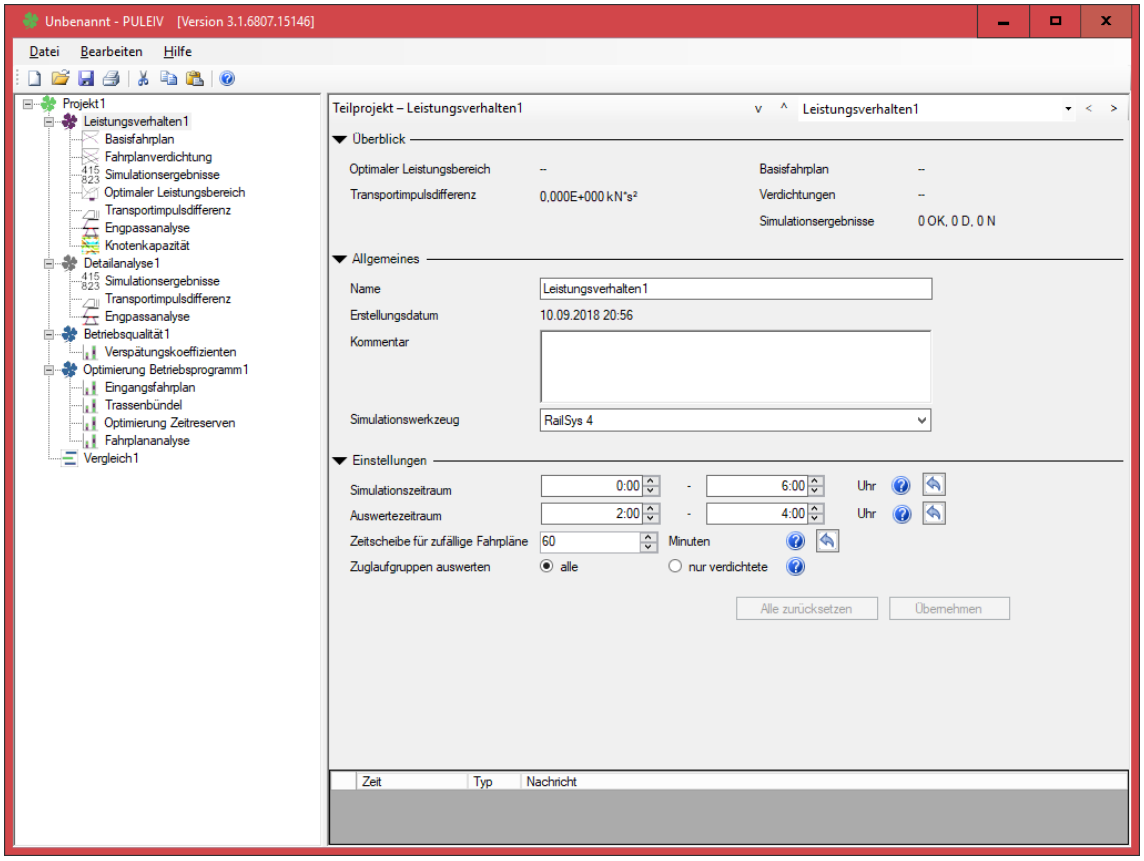

Abbildung 4.1.: Hauptansicht von PULEIV

4.2. Schnittstellen zu Simulationswerkzeugen

Die meisten Berechnungen erfolgen auf Basis einer gegebenen Infrastruktur mit passenden Fahrplänen. Die Simulation der Fahrpläne wird nicht durch PULEIV, sondern durch den Anwender im gewählten Simulationswerkzeug durchgeführt. Dafür greift PULEIV auf die Ergebnisse von Simulationswerkzeugen zurück, für die es entsprechende Datenschnittstellen zum Im- und Export anbietet. Zu den unterstützten Programmen zählen

- **RailSys:** Unter dem Namen RailSys bietet die Rail Management Consultants GmbH einen Verbund aus mehreren Anwendungen für eine Vielzahl von Aufgaben aus dem Bereich der technischen und betrieblichen Planung des Schienenverkehrs an. [17]
- **LUKS:** Die VIA Consulting GmbH, eine Ausgründung des Verkehrswissenschaftlichen Instituts der Universität Aachen, entwickelte LUKS für diverse analytische und simulative Anwendungen in der Infrastruktur- und Betriebsplanung. LUKS wird bei der DB Netz AG und der Infrastruktursparte der belgischen Staatsbahn, Infrabel, eingesetzt. [18]
- **PAULA-Z:** In Kooperation mit der DB Netz AG entwickelte die ASCI Systemhaus GmbH eine Anwendung zur statistischen Auswertung von Fahrplänen. [19]

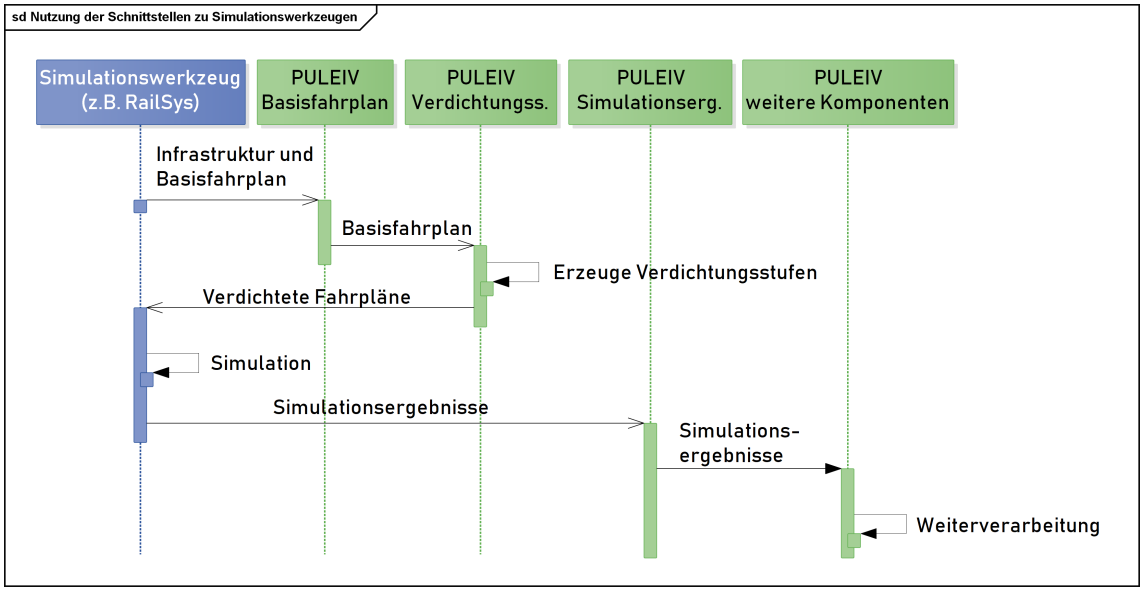

Abbildung 4.2.: Nutzung der Schnittstellen zu Simulationswerkzeugen

Für die meisten Untersuchungen mit PULEIV wird RailSys verwendet, die anderen beiden Werkzeuge waren in der Anfangszeit von PULEIV relevanter. Details zur Datenschnittstelle für RailSys sind in Anhang A aufgeführt.

Abbildung 4.2 zeigt das Zusammenspiel zwischen PULEIV und dem jeweiligen Simulationswerkzeug. Die Infrastruktur- und Basisfahrplandaten werden zunächst aus den vom Simulationswerkzeug exportierten Daten gelesen. Aufbauend auf die Struktur des Betriebsprogramms des Basisfahrplans können dann Verdichtungsstufen generiert werden, die die gleichen Zugarten und Fahrwege verwenden, aber eine höhere oder niedrigere Belastung aufweisen. Das Verhältnis der verschiedenen Zugarten zueinander bleibt aber unverändert. Deshalb spricht man bei den Verdichtungsvarianten vom gleichen Betriebsprogramm.

Nach der Generierung der Verdichtungsstufen können diese Dateien in das Simulationswerkzeug importiert werden. Dort erfolgt dann die Simulation der neuen Fahrpläne. Die Ergebnisse dieser Simulationen werden ebenfalls in PULEIV eingelesen. Auf ihrer Basis erfolgen dann im Anschluss die Berechnungen für die weiteren Teile von PULEIV, also unter anderem der optimale Leistungsbereich und die Engpassanalyse. Diese werden in Kapitel 2 bzw. 7 näher erläutert.

5. Restrukturierung der Teilprojekte

Wie in Kapitel 4 beschrieben, besteht PULEIV aus verschiedenen Teilprojekten und untergeordneten Komponenten. Im Zuge der Erweiterung der mikroskopischen Engpassanalyse in Abschnitt 7 sollen auch Teile der bestehenden Struktur angepasst werden. Zum Verständnis des Aufbaus ist eine Betrachtung der in den Spezifikationsdokumenten unzureichend beschriebenen Softwarearchitektur notwendig. Im folgenden Abschnitt wird diese erläutert.

5.1. Überblick über die Softwarearchitektur

In der Entwurfsphase von PULEIV wurde ein modularisierter Aufbau nach dem Model-View-Controller-Pattern konzipiert:

- **Model:** Als Datenmodell dienen bei PULEIV neben individuellen Klassen für einzelne Aspekte auch sogenannte Ablagen, die es anderen Komponenten über Getter- und Setter-Methoden ermöglichen, auf ausgewählte Daten der jeweiligen Komponente zuzugreifen. Daneben gibt es den Projektmanager, der die persistente Speicherung der Projektdaten verwaltet.
- **View:** Die Oberfläche des Gesamtprogramms wird von einer Instanz der `Hauptfenster`-Klasse verwaltet. Diese enthält auch jeweils eine Instanz der verschiedenen Komponenten mit ihren eigenen Oberflächen-Instanzen.
- **Controller:** Die Funktionsschicht wird neben Event-Listenern bei den Oberflächen vor allem über Komponenten [1] realisiert, die auf ihre zugeordneten Oberflächen und Modellklassen zugreifen.

Da die Teilprojekte jeweils unterschiedliche Anforderungen haben, gibt es pro Teilprojekttyp jeweils eine Komponentenklasse, die von einer generalisierten Klasse erbt. Zur Komponentenklasse gehört jeweils eine individuelle Oberflächenklasse. Der Zusammenhang der verschiedenen Bestandteile wird in Abbildung 5.1 vereinfacht dargestellt.

[1] Der Begriff Komponente ist im Kontext von PULEIV doppelt belegt. Zum einen werden die Bestandteile der Teilprojekte als Komponente bezeichnet, also z.B. die *Simulationsergebnisse* oder die *Engpassanalyse*. Zum Anderen gibt es Komponentenklassen, die pro Teilprojekt bzw. Komponente einmalig instanziiert werden und die Zuordnung der Daten vom aktuell selektierten Teilprojekt an die Oberfläche verwalten.

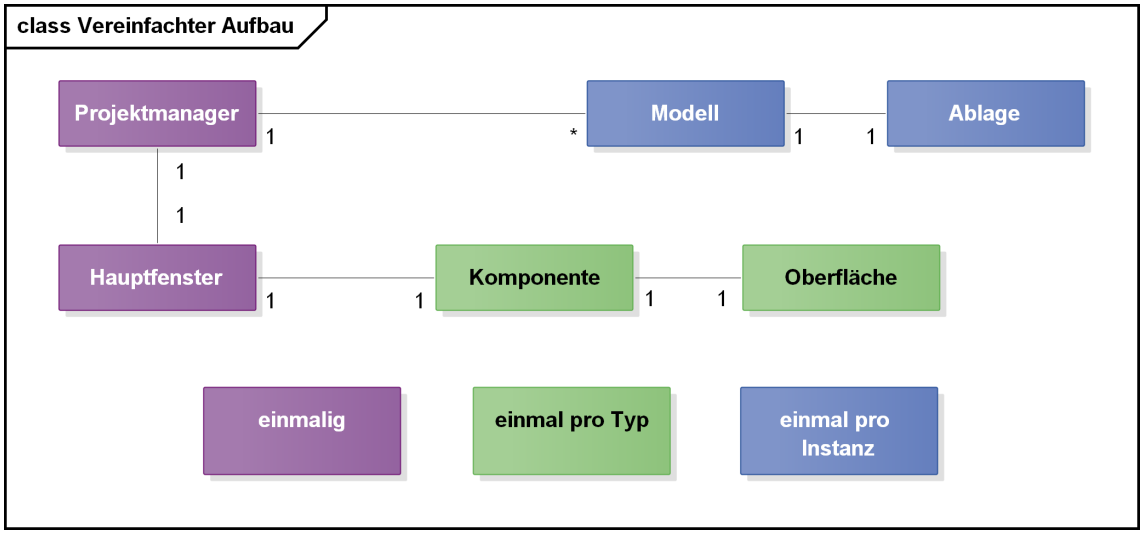

Abbildung 5.1.: Übersicht über die Architektur-Bestandteile von PULEIV

5.2. Konzept des neuen Teilprojekts Engpassanalyse

Für die makroskopische und mikroskopische Engpassanalyse soll ein neues Teilprojekt angelegt werden. Diese Komponenten benötigen die Simulations- und Berechnungsergebnisse aus dem Teilprojekt Leistungsverhalten [2]. Deshalb ist eine Verknüpfung mit einer konkreten Teilprojektinstanz des Typs Leistungsverhalten erforderlich. Der Anwender soll diese Verknüpfung über einen Auswahldialog konfigurieren können.

Analog zu den anderen Teilprojekttypen besteht das neue Teilprojekt Engpassanalyse aus

- Komponente
- Oberfläche
- Datenmodell
- Ablage

und verfügt über entsprechende Serialisierer und Deserialisierer für das Datenmodell und die Ablage. Das vereinfachte Klassendiagramm in Abbildung 5.2 stellt den Zusammenhang der Bestandteile dar.

Zur Auswahl der zu verwendenden Daten bietet das Teilprojekt Engpassanalyse eine Combobox an, die eine Liste der aktuell existierenden Teilprojekte des Typs Leistungsverhalten enthält. Die Aufhebung dieser Selektion ist über einen Zurücksetzen-Button möglich. Abbildung 5.3 zeigt die Anordnung der Komponenten.

[2] Bislang trägt das Teilprojekt den Namen `Optimaler Leistungsbereich`. Zur besseren Unterscheidung von der gleichnamigen Komponente dieses Teilprojekts wird das Teilprojekt zu `Leistungsverhalten` umbenannt.

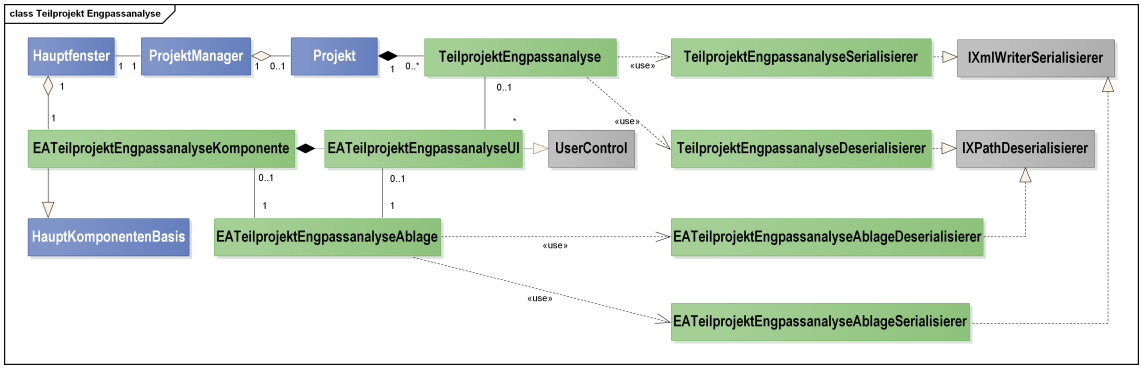

Abbildung 5.2.: Klassendiagramm des Teilprojekts Engpassanalyse

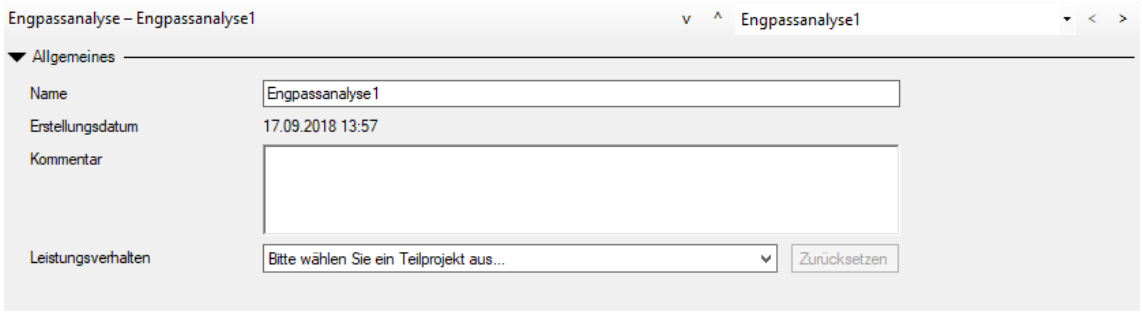

Abbildung 5.3.: Übersicht über die Einstellungen des Teilprojekts

Die aktuelle Auswahl wird in der Ablage des Teilprojekts hinterlegt und kann so von den Komponenten abgerufen werden. Dazu steht eine Getter-Methode bereit.

Die Komponenten des Teilprojekts benötigen zur Engpassanalyse einen Datenkontext mit Referenzen auf die Fahrplandaten des gewählten Teilprojekts Leistungsverhalten. Dazu gibt es die Methode `GetOLBKontext()`. Eine gleichnamige Methode gibt es auch im Bestandscode des Teilprojekts Leistungsverhalten. Wenn ein solches ausgewählt wurde, wird diese aufgerufen und das Ergebnis zurückgegeben. Wenn kein Teilprojekt Leistungsverhalten gewählt wurde, wird eine entsprechende Fehlermeldung in der Meldungskonsole von PULEIV ausgegeben:

„In den Einstellungen des Teilprojekt Engpassanalyse ist kein Teilprojekt selektiert!“

5.2.1. Funktionstests

Die Einführung des neuen Teilprojekts führt zu Anpassungen in vielen Bestandteilen von PULEIV. Zur Qualitätssicherung dieser Anpassungen ist die Durchführung von vielfältigen Tests notwendig.

Oberflächentests

Das Teilprojekt hat zwei eigene Oberflächenelemente, deren Verhalten geprüft werden muss. Tabelle 5.1 enthält manuelle Testfälle für die verschiedenen Zustände der beiden Elemente:

Anzahl LV[3]	ausgewählt	Text der Combobox	Combobox	Button
0	nein	`Bitte legen Sie ein Teilprojekt an...`	aktiviert	deaktiviert
1	nein	`Bitte wählen Sie ein Teilprojekt aus...`	aktiviert	deaktiviert
1	ja	Name des Teilprojekt	deaktiviert	aktiviert
>1	nein	`Bitte wählen Sie ein Teilprojekt aus...`	aktiviert	deaktiviert
>1	ja	Name des Teilprojekt	deaktiviert	aktiviert

Tabelle 5.1.: Testfälle für die Oberflächenelemente des Teilprojekts

Außerdem werden einige Eventlistener eingesetzt. Durch Tests muss sichergestellt werden, dass diese in den richtigen Situationen ausgelöst werden und entsprechende Funktionen aufgerufen werden:

- **Teilprojekt Leistungsverhalten ausgewählt:** Wenn ein Teilprojekt Leistungsverhalten ausgewählt wurde, muss der Name des gewählten Teilprojekt in der Combobox erscheinen, diese deaktiviert und der Zurücksetzen-Button aktiviert werden. Das Ergebnis der Auswahl muss in der Ablage hinterlegt sein.
- **Zurücksetzen-Button betätigt:** Nach Betätigung des Zurücksetzen-Button muss in der Combobox wieder `Bitte wählen Sie ein Teilprojekt aus...` erscheinen und sie muss wieder aktiviert sein. Der Zurücksetzen-Button muss deaktiviert sein. Außerdem muss das Teilprojekt Engpassanalyse zurückgesetzt werden.
- **Neues Teilprojekt Leistungsverhalten:** Nach dem Erstellen eines neuen Teilprojekt Leistungsverhalten muss dieses in der Auswahl der Combobox verfügbar sein. Der Zustand der Combobox darf sich aber nicht ändern.
- **Teilprojekt Leistungsverhalten entfernt:** Wenn ein Teilprojekt Leistungsverhalten entfernt wird, darf es in der Combobox nicht mehr auswählbar sein. Außerdem muss das Teilprojekt Engpassanalyse zurückgesetzt werden.

[3] Die Anzahl der verfügbaren Teilprojekte des Typs Leistungsverhalten

5.3. Komponenten des Teilprojekts

Unter dem Dach des neuen Teilprojekts sollen die makroskopische Engpassanalyse (vormals `Engpassanalyse`) und die mikroskopische Engpassanalyse (vormals `Knotenkapazität`) zusammengefasst sein. Die beiden Komponenten waren bislang im Teilprojekt Leistungsverhalten (vormals Teilprojekt Optimaler Leistungsbereich) enthalten. Dies ermöglichte ihnen einen direkten Zugriff auf die Simulationsergebnisse und Berechnungen dieses Teilprojekts. Aus dem Kontext des Teilprojekts Engpassanalyse ist ein direkter Zugriff nicht möglich, da es keine statische Zuordnung zwischen den Teilprojekten gibt.

Entsprechend muss der Zugriff angepasst werden, damit auf das in der Auswahl des eigenen Teilprojekts Engpassanalyse hinterlegte Teilprojekt Leistungsverhalten zugegriffen wird. Dazu wurde eine Hilfsfunktion entworfen, die unter Eingabe des aktuellen Teilprojekts Engpassanalyse dessen hinterlegte Auswahl ermittelt:

Listing 5.1 Hilfsfunktion zum Auslesen der Auswahl

```
1      private TeilprojektOptimalerLeistungsbereich
           FindeSelektiertesTeilprojektLV(TeilprojektEngpassanalyse teilprojekt)
2      {
3          var engpassanalyseAblage = (EATeilprojektEngpassanalyseAblage)teilprojekt
               .KomponentenDaten[KomponentenNamen.EATeilprojektEngpassanalyse];
4          return engpassanalyseAblage.selektiertesTeilprojektLV;
5      }
```

Weiterhin müssen die Stellen, an denen bislang geprüft wurde, ob ein übergebenes Teilprojekt dem Typ des eigenen Teilprojekts entspricht, geändert werden. Listing 5.2 zeigt ein Beispiel vor, Listing 5.3 dasselbe Beispiel nach der Anpassung [4].

Listing 5.2 Codeauszug vor der Anpassung an das neue Teilprojekt

```
1  private void ErstelleAblage(Teilprojekt tp)
2  {
3      if ((tp is StandardTeilprojekt)
4          && !tp.KomponentenDaten.ContainsKey(KomponentenNamen.RePlan))
5      {
6          tp.KomponentenDaten[KomponentenNamen.RePlan] =
7                  new RePlanAblage(
8                      new Bewerter(((StandardTeilprojekt)tp).GetOLBKontext()));
9      }
10 }
```

[4] Die Anpassungen, die im Rahmen der Verbesserung der Wartbarkeit vorgenommen wurden, sind ebenfalls enthalten.

Listing 5.3 Codeauszug nach der Anpassung an das neue Teilprojekt

```
1  private void ErstelleAblage(Teilprojekt tp)
2  {
3      if ((tp is TeilprojektEngpassanalyse)
4          && !tp.KomponentenDaten.ContainsKey(KomponentenNamen.EAMikroskopischeEngpassanalyse))
5      {
6          tp.KomponentenDaten[KomponentenNamen.EAMikroskopischeEngpassanalyse] =
7                  new RePlanAblage(
8                      new Bewerter(((TeilprojektEngpassanalyse)tp).GetOLBKontext()));
9      }
10 }
```

5.4. Speicher- und Ladefunktion

PULEIV kann Projektdaten persistent in xml-basierte Dateien speichern und diese wieder auslesen. Dazu gibt es für die Modell- und Ablageklassen (De-)Serialisierer, die die Objektstruktur in XML parsen bzw. aus den XML-Dateien auslesen. Kernbestandteil dieser Funktionalität ist dabei die Klasse `KomponentenNamen`, die die verwendeten Schlüssel für die verschiedenen Typen bereithält.

Aufgrund der strukturellen Änderungen und der in Kapitel 6 beschriebenen Umbenennungen ist es in PULEIV nicht mehr möglich, alte Projektdateien direkt einzulesen. Veraltete Schlüssel in den XML-basierten Daten und an anderen Stellen der XML-Struktur erwartete Knoten stellen für den Import mit den Deserialisierern ein Hindernis dar. Abbildung 5.4 bietet einen Überblick über die möglichen Varianten der Speicher- und Ladefunktionalität. Um ältere Projektdateien einlesen zu können, müssen also Übersetzungstabellen vorhanden sein, die bereits beim Einlesen die richtigen Objekte der neuen Struktur erstellen können. Auch die umgekehrte Richtung, also das Schreiben in die Dateien ist für ältere Dateiversionen nicht direkt möglich. Es sind angepasste Serialisierer notwendig, die die alten Schlüssel und den alten Aufbau der XML-Struktur verwenden.

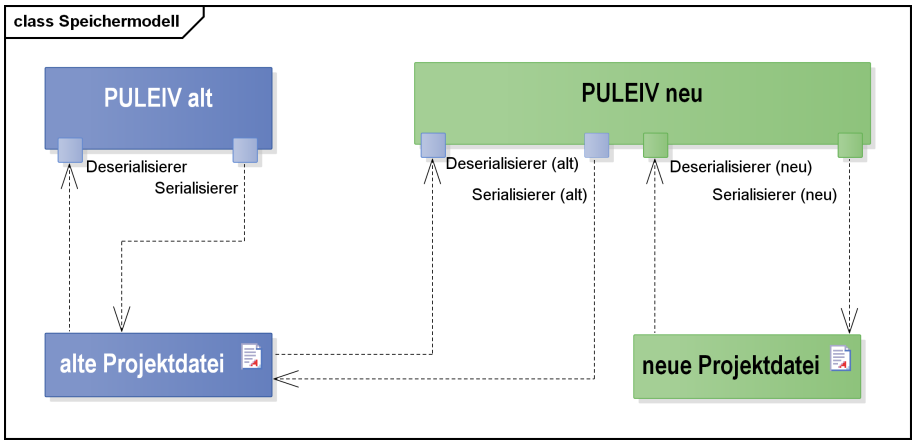

Abbildung 5.4.: Schematisches Speicher- und Ladeverhalten bei verschiedenen Programmversionen

6. Verbesserung der Softwarequalität

Die Grundlagen der Softwarequalität, wie sie in Kapitel 3 beschrieben sind, sollen auch bei PULEIV Anwendung finden. Bei der Einarbeitung in den Code von PULEIV sind vor allem hinsichtlich der Wartbarkeit einige Defizite aufgefallen. Insbesondere das Codeverständnis ist durch die nicht konsistente Wahl der Bezeichner und die unvollständige Kommentierung des Codes sehr erschwert. Durch Refactoringmaßnahmen und Ergänzung fehlender Kommentare sollen deshalb einige dieser Defizite behoben werden.

6.1. Fehlermeldungen in der Anwendung

Bei Fehlbedienungen des Programms gibt PULEIV nicht in jedem Fall eine Fehlermeldung aus. Teilweise zeigt die Anwendung gar keine Reaktion, was es Anwendern und Entwicklern erschwert, die Art des Fehlers einzugrenzen. Auch wenn eine Fehlermeldung in der Meldungskonsole erscheint, ist diese nicht immer aussagekräftig.

Wenn z.B. eine Projektdatei nicht geladen werden kann, erscheint die Fehlermeldung:

„Die Datei <Projektdatei> konnte nicht als Projektdatei geöffnet werden“

Es gibt unterschiedliche Ursachen für diesen Fehler, unter anderem:

- die Datei ist keine Projektdatei
- die Datei ist durch einen anderen Prozess blockiert, d.h. PULEIV erhält keinen Lesezugriff auf die Datei.
- die Datei enthält falsche Daten, also z.B. unbekannte Schlüssel

Mit präziseren Fehlermeldungen könnten Anwender und testende Entwickler die Ursache leichter nachvollziehen. An geeigneten Stellen wurden diese im Rahmen dieser Arbeit hinzugefügt.

6.2. Separate Dokumentation

Die Spezifikations- und Beschreibungsdokumente umfassen nicht alle Bestandteile der Software, da einige erst nachträglich zu PULEIV hinzugefügt wurden. Insbesondere die Softwarearchitektur ist in den bestehenden Dokumenten nur unzureichend erläutert. Neben der Softwarearchitektur wurde im Rahmen dieser Arbeit auch die im Anhang A

Bestandteil	Bezeichner
Datenablage	OpBPFahrplanbewertungAblage.cs
Komponente	BewertungFahrplanOpBP.cs
Oberfläche	OpBPBewertungUI.cs
Komponentenname	FahrplanbewertungOpBP
Kürzel[1]	PULEIV.FBOP

Tabelle 6.1.: Nicht einheitliche Klassenbezeichner in einem Beispiel

befindliche Schnittstellendokumentation erarbeitet. Es wird langfristig empfohlen, die Spezifikationsdokumente zumindest hinsichtlich des allgemeinen Konzepts der fehlenden Bestandteile zu ergänzen.

6.3. Integrierte Dokumentation

Wie in Kapitel 3 beschrieben ist, spielen auch Bezeichner und Kommentare im Code eine große Rolle beim Codeverständnis.

6.3.1. Benennung der Codebestandteile

In Kapitel 5 sind die Bestandteile der Teilprojekte kurz erläutert worden. Im Code ist der Zusammenhang zwischen den Bestandteilen jedoch nur schwer erkennbar, da die Klassen keine einheitlichen Bezeichner haben. Tabelle 6.1 enthält ein Beispiel aus dem Sourcecode von PULEIV, dass die uneinheitlichen Bezeichner verdeutlicht.

Diese Bezeichner haben verschiedene Mängel:

- Das Kürzel des Teilprojekts ist teilweise als Präfix vorangestellt oder teilweise als Suffix angehängt.
- Die Art des Bestandteils ist nur teilweise vorhanden (Ablage, UI).
- Der Bezeichnerstamm unterscheidet sich stark: Fahrplanbewertung, BewertungFahrplan, Bewertung

Um dies zu vereinheitlichen, wurde ein allgemeines Schema aufgestellt und entsprechende Anpassungen vorgenommen:

Präfix – Bezeichner – Typ

Das Beispiel nach der Vereinheitlichung ist in Tabelle 6.2 aufgelistet.

[1] Das Kürzel wird u.a. bei der (De-)Serialisierung der Daten verwendet.

Bestandteil	Bezeichner
Datenablage	OpBPFahrplanbewertungAblage.cs
Komponente	OpBPFahrplanbewertungKomponente.cs
Oberfläche	OpBPFahrplanbewertungUI.cs
Komponentenname	OpBPFahrplanbewertung
Kürzel	PULEIV.OPBPFB

Tabelle 6.2.: Einheitliche Klassenbezeichner im selben Beispiel

Alter Bezeichner	Neuer Bezeichner	Neuer Bezeichner im Code
Optimaler Leistungsbereich	**Leistungsverhalten**	**LVTeilprojektLeistungsverhalten**
Basisfahrplan	Basisfahrplan	LVBasisfahrplan
Fahrplanverdichtung	Fahrplanverdichtung	LVFahrplanverdichtung
Simulationsergebnisse	Simulationsergebnisse	LVSimulationsergebnisse
Optimaler Leistungsbereich	Optimaler Leistungsbereich	LVOptimalerLeistungsbereich
Transportimpulsdifferenz	Transportimpulsdifferenz	Transportimpulsdifferenz
	Teilprojekt Engpassanalyse	EATeilprojektEngpassanalyse
Engpassanalyse	Makroskopische Engpassanalyse	EAMakroskopischeEngpassanalyse
Knotenkapazität	Mikroskopische Engpassanalyse	EAMikroskopischeEngpassanalyse
Detailanalyse	**Detailanalyse**	**DATeilprojektDetailanalyse**
Simulationsergebnisse	Simulationsergebnisse	DASimulationsergebnisse
Verspätungskoeffizienten	**Betriebsqualität**	**BQTeilprojektBetriebsqualitaet**
Verspätungskoeffizienten	Verspätungskoeffizienten	BQVerspaetungskoeffizienten
Optimierung Betriebsprogramm	**Optimierung Betriebsprogramm**	**OPBPTeilprojektOptimierungBetriebsprogramm**
Eingangsfahrplan	Eingangsfahrplan	OPBPEingangsfahrplan
Trassenbündel	Trassenbündel	OPBPTrassenbuendel
Optimierung Zeitreserven	Optimierung Zeitreserven	OPBPOptimierungZeitreserven
Fahrplananalyse	Fahrplananalyse	OPBPFahrplananalyse
Vergleich	**Vergleich**	**Vergleich**

Tabelle 6.3.: Anpassung der Bezeichner von Teilprojekten und Komponenten

Auch die Benennung der Teilprojekte und Komponenten als Ganzes hat Verbesserungspotential. So gibt es z.B. das Teilprojekt Optimaler Leistungsbereich und seine gleichnamige Kompontente Optimaler Leistungsbereich. Im Code werden teilweise nochmals andere Bezeichungen verwendet. Zur besseren Unterscheidung werden auch hier einige Bezeichner angepasst. Tabelle 6.3 enthält die alten und neuen Namen.

6.3.2. Code-Kommentare

Neben den Benennungen ist die Kommentierung des Codes nicht optimal. Die Entwicklungsumgebung VisualStudio zeigt nach dem Öffnen des Projekts mehr als 1100 Warnungen aufgrund fehlender Kommentare an. Dies betrifft alleine die Methoden-Kommentare der nicht privaten Methoden. Darüber hinaus gibt es eine Vielzahl an privaten Methoden, die ebenfalls nicht kommentiert sind. In Verbindung mit nicht-sprechenden Variablennamen (z.B. `num1`) erschweren auch fehlende Zeilenkommentare das Codeverständnis. Im Rahmen dieser Arbeit wurden einige Codeabschnitte umfassend kommentiert, um das Verständnis des Codes langfristig zu verbessern. Die Extrahierung der Kommentierung mit Tools wie Doxygen[11] zu einer zusammenhängenden Bestandsaufnahme des Ist-Standes im Code wird empfohlen.

6.4. CI-Pipeline bei PULEIV

Zur Unterstützung und Entlastung der Softwareentwickler von PULEIV ist die Einrichtung einer CI-Pipeline vorteilhaft. Da in PULEIV viele Bestandteile miteinander verknüpft sind, ist ein erhöhtes Risiko auftretender Nebenwirkungen durch Änderungen am Code gegeben. In Bezug auf PULEIV ist deshalb vor allem das automatisierte Kompilieren und anschließende Testen von Interesse. Die automatisierten Tests sollen sicherstellen, dass keine unerwarteten Fehlerbilder auftreten.

An der Universität Stuttgart wird in einigen Projekten (z.B. SystemTestPortal [20]) das Versionsverwaltungssystem GitLab verwendet. GitLab kann auch auf eigener Hardware gehostet werden und bewahrt somit die Unabhängigkeit von Drittanbietern. Zugleich bietet GitLab auch eine integrierte CI-Lösung zur Umsetzung der Pipeline [21].

Zunächst soll die CI nach einem Commit die Anwendung automatisiert bauen. Im Anschluss sollen automatisierte Tests überprüfen, ob bei den letzten Änderungen bestehende Funktionen beeinträchtigt wurden. Dies erfordert eine Buildumgebung für .NET / C#.

Die von GitLab bereitgestellte CI-Lösung nutzt dazu Docker-Images auf Servern oder eingebundene lokale Rechner. Für .NET gibt es zwei verbreitete Docker-Images, die in Frage kommen:

- Das offizielle Image von Microsoft: microsoft/dotnet
- Die alternative quelloffene Implementierung: mono

Beide Docker-Images bieten prinzipiell eine Buildumgebung für .NET-Anwendungen. Allerdings benötigt PULEIV als Oberflächenanwendung die Unterstützung der Windows Forms Bibliotheken. Das offizielle Image von Microsoft ist noch im Aufbau und unterstützt bislang Windows Forms nicht. Den Dokumentationen zu Mono [22] ist zu entnehmen, dass Mono die Windows Forms 2.0 komplett unterstützt. In der Praxis scheitert der Buildprozess dann aber an einer alten Paketverwaltung, die im PULEIV-Projekt verwendet wurde und nicht kompatibel zu den neueren Buildumgebungen ist. Die letzte Aktualisierung des OpenWrap-Paketmanagers wurde am 29. Mai 2012 bereitgestellt [23].

Da die CI nicht Schwerpunkt dieser Arbeit ist, wurde die Fehlerbehebung nach mehreren erfolglosen Versuchen abgebrochen. Aufgrund der mangelnden Dokumentation dieser OpenWrap-Bestandteile in PULEIV ist das Nachvollziehen des Fehlerbildes nur schwer möglich. In einer nachgelagerten Prozessanalyse, die den Entwicklungsprozess um PULEIV betrachtet, soll der Ansatz einer CI-Pipeline aber nochmals aufgegriffen werden.

Zusammenfassung

Die Defizite und mögliche Lösungsansätze, die in diesem Kapitel beschrieben wurden, geben einen Einblick in den Zustand des Programmcodes von PULEIV. Die diesbezüglich vorgenommenen Anpassungsarbeiten wurden erläutert. Aufgrund des Umfangs der Anwendung ist im Rahmen dieser Arbeit eine allumfassende Betrachtung und Bearbeitung jedoch nicht möglich.

7. Umsetzung des Vier-Phasen-Ansatzes in PULEIV

Aufbauend auf die in Kapitel 2.3 beschriebenen Grundlagen soll der Vier-Phasen-Ansatz in PULEIV integriert werden. Da er ein Bestandteil der mikroskopischen Engpassanalyse ist, wird er neben dem bereits implementierten Drei-Kriterien-Ansatz in der gleichnamigen Komponente „mikroskopische Engpassanalyse “eingesetzt werden.

7.1. Ausgangslage

Die mikroskopische Engpassanalyse (vormals Knotenkapazität) wurde ursprünglich als eigene Anwendung namens **RePlan** entwickelt und später in PULEIV integriert. Deshalb sind ihre Bestandteile nicht im Haupt-Paket `PULEIV`, sondern in separaten Paketen `RePlan` bzw. `Iev.RePlan.Bewertung` enthalten.

7.1.1. Benutzeroberfläche

Die Benutzeroberfläche der mikroskopischen Engpassanalyse gliedert sich in drei Bereiche. Im Kopfbereich können die Simulationsergebnisse eingelesen und verschiedene Layer (de-)aktiviert werden. Darunter befindet sich die eigentliche Darstellung der Infrastruktur mit der Visualisierung der Ergebnisse. Im dritten Abschnitt können die Ergebnisse tabellarisch betrachtet werden. Weiterhin ist es dort möglich, verschiedene Ansichten für den Visualisierungsbereich zu wählen.

Die Layer der Anzeige und ihre Bedeutung sind in Tabelle B.1 im Anhang erläutert.

7.1.2. Engpassanalyse nach dem Drei-Kriterien-Ansatz

Die Layerklassen aus dem `RePLan`-Paket greifen zur Darstellung auf Daten zu, die in einer Objektinstanz der Klasse `Bewerter` gehalten werden. Diese fungiert als Ablage für die mikroskopische Engpassanalyse. Sie wird bei der Verknüpfung mit einem Teilprojekt Leistungsverhalten erzeugt.

Durch das Betätigen des *„Protokolle neu laden “*-Buttons im Kopfbereich wird die Ermittlung der Engpassrelevanzen angestoßen, die sich auf das gesamte Betriebsprogramm beziehen. Die Analyse der Daten im bisherigen Drei-Kriterien-Ansatz erfolgt in mehreren Schritten, die in Abbildung 7.1 dargestellt sind.

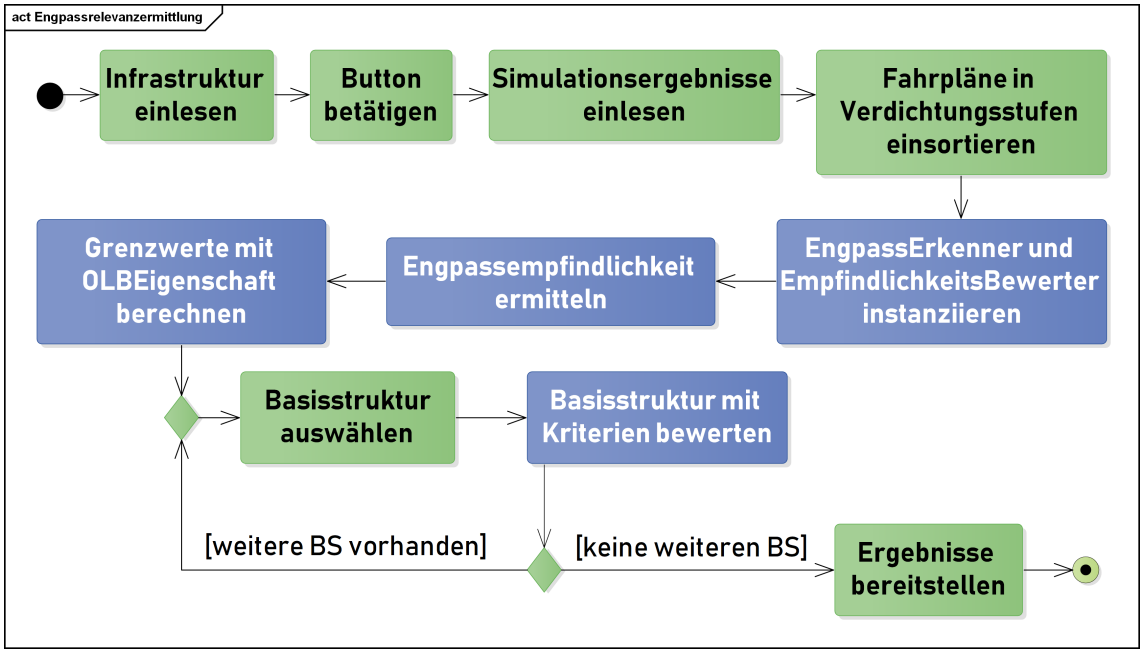

Abbildung 7.1.: Aktivitätsdiagramm zur Ermittlung der Engpassrelevanz mit dem Drei-Kriterien-Ansatz

1. Bei der Initialisierung der Bewerter-Instanz wird zunächst die Infrastruktur ausgewertet.
2. Anschließend werden die Simulationsergebnisse aus den Dateien gelesen und die Fahrpläne in die Verdichtungsstufen einsortiert.
3. Die Fahrwege der Züge aus den Fahrplänen werden in Ketten aus einzelnen Belegungselementen gespeichert.
4. Anhand einer Liste der Basisstrukturen und der dort zugeordneten Fahrwegkomponenten werden die Nicht erfüllbaren Belegungswünsche ermittelt. Dazu wird eine Instanz der Klasse `NEBBewerter` erstellt. Diese speichert auch die Ergebnisse der NEB-Ermittlung.
5. Zur späteren Verwendung werden für den Drei-Kriterien-Ansatz gültige Grenzwerte mit einer Instanz der Klasse `OLBEigenschaft` berechnet.
6. Danach wird eine Instanz der `EngpassErkenner`-Klasse angelegt. Diese enthält wiederum eine Instanz des `EmpfindlichkeitsBewerter`.
7. Letztere Instanz wird benutzt, um auf Basis der Fahrwegkomponenten die Engpassempfindlichkeiten zu ermitteln.
8. Im Anschluss werden diese Empfindlichkeiten, die NEBs und die Belegungsgrade als Kriterien genutzt, um die Basisstrukturen zu klassifizieren. Diese Einstufung wird in der `EngpassErkenner`-Instanz hinterlegt.

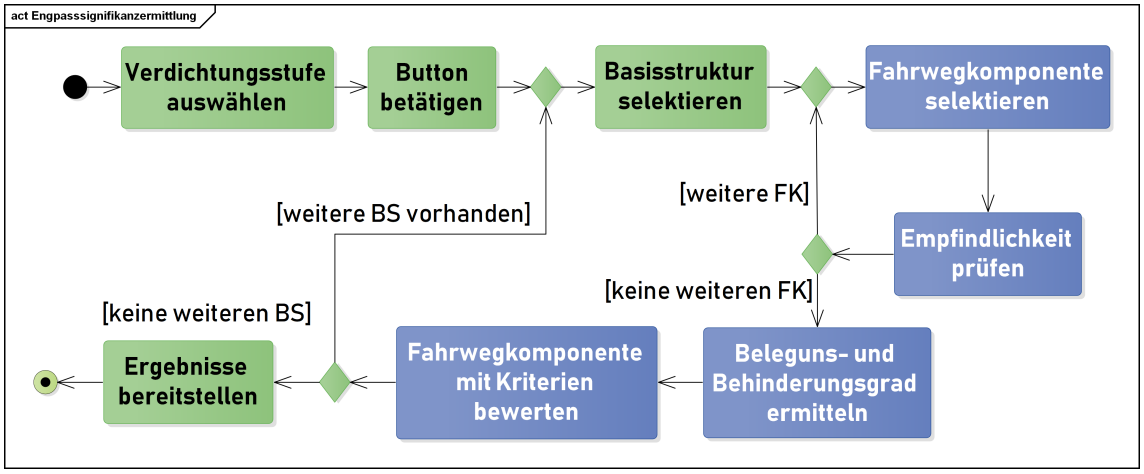

Abbildung 7.2.: Aktivitätsdiagramm zur Ermittlung der Engpasssignifikanz mit dem Drei-Kriterien-Ansatz

Neben dem *„Protokolle neu laden "*-Button gibt es im Kopfbereich auch eine ComboBox zur Auswahl einer Verdichtungsstufe. Nach der Wahl einer Stufe kann mit dem Betätigen des *„Auswerten "*-Buttons die Ermittlung der Engpasssignifikanzen gestartet werden. Die in Abbildung 7.2 dargestellten Schritte sind im Detail:

1. Die Betriebsqualität der einzelnen Basisstrukturen wird anhand der Effizienz, also dem Verhältnis von Belegungs- und Behinderungsgrad, ermittelt. Dazu wird eine Instanz des `EffizienzBewerter` erstellt.
2. Nach der Effizienzbewertung wird jede Bewertungsstufe in der `EngpassErkenner`-Instanz betrachtet und anhand der vorher ermittelten Grenzwerte des Drei-Kriterien-Ansatzes bewertet.
3. Die Ergebnisse der Verdichtungsstufe werden in eine Gesamtergebnisliste eingetragen.

7.2. Anpassungen für den Vier-Phasen-Ansatz

Um eine Auswahl zwischen dem bestehendem Drei-Kriterien-Ansatz und dem neu zu implementierenden Vier-Phasen-Ansatz zu ermöglichen, stehen zwei Radiobuttons zur Verfügung. Abbildung 7.3 zeigt die im Kopfbereich der Benutzeroberfläche eingefügten Bedienelemente.

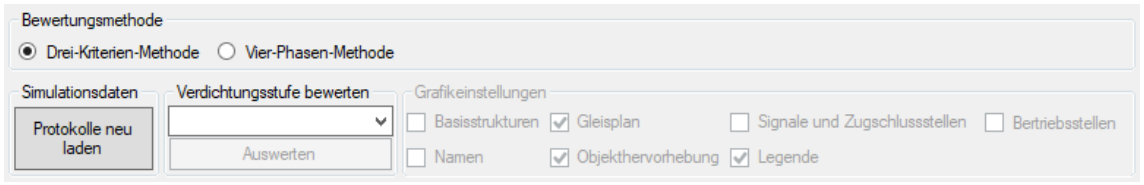

Abbildung 7.3.: Angepasste Benutzeroberfläche mit Auswahl des Ansatzes

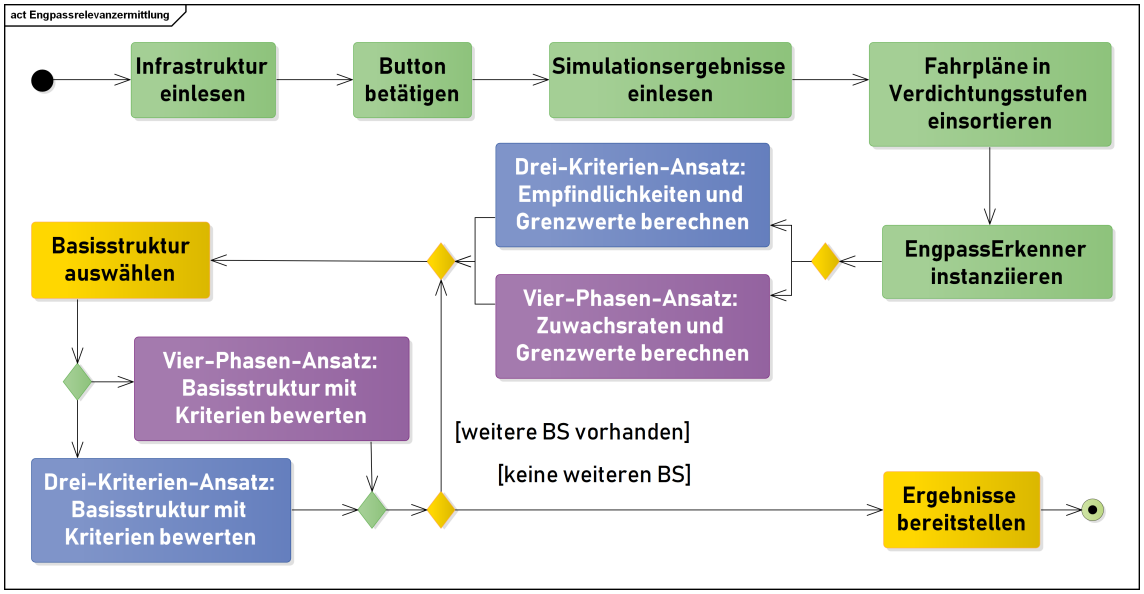

Abbildung 7.4.: Aktivitätsdiagramm zur Ermittlung der Engpassrelevanz mit beiden Ansätzen

Da die Ermittlungen der Engpassrelevanzen und -signifikanzen durch die Buttons angestoßen werden, muss zum Zeitpunkt der Ausführung bekannt sein, ob der Drei-Kriterien-Ansatz oder der Vier-Phasen-Ansatz ausgewählt ist. Deshalb wird ein neuer EventListener eingeführt, um dem `Bewerter` die geänderte Auswahl zu übermitteln. Dann sollen an geeigneten Stellen in die bestehenden Methoden des `EngpassErkenners` Verzweigungen eingebaut werden, um die Methoden für den Vier-Phasen-Ansatz auszuführen. In den Abbildungen 7.1 und 7.2 sind die Bestandteile, die bei der Einführung des Vier-Phasen-Ansatz angepasst werden, blau hervorgehoben. Exemplarisch ist in Abbildung 7.4 der resultierende Ablauf für die Ermittlung der Engpassrelevanzen mit den Verzweigungen dargestellt. Die gelb hervorgehobenen Elemente sind im Code aus Listing 7.1 umgesetzt.

7.3. Die neue Klasse `NEBZuwachsrateBewertung`

Zur Umsetzung der Berechnungen für den Vier-Phasen-Ansatz wird die neue Klasse `NEBZuwachsrateBewertung` eingeführt. Diese wird vom `EngpassErkenner` instanziiert. Die zur Berechnung der Kenngrößen benötigten Daten werden entweder über Parameter bei Methodenaufrufen oder über eine Referenz auf die `Bewerter`-Instanz bezogen.

Einige Berechnungen benötigen komplexere Datenstrukturen. Diese sind deshalb im Kopfbereich der Klasse `NEBZuwachsrateBewertung` definiert:

- **Zuwachsrate** bestehend aus der eigentlichen Rate und y-Achsenabschnitten als double-Werte
- **ZuwachsratenTupel** bestehend aus je einer Zuwachsrate für die vier Phasen
- **ProzentualeGrenzwerte** bestehend aus double-Werten zur Abgrenzung der Phasen

Listing 7.1 Die Methode BewertenRelevanz() der Klasse EngpassErkenner

```
1   public void BewertenRelevanz()
2   {
3     this.EngpassRelDict = new Dictionary<Basisstruktur, EngpassStufe>();
4
5     if (this.bewerter.VierPhasenAnsatz)
6     {
7       //Vier-Phasen-Ansatz (verwendet NEBZuwachsrateBewertung)
8       this.NebZuwachsrateBewerter.Bewerten((IEnumerable < Basisstruktur
            >)this.behälter.Basisstrukturen); ;
9     }
10    else
11    {
12      //Drei-Kriterien-Ansatz
13      HashSet<FahrwegkomponentePair> pairSet = this.knoten.CreateFahrwegkomponentePairSet();
14      this.endeKompDict = this.knoten.CreateEndeKomponenteDict();
15      this.EmpfindlichkeitBewerter.Bewerten(this.knoten.Fahrwegkomponenten, pairSet);
16    }
17
18    //Zuordnung der Stufen
19    foreach (Basisstruktur bs in this.knoten.Basisstrukturen)
20    {
21      EngpassStufe stufe = this.BewertenEngpassRelevanz(bs);
22      if (stufe != EngpassStufe.KEIN)
23          this.EngpassRelDict.Add(bs, stufe);
24    }
25  }
```

Zur Speicherung der Berechnungen und Bewertungen besitzt die Klasse verschiedene Attribute. In ihnen sind die ermittelten Zuwachsraten für die verschiedenen Phasen sowie die berechneten Grenzwerte zur Bestimmung der Engpassrelevanz und Signifikanz abgelegt.

7.3.1. Ermittlung der Grenzwerte

Die Ermittlung der Grenzwerte wird durch einen Aufruf seitens der EngpassErkenner-Instanz gestartet. Sie umfasst die Erfassung aller Nicht erfüllbaren Belegungswünsche getrennt nach Phasen, die Berechnung der Zuwachsraten in einer Basisstruktur und der darauf basierenden Grenzwerte für die Engpassrelevanz und -signifikanz. Das Aktivitätsdiagramm 7.5 stellt diesen Ablauf dar.

7.3.2. Engpassrelevanz

Die Engpassrelevanz wird direkt im Anschluss an die Bestimmung der Grenzwerte bewertet. Der Aufruf erfolgt wiederum durch die Instanz der Klasse EngpassErkenner. Zur Einordnung der Basisstrukturen werden deren NEB-Zuwachsraten mit den Grenzwerten verglichen. Diese Kriterien sind in Tabelle 7.1 aufgelistet.

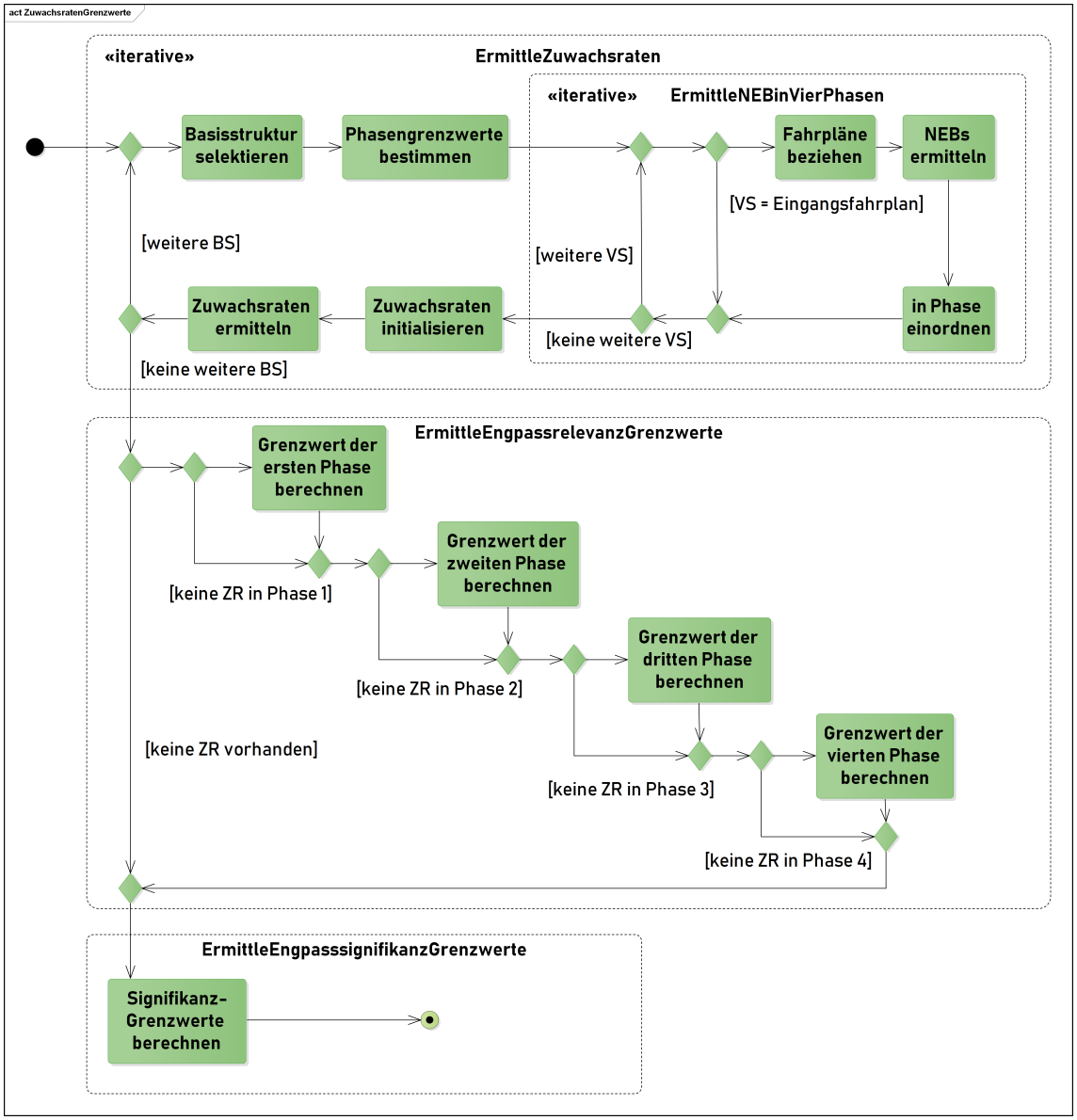

Abbildung 7.5.: Ermittlung der Grenzwerte gemäß dem Vier-Phasen-Ansatz

Die Kriterien lassen sich durch eine Folge von if-Bedingungen prüfen. In Abhängigkeit vom zutreffenden Kriterium wird dann die jeweilige Engpassrelevanzstufe zurückgegeben. Das Ergebnis wird in der `EngpassErkenner`-Instanz abgelegt.

7.3.3. Engpasssignifikanz

In Abhängigkeit von der gewählten Verdichtungsstufe wird analog zur Bewertung der Engpassrelevanz durch den `EngpassErkenner` die entsprechende Methode aufgerufen. Die zuvor ermittelten Grenzwerte werden verwendet, um die Basisstrukturen einer Signifikanzzstufe zuzuweisen.

Kriterium	Relevanz
NEB-Zuwachsrate in Phase 4 ist kleiner als der Grenzwert in Phase 4	sehr hoch
Das vorige Kriterium trifft nicht zu und die NEB-Zuwachsrate in Phase 3 ist größer als der Grenzwert in Phase 3	hoch
Das vorige Kriterium trifft nicht zu und die NEB-Zuwachsrate in Phase 2 ist größer als der Grenzwert in Phase 2	mittel
Das vorige Kriterium trifft nicht zu und die NEB-Zuwachsrate in Phase 1 ist größer als der Grenzwert in Phase 1	niedrig
Das vorige Kriterium trifft nicht zu	keine

Tabelle 7.1.: Kriterien der Engpassrelevanz

Listing 7.2 Die Methode `ErmittleBSSignifikanz()` der Klasse `NEBZuwachsrateBewertung`

```
1  public EngpassStufe ErmittleBSSignifikanz(Basisstruktur bs)
2  {
3      List<BSNEB> listNEBBS = new List<BSNEB>();
4      this.bSNEBdict.TryGetValue(bs, out listNEBBS);
5      foreach(BSNEB neb in listNEBBS)
6      {
7          if (neb.Verdichtungsstufe != double.Parse(this.bewerter.BewertungsstufeId))
8              continue;
9
10         if (neb.NEB > this.GrenzNEBOben)
11             return EngpassStufe.HOCH;
12         if (neb.NEB > this.GrenzNEBUnten)
13             return EngpassStufe.MITTEL;
14         return EngpassStufe.NIEDRIG;
15     }
16     return EngpassStufe.KEIN;
17 }
```

7.3.4. Maßgebender Engpass und mögliche Reserven

Basierend auf den Ergebnissen der Engpassrelevanz und Engpasssignifikanz kann bei Engpässen im Anschluss überprüft werden, ob sie einer der Kandidaten für den maßgebenden Engpass sind. In der Theorie würde ein maßgebender Engpass in der vierten Phase keine Zuwachsrate mehr aufweisen, da er komplett gesättigt ist. Durch die Annäherung aus Simulationsergebnissen ist dies in der Praxis kaum zu erkennen. Deshalb werden in der softwaregestützten Umsetzung mögliche Kandidaten ermittelt, deren Ergebnisse eine vollständigen Sättigung nahe kommen. Dazu wird eine weitere Methode angeboten, die zwei Bedingungen überprüft:

1. NEB-Zuwachsrate der Basisstruktur in Phase 4 ≤ Mittelwert der NEB-Zuwachsraten aller Basisstrukturen in Phase 4

2. Nicht erfüllbare Belegungswünsche der Basisstruktur bei der Verdichtungsstufe der Durchsatzbezogenen Leistungsfähigkeit ≤ Mittelwert der Nicht erfüllbaren Belegungswünsche aller Basisstrukturen bei der Verdichtungsstufe der Durchsatzbezogenen Leistungsfähigkeit

Wenn beide Bedingungen erfüllt sind, ist der Engpass ein Kandidat für den maßgebenden Engpass mit großen Auswirkungen auf die gesamte Infrastruktur.

Zur Planung möglicher Gegenmaßnahmen ist es wichtig, potentielle Reserven zu ermitteln. Um eine mögliche Reserve für eine Verdichtungsstufe zu sein, darf eine Basisstruktur dabei weder eine Engpassrelevanz noch eine Engpasssignifikanz aufweisen. Zur Ermittlung werden die bestehenden Methoden verwendet und dabei deren Rückgabewerte verknüpft.

Überblick

Ein Klassendiagramm zum Überblick über das Zusammenspiel der realisierten Klassen und Methoden befindet sich in Anhang B. Außerdem enthält der Anhang weitere Codeausschnitte zur Erläuterung der hier beschriebenen Teilfunktionen.

8. Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit sollte ursprünglich (nur) der Vier-Phasen-Ansatz als neue Methodik in der softwaregestützten Engpassanalyse spezifiziert und implementiert werden. Während der Arbeit mit dem Code der Bestandssoftware sind Schwierigkeiten in Bezug auf das Verständnis des Codes aufgetreten. Deshalb wurde die Analyse der Defizite der Softwarequalität, besonders hinsichtlich der Wartbarkeit, und deren teilweise Beseitigung ein weiterer Schwerpunkt der Arbeit.

Bevor der neue methodische Ansatz spezifiziert und implementiert wurde, wurden die fachlichen Grundlagen der Leistungsuntersuchung erarbeitet. Diese Grundlagen sind in Kapitel 2 festgehalten. Die praktische Umsetzung des Vier-Phasen-Ansatzes inklusive der Spezifikation wurde in Kapitel 7 erläutert.

Neben der Recherche und dem Verständnis der fachlichen Grundlagen wurden auch die formalen Hintergründe der Softwarequalität betrachtet und der vorhandene Programmcode und die begleitenden Dokumente auf Defizite untersucht. In Kapitel 6 wurden die Befunde festgehalten und mögliche Maßnahmen zur Behebung der Mängel beschrieben. Teilweise wurden diese auch bereits umgesetzt, wie z.B. die Vereinheitlichung der Bezeichner von Klassen und die Nachtragung von fehlenden Kommentaren.

In Vorbereitung auf die Erweiterung mit dem Vier-Phasen-Ansatz wurde die Struktur der Anwendung mit ihren Teilprojekten und Komponenten angepasst. Aus dem Teilprojekt Leistungsverhalten wurde ein neues Teilprojekt Engpassanalyse mit der makroskopischen und mikroskopischen Engpassanalyse ausgegliedert. Im Zusammenhang damit wurde die in der Spezifikation nur unzureichend beschriebene Softwarearchitektur betrachtet und ein Überblick dokumentiert. Diese strukturellen Aspekte und Arbeiten sind in Kapitel 5 erläutert.

Auch zur XML-Schnittstelle der Anwendung RailSys gab es bislang nur unzureichende Informationsdokumente. Die neu erstellte Dokumentation der zentralen XML-Strukturen dieser Schnittstelle befindet sich im Anhang A.

Ausblick

Zur weiteren Verbesserung der Softwarequalität und der Wartbarkeit im Speziellen sollten die in Kapitel 6 beschriebenen Maßnahmen weitergeführt werden. Insbesondere die Vervollständigung der Dokumentation sollte mit hoher Priorität fortgeführt werden.

Außerdem bietet der Entwicklungsprozess rund um PULEIV Potential für weitere Verbesserungen. Die in Kapitel 3.4 betrachtete kontinuierliche Integration mit den automatisierten Tests und die Einführung eines Ticketingsystems sollen in einer Prozessanalyse angegangen werden.

Die unvollständige Dokumentation der Softwarearchitektur hat zu erheblichen Schwierigkeiten beim Verständnis des Zusammenspiels zwischen Oberflächen und Datenmodell geführt. Deshalb mussten mehrere Ansätze zur Anpassung der Struktur wieder verworfen werden. Dieser Mehraufwand und das realisierte Refactoring haben dazu geführt, dass zwar der Vier-Phasen-Ansatz zur Ermittlung der Engpässe und Reserven umgesetzt werden konnte, die Ermittlung und Visualisierung der Engpassursachen aber noch nicht implementiert wurden. Dies kann zu einem späteren Zeitpunkt ergänzt werden.

A. Details zur RailSys-Schnittstelle

Die Infrastruktur-, Fahrplan-, Fahrzeug- und Simulationsdaten werden innerhalb einer Projektstruktur von RailSys in verschiedene Dateien gespeichert. Diese Dateien haben unterschiedliche Dateiendungen, um sie voneinander zu unterscheiden. Inhaltlich basieren sie zumeist auf einer XML-Struktur.

Tabelle A.1 bietet einen Überblick darüber, welche Informationen in welchen Dateien enthalten sind:

Aufbau der Fahrplandateien

.rst-Datei

Die Daten eines Fahrplans werden in XML-Dateien mit der Dateiendung rst gespeichert. Für jede Zugfahrt ist ein eigener **train**-Block angelegt. Ein exemplarischer Aufbau eines solchen Blocks ist in Listing A.1 vereinfacht dargestellt. Jeder der train-Blöcke enthält einen

Datenart	**RailSys-Element**
Fahrplan	
Zuginformationen	`.rst`-Dateien
Laufweg eines Zuges	`.rst`-Dateien
Abfahrts- und Haltezeiten	`.rst`-Dateien
Gruppierung der Zugläufe	`takt.xml`
Informationen zur Regelfahrzeit	Fahrplanexport
Infrastruktur	
Betriebsstellen	`.rsl`-Dateien
Strecken	`.rsl`-Dateien
Vollständiger Spurplan	`.rsl`-Dateien
Fahrzeugdaten	
Masse des Zuges	`.lok`- und `.zug`-Dateien
Simulationsergebnisse	
Störungsinformationen	`.rsd`-Dateien
Abfahrtszeiten - Betriebsstellen	`fahre++.pro`
Abfahrtszeiten - Signale	`fahre++.pro`
Einbruchsverspätung	`fahre++.pro`

Tabelle A.1.: Zuordnung der Daten zu Railsys-Projektelementen (Quelle: [24])

header-Block, der wiederum in einem **service**-Block die Verkehrstage enthält. Außerdem sind die eigentlichen Fahrplandaten als **entry** in einem **timetableentries**-Block angelegt. Die Bedeutung der jeweiligen Attribute wird in Tabelle A.2 erläutert.

Listing A.1 Vereinfachter Aufbau der XML-Datei (.rst) mit den Fahrplandaten

```
1 <train number="1" numbervar="0" name="1" pattern="/FRZ/AHX-LBC" class="FRz"
       constructionState="verkehrt" typeNumber="42">
2     <header><service opday="127" extOpday="0" /></header>
3     <timetableentries>
4         <entry stationID="AHX" trackID="H-1" bstfahrweg="Einbruch-AHX" lineID="a-l" type="pass"
              departure="00:10:00" departureFix="true" timingPoint="true" regularRunningTime="167"
              trainTypeNumber="42" />
5         <entry stationID="BS" trackID="N-3" bstfahrweg="AHX-3-LBC" type="pass" departure="00:12:47"
              departureFix="false" timingPoint="true" regularRunningTime="224" trainTypeNumber="42" />
6         <entry stationID="LBC" trackID="L-1" bstfahrweg="LBC-Ausbruch" type="pass"
              departure="00:16:31" departureFix="false" timingPoint="true" />
7     </timetableentries>
8 </train>
```

takt.xml

In der **takt.xml** sind die verschiedenen Zugläufe in sogenannte `trainpattern` gruppiert. Dabei können auch Eltern-Pattern angegeben werden, um die Pattern hierarchisch zu gliedern. Sie werden z.B. im Basisfahrplanfenster von PULEIV als `Obergruppe/Untergruppe` angezeigt. Ein beispielhafter Aufbau einer takt.xml-Datei ist in Listing A.2 dargestellt:

Listing A.2 Vereinfachter Aufbau der `takt.xml`-Datei

```
1 <trainpatterns>
2     <trainpattern name="GV" parenttrainpattern=""/>
3     <trainpattern name="LBC-EN" parenttrainpattern="/GV"/>
4     <trainpattern name="LBC-EN mit Halt" parenttrainpattern="/GV"/>
5         <trainpattern name="LBC-AHC" parenttrainpattern="/GV"/>
6 </trainpatterns>
```

Block	Attribut	Werte	Bedeutung
train	number	Integer	laufende Nummer für jeden train-Block beginnend mit 1
	numbervar	Integer	Varianten des Zuges
	name	Integer	Verkettung von number und numbervar
	pattern	String	Zuglaufgruppe
	class	String	Zuggattung
	constructionState	String	Status (verkehrt, verkehrt nicht, ...)
	typeNumber	Integer	Zugkonfiguration für die Fahrdynamikberechnung
service	opday	Integer	Verkehrstage
	extOpday	Integer	Ausschlusstage
entry	stationID	String	Kürzel der Betriebsstelle
	trackID	String	Gleis innerhalb der Betriebsstelle
	bstfahrweg	String	Route zwischen den Betriebsstellen
	lineID	String	Streckenbezeichnung
	type	String	Verhalten des Zuges (Passieren, Halten, ...)
	departure	time	Abfahrtszeit
	departureFix	boolean	fixierte Abfahrtszeit (bei einbrechendem Verkehr)
	timingPoint	boolean	Knoten mit Fahrplanzeitangaben
	regularRunningTime	Integer	planmäßige Verweildauer innerhalb des Streckenabschnitts in Sekunden
	trainTypeNumber	Integer	Verwendete Fahrzeugbaureihe
	minStopTime	String	minimale Haltezeit (nicht bei type = 'pass')
	stopTime	String	planmäßige Haltezeit (nicht bei type = 'pass')

Tabelle A.2.: Die Attribute der Datenblöcke in einer `.rst`-Fahrplandatei

Listing A.3 Vereinfachter Aufbau der XML-Datei (.rsl) mit den Infrastrukturdaten

```
1 <line lineID="a-l" name="Ahorn-Lindburg" trafficRouting="on_the_right" type="local"
       dependentLines="">
2     <version major="1" minor="2" patchlevel="0" comment=""/>
3     <stations> ... </stations>
4     <nodes> ... </nodes>
5     <links> ... </links>
6     <blockSections> ... </blockSections>
7     <slipDistances> ... <slipDistances>
8     <stationRoutes> ... </stationRoutes>
9 </line>
```

Aufbau der .rsl-Infrastrukturdateien

In den `.rsl`-Dateien werden alle infrastrukturbezogenen Daten gespeichert. Dazu gehören unter anderem die Streckeninformationen, Signalblöcke, Stationen und Durchrutschwege. Listing A.3 zeigt den grundlegenden Aufbau einer `.rsl`-Datei.

<stations>

Der XML-Knoten `<stations>` enthält Definitionen für die Stationen der betrachteten Infrastruktur. Diese bestehen jeweils aus Einträgen der Form:

<station stationID="`String` "name="`String` "xNet="`Number` "yNet="`Number`"/>

Der Knoten enthält also 4 Attribute:

- *stationID* enthält ein eindeutiges Kürzel zur Identifikation einer Station. Dieses Kürzel wird in anderen Knoten referenziert.
- *name* enthält den ausgeschriebenen Namen der Station.
- *xNet* enthält die X-Koordinate der Station bezogen auf die grafische Darstellung der Infrastruktur.
- *yNet* enthält die Y-Koordinate der Station bezogen auf die grafische Darstellung der Infrastruktur.

<nodes>

Im XML-Knoten `<nodes>` sind verschiedene Elemente der Infrastruktur aufgelistet. Folgende Typen können mehrfach vorkommen:

- *node* enthält einen einfachen Knoten des Gleisnetzes, der z.B. die Informationen eines Vorsignals trägt.
- *switch* enthält die Definition einer Weiche.

Typ	Attribut	Werte	Bedeutung
alle	nodeID	Integer	eindeutige Nummer, die von anderen Elementen referenziert wird
	name	String	Bezeichner
	description	String	Beschreibung des Elements
	kilometre	Number	Streckenkilometer
	x	Number	reale X-Koordinate für die Berechnungen
	y	Number	reale Y-Koordinate für die Berechnungen
	xNet	Number	X-Koordinate in der modellierten Infrastruktur zur grafischen Darstellung
	yNet	Number	Y-Koordinate in der modellierten Infrastruktur zur grafischen Darstellung
node	referenceNode	Integer	Referenz auf eine Gleisdefinition
	presignal	Boolean	Wirkung als Vorsignal
	with1000HzMagnet	Boolean	Wirkung mit Zugbeeinflussung
switch	endOfMainTrack	Integer	Referenz auf einen Knoten für das Weichenende des Hauptgleises
	endOfBranchTrack	Integer	Referenz auf einen Knoten für das Weichenende des Nebengleises
	stationRoutes	Integer	Referenz auf einen Knoten für den Weichenbeginn
signal	referenceNode	Integer	Referenz auf eine Gleisdefinition
	interlockingMachine	String	Referenz auf ein Stellwerk
	type	String	Signaltyp, u.a. „exit“, „entry“
track	trackID	String	eindeutiger Bezeichner
	stationID	String	Referenz auf eine Station aus dem <stations>-Knoten
	interlockingMachine	String	Referenz auf ein Stellwerk
	length	Integer	Länge des definierten Gleises in Metern
	referenceNode	Integer	Referenz auf einen Knoten
stationBorder	lineID	String	Referenz auf eine Infrastruktur aus dem <line>-Knoten
	stationID	String	Referenz auf eine Station aus dem <stations>-Knoten
	exitType	String	Art der Begrenzung
	trackNumber	unbekannt	Referenz auf einen <track>-Knoten
	netBorder	Boolean	Infrastrukturgrenze

Tabelle A.3.: Die Attribute der Datenblöcke im `<nodes>`-Knoten

- *signal* enthält eine Signaldefinitionen.
- *track* enthält eine Gleisdefinition.
- *stationBorder* enthält die Definition für die Begrenzung einer Betriebsstelle.

Die Attribute der verschiedenen Typen sind in Tabelle A.3 erläutert.

Attribut	Werte	Bedeutung
FahrplanId	String	Der Bezeichner des simulierten Fahrplans/Betriebskonzepts
StörungId	Integer	'-1', sofern keine Störung auftrat
ZugId	Integer	Die Nummer des Zuges aus dem Fahrplan
KomponenteId	Integer	Die durch den Zug belegte Fahrwegkomponente
BelegungSekunden	Integer	Die tatsächliche Belegungszeit der Fahrwegkomponente durch den Zug
BehinderungSekunden	Integer	Die Zeit, in der der Zug während der Belegung der Fahrwegkomponente durch etwas behindert war

Tabelle A.4.: Die Attribute der Record-Tags in der Protokoll-XML

Listing A.4 Vereinfachter Aufbau der XML-Datei mit den belegungselementbezogenen Behinderungszeiten

```
<BelegungselementbezogeneBehinderungszeit>
    <Basisstruktur ID="1">
        <BehinderndeFahrwegkomponente>
            <ID>22</ID>
            <SummeBB>46</SummeBB>
            <AnzahlBehinderndeZüge>1</AnzahlBehinderndeZüge>
            <BehinderndeZugnummer>Z0077_225_59</BehinderndeZugnummer>
        </BehinderndeFahrwegkomponente>
        ... <!-- weitere Behindernde Fahrwegkomponenten -->
    </Basisstruktur>
    ... <!-- weitere Basisstrukturen -->
</BelegungselementbezogeneBehinderungszeit>
```

Protokoll

Nach der Simulation in RailSys liegt eine Protokolldatei vor, in der die Ergebnisse der einzelnen Zugfahrten aufgelistet sind. Für jede Belegung einer Komponente liegt ein **Record**-Eintrag vor, dessen Attribute in Tabelle A.4 erläutert sind.

Belegungselementbezogene Behinderungszeit

Die belegungselementbezogene Behinderungszeit wird in einer separaten XML-Datei erfasst. Der Aufbau dieser Datei ist in Listing A.4 vereinfacht dargestellt. Für jede Basisstruktur werden die Fahrwegkomponenten aufgelistet, bei denen es zu einer Behinderung kam. Hierbei bezeichnet der Zahlenwert bei **ID** die ID der Fahrwegkomponente, bei der die Behinderung auftrat. In **AnzahlBehinderndeZüge** und **BehinderndeZugnummer** sind die Anzahl sowie die Bezeichner der behindernden Züge hinterlegt. Die aufsummierte belegungselementbezogene Behinderungszeit ist in **SummeBB** erfasst.

B. Erläuterungen zur Klasse NEBZuwachsrateBewertung

Im Folgenden werden einzelne Codeausschnitte der Klasse `NEBZuwachsrateBewertung` erläutert.

Eingebettete Klassen

Im Kopfbereich der Klasse `NEBZuwachsrateBewertung` sind Datenstrukturen für die verschiedenen Berechnungen definiert:

Listing B.1 Definition der eingebetteten Klassen

```
1         protected class Zuwachsrate
2         {
3             public Zuwachsrate(double rate, double yAchsenabschnittInFunktion, double
                  yAchsenabschnitt)
4             {
5                 this.Rate = rate;
6                 this.YAchsenabschnittInFunktion = yAchsenabschnittInFunktion;
7                 this.YAchsenabschnitt = yAchsenabschnitt;
8             }
9
10            public double Rate                          { get; set; }
11            public double YAchsenabschnittInFunktion    { get; set; }
12            public double YAchsenabschnitt              { get; set; }
13        }
14
15        private class ZuwachsratenTupel
16        {
17            public Zuwachsrate ZuwachsrateUnterOLB      { get; set; }
18            public Zuwachsrate ZuwachsrateInOLB         { get; set; }
19            public Zuwachsrate ZuwachsrateObergrenzeMax { get; set; }
20            public Zuwachsrate ZuwachsrateÜberMax       { get; set; }
21        }
22
23        private class ProzentualeGrenzwerte
24        {
25            public double oLBUntergenze                         { get; set; }
26            public double oLBObergenze                          { get; set; }
27            public double maximaleLeistungsfähigkeit            { get; set; }
28            public double erhoehteMaximaleLeistungsfähigkeit    { get; set; }
29        }
```

Attribute

Die Attribute der Klasse `NEBZuwachsrateBewertung` zur Haltung der Grenzwerte und berechneten Größen:

Listing B.2 Attribute der Klasse `NEBZuwachsrateBewertung`

```
1         //enthält die Zuwachsraten der Basisstrukturen in allen Phasen
2         private Dictionary<Basisstruktur, ZuwachsratenTupel> zuwachsratenBasisstruktur
3         //enthalten die Zuwachsraten der Basisstrukturen in den jeweiligen Phasen
4         private Dictionary<Basisstruktur, Zuwachsrate> zuwachsratenBasisstrukturUnterOLB
5         private Dictionary<Basisstruktur, Zuwachsrate> zuwachsratenBasisstrukturOLB
6         private Dictionary<Basisstruktur, Zuwachsrate> zuwachsratenBasisstrukturObergrenzeMax
7         private Dictionary<Basisstruktur, Zuwachsrate> zuwachsratenBasisstrukturUeberMax
8
9         //enthält die NEBs der Basisstrukturen pro Phase
10        private Dictionary<Basisstruktur, List<BSNEB>> bSNEBdict
11
12        //Zuwachsratengrenzwerte
13        private double zuwachsratenGrenzUnterOLB;
14        private double zuwachsratenGrenzOLB;
15        private double zuwachsratenGrenzObergrenzeMax;
16        private double zuwachsratenGrenzÜberMax;
17        private double zuwachsratenGrenzUnterOLBYAbstand;
18        private double zuwachsratenGrenzOLBYAbstand;
19        private double zuwachsratenGrenzOberMaxYAbtand;
20        private double zuwachsratenGrenzÜberMaxYAbtand;
21
22        //Grenzwerte für die Engpasssignifikanz
23        private double GrenzNEBOben
24        private double GrenzNEBUnten
```

Einordnung in die bestehenden Klassen

Abbildung B.1 bietet einen Überblick über die verschiedenen Klassen, die an der Bewertung beteiligt sind:

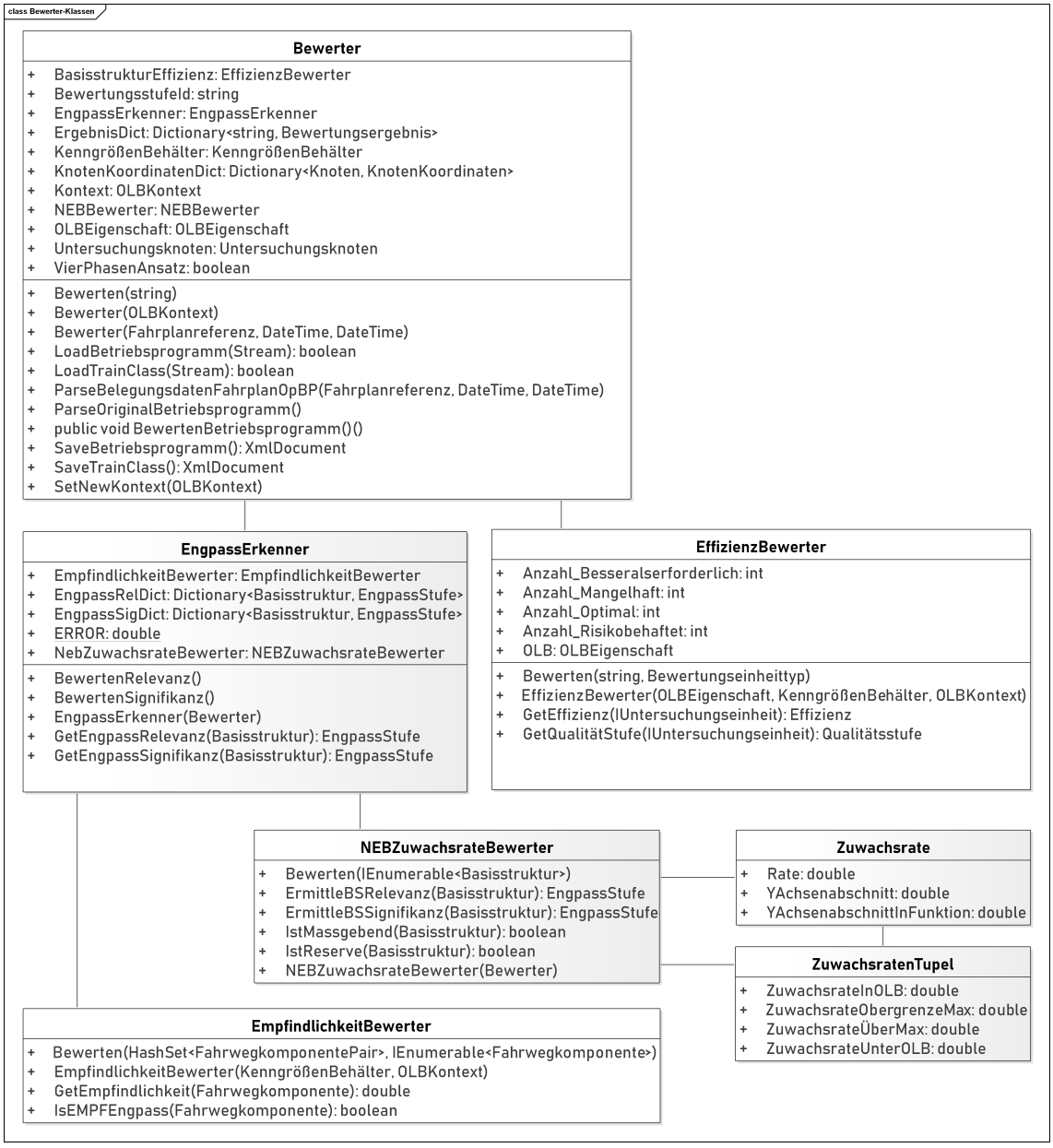

Abbildung B.1.: Klassendiagramm mit den an der Engpassanalyse beteiligten Klassen

Layer der mikroskopischen Engpassanalyse

Die mikroskopische Engpassanalyse bietet folgende Layer in der Darstellung der Infrastruktur an. Dabei sind die allgemeinen Layer beliebig zuschaltbar. Aus den Layern der Bereiche Betriebsqualität und Engpässe kann insgesamt maximal einer gleichzeitig aktiv sein.

Layer	Beschreibung
Allgemeine Layer	
Gleisplan	die Gleise der Infrastruktur
Basisstrukturen	farblich voneinander abgesetze Einteilung in Basisstrukturen
Namen	Namen der Betriebsstellen
Betriebsstellen	Rahmen um die Betriebsstellen
Objekthervorhebung	farbliche Hervorhebung des in der Tabelle selektierten Elements
Signale und Zugschlusstellen	Anzeige der definierten Signale und Zugschlussstellen
Legende	Optionale Legende zur Erklärung der anderen aktivierten Layer
Betriebsqualität	
Qualitätsstufe	Kategorisierung anhand des Belegungs- und Behinderungsgrads
Belegungsgrad	farblich voneinander abgesetze Abstufung des Belegungsgrad
Behinderungsgrad	farblich voneinander abgesetze Abstufung des Behinderungsgrad
Nicht erfüllbare Belegungswünsche	farblich voneinander abgesetze Abstufung der NEBs
Engpässe	
Engpassrelevanz	farblich voneinander abgesetze Abstufung der Engpassrelevanz
Engpasssignifikanz	farblich voneinander abgesetze Abstufung der Engpasssignifikanz

Tabelle B.1.: Layer der Darstellung in der mikroskopischen Engpassanalyse

Literaturverzeichnis

[1] J. Pachl, Hrsg. *Systemtechnik des Schienenverkehrs: Bahnbetrieb planen, steuern und sichern*. 9. Aufl. 2018. Wiesbaden: Springer Fachmedien Wiesbaden, 2018. ISBN: 978-3-658-21407-4. DOI: 10.1007/978-3-658-21408-1 (zitiert auf S. 15).

[2] X. Li. *Mikroskopische Engpassanalyse bei eisenbahnbetriebswissenschaftlichen Leistungsuntersuchungen: Zugl.: Stuttgart, Univ., Diss., 2015*. Bd. 14. Neues verkehrswissenschaftliches Journal. Norderstedt: BoD - Books on Demand, 2015. ISBN: 9783739200439 (zitiert auf S. 16, 17, 22, 23).

[3] DB Netz AG. *Richtlinie 405 - Fahrwegkapazität*. 2008 (zitiert auf S. 18, 20).

[4] C. Schmidt. *Beitrag zur experimentellen Bestimmung der Wartezeitfunktion bei Leistungsuntersuchungen im spurgeführten Verkehr*. 2009. DOI: 10.18419/OPUS-298 (zitiert auf S. 19).

[5] Z. Chu. *Modellierung der Wartezeitfunktion bei Leistungsuntersuchungen im Schienenverkehr unter Berücksichtigung der transienten Phase*. 2014. DOI: 10.18419/OPUS-514 (zitiert auf S. 19).

[6] F. Hantsch, X. Li. „Methoden zur Engpassanalyse bei der Infrastrukturbemessung im Schienenverkehr“. In: *ETR-Eisenbahntechnische Rundschau* 62.3 (2013), S. 30–33. URL: http://www.uni-stuttgart.de/iev/sites/default/files/Ver%C3%B6ffentlichungen/MartinLiHantschEngpassETR13.pdf (zitiert auf S. 20, 21).

[7] U. Martin, X. Li. *Entwicklung einer simulationsbasierten Methodik zur ursachenbezogenen Engpassbewertung komplexer Gleisstrukturen in spurgeführten Verkehrssystemen unter Berücksichtigung stochastischer Bedingungen: DFG-Forschungsprojekt (MA-2326/10-1)*. Bd. 11. Neues verkehrswissenschaftliches Journal. Norderstedt: Books on Demand, 2015. ISBN: 9783734795886 (zitiert auf S. 21).

[8] J. Ludewig, H. Lichter. *Software Engineering: Grundlagen, Menschen, Prozesse, Techniken*. 1. Aufl. Heidelberg: Dpunkt-Verl., 2007. ISBN: 9783898642682 (zitiert auf S. 25, 26).

[9] *ISO/IEC 25010:2011 - Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*. URL: https://www.iso.org/standard/35733.html (besucht am 23. 09. 2018) (zitiert auf S. 25).

[10] *Javadoc Tool Home Page*. URL: https://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html (besucht am 26. 09. 2018) (zitiert auf S. 26).

[11] *Doxygen*. URL: https://github.com/doxygen/doxygen (besucht am 26. 09. 2018) (zitiert auf S. 26, 41).

[12] R. E. Fairley. *Software engineering concepts*. McGraw-Hill series in software engineering and technology. New York: McGraw-Hill, 1985. ISBN: 9780070662728 (zitiert auf S. 26).

[13] M. Fowler, K. Beck. *Refactoring: Improving the design of existing code*. 28. printing. The Addison-Wesley object technology series. Boston: Addison-Wesley, 2013. ISBN: 0201485672 (zitiert auf S. 27).

[14] K. Beck. „Extreme programming: A humanistic discipline of software development“. In: *Fundamental Approaches to Software Engineering*. Hrsg. von E. Astesiano. Bd. 1382. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, S. 1–6. ISBN: 978-3-540-64303-6. DOI: 10.1007/BFb0053579 (zitiert auf S. 27).

[15] M. Fowler. *Continuous Integration*. 2006. URL: https://www.martinfowler.com/articles/continuousIntegration.html (besucht am 14. 09. 2018) (zitiert auf S. 28).

[16] A. Oetting. „Physikalische Maßstäbe zur Beurteilung des Leistungsverhaltens von Eisenbahnstrecken“. Prüfungsjahr: 2005. - Publikationsjahr: 2006; Aachen, Techn. Hochsch., Diss., 2005. Diss. Aachen, 2005, 290 S. : graph. Darst. URL: http://publications.rwth-aachen.de/record/61482 (zitiert auf S. 29, 30).

[17] *RailSys | Produkt der RMCon*. URL: https://www.rmcon.de/railsys/ (besucht am 18. 09. 2018) (zitiert auf S. 31).

[18] *LUKS | VIA Consulting & Development GmbH*. URL: https://www.via-con.de/development/luks/ (besucht am 18. 09. 2018) (zitiert auf S. 31).

[19] *PAULA-Z | ASCI Systemhaus Berlin*. URL: https://www.asci-systemhaus.de/de/softwareloesungen/paulaz/index.php (besucht am 18. 09. 2018) (zitiert auf S. 31).

[20] SystemTestPortal. *SystemTestPortal is a web application that allows to create, run and analyze manual system tests.* URL: https://gitlab.com/stp-team/systemtestportal-webapp (besucht am 10. 09. 2018) (zitiert auf S. 42).

[21] GitLab. *GitLab Continuous Integration & Deployment*. URL: https://about.gitlab.com/features/gitlab-ci-cd/ (besucht am 10. 09. 2018) (zitiert auf S. 42).

[22] M. Project. *Mono Documentation*. 2018. URL: https://www.mono-project.com/docs/gui/winforms/ (besucht am 10. 09. 2018) (zitiert auf S. 42).

[23] OpenWrap. *OpenWrap GitHub Repository*. 2018. URL: https://github.com/OpenWrap (besucht am 10. 09. 2018) (zitiert auf S. 42).

[24] U. Martin, C. Schmidt, P. Breier, Z. Chu, M. B. Lynch, I. Podolski. *PULEIV 2 Spezifikation Version 2.1*. 2011 (zitiert auf S. 53).

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift