Anhang zu:

Entwicklung eines aktuellen Verfahrens zur rechnerischen Dimensionierung gemäß den RDO Beton

von

Thorsten Kathmann
Thorsten Hermes
Uwe Kucera
Jörg Stöver

DTV-Verkehrsconsult GmbH
Aachen

Johannes Neumann
Jan Lehmkuhl
Jan Mirco Pfeifer

ISAC GmbH
Aachen

Berichte der Bundesanstalt für Straßenwesen

Straßenbau Heft S 180

Forschungsprojekt FE 04.0316/2018/ORB
der Bundesanstalt für Straßenwesen

Lastenheft

Konzeption und Erstellung eines Programms zur Rechnerischen Dimensionierung gemäß den RDO Beton

DTV-Verkehrsconsult GmbH

Dr.-Ing. Thorsten Kathmann

Thorsten Hermes

Uwe Kucera

Ingenieurgesellschaft für Straßenwesen Aachen mbH

Dr.-Ing. Johannes Neumann

Dr.-Ing. Jan Lehmkuhl

Inhaltsverzeichnis

Präambel

Dieses Lastenheft besteht aus einem allgemeinen, wiederverwendbaren, projektunabhängigen Teil (Kapitel 1, 2) und einem projektspezifischen Teil (Kapitel 3). Einzelne Aspekte können aufgrund dieses modularen Aufbaus mehrfach aufgegriffen und so in mehreren Schritten präzisiert werden. Sollte es dabei zu Widersprüchen kommen, gilt jeweils die Festlegung des späteren Textblocks, d. h. der projektspezifische Teil überschreibt ggf. den allgemeinen Teil und innerhalb des allgemeinen Teils haben Festlegungen in Kapitel 2 Vorrang vor Kapitel 1.

Innerhalb einiger Unterkapitel kann es mehrere optionale Auswahlmöglichkeiten geben. Es sind nur die angekreuzten Textbestandteile zu berücksichtigen.

1 Universelle Anforderungen

Dieses Kapitel beinhaltet Festlegungen, die nicht nur unabhängig vom konkreten Projekt, sondern darüber hinaus auch unabhängig von der Art der zu entwickelnden Software getroffen werden können.

1.1 Zielumgebung

In diesem Abschnitt werden die Zielumgebungen in allgemeiner, projektunabhängiger Form grundsätzlich spezifiziert. Im projektspezifischen Teil finden sich ggf. weitergehende Angaben, z. B. zur Anzahl der Installation und Details bezüglich zu unterstützender Versionsnummern der Laufzeitumgebungen etc.

1.1.1 Windows-Arbeitsplätze

Die zu entwickelnde Anwendung muss auf einem mit durchschnittlicher Hardware ausgestatteten Arbeitsplatzrechner unter dem Betriebssystem Microsoft Windows 10 voll funktionsfähig sein und nach der Installation einfach ausgeführt werden können.

Sofern Administrator-Rechte für die Installation erforderlich sind, ist dies im Pflichtenheft anzugeben. Der Betrieb muss unter einem Benutzer-Account ohne Administrator-Rechte möglich sein.

Sofern die spezifischen Aufgabenstellung keine explizite (lokale) Netzwerkfunktionalität oder Internet-Anbindung beinhaltet, muss ein Offline-Betrieb vollumfänglich möglich sein.

1.1.2 Internationalisierung / Lokalisierung

Das System ist zunächst bezüglich Bezeichnungen, Einheiten, Zeitzonen und Formatkonventionen auf Nutzer in Deutschland auszurichten. Weitere Varianten sind jeweils zumin-

dest konzeptuell als Erweiterungsmöglichkeiten zu unterstützen, soweit der hierfür erforderliche Mehraufwand nicht erheblich ist.

Das Ausmaß der vorgesehenen Unterstützung für Lokalisierung ist im Pflichtenheft zu beschreiben.

1.2 Architektur

Wichtige Eigenschaften der zu entwickelnden Software sind die langfristige und einfache Verwendung, eine einfache Codeverifizierung/-wartung und Offenheit gegenüber Weiterentwicklungen und Erweiterungen.

Dies soll auf verschiedene Weisen erreicht werden:

- Modularisierung
- Versionsverwaltung für Quellcode und Dokumentation
- im Code eingebettete Entwickler-Dokumentation (Kommentare)
- zukunftssichere Programmiersprachen (verbreitet und plattformunabhängig)
- Debugfähige Stand-Alone Version

1.2.1 Modularisierung

Modularität bedeutet, dass die Software in Form einzelner Teile/Module entwickelt wird, die von den jeweils anderen Implementierungen weitgehend unabhängig klar umrissene Teilaufgaben übernehmen. Die Verknüpfung der einzelnen Module erfolgt über Schnittstellen (API), die einen definierten Datenaustausch zulassen. Dieses Konzept hat viele Vorteile, die gerade bei einer langfristig und auf Weiterentwicklung ausgelegten Software sinnvoll sind:

- Einzelne Module können einzeln und parallel zu anderen von verschiedenen Entwicklern weiterentwickelt und ausgetauscht werden. Dies ist besonders bei Weiterentwicklungen der fachlichen Module relevant, um die konsistente Produktion von Ergebnissen auch zwischen verschiedenen Versionen zu gewährleisten. Die Oberfläche kann beispielsweise zur Unterstützung zusätzlicher Sprachen weiterentwickelt werden, ohne die Berechnungsalgorithmen erneut validieren zu müssen.
- Die Komplexität der einzelnen Module ist geringer als die der gesamten Software, wodurch sich Entwickler einfacher und schneller zurecht finden können. Auch die Softwaretests werden erleichtert, da Module getrennt voneinander verifiziert werden können,

- Die Portierung auf neue Zielsysteme (z.B. Windows-Applikation, Linux-Applikation, Web-Applikation) oder Verteilung des Gesamtablaufs auf mehrere Systeme (Client/Server-Betrieb) wird erleichtert.
- Die Wiederverwendbarkeit von einzelnen, praxiserprobten Teilmodulen auch über Projekte hinweg anstelle von Neuentwicklungen reduziert den Entwicklungsaufwand, die Fehlerwahrscheinlichkeit und erleichtert die Wartung.

Daher ist die umzusetzende Aufgabenstellung im Rahmen der Pflichtenhefterstellung unter funktionalen und technischen Aspekten zu gliedern und ein Modulkonzept zu entwerfen.

1.3 Programmiersprachen

Eine möglichst einheitliche Software-Landschaft erleichtert die Wiederverwendung von Quellcode und Bibliotheken und reduziert den langfristigen Wartungsaufwand. Daher wird auch die Nutzung einer einheitlichen Programmiersprache angestrebt, sofern nicht spezifische Anforderungen dem entgegenstehen.

1.3.1 Standard-Programmiersprache ECMAScript / JavaScript

Die Software ist in ECMAScript / JavaScript zu erstellen.

Der Sprachstandard wird fortlaufend weiter entwickelt. Als Mindestniveau ist ECMAScript 2015 (auch als ECMAScript 6 / ES6 bezeichnet) zu verwenden, da in dieser Version wichtige Sprachbestandteile hinzugefügt wurden, die Wartung und Wiederverwendung von Code deutlich erleichtern, v. a. Klassen und Module.

Spätere Versionen des Sprachstandards können herangezogen werden, insbesondere wenn die zwischenzeitlichen Weiterentwicklungen für die Umsetzung relevant sind. Der genutzte Sprachstandard ist zu dokumentieren. Die Laufzeitkompatibilität zur Zielumgebung ist dann ggf. durch entsprechende Werkzeuge, z. B. Babel (https://babeljs.io/), herzustellen.

1.3.2 Einsatz anderer Programmiersprachen

Durch das jeweilige Einsatzgebiet der Software können Abweichungen und/oder Ergänzungen sinnvoll sein. In der folgenden Tabelle sind Optionen für die Erstellung der Software vorgegeben.

[ ] Es dürfen andere Programmiersprachen gewählt werden, wenn darin bereits vorliegender, Lizenz-kompatibler Quellcode, zu dem kein Äquivalent in JavaScript existiert, die Realisierung deutlich wirtschaftlicher, schneller, kompatibler oder stabiler macht.

[ ] Es dürfen andere Programmiersprachen gewählt werden, weil sich spezifische Funktionalitäten nur darin umsetzen lassen (z. B. Physik-Simulationen auf GPU)

Die mögliche Abweichung von JavaScript als geforderte Programmiersprache gilt:

[ ] Für Teile des Projekts, wobei Anteile, Abgrenzungen und Schnittstellen der verschiedenen Teilsysteme zu skizzieren und motivieren sind.

[ ] Für das gesamte Projekt.

1.4 Tests

Die Lauffähigkeit und die Validität der zu entwickelnden Software ist durch geeignete Verfahren zu gewährleisten. Die eingesetzten Verfahren müssen begründet und dokumentiert werden.

1.4.1 Arten

Für die Software sind Komponententests, Integrationstests und Systemtests zu erstellen. Dabei sind geeignete Testverfahren auszuwählen und zu begründen, warum diese verwendet werden. Die Testverfahren müssen sowohl Positivtests als auch Negativtests umfassen. Es sind Regressionstests auszuführen.

1.4.2 Abdeckung

Es ist eine möglichst hohe Testabdeckung zu erzielen, wobei gleichzeitig der Aufwand für die Durchführung der Tests möglichst gering zu halten ist. Hierfür sind Testverfahren zu automatisieren, wenn es realisierbar und sinnvoll ist. Mit jedem Testverfahren ist eine für dieses Verfahren übliche Testabdeckung zu erzielen. Alle funktionalen und alle nicht-funktionalen Anforderungen sind zu testen.

1.5 Lizenz und Rechte

Die zugrundeliegende lizenzrechtliche Entwicklung ist für die zu entwickelnde Software sowie alle weiteren Module zu dokumentieren.

1.5.1 Lizenzvorgabe: GNU General Public License Version 3

Die zu erstellende Software ist als “freie Software” unter der GNU General Public License der Version 3 aus dem Jahr 2007 (GNU GPLv3) Lizenz zu entwickeln. Die Software unterliegt dem Copyleft, sodass alle Rechte bei Weitergabe, Änderung, Erweiterung und Wiederverwertung erhalten bleiben.

Die Lizenzierung ist in einem geeigneten Format und im vollem Umfang zu dokumentieren.

1.5.2 Einbindung Drittsoftware

Falls zur Realisierung der Software auf Drittsoftware, z. B. Bibliotheken, zurückgegriffen wird, ist durch den Auftragnehmer sicherzustellen, dass diese Nutzung durch die Lizenz(en) der Drittsoftware abgedeckt ist und dass durch die Einbindung dieser Drittsoftware keine Einschränkungen für die zu entwickelnde Software entstehen.

Sofern die Drittsoftware jenseits der lizenzrechtlichen Fragen nicht auch vollumfänglich den nicht-funktionalen Anforderungen an die zu entwickelnde Software bezüglich Dokumentation etc. entspricht, ist die Verwendung jeweils im Vorfeld mit dem Auftraggeber abzustimmen.

1.5.3 Einbringung eigener existierender Software

Eigene existierende Software ist bei Einbringung im Quellcode zu überlassen. Der Auftraggeber ist vor dem Einbau schriftlich zu informieren und über die lizenzrechtlichen Folgen sowie möglichen Konflikten hinzuweisen. Die eigene existierende Software muss alle an die zu entwickelnde Software gestellten Anforderungen in Bezug auf Lizenz, Dokumentation und der Funktionsfähigkeit erfühlen. Die eigene existierende Software ist vollständig im Funktionsumfang, Schnittstellen und Validität zu dokumentieren und zu überlassen. Bei Einbringung von eigener existierender Software ist zu begründen, warum diese eingebracht wird.

1.6 IT-Grundschutz

Als Richtlinie zur Absicherung der zu entwickelnden Applikation ist das IT-Grundschutz-Kompendium des Bundesamtes für Sicherheit in der Informationstechnik (BSI) in der aktuellsten Version heranzuziehen. Eine genaue Beschreibung aller in diesem modular aufgebauten Kompendium enthaltenen Bausteine ist auf der Internetseite des BSI zu finden:

https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKompendium/itgrundschutzKompendium_node.html

Hierbei ist zunächst ein projektübergreifendes definiertes Mindestniveau einzuhalten. Darüber hinaus können sich projektspezifisch weitergehende Anforderungen ergeben, die im entsprechenden Teil dieses Lastenhefts aufgeführt werden.

Im Rahmen der Erstellung des Pflichtenhefts sind vom Anbieter ggf. weitere relevante Bausteine zu identifizieren und zu benennen, die sich z. B. aufgrund des vorgestellten Lösungsansatzes zusätzlich ergeben.

1.6.1 Mindestniveau

Um das Mindestniveau des IT-Grundschutzes zu erreichen, müssen folgende Bausteine des Kompendiums umgesetzt werden:

- OPS.1.1.2 Ordnungsgemäße IT-Administration
- OPS.1.1.3 Patch- und Änderungsmanagement
- OPS.1.1.4 Schutz vor Schadprogrammen
- OPS.1.1.5 Protokollierung
- OPS.1.1.6 Software-Tests und -Freigaben
- CON.5 Entwicklung und Einsatz von Allgemeinen Anwendungen

1.6.2 Protokollierung der Umsetzung

Die Anwendung der verschiedenen Bausteine des IT-Grundschutz-Kompendiums ist im Rahmen des Projektverlaufs zu protokollieren. Dabei ist insbesondere zu dokumentieren, inwiefern die Anwendbarkeit der Einzelbausteine evaluiert und ggf. begrenzt wurde. Weiterhin ist die Anwendung bzw. Nicht-Anwendung von als optional markierten Empfehlungen zu beschreiben und zu begründen.

1.7 Dokumentation

Alle Bestandteile sind ausführlich und verständlich in der Standardprojektsprache zu dokumentieren, sodass für den entsprechenden Nutzer eine zeitnahe Einarbeitung in die Bestandteile wie das Handbuch, die Systemarchitektur, die Softwarearchitektur und den Quellcode möglich ist.

1.7.1 Handbuch

Das Handbuch muss zu Beginn ein Inhaltsverzeichnis, eine Einleitung und ein Kapitel zur Installation und Einrichtung der Software enthalten.

In der Einleitung sind Zielgruppe, mögliche Anwendungsbereiche und ggf. erforderliche Vorkenntnisse zu erwähnen. Zudem ist der Aufbau des Handbuches kurz zu schildern.

Das Kapitel zur Installation und Einrichtung der Software muss die Installation der Software für jede zu unterstützende Zielumgebung schrittweise erklären. Außerdem müssen eventuelle Konfigurationsmöglichkeiten vollständig erläutert werden.

Der Hauptteil des Handbuches muss ein Tutorial und einen Referenzteil umfassen. Das Tutorial ist nach den verschiedenen grundlegenden zu unterstützenden Arbeitsabläufen zu gliedern. Jeder dieser Arbeitsabläufe ist anhand von Beispielen und der vorliegenden Software schrittweise zu durchzuspielen.

Der Referenzteil ist entsprechend der Softwarearchitektur zu gliedern. Jedes von außen ansprechbare Modul der Software muss hier aufgeführt werden. Dabei sind die mit diesem Modul umgesetzten Aktionen und deren mögliche Ergebnisse zu erläutern. Außerdem sind die vom Benutzer eingebbaren Parameter mit ihren gültigen Wertebereichen und ihren Auswirkungen auf das Modul zu beschreiben. Der Referenzteil muss ebenfalls eine Fehlerbehandlung umfassen. In dieser sind bekannte Fehler zusammen mit ihren möglichen Ursachen zu nennen und ein Vorschlag zur Problemlösung darzulegen. Falls nicht gängige Abkürzungen oder Fachbegriffe verwendet werden, muss das Handbuch ein Abkürzungsverzeichnis bzw. ein Glossar enthalten.

1.7.2 Systemarchitektur

Die verschiedenen verwendeten Systeme sind zu kurz zu erläutern und ihr Zusammenspiel untereinander ist anhand von Diagrammen darzustellen und zu beschreiben. Dabei sind Eigenschaften sowie Vor- und Nachteile der verwendeten Systemarchitektur bezüglich der nicht-funktionalen Anforderungen darzulegen.

1.7.3 Softwarearchitektur

Die Architektur des Quellcodes ist mit geeigneten UML-Strukturdiagrammen und UML-Verhaltensdiagrammen ausführlich darzulegen und anhand dieser kurz zu beschreiben.

1.7.4 Quellcode

Alle exportierten Module sind mit Dokumentationskommentaren zu versehen. In diesen muss zu jedem Modul die Version und der Autor notiert sein. Für jedes dieser Module ist der Zweck, die Einsatzmöglichkeiten und eventuelle Grenzen zu beschreiben. Außerdem müssen, falls vorhanden, Eingangsparameter und Rückgabeparameter bezüglich ihres Zwecks, ihres Typs, ihrer Struktur und ihrer gültigen Wertemenge beschrieben werden. Ebenfalls müssen explizit im Code geworfene Fehler mit dem Grund, warum diese ggf. geworfen werden, genannt werden.

Zusätzlich zu den Dokumentationskommentaren sind Kommentare zu ansonsten schwer zu verstehenden Codezeilen oder –Blöcken zu schreiben.

2 Funktionale Module

2.1 Berechnungsprogramme

Dieses Kapitel beinhaltet grundsätzliche Festlegungen für alle Arten von Programmen, bei denen die Berechnung von Werten aus Eingangsgrößen eine Kernfunktionalität darstellt.

2.1.1 Datenformate

Um einen möglichst reibungsfreien Austausch von Daten mit anderen Programmen zu gewährleisten, sollten möglichst gängige Formate verwendet werden, für die eine Vielzahl von Parser-Implementierungen existiert. Textbasierte Formate und dabei insbesondere solche, die auch ohne Parser durch Entwickler oder Benutzer interpretierbar sind, sind dabei zu bevorzugen.

In Ausnahmefällen kann hiervon abgewichen werden, um die Kompatibilität zu bereits existierenden Programmen zu gewährleisten. Hier ist aber die Schaffung eines gesonderten Parsers oder eines zweiten Ausgabeformates zu bevorzugen.

2.1.2 Fehlerbehandlung

Fehler im Berechnungsablauf sollen an allen Stellen explizit abgefangen und behandelt werden.

Dies beinhaltet zuerst die klare Definition und Überprüfung der Wertebereiche für Eingangsparameter und die Behandlung widersprüchlicher oder unsinniger Parameterkombinationen.

Sofern eine Berechnung mehrere Schritte umfasst, also explizit oder implizit mit Zwischenergebnissen gearbeitet wird, sind diese als Eingangsparameter für die nachfolgenden Berechnungsschritte zu werten und entsprechend analog zu behandeln.

Hierbei ist insbesondere auch auf Definitionslücken und Genauigkeitsbereiche der zugrundelegenden Berechnungsfunktionen zu achten. Sofern die Berechnungsfunktionen neben den Ergebnissen z. B. auch Fehlerstatuswerte zurückliefern können, sind diese abzufragen und angemessen zu behandeln.

Für den zu erstellenden Code sind explizite Exceptions als Mechanismus der Fehlerbehandlung zu bevorzugen, weil dadurch das Risiko einer versehentlichen Weiterführung des Programmablaufs mit ungültigen Daten reduziert wird.

Die Fehlerbehandlung beinhaltet die möglichst genaue Beschreibung und Lokalisation des Fehlers. Bei problematischen Eingangsdaten sollten nach Möglichkeit auch Ansätze zur Problemvermeidung gegeben werden.

2.2 Grafische Oberfläche

Dem Endanwender soll eine komfortable, heutigen Oberflächenstandards entsprechende Nutzung ermöglicht werden. Dies kann (unter Berücksichtigung der übrigen Anforderungen) grundsätzlich in Form einer nativen Applikation oder auch browserbasiert realisiert werden.

Programme, die vom jeweiligen Benutzer interaktiv genutzt werden, sollten daher bezüglich ihrer Ausgestaltung den folgenden allgemeinen Anforderungen entsprechen.

2.2.1 Bedienbarkeit

Die Bedienung des Programms sollte mit verschiedenen Eingabegeräten möglich sein:

- Maus
- Tastatur
- Touchscreen (Finger oder Touchpen)

Nach Möglichkeit sollten die Eingabemöglichkeiten innerhalb derselben Benutzeroberfläche kombinierbar sein, um unterschiedlichen Benutzerpräferenzen entgegen zu kommen und den Umstellungsaufwand zu minimieren. Beispielsweise sollte ein explizit aktivierbarer Touch-Modus mit verändertem Bildschirmlayout nur umgesetzt werden, wenn ein gemeinsames Layout z. B. aufgrund der Menge an Eingabefeldern nicht praktikabel ist bzw. mit negativen Auswirkungen für die anderen Eingabemodi einhergehen würde.

Zu Eingabefeldern sollten z. B. beim Überfahren mit der Maus erläuternde Texte eingeblendet werden, in denen auf Wertebereiche und unterstützte Formate eingegangen wird.

Assistenz-Technologien wie Screen Reader sind durch eine sinnvolle Strukturierung der Oberfläche und Hinterlegung entsprechender Meta-Informationen zu Bedienelementen zu unterstützen. Das hierbei angestrebte Ausmaß an Barrierenreduzierung / Barrierefreiheit, sofern nicht ohnehin im spezifischen Teil des Lastenhefts näher vorgegeben, ist im Rahmen des Pflichtenhefts zu dokumentieren.

2.2.2 Standard-Bibliotheken

Für die Realisierung der Benutzeroberfläche soll auf eine vordefinierte Auswahl an Standard-Bibliotheken zurückgegriffen werden. Ähnlich wie die Standardisierung der Programmiersprache erleichtert dies die Wartung und Integration von Code. Weiterhin erleichtert ein gemeinsames Look & Feel der verschiedenen Programme die Benutzung.

Durch die hier getroffene Auswahl der Bibliotheken wird weiterhin die Plattform-Unabhängigkeit gefördert: unabhängig von Betriebssystem, Browser und Version erscheint die Anwendung im einheitlichen Design.

Für browserbasierte Anwendungen ist React (https://reactjs.org/) in Kombination mit der Komponenten-Bibliothek react-md (https://react-md.mlaursen.com/) zu benutzen. Eigene Elemente der Benutzeroberfläche sind aus den dort verfügbaren Komponenten durch Komposition zu erzeugen oder – falls dies nicht möglich ist – an das zugrundeliegende „Material Design“ (https://material.io/) anzulehnen.

Die Datenhaltung der Applikation ist auf Basis von Redux (https://redux.js.org/) und React-Redux (https://react-redux.js.org/) unter Berücksichtigung der jeweiligen „best practices“ zu realisieren.

Die Bibliotheken sind jeweils in der aktuellsten vorliegenden stabilen Version zu verwenden. Bei Inkompatibilitäten aufgrund unterschiedlicher Veröffentlichungszyklen sind sinnvolle Abwägungen zu treffen.

2.2.3 Aufgabenangemessenheit

Die Funktionen der Oberfläche müssen den Benutzer so unterstützen, dass er seine Aufgabe effizient und effektiv erledigen kann. Hierfür sind Abläufe, die ohne eine Benutzereingabe erfolgen können, zu automatisieren. Ausnahmen bilden hierbei Aktionen, die unwiederbringliche Änderungen hervorrufen. Für diese sind vorher vom Benutzer zu bestätigende Rückfragen zu stellen.

Außerdem müssen vom Benutzer einzugebende Konfigurationen mit sinnvollen Standardwerten vorbesetzt werden. Eingegebene Konfigurationen, die nicht zwangsläufig spezifisch für einen einmaligen Gebrauch sind, müssen gespeichert, geladen und gelöscht werden können. In Eingabeformularen werden dem Benutzer zunächst immer nur die notwendigsten und gängigsten Spezifikationen zur Verfügung gestellt. Weitere Spezifikationen sind in einem Eingabeformular über ein Bedienelement für ein erweitertes Eingabeformular aufrufbar.

Der aktuelle Fokus ist entsprechenden des erwarteten Arbeitsablaufes automatisch nach jeder Eingabe so präzise wie möglich neu zu setzen und grafisch hervorzuheben.

In der Oberfläche dürfen keine überflüssigen Informationen angezeigt werden.

2.2.4 Selbstbeschreibungsfähigkeit

Der Benutzer muss in der Lage sein die Oberfläche intuitiv zu nutzen. Dafür müssen in der Oberfläche zu jeder Zeit Orientierungspunkte angezeigt werden, sodass für den Benutzer immer genau ersichtlich ist, an welcher Stelle in der Oberfläche er sich befindet. Außerdem muss in jedem Zustand deutlich erkennbar sein, welche Handlungen wie ausgeführt werden können bzw. müssen. Dies beinhaltet unter anderem, dass Navigationselemente durch ihre Gestaltung schnell und einfach zu identifizieren sind und eindeutig angeben, wohin sie navigieren. Jedes aus Sicht des Nutzers komplexere Element ist mit Kontexthilfen zu versehen.

Für jede Eingabe muss in irgendeiner Form eine Rückmeldung über ihren Erfolg erbracht werden.

2.2.5 Erwartungskonformität

Die Darstellungen in der Oberfläche müssen einheitlich sein, allgemeinen Konventionen entsprechen und konsistent zueinander sein. Insbesondere sind Bedienelemente, für die es Icons gemäß allgemeinen Konventionen gibt, auch mit diesen Icons umzusetzen. Zudem müssen gleichartige Bedienelemente gleichartige Bedeutungen in der gesamten Oberfläche haben. Darüber hinaus dürfen Bedienelemente, die bei einem aktuellen Zustand einer Oberfläche keine Auswirkungen haben oder haben sollen, nicht nutzbar sein und sind von anderen Bedienelementen grafisch abzuheben.

Die Bearbeitungszeiten für die verschiedenen Funktionalitäten der Oberfläche müssen auch für verschiedene Eingaben vorhersehbar sein.

2.2.6 Steuerbarkeit

Der Benutzer muss die Oberfläche eigenständig starten, sowie die Richtung und Geschwindigkeit der Navigation in dieser beeinflussen können. Die Oberfläche muss zu jeder Zeit verlassen und beliebig neugestartet werden können. Die Startseite der Oberfläche

muss von jedem Punkt aus durch ein einzelnes Navigationselement erreichbar sein. In der Oberfläche sollte wenigstens der letzte Schritt rückgängig gemacht und wiederhergestellt werden können. Diese Funktionalität muss nicht explizit auf Anwendungsebene realisiert werden, sofern sie z. B. bereits von Eingabefeldern selbst angeboten wird oder das Wiederherstellen des vorherigen Zustands trivial möglich ist. Hier ist eine Abwägung zwischen gewonnenem Bedienkomfort einerseits und Reduzierung der Oberflächen- und Implementierungskomplexität andererseits zu treffen.

Zudem muss jeder Prozess unterbrochen werden können, der länger als einige Sekunden dauert oder bei dem der Benutzer aus anderen Gründen ein Interesse an einem manuellen Abbruch haben könnte (z. B. Abbruch einer versehentlichen Datenübermittlung vor Vervollständigung/Kontrolle der Eingaben).

Während der Benutzer Datenänderungen in der Oberfläche vornimmt, müssen die Originaldaten weiterhin einsehbar bleiben, falls dies hierfür erforderlich oder von Vorteil ist.

2.2.7 Lernförderlichkeit

Die Oberfläche hat den Benutzer beim Erlernen des Umgangs mit ihren Funktionalitäten zu unterstützen. Hierfür müssen ihre Funktionalitäten durchschaubar bleiben.

2.2.8 Fehlertoleranz

Die Funktion, Leistung und Sicherheit der Oberfläche muss auch aufrechterhalten werden, wenn unvorhergesehene Eingaben oder Fehler in der Software auftreten. Hierfür müssen unter anderem Eingaben plausibilisiert werden.

Aufgetretene Fehler müssen für den Benutzer so erläutert werden, dass dem Benutzer bei Eigenverschuldung eine leichte Fehlerkorrektur ermöglicht wird. Bei nicht kritischen Fehlern sollte eine solche Erläuterung erst nach expliziter Aufforderung des Benutzers angezeigt werden.

2.2.9 Individualisierbarkeit

Folgende Eigenschaften muss ein Benutzer für die Individualisierung der Oberfläche einstellen können:

[X] Schriftgröße

[ ] Farbschema

[ ] Farbschema für Rot-Grün-Sehschwäche

[ ] Umfang der Erläuterungen und Kontexthilfen

2.2.10 Browserbasierte Oberflächen

Dem Endanwender soll eine komfortable, heutigen Oberflächenstandards entsprechende Nutzung ermöglicht werden. Dies kann (unter Berücksichtigung der übrigen Anforderungen) grundsätzlich auch in Form einer browserbasierten Oberfläche realisiert werden. Hierbei sind die aktuellen und weit verbreiteten Browser zu unterstützen. Die unterstützen Browser und deren jeweils erforderliche Version ist in der Dokumentation als Mindestanforderung zu benennen.

3 Spezifische Anforderungen

Die in diesem Kapitel definierten Anforderungen beziehen sich spezifisch auf die Umsetzung der RDO Beton und die hierzu angedachte Softwarelösung.

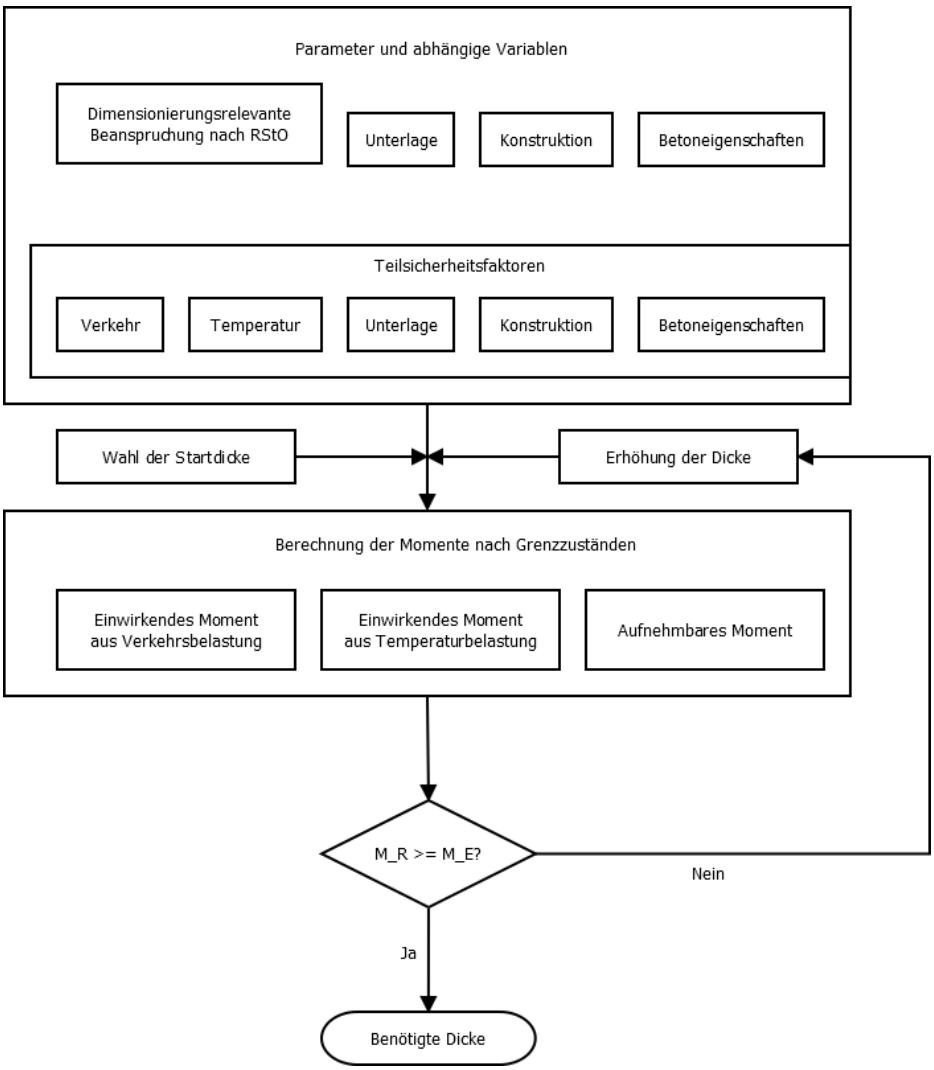
3.1 Berechnung

Die Umsetzung der RDO Beton, d. h. die Dimensionierung auf Basis der Benutzereingaben, stellt die fachliche Kernfunktionalität der zu entwickelnden Software dar.

3.1.1 Umsetzung RDO Beton

Grundlage für die Umsetzung des Projektes ist die Entwurfsfassung der RDO Beton, i.e. RDO Beton 09, Ausgabe 2018 [Forschungsgesellschaft für Straßen- und Verkehrswesen 2018-05-07]. Eine Weiterentwicklung der Dimensionierungspraxis ist nicht vorzunehmen.

Die in Rand- oder Übergangsbereichen auftretenden freien Plattenränder sind entsprechend Abschnitt 4.7 „Querkraftübertragung“ durch nicht ansetzen des Dübelfaktors, m_{bD} = 1, zu berücksichtigen. Eine über die Funktionalität der RDO Beton hinausgehende Logik, wie etwa der Hinweis auf eventuell notwendige Endsporne ist nicht vorgesehen.



1. Abbildung: Möglicher Ablaufplan der RDO Beton.

Dimensionierungsgröße

Die zu dimensionierende Variable ist die Deckendicke h_d. Die Dimensionierung erfolgt in Schritten von 1 mm. Eine Dimensionierung anderer Variablen (E-Modul, Festigkeit) ist derzeit nicht vorgesehen. Für die Dimensionierung bietet sich ein naives iteratives Verfahren an, es können aber andere Verfahren (Bisektion, Newton) verwendet werden. Hierdurch darf die Allgemeingültigkeit nicht beschränkt werden.

Codequalität

Grundsätzlich soll sich durch die strukturierte und gut dokumentierte Programmierweise die Möglichkeit bieten, dass sich ein erfahrener Programmierer in kurzer Zeit in den Code einarbeiten und Änderungen vornehmen kann. Die Implementierung soll möglichst modular und übersichtlich vorgenommen werden. Dafür bietet sich vor allem die Unterteilung der Berechnung in die beiden einwirkenden Momente aus Verkehr und Temperatur sowie

das aufnehmbare Moment an. Weitere Modularisierungsmöglichkeiten ergeben sich beispielsweise für die virtuelle Ersatzaufstandsfläche, die elastische Länge und dergleichen.

Die Parametrisierung der Berechnung ist durch Eingabedateien zu steuern, die von Menschen gelesen und einfach verändert werden können. Die Eingabedateien werden auch durch die grafische Benutzeroberfläche geschrieben werden können. Auf eine strukturierte Angabe der Parameter ist zu achten. Weiterhin bietet sich aus Gründen der Übersichtlichkeit an, die benötigten Parameter zu logischen Einheiten zusammenzufassen und somit die Übersichtlichkeit weiter zu steigern. Dafür bieten sich in den meisten Programmiersprachen zur Verfügung stehende Strukturen oder Objekte an.

Soweit möglich soll eine Fehlbedienung, etwa durch Angabe sinnloser Parameter, verhindert werden.

3.2 Grafische Oberfläche

Die Modularisierung der Berechnung soll sich in der grafischen Oberfläche widerspiegeln. Dazu bietet sich beispielsweise die Verwendung von Reitern, oder einer Baumstruktur an, um den Benutzer durch den Eingabeprozess zu leiten.

Laut Protokoll der Startsitzung vom 18.06.2019, sollen die Eingabemasken die Verwendung unterschiedlicher Sprachen zulassen. Die Codetabellen, etwa für Tooltipps bei Mouse over, werden allerdings nur für Deutsch vorbereitet.

3.2.1 Querschnittsbilder

Zur Vermeidung von konstruktiven Planungsfehlern (z.B. aufgrund der Lage der Fugen in Relation zu der erforderlichen Markierung) wird gefordert, dass die Plattenbreite grafisch dargestellt wird. Eine über die funktionale Beschreibung der RDO Beton hinausgehende Plausibilitätsprüfung, etwa gegen die Richtlinien für die Anlage von Autobahnen (RAA) und die Richtlinien für die Markierung von Straßen, Teil A: Autobahnen (RMS-A) ist nicht notwendig. Weiterhin darf nicht suggeriert werden, dass die Dimensionierung Bezug auf den Querschnitt nimmt. Die zugrundeliegenden Annahmen der Plattentheorie gelten weiterhin nur für die Einzelplatte.

3.2.2 Systembilder für Sonderlasten

In Analogie zu den Querschnittsbildern für Regelbauweisen sind Systembilder für Sonderlasten vorzusehen. Dies kann in Form von möglichst aussagekräftigen Piktogrammen geschehen. Möglich ist beispielsweise die abstrahierte Darstellung eines Containerumschlagplatzes in Form eines Portalkrans mit Container. Die konkrete Wahl eines Piktogramms geschieht in Absprache zwischen AG und AN. Der AN legt hierzu ca. 3 Vorschläge vor.

Das Systembild für Sonderlasten ersetzt das Querschnittsbild für Regelbauweisen sowohl in der GUI, als auch im gedruckten Bericht.

3.2.3 Druckfunktion

Es ist eine Druckfunktion vorzusehen, bei der alle Eingabedaten und die Berechnungsergebnisse kompakt (d. h. nach Möglichkeit auf einer DIN A 4 Seite) dargestellt werden. Hierbei sind die maßgeblichen Eingangsparameter, sowie das spezifische Querschnittsbild oder das Systembild für Sonderlasten anzugeben, bzw. darzustellen.

Nicht relevante Schaltflächen bzw. allgemein interaktive Elemente der Oberfläche sind dabei auszublenden.

Die Zuständigkeit der Anwendung beschränkt sich dabei auf das Layout. Für die Druckerauswahl und die eigentliche Ausgabe können die entsprechenden Browser-eigenen Routinen als vorhanden vorausgesetzt werden.

3.2.4 Datenverwaltung

Eine explizite Datenverwaltung (auf Dateibasis, in einer lokalen Datenbank oder Servergestützt) ist in der gegenständlichen Ausbaustufe nicht umzusetzen, aber nach Möglichkeit bei der Konzeption der Anwendung bereits zu berücksichtigen.

Es ist eine komfortable Lösung zu realisieren, mit der die eingegebenen Eingangsdaten in ihrer Gesamtheit in die Zwischenablage kopiert bzw. aus dieser wieder hergestellt werden können. Um den Versand per E-Mail ebenso wie die Ablage in einer Datei zu ermöglichen, ist dabei ein textbasiertes Austauschformat zu wählen.

3.3 Inbetriebnahme

Die Inbetriebnahme beschreibt den Prozess von der Fertigstellung des Programms durch den Auftragnehmer bis zum Regelbetrieb beim Auftraggeber.

3.3.1 Bereitstellung

Das erstellte Programm ist vom AN in Form einer ZIP-Datei bereitzustellen, die auf oberster Verzeichnisebene eine HTML-Datei zum Ausführen der Applikation enthält.

Sämtliche Programm- und Ressourcendateien sind in Unterordner zu platzieren, um die Übersichtlichkeit zur wahren. Pfadangaben (z. B. zum Laden JavaScript-Datei aus HTML-Datei) sind relativ anzugeben.

3.3.2 Installation

Als browserbasierte Applikation und aufgrund der geforderten Bereitstellung ist eine Installation nicht erforderlich. Die ZIP-Datei kann von Benutzern ohne Administrator-Rechte auf den jeweiligen Zielrechnern entpackt und die Applikation (bei üblicher Systemkonfiguration) durch Doppelklick auf die HTML-Datei gestartet werden.

Die Bereitstellung der ZIP-Datei auf geeigneten Wegen geschieht intern durch den AG.

3.4 Schulung

Die Zielgruppen der Schulung sind entweder Experten, die mit der zugrundeliegenden Theorie der RDO Beton vertraut sind oder aber Praktiker, die nur an den Berechnungsergebnissen, nicht aber am Rechenweg interessiert sind. In jedem Fall sind die zu schulenden Mitarbeiter mit der Nachweisführung der RDO Beton bereits vertraut. Eine Rekapitulation der RDO Beton oder fachliche Diskussion bezüglich der Dimensionierung ist insofern nur soweit Schulungsinhalt, wie es zur Bedienung des Programms unbedingt erforderlich ist.

3.4.1 Schulung vor Ort

Es ist zum Projektabschluss hin eine Schulung in den Räumen der BASt vorzusehen, um voraussichtlich 5 – 10 Benutzer in der Anwendung der erstellten Software zu unterweisen. Hierfür wird ein Umfang von ca. 4 Stunden zuzüglich einer halben Stunde Pause angesetzt.

Die Einladung zur Schulung erfolgt durch den AG (nach vorheriger Abstimmung des Termins mit dem AN).

Die Schulung sollte zu etwa gleichen Teilen eine Frontalpräsentation und eine interaktive Übung durch die Benutzer beinhalten. Die hierzu erforderlichen Arbeitsplatzrechner werden durch den AG gestellt oder durch die Teilnehmer selbst mitgebracht. Die Installation der Software auf den Rechnern erfolgt im Rahmen der Schulung vor Ort (z. B. vor Beginn oder während einer Pause).

Der Auftragnehmer bringt den Präsentationsrechner mit. Ein Beamer wird durch den AG zur Verfügung gestellt. Die für die Schulung erstellten Präsentationsfolien sind nach Abschluss der Schulung dem AG in digitaler Form zur weiteren internen Verwendung zu überlassen.

Zur Sicherstellung eines geeigneten Betreuungsverhältnisses wird der AN mit mindestens 3 Mitarbeitern auftreten. Während der gesamten Schulung sollten auch von allen beteiligten Unterauftragnehmer mindestens jeweils Mitarbeiter zugegen sein.

Pflichtenheft RDO Beton

DTV-Verkehrsconsult GmbH

INHALTSVERZEICHNIS

Inhalt

DOKUMENTENDATEN

Projekt	Umsetzung der rechnerischen Dimensionierung gemäß der RDO Beton als Software für Endnutzer
Auftraggeber	Bundesanstalt für Straßenwesen (BASt) Projektnummer: **FE 04.0316/2018/ORB**
Anbieter	**DTV-Verkehrsconsult GmbH** ISAC GmbH Projektnummer: **06-0460**

Abstimmung Bundesanstalt für Straßenwesen (BASt), DTV-Verkehrsconsult GmbH und ISAC GmbH

BEREICHSZUSTÄNDIGKEIT

Projektphase	Verantwortlicher	Mail
Projektmanagement	Dr. Thorsten Kathmann	kathmann@dtv-verkehrsconsult.de
Qualitätssicherung	Thorsten Hermes	hermes@dtv-verkehrsconsult.de
Erstellung Pflichtenheft	Jörg Stöver	stoever@dtv-verkehrsconsult.de
Umsetzung GUI	Christoph Verbrigghe	verbrigghe@dtv-verkehrsconsult.de
Fachliche Umsetzung Rechenkern	Dr.-Ing. Johannes Neumann	neumann@isac-gmbh.com

Konzeption Rechenkern	Dr.-Ing. Jan Lehmkuhl	lehmkuhl@isac-gmbh.com
Implementierung Rechenkern	Jan Mirco Pfeifer	pfeifer@isac-gmbh.com

1 ZIELBESTIMMUNG

Ziel des Projekts ist die Umsetzung der rechnerischen Dimensionierung gemäß der [RDO Beton] als Software für Endnutzer in Form einer browserbasierten Anwendung. Die Software richtet sich an Benutzer, die ohne programmiertechnische Kenntnisse oder Einarbeitung jenseits einer 3.1.2 Schulung mit Hilfe der Software die Dicke von Betonflächen im Straßenbau berechnen.

1.1 MUSSKRITERIEN

- Die Berechnung gemäß der [RDO Beton] muss möglich sein.
- Der GUI-Client muss die in 3.2.4 GUI-Client näher beschriebenen Eigenschaften erfüllen.
- Der Rechenkern muss die in 3.2.6 Rechenkern und 4.1 Rechenkernfunktionen näher beschriebenen Eigenschaften erfüllen.
- Der Programmcode muss die in 3.2 Software näher beschriebene Qualität erreichen.
- Die Fehlerbehandlung muss die in 4.1 Rechenkernfunktionen näher beschriebene Qualität erreichen.

1.2 WUNSCHKRITERIEN

- Ein möglicher Aufruf bzw. Ansteuerung des Rechenkerns durch ein Command-Line-Interface, dass eine Nutzung ohne den GUI-Client erlaubt.

1.3 ABGRENZUNGS- UND AUSSCHLUSSKRITERIEN

- Die AWDSTAKO-Oberfläche soll nicht als Vorlage oder Richtlinie genutzt werden.

- Konzeptionell und technisch ist ein Grundgerüst für eine Datenverwaltung in Form von Datei-, Datenbank- oder Server angelegt. Die tatsächliche Implementierung der Verwaltung solcher Funktionen ist jedoch nicht Teil des Produkts.

- Andere Lokalisierungen als Deutsch/Deutschland (de_DE) sind nicht Teil des Produkts, jedoch sind alle Textelemente so gehalten, dass eine nachträgliche Lokalisierung in einer anderen Sprache möglich ist.

2 PRODUKTEINSATZ

2.1 ANWENDUNGSBEREICH

Die Software wird auf Arbeitsplatzrechnern in einem Browserfenster ausgeführt, um Berechnungen nach dem Regelwerk [RDO Beton] auszuführen. Dabei kann die Betondeckendichte berechnet werden, die bei der Konstruktion von Verkehrsflächen sowie Stellplätzen benötigt wird. Weitere Faktoren wie Nutzungsdauer oder Sonderbelastungen können dabei in die Berechnungen einfließen.

2.2 ZIELBENUTZER

Personen, die über das entsprechend nötige Fachwissen im Bezug auf die [RDO Beton] sowie durchschnittlich PC- und Softwarekenntnisse verfügen.

Da das Produkt für die Bundesanstalt für Straßenwesen (BASt) entwickelt wird, ist als Programmsprache Deutsch/Deutschland (de_DE) vorgesehen.

2.3 BETRIEBSBEDINGUNGEN

Das Produkt wird als typische Endanwendersoftware entwickelt und ist für einen entsprechenden Einsatz vorgesehen. Ein wartungsfreier Betrieb 24/7 ist grundsätzlich vorgesehen, aber nicht im Rahmen des Projektes garantiert.

Eventuelle Wartung und/oder Anpassung an zukünftige Änderungen in den grundlegenden Prozessen oder Daten können durch die DTV-Verkehrsconsult GmbH ausgeführt werden, soweit dazu entsprechende Vereinbarungen getroffen werden.

Die Implementierung der Software ist durchgehend so gehalten, dass Anpassungen und Erweiterungen durch andere Dienstleister möglich sind.

3 PRODUKTUMGEBUNG

Das Produkt ist generell für den Einsatz unter Microsoft Windows 10 auf einem mit für den Zeitpunkt des Projekts durchschnittlicher Hardware ausgestatteten PC entwickelt und vorgesehen. Als Browser werden Firefox ESR (64 Bit) sowie Google Chrome, jeweils in der zum Zeitpunkt der Fertigstellung der Implementierung aktuellsten Version, unterstützt. Die JavaScript-Funktionalität der Browser muss aktiviert sein. Spezielle Plugins, Add-Ons oder Einstellungen werden nicht vorausgesetzt. Die Lauffähigkeit unter anderen Betriebssystemen und Browsern wird nicht aktiv eingeschränkt, aber weder überprüft noch unterstützt.

Eine Sondersituation ergibt sich bei der Nutzung des CLI-Clients, wenn die erstellten Auswertungen Informationen über den Status des Programmcodes enthalten sollen. Dies ist zur Nachverfolgung von Rechenwegen sinnvoll und wünschenswert, um sicherstellen zu können, dass bei Berechnungen mit der offiziellen Version des Programms gearbeitet wurde. Um diesen Status zu generieren, wird die korrekte Installation des Open-Source-Programms Git vorrausgesetzt.

3.1 DOKUMENTATION

Alle Dokumentation ist in der deutschen Sprache erstellt.

3.1.1 HANDBUCH

Zusammen mit der fertigen Software wird ein Handbuch in digitaler Form zur Verfügung gestellt. Bei dem Handbuch handelt es sich um ein speziell für dieses Produkt produzierte Anleitung und Beschreibung zur internen Nutzung bei der Bundesanstalt für Straßenwesen (BASt). Das Handbuch ist folgendermaßen strukturiert:

1. Inhaltsverzeichnis
2. Einleitung
    - Beschreibung der Zielgruppe.
    - Beschreibung der Anwendungsbereiche.

- Beschreibung nötiger Vorkenntnisse.
- Übersicht über den Aufbau des Handbuchs.

3. Installation und Einrichtung
   - Installation unter Windows 10 in Einzelschritten.
   - Erklärung der vorhandenen Konfigurationsoptionen.
4. Selbstlernabschnitt
   - Gliederung in unterstützte Arbeitsabläufe.
   - Schrittweiser Durchlauf obiger Arbeitsabläufe anhand von Beispielen.
5. Referenzteil - Gliederung nach Softwaremodulen
   - Auflistung aller von außen ansprechbaren Module nach folgendem Schema:
     - Auflistung der jeweilig möglichen Aktionen und Ergebnissen.
     - Auflistung aller vom Benutzer spezifizierbaren Parametern (mit Gültigkeitsbereichen und Auswirkungen auf das Modul).
     - Auflistung bekannter Fehler sowie deren mögliche Ursachen und Vorschlag zur Problemlösung.
6. Systemarchitektur
   - Erläuterung der verwendeten Systemarchitektur.
   - Beschreibung von Zusammenspiel der Systeme anhand von UML-Strukturdiagrammen.
   - Darlegung von Vor- und Nachteilen der verwendeten Systemarchitektur.
7. Softwarearchitektur

- Erläuterung der verwendeten Softwarearchitektur.
- Beschreibung von Zusammenspiel der Software anhand von UML-Strukturdiagrammen.
- Darlegung von Vor- und Nachteilen der verwendeten Softwarearchitektur.

8. Glossar
   - Auflistung genutzter Fachbegriffe und Abkürzungen.
   - Auflistung aller genutzter Softwarebibliotheken.
   - Auflistung aller genutzter Softwarestandards, -spezifikationen und -protokolle.

3.1.2 SCHULUNG

Zum Projektabschluss findet eine Schulung für die Mitarbeiter der Bundesanstalt für Straßenwesen (BASt) zur Präsentation und zum Erlernen der produzierten Software statt. Ziel der Schulung ist es, den zukünftigen Nutzern das Produkt vorzustellen und in eine typische Nutzung einzuweisen.

Für die Schulung gelten folgende organisatorische Rahmenbedingungen:

- Der Termin wird zwischen der Bundesanstalt für Straßenwesen (BASt) und der DTV-Verkehrsconsult GmbH abgestimmt.
- Angesetzter Zeitrahmen für die Schulung ist ca. vier Stunden plus halbstündiger Pause.
- Die Gruppengröße beträgt zwischen fünf und zehn zu schulende Teilnehmer seitens der Bundesanstalt für Straßenwesen (BASt).
- Die DTV-Verkehrsconsult GmbH stellt für die Schulung mindestens drei Mitarbeiter zur Verfügung.
- Die ISAC GmbH stellt mindestens einen weiteren Mitarbeiter für die Schulung bereit.

Folgende logistische Rahmenbedingungen gelten:

- Schulungsort sind durch die Bundesanstalt für Straßenwesen (BASt) gestellte Räumlichkeiten, die durch die Bundesanstalt für Straßenwesen (BASt) festgelegt werden.
- Die Rechner für die Schulung werden durch die Bundesanstalt für Straßenwesen (BASt) oder Teilnehmer gestellt.
- Ein Beamer wird durch die Bundesanstalt für Straßenwesen (BASt) gestellt.
- Der Präsentationsrechner wird von der DTV-Verkehrsconsult GmbH zur Schulung mitgebracht.

Folgende inhaltliche Rahmenbedingungen gelten:

- Die Schulung setzt die Kenntnis und Verständnis der [RDO Beton] der zu schulenden voraus.
- Fachliche beziehungsweise inhaltliche Schulung in oder Diskussion der [RDO Beton] ist nicht vorgesehen.
- Die Schulung beinhaltet zu jeweils ähnlich großen Teilen eine Präsentation sowie eine interaktive Übung bei der die Teilnehmer die Software unter Anleitung selber bedienen.

Im Anschluss werden die genutzten Folien und andere digitale Präsentationsmedien der Bundesanstalt für Straßenwesen (BASt) zur internen Nutzung zur Verfügung gestellt.

3.2 SOFTWARE

3.2.1 GENERELLE ASPEKTE

Die Software wird als ZIP-Datei ausgeliefert, bei der im obersten Verzeichnis eine HTML-Datei zum Start der Applikation existiert. Administratorrechte sind nicht erforderlich.

Eigene existierende Software wird bei Einbringen Teil des Produkts und unterliegt allen definierten Eigenschaften des Produkts. Es wird begründet, warum eigene Software eingebracht wird.

Als Lizenzmodell ist für alle Programmteile gilt [GPL v3]. Die Software wird modular entwickelt.

Dabei gelten folgende Punkte für die Module:

- Die Softwaremodule sind in JavaScript / ECMAScript realisiert. Als Untergrenze ist ES6 gesetzt, höhere Versionen werden aber nach Bedarf verwendet, um auf einem aktuellen Standard der Technik zu bleiben sowie die Entwicklung durch Nutzung neuer Features wirtschaftlicher zu machen.

- Einzelne Module kommunizieren über definierte APIs und sind unabhängig verifizierbar.

- Für alle Module gibt es jeweils Komponenten-, Integrations- und Systemtests.

- Alle Funktionen gemäß 4 Produktfunktionen haben jeweils Positiv-, Negativ- und Regressionstests.

- Weiterhin wird statische Code-Analyse genutzt, um alle Module auf potentiell fehlerhaften oder unsicheren Code frühzeitig zu testen und zu melden.

- Als Testframework für die funktionalen JavaScript-Module wird [Jest] verwendet, da es dem Industriestandard entspricht und durch die hohe Verbreitung eine zuverlässige Plattform bildet.

- Als Testframeworks für die GUI-JavaScript-Module werden [Jest] und [Puppeteer] verwendet, da sie dem Industriestandard entsprechen und durch die hohe Verbreitung zuverlässige Plattformen bilden.

- Die Tests sind automatisiert durchführbar und konfiguriert für ein automatisches Testen von neuem/geänderten Code durch Einbindung in das industrieführende Werkzeug [GitLab].

- Alle exportierten Module sind mit Dokumentationskommentaren versehen:

- (Haupt)-Autor
- Version
- Zweck
- Einsatzmöglichkeiten
- Grenzen
- Parameter (Eingabe und Rückgabe) mit Zweck, Typ, Struktur und gültigen Wertebereichen
- Vom Code geworfene Fehler/Exceptions

Generell werden zur Kommunikation zwischen den Modulen Datenpakete ausgetauscht, die sich in [JSON Notation] repräsentieren lassen.

3.2.2 VERSIONSKONTROLLE

Um Änderungen am Quellcode historisch nachvollziehbar zu halten, ist das Projekt unter Nutzung der weit verbreiteten und etablierten Versionskontrollsoftware [Git] entwickelt. Dabei wird zur initialen Abgabe eine Version erstellt, bei der die interne Entwicklungshistorie bereinigt wurde. Alle weiteren Änderungen im Zuge von Abstimmung oder Weiterentwicklung werden historisch festgehalten.

- Änderungen sind dokumentiert und kommentiert.
- Änderungen sind einem Verantwortlichen zuzuschreiben.
- Parallele Entwicklung mehrerer Programmierer bzw. Features ist möglich durch dezentrale Verwaltung des Quellcodes.
- Eine Hauptversion ist im sogenannten master-Branch entwickelt, in der semantische Versionsnummerierung verwendet wird, um eine nachvollziehbare Historie von Features und Fehlerbehebungen zu ermöglichen.

3.2.3 IT-GRUNDSCHUTZ

- Als Richtlinie wird dem IT-Grundschutz-Kompendium des BSI entsprochen. Dabei werden die folgenden Bausteine umgesetzt:
  - OPS.1.1.2 Ordnungsgemäße IT-Administration
  - OPS.1.1.3 Patch- und Änderungsmanagement
  - OPS.1.1.4 Schutz vor Schadprogrammen
  - OPS.1.1.5 Protokollierung
  - OPS.1.1.6 Software-Tests und -Freigaben
  - CON.5 Entwicklung und Einsatz von Allgemeinen Anwendungen
- Die Anwendung der Bausteine wird dokumentiert und eventuelle Abweichung sowie Anwendung von optionalen Empfehlungen werden begründet.

3.2.4 GUI-CLIENT

Der GUI-Client bildet die Hauptanlaufstelle für typische Nutzer und typische Arbeitsabläufe. Der GUI-Client ist implementiert als JavaScript-Applikation und läuft in einem Browserfenster.

3.2.4.1 SOFTWAREBIBLIOTHEKEN

- [React-MD] wird zur Erzeugung eines einheitlichen, [Material Design] folgenden oder angelehnten Designs genutzt.
- Die Datenhaltung wird durch Nutzung von [Redux] realisiert.
- Die Interaktion zwischen Daten und Interface wird durch Nutzung von [React-Redux] realisiert.
- [ReactJS] wird als Bibliothek für die grundlegende Struktur und Funktionalität verwendet.

3.2.4.2 APPLIKATION

- Der Client ist ohne Administratorrechte in gängigen Browsern ausführbar. Eine Installation, die über dass Kopieren und extrahieren des Clients hinausgeht, ist nicht erforderlich.
- Offline-Betrieb ist vollumfänglich möglich.
- Die Lokalisierung folgt den entsprechenden Regeln für Deutsch/Deutschland (de_DE). Andere Lokalisierungen sind nicht Teil des Produkts, jedoch sind alle Textelemente so gehalten, dass eine nachträgliche Lokalisierung in einer anderen Sprache problemlos möglich ist.

3.2.4.3 OBERFLÄCHE

- Die Oberfläche ist in dem den heutigen Standards entsprechenden [Material Design] realisiert.
- Die Startseite der Oberfläche ist jederzeit durch ein Navigationselement erreichbar.
- Die in 3.2.1 Generelle Aspekte beschriebene Modularisierung ist durch Reiter visualisiert.
- Abläufe, die ohne Benutzerinteraktion erfolgen können und keine unwiederbringlichen Änderungen auslösen, sind automatisiert.
- Jeweils die letzte Eingabe ist rückgängig machbar.
- Jeder lang laufende Prozess kann unterbrochen/abgebrochen werden.
- Die Bedienung der Oberfläche ist selbsterklärend:
    - Grafische Hervorhebung der aktuellen Position.
    - Grafische Hervorhebung möglicher bzw. nötiger Aktionen.
    - Rückmeldung über den Erfolg oder Misserfolg jeder Eingabe.

    - Die Bearbeitungszeit für Funktionen wird basierend auf den eingegebenen Werten vor Nutzung der Funktion angezeigt.

- Die Bedienung ist möglich sowohl mit Maus, mit Tastatur sowie mit Touchscreen.
- Die Benutzeroberfläche ist nach Möglichkeit zwischen den verschiedenen Eingabemöglichkeiten einheitlich gehalten.
- Nach Möglichkeit sind verschiedene Eingabemöglichkeiten gleichzeitig nutzbar.
- Eingabefelder haben Tooltips, in denen Wertebereiche und erlaubte Formate aufgezeigt werden.
- Eingabefelder werden mit sinnvollen Standardwerten vorbesetzt.
- Eingabefelder zeigen falls sinnvoll während der Dateneingabe/-änderung die Originaldaten.
- Eingaben werden wie in 3.2.1 Generelle Aspekte beschrieben plausibilisiert.
- Fehlerhafte Eingaben beeinträchtigen nicht die Stabilität oder Funktion der Oberfläche.
- Der Fokus für Eingabefelder wird entsprechend des erwarteten Arbeitsablauf gesetzt. Dabei wird Rücksicht auf die vom Benutzer erwartete Fokusnavigation genommen, indem GUI-Elemente sinnvoll gruppiert und angeordnet sind.
- Eingabefelder und Informationen die aufgrund der aktuellen Auswahl, Einstellung oder des Arbeitsschritt nicht relevant sind, werden nach Möglichkeit ausgeblendet oder durch Graufärbung grafisch in den Hintergrund versetzt.
- Durch sinnvolle Strukturierung von Eingabeelementen und Hinterlegung von Metainformationen wird die Barrierefreiheit unterstützt.
- Schriftgröße, Helligkeit, Kontrastverstärkung und ähnliche Funktionen sind durch den Browser selber ermöglicht.

3.2.5 CLI-CLIENT

Der CLI-Client dient zum direkten, nicht-interaktiven Aufruf des Rechenkerns von der Kommandozeile aus. Er eignet sich somit insbesondere für automatisierte Ausführungen im Rahmen von Tests oder Massenberechnungen. Als Aufrufparameter werden ein Dateiname und ein Ausgabepfad übergeben:

1. der Name einer existierenden Datei, die Eingangsdaten im JSON-Format und dem erwarteten Schema enthält, und

2. der Pfad zu einem Ausgabeordner, in dem (wiederum im JSON-Format) die Berechnungsergebnisse inklusive Warn- und Fehlermeldungen geschrieben werden.

Eine Installation, die über das Kopieren des Clients hinausgeht, ist nicht erforderlich, sofern Node und git installiert sind. Da der CLI-Client im Wesentlichen ein Interface für den Rechenkern ist, werden nur die Unterschiede bzw. zusätzliche Details spezifiziert.

Der CLI-Client ist ohne Administratorrechte von der Kommandozeile ausführbar. Da der Einsatz ohne Browser erfolgt, muss auf dem System Node.js Version 12 installiert sein.

3.2.6 RECHENKERN

- JavaScript (Version 9 / ECMAScript 2018).

- Industriestandard [JSON Notation] wird als Austauschformat verwendet, um maximale Kompatibilität zu erlauben.

- Eingaben werden wie in Generelle Aspekte beschrieben plausibilisiert.

- Fehlerhafte Eingaben beeinträchtigen nicht die Stabilität oder Funktion des Rechenkerns.

- Der Rechenkern ist funktional und atomar realisiert, eine jeweilige Eingabe führt zu einer Ausgabe und/oder Fehlermeldungen. Der Rechenkern enthält keinen sogenannten „State“.

- Die Ausführungszeit in der spezifizierten Produktumgebung beträgt wenige Millisekunden.

- Der Rechenkern ist beliebig parallelisierbar ausführbar. So kann Rechenkraft für umfangreiche Berechnungen verschiedener Eingabeparameter best möglichst skaliert werden.

3.3 HARDWARE

Es wird ausschließlich dem Durchschnitt aktueller Arbeitsplatzcomputer entsprechende Hardware benötigt.

3.4 INFRASTRUKTUR

Es wird ausschließlich dem Durchschnitt aktueller Arbeitsplatzcomputer entsprechende Infrastruktur benötigt.

4 PRODUKTFUNKTIONEN

4.1 RECHENKERNFUNKTIONEN

Die fachliche Kernfunktion des produzierten Rechenkerns ist eine Implementierung der Richtlinien für die rechnerische Dimensionierung von Betondecken im Oberbau von Verkehrsflächen.

Konkret wird die Entwurfsfassung der [RDO Beton], i.e. RDO Beton 09, Ausgabe 2018 [Forschungsgesellschaft für Straßen- und Verkehrswesen 2018-05-07] umgesetzt.

4.1.1 LAUFZEITLIMIT

Der Rechenkern benötigt auf einem System mit Intel i5-7300U CPU in keinem Durchlauf mehr als 0,2 sec. Daher wird auf die Implementierung einer konfigurierbaren Maximallaufzeit verzichtet.

4.1.2 ZIELGRÖSSE DER DIMENSIONIERUNG

Die einzige Dimensionierungsvariable ist die Solldicke h_d, siehe Kapitel 3, [RDO Beton]. Da das in den [RDO Beton] spezifizierte Verfahren ein nichtlineares Optimierungsproblem darstellt, wird die Solldicke in einem iterativen Verfahren bestimmt. Die Solldicke wird als positive natürliche Zahl auf 1 Millimeter genau bestimmt. Als untere Grenze der Solldicke für den Oberbeton gibt Kapitel 3 der [RDO Beton] 50mm an (h_d >=50). Ob dies ein einzuhaltendes Minimum der gesamten Solldicke darstellt ist unklar. Derzeit wird 50mm als untere Grenze der gesamten Solldicke behandelt, andernfalls wird eine Fehlermeldung ausgegeben, siehe Kapitel 6 „Fehlerbehandlung". Eine obere Grenze für h_d ist in den RDO nicht spezifiziert. Allerdings liegen die typischerweise ermittelten Solldicken von ca. 300mm bereits in einem Bereich in dem die zugrundeliegenden Annahmen der Kirchhoff-Loveschen Plattentheorie (dünne Platte, schubstarr) nicht mehr unbedingt gelten.

4.1.3 INTERPRETATIONSSPIELRAUM

Obwohl als Richtlinie konzipiert enthalten die [RDO Beton] an einigen Stellen Interpretationsspielraum. Zur Identifikation dieser Stellen hat der UAN die Vorgehensweise durch eine bislang nicht mit den [RDO Beton] vertraute Person implementieren lassen. Dabei wurde nur die grobe Struktur vorgegeben, beispielsweise die Berechnung der Einzelmomente. Wie die Momente be-

rechnet werden wurde bewusst offen gelassen, und damit der Interpretation des Mitarbeiters überlassen. Dies hat dazu geführt, dass die beim UAN bestehenden Implementierungen semantische (unabhängig von verwendeter Soft- und Hardware) Unterschiede aufweisen, die jedoch zu identischen Ergebnissen führen können. Diese Unterschiede werden im Rahmen der noch zu komplettierenden Softwaretests herausgearbeitet und mit dem AG diskutiert.

4.1.4 DEFINITIONSLÜCKEN & UNKLARHEITEN

Im Zuge des intensiven Studiums der [RDO Beton] sowie der Implementierung ergaben sich zahlreiche Fragen die nur zum Teil mit dem AN geklärt werden konnten. Diese werden im Folgenden dokumentiert.

4.1.4.1 PLATTENBREITE

Generell erlauben die [RDO Beton] eine untere Plattenbreite von 1200mm. Der Nachweis „Ermüdung“ erfordert allerdings mindestens 2000mm. Falls es sich nicht um Richtungsverkehr handelt, ergeben sich weitere Unklarheiten. Siehe dazu die Ausführungen im Bericht. Als maximal zulässige Plattenbreite werden 4500mm verwendet.

4.1.4.2 DIMENSIONIERUNGSRELEVANTE BEANSPRUCHUNG

Es ist ausschließlich Gleichung 7-3 zu implementieren.

Die dimensionierungsrelevante Beanspruchung wird durch das aus den [RStO] bekannte Verfahren berechnet, wobei zwei Unterschiede besteht. 1. In den [RDO Beton] wird ein gegenüber den [RStO] modifizierter Lastkollektivquotient, q_Bb anstatt q_Bm verwendet. 2. Die Achszahlfaktoren f_a unterscheiden sich. Das Verfahren nach [RStO] ist in den [RDO Beton] daher teilweise enthalten. Zum Einen führt dies aus Sicht des AN zu einer teilweise widersprüchlichen Dopplung der Spezifikation, zum Anderen kann dennoch nicht auf die [RStO] verzichtet werden, da für die Berechnung essentielle Bestandteile fehlen.

Es werden beide Verfahren implementiert.

All dies führt dazu, dass nach den [RDO Beton], komplettiert um Angaben aus den [RStO], die Berechnung von B bzw. B_n stattfinden kann. Dazu sind die unter 5.2.3 angegebenen Parameter notwendig.

4.1.4.3 STRASSENKLASSEN

Es werden lediglich die Straßenklassen Bundesautobahn, Bundesstraße, Landesstraße, Kreisstraße und Gemeindestraße implementiert. Siehe dazu die Ausführungen im Bericht.

4.1.4.4 ANPASSUNGSFAKTOR MB

Die Zeile „Nachweis bei konstanter statischer Last >= 1Tag“ wurde nicht implementiert da dieser Nachweis ansonsten nirgends in den [RDO Beton] vorkommt.

4.1.4.5 REIFENFAKTOR

Die Tabelle 6-5 erscheint redundant. Die Information ist – um den jeweiligen Lastfall ergänzt – in den Tabellen 6-9, 6-10 und 6-11 enthalten. Der Zusatz „bei Anhängerachse“ wurde ignoriert.

4.1.5 FREIE RÄNDER

Die in Rand- oder Übergangsbereichen auftretenden freien Plattenränder sind entsprechend Abschnitt 4.7 „Querkraftübertragung“ durch nicht-ansetzen des Dübelfaktors (mbD = 1) berücksichtigt. Eine über die Funktionalität der [RDO Beton] hinausgehende Logik ist nicht vorhanden.

4.1.6 LÄNGSFUGEN - LÄNGSMARKIERUNG

Die [RDO Beton] fordern unter 4.6 Geometrie, Fugenanordnung und Ausbildung: “Dabei wird vorausgesetzt, dass die Längsfugen des maßgebenden Plattenstreifens außerhalb der Längsmarkierung angeordnet werden.” Solche Voraussetzungen finden sich an etlichen Stellen der [RDO Beton], beispielsweise unter 1. Allgemeines sowie unter 4.5.1 Untergrund/Unterbau. Die Überprüfung dieser Voraussetzungen geschieht außerhalb des Verfahrens der [RDO Beton].

Da allerdings in der Leistungsbeschreibung gefordert wurde: “Visualisierung von schematischen Querschnittsbildern zur Vermeidung von konstruktiven Planungsfehlern (z.B. aufgrund der Lage der Fugen in Relation zu der erforderlichen Markierung)”, stellte sich die Frage was „außerhalb“ bedeutet. Weiterhin wurde der Abgleich mit den RMS-A notwendig. Nach Rücksprache mit dem AK 4.5.7 ergab sich die Klarstellung, dass „außerhalb“ lediglich „neben“ bedeutet und nicht die in 1 gezeigte Anordnung von Fugen und Markierungen des Hauptfahrstreifens.

4.2 CLI-FUNKTIONEN

Die Kommandozeilen-Schnittstelle sieht lediglich den Aufruf des Rechenkerns mit der Übergabe einer JSON-Eingabedatei und dem Pfad der Ausgabedateien vor. Beispielsweise:

```
node src/cli/startRDO.js cli-input/rdo-input.json cli-output/
```

Es gibt aber auch zusätzlich ein Start-Skript, welches den Standard-Input und Standard-Output verwendet und auf oberster Ebene im Repository liegt:

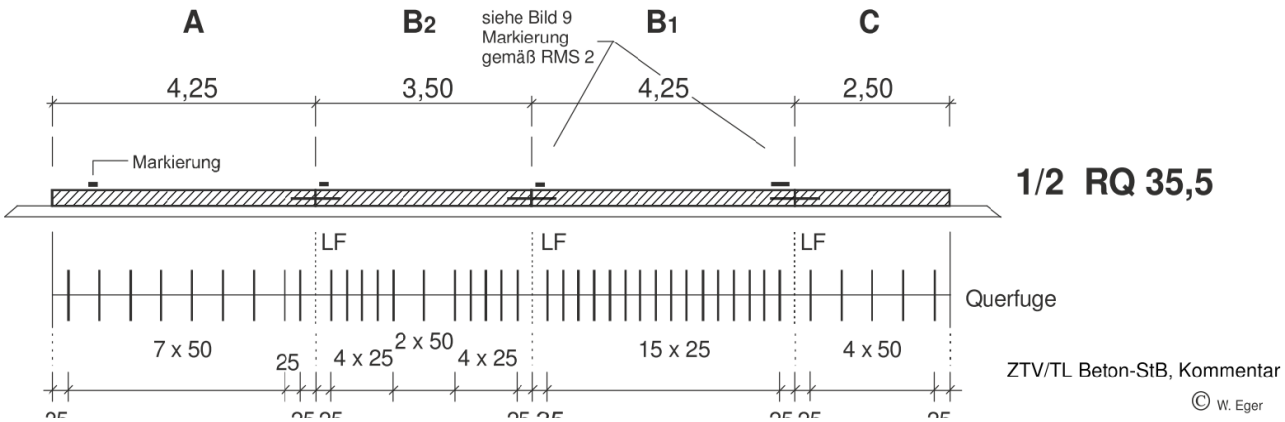
`run-rdo-cli.cmd` oder `run-rdo-cli.sh` auf Unix Systemen

Nach Beendigung des Rechenkerns wird eine Statusmeldung ausgegeben, die aussagt, ob die Solldicke ermittelt werden konnte, oder eine Fehlermeldung enthält. Ebenso werden die Ausgabedateien geschrieben, die die unter 5.3 Ausgabedaten beschriebenen Daten enthält.

4.3 GUI-FUNKTIONEN

Neben der Erfassung der Eingabedaten für den Rechenkern und Ergebnisausgabe und den in 4.4 Daten-Import und -Export beschriebenen Datenaustauschmöglichkeiten realisiert die grafische Benutzeroberfläche folgende Funktionen:

- Schematische Querschnittsbilder zur Vermeidung von konstruktiven Planungsfehlern (z.B. aufgrund der Lage der Fugen in Relation zu der erforderlichen Markierung) werden dargestellt.Siehe dazu die Ausführungen im Bericht.



1. Abbildung: Darstellung des halben Regelquerschnitts inkl. Lager der Fugen und Längsmarkierungen . Quelle: Präsentation Prof. Walter Eger, 2014.

4.4 DATEN-IMPORT UND -EXPORT

Alle eingegebenen Daten können mit einer Exportfunktion als Text in die Zwischenablage kopiert und von dieser wieder übernommen werden.

Konzeptionell und technisch ist ein Grundgerüst für eine weitergehende Datenverwaltung angelegt. Eine Implementierung könnte beispielsweise in Form einer dateibasierten Verwaltung oder durch Nutzung eines Servers mit Datenbankanbindung erfolgen. Eingabekonfigurationen, die für mehrfachen Gebrauch sinnvoll sind, könnten so gespeichert, geladen und gelöscht werden. Die tatsächliche Implementierung einer solchen Verwaltung ist nicht Teil des Produkts.

Es existiert eine Druckfunktion, in der alle Eingabedaten, Zwischenergebnisse, Endergebnisse sowie Symbolbild (Querschnitt) kompakt (DIN-A4) dargestellt werden. Nur relevante/verwendete Werte werden ausgegeben.

5 PRODUKTDATEN

Als „Daten“ im Kontext dieses Artikels werden alle für den Programmablauf bzw. für die Berechnung benutzten oder potentiell benötigten Werte betrachtet. Diese werden unterteilt in drei logische Gruppen (Grunddaten, Eingabedaten sowie Ausgabedaten) sowie einer gesonderten Betrachtung im Sinne des DSGVO.

5.1 GRUNDDATEN

Als Grunddaten sind Daten zu verstehen, die im Rechenkern fest verankert sind und nicht dafür vorgesehen sind, von Benutzern geändert zu werden.

Die [RDO Beton] weisen zahlreiche Tabellen auf, auf Basis derer Werte in Abhängigkeit von den unter 5.2 Eingabedaten spezifizierten Werten ebensolche Grunddaten zu wählen sind. Diese Werte werden hier nicht gesondert aufgeführt sondern sind nach Ansicht des AN vollständig durch die [RDO Beton] spezifiziert.

5.2 EINGABEDATEN

Eingabedaten können durch eine Eingabedatei spezifiziert werden. Diese ist in der grafischen Oberfläche sowie direkt von Menschen (z.B. mit einem Texteditor) veränder- und speicherbar.

Für die Eingabedaten wird JavaScript Object Notation (JSON) verwendet. JSON ist ein kompaktes Datenformat in einer einfach lesbaren Textform. Jedes gültige JSON-Dokument ist gültiges JavaScript. Das Format wird durch den Standard [JSON Notation] spezifiziert. Die Datei ist strukturiert und organisiert, so dass Parameter in logischen Einheiten gruppiert und für den Benutzer übersichtlich und verständlich dargestellt werden.

Als logische Einheiten werden auf oberster Ebene **Projekt** und **Parameter** verwendet. Parameter hat die Untergruppen **Nachweis**, **Konstruktion**, **Verkehr.** Zusätzlich hat **Parameter** ein einzelnes Feld **Beschreibung**, um zwischen mehreren Parametersätze für das selbe Projekt zu differenzieren. **Verkehr** sieht derzeit eine Option für die direkte Angabe der B-Zahl und von B_n oder die Berechnung nach den [RDO Beton] bzw. [RStO] vor. Parameternamen (keys) und zugewiesene Werte (values) werden wie in 5.1 Grunddaten dargestellt gut leserlich getrennt.

```
"Projekt": {
    "Name":         "Dimensionierung mit den RDO Beton 2018",
    "Beschreibung": "JavaScript (RDO Ausgabe 2018)",
    "Kommentar":    "",
    "Benutzer":     "B. Nutzer",
    "ISO_Datum":    "2020-05-21"
},
```

2. Abbildung: JSON-Snippet der Kategorie „Projekt“

Die Variablennamen entsprechen der Nomenklatur der [RDO Beton], insbesondere werden in der Eingabedatei die ausgeschriebenen deutschen Begriffe verwendet. Der JS-Quellcode ist durchgehend Englisch und übersetzt die Begriffe zu Anfang der Berechnung in die besonders für Formeln praktische von der RDO verwendete Kurzform (z.B. L_p) oder falls nicht spezifiziert in einen entsprechenden englischen Begriff (z.B. BaseLayer):

```
L_p:                inputOriginal.Parameter.Konstruktion.Plat-
tenlaenge,
```

```
BaseLayer:    inputOriginal.Parameter.Konstruktion.Unterlage,
```

Einige Parameter sind nach den [RDO Beton] nicht wählbar und werden daher im Quellcode implementiert.

Wenn nicht explizit angegeben, bezieht sich das Feld „Referenz“ immer auf die [RDO Beton].

Undefinierte Wertebereiche sind mit einem „?“ gekennzeichnet.

1. Tabelle: Eingabedaten der Kategorie „Projekt“

JSON-Bz	Code-Bz	Typ	Einheit	Referenz	Wertebereich
Name		str	-		
Beschreibung		str	-		
Kommentar		str	-		
Benutzer		str	-		

5.2.1 NACHWEIS

2. Tabelle: Eingabedaten der Kategorie „Nachweis“

JSON-Bz	Code-Bz	Typ	Einheit	Referenz	Wertebereich
NurDieserNachweis	OnlySpecifiedProof	bool	-	-	{`true`; `false`}
Belastung	Load	str	-	Tab. 6-6	{`"quasidynamisch"`; `"ermuedung"`}
Grenzzustand	LimitState	str	-	Tab. 6-6	{`"tragfaehigkeit"`; `"gebrauchs-tauglichkeit"`}
Fuge	Joint	str	-	Tab. 6-5	{`"laengs"`; `"quer"`}

5.2.2 KONSTRUKTION

3. Tabelle: Eingadeten der Kategorie „Konstruktion“

JSON-Bz	Code-Bz	Typ	Einheit	Referenz	Wertebereich
Strassenklasse	RoadCategory	str	-	Tab. 6-6	{"bundesautobahn"; "bundesstrasse"; "landes-strasse"; "kreisUndGemeindestrasse"}
Strassenbetonklasse	StC	str	-	Tab. 6-1 & 6-2	{"StC 25/30"; "StC 30/37"; "StC 35/45"; "StC 40/50"}
Spaltzugfestigkeit	f_ctk_sp	num	N/mm²	Tab. 4-2, 4-3, 6-1 & 6-2	{2.4; 2.7; 3.0; 3.3; 3.7; 4.0; 4.3; 4.6}
Verformungsmodul-Unterlage	E_V2	num	N/mm²	Tab. 5-1/2 & Abs. 6.2.9	45 - 150
Plattenbreite	B_p	num	mm	Abs. 4.6, Abs. 6.3.6, Tab. 7-2, Gl. 6-21	1200 – 4500 (Quasidynamisch), 2000 – 4500 (Ermüdung)
Plattenlaenge	L_p	num	mm	Abschn. 4.6, Gl. 6-20	1200 - 7500
Unterlage	BaseLayer	str	-	Tab. 6-3	{"asphaltSchicht"; "geotextilAufHGT"; "schotter-TragSchicht"; "kiesTragSchicht"; "sandTrag-Schicht"}
Querkraftuebertra-gung/ Querfuge	TransversalJoint	str	-	Tab. 6-4	{"mitDuebel"; "ohneDuebel"}
Querkraftuebertra-	LongitudinalJoint	str	-	Tab. 6-4	{"mitAnker"; "ohneAnker"; "verstaerkteAnker";

gung/ Laengsfuge					"laengskante"}
AnteilGebrochener-Gesteinskoernung	BrokenAggregateRatio	num		Tab. 6-12	{[0.0;0.2); [0.2;0.4); [0.4;0.6); [0.6; 1.0]}
Regelquerschnitt		str		RAA 2008 Kap. 4.3	{"keiner"; "RQ 43,5"; "RQ 36"; "RQ 31"; "RQ 28"; "RQ 38,5"; "RQ 31,5"; "RQ 25"}
AbrueckungLeitlinie-VonFuge		str		Abs. 4.6 und RMS A 2.2(5)	{"links"; "rechts"}
AbrueckungFahrbahn-begrenzungVonFuge		str		Abs. 4.6 und RMS A 2.2(5)	{"links"; "rechts"}

5.2.3 VERKEHR

4. Tabelle: Eingabedaten der Kategorie „Verkehr“

JSON-Bz	Code-Bz	Typ	Einheit	Referenz	Wertebereich
Geschwindigkeit	Velocity	num	km/h	Tab. 6-7	? z.B. 120
KonzentrationDes-Schwerverkehrs	HeavyTrafficConcen-tration	str/num	-	Tab. 6-14	{"normal"; 0.4 bis 1.0}
Berechnungsmethode	calculationMethod	str			{"RStO/RDO"; "Direkte Angabe"}

5. Tabelle: Direkete Angabe der B-Zahl

JSON-Bz	Code-Bz	Typ	Einheit	Referenz	Wertebereich
B	B	num	1/a	Tab. 6-7	? z.B. 105e+06
B_n	B_n	num		Abschn. 7.1	?

6. Tabelle: Parameter zur Berechnung nach RStO

JSON-Bz	Code-Bz	Typ	Einheit	Referenz	Wertebereich
Fahrstreifen	NumberOfLanes	num	-	RStO12 Tab. A 1.3	1-5 (Richtung: “proFahrtrichtung”), 2-inf (Richtung: “gesamt”)
DTV_SV_Erfassung	HeavyTrafficRecor-ding	str	-	RStO12 Tab. A 1.3	{"gesamt"; "pro Fahrtrichtung"}
Fahrstreifenbreite	LaneWidth	num	mm	RStO12 Tab. A 1.4	z.B. 3750
MaxLaengsNeigung	MaxLongitudinalGra-dient	num	%	RStO12 Tab. A 1.5	{[0;2); [2;4); [4;5); [5;6); [6;7); [7;8); [8;9); [9;10); [10; ?)}
Nutzungsdauer	ServiceLife	num	a	RStO12 S. 33	? z.B. 30
DTV_SV	DTV_SV	num	-	RStO12 Tab. A 2.1	? z.B. 10137

JSON-Bz	Code-Bz	Typ	Einheit	Referenz	Wertebereich
Achszahlfaktor	f_A	str/num		Seite 42, RStO12 Tab. A 1.1	{"entsprechend RstO"; 3.3 bis 4.5}

5.3 AUSGABEDATEN

Die Ausgabedaten werden wie die Input-Daten in einem JSON-Object gespeichert. Zu den Ausgabedaten gehören neben dem finalen Ergebnis auch Zwischenergebnisse, Änderungen am Quellcode, Fehlermeldungen und Warnungen. Vom Rechenkern werden diese Daten vollständig, pro Nachweis-Fall separiert, in einzelnen Dateien abgespeichert.

Die GUI stellt eine Untermenge dieser Daten dar. So wird lediglich die Solldicke h_d als Ergebnis der Dimensionierung ausgegeben. Das vollständige JSON-Object kann aber auch von der GUI als Dateien exportiert werden.

7. Tabelle: Exemplarische Ausgabedaten der Kategorie „Ergebnis“

Bz. in JSON	Bz. in Code	Typ	Einheit	Referenz	Wertebereich
Solldicke	h_d	num	mm	Abschnitt 3, etliche Gleichungen	>=50 für den Oberbeton. Wird als allgemeine untere Grenze angesetzt.Eigentlich ?

5.4 PERSONENBEZOGENE DATEN

Es ist sinnvoll für ein Projekt (eine Berechnung) einen Benutzernamen angeben zu können. Weiterhin werden sowohl ein projektbezogenes Kommentarfeld als auch ein parameterbezogenes Beschreibungsfeld vorgesehen, siehe 5.2 Eingabedaten. Die Angabe ist optional, alle Angaben werden allerdings in der JSON-Datei gespeichert. Daher ist bei einer Weitergabe der Datei(en) vom Anwender selbst auf die Einhaltung entsprechender Rechtsgrundlagen, wie der DSGVO, zu achten. Auf diesen Umstand wird im Benutzerhandbuch hingewiesen.

Weitere personenbezogene Daten werden nicht verarbeitet oder gespeichert.

6 FEHLERBEHANDLUNG

Fehler werden vom 3.2.6 Rechenkern abgefangen und sowohl vom 3.2.4 GUI-Client als auch vom 3.2.5 CLI-Client an den Nutzer zurück gegeben. So werden Aufrufe zwischen den Programmteilen überprüft. Ein generelles Schema für den Umgang mit Fehlern gilt für alle beschriebenen Prüfungen.

6.1 FEHLERPRÜFUNGEN

Fehlerprüfungen folgen einem einheitlichen Schema:

- Klare Definition von Wertebereichen
- Überprüfung von Wertebereichen
- Prüfung und Behandlung von widersprüchlichen und unsinnigen Parameterkombinationen
- Analoge Behandlung für Zwischenergebnisse
- Validierung von Definitionslücken und Genauigkeitsbereiche von zugrundeliegenden Funktionen
- Grundsätzlich werden soweit möglich Exceptions als Fehlermechanismus genutzt um versehentliche weitere Berechnungen mit ungültigen Daten zu verhindern.

6.2 FEHLERBEHANDLUNG

Fehlerbehandlung wird ebenso einheitlich behandelt:

- Lokalisation des Fehlers
- Möglichst genaue Beschreibung

- Bei Fehlern aufgrund unplausiblen Eingangsdaten werden nach Möglichkeit Ansätze zur Fehlervermeidung/-korrekt ausgeben, die eine Korrektur unterstützen.

Bei unkritischen Fehlern wird die Erklärung nur auf Aufforderung des Benutzers ausgeben. Fehlerhafte Grunddaten werden an den Nutzer zurückgemeldet.

7 PRODUKTLEISTUNGEN

/L100/ - *Toleranz*: Aufgrund der Anforderungen an Korrektheit werden bei Eingaben keine Toleranzen erlaubt. Mehrdeutige Eingaben oder fehlende Angaben führen zu Fehlermeldungen.

/L200/ - *Genauigkeit*: Die Solldicke wird auf den mm genau ausgegeben. Etwaige Probleme mit Auslöschung sind dem UAN bislang nicht bekannt und bei den vorkommenden Rechenschritten auch nicht zu erwarten.

8 QUALITÄTSBEWERTUNGEN

Bei der Entwicklung des Produkts werden die verschiedenen Qualitätsaspekte wie folgt bewertet und priorisiert

	SEHR WICHTIG	WICHTIG	WENIGER WICHTIG	UNWICHTIG
Robustheit		X		
Zuverlässigkeit	X			
Korrektheit	X			
Benutzungsfreundlichkeit	X			
Effizienz			X	
Portierbarkeit			X	
Kompatibilität			X	

9 TESTSZENARIEN UND TESTFÄLLE

Zum Testen der Berechnungsroutinen nach Code-Änderungen werden verschiedene Regressionstest implementiert, aus denen abzulesen ist, ob und wenn ja welche Zwischenergebnisse durch diese Änderungen betroffen sind. Dass alle Änderungen erfasst werden, wird nicht gewährleistet, aber eine hohe Code-Abdeckung ist zu erreichen. Außerdem wird das Dimensionierungsbeispiel der [RDO Beton] im Anhang D mit fester Solldicke betrachtet.

10 ENTWICKLUNGSUMGEBUNG

Intern sind die DTV-Verkehrsconsult GmbH und ISAC GmbH frei in der Auswahl der Entwicklungswerkzeuge. Alle im Endprodukt genutzten Produkte, Bibliotheken und Technologien sind Open-Source-Varianten und/oder kostenlos nutzbar durch Dritte und stehen nicht im Widerspruch zur gewählten Lizenz der zu entwickelnden Software.

10.1 SOFTWARE

Eine nicht abschließende Auflistung der genutzten Softwarelösungen findet sich in 13 Referenzen. Zusätzlich werden im Programmcode Softwarebibliotheken genutzt, die den im obigen Absatz ausgeführten Aspekten entsprechen.

11 ERGÄNZUNGEN

Im Folgenden werden potentielle zukünftige Erweiterungen kurz skizziert. Diese sind nicht Bestandteil des bestehenden Auftrags.

11.1 RECHENKERN 2

Als Erweiterung sind quasi beliebige Varianten des Rechenkerns, wie beispielsweise eine Finite-Elemente-Methode möglich. Dazu würden sich dann andere Sprachen, wie beispielsweise C(++) oder Fortran anbieten. Falls sinnvoll kann auch auf GPUs gerechnet werden.

11.2 BROWSER/SERVER-VARIANTE

Eine servergestützte oder -basierte Variante hätte Potential für Funktionen wie Benutzerverwaltung, Einstellungsvorlagen, asynchrone Berechnungen, Bearbeitung durch mehrere Nutzer, Berechnungshistorie sowie arbeitsplatzunabhängiges Laden/Speichern.

11.3 AUTOMATISIERTE MATRIX-BERECHNUNGEN

Unter Angabe von bestimmten Rahmenparametern wäre es möglich, tausende Berechnungsabläufe zu simulieren, und die Ergebnisse aggregiert auszugeben.

12 GLOSSAR

AWDSTAKO - Durch die Villaret Ingenieurgesellschaft mbH vertriebenes Programm zur Bemessung der Dicke von Betondecken für Verkehrsflächen.

Ausgabedaten - Die Ergebnisdaten der Berechnungen des Produkts sind die Ausgabedaten.

Eingabedaten - Eingabedaten sind Daten, die für eine jede Berechnung variabel sind und deshalb zum errechnen eines konkreten Ergebnisses nötig sind und vom Endnutzer eingegeben werden.

Grunddaten - Als Grunddaten verstehen sich Daten, die im Programm fest verankert sind und bei denen nicht abzusehen ist, dass Änderungen nötig sind. Beispiele: Gravitationskonstante.

13 REFERENZEN

Softwarebibliotheken

Git - Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.
https://git-scm.com/

GitLab - GitLab is a web-based DevOps lifecycle tool that provides a Git-repository manager providing wiki, issue-tracking and continuous integration/continuous deployment pipeline features, using an open-source license.
https://gitlab.com/

Jest - Jest is a JavaScript Testing Framework maintained by Facebook, Inc. with a focus on simplicity.
https://jestjs.io/

Puppeteer - Puppeteer is a Node library which provides a high-level API to control Chrome or Chromium over the DevTools Protocol.
https://pptr.dev/

React-MD - This project's goal is to be able to create a fully accessible material design styled website using React Components.
https://react-md.mlaursen.com/

React-Redux - Official React bindings for Redux.
https://react-redux.js.org/

ReactJS - React is a JavaScript library for building user interfaces, maintained by Facebook and a community of individual developers and companies.
https://reactjs.org

Redux - Redux is an open-source JavaScript library for managing application state.
https://redux.js.org/

Regelwerke

JSON Notation: Douglas Crockford (2013) - JavaScript Object Notation ECMA-404.
https://www.json.org/json-en.html

Material Design: Google (2019) - Material design is a comprehensive guide for visual, motion, and interaction design across platforms.
http://material.io

RDO Beton: FGSV (2009) - Richtlinien für die rechnerische Dimensionierung von Betondecken im Oberbau von Verkehrsflächen.
https://www.fgsv-verlag.de/rdo-beton

RStO: FGSV (2012) - Die "Richtlinien für die Standardisierung des Oberbaus von Verkehrsflächen" regeln die Standardfälle bei Neubau und Erneuerung für den standardisierten Oberbau von Verkehrsflächen innerhalb und außerhalb geschlossener Ortslagen.
https://www.fgsv-verlag.de/rsto