Konzipierung und Erstellung eines Parametertabellenkonfigurators und Praxisnahe Mitarbeit im releasten Produkt LC-VISION

Projektarbeit T1000
des Studiengangs Informatik
an der Dualen Hochschule Baden-Württemberg Ravensburg

von

Janik Frick

Kurs: TIT21, Matrikelnummer 4268671
Dualer Partner: Blum-Novotest GmbH, Standort Grünkraut
Betreuer: Mallmann, Guilherme, Dr.-Ing.

Inhaltsverzeichnis

1 Einleitung

Die Blum-Novotest GmbH(Blum) ist im Bereich Maschinenbau für Mess- und Prüftechnik tätig. Am Standort Grünkraut (Baden-Württemberg) ist der Hauptsitz der Firma. Dort befinden sich Entwicklung und Fertigung für den Bereich Messtechnik. Am zweiten Standort in Willich (Nordrhein-Westfalen) sind die Entwicklung und Fertigung im Bereich Prüftechnik untergebracht.
Produkte von Blum werden in vielen anspruchsvollen Industrien im Bereich der Qualitätssicherung eingesetzt. Die Kunden kommen unter anderem aus der Automobil-, Luftfahrt- und der Werkzeugmaschinenindustrie.
An die Produkte dieser Industrien werden höchste Ansprüche in Sachen Qualität gestellt. Daraus resultieren auch für die Produkte der Firma Blum höchste Ansprüche.
Um diese Ansprüche erfüllen zu können, werden sowohl bestehende Produkte laufend optimiert und weiterentwickelt, als auch neue Produkte entwickelt.
Die Arbeit wird im Umfeld von Blum am Standort Grünkraut verfasst.
Ziel der Arbeit ist es die vielfältigen Aspekte der Softwareentwicklung kennenzulernen und den Einstieg zu schaffen.
Um dieses Ziel zu erreichen, besteht der praktische Teil aus zwei Projekten.
Mit der Entwicklung eines „Parametertabellenkonfigurators“ sollen die verschiedenen Aufgaben der Softwareentwicklung veranschaulicht werden.
Durch die Mitarbeit am schon bestehenden Produkt „LC-Vision“ wird die Produktpflege und Weiterentwicklung thematisiert. Außerdem wird dabei das Einarbeiten in unbekannten Code relevant, was ein wichtiger Bestandteil der Produktpflege ist.

2 Softwareentwicklung

Softwareentwicklung besteht nicht nur aus dem Schreiben von Software. Auch die Entwicklung von Software ist ein Projekt. Man beginnt mit dem Sammeln der Anforderungen, startet die Entwicklung von Prototypen und geht in die Implementierung. Der Prozess endet mit der Produktpflege und dem Service.

2.1 Anforderungsanalyse

Eindeutige und präzise Anforderungen bilden die Grundlage für ein funktionierendes Projekt[1].
Daraus folgt, dass es nicht ausreichend ist, die Liste der Anforderungen zu lesen. Anforderungen müssen analysiert und mit weiterem Wissen über das Projektumfeld kombiniert werden.
Dafür gibt es mehrere Techniken, die eingesetzt werden können.

2.1.1 Interviews

Interviews bieten die Möglichkeit in direkten Gesprächen mit Personen aus dem Umfeld des Projekts Informationen zu sammeln. Diese Informationen können Risiken sein, die zu berücksichtigen sind, Schwierigkeiten bei der aktuellen Vorgehensweise, sowie Abläufe die beibehalten werden sollten.
Der Erfolg dieser Vorgehensweise hängt dabei von allen Beteiligten Personen ab. Fehlt Wissen über das Umfeld des Projekts kann es zu Schwierigkeiten beim Sammeln der Fragen kommen, wodurch manche Aspekte nicht berücksichtigt werden. Auch die Befragten können Probleme haben ihre Gedanken und ihr Wissen wiederzugeben[2]. Das kann die Auswertung der Antworten erschweren.

2.1.2 Fragebögen

Fragebögen geben den Befragten Personen die Möglichkeit ohne direkte Beeinflussung durch den Ersteller der Fragen, Informationen bereitzustellen. Zusätzlich reduzieren sie den Zeitdruck unter dem geantwortet werden soll und alle Befragten antworten auf die gleiche Fragestellung.
Fragebögen können eingesetzt werden um Annahmen zu bestätigen oder nach Meinungen und Vorschlägen zu fragen[2].
Bei Einsatz dieser Methode ist darauf zu achten die Fragen offen zu formulieren, um die Antworten nicht in eine Richtung zu lenken.

2.1.3 Beobachtung

Die Beobachtungsmethode dient dazu aus beobachteten Vorgängen Informationen zu sammeln.
Diese Beobachtung kann offen oder verdeckt durchgeführt werden. Bei der verdeckten Vorgehensweise ist das beobachtete Verhalten natürlicher und realistischer, als bei der offenen Vorgehensweise[3].
Mit diesem Vorgehen lassen sich Abläufe bei Aufgaben nachvollziehen. Gründe für dieses Abläufe sollten durch andere Techniken in Erfahrung gebracht werden.

2.1.4 Anforderungstypen

Anforderungen können in zwei Typen unterteilt werden: Die funktionalen und die nicht funktionalen Anforderungen.
Funktionale Anforderungen geben vor, für welche Aufgaben das Produkt geplant wird und welche Funktionen dafür benötigt werden. Die Validierung dieser Anforderungen ist gegeben, da am Ende eindeutig ist, ob eine Funktionalität verfügbar ist.
Die nicht funktionalen Anforderungen geben vor, mit welchem Vorgehen und welchen Tools eine Funktion umgesetzt werden soll[4].
Nicht funktionale Anforderungen sind häufig unpräzise formuliert und somit problematisch in der Validierung[4].

2.2 Dokumentation

Die Dokumentation von Softwareprojekten ist neben der eigentlichen Entwicklung ein relevanter Teilaspekt. Die Dokumentation dient zur Sicherung von Wissen und Grundlagen von Entscheidungen.
Die Dokumentation kann in zwei Bereiche unterteilt werden. Die Dokumentation der Software und die Dokumentation für den Anwender[5].
Software-Dokumentation
Diese Form der Dokumentation soll Wissen über die Software beinhalten. Dazu zählen Grundlagen für Entscheidungen der Softwarearchitektur und die Aufgaben von Funktionen, Methoden und Variablen.
Die Relevanz besteht darin Wissensverlust zu verhindern, der entsteht, wenn der Entwickler nicht zur Verfügung steht. Dieser Verlust beeinflusst die Einarbeitungsdauer eines neuen Mitarbeiters und somit auch die Produktpflege.
Diese Problematik der mangelnden Softwaredokumentation ist in agilen Umfeldern bekannt. Sich ändernde Anforderungen verursachen die Notwendigkeit die Software anzupassen, was Anpassungen der Dokumentation erfordert. Geänderte Anforderungen brin-

gen auch neue kurzfristige Aufgaben mit sich, denen eine höhere Priorität beigemessen wird, als der Aktualisierung der Dokumentation. Kommen viele Änderungen in kurzer Zeit stauen sich die anstehenden Änderungen auf und sorgen für einen hohen Zeitaufwand der Dokumentation. Dies resultiert oftmals in einer vernachlässigten Dokumentation.
Weitere potentielle Schwierigkeiten resultieren aus der Ansicht, dass der Code bereits eine ausreichende Dokumentation für diesen Zweck darstellt und dass Informationen häufig informell und verbal kommuniziert werden, was dazu führen kann, dass diese Informationen nach der Verarbeitung nirgends festgehalten sind und dadurch verloren gehen können[5].

User-Dokumentation

Der Dokumentation für die Anwender und die Betreuer wird meist mehr Bedeutung beigemessen, da sie für den Einsatz des Produktes relevant ist.
Diese Dokumentation beinhaltet Fehlermeldungen, Voraussetzungen für die Installation und Informationen für die Verwendung. Probleme mit dieser Dokumentation können Schwierigkeiten im Kundenservice verursachen. Da Probleme dieser Art einen direkten Einfluss auf den Erfolg eines Produktes oder der Firma haben können. Diese möglichen Konsequenzen sorgen dafür, dass dieser Dokumentation mehr Bedeutung beigemessen wird[5].

2.3 Grobentwurf

Beim Grobentwurf werden die Hauptbestandteile der Software und die Softwarearchitektur geplant. Dabei sollte der Kunde möglichst miteinbezogen werden, denn der Kunde hat großen Anteil am Gelingen des Projekts[6].
Um den Kunden einzubeziehen können Prototypen eingesetzt werden.
Hat das Programm eine Graphische Benutzeroberfläche(GUI) so kann der Grobentwurf unterschiedlich geplant werden.

2.3.1 GUI mit Bildern

Der Grobentwurf der GUI kann mit Hilfe von Bildern in einem beliebigen Bilderstellungsprogramm nachgebildet werden. Die erstellten Entwürfe dienen als Gesprächsgrundlage für eventuell notwendige Veränderungen. Hierbei ist zu beachten, dass die Simulation von Verbindungen zwischen GUI-Elementen zeitaufwendig oder gar nicht möglich ist.

2.3.2 Prototypen

Ein Prototyp ist eine funktionierende, begrenzte Version der Anwendung, die als Basis für Gespräche und die Entwicklung weiterer Prototypen dient[7].

Dieser Prototyp kann durch Software, oder mit einem Mockup-Tool[1], oder auch in einer Software für Präsentationen, realisiert werden. Ein Software basierter Prototyp erfordert die Mitarbeit der Entwickler im Design Prozess.

2.3.3 User-Tests

Bei User Tests wird einer oder mehreren Testpersonen, abhängig von der Verfügbarkeit, der aktuelle Prototyp 2.3.2 vorgelegt. Dann kann man der Testperson eine konkrete Aufgabe geben, oder sie den Prototyp frei entdecken lassen. Bei beiden Vorgehensweisen werden Probleme dokumentiert. Hat man mehrere Testpersonen zur Verfügung können auch Gruppentests durchgeführt werden. Bei diesen arbeiten mehrere Personen an einer Aufgabe. Bei Einzel-Tests lassen sich Probleme im Ablauf besser erkennen. In Gruppen-Tests hingegen werden die tatsächlichen Probleme der Testpersonen besser erkenntlich.[8].
Mit dieser Testmethodik lassen sich nur Prototypen 2.3.2 testen, da nur hier die Verbindungen funktional sind.

2.3.4 30-Sekunden Tests

Bei 30-Sekunden-Tests bekommen die Testpersonen 30 Sekunden Zeit um den Prototyp zu erkunden. Nach 30 Sekunden wird ein Fragebogen ausgefüllt, auf dem die Übersichtlichkeit bewertet wird[9].
30-Sekunden Tests lassen sich mit Bildentwürfen 2.3.1 und mit Prototypen 2.3.2 durchführen, da es bei beiden Entwürfen möglich ist ein Element für 30 Sekunden zu zeigen.

2.3.5 Heatmaps

Heatmaptools markieren farbig welche Bereiche einer Benutzeroberfläche oft genutzt werden und welche seltene[10]. Diese Daten geben Informationen darüber ob Kernfunktionen anders positioniert werden sollten.
Diese Methode lässt sich für Bildentwürfe 2.3.1 und Prototypen 2.3.2 verwenden.

2.4 Implementierung

In der Implementierung ist der Aufwand abhängig von der Vorgehensweise. Wird das GUI mit Hilfe von Bildentwürfen 2.3.1 entwickelt, ist der Aufwand der Implementierung höher, da die Applikation von Grund auf programmiert werden muss.
Wird mit Prototypen 2.3.2 gearbeitet, ist die Implementierung weniger aufwendig. Kommt ein Mockup-Tool zum Einsatz muss die Software ebenfalls von Grund auf programmiert werden, aber der Mock-Up Prototyp dient als Hilfestellung um die Übersicht über die

[1]Tool um ein GUI zu simulieren

notwendigen Verbindungen zwischen GUI Elementen zu behalten.
Ist der Prototyp programmiert werden, ist die Struktur fertig und muss gegebenenfalls um Funktionen ergänzt werden, die im Grobentwurf nicht berücksichtigt wurden.
Entscheidungen was Design und Technologien betrifft, sollten vor Beginn der Implementierung getroffen worden sein. Die Aufgabe der Entwickler besteht darin das geplante Design so in Module zu unterteilen, das es programmiert werden kann[11].

2.4.1 Tests

Durch Tests wird validiert, ob die Software die Anforderungen erfüllt. Dabei sollten alle Ergebnisse von Tests dokumentiert werden, um Fehlerquellen, Lösungen und die Kosten für das Beheben der Fehler im Überblick zu behalten[11].
Diese bestehen aus mehreren Stufen.
Modultests
Programmierte Module werden isoliert getestet, um zu prüfen, ob das Modul die Aufgaben erfüllt. Dazu kann auch ein simulierter Input verwenden, wenn benötigte Module noch nicht verfügbar sind.
Diese Tests können in zwei Arten unterteilt werden. Bei 'white-box' Tests wird das Modul in einzelnen Schritten getestet. Der Tester weiß wie die Daten verarbeitet werden. Bei 'black-box' Tests versucht der Tester das Vorgehen eines Nutzers nachzustellen. Dabei ist dem Tester die genaue Verarbeitung egal[11].
Low-Level Integrationstests
In diesem Stadium werden die Aufrufe und Rückgabewerte von anderen Modulen getestet. Dabei spielt es keine Rolle, von wem die abhängigen Module entwickelt wurden. Stehen keine realen Daten zur Verfügung kann mit simulierten Daten gearbeitet werden.[11].
Ob diese Tests manuell oder automatisiert stattfinden hängt von unterschiedlichen Faktoren ab.

Mit diesem Vorgehen sind alle Module und ihre Abhängigkeiten auf ihre Funktionalität überprüft. Auf dieser Basis wird eine Version erstellt und getestet die in dieser Form praxistauglich ist. Parallel wird daran gearbeitet die benötigten Umgebungsbedingungen bereitzustellen. Dieses Setup wird auf Funktionalität geprüft. Bei erfolgreichem Verlauf der Tests kann die Dokumentation 2.2 fertiggestellt werden. Bestätigt der Kunde das Resultat kann mit dem Training betreuender Personen und Benutzern begonnen werden[11].

2.4.2 Test Automatisierung

Das manuelle Ausführen von Tests kann zeitaufwändig sein, wodurch man auf die Idee kommen kann diese Tests zu automatisieren. Diese Entscheidung ist dabei von unterschiedlichen Faktoren abhängig.
Die Erfahrung des Verantwortlichen beeinflusst die Dauer für die Automatisierung maßgeblich. Auch die Umgebung in der die Anwendung funktionieren soll, muss automatisiert simuliert werden. Basiert die Funktionalität der Software auf Benutzereingaben, oder läuft sie nach dem Start alleine? Anwendungen die selbstständig ablaufen, sind meistens einfacher zu testen, als solche bei denen man eine automatisierte Benutzerinteraktion nachstellen muss[12].

2.5 Objektorientierung

3 Parametertabellenkonfigurator

Mit Hilfe des „Parametertabellenkonfigurators“ sollen die Abläufe der Konfiguration unterstützt werden.

3.1 Problemstellung

Bei bestehenden Programmen für numerische Steuerungen müssen während der Inbetriebnahme Parametertabellen in einem beliebigen Texteditor angepasst werden. Aufgrund der Anzahl, wie auch der Einstellmöglichkeiten der Parameter, wird hierfür eine separate Installationsanleitung benötigt. Der Prozess der Parametrierung ist daher Fehleranfällig und von der Erfahrung des Inbetriebnehmers abhängig. Um den hohen Anforderungen der Kunden gerecht zu werden, gilt es potentielle Fehlerquellen zu eliminieren. Hierfür soll selbstständig ein „Parametertabellenkonfigurator“ entwickelt werden.

3.2 Anforderungen

An den „Parametertabellenkonfigurator“ gibt es mehrere Anforderungen.
Parametertabellen sollen eingelesen und angezeigt werden können. Zu den Parametern soll es ermöglicht werden Beschreibungen aus Text und Bild speichern und anzeigen zu können.
Diese Anforderungen sind aus dem Bereich der funktionalen Anforderungsen 2.1.4
Nicht funktionale Anforderungen 2.1.4 sind die Verwendung der Programmiersprache und das Verwenden des in C++ entwickelten Framework Qt.
Aus diesen Anforderungen lässt sich ableiten welche GUI-Elemente nötig sind und welche Funktionalitäten benötigt werden.

Um die Anforderungen an den „Parametertabellenkonfigurator“ zu sammeln, die nicht im Auftragsdokument stehen, wurde das Interview 2.1.1 gewählt. Die Entscheidung wurde getroffen weil in offenen Gesprächen mehr Zusatzinformationen fließen, als in Umfragen 2.1.2 oder durch Beobachtung 2.1.3. Zusätzlich kann auf Verständnisschwierigkeiten reagiert werden, solange das Gespräch noch aktuell ist und der Kontext noch nicht vergessen ist.
Hauptergebnis ist die Bedeutung eines übersichtlichen GUI.

3.2.1 Notwendige Funktionalitäten

Um die Parametertabellen anzeigen zu können müssen diese aus einer Datei eingelesen werden. Um Konfigurationen festzuhalten wird eine Funktion zum Speichern der Tabellen benötigt.
Für die Beschreibungen wird zusätzlich eine Funktion benötigt um diese mit den richtigen Parametern zu verknüpfen.

3.2.2 Notwendige GUI-Elemente

Das GUI benötigt Anzeigemöglichkeiten für die Parametertabellen und Beschreibungen, Angaben von Namen, Steuerung und Produkttyp zur aktuellen Parametertabelle und den aktuellen Parameter.
Benötigte Steuerelemente sind Möglichkeiten um Parametertabellen und Beschreibungen szu laden und zu speichern.

3.3 GUI-Entwurf

Um das GUI zu entwerfen kommt ein Prototyp 2.3.2 zum Einsatz, der mit dem Mockup-Tool ”Balsamiq Mockup 3érstellt wurde. Das Tool bietet nahezu alle gängigen GUI-Elemente an und kann diese miteinander verknüpfen, so dass schon in einem frühen Stadium die Funktionalität gegeben ist und besprochen werden kann. Zusätzlich bietet das Tool eine unkomplizierte Möglichkeit alternative Entwürfe zu entwerfen und in den Ablauf der Simulation zu integrieren. Das unterstützt den Prozess während der Entwicklung des GUI, da neue Vorschläge in kurzer Zeit besprochen und weiterentwickelt werden können.

Das GUI ist aus mehreren einzelnen Fenstern aufgebaut. Dieser Aufbau unterstützt die Übersichtlichkeit und der Benutzerführung bei der Verwendung, da jeweils nur die benötigten Informationen und Funktionen angezeigt werden.
Insgesamt besteht das GUI aus vier Fenstern die dem Benutzer die benötigten Funktionen zur Verfügung stellen.

Ein Fenster dient dem Zweck bei erster Benutzung den Pfad zu einer Datenbank, in der die Beschreibungen gespeichert sind, einzugeben. Im nächsten Fenster lässt sich die Auswahl der Parametertabellen konfigurieren. Nach der Auswahl einer Parametertabelle wird diese im Hauptfenster angezeigt. In diesem Fenster werden die Beschreibungen der Parameter angezeigt und die Parametertabelle kann angepasst werden. Es gibt Steuerelemente für die Navigation durch die Parametertabelle, für die Navigation durch die Fenster und die Möglichkeit die geänderte Parametertabelle zu speichern.

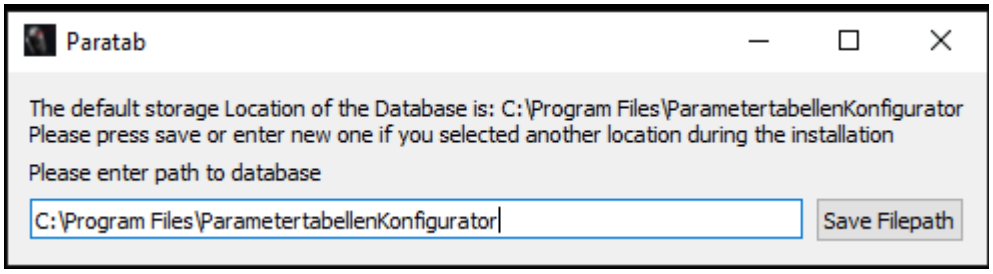

Abbildung 1: Fenster um den Dateipfad der Datenbank einzugeben, es gibt einen Standardpfad

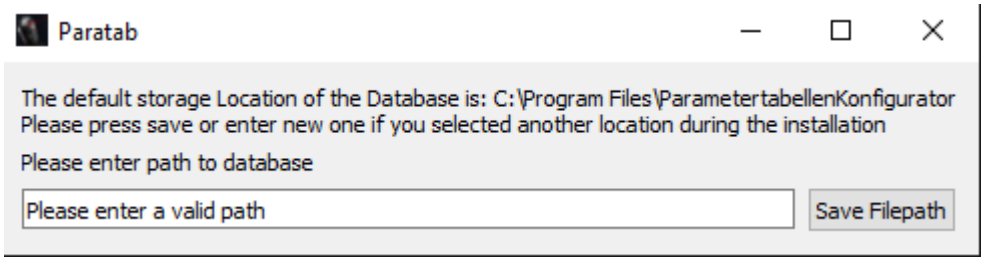

Abbildung 2: Wurde ein nicht vorhandener Pfad eingegeben erscheint die Fehlermeldung "Please enter a valid path"

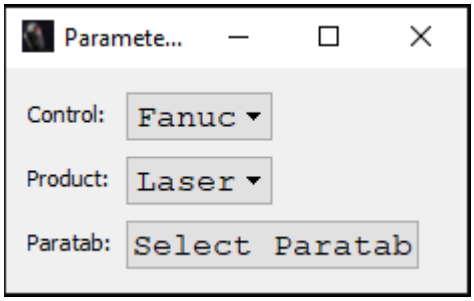

Abbildung 3: Fenster um Auswahl der Parametertabellen zu konfigurieren

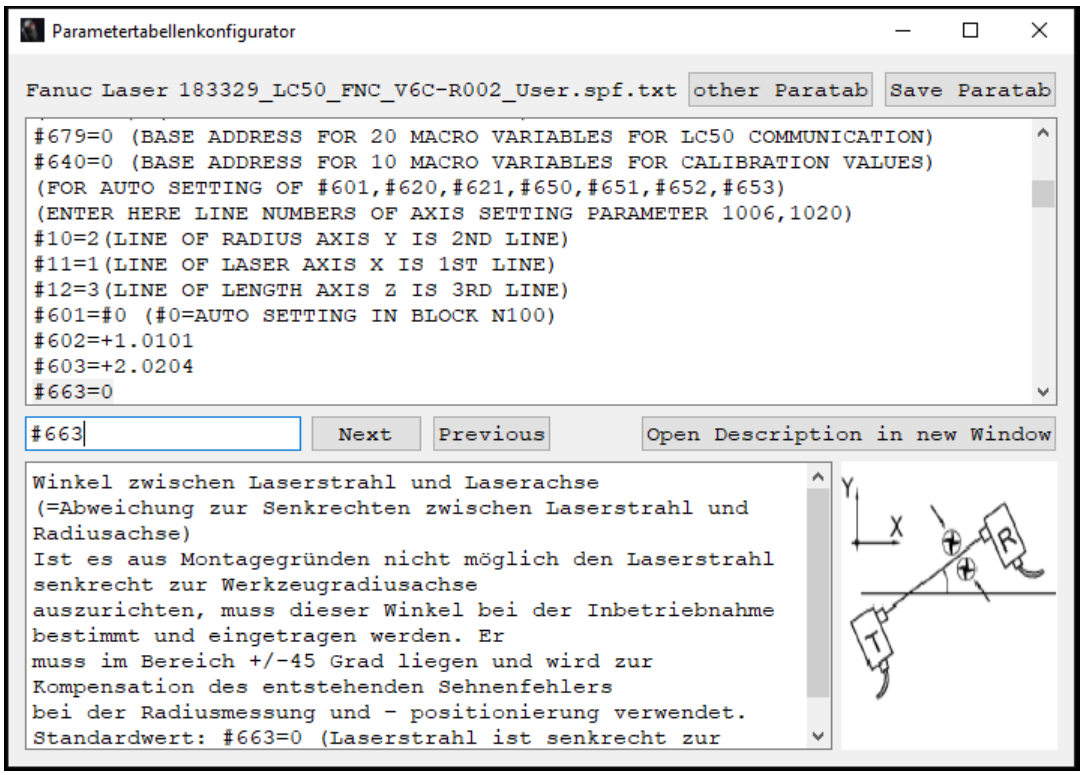

Abbildung 4: Fenster für die Anzeige der Parametertabellen und deren Beschreibungen

In Abbildung 4 werden im Oberen Bereich werden Informationen zur aktuell ausgewählten Parametertabelle angezeigt. Oben rechts gibt es Steuerelemente um eine andere Parametertabelle auszuwählen, oder die Aktuelle zu speichern.
Unterhalb der Anzeige für die Parametertabelle ist der Bereich für detaillierte Informationen zum aktuellen Parameter. Es gibt Steuerelemente um andere Parameter auszuwählen. Dies ist mit den Buttons "Nextünd "Previous"möglich, die jeweils um einen Parameter nach oben oder nach unten springen. Durch direkte Eingabe eines Parameters in das Parameterfeld kann zu einem beliebigen Parameter gesprungen werden. Um größere Beschreibungen übersichtlich sehen zu können gibt es die Möglichkeit die Beschreibung isoliert in einem neuen Fenster zu öffnen.

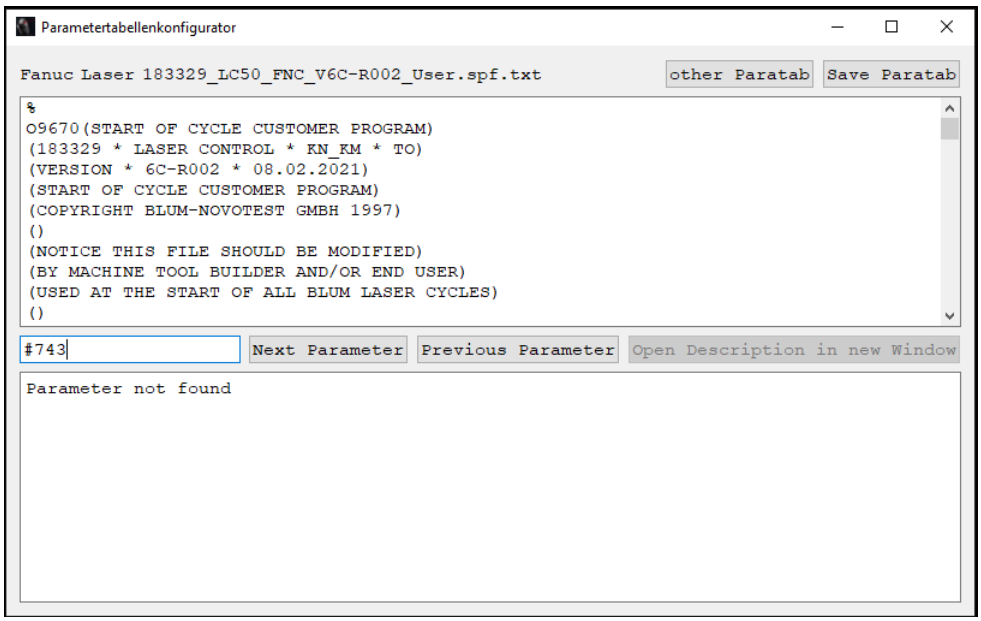

Abbildung 5: Wird ein nicht vorhandener Parameter in das Feld eingegeben wird eine Fehlermeldung angezeigt.

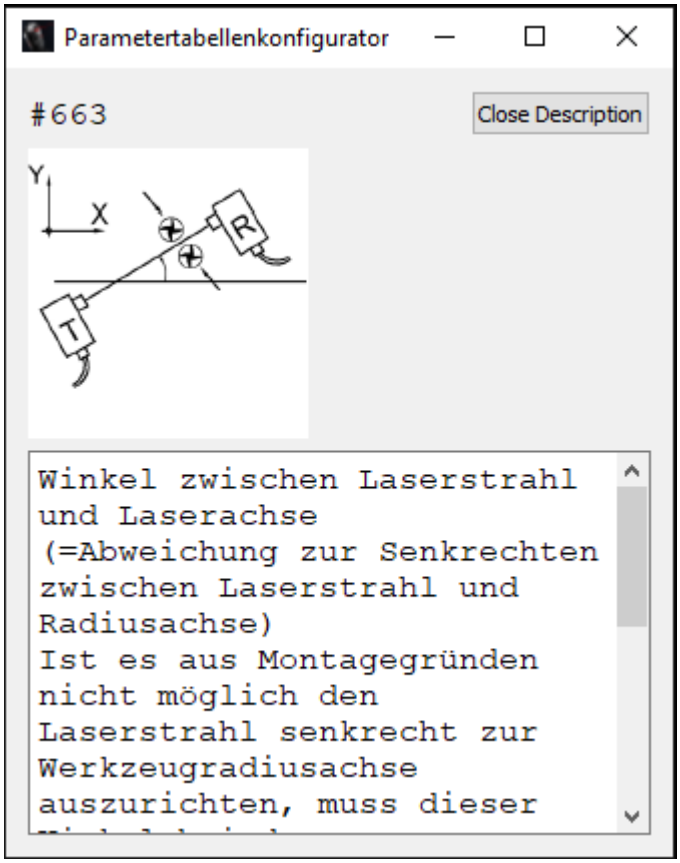

Abbildung 6: Isolierte Ansicht einer Beschreibung eines Parameters

Das in Abbildung 6 sichtbare Bild hat eine konstante Größe. Der Grund dafür ist dass ansonsten bei Änderung der Fenstergröße das Bild verzerrt werden würde und das Erkennen

von wichtigen Informationen im Bild erschwert werden würde.

3.4 Programmierung

Für die Programmierung des „Parametertabellenkonfigurator“ kommt das für GUI optimierte C++ Framework Qt zum Einsatz.

3.4.1 Softwarestruktur

Für die Programmierung kommt die objektorientierte Programmierung 2.5 zum Einsatz, da sich Elemente eines GUI gut als Objekt darstellen lassen. Die Software ist modular aufgebaut. Das bedeutet sie besteht aus einzelnen Modulen, die jeweils eine bestimmte Aufgabe übernehmen. Um die Module miteinander zu verknüpfen gibt es Klassen, die für die Aufrufe der entsprechenden Module verantwortlich sind.
Dieses Design bietet die Möglichkeit die Anzeige der Daten von der grundlegenden Logik getrennt zu verwalten.
Die Module haben vorgegebene Schnittstellen, die von den ‘Managerklassen’ bedient werden. Diese Schnittstellen rufen die internen Methoden des Moduls auf. Somit lässt sich der Inhalt und die Funktionsweise eines Moduls verändern ohne dass der Aufruf des Interface durch die ‘Managerklassen’ verändert werden muss. Dadurch entsteht eine hohe Flexibilität.

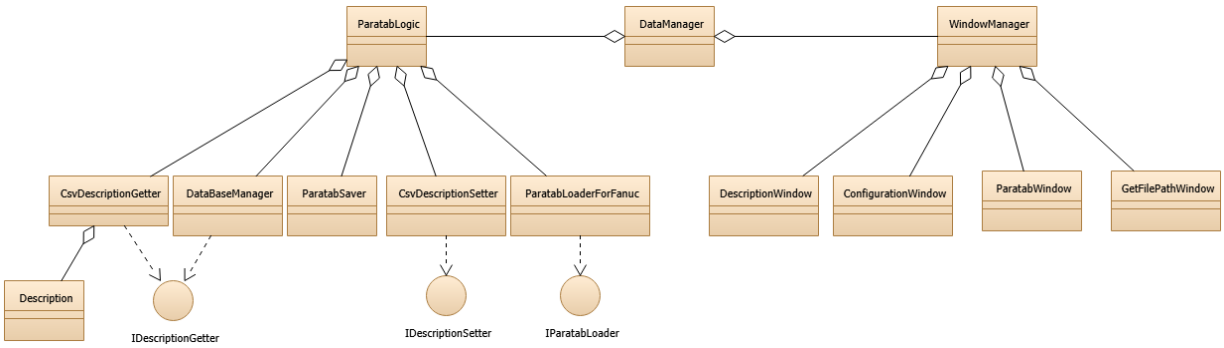

Abbildung 7: UML-Diagramm der Software ohne Inhalte der Klassen

Das in Abbildung 7 dargestellte Diagramm dient zur Veranschaulichung der Modularität der Software, aus Gründen der Übersichtlichkeit wurde daher auf den Inhalt der einzelnen Klassen verzichtet.

4 LC-Vision

Literatur

[1] S. Lane, P. O'Raghallaigh, and D. Sammon, "Requirements gathering: the journey," *Journal of Decision Systems*, vol. 25, no. sup1, pp. 302–312, 2016.

[2] S. Tiwari, S. S. Rathore, and A. Gupta, "Selecting requirement elicitation techniques for software projects," in *2012 CSI Sixth International Conference on Software Engineering (CONSEG)*, pp. 1–10, IEEE, 2012.

[3] R. Silhavy, P. Silhavy, and Z. Prokopova, "Requirements gathering methods in system engineering," *Recent Researches in Automatic Control*, pp. 105–110, 2011.

[4] J. Eckhardt, A. Vogelsang, and D. M. Fernández, "Are "non-functional"requirements really non-functional? an investigation of non-functional requirements in practice," in *Proceedings of the 38th International Conference on Software Engineering*, ICSE '16, (New York, NY, USA), p. 832–842, Association for Computing Machinery, 2016.

[5] T. Theunissen, U. van Heesch, and P. Avgeriou, "A mapping study on documentation in continuous software development," *Information and Software Technology*, vol. 142, p. 106733, 2022.

[6] S. Nerur, R. Mahapatra, and G. Mangalaraj, "Challenges of migrating to agile methodologies," *Commun. ACM*, vol. 48, p. 72–78, may 2005.

[7] R. Budde, K. Kautz, K. Kuhlenkamp, and H. Züllighoven, "What is prototyping?," *Information Technology & People*, 1992.

[8] J. C. Bastien, "Usability testing: a review of some methodological and technical aspects of the method," *International Journal of Medical Informatics*, vol. 79, no. 4, pp. e18–e23, 2010. Human Factors Engineering for Healthcare Applications Special Issue.

[9] P. Šimek, J. Vaněk, and P. Pavlík, "Usability of ux methods in agrarian sector-verification," *Agris on-line Papers in Economics and Informatics*, vol. 7, no. 665-2016-45086, pp. 49–56, 2015.

[10] J. Matejka, T. Grossman, and G. Fitzmaurice, "Patina: Dynamic heatmaps for visualizing application usage," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, (New York, NY, USA), p. 3227–3236, Association for Computing Machinery, 2013.

[11] J. J. Rakos, *Software Project Management*. Prentice Hall, 1990.

[12] M. Fewster and D. Graham, *Software test automation.* Addison-Wesley Reading, 1999.