Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

David Stelter

Entwicklung und Implementierung eines automatisierten Prüfverfahrens für CAD-Modelle und Zeichnungen

Fakultät Technik und Informatik
Department Maschinenbau und Produktion

Faculty of Engineering and Computer Science
Department of Mechanical Engineering and Production Management

David Stelter

Entwicklung und Implementierung eines automatisierten Prüfverfahrens für CAD-Modelle und Zeichnungen

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung

im Studiengang Entwicklung und Konstruktion
am Department Maschinenbau und Produktion
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

in Zusammenarbeit mit:
Rheinmetall Waffe Munition GmbH
Abteilung CITTD
Heinrich-Ehrhardt-Straße 2
29345 Unterlüß

Erstprüfer: Prof. Dr. Hans-Joachim Schelberg
Zweitprüfer: Dipl.-Ing. FH Rüdiger Küch

.
Abgabedatum: 27.08.2020

Zusammenfassung

David Stelter

Thema der Bachelorthesis

Entwicklung und Implementierung eines automatisierten Prüfverfahrens für CAD-Modelle und Zeichnungen

Stichworte

CAD, Workflow, Prüfung, Siemens NX, Siemens Teamcenter, Check-Mate, Microsoft Visual Studio, Visual Basic, Knowledge Fusion

Kurzzusammenfassung

In dieser Arbeit wird ein Verfahren beschrieben, um die Qualität vorliegender CAD-Daten in Siemens NX und Siemens Teamcenter abprüfen zu können. Dabei sollen die Prüfungen manuell vom Anwender oder automatisiert innerhalb eines Workflows durchgeführt werden können. Die Einzelprüfungen bestehen aus mitgelieferten und selbst erstellten Prüfungen. Die selbst erstellten Prüfungen werden in der Programmiersprache Visual Basic in der Entwicklungsumgebung von Microsoft Visual Studio geschrieben. Zur automatisierten Qualitätsprüfung werden die Prüfungen in einen Teamcenter-Workflow eingebunden.

Name of Student

Title of the paper

Development and implementation of an automated test procedure for CAD models and drawings

Keywords

CAD, Workflow, Check, Siemens NX, Siemens Teamcenter, Check-Mate, Microsoft Visual Studio, Visual Basic, Knowledge Fusion

Abstract

This thesis describes a procedure to check the quality of existing CAD data in Siemens NX and Siemens Teamcenter. The checks should be able to be carried out manually by the user or automatically within a workflow. The individual checks consist of supplied and self-created checks. The self-created checks are written in the programming language Visual Basic in the development environment of Microsoft Visual Studio. For automated quality inspection, the checks are integrated into a Teamcenter workflow.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich bei der Erstellung dieser Arbeit unterstützt haben und mich im Laufe der letzten Monate motiviert haben.

Bedanken möchte ich mich bei Prof. Schelberg für die Betreuung meines Hauptpraktikums und die darauf aufbauende Bachelorarbeit.

Ein besonderer Dank gilt Herrn Rüdiger Küch und der gesamten Abteilung CITTD, ohne die diese Arbeit nicht möglich gewesen wäre. Alle Mitarbeiter hatten stets ein offenes Ohr für meine Fragen und haben sich immer Zeit genommen, falls ich mal nicht weitergekommen bin.

Außerdem möchte ich mich bei meiner Familie und meinen Freunden bedanken, die während des gesamten Studiums hinter mir standen.

Abschließend möchte ich mich bei meiner Freundin bedanken, die, obwohl sie an ihrer eigenen Arbeit schreibt, noch Zeit gefunden hat, meine Texte Korrektur zu lesen und Verständnisfehler aufzudecken.

Inhaltsverzeichnis

Abkürzungsverzeichnis

AR	Augmented Reality
Brep	Boundary Representation
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CNC	Computerized Numerical Control
FEM	Finite Elemente Methode
KBE	Knowledge Based Engineering
KF	Knowledge Fusion
PLM	Product Lifecycle Management
RWM	Rheinmetall Waffe Munition GmbH
UI	User-Interface
VB	Visual Basic
VR	Virtual Reality

Abbildungsverzeichnis

Tabellenverzeichnis

Management Summary

Die Qualität der CAD-Daten soll in der Rheinmetall Waffe Munition GmbH signifikant erhöht werden. Außerdem sollen alle deutschen Standorte nach denselben Regeln arbeiten, um eine gleichbleibend hohe Datenqualität standortunabhängig sicherstellen zu können.

Um dies zu erreichen wurde eine standortübergreifende Konstruktionsrichtlinie eingeführt, die für die Entwickler an allen deutschen Standorten verbindlich ist. Die Einhaltung der Punkte der Konstruktionsrichtlinie soll automatisiert abgeprüft werden.

Diese Arbeit befasst sich mit der Entwicklung und Implementierung eines automatisierten Prüfverfahrens für CAD-Modelle und Zeichnungen in Siemens NX und deren Einbindung in den Freigabeprozess im Zusammenspiel mit Siemens Teamcenter. Die Prüfungen erfolgen im NX-eigenen Prüftool CheckMate. Da nicht alle Punkte der Richtlinie durch mitgelieferte Prüfungen abgedeckt werden, müssen eigene Prüfungen geschrieben werden. Diese werden in Verbindung mit NXOpen in Visual Studio geschrieben.

Die Durchführung der Prüfungen kann manuell vom Anwender in NX oder automatisiert im Laufe eines Workflows erfolgen. Nach der Prüfung wird ein Prüfbericht erstellt und in Teamcenter gespeichert. Der Bericht wird danach im Workflow automatisch ausgewertet. Ist die Prüfung bestanden läuft der Workflow weiter. Ist die Prüfung nicht bestanden, bricht der Workflow ab und muss nach Anpassung des Modells oder der Zeichnung von neuem gestartet werden.

Durch die Notwendigkeit eines bestandenen Prüfberichtes bei einer Freigabe steigt die Qualität der neu freigegebenen CAD-Daten an. Dies verringert die Zeit bei der Fehlersuche sowie Korrektur von CAD-Modellen und Zeichnungen. Weiterhin wird die Einarbeitung von neuen Mitarbeitern in Projekte deutlich erleichtert. Um dem Anwender eine schnelle und einfache Möglichkeit der manuellen Durchführung von Prüfungen zu bieten, wird zusätzlich eine eigene Toolbar für die Prüfungen erstellt.

Zukünftig sollen die Einzelprüfungen in einem Visual-Studio Projekt zusammengefasst werden, um Speicherplatz zu sparen. Außerdem bietet sich die Umstrukturierung auf die Programmiersprache C# an, da diese Programmiersprache bei Rheinmetall für die eigenentwickelten Werkzeuge im Siemens NX-Umfeld genutzt wird. Des Weiteren können Anpassungen im User-Interface vorgenommen werden, um dem Anwender die Möglichkeit zu bieten, eigene Prüfprofile zu erstellen.

1. Einleitung

1.1. Motivation

Das Konstruieren von Modellen mittels Computer-Aided-Design (CAD) ist heutzutage ein essenzieller Bestandteil des Entwicklungsprozesses von Produkten. CAD-Modelle sind die Grundlage für viele nachfolgende Prozesse in der Entwicklung. Fehler in einer frühen Entwicklungsphase können enorme Kosten hervorrufen. Deshalb ist eine hohe Qualität der Daten wichtig, um dem Unternehmen Zeit und Geld zu sparen. Bei großen Datenmengen ist die Wahrscheinlichkeit hoch, dass Fehler übersehen werden. Diese können zum Teil durch mitgelieferte Prüfungen der CAD-Systeme abgeprüft werden. Aber nicht alle speziellen Regeln und Richtlinien eines Unternehmens können durch mitgelieferte Prüfungen abgedeckt werden.

1.2. Zielsetzung

Diese Arbeit befasst sich mit der Entwicklung und Implementierung eines automatisierten Prüfverfahrens für CAD-Modelle und Zeichnungen für die Rheinmetall Waffe Munition GmbH. Die Hauptpunkte der Arbeit sind die Analyse der neuen Konstruktionsrichtlinie des Unternehmens, die Programmierung der Prüfungen und deren Einbindung in den Freigabeprozess. Außerdem soll für die Benutzer ein ansprechendes User-Interface (UI) in Form einer Toolbar geschaffen werden. Mit Hilfe der Toolbar können die Prüfungen schnell, einfach und unabhängig vom Freigabeprozess gestartet werden.

1.3. Gliederung

Nach der Einleitung in Kapitel 1 folgen in Kapitel 2 die Grundlagen. Hier werden der Stand der Technik, die Begrifflichkeiten sowie die genutzten Programme erklärt. Außerdem werden Fehler und deren Ursachen in CAD-Modellen und Zeichnungen sowie Verfahren und Lösungen zur automatisierten Prüfung beschrieben.
In Kapitel 3 folgt die Analyse der Konstruktionsrichtlinie. Dazu wird der Stand der Qualitätssicherung im Unternehmen erläutert. Des Weiteren wird ein Auszug aus der Konstruktionsrichtlinie dargestellt und die einzelnen Unterpunkte werden auf ihre Prüfbarkeit mittels Check-Mate analysiert.
Kapitel 4 befasst sich mit der Spezifizierung und Umsetzung der firmeninternen Lösung. Dabei werden die Grundlagen der Programmierung der Prüfungen erläutert. Außerdem werden ausgewählte Prüfungen und deren Einbindung in NX und in einen Workflow vorgestellt. Weiterhin wird die Erstellung von Prüf-Profilen und einer Toolbar erklärt.
In Kapitel 5 werden die Tests der Prüfungen in NX vorgestellt. Des Weiteren wird die Auswertung und Verwaltung der Prüfergebnisse in Teamcenter beschrieben.
Kapitel 6 befasst sich mit der Enduser-Dokumentation.
In Kapitel 7 und Kapitel 8 folgen eine Zusammenfassung der Arbeit sowie Handlungsempfehlungen für die weitere Vorgehensweise.

2. Grundlagen

In diesem Kapitel wird der Stand der Technik im Hinblick auf CAD und die damit einhergehende CAD-Prüfung erläutert. Außerdem werden Begriffe und Programme, die in dieser Arbeit verwendet werden, beschrieben. Weiterhin werden Fehler und Ursachen in CAD-Modellen und Zeichnungen sowie Verfahren und Lösungen zur automatisierten Prüfung beschrieben.

2.1. Stand der Technik

2.1.1. CAD

Ursprünglich war CAD als Software zum Erstellen und Verändern von 2D-Zeichnungen vorgesehen. Sketchpad, 1963 von Ivan Sutherland entwickelt, war das erste Programm, mit dem Skizzen digital erstellt und verändert werden konnten. In den folgenden Jahrzehnten wurde die Technik verfeinert und auf die dritte Dimension ausgeweitet. Anfang der 1980er Jahre kam mit AutoCAD die erste CAD-Software für den PC auf den Markt. Alle Vorgänger basierten auf Großrechnern. Heute ist die CAD-Technik nicht mehr auf das Konstruieren von Modellen und Zeichnungen beschränkt, sondern wird unter anderem durch Zusatzfunktionen, wie die Finite-Elemente-Methode (FEM) oder die Generierung von CNC-Programmen erweitert. Außerdem können die Daten in ein Product-Lifecycle-Management-System (PLM) eingebunden werden. CAD-Software ist auf allen gängigen Betriebssystemen vertreten und kann auf fast allen Geräten genutzt werden. Beim Cloud-Based-CAD gibt es beispielsweise keine Anforderungen an die Hardware des Entwicklers mehr, da alle Prozesse in eine Cloud ausgelagert werden. Außerdem kann CAD in der Entwicklung auch auf die Augmented-Reality (AR) und die Virtual-Reality (VR) Technik zur Erstellung und Präsentation von Modellen zurückgreifen. Des Weiteren besteht die Möglichkeit, das Generative Design zu verwenden. Generative Design bedeutet, dass die CAD-Software in kurzer Zeit mehrere Designvorschläge zu einem Problem herausgibt und der Entwickler zwischen den Vorschlägen auswählen kann. [1] [6]

2.1.2. CAD-Prüfung

Eine automatisierte Prüfung der CAD-Daten war in den ersten Jahrzehnten nicht möglich. Die Entwickler mussten alle inhaltlichen und systeminternen Punkte ihres Modells manuell überprüfen. Diese Vorgehensweise war zeitintensiv und besonders bei großen und komplexen Modellen sehr fehleranfällig. Mit der Zeit wurden Prüftools in die CAD-Systeme implementiert. Diese hatten eine Reihe von mitgelieferten Prüfungen integriert, um definierte systeminterne Punkte abprüfen zu können und die Qualität der Daten zu steigern. Heute haben die meisten, in den Unternehmen angewendeten,

CAD-Systeme eigene Prüftools. Außerdem bieten verschiedene externe Unternehmen ihre eigenen Prüftools für alle gängigen Datenformate an. Dabei gibt es sowohl Prüftools, die in die Oberfläche von CAD-Systemen integriert werden können, als auch externe Tools, die eine CAD-Datei einlesen und prüfen. [2]

2.2. Begriffserläuterungen

2.2.1. Workflow

Ein Workflow ist eine strukturierte Aneinanderreihung von Aufgaben bzw. Arbeitsabläufen im PLM-System mit dem Zweck, Prozesse wiederholbar und zeitsparend abzuschließen. Im Laufe des Workflows werden automatisiert Dokumente, Informationen oder Aufgaben unter den Workflow-Teilnehmern verteilt. Bestimmte Aufgaben können auch vom PLM-System automatisch abgearbeitet werden.

Folgende Aufgaben werden bei Rheinmetall typischerweise innerhalb eines Workflows durchlaufen:
- Prüfen von Objekten
- Generieren von Neutralformaten (JT, PDF)
- Zuweisen von Aufgaben an Workflow-Teilnehmer
- Zuweisen von Prüfungen an Workflow-Teilnehmer
- Setzen von Status auf Objekte und Datasets

Die Vorteile eines Workflows sind:
- erhöhte Effizienz
- erhöhte Flexibilität
- verbesserte Prozesssteuerung

Eine spezielle Art der Workflows ist der Freigabe-Workflow. Dieser besteht aus einer Reihe von Prüfaufgaben, die im Falle des Bestehens zur Freigabe führen. Ein freigegebenes Objekt kann nicht mehr geändert werden. Um das Objekt zu ändern, muss es erneut revisioniert werden.[8]

2.2.2. Master-Modell-Prinzip

Das Master-Modell-Prinzip beschreibt eine Entkopplung von Modell und Zeichnung. Das bedeutet, dass das 3D-Modell als Master angelegt wird. Davon ausgehend wird in einer neuen Datei die Zeichnung abgeleitet. Änderungen in der Zeichnung wirken sich nicht auf das Modell aus. Änderungen im Modell hingegen wirken sich auch auf die Zeichnung aus. Außerdem können andere Dokumente, wie FEM-Berechnungen oder CNC-Programme, auf das Master-Teil referenzieren. Das Master-Modell-Prinzip wird verwendet, um lange Ladezeiten bei großen Baugruppen zu vermeiden. Die Größe der Dateien ist geringer, da nur das 3D-Modell geladen wird und nicht die dazugehörigen Zeichnungen. Diese können bei Bedarf jederzeit nachgeladen werden.

Außerdem können die Zugriffsrechte durch das Trennen von Modell und Zeichnung genauer definiert werden. Der Entwickler, der für das 3D-Modell zuständig ist, muss nicht zwingend auch die Zeichnung erstellen und bearbeiten.

2.2.3. Knowledge Fusion

Knowledge Fusion (KF) ist eine Verbindung des CAD-Systems Siemens NX mit dem Knowledge Based Engineering (KBE). KBE dient der Erfassung von altem und neuem Wissen im Unternehmen und der automatisierten Anwendung dieses Wissens in der Entwicklung. In Verbindung mit NX können so unter anderem automatisiert Prüfungen durchgeführt werden. Als Basis von KF dient die objektorientierte Programmiersprache Intent! der Heide Corp. (USA).[10]

2.2.4. NXOpen

NXOpen beschreibt eine Reihe von Programmierschnittstellen, die die Erstellung eigener Applikationen für NX ermöglicht. Dafür können viele unterschiedliche Hochsprachen (C/C++, Visual Basic, C#, Java und Python) verwendet werden. Mit Hilfe von NXOpen können Prozesse automatisiert, das Interface angepasst oder externe Applikationen eingebunden werden.[11]

2.3. Programme

2.3.1. Siemens NX

Siemens NX ist ein CAD/CAM/CAE-System, welches 1978 auf den Markt kam. Das System bietet viele weitere Funktionen, unter anderem für die Simulation oder die CNC-Programmierung. Ein großer Vorteil von NX ist die einfache Verbindung der Daten mit der PLM-Software Siemens Teamcenter. Für die Erstellung eigener NX-Applikationen kann NXOpen verwendet werden.[7]

2.3.2. Microsoft Visual Studio 2017

Visual Studio ist eine Entwicklungsumgebung und unterstützt verschiedene Hochsprachen, wie z.B. Visual Basic (VB), C, C++, C#. Mit Hilfe von Visual Studio lassen sich unter anderem Windows-Programme, Web-Applikationen sowie Applikationen für Handys und Tablets programmieren. Visual Studio bietet ferner eine Auto-Vervollständigung der Befehle, eine Syntax-Anzeige und einen Debug-Mode.

2.3.3. Siemens NX Check-Mate

Check-Mate ist das interne Prüftool, welches in die Oberfläche von Siemens NX integriert ist. Es bietet eine Vielzahl an mitgelieferten Prüfungen, die auf der Basis von Knowledge Fusion programmiert sind. Die Einbindung der Prüfungen in NX erfolgt über spezielle dfa-Dateien, in denen der KF-Code der Prüfung enthalten ist. Darüber hinaus ist es möglich, zusätzliche Prüf-Dateien in die dfa-Datei einzubinden. Die Prüfungen in Check-Mate werden über ein spezielles User-Interface aufgerufen. Dabei können verschiedene Prüfungen angewählt werden, die anschließend per Mausklick nacheinander durchgeführt werden. Außerdem können Prüf-Profile ausgewählt werden, die eine definierte Gruppe von Prüfungen darstellen. In Abbildung 1 ist das User-Interface von Check-Mate beim Einrichten einer Prüfung dargestellt. Die Ergebnisse der Prüfung werden in Kapitel 5 näher erläutert.[9]

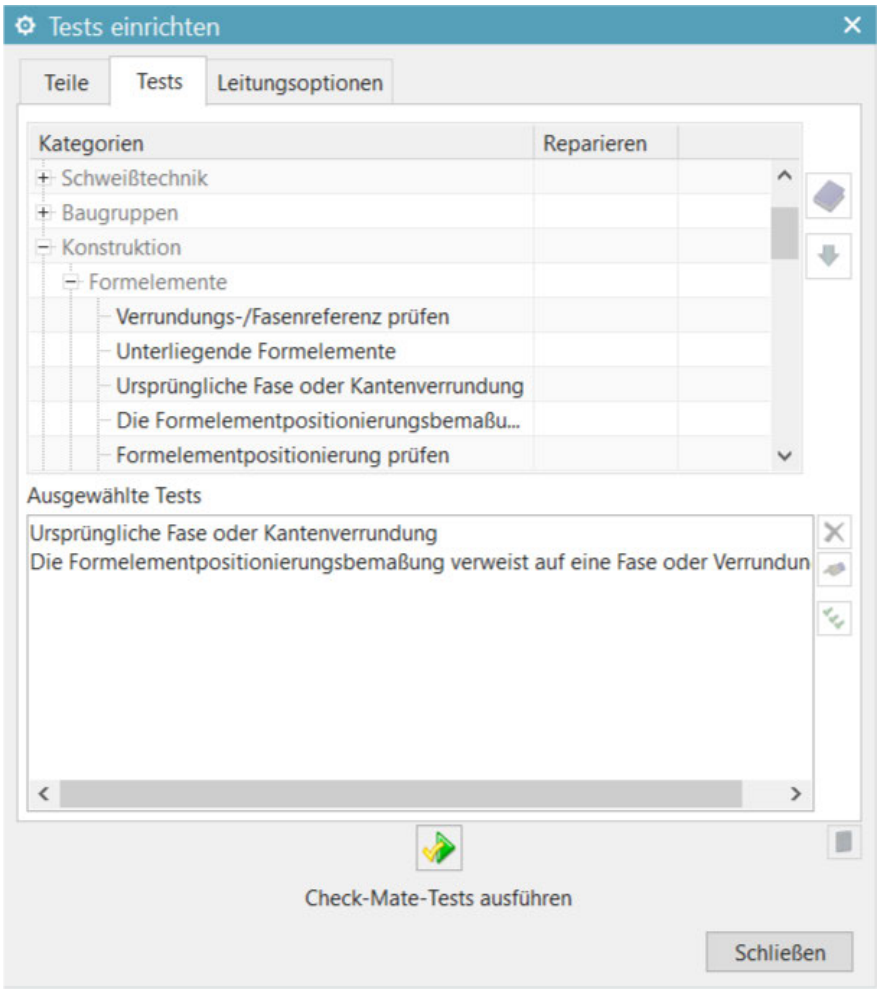

Abbildung 1.: User-Interface Check-Mate

2.3.4. Siemens Teamcenter

Teamcenter ist das PLM-System von Siemens. Durch ein PLM-System können alle Informationen eines Produktes über den gesamten Produktlebenszyklus zusammengeführt werden und sind zentral abrufbar. Dadurch werden die Prozesse innerhalb des Produktlebenszyklus beschleunigt und es kann effektiver gearbeitet werden. In Abbildung 2 werden die verschiedenen Bereiche eines Unternehmens, die Informationen an das PLM-System liefern, dargestellt.

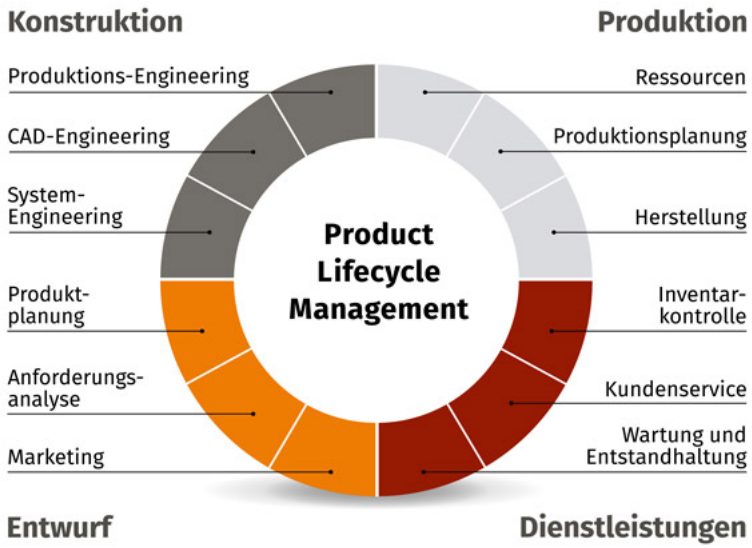

Abbildung 2.: Gliederung PLM-System [4]

Dabei ist die Einbindung nicht auf Daten des Unternehmens beschränkt. Es können auch Datensätze von Kunden oder externen Zulieferern in das System eingebunden werden. Die Daten in Teamcenter sind in einer Baumstruktur gegliedert. An oberster Stelle steht das Item. Ein Item beinhaltet alle Informationen über ein Teil. Auf der ersten Ebene unter dem Item befinden sich die Revisionen. In den Revisionen sind die verschiedenen Änderungsstände des Items abgebildet. Innerhalb jeder Revision finden sich das Modell, die Zeichnung und zusätzliche Dokumente des jeweiligen Änderungsstands. So ist die Historie des Items einsehbar und der Rücksprung auf einen früheren Änderungsstand möglich. Modell, Zeichnung und andere Dokumente sind in den Item-Revisionen als Datasets angelegt. Ein Dataset ist ein Element in Teamcenter, in dem die dazugehörigen Dateien als benannte Referenzen verwaltet werden. Über den Typ des Datasets wird die entsprechende Softwareanwendung verknüpft. Der Aufbau der Baumstruktur ist in Abbildung 3 beispielhaft ersichtlich.[8]

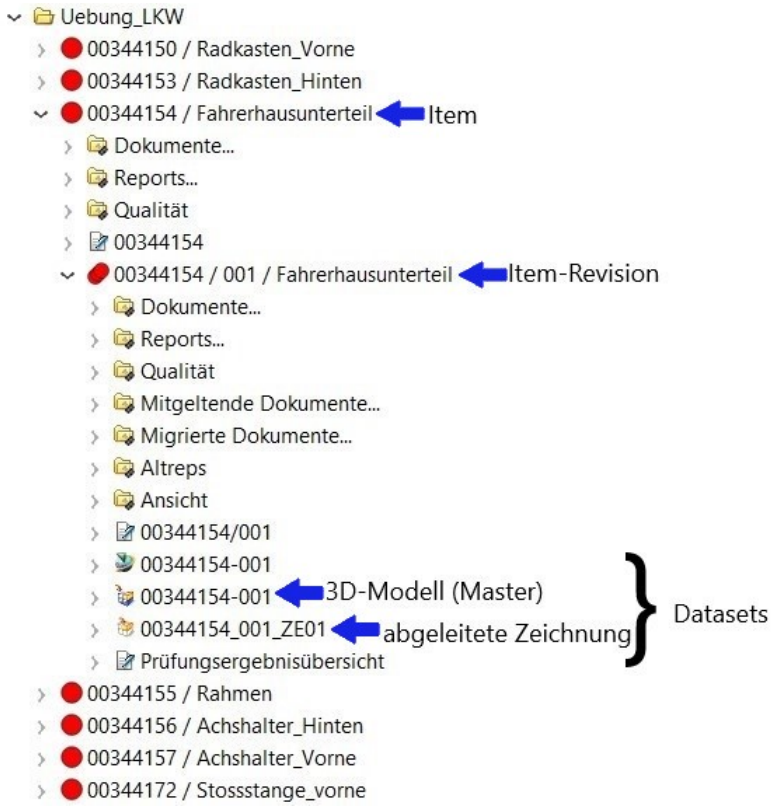

Abbildung 3.: Baumstruktur Teamcenter

2.4. Fehler und deren Ursachen in CAD-Modellen und Zeichnungen

In einem Konstruktionsprojekt gibt es eine Vielzahl von Fehlerquellen. Oft arbeiten mehrere Entwickler an einem Projekt. Ohne einheitliche CAD-Struktur arbeitet jeder Entwickler nach eigenen Methoden und Vorgehensweisen. Durch die daraus entstehenden unterschiedlich aufgebauten CAD-Modelle ist es für einen Entwickler, der neu zu einem Projekt hinzukommt, sehr schwierig, sich in die Daten einzuarbeiten. Um eine einheitliche Struktur und Datenqualität zu erreichen, ist es sinnvoll, eine Richtlinie einzuführen, die die Arbeitsweisen und Strukturen im Umgang mit CAD-Daten regelt. Folgend ist eine Auswahl an Fehlertypen aufgeführt, die die Qualität der Daten beeinflussen können.

Layer:
Layer steuern die Sichtbarkeit von Elementen in CAD-Modellen. Dabei werden verschiedene Gruppen von darstellbaren Objekten, wie Skizzen, Bezugsobjekte, Hilfsgeometrie und Modellgeometrie, unterschiedlichen Layerbereichen zugeordnet. In NX können die unterschiedlichen Layergruppen nach Bedarf ein- und ausgeblendet werden. Je nach Voreinstellung des Systems, können definierte Gruppen auch automatisch ausgeblendet werden. Außerdem können die Layergruppen vom Unternehmen frei gewählt werden. Die Verschiebung der einzelnen Elemente auf die jeweiligen Layer kann manuell oder automatisiert erfolgen. Bei der manuellen Belegung kann eine falsche Layerauswahl dazu führen, dass Hilfsgeometrie nicht ausgeblendet wird und als „echte“ Geometrie angesehen wird.

Löschen bzw. Unterdrücken von Elementen:
Änderungen am Modell können sich auf andere Features auswirken. Löschen oder Unterdrücken einzelner Features kann dazu führen, dass darauf referenzierende Features nicht mehr dargestellt werden können. In Abbildung 4 ist dieser Fehler ersichtlich. Durch Unterdrücken einer Extrusion wurde die darauf referenzierende Kantenverrundung ebenfalls unterdrückt.

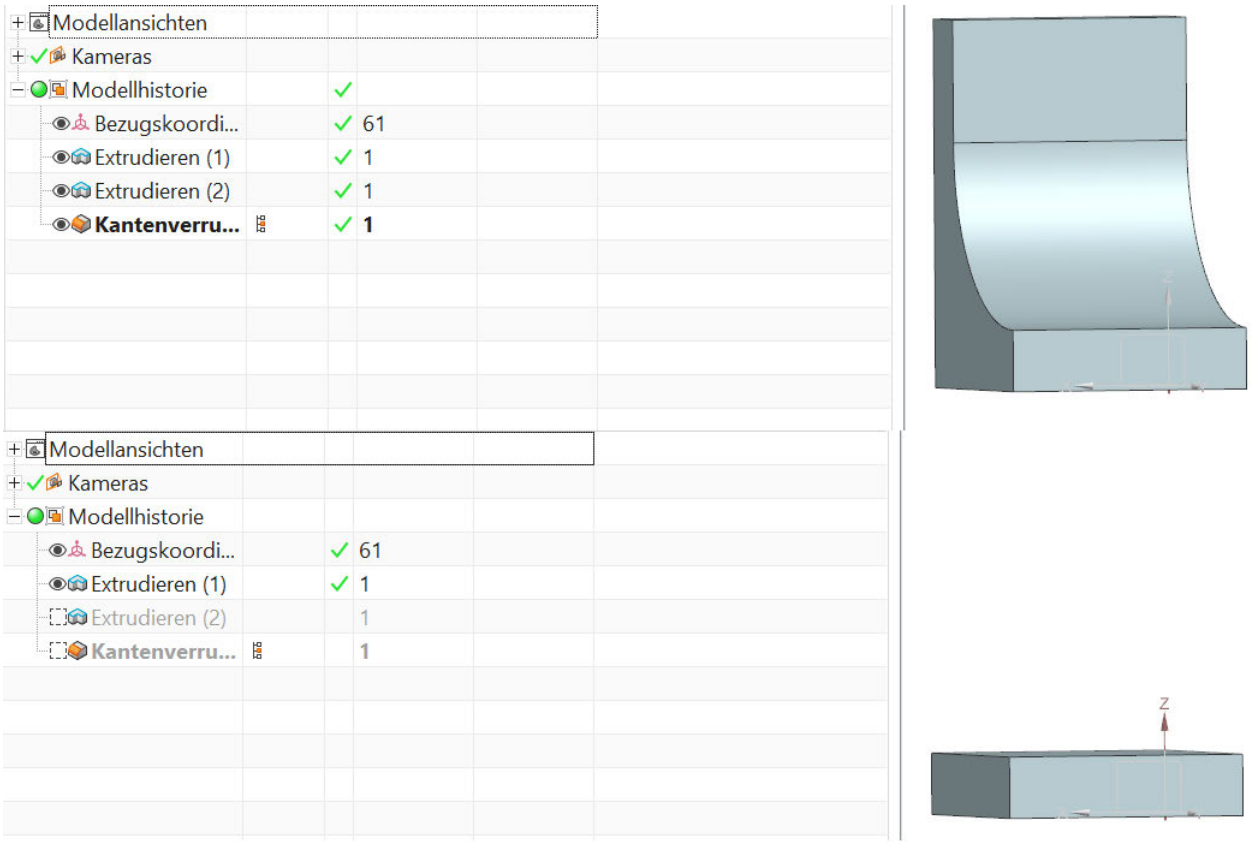

Abbildung 4.: Fehler durch Unterdrücken von Features

Überschreiben von Dateien:
Wenn die CAD-Daten nicht von einem PLM-System verwaltet werden, besteht die Gefahr, dass Dateien unabsichtlich überschrieben werden.

Zwangsbedingungen:
Zwangsbedingungen bestimmen die Freiheitsgrade der Komponenten einer Baugruppe. Des Weiteren bestimmen sie die Interaktionen der Komponenten untereinander. So kann die Funktion einer beweglichen Baugruppe mittels Zwangsbedingungen im 3D-Modell überprüft werden. Ist ein Modell vollständig bestimmt, besitzt es keine Freiheitsgrade und kann sich demnach weder translatorisch bewegen, noch drehen. In einem unterbestimmten Modell können sich eine oder mehrere Komponenten bewegen. In einem überbestimmten System behindern sich die Zwangsbedingungen gegenseitig. Dadurch ist die Behinderung der Freiheitsgrade vom CAD-System nicht eindeutig definierbar und die Zwangsbedingungen werden nicht berechnet. Ein Beispiel eines überbestimmten Modells ist in Abbildung 5 dargestellt. Hier sind die fehlerhaften Zwangsbedingungen vom System rot markiert.

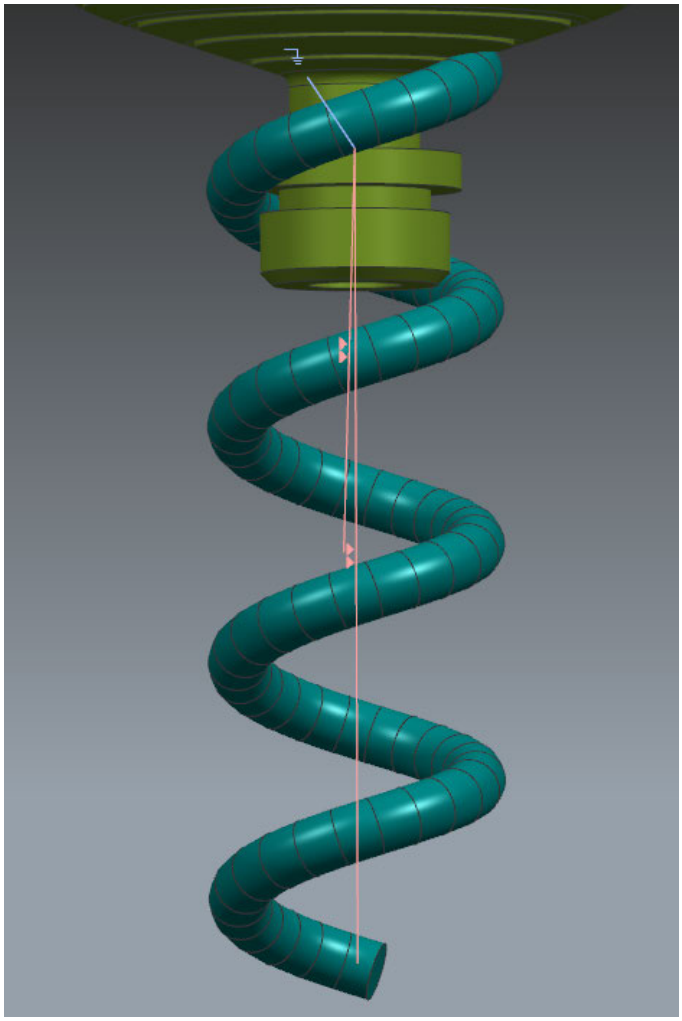

Abbildung 5.: Inkonsistente Zwangsbedingungen

Flächenmodelle:
Bei der Generierung von Volumenmodellen aus Flächen können Änderungen an den Flächen offene Schnittkanten oder sich schneidende Flächen nach sich ziehen, sodass kein Volumenmodell mehr generiert werden kann. In Abbildung 6 wird ein solches Beispiel gezeigt. Bei diesem Beispiel handelt es sich um die Spitze eines Propellers, welcher mit Hilfe einer Studio-Oberfläche generiert wurde.

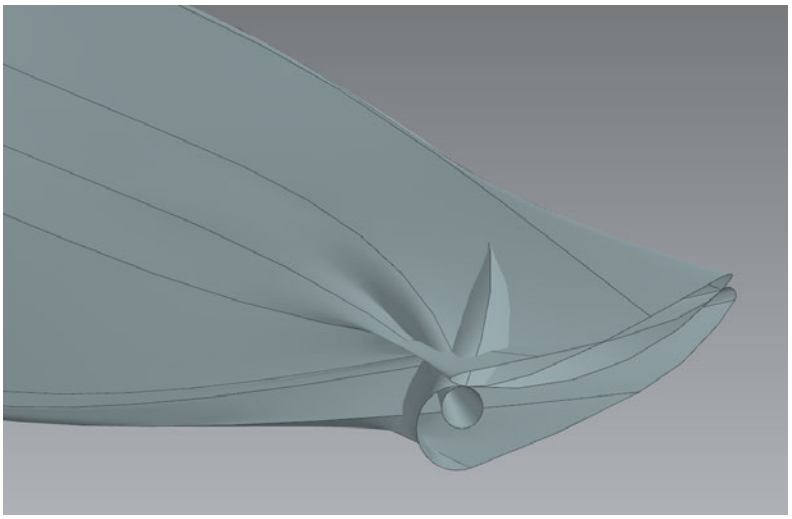

Abbildung 6.: Fehler im Flächenmodell

Kollision von Komponenten:
In einer Baugruppe können einzelne Komponenten kollidieren. Eine Montage der realen Bauteile ist so nicht möglich. In Abbildung 7 ist ein solches Beispiel aufgeführt.

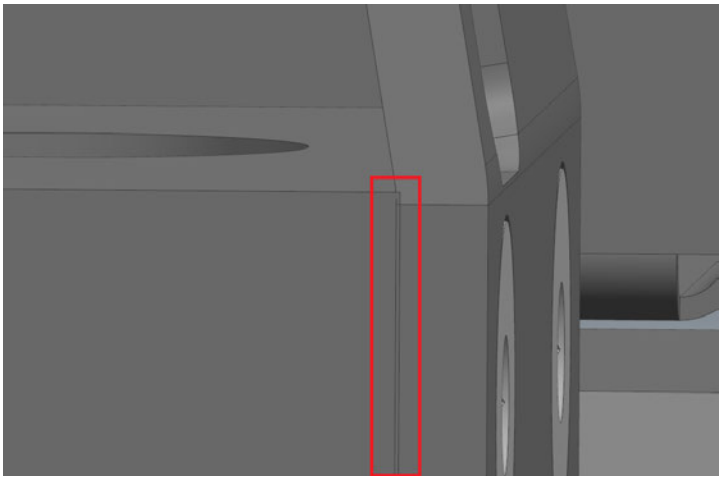

Abbildung 7.: Konturüberschneidung am Modell

Konvertierung:
Eine andere Fehlerquelle ist das Konvertieren von Daten in andere Formate. Jede große CAD-Software hat eigene Datenformate für ihre Daten entwickelt. Da die meisten Datenformate grundsätzlich verschieden sind, werden bestimmte Anteile bei der Konvertierung nicht übertragen. Durch unterschiedliche Import/Export-Einstellungen wird dieser Effekt noch verstärkt. In Abbildung 8 ist ein möglicher Fehler durch Datenkonvertierung dargestellt.

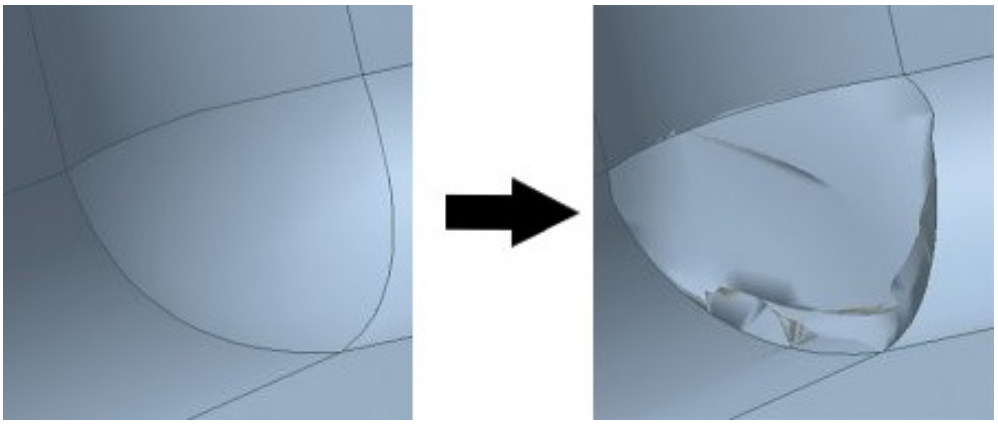

Abbildung 8.: Konvertierungsfehler Außenecke (nach [3])

Um Fehler bei der Datenkonvertierung zu reduzieren, wurden verschiedene Neutralformate entwickelt, die teilweise durch Normen geregelt sind. Diese Neutralformate können von den verschiedenen CAD-Systemen gelesen und geschrieben werden. Wird ein Modell als Neutralformat abgespeichert, erfährt es eine Konvertierung. Wenn das Modell wieder eingelesen wird, wird es erneut konvertiert. Da einige dieser Formate einer Norm unterliegen, sind die Gefahren eines Konvertierungsfehlers nicht so hoch wie bei der direkten Konvertierung in ein anderes CAD-Systemeigenes Format. Die gängigsten Neutralformate sind STEP und JT. JT steht für „Jupiter Tesselation“ und ist in ISO 14306:2012 genormt. Die Geometrie des CAD-Modells kann in JT tesseliert (als Dreicksflächen) oder als exakte Boundary Representation (Brep) dargestellt werden. In Brep werden die Geometrien durch ihre zusammengesetzten Oberflächen beschrieben. Des Weiteren können in JT auch Hintergrundinformationen, wie z.B. Maße und Toleranzen, abgespeichert werden. STEP steht für „Standard for the Exchange of Product model data“ und ist in ISO 10303 genormt. Auch in STEP werden verschiedene Hintergrundinformationen gespeichert. Hierbei entscheidet das Applikationsprotokoll, welche Informationen gespeichert werden. Für die Auswahl des Applikationsprotokolls ist der Anwendungsbereich ausschlaggebend. Es gibt unter anderem Protokolle für die Automobilindustrie, die Luft- und Raumfahrt oder auch den Schiffbau.[5]

Viele der in diesem Abschnitt aufgezählten Fehlertypen können mit Hilfe von automatisierten Prüfungen erkannt werden, wodurch die Datenqualität nach der Bereinigung der Fehler signifikant verbessert wird.

2.5. Verfahren und Lösungen zur automatisierten Prüfung von CAD-Modellen und Zeichnungen

Die meisten großen CAD-Softwarelösungen haben ein eigenes integriertes Prüftool. Dabei sind die Prüftools auf die jeweilige Software und ihren Einsatzzweck zugeschnitten. Diese Prüftools beinhalten vom Hersteller mitgelieferte Prüfungen zu grundlegenden Merkmalen von CAD-Daten, wie z.B. Kollisionsabfragen oder Toleranzabfragen. Viele Unternehmen haben eigene spezielle Regeln und Richtlinien, die nicht durch die mitgelieferten Prüfungen abgedeckt werden. Für diesen Zweck bieten einige Softwareunternehmen ihre eigenen Prüftools an, die entweder in die CAD-Software eingebettet werden können, oder extern bereitgestellt werden. Die externen Prüftools arbeiten meist mit neutralen CAD-Datenformaten[2] und können an die Bedürfnisse des Benutzers angepasst werden. CAD-Software, wie Siemens NX, bietet ebenfalls die Möglichkeit, eigene Prüfungen innerhalb des implementierten Prüftools zu schreiben. Im Falle von NX sind dafür Kenntnisse in Knowledge Fusion nötig.
Die Prüfungen können vom Entwickler selbst innerhalb der CAD-Software oder automatisch vom PLM-System im Zuge eines Workflows durchgeführt werden.

3. Analyse der Konstruktionsrichtlinie

In diesem Kapitel wird der Stand der Qualitätssicherung im Unternehmen erläutert. Außerdem werden die Punkte der Konstruktionsrichtlinie auf ihre Prüfbarkeit mittels Check-Mate analysiert.

3.1. Stand der Qualitätssicherung im Unternehmen

Die Technische Datenverarbeitung betreut das PLM-System, in dem die Daten aller Standorte der Rheinmetall Waffe Munition GmbH (RWM) in Deutschland verwaltet werden. Bis zu diesem Jahr hatte jeder Standort seine eigenen Regeln und Richtlinien in Bezug auf die Konstruktion und die CAD-Daten. Um eine durchgehend hohe und vor allem standortunabhängige Qualität der CAD-Daten zu gewährleisten, hat Rheinmetall eine standortübergreifende Konstruktionsrichtlinie eingeführt. Diese Richtlinie ist für alle Entwickler verpflichtend. Die Einhaltung der Unterpunkte der Richtlinie soll im Laufe eines Freigabeprozesses automatisiert abgeprüft werden. Auch bestehende CAD-Daten sollen nach dieser Richtlinie überprüft werden, wenn ein neuer Freigabestatus vergeben werden soll. Dabei ist das Erstellungsdatum des Modells wichtig. Modelle, die vor dem Inkrafttreten der Konstruktionsrichtlinie generiert worden sind, aber erst nach Inkrafttreten der Konstruktionsrichtlinie freigegeben werden sollen, werden anders geprüft als Modelle, die nach Inkrafttreten der Konstruktionsrichtlinie generiert worden sind. Danach wird in Abhängigkeit des Alters der Daten entschieden, welche Prüfungen durchgeführt werden. Außerdem wird entschieden, welche Fehlermerkmale in den Prüfungen als kritisch angesehen werden und eine Nachbearbeitung erfordern oder ob vorerst nur eine Warnung ausgegeben wird.

3.2. Auszug aus der Konstruktionsrichtlinie

Die Konstruktionsrichtlinie der Rheinmetall Waffe Munition GmbH ist ein Leitfaden, nach dem alle neuen Modelle und Zeichnungen erstellt werden müssen. In der Konstruktionsrichtlinie sind bestimmte Unterpunkte definiert, die automatisiert abgeprüft werden sollen. In Tabelle 1 wird eine Übersicht dieser Unterpunkte gezeigt. Im Folgenden sollen diese Punkte auf ihre Prüfbarkeit mittels Check-Mate analysiert werden.

Tabelle 1.: Auszug der Konstruktionsrichtlinie

1	**Modell**
1.1	Sind die einzelnen Elemente im Modell (Körper, Bezugselemente, Skizzen, usw.) auf den richtigen Layern abgelegt?
1.2	Sind Grundformelemente in dem Modell enthalten?
1.3	In einem Element darf es keine fixierten Bezugselemente geben
1.4	Sind die Skizzen vollständig bestimmt?
1.5	Sind geometrische Elemente innerhalb von Skizzen fixiert? (Ausnahme bilden Splines)
1.6	Sind unterdrückte Formelemente enthalten?
1.7	WAVE-Links dürfen nicht aufgebrochen werden
1.8	Wurden Objekte mit der Funktion „Körper anheben“genutzt?
2	**Baugruppen**
2.1	In einer Baugruppe müssen alle Komponenten mit Hilfe von Baugruppenzwangsbedingungen vollständig bestimmt positioniert werden
2.2	Es dürfen keine inkonsistenten Zwangsbedingungen vorhanden sein
2.3	Es dürfen keine unterdrückten Baugruppenzwangsbedingungen vorhanden sein
2.4	Alle weiteren Komponenten müssen an die erste Komponente assoziativ verknüpft werden
2.5	In Baugruppen ist „fixieren“nicht erlaubt, „binden“ist zu vermeiden
2.6	Die voreingestellte Laderegel „Latest Working“ist einzuhalten
3	**Zeichnung**
3.1	Master-Modell-Prinzip eingehalten?
3.2	Vor der Freigabe muss darauf geachtet werden, dass alle Maße assoziativ sind. Nicht assoziative Maße werden vom System farbig „braun“markiert
4	**Teamcenter**
4.1	In einer Revision darf es nur ein Dataset vom Typ „UGMASTER“geben
4.2	Ist ein „ST-Report“in Teamcenter enthalten?
4.3	Definierte Attribute müssen in Teamcenter gepflegt werden
4.4	Sind die Daten ausgecheckt?

3.3. Analyse der Unterpunkte

3.3.1. Modellprüfungen

- 1.1: Die Prüfung der Layer ist für die Übersichtlichkeit, besonders bei sehr komplexen Modellen, notwendig. Die Layer steuern die Sichtbarkeit bestimmter Elemente im Modell. Je nach Konfiguration des Systems können Skizzen und Bezugselemente automatisch ausgeblendet werden, sobald sie dem jeweiligen Layer zugewiesen werden. Über NXOpen kann auf die einzelnen Elemente eines Modells zugegriffen werden. Dabei kann auch der Layer überprüft werden, auf dem das Element liegt. Durch einen Abgleich mit dem erlaubten Layerbereich können die falschen Layer herausgefiltert werden. Die Layer von Körpern können nicht überprüft werden, da das Programm nicht zwischen Hilfs- und Modellgeometrie unterscheiden kann.

- 1.2: Laut der Konstruktionsrichtlinie sind Einzelteile als Skizze mit anschließender Extrusion, Drehung, o.ä. zu erstellen. Grundformelemente sind nicht erlaubt. Über NXOpen kann das formgebende Feature des Elements abgefragt werden.

- 1.3: Bezugselemente sollen assoziativ zu bestehenden Körpern oder Bezugselementen erzeugt werden. Fixierte Bezugselemente entsprechen nicht der parametrischen Arbeitsweise. Der Status der Bezugselemente kann über NXOpen abgefragt werden. Durch eine Abfrage, ob das jeweilige Bezugselement fixiert ist, können diese Elemente erfasst werden.

- 1.4: Um ein Modell berechenbar modifizieren zu können, müssen die Skizzen des Modells geometrisch vollständig bestimmt werden. Check-Mate hat zu diesem Zweck eine mitgelieferte Prüfung integriert. Diese kontrolliert bei jeder Skizze, ob sie über- oder unterbestimmt ist. Die fehlerhaften Skizzen werden markiert und als Liste ausgegeben.

- 1.5: Laut Konstruktionsrichtlinie ist das Fixieren von geometrischen Elementen innerhalb von Skizzen nicht erlaubt. Splines sind von dieser Vorgabe ausgenommen. Die Eigenschaften von jedem Element einer Skizze werden überprüft. Dabei wird der Typ des Elements abgefragt. Ist dieses vom Typ Spline, wird es von der weiteren Prüfung ausgenommen. Ist das Element kein Spline, wird überprüft, ob das Element fixiert ist.

- 1.6: Unterdrückte Formelemente haben keinen Nutzen für das Modell und sind daher zu löschen. Über NXOpen können alle Körper des Modells auf den jeweiligen Status überprüft werden. Ist ein Körper unterdrückt, ist die Prüfung nicht bestanden.

- 1.7: WAVE-Links sind Verbindungen von Geometrien und Körpern einer Komponente zu einer anderen Komponente in einer Baugruppe. Diese WAVE-Links dürfen nicht aufgebrochen werden. In Check-Mate gibt es eine mitgelieferte Prüfung. Diese prüft, ob alle WAVE-Links intakt sind. Ist ein WAVE-Link aufgebrochen, wird er als Fehler markiert und die Liste aller Fehler wird wieder ausgegeben.

- 1.8: Die Funktion „Anheben“ ist laut der Konstruktionsrichtlinie verboten. In den Eigenschaften der Körper eines Modells kann auf die einzelnen Features zugegriffen werden. Ist das Feature „Anheben“ vorhanden, gilt die Prüfung als nicht bestanden.

3.3.2. Baugruppenprüfungen

- 2.1: In einer Baugruppe muss die geometrische Position aller Komponenten vollständig mittels Baugruppenzwangsbedingungen bestimmt sein, da so unbeabsichtigte Verschiebungen einzelner Komponenten oder Unterbaugruppen bei Verschiebung anderer Komponenten vermieden werden.
Über NXOpen kann auf die einzelnen Rotations- und Translationsfreiheitsgrade jeder Komponente zugegriffen werden.

- 2.2: Durch inkonsistente Zwangsbedingungen werden die Freiheitsgrade einer Komponente nicht beschränkt. Dadurch kann sich die Komponente, je nach Anzahl der Freiheitsgrade, bewegen. Innerhalb der Komponenten des Modells können die Eigenschaften der Zwangsbedingungen angezeigt werden. So können inkonsistente Zwangsbedingungen herausgefiltert werden.

- 2.3: Unterdrückte Zwangsbedingungen haben keinen Nutzen für das Modell. Daher müssen die unterdrückten Zwangsbedingungen gelöscht werden. Wie in Punkt 2.2 kann über die Eigenschaften der Zwangsbedingungen überprüft werden, ob eine Zwangsbedingung unterdrückt ist.

- 2.4: Laut der Konstruktionsrichtlinie soll die erste Komponente einer Baugruppe mit dem Ursprungskoordinatensystem verknüpft werden. Alle anderen Komponenten sollen mit dieser Komponente verknüpft werden. Durch die Eigenschaften der Komponenten einer Baugruppe kann nicht überprüft werden, an welche Komponente diese Elemente angeknüpft sind. Daher ist eine Prüfung in CheckMate nicht möglich.

- 2.5: „Fixieren“ und „Binden“ sind Zwangsbedingungen, die laut Konstruktionsrichtlinie verboten sind, bzw. vermieden werden sollen. Wie Bezugselemente und geometrische Elemente in Skizzen sind fixierte Zwangsbedingungen nicht Teil der parametrischen Arbeitsweise. Der Typ der Zwangsbedingungen jeder Komponente kann über NXOpen abgefragt werden. Wurde „Fixieren“ angewandt, ist die Komponente fehlerhaft, bei „Binden“ wird eine Warnung ausgegeben.

- 2.6: Für eine standortunabhängige und strukturierte Arbeitsweise müssen alle Entwickler nach der gleichen Laderegel arbeiten. Diese bestimmt, welche Revision eines Teils geladen wird. Bei der Laderegel „Latest Working“ wird beispielsweise immer die letzte Revision geladen. Die Laderegel kann nicht über das WorkPart ermittelt werden, sondern wird über die NX-Sitzung abgefragt. Die ermittelte Laderegel wird mit der Laderegel „Latest Working“ abgeglichen.

3.3.3. Zeichnungsprüfungen

- 3.1: In den RWM-Standorten wird nach dem Master-Modell-Prinzip gearbeitet. Um dies zu überprüfen, besitzt Check-Mate eine mitgelieferte Prüfung. Diese untersucht, ob ein Zeichnungsblatt im Master-Modell enthalten ist und ob die Zeichnung auf das Master-Modell referenziert.

- 3.2: Nicht-assoziative Maße ändern sich bei Änderung des Master-Modells nicht mit. Daher ist die Zeichnung nach einer Änderung des Modells fehlerhaft. Um die nicht-assoziativen Maße einer Zeichnung herauszufiltern, werden alle Bemaßungen einer Zeichnung abgefragt. In den Eigenschaften der Bemaßungen kann die Assoziativität abgeprüft werden. Maße, die nicht assoziativ mit einer Geometrie verknüpft sind, werden als Fehler markiert.

3.3.4. Teamcenterprüfungen

- 4.1: Laut Richtlinie darf es unter einer Revision nur ein Dataset vom Typ „UGMaster“ geben. Ist dieser Datentyp schon unter einer Revision vorhanden, kann kein weiteres Dataset diesen Typs angelegt werden. Somit ist keine Prüfung notwendig.

- 4.2: Der Entwickler muss bei der Erstellung einer Baugruppe festlegen, ob diese eine Stückliste (ST-Report) benötigt. Da es Baugruppen ohne Stückliste gibt, ist eine einheitliche Prüfung nicht möglich. Vorhandene Stücklisten werden innerhalb eines Workflows automatisch aktualisiert.

- 4.3: Nicht alle Attribute eines Elements werden standardmäßig von Teamcenter an NX übergeben. Diese Attribute müssten manuell als Übergabeparameter definiert werden. Deshalb werden sie in Teamcenter innerhalb eines Workflows abgeprüft. Sind die Attribute nicht gepflegt wird der Workflow beendet.

- 4.4: Damit Daten innerhalb eines Workflows geändert werden können, müssen sie eingecheckt sein. Daher wird am Anfang des Workflows überprüft, ob alle Daten, die zum Workflow gehören, eingecheckt sind. Sollten Daten ausgecheckt sein, kann der Workflow nicht starten. Außerdem werden bestimmte Elemente im Workflowverlauf nochmals überprüft, da bei der Durchsicht durch die Prüfer diese Elemente ausgecheckt werden. Hat ein Prüfer zur Freigabe die Elemente nicht wieder eingecheckt, wird dies automatisch durch den Workflow übernommen.

4. Umsetzung einer firmeninternen Lösung

In diesem Kapitel wird eine firmeninterne Lösung festgelegt. Außerdem werden die Grundlagen der Programmierung von Prüfungen erläutert. Weiterhin wird der Code ausgewählter Prüfungen sowie deren Einbindung in NX erklärt. Anschließend wird die Erstellung von Prüf-Profilen, die Einbindung der Prüfungen in einen Workflow und die Erstellung eines User-Interface in NX beschrieben.

4.1. Spezifikation der firmeninternen Lösung

Die CAD-Struktur in den RWM-Standorten in Deutschland basiert auf Siemens NX. Verwaltet werden die Daten im PLM-System Siemens Teamcenter. NX bietet das integrierte Prüftool Check-Mate, in welchem auch die Programmierung eigener Prüfungen möglich ist. Außerdem sind NX und Teamcenter aufeinander abgestimmt. Daher erfolgt die Prüfung der Qualität der CAD-Daten innerhalb dieser Systeme. Die Prüfungen werden nicht in Knowledge-Fusion-Code geschrieben, da die zugrunde liegende Programmiersprache nicht trivial ist und eine Schulung in dieser Sprache nicht wirtschaftlich gewesen wäre. Aufgrund der vorliegenden Kenntnisse im Visual Basic Coding, der Auto-Vervollständigung der Befehle und dem Debug-Mode wurde die Erstellung der Prüfungen in Visual Studio ausgelagert. Die Prüfungen sollen manuell in NX und automatisiert im Teamcenter-Workflow ausgeführt werden können. Daher muss am Ende der Prüfung ein Prüfbericht erstellt und in Teamcenter gespeichert werden. Dieser kann dann im Laufe des Workflows ausgewertet werden. Außerdem soll innerhalb eines Workflows zwischen Alt/Neu-Daten unterschieden werden. Die Unterscheidung wurde in Kapitel 3.1 bereits beschrieben.

4.2. Grundlagen der Programmierung von Prüfungen

4.2.1. Ablauf der Programmierung

Die Prüfungen für die Konstruktionsrichtlinie werden in das NX-eigene Prüftool Check-Mate eingebunden. Als Grundlage dazu dient Knowledge Fusion. Die Prüfprogramme werden mit Hilfe der Software Microsoft Visual Studio 2017 geschrieben. Eine Prüfung wird als Klassenbibliothek in der .NET-Framework Umgebung angelegt. Als Programmiersprache dient Visual Basic. Zu Anfang müssen verschiedene dll-Dateien für NXOpen in die Projektmappe geladen werden. Hierzu werden Verweise auf die Dateien angelegt. In Abbildung 9 sind diese Verweise innerhalb der Software dargestellt.

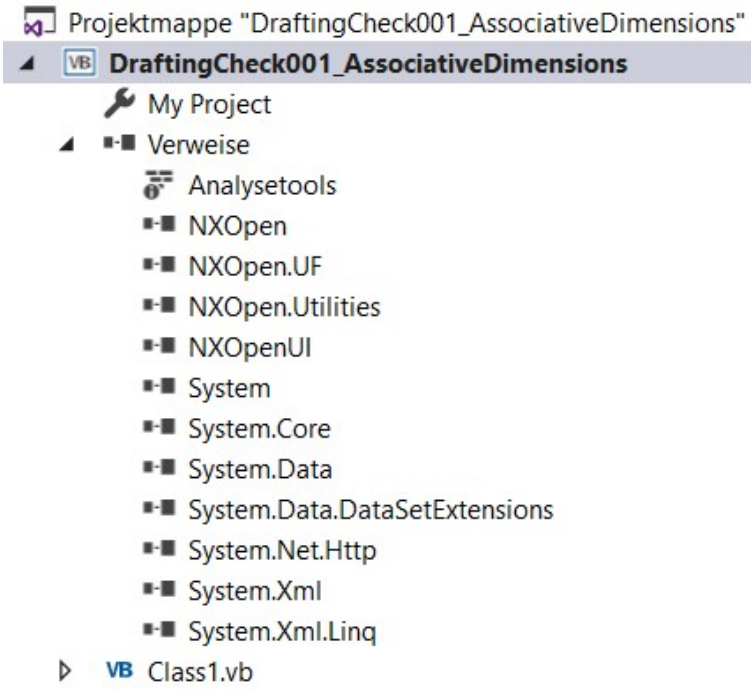

Abbildung 9.: Verweise auf NXOpen

Diese Dateien ermöglichen den Zugriff auf die erforderlichen NXOpen-Befehle. Eine Prüfung ist in mehrere Teile aufgegliedert, von denen jeder Teil eine bestimmte Funktion besitzt. Am Ende wird daraus eine dll-Datei generiert. Diese dll-Datei wird in einer speziellen dfa-Datei referenziert. Die dfa-Datei ist die Schnittstelle zu Check-Mate und beinhaltet den für das Aufrufen der Prüfung nötigen Knowledge-Fusion-Code. In der dfa-Datei wird außerdem der Anzeigename der Prüfung in NX sowie die Kategorie der Prüfung bestimmt. Beim Starten einer Prüfung in NX wird die dfa-Datei aufgerufen, die wiederum die dll-Datei in NX lädt. Nach der Prüfung wird die dll-Datei wieder aus NX entladen. Dies erfolgt automatisch durch eine Teilfunktion der Prüfung.

4.2.2. Liste der häufigsten Befehle

Die Befehle in NXOpen sind in eine Baumstruktur gegliedert. Dabei steht am Anfang eine Klasse, die wiederum in Unterklassen aufgeteilt sein kann. Durch Funktionen können Klassen verändert oder Informationen über die Klasse abgefragt werden. Dabei bezieht sich die Funktion immer auf die zuletzt aufgerufene Klasse. Die Struktur ist in Abbildung 10 beispielhaft dargestellt.

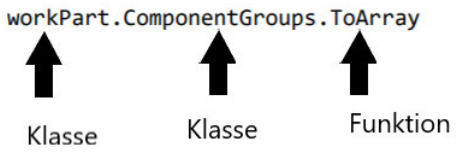

Abbildung 10.: Struktur eines NXOpen-Befehls

Nicht alle Funktionen können auf alle Klassen angewendet werden. Einige Klassen werden am Anfang jeder Prüfung als Variable deklariert. Darunter fallen das WorkPart und die Session. Dies dient dazu, dem Programmierer Arbeit zu sparen. Die wichtigsten Klassen und Funktionen sind in Tabelle 2 bzw. Tabelle 3 näher beschrieben.

Tabelle 2.: Häufige Klassen

Klasse	Beschreibung
Session.GetSession	derzeit geladene NX-Session
UF.UFSession.GetUFSession	derzeit geladene User-Funktion-Session
UI.GetUI	derzeit geladenes User-Interface
theSession.Parts.Work	derzeit geladenes Modell (WorkPart)
Bodies	Körper im WorkPart
Datums	Bezugselemente im WorkPart
Components	Komponenten im WorkPart
Sketches	Skizzen im WorkPart

Tabelle 3.: Häufige Funktionen

Funktion	Beschreibung
toArray	speichert Ergebnis in einem Array
toString	speichert Ergebnis in einem String
Tag	taggt das Element
JournalIdentifier	gibt den Elementnamen im Journal aus
Add	Fügt Element einer Liste hinzu
LibraryUnloadOption	Bestimmt die Ausgabe der Entladefunktion

4.2.3. Aufbau eines typischen Programms

Jedes Prüfprogramm besteht aus einem Deklarationsteil und vier Funktionen. Nach Abschluss der Programmierung wird aus dem Projekt in Visual Studio eine dll-Datei generiert, die durch eine dfa-Datei in NX geladen wird. Bevor eine der Funktionen durchgeführt wird, wird eine globale Variablendeklaration vorgenommen. Die hier deklarierten Variablen sind von allen Funktionen aufrufbar. Außerdem wird NXOpen in das Projekt geladen.

Die vier Funktionen eines Projekts sind:

1. Check-Funktion:

In der Check-Funktion wird die eigentliche Prüfung ausgeführt. In ihr werden die wichtigsten Variablen gesetzt, sowie Fehlerlisten angelegt. In der Prüfung wird ein bestimmtes Attribut des Modells

bzw. der Zeichnung abgefragt. Dabei wird eine Ja-Nein-Entscheidung getroffen und eine dementsprechende Rückmeldung gegeben. In den Fehlerlisten befinden sich die Tags der fehlerhaften Elemente. Tags markieren die betroffenen Elemente im NX-Arbeitsfenster, sodass sie vom Anwender sofort zu erkennen sind.

2. Message-Funktion:

In der Message-Funktion wird anhand der globalen Fehlervariable der Prüfung eine Nachricht ausgegeben.

3. Fehler-Log-Funktion:

In der Fehler-Log-Funktion wird anhand der globalen Fehlervariable entschieden, ob die Prüfung bestanden ist oder nicht.

4. Entladefunktion:

Die Entladefunktion wird nach jeder Funktion durchgeführt. Daher muss in jeder der Funktionen eine Entladeoption als Variable definiert werden. Bei der Entladeoption ist die Reihenfolge wichtig, in der die Funktionen durchgeführt werden. Die zuletzt durchgeführte Funktion ändert die Entladevariable so ab, dass die dll-Datei aus NX entladen wird. Alle anderen Funktionen verhindern dies, indem sie der Entladevariable einen anderen Wert zuweisen. Sollte die dll-Datei nicht aus NX entladen werden, kann sie nicht geändert werden. Dies ist besonders beim Programmieren und gleichzeitigem Testen der Prüfungen wichtig.

4.3. Beispiele ausgewählter Prüfungen

Die Prüfungen der Konstruktionsrichtlinie sind in drei Gruppen aufgeteilt. Im Folgenden wird jede der Gruppen anhand eines Beispiels vorgestellt.

4.3.1. Modellprüfungen

In den Modellprüfungen werden definierte Attribute der 3D-Einzelteile überprüft. Als Beispiel dient hier die Prüfung der Layer für die Skizzen im Modell. In Abbildung 11 ist der Variablendeklarationsteil der Prüfung mit der Einbindung von NXOpen ersichtlich.

```
1  Imports NXOpen
2
3  'Dieser Check prüft, ob die Skizzen auf den richtigen Layern liegen.
4
5  Public Module MainModule
6
7      'Variablendeklaration
8
9      Private SketchLayer As Boolean = True
10     Private UnloadOption As Session.LibraryUnloadOption =
         Session.LibraryUnloadOption.Immediately
11
```

Abbildung 11.: Globale Variablendeklaration

In Zeile 1 werden die Befehle aus NXOpen importiert. Dazu ist ein Verweis auf die NXOpen-dll-Dateien erforderlich. Anschließend werden die globalen Variablen deklariert. Es folgt die Check-Funktion der Prüfung. Diese ist in Abbildung 12 dargestellt.

```
12     Public Function SketchTag() As Tag()
13
14         'Allgemeine Konfiguration
15
16         UnloadOption = Session.LibraryUnloadOption.Explicitly
17
18         Dim theSession As Session = Session.GetSession
19         Dim theUI As UI = UI.GetUI
20         Dim workPart As Part = theSession.Parts.Work
21
22         'Erstellen leerer Fehlerliste
23         Dim SketchList As New List(Of Tag)
24
25         'Überprüfen jeder Skizze auf Fehler
26
27         For Each SketchItem As Sketch In workPart.Sketches.ToArray
28
29             'Überprüfen des richtigen Layers
30
31             If SketchItem.Layer < 20 Or SketchItem.Layer > 39 Then
32                 SketchLayer = False
33                 SketchList.Add(SketchItem.Tag)
34
35             End If
36
37         Next
38
39         'Rückgabe der Fehlerliste
40
41         Return SketchList.ToArray
42
```

Abbildung 12.: Check-Funktion einer Modellprüfung

In Zeile 16 wird die Entladevariable so definiert, dass die dll-Datei am Ende der Funktion nicht aus NX entladen wird. Anschließend folgt die allgemeine Variablendeklaration für die Funktion. Hier werden Variablen für Session und WorkPart gesetzt. Außerdem wird die Variable für die Fehlerliste definiert. In Zeile 27 wird jede Skizze des geladenen Modells in ein Array überführt. Danach wird jede Skizze in diesem Array auf den jeweiligen Layer überprüft. Laut Richtlinie müssen Skizzen in einem Layerbereich zwischen 20 und 39 liegen. Liegt der Layer nicht zwischen 20 und 39, wird die Skizze als fehlerhaft erkannt und mit einem Tag versehen, sowie in die Fehlerliste geschrieben. Außerdem wird die Fehlervariable in Zeile 32 auf falsch gesetzt. In Zeile 41 wird die Fehlerliste mit den Tags an die dfa-Datei übergeben. In Abbildung 13 wird die Message-Funktion der Prüfung gezeigt.

```
45      Public Function GetMessage() As String
46
47          UnloadOption = Session.LibraryUnloadOption.Immediately
48
49          'Ausgabenachricht
50
51          If SketchLayer Then
52              Return "bestanden"
53
54          Else
55              Return "Skizze auf falschem Layer"
56          End If
57
58      End Function
```

Abbildung 13.: Message-Funktion einer Modellprüfung

Im Verlauf einer Prüfung wird zuletzt diese Funktion ausgeführt. Daher muss hier die Entladeoption so eingestellt werden, dass die dll-Datei nach dieser Funktion aus NX entladen wird. In Zeile 51 wird überprüft, ob ein Fehler gefunden wurde. Dementsprechend wird eine Nachricht an die dfa-Datei übergeben. Die Fehler-Log-Funktion ist in Abbildung 14 dargestellt.

```
60      Public Function GetLog() As Boolean
61
62          Call SketchTag()
63          UnloadOption = Session.LibraryUnloadOption.Explicitly
64
65          'Entscheider, ob Fehler gefunden wurden
66
67          If SketchLayer Then
68              Return True
69
70          Else
71              Return False
72
73          End If
74
75      End Function
```

Abbildung 14.: Fehler-Log-Funktion einer Modellprüfung

Da diese Funktion am Anfang jeder Prüfung durchgeführt wird, ist ein Verweis auf die Check-Funktion erforderlich. Dieser ist in Zeile 62 ersichtlich. Die Fehler-Log-Funktion überprüft, ob ein Fehler gefunden wurde und gibt eine entsprechende Variable an die dfa-Datei. Durch diese Variable wird entschieden, ob die Prüfung bestanden ist oder nicht.
Am Ende jeder Prüfung steht die Entladefunktion, die nach jeder der anderen Funktionen durchgeführt wird. In Abbildung 15 ist der Aufbau einer Entladefunktion dargestellt.

```
77      'Entladen der .dll aus NX
78
79      Public Function GetUnloadOption() As Integer
80
81          GetUnloadOption = UnloadOption
82
83      End Function
```

Abbildung 15.: Entladefunktion

GetUnloadOption() ist eine interne Funktion von NX und bekommt nur eine Variable übergeben, in der entschieden wird, ob die dll-Datei entladen wird oder nicht.
Alle Prüfungen für Check-Mate sind ähnlich aufgebaut. Die Message- und Fehler-Log-Funktionen unterscheiden sich nur durch die unterschiedlichen Ausgabenachrichten bzw. Variablennamen. Die Entladefunktion ist in jeder Prüfung identisch. Daher wird in den folgenden Beispielen auf die Erklärung der einzelnen Message-, Fehler-Log- und Entladefunktionen verzichtet.

4.3.2. Baugruppenprüfungen

In den Baugruppenprüfungen werden die Eigenschaften der Gesamtbaugruppe, der Unterbaugruppen und der einzelnen Komponenten geprüft. Die Komponenten sind im System nicht mehr als Körper-Klasse angelegt. Somit sind die Eigenschaften der Einzelteile in einer Baugruppe nicht mehr abprüfbar. In Abbildung 16 wird die Check-Funktion der Prüfung der Komponenten einer Baugruppe auf die einzelnen Freiheitsgrade gezeigt.

```
12      Public Function ConstraintTag() As Tag()
13
14          'Allgemeine Konfiguration
15
16          UnloadOption = Session.LibraryUnloadOption.Explicitly
17
18          Dim theSession As Session = Session.GetSession
19          Dim theUI As UI = UI.GetUI
20          Dim workPart As Part = theSession.Parts.Work
21
22          'Erstellen leerer Fehlerliste
23
24          Dim ConstraintList As New List(Of Tag)
25
26          'Überprüfen von jedem Element
27
28          For Each GroupItem In workPart.ComponentGroups.ToArray
29
30              For Each ComponentItem As Assemblies.Component In
                  GroupItem.GetComponents.ToArray
31
32                  'Überprüfen, ob Komponente vollständig bestimmt ist
33
34                  If Not ComponentItem.GetDegreesOfFreedom.NumRotational = 0 Or
                      Not ComponentItem.GetDegreesOfFreedom.NumTranslational = 0
                      Then
35                      Constraintfully = False
36                      ConstraintList.Add(ComponentItem.Tag)
37                  End If
38
39              Next
40
41          Next
42
43          Return ConstraintList.ToArray
44
45      End Function
```

Abbildung 16.: Check-Funktion einer Baugruppenprüfung

In Zeile 28 werden die verschiedenen Komponentengruppen in ein Array überführt. Im Anschluss werden alle Komponenten aus den einzelnen Gruppen zusammengefasst in ein Array überführt. Danach werden die Freiheitsgrade jeder einzelnen Komponente geprüft. Sollte die Anzahl der Rotations- bzw. Translationsfreiheitsgrade über null liegen, ist die Komponente fehlerhaft. Die Fehlervariable wird auf falsch gesetzt, die Komponente wird getaggt und in die Fehlerliste geschrieben.

4.3.3. Zeichnungsprüfungen

In den Zeichnungsprüfungen werden die Einzelteil- und Baugruppenzeichnungen überprüft. In Abbildung 17 ist die Check-Funktion der Prüfung aller Maßangaben einer Zeichnung auf Assoziativität abgebildet.

```
12      Public Function AssociativeDimensions() As Tag()
13
14          'Allgemeine Konfiguration
15
16          UnloadOption = Session.LibraryUnloadOption.Explicitly
17
18          Dim theSession As Session = Session.GetSession
19          Dim theUfSession As UF.UFSession = UF.UFSession.GetUFSession
20          Dim theUI As UI = UI.GetUI
21          Dim workPart As Part = theSession.Parts.Work
22
23          'Erstellen leerer Liste fehlerhafter Bemaßungen
24
25          Dim DimensionObjects As New List(Of Tag)
26
27          'Überprüfen jeder Bemaßung auf Assoziativität
28
29          For Each DraftingDimension As Annotations.Dimension In
              workPart.Dimensions
30
31              If DraftingDimension.IsRetained = True Then
32                  DimensionObjects.Add(DraftingDimension.Tag)
33                  Fehler = True
34              End If
35
36          Next
37
38          'Rückgabe Fehlerliste
39
40          Return DimensionObjects.ToArray
41
42      End Function
```

Abbildung 17.: Check-Funktion einer Zeichnungsprüfung

Nach der Variablendeklaration werden in Zeile 29 alle Bemaßungen der Zeichnung überprüft. Ist eine Bemaßung nicht assoziativ zum Modell verknüpft, wird sie getaggt und in die Fehlerliste geschrieben. Außerdem wird die Fehlervariable auf falsch gesetzt.

4.4. Einbindung der Prüfungen in NX

Die in Visual Studio programmierten Prüfungen können nicht direkt in NX genutzt werden, da die generierten dll-Dateien nicht eingelesen werden können. Um die Dateien nutzen zu können, müssen sie in speziellen dfa-Dateien referenziert werden. Diese Dateien definieren die Prüfung für NX. Durch die Auslagerung der Prüf-Funktion in Visual Studio sind alle dfa-Dateien der Einzelprüfungen gleich aufgebaut. Zuerst wird der interne Name der Prüfung vergeben. Im Anschluss wird die Kategorie und der angezeigte Name der Prüfung im Check-Mate-UI definiert. In Abbildung 18 ist diese Deklaration ersichtlich.

```
12 #============================================================================
13 # Bezeichnung des Checks:
14 #============================================================================
15 DefClass: %ConstraintCheck003 (%ug_base_checker);
16
17 #============================================================================
18 #++
19
20 Beschreibung: Überprüfung auf Widersprüchliche Zwangsbedingungen.
21
22 #-
23 #============================================================================
24 # History:
25 #============================================================================
26 # - Entwickler David Stelter , 05.07.2020, Check erstellt
27 #
28 #============================================================================
29 #Kategorie und Checkname:
30 #============================================================================
31
32 (string)               %test_category:     "RWM.Test";
33 (string)           %displayed_name:    "ConstraintCheck003 - Widerspruechliche
     Zwangsbedingungen";
```

Abbildung 18.: Deklaration der Prüfung in der dfa-Datei

Nach der Namensdeklaration folgt der Aufruf der Prüfung in NX. Hier wird zunächst anhand der Fehlervariable aus Visual Studio entschieden, ob die Prüfung bestanden ist oder ob ein Fehler bzw. eine Warnung ausgegeben wird. Anschließend wird das Protokoll aus der Fehlervariable der dfa-Datei, der Message-Funktion und der Fehlerliste, jeweils aus Visual Studio, zusammengesetzt. In Abbildung 19 ist dieser Abschnitt einer dfa-Datei dargestellt.

```
36 #============================================================================
37 (Any Uncached)  do_check:
38 #============================================================================
39 @{
40
41         $LogType << IF(GetLog:) THEN LOG_OK ELSE LOG_ERROR;
42         ug_mqc_log($LogType,GetArray:,GetMessage:);
43
44
45 };
```

Abbildung 19.: Aufruf der Prüfung in der dfa-Datei

Im Anschluss folgt die Referenz auf die dll-Datei aus Visual Studio. Im Laufe der Prüfung werden hier die Ausgaben der einzelnen Funktionen in interne Variablen der dfa-Datei umgewandelt. In Abbildung 20 ist ein Beispiel dieser Referenz ersichtlich. Nach dem Verweis auf die dll-Datei folgt das aufzurufende Modul und die im Modul aufzurufende Funktion.

```
46 #=============================================================================
47 (Any Uncached)  GetArray:
48 #=============================================================================
49 @{
50
51     nx_ja_session_execute("[redacted]\Stelter\Checks\TestChecks\VS
         \Constraint_Check_Conflict\Constraint_Check_Conflict\bin\Debug
         \Constraint_Check_Conflict.dll","MainModule","ConstraintTag",{});
52
53
54 };
55 #=============================================================================
56 (Any Uncached)  GetLog:
57 #=============================================================================
58 @{
59
60     nx_ja_session_execute("[redacted]\Stelter\Checks\TestChecks\VS
         \Constraint_Check_Conflict\Constraint_Check_Conflict\bin\Debug
         \Constraint_Check_Conflict.dll","MainModule","GetLog",{});
61
62
63 };
64 (Any Uncached)  GetMessage:
65 #=============================================================================
66 @{
67
68     nx_ja_session_execute("[redacted]\Stelter\Checks\TestChecks\VS
         \Constraint_Check_Conflict\Constraint_Check_Conflict\bin\Debug
         \Constraint_Check_Conflict.dll","MainModule","GetMessage",{});
69
70
71 };
72 #=============================================================================
73 #=============================================================================
```

Abbildung 20.: Referenzen auf Visual Studio in der dfa-Datei

Um die dfa-Dateien als Prüfungen in NX nutzen zu können, ist es möglich, sie im gleichen Ordner wie die mitgelieferten Prüfungen abzuspeichern. Eine weitere Möglichkeit ist die Nutzung von Umgebungsvariablen. Check-Mate bietet zu diesem Zweck die UGCHECKMATE_USER_DIR an. Diese Variable wird in der ugii_env_ug.dat im Systemverzeichnis von NX definiert. Alternativ kann die Variable auch direkt als Windows-Umgebungsvariable definiert werden. In Abbildung 21 ist diese Variable in der ugii_env_ug.dat dargestellt. Diese Konfigurationen sind für einen lokalen Gebrauch der Prüfungen vorgesehen. Um die Prüfungen global (z.B. für einen ganzen Standort) bereitzustellen, können Startroutinen konfiguriert werden. Dabei liegen die Startroutine und die Prüfungen auf einem Server. In der Startroutine wird die UGCHECKMATE_USER_DIR auf den Pfad der Prüfungen auf dem Server gesetzt.

```
#
# UGCHECKMATE_USER_DIR points to the directory for multiple purposes:
#    - where customized checkers and profiles to load from recursively
#    - where Check-Mate authoring tool authored checkers and profiles to save to
#    - if UGCHECKMATE_USER_DIR/startup folder exists, UG will read from
#      this folder menu files, toolbar files and etc
#      The menu files and toolbar files generated via Check-Mate authoring tool
#      can be saved here.
# If it is used as the destination for receiving authored files and/or for saving log files,
# make sure the directory has write permission.
#
# UGCHECKMATE_USER_DIR=

#
# UGII_CHECKMATE_LOG_DIR points to the directory where the Check-Mate results (log files)
#    can be saved to.
#    This defaults to $UGCHECKMATE_USER_DIR
```

Abbildung 21.: Umgebungsvariable für Check-Mate

4.5. Erstellung von Prüf-Profilen

Prüf-Profile beinhalten eine Zusammenstellung von Prüfungen. Genau wie die Einzelprüfungen werden die Prüf-Profile über dfa-Dateien definiert. In der dfa-Datei wird der Aufruf über das Check-Mate-UI bestimmt. Außerdem referenziert sie auf die Einzelprüfungen, die über das Profil abgeprüft werden sollen. Der Aufbau der dfa-Datei ist in Abbildung 22 näher beschrieben. Am Anfang wird die Kategorie und der angezeigte Name des Prüf-Profils festgelegt. Die Einzelprüfungen werden anschließend als sogenannte Childs definiert.

```
#++
<Description>

    This profile examines RWM Datums Checkers

</Description>

#-

DefClass: %RWM_Profil_Datums ( %ug_base_checker_profile );

   ( String )    %test_category:       "RWM.Test";
   ( String )    %displayed_name:      "RWM Datums Profile";

#+
#    Do not allow the profile to be modified with NX interactively.
#-

    (Boolean)    allow_author_changes: false;

#+
# Checkers
#-
#=============================================================================
   ( Child ) %CheckAttribut:
               { Class;                %DatumsCheck001;

               };
#=============================================================================
   ( Child ) %CheckAttribut2:
               { Class;               %DatumsCheck002;

               };
```

Abbildung 22.: dfa-Datei eines Prüf-Profils

4.6. Einbindung der Prüfungen in einen Workflow

Die einzelnen Aufgaben innerhalb eines Workflows sind in Teamcenter durch Bausteine dargestellt. Diese Bausteine werden durch Pfeile verbunden. Das Einbinden der Prüfungen in einen Workflow erfolgt durch einen eigenen Unterworkflow. Dieser kann dann z.B. in einen Freigabeworkflow eingebunden werden. In Abbildung 23 wird der Unterworkflow gezeigt.

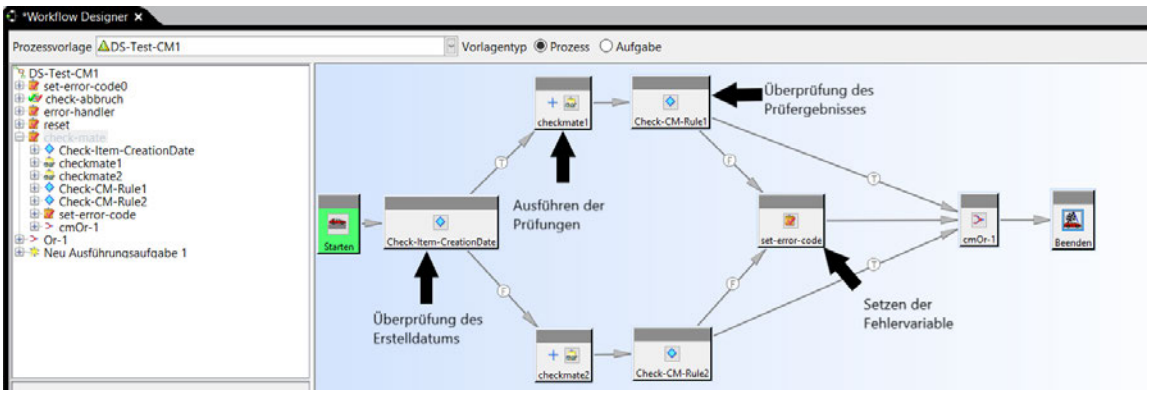

Abbildung 23.: Prüf-Workflow

Die Bereitstellung des Unterworkflows erfolgte durch einen RWM-Mitarbeiter. Teamcenter beinhaltet keine mitgelieferte Funktion, Check-Mate-Prüfungen innerhalb eines Workflows auszuführen. Daher läuft der Aufruf der Prüfungen über einen externen Dienst. Der Unterworkflow besteht aus vier Teilen. Im ersten Teil wird das Erstelldatum des zu prüfenden Teils untersucht. Hier wird zwischen Alt- und Neu-Teilen unterschieden. Der Stichtag hierfür ist der 01.07.2020. Anschließend folgt das Aufrufen der Prüfung in Check-Mate. Dieser Schritt ist in Abbildung 24 näher beschrieben. Im rot markierten Feld werden alle Prüfungen und Prüfprofile definiert, die ausgeführt werden sollen. Da die gesamte Item-Revision geprüft wird, muss nicht zwischen Modell-, Baugruppen- und Zeichnungsprüfungen unterschieden werden.

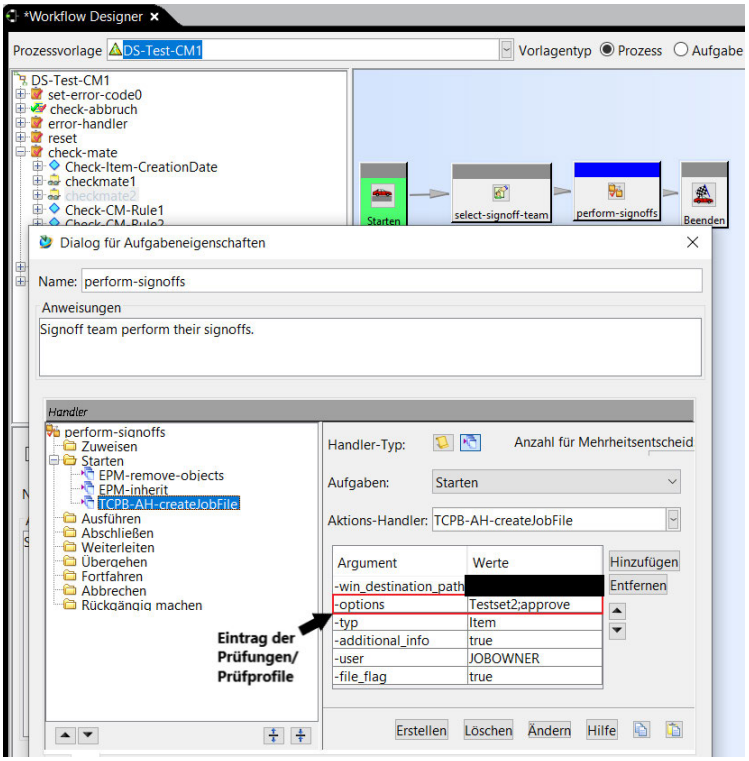

Abbildung 24.: Workflow-Aufruf Check-Mate

Danach erfolgt im dritten Teil die Auswertung der Prüfergebnisse. Dieser Schritt wird in Abbildung 25 gezeigt. Die markierte Funktion überprüft das Endergebnis der Prüfung. Ist die Prüfung bestanden, läuft der Workflow auf dem „True“ Pfad weiter. Ist die Prüfung nicht bestanden, läuft der Workflow auf dem „False“ Pfad weiter.

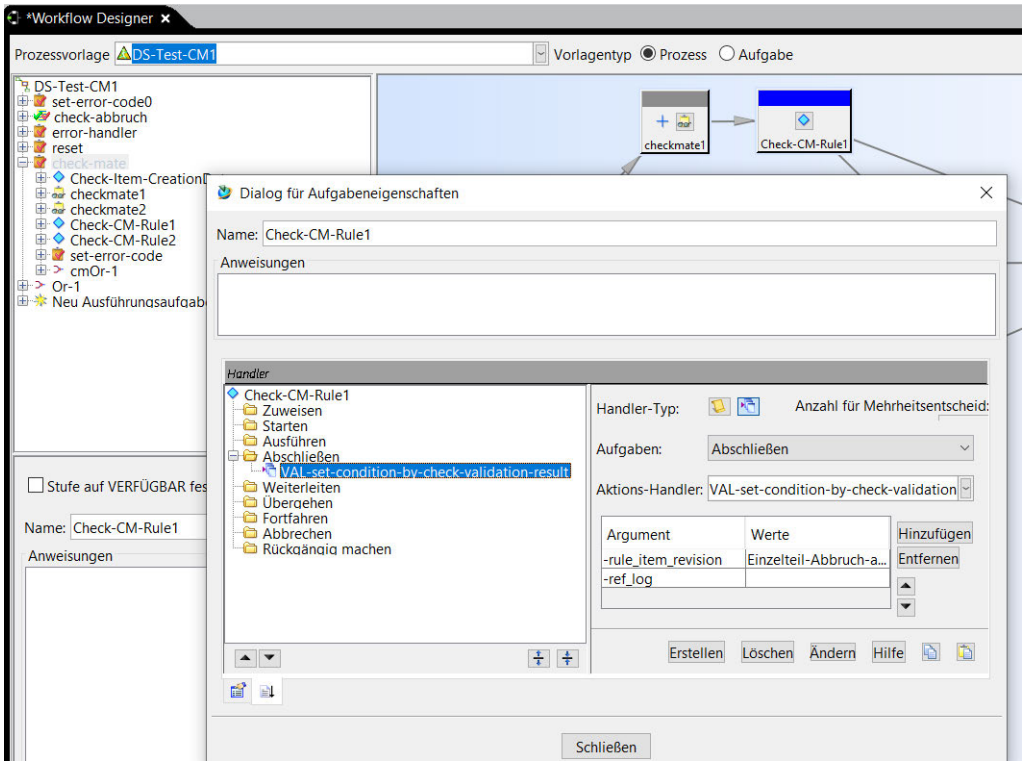

Abbildung 25.: Workflow-Auswertung des Prüfergebnisses

Ist eine Prüfung nicht bestanden, wird im vierten Teil der Fehlercode auf 2 gesetzt. Der Fehlercode kann auf eine beliebige Zahl außer 0 gesetzt werden. Anschließend wird der Unterworkflow abgeschlossen. Ist eine Prüfung bestanden, wird der Unterworkflow ohne Änderung des Fehlercodes abgeschlossen.

Um diesen Unterworkflow in einen bestehenden Workflow einzubinden, muss am Anfang die Fehlervariable auf 0 gesetzt werden. Nach dem Unterworkflow muss abgeprüft werden, ob die Fehlervariable im Unterworkflow verändert worden ist. Ist dies der Fall, wird der Workflow abgebrochen. Steht die Variable weiterhin auf 0, läuft der Workflow weiter. Ein Beispiel eines Freigabeworkflows ist in Abbildung 26 ersichtlich. Die zur Einbindung erforderlichen Bausteine sind rot markiert. Der Fehlerpfad von der Prüfung der Fehlervariable ist gestrichelt dargestellt.

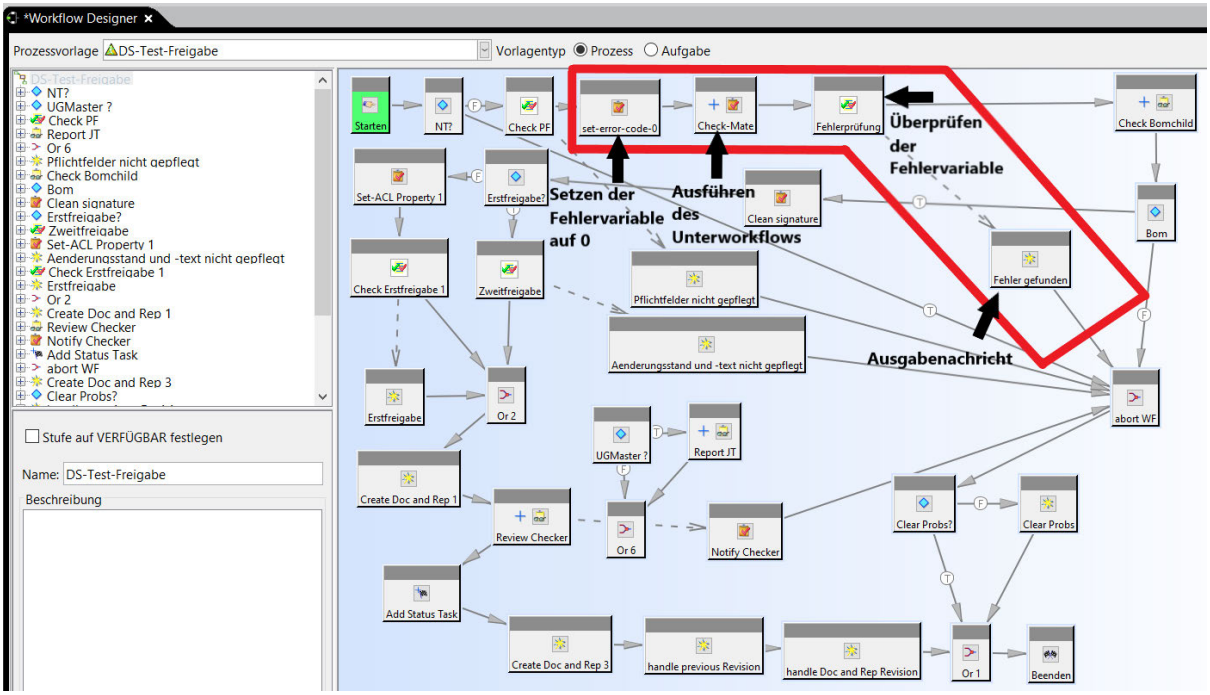

Abbildung 26.: Einbindung der Prüfungen in einen Workflow

4.7. Erstellung eines User-Interface

4.7.1. Erstellung der NX-Toolbar

Eine Toolbar ist ein nützliches Hilfsmittel, um dem Anwender Funktionen einfach und optisch ansprechend anzubieten. In NX werden die Dateien für die Prüfungs-Toolbar im NX-Verzeichnis unter \DESIGN_TOOLS\startup gespeichert. Die Einbindung der Toolbar kann außerdem, wie die Einbindung der Prüfungen, über Startroutinen erfolgen. Zunächst muss eine rtb-Datei erstellt werden. Diese Datei beinhaltet den Namen der Toolbar und die untergeordneten Gruppen von Funktionen. Der Aufbau der Datei ist in Abbildung 27 ersichtlich.

```
1  !
2  ! File Encoding: 'UTF-8'
3  !
4  ! Customer Ribbon Tab
5  !
6  TITLE RWM CheckMate NX 12
7
8  VERSION 170
9
10
11 GROUP    RWM_CheckMate_Sketch.grb
12 GROUP    RWM_CheckMate_3D.grb
13 GROUP    RWM_CheckMate_Datums.grb
14 GROUP    RWM_CheckMate_Constraints.grb
15 GROUP    RWM_CheckMate_Drafting.grb
16 GROUP    RWM_CheckMate_Full.grb
17
```

Abbildung 27.: Startup-Datei der Toolbar

Jede Gruppe von Funktionen ist in einer eigenen grb-Datei definiert. In der grb-Datei werden der Name der Gruppe und die einzelnen Buttons definiert. Jeder Button besteht aus dem Namen (BUTTON), der Beschreibung (LABEL), dem Icon (BITMAP), der auszuführenden Aktion (ACTION) und dem Stil des Buttons (RIBBON_STYLE). Die Aktion verweist auf die dfa-Datei der dazugehörigen Prüfung. Die Dateien für das Icon und die Aktion müssen nicht im NX-Verzeichnis abgespeichert sein. Im dazugehörigen Feld kann auf den jeweiligen Pfad verwiesen werden. In Abbildung 28 ist ein Auszug einer Toolbar-Gruppe dargestellt. Die fehlenden Buttons sind ähnlich der abgebildeten Buttons definiert.

```
1  !
2  ! File Encoding: 'UTF-8'
3  !
4  ! Customer Ribbon Group File
5  !
6
7  TITLE Baugruppe
8  VERSION 170
9  ! GROUP_STYLE DEFAULT
10 ! GROUP_STYLE FLOWLAYOUT
11
12
13   BUTTON        Constraints Profil
14   LABEL         Zwangsbedingungen prüfen
15   BITMAP        [redacted]\Stelter\Tools\bitmaps
       \Constraints_Profile.bmp
16   ACTION        [redacted]\Stelter\Checks\TestChecks\DFA
       \RWM_Profil_Constraints.dfa
17   RIBBON_STYLE  LARGE_IMAGE
18
19   BUTTON        fixed Constraints
20   LABEL         fixieren angewandt
21   BITMAP        [redacted]\Stelter\Tools\bitmaps
       \Constraints_fixed.bmp
22   ACTION        [redacted]\Stelter\Checks\TestChecks\DFA
       \ConstraintCheck001.dfa
23   RIBBON_STYLE  MEDIUM_IMAGE
24
25   BUTTON        suppressed Constraints
26   LABEL         unterdrückte Zwangsbedingungen
27   BITMAP        [redacted]\Stelter\Tools\bitmaps
       \Constraints_supp.bmp
28   ACTION        [redacted]\Stelter\Checks\TestChecks\DFA
       \ConstraintCheck002.dfa
29   RIBBON_STYLE  MEDIUM_IMAGE
```

Abbildung 28.: Beispiel einer Toolbar-Gruppe

Das Aussehen der Toolbar in NX wird in Abbildung 29 gezeigt.

Abbildung 29.: Check-Mate-Toolbar in NX

4.7.2. Erstellung der Icons

Die Icons der Toolbar werden mit Hilfe der Grafik-Software GIMP erstellt. Als Basis der Icons dienen die NX-eigenen Grafiken der Elemente, die überprüft werden. Die fertigen Icons werden dann im Bitmap-Format abgespeichert. Dieses Format bietet keine Transparenzeigenschaften, sodass alle transparenten Flächen des Icons in einem bestimmten „rosa“ Farbton eingefärbt werden müssen. Dieser Farbton wird dann NX-intern ausgeblendet. In Abbildung 30 wird ein Beispiel eines Icons in GIMP gezeigt.

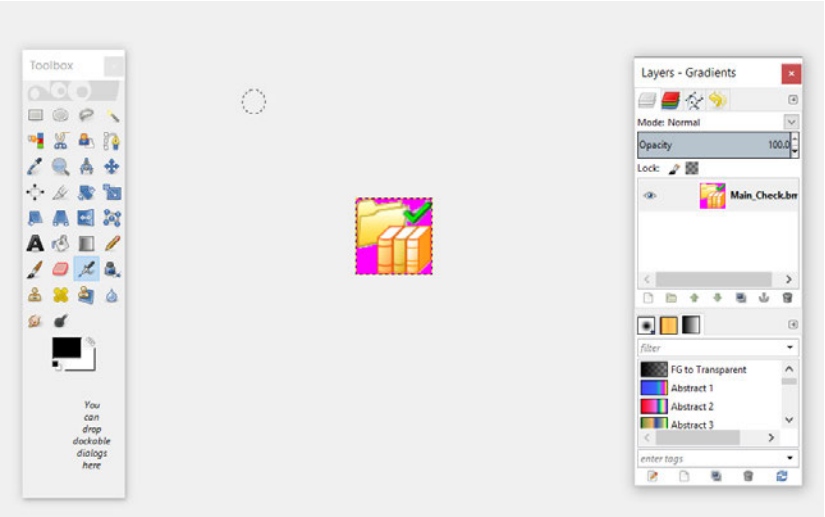

Abbildung 30.: Iconerstellung für die Toolbar

5. Test der firmeninternen Lösung

In diesem Kapitel wird das Testen der Prüfungen in NX erläutert. Außerdem wird die Auswertung und Verwaltung der Prüfergebnisse automatisiert innerhalb eines Workflows und manuell in Teamcenter beschrieben.

5.1. Test der Prüfungen in NX

Die manuellen Prüfungen in NX werden unabhängig von einem Workflow durchgeführt. Durch diese Art der Prüfung soll dem Anwender die Möglichkeit gegeben werden, die Qualität seiner Daten zu prüfen, ohne einen Workflow mit den dazugehörigen Aufgaben durchlaufen zu müssen. Als Testmodelle dienen Originalteile des Unternehmens. Diese Teile wurden von einem Mitarbeiter mit Fehlern versehen, um zu überprüfen, ob die Prüfungen alle eingebauten Fehler erfassen. Bevor eine Prüfung ausgeführt werden kann, muss ein Test eingerichtet werden. Dies erfolgt über einen Button im Check-Mate-UI. Das User-Interface ist in Abbildung 31 hervorgehoben.

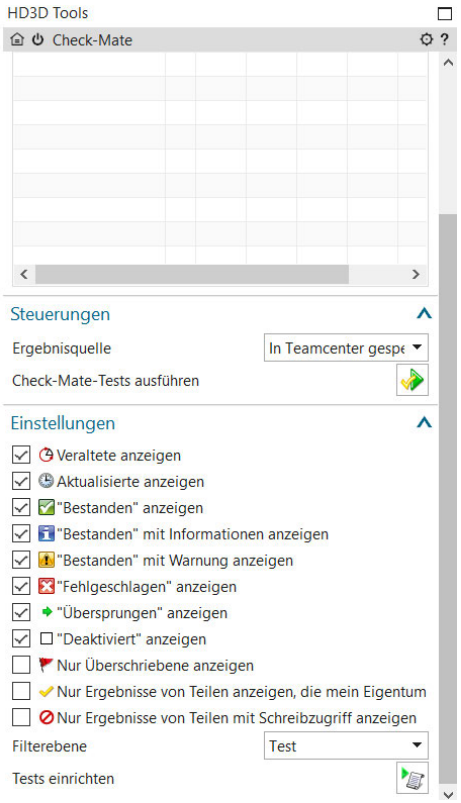

Abbildung 31.: User-Interface von Check-Mate

Im Einrichtefenster können nun die durchzuführenden Prüfungen und die zu prüfenden Teile ausgewählt werden. Die Prüfungen sind in Abbildung 32 ersichtlich. Prüf-Profile sind im Auswahlmenü blau gekennzeichnet.

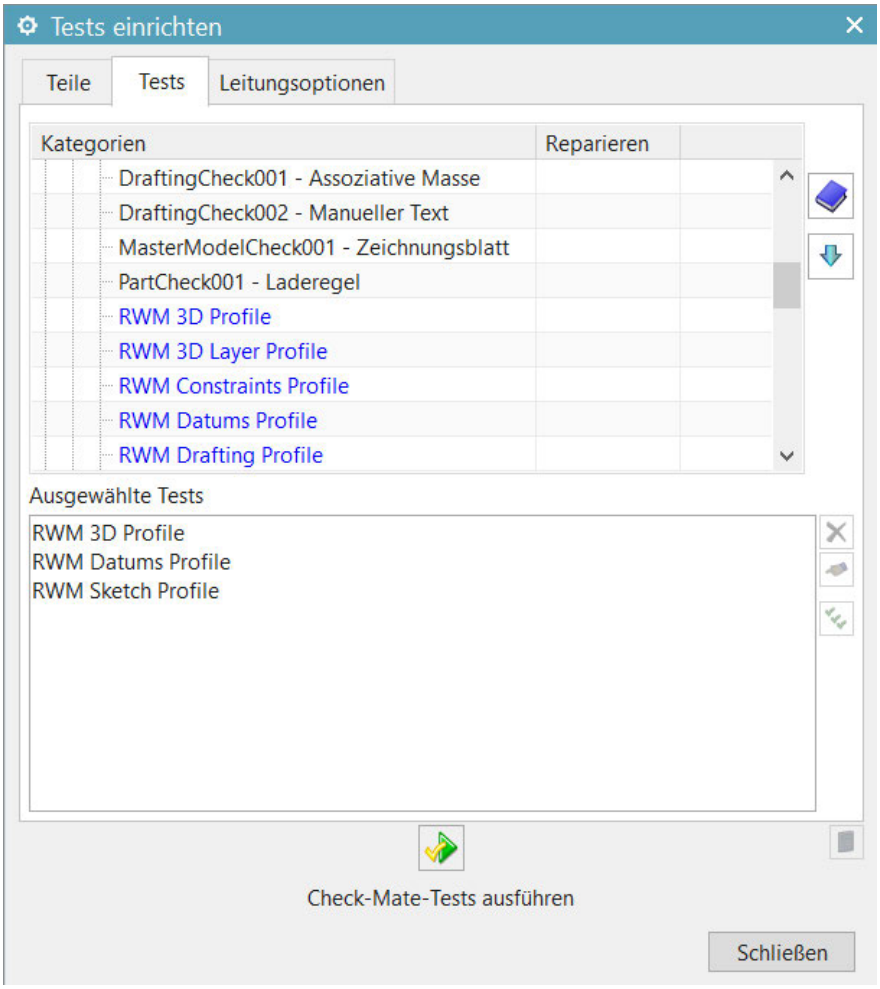

Abbildung 32.: Ausgewählte Prüfungen in Check-Mate

Des Weiteren können zusätzliche Einstellungen in den Leitungsoptionen eingerichtet werden. Hier können Haltepunkte definiert werden. Außerdem kann entschieden werden, ob das Teil nach der Prüfung gespeichert werden soll und ob ein Prüfprotokoll in Teamcenter abgespeichert werden soll. Wenn zusätzlich ein externes Protokoll gewünscht ist, kann hier auch der Speicherpfad angegeben werden. Weiterhin kann eingestellt werden, ob die Ergebnisse nach der Prüfung angezeigt werden sollen oder nicht. Das Fenster der Leitungsoptionen ist in Abbildung 33 dargestellt.

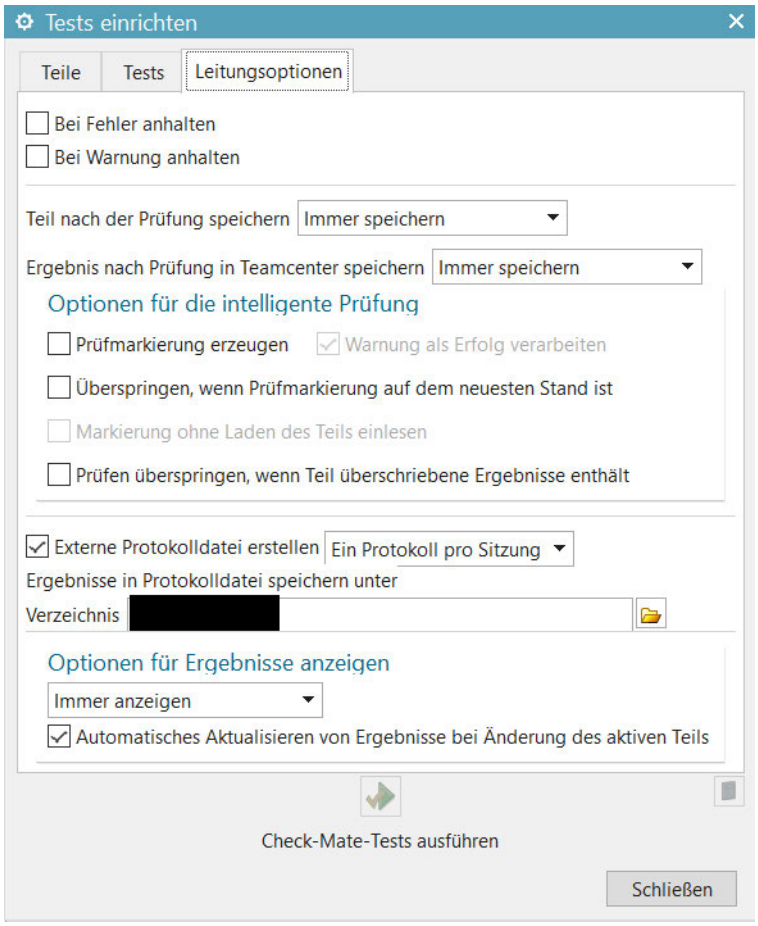

Abbildung 33.: Leitungsoptionen in Check-Mate

Die Prüfungen sind in drei Gruppen (Modell, Baugruppe, Zeichnung) aufgeteilt:

Modellprüfungen:
Die Modellprüfungen umfassen alle Prüfungen zu Skizzen, Bezugsobjekten und Körpern der Einzelteile. Das Ergebnis des Tests ist in Abbildung 34 abgebildet.

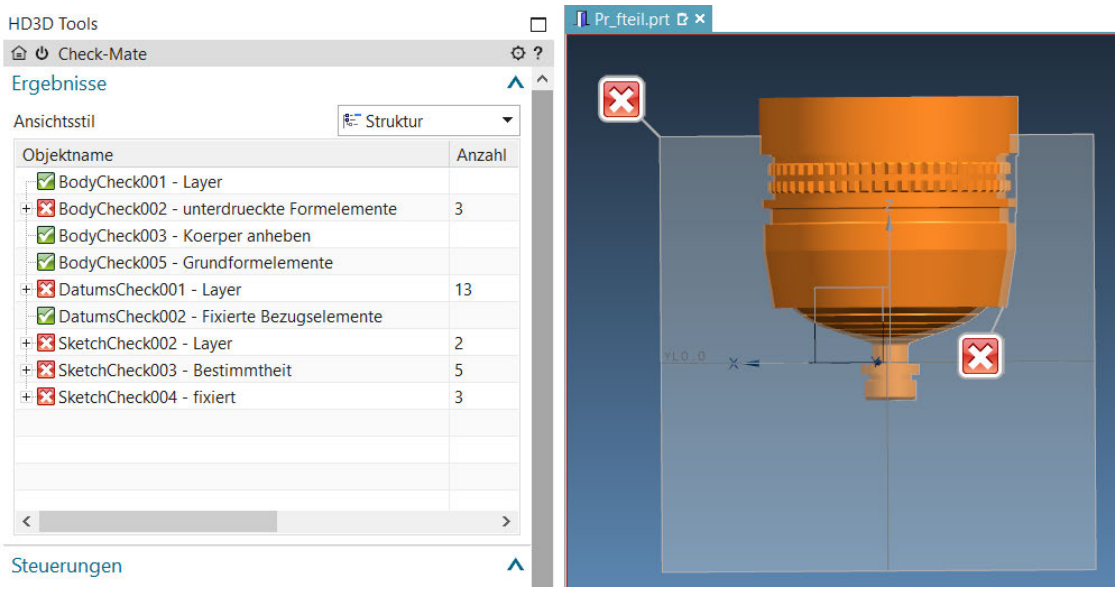

Abbildung 34.: Prüfung des 3D-Einzelteils

Der Test hat einige Fehler im 3D-Modell aufgezeigt. Es sind unterdrückte Formelemente vorhanden. Außerdem sind einige Bezugselemente und Skizzen den falschen Layern zugeordnet. Des Weiteren sind einige Skizzen nicht vollständig bestimmt oder besitzen fixierte Elemente.

Baugruppenprüfungen:

Die Baugruppenprüfungen beinhalten die Prüfungen der Zwangsbedingungen einer Baugruppe. In Abbildung 35 ist das Testergebnis der Baugruppenprüfung ersichtlich.

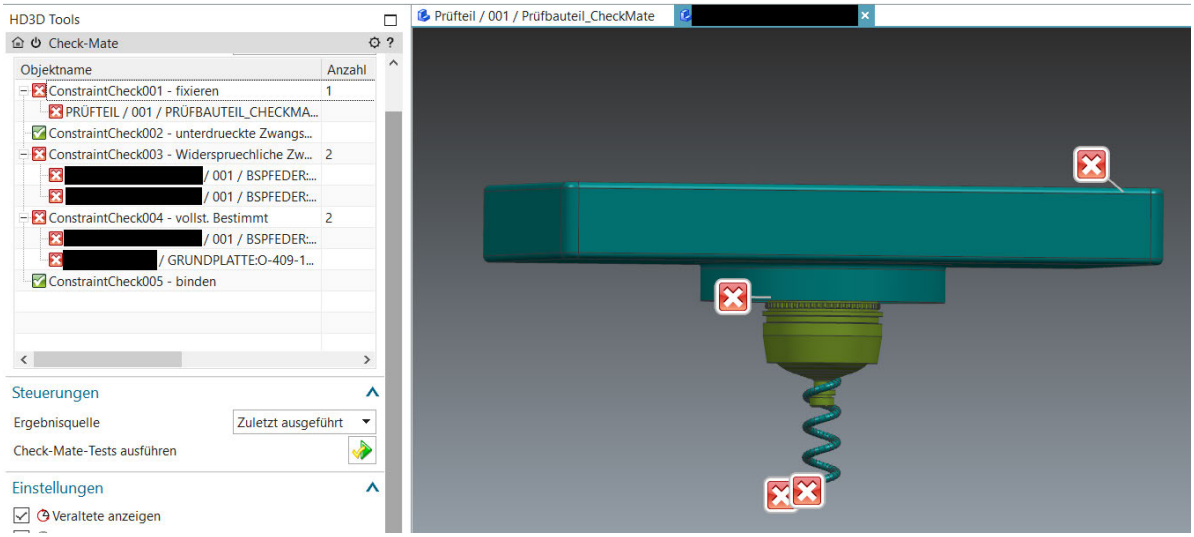

Abbildung 35.: Prüfung der Baugruppe

Die Baugruppe besitzt eine fixierte Komponente. Außerdem widersprechen sich zwei Zwangsbedingungen und zwei Komponenten sind nicht vollständig bestimmt.

Zeichnungsprüfungen:
Die Zeichnungsprüfungen beinhalten die Prüfung auf assoziative Maße und die Einhaltung des Master-Modell-Prinzips. Bei der Prüfung auf die Einhaltung des Master-Modell-Prinzips werden die Zeichnung und das 3D-Modell geprüft. Abbildung 36 zeigt das Ergebnis des Tests.

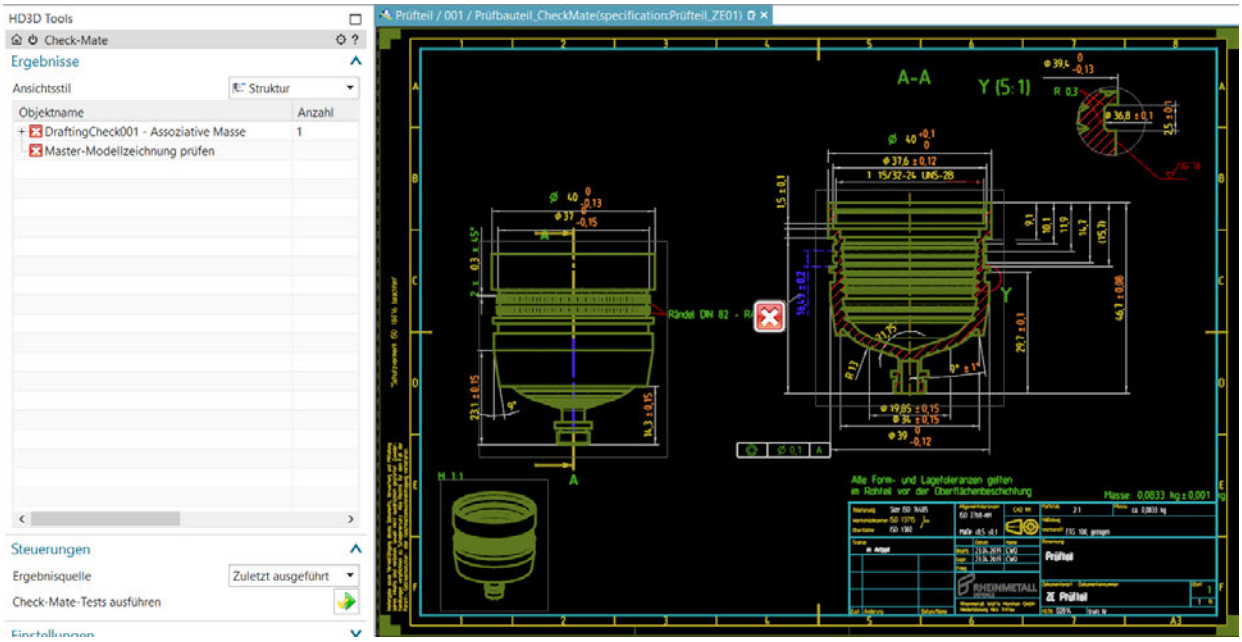

Abbildung 36.: Prüfung der Zeichnung

Die Zeichnung beinhaltet ein nicht an die Modellgeometrie assoziiertes Maß. Dieses ist durch ein Tag gekennzeichnet. Außerdem wurde das Master-Modell-Prinzip nicht eingehalten, da das 3D-Modell ein Zeichnungsblatt enthält. Dies geht aus dem detaillierten Prüfbericht in Abbildung 37 hervor.

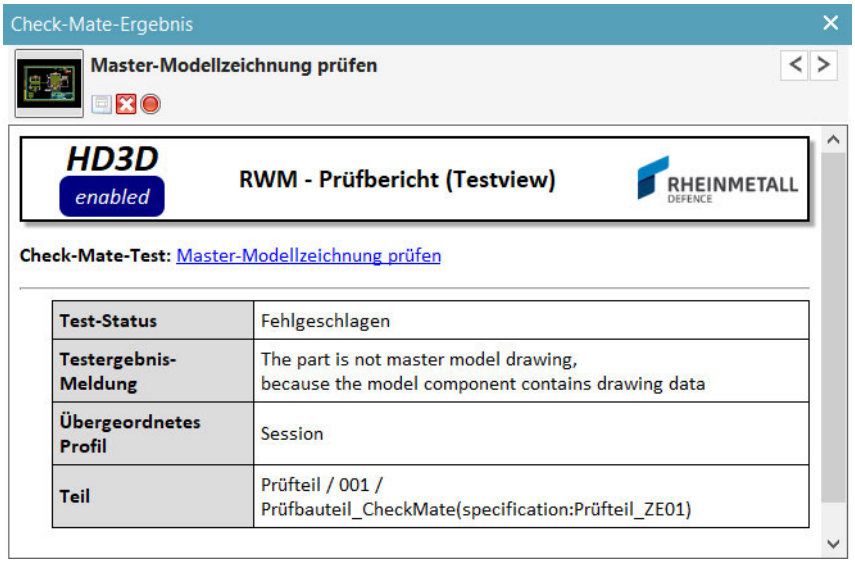

Abbildung 37.: Prüfbericht der Zeichnungsprüfung

5.2. Auswertung und Verwaltung der Prüfergebnisse

Laut Konfiguration werden die Prüfergebnisse in Teamcenter gespeichert. Innerhalb des Workflows wird das Endergebnis der Prüfung ausgewertet. Der Anwender muss zur Fehlererkennung den detaillierten Prüfbericht einsehen. Hier sind alle gefundenen Fehler nach Prüfungen und Prüf-Profilen gegliedert aufgelistet. Außerdem sind auf jeder Gliederungsebene die jeweiligen Prüfergebnisse abgebildet. Der Prüfbericht wird im XML-Format abgespeichert. An oberster Stelle steht der Benutzer, die genutzte Hardware und das Betriebssystem. Anschließend folgen Informationen über das geprüfte Teil. In Abbildung 38 sind diese beiden Punkte dargestellt.

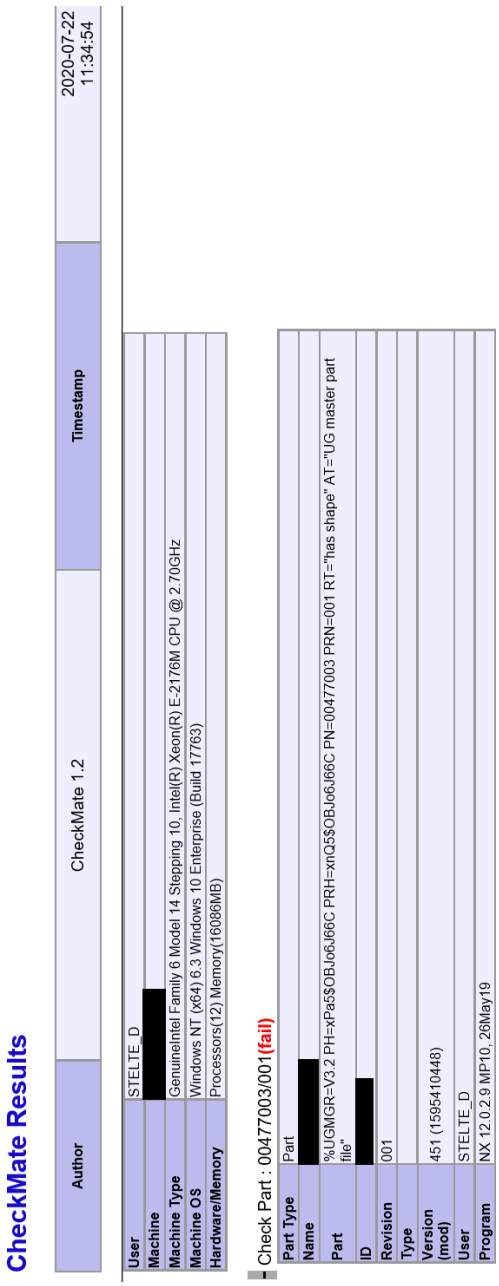

Abbildung 38.: Prüfbericht - Benutzer und Informationen

Unter dem geprüften Teil sind die Prüf-Profile angeordnet. Diese beinhalten die genutzten Einzelprüfungen. Die Gliederung der Profile und der Einzelprüfungen ist in Abbildung 39 ersichtlich.

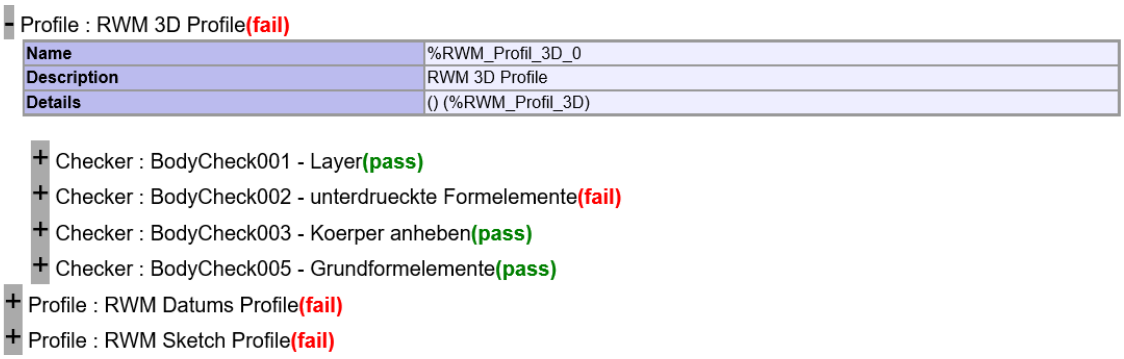

Abbildung 39.: Prüfbericht - Profile und Einzelprüfungen

Jede Einzelprüfung beinhaltet eine detaillierte Liste der ermittelten Fehler. Außerdem wird die Gesamtzahl der Fehler und die Fehlermeldung angezeigt. Das Prüfergebnis einer Einzelprüfung ist in Abbildung 40 dargestellt.

Checker : BodyCheck002 - unterdrueckte Formelemente(fail)

Name	root:%RWM_Profil_3D_0:%CheckAttribut2:
Description	BodyCheck002 - unterdrueckte Formelemente
Details	() (%BodyCheck002)
Version (mod)	451 (1595410448)
User	STELTE_D
Program	NX 12.0.2.9 MP10, 26May19

CheckResult

Status	fail
Total Entity Count	3
Total Entity Set Count	0

Message
Element unterdrückt

Detail Info:

Detail Info

Status	Name	ItemEntity	Points	Vectors	Message	ID	ItemLayer	Image
fail		M8 - Gewindeschnittstelle(23):C-23087-449				CM%UL=V1.0 PH=xPa5$OBJo6J66C C00005a2f000001c1	1	
fail		Zollgewinde zum Zuender(24):C-23120-449				CM%UL=V1.0 PH=xPa5$OBJo6J66C C00005a50000001c1	1	
fail		Fase(31):C-27862-449				CM%UL=V1.0 PH=xPa5$OBJo6J66C C00006cd6000001c1	1	

Abbildung 40.: Prüfbericht - Details einer Einzelprüfung

6. Enduser-Dokumentation

In diesem Kapitel wird das Aufrufen der Prüfungen in NX und das Starten eines Workflows in Teamcenter beschrieben.

6.1. Manuelle Prüfung in NX

Das Aufrufen der Einzelprüfungen oder der vorgefertigten Prüfprofile erfolgt über die Check-Mate-Toolbar in NX. Diese ist in Abbildung 41 dargestellt. Hier können per Mausklick die jeweiligen Einzelprüfungen/Prüfprofile ausgeführt werden.

Abbildung 41.: Check-Mate-Toolbar in NX

Das Erstellen eigener Prüfabläufe mit mehreren Einzelprüfungen und/oder Prüfprofilen erfolgt über das NX-eigene Check-Mate-UI. Dieses wird über die HD3D-Tools aufgerufen. In Abbildung 42 sind diese Schaltflächen dargestellt.

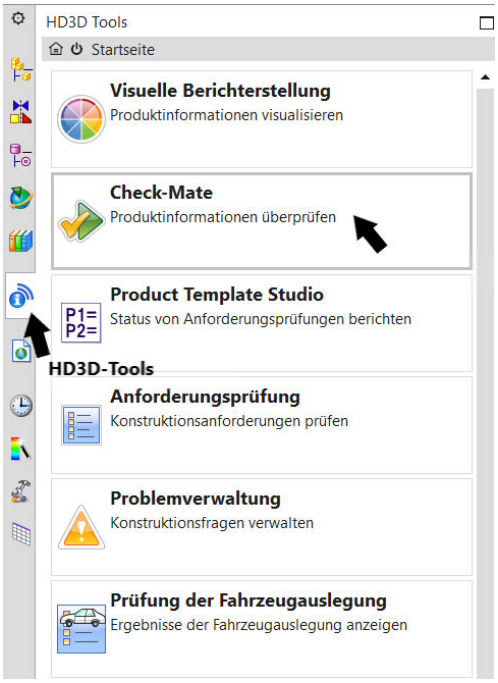

Abbildung 42.: Aufruf des Check-Mate-UI

Das nun angezeigte Fenster ist das Standard-Check-Mate-UI. Hier werden nach abgeschlossener Prüfung die Ergebnisse aufgelistet. Außerdem kann die Anzeige der Prüfergebnisse durch verschiedene Einstellmöglichkeiten angepasst werden. Beispiele zu den Prüfergebnisfenstern sind in Abbildung 34, Abbildung 35 und Abbildung 36 in Kapitel 5.1 dargestellt. Im unteren Bereich des Check-Mate-UI findet sich die Schaltfläche zum Einrichten von Prüfungen. Diese ist in Abbildung 43 markiert.

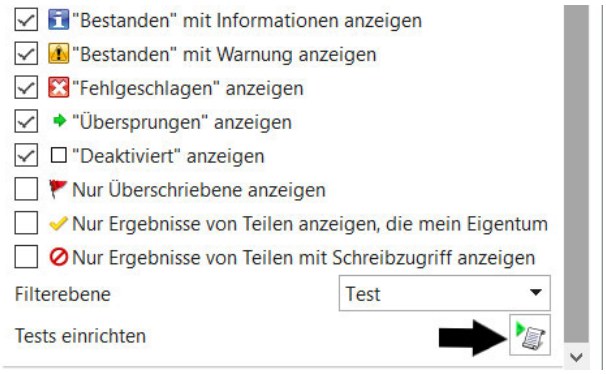

Abbildung 43.: Schaltfläche zum Einrichten von Prüfungen

Es öffnet sich das Einrichtefenster für Check-Mate. Hier können verschiedene mitgelieferte und selbst erstellte Prüfungen ausgewählt werden. Durch einen Klick auf die Schaltfläche „Check-Mate-Tests ausführen“ werden die ausgewählten Prüfungen nacheinander durchgeführt. Das Einrichtefenster ist in Abbildung 44 dargestellt.

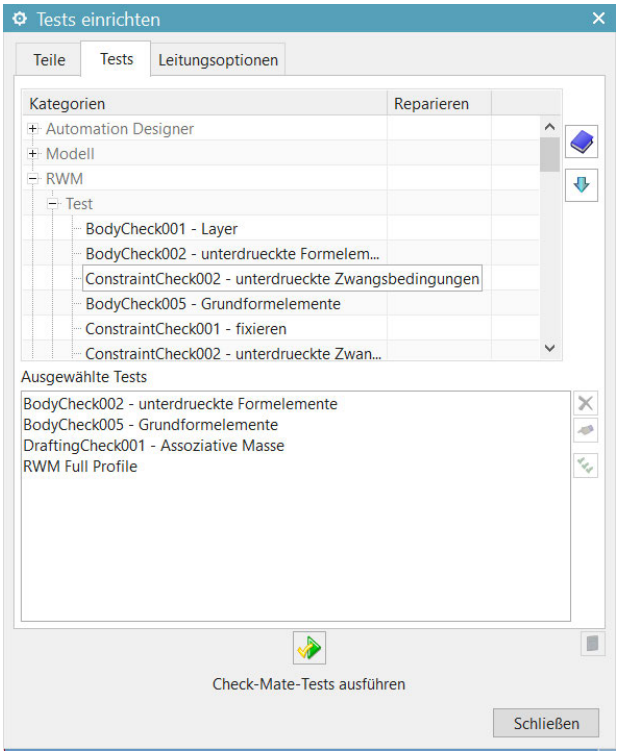

Abbildung 44.: Einrichtefenster für Check-Mate

Nach abgeschlossener Prüfung können die Ergebnisse im Check-Mate-UI eingesehen werden. Außerdem werden die Tags der fehlerhaften Elemente im NX-Grafikfenster angezeigt.
Zur Auswertung innerhalb eines Workflows und zur späteren Einsicht durch den Anwender wird der Prüfbericht in Teamcenter gespeichert. Dieser ist unter der jeweiligen Item-Revision zu finden. In Abbildung 45 ist der Prüfbericht in der Teamcenter-Umgebung markiert.

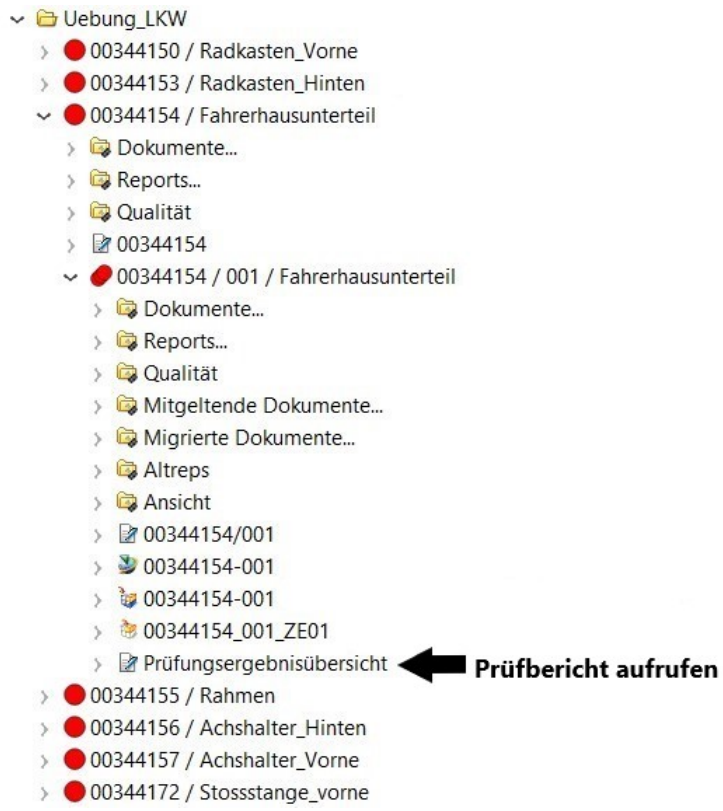

Abbildung 45.: Prüfbericht in Teamcenter

6.2. Starten eines Workflows

Um einen Workflow zu starten, muss die jeweilige Item Revision in Teamcenter ausgewählt werden. In der Übersicht kann nun unten rechts ein neuer Workflow-Prozess gestartet werden. In Abbildung 46 ist das Übersichtsfenster in Teamcenter dargestellt

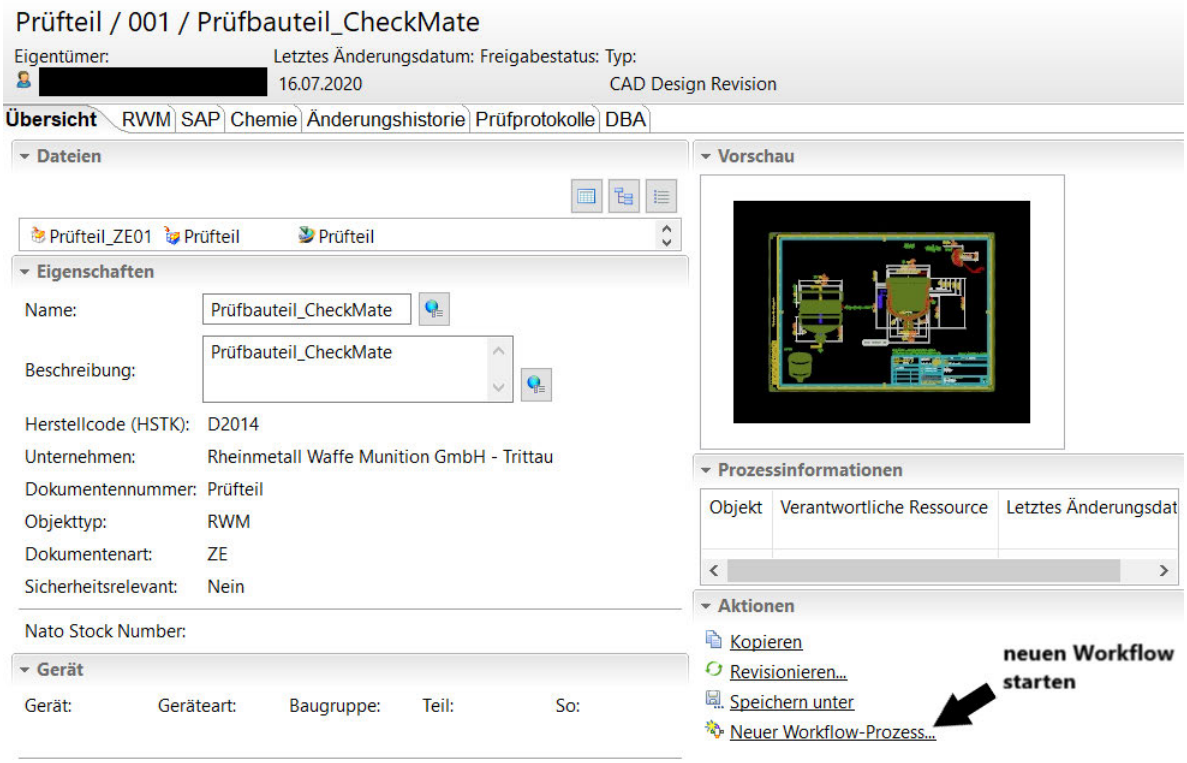

Abbildung 46.: Starten eines Workflows

Es öffnet sich das Einrichtefenster für den Workflow. Aus den Prozessvorlagen kann der gewünschte Workflow ausgesucht werden. Per Klick auf OK wird der Workflow gestartet. In Abbildung 47 wird der Dialog zum Auswählen und Start eines neuen Workflows gezeigt.

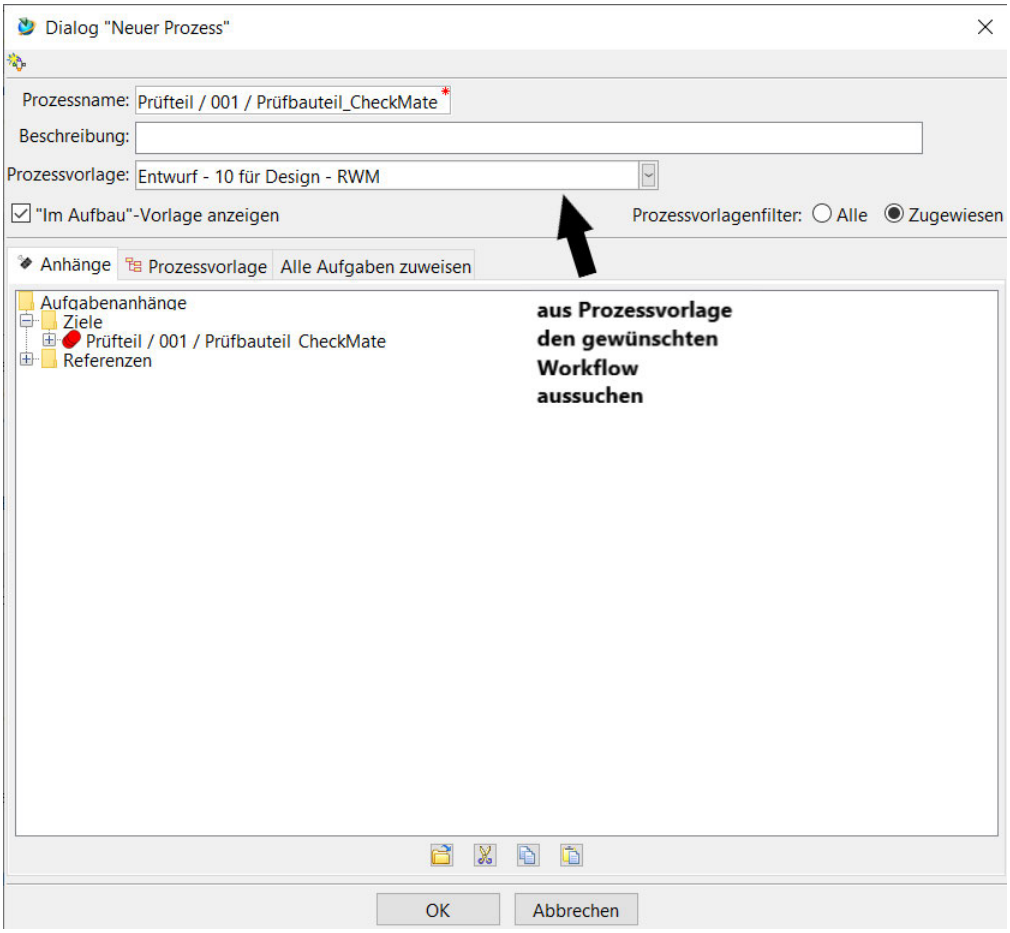

Abbildung 47.: Dialog "Neuer Prozess"

7. Zusammenfassung

Das im Laufe dieser Arbeit implementierte automatisierte Prüfverfahren bietet ein Tool, um langfristig die Qualität der CAD-Daten in den verschiedenen RWM-Standorten in Deutschland zu erhöhen. Kern des Tools ist eine Auswahl an mitgelieferten und selbst erstellten Prüfungen, die die Einhaltung definierter Punkte der Konstruktionsrichtlinie der RWM kontrollieren.

Aufgrund der Vorkenntnisse in der Programmiersprache Visual Basic, sowie der einfachen Handhabung der Entwicklungsumgebung in Microsoft Visual Studio, wurden die selbst erstellten Prüfungen nicht in Knowledge Fusion-Code innerhalb der dfa-Dateien geschrieben, sondern in die Microsoft-Software ausgelagert. Über die dfa-Dateien erfolgt ausschließlich die Einbindung der Visual Studio-Dateien in NX. Daher sind diese Dateien auch sehr ähnlich aufgebaut.

Die Prüfungen können manuell vom Anwender innerhalb der NX-Umgebung gestartet werden. Alternativ werden die Prüfungen automatisch im Laufe des Freigabeprozesses innerhalb eines Teamcenter-Workflows durchgeführt. Zum manuellen Starten der Prüfungen wurde eine Toolbar in NX erstellt. Diese beinhaltet die Einzelprüfungen und Prüf-Profile zu Modellen, Baugruppen und Zeichnungen.

Das automatisierte Starten der Prüfungen innerhalb eines Workflows erfolgt über einen externen Dienst, da Teamcenter keinen systemeigenen Dienst liefert. Die Prüfungen sind in die bestehenden Freigabe-Workflows eingebunden. Das Abprüfen der verschiedenen Teamcenter-Attribute der Punkte 4.1 bis 4.4 des Auszuges der Konstruktionsrichtlinie erfolgt, sofern nötig, im Workflow. Hier ist der Aufwand geringer, als sie in NX abzuprüfen.

Nach abgeschlossener Prüfung, manuell wie automatisch, wird ein Prüfbericht in Teamcenter gespeichert. Dieser kann vom Anwender eingesehen werden. Außerdem wird bei der automatischen Prüfung das Endergebnis des Berichtes innerhalb des Workflows ausgewertet. Bei nicht bestandener Prüfung wird der Workflow abgebrochen. Dabei werden alle Dokumente und Einstellungen, die innerhalb des Workflows geändert bzw. erstellt worden sind, zurückgesetzt und der Workflow wird beendet.

8. Handlungsempfehlungen

Das derzeit implementierte Tool bietet eine Grundlage für das manuelle und automatisierte Prüfen von Einzelteilen, Baugruppen und Zeichnungen nach der Konstruktionsrichtlinie. Einige Elemente dieses Tools zeigen aber noch Verbesserungspotential.

Das Erstellen jeder Prüfung in einer eigenen Projektmappe in Visual Studio ist speicherintensiv. Durch das Verknüpfen der NXOpen-dll-Dateien werden diese in den Projektordner geladen. Durch die Masse an Einzelprüfungen ergibt sich eine Anhäufung der gleichen Dateien. Durch eine Umstrukturierung der Prüfungen in eine Projektmappe mit den Prüfungen als Einzelfunktionen könnte Speicherplatz gespart werden, da nur ein einziges Mal auf die NXOpen-dll-Dateien verwiesen werden muss.

Als Programmiergrundlage der RWM im Umfeld von NX dient die Sprache C#. Um eine einheitliche Programmierstruktur zu bewahren, sollten die Prüfungen in C# umprogrammiert werden.

Die Anwender haben im Moment keine benutzerfreundliche Möglichkeit, mehrere Einzelprüfungen auszuwählen oder eigene Prüfprofile zu erstellen. Diese Funktionalitäten sollten in die Toolbar integriert werden. Eine Alternative wäre die Erstellung eines eigenständigen, fensterbasierten UI´s, in welchem das Erstellen, Starten und Auswerten der Prüfungen möglich ist.

Um die Verwaltung der Prüfungen zu erleichtern, ist die Einbindung eines Source-Code-Management-Systems, wie TortoiseSVN, sinnvoll.

Literatur

[1] Cadenas. *History of CAD*. URL: https://www.cadenas.de/files/cadenas/images/content/infographics/history_of_cad.png (besucht am 08. 06. 2020).

[2] Camtex. *CAD-MODELLE AUTOMATISIERT AUFBEREITEN MIT ASFALIS*. URL: https://www.camtex.de/software/asfalis (besucht am 08. 06. 2020).

[3] Camtex. *CADdoctor*. URL: https://www.camtex.de/software/caddoctor (besucht am 27. 07. 2020).

[4] in-crease. *PDM/PLM*. URL: https://www.in-crease.de/prozessberatung/pdm-plm/ (besucht am 02. 07. 2020).

[5] MTU. *CAD-Konvertierung: Motivation, Probleme und Lösungen*. URL: https://www.mtu.de/fileadmin/DE/6_Karriere/Berufserfahrene/Informationstechnologie/Reim_CAD_Konvertierung.pdf (besucht am 09. 06. 2020).

[6] Scan2CAD. *How CAD Has Evolved Since 1982*. URL: https://www.scan2cad.com/cad/cad-evolved-since-1982/ (besucht am 08. 06. 2020).

[7] Schulungsunterlagen Siemens NX, (Kein Datum), Unterlüß.

[8] Schulungsunterlagen Siemens Teamcenter, (Kein Datum), Unterlüß.

[9] Siemens. *NX Check-Mate*. URL: https://www.plm.automation.siemens.com/en_us/Images/2504_tcm1023-11882.pdf (besucht am 08. 06. 2020).

[10] Siemens. *NX Knowledge Fusion*. URL: https://www.plm.automation.siemens.com/de_de/Images/knowledge_fusion_tcm73-62405.pdf (besucht am 02. 07. 2020).

[11] Siemens. *Programming Tools. NX Open*. URL: https://docs.plm.automation.siemens.com/tdoc/nx/10/nx_api#uid:index (besucht am 02. 07. 2020).

A. Anhang: Code der Modellprüfung

...Checks\VS\Sketch_Check_Layer\Sketch_Check_Layer\Class1.vb 1

```
1  Imports NXOpen
2
3  'Dieser Check prüft, ob die Skizzen auf den richtigen Layern liegen.
4
5  Public Module MainModule
6
7      'Variablendeklaration
8
9      Private SketchLayer As Boolean = True
10     Private UnloadOption As Session.LibraryUnloadOption =
         Session.LibraryUnloadOption.Immediately
11
12     Public Function SketchTag() As Tag()
13
14         'Allgemeine Konfiguration
15
16         UnloadOption = Session.LibraryUnloadOption.Explicitly
17
18         Dim theSession As Session = Session.GetSession
19         Dim theUI As UI = UI.GetUI
20         Dim workPart As Part = theSession.Parts.Work
21
22         'Erstellen leerer Fehlerliste
23         Dim SketchList As New List(Of Tag)
24
25         'Überprüfen jeder Skizze auf Fehler
26
27         For Each SketchItem As Sketch In workPart.Sketches.ToArray
28
29             'Überprüfen des richtigen Layers
30
31             If SketchItem.Layer < 20 Or SketchItem.Layer > 39 Then
32                 SketchLayer = False
33                 SketchList.Add(SketchItem.Tag)
34
35             End If
36
37         Next
38
39         'Rückgabe der Fehlerliste
40
41         Return SketchList.ToArray
42
43     End Function
44
45     Public Function GetMessage() As String
46
47         UnloadOption = Session.LibraryUnloadOption.Immediately
48
```

```
49          'Ausgabenachricht
50
51          If SketchLayer Then
52              Return "bestanden"
53
54          Else
55              Return "Skizze auf falschem Layer"
56          End If
57
58      End Function
59
60      Public Function GetLog() As Boolean
61
62          Call SketchTag()
63          UnloadOption = Session.LibraryUnloadOption.Explicitly
64
65          'Entscheider, ob Fehler gefunden wurden
66
67          If SketchLayer Then
68              Return True
69
70          Else
71              Return False
72
73          End If
74
75      End Function
76
77      'Entladen der .dll aus NX
78
79      Public Function GetUnloadOption() As Integer
80
81          GetUnloadOption = UnloadOption
82
83      End Function
84
85  End Module
86
```

B. Anhang: Code der Baugruppenprüfung

...S\Constraint_Check_Fully\Constraint_Check_Fully\Class1.vb 1

```
1  Imports NXOpen
2
3  'Dieser Check überprüft, ob die einzelnen Elemente im Modell vollständig
     bestimmt sind.
4
5  Public Module MainModule
6
7      'Variablendeklaration
8
9      Private Constraintfully As Boolean = True
10     Private UnloadOption As Session.LibraryUnloadOption =
         Session.LibraryUnloadOption.Immediately
11
12     Public Function ConstraintTag() As Tag()
13
14         'Allgemeine Konfiguration
15
16         UnloadOption = Session.LibraryUnloadOption.Explicitly
17
18         Dim theSession As Session = Session.GetSession
19         Dim theUI As UI = UI.GetUI
20         Dim workPart As Part = theSession.Parts.Work
21
22         'Erstellen leerer Fehlerliste
23
24         Dim ConstraintList As New List(Of Tag)
25
26         'Überprüfen von jedem Element
27
28         For Each GroupItem In workPart.ComponentGroups.ToArray
29
30             For Each ComponentItem As Assemblies.Component In
                 GroupItem.GetComponents.ToArray
31
32                 'Überprüfen, ob Komponente vollständig bestimmt ist
33
34                 If Not ComponentItem.GetDegreesOfFreedom.NumRotational = 0 Or
                     Not ComponentItem.GetDegreesOfFreedom.NumTranslational = 0
                     Then
35                     Constraintfully = False
36                     ConstraintList.Add(ComponentItem.Tag)
37                 End If
38
39             Next
40
41         Next
42
43         Return ConstraintList.ToArray
44
```

```
45      End Function
46
47      Public Function GetMessage() As String
48
49          UnloadOption = Session.LibraryUnloadOption.Immediately
50
51          'Ausgabenachricht
52
53          If Constraintfully Then
54              Return "bestanden"
55
56          Else
57              Return "Komponente nicht vollständig bestimmt"
58          End If
59
60      End Function
61
62      Public Function GetLog() As Boolean
63
64          Call ConstraintTag()
65          UnloadOption = Session.LibraryUnloadOption.Explicitly
66
67          'Entscheider, ob Fehler gefunden wurden
68
69          If Constraintfully Then
70              Return True
71
72          Else
73              Return False
74
75          End If
76
77      End Function
78
79      'Entladen der .dll aus NX
80
81      Public Function GetUnloadOption() As Integer
82
83          GetUnloadOption = UnloadOption
84
85      End Function
86
87  End Module
88
```

C. Anhang: Code der Zeichnungsprüfung

...mensions\DraftingCheck001_AssociativeDimensions\Class1.vb 1

```
1  Imports NXOpen
2
3  'Dieser Check überprüft, ob nicht assoziative Maße in der Zeichnung vorhanden
     sind.
4
5  Public Module MainModule
6
7      'Variablendeklaration
8
9      Private UnloadOption As Session.LibraryUnloadOption =
         Session.LibraryUnloadOption.Immediately
10     Private Fehler As Boolean = False
11
12     Public Function AssociativeDimensions() As Tag()
13
14         'Allgemeine Konfiguration
15
16         UnloadOption = Session.LibraryUnloadOption.Explicitly
17
18         Dim theSession As Session = Session.GetSession
19         Dim theUfSession As UF.UFSession = UF.UFSession.GetUFSession
20         Dim theUI As UI = UI.GetUI
21         Dim workPart As Part = theSession.Parts.Work
22
23         'Erstellen leerer Liste fehlerhafter Bemaßungen
24
25         Dim DimensionObjects As New List(Of Tag)
26
27         'Überprüfen jeder Bemaßung auf Assoziativität
28
29         For Each DraftingDimension As Annotations.Dimension In
             workPart.Dimensions
30
31             If DraftingDimension.IsRetained = True Then
32                 DimensionObjects.Add(DraftingDimension.Tag)
33                 Fehler = True
34             End If
35
36         Next
37
38         'Rückgabe Fehlerliste
39
40         Return DimensionObjects.ToArray
41
42     End Function
43
44     Public Function GetMessage() As String
45
46         UnloadOption = Session.LibraryUnloadOption.Immediately
```

```
47
48          'Ausgabenachricht bei Fehler
49
50          Return "Maße nicht assoziativ"
51
52      End Function
53
54      Public Function GetLog() As Boolean
55
56          Call AssociativeDimensions()
57
58          UnloadOption = Session.LibraryUnloadOption.Explicitly
59
60          'Entscheider, ob Fehler gefunden wurden
61
62          If Fehler = True Then
63              Return False
64
65          Else
66              Return True
67
68          End If
69
70      End Function
71
72      'Entladen der .dll aus NX
73
74      Public Function GetUnloadOption() As Integer
75
76          GetUnloadOption = UnloadOption
77
78      End Function
79
80  End Module
81
82
```

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: Stelter

Vorname: David

dass ich die vorliegende Bachelorarbeit bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Entwicklung und Implementierung eines
automatisierten Prüfverfahrens für CAD-Modelle
und Zeichnungen

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der -bitte auswählen- ist erfolgt durch:

Ort	Datum	Unterschrift im Original
		[redacted]