TECHNISCHE UNIVERSITÄT WIEN

Institut für Fertigungstechnik und Photonische Technologien

Masterarbeit

Erstellung eines Digitalen Schattens für eine flexible Fertigungszelle

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs (Dipl.-Ing. oder DI) unter der Leitung von

Assistant Prof. Dipl.-Ing. Dr.techn. Thomas Fabian Trautner

(Institut für Fertigungstechnik und Photonische Technologien)

Univ.Ass. Dipl.-Ing. Bernhard Johann Wallner

(Institut für Fertigungstechnik und Photonische Technologien)

eingereicht an der Technischen Universität Wien

Fakultät für Maschinenwesen und Betriebswissenschaften

von

Benedikt ZWÖLFER

Wien, im Februar 2025

Benedikt Zwölfer

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

Masterarbeit

nur mit Bewilligung der Prüfungskommission berechtigt bin.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass die vorliegende Arbeit nach den anerkannten Grundsätzen für wissenschaftliche Abhandlungen von mir selbstständig erstellt wurde. Alle verwendeten Hilfsmittel, insbesondere die zugrunde gelegte Literatur, sind in dieser Arbeit genannt und aufgelistet. Die aus den Quellen wörtlich entnommenen Stellen, sind als solche kenntlich gemacht.

Das Thema dieser Arbeit wurde von mir bisher weder im In- noch Ausland einer Beurteilerin/einem Beurteiler zur Begutachtung in irgendeiner Form als Prüfungsarbeit vorgelegt. Diese Arbeit stimmt mit der von den Begutachterinnen/Begutachtern beurteilten Arbeit überein.

Ich nehme zur Kenntnis, dass die vorgelegte Arbeit mit geeigneten und dem derzeitigen Stand der Technik entsprechenden Mitteln (Plagiat-Erkennungssoftware) elektronisch-technisch überprüft wird. Dies stellt einerseits sicher, dass bei der Erstellung der vorgelegten Arbeit die hohen Qualitätsvorgaben im Rahmen der geltenden Regeln zur Sicherung guter wissenschaftlicher Praxis „Code of Conduct“ an der TU Wien eingehalten wurden. Zum anderen werden durch einen Abgleich mit anderen studentischen Abschlussarbeiten Verletzungen meines persönlichen Urheberrechts vermieden.

Wien, im Februar 2025

Benedikt Zwölfer

Kurzfassung

Die Fertigungsindustrie steht vor der Herausforderung, ihre Systeme flexibler und effizienter zu gestalten, um schnell auf veränderte Produktionsbedingungen reagieren zu können. Ein vielversprechender Ansatz ist die Entwicklung eines Digitalen Schattens – einer Digitalen Abbildung eines realen Systems, die es ermöglicht, Prozesse in Echtzeit zu simulieren, zu überwachen und zu optimieren. Diese Arbeit untersucht die Konzeption und Implementierung eines solchen Digitalen Schattens für eine flexible Fertigungszelle und zeigt die Vorteile dieser Technologie für die Produktionsoptimierung auf.

Im Zentrum der Untersuchung steht die Abbildung physikalischer Eigenschaften der Fertigungszelle in einem Simulationsprogramm, das automatische Anpassungen bei Änderungen im realen System ermöglicht. Ein besonderer Fokus liegt auf der effizienten Datenübertragung in Echtzeit, um eine realitätsnahe Simulation zu gewährleisten. Durch die Integration gängiger Schnittstellen und die Entwicklung eines intuitiven Benutzerinterfaces wird der Digitale Schatten zu einem vielseitigen Werkzeug für Analyse und Steuerung. Zudem bietet die Speicherung und spätere Wiedergabe von Zeiträumen in der Simulation die Möglichkeit zur detaillierten Prozessanalyse.

Die Arbeit betont den Mehrwert des Digitalen Schattens für die flexible Fertigung: von der schnellen Aktualisierung physikalischer Änderungen über die Implementierung neuer Systemkomponenten bis hin zur Visualisierung der Aktivität im dreidimensionalen Raum. Eine kritische Auseinandersetzung mit der Datenübertragung, Genauigkeit und Stabilität der Simulation hebt sowohl die Herausforderungen als auch das Optimierungspotenzial dieser Technologie hervor.

Die Ergebnisse zeigen, dass die Implementierung eines Digitalen Schattens eine effektive Methode zur Steigerung der Effizienz in Fertigungsprozessen darstellt und einen wertvollen Beitrag zur digitalen Transformation in der Industrie leistet. Der Ausblick auf zukünftige Arbeiten weist auf Möglichkeiten zur Weiterentwicklung in Richtung Digitaler Zwilling, die Anwendung in weiteren Fertigungszellen sowie die Optimierung von Genauigkeit, Portabilität und zentraler Datenverarbeitung hin.

Abstract

The manufacturing industry faces the challenge of making systems more flexible and efficient to respond quickly to changing production conditions. A promising approach is the development of a Digital Shadow—a digital representation of a real system that enables real-time simulation, monitoring, and optimization of processes. This thesis explores the design and implementation of such a Digital Shadow for a flexible manufacturing cell and highlights the benefits of this technology for production optimization.

The focus of this study is on simulating the physical properties of the manufacturing cell in a simulation program, which allows for automatic adjustments when changes occur in the real system. Particular attention is given to efficient real-time data transfer to ensure accurate simulation. By integrating standard interfaces and developing an intuitive user interface, the Digital Shadow becomes a versatile tool for analysis and control. Additionally, the ability to record and later replay time periods in the simulation offers the potential for detailed process analysis.

The thesis emphasizes the added value of the Digital Shadow for flexible manufacturing: from the rapid updating of physical changes and the implementation of new system components to the visualization of activity in three-dimensional space. A critical examination of data transfer, accuracy, and stability of the simulation highlights both the challenges and optimization potential of this technology.

The results demonstrate that implementing a Digital Shadow is an effective way to enhance efficiency in manufacturing processes and makes a valuable contribution to the digital transformation of the industry. The outlook for future work points to opportunities for further development toward a Digital Twin, application in additional manufacturing cells, and the optimization of accuracy, portability, and centralized data processing.

Abkürzungsverzeichnis

DDS	Data Distribution Service for Real-Time Systems
IoT	Internet of Things
JSON	JavaScript Object Notation
MCD	Mechatronic Concept Designer
MQTT	Message Queuing Telemetry Transport
OPC UA	OPC Unified Architecture
RDF	Resource Description Framework
REST	Representational State Transfer
XML	Extensible Markup Language

Inhaltsverzeichnis

1. Einleitung

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung und Implementierung eines Digitalen Schattens für eine flexible Fertigungszelle. Ziel ist es, das reale System in einer Simulationsumgebung abzubilden, um physikalische Eigenschaften möglichst genau zu simulieren. Dies ermöglicht eine detaillierte Analyse von Produktionsprozessen sowie eine Optimierung der Abläufe.

Im Rahmen dieser Arbeit wurde ein wissenschaftliches Paper veröffentlicht, das sich speziell mit der Integration eines Digitalen Schattens befasst und die entwickelten Methoden detaillierter beleuchtet [1]. Diese Veröffentlichung stellt eine weiterführende Betrachtung der hier beschriebenen Konzepte dar und hebt die praktischen Herausforderungen und Lösungen hervor.

Die Struktur dieser Arbeit orientiert sich am V-Modell, das als bewährte Vorgehensweise für die Entwicklung komplexer Systeme dient. Dieses Modell beschreibt eine strukturierte und phasenorientierte Methodik von der Anforderungsanalyse über die Implementierung bis hin zur Integration des entwickelten Systems.

1.1 Motivation

Die zunehmende Komplexität der Fertigungssysteme und die rapide Weiterentwicklung der digitalen Technologien stellen die Industrie vor bedeutende Herausforderungen. Wie in der Arbeit von Wallner et al. [1] diskutiert, erfordert die effiziente Verwaltung dieser Systeme eine hohe Flexibilität und Anpassungsfähigkeit, um auf Veränderungen schnell reagieren zu können und gleichzeitig Ausfallzeiten zu minimieren. Zudem müssen Sicherheit und Zuverlässigkeit in den Fertigungsprozessen jederzeit gewährleistet sein. Ein spezifisches Problem stellt die frühzeitige Erkennung und Behebung von Logikfehlern und Fehlkonfigurationen dar, um kostspielige Ausfälle und Schäden zu verhindern. Die Nutzung digitaler Technologien wie Digitaler Zwilling und Digitaler Schatten zur Unterstützung dieser Systeme zeigt, dass bestehende Ansätze oft nicht ausreichen, um diese Herausforderungen zu bewältigen [2–4].

Darüber hinaus erfordern diese Technologien häufig einen erheblichen manuellen und personellen Aufwand für die kontinuierliche Aktualisierung und Anpassung an veränderte Produktionsbedingungen. Dies umfasst die regelmäßige Wartung, das Nachziehen von Änderungen und die Integration neuer Maschinen in die Fertigungszelle. Dieser hohe Aufwand führt nicht nur zu höheren Betriebskosten, sondern erhöht auch das Risiko menschlicher Fehler und Verzögerungen im Produktionsprozess [2].

Vor diesem Hintergrund erweisen sich die bestehenden Ansätze zur Integration digitaler Technologien als unzureichend. Die gegenwärtigen Lösungen bieten oft keine ausreichende Flexibilität oder Skalierbarkeit, um sich dynamisch an veränderte Fertigungsbedingungen anzupassen oder individuelle Anforderungen der Benutzer und Benutzerinnen, wie die Integration benutzerdefinierter Visualisierungselemente, effektiv zu unterstützen [3, 4].

1.2 Aufgabenstellung

Das Ziel dieser Arbeit ist die Entwicklung und Implementierung eines Digitalen Schattens für eine flexible Fertigungszelle. Das reale System soll in einem Simulationspro-

gramm abgebildet werden, sodass physikalische Eigenschaften wahrheitsgetreu simuliert werden können. Der Aspekt *Flexibilität* zielt in diesem Kontext auf die Möglichkeit, physikalische Änderungen im realen System mit möglichst wenig manueller Interaktion im Digitalen Schatten zu aktualisieren. Die Simulation muss in der Lage sein, Kollisionen zu erkennen, und es wird besonderer Wert auf flüssige Bewegungsabläufe gelegt. Der Digitale Schatten soll das reale System einerseits zeitnah abbilden (d. h. mit einer möglichst geringen Verzögerung zwischen realer und digitaler Darstellung), andererseits soll es möglich sein, einen Zeitraum aufzuzeichnen, zu speichern und später abzuspielen. Der Schwerpunkt dieser Arbeit liegt auf der Datenübertragung zwischen realem System und Simulation, der physikalischen Simulation, der Integration eines flexiblen Systems sowie dem Laden und Speichern von Daten.

2. Grundlagen und Stand der Technik

Dieses Kapitel widmet sich der Erläuterung und Diskussion verwendeter Begriffe, wobei ein besonderer Fokus auf dem Konzept des Digitalen Schattens liegt. Es beinhaltet eine Erörterung thematisch relevanter Normen und eine Untersuchung der Arbeiten und Ansätze anderer Fachleute und Wissenschaftler in diesem Bereich. Des Weiteren wird ein Überblick über mögliche Softwarelösungen geboten, die in diesem Kontext von Bedeutung sind.

2.1 Physisches und Digitales Objekt

In dieser Arbeit bezieht sich ein physisches Objekt auf ein Objekt, das in der realen Welt existiert. Dies kann eine Maschine, ein Werkzeug oder auch ein Hilfsmittel sein. Das Digitale Objekt ist das digitale Gegenstück dazu, welches das Objekt mit einer definierten Genauigkeit digital abbildet.

2.2 Digitale Darstellungen

Digitale Repräsentationen von Maschinen und Objekten sowie deren Interaktionen mit der realen Welt haben sich in den letzten Jahren stark weiterentwickelt und finden heute in vielfältigen Formen Anwendung. In der Fachliteratur sind insbesondere die Begriffe Digitales Modell (engl. Digital Model), Digitaler Schatten (engl. Digital Shadow) und Digitaler Zwilling (engl. Digital Twin) zentral. Während die Definitionen dieser Begriffe in der Vergangenheit stark variierten, hat sich mittlerweile ein breiter Konsens herausgebildet. Dieser ist das Ergebnis der Entwicklung einheitlicher Standards und Normen. Die von Fuller et al. [5] und Kritzinger et al. [6] vorgestellten Definitionen repräsentieren die Grundlage für das heutige Verständnis. Diese Definitionen differenzieren primär anhand des Datenflusses zwischen dem digitalen und dem physischen Objekt, wobei ein automatischer oder manueller Datenfluss entscheidend ist. In Abbildung 2.1 wird diese Unterscheidung anschaulich visualisiert und trägt somit zu einem besseren Verständnis der einzelnen Konzepte bei.

Digitales Modell

Ein Digitales Modell (engl. Digital Model) ist eine digitale Version eines bestehenden oder geplanten physischen Objekts. Es baut auf dem Digitalen Objekt auf und integriert die Steuerung der physikalischen Elemente wie Motoren, Gelenke, Federn und Dämpfer. Dabei tritt ausschließlich manueller Datenfluss zwischen den Objekten auf [5, 6].

Digitaler Schatten

Ein Digitaler Schatten (engl. Digital Shadow) ist eine digitale Darstellung eines Objekts, die einen automatischen Datenfluss vom Physischen zum Digitalen Objekt aufweist, jedoch nicht in die andere Richtung. Änderungen, die am Physischen Objekt vorgenommen werden, wirken sich auf das digitale Objekt aus, während Änderungen am Digitalen Objekt keine Auswirkungen auf das physische Objekt haben [5, 6].

Digitaler Zwilling

Ein Digitaler Zwilling (engl. Digital Twin) ist eine vollständig integrierte digitale Darstellung eines bestehenden physischen Objekts, bei der die Daten zwischen dem physischen und dem Digitalen Objekt in beide Richtungen fließen. Änderungen am physischen Objekt führen automatisch auch zu Änderungen am Digitalen Objekt und umgekehrt. Diese Verknüpfung ermöglicht eine Echtzeitinteraktion zwischen der physischen und virtuellen Welt und kann somit zur Optimierung von Produktdesign, Fertigung und Service beitragen [5, 6].

Laut Tao et al. [7] ermöglicht der Digitale Zwilling eine datengetriebene Herangehensweise in der Produktentwicklung und im Lebenszyklusmanagement. Durch die Integration von physischen und virtuellen Datenräumen kann die Effizienz, Intelligenz und Nachhaltigkeit in Produktdesign, Fertigung und Service erheblich gesteigert werden. Die Arbeit stellt zudem fest, dass die Entwicklung des Digital Zwillings die Verwendung großer Datenmengen umfasst, um den gesamten Lebenszyklus eines Produkts von der Konzeption bis zur Wartung zu unterstützen. Dadurch können auch Erkenntnisse aus der Analyse von Echtzeit- und historischen Daten gewonnen werden, um die Produktqualität kontinuierlich zu verbessern und Fehler vorherzusagen sowie zu verhindern.

In der Arbeit von Onaji et al. [8] wird noch ergänzt, dass diese Integration nicht nur die physische und virtuelle Ebene umfasst, sondern auch die Interaktion und Konvergenz von historischen und Echtzeitdaten. Dies ermöglicht eine umfassendere und dynamischere Abbildung des Produktionsprozesses, die sowohl das Domänenwissen als auch zeitnah betriebliche Informationen enthält. Darüber hinaus hat sich das Digitaler-Zwilling-Konzept von einer anfänglichen Anwendung in der Produktgestaltung zu einer umfassenden Anwendung über die gesamte Wertschöpfungskette entwickelt, einschließlich Produktionsprozess, Systemleistung und Dienstleistungen.

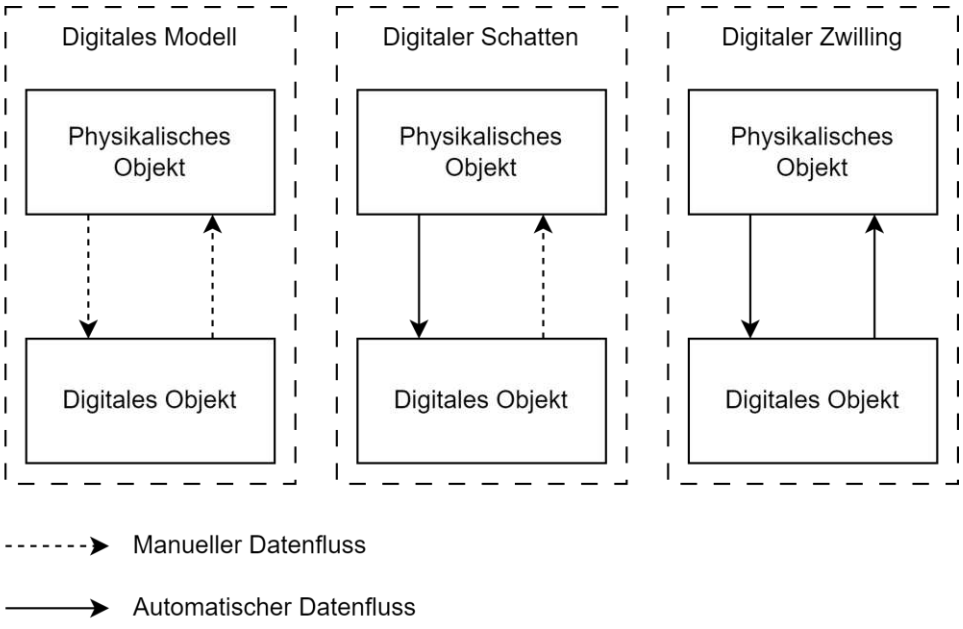


Abb. 2.1: Digitales Modell, Digitaler Schatten, Digitaler Zwilling [5]

Fünfdimensionales Modell

Im Kontext der Digitalen Darstellungen in der Fertigungstechnik ist das Konzept des Digitalen Zwillings, insbesondere dessen fünfdimensionale Modellierung und Rekonfigurierbarkeit, von besonderer Bedeutung. Diese fortschrittliche Modellierungstechnik, die von Zhang et al. [9] vorgestellt wurde, basiert auf den Dimensionen Geometrie, Physik, Fähigkeiten, Verhalten und Regeln und kann auch auf Digitale Schatten angewendet werden. Sie ermöglicht eine detaillierte und dynamische Abbildung von Fertigungssystemen in einer virtuellen Umgebung.

Wie von Tao et al. [7] diskutiert, bietet diese Modellierung die Grundlage für eine datengetriebene Optimierung und Überwachung des gesamten Produktlebenszyklus. Die Analyse von Echtzeitdaten in Verbindung mit historischen Daten kann verwendet werden, um Simulationen durchzuführen, potenzielle Fehler vorherzusagen und den Produktionsprozess effektiv zu steuern. Dies geht über traditionelle Methoden hinaus und ermöglicht es, komplexe Fertigungsprozesse in Echtzeit zu überwachen, zu optimieren und präzise neu zu konfigurieren.

2.3 Normen und Standards

Verschiedene Organisationen, die sich auf die Entwicklung von Standards konzentrieren, haben das Ziel, die Architektur und den Datenfluss von Digitalen Abbildungen zu standardisieren. Dazu gehören das Europäische Institut für Telekommunikationsnormen (ETSI), die Internationale Elektrotechnische Kommission (IEC), die Internet Engineering Task Force (IETF), das Industrial Internet Consortium (IIC), die International Organization for Standardization (ISO), das Open Geospatial Consortium, die Plattform Industrie 4.0, sowie das World Wide Web Consortium (W3C). Viele der Normen sind auf Digitale Zwillinge spezialisiert, doch viele Standards, welche für Digitale Zwillinge konzipiert sind, lassen sich auch auf Digitale Schatten und Digitale Modelle anwenden. Folgende Normen behandeln Digitale Zwillinge in verschiedenen Domänen und aus unterschiedlichen Blickwinkeln. Die jeweiligen Aspekte sind im Folgenden erläutert [10]:

ISO 21597

Die ISO veröffentlichte 2021 die Norm ISO 21597, die als allgemeiner Standard Anforderungen an Design, Implementierung und Wartung Digitaler Abbildungen festlegt. Diese Norm ist besonders relevant für den Lebenszyklus in der Fertigungsindustrie, da sie ein offenes und stabiles Containerformat für den Austausch von Dateien über deren gesamten Lebensdauer bietet. Die Ontologien, also die strukturierten Datenformate innerhalb dieser Container, basieren auf den Technologien RDF(S)/OWL. Diese sind Standards für die Beschreibung von Informationen in einer maschinenlesbaren Form. Die Daten werden entweder im Format Resource Description Framework (RDF)/Extensible Markup Language (XML) oder in einer anderen, vom World Wide Web Consortium (W3C) anerkannten Serialisierung dargestellt. Zudem stimmen die in der Norm definierten Unterscheidungen von Digitalem Modell, Digitalem Schatten und Digitalem Zwilling mit den oben genannten Differenzierungen überein [11].

ITU-T Y.3090

Dieser Standard schlägt vor, ein Netzwerk von Digitalen Zwillingen zu verwenden, um das Netzwerkmanagement über den gesamten Lebenszyklus hinweg in einem geschlossenen Kreislauf zu ermöglichen. Sie gliedert sich in drei Schichten: Die physische Netzwerkschicht (untere Schicht), welche das physische Netzwerk darstellt, Daten sammelt und von der digitalen Einheit des Netzwerks gesteuert wird. Das Netzwerk (mittlere Schicht) enthält ebenfalls drei Subsysteme. Das einheitliche Datenrepository sammelt, speichert und stellt Daten bei Bedarf zur Verfügung. Einheitliche Datenmodelle verarbeiten alle Daten, einschließlich Modellierung, Vorhersage, Planung, Konfiguration und Optimierung. Sie gewährleisten Effektivität und Zuverlässigkeit. Das Management des Netzwerkes führt alle Managementaufgaben des Zwillings durch, wie Lebenszyklusaufzeichnung, Topologiemanagement, Modellmanagement und Sicherheitsmanagement. Die Netzwerkanwendungsschicht (oberste Schicht) ist die Schnittstelle zum Netzwerk der Digitalen Zwillingen [12].

ISO/IEC 23247

Der Standard ISO/IEC 23247 legt ein Framework für Digitale Zwillinge in der Fertigungsindustrie fest. Dies basiert auf der ISO/IEC 30141, einer Referenzarchitektur speziell für das Internet of Things (IoT). Ein Digitales Zwillingsframework bezeichnet dabei ein strukturiertes Modell, das es ermöglicht, virtuelle Abbildungen realer Fertigungsobjekte und -prozesse zu erstellen und zu verwalten. Im Rahmen dieses Standards wird ein domänenspezifisches Referenzmodell vorgeschlagen, das drei Hauptbereiche umfasst: die Nutzerdomäne, die Domäne des Digitalen Zwillings selbst und die Domäne für die Kommunikation zwischen Geräten. Diese Domänen werden durch ein sogenanntes Entitätenreferenzmodell, welches die Struktur und Beziehungen zwischen den verschiedenen Elementen festlegt, geteilt und strukturiert. Innerhalb jeder Domäne werden die Einheiten weiter in Benutzer-Einheiten, die die Interaktion der Nutzer mit dem System beschreiben, und in funktionale Einheiten, die spezifische Aufgaben und Funktionen innerhalb des Systems übernehmen, unterteilt. Der Informationsaustausch ist unterteilt in Nutzer-, Zugriffs-, Nähe- und Servicenetzwerke. Allerdings wird letzteres nicht vom Standard abgedeckt und ist nur notwendig, wenn die Digitale-Zwilling-Einheiten nicht als einzelnes privates System implementiert sind [13, 14].

2.4 Schnittstellen

Um Daten von einem System in ein anderes zu übertragen, werden Schnittstellen benötigt. Sie unterscheiden sich nicht nur durch die Struktur der Daten, sondern vor allem wie sie übertragen werden. Um es vereinfacht darzustellen, wird grundsätzlich unterschieden, welches Gerät für die Übertragung zuständig ist, der Sender oder der Empfänger. Es gibt die Möglichkeit, dass der Sender die Daten in einem bestimmten Intervall aussendet und Empfänger diese Daten dann aufschnappen. Andersherum gibt es Schnittstellen, wo der Empfänger Daten anfragt und diese dann vom Sender übergeben werden. Die in der Fertigungsindustrie gängigsten Schnittstellen werden hier kurz erläutert [15]:

2.4.1 REST

Representational State Transfer (REST) steht für Representational State Transfer, einen Architekturstil, der sich durch die Nutzung standardisierter HTTP-Methoden wie GET, POST, PUT und DELETE auszeichnet, um mit Ressourcen über das Web zu interagieren. RESTful-Schnittstellen sind aufgrund ihrer Einfachheit in der Implementierung und Benutzung besonders beliebt in der Entwicklung von Webdiensten und Microservices. Diese Schnittstellen ermöglichen eine modulare Architektur, bei der einzelne Komponenten mit minimaler Kopplung leicht ausgetauscht oder skaliert werden können. Bei der Verwendung von REST fordert der Empfänger aktiv Daten an, indem er eine HTTP-Anfrage an den Sender richtet, der daraufhin die angeforderten Daten in einer standardisierten Form, häufig im JavaScript Object Notation (JSON)- oder XML-Format, zurücksendet. Diese klare Trennung zwischen Client (Empfänger) und Server (Sender) fördert die Skalierbarkeit und Flexibilität in der Anwendungsentwicklung [16].

2.4.2 MQTT

Message Queuing Telemetry Transport (MQTT) ist ein Protokoll, das speziell für die Bedürfnisse des IoT entwickelt wurde, mit einem Fokus auf minimale Bandbreitennutzung und effiziente Datenübertragung. Es basiert auf einem Publish-Subscribe-Modell, das eine Entkopplung von Datenproduzenten und -konsumenten ermöglicht. In diesem Modell publiziert der Sender (Publisher) Nachrichten zu Themen (Topics), ohne Kenntnis darüber zu haben, welche oder wie viele Empfänger (Subscriber) diese Nachrichten erhalten werden. Die Empfänger abonnieren Themen, die für sie von Interesse sind, und erhalten Updates automatisch, sobald neue Daten publiziert werden. Ein Topic ist dabei ein benannter Kanal, über den Nachrichten publiziert werden. Es dient als Filtermechanismus, sodass Subscriber nur die für sie relevanten Nachrichten erhalten. Diese Art der Kommunikation ist besonders vorteilhaft in IoT-Umgebungen, wo Geräte häufig nur sporadisch Daten senden oder empfangen und Netzwerkbandbreite eingeschränkt ist [17].

2.4.3 OPC UA

OPC Unified Architecture (OPC UA) ist ein fortschrittlicher Kommunikationsstandard in der industriellen Automatisierung, der eine plattformunabhängige, serviceorientierte Architektur zur Verfügung stellt. Diese Architektur ermöglicht eine sichere und zuverlässige Datenübertragung über verschiedene Netzwerkgrenzen hinweg. OPC UA zeichnet sich durch seine Fähigkeit aus, nicht nur einfache Datenpunkte, sondern auch komplexe Datenstrukturen und Maschinenzustände zu übertragen. Es kann sowohl in Client/Server- als auch in Publisher/Subscriber-Konfigurationen implementiert werden, was eine breite Anwendungsvielfalt unterstützt. Im Client/Server-Modell initiiert der Client (Empfänger) die Kommunikation durch Anforderung von Daten vom Server (Sender). Im Publisher/Subscriber-Modell hingegen werden Daten vom Publisher ohne spezifische Empfängeranforderung publiziert, und Subscriber empfangen Daten basierend auf ihren Abonnements. Diese Vielseitigkeit macht OPC UA zu einer idealen Wahl für die Integration in heterogene Systemlandschaften [15].

2.4.4 DDS

Data Distribution Service for Real-Time Systems (DDS) wurde speziell für Echtzeit- und eingebettete Systemanwendungen entwickelt. Es bietet ein robustes Framework für den Datenaustausch, das auf einem Publish-Subscribe-Modell basiert. DDS unterscheidet sich durch seine ausgefeilten Quality-of-Service (QoS)-Parameter, die eine präzise Kontrolle über Aspekte wie Datenlieferung, Zuverlässigkeit und Latenz bieten. Sender (Publisher) teilen Daten ohne direkte Kenntnis der Empfänger (Subscriber), die wiederum die für sie relevanten Datenströme abonnieren können. Diese Architektur unterstützt hochdynamische und skalierbare Systeme, in denen die Anforderungen an die Datenübertragung streng sind [15].

2.4.5 oneM2M

oneM2M ist ein globaler Standard, der eine konsistente und interoperable Plattform für IoT-Netzwerke bereitstellt. Er zielt darauf ab, die Fragmentierung im IoT zu überwinden, indem er eine gemeinsame Architektur für die Integration verschiedenster Anwendungen und Dienste schafft. Durch die Unterstützung von Request/Response und Publish/Subscribe Kommunikationsmodellen ermöglicht oneM2M die effiziente Vernetzung und das Management von IoT-Geräten über verschiedene Industrien und Einsatzgebiete hinweg. Die Fokussierung auf Sicherheit und Datenschutz gewährleistet, dass die über oneM2M realisierten Lösungen den heutigen Anforderungen an sichere IoT-Anwendungen gerecht werden. Die flexible Architektur erlaubt es, dass sowohl Sender als auch Empfänger je nach verwendetem Modell die Datenübertragung initiieren können, was eine breite Palette von Anwendungsfällen unterstützt [15].

2.4.6 IoTivity

IoTivity, ein von der Open Connectivity Foundation unterstütztes Open-Source-Projekt, bietet eine umfassende Plattform zur Vernetzung von IoT-Geräten. Es zielt darauf ab, die Fragmentierung im IoT zu verringern, indem es Standards für die sichere und effiziente Kommunikation zwischen Geräten unterschiedlicher Hersteller und Technologien setzt. Durch die Unterstützung von CoAP, einem speziell für ressourcenbeschränkte Geräte entwickelten Protokoll, ermöglicht IoTivity die Implementierung von Request/Response-Kommunikationsmustern, in denen der Empfänger die Datenübertragung durch das Stellen von Anfragen initiiert. Diese Plattform ist besonders darauf ausgerichtet, Entwicklern die Werkzeuge an die Hand zu geben, die sie benötigen, um interoperable Produkte zu schaffen, die nahtlos in das breitere Ökosystem des Internet of Things integrieren [15].

2.5 Softwaremöglichkeiten

Um ein Digitales Abbild zu erstellen, ist eine Simulationssoftware notwendig. In der Arbeit von Wang et al. [18] wird ein sehr guter Überblick über die Implementierung von Digitalen Zwillingen in wissenschaftlichen Arbeiten gegeben und mit welchen Programmen sie umgesetzt wurden. Sie bietet eine Auflistung von über 100 Paper, etwa 40% davon im Bereich Fertigung. Eine Analyse dieser Arbeit hat gezeigt, dass es in dieser Domäne noch keine dominante Softwarelösung gibt und viele Autoren eigens entwickelte Programme, oft in Kombination mit CAD-Programmen verwendeten. Lo et al.

[19] listet die gängigsten Programme für Digitale Zwillinge von den führenden CAD-Anbietern auf, welche auch in dieser Arbeit in Betracht gezogen werden können.

2.5.1 Simulationsprogramme

Aufgrund dieser vorherigen Arbeiten kann die Auswahl der Software auf folgende Liste zusammengefasst werden (alphabetisch):

- **ABB RobotStudio**: Das von ABB entwickelte Programm RobotStudio ist eine Anwendung für die Simulation und Offline-Programmierung von ABB-Industrierobotern. Die Software ermöglicht es BenutzerInnen, Roboterprogramme offline zu erstellen und in einer virtuellen Umgebung zu testen, bevor sie auf einen physischen Roboter angewendet werden. Das Programm kommuniziert nach außen über Schnittstellen, welche über das Tool IoTGateway vom selben Hersteller weiterverarbeitet werden können.
- **ANSYS Twin Builder**: ANSYS Twin Builder ist effektiv in der Simulation des Verhaltens von Maschinen und Anlagen. Die Software bietet jedoch keine integrierten Lösungen für die direkte Steuerung von Fertigungsprozessen oder für die umfassende Integration in bestehende Produktionssteuerungssysteme.
- **Dassault Systèmes DELMIA**: DELMIA ist spezialisiert auf die Planung und Simulation von Montage- und Automatisierungsprozessen in der Fertigungstechnik. Die Software bietet jedoch keine Unterstützung für die Echtzeit-Überwachung und Kontrolle von Fertigungsanlagen, was in Produktionsumgebungen, die dynamische Anpassungen erfordern, eine Limitation darstellt.
- **PTC ThingWorx**: PTC ThingWorx unterstützt effektiv die Implementierung von IoT-Lösungen in der Fertigung, insbesondere für Überwachung und Analyse von Maschinendaten. Die Plattform ist jedoch nicht für tiefgreifende Analysen komplexer Fertigungsdaten oder für die Integration fortschrittlicher maschineller Lernmodelle konzipiert.
- **Siemens NX**: Siemens NX ist ein leistungsstarkes Programm, das in vielen Branchen verbreitet ist, insbesondere in der Automobilindustrie, der Luft- und Raumfahrtindustrie sowie in der Fertigungsindustrie. Neben der Funktionalität als CAD Software besitzt das Programm Schnittstellen wie OPC UA und NX Open. Während über OPC UA einfach Daten eingelesen werden können, ermöglicht NX Open das Programmieren auf tiefer (low-level) Basis in verschiedensten Sprachen (C#, C++, Java, Python, Visual Basic). Bestehende CAD-Daten können in Siemens NX über gängige Formate wie IGES, STEP, JT und STL importiert werden. Der Mechatronic Concept Designer (MCD) ist ein Modul von Siemens NX, das speziell für die schnelle und effiziente Konzeptentwicklung mechatronischer Systeme entwickelt wurde. Die Software bietet eine physikalische Simulationsumgebung, in der Systeme simuliert und getestet werden können. Über Achsen, Gelenke und Regler können Ist- bzw. Sollwerte eingelesen und ausgegeben werden. Dabei können physikalische Eigenschaften wie Geschwindigkeit oder maximale Beschleunigungen definiert werden. Unter den vielen nach außen führenden Schnittstellen befindet sich auch ein OPC UA Server. Um Bewegungsabläufe darzustellen, gibt es außerdem die Funktion des Sequenz-Editors, wo über Logikbausteine Abläufe aufgebaut werden können.

- **Microsoft Azure Digital Twins**: Microsoft Azure Digital Twins eignet sich für die Datenintegration und -visualisierung in der Fertigung. Die Plattform bietet jedoch keine direkten Funktionen zur Steuerung von Fertigungsanlagen oder zur detaillierten physikalischen Simulation spezifischer Fertigungsprozesse.

Die vorgestellten Programme bieten verschiedene Ansätze zur Integration in reale Systeme. Mit dem Aspekt der automatischen Aktualisierung des Digitalen Schattens, der Datenspeicherung, sowie Laden und Speichern von Konfigurationen wird je nach Programm jedoch sehr schnell auf Grenzen gestoßen. Um dies zu umgehen, besteht die Möglichkeit, Programme oder Tools selbst zu entwickeln, welche Funktionen, wie beispielsweise die Datenspeicherung, übernehmen können. Eine Auflistung zeigt die meistverwendeten Programmiersprachen in der Fertigungstechnik [20] (alphabetisch):

C#

C# ist eine von Microsoft entwickelte Programmiersprache, die in der .NET-Plattform integriert ist. Die Sprache wird oft für Windows- und Webanwendungen sowie neuerdings für Cross-Plattform-Anwendungen verwendet. Ihre Merkmale umfassen automatische Speicherverwaltung und starke Typisierung, welche zur Entwicklung beständiger und wartbarer Code-Strukturen beitragen.

Java

Java ist eine verbreitete Programmiersprache, die besonders für ihre Plattformunabhängigkeit und Robustheit bekannt ist. Sie wird global in der Unternehmensentwicklung eingesetzt und ist stark objektorientiert, unterstützt durch eine umfassende Standardbibliothek. Java profitiert zudem von einer aktiven Community und einer Vielzahl an Open-Source-Bibliotheken, die die Integration mit anderen Systemen und Technologien vereinfachen.

Python

Python ist eine der weltweit verbreitetsten Programmiersprachen, insbesondere in Bereichen außerhalb von Skript-Sprachen [21]. Sie wird aufgrund ihrer einfachen Syntax und klaren Strukturierung in vielen Anwendungsbereichen eingesetzt. Python bietet zahlreiche Open-Source-Bibliotheken zur vereinfachten Interaktion mit verschiedenen Schnittstellen wie OPC UA, MQTT, REST und vielen mehr.

2.5.2 Datenspeicherung

Um Konfigurationen und Simulationen zu speichern und wieder abzuspielen, müssen Daten gespeichert werden. Im Normalfall wird dafür eine Datenbank verwendet. Viele der Simulationsprogramme, vor allem die der großen Hersteller, haben bereits integrierte Datenbanken. Für Programme, die ihren Ursprung aus der 3D-Modellierung beziehen, ist die Nutzung einer externen Datenbank notwendig. Bei der Auswahl der passenden Datenbank sind, abhängig vom Anwendungsgebiet, grundlegende Fragen zu klären [22]:

- **Abfragemuster**: Wie komplex sind die Abfragemuster? Werden Abfragen nach mehreren Schlüsseln durchgeführt? Wird eine ungefähre Suche durchgeführt?

- **Konsistenz**: Ist starke Konsistenz erforderlich?
- **Speicherkapazität**: Wie viel Speicherkapazität wird benötigt?
- **Leistungsfähigkeit**: Wie hoch ist die erforderliche Leistungsfähigkeit und die Latenzzeit?
- **Ausgereiftheit und Stabilität**: Wie viel Erfahrung mit der Datenbank ist notwendig? Wie ausgereift ist sie?
- **Kosten**: Was sind die Kosten? Welche Einschränkungen gibt es?

Diese Fragen werden bei der Auswahl der Datenbank in Unterabschnitt 6.3.3 beantwortet.

3. Zielsetzung

Die vorliegende Arbeit verfolgt das Ziel, einen Digitalen Schatten für eine flexible Fertigungszelle zu entwickeln und zu implementieren. Der Fokus liegt dabei auf der Abbildung des realen Systems in einer Simulationsumgebung, um physikalische Eigenschaften möglichst realitätsnah zu simulieren. Dies ermöglicht eine detaillierte Analyse der Produktionsprozesse, eine verbesserte Fehlererkennung sowie eine Optimierung der Abläufe.

Kernaspekte der Zielsetzung sind:

- Erstellung eines digitalen Modells der Fertigungszelle, das kinematische Bewegungen sowie relevante physikalische Parameter berücksichtigt.
- Integration der Schnittstellen zur Echtzeit-Datenübertragung zwischen der physischen Fertigungszelle und dem Digitalen Schatten.
- Entwicklung einer nutzerfreundlichen Schnittstelle zur Steuerung und Visualisierung der digitalen Abbildung.
- Speicherung und Analyse von Zeiträumen zur späteren Wiedergabe und detaillierten Untersuchung der Produktionsprozesse.
- Sicherstellung einer flexiblen Systemarchitektur, die eine einfache Anpassung und Erweiterung ermöglicht.

Ein besonderes Augenmerk wird auf die Datenübertragung gelegt, um eine möglichst präzise und reaktionsschnelle Simulation zu gewährleisten. Zudem soll durch die Implementierung einer geeigneten Speichermöglichkeit eine umfassende Analyse von Abläufen ermöglicht werden. Die erarbeiteten Konzepte und Methoden dienen als Grundlage für zukünftige Weiterentwicklungen in Richtung eines vollständigen Digitalen Zwillings. Es ist zu beachten, dass der Begriff “Echtzeit” in diesem Kontext eine zeitnahe Datenverarbeitung mit einer akzeptablen Latenz bedeutet. Die genaue Definition von “akzeptabler Latenz” variiert je nach Anwendung und spezifischen Anforderungen des Systems. In der industriellen Automatisierung und Fertigung werden oft Latenzzeiten im Bereich von Millisekunden bis zu einigen hundert Millisekunden als akzeptabel angesehen, um eine effektive Steuerung und Synchronisation zu gewährleisten.

4. Methodische Vorgangsweise

Das V-Modell von [23] ist ein bewährtes Vorgehensmodell für die Entwicklung komplexer Systeme. Es beschreibt eine strukturierte und phasenorientierte Vorgehensweise, die von der Anforderungsanalyse über die Implementierung bis hin zur Integration reicht. Die methodische Vorgehensweise orientiert sich am V-Modell, um eine systematische Entwicklung des Digitalen Schattens sicherzustellen.

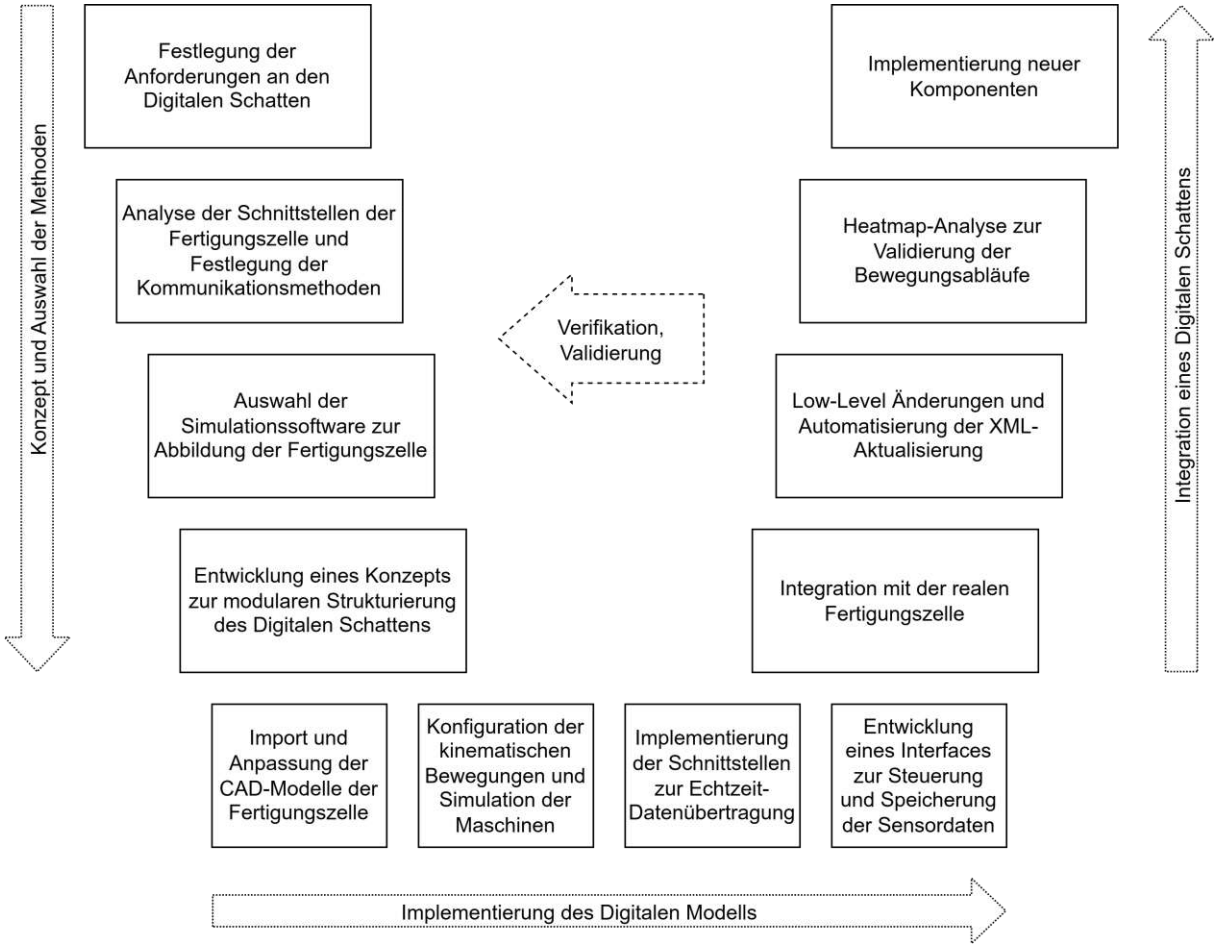


Abb. 4.1: Methodische Vorgangsweise in Anlehnung an das V-Model VDI/VDE 2206 Gausemeler und Moehringer [24]

Wie in Abbildung 4.1 dargestellt sind in dieser Arbeit die drei zentralen Komponenten des V-Modells wie folgt festgelegt:

- Links: Methodenauswahl und Konzeptentwicklung
- Unten: Implementierung des Digitalen Modells
- Rechts: Integration eines Digitalen Schattens

4.1 Methodenauswahl und Konzeptentwicklung

Diese Phase umfasst die Untersuchung bestehender Schnittstellen und Softwarelösungen sowie die Auswahl geeigneter Methoden zur Implementierung des Digitalen

Schattens:

- Festlegung von Anforderungen an den Digitalen Schatten.
- Analyse der Schnittstellen der Fertigungszelle und Festlegung der Kommunikationsmethoden.
- Auswahl der Simulationssoftware zur Abbildung der Fertigungszelle.
- Entwicklung eines Konzepts zur modularen Strukturierung des Digitalen Schattens.

Diese Schritte legten die Basis für die spätere Implementierung des Systems.

4.2 Implementierung des Digitalen Modells

In dieser Phase wurde das Digitale Modell der Fertigungszelle erstellt und die definierten Konzepte umgesetzt:

- Import und Anpassung der CAD-Modelle der Fertigungszelle.
- Konfiguration der kinematischen Bewegungen und Simulation der Maschinen.
- Implementierung der Schnittstellen zur Echtzeit-Datenübertragung.
- Entwicklung eines Interfaces zur Steuerung und Speicherung der Sensordaten.

Diese Implementierung ermöglichte eine präzise Simulation und Interaktion mit der realen Fertigungszelle.

4.3 Integration eines Digitalen Schattens

Die letzte Phase umfasste die Überprüfung und Feinabstimmung des Digitalen Schattens:

- Integration mit der realen Fertigungszelle.
- Low-Level Änderungen und Automatisierung der XML-Aktualisierung.
- Heatmap-Analyse zur Validierung der Bewegungsabläufe.
- Implementierung neuer Komponenten.

Durch diese Schritte wurde sichergestellt, dass der Digitale Schatten eine realitätsnahe Abbildung der Fertigungszelle ermöglicht und zukünftige Erweiterungen erleichtert.

5. Konzept und Auswahl der Methoden

Im vorherigen Kapitel wurde der aktuelle Stand der Technik sowie Lösungen und Arbeiten mit ähnlichen Themen recherchiert. Mit diesem Wissen können nun Technologien für diese Arbeit ausgewählt und die Umsetzung geplant werden. Im ersten Schritt soll die Fertigungszelle, welche für diese Arbeit herangezogen wurde, analysiert und deren Komponenten isoliert betrachtet werden. Weiters soll untersucht werden, welche Schnittstellen bereits mit den Maschinen bestehen und unterstützt werden. Mit Betracht auf die Aufgabenstellung (Abschnitt 1.2), werden ein Simulationsprogramm und eventuell notwendige zusätzliche Programme ausgewählt. Darauf aufbauend kann die Umsetzung geplant und genaue Ziele definiert werden.

5.1 Aufbau der Fertigungszelle

Die Fertigungszelle (Abbildung 5.1) besteht maschinell aus einem ABB IRB 120, einem 6-achsigen Knickarm-Roboter (Abbildung 5.2a) und einer EMCO CM 55, einer 3-Achs-Fräsmaschine mit einem Werkzeugmagazin mit 8 Plätzen (Abbildung 5.2b). Diese sind über einen Aufbau fix miteinander verbunden. Rohmaterial wird auf Werkstückpaletten eingespannt, welche auf 3D-gedruckte Magazine mittels Zentrierbolzen abgelegt werden. Ein Magazin ist von dem/der Benutzer/in erreichbar und dient zum Rüsten von Rohmaterial und Entnehmen des Fertigteils, sechs weitere Magazine dienen als Zwischenlager und sind nur vom Roboterarm erreichbar. Der ABB IRB 120 ist mit einem Schunk NSR mini 100 ausgestattet, mit dem Werkstückpaletten aufgenommen und transportiert werden können. Die CM 55 ist mit einem Nullpunktspannsystem NSE Mini 90 ausgestattet, wodurch die Werkstückpaletten positioniert werden können. Die Steuerung und Überwachung übernimmt ein BECKHOFF Leitrechner, der mit allen Maschinen und Sensoren kommuniziert.

Abb. 5.1: Fertigungszelle

(a) IRB 120 (b) CM 55

Abb. 5.2: Maschinen-Komponenten der Fertigungszelle

5.2 Anforderungen

Die Entwicklung des Digitalen Schattens basiert auf klar definierten Anforderungen, die sicherstellen, dass die angestrebten Ziele erreicht werden. Dabei wird zwischen funktionalen und nicht-funktionalen Anforderungen unterschieden, um eine strukturierte Umsetzung und spätere Überprüfung zu gewährleisten.

5.2.1 Funktionale Anforderungen

Die funktionalen Anforderungen definieren die konkreten Funktionen, die das System bereitstellen muss:

- Der Digitale Schatten muss den Programmablauf frühzeitig testen und Fehler erkennen können.
- Es muss möglich sein, mehrere Objekte absolut und relativ zueinander zu positionieren.
- Die Simulation muss Kollisionen zwischen Objekten erkennen.
- Die Simulation muss eine flüssige Bewegung von Maschinen und Objekten ermöglichen.
- Komplexe Bewegungsabläufe, wie Werkzeugwechsel oder die Bewegung von Knickarm-Robotern, müssen realisierbar sein.
- Das System muss eine Echtzeit-Synchronisation mit der Fertigungszelle ermöglichen.
- Es muss möglich sein, einen Zeitraum aufzuzeichnen, zu speichern und später abzuspielen.

- Simulationen sollen auch unabhängig von Echtzeitdaten der realen Fertigungszelle durchgeführt werden können.
- Das Interface und die grafische Oberfläche müssen eine benutzerfreundliche Steuerung der Simulation ermöglichen.
- Eine Heatmap zur Visualisierung der Verweildauer von Objekten im dreidimensionalen Raum soll integriert werden.

5.2.2 Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen legen fest, wie das System die funktionalen Anforderungen umsetzen soll und welche Qualitätskriterien dabei berücksichtigt werden müssen:

- Die Integration existierender CAD-Dateien in Siemens NX muss einfach und effizient erfolgen.
- Physikalische Änderungen im realen System sollen mit minimalem manuellen Aufwand im Digitalen Schatten aktualisiert werden können.
- Die Simulation soll in Echtzeit oder mit einer definierten Präzision ablaufen können.
- Die Schnittstellen zur Datenübermittlung müssen zuverlässig arbeiten.
- Die Übertragungsintervalle der Echtzeitdaten müssen optimiert und analysiert werden.
- Die Strukturierung der Daten muss sich an den in Abschnitt 2.3 vorgestellten Normen orientieren.

Diese Anforderungen bilden die Grundlage für die Implementierung und dienen als Referenz für die spätere Verifikation und Validierung.

5.3 Analyse der Schnittstellen

Geplant war, dass der Leitrechner für die Datenübertragung innerhalb und außerhalb der Fertigungszelle zuständig sein würde. Aufgrund von Lieferproblemen konnte dieser jedoch nicht in Betrieb genommen werden. Als Alternative wurde untersucht, welche Schnittstellen die einzelnen Komponenten unterstützen, um diese direkt anzusteuern und Daten auszulesen. Ein von Thomas Trautner entwickeltes Programm, der sogenannte RobotServer, ermöglicht die Kommunikation mit dem IRB 120 über REST und MQTT. Dieses Programm nutzt das RobotStudio SDK in C#, um die vorhandenen Funktionen und Daten des Roboters über HTTP bereitzustellen, während Echtzeitdaten über MQTT übertragen werden. Zusätzlich bietet die Zusatzsoftware ABB IoT Gateway die Steuerung über OPC UA und MQTT an. Der Roboterarm erlaubt zudem den Anschluss von digitalen IO-Schnittstellen. Diese Signale sind über alle Schnittstellen des IRB 120 verfügbar und können genutzt werden, um auch die CM 55 über dieselben Protokolle zu steuern. Siemens NX unterstützt sowohl OPC UA als auch NX Open für eine Low-Level-Schnittstelle.

5.4 Kinematische Bewegungen

Um alle Komponenten und die damit verbundenen kinematischen Bewegungen abzubilden, werden die einzelnen Bewegungen analysiert. Der Fräskopf der CM 55 bewegt sich linear in eine Achse, der Schlitten in zwei. Der Werkzeugwechsler bewegt sich in einer rotatorischen Bewegung. Der IRB 120 ist hierbei etwas komplizierter, die einzelnen Gelenke lassen sich zwar als rotatorische Bewegungen beschreiben, werden die Daten allerdings in xyz-Koordinaten übergeben, ist eine Rückrechnung auf die Achswerte mittels inverser Kinematik notwendig. Alle anderen Komponenten sind starr und weisen keine kinematischen Bewegungen auf.

5.5 Auswahl der Software

5.5.1 Simulationssoftware

Mit den gewonnenen Erkenntnissen in diesem Kapitel und der Recherche aus Unterabschnitt 2.5.1, kann ein Programm für die Simulation ausgewählt werden. Ein besonderer Fokus wird auf die Schnittstellen, das Importieren von Modellen und die Integration im Institut für Fertigungstechnik und Photonische Technologien für weiterführende Arbeiten gelegt. Aus diesen Gründen wurde Siemens NX mit dem MCD Modul ausgewählt. Es unterstützt OPC UA, für beide Maschinen gab es bereits native Siemens NX Modelle und das Programm ist derzeit am Institut als Standard-CAM Programm verwendet. Diese Entscheidung stützt sich auch auf die Recherche von Wang et al. [18], wo ersichtlich ist, dass viele aktuelle Digitale Zwillinge in der Forschung im Bereich Fertigungstechnik mittels Siemens NX und dem MCD Modul erstellt wurden. Außerdem kann somit sichergestellt werden, dass es in Zukunft bei der Weiterverwendung nicht zu Problemen der Unterstützung kommt, denn mit dieser Lösung kann herstellerunabhängig jede Maschine abgebildet werden, sobald CAD Daten verfügbar sind. Es unterstützt eine Schnittstelle namens NX Open, welche tiefgreifendere Steuerungen von Siemens NX ermöglicht. Siemens NX unterstützt allerdings keine Möglichkeit, Daten zu speichern, deswegen ist eine externe Lösung notwendig. In dieser Arbeit wird immer wieder von Signalen und Signaladapter gesprochen. Signale in Siemens NX MCD stehen für die Befehle, die an Maschinen gesendet werden, wie z. B. das Starten oder Stoppen einer Aktion. Signaladapter sind die Tools, die diese Befehle in ein Format umwandeln, das von der Maschine verstanden wird. Sie ermöglichen die Kommunikation zwischen der Schnittstelle, in diesem Fall OPC UA, und der realen Maschinensteuerung.

Um die Arbeit in der Entwicklung zu vereinfachen und nicht immer mit der realen Fertigungszelle verbunden sein zu müssen, wurde entschieden, ABB RobotStudio im Entwicklungsprozess zu nutzen. Es kann dafür verwendet werden, einen virtuellen IRB 120 bereitzustellen. Dieser verhält sich wie die reale Maschine und liefert Daten, welche für Entwicklung und Überprüfungen genutzt werden können. Der Vorteil ist, dass die Software vom selben Hersteller zur Verfügung gestellt wird, weswegen davon ausgegangen werden kann, dass die Dimensionen und physikalischen Eigenschaften korrekt sind.

5.5.2 Steuerung

Zwar kann Siemens NX direkt Signale über OPC UA empfangen und diese verarbeiten, wenn jedoch die Daten gespeichert und abgespielt werden müssen, ist es notwendig, dass eine externe Steuerung diesen Teil übernahm. Außerdem können über so eine Steuerung auch die Konfigurationen verwaltet werden. Aufgrund dieser Anforderungen wurde entschieden, diese Steuerung mit einer grafischen Oberfläche auszustatten. Für die Programmierung wurde Python gewählt, da es NX Open Schnittstellen von Siemens NX unterstützt und eine große Auswahl an Packages bietet. Diese Komponente wird nun in dieser Arbeit als das Interface definiert, da es als das Interface zwischen den Schnittstellen der Fertigungszelle und Siemens NX, der grafischen Oberfläche und der Datenspeicherung agiert.

5.5.3 Datenspeicherung

Ein Ziel der Arbeit ist es, Bewegungsvorgänge aufzunehmen und wiederzugeben. Dafür ist es notwendig, einerseits Konfigurationen, wie etwa den Namen, die Zeit der Aufnahme und die aktuelle Konfiguration und Positionen der Maschinen im Raum, andererseits die Daten der Bewegungen, auch Zeitreihendaten genannt, zu speichern. Diese zwei Datensätze unterscheiden sich deutlich voneinander. Während die Daten für die Konfigurationen mehr Informationen beinhalten, werden sie nur selten, also etwa beim Erstellen einer Aufnahme oder Ändern einer Konfiguration, geschrieben oder gelesen. Zeitreihendaten hingegen haben eine sehr hohe Frequenz, im Bereich von 10 bis 60 Einträgen pro Sekunde, um eine möglichst flüssige Bewegung zu gewährleisten. Um passende Datenbanken zu finden, werden die Entscheidungsfragen aus Unterabschnitt 2.5.2 herangezogen.

Konfigurationen

Da Konfigurationen klar definierte Strukturen haben, eignet sich dafür eine relationale Datenbank. Die Speicherkapazität stellt kein Problem dar, da nur geringe Datenmengen (wenige MB) erwartet werden. Was die Performance betrifft, so sind keine hohen Anforderungen zu erwarten, da Daten hauptsächlich beim Speichern und Laden manipuliert werden. Für eine stabile Anwendung und eine relativ einfache Datenbankstruktur kann auf herkömmliche und gut etablierte Systeme zurückgegriffen werden, um Stabilität zu gewährleisten. Mit diesen Anforderungen wurden folgende Möglichkeiten gefunden [25]:

- **SQLite**: Aufgrund ihrer Einfachheit und Kompaktheit eignet sich SQLite besonders für Anwendungen mit geringem Datenvolumen und minimalen Performance-Anforderungen.
- **PostgreSQL**: Als robustes und leistungsfähiges Open-Source-System ist PostgreSQL auch für potenzielles zukünftiges Wachstum geeignet.
- **MySQL/MariaDB**: Beide sind weit verbreitete Open-Source-Datenbanksysteme, die für ihre Zuverlässigkeit und Benutzerfreundlichkeit bekannt sind.
- **Microsoft SQL Server Express**: Diese kostenlose Edition des Microsoft SQL Servers bietet robuste Features und ist für die Entwicklung und den Betrieb kleinerer Anwendungen geeignet.

Mit dem Aspekt, dass Python für die Programmierung verwendet wird, wurde SQLite ausgewählt. Es ist direkt in Python integriert, sodass keine zusätzlichen Treiber oder Plugins benötigt werden.

Zeitreihen

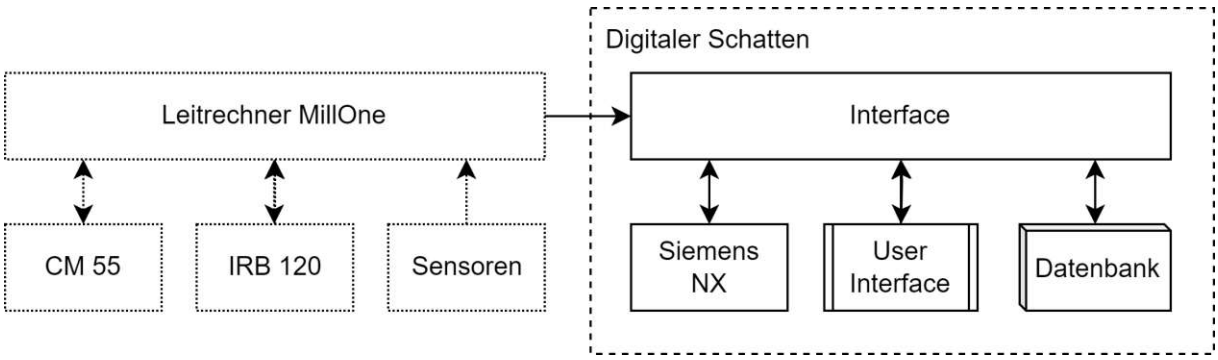
Zeitreihen zeichnen sich durch sehr einfache Strukturen aus, weshalb ein Key-Value Store ausreichend ist. Die eingespielten Zeitreihendaten haben wenig Relationen und müssen auch nicht gleich wieder ausgelesen werden, weshalb eine dokumentenbasierende Datenbank sinnvoll wäre. Zeitreihen beinhalten viele Einträge, worunter die Speicherkapazität leidet. Hier können bei intensiver Nutzung Objektspeicher-Dienste nützlich sein, gegenüber herkömmlichen Datenbanken wie SQL. Anforderungen an die Datenbank sind eine genügend hohe Schreib- und Lesegeschwindigkeit. SQL reicht hier grundlegend aus, sollten jedoch simultane Anfragen auftreten, wäre eine Cloud-Lösung wie DynamoDB und Google's Bigtable eine Option. Folgende Datenbanken wurden für diese Anwendung gefunden [26]:

- **InfluxDB**: Eine Open-Source-Zeitreihendatenbank, die entwickelt wurde, um hohe Schreib- und Abfragegeschwindigkeiten zu ermöglichen. Sie ist gut geeignet für Anwendungen im Bereich des Internet of Things (IoT), der Echtzeitanalyse und der Speicherung von Protokolldaten.
- **TimescaleDB**: Eine Open-Source-Datenbank, die auf PostgreSQL aufbaut und speziell für Zeitreihendaten optimiert wurde. Sie bietet vollständige SQL Unterstützung und erweiterte Funktionen wie Datenkomprimierung und kontinuierliche Aggregation.
- **OpenTSDB**: Eine verteilte und skalierbare Open-Source-Zeitreihendatenbank, die auf HBase (Hadoop) aufbaut. Sie ist gut geeignet für die Verarbeitung großer Datenmengen.
- **Prometheus**: Eine Open-Source-Monitoring- und Alarmierungslösung, die auch eine eingebaute Zeitreihendatenbank bietet. Sie ist sehr beliebt im Bereich der Infrastrukturüberwachung.

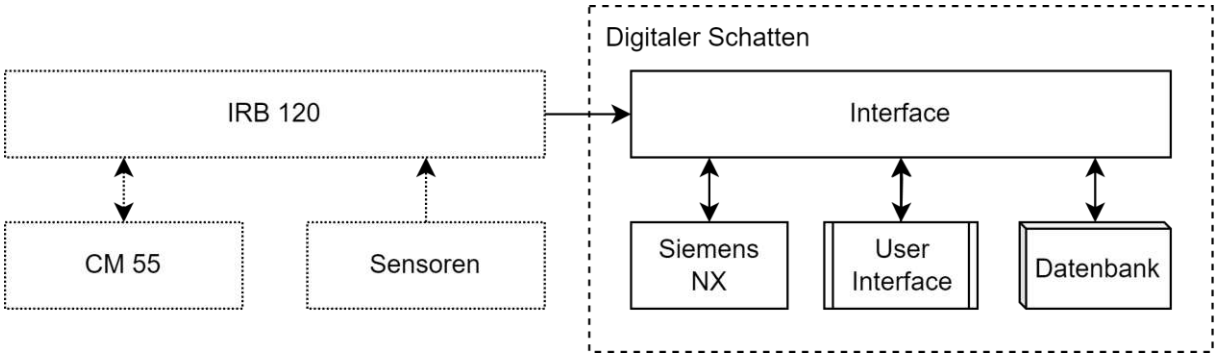
Aufgrund der Performance von SQL [27] und der Einfachheit, keine zweite Datenbank zu benötigen, wurde entschieden, auch alle Zeitreihendaten in der SQLite-Datenbank zu speichern. Für die Zukunft kann ein Wechsel zu anderen, auf Zeitreihen spezialisierten Datenbanken in Betracht gezogen werden, aber für den Moment war diese Lösung ausreichend und vereinfachte den Entwicklungsprozess.

5.6 Konzept

Abbildung 5.3 zeigt die geplante Kommunikation zwischen dem Leitrechner des Digitalen Schattens. In Abbildung 5.3a ist die geplante Konfiguration mittels Leitrechner zu sehen, in Abbildung 5.3b die tatsächliche Umsetzung über die Schnittstellen der CM 55, wie in Abschnitt 5.3 beschrieben.



(a) Mit Leitrechner



(b) Ohne Leitrechner

Abb. 5.3: Konzept des Digitalen Schattens

6. Implementierung

Aufgrund der Fragen aus Abschnitt 5.2 wurde die Umsetzung dieser Arbeit in vier Meilensteine unterteilt, welche hier als Abschnitte angeführt werden: Digitales Modell, Schnittstellen, Interface und Digitaler Schatten. Das Digitale Modell konzentriert sich auf den Aufbau des Modells in Siemens NX sowie die Erzeugung und Steuerung der physikalischen Eigenschaften. Für die Interaktion zwischen Fertigungszelle, Interface und Siemens NX werden Schnittstellen benötigt, die sich auf den Datenaustausch konzentrieren. Der Schwerpunkt der Arbeit liegt auf der Entwicklung des Interfaces, das als Benutzerschnittstelle dient und auf den Schnittstellen zum Ein- und Auslesen von Daten aufbaut. Mit dem Auslesen von Echtzeitdaten aus der Fertigungszelle werden dann Digitales Modell, Schnittstellen und Interface zum Digitalen Schatten zusammengeführt. In allen Kapiteln wird zunächst versucht, einen Prototyp zu entwickeln, der sich nur auf die Funktionalität konzentriert. So kann untersucht werden, wie Funktionen und Anwendungen aufgebaut sein sollen. Mit den gewonnenen Erkenntnissen wird dann versucht, ein modulares System zu entwickeln, das die Aspekte der Flexibilität aus Abschnitt 5.2 widerspiegelt. Abbildung 6.1 spiegelt die geplante Vorgehensweise für die Umsetzung wider. Die Umsetzung des Digitalen Schattens in dieser Arbeit folgt ähnlichen Prinzipien wie die Implementierung des Digitalen Zwillings in der Arbeit von Wallner et al. [1]. Insbesondere die Verwendung von Standards wie ISO 23247 zur Strukturierung und Verwaltung der Daten hat sich als effektiv erwiesen.

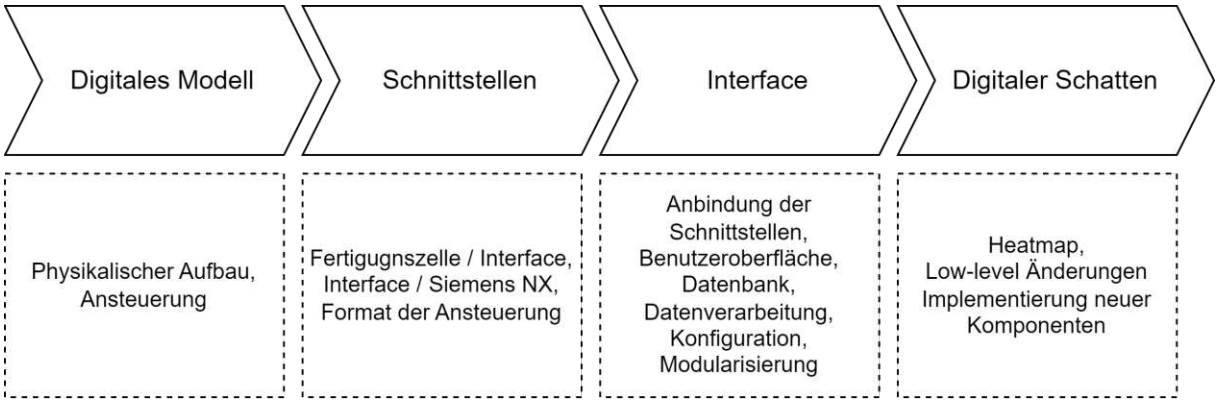


Abb. 6.1: Meilensteine

Die Implementierung folgt der in Kapitel 4 beschriebenen methodischen Vorgehensweise und stellt sicher, dass die in Kapitel 5 definierten Anforderungen erfüllt werden. Durch eine modulare Entwicklung wird sichergestellt, dass alle funktionalen Anforderungen in Teilschritten umgesetzt und validiert werden.

6.1 Digitales Modell

Das Digitale Modell bietet den Grundstein, auf dem der Digitale Schatten aufgebaut werden kann. Dieses inkludiert Abbilder von Maschinen und Objekten, einschließlich ihrer physikalischen Eigenschaften. In diesem Kapitel werden die Fragen, wie CAD-Dateien in die Simulation geladen und Bewegungsabläufe abgebildet werden können, untersucht. Die Herausforderung hierbei liegt an der Umsetzung von komplexeren Bewegungsabläufen wie dem Werkzeugwechsel der CM 55 sowie der Steuerung des IRB

120. Es ist üblich, dass Hersteller CAD-Dateien ihrer Maschinen und Komponenten bereitstellen, was auch hier der Fall war. Diese Dateien sind jedoch oft als vereinfachte Versionen zur Verfügung gestellt, weshalb für spezifische Anwendungen wie die Kollisionsprüfung es daher manchmal erforderlich ist, die Zweckmäßigkeit dieser Dateien zu überprüfen. Bei beiden Maschinen erwiesen sich diese CAD-Daten als geeignet.

6.1.1 Physikalischer Aufbau

Da die Maschinen bereits im Siemens NX Format verfügbar waren, konnten sie direkt in die Baugruppe geladen werden. Andere Elemente wie der Aufbau und selbst entwickelte Objekte wie das Magazin mussten importiert und in das entsprechende Format konvertiert werden. Nachdem alle CAD-Daten in Siemens NX importiert worden waren, konnten die physikalischen Eigenschaften wie die Volumenkörper und beweglichen Elemente definiert werden. Dafür wurde das MCD Modul von Siemens NX genutzt. Von beiden Maschinen konnten die mechanischen Spezifikationen aus den Datenblättern verwendet werden. Dabei wurde darauf geachtet, die einzelnen translatorischen und rotatorischen Bewegungen so genau wie möglich abzubilden und auf Maximalwerte durch Anschläge oder Bauweisen zu achten. In Abbildung 6.2 ist das Digitale Modell der Fertigungszelle zu sehen. Für alle Komponenten, welche vom IRB 120 erreichbar sind, wurden Kollisionen aktiviert.

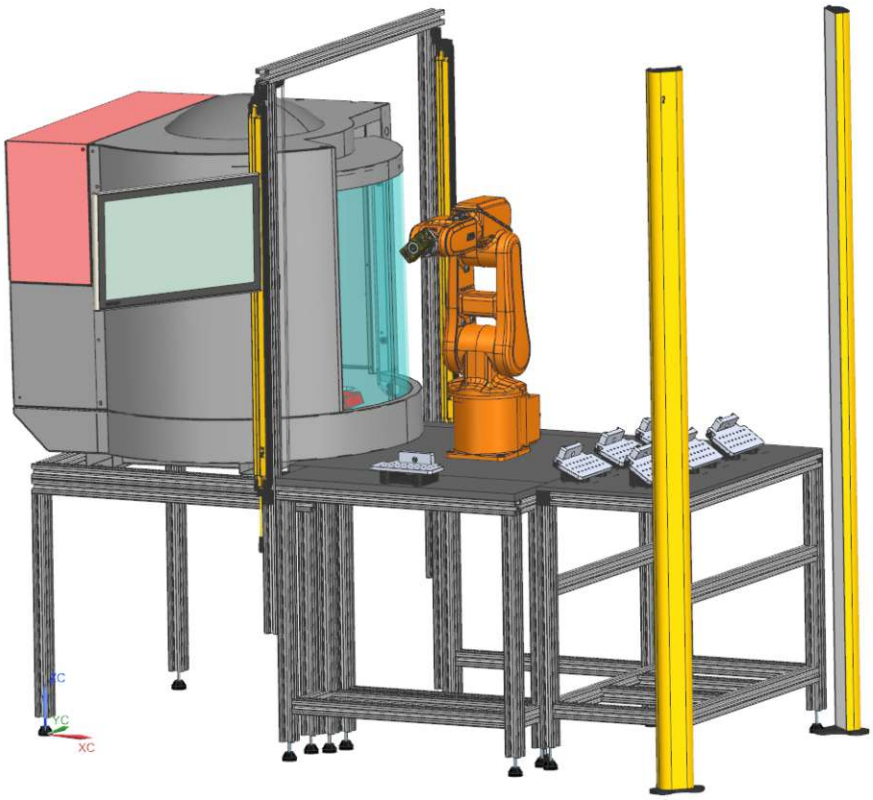
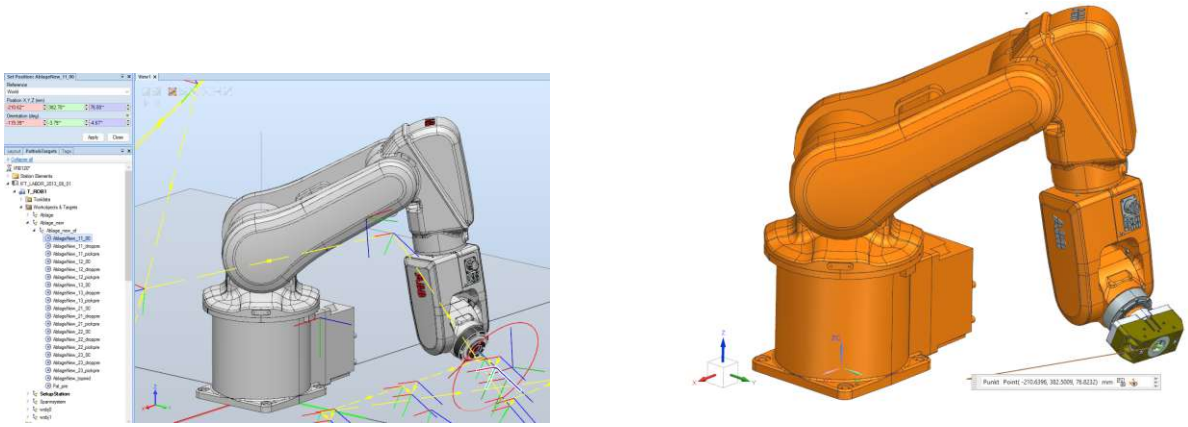


Abb. 6.2: Digitales Modell der Fertigungszelle in Siemens NX

Verifikation des Digitalen Modells

Um sicherzustellen, dass das Digitale Modell dem realen Roboter entspricht, wurden für den IRB 120 10 beliebig gewählte Punkte in RobotStudio angefahren. Die sechs

Achswerte wurden ausgelesen und in das Digitale Modell eingetragen. Der gemessene Punkt in Siemens NX wurde dann mit dem vorher definierten Punkt verglichen. Durch die Übereinstimmung konnte sichergestellt werden, dass das Modell korrekt abgebildet wird (Abbildung 6.3). Dasselbe Verfahren wurde für die CM 55 durchgeführt, wobei nur die drei linearen Achsen verglichen werden mussten. Für alle Punkte wurde die Abweichung gemessen und ein Maximalwert von 0,22 mm festgestellt.

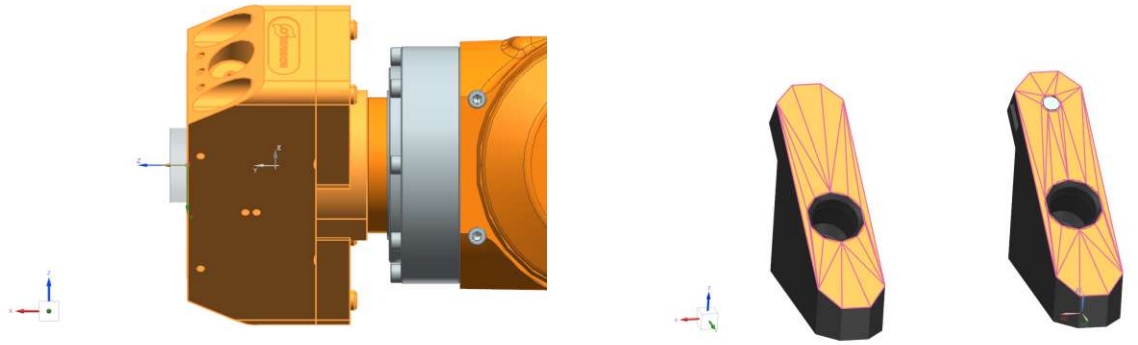


(a) RobotStudio (b) Siemens NX

Abb. 6.3: Überprüfung des Digitalen Modells des IRB 120

Vereinfachungen

Obwohl es theoretisch möglich ist, jede geometrische Form physikalisch als Kollisionskörper abzubilden, stößt man schnell an die Grenzen von Siemens NX. Da die Berechnung der Wechselwirkungen zwischen den Körpern auf mechanischen Grundlagen beruht, wurde die Berechnung bei komplexeren Elementen sehr schnell rechenintensiv. Selbst durch Optimierung der Simulationsparameter konnte eine Echtzeitsimulation nur durch starke Vereinfachungen des Modells erreicht werden. Insbesondere die Positionierung der Werkstückpaletten auf den Magazinen erforderte sehr viel Rechenleistung, sodass die Simulation bei voller Auslastung im Durchschnitt nur mit 50 % der Echtzeitgeschwindigkeit arbeiten konnte. Aus diesem Grund wurden Vereinfachungen vorgenommen, die möglichst wenig Auswirkungen auf das Ergebnis haben. Neben Vereinfachungen von Geometrien wurde versucht, die Zentrierbolzen der Werkstückpaletten von der physikalischen Berechnung auszunehmen. Das Magazin wurde mit einer Oberfläche mit sehr hohem Reibungskoeffizienten vereinfacht und das Loch für die Bolzen entfernt. Der NSR Mini-Greifer wurde durch einen virtuellen Vakuumgreifer angenähert (Abbildung 6.4), anstatt die mechanische Interaktion zwischen Bolzen und Greifer zu simulieren. Diese Vereinfachungen hatten Auswirkungen auf die Zentrierfähigkeit durch die Bolzen, was eine Inkonsistenz bei längeren Simulationen verursachen könnte. Ein Test hat jedoch gezeigt, dass diese Annäherung für einige Arbeitszyklen keine Auswirkungen auf die Positionierung hat. Wie in Abschnitt 5.2 definiert, wird besonders Wert auf eine flüssige Simulation gelegt, deswegen wurde entschieden, diese Vereinfachungen anzunehmen.



(a) Schunk NSR Mini (virtueller Vakuumgreifer - links)

(b) Magazin (Oberfläche mit hohem Reibungskoeffizient)

Abb. 6.4: Physikalische Vereinfachung

6.1.2 Ansteuerung

Grundsätzlich werden die meisten Bewegungen (Achsen, Gelenke) direkt über ihre Ist- bzw. Sollwerte in Siemens NX über Signale angesteuert und ausgelesen. Für Bewegungsabläufe, wie z. B. den Werkzeugwechsler der CM 55 oder die Steuerung des IRB 120, musste untersucht werden, ob eine direkte Weitergabe der Daten möglich ist oder ob eine spezielle Ansteuerung benötigt wird.

Werkzeugwechsel

Der Bewegungsablauf des Werkzeugwechslers (Abbildung 6.5) ist nicht ganz trivial, da die Werkzeuge grundsätzlich von oben gegriffen und dann senkrecht aus dem Magazin entnommen werden. Anders verhält es sich beim Werkzeug in der ersten Position, das mit einer Schwenkbewegung entnommen wird. Um diesen Ablauf darzustellen, wurde versucht, die Sequenzfunktion von Siemens NX zu verwenden. Es stellte sich jedoch schnell heraus, dass dies nur sehr eingeschränkt möglich ist und dass diese Methode nicht sehr gut mit globalen Variablen bzw. Signalen umgehen kann. Somit lässt sich zwar der Bewegungsablauf erstellen, dieser aber nicht mit dem MCD Server verbinden. Daher wurden nur die Ansteuerungen der Motoren definiert und die gesamte Logik des Bewegungsablaufs dem Interface überlassen.

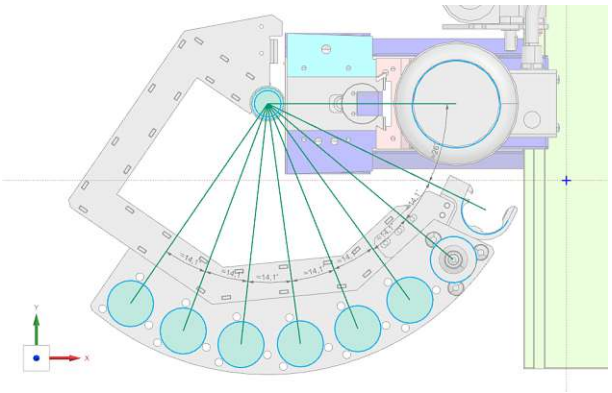


Abb. 6.5: Werkzeugmagazin der CM 55 (CAD)

Knickarm-Roboter

Ein Knickarm-Roboter kann auf zwei verschiedene Arten gesteuert werden: über die Positionen der einzelnen Gelenke oder über absolute xyz-Koordinaten und inverse Kinematik. Unter inverser Kinematik versteht man in der Robotik die Berechnung der Endposition eines mehrachsigen Roboters mit Gelenken bei gegebener Zielposition und Zielausrichtung. Mithilfe geeigneter Steuersoftware können Positionen mit definierter Orientierung angefahren werden. Dabei ist zu beachten, dass es bei der Berechnung der Endposition keine eindeutige Lösung geben kann, sodass sich das Verhalten des Roboters je nach Berechnungsverfahren stark unterscheiden kann [28]. In der Regel werden diese Roboter über eine vom Hersteller entwickelte und auf den Roboter abgestimmte Steuerung angesteuert, die die Berechnung der inversen Kinematik übernimmt. Diese Arbeit umfasst den Aspekt der Planung und Simulation unabhängig von der realen Fertigungszelle, das heißt, es muss auch möglich sein, den IRB 120 ohne die reale Steuerung ansteuern zu können, in Zukunft beispielsweise über den Leitrechner. Ohne diese Steuerung werden nur kartesische Koordinaten und Orientierungen, aber keine Achswerte der Gelenke übermittelt.

Um dies umzusetzen, wurden drei verschiedene Ansätze untersucht: die Entwicklung einer Python-basierten Steuerung, die Verwendung der nativen Funktionen von Siemens NX und die Nutzung von RobotStudio.

Ansatz 1: Python-basierte Steuerung Bei diesem Ansatz wurde ein externes Python-Skript zur Berechnung der inversen Kinematik verwendet. Hierfür kam IKPy von Manceron [29] zum Einsatz, welches die Gelenkpositionen und Rotationsachsen des Roboters in einer URDF-Datei beschreibt. Während dieser Ansatz für einzelne Punkte gut funktionierte, traten bei Bewegungsabläufen erhebliche Probleme auf. Die Berechnung neuer Gleichungssysteme für jeden Punkt führte zu sprunghaften Bewegungen, und die fehlenden Informationen über Minimal- und Maximalwerte der Gelenke verursachten unvorhersehbare Bewegungsstopps. Trotz Versuchen, das Intervall für die Punktberechnung zu erhöhen, konnten keine zufriedenstellenden Ergebnisse hinsichtlich der Bewegungsflüssigkeit erzielt werden.

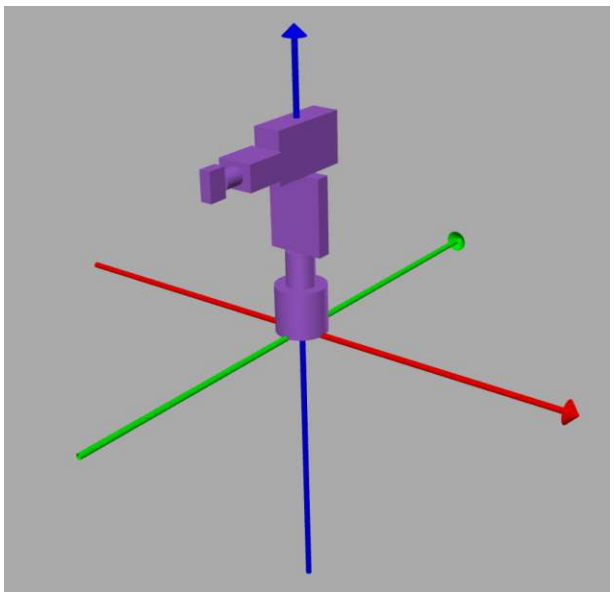


Abb. 6.6: URDF Modell des IRB 120 für inverse Kinematik

Ansatz 2: Siemens NX native Funktionen Der Mechatronic Concept Designer (MCD) von Siemens NX verfügt über eine native Funktion zur inversen Kinematik. Diese ermöglicht es, Zielpositionen des Roboters über die Eingabe von Koordinaten zu ermitteln. Siemens NX bietet hier zwei Modi: den Offline-Modus für vordefinierte Punkte und Bahnen sowie den Online-Modus für dynamische Punkte, die über Signale gesteuert werden können. Während der Online-Modus flüssige Bewegungen ermöglicht, konnte der Weg des Roboters nicht gesteuert werden, was zu unerwarteten Neupositionierungen und Kollisionen führte. Die Einführung von Zwischenpunkten verbesserte die Bewegungsflüssigkeit, jedoch blieben die Bewegungen ruckartig, besonders bei Intervallen unter 0,1 Sekunden.

Ansatz 3: Nutzung von RobotStudio RobotStudio, die Software von ABB, bietet eine virtuelle Integration der Robotersteuerung und kann Pfade automatisch berechnen. Der Vorteil dieser Methode liegt in der präzisen und realitätsnahen Simulation, da die Steuerungsalgorithmen des Herstellers verwendet werden. Ein Test zeigte, dass das Digitale Modell des IRB 120 in RobotStudio eine präzise und konsistente Bewegung auf einer geraden Linie ermöglichte. Allerdings würde dies bedeuten, dass ein weiteres Programm in das finale System integriert werden muss, was die Komplexität erhöht und die nahtlose Integration erschwert.

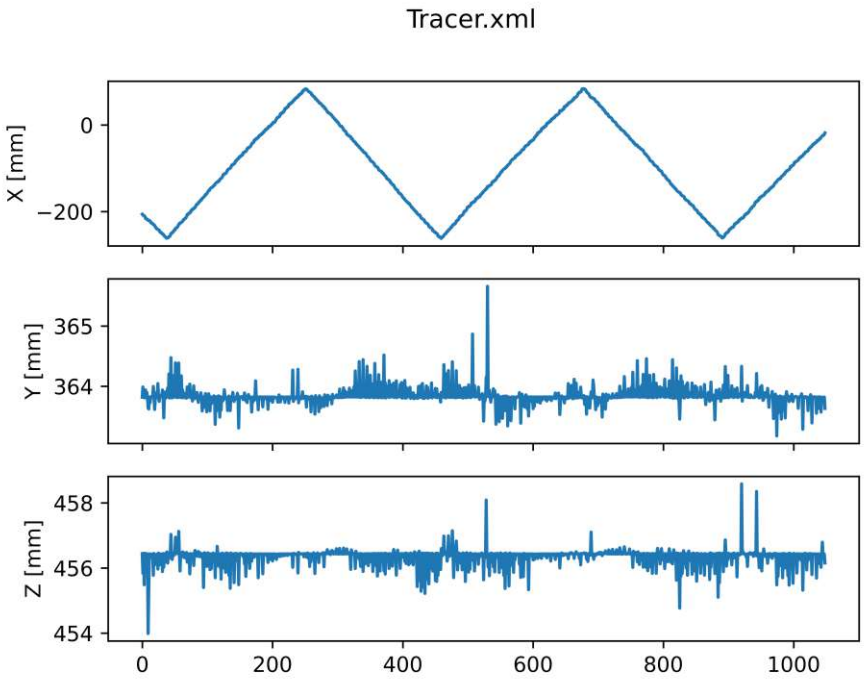


Abb. 6.7: Plot eines Tracers in Siemens NX

Ergebnisse und Entscheidung Keiner der drei Ansätze lieferte vollständig zufriedenstellende Ergebnisse. Die Python-basierte Steuerung scheiterte an der Flüssigkeit der Bewegungsabläufe und der Einhaltung der Gelenkgrenzen. Die native Funktion von Siemens NX zeigte Verbesserungen, blieb aber hinter den Anforderungen zurück. Daher wurde entschieden, auf die Nutzung von RobotStudio zurückzugreifen, um die Präzision und die flüssigen Bewegungsabläufe sicherzustellen. Über den in Siemens NX integrierten Tracer konnten die Werte der Achsen im Abstand von 0,01 s als XML

Datei aufgezeichnet und über ein Python-Programm geplottet werden (Abbildung 6.7). Es zeigte sich, dass beide Achsen erhöhte Ausschläge aufweisen. Eine Auswertung in Python zeigte die Minima und Maxima des Tracers, die mit den Werten der Achsen des RobotStudio MQTT Servers verglichen wurden (Tabelle 6.1). Hierbei konnte eine maximale Schwankung von 4,59 mm gemessen werden. Diese Schwankungen lassen sich dadurch erklären, dass der Roboter in Siemens NX nur durch die Achswerte gesteuert wird. Beim Übergang auf eine neue Position wird dadurch nicht der Weg, den der Roboter fährt, beachtet, sondern nur die Endposition. Dadurch kann es vorkommen, dass auf diesem Weg keine lineare Bewegung zustande kommt. Je höher die Frequenz der Werte und geringer die Geschwindigkeit des Roboters, desto genauer sind die Bewegungen. Die Schwankungen beziehen sich hierbei jedoch nur auf die Bewegungen und nicht auf die angefahrenen Punkte. Wie bereits bei der Überprüfung der CAD Modelle erwähnt, liegt die maximal gemessene stationäre Schwankung bei 0,22 mm.

	Tracer [mm]	**Recording [mm]**
Y min.	363.17	363.80
Y max.	365.66	363.92
Z min.	453.99	456.46
Z max.	458.58	456.61
Y Schwankung	2.49	0.12
Z Schwankung	4.59	0.15

Tab. 6.1: Testergebnisse Genauigkeit bei linearer Bewegung (*n=1200 Datenpunkte*)

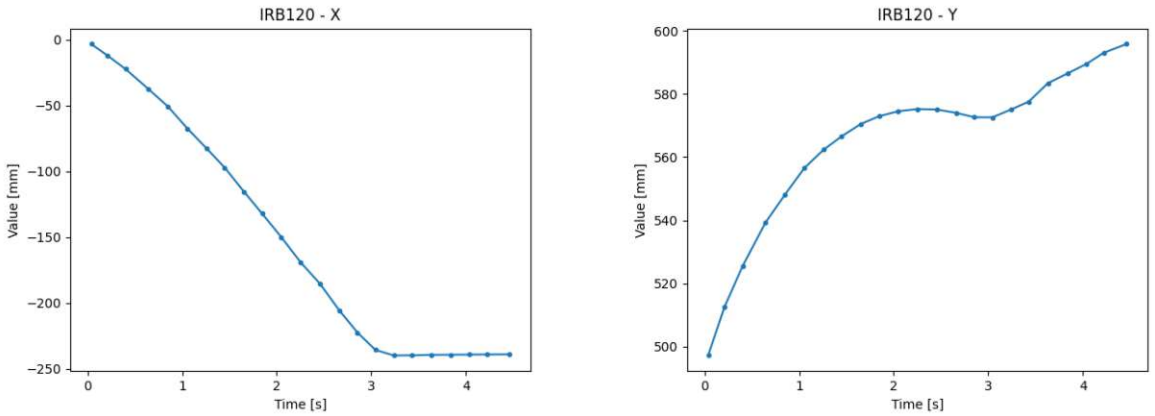
6.2 Schnittstellen

Um Daten zwischen Fertigungszelle, Interface und Siemens NX zu übertragen, sind Schnittstellen notwendig. Wie in Abschnitt 5.3 erörtert, gibt es die Möglichkeit, zwischen der Fertigungszelle und dem Interface über OPC UA, MQTT und REST zu interagieren. Die Datenübertragung zwischen Siemens NX und dem Interface beschränkt sich auf OPC UA und NX Open.

6.2.1 Fertigungszelle / Interface

Das von Thomas Trautner entwickelte Programm, welches die Schnittstelle über REST und MQTT ermöglicht, wurde als Erstes untersucht, da es bereits in vorherigen Projekten erfolgreich als Schnittstelle genutzt wurde. Es wurde entschieden, die MQTT Schnittstelle zu nutzen, da diese häufig in der Industrie verwendet wird und der Leitrechner in Zukunft unterstützt. Die Fertigungszelle wurde über ein Netzwerk an das Digitale Modell angebunden. Die Daten wurden erfolgreich übertragen, gleich fiel aber die niedrige Frequenz auf, in der die Daten aktualisiert werden (Abbildung 6.8). Gemessen wurde ein durchschnittliches Intervall von 0.2s. Dies war zu hoch, denn wie in Unterunterabschnitt 6.1.2 festgelegt wurde, ist ein Intervall von maximal 0.03s erforderlich. Nach längerer Analyse des Programms konnte festgestellt werden, dass die API-Schnittstelle des IRB 120 die hohe Latenz verursachte. Obwohl Verbesserungen

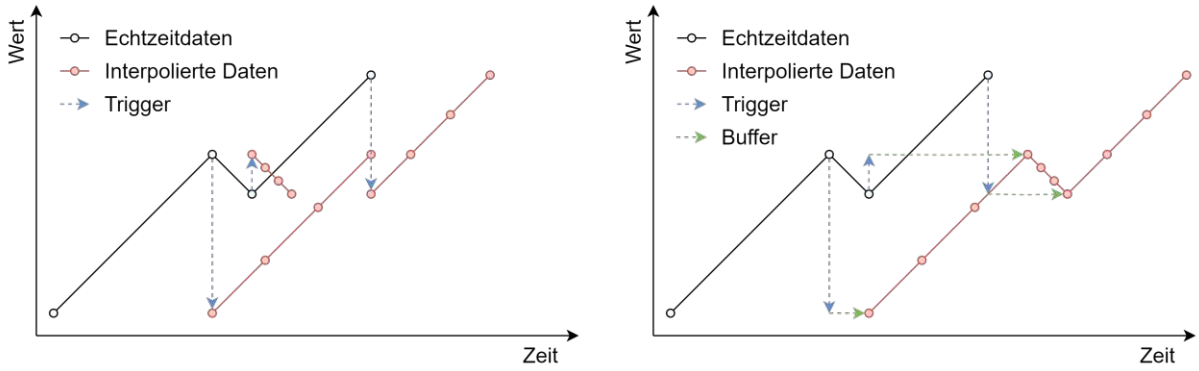
am Threading des Programms vorgenommen wurden, konnte das Intervall auf maximal 0.08s verringert werden. Als zweite Möglichkeit wurde ABB IoTGateway getestet, welches auch in der Lage ist, die Daten des Roboters über MQTT zu übertragen. Mit diesem Programm konnte ein Intervall von 0.05s erzielt werden, jedoch ist auch das zu hoch für die gesetzten Anforderungen.



(a) y-Achse (b) y-Achse

Abb. 6.8: IRB 120 MQTT Server Rohdaten

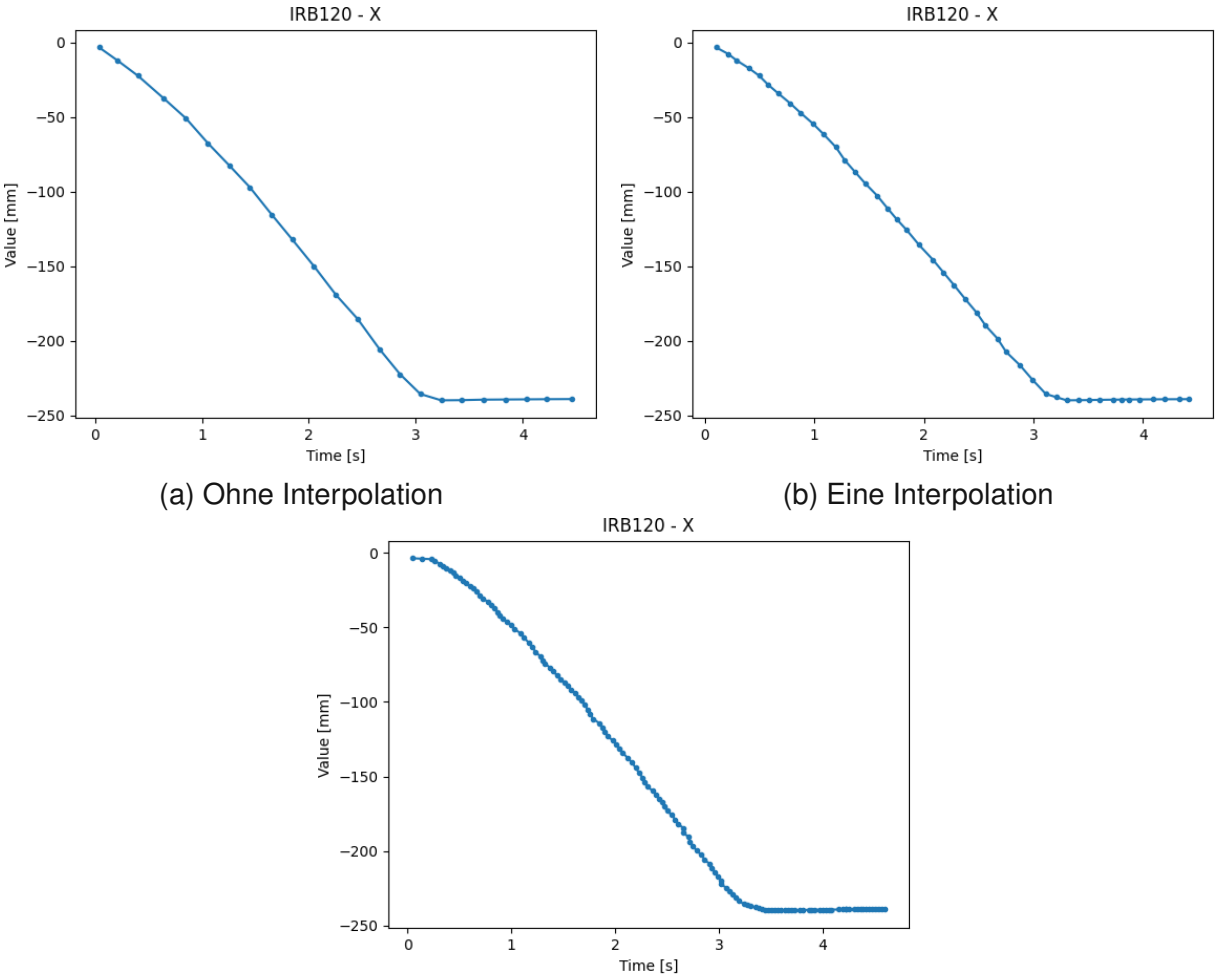
Um die visuelle Darstellung zu verbessern, wurde versucht, die Daten zu interpolieren, bevor an Siemens NX weitergegeben werden. Eine große Herausforderung stellte hierbei das variable Intervall dar, mit dem die Daten von der Fertigungszelle kommen. Hierfür wurde ein variabler Puffer entwickelt. Die Gesamtdauer zwischen zwei Datenpunkten wurde mithilfe eines Timers bestimmt. Durch eine Verzögerung mit einem festen Wert (z. B. 500ms) konnte für jeden Zwischenpunkt ein variabler Puffer berechnet werden. Für den ersten Punkt, der den Endwert der vorherigen Interpolation darstellt, war dies die Verzögerung minus der Gesamtzeit. Für jeden weiteren Zwischenpunkt wurde iterativ eine Zeit addiert, die sich aus der Anzahl der Iterationen und der Gesamtdauer ergab. Die Daten des MQTT Servers kamen mit variabler Frequenz, daher war ein variabler Puffer sehr wichtig, wie in Abbildung 6.9 ersichtlich. Ohne ihn hätten sich einerseits interpolierte Punkte überlappt und andererseits Zwischenpunkte gefehlt. Der in der Grafik genannte Trigger übergibt die interpolierten Daten an den Digitalen Schatten. Die Daten können natürlich erst übergeben werden, wenn der vollständige Pfad mit den interpolierten Punkten berechnet ist, was erst möglich ist, wenn Anfangs- und Endpunkt bekannt sind. Dabei ist auch ersichtlich, dass, wenn dies ohne Puffer (Abbildung 6.9a) geschieht, es zu genannten Problemen führt.



(a) Ohne Puffer (b) Variabler Puffer

Abb. 6.9: Interpolations-Puffer

Das Ergebnis der Interpolation mittels variablen Puffer ist in Abbildung 6.10 ersichtlich. Visuell war in Siemens NX nun kaum ein Unterschied zur Simulation in RobotStudio zu erkennen.



(a) Ohne Interpolation (b) Eine Interpolation

(c) 10 Interpolationen

Abb. 6.10: Interpolation von Daten-punkten (Achswert [mm] / Zeit [s])

6.2.2 Interface / Siemens NX

OPC UA und NX Open bieten beide die Möglichkeit Daten zu übertragen. Im ersten Schritt wird der Fokus auf die Übertragung von Zeitreihendaten gelegt, um die digitalen Maschinen zu steuern. Siemens NX bietet einen integrierten OPC UA Server, der vom Interface angesteuert werden kann. Die zweite Möglichkeit ist die Datenübertragung mittels NX Open. Um eine Entscheidung treffen zu können, ist zu beachten, wie diese beiden Schnittstellen in Siemens NX umgesetzt sind. Während OPC UA ein Industriestandard ist und Daten direkt an Signale der Simulation geknüpft werden können, bietet NX Open die Möglichkeiten auf low-level Kontrolle der Simulation. Es gibt jedoch einen großen Unterschied zwischen den beiden, und zwar läuft OPC UA auf einem Server asynchron im Hintergrund, während NX Open nur synchron ausgeführt werden kann. Das bedeutet, solange ein NX Open Skript aktiv ist, ist es nicht möglich, mit der Simulation zu interagieren. Außerdem werden Positionen von Maschinen erst am Ende das Skript aktualisiert, was eine Simulation unmöglich macht. Aus diesem Grund wurde OPC UA als Schnittstelle für die Datenübertragung von Zeitreihendaten ausgewählt. Aus Gründen der Übersichtlichkeit wurden die Signale der verschiedenen Maschinen in Nodes zusammengefasst, in diesem Fall CM 55 und IRB 120. Mit der Modularisierung, welche in Unterabschnitt 6.3.7 genauer beschrieben wird, mussten auch die Signale der Komponenten in Siemens NX angepasst werden. Die Signale einer Komponente wurden alle in einem Signaladapter zusammengefasst, sodass beim Laden der Komponente in eine Baugruppe ein Suffix mit einem fortlaufenden Index erzeugt wird. Beim Laden mehrerer Komponenten des gleichen Typs werden also die Namen der OPC UA Signale eindeutig benannt (z. B. *IRB120_1*, *IRB120_2*, ...).

Jedoch war es nicht nur notwendig, Zeitreihendaten zu übertragen, sondern auch die Konfigurationen und die Kontrolle über die Simulation. Hierfür konnte NX Open genutzt werden. Objekte können per Code geladen und positioniert werden. Auch das Starten und Stoppen der Simulation ist über NX Open möglich. Die Einschränkung, dass jeglicher Code auf dem Siemens NX Thread synchron läuft, bedeutet aber auch, dass grafische Interfaces Siemens NX blockieren. Dies bedeutet, dass das Interface nicht direkt in Siemens NX gestartet werden kann und somit auch nicht auf die Funktionen von Siemens NX Zugriff hat. Um dieses Problem zu lösen, können NX Open Funktionen als Skripte ausgeführt werden, welche mit dem Interface kommunizieren müssen. Für diese Kommunikation wurde MQTT gewählt, denn durch die Schnittstellen zwischen Fertigungszelle und Interface ist bereits ein MQTT Server verfügbar, der genutzt werden kann.

Um zu testen, ob die Steuerung über einen OPC UA Server möglich ist, wurde ein Programm geschrieben, das die Werte des Servers in einem bestimmten Intervall aktualisiert. Für den Test wurde das erste Gelenk (Rotation des gesamten Arms) des IRB 120 ausgewählt, da es die größte Bewegung verursacht. Die Bewegung wurde als Rotation um 90° in 2 Sekunden definiert. Die einzelnen Schritte in Grad wurden entsprechend dem Intervall berechnet. Schritt für Schritt wurde das Intervall reduziert und die visuelle Darstellung dokumentiert. Die Abtastrate des OPC UA Servers in Siemens NX wurde auf das Intervall angepasst, um möglichst flüssige Ergebnisse zu erhalten. Ab einem Intervall von 0,03 s wurde die Simulation als flüssig empfunden, ab 0,02 s sehr flüssig. Der weitere Unterschied zu 0,01 s war visuell nicht mehr wahrnehmbar.

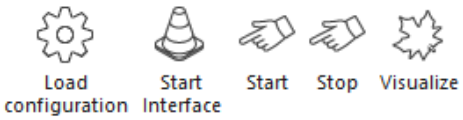


Abb. 6.11: Benutzerdefinierte NX Open Befehle

Befehle zur Steuerung des Digitalen Schattens wurden direkt grafisch in Siemens NX integriert (Abbildung 6.11). Mit dem Button *Start Interface* wird das Interface direkt aus Siemens NX heraus gestartet, sodass es nicht notwendig ist, das Programm manuell über die Kommandozeile aufzurufen. Das Interface wird jedoch trotzdem in einer neuen Python-Instanz gestartet, um oben beschriebene Probleme der Asynchronität zu vermeiden. Die Funktionen *Start* und *Stop* starten bzw. stoppen die Simulation und senden diese Information über MQTT an das Interface. *Load configuration* öffnet ein Fenster, in dem eine Konfiguration ausgewählt werden kann. Die Schaltfläche *Visualize* startet die Erzeugung der Heatmap, welche später in Abschnitt 7.4 beschrieben wird. Ein vollständiger Benutzerablauf des Programms ist in der Abbildung 6.12 dargestellt. Es ist jedoch auch möglich, zuerst das Interface und dann Siemens NX zu starten, da beide Programme unabhängig voneinander arbeiten.

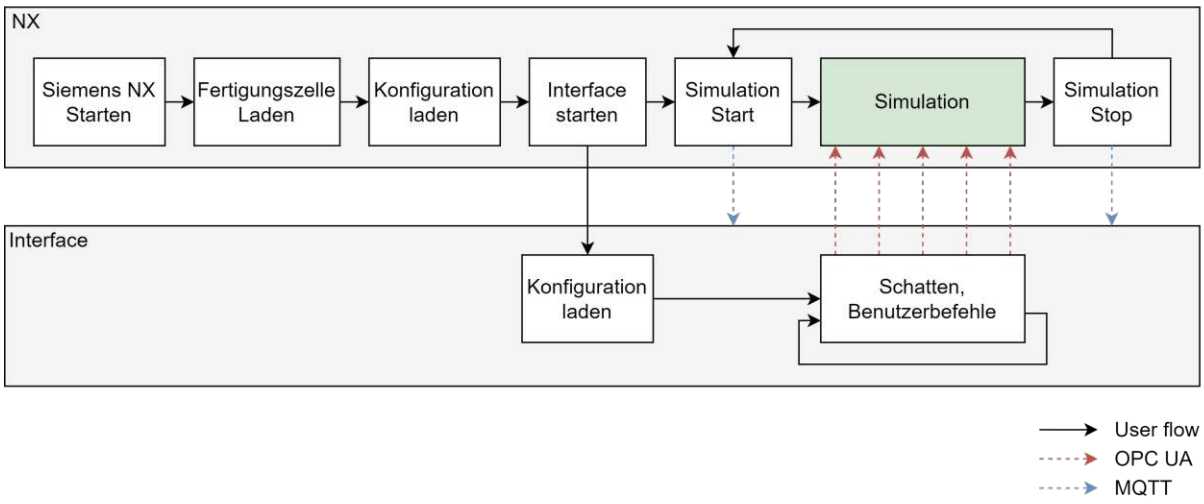


Abb. 6.12: Siemens NX Benutzerablauf

6.2.3 Format der Ansteuerung

Es stellte sich die Frage, in welchem Format die Daten vom Leitrechner kommen sollen. Es wurden zwei verschiedene Lösungen untersucht, die kontinuierliche Datenübertragung und die ereignisgesteuerte Datenübertragung.

Kontinuierliche Datenübertragung Hier werden Ist-Wert von Maschinen oder Sensoren in Echtzeit, meist in sehr kurzen Zeitabständen, übertragen. Je nach verwendeter Schnittstelle werden die Daten entweder in einem definierten Intervall abgefragt (OPC UA) oder bei Änderungen publiziert (MQTT). Beide Methoden führen dazu, dass die Daten in kurzen Zeitabständen eintreffen. Dies hat zur Folge, dass Simulationen sehr gut analysiert werden können, da exakte Rohdaten der Maschinen vorliegen, aber bei größeren Zeitabständen keine flüssige Darstellung in Echtzeit ermöglicht wird.

Ereignisgesteuerte Datenübertragung Bei dieser Methode werden keine aktuellen Maschinenkoordinaten übertragen, sondern nur die Informationen des Befehls. Man kann sich dies wie eine CNC-Programmierung vorstellen, bei der keine einzelnen Koordinaten, sondern Befehle wie z. B. eine lineare Bewegung von einem Punkt zu einem anderen Punkt vorgegeben werden. Der Vorteil ist eine flüssige Simulation, da die Bewegungen im Computermodell erzeugt werden. Nachteil ist eine nicht ganz exakte Abbildung des realen Systems, da Bewegungen von Beschleunigungen und Lasten abhängen. Diese können zwar nachgebildet werden, garantieren jedoch keine exakte Übereinstimmung. Ein weiterer Nachteil ist die wesentlich kompliziertere Steuerungslogik, da alle Bewegungsformen entweder im Interface oder in Siemens NX definiert werden müssen.

Auswahl der Methode Für die in dieser Arbeit betrachteten Anwendungsfälle wurde die kontinuierliche Datenübertragung gewählt. Obwohl die ereignisgesteuerte Datenübertragung den Vorteil einer flüssigeren Simulation bietet, ergaben sich während der Umsetzung erhebliche Herausforderungen, die ihre Anwendung in diesem Kontext erschwerten. Wie bereits im Unterunterabschnitt 6.1.2 beschrieben, war es im Rahmen dieser Arbeit nicht möglich, die inverse Kinematik unter den gegebenen Anforderungen erfolgreich zu implementieren. Zusätzlich erfordert die Steuerung über Befehle eine komplexere Implementierungslogik, was insbesondere bei der Integration neuer Komponenten den Entwicklungsaufwand erheblich erhöht. Aufgrund dieser Einschränkungen wurde die kontinuierliche Datenübertragung bevorzugt, da sie eine direkte Übermittlung von Ist-Werten der Maschinen oder Sensoren ermöglicht. Diese Methode stellt zwar höhere Anforderungen an die Echtzeitübertragung, erlaubt jedoch eine genauere Abbildung des realen Systems und eine effizientere Umsetzung der vorliegenden Arbeit.

6.2.4 Verifikation der Schnittstellen

Die Verifikation der Schnittstellen erfolgte durch gezielte Tests mit definierten Testnachrichten. Dabei wurde überprüft, ob alle Sensor- und Steuerdaten korrekt übertragen werden. Insbesondere wurde sichergestellt, dass die Datenübertragung über OPC UA und MQTT stabil funktioniert und keine unerwarteten Paketverluste auftreten.

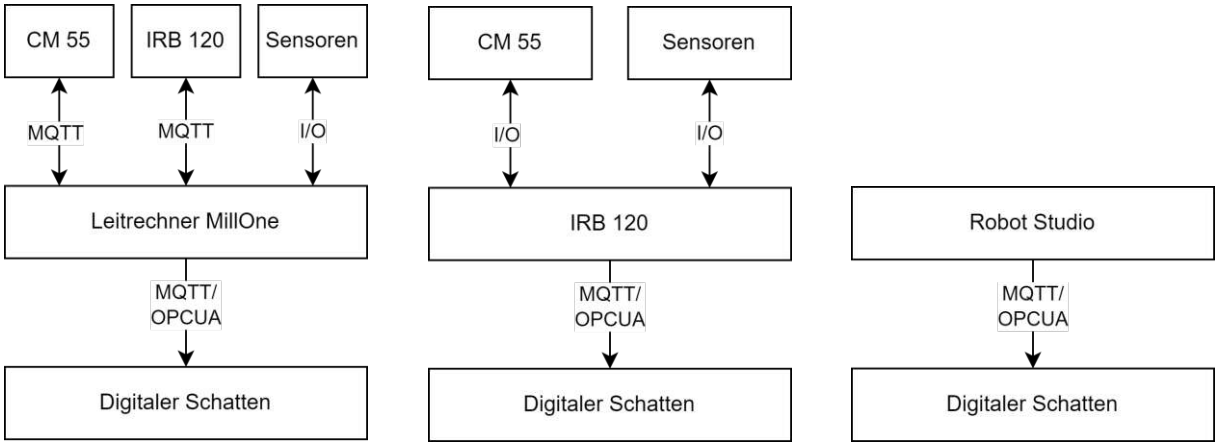
6.3 Interface

Das Interface bildet den Kern der Arbeit. Bei der Entwicklung wurde besonderer Wert auf den Aspekt der Flexibilität gelegt, sodass Wiederverwendung und Umstrukturierung der einzelnen Komponenten (IRB 120, CM 55, Server, Clients) ohne großen Aufwand möglich sein sollen. Um das zu erreichen, wurde auf objektorientierte Programmierung mit logischen und eindeutigen Schnittstellen geachtet. Komponenten sollen auch in mehreren Instanzen arbeiten können, z. B. in einer Fertigungsstraße mit mehreren Komponenten desselben Modells.

6.3.1 Anbindung der Schnittstellen

Oberste Priorität des Interfaces ist die Verarbeitung und Weitergabe der Daten. Dafür wurde im ersten Prototypen versucht, die Daten, wie in Abschnitt 6.2 beschrieben, welche via MQTT vom Interface kommen, mittels OPC UA (Unterabschnitt 6.2.2)

an Siemens NX weiterzugeben. Um mit dem Programm live interagieren zu können, ist eine asynchrone Datenverarbeitung notwendig. Hierfür wurden die Python-Pakete *asyncio_mqtt* und *asyncua* verwendet.



(a) Fertigungszelle mit Leitrechner

(b) Fertigungszelle ohne Leitrechner

(c) Entwicklung ohne physische Fertigungszelle

Abb. 6.13: Anbindung der Schnittstellen

In Abbildung 6.13 ist der Versuchsaufbau dargestellt. In der vollständigen Integration ist der Leitrechner für die Kommunikation zwischen Maschinen, Sensoren und Digitalen Zwilling zuständig. Ohne Leitrechner kommuniziert der Digitale Schatten mit dem IRB 120, welcher über I/O-Schnittstellen mit den Sensoren und der CM 55 verbunden ist. Während der Entwicklung werden RobotStudio und ein lokaler MQTT und OPC UA Server verwendet, um nicht immer mit der physischen Fertigungszelle verbunden zu sein. Der IRB 120 sowie die I/O-Signale der Sensoren und der CM 55 können auch in RobotStudio simuliert werden.

6.3.2 Benutzeroberfläche

Abb. 6.14: Erste Version des Python Interfaces

Ein Ziel der Arbeit ist es, eine Benutzeroberfläche zu erstellen, die den Digitalen Schatten steuert und essenzielle Daten anzeigt. Auch während der Entwicklung ist ein Programm mit grafischer Oberfläche hilfreich, um Fehler zu finden und Tests durchzuführen. Hierfür wird PySimpleGUI verwendet[30]. Der erste Prototyp (Abbildung 6.14) zeigte aktuelle Daten des Digitalen Schattens an und bot die Möglichkeit, manuell Werte zu setzen. Im nächsten Schritt wurden die Anforderungen an das Interface auf die Funktionen der grafischen Oberfläche übertragen:

- **Manual**: Manuelle Steuerung der einzelnen Antriebe wie Achsen, Gelenke, Türen usw.
- **Simulation**: Platz für vordefinierte Simulationen, die ohne Verbindung zum realen Modell ablaufen sollen.
- **Shadow**: Modus, um das Interface im Shadow-Modus laufen zu lassen. Echtzeitdaten sollen in den Digitalen Schatten eingelesen werden.
- **Recordings**: Da Bewegungsabläufe in einer Datenbank gespeichert werden sollen, können hier die Programme abgespielt werden
- **Configuration**: Ein Menü zum Hinzufügen und Aktualisieren von Konfigurationen.

Im Laufe der Entwicklung wurden einige Verbesserungen gemacht. Anstatt die Serveradressen manuell einzugeben, wurde eine Konfigurationsdatei erstellt. Die Verbindung zu den Schnittstellen wurde automatisiert, sodass das Programm immer versucht, eine Verbindung herzustellen, ohne manuelle Interaktionen. Zusätzlich verfügt die Oberfläche über ein Statusfenster, das die wichtigsten Informationen bezüglich Verbindung und Status des Digitalen Schattens anzeigt und die Fehlersuche erleichtert. Es

wird angezeigt, ob der Siemens NX OPC UA Server und der Shadow Server verbunden sind, ob die Simulation in Siemens NX läuft und ob der Shadow Modus aktiviert ist. Die finale Version unterstützt ein modulares System, sodass je nach Konfiguration direkt alle Oberflächen der Komponenten geladen und angezeigt werden (Abbildung 6.15). Auf den Aspekt der Module wird in Unterabschnitt 6.3.7 genauer eingegangen.

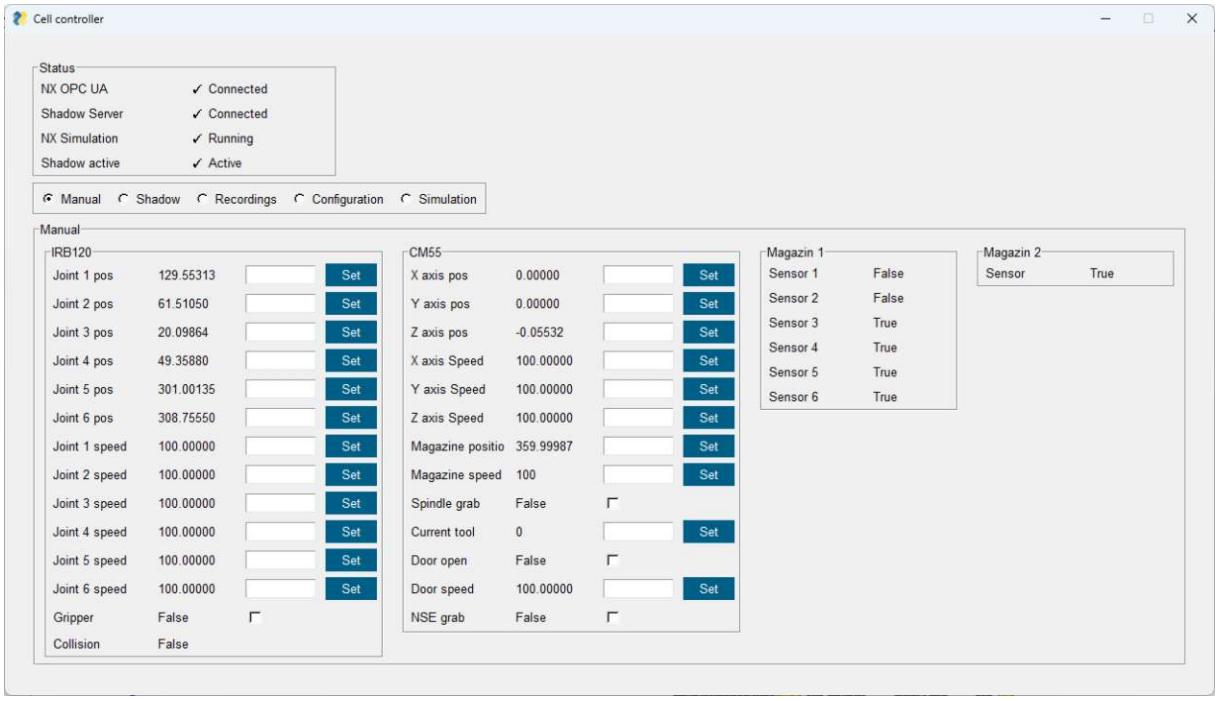


Abb. 6.15: Finale Version der Benutzeroberfläche

6.3.3 Datenbank

Die Datenbank dient zur Speicherung aller Konfigurationen, insbesondere der Positionen der Komponenten, und der Zeitreihendaten. Die in Abschnitt 5.5 ausgewählte Datenbank SQLite wurde in Python mittels SQLite3 in das Programm integriert. Für alle Komponenten wurden Tabellen angelegt, die einer Haupttabelle *recording* untergeordnet sind. Zusätzlich gibt es eine eigenständige Tabelle *settings*, in der Benutzerdaten wie z. B. die Serveradresse gespeichert werden. Zum Zeitpunkt der ersten Version war noch nicht ganz klar, welches Format für die Ansteuerung (Unterabschnitt 6.2.3) gewählt wird, daher wurden vorerst beide Formate (ereignisgesteuerte und kontinuierliche Datenübertragung) implementiert. Die komplette Struktur der Datenbank ist in Abbildung 6.16 zu sehen. Jede Maschine ist dabei Tabellen zugeordnet, die wie folgt beschrieben werden können:

- **configuration**: Zuständig für Position und Ausrichtung der Komponente
- **point_recording**: Für alle Daten der kontinuierlichen Datenübertragung
- **command_recording**: Für alle Daten der ereignisgesteuerte Datenübertragung

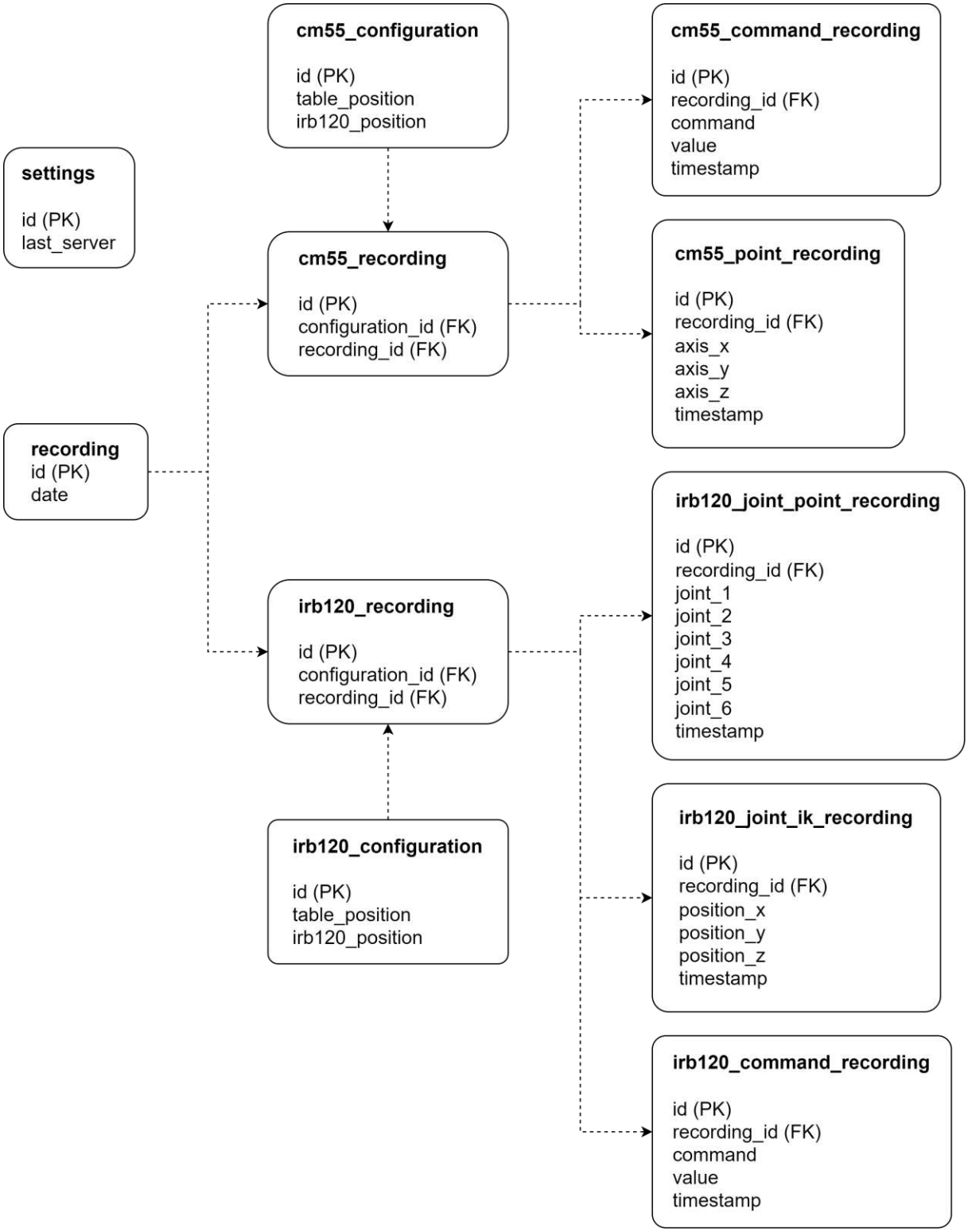


Abb. 6.16: Datenbankstruktur

Durch die Erkenntnis, dass alle Werte in Form von Zeitreihen vom Leitrechner kommen (Unterabschnitt 6.2.3), konnte auf alle *command_recording* Tabellen in der Datenbank verzichtet werden. Dies führte dazu, dass die Tabellen in den einzelnen Komponenten sehr ähnlich waren und Optionen zur Optimierung boten. Mit einem völlig neuen Ansatz, alle Komponenten zusammenzufassen und damit die Integration neuer Komponenten drastisch zu vereinfachen, wurde ein neues Schema entwickelt (Abbildung 6.17). Hierbei wird die Konfiguration als XML-Datei von der Datenbank entkoppelt

und nur noch referenziert. Diese Entscheidung wird in Unterabschnitt 6.3.5 genauer erklärt. Die neue Tabelle *time_series* übernimmt die Erfassung aller Zeitreihendaten in Form des JSON Formats. Das bedeutet, dass bei der Erstellung einer neuen Komponente keine neuen Tabellen angelegt werden mussten. Auch die Tabelle *settings* war hinfällig, da die Einstellungen auf eine Konfigurationsdatei übertragen wurden.

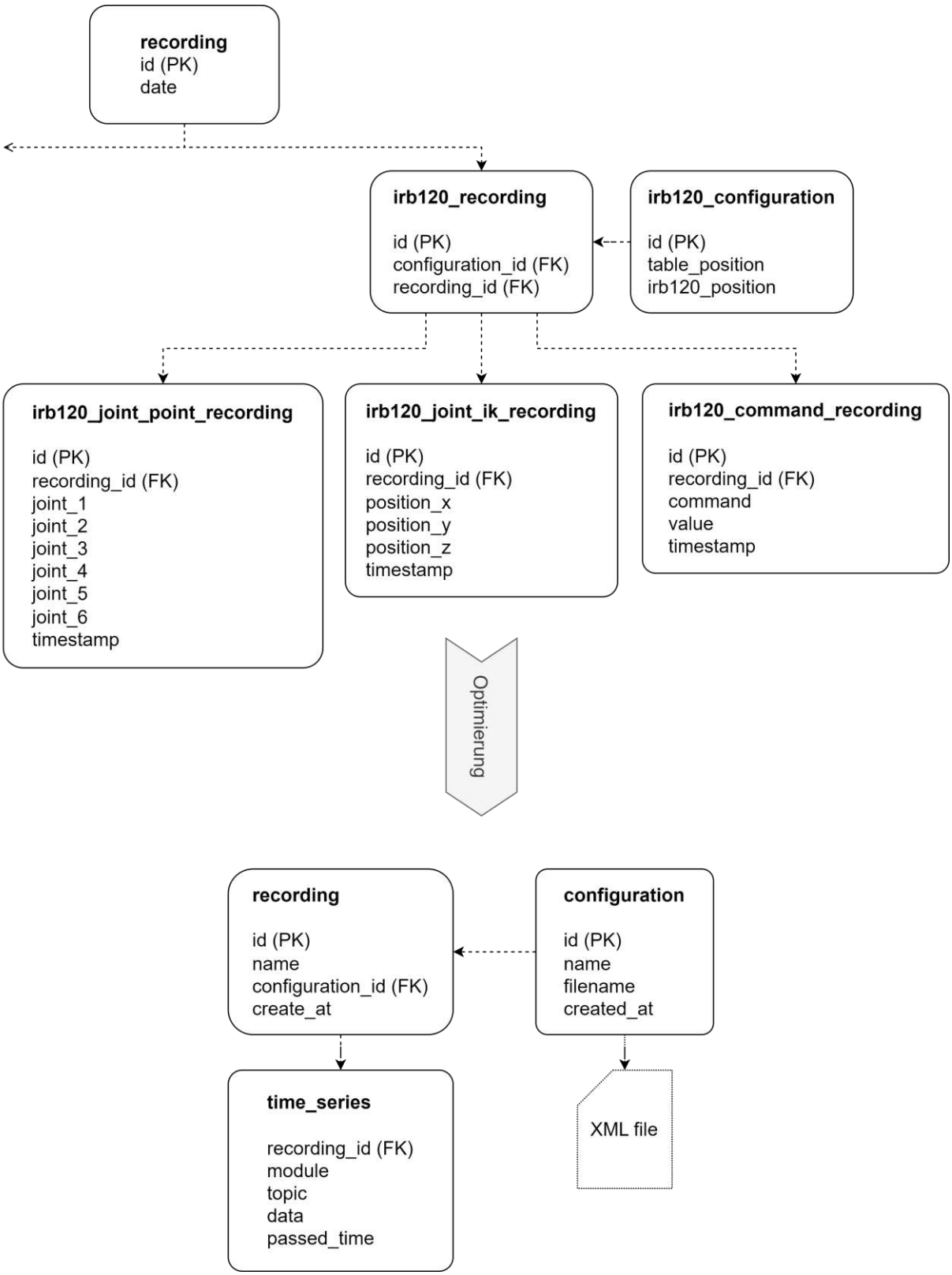


Abb. 6.17: Optimierung der Datenbank

6.3.4 Datenverarbeitung

In der ersten funktionsfähigen Form diente das Modul der Datenverarbeitung (im Programmcode *Shadow* genannt) dazu, Signale zwischen Input und Output weiterzuleiten, wobei die meisten Signale nur weitergeleitet werden mussten, während andere vorher umgewandelt werden mussten. Vor allem komplexere Bewegungsabläufe beschrieben, mussten programmiert werden. Da wie im Unterabschnitt 6.1.2 beschrieben die gesamte Logik im Interface und nicht in Siemens NX gesteuert wird, mussten Bewegungsabläufe wie die Bewegungen für den Werkzeugwechsel programmiert werden. Zum Beispiel kommt als Input der Befehl *Werkzeug 4*, der als Output in eine Sequenz von Befehlen umgewandelt werden muss. Einziger Nachteil dieser Herangehensweise ist die Implementierung neuer Komponenten, da in jeder Komponente alle Ein- und Ausgänge manuell verknüpft werden müssen. Deswegen lag der Fokus auf die Vereinfachung dieser Steuerung.

Mapping Um die Datenverarbeitung zu vereinfachen, wurde versucht, ein System zu bauen, wo über ein vordefiniertes Mapping Daten automatisch verarbeitet werden können. Im Beispiel Listing 6.1 sollen die Daten, die unter dem MQTT Topic *axis-pos* als JSON Format empfangen werden, auf die einzelnen Achs-Positionen im OPC UA Server als Soll-Werte ausgegeben werden. Diese Werte brauchen keinerlei Modifizierungen, deswegen ist kein weiterer Code notwendig. Außerdem werden im Mapping die Endpunkte für die Ist-Werte angegeben, sodass diese in der grafischen Oberfläche angezeigt werden können.

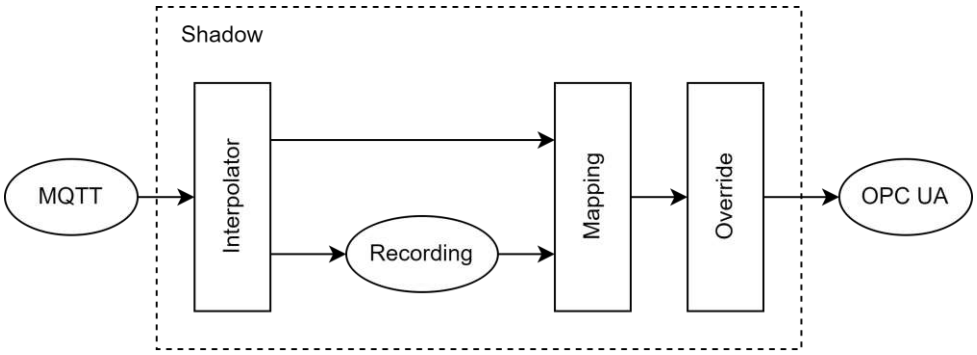


Abb. 6.18: Datenfluss im Interface

Der Datenfluss im Interface kann wie in Abbildung 6.18 beschrieben werden. Die eingehenden Daten durchlaufen vorerst den Interpolator, der im Unterabschnitt 6.2.1 beschrieben wurde. Danach kommen die Daten zum Mapping und zum Recorder (wenn die Daten aufgezeichnet werden). Nach dem Mapping können die Daten je nach Anwendungsfall überschrieben oder modifiziert werden. Diese Funktion wurde beispielsweise für die CM 55 verwendet, um das Signal zur Türöffnung von einem booleschen Wert in einen numerischen Wert für den Türwinkel umzuwandeln. Anschließend werden die Daten an den Siemens NX Server OPC UA übergeben. Mit der Nutzung des Interpolators schrumpft der benötigte Code für die einzelnen Maschinen um durchschnittlich 65 % und vereinfacht den Prozess, diese Klassen zu erstellen.

```
mapping = [
    ShadowTopic(
        mqtt_name="axis-pos",
        units="mm",
        nodes=[
            ShadowNode(
                display_name="X axis pos",
                opcua_get="position_x_ist",
                opcua_set="position_x_soll",
                default=0.0
            ),
            ...
        ]
    ),
    ...
]
```

Listing 6.1: Shadow Mapping

6.3.5 Konfiguration

Vor der Optimierung der Datenbank (Unterabschnitt 6.3.3) wurden die Positionierung und Ausrichtungen der einzelnen Komponenten in der Simulation in der Datenbank gespeichert. Dieser Ansatz erfüllte zwar den Zweck, war aber sehr unflexibel gegenüber der Integration neuer Komponenten. Wie bereits beschrieben, wurde entschieden, die Konfiguration aus der Datenbank herauszunehmen. Als erster Ansatz wurde versucht, die Konfiguration direkt in das Programm zu integrieren. Mit dieser Methode war es auch möglich, mehrere Instanzen einer Komponente mit unterschiedlichen Ein- und Ausgängen gleichzeitig zu laden. Ein Beispiel für die Definition eines Moduls direkt im Programmcode ist in der Listing 6.2 zu sehen.

```
modules = [
    Module(
        nx_module=(
            position=[1143.738, 401.0, 772.0],
            orientation=[[0.0, -1.0, 0.0],[1.0, 0.0, 0.0],[0.0, 0.0,
             ↪  1.0]]
        ),
        shadow=IRB120Shadow(
            receiver_name="irb120",
            opcua_name="IRB120"
        )
    )
]
```

Listing 6.2: Beispiel Interface Konfiguration in Python

Da sich dieser Ansatz sehr gut integrieren ließ, wurde entschieden, diesen weiterzuverfolgen. Wie in Abschnitt 5.2 definiert, wird angestrebt, genormte Datenstrukturen zu verwenden. Die ISO 23247 (Unterunterabschnitt 2.3) schlägt ein XML Format vor. Dafür werden Konfigurationen nicht mehr im Programm selbst geschrieben, sondern als XML Datei abgelegt. Dadurch entsteht die Möglichkeit, diese Dateien über das genormte XML Schema zu erstellen und zu ändern. Unter *MandatoryInformationAttribu-*

tes versteht man in der ISO 23247 alle Attribute, welche unbedingt bei jedem Element erforderlich sind. In dieser Implementierung ist es nur die ID. *OptionalInformationAttributes* beinhaltet demnach optionale Attribute. Da es verschiedene Möglichkeiten der Positionierung gibt, fallen auch diese unter die optionalen Attribute. Ein vereinfachtes Beispiel mit nur einem Modul ist in Listing 6.3 dargestellt. Hierbei ist die ID als erforderliches Attribut und die Informationen zur Position als optionale Attribute definiert.

```
<Modules>
    <IRB120Module>
        <MandatoryInformationAttributes>
            <ID>irb120</ID>
        </MandatoryInformationAttributes>
        <OptionalInformationAttributes>
            <Location>
                <Position>[1143.738, 401.0, 772.0]</Position>
                <Orientation>[[0.0, -1.0, 0.0], [1.0, 0.0, 0.0], [0.0,
                 ↪  0.0, 1.0]]</Orientation>
            </Location>
            <Shadow>
                <ReceiverName>irb120</ReceiverName>
            </Shadow>
        </OptionalInformationAttributes>
    </IRB120Module>
</Modules>
```

Listing 6.3: XML Konfiguration

Neben der absoluten Positionierung eines Objekts kann es auch relativ zu einem anderen Objekt platziert werden. Dies ist insbesondere dann sinnvoll, wenn das zu positionierende Objekt von einem Roboter o. ä. eingelesen und bei Bedarf aktualisiert wird. In den folgenden Beispielen (Listing 6.4 und Listing 6.5) werden die beiden Möglichkeiten zur Erstellung einer relativen Abhängigkeit gezeigt und die Unterschiede farblich hervorgehoben. Für die Positionierung werden entweder eine Position und eine Orientierung (<Position>, <Orientation>) oder drei Punkte (<PointsPosition>) benötigt. Die drei Punkte werden wie folgt behandelt: der erste Punkt als Ursprung, der zweite als Richtung der x-Achse, der dritte als Richtung der y-Achse. Zusätzlich kann ein Referenzsystem angegeben werden, das dazu dient, das Koordinatensystem vom Ursprung des Objekts zu verschieben (Beispiel in Abbildung 6.19). Dieses System wurde gewählt, da es die Möglichkeit bietet, mehrere Instanzen eines Objekts zu implementieren, ohne darauf achten zu müssen, dass jedes Objekt mit dem gleichen Koordinatensystem eingelernt wird. Würde man den Modellnullpunkt als Referenzsystem verwenden, müsste jede Instanz mit den gleichen Punkten angelernt werden, was insbesondere in engen Fertigungszellen oft unmöglich ist. Ist das Koordinatenreferenzsystem einmal definiert, kann die relative Position und Orientierung über die Python-Schnittstelle aktualisiert werden.

```
<OptionalInformationAttributes>
    <LocationRelationship>
        <With>irb120</With>
        <Position>[462.566, 71.6667, 51.6976]</Position>
        <Orientation>
            [[0.027887630136015702, 0.9996030777543804,
             ↪  -0.003995876545532892], [-0.9995874877996973,
             ↪  0.02791420439259338, 0.006756584005154197],
             ↪  [0.006865443881277728, 0.003805803081988296,
             ↪  0.9999619029769883]]
        </Orientation>
        <ReferenceSystem>
            <Position>[-13.5, 114.0, 40.0]</Position>
            <Orientation>[[0.0, -1.0, 0.0], [1.0, 0.0, 0.0], [0.0, 0.0,
             ↪  1.0]]</Orientation>
        </ReferenceSystem>
    </LocationRelationship>
    <UpdateAddress>
        ABB120/RAPID/T_ROB1/CalibData/SetupStation
    </UpdateAddress>
</OptionalInformationAttributes>
```

Listing 6.4: XML Konfiguration eines abhängigen Objektes mittels Position und Orientierung

```
<OptionalInformationAttributes>
    <LocationRelationship>
        <With>irb120</With>
        <PointsPosition>
            [[-210.623, 382.695, 76.804], [267.361, 386.156, 78.511],
             ↪  [-220.853, 603.089, 88.733]]
        </PointsPosition>
        <PointsReferenceSystem>
            [[64.00000039, -8.19037618, 74.58402475], [541.77187858,
             ↪  -3.93048053, 76.67451773], [53.55787858, 213.00251947,
             ↪  86.89651773]]
        </PointsReferenceSystem>
    </LocationRelationship>
    <UpdateAddress>
        ABB120/RAPID/T_ROB1/CalibData/SetupStation
    </UpdateAddress>
</OptionalInformationAttributes>
```

Listing 6.5: XML Konfiguration eines abhängigen Objektes mittels drei Punkten

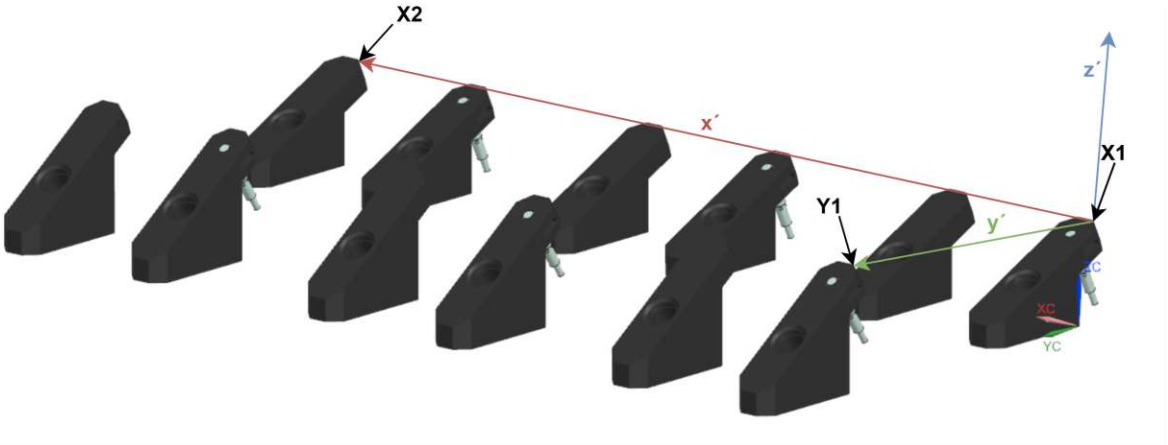


Abb. 6.19: Beispiel eines Referenzkoordinatensystems über drei Punkte

6.3.6 Verifikation der Software-Architektur

Die Implementierung des Interfaces wurde mit den initialen Designvorgaben verglichen. Dabei wurde überprüft, ob die Modulstruktur eingehalten wurde und alle Schnittstellen logisch und konsistent implementiert sind. Zur Verifikation wurden Code-Reviews durchgeführt sowie statische Analysen der Softwarearchitektur vorgenommen.

6.3.7 Modularisierung

Der von Zhang et al. [9] beschriebene Ansatz der fünfdimensionalen Modellierung (Abschnitt 2.2) und Rekonfigurierbarkeit betont die Bedeutung einer modularen Designstruktur sowie die Fähigkeit zur schnellen Anpassung an veränderte Bedingungen in der Fertigungsumgebung. Um diesen Ansatz im Programm umzusetzen, wurde die klassische Programmierung jedes einzelnen Moduls durch die Integration von Mappings (Unterabschnitt 6.3.4) ersetzt.

Durch diese Methode konnten die Komponenten erheblich vereinfacht werden, sodass neue Komponenten sehr einfach in das System integriert werden können. Abbildung 6.20 zeigt, dass vor der Modularisierung jedes Modul mehrere Funktionen hatte, die alle individuell definiert und programmiert werden mussten. Dies musste für jede neue Komponente, wie Roboter, Maschinen oder Objekte, durchgeführt werden.

Wie bereits in Unterabschnitt 6.3.3 erwähnt, wurde im Zuge der Modularisierung eine einheitliche Datenbank verwendet, sodass keine neuen Tabellen mehr für die einzelnen Komponenten erstellt werden mussten. Die individuellen grafischen Oberflächen wurden durch das Mapping mit einer generierten grafischen Oberfläche ersetzt, die nicht mehr manuell erstellt werden musste. Auch der Receiver wurde durch dieses Mapping ersetzt, da Signale nicht mehr per Programmierung weitergeleitet werden mussten. Durch diese Schritte konnte ein Großteil der manuell zu schreibenden Funktionen eliminiert werden, was die Integration neuer Komponenten erheblich vereinfachte.

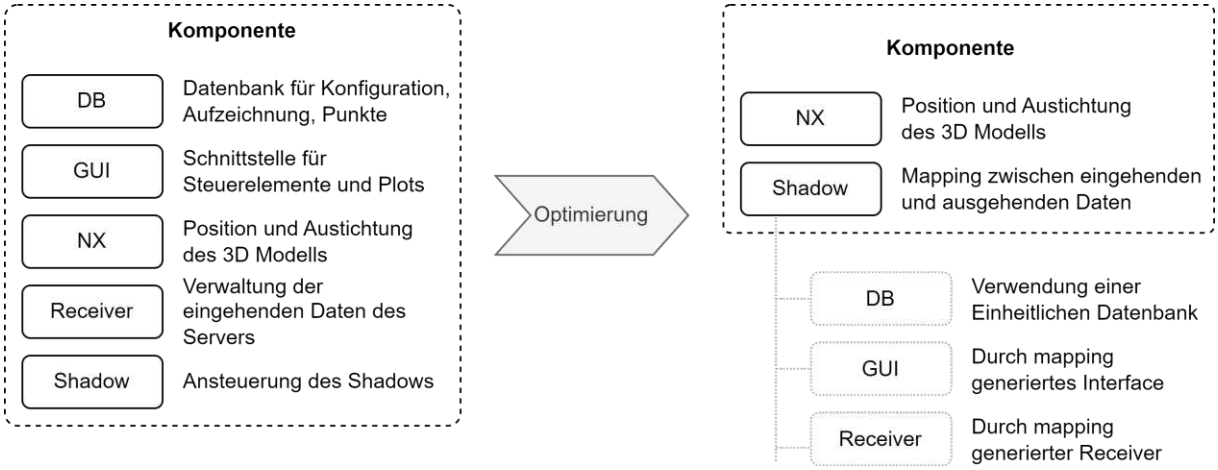


Abb. 6.20: Vergleich vor und nach der Modularisierung

7. Integration eines Digitalen Schattens

Bisher wurde das Programm ausschließlich mit der simulierten Fertigungszelle mittels RobotStudio verwendet. In diesem Kapitel wird die Verbindung des Programms mit der realen Fertigungszelle hergestellt. Darüber hinaus werden Funktionen des Digitalen Schattens, wie Low-Level-Änderungen im System, die Implementierung neuer Komponenten und die Entwicklung der Heatmap, behandelt.

7.1 Integration mit der realen Fertigungszelle

Die Verwendung von RobotStudio für die Simulation ermöglichte eine hardwareunabhängige Entwicklung, die sich durch hohe Flexibilität und Effizienz auszeichnete. Die Verbindung zwischen Interface und simulierter Fertigungszelle wurde bereits in Unterabschnitt 6.2.1 erfolgreich hergestellt und getestet. Für die Integration mit der realen Fertigungszelle wurde die Konfiguration ohne Leitrechner (Abbildung 6.13b) genutzt.

Die Anbindung an die reale Fertigungszelle erfolgte schrittweise und begann mit der Validierung der Schnittstellen. Zunächst wurde die Datenübertragung zwischen der realen Fertigungszelle und dem Interface überprüft. Hierbei wurden die Echtzeitdaten der Fertigungszelle in das Interface eingespeist und deren Integrität sowie Frequenz überprüft. Die Ergebnisse zeigen, dass die Datenübertragung stabil und zuverlässig ist und mit der Simulation übereinstimmt.

7.1.1 Verifikation der Systemintegration

Die Verifikation der Systemintegration stellt sicher, dass alle Module der Implementierung den geplanten Spezifikationen entsprechen. Zu diesem Zweck wurden folgende Prüfungen durchgeführt:

- **Verifikation der Datenübertragung:** Durch gezielte Bewegungen der realen Maschine wurde überprüft, ob alle Schnittstellen exakt die definierten Datenformate verwenden und keine Übertragungsverluste auftreten.
- **Verifikation der Echtzeitfähigkeit:** Die Datenübertragungslatenzen wurden gemessen und mit den geplanten Spezifikationen verglichen. Je nach Einstellung des Interpolators gab es eine fixe Latenz und es wurden alle Daten korrekt übertragen.
- **Verifikation der Maschinensteuerung:** Überprüft wurde, ob alle Bewegungsabläufe korrekt ausgeführt werden.

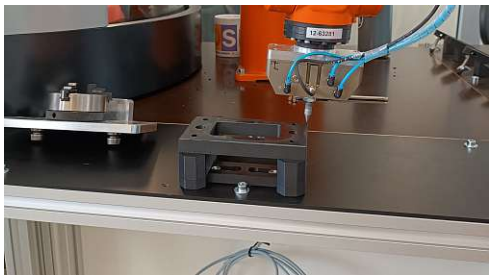
7.2 Low-level Änderungen

Neben der statischen Positionierung über die XML-Konfiguration gibt es auch die Möglichkeit, Objekte oder Maschinen relativ zu positionieren. Diese Funktion eignet sich besonders für eingelernte Positionen, d. h. Positionen, die von einem Roboter angefahren werden, um eine hohe Genauigkeit und Wiederverwendbarkeit zu erreichen. Am Beispiel des Magazins (Listing 7.1) wird die Vorgehensweise beim Einlernen erläutert. Wie bereits bekannt, wird das Modul über die XML-Datei erstmals händisch

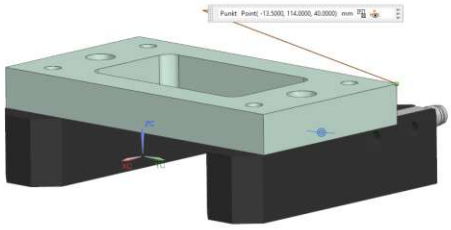
definiert. Statt einer *Location* wird nun eine *LocationRelationship* verwendet. Die folgenden Parameter werden für die relative Positionierung benötigt:

- **With**: Die *ID* des referenzierten Objekts.
- **Position**: Die relative Position zum Objekt, die beim Lernen überschrieben wird.
- **Orientation**: Die Rotation relativ zum Objekt als 3x3 Rotationsmatrix. Auch sie wird beim Anlernen neu definiert. Alternativ zu **Position** und **Orientation** kann auch **PointPosition** für die Definition der Positionierung verwendet werden, um drei Punkte statt einer Position und Orientierung anzugeben.
- **ReferenceSystem**: (Optional) Ein Referenzsystem beschreibt das Koordinatensystem, auf das referenziert wird, wie in Abbildung 7.1b ersichtlich. Im Beispiel des Magazins misst der Roboterarm nicht den Ursprung des Systems, sondern ein Referenzkoordinatensystem. Ist dieses einmal definiert, muss es beim erneuten Einlernen nicht neu definiert werden.
  - **Position**: Die Position des Referenz-Koordinatensystems relativ zum Ursprung.
  - **Orientation**: Die 3x3 Rotationsmatrix relativ zum Ursprung.

  Auch hier kann anstelle von **ReferenceSystem PointsReferenceSystem** verwendet werden.



(a) Abtasten der Eckpunkte



(b) Auslesen des Koordinatenursprungs

Abb. 7.1: Anlernen eines Referenzkoordinatensystems

```
<MagazineModule>
    ...
    <OptionalInformationAttributes>
        <LocationRelationship>
            <With>IRB120</With>
            <Position>...</Position>
            <Orientation>...</Orientation>
            <ReferenceSystem>
                <Position>
                    [8.0, -21.97493321, 59.22015401]
                </Position>
                <Orientation>
                    [[1.0, 0.0, 0.0], [0.0, 0.9972619, 0.0739504],
                     ↪  [0.0, -0.0739504, 0.9972619]]
                </Orientation>
            </ReferenceSystem>
        </LocationRelationship>
    </OptionalInformationAttributes>
</MagazineModule>
```

Listing 7.1: XML Komponente des Magazins mit Referenzkoordinatensystem

Ein Ziel dieser Arbeit war es, die Möglichkeit zu bieten, Änderungen möglichst einfach auf den Digitalen Schatten zu übertragen. Aus diesem Grund wurde versucht, den Vorgang des manuellen Aktualisierens der XML-Datei zu automatisieren.

7.3 Automatisierung der XML-Aktualisierung

Die vom IRB 120 verwendeten Daten für die Positionierung des Magazins werden beim Anlernen aktualisiert und sind über OPC UA am Roboter verfügbar. Diese Daten können genutzt werden, um die Position des Magazins in der XML-Datei automatisch zu beschreiben. Um dies zu erreichen, wurde ein Prozess entwickelt, der es ermöglicht, nach dem Anlernen der Magazine im Interface die Konfiguration zu aktualisieren. Dieser Prozess besteht aus mehreren Schritten:

1. **Anlernen der Magazine**: Zunächst wird die Position der Magazine manuell am IRB 120 eingelernt. Diese Positionen werden im OPC UA Server des IRB 120 gespeichert.

2. **Datenabfrage über das Interface**: Nach dem Anlernen kann im Interface die Aktualisierungsfunktion aufgerufen werden. In Abbildung 7.2 ist zu sehen, dass eine *UpdateAddress* angegeben werden kann, welche den Pfad im OPC UA Server widerspiegelt.

3. **Datenverarbeitung**: Das Skript kommuniziert mit dem OPC UA Server des IRB 120 und holt sich die aktuellen Positionsdaten der Magazine. Diese Daten umfassen genaue xyz-Koordinaten sowie die Orientierung der Magazine.

4. **Automatische XML-Erstellung**: Basierend auf den abgerufenen Positionsdaten generiert das Skript automatisch eine neue XML-Datei. Diese Datei enthält alle notwendigen Informationen zur aktuellen Positionierung der Magazine.

5. **Laden der neuen Konfiguration in Siemens NX**: Die neu erstellte XML-Datei wird dann in Siemens NX geladen, sodass die aktuelle Position der Magazine im digitalen Modell korrekt dargestellt wird.

```
<OptionalInformationAttributes>
    ...
    <UpdateAddress>
        ABB120/RAPID/T_ROB1/CalibData/SetupStation
    </UpdateAddress>
</OptionalInformationAttributes>
```

Listing 7.2: *UpdateAddress* in einer XML-Konfiguration

Diese Automatisierung reduziert den manuellen Aufwand erheblich und minimiert die Fehleranfälligkeit, die bei der manuellen Aktualisierung der XML-Dateien auftreten kann. Darüber hinaus gewährleistet sie, dass die Position des Magazins im Digitalen Schatten aktuell und korrekt bleibt, was für die Präzision und Zuverlässigkeit der Simulationen entscheidend ist. Zudem ermöglicht die Automatisierung eine nahtlose Integration neuer Komponenten und Anpassungen, was die Flexibilität und Anpassungsfähigkeit des Systems erhöht. Die Implementierung dieser Automatisierung verbessert somit die Effizienz und Genauigkeit des Digitalen Schattens erheblich. Diese Methode ist nicht nur auf den spezifischen Fall des IRB 120 beschränkt, sondern kann allgemein verwendet werden, um die Positionen und Konfigurationen verschiedener Roboter und Maschinen in einem digitalen Modell automatisch zu aktualisieren.

7.4 Heatmap

Die Heatmap-Funktion visualisiert die Häufigkeit, mit der ein Roboter, insbesondere sein Werkzeug, in seinem Arbeitsraum agiert. Diese wird durch ein dreidimensionales Gitter dargestellt, wobei die einzelnen Blöcke entsprechend ihrer Häufigkeit in ihrer Deckkraft variieren. Je satter die Farbe eines Blocks, desto häufiger befand sich der Roboter in diesem Bereich. Zur Visualisierung wurde ein Parameter eingeführt, der die Größe des Gitters bestimmt und vom Benutzer eingestellt werden kann. Ein Beispiel für die Heatmap bei der Aufnahme einer Werkstückpalette ist in Abbildung 7.2 zu sehen. Die Funktion kann in Siemens NX über den Button *Visualize* aufgerufen werden (Unterabschnitt 6.2.2). Bei der Erstellung der Heatmap ist darauf zu achten, dass in RobotStudio das richtige Werkobjekt ausgewählt wurde, da es sonst zu Verschiebungen kommen kann.

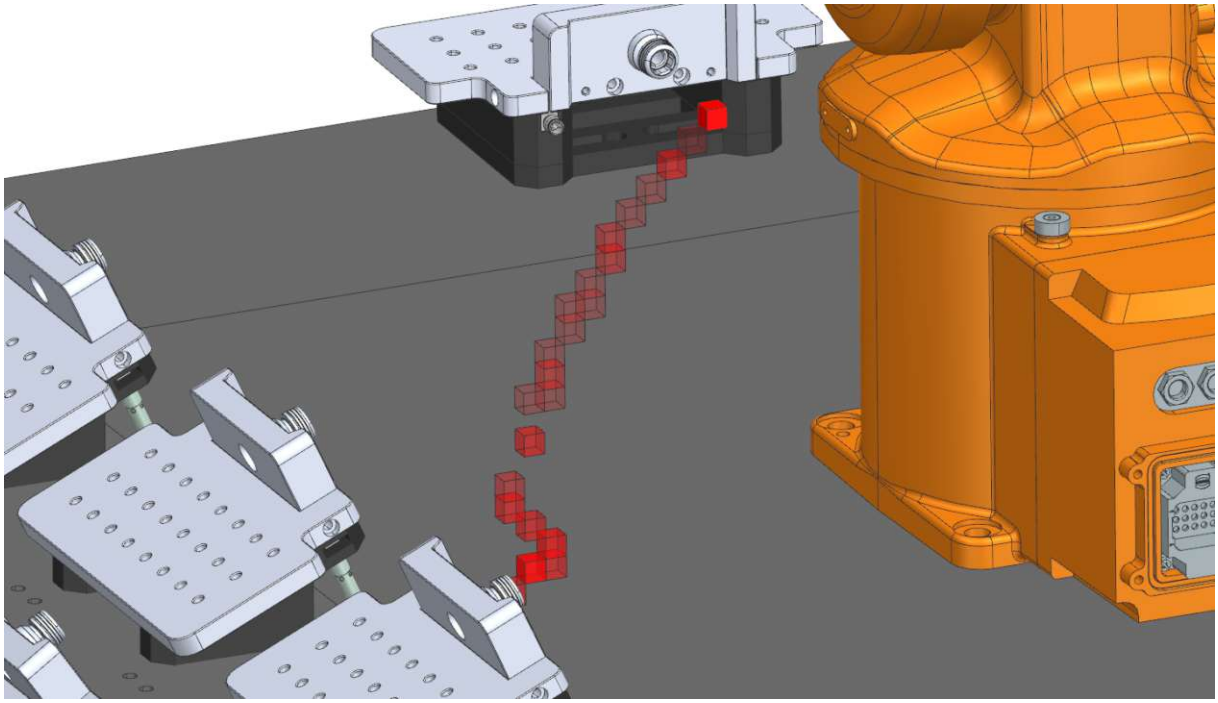


Abb. 7.2: Heatmap Beispiel: Aufnahme einer Werkstückpalette

7.5 Implementierung neuer Komponenten

Um eine neue Komponente zu erstellen, muss zuerst ein Name gewählt werden. Da der Name der Komponente die Ordner- und Komponentenstruktur bestimmt, sollte er im *Pascal Case Format* gewählt werden. Die XML-Konfiguration und das zugehörige Dateisystem können dem Listing 7.4 und dem Listing 7.3 entnommen werden. Das Interface ist so programmiert, dass Komponenten automatisch nach Namen geladen werden, daher ist eine korrekte Benennung von größter Bedeutung. Sobald eine Komponente implementiert ist, wird sie zum Git-Repository hinzugefügt und kann in Zukunft von jedem verwendet werden.

```
1 - src
2     - modules
3         - robot1
4             - robot1\robot1_nx.py -> class Robot1Nx
5             - robot1\robot1_shadow.py -> Class Robot1Shadow
```

Listing 7.3: Implementieren einer neuen Komponente ”Robot1” - Dateisystem

```
<Robot1Module>
    <MandatoryInformationAttributes>
        ...
    </MandatoryInformationAttributes>
    <OptionalInformationAttributes>
        ...
    </OptionalInformationAttributes>
</Robot1Module>
```

Listing 7.4: Implementieren einer neuen Komponente ”Robot1” - XML Konfiguration

8. Ergebnisse und kritische Betrachtung

8.1 Zentrale Erkenntnisse und Analyse

8.1.1 Integration von CAD-Dateien in Siemens NX

Eine der zentralen Fragen dieser Arbeit war, wie CAD-Dateien effizient in Siemens NX integriert werden können, um die physikalischen Eigenschaften der Fertigungszelle realistisch abzubilden. Die CAD-Dateien der Maschinenkomponenten, insbesondere des Knickarm-Roboters ABB IRB 120 und der Fräsmaschine EMCO CM 55, wurden erfolgreich importiert und angepasst. Dadurch konnten die physikalischen Eigenschaften und Bewegungsabläufe der Maschinen präzise simuliert werden. Besonders die komplexen kinematischen Bewegungen, wie Werkzeugwechsel und die Bewegungen des Knickarm-Roboters, stellten eine Herausforderung dar, die jedoch erfolgreich bewältigt wurde.

8.1.2 Sicherstellung der Flexibilität des Digitalen Schattens

Eine weitere zentrale Frage war, wie die Flexibilität des Digitalen Schattens gewährleistet werden kann, um physikalische Änderungen im realen System mit minimalem Aufwand zu aktualisieren. Dies wurde durch die Entwicklung eines modularen und objektorientierten Interfaces erreicht, das Änderungen schnell und effizient im Digitalen Schatten ermöglicht. Neue Maschinen und Objekte konnten problemlos integriert werden.

8.1.3 Erreichung einer flüssigen und präzisen Simulation

Die Frage, wie eine flüssige und präzise Simulation der Fertigungsprozesse erreicht werden kann, war von besonderer Bedeutung. Die kontinuierliche Datenübertragung über OPC UA erwies sich als die geeignetste Methode, um eine exakte Echtzeitabbildung der Maschinenzustände zu gewährleisten. Um die Datenpunkte in Echtzeit zu erhalten, wurde eine Interpolationstechnik verwendet. Diese Technik ermöglichte es, zwischen den empfangenen Datenpunkten zusätzliche Werte zu berechnen, um eine flüssigere Bewegung darzustellen. Ein variabler Puffer wurde eingeführt, der die Zeit zwischen zwei empfangenen Datenpunkten berücksichtigte und entsprechend interpolierte Werte erzeugte. Die Ergebnisse, welche nochmals in Tabelle 8.1 dargestellt sind, zeigen eine deutliche Verbesserung durch die Interpolation.

Intervall (s)	Visuelle Qualität	Max. Schwankung (mm)
0.08	Ruckartig	>10
0.03	Flüssig	<5
0.01	Sehr flüssig	0.22

Tab. 8.1: Auswirkungen der Interpolation auf die Bewegungsqualität

8.2 Genauigkeit der Simulation

Durch Tests wurde die Genauigkeit der Simulation ermittelt. Tabelle 8.2 stellt die gemessenen Positionen des IRB 120 im Vergleich zu den erwarteten Sollwerten dar. Dabei zeigt sich, dass die maximale Abweichung während der Bewegung bei ca. 4,6 mm liegt, während die stationäre Abweichung maximal 0,22 mm beträgt.

	Abweichung bei Bewegung [mm]	**Stationäre Abweichung [mm]**
Y Schwankung	2.49	0.12
Z Schwankung	4.59	0.15

Tab. 8.2: Testergebnisse Genauigkeit bei linearer Bewegung
(*n=1200 Datenpunkte*)

Diese Kombination aus Interpolation und Pufferung hat sich als effektive Methode erwiesen, um eine nahezu verzögerungsfreie und präzise Bewegungsdarstellung zu ermöglichen. Damit konnten die Anforderungen an eine flüssige Simulation erfüllt werden, ohne dass eine übermäßige Rechenlast auf das System entsteht.

8.2.1 Anforderungen und Funktionen des Interfaces

Ein zentrales Anliegen war die Definition der Anforderungen und Funktionen, die das Interface erfüllen muss, um eine effiziente und benutzerfreundliche Steuerung des Digitalen Schattens zu ermöglichen. Das entwickelte Interface ermöglicht eine intuitive und flexible Interaktion mit dem Digitalen Schatten. Die objektorientierte Programmierung und klar definierte Schnittstellen trugen zur Wiederverwendbarkeit und Anpassbarkeit bei. Besondere Aufmerksamkeit wurde auf die Visualisierung der Bewegungen und die Benutzerfreundlichkeit gelegt. Die Implementierung einer Heatmap zur Visualisierung der Verweildauer und die Möglichkeit, Simulationen unabhängig von der realen Fertigungszelle durchzuführen, erfüllten die gestellten Anforderungen.

8.2.2 Erfassung und Nutzung von Echtzeitdaten

Abschließend wurde untersucht, wie Echtzeitdaten erfasst, gespeichert und zur Simulation verwendet werden können. Die Echtzeitdaten der Maschinen wurden erfolgreich erfasst und in einer SQLite-Datenbank gespeichert. Dies ermöglichte eine präzise Echtzeitsimulation sowie die Aufzeichnung und spätere Wiedergabe von Fertigungsprozessen. Diese Vorgehensweise bietet wertvolle Möglichkeiten zur Analyse und Optimierung der Fertigungsprozesse. Die gewählte Datenbanklösung war ausreichend, jedoch könnte für zukünftige Anwendungen der Wechsel zu spezialisierten Zeitreihendatenbanken erwogen werden. Durch diese Architektur kann das System nicht nur aktuelle Maschinenzustände in Echtzeit abbilden, sondern auch historische Daten für Analysen oder zur Optimierung von Fertigungsprozessen heranziehen. Dies stellt eine wichtige Grundlage für zukünftige Erweiterungen dar, insbesondere im Bereich der prädiktiven Wartung oder KI-gestützten Prozessoptimierung.

8.2.3 Einsatz in mehreren Fertigungszellen

Das System wurde konzeptionell so entworfen, dass es in mehreren Fertigungszellen genutzt werden kann. Grundlage dafür ist ein Git Repository, welches das System im-

mer aktuell hält und für Updates zuständig ist (siehe Abbildung 8.1).Die Definition der Komponenten, also der CAD-Dateien, der Struktur und des Mappings wird im Repository gespeichert und ist für alle Systeme immer auf dem gleichen Stand. Jedes lokale System hat jedoch seine eigene Datenbank und Konfiguration über XML-Dateien. Dadurch können Fertigungszellen sehr einfach konfiguriert werden, wobei immer sichergestellt ist, dass alle Systeme auf dem aktuellen Stand sind. Eine praktische Erprobung mit mehreren Fertigungszellen wurde jedoch nicht durchgeführt. Um die Skalierbarkeit zu testen, wurden jedoch mehrere Maschinenkonfigurationen innerhalb einer Testumgebung simuliert. Dabei konnte die Konsistenz der Konfigurationsdateien über ein Git-Repository erfolgreich überprüft werden. Ein realer Test in einer verteilten Umgebung ist jedoch für eine abschließende Bewertung erforderlich.

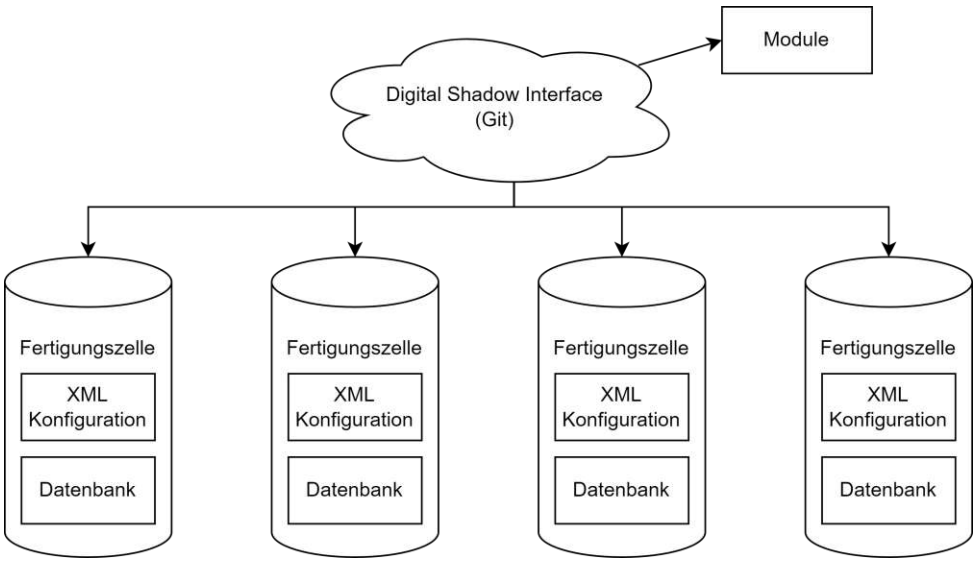


Abb. 8.1: Datenfluss in einem Git-basierten System

8.3 Kritische Betrachtung

8.3.1 Herausforderungen und Lösungen

Weniger positiv war die Performance der Physiksimulation von Siemens NX. Aufgrund der iterativen Berechnungslogik in Siemens NX mussten viele Bauteile vereinfacht werden. Dies führte dazu, dass einige Funktionen, wie die präzise Positionierung über Bolzen, nicht mehr gegeben waren. Diese Vereinfachungen ermöglichten zwar eine effizientere Simulation, hatten jedoch potenzielle Auswirkungen auf die Genauigkeit der Ergebnisse. Eine detaillierte Untersuchung dieser Auswirkungen sollte in zukünftigen Arbeiten erfolgen.

8.3.2 Genauigkeit der Simulation

Die maximale gemessene Abweichung zwischen den simulierten und den realen Koordinaten des Knickarm-Roboters ABB IRB 120 beträgt 4,6 Millimeter. Für eine visuelle Repräsentation ist dies ausreichend, könnte jedoch in Anwendungen mit höheren Präzisionsanforderungen problematisch sein. Industriestandards im Maschinenbau definieren typischerweise eine Toleranz von unter 2 Millimetern. Die stationäre Schwan-

kung wurde mit maximal 0,22 Millimeter gemessen, was auf eine hohe Genauigkeit in Endpositionen hinweist.

8.3.3 Flexibilität und Skalierbarkeit

Die modulare Struktur des Interfaces und die Verwendung von objektorientierter Programmierung ermöglichten eine einfache Anpassung und Erweiterung des Systems. Die Flexibilität des Digitalen Schattens wurde erfolgreich nachgewiesen. Die Möglichkeit, das System auf mehrere Fertigungszellen zu skalieren, wurde konzeptionell betrachtet, jedoch nicht praktisch getestet. Für eine abschließende Bewertung der Skalierbarkeit wären Tests in einer verteilten Umgebung erforderlich, in denen mehrere Digitale Schatten gleichzeitig mit einer realen oder simulierten Fertigungszelle kommunizieren. Besonders kritisch wäre dabei die Untersuchung der Netzwerkstabilität sowie der Latenzen bei mehreren gleichzeitigen Datenquellen.

8.3.4 Benutzerfreundlichkeit

Die Benutzerfreundlichkeit des Interfaces wurde intern getestet und als intuitiv bewertet. Die klare Struktur und die intuitive Bedienung ermöglichen eine einfache Nutzung, auch für Anwender ohne tiefgehende technische Kenntnisse. Eine formale Evaluation durch potenzielle Endnutzer wurde jedoch nicht durchgeführt. Eine solche Studie könnte in einer weiterführenden Arbeit durchgeführt werden, um Schwächen in der Benutzerführung zu identifizieren.

8.3.5 Nutzung von Python für die Entwicklung

Die Entwicklung des Digitalen Schattens wurde mit Python umgesetzt, was sowohl Vor- als auch Nachteile mit sich brachte. Python erwies sich als flexibel und gut geeignet für die schnelle Entwicklung und Prototypisierung. Jedoch traten Herausforderungen bei der Portabilität der Entwicklungsumgebung auf. Das Übersiedeln der Environments gestaltete sich als komplex, da stets die passenden Requirement-Dateien benötigt wurden. Eine kompilierte Software könnte hier Vorteile bieten, da sie eine stabilere und leichter zu verteilende Lösung darstellt.

8.4 Zusammenfassung der Ergebnisse

Die vorliegende Arbeit hat gezeigt, dass der entwickelte Digitale Schatten die in Kapitel 5.2 definierten Anforderungen weitgehend erfüllt. Besonders hervorzuheben sind die erfolgreiche Integration von CAD-Daten, die Realisierung einer flüssigen Simulation und die flexible Anpassbarkeit des Systems. Einschränkungen wie die Abweichung der Positionsgenauigkeit und die vereinfachte Modellierung in Siemens NX verdeutlichen jedoch, dass weiteres Optimierungspotenzial besteht. Die Ergebnisse dieser Arbeit bieten eine fundierte Grundlage für zukünftige Arbeiten, insbesondere im Hinblick auf die Skalierung des Systems auf mehrere Fertigungszellen und die externe Validierung in realen Anwendungsumgebungen.

9. Ausblick

9.1 Weiterentwicklung des Digitalen Schattens

9.1.1 Digitaler Zwilling

Die Weiterentwicklung dieser Arbeit könnte sich auf die Transformation des Digitalen Schattens zu einem Digitalen Zwilling konzentrieren, was ein spannendes Thema für zukünftige Forschungsarbeiten darstellt. Das aktuelle Interface bietet dafür einen soliden Grundstein, muss jedoch an die erweiterten Anforderungen eines digitalen Zwillings angepasst werden. Diese Anforderungen umfassen unter anderem die Fähigkeit zur bidirektionalen Kommunikation mit dem physischen System und fortschrittliche Analysefunktionen.

Eine mögliche Funktion, die auf der bestehenden Lösung aufbaut, wäre die Implementierung einer prädiktiven Wartung. Dabei könnten Sensordaten in Echtzeit analysiert und Wartungsbedarfe vorhergesagt werden, um Ausfallzeiten zu minimieren und die Effizienz der Fertigungszelle zu maximieren. Wie in der Arbeit von Wallner et al. [1] dargestellt, bietet der digitale Zwilling umfassende Möglichkeiten zur Optimierung und Anpassung flexibler Fertigungssysteme durch die Integration solcher fortschrittlichen Funktionen.

9.1.2 Evaluierung der Genauigkeit

In dieser Arbeit wurden zwar einfache Versuche wie die Messung der Genauigkeit des IRB 120 beim Abfahren einer geraden Linie durchgeführt, jedoch wäre eine genauere Untersuchung sinnvoll. Hierbei wäre interessant, ob die Schwankung des Roboters Einfluss auf den Digitalen Schatten hat und die Simulation verfälschen kann. Als erster Ansatz kann versucht werden, manuell die Frequenz der Daten und die Anzahl interpolierter Datenpunkte zu ändern, denn es wird vermutet, dass eine Korrelation bestehen könnte.

9.1.3 Anpassung der Simulationsgenauigkeit

Derzeit basiert jedes Objekt im Digitalen Schatten auf einem leistungsorientierten Modell in Siemens NX. Dies ist für die meisten Simulationen ausreichend. Für genauere Analysen wäre es jedoch praktisch, ein genaueres Modell verwenden zu können. Ziel weiterer Arbeiten könnte es sein, für jedes Objekt mehrere Siemens NX Komponenten mit unterschiedlichen Genauigkeitsstufen zu erstellen und einen Genauigkeitsparameter in die Benutzeroberfläche zu integrieren, der dann die gewünschte Komponente lädt.

9.2 Systemintegration und -einsatz

9.2.1 Einsatz in mehreren Fertigungszellen

Obwohl das System bereits so konzipiert ist, dass es in mehreren Fertigungszellen gleichzeitig eingesetzt werden kann, wurde dies im Rahmen dieser Arbeit nicht praktisch getestet. Zukünftige Arbeiten könnten sich darauf konzentrieren, das System in

verschiedenen Fertigungszellen zu implementieren und die Synchronisation sowie die Aktualisierung der Komponenten über ein zentrales Git Repository zu validieren. Dies würde sicherstellen, dass alle Systeme immer auf dem neuesten Stand sind und zeigen, wie gut die Lösung in einer realen, verteilten Umgebung funktioniert.

9.2.2 Zentralisierte Datenverarbeitung

Derzeit werden alle gesammelten Daten lokal in den jeweiligen Systemen gespeichert. Es wäre jedoch sinnvoll, eine zentralisierte Lösung zur Verfügung zu stellen, die die Daten aus den einzelnen Zellen auslesen kann (siehe Abbildung 9.1). Dies hat den Vorteil, dass die Aufzeichnungen an zentraler Stelle ausgelesen und Fehler erkannt werden können. Ein solches System erfordert größere Änderungen im Code sowie die Integration von Serverstrukturen und stellt eine interessante Weiterführung dieser Arbeit dar. Es muss jedoch berücksichtigt werden, dass eine zentrale Lösung als Single Point of Failure agieren kann, was bedeutet, dass bei einem Ausfall des zentralen Servers alle angeschlossenen Systeme beeinträchtigt werden könnten.

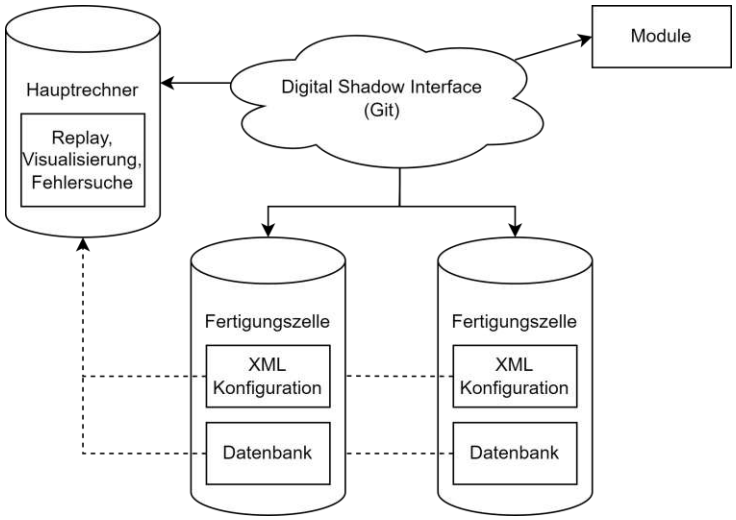


Abb. 9.1: Datenfluss in einem cloudbasierten System

9.3 Technologische Verbesserungen

9.3.1 Portabilität der Entwicklungsumgebung

Die Nutzung von Python für die Entwicklung des Digitalen Schattens hat sowohl Vor- als auch Nachteile gezeigt. Zukünftige Arbeiten könnten sich darauf konzentrieren, die Portabilität der Entwicklungsumgebung zu verbessern. Eine mögliche Lösung könnte die Erstellung einer kompilierbaren Version der Software sein, die eine stabilere und leichter zu verteilende Lösung bietet. Alternativ könnten Containerisierungstechnologien wie Docker genutzt werden, um die Entwicklungsumgebung einfacher zu verwalten und zu verteilen.

9.3.2 Automatischer OPC UA Server über NX Open

Derzeit muss der OPC UA Server bei jedem Laden einer Konfiguration manuell gestartet werden. Die Funktion, den OPC UA Server über NX Open steuern zu können, wäre

von großem Nutzen, da sie die Anzahl der manuellen Aktionen weiter reduzieren und das Programm für jede/n Benutzer/in verständlicher machen würde.

Literatur

[1] B. Wallner, B. Zwölfer, T. Trautner und F. Bleicher, „Digital Twin Development and Operation of a Flexible Manufacturing Cell using ISO 23247,“ in *Procedia CIRP*, Bd. 120, Elsevier B.V., 2023, S. 1149–1154. DOI: 10.1016/j.procir.2023.09.140.

[2] M. Liebenberg und M. Jarke, „Information systems engineering with Digital Shadows: Concept and use cases in the Internet of Production,“ *Information Systems*, 2023. [Online]. Adresse: https://www.sciencedirect.com/science/article/pii/S0306437923000182.

[3] M. Javaid, A. Haleem und R. Suman, „Digital twin applications toward industry 4.0: A review,“ *Cognitive Robotics*, 2023. [Online]. Adresse: https://www.sciencedirect.com/science/article/pii/S2667241323000137.

[4] S. S. Kamble, A. Gunasekaran, H. Parekh und V. Mani, „Digital twin for sustainable manufacturing supply chains: Current trends, future perspectives, and an implementation framework,“ *Technology Forecasting and Social Change*, 2022. [Online]. Adresse: https://www.sciencedirect.com/science/article/pii/S0040162521008799.

[5] A. Fuller, Z. Fan, C. Day und C. Barlow, „Digital Twin: Enabling Technologies, Challenges and Open Research,“ *IEEE Access*, Jg. 8, S. 108 952–108 971, 2020, ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2998358.

[6] W. Kritzinger, M. Karner, G. Traar, J. Henjes und W. Sihn, „Digital Twin in manufacturing: A categorical literature review and classification,“ in *IFAC-PapersOnLine*, Bd. 51, Elsevier B.V., Jan. 2018, S. 1016–1022. DOI: 10.1016/j.ifacol.2018.08.474.

[7] F. Tao, J. Cheng, Q. Qi, M. Zhang et al., „Digital twin-driven product design, manufacturing and service with big data,“ *International Journal of Advanced Manufacturing Technology*, Jg. 94, Nr. 9-12, S. 3563–3576, Feb. 2018, ISSN: 14333015. DOI: 10.1007/s00170-017-0233-1.

[8] I. Onaji, D. Tiwari, P. Soulatiantork, B. Song und A. Tiwari, *Digital twin in manufacturing: conceptual framework and case studies*, 2022. DOI: 10.1080/0951192X.2022.2027014.

[9] C. Zhang, W. Xu, J. Liu, Z. Liu et al., „A reconfigurable modeling approach for digital twin-based manufacturing system,“ in *Procedia CIRP*, Bd. 83, Elsevier B.V., 2019, S. 118–125. DOI: 10.1016/j.procir.2019.03.141.

[10] M. Jacoby und T. Usländer, *Digital twin and internet of things-Current standards landscape*, Sep. 2020. DOI: 10.3390/APP10186519.

[11] „DIN EN ISO 21597-1 Informationscontainer zur Datenübergabe –Austausch-Spezifikation –Teil 1: Container,“ Techn. Ber., 2020.

[12] Itu-t, „ITU-T Rec. Y.3090 (02/2022) Digital twin network – Requirements and architecture,“ Techn. Ber., 2022. [Online]. Adresse: http://handle.itu.int/11.1002/1000/11.

[13] „Automation systems and integration-Digital twin framework for manufacturing-Part 1: Overview and general principles Systèmes d'automatisation industrielle et intégration-Cadre technique de jumeau numérique dans un contexte de fabrication-Partie 1: Vue d'ensemble et principes généraux," Techn. Ber., 2021.

[14] „ISO 23247-4 Automation systems and integration-Digital twin framework for manufacturing-Part 4: Information exchange," Techn. Ber., 2021.

[15] M. P. Jensen und D. Uckelmann, „Standards für das Internet der Dinge," *Zeitschrift für wirtschaftlichen Fabrikbetrieb*, Jg. 115, Nr. 3, S. 171–174, März 2020, ISSN: 0947-0085. DOI: 10.3139/104.112245.

[16] D. Abts, „RESTful Web Services," in *Masterkurs Client/Server-Programmierung mit Java*, Wiesbaden: Springer Fachmedien Wiesbaden, 2022, S. 331–440. DOI: 10.1007/978-3-658-37200-2{_}12.

[17] U. Jumar und J. Jasperneite, „KOMMUNIKATION IN DER AUTOMATION," 2021. DOI: 10.25673/39548. [Online]. Adresse: http://dx.doi.org/10.25673/39548.

[18] Y. Wang, X. Wang, A. Liu, J. Zhang und J. Zhang, *Ontology of 3D virtual modeling in digital twin: a review, analysis and thinking*, 2023. DOI: 10.1007/s10845-023-02246-6.

[19] C. K. Lo, C. H. Chen und R. Y. Zhong, „A review of digital twin in product design and development," *Advanced Engineering Informatics*, Jg. 48, Apr. 2021, ISSN: 14740346. DOI: 10.1016/j.aei.2021.101297.

[20] P. Dymora und A. Paszkiewicz, „Performance analysis of selected programming languages in the context of supporting decision-making processes for industry 4.0," *Applied Sciences (Switzerland)*, Jg. 10, Nr. 23, S. 1–17, Dez. 2020, ISSN: 20763417. DOI: 10.3390/app10238521.

[21] Statista, *Most used programming languages among developers worldwide*. [Online]. Adresse: https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/.

[22] N. Silnitsky, *How to choose the right database for your service*, https://medium.com/wix-engineering/how-to-choose-the-right-database-for-your-service-97b1670c5632, 2021. [Online]. Adresse: https://medium.com/wix-engineering/how-to-choose-the-right-database-for-your-service-97b1670c5632.

[23] L. C. Magalhães, L. C. Magalhães, J. B. Ramos, L. R. Moura et al., „Conceiving a Digital Twin for a Flexible Manufacturing System," *Applied Sciences (Switzerland)*, Jg. 12, Nr. 19, Okt. 2022, ISSN: 20763417. DOI: 10.3390/app12199864.

[24] I. J. Gausemeler und S. Moehringer, „VDI 1206-A NEW GUIDELINE FOR THE DESIGN OF MECHATRONIC SYSTEMS," Techn. Ber., 2002. [Online]. Adresse: www.elsevier.comllocatelifac.

[25] E. Schicker, „Datenbanken und SQL," Techn. Ber.

[26] A. Struckov, S. Yufa, A. A. Visheratin und D. Nasonov, „Evaluation of modern tools and techniques for storing time-series data," in *Procedia Computer Science*, Bd. 156, Elsevier B.V., 2019, S. 19–28. DOI: 10.1016/j.procs.2019.08.125.

[27] A. Sajjanshetty, *Towards Inserting One Billion Rows in SQLite Under A Minute*, https://avi.im/blag/2021/fast-sqlite-inserts/, 2021. [Online]. Adresse: https://avi.im/blag/2021/fast-sqlite-inserts/.

[28] J. Mareczek, *Grundlagen der Roboter-Manipulatoren Band 1*. Springer Berlin Heidelberg, 2020. DOI: 10.1007/978-3-662-52759-7. [Online]. Adresse: https://doi.org/10.1007/978-3-662-52759-7.

[29] P. Manceron, *IKPy*, Mai 2022. DOI: 10.5281/zenodo.6551158. [Online]. Adresse: https://doi.org/10.5281/zenodo.6551158.

[30] *PySimpleGUI*. [Online]. Adresse: https://www.pysimplegui.org/en/latest/.

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis