VOM KABINENPLANER ZUR FERTIGUNG - DAS DLR-PROJEKT DICADEMA

M. Malecha*, V. Srinivasan†, Y. Ghanjaoui†, P. Satwan†, M. Schönheits*, M. Mayer*, A. Schuster*

* Deutsches Zentrum für Luft- und Raumfahrt, Zentrum für Leichtbauproduktionstechnologie, Am Technologiezentrum 4, Augsburg, Deutschland

† Deutsches Zentrum für Luft- und Raumfahrt, Institut für Systemarchitekturen in der Luftfahrt, Hein-Saß-Weg 22, Hamburg, Deutschland

Zusammenfassung

Das Projekt DiCADeMa (Digital Cabin Architectures and Design for Manufacturing) erforscht die digitale Durchgängikeit vom Kabinendesign bis zur Herstellung und Montage von Kabinenkomponenten um den Endkunden (Airlines) eine möglichst späte Entscheidungsmöglichkeit für die Ausgestaltung der Konfiguration der Kabinenelemente zu ermöglichen. In dem betrachteten Use Case werden, ausgehend von einer Sitzkonfiguration, die Positionen der Gepäckablagen berechnet und mit Hilfe der Common Parametric Aircraft Configuration Schema (CPACS) und dem Fuselage Geometry Assembler (FUGA) validiert. Dabei werden die Aufhängepunkte der Gepäckablageelemente an den Spanten bestimmt. Der Prozess wird aus den definierten Einzelschritten geplant (CAMEO) und eine ausführbare Montagefolge aufgebaut. Der Ablauf des Prozesses wird für die teilnehmenden Akteure simuliert und die einzelnen Schritte an die Prozessteilnehmer in der entsprechenden Reihenfolge geschickt: ein Autonomously Guided Vehicle (AGV) mit einem UR10e-Leichtbauroboter fährt die Spante an, bestimmt lokal mit Hilfe der CAD-Daten aus der Simulation und optischer Referenzierung die Aufhängepunkte der Hatracks und markiert die Bohrung die anschließend vom menschlichen Arbeiter manuell gesetzt wird. Das Paper beschreibt die Prozessarchtektur und die für Dezember 2024 geplante Demonstration am Full-Size-Mockup.

Keywords

Cabin Design; Automated Maufacturing; Digital Process; Last Minute Custamization

NOMENKLATUR

Abkürzungen

AGV	Automated Guided Vehicle
CPACS	Common Parametric Aircraft Configuration Schema
CSM	Cameo Systems Modeler
FUGA	Fuselage Geometry Assembler
JSON	JavaScript Object Notation
KOS	Koordinatensystem
MES	Manufacturing Execution Systems
PCL	Point Cloud Library
PPR	Produkt-Prozess-Ressourcen
TCP	Tool Center Point
XML	Extensible Markup Language

1. EINFÜHRUNG

Die Luftfahrtindustrie, ähnlich wie andere Wirtschaftsbereiche, unterliegt einem Wandel in Richtung gesteigerter Wirtschaftlichkeit und kürzeren Innovationszyklen bei steigenden Anforderungen bezüglich Sicherheit und Qualität [1]. Die Digitalisierung und Automatisierung ist hier eine der gängigen Antworten, obwohl im Vergleich zu anderen Branchen, wie z.B. der Autoindustrie, die traditionell auf Kleinserienfertigung ausgerichteter Flugzeugfertigung überproportional hohe Investitionskosten abverlangt. Daher sind Ansätze für flexible Automatisierungslösungen und durchgängige Digitalisierungsketten die attraktivsten [2], wie z.B. flexible robotische Zellen [3]. Diese sollen (bidirektional) bereits im Design beginnen und über die Fertigung bis hin zur Wartung und Recycling hineinreichen. Das DLR-interne Projekt DiCADeMa (Digital Cabin Architectures and Design for Manufacturing) des Deutschen Zentrums für Luft- und Raumfahrt zielt auf die exemplarische Untersuchung einer bidirektionalen Datenverbindung zwischen dem Design und der Fertigung am Beispiel der Kabinengestaltung. Die ausführenden Institute, Institut für Systemarchitekturen in der Luftfahrt

in Hamburg und Institut für Bauweisen und Strukturtechnologie in Augsburg, beteiligen sich mit den ihren Kompetenzen an den jeweiligen Enden der Prozesskette und erarbeiten die Datenarchitektur und Schnittstellen zwischen der digitalen und der realen Welt für einen in Zusammenarbeit mit der Industrie entwickelten Use Case.

2. DEFINITION DES USE CASE

Der Use Case beschreibt das Szenario in dem der Kunde (eine Fluggesellschaft) die Ausgestaltung der Kabine im Vergleich zu heutigen Prozessen sehr spät finalisieren kann (Last Minute Customization), was dem Kunden einen erheblichen Vorteil einer kurzfristigen Anpassung an die aktuelle Marktsituation bietet. Der Kunde kann die Sitzkonfiguration in der Kabine anpassen woraus sich die Position und Größe der über den Sitzen befindlichen Gepäckablagen und deren Befestigungspunkte an der Flugzeugstruktur ergeben. Die Konfiguration wird auf Plausibilität geprüft und anschließend ein Fertigungsprozess generiert und simuliert. Dieser wird an eine automatisierte Anlage in dem realen Fertigungsbereich geschickt die entsprechenden Bohrungen für die Befestigungspunkte für einen menschlichen Werker markiert. Nach dem manuellen Bohrprozess werden die Bohrungen vermessen um eineseits die Dokumentation (den digitalen Zwilling) zu aktualisieren und anderseits um die Ausgleichelemente für die Aufhängung der Gepäckablagen vorzuschlagen. Die beim Fertigungsprozess gewonnenen Daten werden in das Design und die Plaung zurückggeschickt und für die Optimierung der Design- und Fertigungsprozesse verwendet.

3. PROZESSABLAUF

Um die Projektziele zu erfüllen, ist ein Prozessablauf für die Verknüpfung der vorhandenen Tools erstellt worden. Bild 1 zeigt dies schematisch. Hierbei sind die durchgeführten Prozessschritte die folgenden:
Am Anfang der Prozesskette steht ein Kabinenkonfigurator, über den ein Benutzer sich das Kabinenlayout frei zusammenstellen kann. Im Rahmen des DiCADeMa Projektes sollen hierbei Variationen von Gepäckablagen (sogenannte „Hatracks“) sowie frei platzierbare Monumente (Toiletten, Küchen) betrachtet werden. Im nächsten Schritt wird das gewählte Kabinenlayout verwendet, um einen Entwurf für die gesamte Kabine zu erstellen. Dies ermöglicht einen Rückschluss auf die Position der für die Kabinenfertigung notwendigen Strukturanbindungspunkte. Die Positionen der Strukturanbindungspunkte können dann von der Prozessplanung verwendet werden, um die notwendigen Prozessschritte für die Kabinenfertigung zu planen.

Um eine einheitliche, automatisierte Kommunikation zwischen den ersten drei Prozessschritten zu ermöglichen, sind eine REST API sowie eine zentrale Datenbank implementiert worden. Dabei speichert die Datenbank die für spätere Prozesschritte notwendigen Informationen, während die REST API als zentrale Schnittstelle der Prozesse zur Datenbank dient. Im vorliegenden Fall stellt die Prozessplanung die Anfrage nach den Strukturanbindungspunkten über die API. Daraufhin startet die API die notwendigen Prozesse im Kabinenkonfigurator und im Kabinenentwurf, speichert die jeweiligen Ergebnisse in der Datenbank, und übermittelt schlussendlich die Strukturanbindungspunkte an die Prozessplanung. Als Datenbank dient dabei in der ersten Implementierung eine JSON („JavaScript Object Notation“) Datei, da das JSON-format eine für Mensch und Maschine einfach les- und editierbare Textform bereit stellt.

Sobald nun die Prozessschritte für die Kabinenfertigung ermittelt worden sind, können sie im nächsten Schritt für die Bewegungsplanung der verwendeten Ressourcen verwendet werden. Dieser Prozessschritt stellt den Übergang zwischen der digitalen Kabinenfertigungsplanung zur physischen Umsetzung in der Fertigung dar.

In den folgenden Kapiteln 4 und 5 wird näher auf die zentralen Prozessschritte eingegangen.

4. DIGITAL

Dieses Kapitel detailliert die digitalen Prozessschritte (siehe Bild 1), die den Kundenwunsch über einen Kabinenkonfigurator erfassen, digital im Kontext des Kabinenentwurfs und der zugehörigen Prozessplanung verarbeiten und daraus bereits simulativ Fertigungsabläufe planen.

4.1. Kabinenkonfigurator

Wie bereits in Kapitel 3 beschrieben, steht am Anfang des Prozessablaufes ein „Kabinenkonfigurator“. Dieser ermöglicht es einem Benutzer die Zusammenstellung der Kabine im Bezug auf Varianten von Gepäckablagen (auch „Hatracks“ genannt) und Bordküchen (auch „Galleys“ genannt) frei wählen zu können. Konkret bedeutet dies, dass die Flugzeugkabine spantweise segmentiert wird. Für jeden Bereich kann zwischen verschiedenen Hatrack-Längen oder der Platzierung eines Monumentes gewählt werden. Für die Implementierung des Kabinenkonfigurators wurde aus Gründen der einfachen Bedienung und Umsetzung für einen ersten Schritt eine Excel-Datei gewählt. Bild 2 zeigt den Aufbau dieser Excel-Datei: Zum einen wird die Flugzeugkabine in Segmente von Spant zu Spant (sogenannte „Frame Bays“) unterteilt. Basierend auf dem in Kapitel 2 beschriebenen Use-Case ergeben sich damit zehn Frame-Bays für den vorliegenden Fall. Auf diese zehn Frame Bays lassen sich dann beliebig Hatracks verschiedener Längen und Galleys verteilen.

Für das DiCADeMa-Projekt wurde eine symmetrische Kabine angenommen, die vorliegende Methodik lässt sich aber mit etwas Implementierungsaufwand im wis-

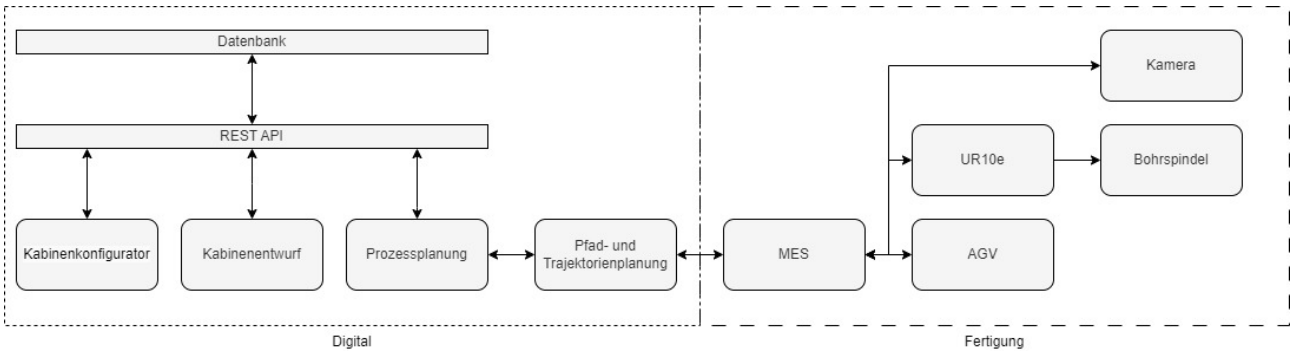

BILD 1. Schematische Darstellung des Prozessablaufes für das DiCADeMa-Projekt.

Frame Bays	1	2	3	4	5	6	7	8	9	10
Monuments	1H	1H	1H	1H	3H			1H	1H	1G

BILD 2. Benutzeroberfläche des Kabinenkonfigurator in Excel zur Beschreibung der spantweisen Belegung einer Flugzeugkabine. Hierbei kann zwischen verschiedenen Hatrack-Längen xH oder der Platzierung einer Galley xG ausgewählt werden.

sensbasierten Kabinenentwurf (siehe Kapitel 4.2) auf asymmetrische Kabinen erweitern.

4.2. Kabinenentwurf

Für die Umsetzung des Kabinenentwurfs wird eine am DLR in Hamburg entwickelte Rumpf- und Kabinenentwurfsumgebung mit dem Namen FUGA („Fuselage Geometry Assembler“) angewandt und weiterentwickelt [4,5]. FUGA verwendet einen wissensbasierten Ansatz, um eine Flugzeugkabine basierend auf einer gegebenen Primärstruktur - Flugzeugrumpf, Flügel, etc. - auszulegen. Als Input für die Primärstruktur dient dabei ein Datensatz im CPACS-Format. CPACS, oder „Common Parametric Aircraft Configuration Schema“ ist eine hierarchische Sammlung an Parametern im XML-Format, um ein bestimmtes Flugzeug möglichst vollständig zu beschreiben [6]. Für den Kabinenentwurf mit FUGA wird dieser CPACS-Datensatz mit Kabinenparametern angereichert - wie z.B. die Sitzverteilung oder die Sitzmodelle - um die gewünschte Kabine zu integrieren [4,5].

Innerhalb von FUGA werden physikalische Grundsätze oder empirische Informationen die den Kabinenentwurf beschreiben als „Regeln“ in der Programmiersprache Python formuliert. Diese Regeln können als Funktionen verstanden werden, die ein bestimmtes Ergebnis generieren. Als Eingangsparameter für eine Regel dient der CPACS-Datensatz, oder direkt eine oder mehrere Regeln die bereits aus dem CPACS-Datensatz neue Informationen berechnet haben. So lässt sich beispielsweise die Position aller Spante direkt aus dem CPACS-Datensatz auslesen, während für die genaue Dimensionierung und Positionierung der Deckenpanele viele weitere Zwischenschritte notwendig sind.

Durch die Verknüpfung von Regeln basierend auf ihren Inputs und Outputs ergibt sich ein gerichteter Graph. Eine vereinfachte Darstellung ist in Abbildung 3 abgebildet. Hierbei lässt sich die Komplexität des sich ergebenden Systems erkennen. Mehrere Regeln die zu einem bestimmten Flugzeugbereich gehören (z.B. Flügel oder Kabine) werden hierbei zu „Regelsätzen“ gruppiert und sind in Bild 3 farblich unterschiedlich dargestellt. Die Eingansparameter des CPACS-Datensatzes sind in grau dargestellt.

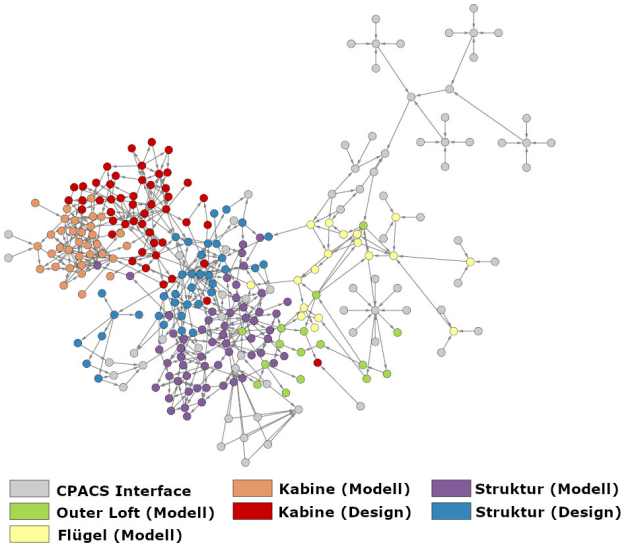

BILD 3. Visualisierung des Verbindungen zwischen FUGA-Regeln als gerichteter Graph. Regeln, die den gleichen Flugzeugteil betreffen werden hierbei zu „Regelsätzen“ verbunden und sind hier farblich getrennt dargestellt [4]. Hierbei wird zwischen Modellgenerierung und Design als verschiedene Aufgaben unterschieden.

Die Modellierung von Regeln als gerichteter Graph hat entscheidende Vorteile: Möchte man das Ergebnis einer bestimmten Regel erfahren - z.B. die Position aller Hatracks - so muss nicht der gesamte Graph durchlaufen werden. Stattdessen kann der Graph auf einen Untergraph reduziert werden, der ausreicht um den gewünschten Endknoten zu erreichen.

Ebenfalls lässt sich durch die aufgebaute digitale Durchgängigkeit überprüfen, ob die Werte überhaupt zueinander passen. So wird beispielsweise die vorgegebene Kabinenkonfiguration aus Unterkapitel 4.1 beim Importieren durch bereits vorhandene Eingabeparameter validiert. Dies dient dazu zu gewährleisten, dass die gewünschte Kabinenkonfiguration auch mit der gegebenen Primärstruktur kompatibel ist.

Um für den DiCADeMa Use-Case eine beliebige Kabinenkonfiguration optional vorgeben zu können, wurde FUGA um entsprechende Regeln erweitert. Bild 4 zeigt eine vereinfachte Visualisierung der durch FUGA ermittelten Ergebnisse als 3D-Modell. Bild 4 beinhaltet hierbei die Spante, den Kabinenboden, die Hatracks und die Galleys für die Kabinenkonfiguration aus Bild 2.

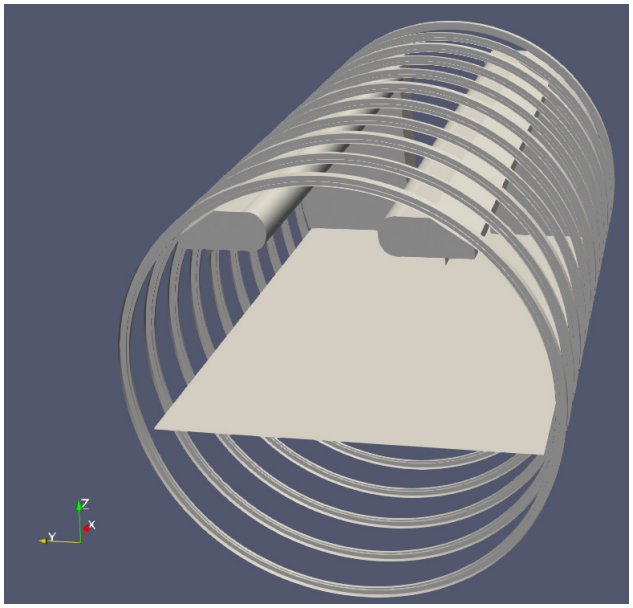

BILD 4. 3D-Darstellung der durch FUGA errechneten Spante, des Kabinenbodens, der Hatracks und der Galleys für die Kabinenkonfiguration aus Bild 2.

Mit Hilfe der resultierenden 3D-Geometrien ist es nun möglich weitere Werte zu ermitteln, die für die spätere Kabinenfertigung notwendig sind. So erlaubt die modellbasierte Umsetzung einen Rückschluss auf die benötigten Montagepunkte durch geometrische Operationen im dreidimensionalen Raum. Für den vorliegenden Fall wurden dabei für jede Hatrack-Variante vier benötigte Montagepunkte angenommen, einer pro Eckpunkt. Für die Montage der Hatracks wurde von Verbindungslösungen (z.B. Brackets) ausgegangen, die - abhängig von den Anbindungsposition in der Kabine - die Hatracks entweder horizontal oder unter einem Winkel mit den Spanten verbinden. Hierbei wurde vereinfachend angenommen, dass jedes Verbindungselement lediglich eine Nietverbindung auf dem Spant benötigt. In der Realität liegen je nach gewähltem Verbindungselement mehr Nietverbindungen vor, jedoch ändert dies nichts an der im Projekt aufgespannten Projektkette bis auf eine Vervielfachung der resultierenden Montagepunkte. Auch die realle Demonstration, wie im Unterkapitel 6.2 beschrieben, wird nur die reduzierte Anzahl an Montagepunkten verwenden.

4.3. Prozessplanung

Das Ziel der Prozessplanung ist es, für einen bestimmten Kabinendesign automatisiert eine Prozesskette mit den auszuführenden Fertigungsschritten zu definieren. Die Prozessplanung erfolgt im Modellierungswerkzeug Cameo Systems Modeler (CSM)[1]. Damit kann das Produkt-Prozess-Ressourcen (PPR) Datenmodell abgebildet werden. Dieses beinhaltet die wichtigsten Montageartefakte, die zugehörigen Metainformationen und wie sie miteinander verlinkt sind. Das Modell entehält ebenfalls eine Zustandsmaschine, die eine automatisierte Prozessgenerierung ermöglicht. Diese ist in Bild 5 dargestellt. Mithilfe der API-Schnittstelle kann das Modell mit externen Anwendungen kommunizieren und verschiedene Aktivitäten in den jeweilige Zuständen triggern.

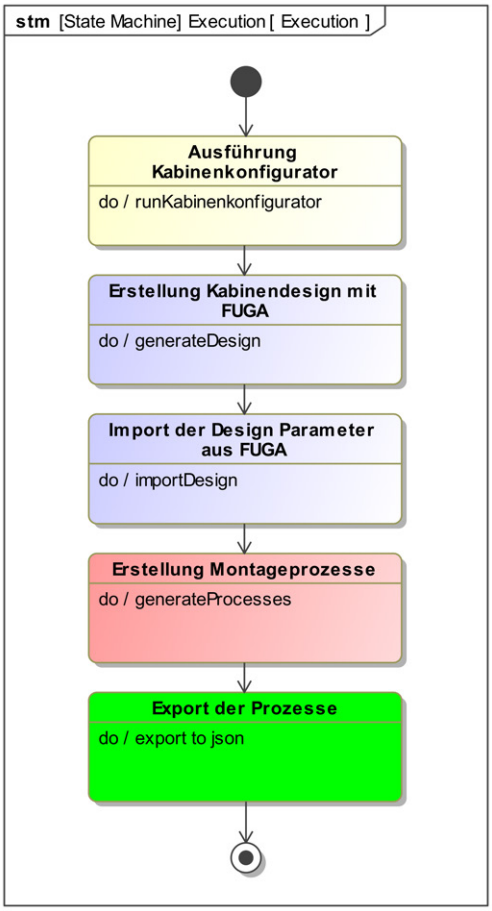

BILD 5. Zustandsmaschine für die automatisierte Ablaufsplanung

Zunächst wird die Kabinenkonfiguration ausgeführt und der Kabinenentwurf wird in FUGA für die ausge-

[1]Cameo Systems Modeler ist eine kollaborative modellbasierte Systems Engineering (MBSE) Umgebung, die Werkzeuge zur Definition, Nachverfolgung und Visualisierung aller Aspekte von Systemen in standardkonformen SysML-Modellen und -Diagrammen bereitstellt [7].

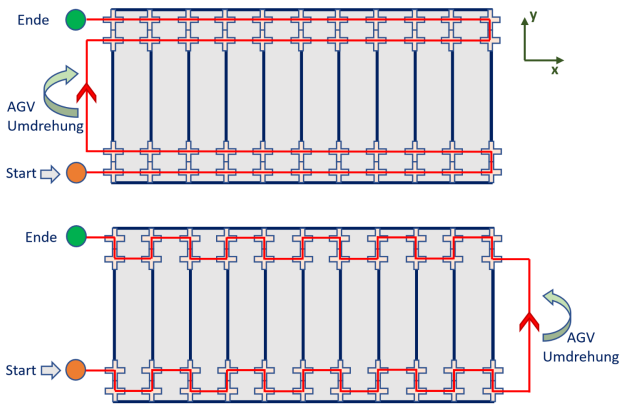

BILD 6. Beispiele für verschiedene Abläufe der Markierungsprozesse

wählte Konfiguration realisiert. Anschließend werden die Hatracks-Modellelemente mit den zugehörigen Positions- und Anbindungsinformationen in der Kabine entsprechend des Datenmodells instanziiert. Im betrachteten Anwendungsfall (vgl. Abschnitt 2) wird ein Markierungsprozess für jeden Anbindungspunkt zwischen Gepäckfächern und Spanten definiert. Die Ablaufplanung des Prozesses wird in einem externen Python-Skript verarbeitet. Dort sind Funktionen für die Prozessabläufe definiert. Das Programm verwendet die kartesischen x-y-Koordinaten der Anbindungspunkte und klassifiziert diese nach einer logischen Reihenfolge. Die ausgewählte Klassifizierungsmethode kann im Modell eingestellt und an ein externes Programm weiter gegeben werden. Bild 6 zeigt zwei Abläufe die von den implementierten Funktionen generiert werden.
Das Modell verwendet die Ergebnisse der Klassifizierung und die Reihenfolge der Prozesse wird mit dem Attribut ranking spezifiziert. Das Modell kann abschließend alle Informationen zu den Prozessen in eine JSON-Datei schreiben und für die detaillierte Planung der Roboterbewegung zur Verfügung stellen.

4.4. Planung und Simulation der Roboterbewegung

Die Berechnungen des Pfads des AGV und Trajektorie des darauf monitierten UR10e-Roboters werden an die Simulation weitergeleitet. Dabei wird die JSON Datei, die von der Prozessplanung übergeben wird, wird als Eingabe für die Simulation benutzt. Ein Teil der bereitgestellten Daten wird in Listing 1 dargestellt.

```
"process_node16":
{
    "tcp_y": 1.42556,
    "tcp_x": 4.2672,
    "tcp_z": 1.787417,
    "robot_id": "UR10e",
    "ranking": 17,
    "id": "mark_attachement_17",
    "kukaA_rad": -1.5707,
    "kukaB_rad": -0.8343,
    "kukaC_rad": 1.5708,
    "cam_x": 4.2830,
    "cam_y": 1.3772,
    "cam_z": 1.8371,
    "drill_x": 4.2830,
    "drill_y": 1.4255,
    "drill_z": 1.7874
}
```

Listing 1. Auszug aus der Eingabe der Simulationsparameter

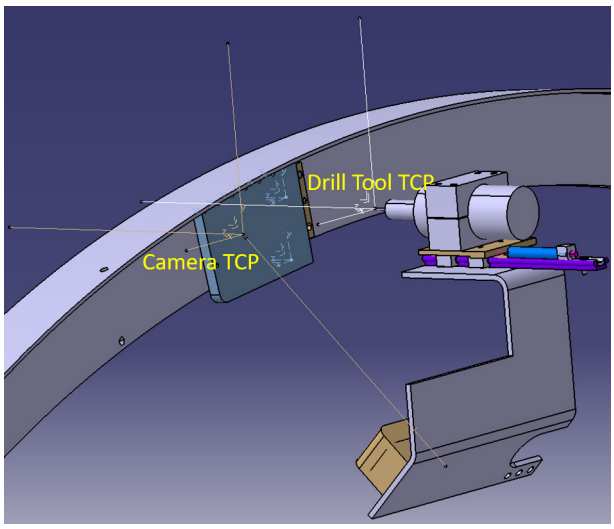

BILD 7. Representätion von Kamera TCP and Bohrer TCP

Für jeden Spant gibt es vier Befestigungpunkten für die Hatracks. Im falle des betrachtetet Use Cases und der Demonstration im DiCADeMa-Projekt gibt es für jeden der Punkte gibt es eine auswechselbare Platte die markiert und gebohrt werden kann, siehe Unterkapitel 6.1. Die Eingabedatei der Prozessplanung enthält die Position der Kamera für die lokale Orientierung (Unterkapitel 5.3) und die Sollposition der Bohrung für den Montagepunkt. Die beiden Positionen sind in Bild 7 dargestellt. Für jede Zielposition an der Rumpfstruktur berechnet die Simulation den Pfad des AGVs und Trajektorie des Roboters unter der Berücksichtigung der Kollisionsvermeidung mit der Umgebung.
Die Berechnung des Pfads- und Trajektorie des Roboters begint mit der Analyse der inversen Kinematik mit dem Fokus auf die Erreichbarkeitsanalyse in Abhängigkeit von Roboterkinematik. Wenn die Zielposition erreichbar ist, wird die entsprechende Trajektorie mit der Berücksichtigung von Kollisionsvermeindung berechnet. Verschiedene Algorithmen werden für die unterschiedlichen Aufgabenkomplexe bewertet, wie z.B. die Entfernung zum Ziel und der Fokuswinkel auf das Ziel, die die Orientierung des Endeffektors beschreiben. Der Prozessablauf ist im Algorithmus 1 dargestellt. Die berechneten Trajektorien sind in der eingebenenen JSON Datei geschrieben. Eine Auszug der Informationen sind in Listing 2 enthalten.

```
"process_node16": {
  "tcp_y": 1.42556,
  "tcp_x": 4.2672,
  "tcp_z": 1.787417,
```

```
"robot_id": "UR10",
"ranking": 17,
"id": "
    mark_attachement_17_for_Bracket",
"kukaA_deg": -90.00,
"kukaB_deg": -47.80,
"kukaC_deg": 90.00,
"kukaA_rad": -1.5708,
"kukaB_rad": -0.8343,
"kukaC_rad": 1.5708,
"cam_x": 4.2830,
"cam_y": 1.3772,
"cam_z": 1.8371,
"drill_x": 4.2830,
"drill_y": 1.4256,
"drill_z": 1.7874,
"euler_angles_of_ur10e_ee": [
  {
    "ee_x": 3.5501,
    "ee_y": 1.2075,
    "ee_z": 1.5637,
    "ee_phi": -1.5699,
    "ee_theta": 0.0,
    "ee_psi": 0.0,
    "euler_angle_repr": "zyx"
  },
  {
    "ee_x": 3.9715,
    "ee_y": 1.5438,
    "ee_z": 1.6398,
    "ee_phi": -1.269,
    "ee_theta": -0.0982,
    "ee_psi": -0.2583,
    "euler_angle_repr": "zyx"
  },
  {
    "ee_x": 4.1793,
    "ee_y": 1.5988,
    "ee_z": 1.6443,
    "ee_phi": -1.1962,
    "ee_theta": -0.1232,
    "ee_psi": -0.3524,
    "euler_angle_repr": "zyx"
  },
  {
    "ee_x": 4.2430,
    "ee_y": 1.6053,
    "ee_z": 1.6424,
    "ee_phi": -1.1789,
    "ee_theta": -0.1285,
    "ee_psi": -0.3784,
    "euler_angle_repr": "zyx"
  },
  {
    "ee_x": 4.2829,
    "ee_y": 1.3769,
    "ee_z": 1.8371,
    "ee_phi": -1.5706,
    "ee_theta": 0.7356,
    "ee_psi": 0.0006,
    "euler_angle_repr": "zyx"
  }
],
"AGV_Path": [ 4.5672, 0.1, 0.0 ]
},
```


Listing 2. Auszug aus der Ausgabe der Simulation

Algorithm 1 Sequenz der Berechnung von Roboterbewegung

input:
list of
  [TargetCameraLocation, TargetDrillLocation] as TargetDrillCameraLoc,
  for all points on the frames

output:
list of
  [AGVTrajectories as $Traj_{AGV}$,
  UR10eTrajectories as $Traj_{UR10eeCam}$,
  UR10eTrajectories as $Traj_{UR10eeDrill}$]
  for all points on the frames

FOR i in len(TargetDrillCameraLoc):
  TDrillLoc = TargetDrillCameraLoc[i][0]
  TCameraLoc = TargetDrillCameraLoc[i][1]
  Procedure ComputeRobotParams($TDrillLoc$, $TCamLoc$)
  $jt\_ang\_drill \leftarrow$ ComputeIK(TDrillLoc)
  $jt\_ang\_cam \leftarrow$ ComputeIK(TCamLoc)

if jt_ang_drill AND jt_ang_cam then
    $Traj_{AGV} \leftarrow$ CompPath(jt_ang_cam)
    $Traj_{Cam} \leftarrow$ CompTraj(jt_ang_cam)
    $Traj_{Drill} \leftarrow$ CompTraj(jt_ang_drill)
  if $Traj_{Cam} \neq NONE$ then
      $World_{T_{Cam}}[ZYX] \leftarrow$ ee_x, ee_y, ee_z, ee_phi, ee_theta, ee_psi where phi, theta and psi denotes the camera end-effector coordinates in euler notation.
  end if
  if $Traj_{Drill} \neq NONE$ then
      $World_{T_{Drill}}[ZYX] \leftarrow$ ee_x, ee_y, ee_z, ee_phi, ee_theta, ee_psi where phi, theta and psi denotes the drill end-effector coordinates in euler notation.
  end if
end if
END FOR
End Procedure

5. FERTIGUNG

Die in den ersten Prozessschritten (Kapitel 4) erzeugte Daten werden an das Shoopfloor übergeben um die Fertigungsschritte auszüführen. Im folgenden Kapitel werden die Komponenten und deren Zusammenwirken beschrieben die verwendet werden um den digitalen Prozess in die reale Fertigungsumgebung zu implementieren.

5.1. Automated Guided Vehicle (AGV)

Als AGV kommt eine mobile Plattform zum Einsatz die im Rahmen anderer Projekte wie „Factory of the Future“ (FoF) und „Drapebot“ [8] zum Transport von Carbonfaser-Zuschnitten mit Hilfe einer aufgesetzten Zuschnitts-Palette verwendet wurde, siehe Bild 8.

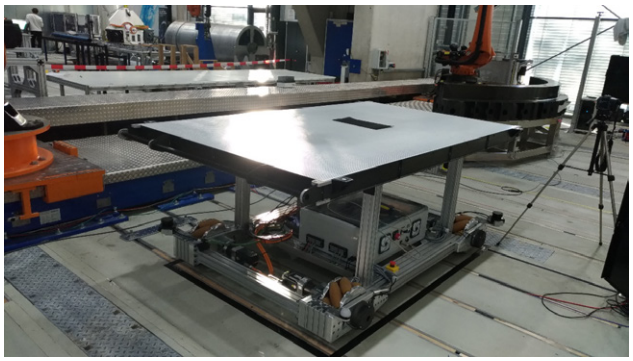

BILD 8. Mobile Plattform mit Zuschnitts-Palette

Das AGV verfügt über 4 Mecanum-Räder um volle 3D-Manövrierfähigkeit zu gewährleisten (zwei translatorische, ein rotatorischer Freiheitsgrad). Der Rahmen besteht aus Aluminium-Profilen sowie CNC-gefertigen Blechen zur Aufnahme der Räder. Als Steuerungsplattform kommt das Beckhoff TwinCAT System zum Einsatz, das sowohl die Bewegegungsteuerung als auch die sichere Überwachung der Achsen, der Bremsen sowie der zwei integrierten SICK nanoscan3 Laserscanner übernimmt.
Für DiCADeMa wurde die mobile Plattform überarbeitet und adaptiert indem ein Universal Robots UR10eee Roboter integriert wurde wie in Bild 9 zu sehen ist.

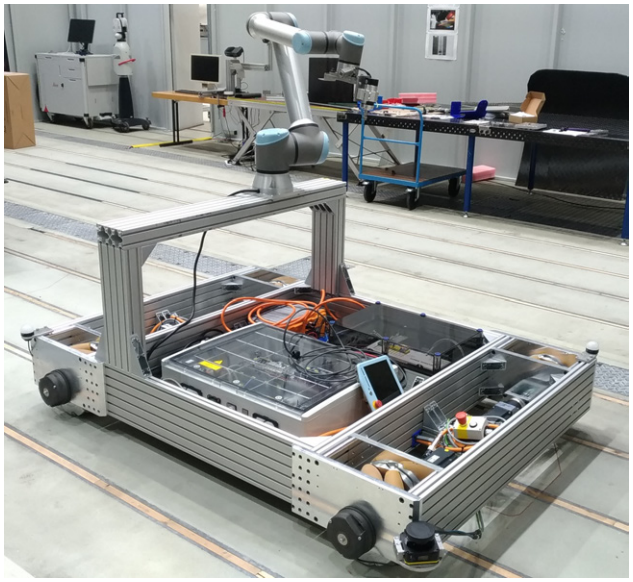

BILD 9. Mobile Plattform für DiCADeMa adaptiert

Für die Kommunikation dem AGV während des Prozesses wurde eine OPC UA Schnittstelle implementiert, die über das in das AGV integrierte WiFi-Netzwerk erreichbar ist. Die Schnittstelle bietet Funktionalitäten wie Fahraufträge zu senden sowie Statusinformationen wie Ist-Position, Stromverbrauch, etc. zu erfassen, die in dir Rückwärtsrichtung des Datenflusses der Plaung und dem Design bereitgestellt werden.
Um die Planung von Pfaden zu ermöglichen muss das AGV selbst oder ein externes System in der Lage sein die Pose des AGV in Bezug auf die Umgebung zu ermitteln. Zusätzlich zu den Odometrie-Daten der Räder wird hierfür auf die kommerzielle Lösung SICK Lidar-Loc zurückgegriffen.

5.2. Manufacturing execution systems (MES)

Die in Listing 2 beschriebenen process nodes müssen im Verlauf der Produktion ausgeführt werden. Ein process node besteht dabei jeweils aus einer Reihe einfacherer Grundschritte, die der Reihe nach auf den vorgesehenen Endgeräten ausgeführt werden müssen. Das ist die Aufgabe des MES (Manufacturing Execution Systems). Hierfür bietet sich eine Client-Server Architktur an, bei der das MES jeweils Client und das Endgerät jeweils Server ist (Bild 10).

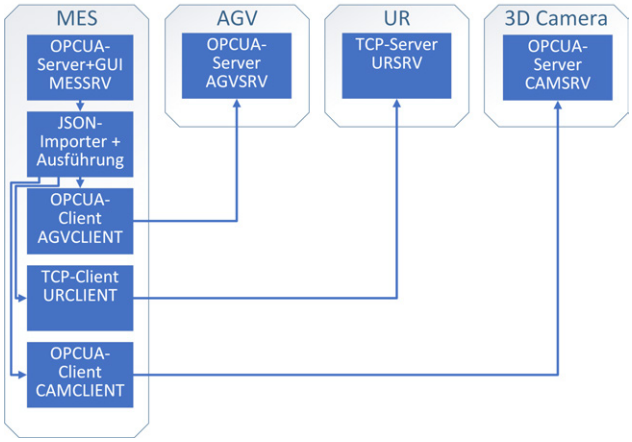

BILD 10. Strukur des MES mit Komponenten-Clients sowie im rechten Bereich die zugehörigen Serverinstanzen von AGV, UR10e und 3D-Kamera.

Als TCP (Transmission Control Protocol) Server wird die von Universal Robot serienmäßig angebotene Lösung verwendet: AGV und Kamera verfügen jeweils über selbst erstellte bzw. konfigurierte OPCUA-Server (vgl. Unterkapitel 5.1 und 5.3). Das MES wird über ein GUI gesteuert, exponiert die Steuerelemente aber zusätzlich über OPCUA, wodurch später beispielsweise ein Leitstand zur Prozessanwahl aufgeschaltet werden kann.
Die Ablaufsteuerung wurde so realsiert, dass die process nodes strikt nach der Siumlationsreihenfolge ausgeführt werden, da eine Änderung des Ablaufs in der Regel eine Neuplanung bedingt, wobei ein neuer AGV-Pfad zur nächsten Bohrposition simuliert wird. Dies stellt einen ersten Schritt dar, im weiteren Verlauf soll eine interaktive Schnittstelle entwickelt und getestet werden, um die Flexibilität auf dem Shopfloor durch eine beliebige Anwahl zu erhöhen.
Der process node enthält neben der Bohrposition einen AGV-Pfad sowie einen Pfad zur Detektionsposition

der Kamera. Aus den nodes werden jeweils die Elementarschritte Anfahren mit AGV, UR10e in Messposition, Messung, Tool-Wechsel, Anfahren der korrigierten Position über Vorposition, Bohrschritt, Rückzug in Vorposition, Tool-Rückwechsel und Rückzug des UR10e in die Safe-Position generiert. Es ergibt sich somit ein Ablauf gemäß Abbildung 11. Ein simultanes Verfahren von AGV und UR10e ist derzeit nicht vorgesehen, da hierfür eine deterministische Echtzeitsynchronisierung der Endgeräte nötig wäre.

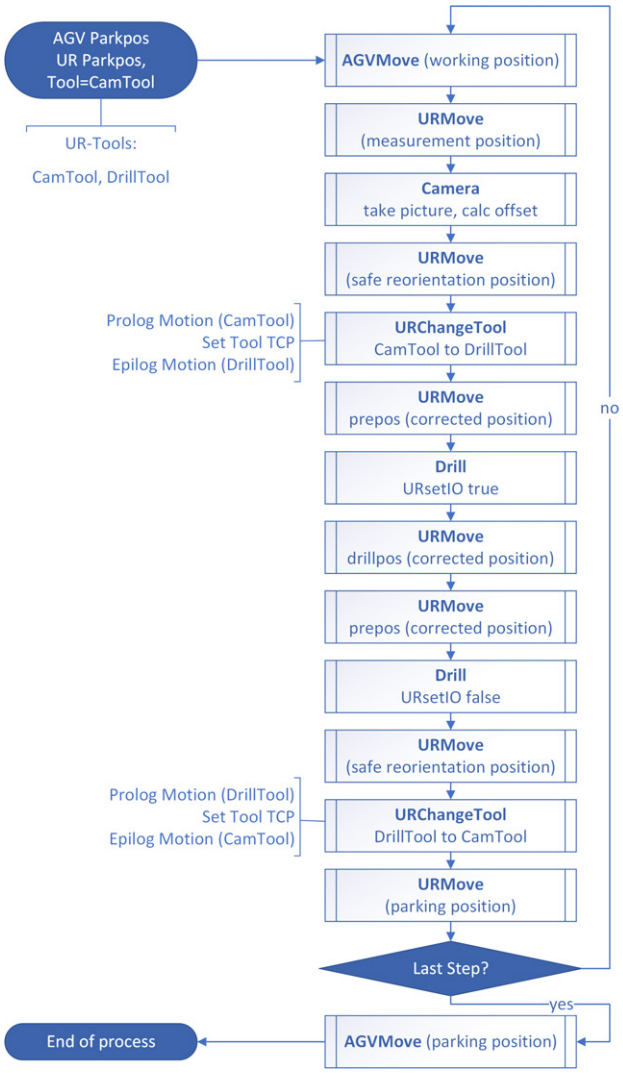

BILD 11. Gesamtprozess aus elementaren Prozesschritten generiert aus der JSON Beschreibung.

Die Implemetierung erfolgt in C++17. Die Elementarschritte werden in einem `std::vector <processStepType>` abgespeichert, wobei der `processStepType` als `std::variant` aus den elementaren Prozesschritten gebildet wird, welche vom abstrakten `processStep` erben, welcher selbst leer ist, aber Verweise auf untergeordnete Prozesschritte enthalten kann. Die `processSteps` sind hiermit polymorphe Objekte, die je nach Art des Prozesschritts `std::shared_ptr<device>executionDevice` Verweise auf spezifische Instanzen der Ausführungsobjekte (devices) enthält. Die konkreten `devices` erben wiederum vom allgemeinen `device`. Die Abarbeitung geschieht mit einem rekursiv aufgerufenen visitor pattern, wodurch eine codemäßige Trennung von Geräteimplementierung und Ausführungsschicht vorgenommen werden kann. Parallel wird noch ein zweiter Ansatz untersucht, in dem die Ausführung Teil des `device` ist, was Redundanzen im Code reduziert, jedoch die Entkopplung verschlechtert. Für eine Beschreibung des visitor-patterns siehe z. B. [9].

5.3. Lokale Orientierung

Um die Hatracks montieren zu können, werden Bohrungen auf den Spanten benötigt. Dazu sollen mittels AGV und einem, auf dem AGV, montierten Roboter (UR10e) die Bohrpositionen markiert werden. Die Positionierung des AGV sowie des Roboters weisen jedoch im mm-Bereich auf. Auch der Boden auf dem das AGV sich bewegt weist Abweichungen vom CAD auf oder biegt sich durch, wie es vom realen Flugzeugboden erwartet wird.
Um diese gesamte Kette an Ungenauigkeiten zu eliminieren wird mittels 3D Kamera eine lokale Neupositionierung vorgenommen. Sowohl die Kamera bzw. deren Aufnahme, als auch der Algorithmus, der die korrigierte Bohrposition ermittelt, müssen die gewünsche Genauigkeit im sub-mm-Bereich garantieren können.
Da die Positionskorrektur nicht nur in einer Ebene sondern auch rotatorisch in allen drei Achsen erfolgen können muss, kommen besonders 3D Kameras in Frage. Bei der Auswahl wurde auf eine hohe Auflösung und Genauigkeit geachtet. Die meisten 3D Kameras besitzen einen größeren Arbeitsabstand als 500 mm. Aufgrund der beengten Verhältnisse zwischen einzelnen Spanten fiel die Wahl auf die Ensenso N36-606-16-BL, die einen Arbeitsabstand von 283 mm ermöglicht [10].
Um sich lokal zu Orientieren kommen Ebenen oder Objekte in Frage. Die Ebene in die gebohrt werden soll kann dabei zur Ermittlung der Normalen dienen. Die Rotationsachsen der Niete bzw. Fastener liefern die Richtung einer weiteren Achse, womit ein lokales Koordinatensystem bzw. eine Transformationsmatrix bestimmt werden kann.
Der Kamera-TCP (Camera TCP) wird wie in Bild 7 im CAD auf die Position des Kamera-Messpunktes bewegt. Die Soll-Position aller bereits am Spant gesetzten Nieten (Fastener) kann somit mit Bezug auf den Kamera-TCP aus dem CAD exportiert werden. In der Fertigung wird der Kamera-TCP ebenfalls auf den Kamera-Messpunkt ausgerichtet, allerdings mit einer, wie weiter oben beschriebenen, Ungenauigkeit. Bild 12 veranschaulicht die optimale Position, wie im CAD vorgegeben und Ist-Position (grünlich markiert) die durch Ungenauigkeiten dann tatsächlich nur erreicht wird. Zur Veranschaulichung wurde der Roboter mehrere cm bewegt.
An der Ist-Position wird die Kamera getriggert und gibt eine 3D Punktwolke aus. Ab hier startet eine in C++ implementierte Pipeline, die am Ende den korrigerten Bohrpunkt mit Bezug auf den Kamera-TCP ausgibt. Dieser Punkt wird weiter, anhand der aktuellen Roboterposition, transformiert und liegt final im

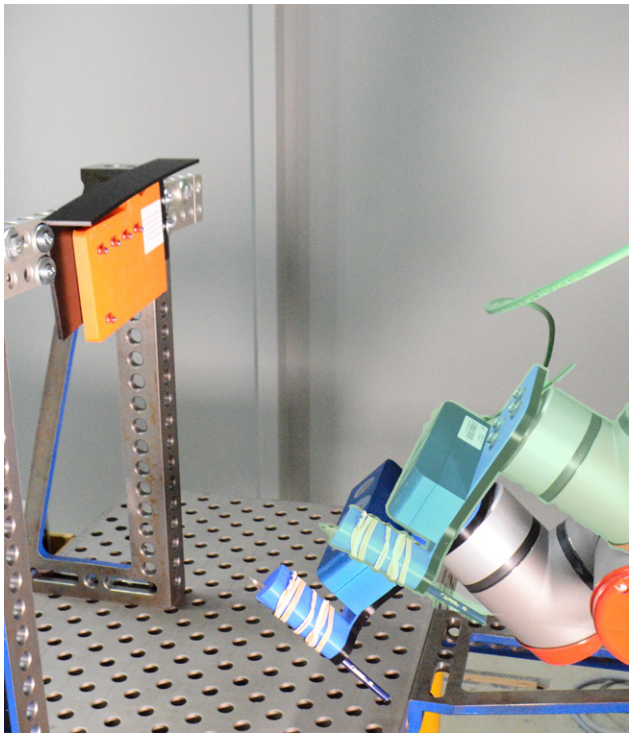

BILD 12. Soll-Position des Roboters bei der keine Korrektur des Bohrpunktes notwendig wäre. Dagegen in hellgrün eine mögliche Fehlpositionierung des Roboters, die eine Korrektur des Bohrpunktes erfordert. Als Referenzierungselemente dienen die Fastener auf dem Spant (rote Punkte auf orangenem Mockup-Spant)

Roboterbezugs-Koordinatensystem vor. Nun kann auf den TCP des Markierwerkzeugs (Drill Tool TCP) gewechselt werden um den Punkt anzufahren.
Die Pipeline verwendet die Point Cloud Library, eine 2D/3D Bild und Punktwolken verarbeitende Bibliothek, und ein eigens entwickeltes Verfahrens um die Achse eines rotationssymetrischen Bauteils zu ermitteln. Zu Beginn wird die Ebene ermittelt auf der sich die Fastener befinden. Anschließend werden die einzelnen Fastener extrahiert. Die Achsen der Fastener werden grob mittels Kovarianzmatrix bestimmt und dann iterativ optimiert, siehe Bild 13. Hier sei angemerkt das die Punktwolke jedes einzelnen Fasteners sehr unvollständig ist, da die Rückseite der Fastener verschattet ist und somit Messpunkte fehlen. Es folgt ein Best-Fit zwischen Soll-Fastener-Positionen und Ist-Fastener-Positionen um final die Transformation bestimmen zu können.

5.4. Markierung

Die korrigierte Position für die Bohrung soll durch den UR10e für den Werker markiert werden. Hierzu kann eine Vielzahl von Methoden eingesetzt werden - in dem hier beschriebenen Use Case wird eine Körnung bzw. Anbohrung der Position eingesetzt um dem Werker bestmögliche Voraussetzung für eine optimale Bohrung anzubieten. Zu dem Zweck enthält der Endeffektor des UR10e eine Motorspindel mit einem Bohrer der die Bohrstelle mit großer Präzision mit einer ca. 1 mm

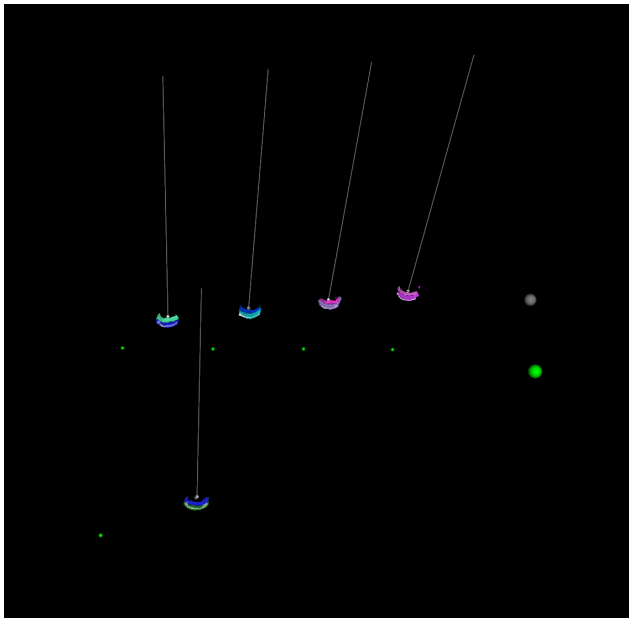

BILD 13. Die kleinen hellgrünen Punkte geben die Soll-Fastener-Positionen an. Der größere hellgrüne Punkt unten im Bild die Soll-Bohrposition. Die Linien zeigen die tatsächlich gefundenen Achsen der Fastener an. Bestimmte Bereiche der Fastener-Punktwolken sind am linken Ende der Achsen sichtbar. Der korrigierte Bohrpunkt ist unten im Bild grau dargestellt. Vergleiche den Versuchaufbau in Bild 12.

tiefen Senkung markiert. Das Design des Endeffektors ist im Bild 14 zu sehen.

6. VORSCHAU

Die Ergebnisse des Projektes werden zum Laufzeitende im Dezember 2024 in einer Demonstration präsentiert. Der Use Case wird in einem durchgängigen Szenario validiert - von der Konfiguration der Größe und der Position der Hatracks - bis hin zum Vermessen der Bohrungen. Dazu wird am ZLP in Augsburg ein Mock-Up für eine Kabine mit elf äquidistant verteilten Spanten aufgebaut innerhalb des das AGV mitsamt dem montierten UR10e die berechneten Bohrpositionen grob anfahren und mit Hilfe der Lokalorientierung präzise markieren soll. Anschließend werden die manuell getätigten Bohrungen vermessen.

6.1. Mock-Up

Das Mock-Up besteht aus elf am Hallenboden verankerten Spanten in der Originalgröße des A320 Airbus, siehe Bild 15. Aus Kostengründen sind die Spante kreisförmig (mit einem Schnitt in der Höhe des Kabinenbodens) und nicht wie im Original elliptisch. Die maximale Abweichung beträgt jedoch unter 3 mm. Die Spante sind identisch und äquidistant in einem Abstand von 0.55 m verteilt. An den Spanten sind Platten abgebracht, die eine Anzahl an Fastenern enthalten die zu lokalen Orientierung verwendet werden. Die Platten sind austauschbar und können markiert und

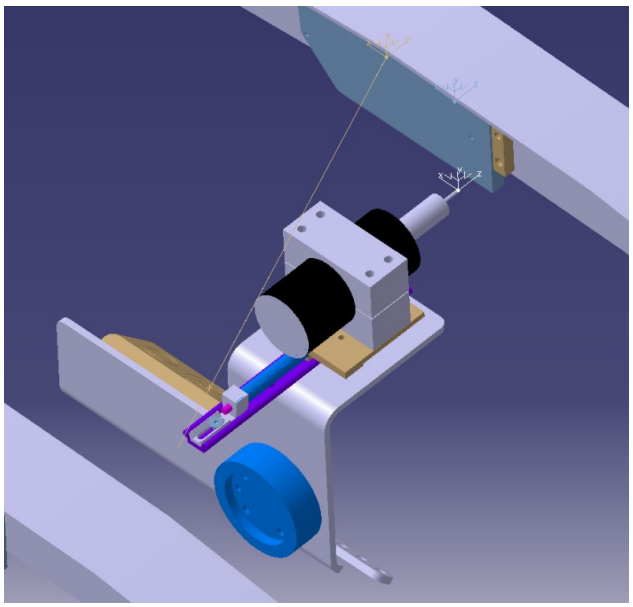

BILD 14. Modell des Endeffektors auf dem UR10e mit Markierungsspindel und Kamera.

anschließend gebohrt werden. Damit können mehrere Versuche durchgeführt werden.

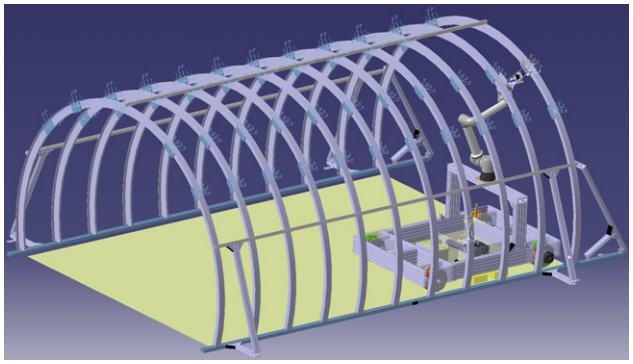

BILD 15. Modell des Mock-Ups für die Demonstration im Dezember 2024.

6.2. Demonstration

Die finale Demonstration zum Projektende im Dezember 2024 soll die komplette durchgängige Kette wie sie in Kapiteln 4 und 5 beschrieben wurde darstellen. Dabei wird eine fiktive Konfiguration der Hatracks im Excel-Tool vorgegeben. Ohne menschliche Einwirkung werden die Daten prozessiert, Pfade generiert und simuliert und an das Shopfloor ausgegeben. Das AGV fährt die relevanten Spante an und markiert anhand einer präzisen lokalen Orientierung die Bohrpositionen. Der Werker bohrt anschließend die markierten Aufhängepunkte. Das AGV bringt den UR10e samt der montierten Kamera an die Spante ein weiteres Mal und die exakte Bohrpositionen werden vermessen. Die Genauigkeit sowie die Leistungsfähigkeit des Systems wird begutachtet und analysiert.

7. ZUSAMMENFASSUNG

In diesem Projekt wird digitale Durchgängigkeit von Ferigungsprozessen von Design bis Produktion im Luftfahrtkontext beschrieben. Dank der digitalen Durchgängigkeit ist es möglich, die (ausgewählte) Anforderungen dynamisch fetszulegen und zu realizieren, sodass ein Designfreeze viel später im Vergleich zu heutigen Prozessen erflogen kann (Last Minute Customization). Use Case für dieses Projekt ist die Markierung und Setzen von Bohrungen in Spanten der Flugzeugkabine an denen die Hatracks montiert werden. Am Anfang werden die kundenspezifischen Informationen (hier in einer Excel-Datei) gesammelt und die eingegebenen Information werden in FUGA validiert. Bei gültigen Eingaben werden die entsprechenden Befestigungspunkte im Kabinenmodell generiert. Die Ausgabe von FUGA wird an Prozessplannung weitergeleitet. Die Prozessplannung definiert die ausführende Fertigungsschritte für notwendige Prozessabläufe, die abhängig von verschiedenen logische Reihenfolge der Zielpunkte (im kartesischen System) klassifiert sind. Danach folgt die Simulation der AGV zusammen mit UR10e-Roboter. Die Simulationssoftware benutzt zwei Eingaben:

- die initiale Position der Kamera (Camera TCP)
- die Bohrpositionen (Drill TCP) an Spanten

aus der Prozessplannung. Mit den Informationen berechnet die Simulation den Pfad des AGVs und die Trajektorie von UR10e. Die Informationen aus Simulation werden an das Shopfloor (MES des AGVs) weitergeleitet. Beim Anfahren der Positionen ist es erforderlich, die Ungenaugigkeit von AGV Bewegung auszugleichen. Diese wird anhand der optischen Referenzierung der Kamera an ausgeprägten Merkmmalen der lokalen Umgebung (hier Nieten/Fastener) berechnet und neue Trajektorie ist ermittelt um die Markierung für die Bohrung präzise in submm-Bereich zu setzen. Der Setup für die Demonstration der gesamten Prozesskette im Dezember 2024 wurde beschrieben.

8. DANKSAGUNG

Der Dank der Autoren gilt dem Projektteam und den externen Uterstützern.
Das Projekt DiCADeMA wird als internes DLR-Projekt aus dem, durch den Deutschen Bundestag beschlossenen Budget, grundfinanziert. Eine Zuwendung durch Dritte findet nicht statt.

Literatur

[1] Wolfgang Kuhn. Digital factory - simulation enhancing the product and production engineering process. In Proceedings of the 2006 Winter Simulation Conference, pages 1899–1906, 2006. DOI: 10.1109/WSC.2006.322972.

[2] Philip Webb, Seemal Asif, Susanne Hogger, Thomas Kosche, and Paul Kiernan. Advanced flexible automation cell control for aerospace manufacturing. Aircraft Engineering and Aerospace Tech-

nology: An International Journal, 87(2):156–164, Jan 2015. ISSN: 0002-2667. DOI: 10.1108/AEAT-11-2012-0204.

[3] Alessandra Caggiano, Fabrizia Caiazzo, and Roberto Teti. Digital factory approach for flexible and efficient manufacturing systems in the aerospace industry. Procedia CIRP, 37:122–127, 2015. ISSN: 2212-8271. CIRPe 2015 - Understanding the life cycle implications of manufacturing. DOI: https://doi.org/10.1016/j.procir.2015.08.015.

[4] Jan-Niclas Walther, Christian Hesse, Jörn Biedermann, and Björn Nagel. Extensible aircraft fuselage model generation for a multidisciplinary, multi-fidelity context. In 33rd Congress of the International Council of the Aeronautical Sciences (ICAS), 2022.

[5] Jan-Niclas Walther. Knowledge-based engineering to provide aircraft fuselage design details for multidisciplinary and multifidelity analysis model generation. Phd thesis, Technische Universität Berlin, Berlin, 2024. DOI: 10.14279/depositonce-20898.

[6] Marko Alder, Erwin Moerland, and Jepsen, Jonas, Nagel, Björn. Recent advances in establishing a common language for aircraft design with cpacs, 2020.

[7] DassaultSystemes. Cameo Systems Modeler: The software tool for systems engineering and modeling, 2024. https://www.3ds.com/products/catia/no-magic/cameo-systems-modeler [visited 20.08.2024].

[8] Marcin Malecha, Alberto Gottardi, Matteo Terreran, Kasper Hald, and Enrico Villagrossi. Human-robot collaboration for complex draping processes of carbon-fibre-reinforced polymers for aerospace parts. In 15th International Conference on Mechanical and Aerospace Engineering (ICMAE), Zagreb, Croatia, 2024.

[9] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1 edition, 1994. ISBN: 0201633612.

[10] Demas Schmorell, Dominik Görick, Alfons Schuster, Monika Mayer, and Andreas Buchheim. Comparison and selection of 3d-camera system for creation of geometric figures by means of 3d-point clouds. In CAMX – the Composites and Advanced Materials Expo, 2024.