ENGINEERING AEROTEC

Universität LÜNEBURG

Prozessoptimierung mit CATIA V5

-

Vereinfachung der Positionierung von Bauteilen eines Versorgungskanals im Flugzeugbau

Diplomarbeit angefertigt an der
Universität Lüneburg,
Fachbereich Automatisierungstechnik,
Studiengang Ingenieur-Informatik
zur Erlangung des akademischen Grades eines
Dipl.-Inform. (FH)

Vorgelegt von
Andreas Krohn
aus Winsen (Luhe)
Matr.Nr.: 153353,
am 09.08.2005

Erstprüfer:	Prof. Dr.-Ing. Heinrich Schleich
Zweitprüfer:	Prof. Dr.-Ing. Wilfried Adami
Betrieblicher Betreuer:	Dipl.-Ing. Torben Fuß

Abstract

Diese Arbeit befasst sich mit der Untersuchung des 3D-Konstruktionswerkzeuges CATIA V5. Es werden die Möglichkeiten analysiert, Arbeitsprozesse mit Hilfe dieses Tools zu optimieren. Anhand des Auftrages zur Positionierung von Bauteilen im Versorgungskanal eines Airbus A340 wird untersucht, in wie weit CATIA V5 hier unterstützende Funktionen anbietet, um die Arbeit schneller, dadurch kostengünstiger und effektiver, also auch weniger fehleranfällig durchzuführen.
Es werden verschiedene Möglichkeiten erarbeitet, analysiert und bewertet. Beispielhaft wird mit der Umsetzung der gewählten Möglichkeit begonnen. Am Ende wird eine Aussicht auf die Zukunft der untersuchten Methodik gegeben.

This diploma thesis involves the examination of the 3D construction tool CATIA V5.
The possibilities will be analysed in order to optimise the work processes. On the basis of the request positioning components in the supply duct of an airbus A380 it will be researched how far CATIA V5 is able to offer support with its capacity, in order to work more quickly, cheaper and effective and additionally to help to cause less mistakes. There will be different possibilities worked out, analysed and evaluated.
For Example it's shown how to implement the chosen possibility.
At the end there will be shown a prospect of the future on the basis of the examined methodic.

Inhalt

1 Erklärung zur Diplomarbeit

Ich versichere, dass ich diese Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Lüneburg, den _____

_____

Andreas Krohn

2 Einleitung

Das Flugzeugbauunternehmen Airbus setzt in der Auftragsausschreibung seit einigen Jahren auf eine neue Philosophie. Nicht mehr jeder Dienstleister kann sich auf diese Ausschreibungen bewerben. Nur so genannte „Key-Supplier“ haben die Möglichkeit, Aufträge zu übernehmen.
Das Unternehmen AEROTEC Engineering GmbH, in der diese Diplomarbeit entstand, bestehend aus den Bereichen

- ‚Engineering & Design’,
- ‚Technical Documentation’,
- ‚Consulting’ und
- ‚IT-Services & Training’

ist in den ersten beiden Genannten ein solcher Airbus-Key-Supplier.
In Zusammenarbeit mit der Abteilung ‘IT-Services & Training’ entstand diese Arbeit.
In allen Bereichen werden auch Dienstleistungen in Form der Arbeitnehmerüberlassung angeboten.
Ein im eigenen Haus, von Engineering & Design, aktuell bearbeiteter Auftrag ist die Konfiguration vom Versorgungskanal des Airbus A340. Wie alle laufenden und neu gestarteten Konstruktionsaufträge wird auch dieser in Zukunft mit einem neuen Konstruktionswerkzeug bearbeitet werden. Airbus hat an alle Zulieferer die Auflage gegeben, dass Konstruktionszeichnungen zukünftig nur noch im Format CATIA V5 ausgeliefert werden dürfen. Hierdurch ergibt sich die Relevanz dieser Arbeit.

3 Problemstellung

3.1 Konstruktionswerkzeuge

In der Luftfahrtbranche wird wie in den meisten Konstruktionsbetrieben bis heute an zweidimensionalen (2D) Konstruktionsmethodiken festgehalten. Bei diesen 2D-Tools ist der Ansatz, das klassische technische Zeichnen durch den Computer zu unterstützen. Man entwirft mit dem Rechner Zeichnungen, die man auch auf dem Papier erstellen könnte. Die Bauteile gewinnen erst den räumlichen Ausdruck durch das Zusammenfügen der verschiedenen Ansichten in der isometrischen Ansicht. Ein 2½D-Programm unterstützt die Erstellung dieser ISO-Ansicht.

Die neue Generation von Konstruktionssoftware verfolgt einen ganz anderen Grundgedanken. Man möchte schon am Rechner 3D entwerfen. Durch die stark angestiegene Rechenleistung der heutigen Computersysteme ist es möglich geworden, räumliche Ansichten zu realisieren, die auf dem Monitor fließend aus allen Blickrichtungen betrachtet werden können, ohne dass das bei der Drehbewegung störend ruckelt. Weiterhin ändert sich die Art der Konstruktion dahingehend, dass man vorwiegend versucht, objektorientiert zu konstruieren. Bauteilelemente werden nicht mehr an Ihrer Position im Raum festgemacht, bezogen auf ein zentrales Koordinatensystem, sondern sind in Ihrer Lage abhängig von ihren Nachbarelementen. Ein Vorteil dieser Methodik ist, dass die Konstruktion nicht auseinander reißt, wenn man ein Teil verschieben muss. Die angehängten Elemente verschieben sich abhängig von der Änderung mit.

Auch im Airbus Umfeld wird nun stark an der Umstellung von der 2D- auf die 3D-Konstruktion gearbeitet. Das schmale Angebot an Konstruktionswerkzeugen, welche es ermöglichen, Zeichnungen im Format CATIA V5 auszuliefern, deutet darauf hin, dass auch AEROTEC Engineering nicht umhin kommt, CATIA V5 von Dassault Systèmes® einzusetzen.

3.2 Aufgabenstellung

Um der geforderten höheren Qualität bei steigendem Kostendruck gerecht zu werden, sollen die nötigen Arbeitsschritte zum Positionieren der Elemente in einem Versorgungskanal minimiert werden. Es sollen unnötige Arbeitsvorgänge erkannt und eingespart werden, ohne Qualitätseinbußen zu verzeichnen. Eine Möglichkeit hierfür ist die Automatisierung der Vorgänge. In der vorliegenden Arbeit soll untersucht werden, inwieweit CATIA V5 hier unterstützend einwirken kann.

Abb. 3-1 Versorgungskanal einer A300-600 [13]

4 Ist-Analyse

4.1 Beschreibung des Ist-Zustandes

Die Abteilung ‚Engineering & Design' hat im Schnitt 10-mal im Jahr den Auftrag zur Positionierung der Bauteile des Versorgungskanals des A340. Der zeitliche Aufwand pro Auftrag entspricht hier im Mittel 130 Mannstunden.

Jeder Kunde hat bezogen auf die Innenausstattung individuelle Wünsche, daher muss auch der Versorgungskanal bei jedem Auftrag angepasst werden.

4.1.1 Der Versorgungskanal

Als Versorgungskanal wird der Bereich in einem Flugzeug bezeichnet, der sich über den Passagieren befindet. Als Fluggast erkennt man ihn, weil dort die Versorgungseinheit (VE) untergebracht ist. In der VE befindet sich die Anzeige „fasten seatbelt“ und „no smoking“. Außerdem sind hier die Individualbeleuchtung und der Serviceknopf untergebracht.

Als weiteres wichtiges Bauteil des Versorgungskanals gilt die O_2-Box. Diese beherbergt die Atemmasken, die im Falle eines Druckabfalls in der Kabine automatisch herunterfallen. Je nach Flugzeugtyp und Ausstattungsmerkmal befinden sich im Versorgungskanal zusätzlich die Individualbelüftung und die Videoscreens. Die freibleibenden Bereiche werden mit Leerblechen, so genannten „Infill-Panels“ aufgefüllt. All diese Bauteile werden in die Schienen des Versorgungskanals eingehängt bzw. eingeschraubt.

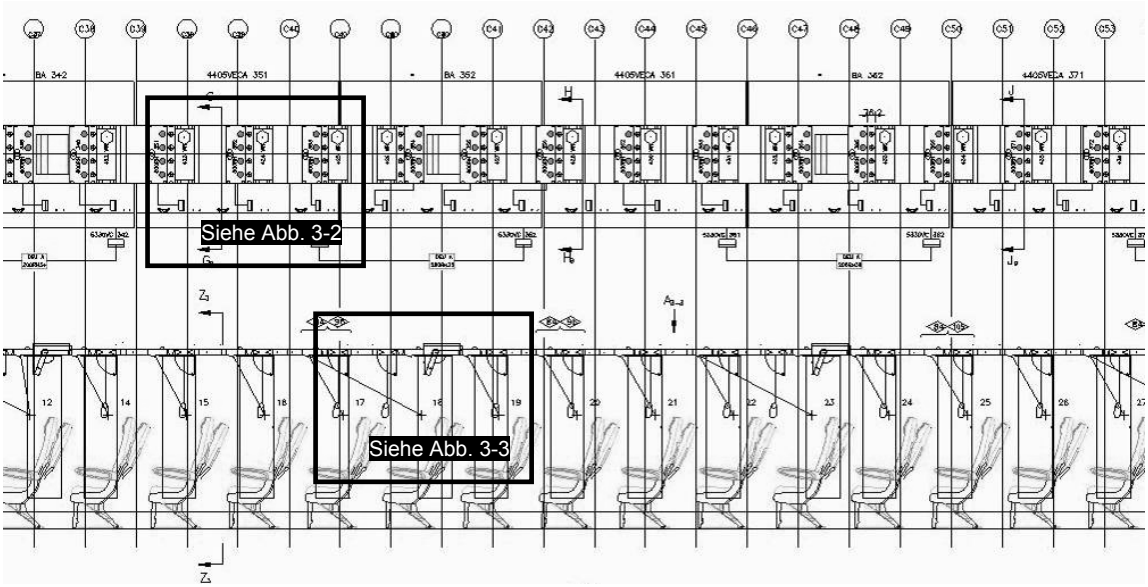


Abb. 4-1 Ausschnitt aus einer angelieferten Airbus-Zeichnung

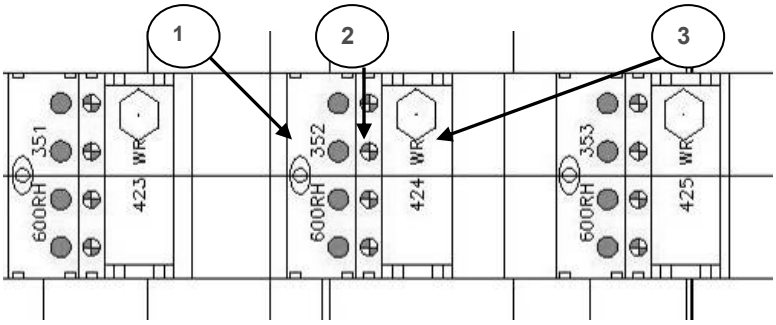


Abb. 4-2 Einbauteile des Versorgungskanals Unteransicht

1- Versorgungseinheit
2- Individualbelüftung
3- O_2-Box

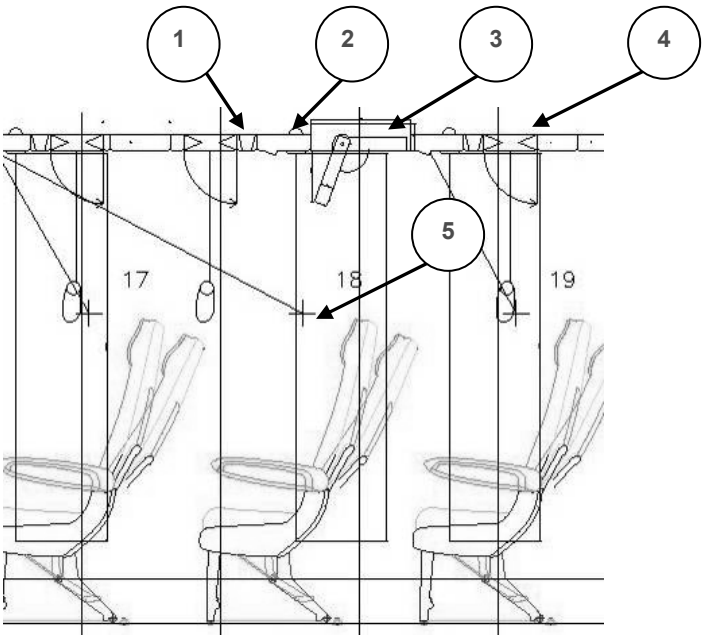


Abb. 4-3 Seitenansicht Sitzreihen mit Versorgungskanal

1- Individualbelüftung
2- Versorgungseinheit
3- Videoscreen
4- O_2-Box
5- Augenpunkt

4.1.2 Das Werkzeug

Zurzeit werden die Zeichnungen mit dem Programm CCD (**CATIA C**adam **D**rafting [CADAM = **C**omputer Graphic **A**ugmented **D**esign **A**nd **M**anufacturing System]) von Dassault Systèmes® erstellt und bearbeitet. Es handelt sich hier um ein interaktives, modular aufgebautes 2½ D-Konstruktionsprogramm.

„Die CAD-Modelle müssen sicht- und koordinatengerecht aufgebaut werden. Koordinatengerechter Aufbau von CAD-Modellen heißt, dass grundsätzlich auf ein gemeinsames, einheitliches Produktkoordinatensystem Bezug genommen wird.“ [26].

2½D bedeutet bei Konstruktionsprogrammen, dass man die verschiedenen Ansichten wie Vorder-, Seiten- und Draufsicht zwar einzeln zeichnet, da sie sich aber alle auf dasselbe Koordinatensystem beziehen, sind die Zeichnungen nicht unabhängig voneinander. Man kann z.B. die isometrische Ansicht aus den Zusammenhängen der anderen Ansichten erstellen.

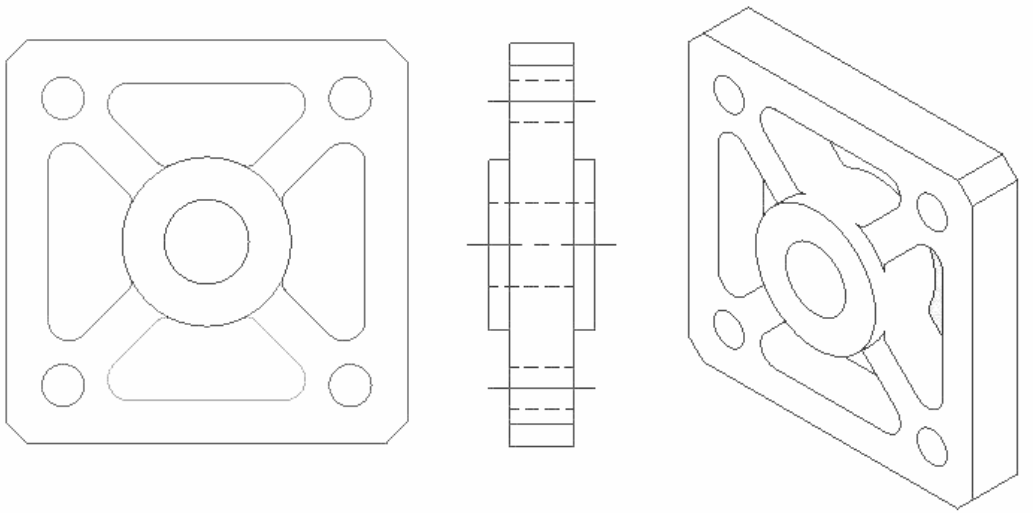


Abb. 4-4 Front-, Seiten- und Isometrische Ansicht eines Bauteils

Bauteile werden im 2D-Raum konstruiert. Teilbausteine können als Detail gespeichert werden. Diese Details können später wiederum in andere Teile eingeladen werden. Wird das Ursprüngliche geändert, so ändern sich alle Eingefügten auch. Diese Philosophie steht auch weiterhin bei CATIA V5 im 3D-Raum zur Verfügung.

4.1.3 Die Arbeitsweise

Wie schon beschrieben liefert Airbus momentan eine CCD-Zeichnung (Abb. 4-1. Hier sind die Sitzreihen, sowie auch die Video-Screens bereits nach Kundenwunsch positioniert. Außerdem befinden sich die VE's, die O_2-Boxen und die Individualbelüftungen bereits in dem Versorgungskanal, sind aber noch nicht an ihrer endgültigen Position. Die Aufgabe des Ingenieurs ist nun, nach den Vorgaben von Airbus die Infill-Panels und die genannten Bauteile zu positionieren. Das bedeutet, dass die Infill-Panels eingefügt werden müssen, und die zu positionierenden Bauteile innerhalb der Toleranzen verschoben werden, bis alle Regeln berücksichtigt wurden. Hierzu müssen nahezu alle Bauteile mehrmals per Hand bewegt werden.

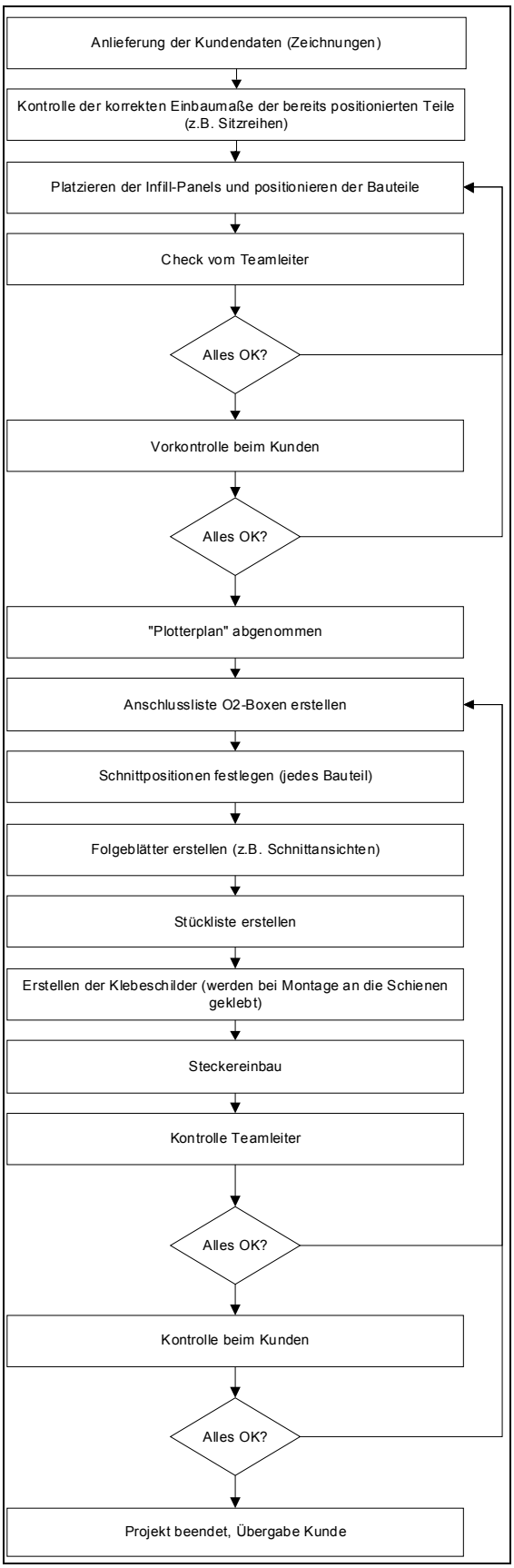


Abb. 4-5 Projektablauf Ist-Zustand

In der Grafik (Abb.4-5) ist der typische Projektablauf als Flussdiagramm dargestellt. Zunächst werden die von der Fachabteilung bei Airbus positionierten Teile mit den vorgegebenen Maßen kontrolliert und nach Bedarf berichtigt. Anschließend wird am vorderen Teil des Versorgungskanals mit der Platzierung der Infill-Panels begonnen. Ist eine Platzierung nicht ohne das Verschieben der angrenzenden Bauteile im Kanal möglich, werden die Elemente innerhalb der unten angegebenen Positionsvorgaben neu positioniert. In dieser Weise wird der gesamte Versorgungskanal durchgearbeitet. Ist diese Positionierung abgeschlossen, so muss der Teamleiter und anschließend die Fachabteilung eine Vorkontrolle durchführen. Wird die Zeichnung hier ohne Mängel abgenommen, so ist dies der so genannte „Plotterplan“. Hier sind noch keine Referenzen zu den genauen Bauteilen angegeben, und es wurden keine Schnittansichten und Zusatzinformationen zur Installation eingearbeitet.
Im Folgenden werden der Steckereinbau, Anschlusslisten, Schnittansichten, Stücklisten, und Klebeschilder erstellt. Sind diese Unterlagen komplett, so erfolgt noch die Endabnahme durch den Teamleiter und den Kunden.

Diese Diplomarbeit konzentriert sich laut Aufgabenstellung auf die Positionierung der Bauteile und der Infill-Panels.

4.1.4 Positionsvorgaben

Folgende Teile sind im Versorgungskanal des A340 enthalten:

- O_2-Boxen
- Videoscreens
- Versorgungseinheiten (VE)
- Individuelle Belüftungen (Air-Box)
- Infill-Panels in den Größen
    - 1, 2, 6, 9 und 16 Inch

Zur Positionierung der Bauteile geht man von den Positionen der Sitzreihen aus. Wie in Abb. 4-3 zu erkennen ist, besitzt jeder Sitzplatz, einen eigenen Augenpunkt, der im statistischen Mittel die Position des Augenmittelpunktes des Fluggastes repräsentiert. Ausgehend von diesem Punkt wird in der 2D-Ansicht über den horizontalen Abstand die Position der Bauteile festgelegt.

- Die **O_2-Box** sollte so eingeplant werden, dass der Mittelpunkt der O_2-Box 0 bis 1 Inch vor dem Augenpunkt liegt. Dies stellt sicher, dass der Passagier die Atemmaske im Notfall leicht erreichen kann.
- Die **VE** sollte vor der O_2-Box positioniert werden.

Die Kombination aus den ersten beiden Auflagen ist notwendig, um zu gewährleisten, dass der Vordermann beim Zurückstellen seiner Sitzlehne keine Schattenbildung der Individualbeleuchtung des Hintermanns verursacht, und dieser z.B. nicht mehr genug Licht zum Lesen zur Verfügung hat.

- Die **Videoscreens**, welche vom Kunden schon vorpositioniert sind, dürfen nur um ± 1 Inch verschoben werden.
- Für die **Air-Box** sind keine Einschränkungen gegeben.
- Nach der Positionierung der Bauteile müssen die frei gebliebenen Bereiche mit den **Infill-Panels** aufgefüllt werden.
  Grundsätzlich soll versucht werden, möglichst große Panels zu nutzen, da diese im Verhältnis zu vielen Kleinen ein geringeres Gewicht aufweisen.
  Das Panel in der Größe 1 Inch kann nicht in die Schienen eingeschraubt werden, sondern muss an die angrenzenden Bauteile geklippt werden. Hier ist zu beachten, dass dies nicht an der O_2-Box und auch nicht an einem Infill-Panel 16 Inch möglich ist.

4.1.5 Wirtschaftliche Betrachtung

Der entstehende Kostenumfang für die Bearbeitung eines Auftrages ergibt sich aus dem gemittelten Bearbeitungsaufwand von 130 Stunden für den Gesamtauftrag. Bei einem internen Kostensatz von 50€ pro Konstrukteursstunde ergeben sich pro Auftrag Kosten in Höhe von 6500€.
Die momentane Auftragslage ergibt eine Häufigkeit von 10 Aufträgen im Jahr.
Die Erstplatzierung der Bauteile im Versorgungskanal umfasst einen zeitlichen Aufwand von 16 Arbeitsstunden, was einem Aufwand von 800€ entspricht.

4.2 Bewertung des Ist-Zustandes

4.2.1 Stärken

Der Vorteil der manuellen 2D-Bearbeitungsart liegt in der Möglichkeit, eine kurzfristige Änderung von Kundenwünschen unkompliziert einzuarbeiten. Auch bei Positionierungsproblemen innerhalb der Vorgaben können sinnvolle Lösungsvorschläge individuell erarbeitet werden. Bei Konflikten mit den Vorgaben des Kunden können Lösungsvorschläge kurzfristig erarbeitet und der Fachabteilung bei Airbus zur Entscheidung vorgelegt werden. Sollte das angelieferte Dokument, in dem die Bauteile noch positioniert werden sollen, schon fehlerbehaftet sein (z.B. falsche Bauteile eingezeichnet), so kann das von dem Konstrukteur aufgrund der Erfahrung leicht erkannt werden.

4.2.2 Schwächen

Die „Erstpositionierung“ nimmt unverhältnismäßig viel Zeit in Anspruch. Außerdem handelt es sich hier um eine Abarbeitung einfacher und logischer Zusammenhänge. Der Auftrag ist im ersten Stadium sicher eher als Fleißarbeit zu bewerten und durch die monotonen Arbeitsschritte fehleranfälliger als andere Arbeiten. Bei einer Erweiterung der Vorgaben seitens des Kunden entsteht eine Art Regelwerk, welches zunehmend schwer zu überschauen ist. In dem Beispiel dieser Arbeit beschränken sich die Vorgaben auf die Genannten. Es kann aber durchaus vorkommen, dass die

Masse der Vorgaben sich verdoppelt oder verdreifacht. Hierdurch wird der zeitliche Aufwand stark erhöht und die Fehleranfälligkeit gesteigert.
Wird bei der Positionierung ein Fehler gemacht, so ist wie bereits in Abb. 4-5 erläutert im Besten Fall eine zeitliche Verzögerung der Auslieferung die Folge. Im Extremfall wird der Fehler weder durch den Teamleiter, noch durch die Fachabteilung erkannt. Die Folge wäre eine Unterbrechung der Installation. Die Kosten hierfür sind schwer vorherzusagen, da bei leicht zu behebenden Fehlern (z.B. falsche „Infill-Panel“-Größe eingeplant) eine Verzögerung im Minutenbereich zu erwarten ist, um das Teil zu tauschen. Ein solcher Fall wird kaum zu höheren Produktionskosten führen. Sind aber falsche Einbauteile eingesetzt oder die Einbaustelle am Ende einer Schiene zu eng berechnet worden, kann es passieren, dass die Zuleitungen (Spannungsversorgung oder Zuluft für die Individualbelüftung) falsch verlegt wurden und ausgetauscht werden müssen. In diesem Fall sind hohe Folgekosten und Verzögerungen zu erwarten. Ein solcher Extremfall ist durch die mehrfache Prüfung unwahrscheinlich, aber nicht auszuschließen.

4.2.3 Verbesserungsbedarf

Wie oben genannt ergibt sich also ein Zustand, in dem einerseits die Individualität einen großen Vorteil bringt (Reaktion auf Änderungen und Fehler), andererseits aber durch diese Individualität auch die Wahrscheinlichkeit für das Entstehen von Fehlern und der zeitliche Aufwand der Projektbearbeitung erhöht wird. Der Verbesserungsbedarf liegt also eindeutig in der Minimierung der Fehleranfälligkeit und der Reduzierung der Dauer einer Projektabarbeitung.
Es ist also eine Lösung gefragt, die bei gleicher oder besserer Qualität zusätzlich Zeit einspart. Eine Möglichkeit ist hier eine Teilautomatisierung der Abläufe. Die sich wiederholenden Vorgänge und die Beachtung der vorgegebenen Regeln könnten von einem unterstützenden Programm erledigt werden. Zeitlich ergibt sich somit ein hoher Gewinn. Durch die anschließende Möglichkeit, trotzdem per Hand Korrekturen durchzuführen, bliebe auch die Individualität erhalten. Diese Möglichkeit soll im Folgenden untersucht werden.

5 Erarbeitung des Soll-Konzepts

5.1 Projektschritte und –plan

	Vorgangsname
1	**Diplomarbeit**
2	**Phase 1**
3	Ist-Analyse
4	Informationen über die Alternativen mit Bewertung
5	**Phase 2**
6	Untersuchungen zur Realisierung
7	**Phase 3**
8	Konzeptaufstellung für den ProgrammCodeTeil
9	Vorbereitung der Modelle in Catia
10	Programm-Realisierung
11	**Phase 4**
12	Ausarbeitung der Dokumentation
13	Korrekturen und Probelesen
14	
15	**Meilensteine**
16	Projektbeginn
17	Absprache mit GF welcher Lösungsweg angedacht ist
18	Absprache mit GF zur Programmstruktur
19	Abgabe der Arbeit bei der GF
20	Abgabe der Diplomarbeit

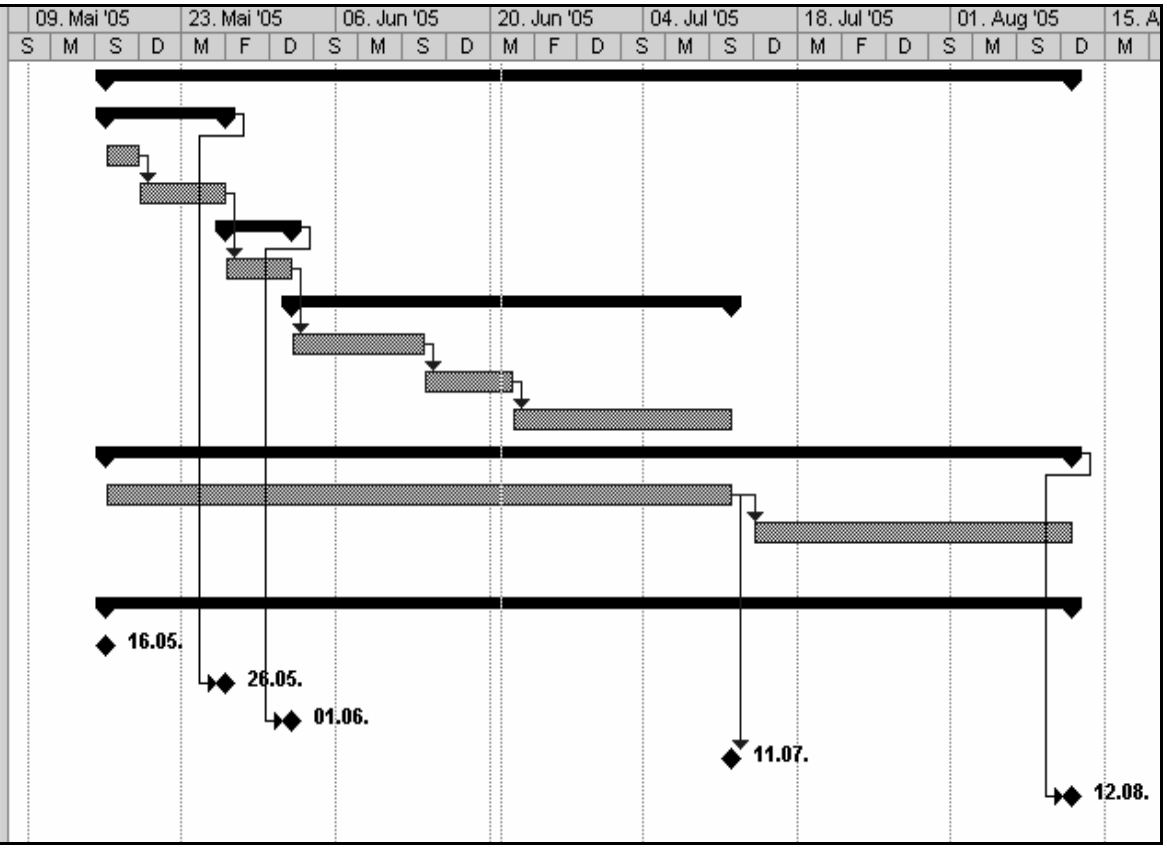


Abb. 5-1 Projektplan aus MS-Project

Wie in Abb.5-1 zu erkennen ist, umfasst die Erarbeitung dieser Diplomarbeit einen Zeitraum von 12 Wochen. Die Bearbeitung teilt sich in 4 Phasen:

- Phase 1
  - Ist-Analyse
  - Informationen über die Alternativen mit Bewertung

Die Phase 1 beinhaltet die Ist-Analyse, in der die momentane Arbeitsweise auf ihre Stärken und Schwächen hin untersucht wird und die Verbesserungsmöglichkeiten erarbeitet werden.
Anhand dieser Verbesserungsmöglichkeiten werden die Alternativen zum Ist-Zustand entwickelt und bewertet.

- Phase 2
  - Untersuchungen zur Realisierung

In der Phase 2 wird die beste Alternative auf die mögliche Realisierung hin untersucht.

- Phase 3
  - Konzeptaufstellung für den Programmcode-Teil
  - Vorbereitung der Modelle in CATIA V5
  - Programm-Realisierung

Phase 3 beinhaltet die Konzeptaufstellung für den Programmcode Teil. Hier entsteht der Programmablauf in Form eines Diagramms. Außerdem werden die nötigen Modelle in CATIA vorbereitet und im Anschluss wird das Konzept in CATIA V5 realisiert.

- Phase 4
  - Ausarbeitung der Dokumentation
  - Korrekturen und Probelesen

Phase 4 ist zeitlich nicht separat zu sehen, die Ausarbeitung der schriftlichen Arbeit läuft vielmehr parallel zur Entwicklung.

Nach jeder vollendeten Phase steht ein Meilenstein, an dem eine Absprache über den Fortgang des Projektes mit den Projektbetreuern stattfindet.

5.2 Anforderungsbeitragende

In der folgenden Abbildung (Abb. 5-2) sind nach der normierten Vorgabe der Unified Modelling Language die Anforderungsbeitragenden dargestellt. An das zu realisierende Programm, welches als „Positioner“ benannt ist, werden von verschiedenen Seiten Anforderungen gestellt.

- Kunde
  - Der Kunde verlangt, dass die entsprechenden Vorgaben zur Positionierung eingehalten werden.
- Konstrukteur
  - Der Konstrukteur möchte eine Arbeitserleichterung durch das Programm erhalten.
- Abteilungsleitung
  - Der Abteilungsleiter verlangt eine qualitativ mindestens gleichwertige Arbeit des Programms im Verhältnis zur manuellen Abarbeitung der Aufträge. Zusätzlich muss die Bearbeitung schneller und damit kostengünstiger erfolgen.

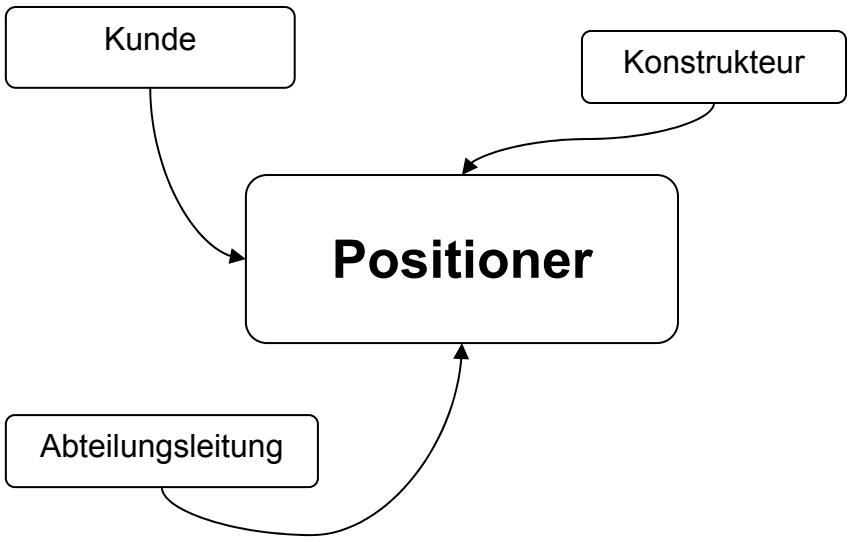


Abb. 5-2 Anforderungsbeitragende

5.3 Lösungsalternativen

Zu beachten ist, dass nicht nur eine Umstellung auf CATIA V5, sondern nach heutigem Stand auch die Positionierung aller Bauteile mit Ausnahme der Sitzreihen und der LCD-Screens erfolgen soll. Es ist wahrscheinlich, dass sich der genaue Workflow noch des Öfteren ändern wird. Da diese Untersuchung aber beispielhaft ist, wird der heutige Stand als Vorgabe angenommen. Als Arbeitsgrundlage steht die CATIA Version 5 Release 14 unter Windows zur Verfügung.

5.3.1 CATIA V5

Da das System CATIA V5 in der heutigen Form noch nicht lange auf dem Markt verfügbar ist, sind auch ergänzende Programme zur Funktionserweiterung von CATIA V5 nicht allzu verbreitet. 1999 wurde CATIA in der Version 5 erstmalig vorgestellt. Jedoch haben anfängliche Probleme zu einem gebremsten Kaufverhalten in der Industrie geführt. Die hohen Kosten für ein Unternehmen, das eine Umstellung auf ein neues System anstrebt, stehen den neuen Möglichkeiten, die man hierdurch erhält, gegenüber. Die Investition in ein Programm, das sich nicht auf dem Markt etabliert, kann schnell zu einem Existenzproblem führen.
Da sich alleine in Deutschland nun mit Audi, BMW, DaimlerChrysler, MAN, Volkswagen, Porsche und Airbus diverse Großunternehmen für die Einführung dieses Systems entschieden haben und auch von Zulieferern verlangen, Ihre Unterlagen in dem Format von CATIA V5 abzuliefern, ist eine gute Marktstellung des Produktes gegeben. Dies bedeutet, dass von nun an mit Sicherheit weitere Firmen in CATIA V5 und funktionserweiternde Tools investieren. Somit wird auch die Neuentwicklung von ergänzenden Systemen vorangetrieben.
Als Zusatzmodul sind zu dem Standardpaket von CATIA V5 auch Teile der Knowledgeware verfügbar. Hierbei handelt es sich hauptsächlich um Regeleditoren. Sie erleichtern Überprüfungen der Einhaltung von Vorgaben in der Konstruktion. Außerdem bietet CATIA V5 eine eigene Programmierschnittstelle, welche es ermöglicht, über Makro-Programmierung direkt im Programm die Funktionen von CATIA V5 zu nutzen und zu automatisieren. Auch diese Möglichkeit soll untersucht werden.

5.3.2 Knowledgeware CATIA V5

Die Knowledgeware von CATIA V5 wurde generell für die Bauteilekonstruktion entwickelt. Sie kann aber auch für das Assembly Design (Zusammenbaudateien [Produkte]) genutzt werden. Die Hauptkomponenten sind der „Knowledge Advisor“ und der „Knowledge Expert“ [7,18].

Knowledge Advisor (KWA):

Die für diese Aufgabe wichtigen Funktionen des KWA sind

- Formeln/Parameter
  - Es können auf einzelne Komponenten der Konstruktion Formeln angewandt werden, wie z.B.:
    Die Länge der Kante A = Länge der Kante B x 2
  - In der weiteren Bearbeitung des Werkstückes ist die Formel (solange sie aktiv geschaltet ist) immer berücksichtigt, wird z.B. die Länge der Kante B geändert, so ändert sich automatisch die Länge der Kante A gemäß der Formel.
  - Im Bereich Parameter können neue Parameter erstellt werden, die im Weiteren mit Regeln oder Prüfungen angesprochen werden können.
  - Allgemeines Ingenieurwissen reicht ohne Kenntnis einer Programmiersprache aus, um solche einfachen Regeln zu erstellen.

- Prüfungen
  - Mit einer Prüfung kann eine Art Vergleich IST/SOLL ausgeübt werden.
  - Darf die Kante A z.B. nicht kürzer als 200mm sein, und durch die Veränderung der Kante B aus der oben genannten Formel ergibt sich ein solcher Regelverstoß, gibt es drei Möglichkeiten dieses bekannt zu geben.
    - Im Strukturbaum erscheint eine rote Ampel  (die ansonsten grün ist)
    - Ampel plus Informationsfeld
    - Ampel plus Warnungsfenster

- Regeln
  - Mit Regeln kann man einfache if/else-Anweisungen formulieren. Es werden einzelne Parameter überprüft,. und wenn eine Bedingung zutrifft, wird die vorgegebene Aktion auf einen anderen Parameter ausgeübt.
  - Ist z.B. die Kante B länger als 300mm, so soll eine dritte Kante (Kante C) die gleiche Länge erhalten.
  - Auch hier wird man beim Erstellen der Regeln vom Programm so unterstützt, dass jeder Konstrukteur ohne weitere Vorkenntnisse eine solche Regel erstellen kann.
- Konstruktionstabellen
  - Man kann Konstruktionstabellen erzeugen oder bestehende Tabellen aus Excel einlesen.
  - In Konstruktionstabellen werden Beziehungen zwischen den einzelnen Komponenten eines Bauteils hinterlegt. Dies geschieht ähnlich dem Erstellen von Regeln, nur dass nicht alles einzeln angelegt wird, sondern alle zusammen in einer Tabelle erstellt werden.
- Verhaltensprozess
  - Hier können alle Aktionen, die mit dem KWA zu realisieren sind, mit Hilfe der Programmiersprache Visual Basic realisiert werden. Somit scheidet diese Möglichkeit für einen Konstrukteur ohne Programmierkenntnisse aus.

Knowledge Expert (KWE):

Grundsätzlich kann sich der KWE neben den Funktionen des KWA nicht nur auf Geometrien beziehen, sondern zusätzlich auch auf so genannte Features, wie z.B. eine Kantenverrundung, ein Offset oder eine Biegung [6].

Zusätzlich werden die erstellten Regeln und Prüfungen auf Wunsch nicht in dem Bauteil mitgespeichert, sondern stehen als selbstständige Regelsets zur Verfügung, was z.B. erlaubt, bestimmte Normvorgaben, die in einem solchen Regelset gespeichert sind, auf alle entworfenen Teile anzuwenden, ohne sie jedes mal gesondert zu implementieren.

- Expertenprüfung
  - Die Funktion ähnelt sehr der KWA-Prüfung. Zusätzlich kann automatisch ein Bericht erstellt werden, in dem die Abweichungen ausgegeben werden. Außerdem kann eine automatische Korrektur veranlasst werden. Entweder wird eine Infobox geöffnet, eine referenzierte Internetseite aufgerufen oder eine vorgegebene Korrektur am Werkstück wird ausgeführt.
- Expertenregel
  - Auch hier ist die KWA-Regelfunktion komplett integriert, kann sich jedoch wie oben beschrieben auch auf die genannten Features beziehen.
- Expert Rule Set
  - Hierüber können gespeicherte Regelstets wie die oben angesprochene Normvorgabe in das aktuelle Dokument geladen werden. So können diverse vorgefertigte Sets in ein Dokument integriert werden.

Ein einfaches Beispiel für den Einsatz von KWE: Es soll ein Loch in ein Blech gebohrt werden. Das Loch ist aber zu dicht an der Kante angesetzt, so dass das Material den entstehenden Kräften bei Belastung der Bohrung nicht standhalten würde. Die Regel erkennt noch bei der Erstellung der Bohrung den Verstoß und schlägt vor, entweder die Bohrung weiter nach innen zu setzten, oder die Materialstärke zu erhöhen. Wahlweise kann die Lösung dann auch gleich automatisiert vom System durchgeführt werden.

5.3.3 Makro-Programmierung in CATIA V5

CATIA V5 bietet schon im Standardpaket die Makroschnittstelle, wie alle gängigen Microsoft Office-Bausteine. Diese Schnittstelle ermöglicht es, einzelne konstruktive Vorgänge sowie komplexe Aufgaben zu automatisieren.
Hierfür werden drei Programmiersprachen angeboten, welche nicht mehr Teil des allgemeinen Ingenieurwissens eines Konstrukteurs sind. Hierzu ist eine spezielle Ausbildung in den Programmiersprachen erforderlich.
Als Untermenge der bestehenden Skriptsprache „Microsoft Visual-Basic-Script“ (MS-VBScript) wird die Sprache CATScript angeboten. Sie arbeitet objektorientiert mit Objekten und Methoden. Es handelt sich hierbei um eine reine codebasierte und dadurch unkomfortable Programmierung, welche jedoch den großen Vorteil hat, dass

der erstellte Code unter der Unix- und Windows-Version von CATIA V5 lauffähig ist. Da die Windowsversion von CATIA V5 jünger ist, kann der erstellte Code auch auf Unix-Maschinen angewandt werden. Da jedoch die Hardware für einen Windowsrechner sehr viel günstiger ist, wird die Verbreitung der Windowsversion extrem ansteigen.
Als weitere Sprache wird CATIA-VBScript angeboten. Hier ist der gesamte Befehlsumfang von MS-VBScript enthalten. Jedoch können diese Programme nur noch in der Windowsversion ausgeführt werden.
Als dritte Möglichkeit steht die Sprache CAT-VBA zur Verfügung. Hierbei handelt es sich um eine CATIA spezifische Version von Microsofts „Visual Basic for Applications“, der Standard Makrosprache im Officebereich von Microsoft [4,5]. Auch diese Programmteile können nur noch unter Windows ausgeführt werden. Die Erstellung der Programme wird hier aber durch einen komfortablen Editor unterstützt, welcher auch schon grafische Elemente zur Erstellung bereithält.

Sprache	Dateiformat	Beschreibung	Anwendungsbereich
CATScript	*.CATScript	Reduziertes VBScript Interpreter (WIN, UNIX)	Makros (WIN und UNIX), CATIA Knowledge Ware
CATIA-VBScript	*.catvbs	Vollständiges VBScript Interpreter (WIN, UNIX seit V5R8)	Makros (WIN), CATIA Knowledge Ware
CATIA-VBA	*.catvba	Visual Basic Application Compiler (WIN)	Menügestützte Anwendungen (WIN)

Tab. 5-1 Übersicht der Makrosprachen von CATIA V5 [2]

Eine weitere Unterstützung bietet der Makrorecorder, mit welchem Vorgehensweisen aufgezeichnet werden können.Als Ergebnis wird ein Programmcode geliefert, welcher im Idealfall schon ein fertiges Makro darstellt. Diese Art der Makroerstellung funktioniert bei leichten Aktionsabfolgen. Sobald die zu automatisierenden Arbeitsschritte umfangreicher werden, wird der Code nur teilweise erstellt.

Als Nebeninformation soll hier angeführt werden, dass CATIA auch als Zusatzmodul eine echte Programmierschnittstelle anbietet, welche es in einer C++ oder Java ähnlichen Programmiersprache (CAA RADE // **CATIA A**pplication Development; **R**apid **A**pplication **D**evelopment **E**nvironment) ermöglicht, echte Programmteile für

CATIA V5 zu entwickeln [15]. „Echt“ soll hier bedeuten, dass das eigentliche Programm CATIA V5 somit kundenspezifisch weiterentwickelt werden kann und nicht nur die freigegebenen Funktionen der Makroschnittstelle genutzt werden können, sondern vielmehr auch Einfluss auf interne Strukturen von CATIA genommen werden kann. Es ist möglich, Workbenches zu entwickeln, wie auch Dassault Systèmes® sie auf den Markt gibt. Eine solche Schnittstelle ist sicher für die Lösung des in dieser Arbeit betrachteten Problems zu mächtig (und zu kostenintensiv), sollte hier aber der Vollständigkeit halber erwähnt werden.

5.3.4 Pace

Die Firma Pace mit Sitz in Berlin hat 1998 erstmals die Plattform „Pacelab“ auf dem Markt vorgestellt. Auf dieser Plattform bauen bis heute diverse Tools auf, die überwiegend für den Einsatz in der Luftfahrt entwickelt wurden. Eines ist das Tool „Pacelab-Cabin“. Hierbei handelte es sich um ein Tool zu Auslegung von Flugzeugkabinen. Der Ansatz von „Pacelab-Cabin“ liegt allerdings schwerpunktmäßig nicht auf der Seite der Konstruktion, sondern auf der Visualisierung der ausgelegten Kabine. Es werden z.B. nicht die originalen Bauteile aus der Konstruktion verlinkt, was bedeutet, dass bei einer konstruktiven Änderung eines Bauteils dieses nicht direkt in der Auslegung der Kabine berücksichtigt wird. Eine Änderung müsste hier per Hand erfolgen.

Das Ergebnis einer entworfenen Kabine ist ein Layoutplan, der darstellt, an welchen Positionen bestimmte Bauteile gesetzt würden. Das Ergebnis ist also ähnlich dem von CCD und CATIA V5, beachtet aber keine konstruktionsbedingten Auflagen. Im Hintergrund liegt der eigentliche Vorteil von Pacelab. Es gibt eine so genannte „Wissensdatenbank“, in der allgemeine Regeln zur Positionierung von Teilen hinterlegt sind. Außerdem werden kundeneigene Daten dort gespeichert, so dass nach Eingabe aller Rahmenbedingungen ein Kabinenlayout nahezu automatisch erstellt wird. Die Positionierung entsteht aber auch hier noch 2½D. Vereinzelt können Übersichten 3D dargestellt werden, was das Verständnis der Zeichnung für den Laien einfacher macht [25].

In Zusammenarbeit mit Airbus hat Pace ein eigenes Tool zu Positionierung und Auslegung eines Versorgungskanals entwickelt, was prinzipiell die von uns

geforderten Eigenschaften erfüllt. Jedoch arbeitet dieses System nur bei relativ einfachen Flugzeugen (Single Aisle) zufriedenstellend. In den größeren Flugzeugtypen (Long Range) funktioniert das Tool nicht akzeptabel, so dass es momentan nicht produktiv eingesetzt wird. Für die neuen Flugzeugprogramme gibt es noch kein Pace-Tool, das diese Funktionalität bietet.

5.4 Bewertung der Alternativen

5.4.1 Technische Betrachtung

Die Knowledgeware von CATIA V5 ist für die Einhaltung von vorgegebenen Regeln prädestiniert, da diese Tools genau dafür entwickelt wurden. Der KWA würde für die gestellte Aufgabe ausreichen, da hier kein Unternehmenswissen von zentraler Stelle verbreitet werden muss. Lediglich das Know-how des Ingenieurs muss in das Produkt implementiert werden. Diese Auflage erfüllt der KWA schon. Jedoch wird hierdurch nur ein geringer Zeitgewinn erzielt. Eine Automation ist nicht zu realisieren, da man hierfür schon die CAA-Schnittstelle oder die Makroprogrammierung nutzen müsste. Die CAA-Schnittstelle ist wie schon erwähnt für die Erstellung von vollwertigen Programmteilen, bzw. zur Erstellung von eigenen Workbenches gedacht und für die gestellte Aufgabe sicher zu mächtig.
Die Makroprogrammierung ermöglicht eine Automation der Vorgänge. Hier muss der Ingenieur bei guter Programmrealisierung nur noch administrativ eingreifen.
Pace liefert zwar eine Schnittstelle für CATIA V5, kann jedoch nicht in CATIA V5 direkt eingebunden werden. Es handelt sich hier um eine eigenständige Software, die zwar die geforderten Funktionen liefert, jedoch für die neuen Flugzeugprojekte nicht zufrieden stellend arbeitet.

5.4.2 Wirtschaftliche Betrachtung

Im Vergleich der Investitionskosten der jeweiligen Lösungsalternativen ist nicht nur die Anschaffung der Tools, sondern auch die Wartung der Programme zu beachten.
Die Firma Pace behält sich vor, Angebote für die Software nur auszugeben, wenn sie von einem realen Kaufinteresse des Kunden ausgehen kann. Die Preise werden hier individuell am Kunden ausgerichtet. Somit konnte kein Listenpreis in Erfahrung gebracht werden. Unternehmensinternen Hinweisen zufolge liegt die Software allerdings deutlich über den Anschaffungskosten der CAA-Schnittstelle.
Die CAA-Schnittstelle ist mit 30.572€ und einem Wartungskostenanteil von 4.585,80€ pro Jahr (entspricht 15% vom Anschaffungspreis) eine der teuersten Alternativen.
Der KWE liegt mit 18.350€ deutlich unter der CAA-Schnittstelle. Die Wartungskosten betragen jährlich 2.580€ (14%).
Nicht einmal die Hälfte (8.927,16€) kostet der KWA in der Anschaffung. Die jährlichen Wartungskosten betragen mit 1.251,64€ auch 14% des Anschaffungspreises.
Die Makro Schnittstelle ist die günstigste Alternative, denn sie ist schon in den Anschaffungskosten der Basissoftware CATIA V5 enthalten. Somit sind auch die Wartungskosten schon inklusive (Tab.5-2).

Tool	Einmalige Kosten (€ Brutto-Listenpreis)	Wartungskosten per anno (€ Brutto-Listenpreis)
KWA	8.927,16	1.251,64
KWE	18.350,00	2.580,00
CAA	30.572,00	4.585,80
Makro Schnittstelle	Inklusive	Inklusive
Pacelab Cabin	Lt. Hersteller keine Angabe. Individuelle Angebote	Lt. Hersteller keine Angabe. Individuelle Angebote

Tab. 5-2 Kostenvergleich Lösungsansätze

Zusätzlich zu den Kosten für die Anschaffung und Wartung sind die Kosten für die Erstellung der Lösungsvariante zu beachten. Außerdem müssen auch die

Einsparungen bei der Auftragsausführung mit Hilfe der einzelnen Lösungen verglichen werden, um die beste Alternative auszuwählen.
Die Pace Software kann hier wieder als Einzige nicht betrachtet werden, da detaillierte Informationen von dem Unternehmen nicht preisgegeben werden.
Der zeitliche Aufwand zur Erstellung der einzelnen Lösungen ist in den übrigen vier Fällen in etwa gleich groß einzuschätzen. In der Makro-Programmierung und der Programmierung mit der CAA-Schnittstelle ergibt sich kein Unterschied, da das gesamte Programm mit reiner Programmierung gelöst wird. Keine der beiden Sprachen ermöglicht gegenüber der Anderen einen zeitlichen Gewinn in der Programmierung.
Die Lösungen KWA und KWE können alleine die gegebene Aufgabenstellung nicht erfüllen und müssen immer in Verbindung mit einer der genannten Programmiersprachen eingesetzt werden. Zeitlich verhält es sich so, dass hier auch weder eine Zeitersparnis erlangt, noch eine längere Zeitdauer benötigt wird, um das Programm zu realisieren. Die Module, die nicht in der Programmiersprache realisiert werden müssen, werden in dem KWA oder KWE erstellt. Die hierfür notwendige Zeit entspricht der benötigten Zeit der Realisierung in CAA oder der Makro-Programmierung.
Der zeitliche Gewinn der einzelnen Module bei der Ausführung der Programme unterscheidet sich untereinander nicht, da alle Lösungswege das gleiche Ziel haben, nämlich die Automation der Positionierung. Die Struktur der Programme ist in jedem Fall gleich und es ergibt sich somit auch hier kein Unterschied.
Als einziges wirtschaftliches Vergleichskriterium sind also die Anschaffungs- und Wartungskosten heranzuziehen. Die genaue Aufschlüsselung in Form einer Kosten-Leistungsrechnung wird im Kapitel „Ist-/ Sollzustand“ näher erläutert.

5.4.3 Fazit

Die Realisierung des Projektes ist mit der Pace-Software im Long Range Bereich momentan nicht zufrieden stellend möglich. Die CAA-Schnittstelle ermöglicht die Umsetzung des geforderten Programms, würde sich aber für den einen Einsatz nicht rentieren. Der Preis macht sich an dem hohen Funktionsumfang fest und ist keinesfalls übertrieben, für die zu realisierende Aufgabe jedoch zu hoch. Der KWE ist

nicht für eine Aufgabe wie die Positionierung der Bauteile gedacht und seinem Funktionsumfang entsprechend kostenintensiv.
Als Möglichkeit ergibt sich der Einsatz von Makroprogrammierung und KWA.
Bei umfangreichen Aufgaben ist der parallele Einsatz zu präferieren, in dem die Regeln im KWA hinterlegt werden und die Automatisierung als Makro realisiert wird. Die Regelabfrage könnte per Makro gestartet werden, und entsprechende Reaktionen werden von dem Programm verarbeitet. Negativ ist hier jedoch, dass sehr schnell bei komplexen Aufgaben eine Überlastung der Rechenleistung entstehen könnte. Sind auch schon Einzelteile mit Regeln versehen, so wird im Zusammenbau bei jedem Verschieben oder Platzieren der KWA angesprochen. Die Datenflut der zu prüfenden Teile würde bei umfangreichen Projektteilen exponentiell ansteigen. Um dem vorzubeugen, und weil alle Prüfungen auch als Makro programmiert werden können, ist die reine Nutzung der Makro-Schnittstelle für die gestellte Aufgabe sinnvoll. Als positiven Nebeneffekt erhält man somit auch die kostengünstigste Variante. Allerdings kann ein solches Programm auch immer nur für ein festes Projekt genutzt werden, aber genau hierum geht es ja in der gestellten Aufgabe.

5.5 Vergleich Ist- und Soll-Zustand

	Ist	Soll
Handling	Manuelle Bearbeitung	Teilweise automatisiert
Technisch	CCD: 2½D	CATIA V5: 3D
Wirtschaftlich	Kosten pro Projekt	Mittelfristige Senkung der Kosten

Tab. 5-3 Ist/Sollvergleich CCD / CATIA V5

In Tab. 5-3 ist zusammengefasst dargestellt, wie in den Bereichen Handling, Technik und Wirtschaftlichkeit der Sollzustand dem Ist-Zustand gegenübersteht. Nachdem die Festlegung auf die Lösung per Makroprogrammierung erfolgt ist, ergibt sich eine gewünschte Teilautomatisierung gegenüber der reinen manuellen Bearbeitung des Auftrages im Handlingbereich. Zusätzlich ist die technische Seite von der 2½D-Umgebung in die 3D-Umgebung zu überführen, und wirtschaftlich soll sich eine mittelfristige Senkung der Kosten pro Projekt ergeben.

Das gewünschte Ergebnis der Positionierung soll hier anhand einer Grafik verdeutlicht werden. Die Bauteile sind positioniert, und auch die Leerflächen sind durch Infill-Panels verdeckt.

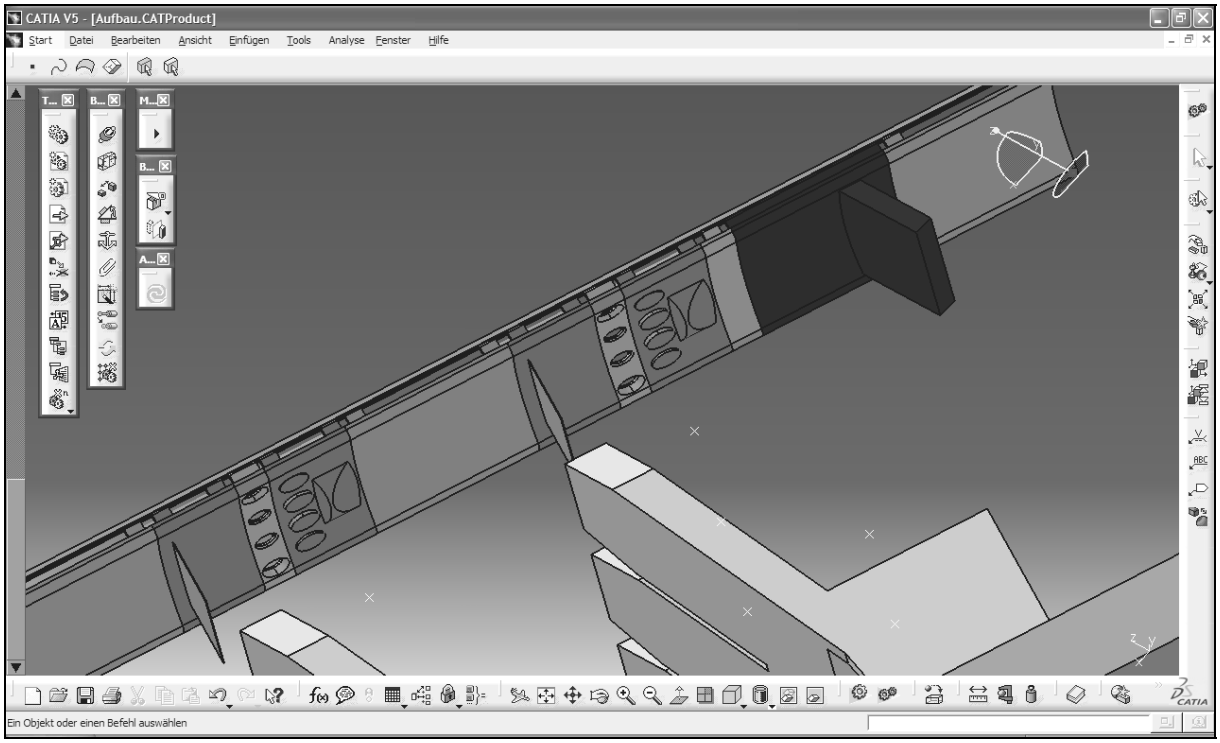


Abb. 5-3 Unteransicht des fertigen Versorgungskanals

5.5.1 Handling

Für den Einsatz des „Positioners“ sprechen folgende Punkte:

- Übernahme der zeitaufwendigen Positionierung durch das Programm.
- Nachträgliche Anpassung möglich.
- Fehler durch ermüdende Tätigkeiten werden ausgeschlossen.
- Im Idealfall nur noch überprüfende Tätigkeiten des Konstrukteurs.

Allerdings müssen die zu positionierenden Bauteile für die Verarbeitung mit dem „Positioner“ vorbereitet werden (jedoch pro Bauteil einmalig).
Diese Vorbereitung ist notwendig, damit das Programm die Bauteile nach den Vorgaben richtig umgehen kann. Auf die Vorbereitung der Bauteile wird hier später noch genau eingegangen.

5.5.2 Technische Betrachtung

Schon aufgrund der vorgegebenen Umstellung von CCD auf CATIA V5 ist technisch gesehen eine starke Umstellung zu erwarten. Auf die Realisierung dieses Projektes bezogen ergibt sich die Konstruktion im 3D-Raum anstelle des 2½D-Raumes aus CCD. Außerdem arbeitet CATIA V5 wie schon erläutert objektorientiert.
Was die Programmierung von Makros angeht ist in CCD keine Schnittstelle vorhanden. Auch Regelassistenten gibt es dort nicht, daher ist bis jetzt nur eine manuelle Abarbeitung der Projekte möglich gewesen. Erst mit CATIA V5 ist es möglich, eine Teilautomatisierung zu realisieren.

5.5.3 Wirtschaftliche Betrachtung

Die Entwicklung eines Programms laut Aufgabenstellung umfasst einen geschätzten Stundenumfang von 240 Stunden bei einem internen Kostensatz von 42€ pro Programmiererstunde. Das ergibt Entwicklungskosten von 10.080€.
Da sich die Zeitersparnis eines gesamten Projektes aber nur auf die Positionierung bezieht, welche nach dem heute eingesetzten System 16 Stunden umfasst, bezieht sich auch die Zeitersparnis pro Projekt nur auf diesen Zeitraum (entspricht bei einem internen Kostensatz für einen Konstrukteur von ca. 50€ = 800€).
Nach der neuen Methode würde die Positionierung ca. 1 Stunde plus maximal 2 Stunden Korrektur/Kontrolle in Anspruch nehmen (=> 150€).
Hinzu kommen pro Bauteil im Schnitt 20 Minuten für die einmalige Veröffentlichung von Flächen und Benennung von wichtigen geometrischen Teilen. Dieser Aufwand ist natürlich bei den ersten Aufträgen am höchsten und umfasst hier im Schnitt 6 Bauteile (=> 100€).
Da die Anschaffung des Systems CATIA V5 unabhängig von der Automatisierung durchgeführt werden muss, können diese Kosten nicht in die Rechnung mit einbezogen werden. Lediglich die Anschaffung der Programmierschnittstelle würde berücksichtigt werden müssen. Da die Makroschnittstelle die wir nutzen aber schon in der Standardversion von CATIA V5 enthalten ist, entstehen auch hier keine weiteren Kosten.
Als Fixkosten gelten also lediglich die Programmentwicklungskosten in Höhe von 10.080€.

Hinzu kommen die genannten variablen Kosten in Höhe von 250€ pro Auftrag gegenüber 800€ nach dem heutigen System[11].

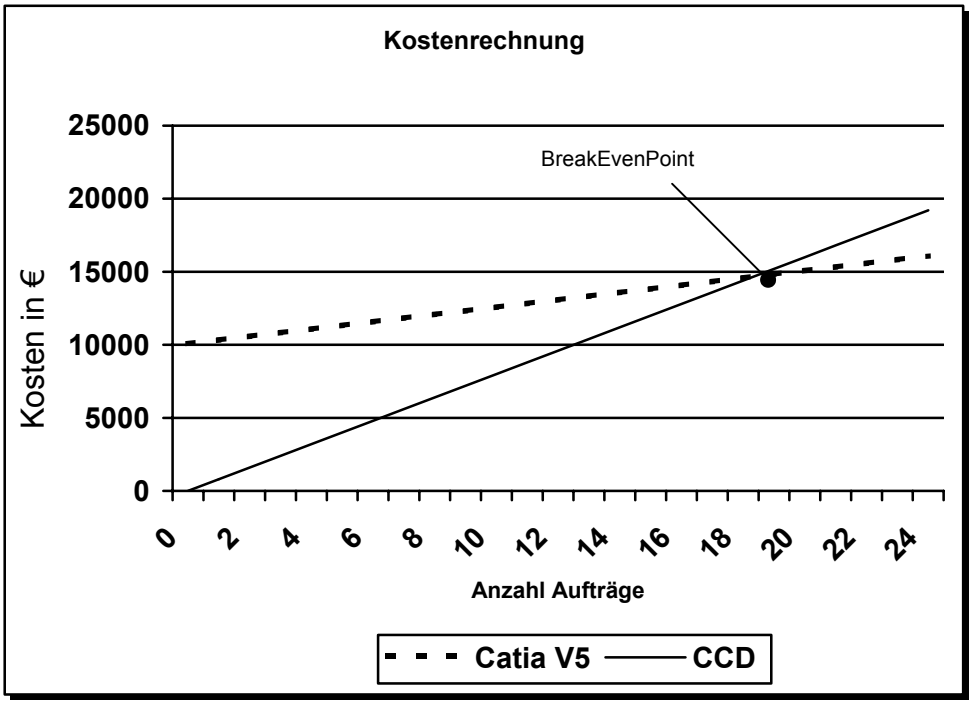


Abb. 5-4 Kosten/Leistungsrechnung

Wie man in der oben gezeigten Grafik sieht, würden sich die Kosten für den Einsatz der neuen Software schon nach dem 18.Auftrag amortisieren. Nach der heutigen Auftragslage entspricht das einem Zeitraum von nicht ganz zwei Jahren.

$$\text{Aufträge bis } \mathbf{B}\text{reak}\mathbf{E}\text{ven}\mathbf{P}\text{oint} = \left(\frac{Fixkosten_{neu}}{\left(vKosten_{alt} - vKosten_{neu}\right)} \right) \qquad (1)$$

$$\text{Aufträge bis BEP} = \left(\frac{10080€}{\left(800€ - 250€\right)} \right) = 18{,}327 \qquad (2)$$

6 Implementierung / Vorgehensweise

6.1 Ablaufdiagramm „Positioner“

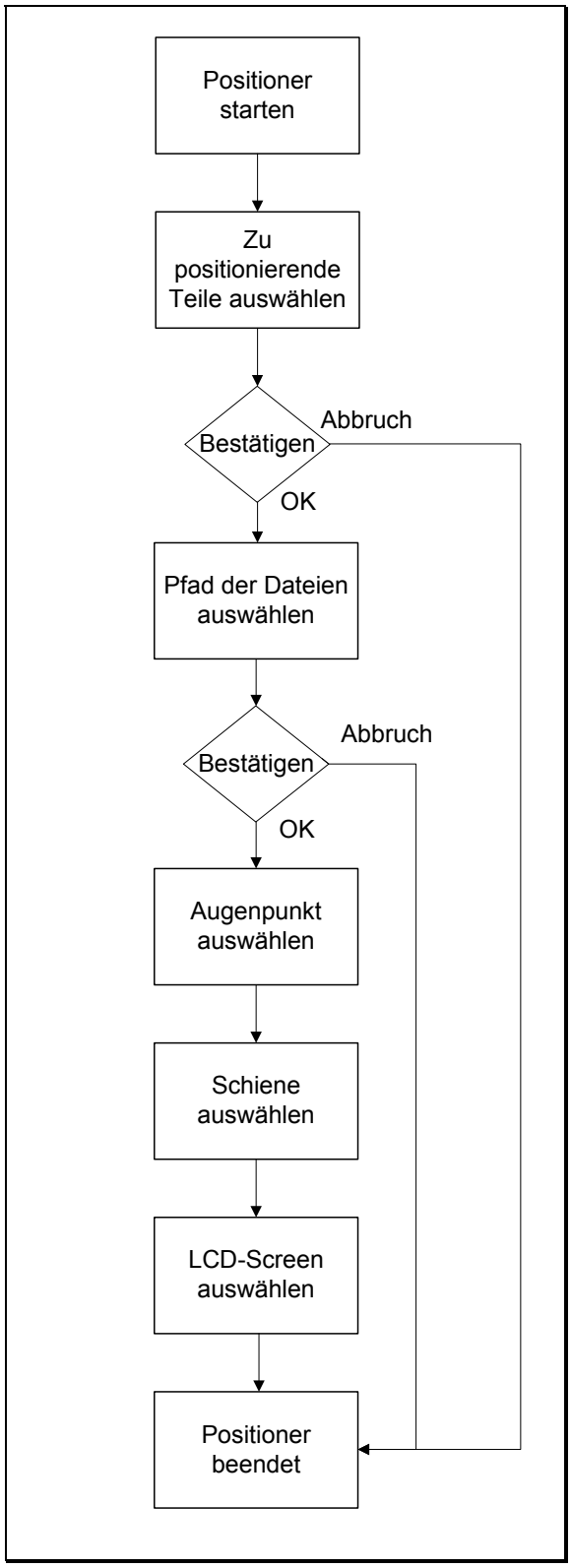


Abb. 6-1 Ablaufdiagramm "Positioner"

In Abb. 6-1 wird grafisch dargestellt, wie der Ablauf des Programms aus Sicht des Konstrukteurs gestaltet sein soll.

Ist der Positioner gestartet worden, so wir der Konstrukteur aufgefordert die zu positionierenden Teile auszuwählen. Diese Auswahl ist nötig, da nicht jeder Kunde alle Bauteile in den Kanal einsetzten möchte. So wird zum Beispiel häufig auf die Individualbelüftung verzichtet.

An dieser Stelle kann das Programm vom User auch abgebrochen werden.

Wurden die Bauteile ausgewählt, so muss der Benutzer jedem Bauteil noch eine CATIA-Datei zuordnen. Dies geschieht über einen Windows Explorer ähnlichen Eingabedialog. Auch hier kann das Programm nochmals vom User abgebrochen werden.

Wurden alle CATIA-Dateien zugeordnet, muss ein Augenpunkt in dem Kabinenlayout gewählt werden. Anschließend muss eine Schiene des Versorgungskanals angewählt werden.

Falls vom Kunden schon LCD-Screens gesetzt wurden, muss auch einer von diesen angewählt werden.

Anschließend arbeitet das Programm im Hintergrund das neue Layout aus und positioniert die Bauteile. Für den Benutzter ist das Programm nun beendet.

6.2 Zustandsdiagramm

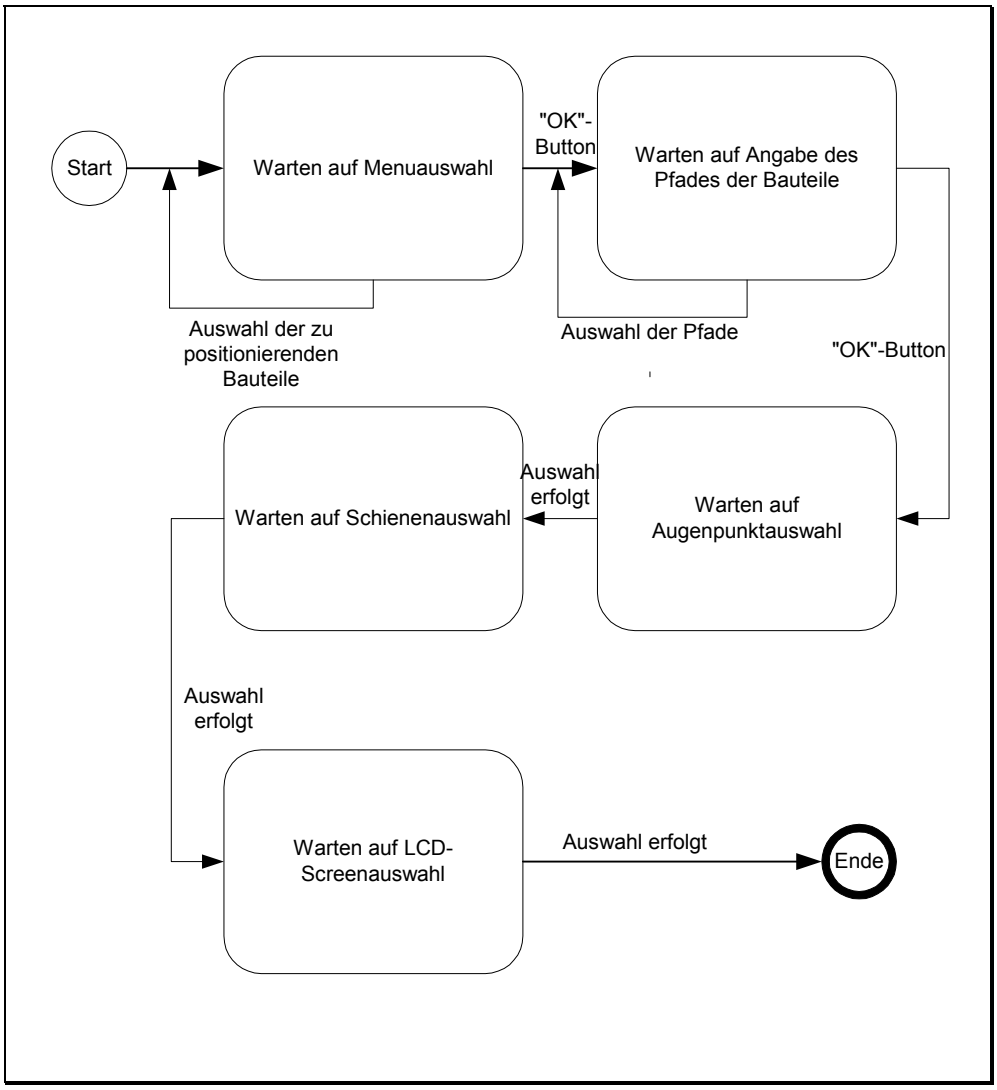


Abb. 6-2 Zustandsdiagramm "Positioner"

In Abb. 6-2 werden die Zustände, in denen sich das Programm befindet, grafisch dargestellt. Diese Darstellungsart stützt sich auch auf die Unified Modelling Language.

Wird das Programm vom User gestartet, so wird auf die Menuauswahl oder auf die Bestätigung durch Drücken des „OK“-Buttons gewartet. Wurde der Knopf gedrückt, so wird auf die Eingabe des Pfades zu jedem Bauteil gewartet. Auch hier schließt der Zustand mit dem Drücken des „OK“-Buttons ab. Daraufhin steht das Programm wartend auf die Auswahl eines Augenpunktes. Ist diese Auswahl erfolgt, steht der Positioner wartend auf die Auswahl einer Schiene und danach eines LCD-Screens.

Ist dies erfolgt, so wird das Layout berechnet, die Teile werden positioniert und das Programm ist beendet.

6.3 Realisierte Skriptbausteine und deren Funktionen

Um die Zusammenhänge der einzelnen Funktionen und Methoden des Programms darzulegen werden sie nach folgendem Schema dargestellt:
In der oberen Zeile findet man den Namen. Die linke Spalte zeigt die Aufgabe und die rechte Spalte zeigt an, welche Methoden oder Funktionen von ihr selbst aufgerufen werden.

OK_1_Click	
• Feststellung, welche Parts positioniert werden sollen • Schreibt die Daten der Parts in das Array „Partfield“	• Tools.ChangeArray • Tools.Path • Infill-Panels • Tools.Measure • Distance.Coords • Distance.Rails

OK_2_Click	
• Feststellung, welche Infill-Panels zur Verfügung stehen • Schreibt die Daten der Infill-Panels in das Array „Inffield“	• Tools.Path

Tools.Distance	
• Errechnung des globalen Abstandes zweier Punkte	

Tools.Path	
• Speicherung des Pfades eines ausgewählten Parts	• FileSelectionBox (VBA-Funktion)

Tools.ChangeArray	
• Vergrößert ein bestehendes Array um die übergebene Größe an der übergebenen Ebene und schreibt die vorhandenen Werte in das neue Array zurück	

Tools.Measure	
• Errechnet mit Hilfe der vorbenannten Flächen und der hinterlegten Informationen in „Partfield“ die Tiefe eines jeden zu positionierenden Bauteils	

Distance.Coords	
• Erzeugung der globalen Positionen der Augenpunkte, um deren Abstände zueinander zu errechnen	• Tools.Distance

Tab. 6-1 Aufgaben und Aufrufe der Skriptbausteine

6.4 Datenstruktur CATIA V5

Die hier erklärte Datenstruktur bezieht sich auf die Datenstruktur der Programmierschnittstelle. Das Wissen über die Zusammenhänge der Strukturelemente ist zwingend notwendig, um die aufgestellte Programmstruktur in ein lauffähiges Programm innerhalb CATIA V5 zu überführen.

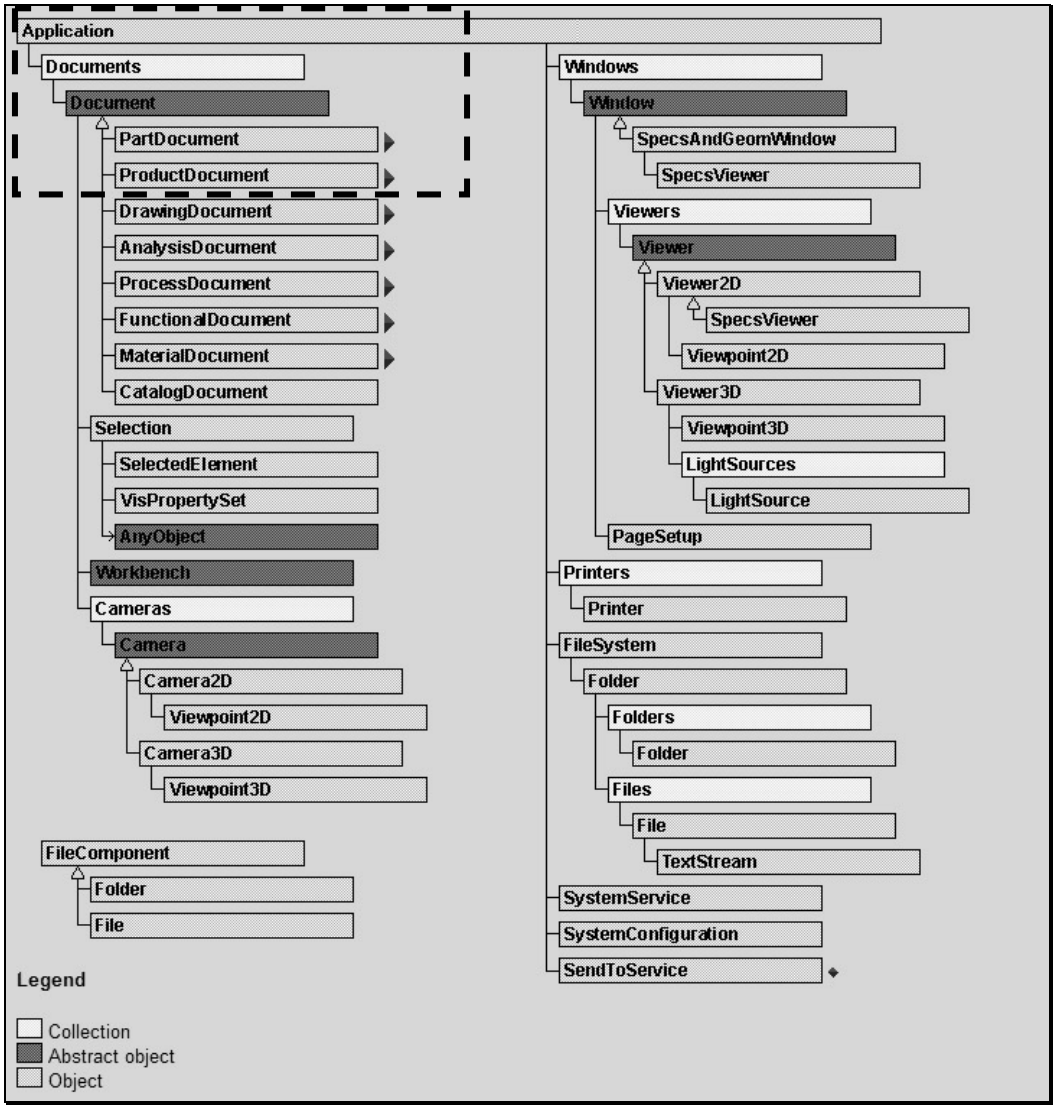


Abb. 6-3 Infrastruktur Automationsobjekte CATIA V5 [10]

In Abb. 6-3 ist die Datenstruktur in CATIA V5 abgebildet. Für dieses Projekt ist in erster Linie der oben eingegrenzte Bereich von Interesse. Es handelt sich hier um die Bereiche „PartDocument“ und „ProductDocument“. Die Struktur besteht aus baumförmigen Zusammenhängen von Collections (Sammlungen), Abstract Objects (abstrakte Objekte) und Objects (Objekte).

Das Root-Objekt (das Wurzelobjekt) ist die CATIA Application selbst, welche die Collection „Documents“ enthält. Hierin befinden sich dann z.B. (falls vorhanden) die Objekte „PartDocument“ und „ProductDocument“.

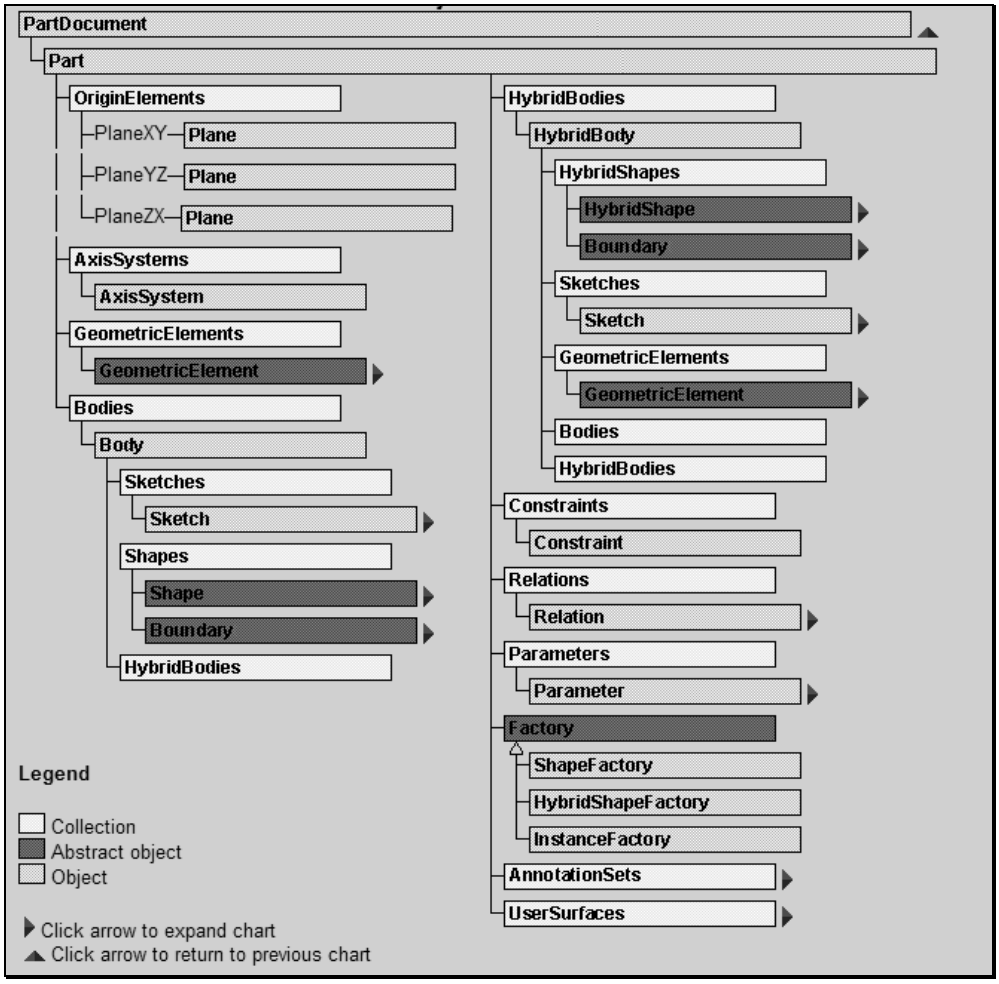


Abb. 6-4 PartDocument Automationsobjekte [10]

In Abb. 6-4 ist die Übersicht über das Objekt „PartDocument“ zu sehen. Ein „Part Document“ ist nichts anderes, als eine CATIA-Datei, in der sich ein Einzelteil befindet. Eine Schraube wird beispielsweise in einem solchen Dokument konstruiert und steht dann weiterhin in einem solchen Dokument zur Verfügung. In jedem „PartDocument“ befindet sich auf oberster Ebene das Objekt „Part“, welches das eigentliche Einzelteil repräsentiert (z.B. die Schraube).
Im Objekt „Part“ befinden sich die verschiedenen Collections. Die wichtigsten sind:

- Origin Elements

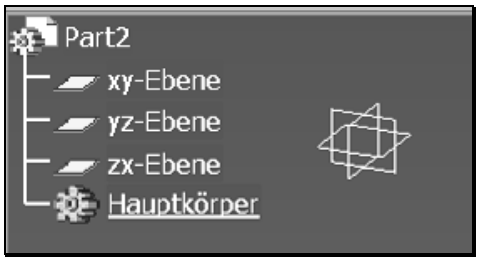


Abb. 6-5 Origin Elements

  - Hier enthalten sind die drei Objekte der Grundebenen (xy-,yz-,zx-Ebene), welche das Hauptachsensystem des Parts bilden, über das die später erklärte Koordinatentransformation berechnet wird.

- AxisSystems

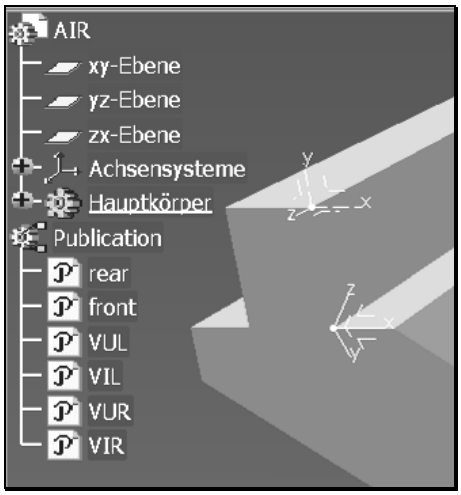


Abb. 6-6 Axis Systems

  - Es ist möglich eigens erstellte Achsensysteme in ein Part einzufügen. Diese sind unter AxisSystems abgelegt.

- Bodies

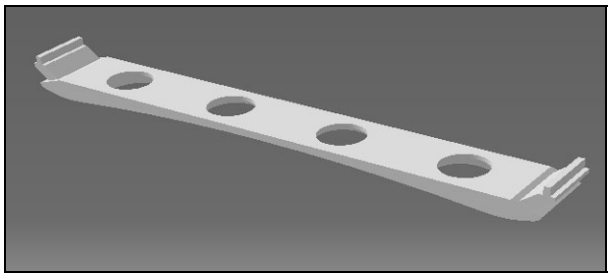


Abb. 6-7 Body (Volumenkörper)

- Sammlung aller kreierten Volumenkörper und HybridBodies im Part, angefangen mit dem „MainBody“, dem Hauptkörper des Parts. Eine Bohrung in einem Quader besteht beispielsweise aus zwei Bodies. Einmal der Quader, und die Bohrung (wird realisiert indem ein Zylinder volumenmäßig vom Quader abgezogen wird). Der Zylinder ist der zweite Volumenkörper.

- Sketches

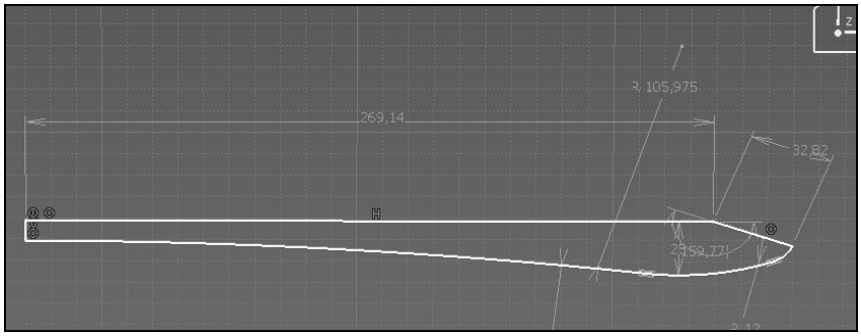


Abb. 6-8 Sketch (Skizze)

  - Sammlung der Entwürfe, welche noch nicht eindeutig einem Body zugeteilt sind.

- HybridBodies
  - Auch geöffneter Körper genannt. Enthält Drahtgeometrien und Flächen.

Des Weiteren sind noch die Daten der Beziehungen, Bedingungen, Parameter usw. hinterlegt.

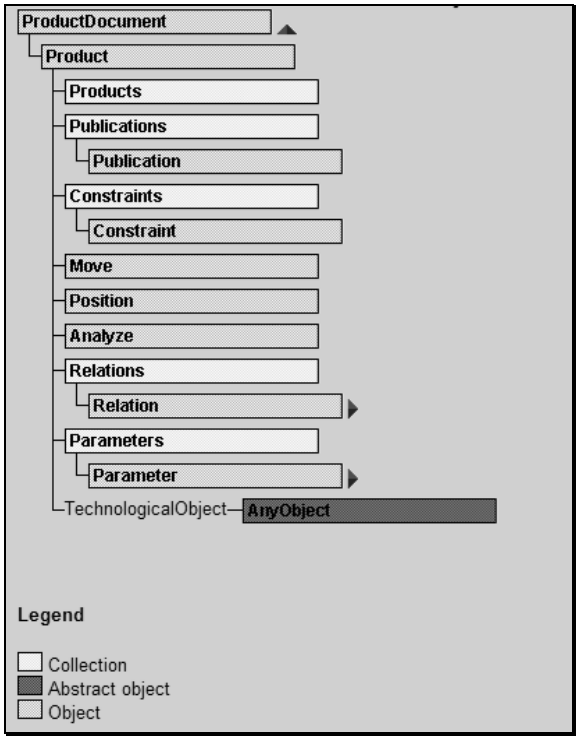


Abb. 6-9 ProductDocument Automationsobjekte[10]

In Abb. 6-9 ist das Objekt „ProductDocument“ dargestellt. Hier ist das oberste Objekt das Product (eine Zusammenbaudatei, z.B. eine Schraube mit einer aufgeschraubten Mutter). Es kann weitere Products enthalten, beherbergt

- Veröffentlichungen
  - Es können Geometrien zusammengefasst werden und unter einer neuen Benennung veröffentlicht und somit ansprechbar gemacht werden
- Die Position des Products in Bezug auf das RootProduct der höchsten Ebene

und noch weitere Parameter, die in dieser Arbeit nicht weiter von Bedeutung sind.

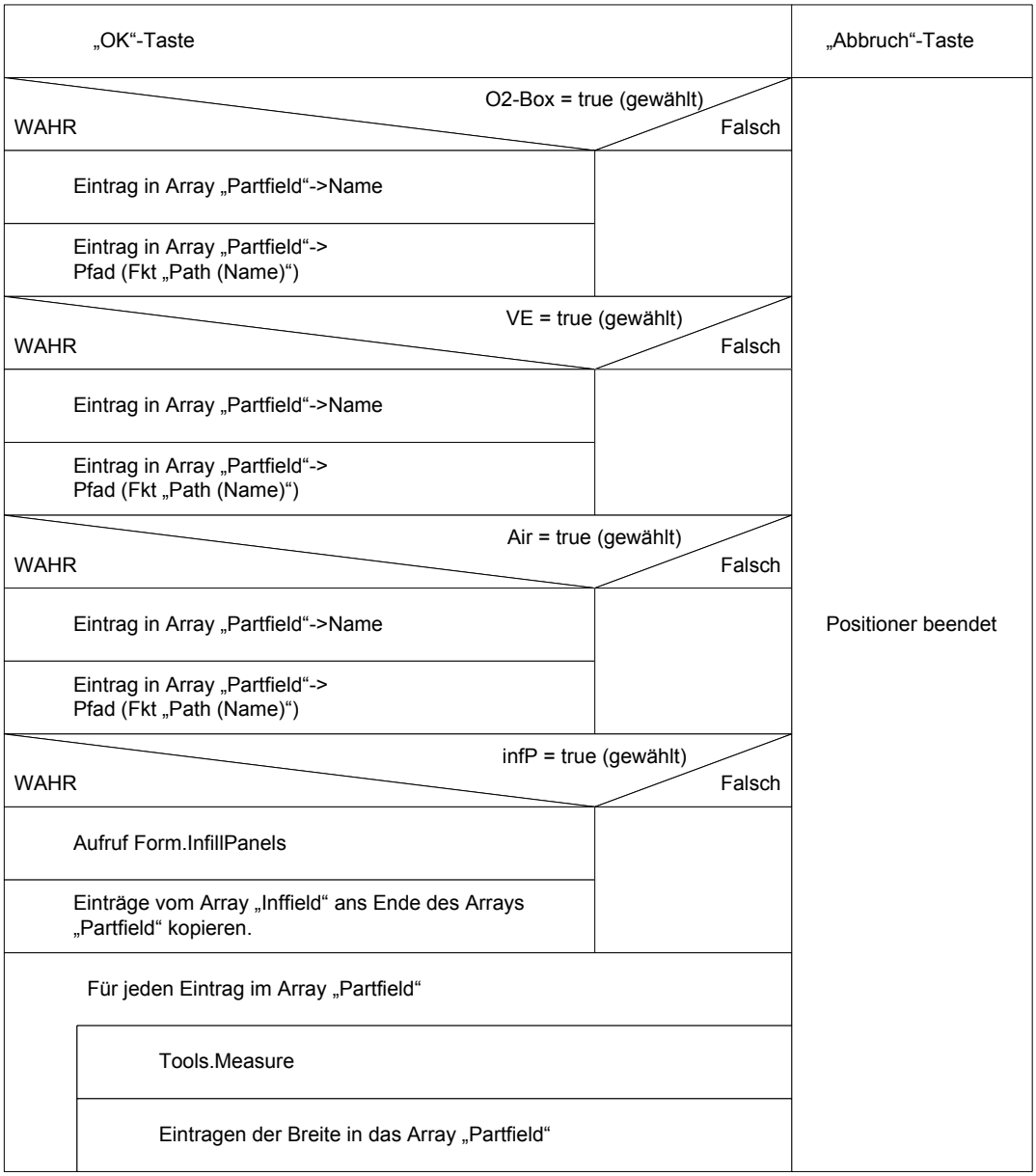
6.5 Programmstruktur

Dieses Kapitel enthält die strukturelle Lösung der Aufgabe. Anhand von Nassi-Schneidermann-Diagrammen werden die zu realisierende Programmstruktur und die notwendigen Datenfelder erläutert.
Zusätzlich ist vor jeder Diagrammdarstellung eine kurze Zusammenfassung des Funktionsumfanges zu finden.

6.5.1 Eingabemasken

- **Selection**
  - Das Formular “Selection” (Abb. 6-10) ist für den User die Eingabemaske, in der ausgewählt wird, welche Bauteile im Versorgungskanal durch das Programm positioniert werden sollen. Die gewünschten Bauteile werden per Mausklick in die zugehörigen Checkboxen ausgewählt. Durch Auswahl des „Abbrechen“-Knopfes wird das gesamte Programm beendet. Im folgenden Diagramm wird die genaue Funktionsweise dargestellt.

Abb. 6-10 Formular "Selection"



Tab. 6-2 Ablauf „Selection“

- **Infill-Panels**
  - Ist bei der Auswahl der zu positionierenden Teile auch die Option „Infill-Panels“ ausgewählt worden, wird im Anschluss das Formular „Infill-Panels“ aufgerufen(Abb. 6-11). Hier wird kommagetrennt eingegeben, welche Größen an Panels zur Verfügung stehen.

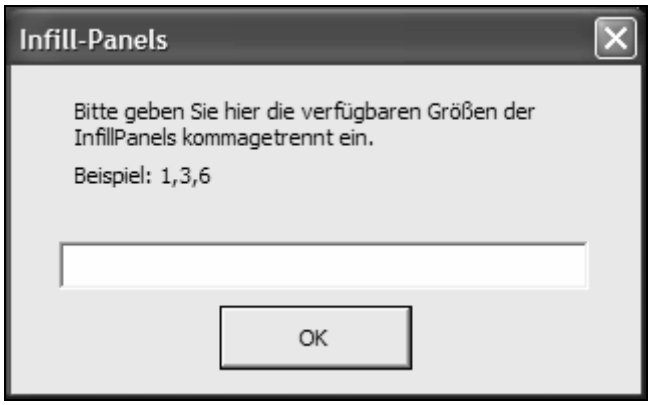


Abb. 6-11 Formular "Infill-Panels"

Warten auf „OK“	
WAHR	Es wurde eine Eingabe gemacht: Falsch
Für jeden Eintrag einen Eintrag in das Array „Inffield“	Array „Inffield“ = null
Eintragen des jeweiligen Pfades in das Array „Inffield“ ->Fkt. "Path"	

Tab. 6-3 Ablauf „Infill-Panels“

6.5.2 Methoden und Funktionen

- **CatMain**
  - Die Hauptmethode ruft nacheinander die abzuarbeitenden Funktionen bzw. Methoden auf. Programme werden generell nach dieser Art gestaltet, um eine bessere Übersichtlichkeit zu erhalten. Die Inhalte der einzelnen Module werden unten näher erläutert

Form.Selection
Distance.Coords
Distance.Rails
Eyepoints
Spaces
FindScreens
Partpositioning
Infiller
LoadParts

Tab. 6-4 Ablauf „CatMain“

- **Path**
  - Die Funktion „Path“ öffnet die FileSelectionBox, ein Auswahlfenster in dem Dateien vom Benutzter ausgewählt werden. Der Rückgabewert der Funktion ist ein String, der die absolute Pfadangabe der gewählten Datei enthält.

Öffne die FileSeletionBox
Titel = Name
path = Ausgewähltes Teil

Tab. 6-5 Ablauf „Path“

- **Tools.Measure**
  - Diese Funktion sucht in den einzelnen Bauteilen die front- und rear-Ebene und misst deren Abstand zueinander. Somit wird die Tiefe des jeweiligen Bauteils ausgemessen und als Rückgabewert übergeben.

Suche alle Ebenen mit dem Namen „front“
Suche alle Ebenen mit dem Namen „rear“
Beide in eine Selection schreiben
Abstand messen
measure = Abstand

Tab. 6-6 Ablauf „Tools.Measure“

- **Distance.Coords**
  - Diese Methode errechnet die Positionen der Augenpunkte im Produkt. Der Benutzer wählt einen beliebigen Augenpunkt aus. Anhand der Daten des Punktes werden alle im Produkt befindlichen Gleichnamigen gesucht und mit dem Namen und Koordinaten gespeichert. Hierzu wird die später erläuterte Koordinatentransformation zur Hilfe genommen. Zusätzlich wird der Abstand zum ersten Augenpunkt gespeichert.

MessageBox: „Bitte wählen Sie einen Augenpunkt“
Warten bis „OK“
Warten auf Auswahl eines Augenpunktes
Speichern des Namens
Suche nach allen Punkten mit dem Namen im Produkt
Menge = Anzahl der gefundenen Punkte
Array „vDistance“ und „vSeatPosition“ mit "ChangeArray" nach „Menge“
Für jeden gefundenen Punkt
Auslesen der Koordinaten im Part
Setze Koordinaten um auf höher liegendes Product
Solange noch ein übergeordnetes Product vorhanden ist
Name in das Array „vDistance“ eintragen
Eintragen des Abstandes in „vDistance [1]“-> Fkt „Tools.Distance“
Sortieren des Arrays nach der Reihenfolge

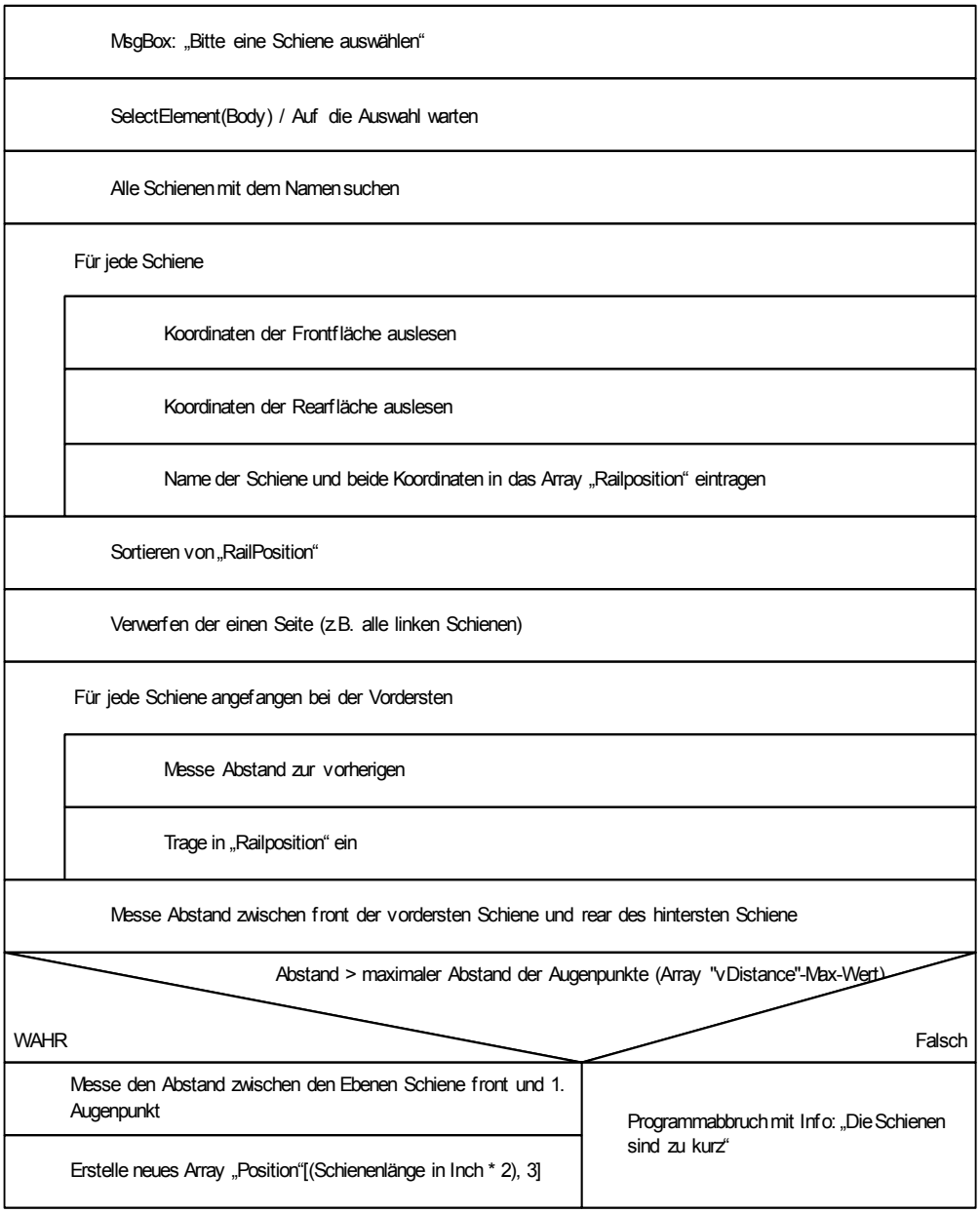
Tab. 6-7 Ablauf „Distance.Coords“

- **Tools.Distance**
  - „Tools.Distance“ errechnet über den Satz des Pythagoras den kürzesten Abstand zweier Punkte im Raum.

X-Wert-Pkt minus x-Wert 1. Pkt
Y-Wert-Pkt minus y-Wert 1. Pkt
Z-Wert-Pkt minus z-Wert 1. Pkt
Tools.Distance = sqrt[(x+y+z)²]

Tab. 6-8 Ablauf „Tools.Distance“

- **Distance.Rails**
  - In dieser Methode wird festgestellt, wie viele Einbauschienen es gibt, wie lang diese sind und welche Abstände sie zueinander besitzen. Auch hier muss der Benutzer erst eine beliebige Schiene per Hand auswählen. Anhand der Daten werden dann alle Schienen gesucht. Die Längen und der Abstand zur jeweils nächsten werden gemessen, womit die Lückengröße zwischen den einzelnen Schienen bekannt ist. Am Ende wird geprüft, ob alle Sitzreihen ausreichend mit Schienen versorgt sind.



Tab. 6-9 Ablauf „Distance.Rails“

- **EyePoints**
  - An den Stellen, an denen ein Augenpunkt vorhanden ist, wird dies im Array „Position“ vermerkt.

Für jeden Augenpunkt in „vDistance“
Position [Abstand*2,AP] = „EP“

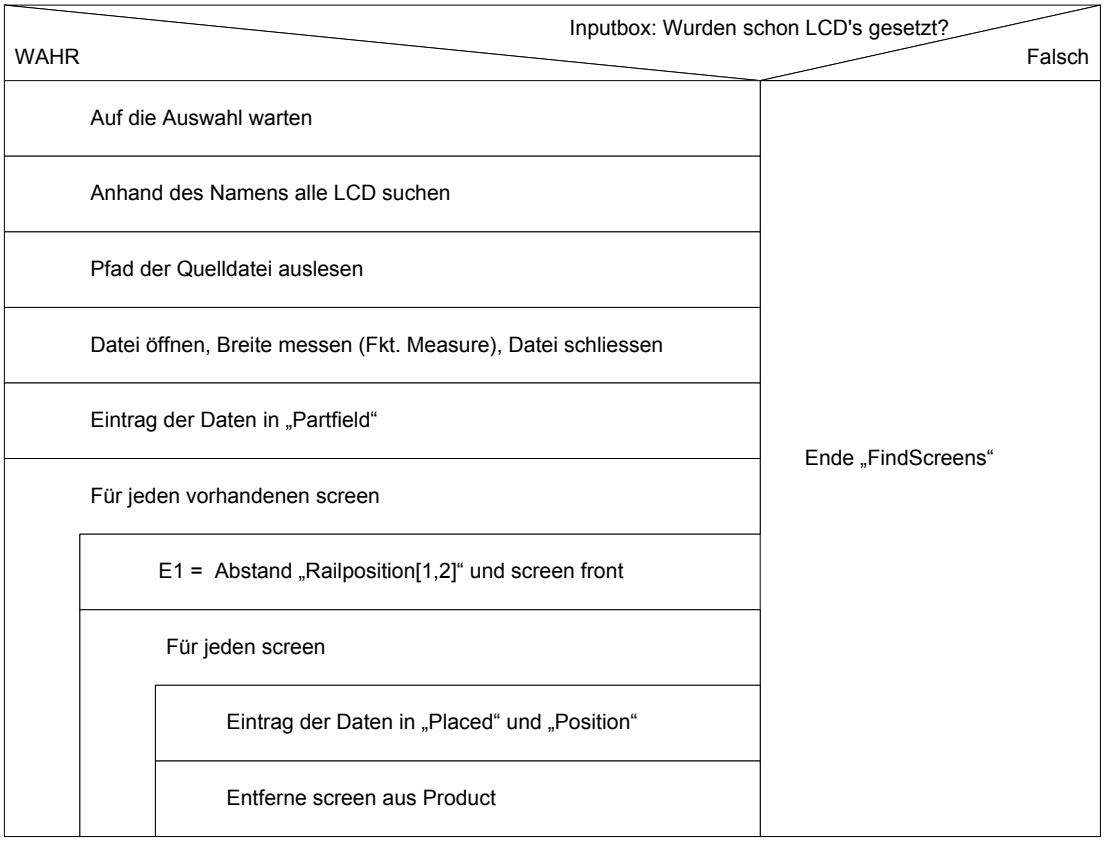
Tab. 6-10 Ablauf „EyePoints“

- **Spaces**
  - Die Lücken zwischen den Schienen werden errechnet und in das „Position“-Array eingetragen.

Für jede Schiene
Abstand E1 = Abstand zu [1] + rear - front
Abstand E2 = front [i+1] - rear[i]
For i = 0 to E2
Position [(E1+i), Lücke] = „x“

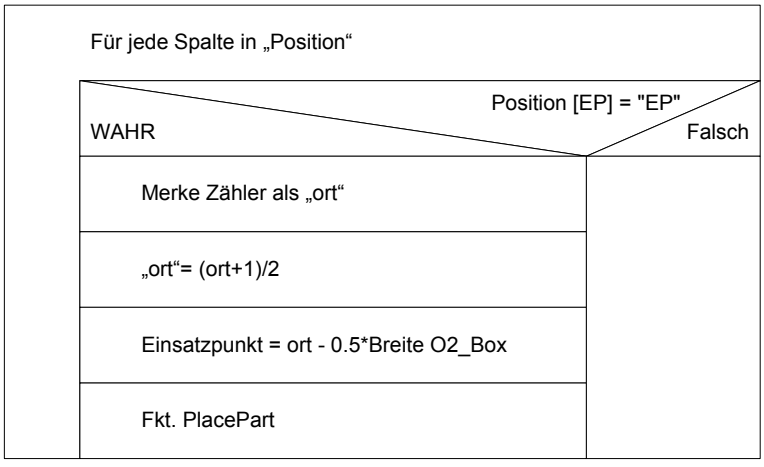
Tab. 6-11 Ablauf „Spaces“

- **FindScreens**
  - Diese Methode ist notwendig, um evtl. vom Kunden vorpositionierte LCD-Screens in das „Position“-Array aufzunehmen. Wurden schon Screens positioniert, so muss der Benutzer einen Screen auswählen. Der Pfad der Bauteildatei wird ausgelesen und gesichert. Die breite des Screens wird ausgelesen. Die Screens werden mit Ihrer Position in die Array „Position“ und „Placed“ eingeschrieben und aus dem Produkt entfernt. Somit werden sie ab hier wie alle anderen zu positionierenden Teile behandelt.



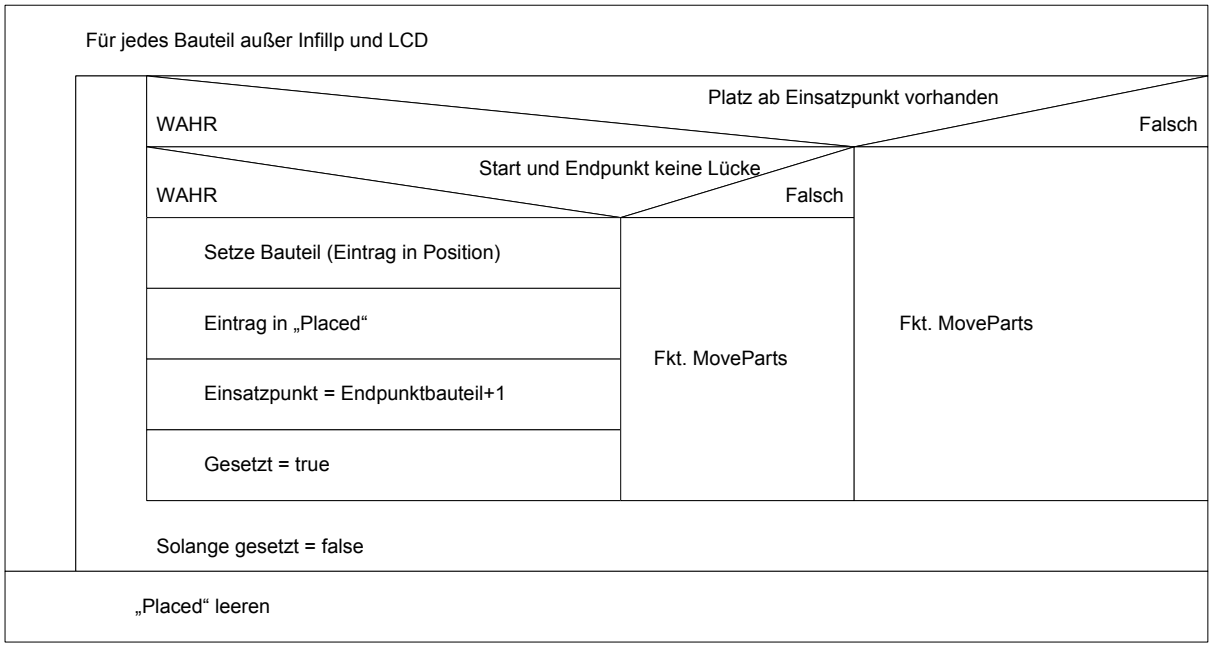
Tab. 6-12 Ablauf „FindScreens“

- **PartPositioning**
  - Das „Position“-Array wird von vorne nach Augenpunkten durchsucht, und bei jedem gefundenen Augenpunkt wird die Positionierung der Bauteile gestartet.



Tab. 6-13 Ablauf „PartPositioning“

- **PlaceParts**
  - Diese Methode berechnet die Einbaustellen der Bauteile zu jedem Augenpunkt. Die Infill-Panels werden später. Sollte ein Einbaukonflikt entstehen (nötiger Platz ist schon belegt), so wird die Fkt. „MoveParts“ aufgerufen. Wurden alle Bauteile zu einem Augenpunkt komplett positioniert, so wird das Array „Placed“ geleert.



Tab. 6-14 Ablauf „PlaceParts“

- **MoveParts**
  - Müssen Bauteile verschoben werden, um eine Positionierung nach den Vorgaben zu realisieren, so wird diese Methode aufgerufen. Ist eine Verschiebung möglich, wird diese durchgeführt. Falls nicht, so wird das zuletzt gesetzte Bauteil aus „Placed“ ausgelesen, entfernt und verschoben wieder neu gesetzt. Wenn keine andere Möglichkeit mehr besteht, so wird das Programm mit einer Information an den Benutzer beendet.

Verschiebung ist möglich	
WAHR	Falsch
Innerhalb der Toleranz verschieben	Lese den letzten Eintrag in „Placed“
	Lösche in „Position“[Part] die Felder aus „Placed“
	Lese den Bauteilnamen aus „Placed“
	Lösche den Eintrag aus „Placed“
	Einsatzpunkt = Einsatzpunkt - Breite Bauteil +1
	Position in "Placed"= 0
	WAHR: Programmabbruch mit Info: „ Automatisches Positionieren innerhalb der Normen nicht möglich“ / Falsch: Fkt. PlaceParts

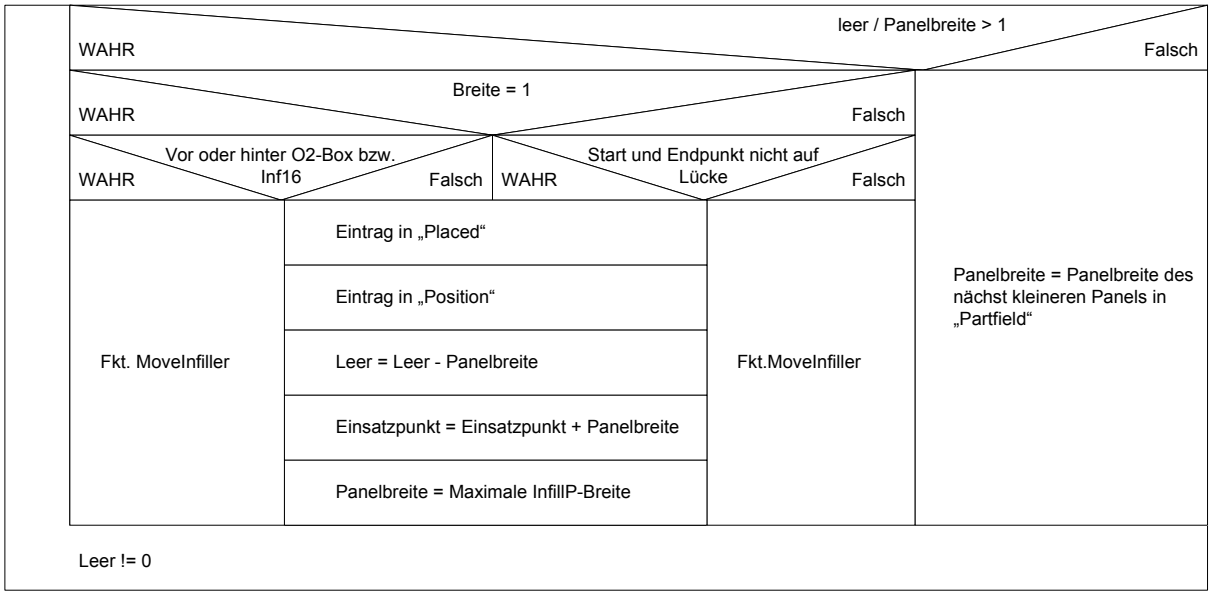
Tab. 6-15 Ablauf „MoveParts“

- **Infiller**
  - Die übrig gebliebenen freien Stellen im Kanal werden nach der Positionierung der Bauteile ausgelesen. Für jede zusammenhängende Leerfläche wird der Wert an die Funktion „PlaceInfiller“ übergeben, in der die Platzierung stattfindet.

Für jede Spalte in „Position“			
Position[Part] = ""			
WAHR		Falsch	
leer = 0		leer > 0	
WAHR	Falsch	WAHR	Falsch
Merke Index als Einsatzpunkt		Panelbreite = Maximale InfillP-Breite	
Leer = Leer + 1		Fkt. PlaceInfiller	

Tab. 6-16 Ablauf „Infiller“

- **PlaceInfiller**
  - Die Lücke wird mit den verfügbaren Infill-Panelgrößen verglichen, um immer möglichst große Infill-Panels zu nutzen. Außerdem wird überwacht, dass die Infill-Panels der Größe 1 Inch nicht an 16 Inch Infill-Panels und nicht an O_2-Boxen grenzen. Zusätzlich darf sich der Start- und Endpunkt nicht in einer Schienenlücke befinden. Ansonsten müssen die Infill-Panels verschoben werden.



Tab. 6-17 Ablauf „PlaceInfiller“

- **MoveInfiller**
  - Das letzte positionierte Infill-Panel wird entfernt und der Einsatzpunkt wird verschoben. Sollte eine Positionierung nicht möglich sein, erhält der Benutzer eine Information hierüber und die gesetzten Bauteile werden anschließend in das Produkt eingeladen.

Lese „Placed“		
Index = Index - 1		
WAHR	Index = 0	Falsch
Info: „ Automatisches Positionieren der Infillpanels innerhalb der Normen nicht möglich“	Lösche in „Position“[Part] die Felder aus „Placed“	
	WAHR: Breite = 1	Falsch
	Breite bleibt = 1	„Breite“ wird die Breite des zuletzt positioniert Teils
Fkt. LoadParts	Lösche den Eintrag aus „Placed“	
	Einsatzpunkt = Einsatzpunkt - alte Breite	

Tab. 6-18 Ablauf „MoveInfiller“

- **LoadParts**
  - Die Methode „LoadParts“ liest die Einträge und somit die Positionen der Bauteile aus dem „Postion“-Array aus und lädt die entsprechenden Bauteile in das Produkt ein. Die Platzierung geschieht über Referenzvergabe zwischen den veröffentlichten Flächen.

Referenz[0] = Schiene 1 front		
Für jede Spalte in Position		
WAHR	Part != ""	Falsch
Lese welches Part		Abstand = Abstand + 0.5 Inch
Lade Part aus „Partfield [Path]“		
Setze front an Referenz [0] mit Offset = Abstand		
Plaziere in Schienen (über die veröffentlichten Flächen)		
Spalte = Spalte + Breite Bauteil + Abstand		
Abstand = 0		

Tab. 6-19 Ablauf „LoadParts“

6.5.3 Datenfelder (Arrays)

- **Partfield**
  - Enthält alle Teilearten, die positioniert werden müssen (also z.B. die O_2-Box mit dem Namen, dem Pfad in dem das Teil liegt und der Breite des Teils).

0	Name
1	Path
2	Width

- **Inffield**
  - Dies ist ein Zwischenspeicher für die zu positionierenden Infill-Panels. Diese werden später in „Partfield“ umkopiert.

0	Name
1	Path

- **vDistance**
  - Hier wird der Abstand einer jeden Sitzreihe zur ersten Sitzreihe mit ihrem Namen gespeichert.

0	Name
1	Distance to first

- **RailPosition**
  - Dieses Array beherbergt die Namen der einzelnen Aufnahmeschienen für die Bauteile, deren Koordinaten der Front- und der Rückseite und deren Abstand zwischen der Front-Seite und der Front-Seite der vordersten Schiene.

0	Name
1	Coord „front“
2	Coord „rear“
3	Distance to first

- **Position**
  - Das Array „Position“ ist das zentrale Array und besitzt für jede 0,5 Inch der Gesamtlänge der Aufnahmeschienen eine Spalte. Leere Spalten bedeuten, dass an dieser Stelle noch kein Teil positioniert ist. Ein ‚EP’ in EyePoint bedeutet, dass hier eine Sitzreihe einen Augenpunkt liegen hat. Ein ‚X’ in Space meint, dass die Schienen hier eine Lücke haben und keine Aufnahme von Bauteilbefestigungen möglich ist. An der Stelle Part wird das hier positionierte Part eingetragen, so dass am Ende ein Array entsteht, welches mit der Seitenansicht der Kabine im Endzustand übereinstimmt. Hieraus werden die Bauteile real in CATIA V5 platziert.

0	EyePoint
1	Space
2	Part

- **Placed**
  - Placed ist ein Zwischenspeicher für die platzierten Bauteile. In PartName wird der Bauteilename hinterlegt. In „TakenPlaces“ stehen die Felder, die das Teil in „Position“ belegt. Müssen nachträglich Teile in „Position“ verschoben werden, so können die letzten Schritte mit Hilfe dieses Arrays nachvollzogen und rückgängig gemacht werden.

0	PartName
1	TakenPlaces

6.6 Umsetzung

6.6.1 Entwurf der Modelle

Zuerst musste eine Umgebung (Kabinenlayout) geschaffen werden. Diese sollte, da es sich um eine beispielhafte Untersuchung handelt, der Umgebung eines typischen Auftrages ähneln. Es entstand eine vereinfachte Kabine mit 17 Sitzreihen. Vereinfacht, da die Sitzreihen alle die gleiche Entfernung zur vorderen Sitzreihe haben und auch nur der mittlere Versorgungskanal betrachtet wird. Trotzdem können aufkommende Schwierigkeiten an dem geschaffenen Modell direkt erkannt werden, und die Umsetzung auf weitere, umfangreichere Kanäle ist ohne weiteres möglich. Aus rechtlichen Gründen konnten keine „echten“ Modelle genutzt werden, da Zulieferfirmen, die schon mit CATIA V5 arbeiten, ihre Modelle nicht nach Außen geben. Außerdem wird überwiegend noch nicht in CATIA V5 entworfen, und somit stehen solche Modelle auch nicht zur Verfügung.
Mit den groben Außenmaßen der originalen 2D-Bauteile wurden eigene, beispielhafte Bauteile entworfen.
Die Sitzreihen bestehen aus je 4 Sitzen. Jeder Sitz hat einen Augenpunkt und einen Lichtpunkt. Der Lichtpunkt wird der Vollständigkeit halber aufgeführt, hat aber im Weiteren für die gemachte Aufgabenstellung keine Bedeutung. Die Sitze sind auf einer Bodenplatte platziert, und über den Sitzen befinden sich die Aufnahmeschienen der Bauteile des Versorgungskanals. Hier mussten für die Untersuchung, wie im realen Modell, Lücken eingebaut werden.

Im Folgenden werden die entworfenen Baugruppen und Bauteile mit kurzen Beschreibungen gezeigt.

Dies ist das entworfene Kabinenteil mit den Sitzreihen und den Schienen, in denen die Bauteile des Versorgungskanals eingebaut werden.

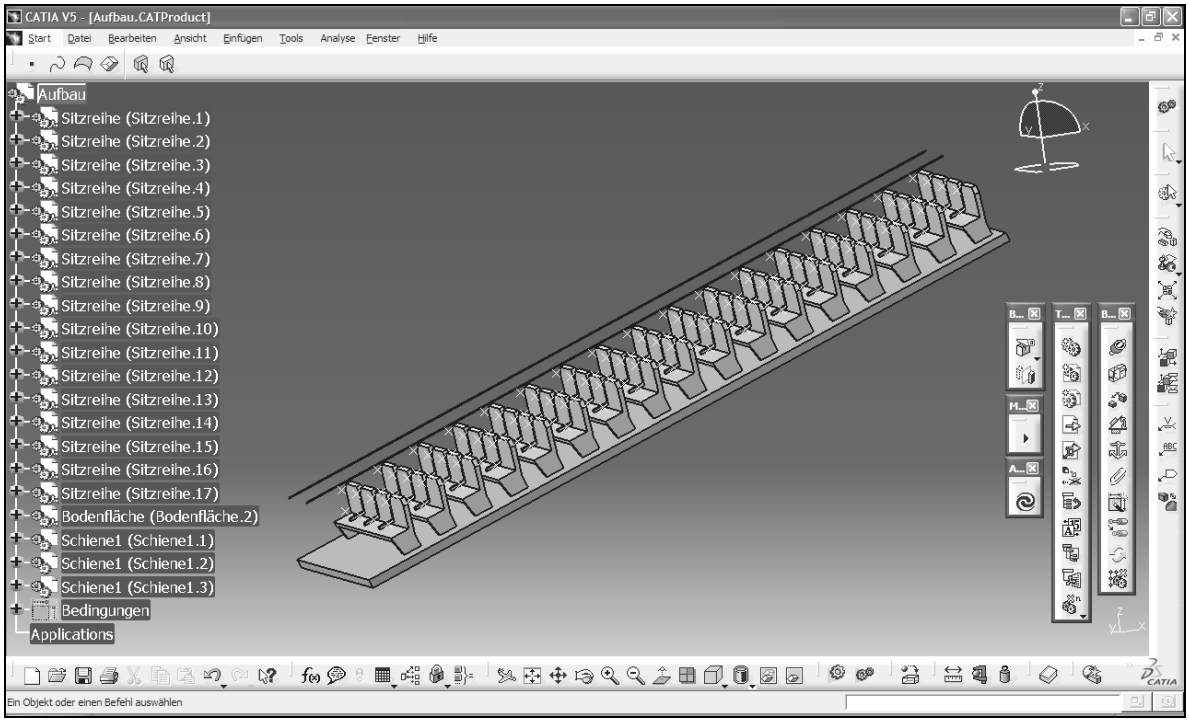


Abb. 6-12 Übersicht Kabinenlayout CATIA V5

Die Platzierung kann rechnerisch 2D erfolgen. Man betrachtet das Modell einfach aus der Seitenansicht.

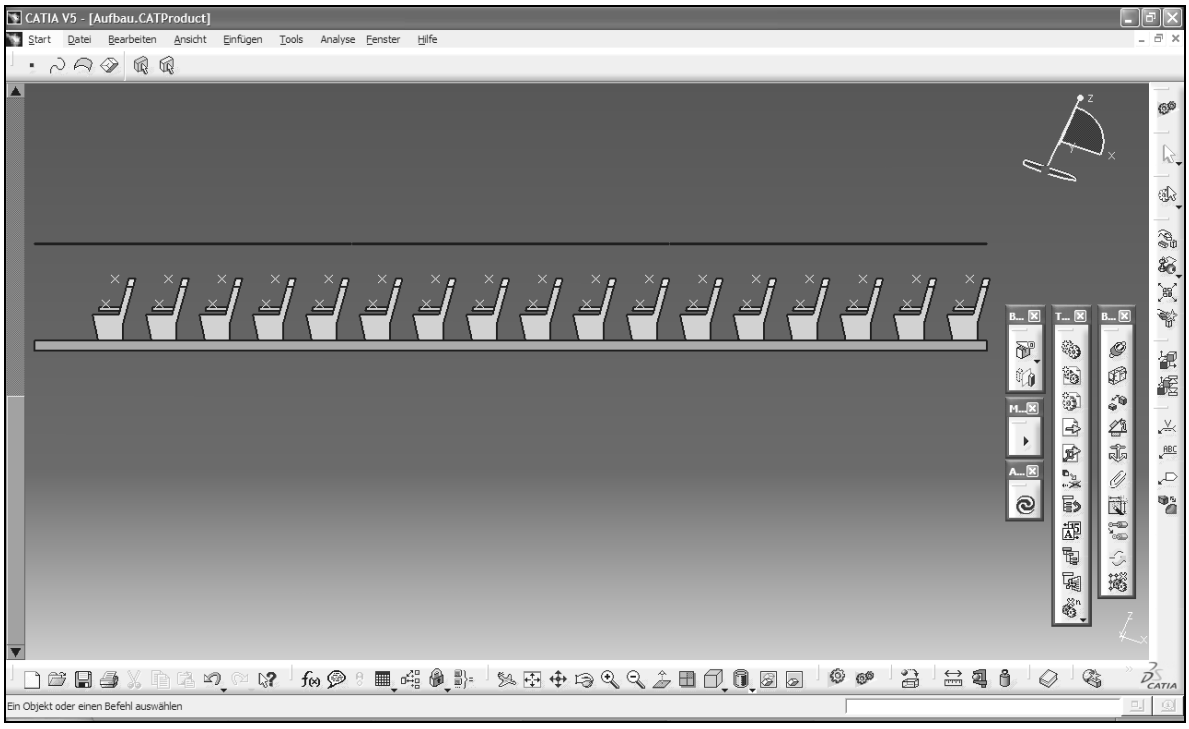


Abb. 6-13 Seitenansicht Kabinenlayout

Die O_2-Box wird mit offener Auswurfklappe dargestellt, um sofort Teile zu erkennen, die beim Öffnen im Weg sein könnten.

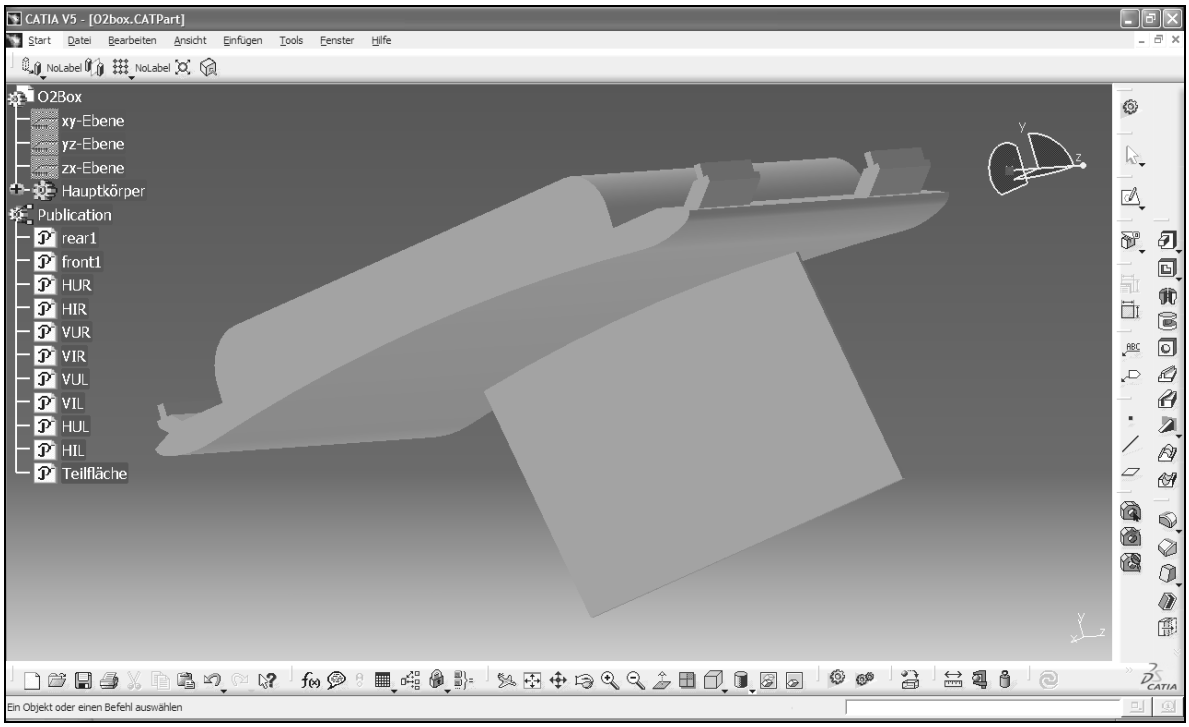


Abb. 6-14 O_2-Box CATIA V5

Bei der Individualbelüftung wurden die Lüftungsauslässe lediglich angedeutet, da alle weiteren Anbauteile für die Positionierung nicht von Bedeutung sind.

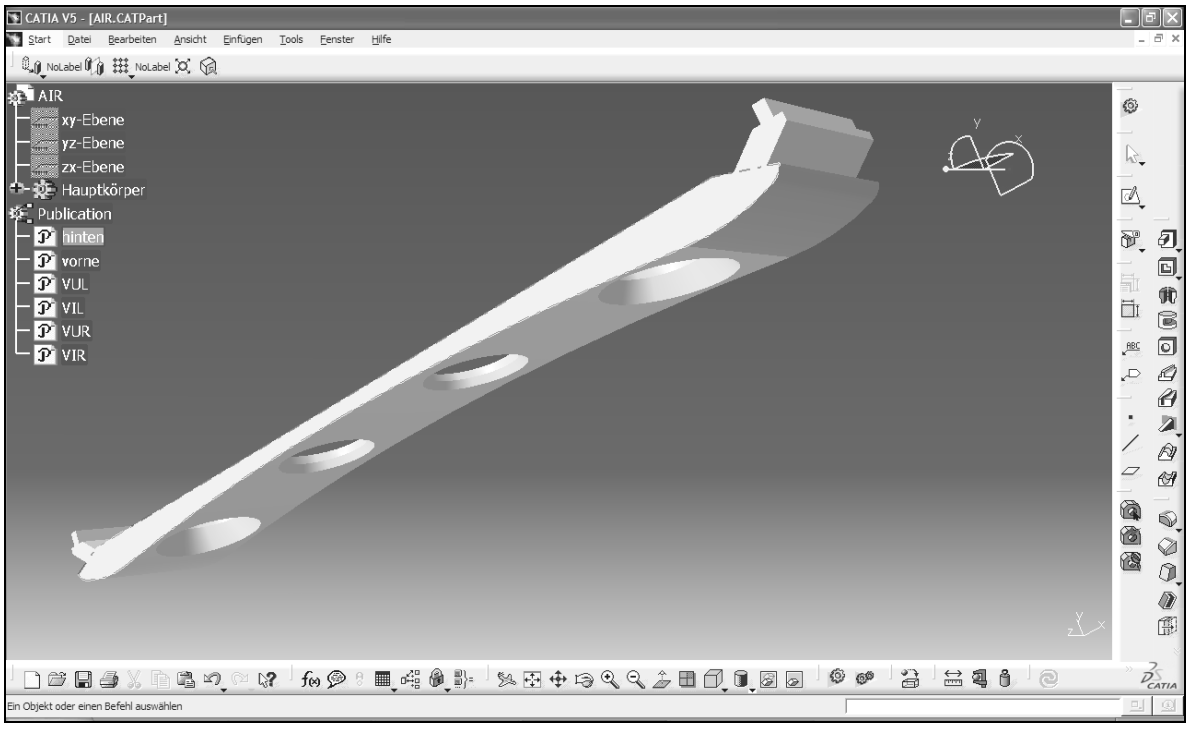


Abb. 6-15 Individuelle Belüftung CATIA V5

Die Versorgungseinheit ist als Bauteil inklusive der Elektronikbox gebaut. Angedeutet sind die Lichter und die Informationsleiste.

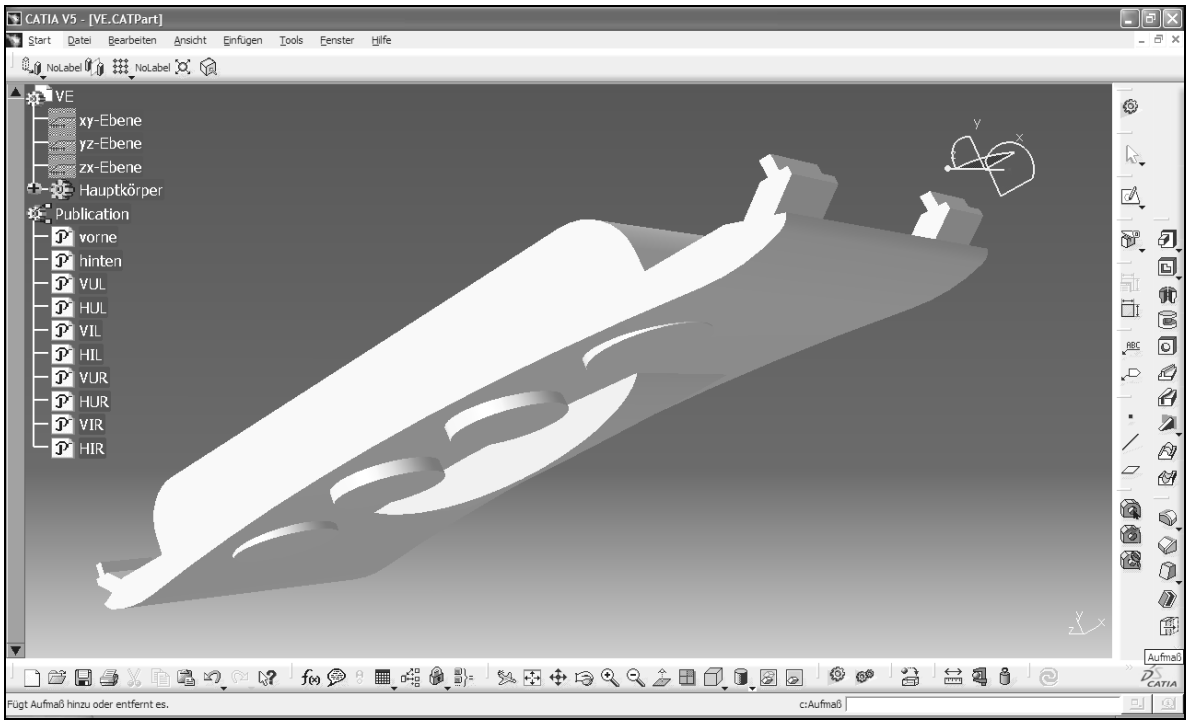


Abb. 6-16 Versorgungseinheit CATIA V5

Beispielhaft wird hier ein 16 Inch Infill-Panel gezeigt. Alle anderen Größen basieren auf dem gleichen Modell, nur die Tiefe ist angepasst.

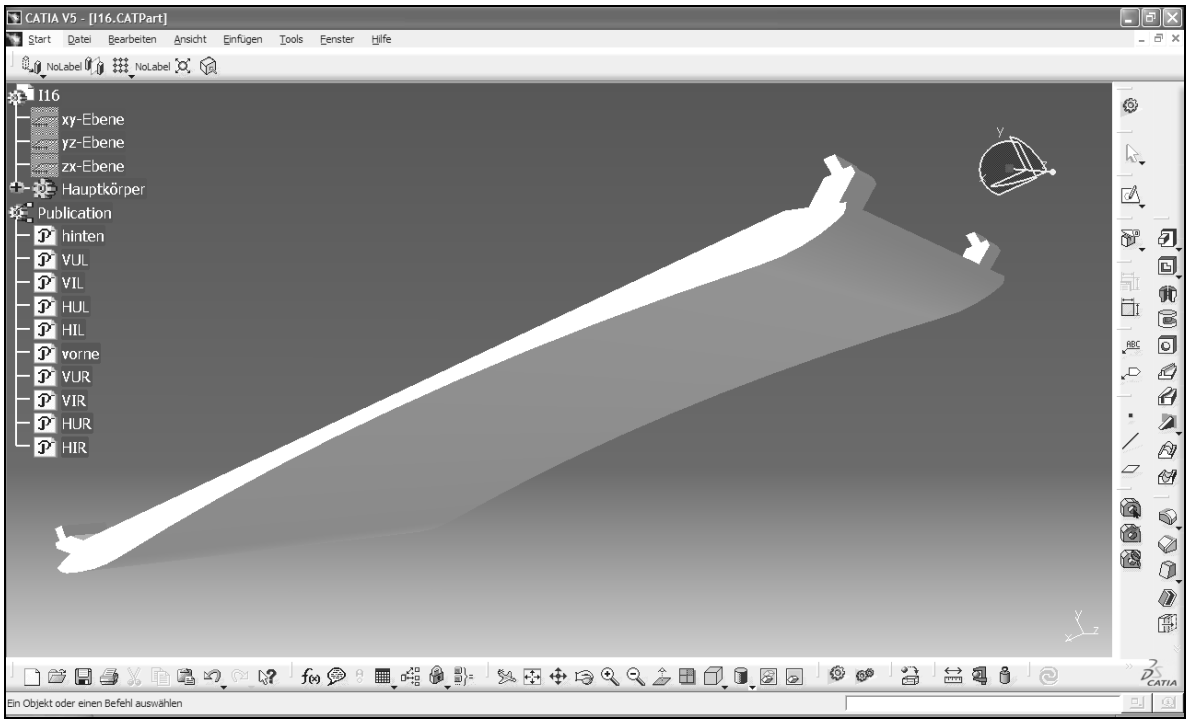


Abb. 6-17 Infill-Panel 16 Inch CATIA V5

6.6.2 Anpassen der Modelle

Bei jedem neu angelieferten Bauteil muss einmalig eine Anpassung durchgeführt werden. Hierbei handelt es sich um Eingaben, die notwendig sind, damit der Positioner mit dem Bauteil umgehen kann.

Damit die Tiefe der Bauteile gemessen werden kann, müssen die beiden Endflächen des Bauteils mit Ebenen versehen werden, zwischen denen der Abstand gemessen werden kann. Diese Tiefe ergibt dann bei der Platzierung die Menge der belegten Plätze im „Position“-Array.

Vorgehensweise:

- Bauteil öffnen
- Vordere Grenzfläche markieren

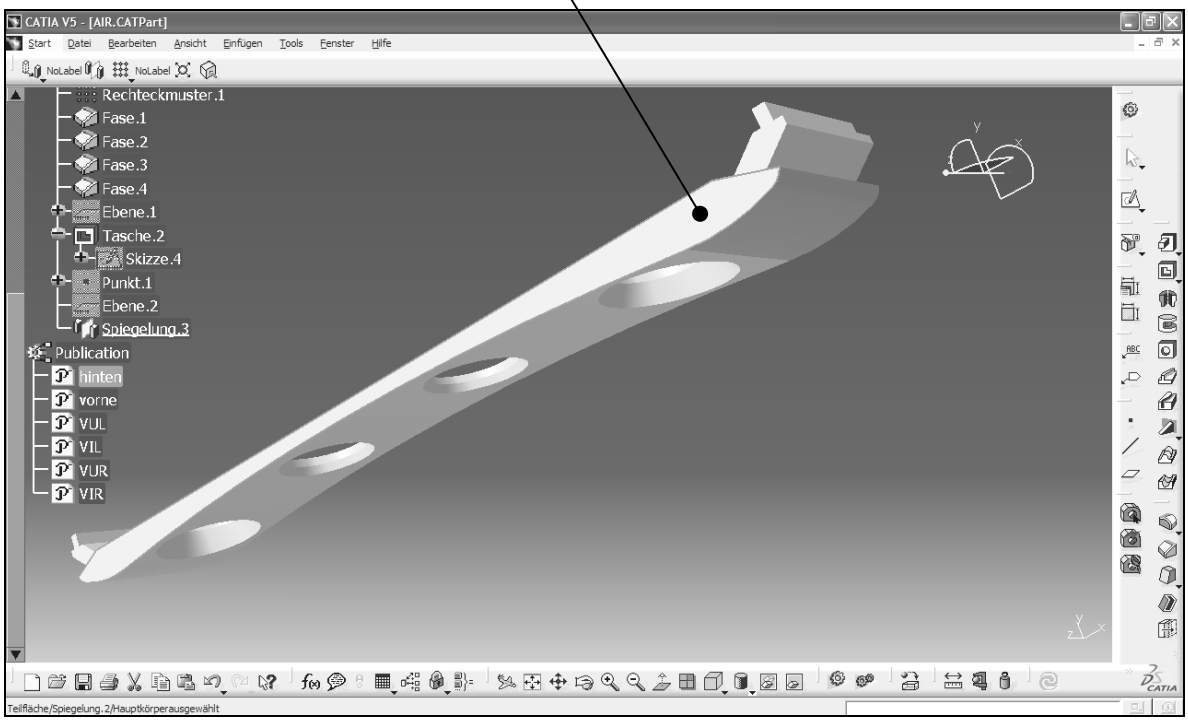


Abb. 6-18 Auswahl der vorderen Ansatzfläche

- „Ebene“ auswählen
  - Ebenentyp: „Offset von Ebene“
  - Referenztyp ist schon durch die Markierung der Fläche eingetragen
  - Offset: 0mm
  - „OK“

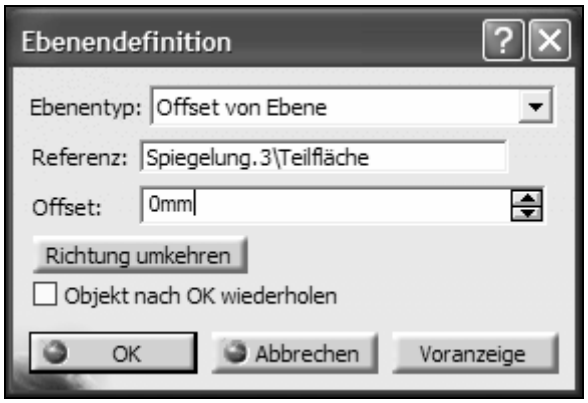


Abb. 6-19 Einstellungen der Ebene

- Die entstandene Ebene im Strukturbaum auswählen und über die rechte Maustaste die Eigenschaften aufrufen
- In den „Komponenteneigenschaften“ den Komponentennamen auf „front“ ändern

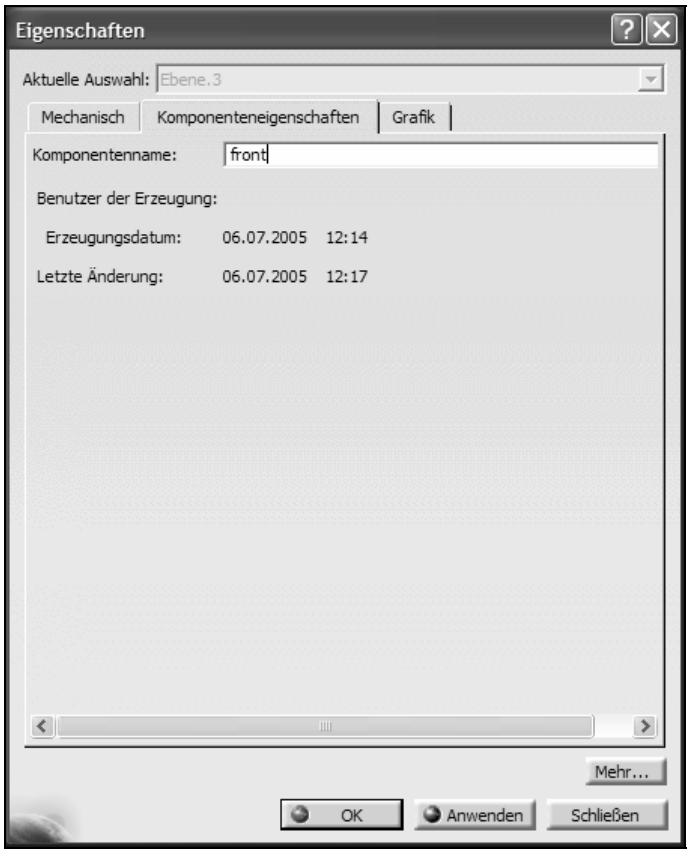


Abb. 6-20 Umbenennung der Namenseigenschaft

- Das gleiche mit der hinteren Fläche wiederholen und hier die Benennung „rear“ wählen.

Zusätzlich müssen für die Platzierung der Bauteile in den Schienen noch bestimmte Flächen veröffentlicht werden, damit beim Einladen der Teile in das Kabinenlayout die Bezüge zwischen den Schienen und den Bauteilen hergestellt werden können.

Am Beispiel des linken vorderen Verbindungsstücks wird die Vorgehensweise erklärt:

- Die gezeigte Fläche auswählen

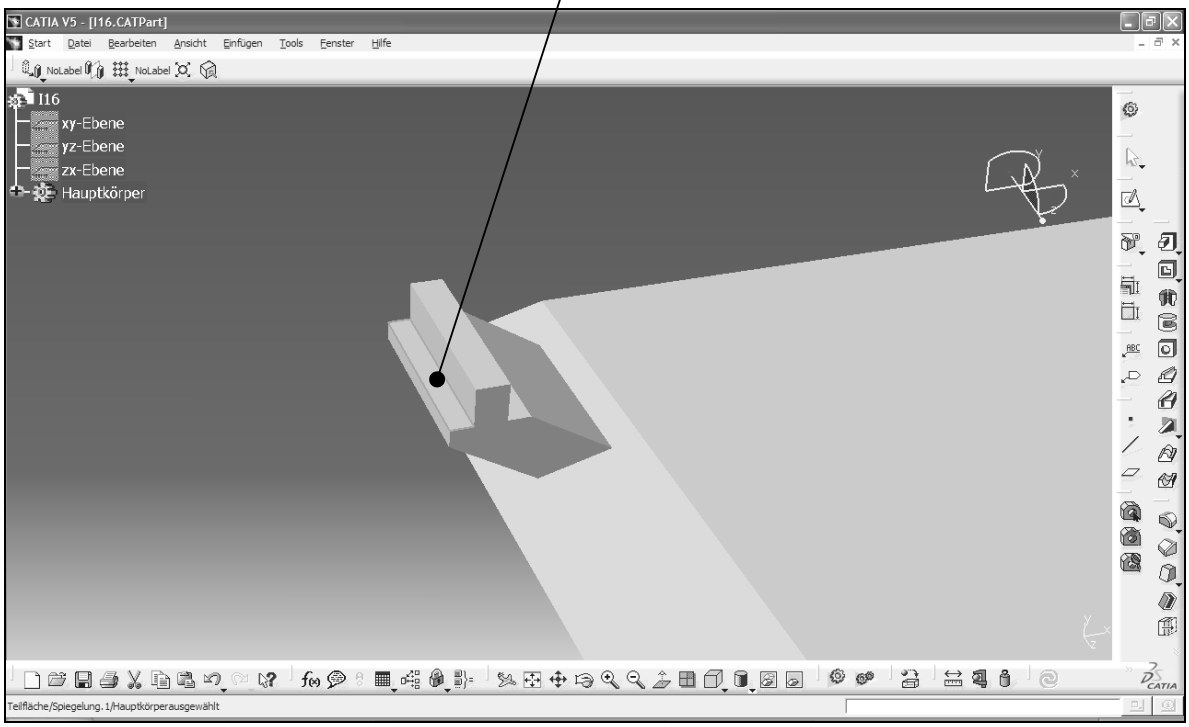


Abb. 6-21 Auswahl der unteren Kontaktfläche

- Im Menu Tools-> Veröffentlichung… die Abfrage mit „Ja“ bestätigen.

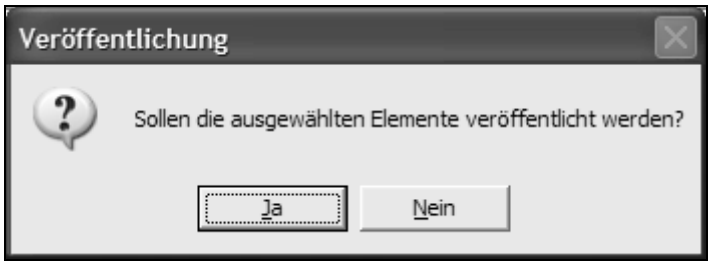


Abb. 6-22 Veröffentlichungsdialog

- Durch Klicken auf den Namen die Umbenennung aktivieren.

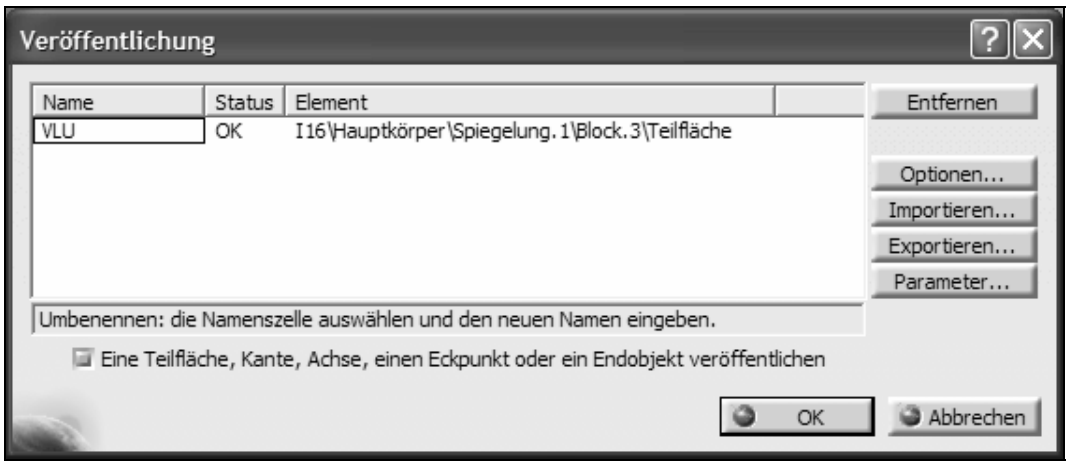


Abb. 6-23 Umbenennung der veröffentlichten Fläche

- Umbenennen in „VUL“ (VorneLinksUnten).
- Die gezeigte Fläche anwählen und umbenennen in „VLI“(VorneLinksInnen).

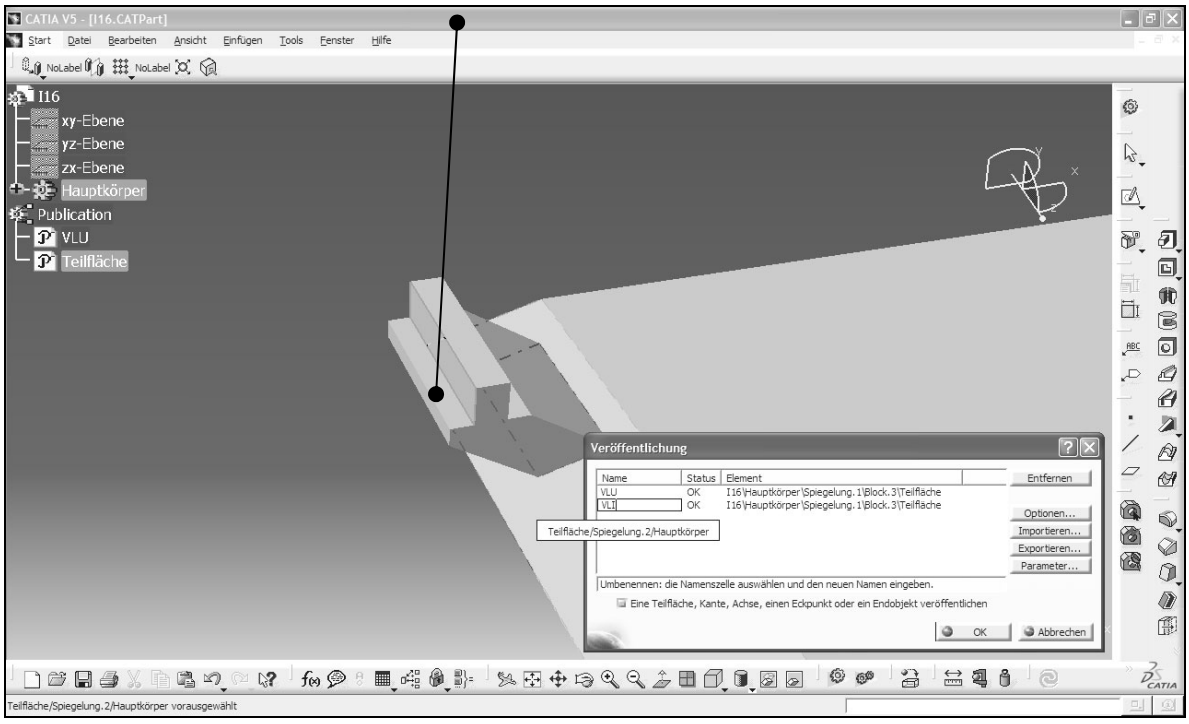


Abb. 6-24 Auswahl der inneren Kontaktfläche

- Veröffentlichungen an allen Verbindungsstücken nach folgendem System fortführen:
  - V = Vorne
  - H = Hinten
  - L = Links
  - R = Rechts
  - U = Unten
  - I = Innen

Bei Bauteilen, die insgesamt nur zwei Verbindungsstücke haben, wird die Benennung der Vorderen gewählt.
An den Schienen werden ebenso die front- und rear-Ebenen eingefügt und die Veröffentlichung der Flächen „Unten“ und „Innen“ durchgeführt. Somit kann die Platzierung der Bauteile später erfolgen, indem die Flächen mit dem Abstand null aufeinander gesetzt werden.

6.6.3 Vorgehen und Probleme

Wie schon in der kompletten Struktur zu erkennen ist, kann man das Programm in drei Hauptteile trennen:

1. Die Vermessung der angelieferten Zeichnung und der zu positionierenden Bauteile.
2. Das Errechnen der Positionen nach den Vorgaben.
3. Das Einladen und Positionieren der Bauteile.

Im ersten Teil liegt der zentrale Punkt in der Vermessung, was bedeutet, dass beispielsweise die Vorder- und Rückseite eines Bauteils im Abstand zueinander vermessen wird, und so die Tiefe des Bauteils feststeht.
Für diesen Fall bietet CATVBA die Methode „GetMinimumDistance()“.
Hier wird zu den zu messenden Flächen jeweils eine Referenz aufgebaut, und diese werden dann zueinander gemessen. Das Ergebnis ist der minimale Abstand der zwei Objekte.
Problematischer wird die Messung im Produkt, da die Methode „GetMinimumDistance()“ leider nur im Part-Design zur Verfügung steht [9, 10].

Bei der Messung der Sitzreihenabstände ist genau dieses Problem aufgetreten. Die Augenpunkte der Sitzreihen müssen hier zueinander vermessen werden.

In einem Part beziehen sich alle Koordinaten auf den Nullpunkt im Part. Betrachtet man ein Produkt, das ein Part enthält, so hat das Produkt einen eigenen Nullpunkt (Abb.6-25 (U,V,W,T)). Im Raum des Produktes liegt der Nullpunkt des Parts (Abb. 6-25 (x,y,z,0)) und auf diesen Nullpunkt bezogen liegt dann die eigentliche Geometrie (Abb. 6-25 *). Die gepunkteten Pfeile (·····►) zeigen die Ortsvektoren im Part bzw. des Parts im Produkt. Der gestrichelte Pfeil (- - - - - ►) ist der resultierende Vektor, der im RootProduct die Position des Punktes global angibt. Ein Produkt kann beliebig viele Unterprodukte enthalten.

Da es keine vorgefertigte Messmethode für dieses Problem gibt, hilft folgende Vorgehensweise:

Die Koordinaten der einzelnen Punkte im Bauteil werden ausgelesen und per Koordinatentransformation bis auf das höchste Produkt (RootProduct) hochgerechnet. Hier kann man dann den räumlichen Vektor zwischen den beiden Punkten errechnen, und der Betrag dieses Vektors ist dann gleich dem Abstand der beiden Punkte zueinander.

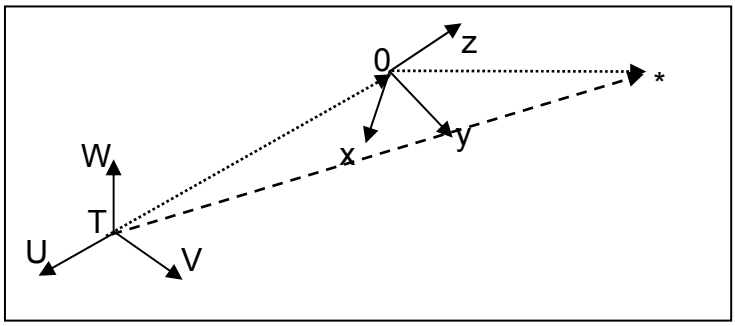


Abb. 6-25 Koordinatensysteme im Produkt [23]

Koordinatentransformation in CATIA V5:

Über die Methode PUNKT.GetCoordinates werden die Koordinaten des Augenpunktes im Part ausgelesen. Es werden die X-, Y- und Z-Komponente übergeben. Anschließend wird über die Methode PRODUCT.Position.getComponents die Koordinatenmatrix des Parts im Product erfragt. Diese Matrix besteht aus 12 Werten:

0	Ux	3	Vx	6	Wx	9	TX
1	Uy	4	Vy	7	Wy	10	Ty
2	Uz	5	Vz	8	Wz	11	Tz

Tab. 6-20 Koordinatenmatrix in CATIA V5

Die ersten neun Werte bezeichnen die Achsenrotation und die letzen Drei die Translation. In dieser Matrix ist also das Lagenverhältnis verpackt, in dem sich das Part zum Produkt befindet.

Die mathematische Erklärung laut [22, 24] fasst die Koordinatentransformation in folgender Formel zusammen:

$$\underline{x}_{neu} = A * \underline{x}_{alt} + \underline{v} \qquad (3)$$

$\underline{x}_{neu}$ = Der Punkt im neuen Koordinatensystem (im Produkt)

A = Die Transformationsmatrix (Tab. 6-2 Einträge 0-8)

$\underline{x}_{alt}$ = Punkt im alten Koordinatensystem (Also unser X,Y,Z)

$\underline{v}$ = Der Verschiebevektor (Tab. 6-2 Einträge 9-11)

Über die Matrizenrechnung kann man nun das gewünschte Ergebnis berechnen. Um dieses zu programmieren, rechnet man die Matrize praktisch per Hand nach folgendem Beispiel durch:

$$X_{neu} = U_x * X + U_y * Y + U_z * Z + T_x \qquad (4)$$

$$Y_{neu} = V_x * X + V_y * Y + V_z * Z + T_y \qquad (5)$$

$$Z_{neu} = W_x * X + W_y * Y + W_z * Z + T_z \qquad (6)$$

Dieses wird so oft wiederholt, bis man sich im RootProduct befindet.
So wird mit allen Punkten verfahren. Aus dem Satz des Pythagoras folgt der Abstand [3]:

$$\text{Abstand} = \sqrt{(X_{p1} - X_{p2})^2 + (Y_{p1} - Y_{p2})^2 + (Z_{p1} - Z_{p2})^2} \qquad (7)$$

Abstand	=	Der Abstand der zwei Punkte zueinander
X_{p1}	=	X-Wert erster Punkt
X_{p2}	=	X-Wert zweiter Punkt
Y_{p1}	=	Y-Wert erster Punkt
Y_{p2}	=	Y-Wert zweiter Punkt
Z_{p1}	=	Z-Wert erster Punkt
Z_{p2}	=	Z-Wert zweiter Punkt

Um Flächen im Produkt zueinander messen zu können, was für die Vermessung der Schienen notwendig ist, benötigt man auch deren Koordinaten. Hier ist einmal der Nullpunkt der Fläche von Nöten und die Flächen aufspannenden Vektoren. Leider gibt CATIA nicht die notwendigen Größen an die Programmier-Schnittstelle weiter. Somit kann eine solche Vermessung nicht durchgeführt werden.

Auch im dritten Teil, der Positionierung, hat sich bei der Umsetzung ein Problem ergeben. Die Positionierung wird in CATIA durchgeführt, indem beispielsweise zwei Flächen zueinander in Beziehung gebracht werden (z.B. Fläche A hat einen Abstand von 0 mm zu Fläche B). Insgesamt können Winkel-, Kongruenz-, Kontakt- und Offsetbedingungen gegeben werden. Für die Positionierung müssen im Produkt die Berührungsflächen der Schienen und der Bauteile öffentlich bekannt sein. CATIA bietet die Möglichkeit, jegliche Art von Geometrie zu veröffentlichen. Diese Geometrien stehen dann unter dem frei zu vergebenen Namen zur Verfügung. Spricht man den Namen an, so ist die Geometrie (oder eine Gruppe von Geometrien) damit referenziert - und genau hier liegt das Problem. Referenzen werden im Bereich Produkt über CATVBA nicht freigegeben. Man kann die veröffentlichten Flächen also über Makros im Produkt nicht ansprechen. Da diese Funktion nicht freigegeben ist, erübrigt sich momentan das „Positioner“-Programm, denn die manuelle Positionierung der Bauteile zu automatisieren ist die Hauptaufgabe des Projektes. Somit konnte das Projekt nicht als praktische Arbeit vollendet werden und wurde, wie

schon belegt, theoretisch behandelt. Bis zum Auftreten der Probleme wurde das Programm umgesetzt und ist im Anhang dieser Arbeit einzusehen.

Ein weiteres grundsätzliches Problem liegt in der Informationsbeschaffung zur Programmierschnittstelle. Wie in den Quellen angegeben, wird zwar bei der Installation der Software CATIA V5 die „AutomationV5“-Referenzdatei [10] gleich mitinstalliert, jedoch bezieht sich diese auf die Programmiersprache CAA. Leider stehen nicht alle CAA-Methoden in den Skriptsprachen zur Verfügung und außerdem ist nicht vermerkt, welche exklusiv in CAA verfügbar sind. Im größten deutschen CATIA V5-Forum [14] stehen zwar grundlegende Informationen zur Verfügung, aber selbst die Administratoren bestätigten, dass die „AutomationV5“-Referenzdatei fehlerbehaftet, und keine weiterführende Literatur auf dem Markt erhältlich ist. Lediglich das Buch „Effiziente Konstruktion mit Makros“ [2] gibt eine Einführung in das Thema. Leider wird hier nicht auf die Makroprogrammierung im Bereich Produkte bzw. Assembly Design, sondern nur auf den Bereich Bauteileerstellung bzw. Part Design eingegangen.

7 Zusammenfassung

Im Zuge der Umstellung der Konstruktionswerkzeuge von 2D auf 3D im Hause Airbus soll schon jetzt bei AEROTEC Engineering untersucht werden, in wie weit die Konstruktionsprozesse mit Hilfe des Tools CATIA V5 unterstützt werden können. Diese Untersuchung sollte an dem Beispiel der Positionierung von Bauteilen in dem Versorgungskanal des Baumusters A340 durchgeführt werden.
In der **Ist Analyse** wurden die heutige Arbeitsweise, das 2½D-Wekzeug CCD und die Vorgaben des Kunden untersucht. Anhand dieser Information konnten die Stärken und Schwächen herausgearbeitet werden. Die Arbeit mit CCD ermöglicht heute die schnelle Reaktion auf Kundenwunschänderungen, jedoch ist die Arbeit sehr zeitaufwendig und durch den immer gleichen Arbeitsablauf sehr fehleranfällig. Es hat sich herausgestellt, dass eine Teilautomatisierung hier einen entscheidenden Vorteil bringen würde.

Die Erarbeitung des **Soll-Konzepts** ergab nach der Untersuchung der verschiedenen Lösungsalternativen, dass für ein Problem dieser Art die Funktionen der CATIA V5 internen Makro-Programmierungsschnittstelle ausreichend sind. Zusätzlich stellte sich heraus, dass dieses gleichzeitig die günstigste Alternative ist, da die Schnittstelle im Standardpaket von CATIA V5 bereits enthalten ist.
Auch im direkten **Vergleich des Ist- und Soll-Zustandes** ergab sich durch Teilautomatisierung der Hauptvorteil in der günstigeren (schnelleren) Ausführung der Aufträge. Bei einer Fertigstellung des Programms würde sich ab dem 19. Auftrag, also am Ende des zweiten Auftragsjahres (bei durchschnittlich 10 Aufträgen dieser Art pro Jahr) eine Kostenersparnis von 550€ pro Auftrag ergeben.
Die theoretische Ausarbeitung des Programmablaufes und die Analyse der Datenstruktur in CATIA V5 ergab ein fertiges Konzept zur Programmrealisierung.

Bei der **Implementierung** jedoch ergaben sich Probleme, die durch mangelnde Informationsquellen nicht voraussehbar waren. Die mitgelieferte Automationsreferenz stellte sich als teilweise fehlerhaft heraus. Auch Literaturquellen stehen wegen der noch sehr jungen Problematik für die gegebene Problemstellung nicht zur Verfügung. Auch die Foren, die sich ansonsten in solchen Fällen als sehr hilfreich erwiesen,

brachten wegen des mangelnden Know-hows nicht viel mehr Informationen. Somit musste es bei der teilweisen Realisierung und der fertiggestellten theoretischen Ausarbeitung des Programmschemas bleiben.

8 Aussicht

Die Aussichten bezogen auf das behandelte Projekt sind insofern gut, als dass sich diese Art der Automation für kleine bis mittlere Projekte durch seine Kostenersparnis in Verbindung mit der Arbeitserleichterung auch schon finanziell rechnet. Man muss jedoch immer beachten, dass CATIA V5 in der Hauptsache nicht zum Positionieren von Teilen, sondern zum Erstellen dieser gedacht ist. Es müssten noch die oben genannten Probleme bei der Makro-Schnittstelle behoben werden, um umfangreiche Funktionen nutzen zu können. Wird dieses realisiert, so ist das betrachtete Projekt sicher gewinnbringend umzusetzen. Zusätzliche Funktionen, wie z.B. das Erstellen der Stückliste, was momentan auch von Hand umgesetzt wird, könnte automatisiert werden, genau wie Schnittbilder und das Erstellen der Aufkleber für die Installation.

Zudem ist aufgefallen, dass weitere Untersuchungen gestartet werden sollten, was den Einsatz der CAA-Schnittstelle betrifft. Der erste Eindruck, der sich beim Erstellen dieser Arbeit ergeben hat zeigt, dass das Know-how über die Programmierung in CATIA V5 im deutschen Markt noch nicht ausreichend vorhanden ist, und sich somit ein möglicher Markt für ein weiteres Standbein eröffnet. Wenn man die Möglichkeit hat, ein wenig Forschung auf dem Gebiet zu betreiben, könnte man extern Programmierungsdienstleistungen für Zusatzmodule (Workbenches) anbieten. Auch bei Airbus ist hier ein Bedarf zu bemerken. Schon die Monopolstellung von z.B. Pace im Bereich Kabinenauslegung mit hinterlegter Wissensdatenbank zeigt, dass in diesem Bereich noch Platz für Alternativlösungen ist.

9 Glossar

2D / 3D	2/3-dimensional
Air-Box	Individualbelüftungssystem im Versorgungskanal
AP	Augenpunkt
Assembly Design	workbench in CATIA V5 zur Produkterstellung (Zusammenbaudatei)
BEP	**B**reak **E**ven **P**oint (Gewinnschwelle)
CAA RADE	C++ ähnliche Programmiersprache in CATIA V5 (**CATIA A**pplication Development; **R**apid **A**pplication **D**evelopment **E**nvironment)
Cadam	Computer Grafic Augumented Design And Manufacturing System
CATScript	Skriptsprache in CATIA V5
CATVBA	Skriptsprache in CATIA V5
CCD	**CATIA C**adam **D**rafting (2½D-Konstruktionstool
EP	Eye Point (Augenpunkt)
Infill-Panels	Leerbleche des Versorgungskanals
Isometrische Ansicht	schiefe Parallelprojektion *(Schrägbild)*
KWA	Knowledge Advisor (Workbench in CATIA V5)
KWE	Knowledge Expert (Workbench in CATIA V5)

Long Range	Flugzeugreihe bestehend aus:

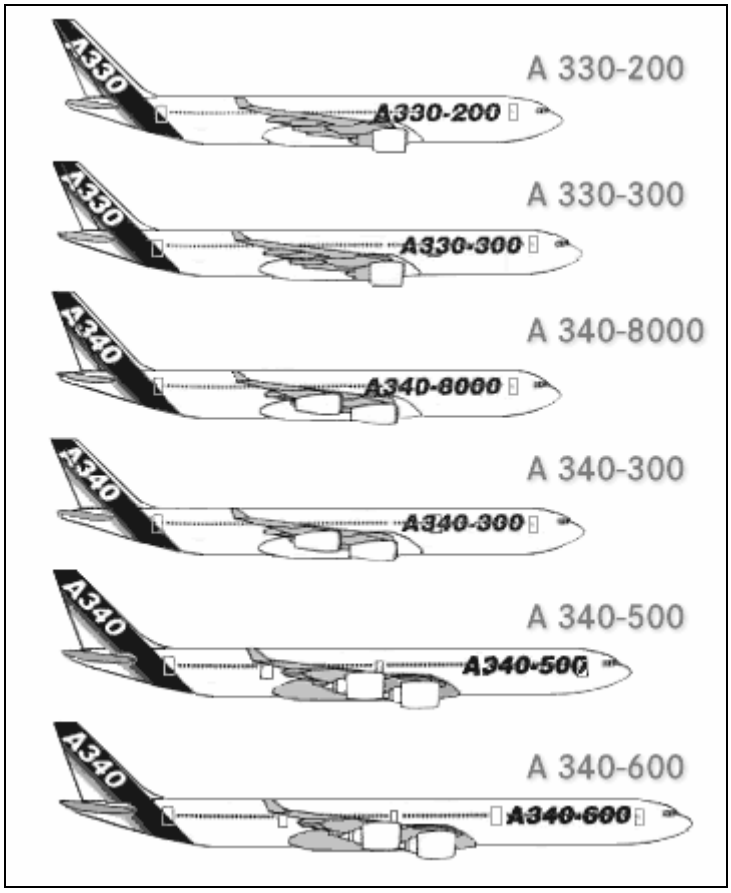


Abb. 9-1 Airbus Flotte Long Range [12]

O_2-Box	Sauerstoffmaskenbox
Pace	Unternehmen, welches eigenständige Positionierungstools anbietet
Plotterplan	fertiger Positionierungsplan ohne Zusatzinformationen (rein grafisch)

Single Aisle	Flugzeugreihe bestehend aus:

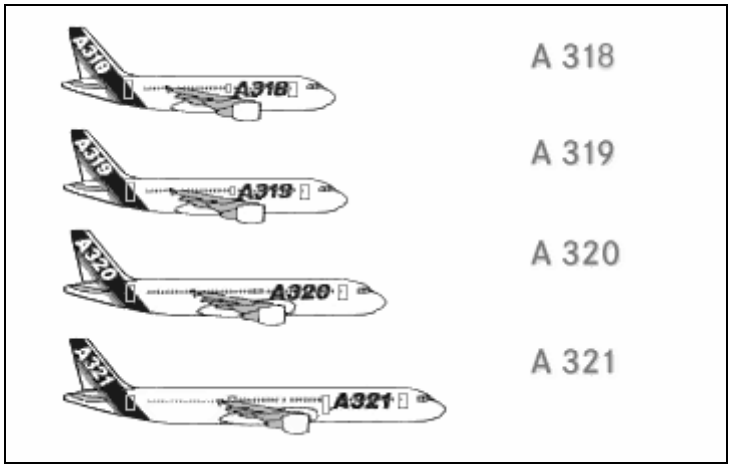


Abb. 9-2 Airbus Flotte Single Aisle [12]

UML	Unified Modeling Language
VE	Versorgungseinheit
Workbenches	Oberflächen (Werkbank) in CATIA V5

10 Bildnachweis

11 Tabellennachweis

12 Quellennachweis

[1] Rudolf W. Rembold
Einstieg in CATIA V5 – Konstruktion in Übungen und Beispielen
2. Auflage 2004
Carl Hanser Verlag München Wien

[2] Dieter R. Ziethen
CATIA V5 – Effiziente Konstruktion mit Makros
2003
Carl Hanser Verlag München Wien

[3] Lothar Papula
Mathematische Formelsammlung – Für Ingenieure und Naturwissenschaftler
7. Auflage 2001
Friedrich Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden

[4] N/A; RRZN
Visual Basic 6.0 – Grundlagen
3.Auflage September 2000

[5] N/A; RRZN
Visual Basic 6.0 – Fortgeschrittene Programmierung
1.Auflage September 2001

[6] N/A; Prostep AG
Handout Knowledge Based Engineering mit CATIA V5 - Workshop
03. Februar 2005

[7] Grégory Lasserre
Diplomarbeit: Development of a knowledge-based CATIA V5 master-part for sheet-metal cleats
Institut Français de mecanique avancee 10.06.2004

[8] Hansdieter Laskowski; CENIT AG Systemhaus
Leitfaden Systemadministration CATIA Version 5
4. Journal August 2004
Olzog Verlag GmbH

[9] N/A; IBM
Deutsche Onlinedokumentation CATIA V5 Release 14

[10] N/A
CAA V5 Visual Basic help
Programm mitgelieferte Automationsreferenz

[11] Schleich, Heinrich
Skript Projektführung und Controlling
2005

[12] Intranet Aerotec Engineering GmbH
15.07.2005

[13] N/A
Bildergalerie
http://www.lightforum.ch
Stand: 19.05.2005

[14] N/A
Größte deutschsprachige CAD-Communitiy
http://www.CAD.de
Stand: 10.06.2005

[15] Dassault Systèmes®
Automation Developers
http://www.3ds.com/alliances/automation-developers
Stand: 22.06.2005

[16] COE
Discussion Forum CATIA V5
http://www.coe.org/forums
Stand: 06.06.2005

[17] Inoffizielle CATIA-Hilfeseite
http://CATIA.cad.de/
Stand: 10.06.2005

[18] IBM
Programminformationen und Workbenches CATIA V5
http://www-1.ibm.com/solutions/plm/country/de/produkte/CATIA V5.html
Stand: 18.05.2005

[19] TU- Darmstadt FB Maschinenbau
CATIA V5 Praktikum TU-Darmstadt
http://eos.dik.maschinenbau.tu-darmstadt.de/projects/cadp_CATIA/deutsch/WebR5
Stand: 15.05.05

[20] FH Ulm
ICT Institut für CAD-Technologien
Lehrunterlagen CATIA V5
http://info.cad.fh-ulm.de/CATIA_v5/
Stand: 15.05.05

[21] Microsoft developer's network
VBA-Sprachreferenz
http://msdn.microsoft.com/
Stand: 21.06.05

[22] Uni Münster
Koordinatentransformation
http://ifgifor.uni-muenster.de/vorlesungen/Geoinformatik/frames/fsteuer.htm
Stand: 28.06.05

[23] Uni Osnabrück
Transformation von Koordinatensystemen
http://www-lehre.informatik.uni-osnabrueck.de/~cg/2000/skript/13_4_Transformation_von.html
Stand: 21.06.05

[24] Wissenschaft Online
Vektor- und Matrizenrechnung, räumliche Koordinatentransformation
http://www.wissenschaft-online.de/spektrum/projekt/quasi5.htm
31.05.2005 14:15Uhr
Stand: 21.06.05

[25] Pace
Unternehmenshomepage
http://www.pace.de
Stand: 18.05.05

[26] Schulungsunterlagen CATIA. CADAM Drafting (CCD)
Basisschulung
DaimlerChrysler Aerospace Airbus

13 Anhang Programmcode

13.1 Formulare

13.1.1 Selection

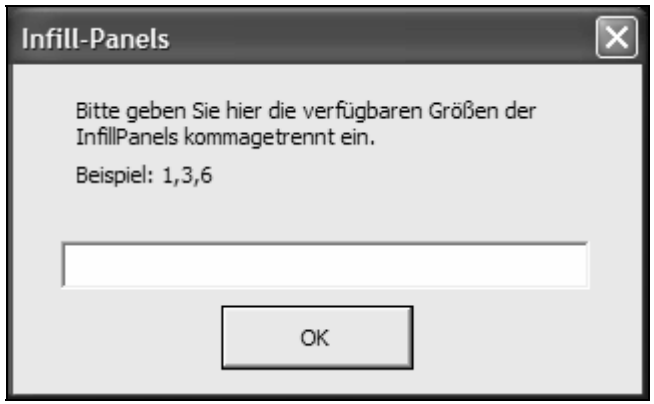

```
Public Partfield
Public iPartquantum As Integer

'*****************************************************************************************************
'   purpose:            cancels the "Positioner"
'
'*****************************************************************************************************
Private Sub ABR1_Click()
      End
      Exit Sub
End Sub
```

```
'*****************************************************************************************************
'   purpose:            changes the status of the checkboxes when "All" is chosen
'
'*****************************************************************************************************

Private Sub All_1_Click()

If (bAll_1 = False) Then
    bAll_1 = True
Else
    bAll_1 = False
End If

If (All_1 = True) Then
    bO2_1 = True
    bVE_1 = True
    bAir_1 = True
    bInf_1 = True
Else
    bO2_1 = False
    bVE_1 = False
    bAir_1 = False
    bInf_1 = False
End If


End Sub
```

```
'*********************************************************************************************
'   purpose:            analyses the inputfields after pressing the "OK"-Button and collects
'                       information of the parts
'
'*********************************************************************************************
Private Sub OK_1_Click()

Selection.Hide

Dim O2box As String
Dim aDoc As document

Dim productDocument1 As ProductDocument
Set productDocument1 = CATIA.ActiveDocument

Dim product1 As product
Set product1 = productDocument1.product

Dim products1 As Products
Set products1 = product1.Products

Dim arrayOfVariantOfBSTR1(0)
iPartquantum = 0
ReDim Partfield(iPartquantum, 2)

If (bO2_1 = True) Then

    Partfield = tools.ChangeArray(Partfield, 1, iPartquantum)
    Partfield(UBound(Partfield, 1), 0) = "O2-Box"
    Partfield(UBound(Partfield, 1), 1) = tools.Path(Partfield(UBound(Partfield, 1), 0))
    iPartquantum = 1

End If
If (bVE_1 = True) Then
    Partfield = tools.ChangeArray(Partfield, 1, iPartquantum)
    Partfield(UBound(Partfield, 1), 0) = "VE-Box"
    Partfield(UBound(Partfield, 1), 1) = tools.Path(Partfield(UBound(Partfield, 1), 0))
    iPartquantum = 1
```

```
End If
If (bAir_1 = True) Then
    Partfield = tools.ChangeArray(Partfield, 1, iPartquantum)
    Partfield(UBound(Partfield, 1), 0) = "Air-Box"
    Partfield(UBound(Partfield, 1), 1) = tools.Path(Partfield(UBound(Partfield, 1), 0))
    iPartquantum = 1

End If
If (bInf_1 = True) Then
    InfillPanels.TextBox1.text = ""
    InfillPanels.Show
    Partfield = tools.ChangeArray(Partfield, 1, UBound(InfillPanels.Inffield, 1) + 1)
    For i = 0 To UBound(InfillPanels.Inffield, 1)

        Partfield(UBound(Partfield, 1) - UBound(InfillPanels.Inffield) + i, 0) = InfillPanels.Inffield(i, 0)
'copying the informations of "Inffield" int "Partfield"
        Partfield(UBound(Partfield, 1) - UBound(InfillPanels.Inffield) + i, 1) = InfillPanels.Inffield(i, 1)

    Next

End If



For h = 0 To UBound(Partfield, 1)       'each Part gets measured in it's width
    Dim sPath As String
    sPath = (Partfield(h, 1))
    Partfield(h, 2) = tools.Measure(sPath)
Next
Distance.Coords
Distance.Rails
End Sub
```

13.1.2 Infill-Panels




```
Public Inffield
'*************************************************************************************************
'   purpose:            Asks for the InfillPanels wich are available
'
'*************************************************************************************************
Private Sub OK_2_Click()

InfillPanels.Hide

    Dim sInput As String
    Dim iQuantity
    Dim stock
    Dim str As String


    If TextBox1.text <> "" Then             'The inputstring gets seperated in the single parts
        sInput = TextBox1.text
        stock = Split(sInput, ",")
        iQuantity = UBound(stock)           'counting how many InfillPanels are available
        ReDim Inffield(iQuantity, 2)
        For i = 0 To iQuantity              'each is stored by name and path in "Inffield"
            Inffield(i, 0) = "InfillPanel " & stock(i) & " Zoll"
```

```
            Inffield(i, 1) = tools.Path("InfillPanel " & stock(i) & " Zoll")
        Next

    End If
End Sub
```

13.2 Methoden / Funktionen

13.2.1 Modul StartMain

```
' Copyright AEROTEC Engineering
'*****************************************************************************************
'   purpose:            The Positioner is designed to automate the positioning procedure for
'                       the PSU-channel
'   author:             Andreas Krohn   (Andreas.Krohn@aerotec.de)
'
'   language:           CATVBA
'   version:            V1.1
'
'*****************************************************************************************

Sub CATMain()

    Selection.Show

End Sub
```

13.2.2 Modul Distance

```
Dim RailArray
Dim vDistance As Variant
Public vSeatPosition As Variant


'*********************************************************************************************
'   purpose:            calculates the global position of the Eye-points of each seatline
'
'*********************************************************************************************
Sub Coords()

Dim sName As String

Set osel = CATIA.ActiveDocument.Selection
osel.Clear

Dim afilter(0)
afilter(0) = "Point"
MsgBox ("Bitte wählen sie einen Augenpunkt aus.")
Status = osel.SelectElement(afilter, "Select a point.", False)

Set opoint = osel.Item(1).Value

oname = osel.Item(1).Value.name
osel.EndSelectElement
osel.Clear
osel.Search (".Punkt.Name=" & oname & ";Alle")
iQuantity = osel.count
ReDim vDistance(iQuantity - 1, 1)
ReDim vSeatPosition((iQuantity * 4 - 1))

Dim acoord_in_part(2)
opoint.GetCoordinates acoord_in_part            'the coordinates of the item in the part (local)
```

```
Dim acoord_part_in_product(11)                  'the coordinates of the part in the product
Dim acoord_product_in_sub_product(11)           'the coordinates of the product in the higher leveled product
Dim atmp(2)                                     'coordinate stock
Dim acoord_in_product(2)                        'the coordinates of the item in the product

For j = 1 To osel.count
    Set opart_product = osel.FindObject("CATIAProduct")

    opart_product.Position.GetComponents acoord_part_in_product         'ask for the coordinate-matrix of the part in
the product, including translation and rotation
    ' first transformation of the coordinates from item-coordinates into product-coordinates
    acoord_in_product(0) = acoord_part_in_product(0) * acoord_in_part(0) + acoord_part_in_product(3) *
acoord_in_part(1) + acoord_part_in_product(6) * acoord_in_part(2) + acoord_part_in_product(9)
    acoord_in_product(1) = acoord_part_in_product(1) * acoord_in_part(0) + acoord_part_in_product(4) *
acoord_in_part(1) + acoord_part_in_product(7) * acoord_in_part(2) + acoord_part_in_product(10)
    acoord_in_product(2) = acoord_part_in_product(2) * acoord_in_part(0) + acoord_part_in_product(5) *
acoord_in_part(1) + acoord_part_in_product(8) * acoord_in_part(2) + acoord_part_in_product(11)


    Do
        Set oparent_product = opart_product.Move.Parent.Parent
        oparent_product.Position.GetComponents acoord_product_in_sub_product    'ask for the coordinate-matrix of the
Product in the higher leveled product

        atmp(0) = acoord_in_product(0) 'copying the transformed coordinates in another stock
        atmp(1) = acoord_in_product(1)
        atmp(2) = acoord_in_product(2)
        'transformation of the coordinates from (sub)product-coordinates into (higher leveled) product-coordinates
        acoord_in_product(0) = acoord_product_in_sub_product(0) * atmp(0) + acoord_product_in_sub_product(3) * atmp(1)
+ acoord_product_in_sub_product(6) * atmp(2) + acoord_product_in_sub_product(9)
        acoord_in_product(1) = acoord_product_in_sub_product(1) * atmp(0) + acoord_product_in_sub_product(4) * atmp(1)
+ acoord_product_in_sub_product(7) * atmp(2) + acoord_product_in_sub_product(10)
        acoord_in_product(2) = acoord_product_in_sub_product(2) * atmp(0) + acoord_product_in_sub_product(5) * atmp(1)
+ acoord_product_in_sub_product(8) * atmp(2) + acoord_product_in_sub_product(11)


    Loop Until CATIA.ActiveDocument.product.name = oparent_product.name ' loop until there is no higher product
```

```
    vSeatPosition(j * 4 - 4) = opart_product.name       'storing the name and coordinates of the points into
vSeatPosition
    vSeatPosition(j * 4 - 3) = acoord_in_product(0)
    vSeatPosition(j * 4 - 2) = acoord_in_product(1)
    vSeatPosition(j * 4 - 1) = acoord_in_product(2)



Next
For f = 1 To (iQuantity)

    vDistance(f - 1, 0) = vSeatPosition(f * 4 - 4)
    vDistance(f - 1, 1) = tools.Distance(vSeatPosition, (f * 4 - 3))

Next

End Sub


'*********************************************************************************************
'   purpose:            calculates the global position of the Rails
'   problems:           does not work because of missing procedures by 3DS
'
'*********************************************************************************************

Sub Rails()

    Dim iStop As Integer
    iStop = 1


    If iStop = 1 Then

    Set FS = CreateObject("Scripting.FileSystemObject") 'creates a textfile and saves the content of "Partfield" and
"Seatposition"
    Set A = FS.CreateTextFile("c:\testfile.txt", True)
```

```
    A.Writeline ("Array Partfield:")
    A.WriteBlankLines (1)

    For i = 0 To UBound(Selection.Partfield, 1)

        A.Writeline (Selection.Partfield(i, 0) & "   " & Selection.Partfield(i, 1) & "   " & Selection.Partfield(i, 2))
    Next

    A.WriteBlankLines (2)
    A.Writeline ("Array vSeatPosition:")
    A.WriteBlankLines (1)

    For i = 0 To UBound(Distance.vSeatPosition, 1) Step (4)

        A.Writeline (vSeatPosition(i) & "   " & vSeatPosition(i + 1) & "   " & vSeatPosition(i + 2) & "   " &
vSeatPosition(i + 3))
    Next

    A.WriteBlankLines (2)
    A.Writeline ("Array vDistance:")
    A.WriteBlankLines (1)

    For i = 0 To UBound(Distance.vDistance, 1)

        A.Writeline (Distance.vDistance(i, 0) & "   " & Distance.vDistance(i, 1))
    Next

    A.Close

    End If
End Sub
```

13.2.3 Modul Tools

```
'*********************************************************************************************
'   purpose:            calculates the global distance of the given Points
'   inputs:             ipoint1:        the vSeatPosition Array
'                       ipoint2:        the vSeatPosition Array
'                       iStartPoint1:   Index of the first SeatRow
'                       iStartPoint2:   Index o the SeatRow to be measured
'
'*********************************************************************************************
Function Distance(iPoint, iStartPoint)

    Dim Result(2) As Variant

    Result(0) = iPoint(iStartPoint) - iPoint(1)
    Result(1) = iPoint(iStartPoint + 1) - iPoint(2)
    Result(2) = iPoint(iStartPoint + 2) - iPoint(3)

    Distance = (Result(0) ^ 2 + Result(1) ^ 2 + Result(2) ^ 2)
    Distance = Distance ^ (1 / 2)

End Function


'*********************************************************************************************
'   purpose:           Opens the fileselectionbox and returns the papth of the selected file
'   input:             sName:   The name of the searched Part
'
'*********************************************************************************************

Function Path(sName)

    Path = CATIA.FileSelectionBox("Bitte die " & sName & " wählen.", "*.CATPart", CatFileSelectionModeOpen)

End Function
```

```
'*************************************************************************************************
'   purpose:            changes the dimension of the given array (max. 2 dimension arrays)
'   inputs:             array1      The array wich needs to be changed in size
'                       dim1        which dimension of the array needs to be changed
'                       iAddValue   Value wich has to be added to the dim1
'
'*************************************************************************************************

Function ChangeArray(array1, dim1, iAddValue)

Dim array2
Dim iKeepValue As Integer
Dim iChangeValue As Integer

array2 = array1             'saving the content of the given array
If dim1 = 1 Then
    iKeepValue = UBound(array2, 2)
    iChangeValue = UBound(array2, 1)
    iChangeValue = iChangeValue + iAddValue
    ReDim array1(iChangeValue, iKeepValue)  'redimension of the given array in the first dimension
ElseIf (dim1 = 2) Then
    iKeepValue = UBound(array2, 1)
    iChangeValue = UBound(array2, 2)
    iChangeValue = iChangeValue + iAddValue
    ReDim array1(iKeepValue, iChangeValue)  'redimension of the given array in the second dimension
End If

max1 = UBound(array2, 1)
max2 = UBound(array2, 2)

For i = 0 To max1                   'write back the stored content of the given array
    array1(i, 0) = array2(i, 0)
    For j = 1 To max2
        array1(i, j) = array2(i, j)
    Next
Next
    ChangeArray = array1
End Function
```

```
'*********************************************************************************************
'   purpose:            calculates the width of a part
'   input:              sElementPath    The path of the Part wich has to be measured
'
'*********************************************************************************************

Function Measure(sElementPath As String)


Dim documents1 As Documents
Set documents1 = CATIA.Documents

Dim partDocument1 As PartDocument
Set partDocument1 = documents1.Open(sElementPath)

Set partDocument1 = CATIA.ActiveDocument

Dim part1 As Part
Set part1 = partDocument1.Part

Set osel = partDocument1.Selection
osel.Clear
                'search for the plane named "front" by he user
osel.Search ("(Name=front & (((FreeStyle.Ebene + 'Part Design'.Ebene) + 'Generative Shape Design'.Ebene) + 'Funktionale
Gussform'.Ebene));Alle)")
Set ofl1 = osel.Item(1).Value


osel.Clear

                 'search for the plane named "rear" by he user
osel.Search ("(Name=rear & (((FreeStyle.Ebene + 'Part Design'.Ebene) + 'Generative Shape Design'.Ebene) + 'Funktionale
Gussform'.Ebene));Alle")
Set ofl2 = osel.Item(1).Value

osel.Clear
osel.Add ofl1
osel.Add ofl2
```

```
Dim ref1 As reference
Dim ref2 As reference

Set ref1 = part1.CreateReferenceFromObject(osel.Item(1).reference)
Set ref2 = part1.CreateReferenceFromObject(osel.Item(2).reference)

Set thespaworkbench = CATIA.ActiveDocument.GetWorkbench("SPAWorkbench")
Set themeasurable = thespaworkbench.GetMeasurable(ref2)
Measure = themeasurable.GetMinimumDistance(ref1)        'measure between the to references (not working in Products)
CATIA.ActiveDocument.Close


End Function
```