



## **Studienarbeit**

# **Vollautomatisiertes Erstellen eines parametrisierten Bauteils in CATIA V5, gesteuert über die COM-Schnittstelle, mittels Scriptsprache Python**

Eingereicht bei

Fachgebiet Maschinenbau-Informatik

Prof. Dr.-Ing. H.-B. Woyand

von

Harry Brodd

Wuppertal, den 17.01.2006

# Inhaltsverzeichnis

<b>1</b>	<b>Zielsetzung</b> .....	<b>- 1 -</b>
<b>2</b>	<b>Funktionalität innerhalb CATIA V5</b> .....	<b>- 2 -</b>
<b>3</b>	<b>Python</b> .....	<b>- 4 -</b>
3.1	Installation & Konfiguration .....	- 4 -
3.2	Programmaufbau .....	- 4 -
3.3	early Binding / late Binding.....	- 5 -
<b>4</b>	<b>Unterschiede zu CATScript/VBScript</b> .....	<b>- 6 -</b>
4.1	Enumerated Types .....	- 6 -
4.2	Aufruf von Methoden / User-Commands .....	- 7 -
4.3	Statische / Dynamische Typbindung .....	- 8 -
<b>5</b>	<b>Literaturhinweise &amp; Hilfeseiten im WWW</b> .....	<b>- 9 -</b>

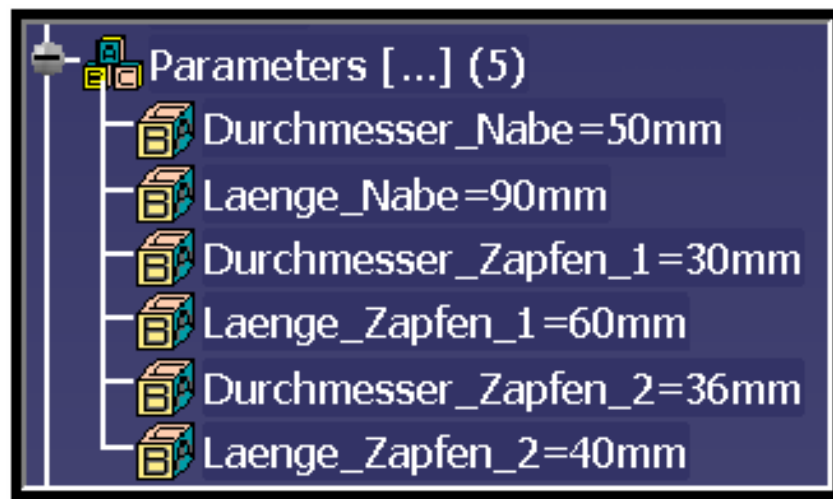


## 2 Funktionalität innerhalb CATIA V5

Das Python-Script CATIA.py erzeugt vollautomatisch eine vereinfachte abgesetzte Welle mit Passfedernut, Fasen und Radien.

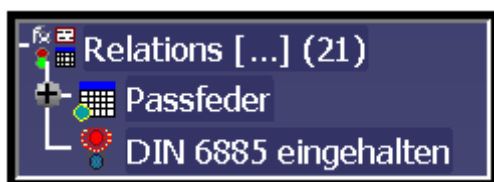
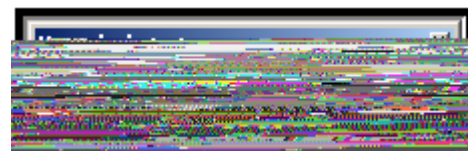
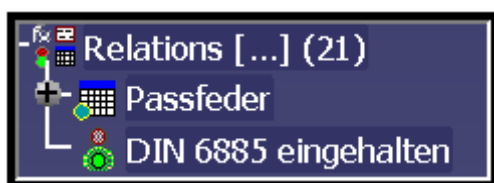
Das Bauteil ist vollständig parametrisiert aufgebaut. Die Abmessungen der Welle werden über Parameter gesteuert, die der Benutzer verändern kann.

Die steuernden Parameter sind in dieser Abb. zu sehen::



(Alle Screenshots aus CATIA V5 wurden mit Umgebungssprache Englisch erstellt.)

Die Abmaße der Passfedernut werden über eine Konstruktionstabelle bestimmt. Die Tabelle wird durch das Script zur Laufzeit erstellt, in der Datei Passfeder.txt gespeichert und in CATIA V5 eingelesen. Der Benutzer wählt die gewünschte Größe durch die entsprechende Zeile der Konstruktionstabelle. Ob die gewählte Passfedernut der DIN 6885 entspricht wird anhand eines ‚Checks‘ geprüft. Das Ergebnis der Prüfung wird im Strukturbaum sowie im Knowledge-Befehlsfenster anhand einer grün bzw. rot leuchtenden Ampel dargestellt.



Die Fasen und Radien werden über Regeln gesteuert und passen sich der Bauteilgröße entsprechend an.

Verändert der Benutzer einen Parameter so, dass er außerhalb des gültigen bzw. sinnvollen Bereichs liegt, wird in einem Pop-Up-Fenster eine entsprechende Fehlermeldung ausgegeben. Die Prüfung der Parameterwerte erfolgt in einer Regel.

Die Grenzen unabhängiger Parameter sind direkt als Parametergrenzwerte festgelegt.

Parameter, Formeln und Regeln, die der Benutzer nicht verändern bzw. einsehen muss, werden aus Gründen der Übersichtlichkeit im Strukturbaum nicht angezeigt (Hidden). Versteckte Elemente können vom Benutzer wieder eingeblendet werden. Um z.B. versteckte Parameter wieder einzublenden wird nach einem Rechtsklick auf *Parameters* im Strukturbaum, im aufgeklappten Menu der letzte Eintrag *Parameters Object* →, *Hidden Parameters* gewählt. Im Pop-Up-Fenster *Hidden Parameters* können nun alle versteckten Elemente angewählt und mit *Show* wieder eingeblendet werden.

## 3 Python

### 3.1 Installation & Konfiguration

Um die COM-Schnittstelle über Python ansprechen zu können, muss das Modul `win32com` für Python installiert werden. Empfehlenswert ist der Download des kompletten Installationspakets *ActivePython* von folgender Webseite: <http://www.activestate.com/Products/ActivePython/>

Nach der erfolgreichen Installation von ActivePython, startet man *PythonwIDE* über das Start-Menue.

Um zu überprüfen, ob CATIA V5 (COM-Server) über Python (COM-Client) angesprochen werden kann, wählt man im Pulldown-Menue *Tools* den Eintrag *COM Makepy Utility* und sucht in der Liste nach Einträgen die mit *CATIA V5...* beginnen.

Sind keine CATIA-Einträge vorhanden, muss CATIA V5 noch freigeschaltet werden. Hierzu wechselt man in der Eingabeaufforderung ins CATIA-Programmverzeichnis: `...\intel_a\code\bin` und ruft `cnext /regserver` auf.

Weitere Informationen hierzu findet man auch in der CATIA Online-Dokumentation unter: *About Object Linking and Embedding (OLE)* im *CATIA Infrastructure User Guide*

Danach ist evtl. ein Neustart von Windows nötig.

Das Script wird über *File* → *Open* in PythonWin geladen und mit *File* → *Run* („Strg+R“) und anschließendem bestätigen mit *OK* gestartet.

Es erscheint eine Input-Box, die zum eingeben eines Namens auffordert. Nach bestätigen mit *OK* wird das CATPart erzeugt.

Es ist nicht zwingend erforderlich CATIA V5 vorher gestartet zu haben, aber wegen der relativ langen Wartezeit beim Start von CATIA V5 durchaus zu empfehlen.

### 3.2 Programmaufbau

Die Basis eines Python-Skripts für CATIAV5 besteht aus folgendem Code:

```
import win32com.client  
  
CATIA = win32com.client.Dispatch("CATIA.Application")
```

In der ersten Zeile wird das Modul `win32com` importiert, Python kann jetzt als COM-Client fungieren.



## 4 Unterschiede zu CATScript/VBScript

Da die Syntax bzw. die zu übergebenden Parameter bei der Programmierung unter Python nicht immer mit den Angaben in den oben genannten Quellen übereinstimmen, folgt hier eine Auflistung einiger Besonderheiten bzw. Unterschiede.

### 4.1 Enumerated Types

Bei der Übergabe von *Enumerated Types* (Aufzählungstypen), darf nicht der Name übergeben werden, sondern es muss der entspr. Integerwert angegeben werden. Das ist unter anderem im Skizzier-Modus bei der Erzeugung von Constraints (Bedingungen) und Dimensions (Bemaßung) der Fall.

Auch bei der Erzeugung Körpern müssen die Parameter als Integerwert angegeben werden, z.B. die Begrenzungen bei Block oder Tasche.

Die folgenden Tabellen zeigen verschiedene ‚Enumerated Types‘ und den zugehörigen Integerwert:

#### Constraint-Typen

catCstTypeReference	0	catCstTypeMidPoint	16
catCstTypeDistance	1	catCstTypeEquidistance	17
catCstTypeOn (=Koinzidenz)	2	catCstTypeMajorRadius	18
catCstTypeConcentricity	3	catCstTypeMinorRadius	19
catCstTypeTangency	4	catCstTypeSurfContact	20
catCstTypeLength	5	catCstTypeLinContact	21
catCstTypeAngle	6	catCstTypePoncContact	22
catCstTypePlanarAngle	7	catCstTypeChamfer	23
catCstTypeParallelism	8	catCstTypeChamferPerpend	24
catCstTypeAxisParallelism	9	catCstTypeAnnulContact	25
catCstTypeHorizontality	10	catCstTypeCylinderRadius	26
catCstTypePerpendicularity	11	catCstTypeStContinuity	27
catCstTypeAxisPerpendicularity	12	catCstTypeStDistance	28
catCstTypeVerticality	13	catCstTypeSdContinuity	29
catCstTypeRadius	14	catCstTypeSdShape	30

#### DimensionType

(Art der Bemaßung: treibend – getrieben)

catCstModeDrivingDimension	0
catCstModeDrivenDimension	1

### **CatLimitMode**

(Begrenzung bei Block/Tasche)

catOffsetLimit,	0
catUpToNextLimit,	1
catUpToLastLimit,	2
catUpToPlaneLimit,	3
catUpToSurfaceLimit,	4
catUpThruNextLimit	5

## **4.2 Aufruf von Methoden / User-Commands**

Die hier aufgeführten Besonderheiten gelten teilweise auch beim Vergleich von CATScript mit BasicScript.

Der Aufruf von Subroutinen (Ohne Wertübergabe) muss mit einer leeren Klammer am Ende erfolgen:

**MySelection.Clear()**

Auch einige Methoden müssen zwingend mit einer leeren Klammer am Ende aufgerufen werden.

**NewBody=MyBodies.Add()**

Unter CATScript und VBScript geht es auch ohne.

Um in den Skizziermodus zu gelangen, bzw. ihn wieder zu verlassen genügen unter CATScript/VBScript folgende Zeilen:

**MySketch.OpenEdition()**

**MySketch.CloseEdition()**

Unter Python scheinen diese Aufrufe keinen Einfluss zu haben, da es aber keine negativen Auswirkungen gab, wurden Sie im erstellten Script eingefügt.

In Python müssen sog. *User-Commands* zum Einsatz kommen. Diese *User-Commands* entsprechen einer Eingabe in der CATIA Kommando-Zeile, rechts unten im CATIA-Fenster.

Um in den Skizziermodus zu gelangen lautet die Befehlszeile

**CATIA.StartCommand("Sketch"),**

und zum verlassen der Skizzierumgebung

**CATIA.StartCommand("Exit workbench").**

Im der Kommandozeile von CATIA würden die entsprechenden Befehle so eingegeben (c steht für command):

*c:Sketch* zum öffnen, bzw. *c:Exit workbench* zum verlassen des Sketchers.

Bei deutscher Sprachumgebung müssen die Befehle und Eingaben entsprechend angepasst werden (*b:Skizze, b:Umgebung verlassen*).

### 4.3 Statische / Dynamische Typbindung

Eine explizite Deklaration der Variablen und Objektnamen wird mit Python nicht durchgeführt. Der Datentyp ist an das Objekt (den *Wert*) gebunden und nicht an eine Variable, d. h. Datentypen werden dynamisch vergeben → dynamische Typbindung.

Einfach gesagt: Variablen existieren von dem Moment an, da ihnen etwas zugewiesen wird, und sie verschwinden wieder, wenn sie nicht mehr verwendet werden.

Folgender Auszug des gleichen Programm-Abschnittes zeigt den Unterschied zwischen Python (dynamisch) und VisualBasic (statisch):

<u>Python</u>	<u>VBA</u>
	Dim MyGeo1 As GeometricElements
MyGeo1= MySketch.GeometricElements	Set MyGeo1 = MySketch.GeometricElements
	Dim Axis As Axis2D
Axis=MyGeo1.Item("AbsoluteAxis")	Set Axis = MyGeo1.Item("AbsoluteAxis")
	Dim AxisH As Line2D
AxisH = Axis.GetItem("HDirection")	Set AxisH = Axis.GetItem("HDirection")
	Dim AxisV As Line2D
AxisV = Axis.GetItem("VDirection")	Set AxisV = Axis.GetItem("VDirection")
	Dim PW0 As Point2D
PW0 = MyFac.CreatePoint(x[0],y[0])	Set PW0 = MyFac.CreatePoint(x(0), y(0))

## 5 Literaturhinweise & Hilfeseiten im WWW

Als Informationsquelle zur Makro-Programmierung von CATIA V5 ist zu allererst die Hilfedatei ‚CAA V5 Visual Basic help‘ zu nennen. Sie befindet sich im CATIA Programmverzeichnis ...\\intel\_a\\code\\bin und heißt **V5Automation.chm**

Als Einstieg in die Programmierung ist das Buch ‚**CATIA V5 – Effiziente Konstruktion mit Makros**‘ von Dieter R. Ziethen zu empfehlen.

Es gibt zur Zeit leider keine weiteren Publikationen zu dieser Thematik, so dass hier auf entsprechende Foren im Internet verwiesen werden muss:

- sehr gutes CATIA-Forum auf [www.cad.de](http://www.cad.de)

Support-Seite von Dassault-Systemes <http://www.3ds.com/alliances/automation-developers/>

Die hier genannten Informationsquellen zur Makro-Programmierung beziehen sich natürlich auf CATSkript , CATIA-VBScript bzw. CATIA-VBA, so dass sie nicht uneingeschränkt für die Programmierung unter Python gelten.

Weitere Informationen zu Python, den Win32-Extensions und Python/COM findet man im Internet auf folgenden Seiten:

<http://www.python.org/>

<http://starship.python.net/crew/mhammond/>

<http://www.win32com.de/>

Unter win32com.de findet man auch einige kleine Beispiel-Scripte zur Ansteuerung von CATIA V5.