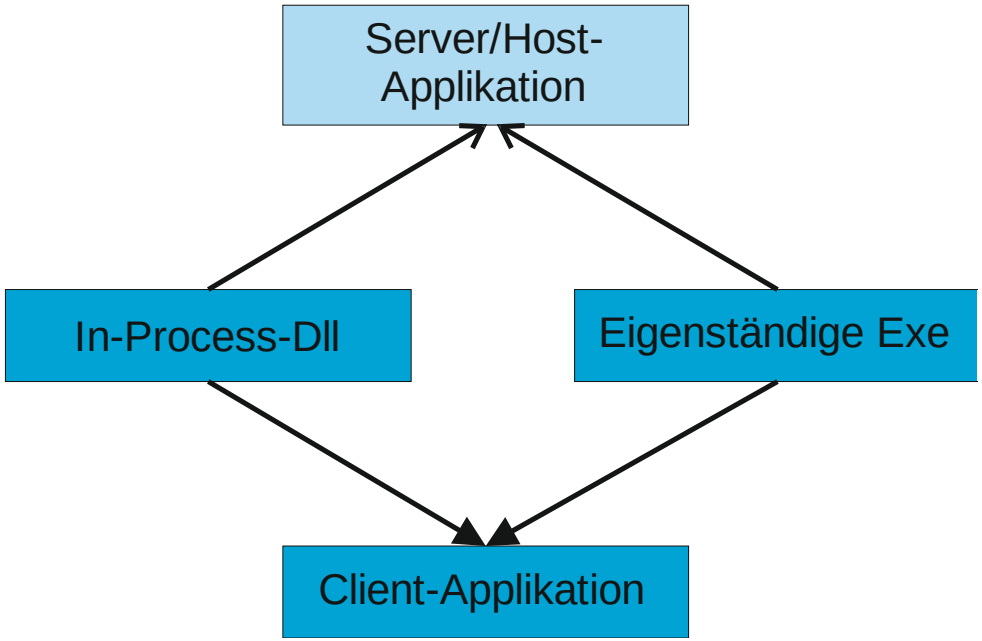
Automatisierung

Grundlegendes

- Damit ein Programm automatisiert werden kann, müssen folgende Voraussetzungen erfüllt sein
  - das Vorhandensein einer konventionellen API (z.B. als Dll) mit Header-Files (DllImport – Attribut in C#), mit der ein äußerer Zugriff auf das Programm geschaffen wird oder
  - die VBA-Unterstützung des Programms, die dann als COM-Interface für den Zugriff genutzt werden kann oder
  - eine .NET API, die direkt über einen Verweis eingebunden werden kann oder
  - die Möglichkeit, ein Programm über AddIns zu erweitern oder
  - eine eigene ins das Server-Programm integrierte Makrosprache (man könnte VBA als solches bezeichnen) oder als letzte (schlechteste) Möglichkeit
  - die Steuerfähigkeit des Programms über Tastatur-Eingaben

- Durch die weite Verbreitung von VBA, das quasi eine Art Standard für Programmierschnittstellen geworden ist, lassen sich im professionellen Bereich fast alle Applikationen über COM automatisieren. Systemeigene Makrosprachen finden in heutigen Projekten kaum noch Verwendung und werden von den System i.d.R. nur noch aus Kompatibilitätsgründen für alte Entwicklungen angeboten
- Durch die Problematik, dass es VBA auf 64bit-Systemen nicht regulär gibt und nur über Tricks verfügbar ist und daher teilweise nicht sauber läuft, bietet COM z.B. über C# eine wirkliche Alternative

Zugriff auf Applikationen



- Unterschieden werden kann, ob das Automatisierungsmodul im Hostprozess läuft (z.B. als Dll dynamisch gebunden) oder als eigenständige Exe den Hostprozess steuert
  - Funktionalität des Programms entscheidet i.d.R. die Anbindungsvariante
  - Eigener Prozess bei Massendatenverarbeitung häufig günstiger
- Als Exe wird ein zweites Programm gestartet, das die Server-Applikation fernsteuert.
- Bei der Dll wird die Erweiterung i.d.R. über entsprechende Befehle hinzugeladen (Add-In-Dialoge)

Einbindung als Dll

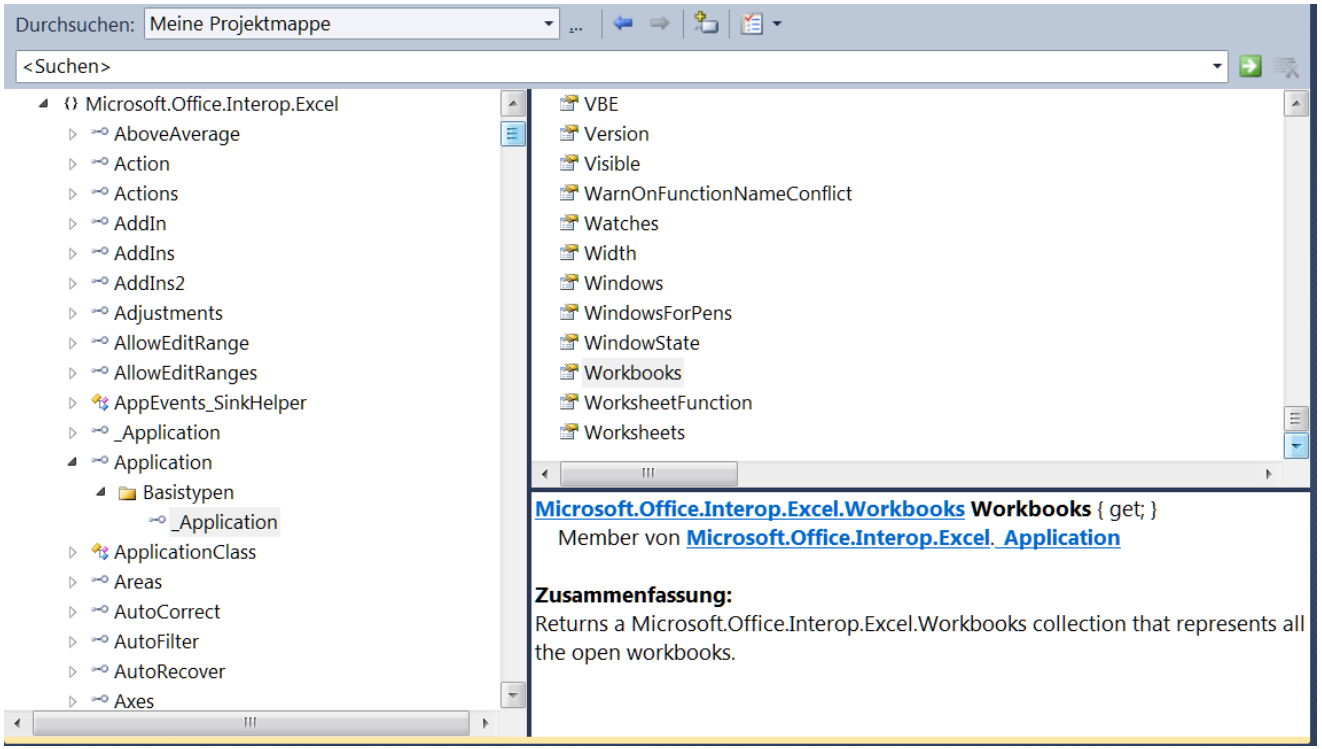
- Damit eine Dll vom Hostprozess als Erweiterung angenommen wird, muß die Dll über eine vom zu automatisierenden Programm vorgegebene Schnittstelle verfügen
  - Es gibt vordefinierte Funktionen für Lade/Entladeroutinen, die das Hostprogramm automatisch ausführt. Dabei werden z.B. die Befehle, die von der Dll bereitgestellt werden, im Programm registriert und können danach ausgeführt werden.
  - Für eine komfortable Anwendungsentwicklung stellen manche Programmhersteller Assistenten bereit, die ein Rumpfprojekt z.B. in Visual Studio erstellen und damit die Grundvoraussetzungen für die Ladefähigkeit sicher stellen (VSTO für Office-Apps)
  - Dlls können als reguläre, COM oder .NET Bibliothek erstellt werden. Mit C# können nur .NET Bibliotheken erstellt werden, COM mit entsprechenden Einschränkungen. Ein C#-Addon als Dll funktioniert folglich nur bei Programmen, die eine .NET-Schnittstelle anbieten
- Damit in der Dll auf das Programm Einfluss genommen werden kann, sind Programmbibliotheken einzubinden, mit denen Befehle zur Steuerung der Applikation bereit gestellt werden
  - Bibliotheken können in verschiedener Form angeboten werden. Obwohl in C# nur .NET-Bibliotheken erstellt werden können, ist es dennoch möglich, auch reguläre und COM Bibliotheken einzubinden
  - Um eine Applikation über eine Dll zu automatisieren, muss sowohl die „Signatur“ für das AddIn bekannt sein als auch der Befehlsumfang der Bibliothek zum Steuern der Anwendung, auch Anwendungswissen ist notwendig, da die Anwendungen selber teilweise auf den APIs aufbauen

Steuerung über eine Exe

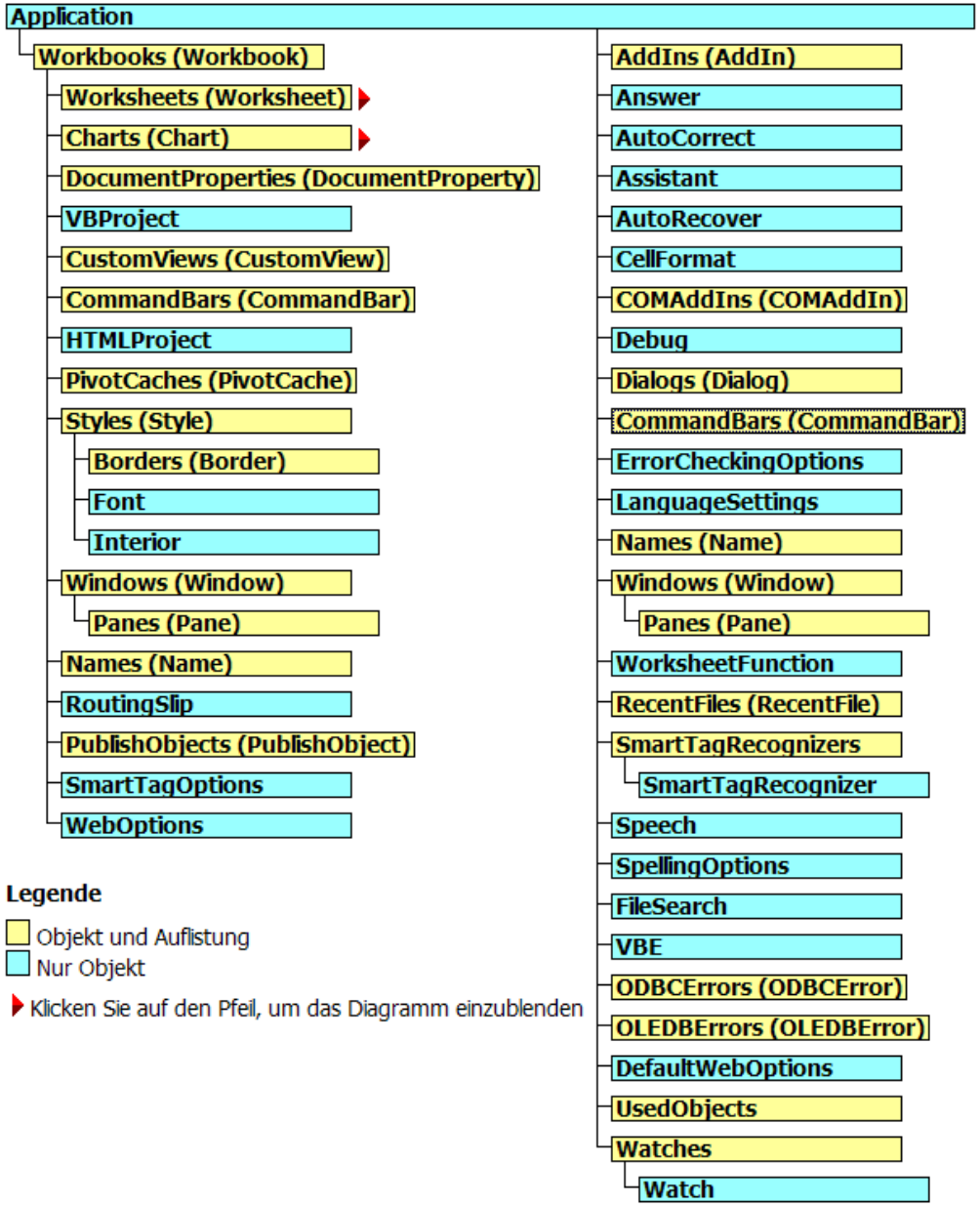
- Da die Exe nicht im Prozessraum des zu steuernden Programms läuft, interessiert hierbei nicht eine vordefinierte Struktur von Funktionen in der Steuer-Exe wie bei den Dlls
- Da die Exe selbständig läuft, kann sie keine unmittelbare Erweiterung des Befehlssatzes des zu automatisierenden Programmes bereit stellen. Durch den Aufruf der Exe evtl. mit Parametern aus dem Hostprogramm selber heraus lässt sich jedoch teilweise ein ähnlicher Effekt erzielen
- Um die Applikation über eine Exe zu steuern, sind folgende Möglichkeiten denkbar
  - Senden von Tastatur-Zeichen an die Applikation. Sofern das Programm durch Tastaturfolgen zu steuern ist, aber keine API besitzt, ist dies manchmal die einzige Möglichkeit, zu automatisieren. Nachteilig ist die hohe Störanfälligkeit
  - Über eine API kann die Exe sich mit dem laufenden Programm verbinden oder direkt eine neue Instanz des Programms starten. Das laufende Programm steht in der Exe als Objekt zur Verfügung, über das in einer objektorientierten Struktur auf die von der API bereit gestellten Befehle zugegriffen werden kann. Dies ist typischerweise bei den COM-automatisierten Programmen der Fall.
- Durch die in einem eigenen Prozess laufende Exe ist die Überwachung des eigentlichen Programms besser und sicherer zu gewährleisten

Basis der objektorientierter APIs

- Das Objektmodell der zu steuernden Applikation zeigt die Zusammenhänge zwischen den Klassen und Objekten, die für die Automatisierung bereit stehen
- Das Modell orientiert sich i.d.R. an der interaktiven Funktionalität des Programms
- In C# wird ein Objektmodell über einen Verweis eingebunden, Mit entsprechenden using-Direktiven kann die Handhabung vereinfacht werden
- Das Objektmodell in Visual Studio kann durch Doppelklick auf den entsprechenden Verweis eingesehen werden.



Das Excel-Objektmodell



Quelle: Microsoft

Das Excel-Worksheet-Objektmodell

Quelle: Microsoft

Das Excel-Objektmodell

- Wichtige Klassen/Objekte des Excel-Objektmodells sind:
  - Microsoft.Office.Interop.Excel.Application
  - Microsoft.Office.Interop.Excel.Workbook
  - Microsoft.Office.Interop.Excel.Worksheet
  - Microsoft.Office.Interop.Excel.Range
- Das Objektmodell orientiert sich sehr stark an der Programmoberfläche. Das Application-Objekt stellt die gesamte Anwendung dar, und jedes Workbook-Objekt (Arbeitsblatt = Excel-Dokument) enthält eine Auflistung von Worksheet-Objekten (Tabellen-Blätter)
- Davon ausgehend gibt es als bedeutende Abstraktion das Range-Objekt, das Zellen darstellt und für die Arbeit mit einzelnen Zellen oder Zellgruppen verwendet werden kann
- In den Office-Produkt- und vielen anderen APIs werden die Klassen/Objekte, die eine Liste von Elementen beinhalten (z.B. Application.Workbooks) immer in Pluralform mit angehängtem S bezeichnet.
- Wenn in der Liste der geöffneten Dateien auf ein konkretes Element zugegriffen wird, ist das einzelne Element folglich in einer Singular-Bezeichnung angegeben.

Quelle: Microsoft

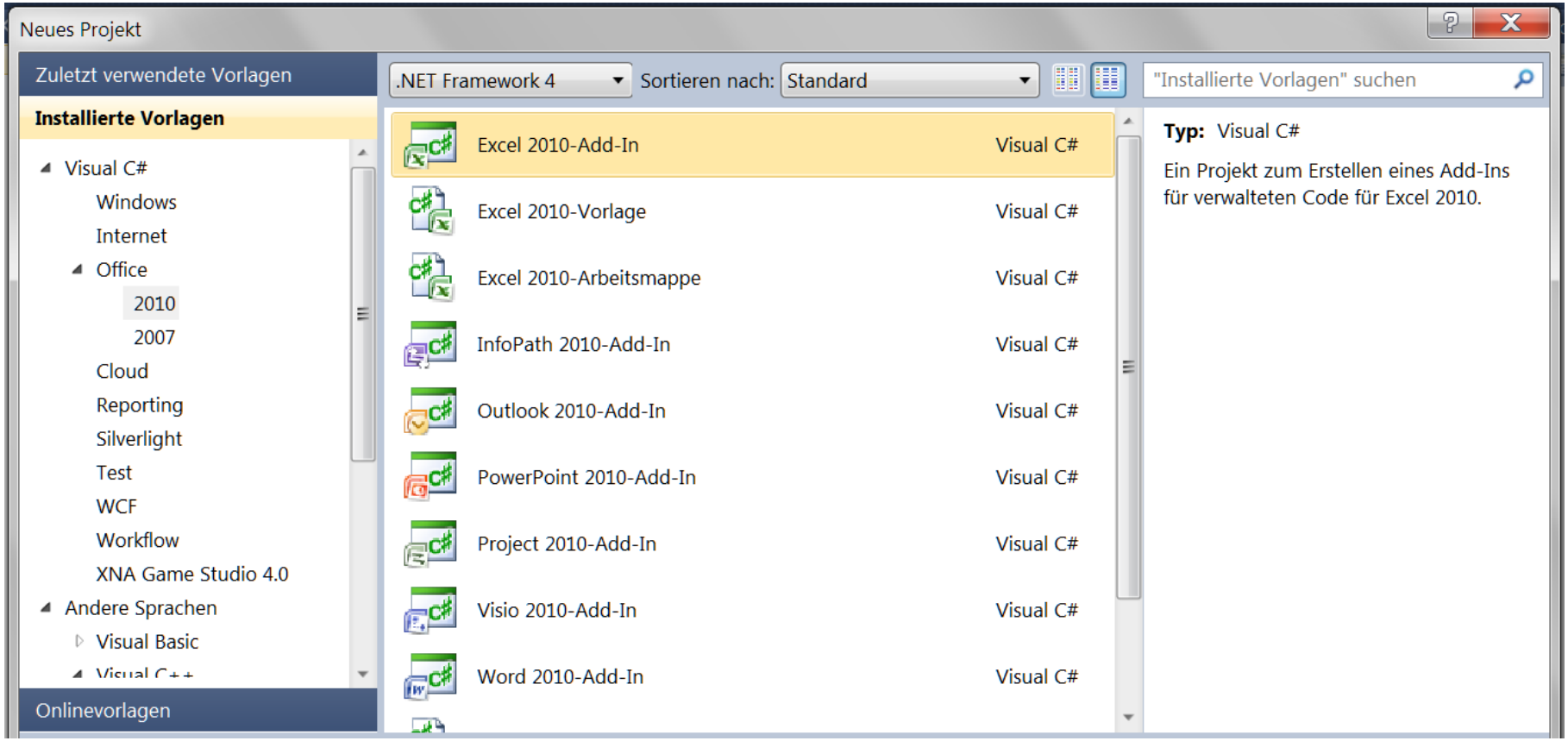
Verbindung zum Excel-Application-Objekt herstellen (Exe)

- Mit using Excel = Microsoft.Office.Interop.Excel; und den Verweisen auf die Excel-Objektbibliothek kann folgender Quelltext verwendet werden, um Excel aus einer anderen Applikation zu starten:

```
Excel.Application excel = null;
try
{
    //Mit laufendem Excel verbinden...
    excel = (Excel.Application)System.Runtime.InteropServices.Marshal.GetActiveObject("Excel.Application");
}
catch (System.Runtime.InteropServices.COMException ex)
{
    Console.WriteLine("Excel läuft nicht, Neustart!\n" + ex.Message );
    //bei Fehler Excel neu starten
    excel = new Excel.ApplicationClass();
    /*
     * Wenn für ActiveX-Anwendung keine ApplicationClass existiert, anderen Startprozeß verwenden
     */
    //Type excelType = System.Type.GetTypeFromProgID("Excel.Application");
    //excel = (Excel.Application)System.Activator.CreateInstance(excelType);
}
catch
{
    MessageBox.Show("Anderer Fehler aufgetreten!");
}
```

Erstellen von Office-Addins (Dll)

- Verwenden des Assistenten in Visual Studio (Visual Studio Tools für Office VSTO)



- Excel-Vorlage und Arbeitsmappe erzeugen Add-Ins, die an eine Vorlage oder eine Datei gebunden sind, damit nur in diesen Dokumenten zur Verfügung stehen

Neue Befehle durch ein eigenes Ribbon-Menü einbringen

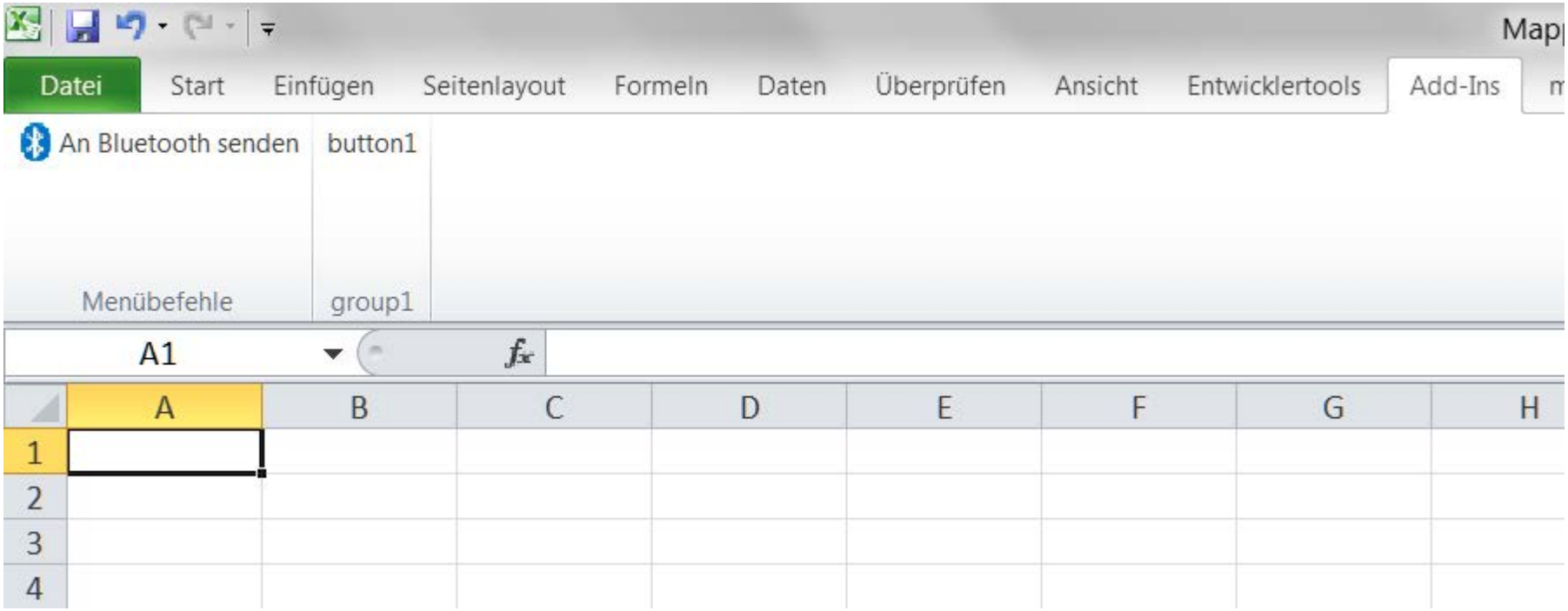
- Erstellen einer Ribbon-Klasse (Menüband), die mit dem Designer individuell gestaltet werden kann.
- Auf die Steuerelemente wird wie bei den Windows-Forms reagiert, also mit entsprechend zu wählendem Event

- Das Application-Objekt wird dabei über Globals.ThisAddIn.Application bereit gestellt.

```
private void button1_Click(object sender, RibbonControlEventArgs e)
{
    Globals.ThisAddIn.Application.ActiveCell.Value = "Hallo";
}
```

Debuggen

- Beim Debuggen wird Excel gestartet, wobei das Add-In geladen wird

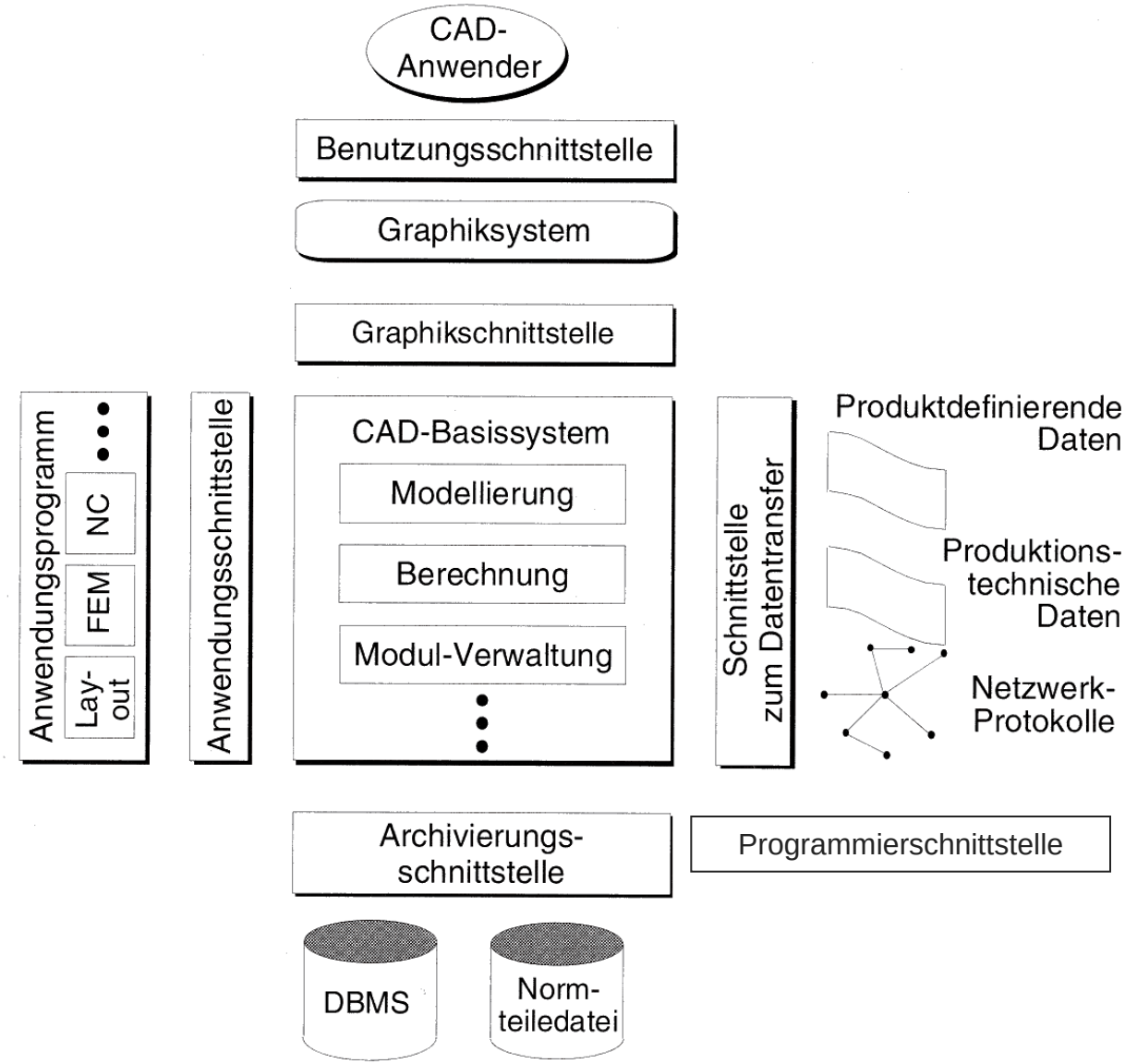


- Wird der Befehl selektiert und ist ein Breakpoint gesetzt, kann das Add-In in Visual Studio debuggt werden

Andere Office-Programme

- Durch VSTO können auch Applikations- und dokumentbezogene Add-Ins für andere Office-Anwendungen erstellt werden.
- Die grundsätzliche Vorgehensweise bleibt immer gleich
- Jedes Mal anders ist das Objektmodell der betreffenden Applikation – eine Programmierung ohne Anwendungskenntnisse wird nicht erfolgreich sein können. D.h. eine Applikation beruht in diesem Bereich sehr stark auf den Anwendungskenntnissen des oder der Programmierenden
- Neben VSTO gibt es Alternativen zur Erstellung von Add-Ins wie add-in-express
- Eine einfache Möglichkeit, Add-Ins umzusetzen, bietet die Möglichkeit mit VBA den Makro-Befehl selber zu definieren und deren Implementierung in C# vorzunehmen und über eine COM Schnittstelle von VBA aus aufrufbar zu machen. Damit kann der Quellcode besser als in VBA geschützt werden. Durch VSTO steht aber eine ebenso einfache Möglichkeit bereit.
- Ein wichtiger Punkt bei den Add-Ins stellt auch die Verteilung bzw. die Installation dar. Um dem Benutzer die manuelle Installation zu sparen, sollte ein Installationsprogramm ausgeliefert werden bzw. durch ein Skript die entsprechenden Registry-Einträge vorgenommen werden

Komponenten und Schnittstellen von CAD-Systemen

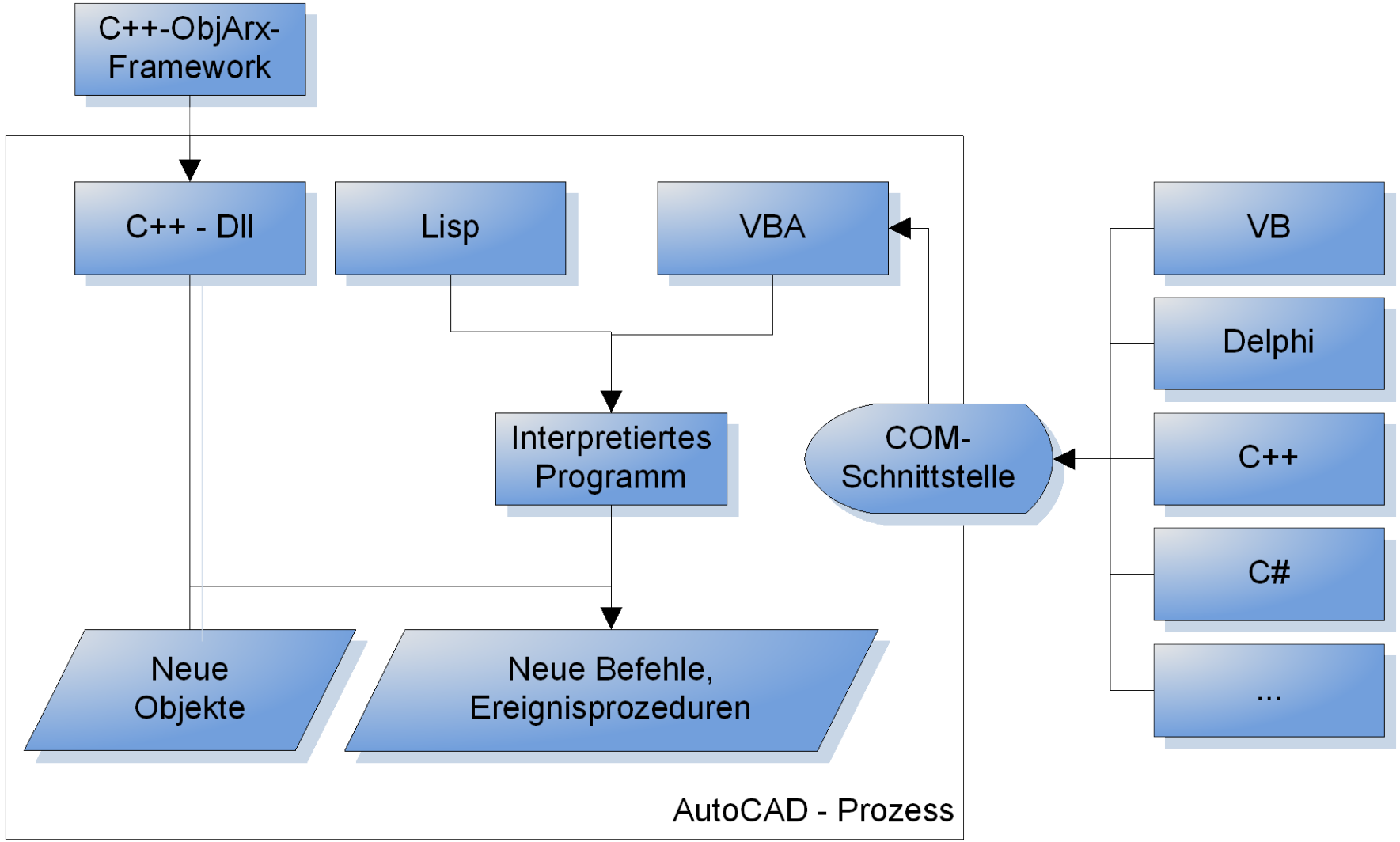


Quelle: Spur/Krause

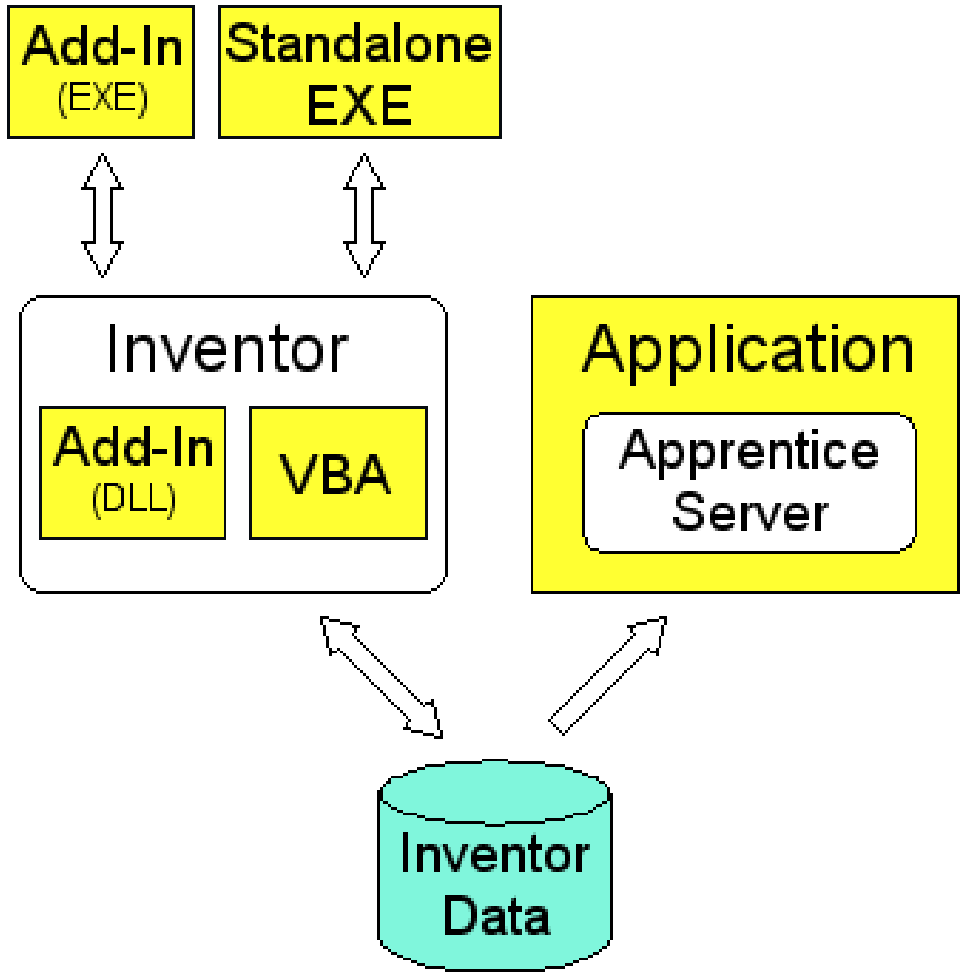
Was bietet eine CAD-API

- Zugriff auf vorhandene CAD-Funktionen
  - Automatisierung von Befehlsabfolgen (häufig als Makro bezeichnet)
  - In der Regel ist der gesamte manuell zu bedienende Befehlsumfang abgebildet
- Zugriff auf die internen Datenstrukturen des CAD-Systems, im Falle von Inventor ist das bspw.:
  - Anlegen und Manipulieren von Skizzen
  - Erstellen und Manipulieren von Feature-Operationen
  - Entstehungshistorie der Modelle, CSG-Baum, Punkte, Flächen, Kanten
  - Bearbeiten der Dokumentenlinks, …
- Erstellung neuer Dialoge und Benutzer-Oberflächen
  - Erstellung von Oberflächen im Rahmen der Möglichkeiten der Programmierumgebung
- Häufig anzutreffende Schnittstellen zur Automatisierung von CAD-Systemen
  - eigene Makro-Sprache (im Prozess des Systems laufend, i.d.R. interpretiert)
  - offene Schnittstellen zur Verwendung aus verschiedenen Programmierumgebungen
    - COM
    - .NET
    - Dll
  - Framework, hierbei stehen die CAD-eigenen Klassen als Basisklassen für eigene CAD-Objektklassen zur Verfügung, was eine extrem hohe Anpassbarkeit gewährleistet

Programmierschnittstellen von AutoCAD (ohne .NET)

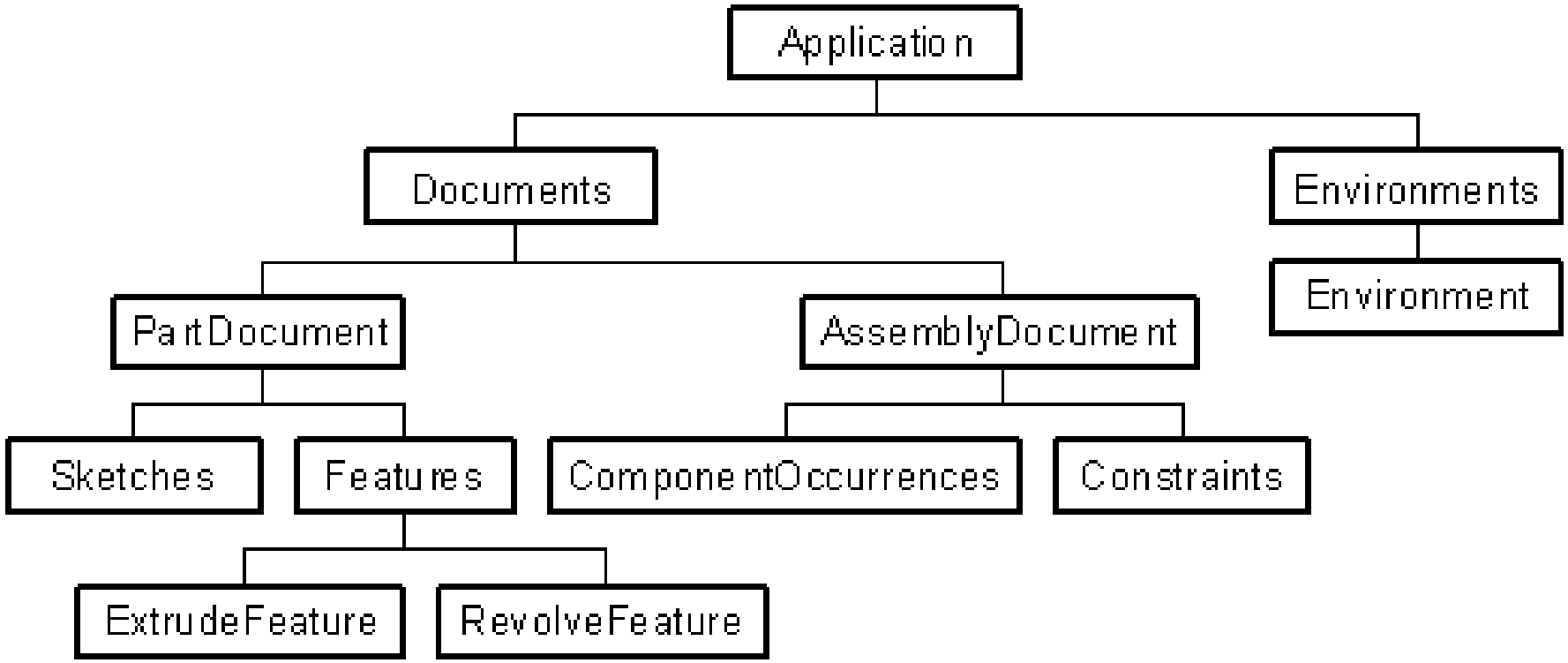


Möglichkeiten des Zugriffs auf Inventor

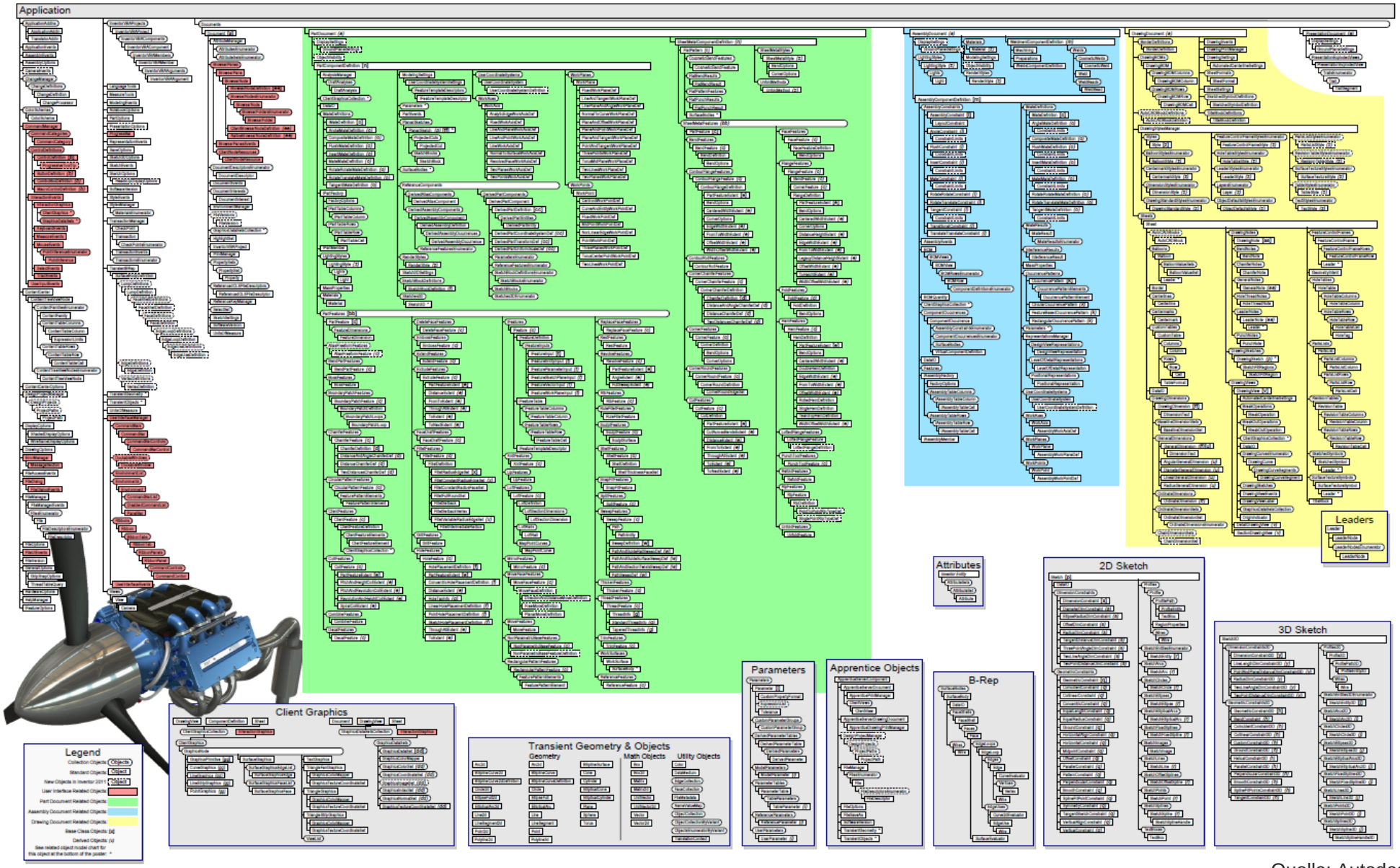


Quelle: Autodesk

Das Basis-Inventor-Objektmodell



Das Inventor-Objektmodell



Quelle: Autodesk

CAD-Programmierschnittstelle – Wozu?

- Beispiele von Anwendungsprogrammen?
- Warum werden Anwendungsprogramme erstellt?
  - Vorteile
  - Nachteile
- Wann ist die Entwicklung eines Anwendungsprogramms zweckmäßig – was sind Entscheidungskriterien und wie können diese ermittelt werden?
- Welche Randbedingungen gilt es für die Anwendungsprogrammentwicklung zu klären
- Wer entwickelt Anwendungsprogramme – Personenprofil?

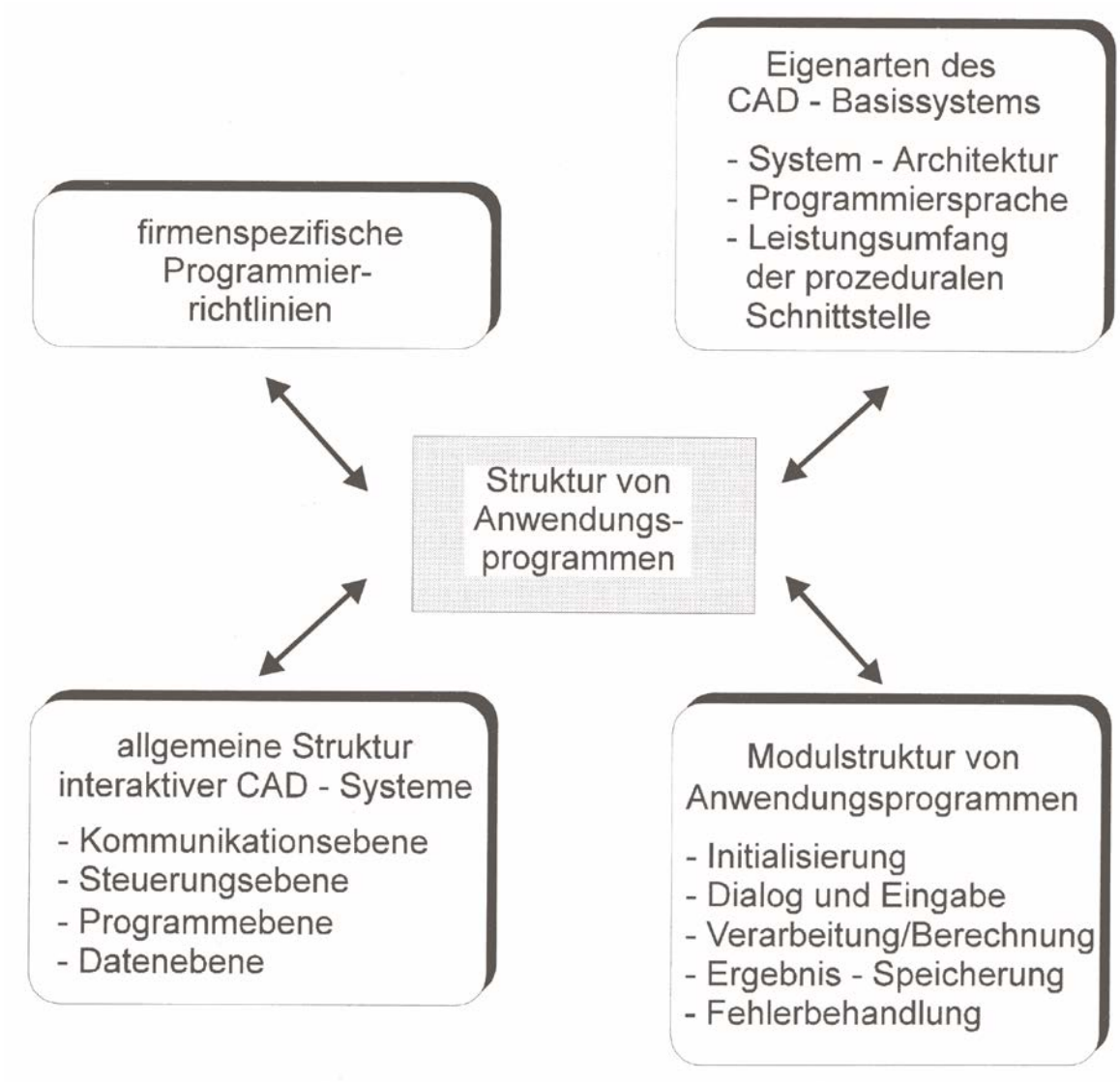
Software-Design

- Die Planung der GUI gehört wie die Planung der Algorithmen zum Softwaredesign
- Aus der Anwendersicht muss die GUI alle Anwendungsfälle in geeigneter Form unterstützen
- In der Planung sollte, bei Kenntnis der Anwendungsfälle, eine dafür geeignete GUI konzipiert werden
- Die Qualität der GUI entscheidet oft über die Akzeptanz einer Software
  - eine gut programmierte Software mit schlechter GUI wird als unprofessionell empfunden
  - eine gute GUI mit schlampiger Programmierung blendet den Anwender zunächst, Fehler in der Logik werden zunächst nicht wahrgenommen
- Die GUI muss häufig komplexe Zusammenhänge eines Algorithmus in einfacher Form präsentieren
- Für die Konzeption einer guten GUI muss der Designer über (umfangreiche) Erfahrungen aus Anwendersicht verfügen

Gestaltungsaspekte für CA(D)-Anwendungen

Quelle: Spur/Krause

strukturelle Anforderungen



Quelle: Spur/Krause

Aufgabe

- Ein Hersteller von Elektromotoren mit angeflanschtem Getriebe will einen Konfigurator erstellen lassen, mit dem die unterschiedlichen Getriebe- und Motorenvarianten interaktiv konfiguriert werden können. Dabei soll das jeweilige Ergebnis direkt im 3D-CAD-System visualisiert werden. Stellen Sie die wesentlichen Anwendungsfälle dar und konzipieren Sie das GUI.
  Erarbeiten Sie eine mögliche Strukturierung des Produktes.
  - Varianten werden gebildet über:
    - Gehäuseanschluss, Wellendurchmesser, Wellenanschluss, Auftretende Kräfte an der Abtriebswelle
    - Drehzahlen, Momente, Leistungen, Motorentyp
    - Spannung, Frequenz, Spannungsart
    - Isolation, Explosionsschutz, Temperaturbereich
    - Getriebeart, Getriebequalität
  - Welche logischen Zusammenhänge zwischen den einzelnen Komponenten wären bei der Abbildung zu berücksichtigen?
  - Wie könnte eine Benutzeroberfläche aussehen?