IN DIESEM KAPITEL

Kapitel 1
Controlling und Excel – geschaffen füreinander

Um es gleich vorwegzunehmen: Wir werden uns in diesem Kapitel durchaus kritisch zu Excel äußern, aber auch Einblicke in die Sicht von »Abhängigen« geben, denn: Ein Leben ohne Excel können wir uns als Controller nicht vorstellen. Es ist einfach der hilfreiche Assistent in zahllosen Situationen geworden, in denen es darum geht, schnelle und pragmatische Lösungen zu durchaus kniffligen Problemstellungen zu finden.

Aber wir haben in unserer mehrjährigen Excel-Praxis auch schon so viele Modelle erstellt, die man getrost in den Mülleimer werfen konnte. Nicht, weil sie ihren Dienst nicht erfüllt hatten, sondern weil die Modelle weder für andere noch für uns selbst nach einer gewissen Zeit noch brauchbar waren. Vielleicht ist es Ihnen auch schon so gegangen: Voller Begeisterung stürzen Sie sich in die Umsetzung einer tollen Idee. Sie verfeinern und verfeinern – und schließlich finden Sie sich in einem Formelgewirr wieder, das Sie selbst nicht mehr verstehen. Das muss nicht sein. In diesem Kapitel werden wir Ihnen einige Hilfsmittel mit auf den Weg geben, um genau das zu vermeiden.

Die Erfolgsgeschichte

Wenn man sich die Erfolgsgeschichte von Excel anschaut, dann kann man mit Recht sagen, dass die Controller-Community auf das Programm gewartet hat. Zwar gehen alle Tabellenkalkulationen auf VisiCalc von Dan Bricklin zurück (und das konnte schon eine ganze Menge), allerdings gelang der große Durchbruch erst mit der grafischen Oberfläche. Hier hatte Excel gegenüber Lotus 1-2-3 einfach die Nase vorn.

Wenn wir Controller fragen, welches Werkzeug sie am intensivsten nutzen, bekommen wir immer eine eindeutige Antwort: Natürlich Excel! Was macht also den Reiz dieses Werkzeugs aus?

- ✔ Die ersten Schritte sind extrem einfach. Man kann einfach loslegen und erhält recht schnell erste Ergebnisse. Excel bietet eine intuitive Benutzerführung, was es schnell nach dem Erscheinen Ende der 1980er-Jahren zum Marktführer bei den Tabellenkalkulationen machte. Man bekommt schnell Lust nach mehr. Und mit ein wenig Erfahrung kommt man sich schnell wie ein Software-Entwickler vor: Man kann seine Aufgaben automatisieren!
- ✔ Das Denken in Zahlen und Tabellen ist es, was Controlling und Excel verbindet. Controlling-Konzepte werden dann nachvollziehbar, wenn es etwas zu rechnen gibt. Und im Gegensatz zum Taschenrechner bieten sich in Excel vielfältige Möglichkeiten, die erstellten Rechenmodelle mehr als einmal zu verwenden. Und Ihre Excel-Blätter wachsen und wachsen und wachsen – leider manchmal auch etwas unkontrolliert.
- ✔ Es ist den Excel-Entwicklern hervorragend gelungen, Komplexität zu verbergen. Sie können sofort mit der Umsetzung Ihrer Ideen anfangen, ohne viel lernen zu müssen. Trotzdem ist so viel Funktionalität enthalten, dass die meisten Anwender niemals an die Grenzen der Möglichkeiten stoßen werden.

Deshalb ist Excel fester Bestandteil im Werkzeugkasten des Controllers. Daran haben auch alternative und meistens sogar günstigere Werkzeuge wie OpenOffice bislang noch nicht viel ändern können.

Für welches Problem das richtige Werkzeug?

Es wäre doch schön, für jede noch so spezielle Fragestellung eine passende Lösung zu haben, ohne sie selbst erst (zum Beispiel mit Excel) entwickeln zu müssen. Also Controlling von der Stange? Ein schöner Traum oder doch nicht ganz so weit entfernte Vision? Wir glauben, dass es in den nächsten Jahren doch eher ein Traum bleiben wird. So gibt es keine Software-Lösung für einen Führungsteilprozess, wie beispielsweise die Budgetierung, die für alle Unternehmen – oder wenigstens für eine Mehrheit – ohne Anpassungen auskommt. Wo liegt das Problem? Die unternehmensindividuellen Faktoren, wie Größe, Dynamik, Verfahren (beispielsweise Top-down, Bottom-up, Gegenstrom), sind einfach zu unterschiedlich.

Welches Programm ist also das richtige für Ihre Bedürfnisse?

Nicht immer sind die neuesten Business-Intelligence-Technologien (BI) auch die beste Lösung (insbesondere dann nicht, wenn man Wirtschaftlichkeitsaspekte heranzieht).

Lassen Sie uns an dieser Stelle kurz untersuchen, nach welchen Kriterien die Auswahl eines Programms vorgenommen werden kann. In der Betriebswirtschaft wird dazu häufig

ein sogenanntes Turbulenz-Portfolio verwendet. In diesem spielen zwei strukturbildende Merkmale eine Rolle:

- **Komplexität**, definierbar als Anzahl und Verschiedenartigkeit der für die Unternehmung relevanten Umwelttatbestände in einzelnen Umweltsegmenten. Für ein System bedeutet dies, dass eine hohe Anzahl von Objekten und Objektdimensionen sowie die zahlreichen Abhängigkeiten zwischen den Objekten zu berücksichtigen sind.
- **Dynamik**, definierbar als Häufigkeit, Geschwindigkeit, Stärke, Regelmäßigkeit und Vorhersehbarkeit von Veränderungen von für die Unternehmung relevanten Umwelttatbeständen in einzelnen Umweltsegmenten. Systeme sind also entsprechend häufig anzupassen.

Diese beiden Dimensionen sind bei der Auswahl von Software-Lösungen hilfreich.

- Eine höhere Komplexität erfordert ein methodisches Vorgehen bei der Automatisierung von Verfahren. Man nähert sich dem Ziel nicht in einem Schritt, sondern zerlegt das Problem in kleine Abschnitte. Bei einer geringen Komplexität lässt sich eine Ad-hoc-Vorgehensweise, beispielsweise mithilfe von Excel, noch am ehesten verzeihen.
- Je höher die Dynamik der Anforderungen ist, desto flexibler müssen Anwendungen sein. Hierfür eignen sich Speziallösungen. In einem eher statischen Umfeld kann man tendenziell mehr Zeit in eine individuelle Lösung investieren. Schnittstellen sind stabiler, das heißt, das Entwicklungsergebnis wird noch lange nutzbar sein.
- Bei der Betrachtung beider Ansätze ist es wichtig, zwischen der Konfigurationsflexibilität und der Laufzeitflexibilität zu unterscheiden. Individualprogrammierung glänzt beispielsweise bezüglich der Konfigurationsflexibilität, kommt jedoch schnell an seine Grenzen, wenn ein extrem dynamisches Umfeld häufige Änderungen erfordert.

In das Turbulenz-Portfolio können nun mögliche Lösungsansätze eingetragen werden (Abbildung 1.1). Es ergeben sich vier Quadranten:

- **Geringe Komplexität, geringe Dynamik.** Dies könnte beispielsweise bei einem kleinen Einzelhändler oder Handwerker der Fall sein. Benötigt man hier wirklich eine Speziallösung? Kleine Unternehmen können mit einer Abrechnungslösung mit einfachstem Reporting durchaus leben. Wenn keine Finanzbuchhaltung im Einsatz ist, ist natürlich auch Excel geeignet.
- **Geringe Komplexität, hohe Dynamik.** Ist das Geschäft einfach, aber dynamisch, reicht es häufig aus, Ad-hoc-Modelle zu verwenden, denn die Neuerstellung oder Anpassung ist in der Regel nicht sehr aufwendig.
- **Hohe Komplexität, geringe Dynamik.** Erhöht sich die Komplexität bei geringer Dynamik beispielsweise durch eine höhere Anzahl an Mitarbeitern, Produkten oder Kunden, wird die Steuerung komplizierter. Man behilft sich mit Berichtsergänzungen. In einem moderat dynamischen Umfeld besteht die Notwendigkeit, den Informationsprozess in mehrere Schritte zu unterteilen. Reporting und ein Data Warehouse scheinen hier die geeignete Lösung zu sein.

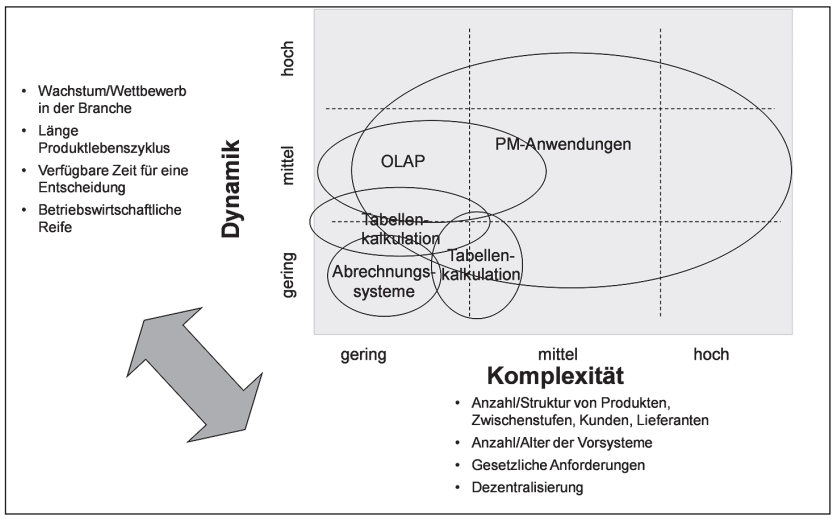


Abbildung 1.1: Unternehmenssteuerung im Turbulenz-Portfolio

- ✔ **Hohe Komplexität, hohe Dynamik.** Die schwierigste Situation ist schließlich bei einer hohen Dynamik und einer hohen Komplexität anzutreffen. Das trifft mittlerweile auf einen Großteil der Unternehmungen zu. Es ist der Bereich, der in der Regel durch spezielle Corporate-Performance-Management-Anwendungen adressiert wird. Auf die Arbeit mit Excel wird hier allerdings nicht verzichtet. Vielmehr findet eine Aufteilung statt: Kollaboratives Arbeiten und Massenkalkulationen werden durch Spezialwerkzeuge unterstützt, während Nebenrechnungen mit Excel durchgeführt werden.

Was ist die Konsequenz? Sie müssen die individuelle Situation Ihres Unternehmens und auch die spezielle Problemstellung, die Sie gerade bearbeiten, genau einschätzen, bevor Sie die Werkzeugauswahl angehen. Vielleicht benötigen Sie ja für die Planung ein Spezialwerkzeug, während für die Analyse Excel sehr gut geeignet ist.

Excel sollten Sie übrigens auch im komplex-dynamischen Umfeld nicht generell ablehnen, was Verfechter von BI-Werkzeugen gerne machen. Denn mittlerweile sind Lösungen im Einsatz, die die in Bezug auf Dynamik und Komplexität kritischen Aspekte aus der Tabellenkalkulation in Spezialanwendungen auslagern. Insofern kann das Beste aus verschiedenen Konzepten miteinander kombiniert werden.

Lassen Sie uns noch einen Aspekt besonders herausstellen, an dem sich die Geister wohl am häufigsten scheiden: Eines der Haupteinsatzfelder von Excel im Controlling ist die Planung. Spitzenreiter bei der Unterstützung von Planungssystemen ist eindeutig Excel, und die meisten Planungsanwendungen werden auf Basis von Excel aufgebaut. Selbst Großunternehmen nutzen es für den Planungsprozess. Natürlich ist der Einsatz von Tabellenkalkulationen in der Budgetierung nicht unproblematisch.

Aber die Vorteile liegen auf der Hand: Die Tabellenkalkulation ist schnell einsetzbar, da geringe beziehungsweise keine Anschaffungskosten entstehen (sie ist meist schon unternehmensweit vorhanden) und die meisten Benutzer mit dem System umgehen können. Sie bietet eine hohe Flexibilität. Allerdings ist diese Flexibilität trügerisch.

Die englische Tochter der KPMG (jetzt Origin) hat bei Kunden vor einigen Jahren die mit Tabellenkalkulationen erstellten Anwendungen untersucht. Die Ergebnisse sind wenig überraschend:

- ✔ 95 Prozent der Anwendungen enthielten wesentliche Fehler.
- ✔ 59 Prozent der Anwendungen hatten ein mangelhaftes Design.
- ✔ 92 Prozent der Anwendungen hatten wesentliche Fehler in der Steuerberechnung.
- ✔ 75 Prozent enthielten wesentliche Fehler in der Controlling-Methodik.
- ✔ 78 Prozent der Abteilungen verfügten über keine formale Qualitätssicherung.

Das Erstaunliche an dieser Studie war jedoch, dass 81 Prozent der Anwender davon ausgingen, dass sie mit ihren tabellenkalkulationsbasierten Modellen einen wesentlichen Vorteil gegenüber Wettbewerbern erlangen würden – sicherlich in Unkenntnis der vielfältigen Fehlerquellen. Fazit: Sie müssen beim Einsatz von Excel durchaus aufpassen.

Die Alternative, um eine Tabellenkalkulation effizient zu nutzen, besteht in der Einbindung dieses Werkzeugs in spezialisierte Planungssysteme. Zahlreiche Anbieter derartiger Systeme nutzen Excel mittlerweile als flexibles Eingabe- und Analysewerkzeug in Kombination mit Regel- und Speicherungskomponenten ihrer Planungsanwendung. Die Tabellenkalkulation ist also vollständig in die Planungsumgebung integriert und stellt dem Planer eine vertraute Umgebung bereit.

Die Grenzen von Excel

Excel kann viel, aber nicht alles. Deshalb ist es wichtig zu verstehen, wo die Grenzen des Programms liegen.

»Grenzen«, das hört sich zunächst nach festen Restriktionen an, wie »maximal 64.000 Zeilen«. Doch diese Zeiten sind seit der Version 2007 vorbei. Es sind eher konzeptionelle Fragen, bei denen Excel an seine Grenzen stößt, wenn es also für etwas verwendet wird, für das das Programm grundsätzlich gar nicht geeignet ist. Überschreiten Sie diese Grenze, dann werden Sie wahrscheinlich ein 4K-Problem bekommen. Die vier Ks stehen für:

- ✔ **Komplexität:** Ein Controller hat uns einmal ganz stolz gezeigt, wie er seine neue Planungslösung mit Verknüpfungen über 20 Dateien aufgebaut hatte. Als wir ihn ein Jahr später trafen, war von dieser Lösung keine Rede mehr. Wir sprachen ihn darauf an. »Die hat leider nie richtig funktioniert«, war seine Antwort. Mehr wollte er dazu nicht sagen. Sie kennen das bestimmt auch: Alles fängt mit einer kleinen, einfachen Lösung an, dann wird hier etwas erweitert, dort etwas hinzugefügt, und nach einer Weile traut man sich kaum noch, Änderungen vorzunehmen.
- ✔ **Konsistenz:** Wenn falsche Zahlen auftauchen, breitet sich schnell Panik aus. Die Ursache können fehlerhafte Funktionen sein. Schlimmer sind jedoch Inkonsistenzen, die durch nachträgliche Anpassungen auftreten: Die Anpassung einer Kennzahlberechnung wird beispielsweise nur auf elf statt auf zwölf Monate kopiert, bei der Neuanlage einer

Kostenstelle wird vergessen, auch die Umlagen entsprechend anzupassen, und, und, und ... Auch die Makro-Automatisierung birgt Risiken: Mit der Aufzeichnungsmethode scheint die Erstellung von Makros eine einfache Sache zu sein. Wirklich? Es werden nur die Aktivitäten aufgezeichnet! Zum Programmieren gehört allerdings wesentlich mehr. Das Makro eines Kunden hatte über 500 Zeilen Aufzeichnung in einem Programm, die natürlich nicht hintereinander aufgenommen, sondern zusammenkopiert worden waren. Die Fehlersuche war extrem kompliziert. Verschiedene Makro-Strecken enthielten immer wieder den gleichen Code, sodass Änderungen äußerst aufwendig wurden.

- ✔ **Kapazität:** Die Excel-Datei eines Kunden hatte einen Umfang von mehr als 100 MB. Hier wurde Excel als Datenbank missbraucht. Kein Wunder, dass sich der Controller über Performance-Probleme beklagte. Für die Verarbeitung von großen Datenmengen ist Excel nie konzipiert worden. Richtig glücklich werden Sie nur bei einfachen Datenstrukturen und wenigen Datensätzen.

- ✔ Und schließlich die **Kommunikation:** Wenn viele Mitarbeiter an einer gemeinsamen Lösung arbeiten, dann möchten sie gegebenenfalls auch gleichzeitig auf die Modelle zugreifen können beziehungsweise nicht immer aktualisierte Ergebnisse per E-Mail versenden. Auch soll nicht immer jeder alles sehen. Typisch für eine intensive Kommunikation sind Planungsprozesse, in die bereits in mittelgroßen Unternehmen zahlreiche Personen involviert sind. In einem derartigen Prozess arbeiten mehrere Mitarbeiter an denselben Planungsmasken. Es müssen Ergebnisse konsolidiert werden. Wenn hier Excel zum Einsatz kommt, müssen viele Tätigkeiten manuell durchgeführt werden. Die Fehleranfälligkeit steigt damit.

Das sollte Sie aber nicht entmutigen, mit Excel zu arbeiten. Wenn Sie einige Bedenken beherzigen, dann wird Excel zu einem fantastischen Universalwerkzeug, also quasi zum Schweizer Offiziersmesser für den Controller!

Die »sieben Todsünden« und was man dagegen machen kann

Lassen Sie uns die Kritik an der Arbeitsweise mit Excel noch etwas vertiefen. Woran sollten Sie denken, bevor Sie Ihr nächstes Excel-Projekt starten? Wir haben sieben »Todsünden« bei der Arbeit mit Excel formuliert, mit deren Hilfe Sie unserer Meinung nach zwischen guten und schlechten Excel-Sheets unterscheiden können.

Ein Spreadsheet auf die Schnelle

Dass Sie Excel als Ad-hoc-Werkzeug gut benutzen können, ist eine Stärke dieses Programms. Man kann einfach anfangen und schauen, was dabei herauskommt. Wenn Sie wirklich absehen können, dass Sie eine »Wegwerf«-Datei erstellen, dann ist das in Ordnung. Für alle anderen Arbeiten sollten Sie sich zu Beginn folgende Fragen stellen:

- ✔ Benötige ich das Dokument später noch einmal?

- ✔ Möchte ich die gefundene Lösung weiterentwickeln?
- ✔ Werden auch andere damit arbeiten?

Wenn Sie diese Fragen mit Ja beantworten, schlagen wir vor, dass Sie nicht gleich mit Excel loslegen, sondern sich zunächst auf einem Blatt Papier Gedanken darüber machen, welche Funktionen Sie brauchen und ob Word, Access oder auch eine Kombination der Programme für Ihre Aufgabe nicht besser geeignet sein könnte. Erst wenn Sie eine klare Vorstellung von Ihrer Aufgabe haben, sollten Sie mit der Umsetzung beginnen.

Sie sollten sich außerdem überlegen, ob Sie Ihre Aufgabe nicht in leicht verdauliche und nach Möglichkeit unabhängige Teillösungen einteilen können. Teilen Sie beispielsweise eine Unternehmensplanung in einzelne Blätter auf. Achten Sie darauf, dass Sie pro Blatt einen Eingabe- und einen Ausgabebereich definieren. Wenn es zu komplex wird, definieren Sie für jedes Arbeitsblatt einen Bereich, der Daten empfängt, und einen anderen Datenbereich, aus dem sich andere Blätter (beziehungsweise Module) bedienen.

Und schließlich: Dokumentation ist im Controlling das halbe Leben. Alles, was nicht selbsterklärend ist, gehört dokumentiert. Damit Sie aber nicht zu viel dokumentieren müssen, nutzen Sie die eingebauten Möglichkeiten von Excel. Wenn Sie häufig mit Namen arbeiten, erfüllen Sie damit schon einen Teil der Dokumentationsaufgabe, denn die Formeln sind mit einem Mal selbsterklärend. Ein kleines Beispiel zeigt Abbildung 1.2:

	A	B	C	
1				
2		Umsatz	100	
3		Kosten	80	
4		Deckungsbeitrag	20	
5				

Abbildung 1.2: Selbsterklärende Datenblätter übernehmen Ihre Dokumentationsaufgabe.

Natürlich können Sie den Deckungsbeitrag in Zelle C4 auch mit Zellreferenzen beschreiben:

```
C4:=C2-C3
```

Aber betrachten Sie jetzt nur die Formel: Da könnte alles möglich sein, zum Beispiel »Aktiva – Passiva«, »Anfangsbestand – Abgang« und so weiter. Ohne die Texte in Spalte B fällt die Interpretation der Daten schwer. Zudem handelt es sich hier um ein sehr einfaches Beispiel, denn in der Regel stehen die Bestandteile einer Formel nicht untereinander, sondern sind zum Teil über verschiedene Arbeitsblätter verteilt. Da hilft Ihnen (vor allem aber Ihren Kollegen) die Arbeit mit Namen:

1. Sie können die Namen einzeln angeben oder über Formeln | Aus Auswahl erstellen automatisieren. Markieren Sie hierzu die Zellen und wählen Sie im Register Formeln | Aus Auswahl erstellen aus (Abbildung 1.3).
2. Excel erzeugt dann Namen für jede Zelle. Zunächst werden Sie gefragt, wie die Namen zu den Zellen stehen sollen (Abbildung 1.4).

Abbildung 1.3: Anlage von Namen

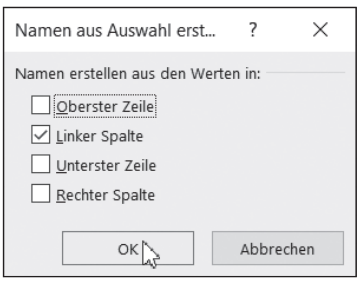


Abbildung 1.4: Automatische Erstellung von Namen

Und Excel hilft Ihnen sogar bei der Erstellung der Formel (Abbildung 1.5). Geben Sie nur die ersten Buchstaben ein, und Excel kann es kaum erwarten, Ihnen den ganzen Namen vorzuschlagen.

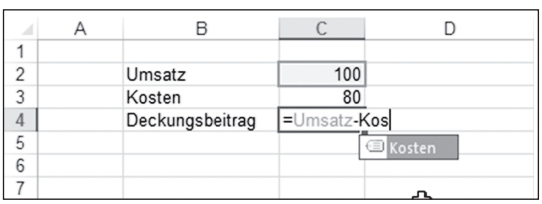


Abbildung 1.5: Eingabeunterstützung

Und schon liegt uns eine selbsterklärende Formel vor:

```
C4:=Umsatz-Kosten
```

Damit ist nicht viel Aufwand verbunden, die Verständlichkeit wird aber deutlich verbessert.

Überladen von Excel

»Ich weiß schon, was ich tue!« Das möchten wir gar nicht infrage stellen. Aber können Sie sich noch daran erinnern, warum Sie dieses oder jenes vor einem halben Jahr erstellt haben? Wir wissen es leider meistens nicht mehr. Und dann folgt die mühselige Arbeit, sich durch Arbeitsblätter zu quälen.

In der Programmierung gibt es einen Begriff, der einen schlechten Programmierstil kennzeichnet: »Spaghetti-Code«. Ein Programmfluss ist durch Sprünge und Verzweigungen gekennzeichnet. Nun, »Spaghetti-Excel« gibt es auch:

```
A12/A25*(SUMME('Tabelle 1'!F17:F24) + SUMME('Tabelle 2'!.....)
```

Wir möchten, dass Ihre Arbeitsblätter ganz anders aussehen (dieses Buch wird Ihnen dabei helfen). Der Schlüssel zum Erfolg ist, die Aufgaben in möglichst kleine Abschnitte zu unterteilen. Ab einer gewissen Komplexität schleichen sich Fähler (huch!) ein. Arbeiten Sie deshalb mit Zwischenergebnissen, sodass die Schritte nachvollziehbar sind.

Nutzen Sie Kommentare! Dies können Zellen neben Ihren Berechnungen oder Zellkommentare sein. Zellkommentare können Sie über das Kontextmenü erstellen.

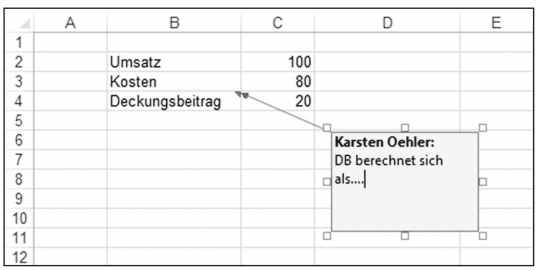


Abbildung 1.6: Zellkommentar

Für Kommentierungen in Formeln gibt es übrigens einen einfachen Trick: Hängen Sie einfach die Funktion `N("hier steht der Kommentar")` an. `N` liefert bei einem Textfeld den Wert null. Im Gegensatz zum Excel-Kommentar können Sie so auch einzelne Formelbereiche direkt kommentieren.

```
=Umsatzerlöse+N("Aus der GuV")-
Forderungen_aus_Lieferungen_und_Leistungen+
N("Hier greifen Sie auf die Bilanz zu.")+
erhaltene_Anzahlungen_auf_Bestellungen+N("Und hier auch")
```

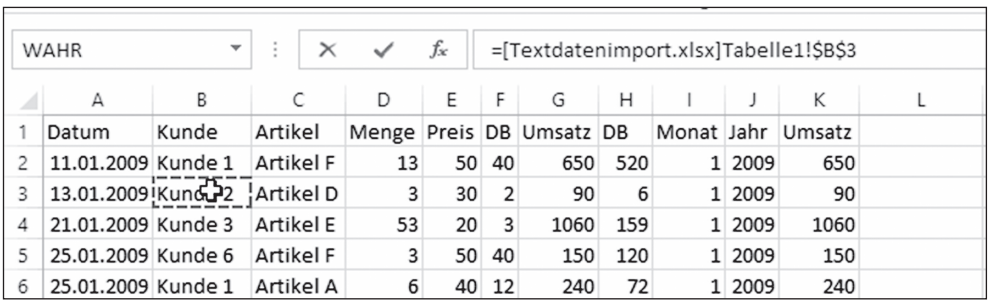
Wenn Referenzen ins Leere führen ...

Genauso wie Sie sich fragen sollten, welche Aufgaben ein Excel-Sheet tatsächlich zu lösen hat, müssen Sie hinterfragen, ob ein verteiltes Arbeiten mit Excel sinnvoll ist. Antworten auf diese Frage könnten sein:

- ✔ Eine einzelne Datei wird zu groß. (Lade- und Berechnungszeiten sind typische Argumente dafür.)
- ✔ Die Datei berührt unterschiedliche Aufgabenbereiche, die von verschiedenen Mitarbeitern betreut werden.
- ✔ Die Verteilung über mehrere Dateien wird grundsätzlich als Modularisierungskonzept verwendet.

Um auf andere Dateien zu verweisen, nutzen Sie Referenzen. Excel setzt automatisch eine Referenz, wenn Sie auf eine Zelle außerhalb der aktuellen Arbeitsmappe verweisen.

Öffnen Sie dazu mindestens zwei Arbeitsmappen und geben Sie in der ersten in einer Zelle ein »=« ein (nicht mit ↵ bestätigen). Mit Strg+F6 können Sie jetzt Zellen aus anderen Arbeitsblättern auswählen (Abbildung 1.7).



	A	B	C	D	E	F	G	H	I	J	K	L
1	Datum	Kunde	Artikel	Menge	Preis	DB	Umsatz	DB	Monat	Jahr	Umsatz	
2	11.01.2009	Kunde 1	Artikel F	13	50	40	650	520	1	2009	650	
3	13.01.2009	Kunde 2	Artikel D	3	30	2	90	6	1	2009	90	
4	21.01.2009	Kunde 3	Artikel E	53	20	3	1060	159	1	2009	1060	
5	25.01.2009	Kunde 6	Artikel F	3	50	40	150	120	1	2009	150	
6	25.01.2009	Kunde 1	Artikel A	6	40	12	240	72	1	2009	240	

Abbildung 1.7: Eine externe Verknüpfung erzeugen

Eine Referenz wird wie folgt gespeichert:

```
=Pfad [Datei]Blatt!Zellreferenz
```

Für dieses Buch könnte eine Referenz beispielsweise folgendermaßen aussehen:

```
=C:\Excelbuch\[Ref1.xlsx]Tabelle 1!$A$1
```

Referenzen auf andere Dateien (externe Referenzen) machen aber nicht nur Freude. Folgende typische Fehler können auftreten:

- ✔ Die Quelldatei wurde gelöscht, geändert oder verschoben.
- ✔ Die Referenz innerhalb der Quelldatei ist nicht mehr gültig.
- ✔ Die Aktualisierung macht Probleme. Das tritt beispielsweise dann auf, wenn »transitive« Referenzen verwendet werden. Mit anderen Worten: Eine Zelle aus Datei A bezieht sich auf eine Zelle in Datei B, die wiederum direkt oder indirekt auf eine Zelle in einer dritten Datei C referenziert. Ändert man die dritte Datei C und öffnet Datei A, ohne dass zuvor Datei B aktualisiert wurde, enthält A veraltete Daten. Dieser Fall tritt häufig in hoch verteilten Planungsprozessen auf.

Excel ist aber höflich und fragt beim Öffnen einer Datei, ob die externen Referenzen aktualisiert werden sollen.

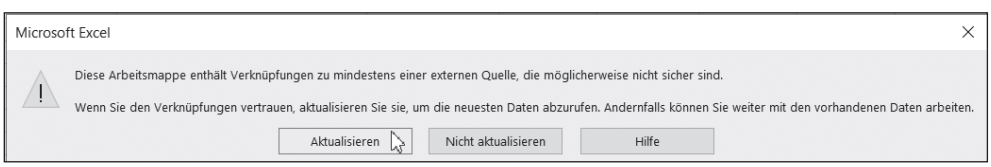


Abbildung 1.8: Entdeckung externer Verknüpfungen

Allerdings wird nicht geprüft, ob die Quelle ihrerseits auch wieder Referenzen enthält. Das kann dazu führen, dass Sie zwar in Bezug auf die Quelle aktualisierte Daten erhalten, die aber trotzdem falsch sind. Was ist hier zu tun?

Beschränken Sie zunächst die Anzahl der Verknüpfungen auf ein Mindestmaß. Trotzdem kann es passieren, dass Sie eine Doppelreferenz nicht bemerken.

Verwenden Sie für die Übergaben Namen, zum Beispiel:

```
='C:\Excelbuch\Ref1.xlsx'!Referenz
```

Wenn das alles nicht hilft, sollte zumindest ein Prozess vereinbart werden, der verbindlich festlegt, wann Daten aktualisiert werden.

Excel als Datenbank

Excel kann sicherlich eine gute Datenbank abgeben. Allerdings werden alle Daten im Arbeitsspeicher gehalten. Und der ist immer lokal auf dem PC oder Laptop. Auch wenn die Leistungsfähigkeit der Rechner immer weiter steigt, wird Excel trotzdem irgendwann langsamer.

Bevor Sie also Excel auch als Datenbank benutzen, stellen Sie sich die folgenden Fragen:

- ✔ Muss ich alles in einer Datei abspeichern? Überlegen Sie sich ein Modulkonzept, wobei die Maßgabe gilt, die Referenzen auf ein Mindestmaß zu beschränken.
- ✔ Der Zugriff auf eine Datenbank stellt eine Alternative zu Excel dar. Laden Sie nur die Datenelemente, die Sie benötigen. Die Daten der letzten drei Jahre sind vielleicht für Vergleiche sinnvoll, trotzdem brauchen Sie nicht jeden einzelnen Beleg, sondern nur die Kontensalden.

Die Nutzung von Excel als Datenbank ist eigentlich recht einfach. In Kapitel 3 wird detailliert erklärt, wie Sie auf Datenbanken zugreifen können.

Mit den Analysis Services besteht die Möglichkeit, auf größere Datenbestände zuzugreifen, ohne alle Daten in den Arbeitsspeicher zu laden. Das wird noch genauer in Kapitel 6 erläutert.

Excel als Kollaborationswerkzeug

Excel wurde ursprünglich als Einzellösung konzipiert: Eine Person arbeitet an einer Aufgabenstellung. Der Alltag eines Controllers sieht heute anders aus. Es gibt Prozesse, bei denen ein Datenaustausch zwingend ist. Planung und Budgetierung sind derartige Prozesse. Und ein typischer Planungsprozess verlangt einen klar definierten Workflow:

- ✔ Planungsdaten werden zentral vorbereitet und dann in »Scheiben« an viele Planungsverantwortliche versendet.
- ✔ Es wird geplant und zum Teil noch weiter verteilt, also »kaskadierend« geplant.

- ✔ Die Planstände werden wieder eingesammelt.
- ✔ Dieser Prozess kann in mehreren Iterationsstufen ablaufen, zum Beispiel wenn es zentrale Planungsänderungen gibt oder die zentrale Planungsinstanz mit den Ergebnissen der verteilten Planung nicht zufrieden ist.

Vergleichbare Prozesse gibt es für

- ✔ die Konsolidierung (über sogenannte Reporting-Packages),
- ✔ das Risikomanagement,
- ✔ das Forecasting und
- ✔ das Berichtswesen (Kommentierung).
- ✔ Und überall ist Excel im Spiel ...

Natürlich wird Excel immer leistungsfähiger, und in späteren Kapiteln werden wir noch darstellen, wie Sie Excel für alle möglichen Controlling-Aufgaben einsetzen können. Aber ab einem gewissen Umfang empfiehlt es sich, Spezialwerkzeuge einzusetzen, die man mit Excel kombinieren kann.

Ein Problem bleibt jedoch auf jeden Fall bestehen: Die verteilten Pläne sind häufig voneinander abhängig. Was beispielsweise der Vertrieb plant, beeinflusst die Planung der Produktion und so weiter. Und das Versenden von Excel-Blättern macht diesen Prozess so richtig schwerfällig.

Excel bleibt jedoch das ideale Werkzeug, solange die Planungsstruktur einfach ist und mit wenigen Planern auskommt.

Kopieren ohne System

Das Kopieren von Funktionen geht mit Excel sehr einfach, ist aber auch eine der häufigsten Fehlerquellen. Deshalb ist hier methodisches Arbeiten angebracht.

Kopieren ist nicht gleich Kopieren, weshalb wir empfehlen, sich vor dem Kopieren Gedanken über die Art und Weise zu machen, wie Excel mit Zellverweisen umgehen soll. Standard ist die komplett relative Adressierung: Wenn Sie eine Formel von der Zelle C1 (beispielsweise `=A1*0,3`, vielleicht eine Personalnebenkostenberechnung) auf D2 kopieren, wird aus der Formel `=B2*0,3`. Aber das ist nicht unbedingt gewünscht, denn der Referenzwert steht in A2.

Ein paar Beispiele zur Funktionsweise des Kopierens sehen Sie in Abbildung 1.9.

Der Umsatzanteil in Bezug auf das Jahr ergibt sich aus dem Monatsumsatz dividiert durch den Jahresumsatz.

Der erste Versuch für die Zelle F3 wäre

```
F3:=C3/C15
```

Da Sie aber die Zelle C3 auf die Zellen C4 bis C14 kopieren wollen, muss die Zelle C15 für das Kopieren durch die absolute Adressierung fixiert werden. Markieren Sie in der Formel den Formelteil C15 und drücken Sie so oft F4, bis die richtige Adressierung gewählt wurde.

```
F3:=C3/$C$15
```

	A	B	C	D	E	F	G	H	I
1									
2			Umsatz	var. Kosten	fixe Kosten	Umsatzanteil	var. Kostena	relativer DB I	relativer DB II
3		Jan	100	83	25	7,44%	6,93%	17,00%	-8,00%
4		Feb	118	118	20	8,78%	9,85%	0,00%	-16,95%
5		Mrz	123	103	27	9,15%	8,60%	16,26%	-5,69%
6		Apr	100	87	25	7,44%	7,26%	13,00%	-12,00%
7		Mai	111	87	24	8,26%	7,26%	21,62%	0,00%
8		Jun	107	86	26	7,96%	7,18%	19,63%	-4,67%
9		Jul	126	98	23	9,38%	8,18%	22,22%	3,97%
10		Aug	105	106	23	7,81%	8,85%	-0,95%	-22,86%
11		Sep	105	113	20	7,81%	9,43%	-7,62%	-26,67%
12		Okt	116	110	24	8,63%	9,18%	5,17%	-15,52%
13		Nov	116	94	21	8,63%	7,85%	18,97%	0,86%
14		Dez	117	113	23	8,71%	9,43%	3,42%	-16,24%
15		Jahr	1344	1198	281				
16									

Abbildung 1.9: Geschicktes Kopieren

Wenn Sie aber planen, auch den Monatsanteil der variablen und fixen Kosten zu ermitteln, würde ich Folgendes vorschlagen:

```
F3:C3/C$15
```

Und schon können Sie mit einem Kopiervorgang kopieren:

Gehen Sie auf die Zelle F3 und drücken Sie Strg+C. Markieren Sie anschließend die Zellen F3 bis F15 und drücken Sie Strg+V.

Sie können auch mit gemischt adressierten Bereichen arbeiten. Ein kurzes Beispiel: Der relative Deckungsbeitrag I (DB I) berechnet sich aus:

```
H3:=($C3-D$3)/$C3
```

Der relative DB II lässt sich durch Kopieren nicht so einfach erzeugen. An dieser Stelle hilft Ihnen die Arbeit mit gemischt adressierten Bereichen. Ändern Sie H3 zu:

```
H3:=($C3-SUMME($D3:D3))/$C3
```

Kopieren Sie nun auf I3. Der Wert in Zelle I3 ergibt sich dann aus

```
I3:=($C3-SUMME($D3:E3))/$C3
```

In Tabelle 1.1 finden Sie die Adressierungsarten, die Excel kennt.

Das Arbeiten mit selbsterklärenden Namen haben wir bereits vorgestellt. Wenn sich Namen auf einzelne Zellen beziehen, verändern sich diese beim Kopieren nicht. Allerdings können sich Namen auch auf Zellbereiche beziehen. Dann können Sie auch relative Namensbezüge

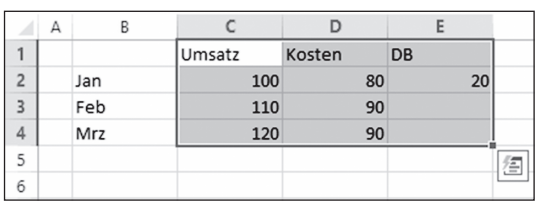
kopieren. Die Namen lassen sich übrigens auch für Bereiche automatisch erstellen. Hier ein kleines Beispiel (Abbildung 1.10) für eine Deckungsbeitragsrechnung über drei Monate:

Adressierungsart	Wirkung	Shortcut
SpalteZeile(relativer Bezug)	Beim waagerechten Kopieren nach rechts oder links wird der Spaltenbuchstabe angepasst.Beim senkrechten Kopieren nach oben oder unten wird die Zeilennummer angepasst.	Umschalten mit F4
$SpalteZeile(gemischter Bezug)	Beim waagerechten Kopieren wird der Spaltenbuchstabe angepasst.Beim senkrechten Kopieren bleibt die Zeilennummer unverändert.	
Spalte$Zeile(gemischter Bezug)	Beim waagrechten Kopieren bleibt der Spaltenbuchstabe unverändert.Beim senkrechten Kopieren wird die Zeilennummer angepasst.	
$Spalte$Zeile(absoluter Bezug)	Sowohl beim senkrechten als auch beim waagrechten Kopieren bleiben Spaltenbuchstabe und Zeilennummer unverändert.	

Tabelle 1.1: Adressierungsarten in Zellbezügen

1. **Markieren Sie die Zellen C1 bis E4 und drücken Sie dann** Strg+⇧+F3.

   Excel erkennt die Namen in der obersten Zeile.



	A	B	C	D	E
1			Umsatz	Kosten	DB
2		Jan	100	80	20
3		Feb	110	90	
4		Mrz	120	90	
5					
6					

Abbildung 1.10: Deckungsbeitragsrechnung

2. **Bestätigen Sie mit OK (Abbildung 1.11).**

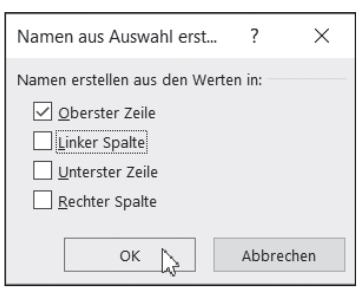


Abbildung 1.11: Namen aus Auswahl erstellen

3. Nun können Sie in die Zelle E2 = `Umsatz-Kosten` eingeben.

Obwohl die Namen »Umsatz« und »Kosten« als Bereiche definiert sind, weiß Excel, dass es in E2 auf die Zellen C2 und D2 zugreifen muss. Und es kommt noch besser:

Kopieren Sie die Zelle E2 auf die Zellen E3 bis E4 (Abbildung 1.12).

	A	B	C	D	E
1			Umsatz	Kosten	DB
2		Jan	100	80	20
3		Feb	110	90	
4		Mrz	120	90	
5					

Abbildung 1.12: Kopieren von Referenzen

Die Referenzen werden identisch kopiert, aber Excel greift immer auf die richtigen Zeilen zu (Abbildung 1.13).

	A	B	C	D	E	F
1			Umsatz	Kosten	DB	
2		Jan	100	80	20	=Umsatz-Kosten
3		Feb	110	90	20	=Umsatz-Kosten
4		Mrz	120	90	30	=Umsatz-Kosten
5						

Abbildung 1.13: Referenzieren auf Namensbereiche

Nun möchten wir in unserer Deckungsbeitragsrechnung noch den Monat April berücksichtigen:

1. Kopieren Sie die Zelle E4 auf E5. Excel meldet einen Fehler (Abbildung 1.14).

	A	B	C	D	E	F
1			Umsatz	Kosten	DB	
2		Jan	100	80	20	=Umsatz-Kosten
3		Feb	110	90	20	=Umsatz-Kosten
4		Mrz	120	90	30	=Umsatz-Kosten
5		April	130	100	#WERT!	
6						
7						

Abbildung 1.14: Zusätzliche Zeilen

2. Sie müssen noch die Namen neu definieren. Das geht ganz einfach: Sie markieren die Zellen C1 bis E5 und drücken wieder Strg+⇧+F3. Anschließend bestätigen Sie die Auswahl des Namens (Abbildung 1.11) und die Frage nach dem Ersetzen der vorhandenen Definition (Abbildung 1.15).

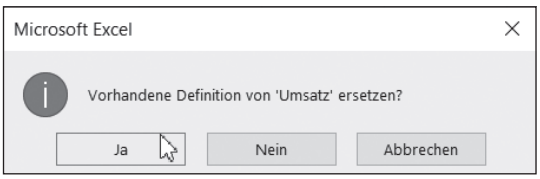


Abbildung 1.15: Ersetzen

Und siehe da, es passt (Abbildung 1.16).

	A	B	C	D	E	F
1			Umsatz	Kosten	DB	
2		Jan	100	80	20	=Umsatz-Kosten
3		Feb	110	90	20	=Umsatz-Kosten
4		Mrz	120	90	30	=Umsatz-Kosten
5		April	130	100	30	
6						

Abbildung 1.16: Namensverlängerung

Etwas komplizierter wird es, wenn Sie die Bereiche nicht manuell, sondern dynamisch anpassen wollen.

Eine besonders dynamische Form, mit Namen umzugehen, ist die folgende: Statt auf einen Zellbereich kann ein Name auch auf Formeln verweisen. Die Funktion `BEREICH.VERSCHIEBEN` hat eine schöne Eigenschaft: Der adressierte Bereich kann sich verschieben.

Die Funktion benötigt fünf Parameter: Startpunkt, Zeilenabstand vom Startpunkt, Spaltenabstand vom Startpunkt, Bereichshöhe und Bereichsbreite. Die letzten vier Parameter werden als »Anzahl der Zellen« angegeben. Das können Sie einfach abzählen.

Jetzt müssen Sie nur noch die Höhe des Bereichs dynamisch ermitteln. Hierzu gibt es die Funktion `ANZAHL`, die die Anzahl der mit Zahlen belegten Zellen eines Bereichs zurückgibt.

Damit ergibt sich der Umsatz als

```
Umsatz2:=BEREICH.VERSCHIEBEN(Namen!$C$2;0;0;ANZAHL(Namen!$C:$C);1)
```

und die Kosten ergeben sich als

```
Kosten2:=BEREICH.VERSCHIEBEN(Namen!$D$2;0;0;ANZAHL(Namen!$D:$D);1)
```

1. **Mit Strg+F3 kommen Sie in den Namens-Manager (Abbildung 1.17).**
2. **Klicken Sie auf Neu und geben Sie die beiden Namen ein.**

Jetzt können Sie beliebig viele Zeilen anhängen, ohne sich Gedanken über die Bereiche zu machen (Abbildung 1.18).

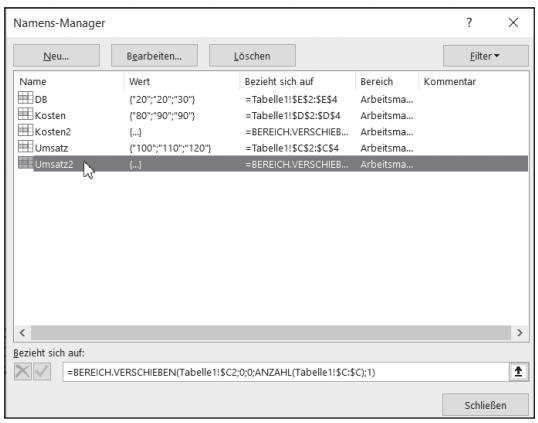


Abbildung 1.17: Namensdefinition

	A	B	C	D	E	F	G	H
1			Umsatz	Kosten	DB		DB	
2		Jan	100	80	20	=Umsatz-Kosten	20	=Umsatz2-Kosten2
3		Feb	110	90	20	=Umsatz-Kosten	20	=Umsatz2-Kosten2
4		Mrz	120	90	30	=Umsatz-Kosten	30	=Umsatz2-Kosten2
5		April	130	100	30		30	=Umsatz2-Kosten2
6		Mai	140	120	#WERT!		20	=Umsatz2-Kosten2
7								

Abbildung 1.18: Dynamische Namensverweise

Makros first

Natürlich können Programmierkenntnisse nicht schaden. Vorsicht ist allerdings geboten, wenn man sofort mit der Programmierung von Makros beginnt, ohne darüber nachgedacht zu haben, ob es in Excel nicht doch eine andere, funktionale Unterstützung gibt.

Excel verfügt wie alle Office-Programme über eine komplette Programmierumgebung, mit deren Hilfe sich Aufgaben automatisieren lassen. Der Begriff »Makro« ist da eigentlich nicht mehr angemessen, denn die Sprache Visual Basic for Applications (VBA) ist durchaus für eine professionelle Programmierung geeignet.

Gute Programmierung ist aufwendig und erfordert spezielle Fähigkeiten. Das betrifft weniger die Syntax oder die speziellen Funktionen der Programmiersprache. Dabei unterstützt Sie die Makro-Aufzeichnungsfunktion. Mithilfe dieser Funktion kann zwar Code erzeugt werden, der aber noch lange keine gute Programmierung darstellt.

Um die Nutzung von Makros wird man nicht ganz herumkommen. Aber: Verzichten Sie, wenn es geht, auf Makro-Programmierung. Nicht zuletzt kann man auch ohne Programmierung elegante Lösungen in Excel schaffen.

Makros – im Zweifelsfall ohne

Wir haben lange überlegt, ob wir in diesem Buch überhaupt auf Makros eingehen sollten. Aber bestimmte Aufgaben lassen sich mit Excel (noch) nicht ohne den Einsatz von Makros lösen. Allerdings präferieren wir im Zweifelsfall immer eine Lösung über Formeln.

Aufzeichnen von Makros

Sie müssen kein ausgewiesener Programmierfuchs sein, um mit Makros arbeiten zu können, denn es ist eigentlich gar nicht so schwer. Gerade wenn Sie unter Zeitdruck stehen, kann die Arbeit mit Makros schneller gehen, als wenn Sie sich durch die äußerst umfangreiche Objektbibliothek mittels Hilfefunktion kämpfen.

Starten wir also mit der Entwicklung unseres ersten Makros. Dazu nutzen wir die Aufzeichnungsfunktion von Excel. Wir wollen eine spezielle Formatierungseigenschaft auf ein Makro legen, und zwar den Zeilenumbruch:

1. **Erzeugen Sie eine leere Arbeitsmappe.**

2. **Wählen Sie den Befehl Makro aufzeichnen aus dem Menü Entwicklertools der Multifunktionsleiste aus.**

3. **In der erscheinenden Maske (Abbildung 1.19) geben Sie den Namen `Mein_erstes_Makro` ein.**

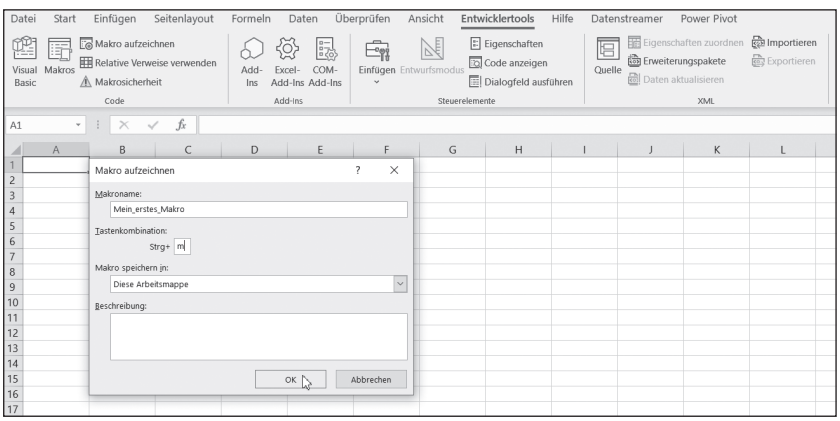


Abbildung 1.19: Start der Makro-Aufzeichnung

Sie können auch einen anderen Namen vergeben, allerdings sind spezielle Sonderzeichen und insbesondere Leerzeichen nicht erlaubt.

Geben Sie als Tastenkombination Strg+M ein und bestätigen Sie mit OK. Die Tastenkombination ist ganz hilfreich, weil Sie dieses Makro damit später aufrufen können. Der Wermutstropfen dabei ist, dass Sie sich die Kombination merken müssen. Und wenn Sie diesen Komfort für viele Makros nutzen, kann das schon verwirrend werden.

4. **Nun wollen wir die eigentliche Aktion ausführen. Rufen Sie dazu die Zellformatierung über das Kontextmenü auf (Abbildung 1.20) und klicken Sie dann auf Textumbruch. Schließen Sie die Formatierungsmaske mit OK.**

5. **Beenden Sie die Aufzeichnung, indem Sie im Menü Entwicklertools auf Aufzeichnung beenden klicken. Ihr erstes Programm ist damit geschrieben. Doch wo befindet es sich?**

Mit Alt+F11 gelangen Sie in die Programmierumgebung von Excel (Abbildung 1.21). Visual Basic for Applications (VBA) ist eine vollwertige Programmierumgebung, die auch viele Profis schätzen. Die Umgebung hat diverse Fenster, die Ihnen die notwendigen Informationen zur Verfügung stellen.

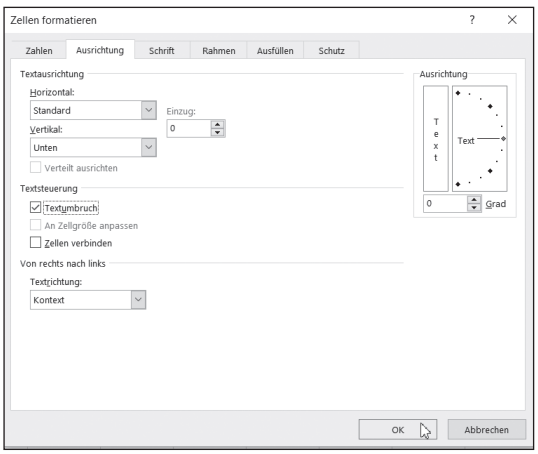


Abbildung 1.20: Aktion während der Aufzeichnung

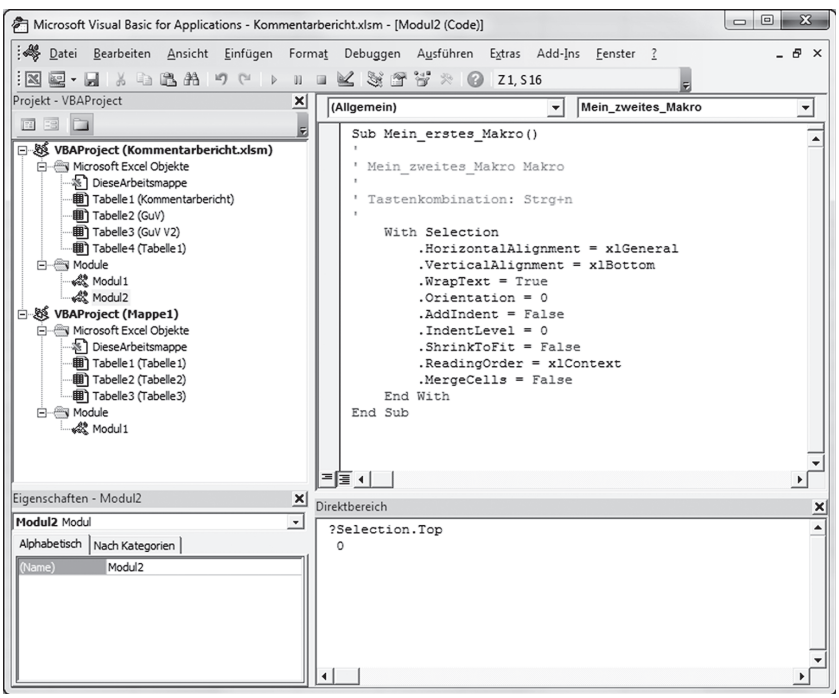


Abbildung 1.21: Die VBA-Entwicklungsumgebung

✔ Das Projektfenster (links oben) enthält alle Objekte des VBA-Projekts. Dazu gehören auch die Arbeitsblätter, die selbst VBA-Programme aufrufen können.

- ✔ Das Eigenschaftsfenster (links unten) stellt alle Attribute zum aktuell ausgewählten Objekt dar.
- ✔ Das Codefenster (rechts oben) zeigt den eigentlichen BASIC-Code.
- ✔ Das Direktfenster (rechts unten) benötigen Sie in der Regel zur Fehlersuche.

Ihr erstes Makro ist im »Modul1« gelandet (Abbildung 1.21). Die Aufzeichnungsfunktion legt automatisch einen solchen Programmbereich an. Jetzt müssen Sie noch testen, ob Ihr Makro auch funktioniert.

1. **Gehen Sie zurück zu Excel (am einfachsten über die Windows-Steuerung). Windows behandelt die Programmierumgebung übrigens wie ein eigenständiges Programm.**
2. **Geben Sie in eine Zelle eines Arbeitsblatts zum Beispiel `Dies ist mein erstes Makro` ein.**

   Der Text sollte länger als die Zellenbreite sein. In der Standardeinstellung würde Excel keinen Zeilenumbruch vornehmen und über die Zellengrenze hinausschreiben, falls die Nachbarzelle leer ist.
3. **Betätigen Sie nun Strg+M. Was passiert? Wenn Sie alles richtig gemacht haben, dann wird die Zelle neu formatiert, und der Zelleninhalt wird umbrochen (Abbildung 1.22).**

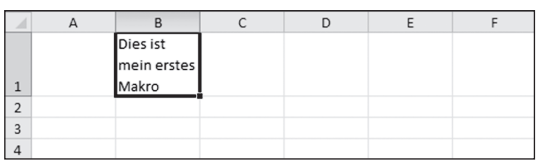


Abbildung 1.22: Das Makro-Ergebnis

Die Makro-Analyse

Für unsere kleine Änderung wird doch ein recht langes Makro benötigt: viel Code für wenig Ausführung. Das geht auch effektiver! Sie müssen einfach alles aus dem Code entfernen, was nicht benötigt wird. Das gilt insbesondere für die »nicht bestellten« Bestandteile. Wir zeigen Ihnen gleich im Detail, wie Sie vorgehen müssen. Wir wollen Ihnen aber hier schon einmal das viel schlankere Ergebnis unserer Korrektur vorstellen. Von den Attributen benötigen Sie nur `.WrapText = True`.

```
Sub Mein_erstes_Makro()
    With Selection
        .WrapText = True
    End With
End Sub
```

Die Makro-Aufzeichnung ist also alles andere als effektiv. Und das Makro macht eine ganze Menge Sachen, die wir gar nicht möchten. Das liegt daran, dass Excel nicht entscheiden kann, was mit den Auswahloptionen passieren soll, die Sie beim Aufruf der Maske nicht angeklickt haben.

Doch nun kurz zu dem Makro im Detail:

- ✔ Eine Prozedur wird durch `Sub <Prozedurname>()` eingeleitet. Zwischen den Klammern können noch Parameter übergeben werden. Alle folgenden Ausführungen werden sequenziell bis `End Sub` abgearbeitet.
- ✔ Das Objekt `Selection` kennzeichnet die aktuelle Markierung im Arbeitsblatt. Das können eine Zelle, ein Zellbereich oder auch mehrere Zellbereiche sein.
- ✔ Ein Objekt hat diverse Eigenschaften und Methoden. Unser Objekt hat beispielsweise die Eigenschaft `.WrapText` mit den Attributen `True` oder `False`. In der Formatierungsmaske wird dies durch eine Checkbox gekennzeichnet. Objekteigenschaften werden dargestellt als `<Objektname>.<Objekteigenschaft>`.
- ✔ `With` ist ein »Faulenzer-Befehl«. Bei allen Statements zwischen `With<Object>` und `End With` können Sie den Objektbezeichner weglassen und müssen lediglich `<Object>` schreiben.

In unserem Beispiel bedeutet das allerdings keine Ersparnis. Insofern können wir nochmals vereinfachen zu:

```
Sub Mein_erstes_Makro()
    Selection.WrapText = True
End Sub
```

Sie können das Makro übrigens auch Schritt für Schritt ausführen. Gehen Sie dazu in die VBA-Umgebung und klicken Sie auf eine Position innerhalb der Prozedur. Wählen Sie im Menü DEBUGGEN den Befehl EINZELSCHRITT aus beziehungsweise drücken Sie F8.

Sie sehen, dass die erste Zeile der Prozedur gelb hinterlegt ist (Abbildung 1.23).

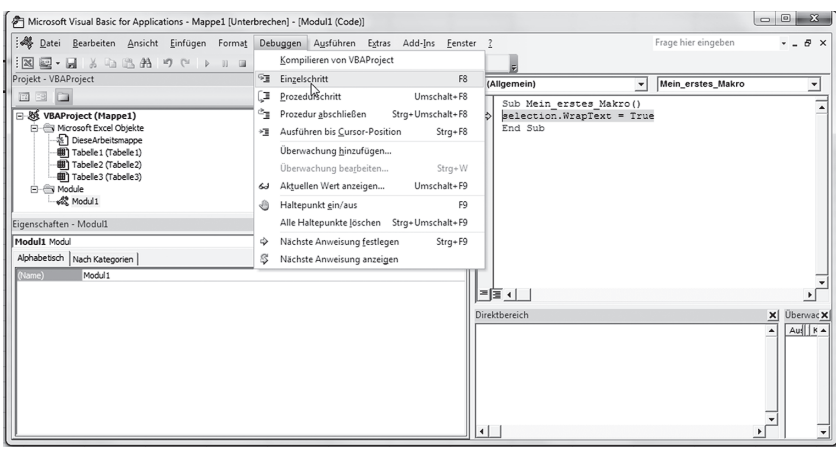


Abbildung 1.23: Makros Schritt für Schritt ausführen

Nun ist es durchaus aufwendig, jedes Mal die Makro-Aufzeichnungsfunktion zu starten und alles an Code zu entfernen, was nicht benötigt wird. Der VBA-Editor bietet auch eine andere Möglichkeit, die vielen Hundert Objekteigenschaften auszuwählen.

Wir schlagen noch eine kleine Ergänzung vor:

```
Sub Mein_erstes_Makro()
Dim S As Range
Set S = Selection
S.WrapText = True
End Sub
```

Definieren Sie eine Variable `S` mit der Einleitung `DIM`. Mittels `As` weisen Sie den Objekttyp `Range` zu. Dieser Objekttyp referenziert auf eine Zelle oder auf Zellbereiche. Man kann damit Zellinhalte lesen, schreiben oder auch alle Formatierungen automatisieren.

Mit der `Set`-Anweisung referenzieren Sie auf die aktuelle Auswahl. Wenn Sie jetzt im VBA-Code am Anfang einer Zeile `S.` eingeben, öffnet sich sofort eine Dropdown-Liste, und Sie sehen alle zu dem Objekt verfügbaren Eigenschaften und Methoden (Abbildung 1.24).

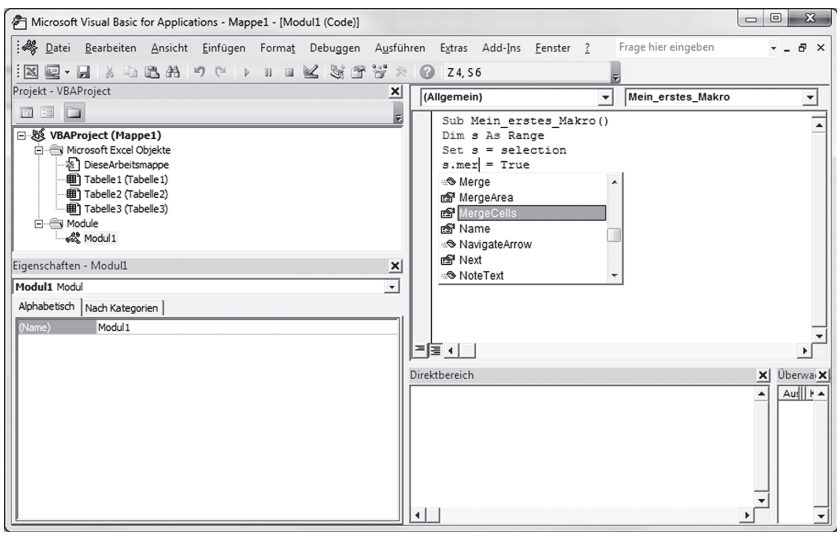


Abbildung 1.24: Automatische Ergänzung durch den VBA-Editor

Ein wenig VBA

Von unseren Kunden wurde immer wieder der Wunsch geäußert, die Kommentare in Arbeitsblättern auswerten zu können. Das wollen wir Ihnen deshalb nicht vorenthalten.

Das Beispiel finden Sie unter `\Beispiele\Kommentarbericht.xlsm`.

In Abbildung 1.25 sehen Sie eine Gewinn-und Verlust-Rechnung (GuV) mit Zellen, deren Ecken rechts oben markiert sind (zum Beispiel Zelle D4). Bei diesen Zellen sind Kommentare hinterlegt. Nun soll ein Bericht erstellt werden, der die Kommentare als Zellen auswertet und nach Möglichkeit auch die Zeilenhöhe dynamisch anpasst, sodass auf jeden Fall der gesamte Kommentar dargestellt wird.

	A	B	C	D	E	F	G
1		**GuV**					
2		Positionen	2015	2016	2017	2018	2019
3		Umsatzerlöse	3000	3100	3200	3300	3400
4		Bestandsveränderung unfertige u. fertige Erze	300	-200	-300	300	200
5		andere aktivierte Eigenleistungen	200	200	200	200	200
6		sonstige betriebliche Erträge	300	300	200	300	300
7		Aufwendungen für RHB und für bezogene Wa	600	650	600	700	700
8		Aufwendungen für bezogene Leistungen	600	600	600	600	600
9		Materialaufwand	1200	1250	1200	1300	1300
10		Löhne und Gehälter	400	400	400	400	400
11		soziale Aufwendungen und Aufw. für Altersve	200	200	200	200	200
12		Personalaufwand	600	600	600	600	600
13		Abschreibungen	400	400	400	400	400
14		sonstige betriebliche Aufwendungen	100	100	100	100	100
15		Betriebsergebnis	1500	1050	1000	1700	1700
16		Erträge aus Beteiligungen	100	100	100	100	100
17		Erträge aus anderen Wertpapieren und Auslei	100	100	100	100	100
18		sonstige Zinsen und ähnliche Erträge	100	100	100	100	100
19		Aufwendungen aus Verlustübernahme	100	100	100	-400	100
20		Aufwendungen aus assoziierten Unternehmer	100	100	100	100	100
21		Abschreibungen auf Finanzanlagen und Wertp	100	100	100	100	100
22		Zinsen und ähnliche Aufwendungen	100	100	100	100	100
23		Finanzergebnis	-100	-100	-100	400	-100
24		Ergebnis des gewöhnlichen Geschäftstätigkeit	1400	950	900	2100	1600
25		außerordentliche Erträge	100	200	100	200	100
26		außerordentliche Aufwendungen	100	100	200	100	200
27		außerordentliches Ergebnis	0	100	-100	100	-100
28		Steuern vom Einkommen und vom Ertrag	500	500	500	500	500
29		sonstige Steuern	200	200	200	200	200
30		Jahresüberschuß (Jahresfehlbetrag)	700	350	100	1500	800
31		anderen Gesellschaftern zustehendes Ergebn	300	100	100	100	100
32		Bilanzgewinn (Bilanzverlust)	400	250	0	1400	700
33							

Abbildung 1.25: GuV mit kommentierten Zellen

Der erste Entwurf könnte wie in Abbildung 1.26 aussehen.

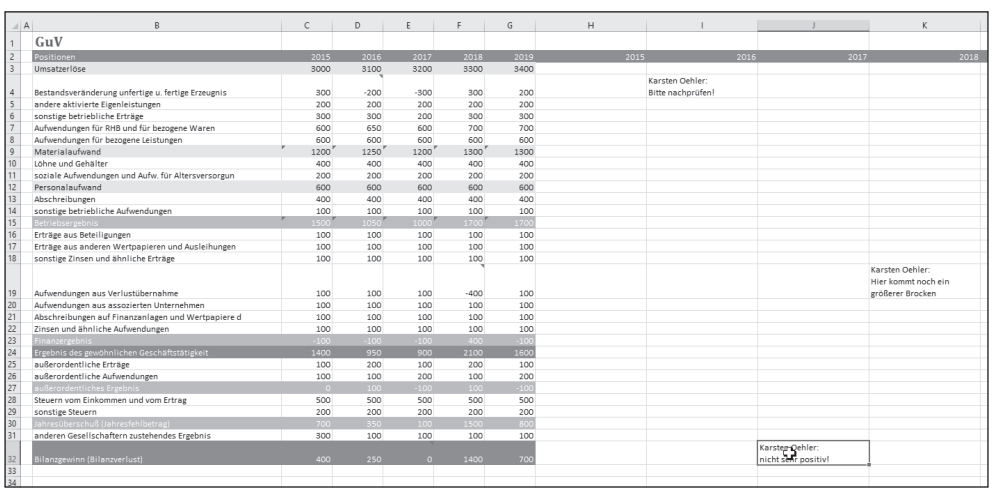


Abbildung 1.26: Kommentarbericht

Leider steht eine Funktion KOMMENTAR(<Zelle>) in den Standardeinstellungen von Excel nicht zur Verfügung. Sie können aber eine eigene Funktion schreiben. Wechseln Sie dazu mit Alt+F11 in die VBA-Umgebung. Und so könnte der Code dazu aussehen:

```
Function Kommentar(Bezug As Range) As String
 Kommentar = Bezug.Comment.Text
End Function
```

Dazu einige Anmerkungen:

1. Anstatt mit Sub <Prozedurname> leiten Sie den Code mit Function <Funktionsname> ein. Beenden Sie diese Funktion mit End Function.
2. Die Funktion muss einen Wert zurückgeben. Dieser ist vom Typ »String«, also ein Text.
3. Diese Funktion nimmt einen Parameter auf. Wir übergeben eine Zelle. Deshalb übernehmen wir den Parameter als Typ »Range«. »Range« kann eine einzelne Zelle, aber auch ein Zellbereich sein.
4. »Range« hat eine Eigenschaft »Comment«, die unseren Kommentar enthält. Diese ist ein Subobjekt und hat selbst wieder Eigenschaften. Uns interessiert hier die Eigenschaft »Text«.

Probieren Sie es aus! Es gibt allerdings noch einen unschönen Nebeneffekt: Wenn eine Zelle keinen Kommentar enthält, wird ein Fehlercode (#WERT) übergeben. Diesen Fehler müssten Sie mit einer If-Schleife abfangen. Die Bedingung hierzu lautet Bezug.Comment Is Nothing. Beim Vergleich wird kein Gleichheitszeichen verwendet, sondern Is zeigt an, dass hier die Arbeitsspeicheradressen von Objekten verglichen werden. Nothing ist damit die Prüfung, ob eine Adresse für das Objekt existiert.

```
Function Kommentar(Bezug As Range) As String
If Bezug.Comment Is Nothing Then
 Kommentar = ""
Else
 Kommentar = Bezug.Comment.Text
End If
End Function
```

Das Ergebnis kann sich sehen lassen (Abbildung 1.27).

Dafür haben wir eine zweite Funktion (KommentarRange) geschrieben:

```
Public Function KommentarRange(Bezug As Range) As String
KommentarRange = ""
For Each r In Bezug.Cells
If Not r.Comment Is Nothing Then
 KommentarRange = KommentarRange & vbLf & Cells(2, r.Column) & _
 ": " & r.Comment.Text
End If
Next
End Function
```

	A	B	C	D	E	F	G	H
1		**GuV**						
2		Positionen	2015	2016	2017	2018	2019	Kommentare
3		Umsatzerlöse	3000	3100	3200	3300	3400	
4		Bestandsveränderung unfertige u. fertige Erzeugnis	300	-200	-300	300	200	Karsten Oehler: Bitte nachprüfen!
5		andere aktivierte Eigenleistungen	200	200	200	200	200	
6		sonstige betriebliche Erträge	300	300	200	300	300	
7		Aufwendungen für RHB und für bezogene Waren	600	650	600	700	700	
8		Aufwendungen für bezogene Leistungen	600	600	600	600	600	
9		Materialaufwand	1200	1250	1200	1300	1300	
10		Löhne und Gehälter	400	400	400	400	400	
11		soziale Aufwendungen und Aufw. für Altersversorgun	200	200	200	200	200	
12		Personalaufwand	600	600	600	600	600	
13		Abschreibungen	400	400	400	400	400	
14		sonstige betriebliche Aufwendungen	100	100	100	100	100	
15		Betriebsergebnis	1500	1050	1000	1700	1700	
16		Erträge aus Beteiligungen	100	100	100	100	100	
17		Erträge aus anderen Wertpapieren und Ausleihungen	100	100	100	100	100	
18		sonstige Zinsen und ähnliche Erträge	100	100	100	100	100	
19		Aufwendungen aus Verlustübernahme	100	100	100	-400	100	Karsten Oehler: Hier kommt noch ein größerer Brocken
20		Aufwendungen aus assozierten Unternehmen	100	100	100	100	100	
21		Abschreibungen auf Finanzanlagen und Wertpapiere d	100	100	100	100	100	
22		Zinsen und ähnliche Aufwendungen	100	100	100	100	100	
23		Finanzergebnis	-100	-100	-100	400	-100	
24		Ergebnis des gewöhnlichen Geschäftstätigkeit	1400	950	900	2100	1600	
25		außerordentliche Erträge	100	200	100	200	100	
26		außerordentliche Aufwendungen	100	100	200	100	200	
27		außerordentliches Ergebnis	0	100	-100	100	-100	
28		Steuern vom Einkommen und vom Ertrag	500	500	500	500	500	
29		sonstige Steuern	200	200	200	200	200	
30		Jahresüberschuß (Jahresfehlbetrag)	700	350	100	1500	800	
31		anderen Gesellschaftern zustehendes Ergebnis	300	100	100	100	100	
32		Bilanzgewinn (Bilanzverlust)	400	250	0	1400	700	Karsten Oehler: nicht sehr positiv!

Abbildung 1.27: Sammlung der Kommentare

Hier wird die Schleife `For Each` verwendet. Der übergebene Bereich enthält eine Sammlung von Zellen, die mit `Bezug.Cells` abgefragt werden kann. Es wird ein Verweis auf eine sogenannte »Collection« von Objekten übergeben. Jede einzelne Zelle wird einzeln mit der `For`-Schleife abgefragt. Mit `&` werden dann die Kommentare »montiert«. `vbLf` ist eine Systemkonstante, die den Zeilenumbruch in einer Zelle erzwingt.

Vielleicht wollen Sie aber auch einen komplett eigenständigen Kommentarbericht erzeugen. Auch hierbei unterstützt Sie VBA. In diesem Fall schreiben Sie keine Funktion, sondern eine Prozedur, die mit `Sub` eingeleitet wird;

```
Public Sub Report()

With Sheets("Kommentarbericht")
k = 2
While .Cells(k, 1) <> ""
 .Cells(k, 1).Delete
Wend
End With

k = 2
With Range("GuV")
```

```
For i = 2 To .Rows.Count
 For j = 2 To .Columns.Count
  If Not .Cells(i, j).Comment Is Nothing Then
   Sheets("Kommentarbericht").Cells(k, 1) = .Cells(i, 1) & _
   " / " & .Cells(1, j) & " : " & .Cells(i, j).Comment.Text
   k = k + 1
  End If
 Next
Next
End With
End Sub
```

Einige Erläuterungen zu dieser Prozedur:

- ✔ Da die Prozedur mehrfach aufgerufen werden sollte, müssen eventuell bestehende Zellen gelöscht werden. Man kann einen »Range« löschen, aber wir wissen nicht, wie groß der Bereich ist. Deshalb löschen wir einfach die Zellen so lange, bis kein Text mehr im Kommentarreport steht. Hierzu gibt es die Schleifenkonstruktion `While Wend`: Der in dieser Schleife stehende Code wird so lange ausgeführt, wie die Bedingung nach `While` gültig ist.
- ✔ Mit zwei `For`-Schleifen (horizontal und vertikal) prüfen wir im GuV-Bericht, ob Kommentare existieren.
- ✔ Wir greifen hier über Excel-Namen auf Zellbereiche zu. `Range( ("GuV")` liefert den Zellbereich zurück, der durch `"GuV"` adressiert wird. Mit Namen zu arbeiten, hat in VBA den Vorteil, dass Sie bei Änderungen im Bericht nicht den Code ändern müssen.

Das Ergebnis sehen Sie in Abbildung 1.28.

	A
1	Auflistung der Kommentare
2	Bestandsveränderung unfertige u. fertige Erzeugnis / 2016 : Karsten Oehler: Bitte nachprüfen!
3	Aufwendungen aus Verlustübernahme / 2018 : Karsten Oehler: Hier kommt noch ein größerer Brocken
4	Bilanzgewinn (Bilanzverlust) / 2017 : Karsten Oehler: nicht sehr positiv!
5	
6	
7	

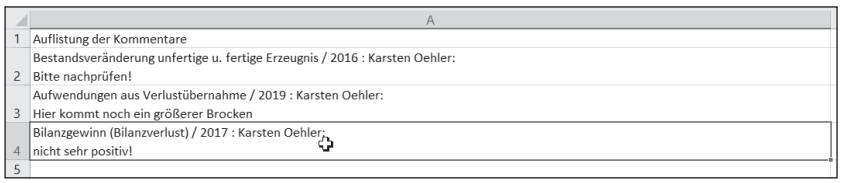
Abbildung 1.28: Eigenständige Kommentarliste

Die Worksheet-Funktionen

Um das Verständnis einer eigenständigen Kommentarliste zu erhöhen, hilft es, bei Doppelklick auf einen Kommentar (Abbildung 1.29) in die GuV zu springen, und zwar genau an die Stelle, auf die die Zelle im Kommentarbericht verweist (Abbildung 1.30).

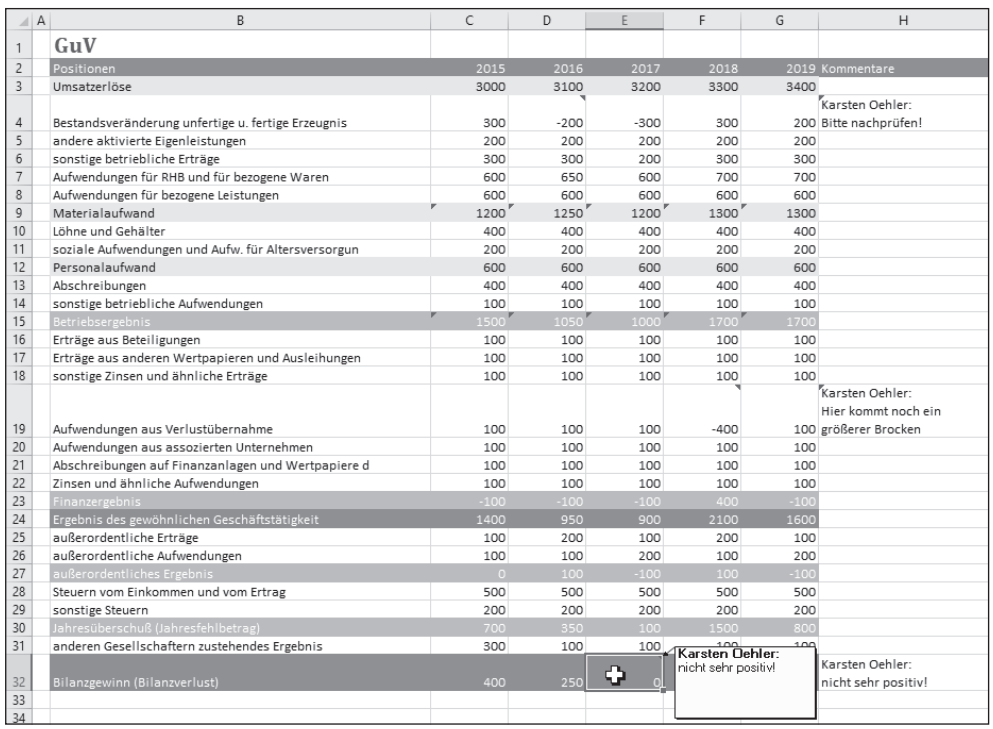
Der Code für diese Funktion muss allerdings in einem besonderen Bereich stehen.

1. **Öffnen Sie die VBA-Umgebung mit Alt+F11 des VBAProject »Kommentarbericht .xlsm«.**



	A
1	Auflistung der Kommentare
2	Bestandsveränderung unfertige u. fertige Erzeugnis / 2016 : Karsten Oehler: Bitte nachprüfen!
3	Aufwendungen aus Verlustübernahme / 2019 : Karsten Oehler: Hier kommt noch ein größerer Brocken
4	Bilanzgewinn (Bilanzverlust) / 2017 : Karsten Oehler: nicht sehr positiv!
5	

Abbildung 1.29: Doppelklick auf einen Kommentar



	A	B	C	D	E	F	G	H
1		**GuV**						
2		Positionen	2015	2016	2017	2018	2019	Kommentare
3		Umsatzerlöse	3000	3100	3200	3300	3400	
4		Bestandsveränderung unfertige u. fertige Erzeugnis	300	-200	-300	300	200	Karsten Oehler: Bitte nachprüfen!
5		andere aktivierte Eigenleistungen	200	200	200	200	200	
6		sonstige betriebliche Erträge	300	300	200	300	300	
7		Aufwendungen für RHB und für bezogene Waren	600	650	600	700	700	
8		Aufwendungen für bezogene Leistungen	600	600	600	600	600	
9		Materialaufwand	1200	1250	1200	1300	1300	
10		Löhne und Gehälter	400	400	400	400	400	
11		soziale Aufwendungen und Aufw. für Altersversorgun	200	200	200	200	200	
12		Personalaufwand	600	600	600	600	600	
13		Abschreibungen	400	400	400	400	400	
14		sonstige betriebliche Aufwendungen	100	100	100	100	100	
15		Betriebsergebnis	1500	1050	1000	1700	1700	
16		Erträge aus Beteiligungen	100	100	100	100	100	
17		Erträge aus anderen Wertpapieren und Ausleihungen	100	100	100	100	100	
18		sonstige Zinsen und ähnliche Erträge	100	100	100	100	100	
19		Aufwendungen aus Verlustübernahme	100	100	100	-400	100	Karsten Oehler: Hier kommt noch ein größerer Brocken
20		Aufwendungen aus assoziierten Unternehmen	100	100	100	100	100	
21		Abschreibungen auf Finanzanlagen und Wertpapiere d	100	100	100	100	100	
22		Zinsen und ähnliche Aufwendungen	100	100	100	100	100	
23		Finanzergebnis	-100	-100	-100	400	-100	
24		Ergebnis des gewöhnlichen Geschäftstätigkeit	1400	950	900	2100	1600	
25		außerordentliche Erträge	100	200	100	200	100	
26		außerordentliche Aufwendungen	100	100	200	100	200	
27		außerordentliches Ergebnis	0	100	-100	100	-100	
28		Steuern vom Einkommen und vom Ertrag	500	500	500	500	500	
29		sonstige Steuern	200	200	200	200	200	
30		Jahresüberschuß (Jahresfehlbetrag)	700	350	100	1500	800	
31		anderen Gesellschaftern zustehendes Ergebnis	300	100	100			
32		Bilanzgewinn (Bilanzverlust)	400	250	0			Karsten Oehler: nicht sehr positiv!
33								
34								

Karsten Oehler:
nicht sehr positiv!

Abbildung 1.30: Das Ziel: die dazugehörige Zelle zu einem Kommentar

2. **Im linken Fenster sehen Sie die verschiedenen Module. Klicken Sie nun doppelt auf `Tabelle 1 (Kommentarbericht)`. Hier sind die arbeitsblattspezifischen Prozeduren hinterlegt.**

   Für jedes Blatt existieren Standardfunktionen, die aufgerufen werden, wenn ein bestimmtes Ereignis, wie zum Beispiel der Doppelklick oder eine Zelländerung, auftritt.

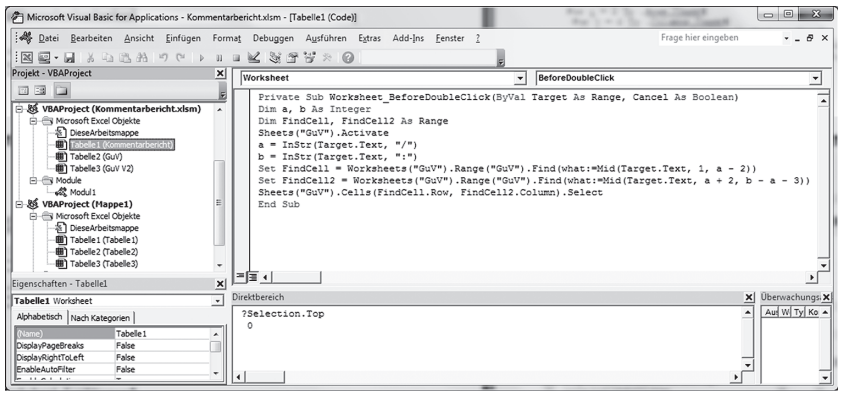


Abbildung 1.31: Das Makro-Ergebnis

3. **Sie finden die vordefinierten Zellen, indem Sie das Kombinationsfeld rechts anklicken (Abbildung 1.32). Bereiche, für die Code hinterlegt wurde, werden fett markiert.**

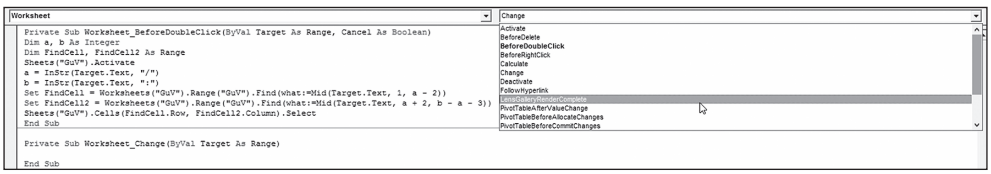


Abbildung 1.32: Die eingebauten Funktionen

Wenn Sie die Prozedur `BeforeDoubleClick` zum ersten Mal auswählen, um Code einzufügen, erzeugt VBA einen Rumpf:

```
Private Sub Worksheet_BeforeDoubleClick
(ByVal Target As Range, Cancel As Boolean)
End Sub
```

Es werden folgende Parameter übergeben: `Target` ist eine Referenz auf die Zelle, auf die geklickt wurde. `Cancel` ist eine »True/False«-Variable. Wenn Sie diese auf `True` setzen, wird das Standardverhalten von Excel ausgeschaltet. Klicken Sie beispielsweise doppelt auf eine Grafik, dann öffnen sich üblicherweise die Formatierungseigenschaften. Wenn Sie darauf verzichten wollen, weil nur die von Ihnen geschriebene Prozedur ausgeführt werden soll, dann müssen Sie `Cancel` auf `True` setzen.

Hier ist der Code zu unserer Prozedur:

```
Private Sub Worksheet_BeforeDoubleClick _
(ByVal Target As Range, Cancel As Boolean)
Dim a, b As Integer
Dim FindCell, FindCell2 As Range
```

```
Sheets("GuV").Activate
a = InStr(Target.Text, "/")
b = InStr(Target.Text, ":")

Set FindCell = _
Worksheets("GuV").Range("GuV").Find _
(what:=Mid(Target.Text, 1, a - 2))
Set FindCell2 = _
Worksheets("GuV").Range("GuV").Find _
(what:=Mid(Target.Text, a + 2, b - a - 3))

Sheets("GuV").Cells(FindCell.Row, FindCell2.Column).Select
End Sub
```

Einige hilfreiche Erläuterungen:

- ✔ In der ausgewählten Zelle suchen wir nach Referenzen. `InStr` gibt die Position im Text zurück, an der das gesuchte Zeichen steht. Wir suchen nach »/« und »:«. Aber aufgepasst: Stellen Sie sicher, dass diese Sonderzeichen nicht in den GuV-Positionen verwendet werden.
- ✔ Über die `Find`-Funktion des `Range`-Objekts können wir nach dem Text suchen. Die `Find`-Funktion liefert die Zelladresse. Damit haben wir aber erst den Zeilen- und den Spaltenkopf.
- ✔ Die gesuchte Zelle ergibt sich aus der Zeile des Zeilenkopfs und der Spalte des Spaltenkopfs. Mit der `Select`-Funktion aktivieren wir die gefundene Adresse.

Jetzt haben Sie einige Grundlagen von VBA kennengelernt: ein gutes Rüstzeug für die nächsten Kapitel. In diesen kommen wir immer wieder auf VBA zurück, sodass Sie sukzessive Ihr Know-how über diese Programmiersprache ausbauen können.