Thorsten Spitta
(Vatter Unternehmensgruppe)

Sechs Jahre Anwendungsentwicklung mit Prototyping

- Revision von Begriffen und Konzepten -

Zusammenfassung: Es wird gezeigt, daß eine Revision des Begriffs (Software-)Prototyp zweckmäßig ist und bisherige Klassifikationen zum Prototyping eingeschränkt werden sollten. Es wird empfohlen, zwischen Modellen, Prototypen und Pilotsystemen zu unterscheiden. Die Sinnhaftigkeit der Begriffseinschränkung und der tatsächliche Verlauf großer Projekte wird an Fallstudien gezeigt. Die Fallstudien entstammen der Entwicklung eines integrierten Systems mit über 40 MJ Entwicklungsaufwand. Das System besteht aus: Tourenplanung, Produktionssteuerung, Bestandsführung und Vertriebsabwicklung. Anhand der Fallstudien werden auch Nutzen und Gefahren des Prototyping gezeigt.

1. Begriffe

1.1 (Software-)Prototypen

Die Begriffe Prototyp/Prototyping werden in der Softwareentwicklung trotz einer begriffsklärenden Tagung 1983 [Approaches to Prototyping: 2] bis heute verschieden benutzt [vgl. 2, 4, 7, 12, 13]. Der Begriff Prototyp soll hier so eingegrenzt werden, wie er der Projektarbeit unter industriellen Bedingungen entspricht. Dabei rückt der Autor auch von dem bisher selbst praktizierten Sprachgebrauch ab. Ich beziehe mich auf den umgangssprachlichen Gebrauch von Prototyp.

Aus praktischer Anschauung versteht wohl jeder dasselbe unter dem **Prototyp eines Automobils**. Dies ist ein real funktionierendes Auto, an dem man systematisch erprobt, was es unter Extrembedingungen leistet und aushält. Bei der Softwareentwicklung gibt es keine einheitliche, sondern üblicherweise drei Betrachtungsebenen von Prototypen [vgl. Floyd in: 2]:

1. Der **Entwicklungsschritt**, in dem Prototypen angewendet werden: Dies sind gelegentlich die Vorstudie, meist Spezifikation und Softwareentwurf [vgl. hierzu Abb.1 Folgeseite].
2. Die **Vorgehensweise bei der Entwicklung**: Sie ist linear oder zyklisch (Folge von Versionen).
3. Der **Gegenstand**, den ein Prototyp abbildet: Das Produktionssystem oder ein davon verschiedenes Werkzeug.

Statt Definitionen noch einige Beispiele aus dem Alltag:

- ein Modellauto ist kein Prototyp;
- ein Musterhaus ist ein **Prototyp** und kein Modell;
- ein Kapazitätsbelegungsmodell bleibt ein **Modell**, ob es auf dem Papier oder einem Computer gerechnet wird;
- ein Buchhaltungssystem, ob mit "Haupt- und Nebenbüchern" manuell verwaltet oder mit einem Computer, ist ein **ausführbares Modell** des Werte- und Mengenflusses eines Unternehmens. Es ist sicher kein Prototyp;
- die ersten BTX-Anwender haben mit einem **Pilotsystem** gearbeitet und nicht mit einem Prototyp.

Prototypen sollen sich demnach von Modellen, von Pilotsystemen [vgl. Rezevski in: 2] und von Produktivsystemen unterscheiden. Sie sind ein Stück Software, das vor dem Produktivsystem entwickelt und mit dem dann gearbeitet wird (Exploration oder Experiment [vgl. Floyd in: 2]). Es ist jedoch eine Einschränkung notwendig:

Man kann Prototypen im Rahmen eines linearen Vorgehensmodells einsetzen ["Wasserfallmodell" nach Boehm] oder in einem zyklischen [Floyd 81: 6]. Prototyping ist jedoch nicht gleichzusetzen mit evolutionärer (=zyklischer) Softwareentwicklung.

Prototypen können wie jede Software evolutionär zum Endprodukt weiterentwickelt werden, wenn der Prototyp in der Zielsprache und -umgebung erstellt wird. Sehr leicht wird jedoch eine notwendige Bedingung hierfür übersehen: Gerade evolutionäre Softwareentwicklung verlangt ein kontrolliertes Vorgehen, d.h. es werden Spezifikationen und Entwürfe erstellt und begutachtet. Ohne die Beachtung dieser Bedingung gleitet evolutionäres Prototyping [vgl. Floyd in: 2] leicht in eine 'quick and dirty'-Entwicklung ab. Um das zu verhindern, muß entweder ein hoher Kontrollaufwand betrieben werden oder man räumt die Möglichkeit für diese Art der Entwicklung gar nicht erst ein.

Für das Weitere seien folgende Entwicklungsschritte eines vereinfachten zyklischen Vorgehensmodells unterstellt:

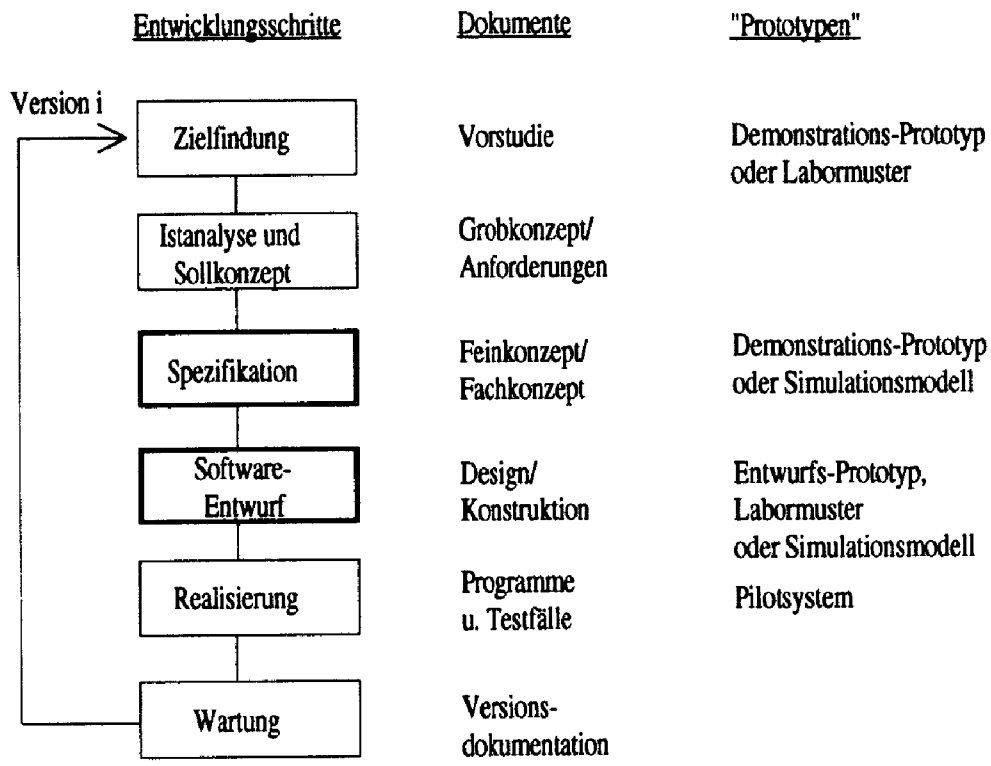

Abb. 1: Entwicklungsschritte mit Prototypen (/= Synonym)

Prototypen werden überwiegend in den doppelt umrandeten Schritten verwendet, ohne textuelle oder formale Spezifikationen und Software-Entwürfe zu ersetzen. Nennt man alle vor der Realisierung hilfsweise erstellten Programme Prototypen -dies der heute gängige Sprachgebrauch-, dann ist "Prototyping" das Arbeiten hiermit (gelegentlich auch das Erstellen).

Folgende Abgrenzung sollte jedoch vorgenommen werden:

Modell	Prototyp	Produktivsystem
Labormuster/ [7]	**Demonstrations-PrT/** [7]	**Pilotsystem** [in 2]
Testprogramm; [13]	Kommunikations-PrT;[14]	
Simulationsmodell	**Entwurfs-Prototyp/** [14]	evolutionär entwickelte
	Durchstich/ [3]	Version [in 2]
	Architektur-PrT/ [12]	
	technischer PrT. [13]	

Tab. 1: Zuordnung von Begriffen zu Prototypen (PrT)

Der hier skizzierte Sprachgebrauch [Feudruck] ist als Diskussionsbeitrag gedacht und in der starken Eingrenzung des Prototyping zugegebenermaßen radikal. Wozu ist das nützlich?

Prototypen i.S. von Tabelle 1 sind in der Zielsprache und Zielumgebung erstellt. Dies ist m.W. bei der überwiegenden Mehrzahl der praktisch verwendeten Prototypen der Fall, da man meist den Aufwand einer Re-Implementierung und den Bruch an der Benutzerschnittstelle scheut. Die Gefahr einer unkontrollierten Weiterentwicklung ist genau bei diesem "Typ von Prototyp" sehr groß. Sie besteht bei Modellen nicht, bei Pilotsystemen ist eine iterative Weiterentwicklung gewollt. Bei einem wie oben dargestellten Prototyping kann der Entwicklungsprozeß gezielt überwacht werden.

Abschließend ein Bild, das verdeutlicht, wie die am häufigsten verwendeten Prototypen in das zu erstellende Produkt einzuordnen sind:

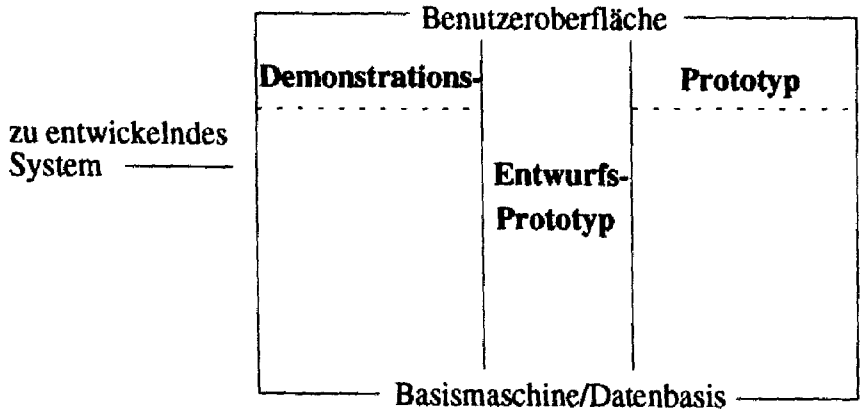

Abb. 2: Zwei Arten von Prototypen nach Denert [3, S. 53]

Die Eingrenzung von Prototyping läßt Raum auch für andere Programme während des Entwicklungsprozesses, die weder Prototypen noch Teil des Echtsystems sind. Demonstrationsmodelle, Labormuster, Testprogramme und Simulationsmodelle sind nützliche Bestandteile vieler Softwareprojekte. Pilotsysteme auf der anderen Seite gehen über in eine "Nullversion" des Zielsystems.

Das hier gegebene eingeschränkte Verständnis von (Software)-Prototyp deckt sich mit dem Sprachgebrauch der übrigen Ingenieurwissenschaften.

1.2 Beteiligte

Bei einer Entwicklung mit Prototyping spielen folgende Gruppen von Beteiligten eine Rolle. Der Sprachgebrauch folgt weitgehend Kieback et.al. [7, S.66]:

Anwender sind alle Mitglieder der Organisationseinheit/en, die vom Anwendungssystem direkt oder indirekt betroffen ist/sind. Demgegenüber sind **Benutzer** diejenigen Personen, die das System direkt benutzen, sei es durch Datenein- oder -ausgaben am Bildschirm (**direkte Benutzer**), oder durch das Benutzen von Listen (**indirekte Benutzer**). In der Regel gehören die Benutzer zu den Anwendern. Mitarbeitende Entwickler aus dem Fachbereich sind von ihrer Tagesarbeit teilweise oder ganz freigestellt. Sie werden hier **Anwendervertreter** genannt, obwohl sie auch Benutzervertreter sein können. Vor allem bei buchenden oder bei einigen Auskunftsystemen ist die Zahl der Benutzer so groß, daß man nur "Vertreter" am Prototyping beteiligen kann. Ein Anwendervertreter, der nicht Benutzer ist, ist z.B. der Projektleiter des Fachbereichs, häufig ein Manager. **Entwickler** gehören der DV-Abteilung oder einem Softwarehaus an.

2. Rahmenbedingungen für Prototyping

Wenn Prototyping eine wichtige Technik bei der Kommunikation mit dem Benutzer oder Anwender ist, dann muß man die Rahmenbedingungen betrachten, denen diese Kommunikation unterliegt. Ich sehe folgende grundsätzlichen Einflußfaktoren auf Verlauf, Aufwand und Nutzen der Kommunikation:

(1) Organisationskultur
(2) Qualifikation
(3) Entwicklungsanlaß
(4) Kapazität.

Zu (1): Eine **Organisationskultur** kann **konstruktiv** oder **konfrontativ** sein. Sicher ist, daß überall der Abbau eines konfrontativen Umgangs der Fachabteilungen miteinander angestrebt wird. Man muß jedoch sehen, daß in Zeiten autoritärer Führungsstrukturen über Jahrzehnte die konfrontative Kultur gepflegt wurde unter dem Motto "Konkurrenz belebt das Geschäft". Konstruktiv heißt nicht fraternisierend. Eine Organisation behält ihre Spannung nur durch Interessengegensätze.

Es ist fast zu trivial, es zu schreiben: Prototyping kann nur bei einer konstruktiven Kultur Nutzen bringen.

Zu (2): Die **Qualifikation** der Entwickler und auch der Anwender hinsichtlich Betriebswirtschaft ist oft unzureichend. Trotzdem muß unter dieser Randbedingung entwickelt werden. Häufig ist sogar das notwendige know how für ein zeitgerechtes System in der Organisation überhaupt nicht vorhanden. **Hier würde Prototyping die falschen Anforderungen evaluieren.** Wir haben in diesen Fällen -soweit wir sie erkennen konnten- das know-how von Beratern und Softwarehäusern "importiert". Wichtig ist, für einen Wissenstransfer an die eigenen Mitarbeiter zu sorgen.

Wenn es im übrigen schon DV-Fachleuten immer schwerer fällt, der stürmischen technologischen Entwicklung zu folgen und Mode von Ernsthaftem zu unterscheiden, dann kann man Anwender verstehen, die sich nach jahrelang verschobenen Neuinvestitionen in Software überfahren fühlen. Prototyping kann hier **helfen**, ist aber keine Patentlösung für Lernprozesse.

Zu (3): Der **Entwicklungsanlaß** ist hier im Gegensatz zu dem Bericht von Kieback et.al. [7] in der überwiegenden Anzahl der Fälle eine Ersatz- und keine Neuinvestition von Software. Dies bedeutet, daß es kaum noch Benutzer ohne Erfahrung mit Computern gibt, aber sehr viele, die veraltete Technik und unzulängliche betriebswirtschaftliche Konzepte gewohnt sind. Folgendes Urteil sei erlaubt:

Es ist viel schwerer und ggf. konfrontativer, die eingefahrenen Muster computererfahrener Benutzer aufzubrechen, als mit Anwendern zu kommunizieren, die seit Jahren auf Computerunterstützung **warten**.

Zu (4): Die **Kapazität** in den DV-Abteilungen ist bekanntlich überall zu gering (sog. Anwendungsstau). In den Fachabteilungen ist sie häufig geradezu dramatisch knapp, und zwar nicht an Zahl, sondern an "Köpfen". Es sind ausschließlich die Schlüsselfiguren der Fachbereiche, mit denen man erfolgreich Software entwickeln kann. Genau diese sind aber durch Tagesbetrieb blockiert, da sie überall unentbehrlich sind. Die Folge ist, daß die Kommunikation ständig unterbrochen wird oder Termine verschoben werden. Wir haben in fast allen Projekten Sitzungen in Hotels auslagern müssen, ohne dort über die technische Infrastruktur für Prototyping zu verfügen, z.B. Bildschirmanschlüsse.

Unter diesen Randbedingungen läuft dann ein Kommunikationsprozeß ab, den man in eine **Informationsphase** und eine **Meinungsaustauschphase** zerlegen kann. In der Informationsphase vermittelt der Entwickler dem Benutzer die (standardisierten) Konzepte der Benutzeroberfläche. Diese Phase ist einseitig. In der Meinungsaustauschphase kommunizieren Entwickler und Benutzer bzw. Anwender wechselseitig über Fachinhalte und die jeweiligen Vorstellungen von den zu implementierenden Konzepten [Näheres in: 13, Kap.9].

3. Sechs Projekte in sechs Jahren

Die im folgenden dargestellten Systeme und Projektabläufe umfassen wesentliche Abläufe des Tagesgeschäftes jedes gleichzeitig produzierenden und verkaufenden Unternehmens. Ausnahme ist das erste System, das man nur dann benötigt, wenn man einen großen Außendienst steuern will. Es handelt sich um die Systeme

TOUR: Tourenplanung Außendienst

GAR: Grunddaten Artikel
(Stammdaten, Stücklisten, Arbeitspläne)

PAU: Produktionsaufträge

LGB: Lagerbestandsführung
(Fertig- und Halbfabrikate, Material)

FAKT: Fakturierung

VAU: Verkaufsauftragsabwicklung
(Kundenaufträge und Versand).

3.1 Allgemeines zum Projektablauf

Alle Projekte

- waren Ersatz für veraltete Systeme;
- dauerten mehrere Jahre;
- wurden stufenweise in Echtbetrieb genommen;
- wurden an mehreren Standorten auf einem Zentralrechner entwickelt für den Einsatz an zwei bis drei Standorten;
- waren trotz des Einsatzes von Prototypen geprägt von nachgeschobenen Anforderungen spätestens ab Beginn des Echtbetriebes;
- mußten so eingeführt werden, daß die Primäraufgabe der Benutzer (produzieren, verkaufen etc.) möglichst wenig beeinträchtigt wurde.

Drei Projekte waren verbunden mit einem Wechsel des Zielrechners.

Das folgende Bild gibt grob die Zusammenhänge und die Implementierungsreihenfolge wieder. Unterteilte Nummern kennzeichnen eine parallele Einführung. Die "Platten"-Symbole drücken aus, daß primär ein Datenbestand verwaltet wird, die Kästchen, daß der funktionale Aspekt überwiegt:

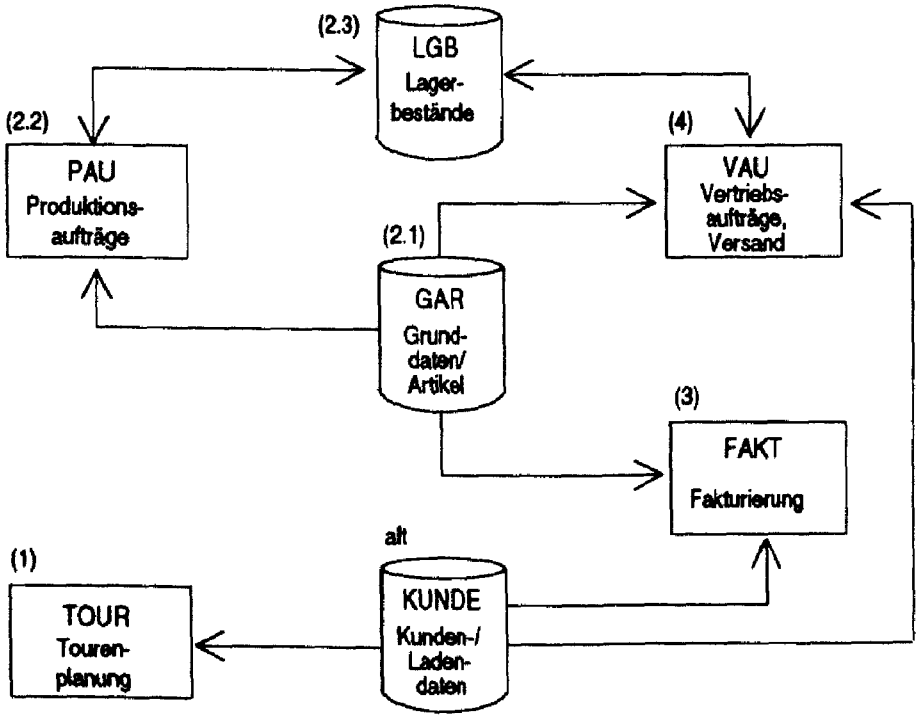

Abb. 3: Abhängigkeiten der Systeme

3.2 Mengengerüste

Die folgende Tabelle zeigt die wichtigsten Eckdaten, die man für eine Beurteilung des Prototyping der Systeme benötigt. Alle Entwicklungsaufwände sind in einer Projekt- (und Wartungs-) Datenbank nachvollziehbar abgelegt. Die Spaltenüberschriften geben einen ersten Eindruck, welche Zusammenhänge zwischen Entwicklungsdauer, Zahl der Beteiligten, Systemgröße und Aufwand bei der Bewertung von Prototyping berücksichtigt werden sollten:

	Dauer (Mon)			EW	AwV	Benutzer		Mas-	Mo-	DB-V	Aufwand	
	Ent	Pil	Ein			dir	ind	ken	duln		Std	MJ
TOUR	24	-	4	4	2	2	50	36	55	23	5.962	4,3
GAR	25	13	7	10	3	8	20	120	368	25	18.900	13,5
PAU	21	13	8+	6	4	16	10				13.808	9,9
HOST								31	154	26		
PC								12	32	49		
LGB	19	13	8+	5	6	40	20	142	282	25	9.563	6,8
FAKT	8	-	5	4	1	-	10	-	19	29	1.259	0,9
VAU	21	3	5+	7	2	30	15	36	111	15	12.457	8,9
Σ								377	1.021	192	61.965	44,3

Tab.2: Mengengerüste

Dauer:

Ent: **Entwicklung** von der Vorstudie bis zur Übergabe an den Anwender in Monaten.

Pil: **Pilotanwendung** als Probebetrieb. Der Benutzer muß alte und neue Anwendung parallel betreiben. Dies ist eine erhebliche Zusatzbelastung.

Ein: **Einführung**; in dieser Zeit erfolgen noch viele Änderungen und Fehlerkorrekturen. Nach Abschluß der Einführung befindet sich das System in der **Wartung**. Ein '+' hinter der Dauer bedeutet, daß die Einführung bei Redaktionsschluß dieses Papiers noch nicht beendet war.

Beteiligte:

EW: Zahl der überwiegend oder maßgeblich beteiligten **Entwickler** incl. Projektleiter. [rechnerische Prüfung EW gegen Aufwand in dieser Tabelle **nicht** möglich! (MJ meist > EW, in einem Fall <)]

AwV: **Anwendervertreter** als Mitglied des Projektteams.

Benutzer: dir= direkte , ind= indirekte. Die Zahl der **Anwender** ist bei einigen Systemen >> AwV+Benutzer.

Software:

Moduln: Zahl ausführbare, separat compilierbare Programme, nicht aber externe Daten-, Maskendefinitionen u.ä. oder systemweit benutzte Moduln.

DB-V: Zahl Datenbank-Views bzw. Dateien (Dateisystem).

Aufwand: kumulierter **Gesamtaufwand** in Stunden: Ew + Pil + Ein bis 30.06.92. Die **Kosten** betragen mehr als 100 DM/Stunde.

3.3 Die Projekte

3.3.1 Tourenplanung Außendienst

Das System wurde in zwei Versionen entwickelt, von denen die erste 18 Monate im Einsatz war. **TOUR1** war die Ablösung eines unwartbaren, batchorientierten Altsystems mit zentralistischer Planung und starr "verdrahteten" Besuchsdistanzen.

Die Forderung nach einer dezentralen Planung hätte einen so gravierenden Umbau des Systems erfordert, daß eine Neuimplementierung als **TOUR2** wirtschaftlicher war. TOUR2 ist seit 2 Jahren in Echtbetrieb.

Verlauf: In TOUR1 und TOUR2 wurde mit dem in [13, Kap.8 u. 11] beschriebenen Werkzeug ein Prototyp der Benutzeroberfläche vor Erstellung der Spezifikation mit den Benutzern durchgesprochen. In TOUR1 wurde er **präsentiert**, in TOUR2 **arbeiteten die Benutzer selbst** am Bildschirm mit dem Prototyp. Dies war erheblich wirkungsvoller. Der aktive Aspekt war eine Folge aus den Erfahrungen in TOUR1.

Bewertung: **Prototyping** bedeutet hier, daß der Dialogteil mit den zwei späteren Benutzern evaluiert wurde. Selbstverständlich wurden auch die Listen mit Vertretern der indirekten Benutzer, 50 Gebietsverkaufsleitern im Außendienst, anhand eines Layout-Entwurfs besprochen. Dies ist kein Prototyping. Ebenfalls war TOUR1 kein Prototyp, sondern eine Version. Ohne ein Prototyp-Werkzeug, das auch Produktionswerkzeug ist (hier eine 4GL), wäre das Projekt sehr wahrscheinlich zu teuer geworden und hätte zu lange gedauert.

3.3.2 Grunddaten Artikel

Artikelstammdaten mit Stücklisten und Arbeitsplänen sind **der** zentrale Stammdatenbestand jedes Industriebetriebes. Er hat ein besonderes Gewicht für die Herstellung

modischer Produkte mit vielen Varianten, bei denen sich halbjährlich zwischen 30 und 70% der Stammdaten gegenüber der Vorsaison ändern. Die vorhandenen zwei Systeme für zwei Werke auf verschiedenen Rechnern genügten den Anforderungen schon lange nicht mehr. Es waren über 600.000 Artikelvarianten gespeichert, da man wegen der vielen Abhängigkeiten und der Undurchschaubarkeit der Systeme kaum noch Daten alter Saisons löschen konnte.

Ziel war die Entwicklung eines möglichst redundanzfreien Systems mit kontrollierbaren Abhängigkeiten. Benutzer waren die für die Datenpflege Verantwortlichen a) in der Produktion und b) im Vertrieb. Das System sollte zunächst in Werk A eingesetzt werden, danach in Werk B.

Verlauf: Die Entwicklung war von Anfang an durch Skepsis der Benutzer und auch einiger Mitarbeiter des DV-Bereiches begleitet. Die Gründe waren genau gegensätzliche: Die Benutzer hatten keine Vorstellung von der Komplexität einer integrierten Stammdatenverwaltung mit mehrfach ineinander verschachtelten Baumstrukturen. Sie hielten das Projekt für überflüssig. Die DV-Mitarbeiter wiederum hatten Bedenken wegen der ihnen sehr wohl bekannten Komplexität. Die erste Benutzerin in Werk A wurde für das Projekt neu eingestellt und kannte solche Strukturen. Mit ihr wurde der **Demonstrations-Prototyp** der Benutzeroberfläche durchgesprochen. Diese Oberfläche verbarg die Komplexität durch logische Sichten. Beim Prototyping mußten Strukturskizzen auf Flipchart verwendet werden, die der Prototyp nicht liefern konnte. Es war zuviel verlangt, daß die Benutzerin die korrekte Abbildung der Strukturen glauben sollte. Erst der fast 3 Monate später verfügbare **Entwurfs-Prototyp** des Datenbankdesigns, ergänzt um einige Listen, konnte die gewünschten Nachweise erbringen und die Benutzerin überzeugen.

Mit diesem Prototyp wurden vor allem die kritisch zu sehenden Zugriffszeiten erstmals getestet; "kritisch" deshalb, weil später fast alle neuen Systeme lesend auf GAR zugreifen würden. Das System wurde problemlos eingeführt, obwohl es immer wieder von indirekt betroffenen Anwendern attackiert wurde.

Bewertung: Bei diesem schwer durchschaubaren System zeigten sich die Grenzen des Prototyping der Benutzeroberfläche. Der Prototyp war nützlich, genügte aber nicht. Für ein anschauliches Prototyping mußte ein erheblicher Teil der echten Funktionalität hergestellt werden. Hier hätte ein Werkzeug für Entwurfs-Prototypen (das es nicht gab!) wenig geholfen, da ein spezifischer Entwurf zu erstellen war. Der hohe Vernetzungsgrad einer Stammdatenverwaltung läßt sich Anwendern nicht durch Prototypen vermitteln. Die Emotionen hätten nur durch intensive Schulung und Demonstration von Listen beseitigt werden können. Hierfür fehlte jedoch die Zeit.

3.3.3 Produktionsaufträge Werk A

In Ablösung und Erweiterung des Altsystems in Werk A sollte eine Produktionsauftragsverwaltung erstellt werden. Ein solches System besteht aus einem zentralen Teil (Logistik/Produktionsplanung) und dezentralen Teilen (Werkstattsteuerung). Es wurde ein **verteiltes Hardware- und Softwaresystem** konzipiert und damit technisches Neuland betreten. Der Logistik-Teil sollte auf dem Host-Rechner in Form der Artikelstammdaten und Lagerbestandsführung, der Werkstatt-Teil auf einem PC-Netz installiert werden. Die Host-Datenbank war ADABAS, die PC-Datenbank DBASE. Die Verteilung bestand darin, daß mangels Verfügbarkeit einer echt verteilten Datenbank zwischen beiden Systemen vollautomatisch Daten per File-Transfer ausgetauscht werden mußten.
Es entstanden mehrere Modelle, zwei Prototypen und ein Pilotsystem:

1. **Modell** Benutzeroberfläche des Systems,
2. **Prototypen** der Benutzeroberfläche Host- und PC-System,
3. **Modelle** (Testprogramme) des Datenbank-File-Transfers (ADABAS <--> DBASE). Dabei sollte ein Produkt verwendet werden, das automatisch aus ADABAS-Dateien DBASE-Dateien erzeugt und umgekehrt,
4. **Pilotsystem** des Gesamtsystems.

Verlauf:

Zu 1 u. 2: Da viele Benutzer direkt beteiligt waren, entstand für das Prototyping jedesmal erheblicher Aufwand durch temporäre Installation von Bildschirmen in Besprechungsräumen. Das Prototyping war belastet durch organisatorische Störungen.
Als nützlicher erwies sich ein vor den Prototypen für die Vorstudie auf einem tragbaren PC erstelltes **Modell** der Benutzeroberfläche, das ohne Aufwand überall mitgenommen und gezeigt werden konnte. Der (verlorene) Erstellungsaufwand war im Nachhinein infolge des hohen Nutzens gering. Das Modell wurde noch während des Pilotbetriebes benutzt.

Zu 3: Es mußte ermittelt werden, ob und wie ein automatisierter File-Transfer realisierbar war. Es mußte vor allem das Ausfallverhalten einer Vielzahl von Hardware- und Betriebssystem-Komponenten getestet werden. Dabei wurden durch die Tests gefunden: 1) Fehler in der gekauften Transfer-Software. Sie wurden vom Hersteller sehr schnell korrigiert. 2) Ausfallkonstellationen, die niemand vorhergesehen hatte.

Zu 4: Das System wurde 13 Monate lang mit stufenweise erweiterter Funktionalität als **Pilotsystem** in einem Probebetrieb parallel zum Altsystem von den Anwendern eingesetzt. Die hierbei entdeckten Schwächen lagen vor allem im technischen Bereich (Antwortzeiten

Host, Anbindung PC-Netz an Host, dezentrale online-Drucker u.a.). Mit Prototypen oder mit Modellen hätte man die Schwächen nicht entdecken können.
Die Benutzer waren am Probebetrieb beteiligt, was ihnen bei Systemfehlern nicht unerhebliche Geduld abverlangte. Die Belastung der Beteiligten sowie der Erklärungsbedarf des DV-Bereiches waren groß.

Bewertung: Da das System nur mit den Ergebnissen von GAR und LGB [vgl. 3.3.4] funktionieren konnte und damit fast 60 Benutzer hatte, wurde großer Wert auf ein intensives Prototyping gelegt. Das Prototyping hatte die Benutzer frühzeitig in die Entwicklung mit einbezogen, dies zahlte sich aus durch eine breite Unterstützung während des Pilotbetriebes. Ohne Prototyping und das Pilotsystem wäre es nicht möglich gewesen, ein so komplexes System in Echtbetrieb zu nehmen. Die Prototypen sicherten die Unterstützung der Benutzer, das Pilotsystem deckte die technischen Probleme auf und diente der Schulung.

3.3.4 Lagerbestandsführung

Ein Bestandsführungssystem ist zentral für jede Produktions- als auch Kundenauftragsabwicklung (Versand). Es spielt in einem integrierten System eine ähnlich fundamentale Rolle wie GAR, mit dem Unterschied, daß es von sehr vielen Buchungen "lebt", speziell bei einem Massenfertiger. Bei schlechten Antwortzeiten wird es zum Flaschenhals für viele Betriebsabläufe. 10.000 - 15.000 Lagerbewegungen sind pro Tag zu bewältigen. Die Spitzenlast beträgt bis zu 200 Buchungen pro Minute (ausgelöst durch automatische Hintergrund-Tasks).

Verlauf: Für das System wurde ein **Demonstrations-Prototyp** erstellt und auf dieser Basis eine schon vorformulierte Spezifikation abgestimmt. Zu Beginn des Pilotbetriebes mit PAU wurden viele bereits verabschiedete Konzepte durch die Benutzer in Frage gestellt, da die tatsächliche Lagerorganisation nicht mit der vorher vereinbarten übereinstimmte. Die Benutzer hatten beim Prototyping komplexe Lagerbelegungs- und Reservierungsstrategien gefordert, die sie im Pilotbetrieb nicht handhaben konnten. Dies war eine Ursache für viele Änderungswünsche, eine andere war, daß Funktionen mit realen Daten einfach nicht handhabbar waren.

Die Entwickler wiederum hatten zuwenig Vorstellungen von der Arbeitssituation der Benutzer. Dies hatte Funktionen mit überflüssigem Komfort und Überforderung der Benutzer zur Folge.

Beide Ursachen führten zu aufwendigen Änderungen am fertigen System (Benutzeroberfläche, Datenbasis und Funktionalität). Die Änderungen waren z.T. instabil

und nahmen sehr viel Zeit in Anspruch. Dies hätte von Projektleiter und Management als Alarmsignal gewertet werden müssen, wurde es aber nicht, da der Fertigstellungsdruck durch die Fachabteilung sehr groß war. Der Echtbetrieb mit größeren Mengengerüsten war geprägt von schwerwiegenden Antwortzeitproblemen. Eine Detailanalyse nach Beginn des Echtbetriebes ergab, daß das System überarbeitet werden mußte.

Die Entwickler hatten mit rudimentären Spezifikationen und ohne Entwurf aus dem Prototyp heraus "durchrealisiert". Es zeichneten sich Designfehler bei Modularisierung und Datenbankentwurf [10] ab. Management und Projektleiter hatten unter Zeitdruck den erfahrenen Entwicklern vertraut und keine Kontrollen vorgenommen.
Das System mußte als neue Version grundlegend überarbeitet werden. (Die Performanceprobleme waren zunächst behelfsmäßig auf Datenbank-Ebene beseitigt worden.)

Bewertung: In diesem Projekt war "evolutionäres Prototyping" in falsch verstandener Weise betrieben worden, nämlich als Freibrief für 'quick and dirty' durch Programmierer der "alten Schule". Da die notwendigen Kontrollen aus Zeitdruck unterblieben waren, wurde die Fehlentwicklung erst so spät bemerkt. Mit Entwurfs-Prototypen hätte man sowohl die Handhabungs- als auch die Antwortzeitprobleme viel früher erkannt. Es wäre gar nicht erst zu den Designfehlern gekommen.

3.3.5 Fakturierung

Die 25 Jahre alte, längst unwartbare Fakturierung mußte ersetzt werden. Das System ist ein der Kundenauftragsverwaltung und dem Versand (=Dialogsysteme) nachgelagertes Batchsystem.

Verlauf: Mit dem Benutzer mußten die Berechnungsverfahren sorgfältig abgestimmt werden, am Rande auch modifizierte Rechnungsformulare. Dies erfolgte mit Hilfe der **Spezifikation. Es gab keine Prototypen**; sie wurden auch nicht für erforderlich gehalten. Das System wurde mit einer Schnittstelle zur alten Verkaufsabwicklung problemlos in Betrieb genommen. Vorangegangen waren intensive Paralleltests mit dem Altsystem. Implementiert wurde in COBOL.

Bewertung: Für reine Batchsysteme braucht man nur dann Prototypen der Benutzeroberfläche, wenn es sich um umfassende Auswertungssysteme handelt. Dann sollte man aber als Werkzeug einen leistungsfähigen Listengenerator einsetzen. Heute wird in der Vatter Gruppe auf dem Host hierzu NATURAL 2 verwendet, wobei Langläufer als Produktionsversion in COBOL neu geschrieben werden. In einem solchen Fall wäre das NATURAL-Programm i.S. unserer Terminologie ein **Simulationsmodell**.

3.3.6 Verkaufsauftragsabwicklung

Eine Verkaufsauftragsabwicklung besteht grob aus der Verwaltung der Kundenaufträge und dem Versand. Da es kurz-, mittel- und langfristige Aufträge gibt, sind komplexe Reservierungsstrategien für Aufträge erforderlich, die mit den Beständen korrespondieren müssen. Da trotz dieser Steuerung bei der Auslieferung Ware physisch fehlen kann, muß der Versand über Funktionen verfügen, entweder alternative Artikel zu liefern oder Teilaufträge zu verwalten. Eine Verkaufsauftragsabwicklung ist betriebswirtschaftlich anspruchsvoll und entsprechend schwierig zu spezifizieren und zu entwerfen.

Verlauf: Es wurde ein **Demonstrations-Prototyp** erstellt und den Benutzervertretern zweier Standorte vorgestellt. Die Gespräche verliefen stockend, weil den Benutzern das Abstraktionsvermögen fehlte, sich hinter den größtenteils leeren oder fiktiv gefüllten Feldern (die berühmte Artikel-Nr 4711 !) etwas vorzustellen. Daraufhin wurde der Prototyp um eine behelfsmäßige Datenbank erweitert und Extraktionsprogramme auf die alte Datenbasis geschrieben. So konnte ein **Prototyp mit Echtdaten** vorgestellt werden, auf dessen Basis ein intensives Prototyping erfolgen konnte. Hierauf basierte die Spezifikation.

Da das System hohe Transaktionsraten vor allem im Versand zu bewältigen hatte, wurde ein **kollektiver Entwurf** [vgl. 7] durch die besten verfügbaren Entwickler und den Projektleiter erstellt, durch **Testprogramme** belegt und noch einmal einem Review durch eine Fremdfirma mit entsprechenden Erfahrungen unterzogen.

Das System ging ohne technische Probleme in Echtbetrieb, Erweiterungen und Fehlerkorrekturen waren unproblematisch. Es wird seitdem iterativ weiterentwickelt und stufenweise im zweiten Werk eingeführt. Es gab lediglich übergangsweise Beeinträchtigungen durch unperformante Lagerbuchungen bzw. Restriktionen der Hardware (zu langsame Platten).

Auf der Ebene der betriebswirtschaftlichen Verfahren hatten die Benutzer zunächst Schwierigkeiten. Sie konnten die harten Regeln der Reservierungsstrategien nicht umsetzen, so daß einige Aufträge nicht termingerecht ausgeliefert wurden. Nach Beendigung dieser Lernprozesse konnten die wirtschaftlichen Vorteile des Systems gegenüber dem abgelösten Altsystem zum Tragen kommen.

Bewertung: Unter dem Aspekt Prototyping ist dies das am günstigsten verlaufene Projekt, es gab allerdings auch weniger Risiken als bei GAR und PAU: Die technischen Risiken waren aus dem Altsystem bekannt, die Entwickler und die Benutzer kannten die Problemstellung. Die Neuerungen konnten den Anwendern trotz einiger Schwierigkeiten vermittelt werden. Eine evolutionäre Weiterentwicklung in Versionen war aufgrund der soliden Architektur unkritisch. Verallgemeinern läßt sich folgendes: Demonstrations-Pro-

totypen sollten grundsätzlich Echtdaten verwenden und keine "Spieldaten", auch wenn der Aufwand höher ist.

4. Gesamtbewertung

4.1 Nutzen des Prototyping

Eine Entwicklung ohne Prototypen nach dem "klassischen" Vorgehensmodell bewegt sich ausschließlich auf der Modellebene. Die Darstellungsform (Texte, Grafiken, mathematische Ausdrücke) ist vom Realsystem verschieden. Das Verhalten des Realsystems und das Verständnis des Benutzers von diesem System werden **theoretisch** antizipiert.

Mit Prototypen wird auf der Ebene des Realsystems gearbeitet, gefolgert und kommuniziert. Dies vermeidet Strukturbrüche, die Aufwand bedeuten und Risiken bei der Umsetzung beinhalten.

Bei der Entwicklung der Systeme wurden die Prototypen der Benutzeroberfläche immer wieder mit den Benutzern oder Anwendervertretern durchgesprochen und angepaßt. Parallel dazu entstand die Spezifikation aus den Gesprächen. Auf diese Weise erstellte Spezifikationen sind knapper, realitätsnäher und werden eher vom Benutzer getragen als ohne Prototyp erstellte. Die Spezifikationen wurden später in Benutzerhandbücher überführt. Diese Entwicklungsmethode ist effizienter als die Durchsprache von Spezifikationen.

Bei VAU zeigte sich, daß die behelfsmäßige Einbeziehung realer Daten das Prototyping noch wirksamer macht, da hierdurch eine bessere Verständigung mit dem Benutzer möglich war. Der Benutzer hatte mit den realen Daten in den Masken einen besseren Bezug zu den Inhalten, die er kannte, auch wenn die **Form** (die Maske) für ihn neu war. Vergleicht man GAR, PAU und VAU, so könnte man schließen: Je risikoreicher ein System ist, desto wichtiger und nützlicher sind Prototypen.

Durch die Verwendung eines Werkzeuges konnten die Dialogsysteme nach dem Prototyping ohne Strukturbruch bis zum Echtsystem weiterentwickelt werden. Diesem Werkzeug liegt eine vielfach bewährte Software-Architektur zugrunde. Dies bedeutet ebenfalls Nutzen durch effiziente Entwicklung.

Vergleichzahlen zur **Messung des Nutzens** wird man bei realen Systemen kaum erhalten, da solche Entwicklungen nie alternativ durchgeführt werden können. Uns ist nur eine von Boehm veröffentlichte Studie bekannt, in der Softwareentwicklung mit und ohne

Prototyping empirisch verglichen wurde. Dies erfolgte mit Studentengruppen unter den Laborbedingungen einer Universität [1]. Danach ist Softwareentwicklung mit Prototyping wirtschaftlicher als ohne.

Es gibt jedoch ein Maß zur Messung der Qualität und damit des Nutzens einer Software im Echteinsatz, den **Wartungsaufwand**. Die Daten müssen pro System festgehalten werden, lassen sich aber sinnvoll frühestens 2 Jahre nach dem ersten Echteinsatz auswerten. Sie stehen in der Vatter GmbH in einer Datenbank zur Verfügung.
Für TOUR1 und TOUR2 wurden 706 Stunden in 3,5 Jahren aufgewendet. Dies ist extrem wenig für ein so komplexes System. Man kann jedoch nicht ein Tourenplanungssystem, das höchstens 2-3 mal im Jahr benutzt wird, mit einer tagfertigen Versandabwicklung vergleichen. Für die anderen Systeme liegen noch keine Wartungsdaten vor.

4.2 Gefahren des Prototyping

Die Gefahren des Prototyping liegen darin, daß die Vorgehensweise mit Prototypen der "urwüchsigen" Softwareentwicklung, d.h. iterative Realisierung ohne Spezifikation und Entwurf näher ist als das "geregelte" Vorgehensmodell des Software Engineering.

Diese "urwüchsige" Vorgehensweise ist u.W. noch sehr verbreitet und liegt auch der menschlichen Ungeduld nahe, schon früh etwas "laufen" zu sehen. Dann ist die Gefahr sehr groß, einen Prototypen mit Zugriff zu Echtdaten iterativ weiterzuentwickeln, als sich noch mit Spezifikation und Entwurf "aufzuhalten". LGB ist in der Industriepraxis kein Einzelfall. Dies gilt besonders, wenn dem Benutzer der Unterschied zwischen Prototyp und Echtsystem nicht deutlich vermittelt wurde. Ein Prototyp kann leicht falsche Erwartungen beim Benutzer wecken.

Wegen der Nähe zu 'quick and dirty' sollte man evolutionäre Entwicklung nicht mit Prototyping vermischen. Es muß eine tragfähige Architektur existieren, bevor man iterativ entwickeln kann.

In keinem Fall dürfen Prototypen eine Spezifikation und damit eine pflegbare Dokumentation **ersetzen**. Dies liefe auf die Wartung undokumentierter Software und damit den Zustand vor Schaffung der Disziplin Software Engineering hinaus.

5. Fazit und Abschluß

Die Fallstudien haben das Einsatzspektrum von Modellen, Prototypen und Pilotsystemen gezeigt. Prototypen werden hier auf diejenigen Programme eingeschränkt, die in der Zielumgebung erstellt sind, also iterativ weiterentwickelt werden können. Dieser Ausschnitt der Entwicklung sollte explizit und kontrolliert als **Prototyping** gehandhabt werden. Andere Programme, die nicht Prototypen oder Teil des Echtsystems sind (**Modelle**), sollten freizügiger verwendet werden, um die Kreativität der Entwickler nicht einzuschränken. Teil- oder Vorabsysteme, die produktiv genutzt werden, sind als **Pilotsysteme** wiederum anders zu benutzen als Prototypen.

Vom Modell zum Pilotsystem nimmt die Anwendung durch den Entwickler ab und durch den Benutzer zu. Durch den hier dargestellten Einsatz von Prototypen kann die Gefahr des Mißbrauchs des Prototyping reduziert werden.

An der Entwicklung der Systeme und damit dem Zustandekommen dieses Papiers waren beteiligt: Meine Kollegen und Mitarbeiter in der Vatter GmbH in Rheine/Westf. und Schongau/Allgäu. Den Gutachtern der Tagung danke ich für wertvolle Hinweise und Anregungen.

Literatur

[1] Boehm, B.W., Gray, T.E., Seewaldt, *T.: Prototyping Versus Specifying: A Multiproject Experiment*,in: IEEE Trans. on SE 10(1984) No.3, pp.290-303

[2] Budde, R., Kuhlenkamp, K., Mathiassen, L., Züllighoven, H. (eds.): *Approaches to Prototyping*. Springer, Berlin - Heidelberg - New York - Tokyo 1984

[3] Denert, E.: *Software-Engineering*. Springer, Berlin - Heidelberg - New York et.al. 1991

[4] Dobertkat, E., Fox, D.: *Software Prototyping mit SETL*, Teubner, Stuttgart 1989

[5] Fittkau, B., Müller-Wolf, H.-M., Schulz von Thun, F.: *Kommunizieren lernen (und umlernen)* (2.Aufl.). Westermann, Braunschweig 1980

[6] Floyd, C.: *A Process Approach to Software Development*, in: Systems Architecture, Proc. of the 6.European ACM Regional Conference, Westbury House 1981, pp.285-294

[7] Kieback, A., Lichtner, H., Schneider-Hufschmidt, M., Züllighofen, H.: *Prototyping in industriellen Software-Projekten*, in: Informatik-Spektrum 15(1992) H.2, S.65-77

[8] Lehmann, M.M.: *Programs, Life Cycles and Laws of Software Evolution*, in: IEEE Proceedings 68(1980) No.9, pp.1060-1076

[9] Maibaum T.S.E., Turski, W.M.: *On what exactly is going on when Software is developed Step-by-Step*, in: 7th ICSE, pp.528-533. Orlando/ Florida 1984

[10] Nagl, M.: *Softwaretechnik: Methodisches Programmieren im Großen*. Springer, Berlin - Heidelberg - New York et.al. 1990

[11] Pasch, J.: *Dialogischer Software-Entwurf*, Dissertation TU Berlin, FB Informatik 1991

[12] Pomberger, G., Pree, W., Stritzinger, A.: *Methoden und Werkzeuge für das Prototyping und ihre Integration*, in: Informatik Forsch.Entw. 7(1992) H.2, S. 49-61

[13] Spitta, Th.: *Software Engineering und Prototyping*. Springer, Berlin - Heidelberg - New York et.al. 1989

[14] Spitta, Th.: *Prototyping: Ziele und Möglichkeiten - Kosten/ Nutzen und Relevanz in der anwendungsorientierten Systementwicklung*, in: CW/CSE 85, Software-Forum 1985, S.493-533