

# Foxtrott



Lesen Sie, ob ein Standard-CMS oder eine selbst entwickelte Lösung für Sie mehr Sinn macht!

Von Dieter Gust, Leitung Forschung und Entwicklung bei der itl AG, München

**Der Fuchs zeigt, wie der Foxtrott getanzt wird:  
Langer Schritt, kurzer Schritt, wie die Fuchsspur  
im Schnee. Genauso sollten Sie ein CMS-Projekt  
dirigieren. Schrittweise! Dabei darf nicht  
vergessen werden, dass der Anwender führt!**

Content Management Systeme (CMS) bleiben auf dem Vormarsch. Die Zahl der Unternehmen, die in den vergangenen Jahren ein solches System implementierten, ist beträchtlich. Getragen vom Bestreben, die eigenen Bedürfnisse exakt abzubilden, entwickelte man dabei vielfach individuelle Lösungen auf der Grundlage von SGML/XML. Denn schließlich – so dachte man – weiß man selbst am besten, was die eigenen Redakteure wünschen, und ein vorgefertigtes CMS kann nie zu 100 Prozent den gewünschten Funktionsumfang bieten. Oft jedoch fiel die abschließende Bilanz ernüchternd aus, weil die

Anwenderbedürfnisse entgegen dem ursprünglichen Ansatz nicht zur Genüge umgesetzt wurden und die Kosten erheblich waren. Ein Standardsystem stellt so gesehen mit Sicherheit das geringere Risiko dar. Allerdings muss man sich bei so einem System immer die Frage stellen, ob es die Anwenderbedürfnisse auch ausreichend abdeckt. Die Erfahrungen aus der Praxis lehren vor allen Dingen eins: Implementierung in kleinen Schritten!

Die Crux bei der Einführung eines jeden Softwaresystems besteht darin, dass man zunächst eine philosophische Grund-

## CHECKLISTE ZUR EINFÜHRUNG EINES CONTENT-MANAGEMENT-SYSTEMS

1. Warum soll das System eingeführt werden? Was genau sind die zentralen Anwenderbedürfnisse?
2. Wie viele Redakteure sollen das System nutzen? Sind diese hinreichend beim Erfassen der Anforderungen eingebunden?
3. Welche „CMS-Kultur“ herrscht im Unternehmen vor? Genügen die Funktionalitäten eines Standardsystems oder ist eine Eigenentwicklung sinnvoll?
4. Welches Systemhaus ist der geeignete Partner für Entwicklung und Implementierung? Welche Referenzen haben die in Frage kommenden Systemhäuser aufzuweisen?
5. Wie gestalten sich Lasten- und Pflichtenhefte? Vermitteln sie ein konkretes Bild oder sind sie zu allgemein gehalten? Stehen „Use Cases“ im Mittelpunkt?
6. Welche Basisarchitektur des zu implementierenden Systems liegt vor? Wie gestaltet sich das Geflecht aus Modulen, Dokumenten und Prozessen?
7. Lassen sich die zentralen Anforderungen an das System in Form eines Pilotprojektes abdecken? Muss man gegebenenfalls die Konzeption revidieren, damit ein Pilotprojekt möglich ist?

frage beantworten muss: Wählt man ein im Markt erhältliches Standardsystem oder entwickelt man selbst eine maßgeschneiderte Lösung? Meist sehen Entscheider in solchen Neueinführungen einen zentralen Einschnitt, dessen Aufwand sich durch einen potenzierten Nutzen in Form von Vereinfachungen, optimierten Prozessen und vor allem mehr Komfort niederschlagen muss. Vielfach geht man aufgrund dessen dazu über, sich seine

eigene individualisierte Lösung quasi „mundgerecht“ selbst entwickeln zu lassen. Denn – so die Über-

legung – die höheren Investitionen rechnen sich langfristig.

### Das Wasserfallmodell

Die Einführung eines CMS bildet dabei keine Ausnahme. Klassisch geht man hierbei nach dem so genannten Wasserfallmodell vor. Grundgedanke dieses Konzepts bildet die Annahme, dass alle Prozesse bis zum Produktivstart fließend ineinander übergehen. Zunächst erstellt man dementsprechend im Rah-

men einer Ist-Analyse ein Lastenheft mit einem Anforderungskatalog. Dieses bildet die Grundlage für das Pflichtenheft sowie das Systemdesign. In einem durchgängigen Prozess will man so das entworfene System bis zum Abschluss immer konkreter modellieren.

So schlüssig dieser Ansatz auf den ersten Blick auch erscheinen mag, so selten geht er in der Praxis auf. Denn meist sind fachorientierte Lastenhefte zu allgemein gehalten. In rudimentären Illustrationen mit Tonnen und Pfeilen finden sich so etwa Anforderungen wie „aus Modulen sollen sich Dokumente erzeugen lassen“. Anhand der Lastenhefte erstellt man datentechnisch orientierte Pflich-

tenhefte. Diese reflektieren zwar eine Vielzahl neuer Funktionen, aber enthalten meist weder eine Benutzerschnittstellen-Modellierung noch ausgearbeitete Use Cases (Anwendungsfälle). Denn Systemspezialisten sind in der Regel bestrebt eine funktionale Basisarchitektur zu gewährleisten, bevor sie sich Gedanken über Buttons und Menüs machen. In dieser

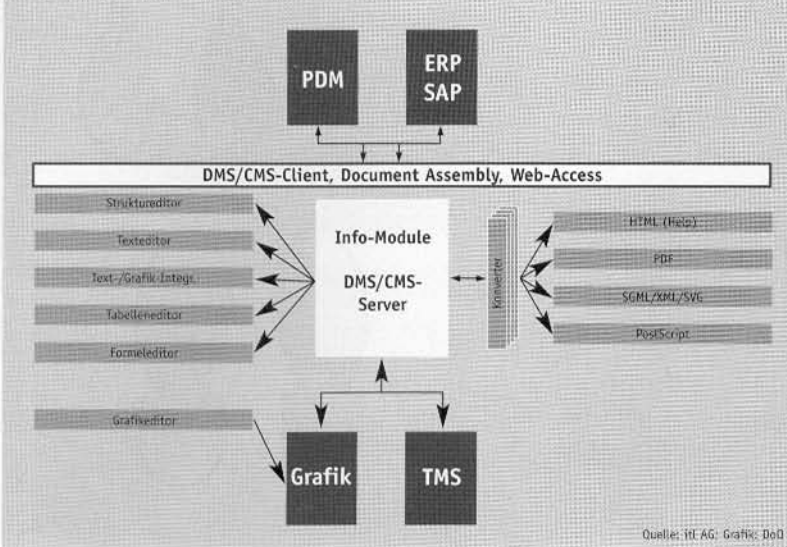
Phase können auch die Anwender nicht unbedingt als Hilfestellung fungieren, weil sie meist nur über die Benutzerschnittstelle, also durch praktisches Probieren, die Tauglichkeit einer Funktion beurteilen können.

So sind die Entscheider gezwungen, Pflichtenhefte „abzunicken“, ohne zu wissen, wie gut das System die Anwenderbedürfnisse wirklich abdeckt. Die böse Überraschung ist damit jedoch vielfach vorprogrammiert, wenn das System realisiert ist. Erst jetzt zeigt sich nämlich der mitunter eklatante Unterschied zwischen einer neuen Funktion und dem für Redakteure relevanten Anwendungsfall.

Natürlich versucht man, dieses Dilemma umgehend zu korrigieren, indem man Änderungsaufforderungen (Change Request) formuliert. Dies führt jedoch oft nach dem Schneeballprinzip zu immer weiteren Änderungswünschen, was sowohl teuer als auch gegen die Logik des Lasten-/Pflichtenheft ist.

Lässt sich das Pflichtenheft im Nachhinein als „qualitativ untauglich“ bewerten, kann man die betrauten Systemhäuser zur kostenlosen Nachbesserung „verdonnern“. Letztere sind al-

## CMS-REFERENZMODELL



lerdings in dieser Hinsicht nicht selten bereits gebrannte Kinder und versuchen, dem frühzeitig gegenzusteuern. Hierzu bedient man sich eines Pilotprojektes, das man quasi dazwischen schiebt, um die neue Umgebung möglichst genau widerzuspiegeln. Außerdem nehmen neuerdings Pflichtenhefte auch den Begriff „Use Case“ auf. Ob darin aber wirklich Anwendungsfälle aus Anwendersicht beschrieben sind, ist eher fraglich. Denn nicht die Anwender, sondern die Entwickler kommen in der Regel bei deren Dokumentation zum Zuge.

Und so gilt trotz aller Verbesserungsversuche im Projektablauf: Die Regel ist, dass die „Systemfindungsphase“ länger als geplant dauert, weil weit mehr Funktionalitäten als ursprünglich angenommen nachzuimplementieren sind. Zudem sind Teile des Systems meistens dann fehleranfällig, wenn man sie im Zusammenspiel mit einer hochkomplexen Dokumentengenerierung aus einer Datenbank nutzt. Dementsprechend wird das Projekt, will man es erfolgreich beenden, teurer als erwartet.

#### Kleinere Brötchen backen

Die Dokumentenproduktion eines Unternehmens muss sich schritt-

weise ändern. Klassische Desktop-Publishing-Lösungen lassen sich nicht in einem Schritt zu einer voll automatischen durch SAP angestoßenen Dokumentenerzeugung aus XML-Modulen überführen. Diesen Anspruch vermag weder ein System noch ein Projekt in einem überschaubaren Rahmen von unter einem Jahr zu erfüllen.

Pilotprojekt oder Prototyp müssen dennoch bereits eine Umgebung darstellen, die sich vom System-Anwender produktiv nutzen lässt. Andernfalls droht eine frappante Lücke zwischen der Anwendervision, dem Pflichtenheft und der Realisierung.

Verfehlt ein Prototyp beziehungsweise ein Pilotprojekt diesen Produktivzustand, muss man das Projektziel entsprechend modifizieren. Selbstverständlich kann auch das anvisierte System von Grund auf nicht geeignet sein.

Die folgenden Stufen der Weiterentwicklung vom Prototypen zum endgültigen System muss für die Anwender in überschaubaren Schritten erfolgen und als eigene Teilprojekte definiert sein. Moderne Softwareentwicklung kann im Grunde nur dann erfolgreich sein, wenn man sie als iterativ-inkrementellen Prozess verfolgt. Sie besteht also streng genommen ausschließlich aus permanenten Analysen, „lebendigen“ Konkretisierungen, die immer im Fluss bleiben, und „kontinuierlichem Re-Development“. Und eigentlich alles ist demnach „Change Management“.

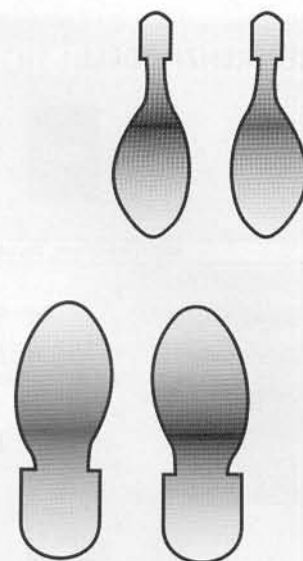
Ob dieser radikale Paradigmenwechsel immer funktioniert, lässt sicher in Frage stellen. Eine Alternative besteht aber lediglich in einem fertigen Standardsystem, das alle Anforderungen abdeckt.

#### Weg von semantisch überladenen Dokumenttypen-Definitionen

Eine solide Basisarchitektur steht häufig im Mittelpunkt bei der Planung eines „perfekten“ CMS, denn man will ja kein System implementieren, das alsbald in sich „zusammenstürzt“. Dabei verlegt man sich jedoch zu oft darauf, Dokumenttypen akribisch zu pla-



Dieter Gust, Leitung Forschung und Entwicklung bei der itl AG, München



## GLOSSAR

**Wasserfallmodell:** Entwicklungsmodell, wonach alle Entwicklungsetappen linear ineinander übergehen. Erst am Ende des Projekts führt man alle Teilprojektstränge zusammen.

**Iterativ-inkrementelle Entwicklung:** Entwicklungsansatz, der eine modulare Projektstruktur propagiert. Jedes Projekt gliedert sich in Teilprojekte, die auf mehreren hierarchischen Ebenen das Gesamtprojekt ergeben. Der Fortgang eines (Teil)Projekts durchläuft die verschiedenen Hierarchieebenen, wobei man erst auf die jeweils nächste Ebene übergeht, wenn alle Teilprojekte einer Ebene zufrieden stellend abgeschlossen sind und sich miteinander verbinden lassen.

**Use Case:** Anwendungsfall – soll die tatsächlichen Anwendungen eines Systems beschreiben, um dann auf dieser Basis die neuen Funktionalitäten zu entwickeln.

**Re-Development:** Revision der Datenstruktur eines Systems wegen bedeutsamer Änderungen der Funktionalität.

**Change Management:** Methoden sowie Verfahren zum Planen, Vorbereiten und Durchführen von Änderungen.

**Change Request:** Änderungswünsche, die von den bisherigen Projektanforderung abweichen.

**DTD:** Document Type Definition. In einer DTD lässt sich mittels einer formalen Grammatik bestimmen, welche Elementtypen innerhalb eines XML-Dokuments erlaubt sind, in welchen Strukturen sie festzulegen sind und welche Attribute die einzelnen Elemente haben.

**SGML:** Standard Generalized Markup Language. Standardisierte Sprache zum Codieren elektronischer Dokumente. Basis für HTML und XML.

**XML:** Extended Markup Language. Mit XML lassen sich Datenformate definieren. XML erlaubt unter anderem eine bessere Website-Gestaltung. Es beschreibt die Datendefinition, so dass sich Anwendungen darauf beziehen können.

**Astoria CMS:** Dokumentenmanagement-System auf der Basis von XML und SGML. Hersteller Astoria Software, in Europa vertreten durch DigitalML.

**Poet CMS:** Objektorientiertes Datenbanksystem von Poet Software.

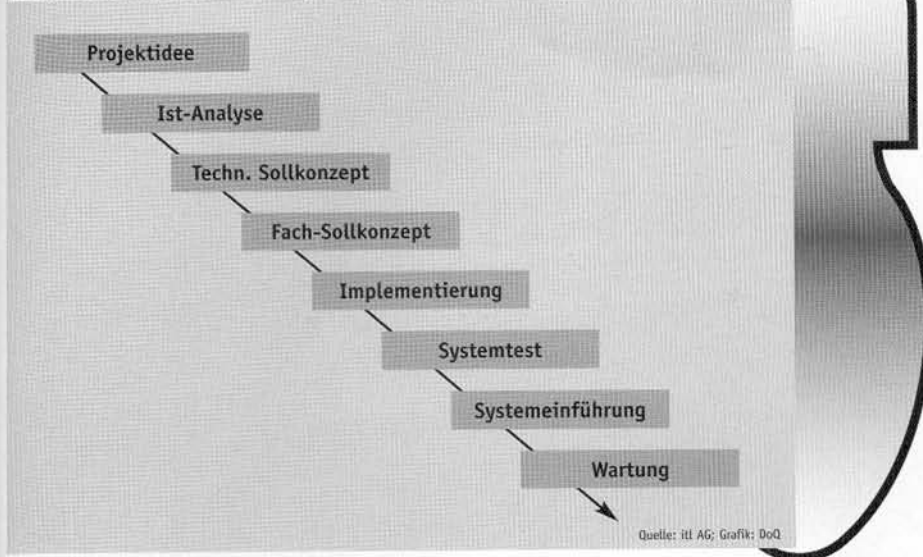
**TMS:** Translation Memory System. Ein System, das beim Übersetzen von in sich ähnlichen Texten zum Einsatz kommt. Es erkennt wiederkehrende Textpassagen und bietet eine vorgefertigte Übersetzung an.

nen, um sich gegen alle Eventualitäten abzusichern. So entstehen gigantische Document Type Definitions (DTDs), die dem System die Luft abschnüren und es unflexibel machen. In der Konsequenz heißt das in erster Linie eins: Weg mit den Mammut-DTDs!

Einen möglichen Ausweg weisen Systeme, deren Datenbasis sauber formulierten XML-Code enthält. Bei diesen Systemen parst man die Daten nicht gegen eine DTD, damit werden sie beweglicher und überschaubarer. Die Forderung nach schlankeren DTDs ist vor allem für solche Lösungen problematisch, die auf Systemen wie Astoria oder Poet-CMS basieren. Deren Ansatz legt ein Datenmodell nahe, das auf einer durchgestylten alle Aspekte berücksichtigenden DTD beruht. Das bedeutet unter anderem das Abbilden der Produktstrukturen, das Attributieren gemäß der Produktdatenorganisation sowie das Berücksichtigen aller möglichen Medien bei der Dokumentausgabe.

So gut der Ansatz einer stabilen Systemarchitektur mithilfe von detaillierten DTDs theoretisch auch sein mag, Hindernisse begegnen ihm praktisch von Anfang an. Gerade in frühen Projektphasen sind Anwender nämlich damit überfordert, alle notwendigen Informationen zu liefern, die derartige

## DAS TRADITIONELLE WASSERFALLMODELL



DTDs voraussetzen. Dies gilt insbesondere im Hinblick auf die geplante Modulbildung. Es ist also von vornherein fraglich, ob dieses Vorgehen zum Erfolg führen kann.

Andererseits ist auch das Abrücken von einer kompletten Datenmodellierung innerhalb der DTD kein Allheilmittel. Denn damit läuft man Gefahr, ein System zu planen, das irgendwann an seine Grenzen stößt. Kleinere Lösungen zu bevorzugen geht also eindeutig einher mit dem kalkulierten Risiko, eine eingeführte Systemlösung wieder aufgeben zu müssen. Diese Gefahr kann man letztlich aber auch nicht mit einer Mammut-DTD umgehen.

Wie aber lässt sich das Problem lösen? Eine Möglichkeit besteht darin, die Metadatenmodellierung und die Elementmodellierung in einer eher flachen DTD logisch zu trennen. XML-SGML-Spezialisten werden dies möglicherweise mit einem Kopfschütteln quittieren, aber dieser Weg wird der sich ständig wechselnden Praxis besser gerecht.

**Standard-Systeme unter der Lupe**  
Was ein CMS leisten muss, ergibt sich in letzter Instanz aus den Anforderungen der Redakteure. Unternehmen, die gerade die Implementierung eines CMS erwägen, sollten sich aber auf jeden

Fall auch mit Standard-Systemen auseinander setzen. Je nach Ausgangslage können sie sicher als Referenzimplementierung fungieren.

### Schritt für Schritt ist besser

Es bleibt dabei: Der Schlüssel für ein erfolgreiches Softwareprojekt ist ein iterativ-inkrementeller Ansatz, der einen Schritt nach dem anderen macht und jeden Schritt auf seine Richtigkeit hin überprüft. Von Beginn an muss man viel mehr als bisher üblich von Use Cases anstatt von Funktionen sprechen. Dies trifft auch für das Erweitern eines bestehenden Systems oder einer Systemlösung zu.

Will man ein System erweitern, muss man die Anforderungen beschreiben. Sind die Anforderungen fixiert und umgesetzt, muss man sie über Praxistests verifizieren. Dies gilt umso mehr, da sich Anwender ihre Use Cases in der Regel erst durch das konkrete Tun bewusst machen. Ein ausgereifter und produktiv nutzbarer Prototyp ist also unumgänglich.

Vielfach glaubte man gerade bei Erweiterungen, durch geeignete Schnittstellen zum Erfolg kommen zu können. Man wollte so quasi weitere vorgefertigte Module an existierende Systeme einfach „anflanschen“. Doch weit ge-

fehlt: Das Beispiel „wir bieten eine Schnittstelle zum System „XY“ besagt praktisch nichts. Dass sich Daten für die Übernahme in ein anderes System ausspielen und wieder einspielen lassen, ist mehr oder weniger banal.

Stattdessen gilt es, das gesamte Beziehungsgeflecht zwischen Modulen, Dokumenten und den Prozessen der Datensynchronisation sowie der möglichen Weiterarbeit an den Modulen detailliert zu betrachten.

### Fazit

Die Entscheidung, welchen Ansatz man bei der Implementierung eines CMS verfolgt, hängt maßgeblich von den Bedürfnissen der Anwender ab.

Jedes Standardsystem bietet dabei im Grund nur eine Good-Enough-Variante, die nicht zwangsläufig die richtige Lösung sein muss. Der entscheidend minimierte Risikofaktor jedoch macht ein Vorgehen auf dieser Grundlage durchaus für viele Unternehmen zu der sinnvolleren Option.

Wer aber dennoch den Schritt hin zur Entwicklung nach Maß wagt, sollte sich vom traditionellen Wasserfallmodell verabschieden und einen von Use Cases bestimmten schrittweisen Ausbau seines CMS wählen. 