The CoreMedia Process for Content Applications

Martin Pakendorf
CoreMedia AG, martin.pakendorf@coremedia.com

Zusammenfassung

Die CoreMedia AG hat aus den Erfahrungen mehrerer Projekte mit dem Content Management System „CoreMedia Publisher" einen Software Engineering Prozess für die Anforderungen von Internet-Projekten gewonnen. Der CoreMedia Prozess adressiert mehrere Aspekte: den Umgang mit unklaren fachlichen Anforderungen, kurze Realisierungszeiträume, hohe Last und die Integration mehrerer Standardprodukte. Die Verwendung von Architekturmustern ist ein hilfreiches Instrument für die Planung von CoreMedia-Projekten und ein zentrales Element des Vorgehensmodells, das sowohl iterative als auch nicht-iterative Aktivitäten beinhaltet.

Schlüsselwörter: Content Management, Software Engineering Prozess, Architekturmuster

Umfeld und Positionierung der CoreMedia AG

Die CoreMedia AG entwickelt und vertreibt als Produktanbieter ein Content Management System (CMS), den *CoreMedia Publisher*, für den Betrieb von high-end Internet Sites, wie z.B. www.t-online.de. Dieses Standardprodukt wird in Implementierungsprojekten von Realisierungspartnern wie Cap Gemini Ernst & Young, CSC Ploenzke, Skillberry und anderen mit Unterstützung von CoreMedia Beratern auf Kundenbedürfnisse zugeschnitten und eingeführt. Für die Durchführung derartiger Projekte verfügt die CoreMedia AG über ein Vorgehensmodell, welches allen CoreMedia-Partnern zur Verfügung gestellt wird. Dieses Vorgehensmodell soll im folgenden vorgestellt werden.

Projekte für Content Applications

Ein Content Management System dient im wesentlichen dazu, digitale Inhalte (*Content*[1]) für Internet-Auftritte zu verwalten. Je nach Ausprägung und Historie der Systeme unterschiedlicher Hersteller umfasst die Funktionalität dieser Systeme mehrere oder alle Stufen der Verarbeitungskette eines Redaktionssystems, eines Content-Verwaltungsssystems und eines Content-Delivery Systems zur Erzeugung von Webseiten. Die Systeme verschiedener Hersteller beruhen auf unterschiedlichen Technologien von HTML-Schablonen mit einer einfachen dateibasierten Contentverwaltung bis hin zu objektorientierten, datenbankgestützten, skalierbaren und hochverfügbaren Architekturen für hochgradig aktualisierte Sites.

Schon eine „einfache" nachrichtenintensive Website benötigt heutzutage ein komplexes Anwendungssystem aus CMS, Newsfeeds, Datenbanken und Webservern. Reine Informations- Websites mit werbefinanzierten Präsentation von Inhalten gehen in komplexe Internet-Anwendungen über, zum Beispiel in Content-Commerce-Community Portale, in denen redaktionelle Inhalte, Community-Funktionen (Chat-Systeme, Newsforen) und Ecommerce-Systeme über einer einheitlichen Oberfläche Anreize zum Kaufen bieten sollen. Desweiteren müssen neue Endgeräte (WAP-Telefone und PDA's) in die Oberfläche eingebunden werden so dass in Zukunft HTML-Websites eine von vielen Interaktionspunkten mit komplexen Anwendungen darstellen werden. Sogenannte *Multi-Touchpoint* Systeme [1] werden längerlaufende Transaktionen über eine Vielzahl von Endgeräten und Oberflächen anbieten. Diese Anwendungen benötigen ein großes Volumen an aktuellem Content verschiedener Quellen, weshalb sie im folgenden als *Content Applications* bezeichnet werden.

Best-of-breed Content Applications sind um zusätzliche funktionale Module unterschiedlicher Anbieter erweiterte CMS-Systeme, denn in der Regel deckt kein Produkt eines einzigen Herstellers die gesamte Bandbreite der benötigten Funktionalität mit ausreichender Qualität und Tiefe ab. Ecommerce-Produkte, Community-Systeme, „intelligente" Suchmaschinen, Customer Relationship Management (CRM) Systeme, Personalisierungssysteme und bestehende Back-End-Systeme müssen in Projekten mit einem CMS integriert werden, das meistens die führende integrative Komponente derartiger Architekturen darstellt.

Diese Integrationsanforderungen erhöhen die Schwierigkeiten für Internet-Projekte, die im Gegenzug

[1] Im vorliegenden Artikel werden Industrie- und marktübliche Anglizismen bewusst nicht eingedeutscht, damit der Bezug zum neuen Marktsegment der Content Management Systeme erhalten bleibt.

zu klassischen IT-Projekten folgende zusätzliche Hürden besitzen, die in Kombination bewältigt werden müssen:

- Hoher Zeitdruck für Projektergebnisse
- Unklare funktionale Anforderungen
- Unberechenbare Lastanforderungen
- Hohe Sichtbarkeit der Ergebnisse
- Einbindung von Design-Agenturen
- Integration potentiell mehrerer Produkte
- Schnell entwickelnde technologische Basis

Anhand von mehreren erfolgreichen Projekten für unterschiedlichste Content Applications hat die CoreMedia AG einen allgemeinen Software Engineering Prozess für die Projektdurchführung entwickelt, der für Content-intensive Projekte Art geeignet ist. Dieser Prozess ist auf die Besonderheiten von Internetprojekten zugeschnitten. Ein wesentliches Merkmal des Prozesses ist die Orientierung an pragmatischen Architekturmustern, wie im folgenden erläutert wird.

Grundkonzepte der CoreMedia Content Applications

CoreMedia Content Applications sind Internet-Anwendungen, die auf dem CoreMedia Publisher aufgebaut sind. Die Architektur des CoreMedia Kernprodukts ist vereinfacht als UML-Komponentendiagramm in Abbildung 1 dargestellt.

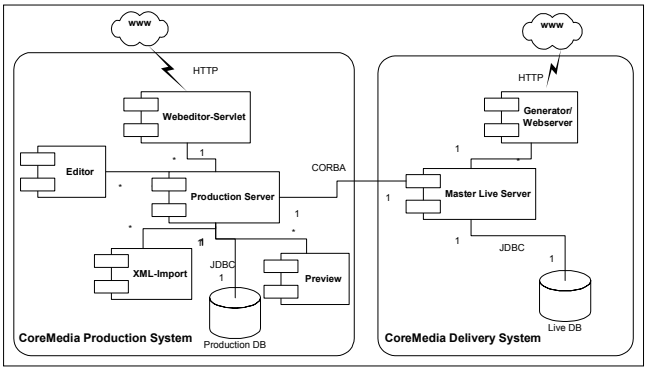

Abbildung 1 Die CoreMedia Systemarchitektur

Das System besteht aus zwei unterschiedlichen Servertypen: einem internen Produktionsserver (CoreMedia Production System) für Inhalte in Bearbeitung und einem Live- bzw. Delivery- System bestehend aus mehreren externen Live-Servern[2] und Generatoren. In dem Delivery-System werden freigegebene Inhalte für die Darstellung im Internet vorgehalten. Inhalte werden durch ein Redaktionssystem und durch Importprozesse in das System übernommen. Erst durch explizite Publikationen gelangen Inhalte vom Produktionssystem auf das Livesystem.

Konform mit einer Mehrschichten-Architektur werden in einer Schicht für Business-Logik und Präsentations-generierung Inhalte in Zielformate (meist HTML) durch Java Server Pages transformiert. Dies findet in den CoreMedia Generatoren statt, von denen aus Skalierungsgründen mehrere Instanzen eingesetzt werden können. Sämtliche Komponenten sind auf unterschiedliche Rechner verteilbar. Eine detaillierte Beschreibung der Architektur findet sich in [2].
Im Kontext des CoreMedia Publishers besteht Content nicht aus technischen Artefakten wie fertige HTML-Seiten, sondern aus beliebig strukturierbaren, multimedialen Inhalten unabhängig von der späteren Verwendung. Diese Inhalte werden üblicherweise für die Generierung von HTML- oder WML-Seiten verwendet, es gibt allerdings auch Einsatzbeispiele für andere Formate wie NITF-SGML (News Industry Text Format), PDF (Portable Document Format), MHP (Multimedia Home Platform) und natürlich XML.
Das CoreMedia System (und dementsprechend auch der CoreMedia Prozess) erzwingt eine starke Strukturierung des Contents in einem Contentmodell, das als persistentes Schema der in einer Datenbank zu speichernden Inhalte dient. Die starke Ausrichtung auf Contentstrukturierung ist ein zentrales Element der Architektur und stellt laut dem Forrester Report[3] eines der wesentlichen Faktoren für den erfolgreichen Einsatz von CMS-Anwendungen dar.

Entwurfsmuster für Systemarchitekturen als Orientierung für Prozesse

Entwurfsmuster für Systemarchitekturen sind für die schnelle Umsetzung von CMS-Projekten wesentlich, denn in diesen sind Erfahrungen zusammengefasst, die nicht in jedem Projekt erneut gemacht werden müssen. Zusätzlich bieten sie die Möglichkeit, Prozesse zu strukturieren, denn sie stellen ein Gerüst für Projektaktivitäten dar, die in einem Prozess und letztlich in einer Projektplanung abgebildet werden müssen. Daher ist der CoreMedia Prozess, ähnlich wie der Rational Unified Process [4] stark an Architekturbetrachtungen ausgerichtet, was durch den Einsatz eines festen Produkts begünstigt wird, denn es bilden sich aus diversen Projekten schnell leistungsfähige Architekturmuster für das Produkt heraus. Allerdings sind auch unabhängig von dem Einsatz eines konkreten CMS-Produkts viele der geschilderten Ansätze übertragbar.

[2] In der Abbildung ist die Grundkonfiguration mit nur einem Live Server dargestellt.

Architekturmuster bzw. deren Grundkonstrukte sind wesentlicher Bestandteil des CoreMedia Produktkerns:

- Ein zentrales strukturiertes Contentmodell bildet die Grundlage für die zukunftssichere Verwaltung von Content. Zusammen mit den Datenbeständen externer Systeme stellt es einen Teil eines systemübergreifenden Domänenmodells dar, das sowohl das Contentmodell als auch die Daten- bzw. Klassenmodelle weiterer Systeme umfasst.
- Contentströme und Contentverwendungsarten stellen den Weg vom Content externer Quellen ins System und aus dem System heraus dar.
- Ein verteiltes Ereignismodell erlaubt die flexible Anbindung externer Systeme, die bei Content-Änderungen benachrichtigt werden. Dieses aus der GUI-Programmierung als Observer-Pattern[5] bekannte Entwurfsmuster ist, vergröbert auf Architekturebene, der wesentlichen Schlüssel zur Integrationsfähigkeit mit externen Systemen.
- Eine Trennung von Produktionsumgebung und Live-System legt durch die Architektur redaktionelle Abläufe (Content-Staging), Konsistenzprüfung des Contents, Performance-Entkopplung aber auch das Ausfallverhalten des Gesamtsystems fest und ist für high-end Content Applications erforderlich.

Die Grundkonstrukte von Systemarchitekturen für Integrationsszenarien sind das Contentmodell, das Ereignismodell und die Content-Austausch-Verfahren.
Für häufige Integrationszenarien wie Personalisierung oder Content-Commerce-Integration verfügt CoreMedia AG über einen wachsenden Katalog an Standard-Architekturmustern. Vollkommen neue Anforderungen lassen sich auf die genannten Grundkonstrukte der CoreMedia Architekturmustern abbilden, so das auch für bisher unbekannte Systeme (z.B. neue CRM-Produkte) meist frühzeitig Integrationsarchitekturen gefunden werden können. Bevor Architekturmuster für Integrationsszenarien detailliert vorgestellt werden, wird im folgenden Abschnitt der wichtigste Bestandteil der Architekturmuster und dementsprechend auch der Content Applications beschrieben: das CoreMedia Contentmodell.

Das Contentmodell als Kern einer Content Application

Aus Sicht einer Architektur für CoreMedia Content Application stellt das frei definierbare, multimediale Contentmodell den wichtigsten Aspekt dar. Der CoreMedia Publisher setzt das Contentmodell automatisch in Datenbankstrukturen für sämtliche multimediale Inhalte um und ermöglicht die Bearbeitung des Contents auch ohne zusätzliche Editor-Programmierung im CoreMedia Redaktionssystem. Unabhängig vom Einsatz eines konkreten CMS-Produkts bildet die saubere Modellierung des Contents die Grundlage für die dauerhafte Speicherung digitaler Firmenwerte und, noch wesentlicher, den entscheidenden Faktor für die spätere Verwendung des Contents (z.B. Präsentation in mehreren Zielformaten).
Für die Contentmodellierung erlaubt der CoreMedia Publisher die Definition von multimedialen Datentypen (oder Dokumenttypen) durch die Aggregation von Basisdatentypen (String, Integer, Date, SGML/XML-Text, MIME-codierten BLOBs) und durch die Assoziation zu anderen Typen und die Vererbung von Dokumenteigenschaften. Auf diese Weise kann ein objektorientiertes Contentmodell ausgedrückt werden.
In CoreMedia Projekten werden UML-Klassendiagramme und Objektdiagramme genutzt, um ein Contentmodell auszudrücken. Java Server Pages dienen zur Darstellung von Objekten in Zielformaten (z.B. HTML). Das System unterstützt einen objektorientierten Aufrufmechanismus für die Ausführung der Java Server Pages, die als *Methoden der Dokumentobjekte* betrachtet werden (im Gegensatz zur herkömmlichen Sicht als prozedurale serverseitige Skripte).

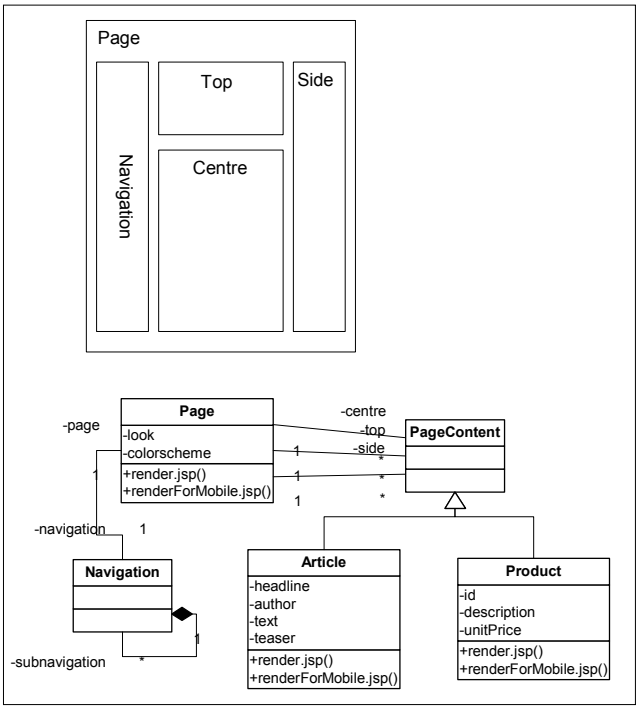

Abbildung 2 Beispiel eines CoreMedia Contentmodells

Ein Beispiel für ein objektorientiertes Contentmodell ist in Abbildung 2 angedeutet, die abgebildete HTML-Seite soll aus Instanzen des Klassenmodells aufgebaut werden.

Ein *Page*-Objekt verweist auf eine Navigationsstruktur und mehrere *PageContent*-Listen, aus denen jeweils die Bereiche „Top“, „Centre“ und „Side“ gefüllt werden, und zwar sowohl mit *Article*-Objekten als auch mit *Product*-Objekten. Verschiedene Java Server Pages übernehmen die Darstellung in ein Zielformat, z.B. *Page->render.jsp*, *Article->render.jsp* und *Product->render.jsp* für die HTML-Erzeugung. Durch die objektorientierte Herangehensweise kommen Entwurfmuster auf Klassen- und Objekt-Ebene erneut als Design-Hilfe zur Geltung.
Für die XML-Beschreibung der Inhalte, also der Instanzen des Contentmodells, existiert ein einziges XML-Format[3]. Gerade die Kombination eines generischen, frei definierbaren Contentmodells und einem einzigen XML-Formats ist eine Stärke der Architektur, die für Projekte deutliche Vorteile bringen:

- Die erzwungene Modellierung eines Contentmodells ergibt eine saubere Grundlage für die darauf aufbauenden Anwendungen.
- Die Verwendung einem einzigen XML-Format im Kern erlaubt die Transformation in beliebige XML-Formate und die Übernahme beliebiger Formate an den Schnittstellen des Systems. Statt „M:N“-Transformationen müssen nur „M:1“- und „1:N“-Transformationen durchgeführt werden.

Durch die Betrachtung des Contents als Objektmodell lassen sich sehr mächtige Anwendungsdomänen analysieren, formalisieren und ausdrücken. Best-of-breed Systemarchitekturen haben in der Regel eine Entsprechung im CoreMedia Contentmodell. Zum Beispiel bedeutet die Back-End Integration von Ecommerce-Systemen für das Contentmodell, dass Produkte als entsprechende Objekte abgebildet werden können und durch Online-Redakteure auch in den redaktionellen Workflow zur Zusammenstellung einer Site berücksichtigt werden können, wie im Objektmodell in Abbildung 2 angedeutet.

Architekturmuster für Integrationsszenarien

Neben den im Produktkern verankerten Architekturmustern spielen Muster für Integrationsarchitekturen eine wichtige Rolle, vor allem beim Einsatz von zusätzlichen Produkten und Systemen. Architekturmuster für Integrationen bestehen aus folgenden Inhalten:

Architekturdiagramm: Das Architekturdiagramm beschreibt die Verteilung der Softwarekomponenten der verschieden Produkte und Teilsysteme. Ein Beispiel für ein Architekturdiagramm ist die in Abbildung 1 dargestellte CoreMedia Systemarchitektur.

Use Cases: Der Einsatz von mehreren eigenständigen Produkten mit eigenen Datenbeständen und Pflegeoberflächen macht eine frühzeitige Betrachtung der Arbeitsabläufe eines integrierten Gesamtsystems und eine Beschreibung der Interaktionsmöglichkeiten der Internet-User erforderlich. So sind in einer Content-Commerce-Integrationslösung die Use Cases „Produktdatenpflege im Commerce-System“, „Produktdatenübernahme ins CMS“, „Redaktionelle Verwendung von Produktdaten im CMS“ für die Site-Produktion relevant. Bei der Betrachtung der Use-Cases der Internet-User sind für ein Architekturmuster nur die Use Cases im Bereich der Schnittstelle beider Systeme relevant. Für das Beispiel der Content-Commerce-Integration wären dies „Login“ und „Wechsel vom redaktionellen Bereich zum Shop-Bereich und zurück“, nicht zwingend aber die durch ein Ecommerce-System vollständig abgedeckten Funktionen („Bestellung abschicken“), da diese nicht in den Bereich der Schnittstellen zwischen beiden Systemen fallen.

Domänenmodell: Das systemübergreifende Domänenmodell einer Integrationslösung ist für die Entwicklung von Back-End Schnittstellen, Business-Logik und der Integrationsoberfläche entscheidend. Neben dem Contentmodell des CoreMedia Systems und den Daten- oder Objektmodellen der zusätzlichen Systeme werden besonders die Assoziationen zwischen den Objekten verschiedener Systeme und die damit verbundene Schlüsselwertverwaltung beschrieben, so dass ein system-übergreifendes Domänenmodell entsteht.

Systemdynamik: Die Beschreibung der Dynamik einer Integrationslösung beinhaltet Abgleichverfahren für Daten, Ereignis-gesteuerte Handlungen und weitere für die Kopplung mehrerer Produkte nötigen Systemabläufe. Zum Beispiel findet bei der Integration eines externen Volltext-Indexers mit dem CoreMedia-System folgende Systemdynamik statt: „Angestoßen von Publikations-ereignissen werden die neu publizierten Dokumente samt ID an den Indexer übertragen. Diese können per Volltextrecherche über JSP’s abgefragt werden.“

Technische Details: Wichtige technische Details eines Musters werden festgehalten, damit diese in iterativen Prozessen rechtzeitig berücksichtigt werden können, auch wenn das Muster in ersten Phasen noch nicht funktionsfertig sein soll. Als Beispiel soll folgendes dienen: Häufig wird die Anforderung nach einer „in einer späteren Stufe personalisierbaren Site“ gestellt. Hierfür müssen schon in ersten Realisierungsstufen die Session-

[3] Dies ergibt sich aus der Design-Entscheidung, das die *Container* des Contents (Objekte mit benannten Attributen) zusammen mit den Inhalten der Container in XML ausgedrückt werden. Daher ist nur eine DTD für beliebige Strukturen erforderlich.

Behandlung berücksichtigt werden, damit eine spätere Hinzunahme von personalisierter Funktionalität nicht durch aufwendige Umkonfiguration von Loadbalancern, Webservern und aller JSP's verteuert wird. Dies bedeutet, dass Session-ID's in URI's Caching-konform hineingeneriert werden (vgl. das in [6] beschriebene URI-Konzept für Caching und Personalisierung) und eine Session-Zuordnung von HTTP-Request zu antwortendem CoreMedia Generator über den Lastverteiler erfolgen muss, damit alle Anfragen einer Session in dem gleichen Servlet-Container behandelt werden können. Technische Details dieser Art fallen aus der allgemeinen Detaillierungsstufe der Architekturmuster heraus, sind aber für eine zügige Implementierung unerlässlich.

Für eine Projektplanung und ein Vorgehensmodell haben Architekturmuster einen hohen Wert. Aktivitäten und kausale Zusammenhänge zwischen Aktivitäten sind vordefiniert, wie auch Lösungsmodelle für komplexe Zusammenhänge. Daher sind Phasen einer Planung anhand der Architektur „ablesbar": z.B. folgt die Definition des Contentmodells der Analyse der Contentverwendungsarten und der Contentströme, anschließend kann erst mit der Entwicklung von JSP's für die Ausgabeseite und XSLT-Definitionen für die Contentübernahme begonnen werden. Für Standardprojekte existieren Schablonen für Standardplanungen. Projekte, in denen teilweise vollkommen neue Anforderungen gestellt werden, profitieren auch von den bestehenden Architekturmustern, da diese die Koordinaten potentieller Lösungsräume beschreiben.
Oft werden sehr vage Anforderungen an eine Endausbaustufe eines best-of-breed Systems gestellt, das aus mehreren zugeschnitten Produkten besteht. Systeme müssen etwa für Content-Syndication oder Personalisierung offen sein, oder gänzlich neue Zielformate unterstützen. Die Konsequenzen derartiger Anforderungen sollten möglichst früh in einem Projekt bekannt sein. Gerade bei der Einführung des Produkts zusammen mit anderen Systemen helfen Architekturmuster, die Konsequenzen integrations-relevanter Themen zu identifizieren und zu bewerten.
Der Katalog von Architekturmustern wird bei CoreMedia AG kontinuierlich erweitert und steht den CoreMedia Beratern und den Projektleitern der Integrationspartner für die Planung von Projekten zur Verfügung. Er ist ein wertvolles Hilfsmittel im Software-Engineering Prozess für CoreMedia Projekte.

Der CoreMedia Prozess

Der architekturzentrische CoreMedia Prozess stellt eine Synthese aus **iterativer** Software-Entwicklung und **nicht-iterativer** System-Organisation und –Installation (*Deployment*) dar. Iterative Software-Entwicklung ist fester Bestandteil moderner Standardprozesse wie dem Rational Unified Process [5] oder Firmenmethodologien wie PERFORM von Cap Gemini und dient als sinnvoller Mechanismus für Anforderungsmanagement, vor allem bei unklaren oder wechselnden Anforderungen. Nicht-iterative Prozesse sind für besonders zeitkritische Projektanteile geeignet, bei denen die Anforderungen im wesentlichen bekannt sind und ein großer Koordinationsaufwand erforderlich ist, z.B. beim Management von Mitwirkungsleistungen oder Unteraufträgen.
Der CoreMedia Prozess ist durch folgende Merkmale gekennzeichnet:

- Aufgrund des rasanten Internet-Umfelds sollten Projektlaufzeiten bis zu einer Live-Schaltung eines Systems oder eines Teilsystems nicht länger als 6 Monate dauern. Anforderungen, deren Komplexität eine längere Projektdauer für eine erste Live-Schaltung erforderlich machen, sollten derart vereinfacht werden, dass frühere Inbetriebnahmen von Funktionsteilen möglich sind.
- Am Anfang findet eine als „Iteration 0" sogenannte „Scope"-Phase für einen schnellen Start statt. Das Ergebnis einer Scope-Phase ist eine Projektplanung und häufig auch ein Angebot für ein Projekt, das zu Projektbeginn (vor der Scope-Phase) noch nicht abschätzbar ist. Für eine Scope-Phase, die in der Regel einen festen Zeitraum umfasst, kann ein Festpreis vereinbart werden. Weitere wichtige Ergebnisse der Scope-Phase sind eine Anforderungsbeschreibung, oft beruhend auf Site-Designs und Storyboards, ein initiales Architekturdesign unter Zuhilfenahme der Architekturmuster, ein erstes Contentmodell und eine Hardware-Konfigurationsplanung. Architektur-muster fließen in die Planung ein und filtern technische Details heraus, die frühzeitig betrachtet werden müssen.
- Software-Entwicklung erfolgt iterativ. Im wesentlichen umfasst dies die Aktivitäten Contentmodellierung, JSP-Entwicklung, XSL-Definition für Content-Importer und, in einer späteren Phase, optionale GUI-Anpassungen für Domänenvalidierungen. In CoreMedia Projekten müssen Entwickler zwingend an der Modellierung beteiligt sein, es darf keine scharfe Trennung zwischen „Content-Design Team" und „Programmierteam" geben. Eine solche Trennung widerspricht dem objektorientierten Entwicklungsansatz.
- Begleitend zur Softwareentwicklung findet ein nicht-iterativer „Deployment Track" statt. Hier ist

Iteration nicht erforderlich, da es um die Durchführung von Projektaufgaben mit bekannten Anforderungen geht. Da das Deployment von großen CMS-Anwendungen in Projekten erfahrungsgemäß eher als die Softwareentwicklung auf dem kritischen Pfad liegt, muss hierauf ein großer Anteil des aktiven Projektmanagements liegen. Zum Deployment Track gehören die Aktivitäten Konfigurationsplanung, Hardwarebeschaffung, Erstellung eines Betriebskonzepts, Installation, Content-Migration, Anwendertests und Lasttests.

Ein Beispiel für einen groben, am CoreMedia Prozess ausgerichteter Projektplan ist in Abbildung 3 dargestellt. Die Aufteilung des Projekts in einen iterativen Software-Entwicklungsstrang und einen nicht-iterativen Deployment-Strang ist erkennbar, wie auch der deutliche Fokus auf eine frühzeitige Aufnahme der Deployment-Aktivitäten. Eine für ein CoreMedia-Projekt exemplarische mittlere Teamgröße ist ebenfalls ablesbar.

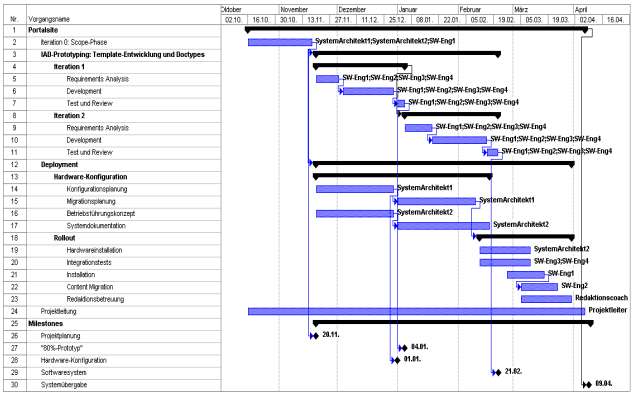

Abbildung 3 Beispiel einer CoreMedia Projektplanung

Entwurfsmuster spielen in dem CoreMedia Prozess eine wichtige Rolle, denn sie helfen, auch unklare Anforderungen in Lösungen und Architekturen umzusetzen. Sie geben Orientierung zum Herausfiltern der benötigten Detailtiefe und reduzieren dadurch die Komplexität. CoreMedia Projekte profitieren von dieser Betrachtungsweise und von dem an Entwurfsmustern ausgerichteten Prozess, der, zusammen mit dem CoreMedia Produkt, erfolgreiche und zukunftsweisende Content Applications Realität werden lässt.

Literaturverzeichnis

[1] Patricia B. Seybold. „Get Ready for M-Commerce and Interactive TV". *Customers.com*, Februar 2000.

[2] „CoreMedia Publisher Whitepaper*". CoreMedia AG*, www.coremedia.com

[3] John P. Dalton, Harley Manning, Katherine M. Gardiner. „Managing Content Hypergrowth". *The Forrester Report*, Januar 2001.

[4] Philippe Kruchten. „The Rational Unified Process – An Introduction, Second Edition". *Addison-Wesley* 2000

[5] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. „Design Patterns – Elements of Reusable Object-Oriented Software". *Addison-Wesley* 1995

[6] "CoreMedia Publisher HowTo Letter: Webserver- / Generator-Konfigurationskonzepte bei personalisierten Inhalten". *CoreMedia AG*, www.coremedia.com