Product Approach For Next-generation Content Management System
About the Authors

Jayant Dani,
Jayant Dani specializes in Data Management, Product Management, and Performance and Solution Architecture in large scale implementations across various industries. He has over 20 years of industry experience, received his Masters in Computer Science from BITS, Pilani, India and is a TOGAF Certified Architect. Jayant currently heads the Product Technology Group within the Solution Engineering Group, CEG at Tata Consultancy Services Ltd.

Sandeep Krishna S
Sandeep Krishna S is an Enterprise Content Management (ECM) specialist with six years of experience in the IT Industry, and has expertise in various ECM Solutions. He has worked with key clients in customization and enterprise solution implementation. Sandeep received his Masters in Software Systems from BITS, Pilani, India. He is currently the Technical and Product Architect of ECM products and solution conceptualization within the Solution Engineering Group, CEG at Tata Consultancy Services Ltd.
According to recent trends, the IT service industry is focusing on building products to effectively compete in an evolving market and sustain growth. In keeping with this trend, Tata Consultancy Services (TCS) has developed several proprietary products and implements software solutions by leveraging its own products. TCS’ Solution Engineering Group (SEG) aims to develop world-class domain-based products that are cost effective, technologically advanced, and deliver better value. SEG’s Software Product Line (SPL) supports the rapid creation of a structured collection of software systems from a shared set of software assets.

With content growing exponentially in size and variety, Content Management Systems (CMS) are fast gaining popularity for managing the life cycle of both structured and unstructured content. We identified that a Product Line Approach is needed to architect our CMS in a secured manner to meet the next-generation (next-gen) needs of an enterprise. We carried out extensive research and analyses to incorporate industry best practices and standards to meet the scalability and interoperability needs of the next-generation CMS.

This paper describes the experience and journey of developing a full-fledged CMS and its Enterprise Content Management (ECM) products across solution conceptualization, architecting, designing, planning, and building core assets. The paper also covers the integration of these while developing a complete product by adhering to standards, technology, cost, performance, and quality processes.

Keywords: Content Management System, Product Line Approach, Architecture, Enterprise, Core Assets, Big Data Store, Research, Open Standards.
## Contents

1. Introduction .......................... 5
2. Product Journey ...................... 6
3. Product Innovations ................. 21
4. Product Implementations .......... 22
5. Future Scope ........................ 22
6. Summary ............................. 23
7. References ........................... 23
8. Declaration ........................... 23
1. Introduction

A CMS can manage the information or data specific to a website, application, or data repository. CMS allows content to be created, edited, versioned, published, searched, and maintained, all from a central interface. Figure 1 represents a CMS.

There are a number of good CMS products available in the market today. However, the plan was to come up with a home-grown solution to address the following objectives — elimination of software license cost, no vendor lock-in, full ownership and control of source code of the asset, and usage of license-friendly open source products to avoid licensing and legal obligations.

This paper describes the steps and thought processes undertaken to build such an in-house CMS product.

To begin with, the architecture of the product was created based on a Product Line Approach to serve as the initial reference architecture. Then, based on specific business needs, independent products were developed. In this context, the reference architecture for the base model of CMS was created first, keeping in mind the business demands of today’s modern world. Subsequently, Enterprise Content Manager products for Web Content, Digital Asset, Document and Knowledge System were developed.
2. Product Journey

We have amassed vast experience in using many enterprise and CMS products for designing and creating websites, securing and managing content, streamlining the content lifecycle, enabling and disabling features, and customizing products. We leveraged this to develop a new product which meets SEG customer needs by using internal reusable software assets. Figure 2 represents the product journey path.

![Figure 2: Product Journey](image)

The journey began with identifying the core CMS engine, building other associated components, and wiring them to build the end-to-end enterprise Web Content Management (WCM) Product. The following sections describe it in detail.

2.1 ECM Components

Research on various CMS components was done based on the enterprise requirements to build an enterprise-specific CMS. The key components identified for CMS included:

- Content storage, creation, and retrieval
- Content security
- Content workflow
- Content services – search and version
- Web content – storage, editing, and management
- Content preview
- User Interface (UI) for content/website creation, preview and management
First, we analyzed the open source CMS products and frameworks with respect to Java technology and CMS to understand the design and development process for building an enterprise CMS to manage web content. We used Java technology as it is platform-independent and supports technology agnostic product deployment. The products analyzed were Alfresco, Magnolia and Hypo. These products are developed using the Java Content Repository (JCR 2.0) specification and Apache Jackrabbit (AJR).

AJR is a fully conforming implementation of the content repository for Java Technology API (JCR, specified in JSR 170 and 283) and provides most CMS features such as content creation, categorization, versioning, search, index, and a repository which are required for a CMS system.

AJR based on Apache License 2.0 offers a liberal license and allows modification to its source code as well as packaging in a product, without any licensing/legal obligations. We chose it for the core CMS engine features.

Our technology research also addressed the other key components required to build an Enterprise Content Management (ECM) system to manage Web Content.

2.1.1 Content Security

This feature was a pre-requisite to secure and protect the content and provide access to users based on roles and privileges. Access Control Lists (ACLs) were created and roles defined to secure access to the data repository at an interface level for users. We used Apache Jackrabbit to provide the default security and access management.

To offer complete access management with User Role Management, we considered various external components such as Spring Security, Java Authentication, and Authorization Service. However, we decided to use TCS SEG’s Core Asset for URM, as it provides ready-to-consume services without the need to get into the finer implementation details.

2.1.2 Workflow Manager

A key component for any CMS is managing the content lifecycle. We considered two options – open source JBoss JBPM (Business Process Management) and TCS Workflow Manager – and conducted proofs of technology on both. JBPM is a full-fledged process orchestration framework with flexible options, exceeding WCM needs. On the other hand, TCS Mastercraft Workflow Manager provides most of the workflow features required for WCM. We decided to go ahead with this option given that it is an internal product with no licensing or third-party dependency concerns. It provides the Bizagi process modeler to design the workflow process which generates XML Process Definition Language (XPDL) – that serves as an input to TCS MasterCraft Workflow Manager.

2.1.3 Template Engine

This is required to preview the content in a particular format. Templates play an important role in the case of WCM where web pages and content created are previewed through the template to provide better visualization. We evaluated open source template engines such as FreeMarker, Velocity, Thymeleaf, and other Java Server Pages (JSP) viewers. Based on our evaluation, we chose FreeMarker as it is lightweight, has a powerful template engine, and supports the JSP tag library too. This ensured that writing/previewing the templates using FreeMarker was standardized while providing options to use other formats such as JSP and HTML as part of WCM with the corresponding view resolver.
2.1.4 Content Search
Content search is another key feature of a CMS. A content search is done using a content name/ID. A refined search can be done based on the content metadata. An elastic or full text search can be performed on the file attachments such as MS Word documents, PDF files, and so on by providing keywords. Apache Lucene and Apache Solr are search engines that present fast, open source enterprise search platforms. Their major features include powerful full-text search, hit highlighting, faceted search, near real-time indexing, dynamic clustering, database integration, rich document handling, and geospatial search. The Lucene search engine, available in AJR by default, met our needs.

2.1.5 Data Repository
Data Repository is a file system or Relational Database Management System (RDBMS) based repository. JCR supports both repository types. As an open source system, Postgres is a fully compliant, freely available database system that can be configured when initializing a repository using JCR. JCR by itself has a default file system based repository for data storage that offers all the features related to data: search, indexing, versioning, workspace creation, and categorization of content.

We considered scalability as well as the need to effectively handle the growing volume of data and functions to meet user requirements of performance, concurrency, and load. We also evaluated the option of storing the content in a Big Data system, using the optimal open source solution based on Apache Hadoop technology.

We identified the key components of the solution and the respective technologies. The component services were developed and packaged as core assets and utilized across products as per ECM needs. Figure 3 summarizes the core assets and associated services involved in building the ECM product.

\[\text{Figure 3: ECM Core Assets}\]

\[\text{CRUD} \rightarrow \text{Create, Retrieve, Update, Delete}\]
\[\text{ACL} \rightarrow \text{Access Control List}\]
2.2 Product Line Approach

Product Line Approach refers to a suite of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission. These are developed from a common set of core assets in a prescribed way, allowing developers to leverage common characteristics among a family of products they produce, and as a result, improve quality, reduce cost, and save time. The aim is to create the Reference Architecture first and then develop independent products based on specific business needs.

The next-gen Product Line Architecture focuses on handling the immense growth in the volume of content. In addition, it helps overcome the challenges of capturing, storage, transformation, searching, sharing, and visualization of data, where the data can be structured or unstructured. The architecture offers measurable advantages for parameters such as scalability, flexibility, interoperability, customizability, security, auditability and quality. Product Line Architecture also addresses key architecture principles such as leveraging open standards and open source technologies for complying with industry standards (such as JCR 2.0), faster time-to-market, no vendor lock-in, and cost effectiveness.

The Product Line was built by creating Core Asset Libraries and then wiring the assets to meet specific business requirements. The Core Asset Libraries involved in creating the Base CMS model were the Content Hierarchical Model, Content Security, Content Lifecycle Management, Multi-Channel Responsive Layouts, Social Content Meta-Data Model, and Multiple Content Repositories – File System, RDBMS and Big Data Store. We also deployed widely used industry design patterns for wiring the Core Asset Libraries to come up with the base model of CMS.

![Product Line Approach](image)
The major factors involved in adopting the Product Line Approach for ECM were:

- **Architecture** – The architecture supported the variation inherent in the product line. It offered the flexibility of adding features to the system and handling the immense growth in the volume of content.

- **Software components** – We designed components and assets for easy plugin without a loss of performance and to ensure built-in support for variation points.

- **Test plans, test cases, and test data** – We prepared and executed the test plan and test cases considering variation points and multiple instances of the product line.

- **Business case and market analysis** – These addressed the entire family of software products and not just one product.

- **Project plans** – We followed the Agile methodology to ensure extensibility to accommodate product variations.

- **Tools and processes** – As the core asset and products were to be created on the Java platform, we took into consideration Java best practices and coding standards for implementation. We used our robust, in-house solutions, processes, and best practices in WMS. The tools used were:
  - **TCS MasterCraft Application Lifecycle Manager (ALM)**: Used to manage the requirements, test cases, and defects observed. We dynamically generated traceability matrices for the requirements use cases, test cases, and defects logged.
  - **TCS MasterCraft Application Quality Manager (AQM)**: This was our wrapper over the Eclipse integrated development environment (IDE). Developers could use this to perform code quality checks, detect bugs, analyze coverage, and generate test cases with ease and efficiency. Using AQM also guaranteed quality code and optimized services.
  - **SEG Workbench**: We used Hudson and Sonar, hosted on TCS SEG Workbench, for continuous build and generation of the Code Quality Check (CQC) report of the asset. Having set extremely high standards for all its products (above 98% in the Sonar report), SEG leveraged Hudson and Sonar to ensure that these standards were met. We also used the GIT version control system hosted on SEG Workbench to maintain the code and its versions of assets and products as well as to tag these to the respective release.
  - **TCS MasterCraft Java-Profiler**: Java-Profiler was the TCS wrapper over the performance tool of Jensor. Java-Profiler generated the performance report for the code executed. It adhered to the development guidelines and validated the report generated with the benchmark specified, improving the product performance quality.
  - **Fortify**: We tested code security and vulnerabilities using the Fortify security tool – this is a suite of tightly integrated solutions for identifying, prioritizing, and fixing security vulnerabilities in the software.

- **People, skills, and training** – This involved bringing resources on board, training and expertise building centered on the procedures associated with the product line.
2.3 Core Asset Building Process

The key component in implementing the Product Line Approach is to understand the non-variable and pluggable components of multiple product architecture and create core assets for it. The assets developed can be leveraged for any product to include the features provided in the core assets.

The major steps followed in the CMS’ core asset building are explained in the following sections:

2.3.1 Requirements Engineering

The functional and non-functional requirements were captured and evaluated as a part of the requirements gathering phase.

TCS’ trademark framework, Quality Operational Process Definition (QOPD), comes with a defined set of quality principles and standards. The functional requirements were validated with respect to business requirements and evaluated against the non-functional requirements pertaining to security, compatibility, reliability, usability, performance, maintainability, and portability.

2.3.1.1 Non-Functional Requirements

Non-Functional Requirements (NFRs) were evaluated pre- and post-development of the core assets to deliver these with the utmost quality. The NFRs considered in the product line were:

- Security
- Performance
- Usability
- Infrastructure

Once the core assets were developed and tested, they were validated against NFRs. The NFR team validated the initially agreed upon NFRs of the core assets which were logged in the QOPD checklist.

The Center of Excellence (COE) teams for usability, infrastructure, security, and performance evaluated the functional requirements and provided the non-functional standards to be considered and included as a part of the design, architecture, and development of a product. The core asset developed was evaluated with the non-functional requirements considered in each of the NFR areas. The COE team provided the necessary sign-off after considering all the necessary non-functional requirements that the assets must meet.

2.3.1.2 Functional Requirement

The functional requirement consisted of use cases for each core asset. Use cases were recorded in line with business needs and evaluated with the corresponding business team to ensure comprehensive requirements coverage.

We decided to build the core asset based on the functional requirements received from the product teams to enable WCM in their system. Figure 5 represents the Core Asset Architecture.
The plan involved building a generic CMS core asset, followed by specific core assets for WCM and its dependent components.

(i) Generic Core Asset – CMS

This process entails developing a generic core asset with all the basic features required for a CMS. Based on the research performed in Section 2.1, Apache Jackrabbit was used to develop this core asset. This asset had all the basic features such as:

- Content Repository – Create/Edit
- Workspace – Create/Delete
- Basic Content Categorization
- Content Metadata – Create
- Content – Create/Retrieve/Update/Delete
- Content Versioning
- Basic Content Search

We identified and developed the API services for these use cases. The asset focused on creating a content repository – an RDBMS or a file system. We also created multiple workspaces inside the repository as well as content...
metadata and definitions. We then created and uploaded the content with the created definitions and content structure; and updated it as per need. Further, we enabled a basic search on name and ID to support content search and retrieval.

We created the Application Programming Interface (API) services with the package name com.tcs.seg.wcm. These services were built and packaged as a reusable Java archive file CMS.jar.

(ii) Enterprise Core Asset – WCM

After the generic core assets, the requirement specific core assets needed to be developed. In this case, a WCM core asset was developed to manage web content to meet the enterprise needs. The specific core assets were developed on top of the core asset (CMS.jar).

The WCM core asset was planned for development with additional complex use cases addressing all the basic features of WCM such as:

- Advance Content Categorization
- Template Management
- Web Page Management
- Content Security
- Content Workflow

The API services for each of these use cases were developed based on the research and analysis of technologies. Here, some design patterns were used to develop this asset so as to enable a plug-in for coupling and decoupling of third-party assets with the core WCM asset, and also for the advanced content categorization and dynamic update of the content’s metadata. The design patterns used were:

- Adapter Design Pattern – The adapter pattern, a wrapper class (the adapter) was used to translate requests from it to another class (the adaptee). In effect, an adapter provided particular interactions with the adaptee that are not offered directly by the adaptee.
Composite Design Pattern – The composite pattern existed as a tree structure where identical operations could be performed on leaves and nodes. (A node in a tree is a class that can have children.)

Figure 7: Composite Design Pattern – JCR Model

Advance Content Categorization was done based on the parent-child relationship to develop a hierarchy of content. It also enabled definition and categorization of content metadata based on the type of content.

Content Lifecycle was defined by plugging in TCS MasterCraft Workflow Manager. Workflow APIs were created to enable easy plug-in to TCS Workflow Manager.

Template services were designed to support different types of views based on the template engine. Templates can be created in FTL/JSP/HTML. Template API services were designed to be used for creating, updating, retrieving, deleting, and previewing content. We also provided an option to reuse the templates and associate templates as part of the API services.

Web Page services were defined to create, update, retrieve, delete, and preview web pages. The web pages were categorized based on the type of content. The web page services offer options to relate pages, select workflows, associate content, and templates.

Content Security was provided to restrict user privileges to the repository by providing the relevant Access Control List (ACL). User access to functionality was restricted based on the user’s role. Content security APIs were created for these services using TCS User Role Management (URM).
We then developed the WCM core asset containing all the service APIs. The required services for content security, template, pages, and workflow were a part of this package and packaged as WCM.jar. They were connected through the adapter to invoke the external asset services. Figure 9 represents the assembly of core assets.
2.3.2 Testing Services

Testing is the process of executing a program with the intent of identifying errors. Test cases were prepared and the test results logged. The various testing activities performed as part of this process were:

- Unit Test
- Validation Test
- Performance Smoke Test
- Integration Test
- System Test
- Functional Test

Unit testing was conducted for the unit testing of all the services developed. Validation testing was done for the parameters inserted into the services and the return result. System testing was carried out whenever the software was deployed on a different system or environment and all the prerequisites for running the software were made available on the target system. Functional testing helped check the end-to-end execution of a function – a combination of different unit test cases. We then conducted performance testing of the software with respect to the time, resource, and size of a function or business service.

2.4 Product Building Approach

According to the Product Line Approach, software components and assets are developed and integrated based on the business case and market analysis for leveraging them across products that require a CMS.
We planned the development of a WCM product founded on the CMS assets. The required core assets were integrated and assembled to build the WCM product.

Initially, a project plan was prepared based on the sequence of operations to be performed. Agile development methodology was adopted to execute the plan. In general, Agile leads to frequent, incrementally delivered features, and functions, even in the case of complex and uncertain requirements. In our case, the architecture and design was defined such that it adopted the requirements-based incremental changes and leveraged any new assets to incorporate additional features and functions in the system.

The sequence of items executed as per the product development plan included:

1. Product architecture
2. User experience design
3. Design of product services
4. Deployment and integration of core assets with product services
5. Content repository setup
6. Integration of user interface with product services
7. Product development
8. Product testing
9. Sample website design and creation

2.4.1 Product Architecture

Product architecture represents an arrangement of functional elements in physical chunks which become the building blocks for the product or family of products.

![Figure 11: High Level Technical Architecture](image)
Figure 11 depicts the high-level technical architecture of the ECM product.

2.4.2 User Experience Design

User Experience design was done in consultation with TCS’ Usability COE to provide better navigation and a user-friendly interface. Initially, we prepared the wireframes and after a peer review, designed the visual look of the WCM product. Subsequently, we transformed the visual design into HTML screens, navigation for which was as per the designed wireframes. Figures 11 and 12 represent the WCM Login page and Admin interface.

The Admin interface in Figure 12 shows a dashboard with the list of content for a user and the status of the content in the content lifecycle. The tree on the left represents the content hierarchy, classified according to content type. The top bar provides an option to create and manage content.
2.4.3 Design of Product Services

We designed and prepared the list of product services based on the functionality and relationship between each service. Design and categorization of the flow of services was based on their functionality and interfaces. Product services included model, controller, and service implementations.

2.4.4 Core Asset Wiring for Product

Core assets were now deployed in the project to implement and execute product services. The core assets developed for WCM addressed almost all the required functionality. Executing the services in core assets developed took care of the content schema. In this case, content categorization and the relationship between content such as pages, templates, and content types was designed and made available as a part of the WCM core assets. The URM was also available as a part of the URM core assets. Since schemas were a part of the core assets, they did not require any specific design and implementation for the WCM product architecture.
2.4.5 Content Repository Setup

On executing the required service from the core asset, we completed the repository setup and made it configurable. The content repository was set up to store the product and application contents.

2.4.6 Integration of User Interface with Product Services

After the repository was set up and the product services developed, they were integrated with the user interface designed by the usability team. The corresponding services were mapped to the modules and areas of the user interface as designed in the wireframes.

2.4.7 Product Development Standards

We followed product development standards such as Spring framework for MVC, coding standards, and design patterns. We also considered options for the configuration and extendibility of services, deployment of any core assets, and upgrading of the product with new features; and developed these options accordingly.

2.4.8 Product Testing

Testing is a critical element following development. Initially, test cases were prepared and executed with respect to all the areas mentioned in section 2.3.2 for core assets. Along with this, we also carried out User Acceptance Testing for all the product functionalities. The bugs detected were fixed to deliver a high quality product.

2.4.9 Sample Website

Once the product was ready, we created a sample website using the product. The content taxonomy for the site was created with different content types and content for each of them and templates designed to preview the contents in the required format. Finally, required pages were designed for the website and associated with the corresponding content, templates, and pages to be navigated.

Figure 14: Sample Website Using WCM
3 Product Innovations

The key differentiators for the WCM product were a scalable system for the growing volume of content, and a dynamic layout editor to enable end business users to design their layout with no dependency on IT developers.

3.1 Big Data Store

To achieve a scalable system, integration with a Hadoop-based Big Data system was evaluated and a plug-in was designed to store the huge amount of data in the Big Data system. This ensured the scalability of the WCM product.

A patent has been filed for, based on this connectivity for content storage, named Content Management System – Big Data JCR Plugin.

3.2 Layout Editor

The Layout Editor was designed as an independent component to provide the list of widgets. Each widget had its set of properties which were configured and designed in the working pane. Once the required design items were added to the working pane, we generated the respective template code for the layout.

Figure 15: WCM Layout Editor
4. Product Implementations

The core asset deployment and its implementation in various products is explained in the following sections.

4.1 Health Insurance Project

The CMS and WCM core assets are being used in a Health Insurance product for content storage/retrieval. They provide a complete enrolment solution by offering high quality Enrolment-as-a-Service at an optimal cost. The content includes Electronic Data Interchange documents, Pentaho transformation jobs, and reference documents.

4.2 eCommerce Platform

The core assets are being used in an eCommerce platform to support a host of online business models to help retailers and brands successfully execute their online business strategy for buying and selling products and services. The eCommerce platform requires a dynamic layout editor to enable end users to design the layout pages for retail products, with a content lifecycle provided using the WCM core assets.

4.3 Product Information Portal

The Product Information Portal is a feature-rich platform which provides collaborative structured authoring that is implemented using MediaWiki. A self-sufficient package, it provides users with personalized facilities in terms of preferences and feedback and also allows for immediate modification and publishing of content. The long-term plan is to build this Product Information Portal using the WCM product to own the complete IP.

5. Future Scope

The entire product journey has offered several learning opportunities in terms of thought, approach, functionality, non-functionality, process, and innovation that led to the development of a WCM product; and identified its future scope.

The CMS product line can enable the implementation of other such initiatives by leveraging the learning for future content management products.

Some key areas for future enhancements and products of CMS are:

- Web Experience Management
- Campaign Management
- Digital Asset Management
- Multi-Channel Responsive Layouts
- Mobile Application Integration
- Social Network Integration
- Web Analytics

The implemented Product Line Approach will support the incremental scope of CMS products as required for an enterprise.
6. Summary

Current market trends and research on various products and technology to develop core assets for SEG products have directed the journey for identifying this approach. Spanning across product architecting and development, this approach envisions future business applicability, as well as the need to manage the growing content. The Software Product Line Approach is effectively deployed and its benefits have been realized. Core asset engineering produces reusable components with the required processes for a quality release. The product planning and implementation on top of core assets has led to an enterprise implementation of a full-fledged enterprise product WCM.

In terms of implementation, it has been deployed across products as a core asset and also led to innovation in terms of scalability and layout editor. The journey of this product development has resulted in the successful building of a WCM product and paved the way for future CMS products.

7. References

2. TCS Tools Group & Solution Engineering Group - References and Best Practices.
3. Arron Ferguson., Creating Content Management Systems In Java ed, 2007 (Cengage Charles River Media, Boston)
4. http://open-source.gbdirect.co.uk/migration/benefit.html#reliability

8. Declaration

We hereby declare that I am the author of this paper, written exclusively for Tata Consultancy Services Ltd. I acknowledge that this content is original, is not plagiarized and bears no resemblance to any existing or previously used work. I have ensured that the content does not infringe upon or violate the copyright, trademark rights or any other intellectual property rights of, any person, firm or corporation. Any and all copyrighted third party content or data and any derivative thereof has been used under express written permission / license from, and attributed to, the legal owner of the said content or data.
Contact
For more information about TCS’ consulting services, contact global.consulting@tcs.com

Subscribe to TCS White Papers
Feedburner: http://feeds2.feedburner.com/tcswhitepapers

About Tata Consultancy Services (TCS)
Tata Consultancy Services is an IT services, consulting and business solutions organization that
delivers real results to global business, ensuring a level of certainty no other firm can match.
TCS offers a consulting-led, integrated portfolio of IT and IT-enabled infrastructure, engineering and
assurance services. This is delivered through its unique Global Network Delivery Model™,
recognized as the benchmark of excellence in software development. A part of the Tata Group,
India’s largest industrial conglomerate, TCS has a global footprint and is listed on the National Stock
Exchange and Bombay Stock Exchange in India.

For more information, visit us at www.tcs.com

IT Services
Business Solutions
Consulting