**FlexPLM BOM Costing Data Flow**

A PROJECT REPORT

submitted by

**VARUN MITTAL**

**( 11BCE0111 )**

*in partial fulfillment for the award of*

**B.Tech**

Degree in

**COMPUTER SCIENCE & ENGINEERING**

****

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

January 2015 to May 2015

**School of Computing Science and Engineering**

**DECLARATION**

I hereby declare that the project entitled **“FlexPLM BOM Costing Data Flow”** submitted by me to the School of Computing Science and Engineering, VIT University, Vellore-14 in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out by me under the supervision of **Prof. Anisha M. Lal, Assistant Professor (SG).** I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

Signature

**VARUN MITTAL**

**(11BCE0111)**

**School of Computing Science and Engineering**

**CERTIFICATE**

The project report entitled **“FlexPLM BOM Costing Data Flow”** is prepared and submitted by **Varun Mittal (11BCE0111).** Ithas been found satisfactory in terms of scope, quality and presentation as partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** in VIT University, India.

**(Name & Signature of the Internal Guide)**

**Internal Examiner External Examiner**

**ACKNOWLEDGEMENT**

The completion of any task depends upon the co-operation, coordination and consolidated efforts of several resources of knowledge, energy, time and above all the proper guidance of the experienced. Therefore I approach this matter of acknowledgement through these lines trying my best to give full credit where it is deserved.

I wish to express my gratitude to my mentor Mr. Chandra Dev Singh**,** Lead Consultant, Infosys Pvt. Limited, who generously helped me to compile this project with his knowledge and expertise. He gave me an opportunity to work on part of the project, and the necessary guidelines regarding the project, helping me in tracking first-hand information as well as insulating a belief in me which proved essential for the completion of this project.

I also thank my Project Guide Prof. Anisha M. Lal, Assistant Professor (SG.) VIT University, Vellore who lent me continual support and reviewed my progress periodically which enabled me to complete the task assigned successfully. I also thank Prof. Hari Seetha, Program Chair SCSE VIT University, Vellore and Prof. Siva Venkateswaran, Dean SCSE, VIT University, Vellore. I finally thank VIT University, Vellore which provided me with an excellent platform to demonstrate my engineering skills in the field of Computer Science and Engineering.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE NO.** |
|  |  |  |
|  | **EXECUTIVE SUMMARY** | 6 |
| **1.** | **INTRODUCTION** | 7 |
|  | 1.1 OBJECTIVE | 7 |
|  | 1.2 MOTIVATION | 7 |
|  | 1.3 BACKGROUND | 7-8 |
| 2. | **PROJECT DESCRIPTION AND GOALS** | 9-10 |
| 3. | **TECHNICAL SPECIFICATION** | 11 |
|  | 3.1 HARDWARE REQUIREMENTS | 11 |
|  | 3.2 SOFTWARE REQUIREMENTS | 11 |
| 4. | **DESIGN APPROACH AND DETAILS** | 12 |
|  | 4.1 DESIGN APPROACH | 12 |
|  | 4.1.1 PROPOSED SYSTEM ARCHITETURE | 12-13 |
|  | 4.1.2 INTEGRATION PROCESS | 13-14 |
|  | 4.1.3 LOGICAL DATA MODEL | 14 |
|  | 4.1.4 CUSTOMIZATION PROCESS FLOW | 15 |
|  | 4.1.5 CONFIGURATION CHANGES | 16 |
|  | 4.1.6 ALGORITHM FOR SSP CUSTOMIZATION | 17 |
|  | 4.1.7 SYSTEM INTERFACE DESCRIPTION | 18-20 |
|  | 4.2 CODES AND STANDARDS | 21 |
|  | 4.3 CONSTRAINTS,ALTERNATIVES AND TRADEOFFS | 22 |
|  | 4.3.1 ASSUMPTIONS | 22 |
|  | 4.3.2 DEPENDENCIES | 23 |
|  | 4.3.3 ALTERNATIVE SYSTEM DESIGNS | 23-24 |
| 5. | **SCHEDULES,TASKS AND MILESTONES** | 25 |
| 6. | **PROJECT DEMONSTRATION** | 26 |
|  | 6.1 SCREENSHOTS | 26-28 |
| 7. | **MARKETING AND COST ANALYSIS** | 29 |
|  | 7.1 COST ANALYSIS | 29 |
| 8. | **SUMMARY** | 30 |
| 9. | **REFERENCES** | 31 |
| 10. | **APPENDIX** | 32 |
|  | - GANTT CHART | 32 |
|  | -SOURCE CODE FOR SSP CUSTOMIZATION | 33-34 |

**EXECUTIVE SUMMARY**

Dealing with fast fashion, private-label goods and branded lines is challenging in today’s world. In today’s challenging retail economy, top brands and retailers must constantly develop innovative, high-quality products at affordable prices to meet consumer expectations .Footwear and Apparel industries have volatile supply chains & require a different form of PLM flexibility compared to traditional PLM (manufacturing driven PLM). Footwear and apparel product designs are intensely driven by emotional concepts and decisions, fashion trends, material innovation, and new fabric technology –development considerations not taken into account in manufacturing-driven PLM methodologies. The most efficient way to control this process is by using a Product Lifecycle Management (PLM) system to manage products from idea through end of life.

PLM has gained wide acceptance as the primary means to improve product development processes across the product development chain to deliver the most business value. Thus, traditional PLM offerings have not adequately addressed the collaborative and creative nature of the footwear and apparel industry. The disadvantages of traditional PLM development process are that there are multiple points of communication, multiple versions of product, needless duplication of effort, an overall lack of control of the process, limited process visibility and no centralization in the system. To overcome the above disadvantages companies have implemented the FlexPLM process. In the FlexPLM process we have taken the problem of developing a re-usable and configurable customization in FlexPLM which can flow BOM level costing data to FlexPLM Cost Sheets. The solution to the above mentioned problem is to design a configurable customization in FlexPLM which can flow the line level cost information from a Bill Of Materials to Cost Sheet executed on a Server Side Plugin.

**1. INTRODUCTION**

Product life cycle management (PLM) at a conceptual level deals with optimization of ‘creation’ processes for product / services or design of production processes, across enterprise. PLM is an enterprise strategy as it lets global organizations work as a single team to design, produce, support and retire products, while capturing best practices and lessons learned along the way. PLM is viewed as an information strategy, an enterprise strategy, a transformational business strategy and ultimately, as a comprehensive approach to innovation built on enterprise-wide access to a common repository of product information and processes.[1]

* 1. **OBJECTIVE**

To design, develop and to test a configurable Customization in FlexPLM which can flow the line level cost information from a Bill of Materials to Cost Sheet to be implemented on a Server Side Plugin(SSP).

* 1. **MOTIVATION**

Dealing with fast fashion, private-label goods and branded lines is challenging. In today’s challenging retail economy, top brands and retailers must constantly develop innovative, high-quality products at affordable prices to meet consumer expectations. The motivation for reducing time to-market and to maintain margins in a complex global sourcing environment, having improved quality, better efficiency and process optimization, having improved regulatory compliance for market segments and reduced direct material costs drives the leading companies to quickly adopt Product Lifecycle Management (PLM) solutions across the enterprise to gain a key competitive advantage[2] .

* 1. **BACKGROUND**

Challenges in the ‘creation’ processes that drive the design of a robust PLM strategy and implementation are:

1. Product information not structured around ‘creation’ processes (i.e., no new product development and introduction (NDPI) view in PLM strategy)

2. Inability to leverage collaboration across the extended enterprise (suppliers, vendors, customers), in the context of the product life cycle

3. Inability to measure and optimize product life cycle cost: Product performance and analytics, total cost to the customer, knowledge management (of creation process) are some of the life cycle-wide characteristics that have a direct impact on business metric

To overcome all these challenges global leaders like Liz Claiborne, Nike and Patagonia trust their product development to FlexPLM for the retail, footwear and apparel industry. The Siemens PLM Software Product Development solution for aerospace and defense enables companies and their suppliers to be flexible and responsive with standard design applications and assured migration paths. Agile Product Lifecycle Management 9.3.1 developed by Oracle for the health care industry allows pharmaceutical companies to better manage the drug development process from end to end

**2. PROJECT DESCRIPTION AND GOALS**

The goal of the project is to design a re-usable and configurable customization in FlexPLM which can flow the line level cost information from a Bill Of Materials to Cost Sheet executed on a Server Side Plugin. The project has been divided into three broad areas to achieve the targeted functionality:

* Extracting Costing specific information from a FlexPLM BOM(Bill of Materials) specific to a Product and Sourcing Configuration .
* Transformation logic is to be applied i.e. the data to be applied on Cost Sheet would also be configurable via properties file or FlexPLM Type Manager Configurations.
* Transformed / Translated costing data should be applied to respective attributes on Costsheet and added with other Factory level attributes so as to calculate a total Cost.

FlexPLM is an automated process and a generic solution that brings a structured, centralized control approach in information exchange to the retail, footwear and apparel (RFA) enterprises. It provides the companies with an optimized and efficient formula for manufacturing their products. FlexPLM allows for better communication and collaboration between different types of vendors and customers, improved design productive techniques and better adherence to quality standards. [3]

With the above mentioned goals in mind, the FlexPLM module will have the following

functions:

* There will be a customization process that will be able to extract Costing specific information from a FlexPLM BOM.
* The customization will have a provision for applying any transformation logic i.e. the data to be applied on Cost Sheet would also be configurable via properties file or FlexPLM Type Manager Configurations.
* Different type of attributes would be used based on the need and business requirements The system will be able to configure relevant Attributes in the modules listed below:
  + Season
  + Product
  + Sourcing Config
  + Specification
  + BOM
  + Costsheet
* The system would be able to configure a System View on BOM to show Costing related attributes and columns
* There will be a process that will be able to leverage OOTB (Out of the BOM) Cost Calculations in BOM to sum up the Raw Material Cost
* There will be a feature that will allow company to use Configuration in Cost sheet to calculate Total Factory Cost – using BOM level cost + Factory Level Cost

In addition, the FlexPLM module will also have the following features:

* Server Side Plugin package would not be impacted by addition of a new server side code that is used to configure the whole system.
* The customization should will not degrade the end user perception of the system performance
* The module will not add any additional resourcing load on the Servers

**3. TECHNICAL SPECIFICATION**

The hardware and software requirements of the project have been given below as follows:

**3.1 HARDWARE REQUIREMENTS**

Operating Systems: - Microsoft Windows 7

4GB RAM

**3.2 SOFTWARE REQUIREMENTS**

Language: Java Servlet Faces (JSF), Java

Compiler- JAVA Eclipse IDE

PLM Package: FlexPLM 10.1

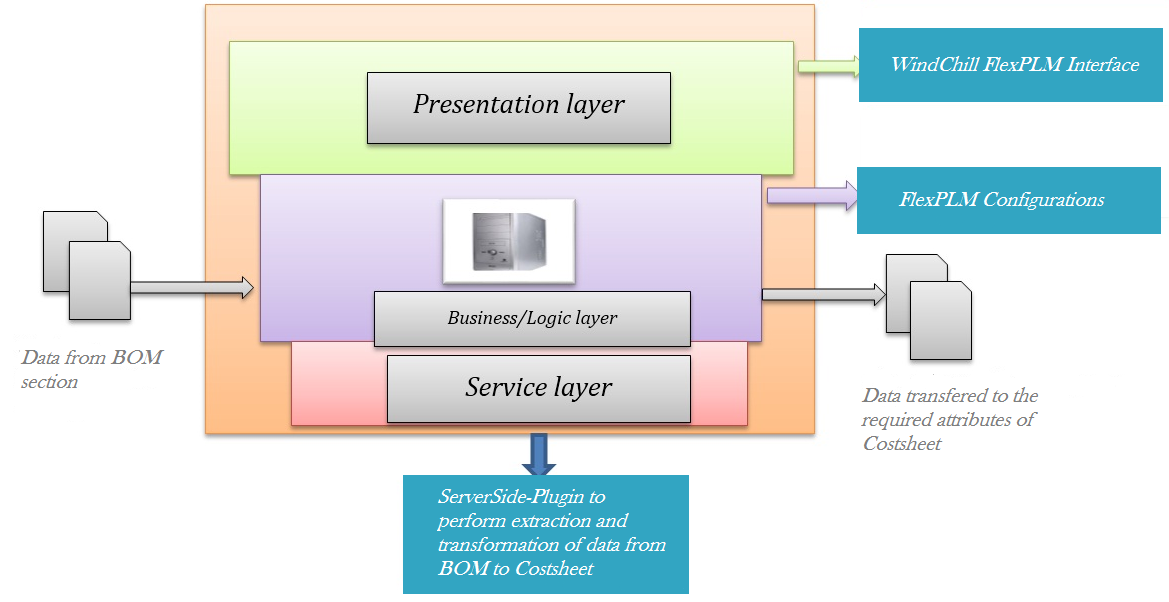
Browsers: Internet Explorer, Mozilla Firefox

DB: Oracle 11g, SQL

**4. DESIGN APPROACH AND DETAILS**

**4.1 DESIGN APPROACH**

**4.1.1 PROPOSED SYSTEM ARCHITECTURE**



**Fig.1 Proposed System Application Architecture**

**Presentation Tier:**

The presentation tier consists of application logic that prepares data for delivery to the client tier and processes requests from the client tier for delivery to back-end business logic. The logic in the presentation tier typically consists of J2SE components such as Forms that receive requests for processing. This tier might also include a portal service that can provide personalized, secure, and customized access to business services in the business service tier.

**Business Tier:**

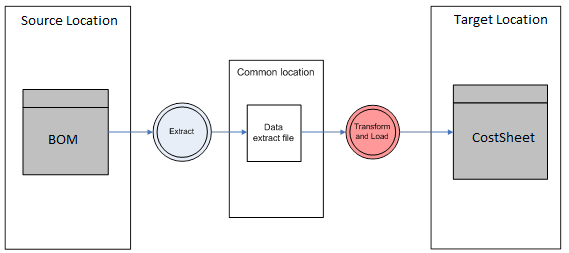
The business service tier consists of logic that performs the main functions of the application: processing data, implementing business logics, coordinating multiple users, and managing external resources such as databases or legacy systems. Typically, this tier consists of tightly coupled components. Individual J2SE components can be assembled to deliver complex business services.Business services can also be built as standalone servers.

**Service Tier:**

The data tier consists of data used by business logic. The data can be persistent application data stored in a database management system or it can be resource and directory information stored. The data can also include data feeds from external sources or data accessible from legacy computing systems.

Presentation layer has forms which provide a better User Interface. Through the UI the user can interface with the utility tool. There is only one user for this tool who is the admin of the system. The file that has to be loaded is browsed from the source system and the business layer will convert the excel file into tab separated file according to the corresponding map file of the business object. The generated tab separated file is the output of the business layer which can be loaded in the target FlexPLM System. Service layer has the java class which is used to perform the Extraction and Transformation functions from Bill of Materials (BOM) to Cost Sheet.

## **4.1.2 INTEGRATION PROCESS**



**Fig.2 Integration Diagram for Data Flow**

The source location contains the Bill of Materials (BOM) sheet wherein the various attributes have been added according to the configurations given by the client. Then the customization logic extracts the BOM information specific to a Product and Sourcing Configuration. Then the customization applies a type of transformation logic in which the data that is to be applied on Cost Sheet would also be configurable via properties file or FlexPLM Type Manager Configurations. After the data is transformed then we load the translated data into the target location containing the Cost Sheet in which some factory level attributes are added to the translated data received via some costing calculations that give us a total Cost incurred by the company.

**4.1.3 LOGICAL DATA MODEL**

Season

Product

Sourcing

Configuration

Specification

Materials

Costing

DataFlow

Bill of Materials

Costsheet

**Fig.3 Logical Data Model for FlexPLM**

As it is evident from the above diagram that Season module contains the product module. The product module contains the sourcing configuration module that contains two sub-modules that are costing and specification respectively. Specification contains the materials module that has a Bill of Materials section which is the source from which the customization extracts the data and applies transformation logic to it. The costing module contains the costsheet module in which the transformed data flows into from the source BOM section.

**4.1.4 CUSTOMIZATION PROCESS FLOW**

Persistence Event from WindChill FlexPLM

Bill Of Materials

(BOM) Module

Server Side Plugin

***findBOMPartsforOwner()***

***findMOACollectionData()***

Transformation (Mapping of Data)

Extraction from BOM

Loading of Data to costSheet

***findFlexBOMData()***

***setValue()***

CostSheet

**Fig4. FlexPLM Customization Process Flow**

**4.1.5 Table 1. CONFIGURATION CHANGES**

|  |  |  |
| --- | --- | --- |
| **Module** | **Attributes** | **Configurations** |
|  |  |  |
| Product | Product Name | Name to be derived from Number and Style |
|  | Product Number |  |
|  | Product Style Description |  |
|  | Status |  |
|  | Product Developer |  |
|  | Product Category | Users to select from Dropdown |
|  | Carryover Product | Users to check off |
|  | Target Retail |  |
|  | Target Retail Margin |  |
|  | Comments |  |
|  |  |  |
| BOM | Fabrics / Trims / Finishes as Sections |  |
|  |  |  |
| Cost Sheet | Cost Sheet Name |  |
|  | Cost Sheet Number |  |
|  | Status |  |
|  | Description |  |
|  | Fabric Cost | From BOM Fabric Section |
|  | Fabric Cost Adjustment | Adjustment by User |
|  | Total Fabric Cost | Total of BOM Section + Adjustment |
|  | Trim Cost | From BOM Trim Section |
|  | Trim Cost Adjustment | Adjustment by User |
|  | Total Trim Cost | Total of BOM Section + Adjustment |
|  | Finishes Cost | From BOM Finishes Section |
|  | Finishes Cost Adjustment | Adjustment by User |
|  | Total Finishes Cost | Total of BOM Section + Adjustment |
|  | Total Raw Material Cost | Sum of All Section level Cost |
|  | CMT Cost |  |
|  | Doc Charge |  |
|  | Misc. Cost |  |
|  | Factory Cost |  |
|  |  |  |
| Sourcing Config | Vendor |  |
|  | Comment |  |
|  | Date Tech Pack Sent |  |
|  |  |  |
| Specification | Name |  |
|  | Status |  |
|  |  |  |
| Season | Brand = ABC / XYZ / MNO |  |

**4.1.6 ALGORITHM FOR SSP CUSTOMIZATION**

|  |
| --- |
| **Name: costSheetPlugin** |
| **Package: com.lcs.customfiles.wc.costsheet** |
| **Description: ssp created for extraction of BOM data** |
| **Extends: nil** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method Name & Type** | **Description** | **Parameters** | **Returns** | **Exception thrown** |
| dataflow | Extraction of required attributes from bom and loading to costsheet | WTObject | void | wtexception |
| * Check if the WTObject is an instance of costSheet. * Get the corresponding product and sourcingCOnfig Id. * Get the collection of BOMparts using the product and BOM type. * Iterate through the collection and get the collection of BOM data of required BOMpart using product and sourcing Id. * Initialize a variable called bomCost to 0. * Get the Business Object created for mapping logic. * Iterate through the collection of bomData and get the corresponding section value. * Compare the section value with the section column in the businessObject . * If it equals with any value, then get the corresponding costSheet attribute value from the businessObject. * Get the corresponding bomLink cost and set it to the costSheet attribute returned in the previous step. * The above steps are repeated for other BOMLinks in the required BOMpart. | | | | |

**Property file entries**

* Custom3.plugin.properties
  + TARGET CLASS- com.lcs.wc.sourcing.LCSCostSheet
  + PLUGIN CLASS- com.customfiles.wc.costsheet.CostSheetPlugin
  + PLUGIN METHOD- dataFlow
  + EVENT- PRE\_CREATE\_DERIVE
  + PRIORITY- 1

**4.1.7 SYSTEM INTERFACE DESCRIPTION**

FlexPLM is extension of Windchill model [4]. FlexPLM package contains of four major components

* Flexible Data Definition
* Standard Client/Server Architecture
* Standardized Data Representation
* Data Retrieval

**Flexible Data Definition-**

* FlexPLM allows for extension of a supplied set of PBOs.
* User creates a hierarchal structure by defining types and adding attributes to the types and subtypes.
* No need to modify Object models, code, or database.
* “Flextyped” interface provides this functionality.
* Associates attributes to generic storage locations.
* Accommodates assignment of attribute value types.
* It Consists of 3 core classes:
* **FlexType**: defines a particular classification of an object. E.g. References
* **FlexTypeAttribute**: defines an attribute on a FlexType instance.
* **FlexTyped** **interface**: to enable the classification of an object type, the Flextyped interface must be implemented by it.

**Standard Client/Server Architecture-**

* FlexPLM architecture based on MVC design pattern.
* Each PBO in FlexPLM implemented as an MVC Module.
* Module: Patterned set of functional components working together to implement one core business object or relationship.

**MVC** Design pattern

**Model** :

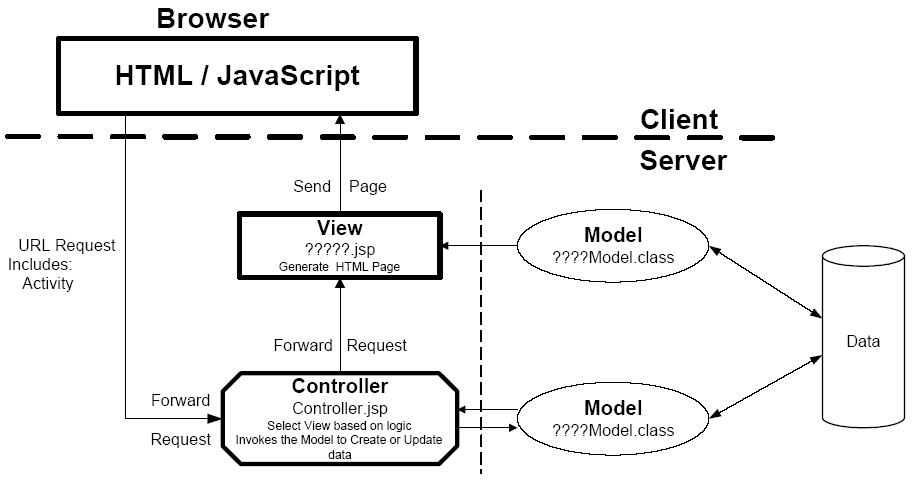
* The core of the application. This maintains the state and data that the application represents. When significant changes occur in the model, it updates all of its views.
* A bridge between View, Controller and database.
* Java Beans for each PBO.

**Controller** :

* The user interface presented to the user to manipulate the application.
* Routing requests and response, must know what is requested and what should be shown.
* JSP

**View** :

* The user interface which displays information about the model to the user. Any object that needs information about the model needs to be a registered view with the model.
* HTML+JSP, DHTML, ASP.
* JSP+ embedded JAVA code + Javascript + HTML



**Fig.5 MVC implementation of FlexPLM**

**Standardized Data Representation-**

* Consistent and configurable presentation achieved.
* GUI rendering Tools and methods offered by FlexPLM.
* Remove all formatting and implementation from the forms, tables, borders, headers, and footers.
* Each variable type of an attribute has standard display mechanism associated.
* This display mechanism is derived from the API Toolkit

**Data Retrieval-**

* Simplified and direct Query capabilities built on JDBC
* DB Query API will do the following:

1. Building of a SQL query – SQL Statement
2. Setting of Criteria within the query – Criteria
3. Joining of Tables = JoinCriteria
4. Execution of the query – query
5. Connection to the Database – DBConnectionManager

**4.2 CODES AND STANDARDS**

Engineering Standards used are:

1.   830-1998 – IEEE Recommended Practice for Software Requirements Specifications

2.   1074-1997 - IEEE Standard for Developing Software Life Cycle Processes

3.  12207.2-1997- IEEE/EIA Standard: Industry Implementation of International Standard

ISO/IEC 12207:1995 Standard for Information Technology-- Software Life Cycle

Processes--Implementation considerations. [5]

4.   Google Java Style for java coding standards [6] [9]

* Package names are all lowercase, with consecutive words simply concatenated together (no underscores). For example, com.example.deepspace, not com.example.deepSpace or com.example.deep\_space.
* Class names are written in UpperCamelCase.Class names are typically nouns or noun phrases. For example, Character or ImmutableList. Interface names may also be nouns or noun phrases (for example, List), but may sometimes be adjectives or adjective phrases instead (for example, Readable).
* Test classes are named starting with the name of the class they are testing, and ending with Test. For example, HashTest or HashIntegrationTest.
* Method names are written in lowerCamelCase.Method names are typically verbs or verb phrases. For example, sendMessage or stop.
* Underscores may appear in JUnit test method names to separate logical components of the name. One typical pattern is test<MethodUnderTest>\_<state>, for example testPop\_emptyStack. There is no One Correct Way to name test methods.

**4.3 CONSTRAINTS, ALTERNATIVES AND TRADEOFFS**

The engineering constraints used are [7]:

1. **Economic constraint**- The cost of implementing FlexPLM and Windchill technologies is very expensive and can only be afforded by high budget companies like Nike, Ralph Lauren and Abercrombie that sell luxury products which are bought by a very less number of customers.

2. **Legality constraint**- The FlexPLM is a third party owned package. The source code for implementing the package is bound and protected by some confidentiality clauses and information rights. The client can only get an abstracted view of the package.

**4.3.1 ASSUMPTIONS**

* The assumption made is that all the other modules besides BOM and CostSheet have been pre-implemented which includes product, material, sourcingconfig and specification.
* The system is configured by creating libraries for vendors, materials.
* There will be a mapping logic created through business object having an attribute of type multi-object.
* There will only be a single BOM created to the corresponding sourcing config and product.
* There will only be a single business object created under the sub-type created for using mapping logic.

**4.3.2 DEPENDENCIES**

* The target system should have same FlexPLM configuration as the source system.
* BOM, CostSheet and other modules are dependent on specification, season and product modules. For costsheet to be created there should be a specification created under required sourcing config.
* Extraction of BOM requires the product and sourcingconfig Id related to the costsheet which are used as parameters in the methods findBOMPartsforOwner( ) and findFlexBOMData( ).
* The sections inside BOM are related to the corresponding section column in the business object which is created for storing mapping logic.

**4.3.3 ALTERNATIVE SYSTEM DESIGNS**

The FlexPLM package used by us has been implemented on a Server Side Plugin. The alternative design is to implement the system on a Client Side Plugin [8]. The differences between the two designs are as follows:

|  |  |
| --- | --- |
| **Server Side Plugin** | **Client Side Plugin** |
| 1. A modular piece of code that enforces business logic and is executed against server-side components of the FlexPLM application. | 1. A modular piece of code that enforces business logic and is executed against User-Interface components of the FlexPLM application. |
| 2. These pieces of code are executed on the FlexPLM application server and typically apply business logic during persistence functionality. | 2. JSP design pattern whose purpose is to generate Javascript code that is inserted into a generated FlexPLM HTML page. |
| **Server Side Plugin** | **Client Side Plugin** |
| 3. SSP are Java methods written in Java classes as POJOs (Plain Old Java Objects) that are called when persistence events happen to an object. SSP are a means of ‘plugging in’ logic on the server without impacting or replacing existing out-of-the-box-logic. | 3. These pieces of code are executed on the client machine, in the user’s web browser. |

Although there are economic and legality constraints with the FlexPLM process, both of these constraints are considered as tradeoffs with respect to the vast number of advantages FlexPLM provides like:

* Reduced cycle time -- ensures on-schedule product launches and first-mover market-share advantages
* Improved quality -- boosts brand value and customer loyalty, and enables premium pricing
* Increased efficiency and process optimization -- drives team productivity and enhances breadth of portfolio
* Improved regulatory compliance for market segments such as consumer products -- ensures adherence to mandates such as REACH and the Consumer Products Safety Act; and, Reduced direct material cost -- improves margins and profit contribution.

**5. SCHEDULES, TASKS AND MILESTONES**

There are three major milestones as well as sever smaller tasks that must be achieved in order to reach the milestones. The three milestones are:

* Extracting Costing specific information from a FlexPLM BOM(Bill of Materials)
* Transformation logic is to be applied i.e. the data to be applied on Cost Sheet would also be configurable via properties file or FlexPLM Type Manager Configurations.
* Transformed / Translated costing data should be applied to respective attributes on Costsheet.

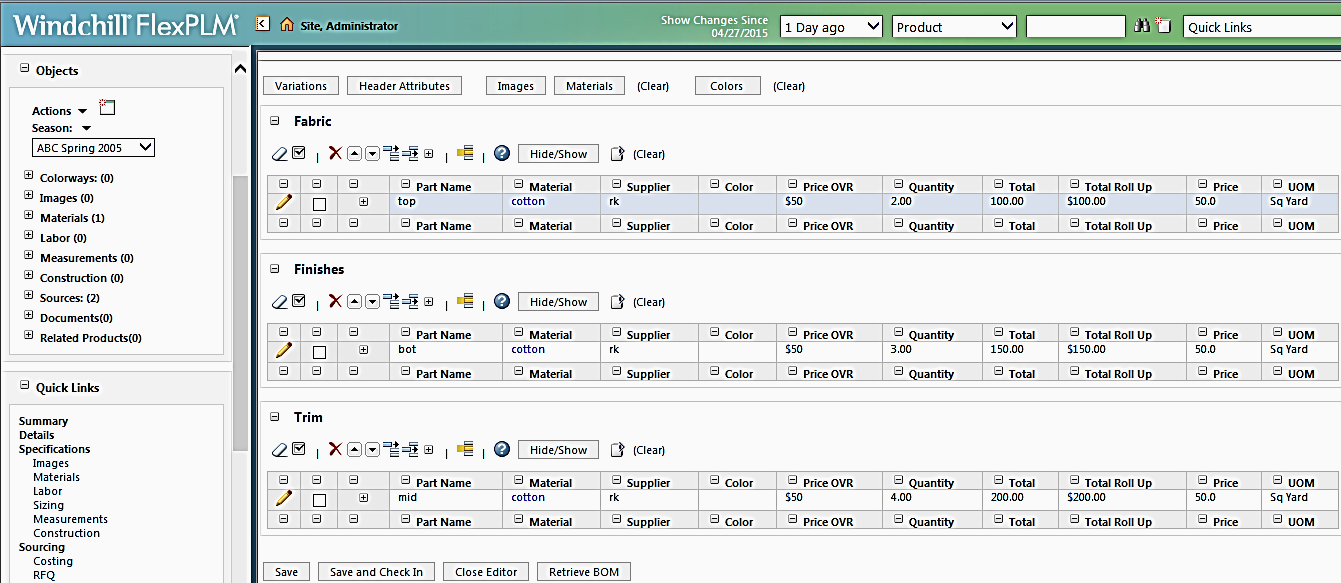
**6. PROJECT DEMONSTRATION**

As there is no automated process in windchill flexPLM for the dataflow between different flex objects, the most important specification for the project to meet is to maintain the dataflow between costsheet and BOM .The main objective of the project is to provide the user with the cost specific information of different sections of BOM to be displayed in the cost sheet.

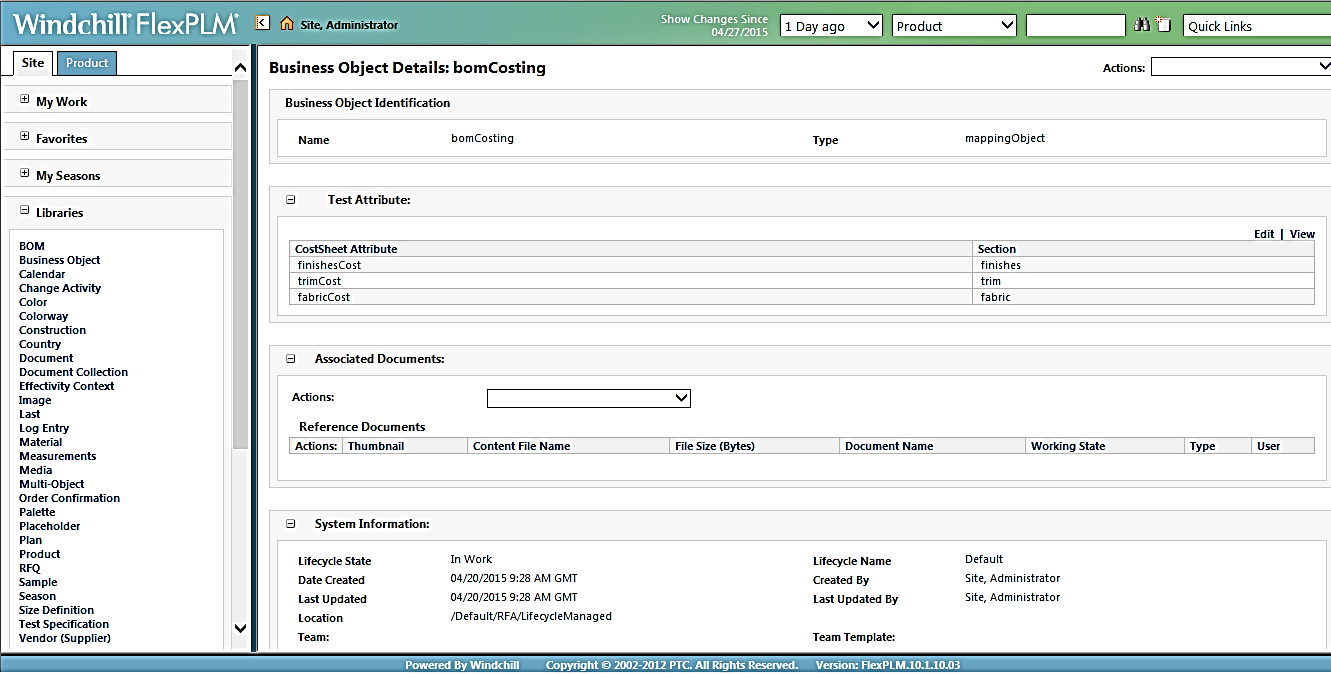
Apart from the dataflow between these two modules the customization should be in such a way that the user should be able to configure any new attributes, sections and also maintain the dataflow between them. The data to be applied on Cost Sheet should also be configurable via properties file or FlexPLM Type Manager Configurations.

A FlexPLM Costsheet may have other attributes and Factory level costing calculations. Such calculations are out of scope for the purpose of this project. But this customization should be implemented with a higher priority of execution, where in additional custom logic can be added on Cost Sheets to calculate a total Cost.

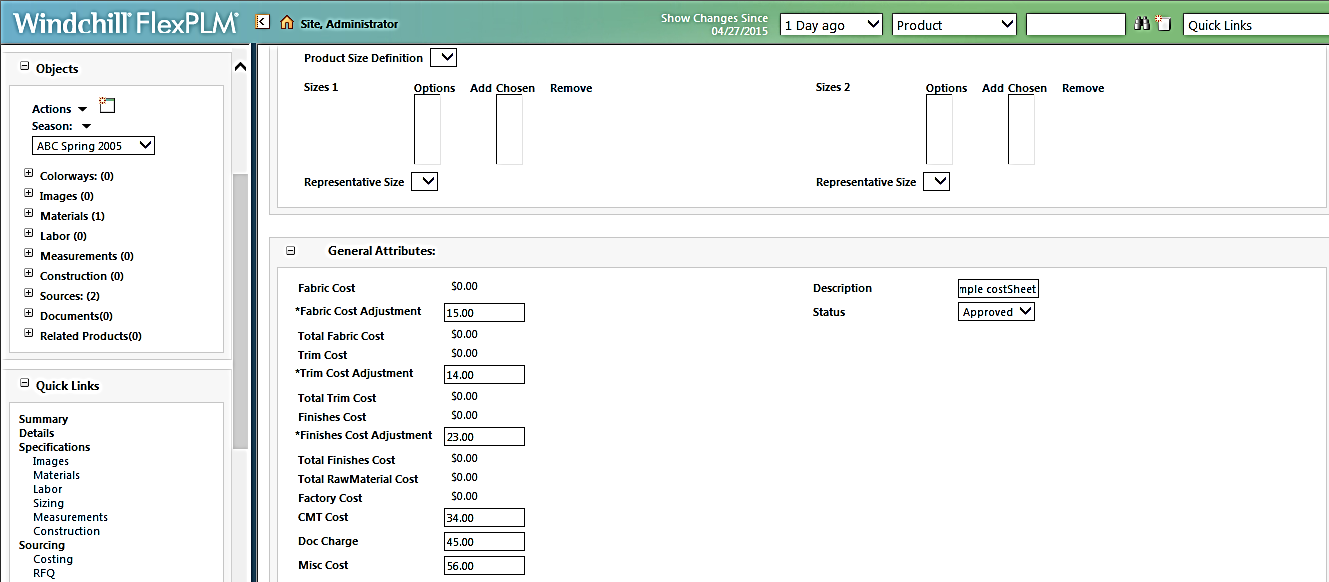
**6.1 SCREENSHOTS**



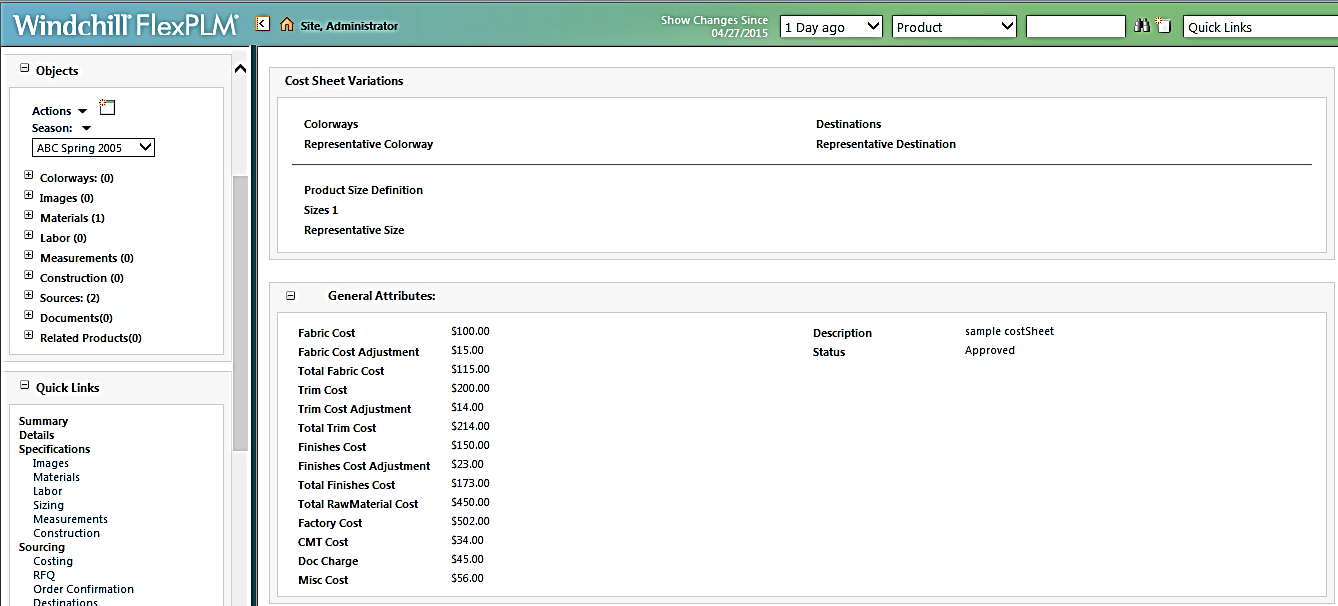
**Fig.6 Bill Of Materials**

****

**Fig.7 Mapping Logic in Business Object**

****

**Fig.8 CostSheet Before Saving**

****

**Fig.9 Final CostSheet**

**7. MARKETING AND COST ANALYSIS**

**7.1 COST ANALYSIS**

Windchill FlexPLM is a loaned asset software and will be provided by the client for the developers to develop the package as per the requirements. So once the package is developed with necessary customizations and configurations the customers do not have to go through the development phase for multiple instances of manufacturing there by saving a lot of cost. And hence the project is very economical for the customers to invest on.

The package shall form a base package and will find a use with various clients of the industry and can be implemented with ease after due customization as per their requirements. The package can be further modified with the dynamic demands of the environment where the client operates and with margins shrinking due to competition and demands from their customers.

Thus optimization of cost shall play a very important role in near future , which can be cut down substantially by using this package.

**8. SUMMARY**

The objective of flowing the line level cost information from a Bill Of Materials to Cost Sheet executed on a Server Side Plugin has been completed successfully. Some more additional customizations and configurations will be implemented in the FlexPLM package in the near future due to various factors arising in the industry using this package.

The various factors identified as of now which would form the base for customizing this package are:

* Dynamically changing business environment for the user.
* Shrinkage of profit margins arising due to competition.
* The market emerging as a buyers market solely in contrast to the sellers market earlier.

Thus FlexPLM , an intuitive web based package solution, should be viewed as an information strategy, an enterprise strategy, a transformational business strategy and ultimately, as a comprehensive approach to innovation built on enterprise-wide access to a common repository of product information and processes.

**9. REFERENCES**

1. http://www.plm.automation.siemens.com/en\_us/plm/

2. http://en.wikipedia.org/wiki/Product\_lifecycle\_management

3. http://www.ptc.com

4. http://www.plm-training.com

5. https://standards.ieee.org/findstds/standard/software\_and\_systems\_engineering.html

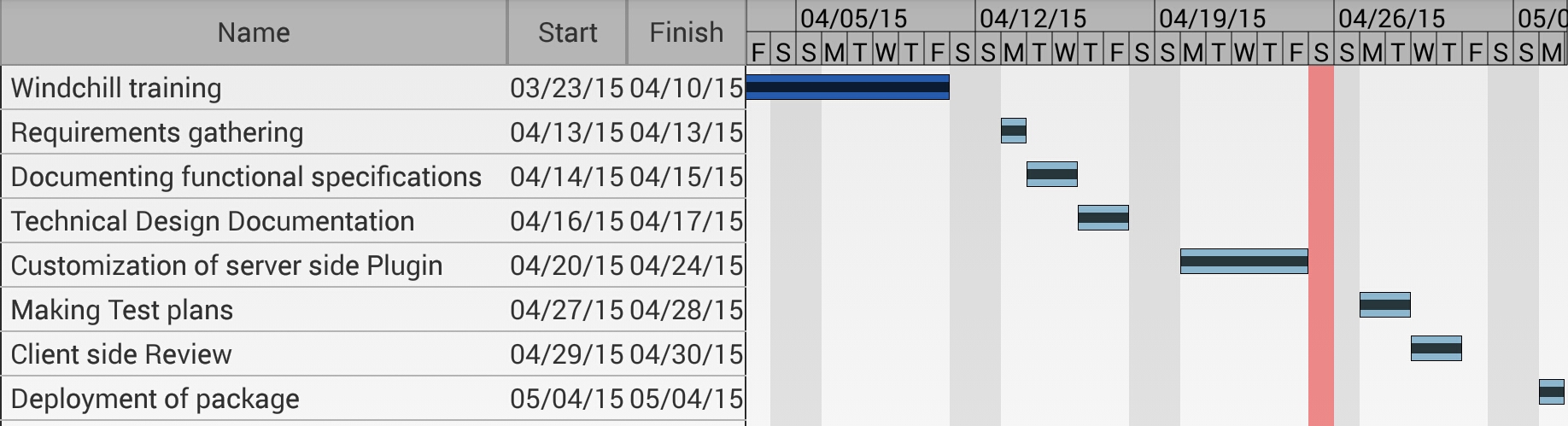
6. https://google-styleguide.googlecode.com/svn/trunk/javaguide.html

7. http://www.product-lifecycle-management.com

8. http://www.motive.co.nz/glossary/client-server.php

9. http://www.tutorialspoint.com/java/java\_files\_io.htm

**APPENDIX**



**Fig. 10 Gantt Chart**

**SOURCE CODE FOR SSP CUSTOMIZATION**

public class CostSheetPlugin {

@SuppressWarnings("unchecked")

public static void dataFlow(WTObject wtobject){

try{

if(wtobject instanceof LCSCostSheet){

Double bomCost=0.0;

String key=null;

/\*Type casting WTObject to LCScostsheet.

\* Retrieving product and source id from Costsheet

\*/

LCSCostSheet costSheet = (LCSCostSheet)wtobject;

LCSProduct product = (LCSProduct)VersionHelper.latestIterationOf(costSheet.getProductMaster());

String sourceId=Double.toString(costSheet.getSourceMasterId());

/\*

\* Setting system generated variable names of BOM attributes to readable variable names.

\*/

FlexTypebom FlexType=FlexTypeCache.getFlexTypeFromPath("BOM\\Materials\\Apparel");

FlexTypeAttribute price=bomFlexType.getAttribute("priceOverride");

FlexTypeAttribute quantity=bomFlexType.getAttribute("quantity");

FlexTypeAttribute section=bomFlexType.getAttribute("section");

String bomQuantity=quantity.getVariableName().toUpperCase();

String bomPrice=price.getVariableName().toUpperCase();

String bomSection=section.getVariableName().toUpperCase();

/\*

\* Setting system generated variable names of Multi-Object attributes to readable variable names.

\*/

FlexTypemultiObjectFlexType=FlexTypeCache.getFlexTypeFromPath("Multi-Object\\bomCostingDataFlow");

FlexTypeAttribute moaSection1=multiObjectFlexType.getAttribute("scope");

FlexTypeAttribute costSheetAtt=multiObjectFlexType.getAttribute("costSheetAttribute");

String costSheetAttribute=costSheetAtt.getVariableName().toUpperCase();

String moaSection=moaSection1.getVariableName().toUpperCase();

/\*

\* Fetching the Business object which was created to store mapping logic.

\*/

LCSLifecycleManagedClientModel lifeCycle=new LCSLifecycleManagedClientModel();

LinkedHashMap map=new LinkedHashMap();

FlexTypebusinessObjectFlexType=FlexTypeCache.getFlexTypeFromPath("Business Object\\mappingObject");

SearchResultsbusinessObjectResults=new LCSLifecycleManagedQuery().findLifecycleManagedsByCriteria(map, businessObjectFlexType, null, null, null);

Vector businessObjectCollection=new Vector(businessObjectResults.getResults());

FlexObject businessObject=(FlexObject)businessObjectCollection.firstElement();

WTObject obj=lifeCycle.findByOid("OR:com.lcs.wc.foundation.LCSLifecycleManaged:"+businessObject.getString("LCSLIFECYCLEMANAGED.IDA2A2"));

SearchResults multiObjectResults=LCSMOAObjectQuery.findMOACollectionData(obj, businessObjectFlexType.getAttribute("attribute"));

Vector multiObjectCollection=new Vector(multiObjectResults.getResults());

/\*

\* Fetching BOM Parts for product owner and the BOM Data.

\*/

LCSFlexBOMQuery query=new LCSFlexBOMQuery();

Vector bomParts= new Vector(query.findBOMPartsForOwner(product,"MAIN"));

Iterator i=bomParts.iterator();

while(i.hasNext()){

SearchResults bomResults=LCSFlexBOMQuery.findFlexBOMData((FlexBOMPart)i.next(), sourceId, null, null, null, null, null, null, false, true, null, null, null, null);

Vector bomLinkCollection= new Vector(bomResults.getResults());

for(int j=0;j<bomLinkCollection.size();j++){

FlexObject bomLink=(FlexObject) bomLinkCollection.get(j);

key=bomLink.getString("FLEXBOMLINK."+bomSection);

for(int x=0;x<multiObjectCollection.size();x++){

FlexObject multiObject=(FlexObject)multiObjectCollection.get(x);

String attKey=multiObject.getString("LCSMOAOBJECT."+costSheetAttribute);

/\*

\* Checking the attributes with mapping logic and setting it to costSheet.

\*/

if(key.equalsIgnoreCase(multiObject.getString("LCSMOAOBJECT."+moaSection))){

bomCost=(bomLink.getDouble("FLEXBOMLINK."+bomPrice)\*bomLink.getDouble("FLE

XBOMLINK."+bomQuantity))+(Double)costSheet.getValue(attKey);

costSheet.setValue(attKey, bomCost);

}

}

}

}

}

}

catch(Exception e){

e.printStackTrace();

}

}

}