Additional Productivity Features For Human Resources Management Systems Benefits

A graduate project proposed in partial fulfillment of the requirements
For the degree of Master of Science in Software Engineering

By

Sphoorti Madugula

December 2011
The graduate project of Sphoorti Madugula is approved by:

_______________________________  __________________________
Robert McIlhenny                  Date

_______________________________  __________________________
Peter Gabrovsky                   Date

_______________________________  __________________________
Robert Lingard, Chair             Date

California State University, Northridge
DEDICATIONS

This project is dedicated to Prasad and Padmaja Bodla, who have inspired and helped me immensely throughout this project; and to Professor Robert Lingard, for all the guidance and support from the beginning to completion of the project.

This project is as much theirs, as it is mine.
ACKNOWLEDGEMENTS

I am indebted to my parents for their patience, guidance and assistance in providing the necessary resources enabling me to work on this project. I am also grateful to all the stakeholders who have helped me with ideas and suggestions on what features to implement in this project, namely Prasad Bodla, Padmaja Bodla, Srinath Parepally, and Sreenath Gajulapalli.

I would also like to thank Professors Peter Gabrovsky and Robert McIlhenny for being on the Committee of Board Member for my project.
**TABLE OF CONTENTS**

SIGNATURE PAGE ......................................................................................................................... ii
DEDICATIONS ................................................................................................................................. iii
ACKNOWLEDGEMENTS .................................................................................................................... iv
TABLE OF CONTENTS ..................................................................................................................... v
TABLE OF FIGURES .......................................................................................................................... viii
LIST OF TABLES ............................................................................................................................... xii
ABSTRACT ............................................................................................................................................ xiii
Introduction ........................................................................................................................................ 1
Oracle E-Business Suite .................................................................................................................. 1
Oracle HRMS ...................................................................................................................................... 1
Oracle Advanced Benefits .............................................................................................................. 2
Statement of the Problems ............................................................................................................... 6
Software Development Life Cycle .................................................................................................... 7
Motivation behind this Project ......................................................................................................... 8
Specific Tools and Technologies Required To Do This Project ...................................................... 9
Project Complexity ........................................................................................................................... 10
Requirements and Specifications ....................................................................................................... 12
Identification of Stakeholders .......................................................................................................... 12
Feature List ......................................................................................................................................... 13
Use Cases ......................................................................................................................................... 14
Use Case 1 .......................................................................................................................................... 14
Use Case 2 .......................................................................................................................................... 16
Use Case 3 .......................................................................................................................................... 18
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conclusion</td>
<td>54</td>
</tr>
<tr>
<td>Textual References</td>
<td>57</td>
</tr>
<tr>
<td>Image References</td>
<td>58</td>
</tr>
<tr>
<td>Appendix A: Stakeholder Interviews</td>
<td>59</td>
</tr>
<tr>
<td>Interview 1</td>
<td>59</td>
</tr>
<tr>
<td>Interview 2</td>
<td>63</td>
</tr>
<tr>
<td>Interview 3</td>
<td>67</td>
</tr>
<tr>
<td>Interview 4</td>
<td>71</td>
</tr>
<tr>
<td>Appendix B: Code</td>
<td>75</td>
</tr>
<tr>
<td>Feature Set 1</td>
<td>75</td>
</tr>
<tr>
<td>Feature Set 2</td>
<td>82</td>
</tr>
<tr>
<td>Appendix C: Tables and Insert Statements</td>
<td>93</td>
</tr>
<tr>
<td>Feature 1</td>
<td>93</td>
</tr>
<tr>
<td>Feature 2</td>
<td>94</td>
</tr>
<tr>
<td>Appendix D: Screenshots</td>
<td>96</td>
</tr>
<tr>
<td>Feature 1</td>
<td>96</td>
</tr>
<tr>
<td>Feature 2</td>
<td>119</td>
</tr>
</tbody>
</table>
TABLE OF FIGURES

Figure 1: Oracle E-Business Suite R12 ................................................................. 1
Figure 2: Oracle HRMS Information Model .......................................................... 1
Figure 3: Oracle HRMS Modules ......................................................................... 2
Figure 4: High Level Data Integration in Oracle HRMS ....................................... 4
Figure 5: An Example of Health Benefit Rates ...................................................... 6
Figure 6: High-level view of Oracle HRMS .......................................................... 25
Figure 7: High-level Logical Model of the Oracle Advanced Benefits Module ....... 27
Figure 8: Setting up Plan Options ........................................................................ 29
Figure 9: Matching Plan Options to Plans ............................................................. 30
Figure 10: Plan Design of SM Employee Benefits Program .................................. 31
Figure 11: Entity Relationship Diagram for the Entities in Feature 1 – Part 1......... 33
Figure 12: Entity Relationship Diagram for the Entities in Feature 1 - Part 2 ......... 34
Figure 13: Entity Relationship Diagram for the Entities in Feature 2 .................... 37
Figure 14: Summary of Plan Design .................................................................... 42
Figure 15: Life Event Reasons – Creating a life event reason to enroll a certain person in Benefits. In this case, the life event reason created was for a New Hire ......................... 96
Figure 16: Plan Types – Creating a Medical Plan Type ........................................ 97
Figure 17: Plan Types – Creating a Dental Plan Type .......................................... 97
Figure 18: Plan Types – Creating a Vision Plan Type ............................................ 98
Figure 19: Plan Types – Creating a Life Insurance Plan Type ............................... 98
Figure 20: Plan Types – Creating a Spouse Life Insurance Option ...................... 99
Figure 21: Plan Types – Creating a Dependent Life Insurance Option ................ 99
Figure 22: Options – Creating an Employee Only Option .................................. 100
Figure 23: Options – Creating an Employee Plus Spouse Option ....................... 100
Figure 24: Options – Creating an Employee Plus Family Option .............................................. 101
Figure 25: Options – Creating a Waive Option ........................................................................ 101
Figure 26: Plans and Options– Creating the United Health Care (UHC) Medical Plan and
applying the Options, SM Employee Only, SM Employee Plus Family, SM Employee Plus
Spouse, and SM Waive to the plan. ......................................................................................... 102
Figure 27: Plans and Options– Creating the Kaiser Medical Plan and applying the Options, SM
Employee Only, SM Employee Plus Family, SM Employee Plus Spouse, and SM Waive to
the plan .................................................................................................................................. 103
Figure 28: Plans and Options– Creating the Cigna Medical Plan and applying the Options, SM
Employee Only, SM Employee Plus Family, SM Employee Plus Spouse, and SM Waive to
the plan .................................................................................................................................. 104
Figure 29: Plans and Options– Creating the Cigna Dental Plan and applying the Options, SM
Employee Only, SM Employee Plus Family, SM Employee Plus Spouse, and SM Waive to
the plan .................................................................................................................................. 105
Figure 30: Plans and Options– Creating the United Health Care (UHC) Dental Plan and applying
the Options, SM Employee Only, SM Employee Plus Family, SM Employee Plus Spouse,
and SM Waive to the plan ........................................................................................................ 106
Figure 31: Plans and Options– Creating the Kaiser Dental Plan and applying the Options, SM
Employee Only, SM Employee Plus Family, SM Employee Plus Spouse, and SM Waive to
the plan .................................................................................................................................. 107
Figure 32: Plans and Options– Creating the Cigna Vision Plan and applying the Options, SM
Employee Only, SM Employee Plus Family, SM Employee Plus Spouse, and SM Waive to
the plan .................................................................................................................................. 108
Figure 33: Plans and Options– Creating the Kaiser Vision Plan and applying the Options, SM
Employee Only, SM Employee Plus Family, SM Employee Plus Spouse, and SM Waive to
the plan .................................................................................................................................. 109
Figure 34: Plans and Options—Creating the United Health Care (UHC) Vision Plan and applying the Options, SM Employee Only, SM Employee Plus Family, SM Employee Plus Spouse, and SM Waive to the plan.

Figure 35: Plans – Creating the Employee Supplemental Life Insurance plan.

Figure 36: Plans – Creating the Spouse Supplemental Life Insurance plan.

Figure 37: Plans – Creating the Dependent Supplemental Life Insurance plan.

Figure 38: Programs – Making the SM Employee Benefits Program.

Figure 39: Plan and Plan Type – Applying all the plans created to the SM Employee Benefits Program.

Figure 40: Plan and Plan Type – Applying all the plans created to the SM Employee Benefits Program.

Figure 41: Setting up the Enrollment Coverage Start and End Dates.

Figure 42: Creating a Life Event titled SM New Hire.

Figure 43: Applying Timings for the Life Event as far as when Coverage Ends.

Figure 44: Applying Start and End Dates for the SM New Hire Life Event.

Figure 45: Putting the Life Event SM New Hire into Motion.

Figure 46: Selecting a Random Imaginary Employee, Tamara Aaron, and applying the Life Event SM New Hire, and Enrolling her into Plans and Options.

Figure 47: Error message displayed when user attempted to enroll a Spouse in Life Insurance before enrolling himself in the Employee Only Life Insurance first.

Figure 48: Error message displayed when user attempted to enroll himself in a Dental plan before enrolling himself in the Medical plan first.

Figure 49: Creating a new life event called SM Move Enrollments for the movement of the benefits enrollments.

Figure 50: Applying the SM Move Enrollments Life Event reason to the SM Employee Benefits Program.
Figure 51: Test employee: Mrs. Tamara Aaron – who was previously enrolled into the SM Benefits Program under the Life Event Reason - “New Hire” is backed out of the Life Event since only one life event can be happening to an employee at a time......................... 120

Figure 52: Test employee: SM New Hire Life Event is voided.................................................. 121

Figure 53: Mrs. Tamara Aaron – who was previously enrolled into the SM Benefits Program under the Life Event Reason - “New Hire” is switched to the “Move Enrollments” Life Event .............................................................. 121

Figure 54: Life event has been processed for this employee ......................................................... 122

Figure 55: Existing Medical Plan: SM Kaiser Medical SM Employee Only. .......................... 122

Figure 56: Successful movement of the Enrollment Plan and option: From Kaiser Medical Employee Only to Cigna Medical Employee Only................................................................. 123
**LIST OF TABLES**

Table 1 Features that are going to be implemented in this project ........................................... 13

Table 2: Use Case 1 ......................................................................................................................... 15

Table 3: Use Case 2 ......................................................................................................................... 17

Table 4: Use Case 3 ......................................................................................................................... 19

Table 5: Use Case 4 ......................................................................................................................... 21

Table 6: Use Case 5 ......................................................................................................................... 23

Table 7: Use Case 6 ......................................................................................................................... 24

Table 8: Test Steps for Feature 1 .................................................................................................... 48

Table 9: Plan Design ....................................................................................................................... 50

Table 10: Test Steps for Feature 2 .................................................................................................. 53
This project intends to provide enhanced productivity features to the Benefits Module of Oracle Human Resources Management System, which is part of Oracle Enterprise Business Suite of Oracle Applications, using the Waterfall method of Software Development. The following pages describe the gathering of the requirements, design, implementation, testing and release of two primary features.

The first feature offers enhanced dependency validation between the benefits plan enrollments. This feature is very user friendly and eliminates the use of Oracle FastFormula, which are written and used for validation, to perform calculations, and to specify rules. FastFormula is a way to plug in customer code or customizations to implement the business rules. Even though FastFormula can be written in English like...
code and are simpler than programming languages like C or Java, they’re still considerably difficult for functional users, who don’t have a computer programming background, to author or easily decipher. So important requirements such as cross-plan validations and dependency checks cannot easily be implemented by functional users and business analysts using FastFormula. The best way for customers to extend the existing Oracle Applications functionality is by providing such an additional feature, which is more business user oriented, easy to understand, and can be modified easily by business users based on their requirements. In fact these validations are part of any implementation and merit being a part of the base product itself.

The second feature in this project will help automate the process of moving benefits enrollments from old benefits plans to new ones. This feature is also setup data driven, which a business user can configure to move the enrollments. It is a common practice to evaluate the benefit offerings every year. Acquisitions are also a common practice in business world, making this feature pretty much required by a majority of application users.

Considering how many employees are in each company that uses the HR Benefits module of Oracle Applications, these features are of significant use, because they would automate things that otherwise would need labor by HR professionals. Challenging aspects of this project include learning new languages, SQL and PL-SQL, working in-depth with Databases, understanding of oracle application architecture, learning technology components like Forms, OAFramework, Oracle application foundation objects like batch submission capabilities.

The software development methodology used for this project is a Linear Project
Management Life Cycle, or the waterfall method. The waterfall method is the best one to use for this project because the number of features that are going to be implemented is limited, and they are not subject to any change.
Introduction

Oracle E-Business Suite

Oracle E-Business Suite (EBS), also known as Applications/Apps or EB-Suite, provides a comprehensive suite of integrated global business applications providing software solutions to Human Resources, Financials, Manufacturing, Customer Relationship Management and Industry Verticals. EBS uses Oracle's core Oracle relational database management system (RDBMS) technology. Oracle EBS is used to automate the business processes in corporations of various sizes worldwide.

The current version of Oracle EBS, 12.1, consists of the following product lines, each of which comprises a number of modules: Oracle Customer Resource Management, Oracle Financials, Oracle Human Resource Management Systems (HRMS), Oracle Mobile Supply Chain Applications, Oracle Order Management, Oracle Procurement, Oracle Project Portfolio Management, Oracle Quotes, Oracle Transportation Management, Oracle Warehouse Management Systems, Oracle Inventory, and Oracle Enterprise Asset Management. [1]

Oracle HRMS

[2] Oracle HRMS allows an organization to create an information model to represent their employees' compensation, benefits, employment, and other related data.
enterprise in terms of Work, Pay and People. [Figure 2]. Specifically this system enables an organization to enter and track employees and contingent workers from the day they apply for a job until the day they leave the company and beyond as retirees. HRMS is used to hold, inquire, report and keep track of personal and employment information such as job details, benefit and compensation, work schedules, personal contacts, and preferred language for correspondence. At its core, it maintains basic personal information such as date of birth, employee number, social security number, marital status or family related information and addresses. Oracle HRMS can be deployed to configure the same HR system to represent many different types of ventures, for example, commercial, public sector, healthcare, telecom and education. There are several sub-modules within HRMS as shown below that includes Advanced Benefits. [3]

![Oracle HRMS Modules](image)

**Oracle Advanced Benefits**

Oracle Advanced Benefits (OAB) is a flexible, rules-based benefits administration application that enables improved benefits data management and analysis while reducing overall costs through easy setup and process automation. This product is widely used in Business Process Outsourcing (BPO) to individual large to small corporations. OAB
enables professionals to adapt to evolving business needs by simplifying the implementation, management, and monitoring of benefits offerings. [4]

In any company, every employee is eligible for certain benefits such as Medical insurance, Dental insurance, Life Insurance, Pension Plans and so on. While the employer provides certain benefits other benefits require an employee to contribute, the OAB module basically helps handle the entire advanced benefits administration effectively. [5]

Figure 4 displays how the OAB module is structured, and where it lies in reference to other modules in Oracle Human Resources Management System. [8]
The OAB module defines the benefits plans, eligibility, rates and associated business rules. This module provides the capability to manage the enrollments and associated interfaces to the service providers. Technically, the OAB module contains a forms-based interface to setup and manage the benefits plans a professional and self-service interface
to manage participant enrollments and batch processes to manage the eligibility and rate evaluations.

*The OAB structure consists of benefits programs, plans, plan types, options, enrollments, enrollment rates and beneficiaries. Plans are the benefits plans that are available under a program. Plan types include categories like medical, dental, vision, spouse life, dependent life, 401K etc. Options available for the employees to choose from may consist of Employee Only, Employee plus Family, Employee plus Spouse, Employee plus Dependents etc. With the help of eligibility criteria set up in the system, the system determines which employee becomes eligible for which benefit programs, plans, options and associated rates. Life events are set up in the system to help the employee to opt for enrollment changes. For example, an employee who is single can opt for the employee plus spouse benefits when married. Therefore, in this case, the life event is Marriage.*

[5]

Although the OAB module is a very flexible and highly automated product, additional features can be added to enhance the current functionality to provide user friendly validations and automated processes. These validations and processes in turn reduce the dependency on the development team during the implementation and ongoing usage of the product. Sample plan design is specified below.
Statement of the Problems

One of the problems that benefits administrators face is evaluating the current benefits offerings, like plans, provided to employees year after year. Reasons such as reducing the expenses or a carrier discontinuing the plans may necessitate ending the existing benefits plans and introducing new ones. It may be necessary to move all participants from the old plans to new plans before open enrollment begins. This module intends to solve this problem by automating the process of deactivating the old plan for future enrollment and providing an analogous new plan. Any design to solve this requirement should be open, extensible and reusable for future needs.
Currently, there is no easy way to build the enrollment dependencies between benefits plans. For example, a participant is not allowed to enroll in a spouse life insurance plan without enrolling in an employee life insurance plan first. To introduce this validation, a benefits administrator needs to take the help of a developer to write such rules called Fast Formulas. Fast Formulas, albeit a simple method to write using English words & basic mathematical functions, do take time and effort. This project proposes a feature such that a business analyst can define the dependency rules on a set of validation tables and a generic validation rule will interpret and apply the rules as a part of enrollment. Essentially, it will make the benefits administrator’s life easier and cut down the cost of development and also the implementation time.

**Software Development Life Cycle**

The software development methodology used for this project is a Linear Project Management Life Cycle, or the waterfall method. The waterfall method is the best one to use for this project because the number of features that are going to be implemented is limited, and they are not subject to any change. The features, functions, requirements and specifications are clearly defined and complete and the solution is known. Additionally, the scope of the project is clear, and not likely to change. For these reasons, using the traditional method of software development would be beneficial for this project. This process includes five phases:

1. **Requirements**: Come up with the use cases, deciding the software features, the constraints and interfaces.

2. **Design**: Decide how the software will satisfy the requirements, and figuring out
which design decision to make, and why.

3. **Implementation**: Come up with code.

4. **Testing**: Verify and validate the software and the results.

5. **Documentation**: This phase will last for almost the entire duration of the project. Each phase will include its own documentation.

Essentially, this project will be use-case driven. The requirements engineering phase will involve coming up with use-cases for features that are going to be implemented. Design and code will basically be the realization of these use-cases. The testing will also be done in terms of the use-cases.

**Motivation behind this Project**

This project was motivated by a desire to learn more about Database Management Systems. This project seemed like a very good opportunity to learn about Oracle Applications and Databases in general. It also involved learning a variety of things such as database structured query languages such as SQL and PL-SQL. Enterprise Business software is very complex as it includes databases, uses various languages and has a number of user interfaces. Since this project involves the use of Oracle EBS, it provides a good way of getting exposure to these technologies and it also helps to understand the implementation life cycle project management. Additionally, this project intends to implement a complete solution, which is the reason why two prime real world scenarios were chosen. It provided an opportunity to comprehend complex software solutions (Oracle Applications) and to extend them to accommodate real world programs using software development methodologies and industry leading coding standards.
Specific Tools and Technologies Required To Do This Project

1. Oracle Database Server 10g

2. Oracle Application (OA) Framework

3. **Oracle Applications (R12):** “R12 virtual machine” was purchased from the company, Focus Thread Consulting. This configuration uses the “VMware” virtual machine to run on any Personal Computer (PC), as long as the minimum memory meets 4GB. Steps involved in using this are – installing the VMware server on a machine, which is a one time job, then accessing the VMware image from VMware software. The configuration used for this project consists of a standard PC with Microsoft Windows 7 Home edition. The memory for this computer was 8GB, and hence made it easy to run VMware.

4. **VMware:** The VMware Server is a hosted virtualization platform that installs like an application on any existing server hardware and partitions a physical server into multiple virtual machines. [6]

5. **Report, Interface, Customization, Extensions (RICE) Methodology:** Functionality of the benefits product was extended using interface tables. This project proposes to provide the customizations to two key business requirements stated earlier:
   a. Automating the movement of the enrollment in plans
   b. Enforcing plan validations

6. **Internet Explorer:** Oracle Applications (R12) works best on Internet Explorer when it’s not installed as a desktop application. Its functionality is impaired when used in different browsers.
7. Editor (Notepad / Wordpad)

8. **PuTTY**: PuTTY is a free implementation of Telnet and SSH for Windows and Unix platforms, along with an xterm terminal emulator. It was originally developed by Simon Tatham. [7]

9. UNIX

**Project Complexity**

This project is very complex as it involves extending complex business applications. It involves understanding the existing system from both a functional as well as a technical perspective. From a functional perspective, one has to understand the operations of Human Resources and Benefits Management. Then, one needs to get a grip of how the Enterprise Resource Planning (ERP) system provides the functionality. One is also required to gather the business requirements and then define and figure out how to fill the functional gaps.

The second dimension of complexity is understanding the technology stack behind the ERP system. The technological components that need to be mastered are the Oracle database architecture, the Forms / Oracle Applications (OA) Framework development environment and oracle approved customization mechanisms. It involves considerable effort in designing the system based on the underlying technology stack.

In addition to making all the coding and development methodology be in line with Oracle Applications, by understanding and following the coding standards and customization best practices, one also has to ensure that the designs conform to standard. For example,
one has to learn how to make error messages follow the pattern of other error messages displayed in the application, or what type of naming conventions to follow for tables and procedures and so on.
**Requirements and Specifications**

**Identification of Stakeholders**

The stakeholders in the case of this problem are:

1. **HR Administrators**: HR Administrators maintain and manage Human Resources activities for a particular division or a company. Apart from taking care of every formality starting from the time an employee joins the company, until he quits or gets terminated, the administrators deal with the government or local agencies in complying and enforcing the local labor laws.

2. **HR Business Analysts**: Business Analysts gather the requirements and work with the end users and the developers. They validate the system based on the requirements and find the gaps or missing functionalities and find ways to design and implement them. They also validate the functionality developed before rolling out to the users.

3. **HR Techno-functional Developer**: They use the requirements documents that the Business Analyst provides to come up with detailed functional design documents.

4. **HR Director**: He is responsible for implementation of HR policies across all mediums of HR like Payroll, Learning, and Recruiting. He also lays out policies for ethics, and for compliance of harassment and equal employment opportunities and so on.

5. **IT Director**: He is in charge of finalizing the functional side designs, signing off the functional designs, project management, prioritization, resource allocation, and hiring for the projects involved.
Interviews were conducted with an IT Director, two HR Techno-functional Developers and a HR Business Analyst for gathering requirements to this project. These interviews have been listed in the Appendix for reference.

After careful consideration of the stakeholder interviews, the following items were finalized as a list of features to implement for this project.

**Feature List**

Based on the information gathered from the stakeholder interviews, the following feature list was come up with to enhance existing functionality of the OAB module:

<table>
<thead>
<tr>
<th>Proposed Feature</th>
<th>Details</th>
<th>Implementation?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Automation of switching plan enrollments</strong></td>
<td>1. Batch Process of switching employees in the existing company from the old plans to a new one. &lt;br&gt;2. Batch Process of switching employees from the newly acquired company from the old plans to a new one.</td>
<td>Yes.</td>
</tr>
</tbody>
</table>

*Table 1 Features that are going to be implemented in this project*
Use Cases

Using the feature list above, six use cases were identified. The details of use cases are listed in the following pages.

Use Case 1

<table>
<thead>
<tr>
<th>Use Case ID:</th>
<th>UC-001</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use Case Name:</strong></td>
<td>User tries to enroll in a Dental Plan and the system applies Enrollment dependency validations.</td>
</tr>
<tr>
<td><strong>Actors:</strong></td>
<td>Employee or Benefits User or HR Personnel.</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>This use case describes the steps a user takes in order to enroll in a Dental Plan and how the system should respond.</td>
</tr>
<tr>
<td><strong>Trigger:</strong></td>
<td>The user selects a Dental Plan in the SM Benefits Program and hits the save button.</td>
</tr>
</tbody>
</table>
| **Preconditions:** | 1. The user has a working internet connection.  
2. The user is logged into Oracle Advanced Benefits application.  
3. User has a started life event to make the enrollment changes.  
4. The user has queried the participant’s (employee’s) profile and can see and edit the data on benefits enrollment pages. |
| **Normal Flow:** | 1. User follows prompts to make enrollment changes for the life event in progress.  
2. User selects one of the Dental Plans in the SM Benefits Program.  
3. User clicks on the save button. |
### Alternate Flow:

1. User follows prompts to make enrollment changes for the life event in progress.
2. User selects one of the Dental Plans in the SM Benefits Program.
3. User clicks on the save button.
4. Application checks to see if the user is already enrolled in the Medical plan.
5. If user isn’t enrolled in Medical Plan under the SM Benefits Program, application displays an error message, asking the user to add a Medical Plan in the SM Benefits program first. This is the cross plan enrollment dependency validation.
6. User clicks on the ‘Ok’ button to make the error message disappear.

### Post-conditions:

1. In case of normal flow, the system saves the changes and displays the form with the updated information.
2. In case of the alternate flow, the system displays the form without saving any of the changes made. At this point, the user has the option to enroll in one of the Medical Plan available in the Benefits Program and save the data and redo the steps in the Normal Flow of this use case.

**Table 2: Use Case 1**
# Use Case 2

<table>
<thead>
<tr>
<th>Use Case ID:</th>
<th>UC-002</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use Case Name:</strong></td>
<td>User tries to enroll in a Vision Plan and the system applies Enrollment dependency validations.</td>
</tr>
<tr>
<td><strong>Actors:</strong></td>
<td>Employee or Benefits User or HR Personnel.</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>This use case describes the steps a user takes in order to enroll in a Vision Plan and how the system should respond.</td>
</tr>
<tr>
<td><strong>Trigger:</strong></td>
<td>The user selects a Vision Plan in the SM Benefits Program and hits the save button.</td>
</tr>
</tbody>
</table>
| **Preconditions:** | 1. The user has a working internet connection.  
2. The user is logged into Oracle Advanced Benefits application.  
3. User has a started life event to make the enrollment changes.  
4. The user has queried the participant’s (employee’s) profile and can see and edit the data on benefits enrollment pages. |
| **Normal Flow:** | 1. User follows prompts to make enrollment changes for the life event in progress.  
2. User selects one of the Vision Plans in the SM Benefits Program.  
3. User clicks on the save button.  
4. Application checks to see if the user is already enrolled in the Medical plan.  
5. If user is enrolled in Medical Plan, the application processes the request and saves the changes. |
| **Alternate:** | 1. User follows prompts to make enrollment changes for the life
<table>
<thead>
<tr>
<th>Flow:</th>
</tr>
</thead>
<tbody>
<tr>
<td>event in progress.</td>
</tr>
<tr>
<td>2. User selects one of the Vision Plans in the SM Benefits Program.</td>
</tr>
<tr>
<td>3. User clicks on the save button.</td>
</tr>
<tr>
<td>4. Application checks to see if the user is already enrolled in the Medical plan.</td>
</tr>
<tr>
<td>5. If user isn’t enrolled in Medical Plan under the SM Benefits Program, application displays an error message, asking the user to add a Medical Plan in the SM Benefits program first. This is the cross plan enrollment dependency validation.</td>
</tr>
<tr>
<td>6. User clicks on the ‘Ok’ button to make the error message disappear.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Post-conditions:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. In case of normal flow, the system saves the changes and displays the form with the updated information.</td>
</tr>
<tr>
<td>2. In case of the alternate flow, the system displays the form without saving any of the changes made. At this point, the user has the option to enroll in one of the Medical Plan available in the Benefits Program and save the data and redo the steps in the Normal Flow of this use case.</td>
</tr>
</tbody>
</table>

Table 3: Use Case 2
## Use Case 3

<table>
<thead>
<tr>
<th>Use Case ID:</th>
<th>UC-003</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use Case Name:</strong></td>
<td>User tries to enroll in an Employee Plus Spouse Life Insurance Plan and the system applies Enrollment dependency validations.</td>
</tr>
<tr>
<td><strong>Actors:</strong></td>
<td>Employee or Benefits User or HR Personnel.</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>This use case describes the steps a user takes in order to enroll in an Employee Plus Spouse Life Insurance Plan and how the system should respond.</td>
</tr>
<tr>
<td><strong>Trigger:</strong></td>
<td>The user selects an Employee Plus Spouse Life Insurance Plan in the SM Benefits Program and hits the save button.</td>
</tr>
</tbody>
</table>
| **Preconditions:** | 1. The user has a working internet connection.  
2. The user is logged into Oracle Advanced Benefits application.  
3. User has a started life event to make the enrollment changes.  
4. The user has queried the participant’s (employee’s) profile and can see and edit the data on benefits enrollment pages. |
| **Normal Flow:** | 1. User follows prompts to make enrollment changes for the life event in progress.  
2. User selects one of the Employee Plus Spouse Life Insurance Plans in the SM Benefits Program.  
3. User clicks on the save button.  
4. Application checks to see if the user is already enrolled in the Employee Only Life Insurance plan.  
5. If user is enrolled in Employee Only Life Insurance plan, the application processes the request and saves the changes. |
| **Alternate Flow:** | 1. User follows prompts to make enrollment changes for the life event in progress.  
2. User selects one of the Employee Plus Spouse Life Insurance Plans in the SM Benefits Program.  
3. User clicks on the save button.  
4. Application checks to see if the user is already enrolled in the Employee Only Life Insurance plan.  
5. If user isn’t enrolled in Employee Only Life Insurance Plan under the SM Benefits Program, application displays an error message, asking the user to add an Employee Only Life Insurance Plan in the SM Benefits program first. This is the cross-plan enrollment dependency validation.  
6. User clicks on the ‘Ok’ button to make the error message disappear. |
| **Post-conditions:** | 1. In case of normal flow, the system saves the changes and displays the form with the updated information.  
2. In case of the alternate flow, the system displays the form without saving any of the changes made. At this point, the user has the option to enroll in one of the Employee Only Life Insurance Plans available in the Benefits Program and save the data and redo the steps in the Normal Flow of this use case. |

Table 4: Use Case 3
Use Case 4

<table>
<thead>
<tr>
<th>Use Case ID:</th>
<th>UC-004</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Name:</td>
<td>User tries to enroll in an Employee Plus Dependents Life Insurance Plan and the system applies Enrollment dependency validations.</td>
</tr>
<tr>
<td>Actors:</td>
<td>Employee or Benefits User or HR Personnel.</td>
</tr>
<tr>
<td>Description:</td>
<td>This use case describes the steps a user takes in order to enroll in an Employee Plus Dependents Life Insurance Plan and how the system should respond.</td>
</tr>
<tr>
<td>Trigger:</td>
<td>The user selects an Employee Plus Dependents Life Insurance Plan in the SM Benefits Program and hits the save button.</td>
</tr>
</tbody>
</table>
| Preconditions: | 1. The user has a working internet connection.  
2. The user is logged into Oracle Advanced Benefits application.  
3. User has a started life event to make the enrollment changes.  
4. The user has queried the participant’s (employee’s) profile and can see and edit the data on benefits enrollment pages. |
| Normal Flow: | 1. User follows prompts to make enrollment changes for the life event in progress.  
2. User selects one of the Employee Plus Dependents Life Insurance Plans in the SM Benefits Program.  
3. User clicks on the save button.  
4. Application checks to see if the user is already enrolled in the Employee Only Life Insurance plan.  
5. If user is enrolled in Employee Only Life Insurance plan, application processes the request and saves the changes. |
### Alternate Flow:

1. User follows prompts to make enrollment changes for the life event in progress.
2. User selects one of the Employee Plus Dependents Life Insurance Plans in the SM Benefits Program.
3. User clicks on the save button.
4. Application checks to see if the user is already enrolled in the Employee Only Life Insurance plan.
5. If user isn’t enrolled in Employee Only Life Insurance Plan under the SM Benefits Program, application displays an error message, asking the user to add an Employee Only Life Insurance Plan in the SM Benefits program first. This is the cross-plan enrollment dependency validation.
6. User clicks on the ‘Ok’ button to make the error message disappear.

### Post-conditions:

1. In case of normal flow, the system saves the changes and displays the form with the updated information.
2. In case of the alternate flow, the system displays the form without saving any of the changes made. At this point, the user has the option to enroll in one of the Employee Only Life Insurance Plans available in the Benefits Program and save the data and redo the steps in the Normal Flow of this use case.

Table 5: Use Case 4
**Use Case 5**

<table>
<thead>
<tr>
<th>Use Case ID:</th>
<th>UC-005</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use Case Name:</strong></td>
<td>HR Administrator wants to shift current employees from one benefits plan and option to another benefits plan and option.</td>
</tr>
<tr>
<td><strong>Actors:</strong></td>
<td>HR Administrator</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>This use case describes the steps a HR Administrator can take to switch current employees from one benefits plan and option to another benefits plan and option. Essentially, this is a batch process, and it will process a number of employees at once.</td>
</tr>
<tr>
<td><strong>Trigger:</strong></td>
<td>The user calls on the Mass Move Enrollments API.</td>
</tr>
<tr>
<td><strong>Preconditions:</strong></td>
<td>1. The HR Administrator has a working internet connection. 2. The HR Administrator is on the OAB application.</td>
</tr>
<tr>
<td><strong>Normal Flow:</strong></td>
<td>1. The HR Administrator populates the Mapping Table with the required data: i.e. the life event he would like to use for moving the enrollments, and the date he wants the move to be done. E.g. Move all enrollments on November 1st, 2010 with Life Event: ‘Mock Movement’ 2. The HR Administrator then populates the Mapping Detail Table with the required data: Old Plan ID, Old Option ID, New Plan ID, New Option ID. 3. The HR Administrator provides the Life Event ID and the Life Event Occurred Date when prompted. 4. The HR Administrator then calls the mass move enrollments procedure, passing parameters that run each employee in the</td>
</tr>
</tbody>
</table>
The employees are enrolled in the corresponding new plan and options.

Table 6: Use Case 5

Use Case 6

<table>
<thead>
<tr>
<th>Use Case ID:</th>
<th>UC-006</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Name:</td>
<td>HR Administrator wants to shift current employees from one benefits plan and option to another benefits plan and option.</td>
</tr>
<tr>
<td>Actors:</td>
<td>HR Administrator</td>
</tr>
<tr>
<td>Description:</td>
<td>This use case describes the steps a HR Administrator can take to switch current employees from one benefits plan and option to another benefits plan and option. Essentially, this is a batch process, and it will process a number of employees at once.</td>
</tr>
<tr>
<td>Trigger:</td>
<td>The user calls on the Mass Move Enrollments API.</td>
</tr>
<tr>
<td>Preconditions:</td>
<td>1. The HR Administrator has a working internet connection. 2. The HR Administrator is on the Oracle Advanced Benefits application.</td>
</tr>
<tr>
<td>Normal Flow:</td>
<td>1. The HR Administrator populates the Mapping Table with the required data; i.e. the life event he would like to use for moving the enrollments, and the date he wants the move to be done. E.g. Move all enrollments on November 1st, 2010 with Life Event: ‘Mock Movement’</td>
</tr>
</tbody>
</table>
2. The HR Administrator then populates the Mapping Detail Table with the required data: Old Plan ID, Old Option ID, New Plan ID, New Option ID.

3. The HR Administrator provides the Life Event ID and the Life Event Occurred Date when prompted.

4. The HR Administrator then calls the mass move enrollments procedure, passing the parameters so as to make just the employees in the newly acquired company go through the procedure.

| Postcondition: | The employees are enrolled in the corresponding new plan and options. |

Table 7: Use Case 6
Functional Design

Oracle Benefits Data Model

As mentioned earlier, Oracle HRMS is structured the following way:

![Diagram of Oracle HRMS structure]

Figure 6: High-level view of Oracle HRMS
The key data that define OAB Benefit offerings is:

- **Program** (overriding umbrella for clustered benefits)
- **Plan Type** (grouping of similar plans)
- **Plan** (the actual benefit made available to employees/dependents)
- **Options** (choices in the plan possible for employees/dependents)
- The pool of **rules** governing program participation [8]

The following high-level logical model depicts benefits objects as delivered by Oracle Benefits Application. The objects in red are oracle data objects with which the code written for Features one and two are linked.
Figure 7: High-level Logical Model of the Oracle Advanced Benefits Module

The rectangular items in the Entity-Relationship Diagram above represent entities, and the diamond shaped items represent the relationships between the entities. The branches at the ends of the links between the items depict many-to-many relationships. For example, a program can have many plans; similarly, one plan can be a part of many different programs.

This section covers the functional design required for two features built as part of this
project. Before coming up with the design for the problem, it is necessary to complete the setups on HR and benefits modules. A benefits program in OAB was set up. It was named “SM Employee Benefits Program”.

This plan design is used to validate the unit test cases before release. The Program has three main Plans: Kaiser, UHC (United Health Care) and Cigna. Each plan has the plan types: Medical, Dental, Vision and Supplemental Life Insurance. Each plan type has the options: Employee only, Employee Plus Spouse, Employee Plus Dependent, and Waive plans.

The consultants interviewed for this project are acquaintances. They worked as implementation and techno-functional consultants in the Oracle applications domain. Since they have exposure to Oracle Applications and several implementations of HRMS modules, their help was taken in trying to learn more about real life problems faced by companies that use Oracle HRMS.

The following figures show the steps involved in making Plan Options and Plan Types and putting them together. For example, after the Employee only option was created, the plan types – Medical, Dental and Vision were added, so that the application applied this Employee only option to each of those plan types.
Figure 8: Setting up Plan Options

The next image shows how plans are set up. Once the options are set up, you can apply the options to a new plan that you’re trying to create. For example, in the image below, the Plan Options, Employee only, Employee Plus Spouse, Employee Plus Dependents, and Waive, were being applied to the new plan that was being created, i.e. Kaiser Medical Plan.
Figure 9: Matching Plan Options to Plans.

The overall structure of this benefits program after all the Plans, Plan Types, Options were created is displayed in the following figure:
Figure 10: Plan Design of SM Employee Benefits Program
Feature Set 1: Enforcement of Enrollment Dependencies

Description

It is required to build an extension feature to enforce the Enrollment Dependencies as a generic solution. Currently, this can be achieved using the fast formulas and PL/SQL functions. Authoring and implementation of fast formulas is not intuitive to functional users, additionally it requires developer effort during development and ongoing maintenance. So, the solution proposed needs to be generic, extensible and easy to use for functional users.

Essentially this feature should enforce the plan enrollment dependencies. For example, a participant is not allowed to enroll in Spouse Life Insurance Plan if not already enrolled in the Employee Life Insurance Plan. Similar dependencies exist for other plans and options as well.

Logical Data Model

Two entities are required to store the enrolling plan (desired plan) and required plan (the plan to be enrolled in in order to enroll in the desired plan). Additionally, these entities should be date effective. Following is the description of the entities and entity relationship diagram.
Figure 11: Entity Relationship Diagram for the Entities in Feature 1 – Part 1
Algorithm for Enrollment Dependency Validation

This logic should be defined as a PL/SQL package and should be integrated into enrollment business process.
For this feature set, the overall design involves the following steps:

1. Create plan level and plan type + option level enrollment dependency tables.
   a. Create sample data.

2. Create a package which validates the plan and option + plan type validations.
   a. For each row in the plan level validations table:
      i. Check enrolled in “enrolled_plan”;
         If enrolled, then:
         Check enrolled in “required_plan”
         If not enrolled, then:
         Form an error message and raise it.
         End if
         End if
   End loop;
   b. For each row in plan type + option validations table
      i. Check if the user is enrolled in “enrolled_option” + “enrolled_pl_typ”
         If enrolled, then:
         Check enrolled in “rqd_pl_typ” + “rqd_option”
         If not enrolled, then,
         Form error message and raise it.
         End if;
         End if;
   End loop;

3. As the fast formulas are no longer required, if any existing formulas are defined, they
should be removed.

**Feature Set 2: Automation of Switching Plan Enrollments**

**Description**

It is required to build an extension to the Oracle Applications Benefits Module that performs a batch process to move the participant enrollments from old plans to new ones. This should be a generic solution and very user friendly, and something that can be done with very little help from developers. Either due to changes in vendors, to reduce the costs or on acquiring a new company, it is necessary to move the enrollments for employees from old plans to new plans en-masse. For example, all the current employees who are enrolled in UHC Health Plan should be moved to the Kaiser Health plan starting from 1\textsuperscript{st} of January, 2012.

**Logical Data model**

Two entities are required to store the old plan and corresponding new plan. These entities should support what life event needs to be used for this mass movement and on what date these changes should be effective. Additionally, the plans and options that are not impacted by this movement should be retained for each employee. This should be a batch process and should require no integration into existing oracle processes. Following is the entity relationship diagram:
Figure 13: Entity Relationship Diagram for the Entities in Feature 2
Algorithm for Enrollment Dependency Validation

For this feature set, the overall design involves the following steps:

1. Create your own life event. (“SM Move Enrollment”)
2. Configure
   a. Create potential life event
   b. Run ‘benmngle’
   c. Make enrollment into UHC
   d. Apply defaults
   e. Close life event
3. Setup Steps:
   1. Create Life Event “SM Move Enrollment”
   2. Attach Life Event to program on “Program Enrollment form”
   3. Create Kaiser plan with options.
   4. Attach Kaiser plan to program with corresponding year periods.
   5. Validate setup for one person.
4. For each person: Loop:
   1. Check if there are any ongoing life events: (ben_per_in_ler = strtd)
   2. Check if there are any future processed life events.
      If yes:
      Skip person.
   3. Give potential life event of a given ler_id to the person given lf_evt_ocrd_dt
   4. Process the person for the given potential (benmngle)
   5. For each row in detail table
Loop:

Make enrollment into new plan if person is enrolled in old plan and option combination

End Loop

6. Make enrollments into old plans not in detail table.

7. Close the life event.

8. End person loop.

5. Create corresponding batch process components.
Technical Design and Coding

Feature Set 1

1. Naming Conventions:
   a. All packages and table names begin with sm_
   b. Parameter names begin with p_
   c. Wherever possible, Oracle Advanced Benefits module abbreviations are used to be consistent with delivered code.

2. Physical Data Model: Following are the tables created to implement the feature. Please see appendix for table definitions and their associated sample insert statements.
   a. SM_pl_enrt_dep
   b. SM_pl_typ_opt_dep

3. Package sm_enrt_validations_api created for the algorithm specified in the functional design.

Feature Set 2

1. Naming Conventions:
   a. All packages begin with sm_ , table names begin with:
   b. Parameter names begin with p_
   c. Wherever possible Oracle Advanced Benefits abbreviations are used to be consistent with delivered code.

2. Physical Data Model: Following are the tables created to implement the feature. Please see Appendix C for table definitions and their associated sample insert statements.
a. SM_pl_mapping_master

b. SM_pl_mapping_details

3. Package sm_pl_move_enrollments_api created for the algorithm specified in the functional design.
Testing and Release

Enrollment Dependency Validation: Feature 1

Test Plan:

The following plan design was created to test this functionality of Feature 1. See the following table for summary of plan design. The actual Screenshots are attached in Appendix D.

<table>
<thead>
<tr>
<th>PLAN TYPES</th>
<th>PLANS</th>
<th>OPTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM Medical</td>
<td>Cigna</td>
<td>Employee Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Family</td>
</tr>
<tr>
<td></td>
<td>Kaiser</td>
<td>Employee Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Family</td>
</tr>
<tr>
<td></td>
<td>UHC</td>
<td>Employee Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Family</td>
</tr>
<tr>
<td>SM Dental</td>
<td>Cigna</td>
<td>Employee Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Family</td>
</tr>
<tr>
<td></td>
<td>Kaiser</td>
<td>Employee Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Family</td>
</tr>
<tr>
<td></td>
<td>UHC</td>
<td>Employee Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Family</td>
</tr>
<tr>
<td>SM Vision</td>
<td>Cigna</td>
<td>Employee Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Family</td>
</tr>
<tr>
<td></td>
<td>Kaiser</td>
<td>Employee Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Family</td>
</tr>
<tr>
<td></td>
<td>UHC</td>
<td>Employee Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Family</td>
</tr>
<tr>
<td>SM Life Insurance</td>
<td>Cigna</td>
<td>Employee</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dependent</td>
</tr>
</tbody>
</table>

Figure 14: Summary of Plan Design
Unit Testing Enrollment Dependency Validation in SQL Plus

This process involves the following steps:

a. Run the insert statements to define the dependencies between the plan and option combinations.

**Insert Statements:**

```sql
INSERT INTO SM_PL_ENRT_DEP VALUES (SM_PL_ENRT_DEP_SEQ.nextval, 202, 7042, 7041, to_date('01-Jan-2000', 'DD-MON-YYYY'), null);
```

-- This insert statement specifies that in order for an employee to add 7042, which is the Spouse Supplemental Insurance, the employee needs to be enrolled in 7041, which is the Employee Supplemental Life Insurance plan.

```sql
INSERT INTO SM_PL_ENRT_DEP VALUES (SM_PL_ENRT_DEP_SEQ.nextval, 202, 7043, 7041, to_date('01-Jan-2000', 'DD-MON-YYYY'), null);
```

-- This insert statement specifies that in order for an employee to add 7043, which is the Dependent Supplemental Insurance, the employee needs to be enrolled in 7041, which is the Employee Supplemental Life Insurance plan.

-- 202 - Business Group ID
-- 7041 - Plan ID of Employee Supplemental Insurance
-- 7042 - Plan ID of Spouse Supplemental Insurance
-- 7043 - Plan ID of Dependent Supplemental Insurance

b. Unit test “Enrollment Dependency Validation” code from sql*plus.

**Unit Testing Script:**

```sql
set serveroutput on
DECLARE
BEGIN
-- sm_enrt_validations_api.pl_lev_validations
(p_business_group_id => 202,
 p_person_id =>4426,
```
p_effective_date =>to_date('01-Jan-2000', 'DD-MON-YYYY'));

--
--
sm_enrt_validations_api.pl_lev_validations
(p_business_group_id =>202,
 p_person_id =>4426,
 p_effective_date =>to_date('01-Jan-2000', 'DD-MON-YYYY'));

--
END;
/

c. Release: Add the validation code into ben_pen_api packages so that validations will be applied irrespective of initiation of enrollments.

**Test Steps for checking Dependency Validation using the UI**

In this step, one can test the dependency validation from the User Interface. Error messages saying that you cannot enroll in a Dental plan without enrolling in a Medical Plan first, or that you cannot enroll in a Vision plan without enrolling in a Medical Plan first, or that you cannot enroll in a Spouse Life Insurance plan without enrolling in an Employee Life Insurance plan first, or that you cannot enroll in a Dependent Life Insurance Plan without enrolling in an Employee Life Insurance plan first should be displayed.

<table>
<thead>
<tr>
<th>Test Step</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. Verify that if you try to enroll a user in a Dental plan without enrolling in a medical plan first, the system displays a popup message explaining that the user needs to enroll in a Medical Plan first.</strong></td>
<td></td>
</tr>
<tr>
<td>1.1 User selects an employee who hasn't enrolled in benefits.</td>
<td>Employee is selected.</td>
</tr>
</tbody>
</table>
1.2 User follows prompts to make enrollment changes for the life event in progress.

1.3 User selects one of the Dental Plans in the SM Benefits Program and adds it to the user's Benefits.

| 1.4 | User clicks on the save button. | Application checks to see if the user is already enrolled in the Medical plan. |
| 1.5 | User selects one of the Medical Plans in the SM Benefits Program and adds it to the same user's Benefits. | Application displays an error message saying - "Please enroll in a Medical plan before trying to enroll in a Dental plan" |
| 1.6 | User clicks on the save button. | Application saves the enrollment successfully. |
| 1.7 | User selects one of the Dental Plans in the SM Benefits Program and adds it to the user's Benefits. | Application checks to see if the user is already enrolled in the Medical plan. |
| 1.8 | User clicks on the save button. | Application saves the enrollment successfully. |

2. Verify that if you try to enroll a user in a Vision plan without enrolling in a medical plan first, the system displays a popup message explaining that the user needs to enroll in a Medical Plan first.

<p>| 2.1 | User selects an employee who hasn't enrolled in benefits. | Employee is selected. |
| 2.2 | User follows prompts to make enrollment changes for the life event in progress. | |</p>
<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3</td>
<td>User selects one of the Vision Plans in the SM Benefits Program and adds it to the user's Benefits.</td>
<td>Application checks to see if the user is already enrolled in the Medical plan.</td>
</tr>
<tr>
<td>2.4</td>
<td>User clicks on the save button.</td>
<td>Application displays an error message saying - &quot;Please enroll in a Medical plan before trying to enroll in a Vision plan&quot;</td>
</tr>
<tr>
<td>2.5</td>
<td>User selects one of the Medical Plans in the SM Benefits Program and adds it to the same user's Benefits.</td>
<td></td>
</tr>
<tr>
<td>2.6</td>
<td>User clicks on the save button.</td>
<td>Application saves the enrollment successfully.</td>
</tr>
<tr>
<td>2.7</td>
<td>User selects one of the Vision Plans in the SM Benefits Program and adds it to the user's Benefits.</td>
<td>Application checks to see if the user is already enrolled in the Medical plan.</td>
</tr>
<tr>
<td>2.8</td>
<td>User clicks on the save button.</td>
<td>Application saves the enrollment successfully.</td>
</tr>
</tbody>
</table>

3. Verify that if you try to enroll a spouse in a medical plan without enrolling the employee in a medical plan first, the system displays a popup message explaining that the employee needs to be enrolled in a medical plan first.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>User selects an employee who hasn't enrolled in benefits.</td>
<td>Employee is selected.</td>
</tr>
<tr>
<td>3.2</td>
<td>User follows prompts to make enrollment changes for the life event in progress.</td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>User selects one of the Medical Plans in the SM Benefits Program and tries to enroll the Spouse in the Medical plan.</td>
<td>Application checks to see if the employee himself is already enrolled in a Medical plan.</td>
</tr>
<tr>
<td>3.4</td>
<td>User clicks on the save button.</td>
<td>Application displays an error message saying - &quot;Please enroll this employee in a Medical plan before enrolling the Spouse in the Medical plan.&quot;</td>
</tr>
<tr>
<td>3.5</td>
<td>User selects one of the Medical Plans in the SM Benefits Program and enrolls the employee in the Medical Plan.</td>
<td></td>
</tr>
<tr>
<td>3.6</td>
<td>User clicks on the save button.</td>
<td>Application saves the enrollment successfully.</td>
</tr>
<tr>
<td>3.7</td>
<td>User selects one of the Medical Plans in the SM Benefits Program and tries to enroll the Spouse in the Medical plan.</td>
<td>Application checks to see if the employee is already enrolled in a Medical plan.</td>
</tr>
<tr>
<td>3.8</td>
<td>User clicks on the save button.</td>
<td>Application saves the enrollment successfully.</td>
</tr>
</tbody>
</table>

### 4. Verify that if you try to enroll a spouse or a dependent in a Life Insurance plan without enrolling the employee in a Life Insurance plan first, the system displays a popup message explaining that the employee needs to be enrolled in the Life Insurance plan first.

<p>| 4.1  | User selects an employee who hasn't enrolled in a life insurance plan. | Employee is selected. |
| 4.2  | User follows prompts to make enrollment changes for the life event in progress. |  |
| 4.3  | User selects one of the Life Insurance plans in the SM Benefits Program with the Spouse option and adds the plan. |  |</p>
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Expected Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4</td>
<td>User clicks on the save button.</td>
<td>Application checks to see if the employee himself is already enrolled in a Life Insurance plan. Since the employee isn't enrolled in a Life Insurance plan himself, the application displays an error message saying - &quot;Please enroll this employee in a Life Insurance plan before enrolling the Spouse in the Medical plan.&quot;</td>
</tr>
<tr>
<td>4.5</td>
<td>User selects one of the Life Insurance Plans in the SM Benefits Program with the Employee only option and enrolls himself in it.</td>
<td></td>
</tr>
<tr>
<td>4.6</td>
<td>User clicks on the save button.</td>
<td>Application saves the enrollment successfully.</td>
</tr>
<tr>
<td>4.7</td>
<td>User selects one of the Life Insurance plans in the SM Benefits Program with the Spouse option and adds the plan.</td>
<td></td>
</tr>
</tbody>
</table>
Automation of Switching Plan Enrollments: Feature 2

Test Plan:

The following plan design and a new life event were created to test this functionality of Feature 2. See the following table for summary of plan design. The actual Screenshots are attached in Appendix D.

<table>
<thead>
<tr>
<th>PLAN TYPES</th>
<th>PLANS</th>
<th>OPTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM Medical</td>
<td>Cigna</td>
<td>Employee Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Family</td>
</tr>
<tr>
<td></td>
<td>Kaiser</td>
<td>Employee Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Family</td>
</tr>
<tr>
<td></td>
<td>UHC</td>
<td>Employee Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Family</td>
</tr>
<tr>
<td>SM Dental</td>
<td>Cigna</td>
<td>Employee Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Family</td>
</tr>
<tr>
<td></td>
<td>Kaiser</td>
<td>Employee Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Family</td>
</tr>
<tr>
<td></td>
<td>UHC</td>
<td>Employee Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Family</td>
</tr>
<tr>
<td>SM Vision</td>
<td>Cigna</td>
<td>Employee Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Family</td>
</tr>
<tr>
<td></td>
<td>Kaiser</td>
<td>Employee Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Family</td>
</tr>
<tr>
<td></td>
<td>UHC</td>
<td>Employee Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Plus Family</td>
</tr>
<tr>
<td>SM Life Insurance</td>
<td>Employee</td>
<td>Spouse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dependent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Waive</td>
</tr>
</tbody>
</table>
Table 9: Plan Design

The Life Event that was created for the purpose of testing this feature was called “Move Enrollments”

**Unit Testing Enrollment Dependency Validation in SQL Plus**

This process involves the following steps:

**a.** Run the insert statements to define the old plan and new plan and associated options to switch enrollments from old plan plus option to new plan plus option to move enrollments.

**Insert Statements:**

```
INSERT INTO SM_PL_MAPPING_MASTER VALUES
(SM_PL_MAPPING_MASTER_SEQ.nextval, 13594, to_date('01-Jan-2012', 'DD-MON-YYYY'), sysdate-10, null);

INSERT INTO SM_PL_MAPPING_DETAILS VALUES
(SM_PL_MAPPING_DETAILS_SEQ.nextval, 1, 7033, 4194, 7032, 4194, to_date('01-Jan-2012', 'DD-MON-YYYY'), null);

/*
Old plan Id: Cigna: 7033
Old option ID: Employee Only: 4194
New Plan ID: Kaiser: 7032
New Option ID: Employee Only: 4194
*/
```

**b.** Unit test “Automate switching Enrollments” code from sql*plus.

**Unit Testing Script:**

```
set serveroutput on
DECLARE
BEGIN
sm_pl_move_enrollments_api.pl_move_enrollments
```
(202, 4426, 13593, TO_DATE('01-jan-2011', 'dd-mon-yyyy')+160);

END;
/

c. **Release**: The package is created in the database so it can be called from any program or interface.

d. Sample movement from old plan to new plan is executed and results can be seen in given screen shot, which have been included in the Appendix D, Part 2.

**Test Steps for Moving Enrollments**

<table>
<thead>
<tr>
<th>Test Step</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>5. Verify that you can run a batch process successfully to transfer all employees in the existing company from the old benefits plan plus option to the corresponding new benefits plan plus option.</td>
<td></td>
</tr>
<tr>
<td>5.1 The HR Administrator populates the Mapping Table with the required data: i.e. the life event he would like to use for moving the enrollments, and the date he wants the move to be done. E.g. Move all enrollments on November 1st, 2010 with Life Event: 'Mock Movement'</td>
<td>Life event gets created.</td>
</tr>
<tr>
<td>5.2 The HR Administrator then populates the Mapping Detail Table with the required data: Old Plan ID, Old Option ID, New Plan ID, New Option ID.</td>
<td>Mapping table gets populated, and the application requests the user to provide the Life Event ID and Life Event Occurred Date.</td>
</tr>
<tr>
<td>5.3 The HR Administrator provides the Life Event ID and the Life Event Occurred Date.</td>
<td>The application stores the data input by the HR Administrator.</td>
</tr>
<tr>
<td>5.4</td>
<td>The HR Administrator then calls the mass move enrollments procedure</td>
</tr>
<tr>
<td>5.5</td>
<td>The HR Administrator selects one of the employees on whom this procedure was applied.</td>
</tr>
<tr>
<td>5.6</td>
<td>The HR Administrator checks to see what plan and plan type the employee is enrolled in.</td>
</tr>
</tbody>
</table>

**6. Verify that you can run a batch process successfully to transfer all employees and their corresponding benefits plans and options from the newly acquired company.**

| 6.1 | The HR Administrator populates the Mapping Table with the required data: i.e. the life event he would like to use for moving the enrollments, and the date he wants the move to be done. E.g. Move all enrollments on November 1st, 2010 with Life Event: ‘Mock Movement’ | Life event gets created. |
| 6.2 | The HR Administrator then populates the Mapping Detail Table with the required data: Old Plan ID, Old Option ID, New Plan ID, New Option ID. | Mapping table gets populated, and the application requests the user to provide the Life Event ID and Life Event Occurred Date. |
| 6.3 | The HR Administrator provides the Life Event ID and the Life Event Occurred Date. | The application stores the data input by the HR Administrator. |
| 6.4 | The HR Administrator then calls the mass move enrollments procedure | The application passes parameters corresponding to each employee in the newly acquired company through the |
The HR Administrator selects one of the newly acquired employees on whom this procedure was applied. The employee information is pulled up.

The HR Administrator checks to see what plan and plan type the newly acquired employee is enrolled in. The employee is enrolled in the new benefits plan as expected.

| 6.5 | The HR Administrator selects one of the newly acquired employees on whom this procedure was applied. | The employee information is pulled up. |
| 6.6 | The HR Administrator checks to see what plan and plan type the newly acquired employee is enrolled in. | The employee is enrolled in the new benefits plan as expected. |

Table 10: Test Steps for Feature 2
**Conclusion**

This project intended to provide enhanced productivity features to the Benefits Module of Oracle Human Resources Management Systems (HRMS). It was aimed to help move benefits enrollments from an old benefits plan to a new one. It was also meant to provide enhanced dependency validation between the benefits plan enrollments. After a successful implementation of the waterfall model of project management, requirements, design and code were come up with and the features were tested. As a part of Unit Testing using PL-SQL based test scripts, when code for both features didn’t work, the code was debugged using the hr_utility package and was rectified accordingly. Once the code worked, it was integrated into the enrollment package and deployed standalone for features one and two respectively. The tests were successful, and everything worked as expected even after integration. Success was measured by the feedback given by two of the stakeholders who were previously interviewed, Padmaja Bodla, Srinath Parepally and Sreenath Gajulapalli, who have all had numerous years of experience in the fields of Oracle and SAP. They said first feature proved to be a very generic and user friendly solution. They felt that feature 1 eliminates the development time of one week per each validation. At least twenty validations are required for a typical implementation. So, overall, it would eliminate twenty weeks of developer effort.

When the designs were validated, consultants mentioned that such high volume enrollment changes can be performed in a few hours (by running the batch process) compared to the weeks it would take without the feature. Feature 2 essentially eliminates custom program development. No developer effort is required after the initial implementation. The batch process can be created and run on the coded package for all...
the employees for whom the benefits plan needs to be switched from one to another. Time and effort put into the process by HR professionals when they change the benefits options for one employee at a time – something that can take a few weeks to a few months – on automation, would take a few hours.

The following list provides a summary of all the tasks performed in this project:

- Coming up with the requirements for the two features that were implemented.
- Coming up with the design and implementation.
- Creating supporting eligibility profiles.
- Coming up with a generic validation package to support the enrollment dependencies, which were then integrated into the plan design setups.
- Unit testing before the code was integrated.
- Testing using the UI after the code was integrated.

Like any project, this project took a little longer than the original schedule mentioned in the proposal. Two packages were coded, one package for each feature. The code was modularized and compliant with Oracle Application Coding Standards. The amount of code expected was less than the amount of code actually developed. This was one of the reasons why the project took a little longer than the proposed schedule. Another reason the project took longer to finish than expected had to do with getting familiarized with the languages, SQL and PL SQL, installing Oracle Applications, and learning how to use the Benefits module of the Oracle Applications as well as the software, Putty. However, the features were implemented successfully. In fact, the consultants who viewed the
demo expressed that these can be features of the product and that it would enhance the functionality of the product as a whole.
Textual References


  <http://idealpenngroup.tripod.com/sitebuildercontent/OAUG2008/Collaborate2
Image References


Appendix A: Stakeholder Interviews

Interview 1

A. Stakeholder Profile

1. What is your name?
   Padmaja Bodla

2. Where do you currently work?
   Activision, Inc.

3. What is your position there?
   Oracle Applications Developer

4. How long have you worked there?
   6 months

5. Can you tell me what kinds of tasks your job entails?
   - Development of customizations to HRMS modules of Oracle Applications.
   - Support production instance for issues raised by users and functional analysts.
   - Lead development team

6. What other positions have you held in your career?
   I have worked as developer, a consultant, and as a support engineer for more than 10 years. My positions at various companies have been as follows:
   - Senior Applications Developer, Oracle Corporation.
   - Senior Software Engineer, KBACE Technologies, Inc.
Techno-functional and Implementation Consultant, Deloitte.

B. Assessing the Problem

7. Do you think that the OAB Module of Oracle Applications could do with some better functionality or an enhanced set of features?

OAB module of Oracle applications (EBS) is a global benefits management product. Its flexible architecture is leveraged in benefits, compensation workbench and HR modules. Even though it accommodates requirements of business outsourcing organizations and big corporations, there is still some scope for improvement in areas such as easy and rapid setups, guided plan design setups, employee and employer communications, extracts and document management.

8. Can you explain what kinds of functionality you feel are missing in Oracle Applications in general?

I think communications, document management and vendor interfaces are generally lacking. In some cases, simple features have to be implemented using customizations.

9. Can you explain what kinds of functionality the Benefits module of the HRMS part of Oracle Applications seems to be missing?

- Rapid implementation templates to create plan designs
- Communications module to send benefits documents, store employee certifications and so on.
- Mass data uploads to move enrollments from one plan to other plans.
- Sometimes customizations are required for common features like cross
plan validations.

10. Have you personally spoken to anyone who seemed to have experienced this particular problem?

When I was a consultant and lead implementation engineer, I happened to discuss these features with functional analysts and business users.

11. Can you tell me how they dealt with such a situation?

We implemented cross-plan validations using fast formulae. This particular way of implementing the simple validations was not intuitive. It will be a good idea to use base tables to track these validations so that business users without help of developers can manage year over year. Additionally, the implementation time will be reduced from weeks to few hours of setup. Good documentation practices help a lot too.

12. What are some of the other types of problems you have faced while working with the OAB Module?

Mass movement of employee population from one plan to other, in case of reorganization or streamlining the plans tended to be a bit of a problem.

13. Can you tell me about some of the problems that companies you know to use OAB faced?

Training users as the plan design is complex. If fast formulas are used it is not easy for functional users to understand the logic involved.

C. Assessing Our Solution

14. If there were a way to automate such a process to make things easier, would you like to use it? Do you think it would help you?
Certainly. A better User Interface that promotes ease of use or a base table to define plan validations and any automated batch process to move enrollments will help.

15. Can you think of any problems such a solution may cause if it were integrated into the OAB module?

As long as these features are generic and useful for majority of customers there is no issue. The features that can be implemented cannot be client specific.

D. Assessing the Opportunity

16. Do you believe the functionality that the OAB Module offers is sufficient?

OAB offers functionality sufficient to implement benefits for any size organization. There is certainly some scope to improve or provide more features as stated earlier.

E. Outside the Box

17. In a dream world, what would you expect from OAB Module?

To accommodate all the requirements of clients, with almost no customizations.

18. What are the pitfalls with the way OAB Module is right now?

Plan design is complex and not easy to understand for simple functional users.
Interview 2

A. Stakeholder Profile

1. What is your name?
   Prasad Bodla

2. Where do you work?
   Oracle Corporation.

3. What is your current position at Oracle?
   Senior Director, Product Development

4. How long have you worked there?
   14 years

5. What other positions have you held for the time you've worked at Oracle?
   Senior Developer, Principal Applications Developer, Development Manager, Senior Development Manager, Group Manager, Director

6. Can you tell me about what kinds of tasks your current position entails?
   - Managing Fusion Advanced Benefits, Fusion Global absences Development teams.
   - Managing project planning, resourcing, and signing off requirement, functional design, and technical design documents. Coordinating the work between the product management, Quality assurance and Strategy teams.
   - Guiding the Senior Managers and Architects in product designs and development plan executions.
- Coordinating the work for customer interactions and release operations related activities.
- Guiding product strategy, planning, competitive analysis and future direction.

**B. Assessing the Problem**

7. Do you think that the OAB Module of Oracle Applications is sufficient to meet the requirements of mid-sized to large corporations? Do you think an enhanced set of features and functionality required to meet requirements easily?

Yes, the OAB module is a feature rich rules-based product used to implement benefits for any size corporation. Even though the fresh-out-of-the-box product supports a multitude of requirements, there is scope to improve features in vendor extracts, communications and to be made less reliant on customizations using fast formulas.

8. Can you explain what kinds of functionality will help users reduce customizations, deployment time?

Customers generally customize the benefits product using the fast formulas. Fast formulas can be attached at more than 100 locations. Some of these customizations like cross plan enrollment validations are used in almost all implementations. To implement these validations developers are required to write custom code and hard for functional analysts to understand. So it will help to enhance the product by creating new set of tables and user friendly way to define these validations.
9. Can you explain what kinds of functionality the Benefits module of the HRMS part of Oracle Applications seems to be missing?

Few features which will reduce the implementation time are – out of the box vendor extracts, employee communication, reduced reliance on fast formulas and mass enrollment of employees to due to acquisitions or introducing new plans.

10. Have you personally spoken to anyone who seemed to have experienced this particular problem?

I worked with various customers during their implementation cycles and observed that some of these customizations will be good candidates to build into the product.

11. Can you tell me how they dealt with such a situation?

Almost always, customers either write fast formulas to plug in their business requirements or they build the custom extract and communication modules.

12. What are some of the other types of problems customers faced while working with the OAB Module?

Some of the problems are billing, third party interfaces, and complexity of plan design and rate changes implementation.

13. Can you tell me about some of the problems that companies you know to use OAB faced?

Data migration, user training, employee notification system, correcting the data for few employees are difficult due to various reasons. One of the problems is to make mass changes to employee participation and enrollment.
like deactivating a benefit plan and introducing new one in its place.

C. Assessing Our Solution

14. If there was a way to automate such a process to make things easier, would you like to use it? Do you think it would help you?

Yes, it will help customers to use the automations, like mass disenrollment and enrollment and easy cross plan enrollment validations.

15. Can you think of any problems such a solution may cause if it were integrated into the OAB module?

As the product is used by several corporations with various sizes, any feature or enhancement must be useful to customer base and generic.

D. Assessing the Opportunity

16. Do you believe the functionality that the OAB Module offers is sufficient?

Functionality offered by OAB module is more than sufficient for vast customer base. As each corporation is unique, their benefits offerings will be unique. It will be impossible to provide everything out of the box. However, the product is rules based, and it provides infrastructure to support any benefits plans.

E. Outside the Box

17. In a dream world, what would you expect from OAB Module?

Every customer requirement should be met out of the box, without customizations.

18. What are the pitfalls with the way OAB Module is right now?

None.
Interview 3

A. Stakeholder Profile

1. What is your name?
   Srinath Parepally

2. Where do you currently work?
   Aptude Inc.

3. What is your position there?
   Vice-President

4. How long have you worked there?
   6 years

5. Can you tell me what kinds of tasks your job entails?
   - Overall in charge of the operations
   - Project Management, Resource Fulfillment and Customer Interaction

6. What other positions have you held in your career?
   My positions at various companies have been as follows:
   - Technical Architect - Washington Mutual
   - Principal Consultant – Vitria Technologies
   - Programmer Analyst – J.J & Associates
   - Techno-functional Developer – Oracle HRMS Implementation

B. Assessing the Problem

7. Do you think that the OAB Module of Oracle Applications could do with some better functionality or an enhanced set of features?
OAB module is rules based and pretty much accommodates any complex benefits plan design. But there are few features really useful if available out of the box for customers. Especially some of the following features which are useful for customers and I am aware custom programs are written by clients. Main Communications features lacking are – employee communications and vendor communications. Employee communications include – notification indicating life event is started, reminder notifications during the open enrollment, communication about plan enrollment defaulting, communication about any certifications and associated reminders and any common benefits related emails and notifications. Also it is important that employees must be able to access these communications on demand any time on benefits portal. Also important is mass communication to selected employees. Second category of communications include the communication with the vendors.

8. Can you explain what kinds of functionality you feel are missing in Oracle Applications in general?

I feel Web center technologies like chat, social connection and group spaces are missing from applications.

9. Can you explain what kinds of functionality the Benefits module of the HRMS part of Oracle Applications seems to be missing?

As mentioned earlier automated end to end solution of communication is missing and other major area is billing module if employees have to pay when they are not earning sufficient pay.

10. Have you personally spoken to anyone who seemed to have experienced
this particular problem?

Yes I was a consultant and implemented above mentioned features as customizations for some of my clients.

11. Can you tell me how they dealt with such a situation?

We have consulted our clients and designed custom programs using the oracle infrastructure to send the notifications, emails, regular ground mail of generated documents using mail merge features and workflow notifications.

12. What are some of the other types of problems you have faced while working with the OAB Module?

I felt Fast formulas are easy for developers and it is easy to maintain, but very difficult for business analysts to understand. I felt this language if modified to be useful for business analysts will greatly enhance the capabilities if business analysts.

13. Can you tell me about some of the problems that companies you know to use OAB faced?

Mostly mentioned above. Also the plan design pages are complex.

C. Assessing Our Solution

14. If there was a way to automate such a process to make things easier, would you like to use it? Do you think it would help you?

Yes with development software to automate communications like BI publisher these features can be built into base product.

15. Can you think of any problems such a solution may cause if it were integrated into the OAB module?
No, as long as features are built generic then fine.

D. Assessing the Opportunity

16. Do you believe the functionality that the OAB Module offers is sufficient?
   Yes features are sufficient, just need to extend where the gaps are like communications.

E. Outside the Box

17. In a dream world, what would you expect from OAB Module?
   I expect no customizations and fast formulas can be authored by business analysts.

18. What are the pitfalls with the way OAB Module is right now?
   I think multitude of technologies are used like forms, OA framework etc., I guess using one tech stack will be ideal.
Interview 4

A. Stakeholder Profile

1. What is your name?
   Sreenath Gajulapalli

2. Where do you currently work?
   GP Consulting, LLC.

3. What is your position there?
   Vice-President for Business Development, Customer Relations, Account Management

4. How long have you worked there?
   8+ Years

5. Can you tell me what kinds of tasks your job entails?
   As a vice-president to a small emerging company, my job entails me to wear different hats like:
   - Winning new clients
   - Managing existing client accounts
   - Managing Customers & Vendors relations
   - Company Marketing
   - Company Accounting
   - Employee Management etc.

6. What other positions have you held in your career?
   My positions at various companies have been as follows:
   - Worked as Senior ERP (SAP) Consultant at PSS Fed, Inc. for their
clients Accenture & Defense Logistics Agency

- Worked as Senior SAP Consultant at AtosOrigin for their various clients like Conair, Alaska Telephone & Utilities, Avaya, Philips Lighting etc.
- Worked as a Senior IT Consultant at Keane, Inc. for their client, Bose Corporation
- Worked as an IT Consultant at Overseas Technologies for their client, Fidelity Investments
- Worked as an IT Consultant for Sumac Enterprises for their client, State Street Bank
- Worked as a functional consultant for Oracle HRMS Implementation.

F. Assessing the Problem

7. Do you think that the OAB Module of Oracle Applications could do with some better functionality or an enhanced set of features?

Yes there are certain features will be useful like employee and vendor billing, financials integration, document management.

8. Can you explain what kinds of functionality you feel are missing in Oracle Applications in general?

Main features useful are:

a) Employee Billing: If earnings are not sufficient for employees or non-employee participants, benefits costs should be billed. This module of billing and collection reconciliation is not there in the product.
b) Vendor billing and payments if available will eliminate the customizations

c) For above two features if integrated with account receivables and payables it will be ideal.

d) Document management: features to store documents sent to employees and documents received from employees to be stored in central repository and available for administrators and in some cases available for employees on portals.

9. Can you explain what kinds of functionality the Benefits module of the HRMS part of Oracle Applications seems to be missing?

Above features are missing even though oracle tech stack provides all the software needed. Basically, seamless integration is missing.

10. Have you personally spoken to anyone who seemed to have experienced this particular problem?

As a consultant I implemented document management and billing as extensions.

11. Can you tell me how they dealt with such a situation?

By using Oracle file system and custom programs.

12. What are some of the other types of problems you have faced while working with the OAB Module?

Mainly the plan design pages are not simple. If the number of benefits objects are more, it is difficult to navigate to a specific row especially on plan and program enrollment requirements pages.
13. Can you tell me about some of the problems that companies you know to use OAB faced?

As mentioned earlier some of that functionality was built using the custom programs. Also found it is difficult for business users to understand forms interface.

G. Assessing Our Solution

14. If there was a way to automate such a process to make things easier, would you like to use it? Do you think it would help you?

Yes, the customizations we built can be embedded to base product.

15. Can you think of any problems such a solution may cause if it were integrated into the OAB module?

As long as they are generic and not specific to any client then no issue.

H. Assessing the Opportunity

16. Do you believe the functionality that the OAB Module offers is sufficient?

Yes, I think OAB module one of the leading product in the ERP space.

I. Outside the Box

17. In a dream world, what would you expect from OAB Module?

To include above mentioned functionality and simple and streamlined UI.

18. What are the pitfalls with the way OAB Module is right now?

Just that more features are needed.
CREATE OR REPLACE PACKAGE sm_enrt_validations_api AS
  --
  PROCEDURE pl_lev_validations (p_business_group_id IN NUMBER, p_person_id IN NUMBER, p_effective_date IN DATE);
  --
  PROCEDURE pl_type_opt_validations (p_business_group_id IN NUMBER, p_person_id IN NUMBER, p_effective_date IN DATE);
  --
END sm_enrt_validations_api;
/
CREATE OR REPLACE PACKAGE body sm_enrt_validations_api AS
  --
  PROCEDURE wl (l_text in varchar2) IS
  --
  BEGIN
  --
  dbms_output.put_line(l_text);
  --
END wl;

  PROCEDURE pl_lev_validations (p_business_group_id IN NUMBER, p_person_id IN NUMBER, p_effective_date IN DATE) IS

  CURSOR SM_PL_ENRT_DEP_CURSOR (cv_business_group_id IN NUMBER) IS
  SELECT * FROM SM_PL_ENRT_DEP
  WHERE business_group_id = cv_business_group_id;

  CURSOR PEN_CURSOR (cv_business_group_id IN NUMBER, cv_person_id IN NUMBER, cv_pl_id IN NUMBER) IS
  SELECT pen.person_id FROM BEN_PRTT_ENRT_RSLT_F pen
WHERE pen.business_group_id = cv_business_group_id
AND pen.person_id = cv_person_id
AND pen.pl_id = cv_pl_id
AND pen.prtt_enrt_rslt_stat_cd IS null
AND pen.effective_end_date = hr_api.g_eot
AND pen.enrt_cvg_thru_dt = hr_api.g_eot;

CURSOR RQD_PEN_CURSOR (cv_business_group_id IN NUMBER,
cv_person_id         IN NUMBER,
cv_pl_id             IN NUMBER) IS
SELECT pen.person_id
FROM BEN_PRTT_ENRT_RSLT_F pen
WHERE pen.business_group_id = cv_business_group_id
AND pen.person_id = cv_person_id
AND pen.pl_id = cv_pl_id
AND pen.prtt_enrt_rslt_stat_cd IS null
AND pen.effective_end_date = hr_api.g_eot
AND pen.enrt_cvg_thru_dt = hr_api.g_eot;

CURSOR get_plan_cursor (cv_business_group_id IN NUMBER,
cv_pl_id             IN NUMBER,
cv_effective_date    IN DATE) IS
SELECT pln.name
FROM BEN_PL_F pln
WHERE pln.business_group_id = cv_business_group_id
AND pln.pl_id = cv_pl_id
AND cv_effective_date between pln.effective_start_date AND
pln.effective_end_date;

l_pl_name      ben_pl_f.name%type;
l_pl_name1     ben_pl_f.name%type;
l_number       number;
l_err_message  varchar2(150);

BEGIN

wl (p_business_group_id);
w1 (p_person_id);

FOR L_PL_ENRT_DEP_REC IN SM_PL_ENRT_DEP_CURSOR
(p_business_group_id)
LOOP

END;
wl ('Inside FOR LOOP SM_PL_ENRT_DEP_CURSOR');
l_err_message := null;

OPEN PEN_CURS0R (p_business_group_id, p_person_id,
                  L_PL_ENRT_DEP_REC.enrolled_pl_id);
FETCH PEN_CURS0R into l_number;

IF PEN_CURSOR%FOUND THEN
  OPEN RQD_PEN_CURS0R (p_business_group_id, p_person_id,
                        L_PL_ENRT_DEP_REC.required_pl_id);
  FETCH RQD_PEN_CURS0R into l_number;

  IF RQD_PEN_CURS0R%NOTFOUND THEN
    OPEN get_plan_cursor (p_business_group_id,
                           l_pl_enrt_dep_rec.required_pl_id,
                           p_effective_date);
    FETCH get_plan_cursor into l_pl_name;
    CLOSE get_plan_cursor;
  END IF;

  OPEN get_plan_cursor (p_business_group_id,
                        l_pl_enrt_dep_rec.enrolled_pl_id,
                        p_effective_date);
  FETCH get_plan_cursor into l_pl_name1;
  CLOSE get_plan_cursor;
  l_err_message := 'You must enroll in plan with id ' ||
                   l_pl_name ||
                   ' when enrolled in ' ||
                   l_pl_name1;

END IF;

CLOSE RQD_PEN_CURS0R;
END IF;

CLOSE PEN_CURS0R;

-- Raise error IF message IS formed.

IF l_err_message IS not null then
fnd_message.set_name('BEN','BEN_92187_POST_ELCN_NOT_PASS');
    fnd_message.set_token('ERROR_MESSAGE',l_err_message);
    fnd_message.raise_error;

--
END if;
--
END LOOP;
--
END pl_lev_validations;
--
PROCEDURE pl_type_opt_validations (p_business_group_id IN NUMBER,
    p_person_id IN NUMBER,
    p_effective_date IN DATE)
IS
--
CURSOR sm_pl_typ_opt_dep_cursor (cv_business_group_id IN NUMBER) IS
SELECT *
FROM SM_PL_TYP_OPT_DEP
WHERE business_group_id = cv_business_group_id;
--
CURSOR pen_cursor (cv_business_group_id IN NUMBER,
    cv_person_id IN NUMBER,
    cv_pl_type_id IN NUMBER,
    cv_option_id IN NUMBER) IS
    SELECT pen.person_id
    FROM BEN_PRTT_ENRT_RSLT_F pen
    WHERE pen.business_group_id = cv_business_group_id
    AND pen.person_id = cv_person_id
    AND pen.pl_typ_id = cv_pl_type_id
    AND pen.prtt_enrt_rslt_stat_cd IS null
    AND pen.effective_end_date = hr_api.g_eot
    AND pen.enrt_cvg_thru_dt = hr_api.g_eot
;
--
CURSOR rqd_pen_cursor (cv_business_group_id IN NUMBER,
    cv_person_id IN NUMBER,
    cv_pl_type_id IN NUMBER,
    cv_option_id IN NUMBER) IS
    SELECT pen.person_id
    FROM BEN_PRTT_ENRT_RSLT_F pen
    WHERE pen.business_group_id = cv_business_group_id
    AND pen.person_id = cv_person_id
    AND pen.pl_typ_id = cv_pl_type_id
    AND pen.prtt_enrt_rslt_stat_cd IS null
    AND pen.effective_end_date = hr_api.g_eot
    AND pen.enrt_cvg_thru_dt = hr_api.g_eot
;
AND pen.person_id = cv_person_id
AND pen.pl_typ_id = cv_pl_type_id
AND pen.prtt_enrt_rslt_stat_cd IS null
AND pen.effective_end_date = hr_api.g_eot
AND pen.enrt_cvg_thru_dt = hr_api.g_eot

CURSOR get_pl_type_cursor (cv_business_group_id IN NUMBER, 
cv_pl_type_id        IN NUMBER, 
cv_effective_date    IN DATE) IS
SELECT pln.name
FROM BEN_PL_TYP_F pln
WHERE pln.business_group_id = cv_business_group_id
AND pln.pl_typ_id    = cv_pl_type_id
AND cv_effective_date between
    pln.effective_start_date AND
pln.effective_end_date;

CURSOR get_option_cursor (cv_business_group_id IN NUMBER, 
cv_option_id         IN NUMBER, 
cv_effective_date    IN DATE) IS
SELECT pln.name
FROM BEN_OPT_F pln
WHERE pln.business_group_id = cv_business_group_id
AND pln.option_id     = cv_option_id
AND cv_effective_date between
    pln.effective_start_date AND
pln.effective_end_date;

l_type_name    ben_pl_typ_f.name%type;
l_type_name1   ben_pl_typ_f.name%type;
l_option_name  ben_opt_f.name%type;
l_option_name1 ben_opt_f.name%type;
l_number       number;
l_err_message  varchar2(150);

BEGIN

FOR l_pl_typ_opt_dep_rec IN sm_pl_typ_opt_dep_cursor
    (p_business_group_id)
LOOP

    l_err_message := null;

    OPEN pen_cursor (p_business_group_id, p_person_id,
L_PL_TYP_OPT_DEP_REC.enrolled_pl_type_id,
L_PL_TYP_OPT_DEP_REC.enrolled_option_id);

FETCH pen_cursor into l_number;

IF pen_cursor%FOUND THEN

OPEN rqd_pen_cursor (p_business_group_id, p_person_id,
l_pl_typ_opt_dep_rec.required_pl_type_id,
l_pl_typ_opt_dep_rec.required_option_id);

FETCH rqd_pen_cursor into l_number;

IF rqd_pen_cursor%NOTFOUND THEN

OPEN get_pl_type_cursor (p_business_group_id,
l_pl_typ_opt_dep_rec.required_pl_type_id,
p_effective_date);

FETCH get_pl_type_cursor into l_type_name;
CLOSE get_pl_type_cursor;

OPEN get_option_cursor (p_business_group_id,
l_pl_typ_opt_dep_rec.required_option_id,
p_effective_date);

FETCH get_option_cursor into l_option_name;
CLOSE get_option_cursor;

OPEN get_pl_type_cursor (p_business_group_id,
l_pl_typ_opt_dep_rec.enrolled_pl_type_id,
p_effective_date);

FETCH get_pl_type_cursor into l_type_name1;
CLOSE get_pl_type_cursor;

OPEN get_option_cursor (p_business_group_id,
l_pl_typ_opt_dep_rec.enrolled_option_id,
p_effective_date);

FETCH get_option_cursor into l_option_name1;
CLOSE get_option_cursor;

l_err_message := 'You must enroll in the plan type'

||
l_type_name || ' with option ' ||
   l_option_name ||
   ' When enrolled in ' ||
   l_type_name1 || ' with option ' ||
   l_option_name1;

--
END IF;
--
CLOSE rqd_pen_cursor;
--
END if;
--
CLOSE pen_cursor;
--
-- Raise error IF message IS formed.
--
IF l_err_message IS not null then
--

fnd_message.set_name('BEN','BEN_92187_POST_ELCN_NOT_PASS');
fnd_message.set_token('ERROR_MESSAGE',l_err_message);
fnd_message.raise_error;
--

END if;
--
END LOOP;
--
END pl_type_opt_validations;
--
END sm_enrt_validations_api;
/

81
CREATE OR REPLACE PACKAGE sm_pl_move_enrollments_api AS

PROCEDURE pl_move_enrollments (p_business_group_id IN NUMBER,
                            p_person_id     IN NUMBER,
                            p_ler_id        IN NUMBER,
                            p_lf_evt_ocrd_dt IN DATE);

PROCEDURE chk_lf_evt_stat (p_business_group_id IN NUMBER,
                            p_person_id     IN NUMBER,
                            p_lf_evt_ocrd_dt IN DATE,
                            p_le_ongoing    OUT CHAR);

PROCEDURE chk_future_lf_evts (p_business_group_id IN NUMBER,
                             p_person_id     IN NUMBER,
                             p_lf_evt_ocrd_dt IN DATE,
                             p_le_exists     OUT CHAR);

PROCEDURE sm_process_le (p_business_group_id IN NUMBER,
                            p_person_id     IN NUMBER,
                            p_ler_id        IN NUMBER,
                            p_lf_evt_ocrd_dt IN DATE);

PROCEDURE sm_switch_enrollment (p_business_group_id IN NUMBER,
                                p_person_id     IN NUMBER,
                                p_ler_id        IN NUMBER,
                                p_lf_evt_ocrd_dt IN DATE);

END sm_pl_move_enrollments_api;
/

CREATE OR REPLACE PACKAGE BODY sm_pl_move_enrollments_api AS

PROCEDURE wl(p_line in varchar2) IS
BEGIN
   dbms_output.put_line(p_line);
END;

PROCEDURE chk_lf_evt_stat (p_business_group_id IN NUMBER,
                            p_person_id     IN NUMBER,
                            p_lf_evt_ocrd_dt IN DATE,
                            p_le_ongoing    OUT CHAR) IS

   -- To check IF there's an ongoing life event
---
CURSOR sm_lf_evt_stat_cursor (cv_business_group_id IN NUMBER,
    cv_person_id IN NUMBER,
    cv_lf_evt_ocrd_dt IN DATE)
IS
SELECT pil.person_id
FROM ben_per_in_ler pil,
    ben_ler_f lerf
WHERE pil.business_group_id = cv_business_group_id
    AND pil.ler_id = lerf.ler_id
    AND pil.person_id = cv_person_id
    AND pil.per_in_ler_stat_cd = 'STRTD'
    AND cv_lf_evt_ocrd_dt BETWEEN lerf.effective_start_date
        AND lerf.effective_end_date
    AND lerf.typ_cd NOT IN ('UNRSTCD', 'IREC',
        'CWB', 'COMP', 'GSP');

--
ignore_id NUMBER;
--
BEGIN
--
OPEN sm_lf_evt_stat_cursor (p_business_group_id,
    p_person_id,
    p_lf_evt_ocrd_dt);
FETCH sm_lf_evt_stat_cursor INTO ignore_id;
CLOSE sm_lf_evt_stat_cursor;
--
IF ignore_id IS NOT NULL THEN
--
p_le_ongoing := 'Y';
--
ELSE
--
p_le_ongoing := 'N';
--
END IF;
--
w1('inside chk_lf_evt_stat : ' || p_le_ongoing);
END chk_lf_evt_stat;
--

PROCEDURE chk_future_lf_evts (p_business_group_id IN NUMBER,
    p_person_id IN NUMBER,
    p_lf_evt_ocrd_dt IN DATE,
    p_le_exists OUT CHAR) IS
-- To check IF there IS a future processed life event
--
CURSOR sm_chk_future_lf_evts_cursor (cv_business_group_id IN NUMBER,
cv_person_id IN NUMBER,
cv_lf_evt_ocrd_dt IN DATE) IS
    SELECT pil.person_id
    FROM ben_per_in_ler pil,
        ben_ler_f lerf
    WHERE pil.business_group_id = cv_business_group_id
        AND pil.ler_id = lerf.ler_id
        AND pil.person_id = cv_person_id
        AND pil.per_in_ler_stat_cd = 'PROCD'
        AND pil.lf_evt_ocrd_dt >= cv_lf_evt_ocrd_dt
        AND cv_lf_evt_ocrd_dt BETWEEN lerf.effective_start_date
                                    AND lerf.effective_end_date
        AND lerf.typ_cd NOT IN ('UNRSTCD', 'IREC', 'CWB',
                      'COMP','GSP');
--
    ignore_id2 NUMBER;
--
BEGIN
--
    OPEN sm_chk_future_lf_evts_cursor (p_business_group_id,
            p_person_id,
            p_lf_evt_ocrd_dt);

    FETCH sm_chk_future_lf_evts_cursor INTO ignore_id2;
    CLOSE sm_chk_future_lf_evts_cursor;
--
    IF ignore_id2 IS NOT NULL THEN
--
        p_le_exists := 'Y';
--
    ELSE
--
        p_le_exists := 'N';
--
    END IF;
--
    wl('inside chk_future_lf_evts : '|| p_le_exists);
END chk_future_lf_evts;
PROCEDURE CREATE_ptnl_lf_evt (p_business_group_id IN NUMBER,
p_person_id IN NUMBER,
p_ler_id IN NUMBER,
p_lf_evt_ocrd_dt IN DATE) IS

-- CREATE a potential life event of a given ler_id FOR the person given lf_evt_ocrd_dt

CURSOR sm_chk_if_le_exists (cv_business_group_id IN NUMBER,
cv_person_id IN NUMBER,
cv_ler_id IN NUMBER,
cv_lf_evt_ocrd_dt IN DATE) IS

SELECT 'Y'
FROM ben_ptnl_ler_for_per plfp
WHERE plfp.business_group_id = cv_business_group_id
  AND plfp.person_id = cv_person_id
  AND plfp.ler_id = cv_ler_id
  AND plfp.lf_evt_ocrd_dt = cv_lf_evt_ocrd_dt
  AND plfp.ptnl_ler_for_per_stat_cd = 'DTCTD';

l_ovn NUMBER;
l_ptnl_id NUMBER;
l_le_exists VARCHAR2(1);

BEGIN

l_le_exists := 'N';
OPEN sm_chk_if_le_exists(p_business_group_id,
p_person_id,
p_ler_id,
p_lf_evt_ocrd_dt);

FETCH sm_chk_if_le_exists INTO l_le_exists;

IF sm_chk_if_le_exists%NOTFOUND THEN
  -- ben_CREATE_ptnl_ler_for_per.CREATE_ptnl_ler_event
  wl('Creating potential life event');
  ben_ptnl_ler_for_per_api.CREATE_ptnl_ler_for_per
    (p_validate => false,
     p_ptnl_ler_for_per_id => l_ptnl_id,
     p_ntfn_dt => SYSDATE,
     p_lf_evt_ocrd_dt => p_lf_evt_ocrd_dt,
     p_ptnl_ler_for_per_stat_cd => 'DTCTD')
END IF;

END;
/
PROCEDURE sm_process_le (p_business_group_id IN  NUMBER, 
p_person_id         IN  NUMBER, 
p_ler_id            IN  NUMBER, 
p_lf_evt_ocrd_dt    IN  DATE) IS

--
l_errbuf VARCHAR2(1000); 
l_retcode NUMBER;
--
BEGIN
--
ben_manage_life_events.process
(
  errbuf => l_errbuf, 
  retcode => l_retcode, 
  p_effective_date => p_lf_evt_ocrd_dt, 
  p_mode => 'L', 
  p_benefit_action_id => null, 
  p_derivable_factors => 'Y', 
  p_validate => 'N', 
  p_person_id => p_person_id, 
  p_person_type_id => null, 
  p_business_group_id => p_business_group_id, 
  p_pgm_id => null, 
  p_pl_id => null, 
  p_popl_enrt_typ_cycl_id => null, 
  p_no_programs => 'N', 
  p_no_plans => 'N', 
  p_comp_selection_rule_id => null, 
  p_person_selection_rule_id => null, 
  p_ler_id => null,
PROCEDURE pl_move_enrollments (p_business_group_id IN NUMBER,
p_person_id IN NUMBER,
p_ler_id IN NUMBER,
p_lf_evt_ocrd_dt IN DATE) IS

CURSOR C_pil IS
SELECT b.PER_IN_LER_ID
    FROM ben_per_in_ler b,
         ben_ler_f ler
WHERE b.per_in_ler_stat_cd = 'STRTD'
    AND b.person_id = p_person_id
    AND b.ler_id = ler.ler_id
    AND ler.typ_cd NOT IN ('UNRSTCD', 'IREC', 'CWB', 'COMP', 'GSP')
    AND p_lf_evt_ocrd_dt between
        ler.effective_start_date and
        ler.effective_end_date
;

l_le_ongoing char(1);
l_le_exists char(1);
l_per_in_ler_id NUMBER;

BEGIN

wl('BEGIN   processing of Person with person id : ' || p_person_id);

END pl_move_enrollments;
sm_pl_move_enrollments_api.chk_lf_evt_stat
(p_business_group_id,
    p_person_id,
    p_lf_evt_ocrd_dt,
    l_le_ongoing);

--
sm_pl_move_enrollments_api.chk_future_lf_evts
(p_business_group_id,
    p_person_id,
    p_lf_evt_ocrd_dt,
    l_le_exists);

--
IF l_le_ongoing = 'Y' OR l_le_exists = 'Y'
then
    wl('Skipped processing of Person with person id : '||
        p_person_id);
    END if;
--
IF l_le_ongoing = 'N' AND l_le_exists = 'N'
then
    --
    wl('Processing of Person with person id : '||
        p_person_id);
    sm_pl_move_enrollments_api.create_ptnl_lf_evt
    (p_business_group_id,
        p_person_id,
        p_ler_id,
        p_lf_evt_ocrd_dt);
    --
    wl(' Now process the potential life event created above');
-- Now process the potential life event created above
--
    sm_process_le (p_business_group_id,
        p_person_id,
        p_ler_id,
        p_lf_evt_ocrd_dt);
--
-- Now move enrollments
--
    sm_switch_enrollment (p_business_group_id,
        p_person_id,
        p_ler_id,
        p_lf_evt_ocrd_dt);
--
-- Close the enrollments

open c_pil;
fetch c_pil into l_per_in_ler_id;
close c_pil;

ben_close_enrollment.close_single_enrollment(
    p_per_in_ler_id => l_per_in_ler_id,
    p_effective_date => p_lf_evt_ocrd_dt,
    p_business_group_id => p_business_group_id,
    p_validate => false,
    p_close_uneai_flag => 'Y',
    p_uneai_effective_date => p_lf_evt_ocrd_dt
);

END if;

END pl_move_enrollments;

PROCEDURE sm_switch_enrollment (p_business_group_id IN NUMBER,
    p_person_id         IN NUMBER,
    p_ler_id IN NUMBER,
    p_lf_evt_ocrd_dt    IN DATE) IS

CURSOR sm_switch_enrollment_cursor (cv_business_group_id
    IN NUMBER,     cv_person_id
    IN NUMBER,     cv_lf_evt_ocrd_dt
    IN DATE,       cv_old_pl_id
    IN NUMBER,     cv_old_opt_id
    IN NUMBER) IS

SELECT md.new_pl_id, md.new_opt_id
    FROM SM_pl_mapping_master mm,
         SM_pl_mapping_details md
WHERE mm.pl_mapping_master_id = md.pl_mapping_master_id
    AND mm.ler_id = p_ler_id
    AND p_person_id = cv_person_id
    AND mm.lf_evt_ocrd_dt = cv_lf_evt_ocrd_dt;
CURSOR PEN_CURSOR (cv_business_group_id IN NUMBER,
     cv_person_id     IN NUMBER) IS
     SELECT pen.pl_id, pen.oipl_id, pen.pgm_id
     FROM BEN_PRTT_ENRT_RSLT_F pen
     WHERE pen.business_group_id  = cv_business_group_id
     AND pen.person_id          = cv_person_id
     AND pen.prtt_enrt_rslt_stat_cd IS null
     AND pen.effective_end_date = hr_api.g_eot
     AND pen.enrt_cvg_thru_dt   = hr_api.g_eot
     ;

CURSOR get_option_cursor (cv_business_group_id IN NUMBER,
     cv_oipl_id           IN NUMBER,
     cv_effective_date    IN DATE) IS
     SELECT oipl.opt_id
     FROM BEN_OIPL_F oipl
     WHERE oipl.business_group_id = cv_business_group_id
     AND oipl.oipl_id           = cv_oipl_id
     AND cv_effective_date between
         oipl.effective_start_date AND
         oipl.effective_end_date;

     l_option_id  NUMBER;
     l_new_pl_id  NUMBER;
     l_new_opt_id NUMBER;
     l_pgm_id     NUMBER;

     BEGIN

     FOR L_ENRT_REC IN PEN_CURSOR (p_business_group_id, p_person_id)
     LOOP

     l_new_pl_id := NULL;
     l_new_opt_id := NULL;
     l_option_id := NULL;
     l_pgm_id     := l_enrt_rec.pgm_id;

     OPEN get_option_cursor (p_business_group_id, l_enrt_rec.oipl_id, p_lf_evt_ocrd_dt);

     FETCH get_option_cursor INTO l_option_id;
     CLOSE get_option_cursor;
OPEN sm_switch_enrollment_cursor (p_business_group_id, p_person_id, p_lf_evt_ocrd_dt, l_enrt_rec.pl_id, l_option_id);

FETCH sm_switch_enrollment_cursor INTO l_new_pl_id, l_new_opt_id;
CLOSE sm_switch_enrollment_cursor;

IF l_new_pl_id IS NOT NULL THEN
  ben_enrollment_process.enrollment_information_detail (p_validate => FALSE, p_ler_id => p_ler_id, p_pgm_id => L_ENRT_REC.pgm_id, p_pl_id => l_new_pl_id, p_opt_id => l_new_opt_id, p_life_event_date => p_lf_evt_ocrd_dt, p_ended_pl_id => l_enrt_rec.pl_id, p_ended_opt_id => l_option_id, p_effective_date => p_lf_evt_ocrd_dt, p_person_id => p_person_id, p_business_group_id => p_business_group_id);
ELSE
  ben_enrollment_process.enrollment_information_detail (p_validate => FALSE, p_ler_id => p_ler_id, p_pgm_id => L_ENRT_REC.pgm_id, p_pl_id => l_enrt_rec.pl_id, p_opt_id => l_option_id, p_life_event_date => p_lf_evt_ocrd_dt, p_ended_pl_id => NULL, p_ended_opt_id => NULL, p_effective_date => p_lf_evt_ocrd_dt, p_person_id => p_person_id, p_business_group_id => p_business_group_id);
END IF;

END LOOP;

IF l_pgm_id IS NOT NULL THEN
  ben_enrollment_process.post_enrollment (p_validate => FALSE, p_ler_id => p_ler_id, p_pgm_id => L_ENRT_REC.pgm_id, p_pl_id => l_enrt_rec.pl_id, p_opt_id => l_option_id, p_life_event_date => p_lf_evt_ocrd_dt, p_ended_pl_id => NULL, p_ended_opt_id => NULL, p_effective_date => p_lf_evt_ocrd_dt, p_person_id => p_person_id, p_business_group_id => p_business_group_id);
END IF;
, p_person_id => p_person_id,
, p_ler_id => p_ler_id,
, p_life_event_date => p_lf_evt_ocrd_dt,
, p_pgm_id => l_pgm_id,
, p_pl_id => NULL,
, p_proc_cd => NULL,
, p_business_group_id => p_business_group_id,
, p_effective_date => p_lf_evt_ocrd_dt);

END IF;

--

END sm_switch_enrollment;

--

--

END sm_pl_move_enrollments_api;
/

Appendix C: Tables and Insert Statements

Feature 1

DROP TABLE SM_PL_ENRT_DEP;
DROP TABLE SM_PL_TYP_OPT_DEP;

REM Create Tables

connect ben/ben;

CREATE TABLE SM_PL_ENRT_DEP
(PL_ENRT_DEP_ID NUMBER CONSTRAINT SM_PL_ENRT_DEP_pk primary key,
business_group_id NUMBER,
enrolled_pl_id NUMBER,
required_pl_id NUMBER,
start_date DATE,
end_date DATE
);

CREATE TABLE SM_PL_TYP_OPT_DEP
(PL_TYP_OPT_DEP_ID NUMBER CONSTRAINT SM_PL_TYP_OPT_DEP_pk primary key,
business_group_id NUMBER,
enrolled_pl_type_id NUMBER,
enrolled_option_id NUMBER,
required_pl_type_id NUMBER,
required_option_id NUMBER,
start_date DATE,
end_date DATE
);

REM Create Sequences for both tables

CREATE SEQUENCE SM_PL_ENRT_DEP_SEQ INCREMENT BY 1 START WITH 1;
CREATE SEQUENCE SM_PL_TYP_OPT_DEP_SEQ INCREMENT BY 1 START WITH 1;

REM Give grants to Apps Schema

GRANT ALL ON SM_PL_ENRT_DEP TO Apps;
GRANT ALL ON SM_PL_TYP_OPT_DEP TO Apps;
GRANT ALL ON SM_PL_ENRT_DEP_SEQ TO Apps;
GRANT ALL ON SM_PL_TYP_OPT_DEP_SEQ TO Apps;
connect apps/apps;

REM PUBLIC SYNONYMS UNDER APPS SCHEMA

create public synonym SM_PL_ENRT_DEP for ben.SM_PL_ENRT_DEP;
create public synonym SM_PL_TYP_OPT_DEP for ben.SM_PL_TYP_OPT_DEP;
create public synonym SM_PL_ENRT_DEP_SEQ for ben.SM_PL_ENRT_DEP_SEQ;
create public synonym SM_PL_TYP_OPT_DEP_SEQ for ben.SM_PL_TYP_OPT_DEP_SEQ;

Feature 2

drop table SM_pl_mapping_details;
drop table SM_pl_mapping_master;

REM Create Tables

connect ben/ben;

CREATE TABLE SM_pl_mapping_master
(   pl_mapping_master_id NUMBER(15) CONSTRAINT SM_pl_mapping_master_pk primary key,
    ler_id NUMBER,
    lf_evt_ocrd_dt DATE,
    start_date DATE,
    end_date DATE
));

CREATE TABLE SM_pl_mapping_details
(   pl_mapping_details_id NUMBER(15) CONSTRAINT SM_pl_mapping_details_pk primary key,
    pl_mapping_master_id NUMBER(15) CONSTRAINT SM_pl_mapping_master_fk references SM_pl_mapping_master(pl_mapping_master_id),
    old_pl_id NUMBER,
    old_opt_id NUMBER,
    new_pl_id NUMBER,
    new_opt_id NUMBER,
    start_date DATE,
    end_date DATE
);
REM Create Sequences for both tables

CREATE SEQUENCE SM_pl_mapping_master_seq INCREMENT BY 1 START WITH 1;
CREATE SEQUENCE SM_pl_mapping_details_seq INCREMENT BY 1 START WITH 1;

REM Give grants to Apps Schema

GRANT ALL ON SM_pl_mapping_master TO apps;
GRANT ALL ON SM_pl_mapping_details TO apps;
GRANT ALL ON SM_pl_mapping_master_seq TO apps;
GRANT ALL ON SM_pl_mapping_details_seq TO apps;

connect apps/apps;

REM PUBLIC SYNONYMS UNDER APPS SCHEMA

create public synonym SM_pl_mapping_master for ben.SM_pl_mapping_master;
create public synonym SM_pl_mapping_details for ben.SM_pl_mapping_details;
create public synonym SM_pl_mapping_master_seq for ben.SM_pl_mapping_master_seq;
create public synonym SM_pl_mapping_details_seq for ben.SM_pl_mapping_details_seq;
Appendix D: Screenshots

Feature 1

The following screenshots were taken when the SM Benefits Enrollment Plan was created using the Benefits module of the Oracle Application. Each screenshot has a caption which explains what plan type or plan or option is being created.

Figure 15: Life Event Reasons – Creating a life event reason to enroll a certain person in Benefits. In this case, the life event reason created was for a New Hire.
Figure 16: Plan Types – Creating a Medical Plan Type

Figure 17: Plan Types – Creating a Dental Plan Type

97
Figure 18: Plan Types – Creating a Vision Plan Type

Figure 19: Plan Types – Creating a Life Insurance Plan Type
Figure 20: Plan Types – Creating a Spouse Life Insurance Option

Figure 21: Plan Types – Creating a Dependent Life Insurance Option
Figure 22: Options – Creating an Employee Only Option

Figure 23: Options – Creating an Employee Plus Spouse Option
Figure 24: Options – Creating an Employee Plus Family Option

Figure 25: Options – Creating a Waive Option
Figure 26: Plans and Options—Creating the United Health Care (UHC) Medical Plan and applying the Options, SM Employee Only, SM Employee Plus Family, SM Employee Plus Spouse, and SM Waive to the plan.
Figure 27: Plans and Options—Creating the Kaiser Medical Plan and applying the Options, SM Employee Only, SM Employee Plus Family, SM Employee Plus Spouse, and SM Waive to the plan.
Figure 28: Plans and Options—Creating the Cigna Medical Plan and applying the Options, SM Employee Only, SM Employee Plus Family, SM Employee Plus Spouse, and SM Waive to the plan.
Figure 29: Plans and Options – Creating the Cigna Dental Plan and applying the Options, SM Employee Only, SM Employee Plus Family, SM Employee Plus Spouse, and SM Waive to the plan.
Figure 30: Plans and Options—Creating the United Health Care (UHC) Dental Plan and applying the Options, SM Employee Only, SM Employee Plus Family, SM Employee Plus Spouse, and SM Waive to the plan.
Figure 31: Plans and Options—Creating the Kaiser Dental Plan and applying the Options, SM Employee Only, SM Employee Plus Family, SM Employee Plus Spouse, and SM Waive to the plan.
Figure 32: Plans and Options—Creating the Cigna Vision Plan and applying the Options, SM Employee Only, SM Employee Plus Family, SM Employee Plus Spouse, and SM Waive to the plan.
Figure 33: Plans and Options—Creating the Kaiser Vision Plan and applying the Options, SM Employee Only, SM Employee Plus Family, SM Employee Plus Spouse, and SM Waive to the plan.
Figure 34: Plans and Options—Creating the United Health Care (UHC) Vision Plan and applying the Options, SM Employee Only, SM Employee Plus Family, SM Employee Plus Spouse, and SM Waive to the plan.
Figure 35: Plans – Creating the Employee Supplemental Life Insurance plan.

Figure 36: Plans – Creating the Spouse Supplemental Life Insurance plan.
Figure 37: Plans – Creating the Dependent Supplemental Life Insurance plan.

Figure 38: Programs – Making the SM Employee Benefits Program
Figure 39: Plan and Plan Type – Applying all the plans created to the SM Employee Benefits Program.
Figure 40: Plan and Plan Type – Applying all the plans created to the SM Employee Benefits Program.

Figure 41: Setting up the Enrollment Coverage Start and End Dates
Figure 42: Creating a Life Event titled SM New Hire

Figure 43: Applying Timings for the Life Event as far as when Coverage Ends.
Figure 44: Applying Start and End Dates for the SM New Hire Life Event

Figure 45: Putting the Life Event SM New Hire into Motion
It's not visible in this screenshot, but the user was attempted to enroll in the Spouse Supplemental Life Insurance plan, and the Dependent Supplemental Life Insurance Plans without enrolling in the Employee Supplemental Life Insurance Plan, and an error message was displayed, asking the user to enroll in the Employee Supplemental Life Insurance Plan first. Here's a screenshot of the error message that was displayed:

Figure 47: Error message displayed when user attempted to enroll a Spouse in Life Insurance before enrolling himself in the Employee Only Life Insurance first.
The same results were seen when the user was attempted to enroll in the Dental and Vision plans without enrolling in the Medical Plan first.

Figure 48: Error message displayed when user attempted to enroll himself in a Dental plan before enrolling himself in the Medical plan first

After validations were done, the user was enrolled in all the plans in the correct order, and was successfully able to do so, without any error messages being raised.
Feature 2

Figure 49: Creating a new life event called SM Move Enrollments for the movement of the benefits enrollments
Figure 50: Applying the SM Move Enrollments Life Event reason to the SM Employee Benefits Program

Figure 51: Test employee: Mrs. Tamara Aaron – who was previously enrolled into the SM Benefits Program under the Life Event Reason - “New Hire” is backed out of the Life Event since only one life event can be happening to an employee at a time.
Figure 52: Test employee: SM New Hire Life Event is voided.

Figure 53: Mrs. Tamara Aaron – who was previously enrolled into the SM Benefits Program under the Life Event Reason - “New Hire” is switched to the “Move Enrollments” Life Event.
Figure 54: Life event has been processed for this employee

Figure 55: Existing Medical Plan: SM Kaiser Medical SM Employee Only.
Figure 56: Successful movement of the Enrollment Plan and option: From Kaiser Medical Employee Only to Cigna Medical Employee Only.