FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Especificação da adaptação do módulo Oracle Retail Merchandising System (ORMS) aos processos de negócio dos retalhistas.

Frederico Brandão Figueiredo

VERSÃO DEFINITIVA

Relatório de Projecto/Dissertação
Mestrado Integrado em Engenharia Informática e Computação

Orientador FEUP: João Moreira

Orientador Wipro Retail: José Pereira

Junho de 2009

Especificação da adaptação do módulo Oracle Retail Merchandising System (ORMS) aos processos de negócio dos retalhistas.

Frederico Brandão Figueiredo

Relatório de Projecto
Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: M. Cristina Ribeiro

_____

Arguente: Vítor Pereira (ULus)

Vogal: João Moreira

29 de Junho de 2009

Resumo

O controlo da informação que um retalhista possui dos seus artigos é uma chave muito importante no que diz respeito a tomada de decisões que afectam os seus processos de negócio. Manter toda a informação de todos os artigos armazenada nos sistemas de software revela-se crucial para o sucesso do negócio. Manter esta informação actualizada torna-se ainda mais importante.

O objectivo deste trabalho passa por incluir novos atributos e funcionalidade a uma ferramenta já de si completa, o Oracle Retail Merchandising System (ORMS). A esta ferramenta pretende-se adicionar atributos que permitem ao retalhista identificar se determinado artigo tem o preço incluído no código de barras, se tem o preço marcado na embalagem ou se é necessário saber o seu preço à unidade de forma a facilitar a comparação com outros artigos.

A solução foi encontrada seguindo a metodologia usada na Wipro Retail. Depois de realizado um Conference Room Pilot (CRP), foi gerado um documento, juntamente com o retalhista, que descreve as necessidades do negócio assim como todas as funcionalidades que deverão estar presentes após as alterações: Functional Requirement Document (FRD). Baseado no FRD é construído um novo documento de teor técnico, o Technical Requirement Document (TRD). Após a construção e aprovação de ambos os documentos, seguem-se as fases de desenvolvimento, testes e implementação.

Durante o decorrer do projecto, foram desenvolvidos os FRD e TRD para as alterações propostas. Aquando da entrega deste relatório, as alterações encontravam-se em fase de desenvolvimento, estando previsto que este esteja concluído até Setembro 2009.

No final do projecto, o aluno adquiriu conhecimentos de negócio nos processos mais críticos na área do retalho. Além disso, conheceu as metodologias associadas a um projecto de Delivery e adquiriu um profundo conhecimento das tecnologias usadas.

Abstract

The information control a retailer can get on its products is a very important key when decisions, that concern its business processes, need to be made. Maintaining all this information, for all the products, in the software system is crucial for the success of the business. To maintain all this information updated becomes even more important.

The goal of this work is to include new attributes and functionality to a tool already very complete by itself, the Oracle Retail Merchandising System (ORMS). This tool needs to be updated by including attributes that allow the retailer to identify if certain product has its price embedded into the barcode, if it has the price printed on the package, or if it's necessary to know the Unit Equivalent Price (UEP) of the product so that products can be compared to each other.

The solution was found by following Wipro Retail Methodology. After the Conference Room Pilot (CRP), a document was designed with the retailer, which fully describes every business need and functionality that must be present after the changes are done: the Functional Requirement Document (FRD). Based on this document, the Technical Design Document (TRD) was built. After both of these documents being designed, the following steps were development, testing and implementation.

During the project lifecycle, these FRD and TRD documents were design ed for the proposed changes. By the time this report was delivered, these changes were being developed, and they were expected to be delivered by September 2009.

By the end of this project, the student has acquired business knowledge on the most critical processes for the retail area. Above that, he has gained contact with the methods related with a Delivery project and has acquired a deep knowledge on the technologies used.

Agradecimentos

Na certeza de que não mencionarei todos quantos contribuíram para que eu fosse capaz de chegar a esta fase, não poderia deixar de referir a minha gratidão pelos que me auxiliaram e acolheram e que, de diversas formas, me ajudaram e apoiaram nesta fase final do curso e da realização do meu projecto.

Aos meus pais, um agradecimento especial, pelo apoio e amizade que sempre me dispensaram, nos bons e nos maus momentos, acreditando sempre que eu era capaz.

À Wipro Retail, pela oportunidade de efectuar o projecto numa empresa jovem, dinâmica e ambiciosa, proporcionando-me as melhores condições.

Agradeço ao José Pereira pela orientação e apoio que me deu em todos os momentos que compuseram este projecto.

Não posso, também, esquecer de agradecer a todos os meus colegas de trabalho pela amizade e boa disposição com que fui recebido. Uma palavra especial de agradecimento para a Regina e Dória.

Um agradecimento especial também ao meu orientador da FEUP, professor João Moreira, por todo o apoio e pelas sugestões dadas.

Agradeço, ainda, a todos os meus amigos, que sempre me apoiaram e que me fizeram esquecer que a vida nem sempre é fácil!

E finalmente, à Sandra, por tudo!!

Índice

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

BRD	Business Requirement Document
CRP	Conference Room Pilot
EDI	Electronic Data Interchange
ERP	Enterprise Resource Planning
FRD	Functional Requirement Document
MRP	Material Resource Planning
MRP-II	Manufacturing Resource Planning
OR	Oracle Retail
ORMS	Oracle Retail Merchandising System
ORPM	Oracle Retail Price Management
TRD	Technical Requirement Document

1 Introdução

A necessidade de se manterem actualizados, com capacidade de fazer frente à concorrência e optimizar o seu processo de negócio, faz com que vários retalhistas em todo o mundo recorram a empresas especializadas, com o intuito de adaptar um Enterprise Resource Planning (ERP) aos seus processos de negócio, com o objectivo de manter a vantagem competitiva face aos seus competidores.

A Wipro Retail é uma empresa de referência no desenho, implementação e suporte de sistemas de informação para retalho, bem como de programas de transformação de negócio (Business Transformation). O principal objectivo da Wipro Retail em qualquer implementação onde participa é o de conseguir conciliar um ERP com as necessidades do cliente, mantendo o que muitos clientes chamam da “magia” da empresa (certos processos de negócio que determinada empresa acha crucial para o desempenho desta) e que dá ao cliente um cunho pessoal e algo que na ideia de cada cliente lhe permite aumentar a competitividade face aos seus concorrentes.

Pelo facto de trabalhar com clientes em diversas partes do mundo, com diferentes culturas, a Wipro Retail é uma empresa com um vasto conhecimento em negócio de retalho em diversas situações. Esta experiência, aliada ao conhecimento e à experiência em Tecnologias de Informação (IT) da Wipro Retail, assegura que esta encontre as melhores soluções para os diversos casos e reduza o risco na hora da implementação de projectos considerados críticos.

O projecto realizado está relacionado com o sistema de gestão de artigos presente no módulo Oracle Retail Merchandising System (ORMS). Durante o CRP (Conference Room Pilot) do projecto foram identificados algumas lacunas do sistema de gestão de artigos base do ERP. Assim, o trabalho realizado, concentrou-se na

análise e resolução destas lacunas através da adaptação do sistema base aos processos de negócio do retalhista.

Utilizando o sistema base, o retalhista em questão viu-se em muito limitado face a determinados atributos que utilizava para caracterizar os seus artigos. Atributos esses que considerava essenciais e que portanto não os queria perder neste novo sistema. Dado que o sistema base a implementar não contemplava grande parte destes atributos, a solução encontrada passou pela completa re-estruturação de um formulário/ecrã do sistema base, para que, através da adição de novas funcionalidades e atributos, o retalhista seja capaz de optimizar os seus processos de negócio, deixando-o assim melhor capacitado para acompanhar a evolução do mercado do retalho.

1.1 Projecto

Os retalhistas enfrentam um ambiente altamente competitivo e em mutação constante para o qual são necessárias respostas rápidas e ajustadas à lei da oferta e da procura.

Problemas típicos com que se debatem actualmente são, a título de exemplo, o aumento da concorrência, a redução dos períodos de reacção às alterações impostas pelo mercado e o aumento da complexidade crescente das operações de retalho. Por estas e outras razões, os retalhistas têm vindo a substituir os seus modelos tradicionais por estratégias que lhes permitam compreender o mercado e agir em tempo útil, protegendo as suas margens e ao mesmo tempo garantindo a satisfação dos clientes.

Este projecto consiste na implementação de uma solução que permita a um retalhista fazer face a esta dinâmica apoiando-o na gestão eficiente e no controlo das actividades diárias de merchandising permitindo a optimização do ciclo de vida de gestão de produtos.

O projecto foca-se na adaptação do sistema de gestão de artigos base aos processos de negócio do retalhista. Esta fase do projecto realiza-se após o levantamento de todos os requisitos do negócio, de forma a que todas as lacunas do sistema base sejam preenchidas. Um projecto de implementação passa por diversas fases, sendo que numa primeira fase é necessário definir quais as potenciais diferenças que existem entre o sistema que será implementado e as necessidades do cliente. Durante um conjunto de reuniões, o Conference Room Pilot (CRP), são apresentadas ao cliente as diversas funcionalidades do sistema, e é discutido com este se aquilo que o sistema apresenta assegura as necessidades em termos de negócio do cliente. Nestas reuniões são detectadas e registadas todas as adaptações

que serão precisas fazer ao sistema base aos processos de negócio do retalhista. Depois para cada um destas adaptações é feita uma análise funcional onde é discutido com o retalhista mais pormenorizadamente cada alteração e é acordado com ele a funcionalidade a adoptar. Depois passamos para a análise técnica, onde é efectuado o desenho a nível da aplicação, alterações em termos de código. Depois desta fase de desenho, segue-se o desenvolvimento e testes: nestas fases as alterações são produzidas e testadas até que se obtenha o produto pretendido. Segue-se a fase de implementação, nova fase de testes e finalmente o go-live: o produto com as alterações passa a ser usado pelo cliente pela primeira vez.

As fases seguintes, tal como é mostrado no gráfico que se segue (Imagem 1), passam pelo desenho funcional e técnico destas modificações; assim como pela passagem de conhecimento e acompanhamento de uma equipa externa, que será responsável pelos desenvolvimentos definidos nos desenhos funcional e técnico.

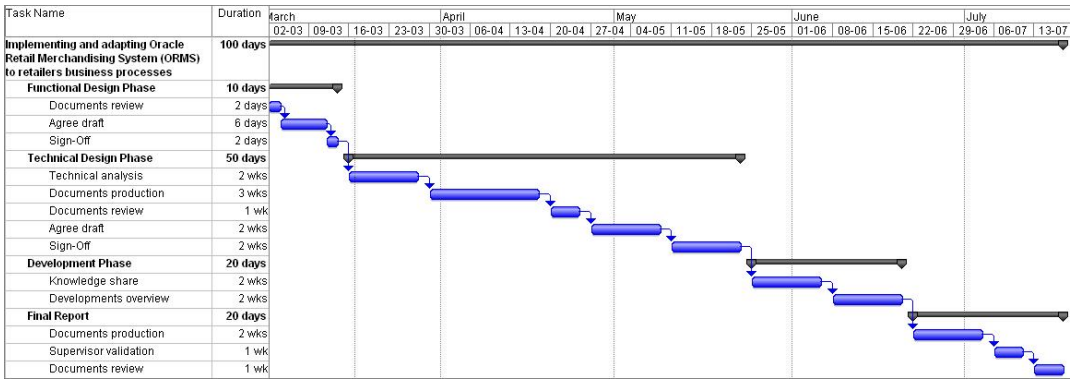


Imagem 1: Plano de Actividades

1.2 Motivação e Objectivos

O negócio do retalho é muito complexo e alargado, desde o planeamento até à execução das tarefas muitas pessoas, decisões e aplicações estão envolvidas. A motivação para este projecto é obtida da possibilidade de aprender mais sobre este negócio complexo, que é o mundo do retalho.

O projecto decorreu no âmbito de um projecto global de implementação de um ERP para suporte aos processos críticos de um grande retalhista, sendo a participação do autor focada no módulo de Merchandising que engloba as actividades básicas de gestão de produto, incluindo toda a parametrização do produto no sistema, o reaprovisionamento deste relativamente a lojas e armazéns, compras, gestão de stock, gestão de custos/preços e gestão de fornecedores, num ambiente global e através de múltiplos canais de distribuição.

No final do projecto o aluno adquiriu conhecimentos de negócio sobre alguns dos processos mais críticos da área de retalho: gestão de categorias, gestão de produto, negociação e compras, gestão de inventário, gestão de canais de distribuição e definição de preços de venda. Além disso, o autor adquiriu conhecimento das metodologias associadas às fases de um projecto de delivery (levantamento de requisitos, análise funcional, desenho técnico, desenvolvimento, testes e implementação) e fortes conhecimentos na ferramenta e tecnologias utilizadas.

1.3 Estrutura da Dissertação

Para além da introdução, esta dissertação contém mais 4 capítulos. No capítulo 2 é descrito o ambiente em que o projecto decorreu, são apresentados a Wipro Retail e o mundo do Retalho. Neste capítulo é também apresentada a história dos sistemas de informação, a importância destes sistemas no mundo de retalho e finalmente é apresentado o sistema de informação que foi alvo de adaptações neste projecto. No capítulo 2 é ainda apresentada a metodologia usada nas customizações e implementações da Wipro Retail. No capítulo 3 são apresentadas as funcionalidades base do sistema, as razões que motivaram o retalhista a efectuar alterações no sistema base, e a solução encontrada. A solução apresentada no capítulo 3 é detalhada no capítulo 4 tanto a nível funcional como técnico. No capítulo 5, o último capítulo, são apresentadas conclusões do trabalho efectuado, e é abordado o tema de trabalho futuro. Depois dos capítulos encontram-se os anexos.

2 Revisão Bibliográfica

Neste capítulo 2 é apresentado o ambiente em que o projecto decorreu, são apresentados a Wipro Retail e o mundo do Retalho. Neste capítulo é também apresentada a história dos sistemas de informação, a importância destes sistemas no mundo de retalho e finalmente é apresentado o sistema de informação que foi alvo de adaptações neste projecto. No capítulo 2 é ainda apresentada a metodologia usada nas customizações e implementações da Wipro Retail.

2.1 O Retalho e a Wipro Retail

O Retalho e a Wipro Retail andam de mãos dadas, num mercado cada vez mais competitivo. Em seguida vão ser descritos em mais detalhe, tanto o mundo do Retalho, como a empresa que é a Wipro Retail.

2.1.1 O Negócio do Retalho

Genericamente, reduz-se o sucesso de uma empresa de retalho a um conceito extremamente simplista: comprar bem para poder vender bem. Isso é lógico, mas não é tudo. O Retalho é uma arte das mais complexas e sofisticadas do marketing, envolve merchandising, “vitrinismo”, promoção e uma infinidade de técnicas que levam a uma diferença significativa no desempenho de uma loja.

O Retalho consiste basicamente em entender as necessidades dos consumidores e superar suas expectativas de forma melhor que a concorrência. Pode-se dizer que as expectativas básicas do consumidor podem ser descritas como [Ref 2]:

- Qualidade;
- Preços competitivos;
- Localização conveniente;
- Estacionamento adequado;
- Horário de funcionamento razoável;
- Loja com um aspecto agradável;
- Vendedores com formação.

Os mercados são caracterizados por uma grande competitividade, e consequente concorrência cada vez mais intensas. A competitividade dos mercados deriva da cada vez maior exigência dos clientes, das quais se podem destacar [Ref 2]:

- Um leque mais variado de produtos para escolha;
- Preços baixos;
- Abertura ao fim de semana;
- Especialistas no atendimento a cliente;
- Menor tempo de espera para ser atendido;
- Maior número de modalidades de pagamentos;
- Maior número de serviços disponíveis;
- Lay-out e design interessante;
- Qualidade do serviço pós-venda.

O sucesso de um retalhista depende em grande parte da sua capacidade negocial para com os fornecedores e da capacidade de adaptação às exigências sazonais dos seus clientes.

O diagrama seguinte demonstra a posição do retalhista na cadeia que vai desde o fabricante até ao cliente.

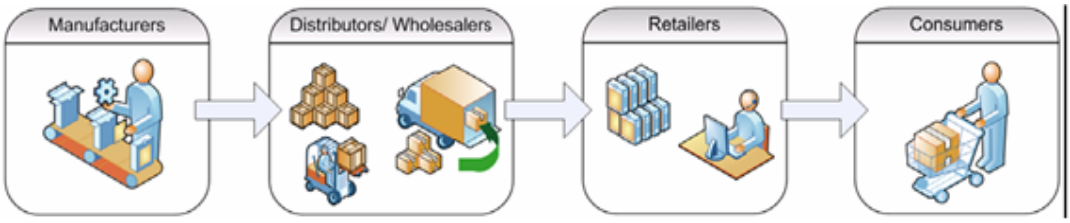


Imagem 2: Cadeia de Retalho

Os retalhistas procuram fidelizar os seus clientes assegurando a satisfação das exigências acima enumeradas, ao mesmo tempo que procuram maximizar o retorno do seu investimento através da diminuição dos custos operacionais ou do aumento do poder negocial face aos fornecedores.

2.1.2 A Wipro Retail

A Wipro Retail, divisão da Wipro Technologies, é uma empresa multinacional na área das tecnologias de informação (IT) que oferece serviços de elevado valor acrescentado e rápido retorno do investimento aos retalhistas mais prestigiados a nível mundial. A sua reputação assenta na prestação consistente de um serviço de alta qualidade e num sólido historial de excelentes resultados.

A Enabler, agora Wipro Retail, foi fundada em 1997, a partir da autonomização da Direcção de Sistemas de Informação da Sonae Distribuição, cuja actividade e experiência na concepção e desenvolvimento de Sistemas de Informação para a Modelo Continente proporcionou uma sólida base de conhecimento dos processos e sistemas de retalho [Ref 1].

A empresa cresceu rapidamente e conta, actualmente, com escritórios em Portugal, Reino Unido, Alemanha, Itália, Espanha, França e Brasil, e com clientes em vários países da Europa e cada vez mais na América do Norte, América Latina e Ásia.

Em Junho de 2006, a empresa foi adquirida pela Wipro, uma aquisição que veio reforçar o portfolio de serviços de retalho da multinacional indiana e que permite aos clientes da Wipro Retail beneficiarem da dimensão da Wipro.

Conhecimento, inovação e pragmatismo são as palavras que mais caracterizam a Wipro Retail. O profundo conhecimento do negócio proporciona uma compreensão única sobre os factores de motivação empresarial e as formas como se podem combinar a tecnologia e o processo de negócio, de modo a melhorar o rendimento comercial. É uma empresa dedicada a ajudar empresas de retalho a optimizar os processos dos sistemas de informação, convertendo assim complexas infra-estruturas em ferramentas de apoio ao negócio.

A reputação da empresa baseia-se, portanto, num serviço de alta qualidade e um acompanhamento constante, que reflecte os excelentes resultados até agora obtidos.

A Wipro Retail oferece aos seus clientes serviços ao nível da integração de sistemas, desenvolvimento de soluções, implementação e suporte, baseados nas melhores soluções existentes no mercado. Neste sentido, assegura ferramentas de suporte ao negócio para as seguintes áreas [Ref 1]:

- Sistemas Operacionais (Gestão de Mercadorias, Vendas e Gestão de Armazém, Operações de Back Office, etc.);
- Gestão de Sistemas de Informação (Managament Information Systems);
- As soluções MIS da Enabler – Wipro são baseadas em dados multi-dimensionais e OLAP permitindo aos retalhistas chegar aos dados que

necessitam e procurar informação no formato que melhor satisfaz os seus requisitos;

- Business Intelligence – Ajuda os retalhistas a melhorar o seu desempenho e eficiência na sua tomada de decisão;
- Digital Supply Chain (Cadeia de Fornecimento Digital) – Permite aos retalhistas partilhar uma única comunicação com os seus clientes, fornecedores e parceiros;
- Sistemas Integrados – Cobre os seguintes requisitos: operações, suporte à decisão, Business Intelligence e soluções de interacção. Para integrar todos estes componentes a Enabler - Wipro fornece tecnologias de integração – Enterprise Application Integration (EAI).

Da sua lista de clientes constam os nomes de alguns dos retalhistas e grossistas mais reconhecidos a nível internacional, nomeadamente Tesco, AVA, Modelo Continente, Nisa Today's, Esprit, Despar, Renner, Galeries Lafayette, Sabeco, Fortress, Vetura, Dubai Duty Free e Miquel. Com eles desenvolvem operações em diversos países europeus e continuam a fortalecer a sua presença na América do Norte, América Latina e Ásia Pacífico.

O seu know-how e experiência na área das tecnologias de informação para o retalho são uma ajuda fundamental para os retalhistas, nomeadamente na redução do risco em projectos críticos para o seu negócio.

Detentores de uma certificação CMM Level 5, as suas metodologias de desenvolvimento representam o estado da arte na engenharia de software.

2.2 Os Sistemas de Informação

Um ERP (Enterprise Resource Planning) é um sistema de gestão integrada que surgiu da necessidade de optimização/evolução do Material Resource Planning (MRP) e Manufacturing Resource Planning (MRP-II) [Ref 3].

O MRP foi desenvolvido na década de 60 e era utilizado para gerir stocks de materiais através da planificação de ordens de compra e de ordens de fabrico. Na década de 70 foi introduzido o MRP II que veio incorporar no MRP outras funções prioritárias na conclusão de processos de produção [Ref 3].

O conceito de ERP surgiu na década de 90 quando a palavra-chave passou a ser integração. Este é considerado, por alguns autores, o nível mais avançado dos sistemas tradicionalmente chamados de MRP II [Ref 4]. Incorpora além das funções anteriormente contempladas, funcionalidades relacionadas com finanças, custos, vendas, gestão de recursos humanos, entre outras, que anteriormente ao

aparecimento deste conceito eram controladas através de inúmeros sistemas díspares e não integrados.

Actualmente, é sabido que a Informação, bem como a sua correcta e eficiente gestão, são requisitos essenciais a qualquer negócio. Assim, cria-se a necessidade de existirem, no seio de uma organização, sistemas capazes de realizar tarefas de recolha, tratamento e disponibilização da informação, de forma correcta e eficiente. Se por um lado é verdade que muita desta informação é vital para o funcionamento e para a sobrevivência da organização, também não deixa de ser verdade que informação em timings errados e/ou o excesso de informação resultante da má filtragem dos dados, ou mesmo da falta dessa mesma filtragem, podem ser nocivas para a organização, influenciando negativamente o processo de tomada de decisões.

A grande diferença entre um ERP e um sistema de suporte à gestão tradicional reside no facto do ERP desenvolver mais as possibilidades de integração e coerência na produção de informação. Enquanto que nos sistemas tradicionais as componente informáticas dos diferentes departamentos da empresa normalmente trabalhava de forma isolada, nas aplicações ERP os vários departamentos organizacionais encontram-se integrados, partilhando a mesma interface com o utilizador e uma única base de dados. A integração de todo um sistema de informação de uma empresa num software ERP visa reduzir custos, melhorar os procedimentos de negócio, aumentar a eficácia, melhorar a troca de informação em ambientes distribuídos e, principalmente, facilitar o processo de tomada de decisões [Ref 5].

2.2.1 Os Sistemas de Informação no Retalho

Um ERP para o retalho deverá ser capaz de satisfazer as crescentes exigências em termos de processamento de informação, e oferecer soluções que forneçam um suporte eficiente à realização de todas as operações essenciais ao retalhista. Entre estas funções encontram-se: gestão de entrepostos, gestão de fornecedores (controlo do processo de ordens de compra, contratos de fornecimento, gestão de custos), gestão logística (gestão de espaço, distribuição), gestão de lojas, reaprovisionamento, planeamento de capacidades (modelos previsionais), gestão de preços e promoções e gestão de artigos. Encontram-se também funções de planeamento financeiro (orçamentação), planeamento de gama (sortido de artigos por loja) e planeamento de colecções (importante para vestuário e calçado) [Ref 6].

Um ERP para o retalho deverá ajudar no processo de tomada de decisão fornecendo ferramentas capazes de realizar previsões, que permitam identificar, por exemplo, que produtos deverão ser colocados à venda, em que locais e qual o preço a que deverão ser disponibilizados. Estas ferramentas de previsão permitem a definição de políticas comerciais centradas no cliente ao mesmo tempo que possibilitam a

optimização dos níveis de inventários através da definição mais precisa e eficiente das operações de reaprovisionamento, reduzindo custos de armazenamento [Ref 6].

Para maximizar a eficiência de um retalhista, um ERP para o retalho deve permitir a manipulação da informação com um máximo de precisão. Esta precisão é encontrada manipulando informação ao nível das combinações artigo/loja. Este nível de precisão é essencial, pois possibilita o controlo eficiente de todo o negócio e também a personalização do serviço prestado aos clientes finais, uma vez que torna possível a definição de políticas comerciais específicas ao contexto em que o produto é comercializado. Entre estas políticas encontram-se a gestão de preços e a criação de promoções ao nível de cada loja, ou ao nível de zonas [Ref 6].

Um ERP para o retalho tem ainda de possuir mecanismos que lhe permitam lidar com a complexidade associada aos artigos. Um exemplo desta complexidade é a constituição de packs através da junção de quantidades de um ou vários componentes. Para estes artigos é necessário manter informação acerca de preços, inventários ou outros atributos simultaneamente para o próprio artigo pack como para os diferentes componentes, uma vez que estes últimos poderão ser vendidos separadamente. Ainda em relação aos artigos, um ERP para retalho deve possuir mecanismos que permitam reduzir o risco associado a algumas características enumeradas anteriormente, como a sazonalidade ou os reduzidos ciclos de vida. As ferramentas de previsão acima referenciadas, constituem um exemplo de mecanismos que permitem lidar com esta complexidade, ao produzirem estimativas acerca da futura procura dos artigos, o que fornece um ponto de partida para as decisões relacionadas com as compras a fornecedores, reduzindo o risco associado às suas características [Ref 6].

As ferramentas de suporte às operações de loja, os POS's (Point of Sale), são outra característica essencial de um ERP de retalho. A sua utilização é essencial para as superfícies comerciais e a sua integração num ERP possibilita a gestão em tempo real dos inventários [Ref 6].

Também muito importante para o mercado de retalho é a utilização de novas tecnologias de comunicação como meio de contacto quer com fornecedores como com os próprios clientes. A utilização de tecnologias como o EDI (Electronic Data Interchange) permite agilizar o processo de comunicação com os fornecedores enquanto a utilização da Internet como ponto de venda permite reduzir os custos operacionais e personalizar ainda mais o serviço de atendimento aos clientes [Ref 6].

Estas são algumas das características mais relevantes que um ERP para o retalho deverá possuir. De uma maneira genérica, pode dizer-se que a aplicação dos sistemas ERP ao mercado do retalho deverá traduzir-se na redução de inventários, agilização do processo de interacção com fornecedores e clientes e num controlo integrado sobre todos os elos da cadeia de valor [Ref 6].

2.2.2 Oracle Retail Merchandising System (ORMS)

O ORMS (Oracle Retail Merchandising System) é a base do sistema integrado de gestão que permite o armazenamento e controlo de toda a informação necessária e vital para o funcionamento do negócio de um retalhista. O ORMS serve como repositório central onde toda a informação fica armazenada assegurando assim a integridade entre todos os diferentes sistemas da organização que estarão ligados ao ORMS. Devido ao facto de ser o sistema que gere toda a informação, é o ORMS que suporta as funções chave do negócio do retalho, entre as quais se encontram a gestão de inventário, compras, estabelecimento de preços e custos, gestão de promoções e reaprovisionamento.

Este sistema permite aceder rápida e facilmente a toda a informação crucial para o dia a dia das actividades do retalho e potencia o alcance das metas de vendas e de resultados financeiros. O ORMS acompanha os processos de negócio, tornando-os assim mais eficientes, e permite harmonizar os sistemas informáticos de suporte ao negócio do retalho, tudo com o objectivo de melhorar o serviço prestado aos clientes.

Em termos técnicos, o ORMS é suportado pela plataforma Oracle e funciona numa arquitectura de três camadas, tal como se encontra representado na Figura seguinte – Arquitectura do ORMS.

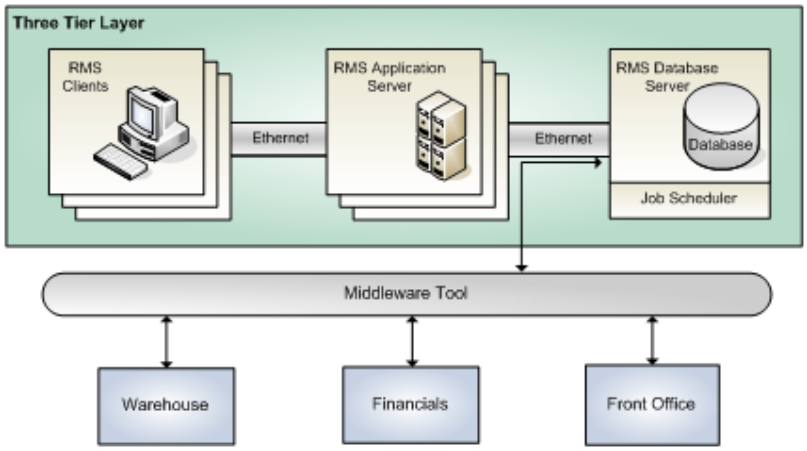


Imagem 3: Arquitectura do ORMS

O RMS é constituído por uma base de dados Oracle, na qual é armazenada toda a informação, mas também muita da lógica de negócio, através de código procedimental desenvolvido em PL/SQL, as chamadas *stored procedures* (conjunto de comandos SQL que podem ser armazenados no servidor; uma vez que isto tenha sido feito, os clientes não precisam de reenviar os comandos individuais mas pode fazer

referência às *stored procedures* aumentando assim a eficiência). O utilizador acede à aplicação através de um *browser*, e realiza as operações através de ecrãs desenvolvidos em Oracle Forms.

Devido ao grande volume de informação que este sistema tende a processar, muitas das operações de suporte ao negócio são realizadas através de um sistema batch. Este sistema *batch* consiste num conjunto de programas desenvolvidos em Pro*C, que correm autonomamente em fases previamente planeadas (existem várias fases, cada uma com uma sequência lógica de comandos). Encontram-se no processamento *batch* as tarefas que implicam a realização de transacções que envolvam acessos maciços à base de dados e cuja realização em concorrência com a normal utilização do sistema por parte dos utilizadores seria impossível. Como exemplos de operações executadas durante o processamento *batch* encontram-se alterações de preços, consolidação de dados históricos, reaprovisionamento automático e limpeza (purga) de dados antigos.

2.3 Metodologia Wipro Retail

O serviço de integração de soluções da Wipro Retail baseia-se no desenvolvimento de soluções de negócio para um cliente final a partir de soluções existentes e das quais a Wipro Retail tem um conhecimento profundo, ou a partir de software de base (tipo middleware). A Wipro Retail possui uma metodologia sobre *best practices* para a prestação deste tipo de serviços. Esta metodologia consiste em sete fases: identificação de requisitos, análise, especificação, construção, documentação, implementação e estabilização, tal como se pode verificar na Imagem 4 – Metodologia de Integração de Soluções da Wipro Retail.

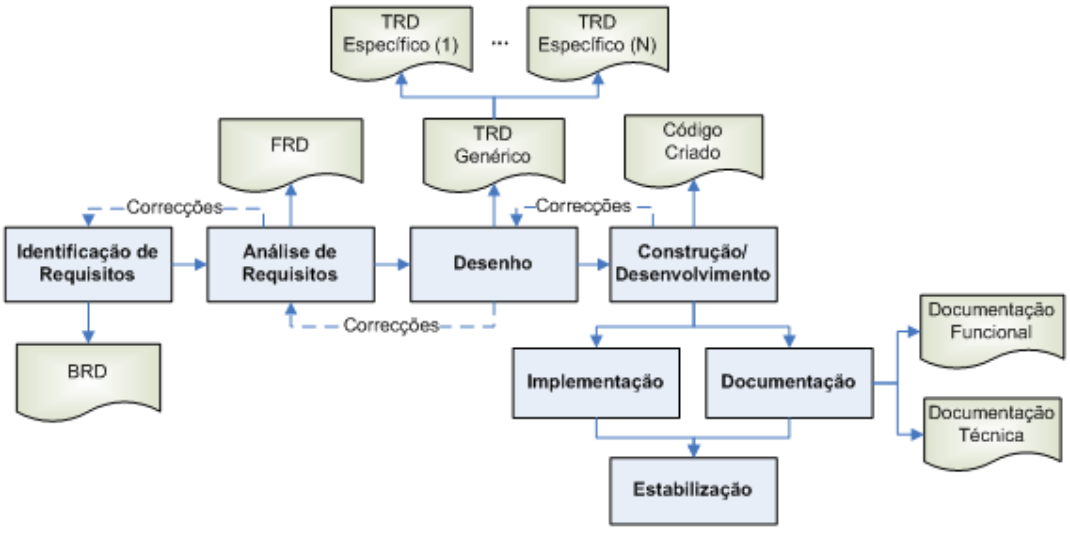


Imagem 4: Metodologia de Integração de Soluções

2.3.1 Identificação de requisitos

A identificação de requisitos tem como objectivo a produção do BRD (Business Requirement Document). Na fase de produção do BRD a interacção com os analistas de negócio do cliente final é essencial. Reuniões periódicas permitirão um alinhamento de conhecimento e expectativas durante a fase de escrita, facilitando uma assinatura rápida do documento por todos os envolvidos. O objectivo nesta fase é conseguir obter todas as necessidades do cliente em relação ao seu processo de negócio, e com isso validar se todas essas necessidades estão contempladas no sistema a implementar. Todas as falhas existentes entre o processo de negócio do cliente e a aplicação a implementar são registadas como novos requisitos que serão alvo de análise futura. A enumeração clara destes requisitos é essencial para permitir o rastreio dos mesmos ao longo de todo o projecto. A assinatura deste documento garante o entendimento e o comprometimento de todos os envolvidos relativamente ao seu conteúdo, sendo assim uma base sólida e fechada para as fases seguintes do projecto.

2.3.2 Análise de Requisitos

A análise inicia-se utilizando o documento BRD como input. O objectivo será a produção do FRD (Functional Requirement Document) que concretiza em termos funcionais o BRD. Todos os requisitos levantados na produção do BRD são analisados mais pormenorizadamente, de forma a verificar se de facto são requisitos válidos ou podem ser ultrapassados através de soluções alternativas. Para todos aqueles que são válidos, então é feito para cada requisito um FRD que explica em termos funcionais como é que o requisito levantado vai funcionar no sistema. Os envolvidos na produção do FRD devem fazer uma leitura cuidadosa e detalhada do BRD de forma a adquirirem um conhecimento profundo das necessidades do projecto. São feitas reuniões regulares (menos frequentes do que na fase anterior) com os analistas de negócio, aqui com um número mais restrito de pessoas (normalmente pessoas que têm um conhecimento mais profundo da área de negócio a que o requisito em questão se refere) e com os analistas tecnológicos do cliente para permitir um alinhamento de conhecimento e expectativas durante a fase de escrita, facilitando uma assinatura rápida do documento por todos os envolvidos. Neste documento deve ser feita uma enumeração dos requisitos funcionais que permita uma ligação clara aos requisitos do BRD e facilite o rastreio dos mesmos ao longo do projecto. A assinatura deste documento garante o entendimento e o comprometimento de todos os envolvidos relativamente ao seu conteúdo, sendo assim uma base sólida e fechada para as fases seguintes do projecto.

2.3.3 Especificação

A especificação inicia-se utilizando os documentos BRD e FRD como input. O objectivo será a produção de um TRD (Technical Requirement Document) genérico e dos vários TRD específicos que concretizam em termos técnicos o FRD. Os gestores de projecto devem decidir a divisão mais eficaz, de acordo com as equipas, para produção dos diferentes TRD específicos. Esta divisão poderá ser por processos, funcionalidades, áreas tecnológicas ou outra. Devem também ser especificados os testes unitários mais específicos a executar na fase de construção. Os envolvidos na produção dos TRD devem fazer uma leitura cuidadosa e detalhada do BRD e do FRD de forma a adquirirem um conhecimento profundo das necessidades globais do projecto. Devem ser feitas reuniões pontuais com os analistas de negócio e com os analistas tecnológicos do cliente para permitir um alinhamento de conhecimento e expectativas durante a fase de escrita, facilitando uma assinatura rápida do documento por todos os envolvidos. Neste documento deve ser feita uma enumeração dos requisitos técnicos que permita uma ligação clara aos requisitos do BRD, FRD e facilite o rastreio dos mesmos ao longo do projecto. A assinatura destes documentos deve garantir o entendimento e o comprometimento de todos os envolvidos relativamente ao seu conteúdo, sendo assim uma base sólida e fechada para as fases seguintes do projecto.

2.3.4 Construção/Desenvolvimento

A fase de construção inicia-se utilizando o documento TRD genérico e TRD's específicos. Os gestores de projecto devem decidir a divisão mais eficaz do trabalho de desenvolvimento de acordo com as equipas. Os diferentes TRD produzidos constituirão uma das bases lógicas para esta divisão. Os envolvidos na produção do código devem fazer uma leitura cuidadosa e detalhada dos TRD e ler também o BRD e o FRD de forma a adquirirem um conhecimento profundo das necessidades globais do projecto. O desenvolvimento do código deve obedecer aos Princípios de Desenvolvimento de Software da Wipro Retail. Nestes Princípios de Desenvolvimento, podemos encontrar uma definição clara da estrutura do ambiente de desenvolvimento e das regras de codificação específicas do projecto, que permitam garantir a fácil localização e uniformidade do código produzido. Estas regras estão documentadas e devem ser do conhecimento de toda a equipa. Devem existir, a partir do 1º dia de desenvolvimento, regras claras para controlo de versões do código produzido. Deverão ser executados testes unitários exaustivos pelos elementos da equipa de desenvolvimento, nomeadamente os testes mais específicos definidos na fase de Especificação em documentos próprios. Nestes documentos devem ser registados os resultados da execução destes testes. No final desta fase deverá existir na ferramenta

de gestão de versões em uso na Wipro Retail o código fonte que dará origem aos componentes de software que compõem a solução final a entregar à equipa de testes ou ao cliente final para testes.

2.3.5 Documentação

A documentação inicia-se utilizando os documentos FRD, TRD genérico e TRD específicos. O objectivo será a produção da documentação de utilizador e documentação para o suporte. A documentação para utilizadores tem de ser clara e objectiva. Deve focar-se particularmente nas diferentes componentes funcionais da aplicação. Sempre que for considerado relevante, deve ser feito um enquadramento da funcionalidade com os processos em que está inserida para melhorar a percepção global dos impactos da execução da funcionalidade. A documentação técnica deve também ser clara e objectiva. Tem de descrever com detalhe todos os processos, componentes e tarefas relevantes para a função de suporte. O objectivo é que este documento seja a base principal de trabalho das equipas dos serviços profissionais. O objectivo final é os serviços profissionais serem capazes de executar a sua função com qualidade sem intervenção do desenvolvimento.

2.3.6 Implementação

A implementação tem por objectivo a colocação em produção do sistema aplicativo desenvolvido para o cliente. A equipa envolvida deve ter particular atenção às características de todo o ambiente de produção, que pode ter algumas diferenças para o de desenvolvimento. Todas as actividades durante este período devem ser executadas com particular cuidado pois estamos a trabalhar sobre um sistema de produção. Todos os aspectos de segurança e execução de comandos relacionados com acessos a bases dados, sistema operativo, escalonadores de trabalhos, processos *batch* e interactivos devem ser abordados com especial cuidado e rigor.

Deve ser produzida documentação de apoio às equipas que vão gerir a infra-estrutura de produção.

No fim deste período de arranque, o sistema deve ficar num estado estável e seguro sendo o acesso limitado às equipas responsáveis pelo suporte aplicativo e pela estabilização da aplicação.

2.3.7 Estabilização

A estabilização tem por objectivo dar garantia que o sistema aplicativo desenvolvido para o cliente se encontra com um correcto funcionamento. Todas as actividades durante este período devem ser executadas com particular cuidado pois estamos a trabalhar sobre um sistema de produção. Todos os aspectos de segurança e execução de comandos relacionados com acessos a bases dados, sistema operativo, escalonadores de trabalhos, processos *batch* e interactivos devem ser abordados com especial cuidado e rigor. O objectivo desta fase é que no fim deste período de estabilização, o sistema tem de ficar num estado estável e seguro sendo o acesso limitado às equipas responsáveis pelo suporte aplicativo. Esta fase deve terminar com o fecho formal do projecto por parte da gestão de projecto, sendo feita a passagem do sistema para as equipas de serviços profissionais terminando o envolvimento da equipa de projecto.

2.4 Conclusões

O tempo em que o retalho se definia por comprar bem e vender melhor acabou. Ao longo do tempo, até aos dias de hoje, o mercado retalhista evoluiu naturalmente e tornou-se num dos mais competitivos mercados na realidade actual.

A constante procura por redução de custos, a melhoria da qualidade de serviço prestada ao cliente, horário de funcionamento, formação do pessoal, por si só já não chegam para se distinguir entre retalhistas.

A Wipro Retail, ex Enabler, entra neste mercado como um apoio ao retalhista para actualização dos seus sistemas de informação e, a partir daí, obter vantagem competitiva sobre outros retalhistas. A antiga Enabler começou por ser um apoio aos sistemas de informação do Modelo/Continente, mas é agora reconhecida internacionalmente, sendo o seu trabalho é sinónimo de qualidade.

Os ERPs são vistos pelos retalhistas como uma actualização necessária aos seus sistemas de informação, de forma a integrar todas as áreas de um processo que é o retalho, num só ponto. Este processo é necessário por si só para garantir a necessária competitividade.

O ORMS é a solução desenvolvida pela Oracle para ser usada pelos grandes retalhistas de forma a controlar o seu processo de negócio. Este sistema permite aceder rápida e facilmente a toda a informação crucial para o dia a dia das actividades do retalho e potencia o alcance das metas de vendas e de resultados financeiros. O ORMS acompanha os processos de negócio, tornando-os assim mais eficientes, e permite harmonizar os sistemas informáticos de suporte ao negócio do retalho, tudo com o objectivo de melhorar o serviço prestado aos clientes.

3 Gestão de Artigos

Será descrito neste capítulo basicamente o porquê das modificações que deram origem a este trabalho. Devido a certas particularidades sobre o seu processo de negócio, o retalhista não ficou completamente satisfeito com a funcionalidade base do sistema a implementar, requisitando portanto alterações no sistema que lhes permite assegurar a continuidade do seu processo de negócio.

Será apresentado neste capítulo quais as funcionalidades base do sistema a implementar, bem como o que motivou o retalhista a pedir as alterações à funcionalidade inicial. Será também apresentada a solução encontrada, que teve em conta não só o pedido do cliente, como também o impacto que a solução teria com toda a restante aplicação. A solução encontrada e proposta, foi a solução que teve a melhor avaliação considerando estes dois pontos.

3.1 Funcionalidade base

O módulo Oracle Retail Merchandising System (ORMS) da Oracle Retail dá ao retalhista a possibilidade de configurar tudo o que diz respeito aos artigos que mantém na sua empresa. É através do ORMS que o retalhista consegue criar/manter os seus artigos, configurando-os da maneira que achar mais conveniente conforme a situação, e conforme a sua finalidade.

Nesta secção é explicado de como funciona a criação e manutenção de artigos no módulo ORMS.

Como já foi referido anteriormente o ORMS é o sistema que gere toda a gama de produtos de uma companhia. É aqui que o cliente cria, define e mantém todos os seus

artigos. Todo este processo de configuração de artigos é crucial dado que só depois desta acção é que o retalhista pode efectuar qualquer tipo de transacção com o artigo, como a compra ou venda deste.

É de acordo com as características de um determinado produto que é definido o principal propósito desse produto. De forma a melhor distinguir os diferentes produtos, estes podem ser separados em diferentes "tipos". Ao criar um novo artigo no ORMS, é apresentado ao retalhista o seguinte leque de tipos de produtos: "**Regular Item**"; "**Pack**", que pode ser simples ou complexo; "Deposit Items"; "Consignment Items"; "Concession Items"; e finalmente "Transformable Items".

Dentro do tipo "**Regular Item**" existe também o conceito de nível do artigo. Um artigo pode ter até 3 níveis:

- O **primeiro nível** é chamado de nível agregador. Os retalhistas usam este nível quando pretendem agrupar um conjunto de items com as características básicas iguais dentro da mesma família. Normalmente a um artigo deste nível é chamado a artigo pai, pois todos os artigos criados a um nível abaixo deste, terão sempre as suas características herdadas deste artigo agregador. Isto facilita em muito algumas actividades que o retalhista tem de fazer, onde pretende agrupar uma quantidade de artigos com características semelhantes para efectuar certas acções;
- No **segundo nível** estariam presentes todas as variantes do artigo. Os artigos a este nível têm as mesmas características base que o pai, no entanto o retalhista pode a este nível alterar outras características que lhe vai permitir diferenciar os diferentes artigos a este nível (esta hierarquia é frequentemente usada nos retalhistas do mundo da moda, pois esta funcionalidade apresenta uma forma extremamente fácil e intuitiva de cadastrar diferentes artigos em que a única diferença é o tamanho ou a cor, por exemplo, t-shirts que podem ter várias cores e vários tamanhos);
- No **terceiro nível** estariam os códigos de barras associados a cada um dos artigos de nível 2.

Na imagem 5 pode-se observar um exemplo da estrutura de um artigo com 3 níveis.

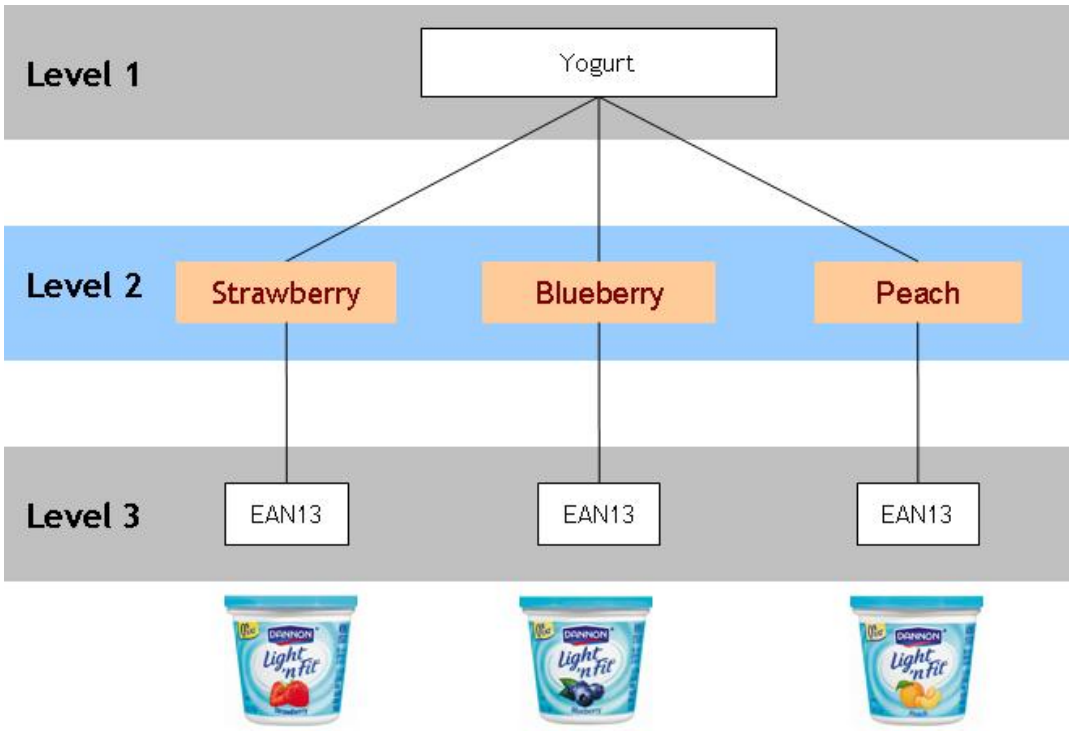


Imagem 5: Artigo com 3 níveis

Artigos com 1 (um artigo sem código de barras associado) ou 2 (artigo mais os respectivos códigos de barras) níveis são chamados de artigos simples, artigos com 3 níveis (artigos em que têm um elemento agregador) são chamados de artigos complexos.

Os **Simple Pack** têm normalmente 2 níveis (no primeiro nível seria definido o artigo em si, e no segundo nível os códigos de barras associados ao primeiro nível). Um Simple Pack é um tipo de artigo cuja função é de agrupar um número variável de um "Regular Item". Em situações em que o mesmo artigo pode ser vendido separadamente ou num pack, são criados este tipo de artigos no sistema. Assim o retalhista pode controlar não só quantas unidades vende de um determinado produto, mas também quantas unidades vende em separado e num pack. As habituais embalagens de coca-cola são um bom exemplo de um "Simple Pack", já que o cliente pode comprar a embalagem como um todo, ou então comprar apenas uma garrafa isolada. O retalhista pode criar quantos packs quiser para um determinado "Regular Item", com a quantidade que desejar. A única limitação que o retalhista tem ao fazer um pack deste tipo é que este só pode ser constituído por apenas um "Regular Item". Ou seja ao criar o simple pack de coca-cola, o retalhista só pode ter dentro do pack garrafas de coca-cola e não de outra bebida, ou qualquer outro artigo.

Os **Complex Pack** são em tudo igual aos Simple Pack, excepto que podem ser constituídos por múltiplos Regular Item, e cada um destes em quantidades variáveis. Neste caso o retalhista já tem a possibilidade, se assim o desejar, de misturar

diferentes tipos de artigos numa mesma embalagem. Pegando no exemplo anterior, seria possível ao retalhista criar um pack com garrafas de coca-cola e garrafas de Pepsi, por exemplo.

Apesar do ORMS permitir a criação de outros tipos de Items, estes não se encontram no âmbito das modificações feitas ao sistema base, sendo por isso desnecessária a sua descrição detalhada.

Para alem da definição base dos atributos de um artigo, o ORMS permite também fazer a associação dos artigos tanto a fornecedores (o que vai permitir ao retalhista encomendar o artigo) como a lojas (definindo assim as lojas onde o artigo vai ser vendido). Existem também atributos relacionados com estas ligações que o retalhista terá também de parametrizar no sistema para que este funcione correctamente

3.2 Requisitos do Cliente

Como já foi referido anteriormente, durante o processo de CRP são levantados requisitos que têm a ver com necessidades do cliente que o sistema não consegue acomodar. Nos próximos capítulos serão apresentados alguns requisitos levantados durante a primeira fase do projecto, requisitos esses que depois se tornaram alterações ao sistema de base.

3.2.1 Requisito #39:

Para melhor comparar preços entre dois produtos semelhantes, é normal o cliente utilizar para essa comparação o preço à unidade e não o preço geral do produto. O preço à unidade permite ao cliente ter um melhor termo de comparação em relação a dois ou mais produtos. Através deste valor o cliente consegue ter a ideia exacta de quanto está a gastar por unidade. Por exemplo: uma garrafa de Sumol de 1,5l custa 3€ e uma lata de Sumol (33cl) custa 0.70€; comparar directamente é impossível, assim se fizermos o cálculo de cada um dos artigos para a mesma unidade (1l) ficamos a saber que na garrafa estamos a pagar 2€/l e na lata 2,10€/l.

Apesar da aplicação de base acomodar já o conceito de preço por unidade, não satisfaz plenamente todas as necessidades desejadas pelo retalhista, pelo que foi levantado o seguinte requisito para que a funcionalidade passasse a funcionar mais de acordo com as necessidades deste:

Requisito #39:

O preço à unidade de um produto precisa de ser mantido no sistema a nível central.

Como se pode ver, o requisito na altura do seu levantamento é apenas uma ideia. É fruto de uma primeira discussão com o cliente sobre as potencialidades do sistema a implementar e a forma como este gere o seu negócio. Normalmente nesta primeira fase não é possível chegar a um nível de detalhe muito elevado. Há que registar apenas que nesta área existe uma potencial falha no sistema que não permite o retalhista trabalhar como pretende (processos de negócio). Futuramente durante a análise funcional, e após várias reuniões com o retalhista, o requisito previamente referido foi reformulado, tal como se mostra a seguir.

Os requisitos do negócio para o sistema relacionados com esta área são os seguintes:

- Permitir diferentes unidades de medida para unidades e preço;
- Definir os valores do preço à unidade ao nível do artigo e não do artigo/loja (como existe no sistema base);
- Visualizar o valor do preço à unidade tal como vai ser enviado para as lojas;
- Forçar todas as lojas a terem o preço à unidade definido ao nível do artigo;
- Permitir definir por defeito uma unidade de medida para todos os artigos que pertencem a uma sub-classe (a sub-classe é o nível mais baixo da hierarquia mercadológica utilizada no ORMS para organizar os artigos criados);
- Permitir ao retalhista adicionar novas unidades de medida no futuro.

De modo a cumprir com o pedido do retalhista relacionado com o preço à unidade, e replicar os processos de negócio que o retalhista possui actualmente, foi decidido que o modo como o preço à unidade é definido no ORMS base tinha de ser alterado. No ORMS base, o preço à unidade é definido de um modo diferente: se por um lado tinha maior liberdade por permitir ter diferentes valores de UEP (Unit Equivalent Price) de acordo com a combinação artigo/loja (o retalhista pretende que o preço por unidade seja uma característica ao nível do artigo e não ao nível do artigo/loja), por outro obrigava a que a unidade de medida da quantidade do artigo fosse o mesmo que o valor do UEP.

Actualmente, o retalhista usa um conjunto de códigos que estão associados às unidades de medida a serem usadas para calcular o valor do preço à unidade do artigo. Esta lógica vai ser replicada e implementadas no ORMS. Para esse efeito, os seguintes campos terão de ser criados para que o preço à unidade seja definido da forma pretendida pelo retalhista:

- **UEP (Unit Equivalent Price) Code**: este campo vai determinar automaticamente quais vão ser as unidades de medida associadas. Por exemplo:
    - J – quantidade do artigo inserida em ml e o preço à unidade calculado e mostrado para 100ml;
    - D – quantidade do artigo inserida em cm e o preço à unidade calculado e mostrado em metros.
- **Net Content**: quantidade do artigo.

No exemplo apresentado no início desta secção (exemplo do Sumol), o Net Content da garrafa de Sumol seria 1,5L e da lata seria 33cl. O valor do UEP para a garrafa seria L (pois a quantidade está em litros e queremos mostrar o UEP ao litro) e para a lata seria C. (A lista com os valores possíveis para o UEP Code é apresentada na Imagem 6.

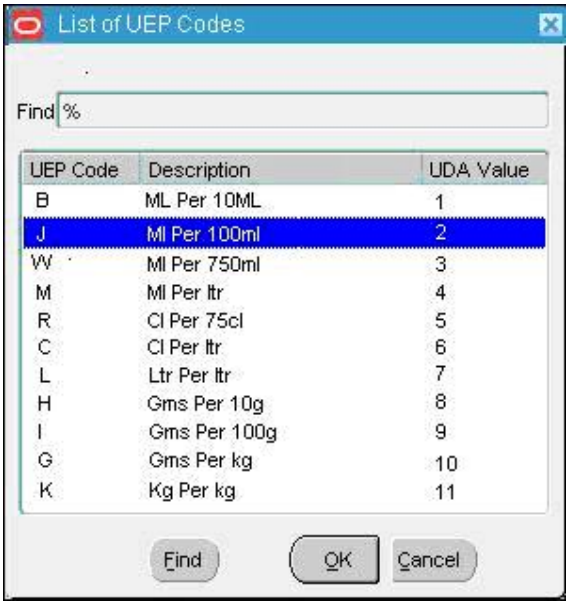


Imagem 6: Lista de Códigos UEP

Ao criar a sua estrutura mercadológica no sistema, o retalhista vai ter a possibilidade de atribuir um valor do "UEP code" ao nível da sub-classe. Este valor vai ser o valor por defeito para todos os artigos criados dentro desta sub-classe. No entanto, o retalhista terá sempre a possibilidade de alterar este valor ao nível do artigo se assim o entender.

Outra diferença para as funcionalidades base do ORMS é que o preço à unidade vai deixar de ser definido ao nível do artigo e loja e vai passar a ser calculado apenas ao nível do artigo (mesmo valores para todas as lojas).

3.2.2 Requisito #40:

É comum em determinados tipos de produtos a inclusão do preço de venda na própria embalagem do artigo, por parte dos fornecedores. Estes artigos passam a ser referidos como artigos com o preço marcado.

Dado que o sistema a ser implementado não contempla este tipo de artigos, o retalhista levantou o seguinte requisito:

> Requisito #40:
>
> Para artigos com o preço marcado no produto são necessárias as seguintes funcionalidades no sistema:
>
> 1 – Identificar esse tipo de produtos: Preço Marcado e Preço Embebido;
>
> 2 – Guardar o preço marcado;
>
> 3 – Não permitir que o preço de venda seja superior ao preço marcado.

O último ponto deste requisito não diz respeito ao ORMS. Este ponto está relacionado com outra aplicação da Oracle Retail, o Oracle Retail Price Management (ORPM), cuja função é a gestão de preços e promoções dos artigos, logo este últmo ponto não se encontra no âmbito da modificação aqui apresentada.

O ORMS vai guardar e manter um indicador de produto com preço marcado e o valor marcado. Como estes dois campos não existem no ORMS base, estes vão ter de ser criados e a aplicação terá de ser modificada de forma a tratar destes valores.

3.2.3 Requisito #70:

Com a evolução das tecnologias, e a maior flexibilidade dos códigos de barras, o preço, para certos produtos, passou a ser incluído no código de barra do produto. A estes produtos chamam-se artigos com o preço embebido.

Mais uma vez, a aplicação base não permite o tratamento deste tipo de artigos, que são frequentemente usados por este retalhista. Logo, a necessidade fez com que o seguinte requisito fosse levantado:

Requisito #70:

O requisito para os artigos com preço embebido no código de barras (barcode) é:

1 – Identificar estes artigos tanto no ORMS como no ORPM.

O ORMS terá de guardar e manter um indicador que será responsável por identificar se o artigo tem ou não o preço embebido no código de barras. Como este campo não está disponível na versão base do ORMS, então terá de ser criado e a aplicação terá de ser modificada de forma a tratar deste valor. Esta modificação está relacionada com o requisito #40 de forma que um artigo ou é regular, ou é do tipo definido no requisito #40 (artigo com preço marcado na embalagem) ou no requisito #70 (artigo com o peço embebido no código de barras). O retalhista terá, ao definir o artigo, de optar por uma destas três opções.

Para além destas alterações, foi identificado que o código de barras usado pelo retalhista nestes casos é um tipo que não se encontra no ORMS. Será portanto também necessário a inclusão deste novo tipo de código de barras no ORMS

3.2.4 Requisito #211:

Outro requisito registado durante a primeira fase do projecto, depois de ser apresentado ao retalhista quais os atributos que este tinha à sua disposição para definir um artigo, foi o seguinte:

Requisito #211:

Alguns atributos do artigo presentes nos sistemas actuais do retalhista têm de ser mantidos no futuro.

O retalhista considerou crucial a manutenção de alguns dos atributos que caracterizam no sistema actual os seus artigos. Tiveram de ser incluídos no ORMS todos os atributos existentes no sistema actual que não conseguiram ser mapeados para atributos existentes no sistema base do ORMS.

Esta modificação garante ao retalhista que não perde informação considerada essencial durante a passagem de um sistema para o outro.

Poder-se-ão encontrar novos atributos noutras fases do projecto. Assim a adição de novos atributos não está completamente fechada. Esta adição de atributos poderá ser feita distribuída pelas várias transições do projecto.

3.3 Solução do Problema

Depois de se ter terminado com a definição dos requisitos do cliente seguiu-se uma fase de análise de modo a conseguir desenhar uma solução com o menor impacto possível na aplicação base mas que consiga cumprir com todas as necessidades do retalhista. Esta é sempre uma preocupação primordial – tentar encontrar sempre a solução que menos impacto tem no sistema, isto porque diminui consideravelmente o risco de problemas no futuro visto que a Oracle lança regularmente actualizações às suas aplicações.

Para parte dos requisitos 39, 40, 70 e 211 (parte que o aluno ficou responsável) foi decido que a melhor abordagem seria criar praticamente um novo ecrã/formulário, de raiz, de modo a poder incluir todas as novas funcionalidades, minimizando assim o impacto no resto da aplicação. O ecrã escolhido foi o ecrã dos Item Attributes. Este ecrã já possuía alguns atributos relacionados com os artigos, no entanto, e visto que o retalhista não os considerava úteis, estes foram retirados, dando assim espaço a que os novos atributos fossem implementados.

Para cumprir com os requisitos na sua totalidade (não só o trabalho do aluno), foi decidido alterar outros ecrãs existentes, não só no que diz respeito ao seu conteúdo mas também em relação a funcionalidades.

3.4 Conclusões

Durante o CRP, foi possível ao retalhista mapear os seus processos de negócio às funcionalidades base do ORMS. Em todas as situações em que este mapeamento não foi possível, pois não satisfaziam as necessidades de negócio do retalhista, foram identificados requisitos de forma ao retalhista criar na nova aplicação funcionalidades que lhe permite manter os seus processos de negócio.

Parte dos requisitos levantados estão relacionados com a criação e manutenção de artigos no novo sistema. Assim, os requisitos apresentados neste capítulo são parte desses requisitos.

O retalhista pretende principalmente, que toda a informação que considera essencial existente no sistema anterior (material gasto nas embalagens, preço à unidade, preços marcados, marcas), sejam replicados na nova aplicação a ser implementada (ORMS). Deste modo, o retalhista consegue assegurar que mantém os processos de negócio que considera os mais correctos.

A solução encontrada (para a parte da responsabilidade do aluno), passa por adaptar um ecrã do ORMS, e incluir todos estes atributos considerados essenciais pelo cliente.

4 Implementação

Como já foi referido anteriormente, depois do levantamento dos requisitos na primeira fase do projecto, existe depois o exercício de encontrar a solução para esses requisitos. Para isso é feito primeiro uma análise funcional do requisito, onde se define com o retalhista claramente o que é que este pretende para cada requisito a nível funcional, e depois é feita uma análise técnica dessa parte funcional que apresentará já um nível de detalhe técnico da aplicação bastante elevado, e que servirá como base para depois fazer o desenvolvimento.

Neste capítulo é detalhada tanto a nível funcional como técnico dos requisitos apresentados no capítulo 3.

4.1 Desenho Funcional

As modificações que serão em seguida explicadas dizem respeito a apenas um ecrã da aplicação ORMS e tudo o que se suporta esse ecrã. Este projecto centrou-se no ecrã dos Item Attributes, no entanto, outros ecrãs foram também modificados para cumprir com os requisitos na sua totalidade.

4.1.1 Modificação 39

A modificação 39 inclui um conjunto de campos que não estão incluídos no ORMS base. Por isso, a primeira decisão passava por escolher em que ecrã se

deveria incluir estes campos. As opções eram: Item Master (ecrã principal dos items), Item Attributes (ecrã dedicado a novos atributos) ou um ecrã novo criado de raiz.

A escolha foi incluir estes novos atributos no ecrã Item Attributes, na medida em que era a solução que traria menos impactos à aplicação. O ecrã Item Master sendo o ecrã principal do artigo, é um ecrã mais sujeito a futuras actualizações da Oracle, e portanto qualquer alteração aqui iria obrigar a uma análise mais detalhada sempre que um novo patch da aplicação saísse. Seria portanto muito mais susceptível a erros.

Ecrã Item Attributes

Este ecrã vai ser actualizado para que se possa incluir alguns dos atributos considerados essenciais no ORMS. Vai ser neste ecrã que os utilizadores vão poder definir os campos relacionados com o preço à unidade (UEP Code e Net Content) como se pode ver na Imagem 7.

O campo que mostrará o preço à unidade (UEP) vai ser calculado com base nestes parâmetros e no preço de venda do artigo.

Este campo (UEP) não vai ser editável. Caso o artigo a ser editado tenha preços diferentes de loja para loja, o valor que vai ser calculado e mostrado é o preço à unidade para a zona base do grupo de zonas primário (estrutura que organiza os preços no ORPM).

Caso o utilizador entre neste ecrã sem primeiro definir o preço para o artigo, o campo que mostraria o valor do preço à unidade vai mostrar um 0.00, porque ainda não tem um preço com que trabalhar.

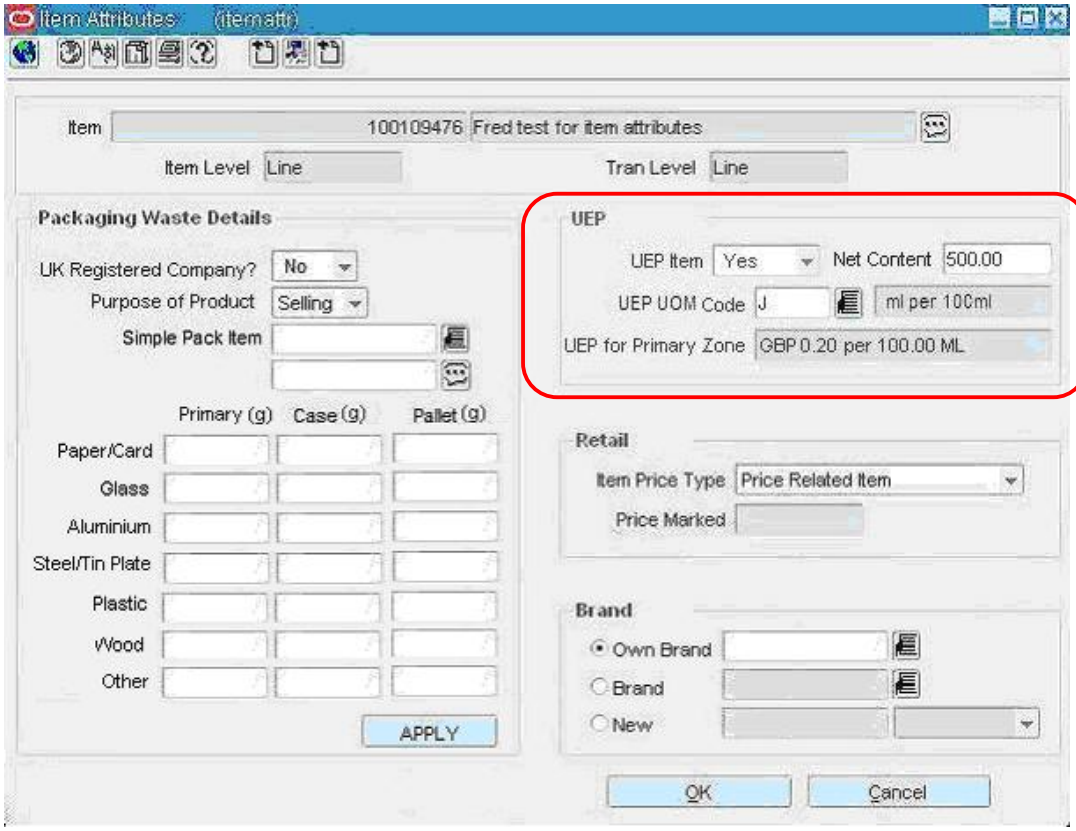


Imagem 7: Secção do UEP em destaque

O indicador de preço à unidade (campo UEP Item) será responsável pelo estado dos outros campos: se este estiver a YES então os outros campos estarão activos, caso contrário estarão desactivados.

No ecrã acima, o utilizador está a definir que o item vai ter preço à unidade. A quantidade do artigo são 500 e o código do preço à unidade é o J, ou seja, são 500 ml no artigo e quer saber o preço aos 100 ml. Baseado neste conjunto de informações, e tendo em conta que o preço do artigo é uma libra, o valor do UEP será:

- **UEP = (£1*100ml) / 500ml** è **£0.2 per 100ml**

Os códigos do UEP podem ser escolhidos a partir de uma lista de valores disponível através de um botão ao lado do campo “UEP UOM Code” (Imagem 8).

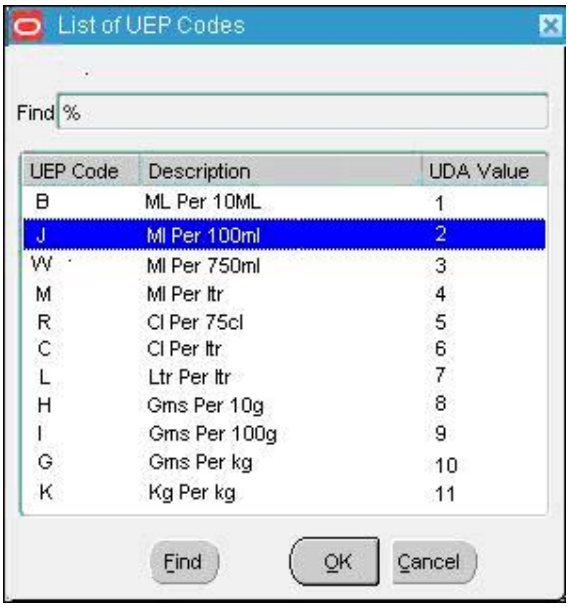


Imagem 8: Lista de Códigos UEP

Esta lista não é editável ao utilizador da aplicação. No entanto é possível ao retalhista criar novos UEP codes através da sua equipa de IT, inserindo os valores directamente na base de dados.

Os items de nível 2 vão herdar os valores dos respectivos items de nível 1, e os items de nível 1 herdarão o valor do UEP code da sub-classe caso esteja definido a este nível. Contudo estes serão apenas valores por defeito e podem ser alterados.

4.1.2 Modificação 40:

Na imagem seguinte (Imagem 9) encontra-se o aspecto do ecrã Item Attributes depois de incluídos os campos relativos ao preço marcado que se encontram destacados.

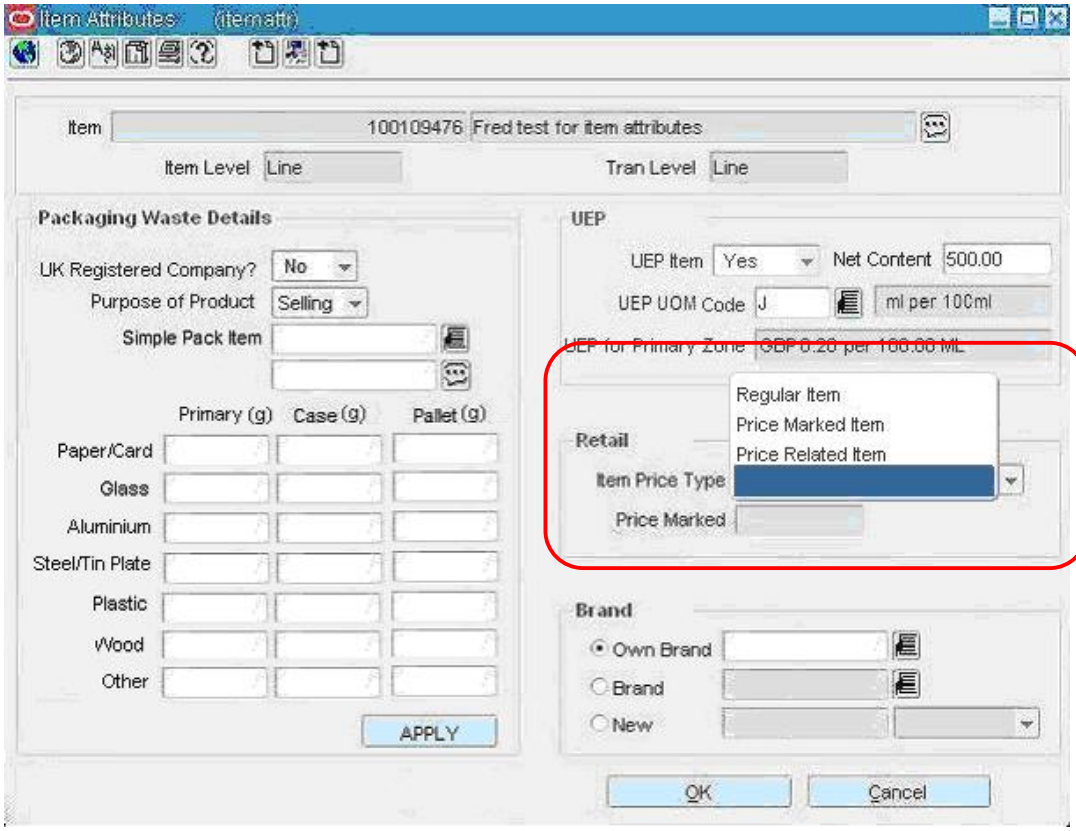


Imagem 9: Opções do campo Item Price Type

Como um artigo, ou é regular, ou tem o preço marcado ou tem o preço embebido, foi decido que o campo para definir se um artigo tinha o preço marcado seria o mesmo para definir se um artigo tinha o preço embebido.

O campo Item Price Type vai ser obrigatório, e vai ser partilhado com a modificação 70. Como se pode ver pela imagem este campo vai ser uma drop down list sem qualquer valor por defeito, obrigando assim ao utilizador escolher um dos 3 valores disponíveis.

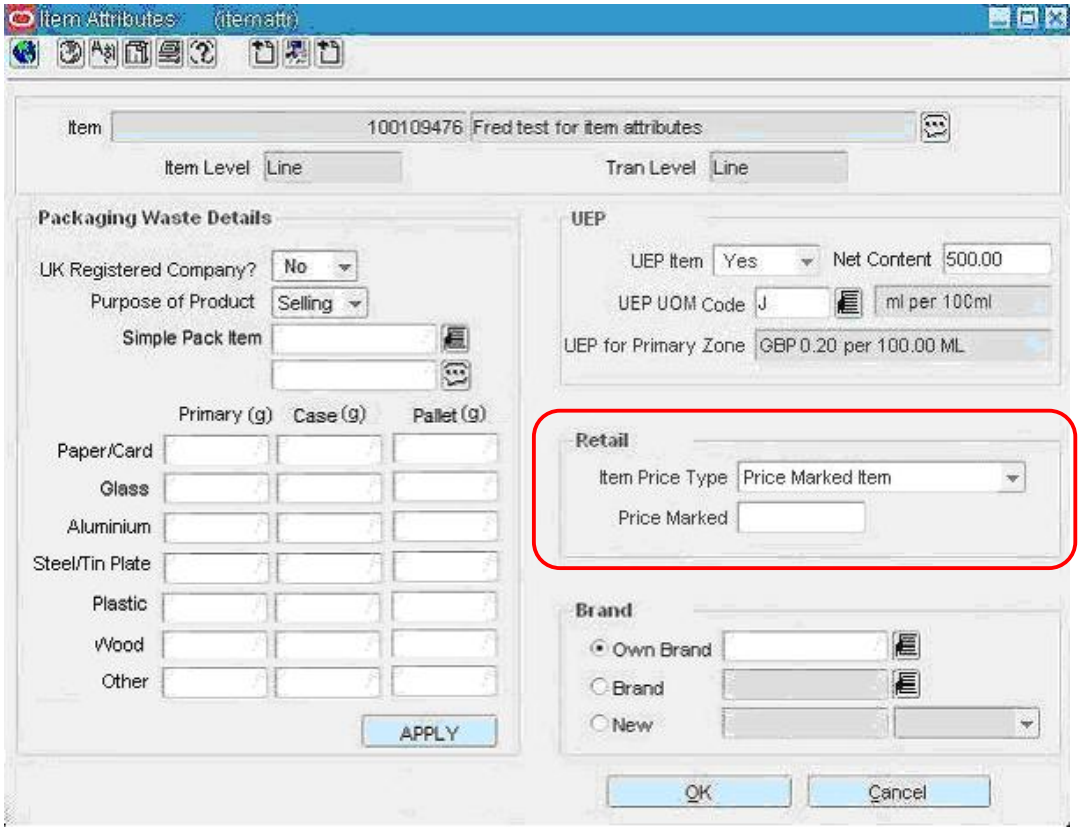


Imagem 10: Campo Price Marked está activo

Na altura de criação de um artigo, se a opção “Price Marked Item” (artigo com preço na embalagem) for escolhida (Imagem 10), o campo “Price Marked” é activado e o utilizador tem de inserir aqui o preço que vem registado na embalagem do produto. O facto de um artigo ter o preço marcado significa que o retalhista não o pode vender a um preço superior, por questões legais. Assim, como o sistema passa a ter o valor do preço marcado, é possível controlar e impedir que o sistema ou um utilizador definam um preço de venda superior ao preço marcado. Depois de um artigo ser criado e aprovado, o campo “Item Price Type” passa a ficar desactivo, ou seja, o utilizador não poderá alterar o tipo do artigo, contudo, o preço marcado pode ser alterado, mas não por um qualquer utilizador. Este valor só poderá ser alterado através de um utilizador com esse privilégio. Isto é um outro requisito do retalhista para impedir que este valor possa ser alterado por um qualquer utilizador.

Convém referir que o campo “Price Marked” se torna obrigatório caso a opção “Price Marked Item” for escolhida no campo “Item Price Type”.

Quando um utilizador com privilégios altera o preço marcado de um artigo já criado e aprovado aparecerá o aviso apresentado na Imagem 11, avisando o utilizador que devido a esta alteração poderão surgir conflitos devido ao preço do produto. Por

isso é que esta alteração a ser feita deve ser analisada antes, para ver todos os impactos que poderá causar.

Imagem 11: Aviso sobre mudar o preço marcado

Se o utilizador carregar em “Yes”, o novo preço será confirmado. Se carregar em “No” o preço marcado voltará para o preço previamente definido.

4.1.3 Modificação 70

Na imagem 12 encontra-se o aspecto do ecrã Item Attributes depois de incluídos os campos relativos ao preço embebido (“Price Related Item” na Imagem 13) que se encontram destacados.

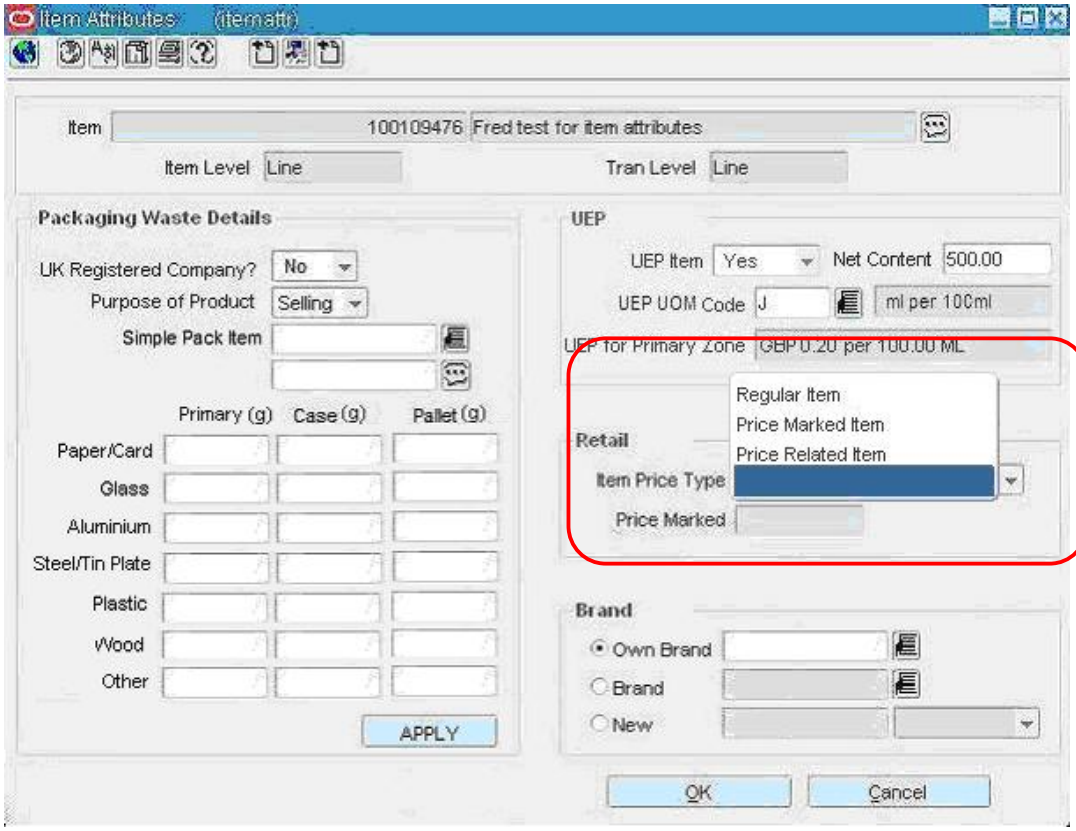


Imagem 12: Opções do campo Item Price Type

Como se pode observar, o campo é o mesmo que o referenciado na modificação 40. Este campo “Item Price Type” é obrigatório.

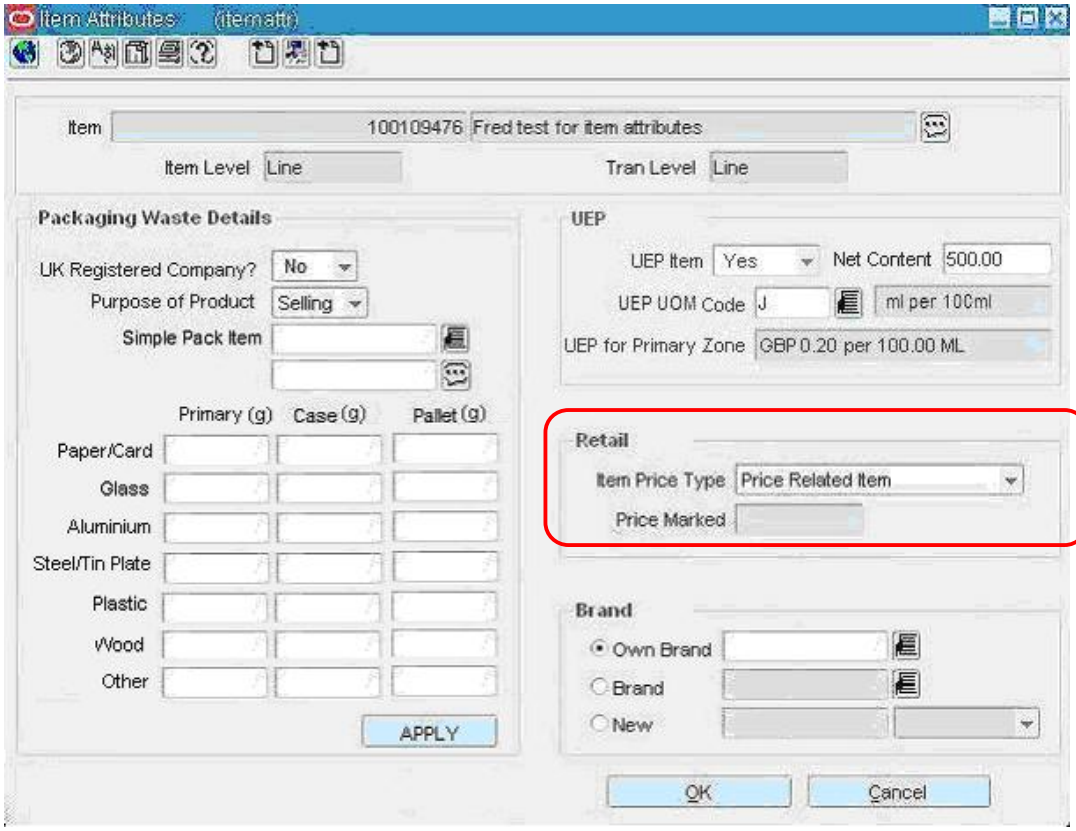


Imagem 13: O campo Price Marked permanece desactivo

Na altura de criação de um novo artigo, se a opção “Price Related Item” for escolhida, isso vai resultar numa limitação nos tipos de códigos de barras que se podem usar para este artigo (no terceiro nível do artigo).

4.1.4 Modificação 211

Como foi descrito previamente, esta modificação está relacionada com o requisito 211 em que o retalhista precisa que certos atributos considerados essenciais assim como as funcionalidades associadas passem para o novo sistema.

A Tabela 1 mostra os novos atributos que precisam de ser incluídos no ORMS, para ser mais preciso, no ecrã de Item Attributes.

Tabela 1: Lista de Atributos da Mod 211[1]

ATTRIBUTE	TYPE	DESCRIPTION
Purpose of Product	List Item	Will hold the value I: Internal S: Selling to customers. This field will be enabled when creating or editing Item Attributes for an item. Disabled in the view mode.
UK Registered Company?	List Item	Will hold the values Yes or No This field will be enabled when creating or editing Item Attributes for an item. Disabled in the view mode.
Paper/Card, Glass, Aluminium, Steel/Tin plate, plastic, wood, other.	Free Form Numeric	User can enter the values here in new/edit modes. These fields will be disabled in view mode. The columns case and pallet are always disabled except when "UK Registered Company?" is No. The values will always be entered in grams.
Brand	Radio button + Text Box + LOV	User can enter the values here in new/edit modes. Only one radio button can be chosen. If Brand is select the user will have the option to select the brand from a LOV.
Own Brand	Radio button + Text Box + LOV	User can enter the values here in new/edit modes. Only one radio button can be chosen. If Own Brand is select the user will have the option to select the own brand from a LOV.
New	Radio button + Text Box + Combo Box	User can enter the values here in new/edit modes. Only one radio button can be chosen. If New is selected, the user will have the option to create a new "brand" and add it to one of the LOVs (Brand or Own Brand)

[1] Tabela proveniente do documento funcional produzido pelo aluno durante o projecto.

Nesta modificação foi também pedido pelo retalhista, que os atributos existentes neste ecrã no ORMS base fossem retirados, pois não tinham utilidade para o retalhista. Essa informação como não seria utilizada, poderia causar confusão às pessoas responsáveis pela criação e manutenção de produtos, dado que se veriam numa situação em que teriam campos para preencher com informação que não tinham ao seu dispor. Outra das razões para a remoção dos atributos base foi a do espaço que é criado para os novos campos necessários das modificações 39, 40, 70 e 211.

A informação apresentada ao utilizador relativa ao "Packaging Waste Details" de um artigo vai depender se o artigo editado é um regular item ou um pack item.

Regular Item: se o artigo for um regular item e o utilizador escolher "No" no campo "UK Registered Company", então o utilizador terá de escolher um pack e definir a informação para esse pack. É obrigatório definir esta informação para todos os packs associados ao artigo.

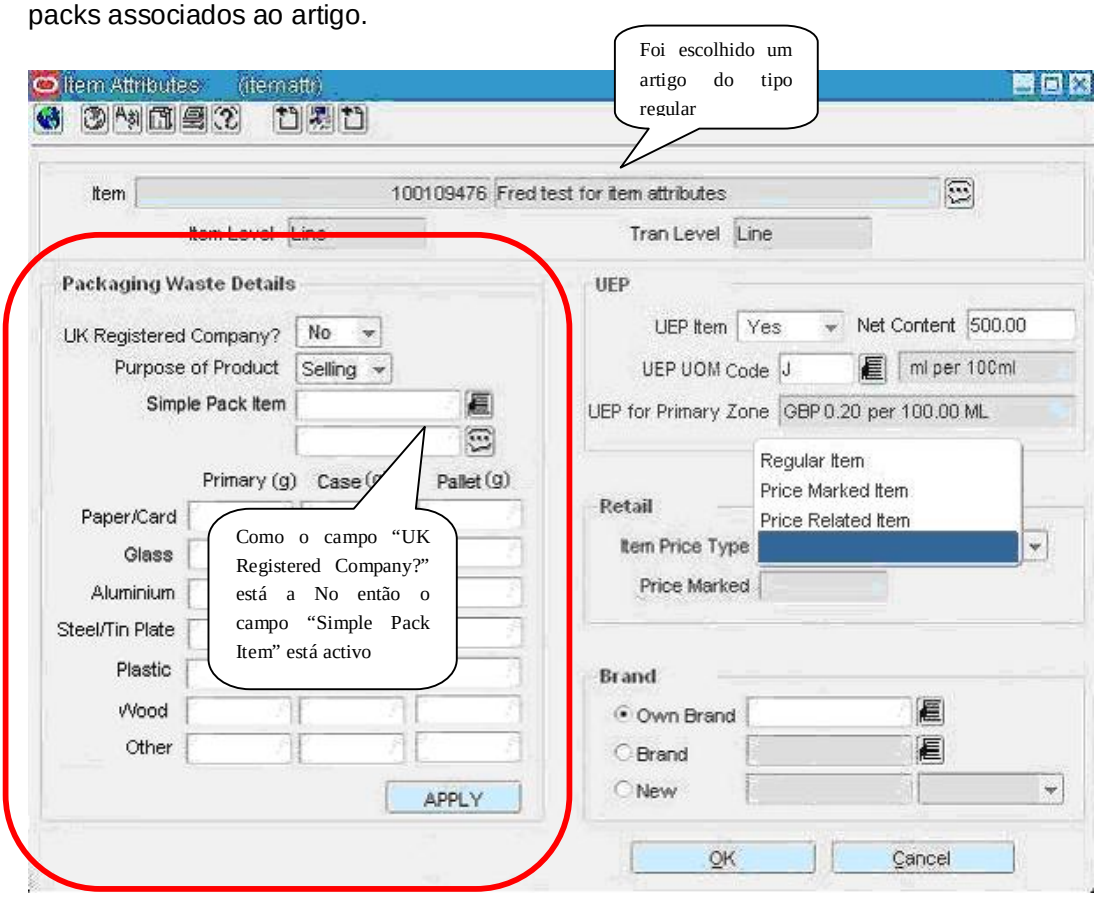


Imagem 14: Packaging Waste em destaque

No caso de o utilizador popular a informação deste ecrã sem definir primeiro um Item Pack para o artigo, e o campo "UK Registered Company" estiver a "No", vai ser apresentado a seguinte mensagem de erro (Imagem 15):

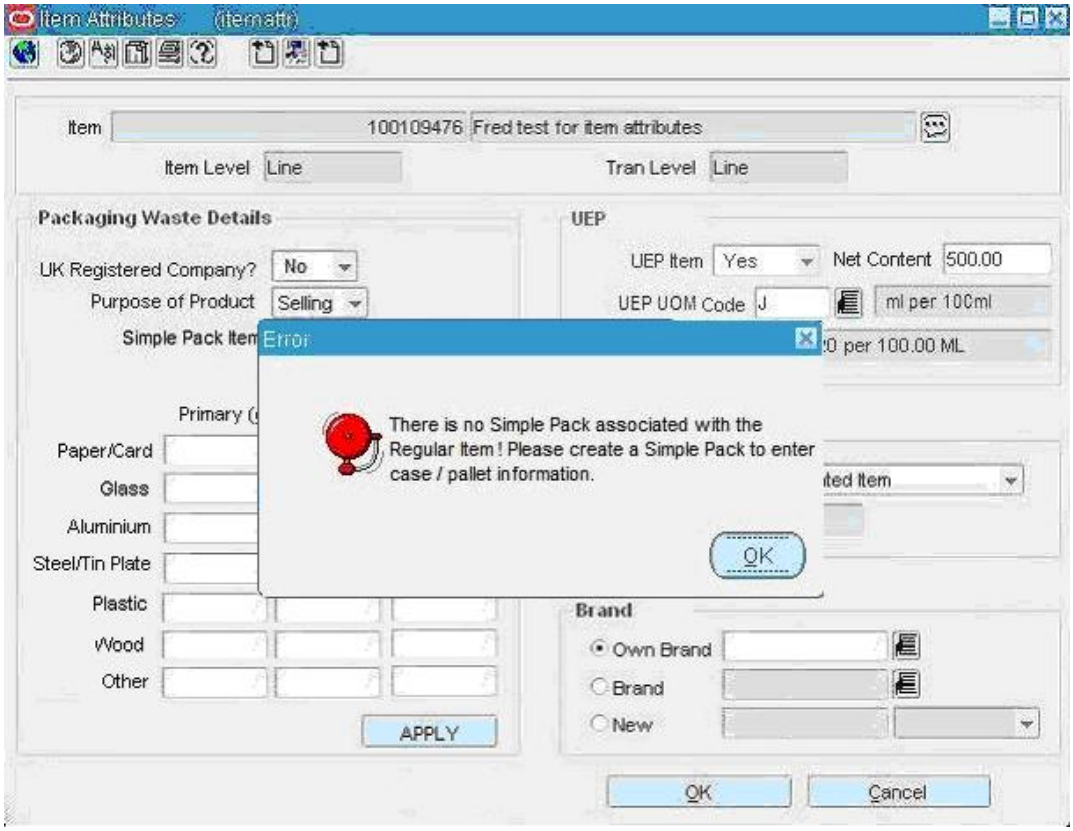


Imagem 15: Erro de Pack

Se o campo “UK Registered Company” estiver a “Yes”, então o utilizador não terá de definir esta informação ao nível do Pack, portanto as colunas “Case” e “Pallet” vão estar desactivadas. Apenas a coluna “Primary” estará activa e será possível editar por forma ao utilizador definir esta informação apenas ao nível do Regular Item que está a editar.

Pack Item: se o artigo a ser editado, neste ecrã, for um Simple Pack então apenas as colunas “Case” e “Pallet” estarão activas e será possível ao utilizador inserir informação, pois neste caso o utilizador não está a tratar o Regular Item, mas apenas o Simple Pack. A imagem 16 demonstra este cenário.

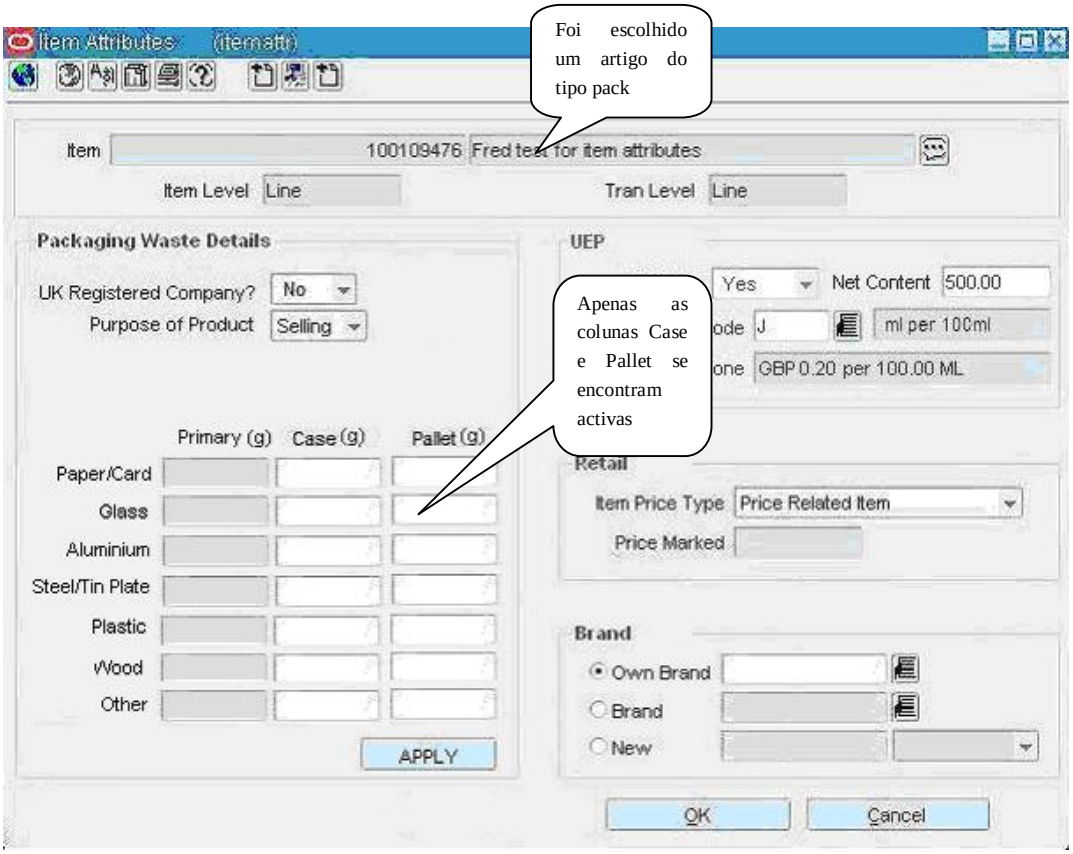


Imagem 16: Colunas Case e Pallet estão activas

Um resumo dos cenários possíveis neste conjunto de atributos encontra-se na tabela seguinte, tabela 2.

Tabela 2: Resumo dos cenários possíveis [2]

ITEM TYPE	UK REGISTERED COMPANY?	SIMPLE PACK LOV	PRIMARY	CASE	PALLET
Regular Item	Yes	Field Disabled	Field Disabled	Field Disabled	Field Disabled
Regular Item	No	Enabled for the user to choose the specific pack.	Field Disabled	Field Enabled	Field Enabled
Simple Pack	Field Disabled set to value 'No'	LOV not shown on screen for a Simple Pack item	Field Disabled	Field Enabled	Field Enabled

[2] Tabela proveniente do documento funcional produzido pelo aluno durante o projecto.

A área relacionada com as marcas também vai ser incluída no ecrã dos Item Attributes. Esta área vai ser responsável por manter os campos “Brand” e “Own Brand”.

Esta área vai ter 3 opções e apenas uma delas pode ser escolhida: ou escolhe uma “Brand”, ou escolhe uma “Own Brand” ou cria uma nova e define se é “Br and” ou “Own Brand”, como se pode ver na imagem seguinte (Imagem 17).

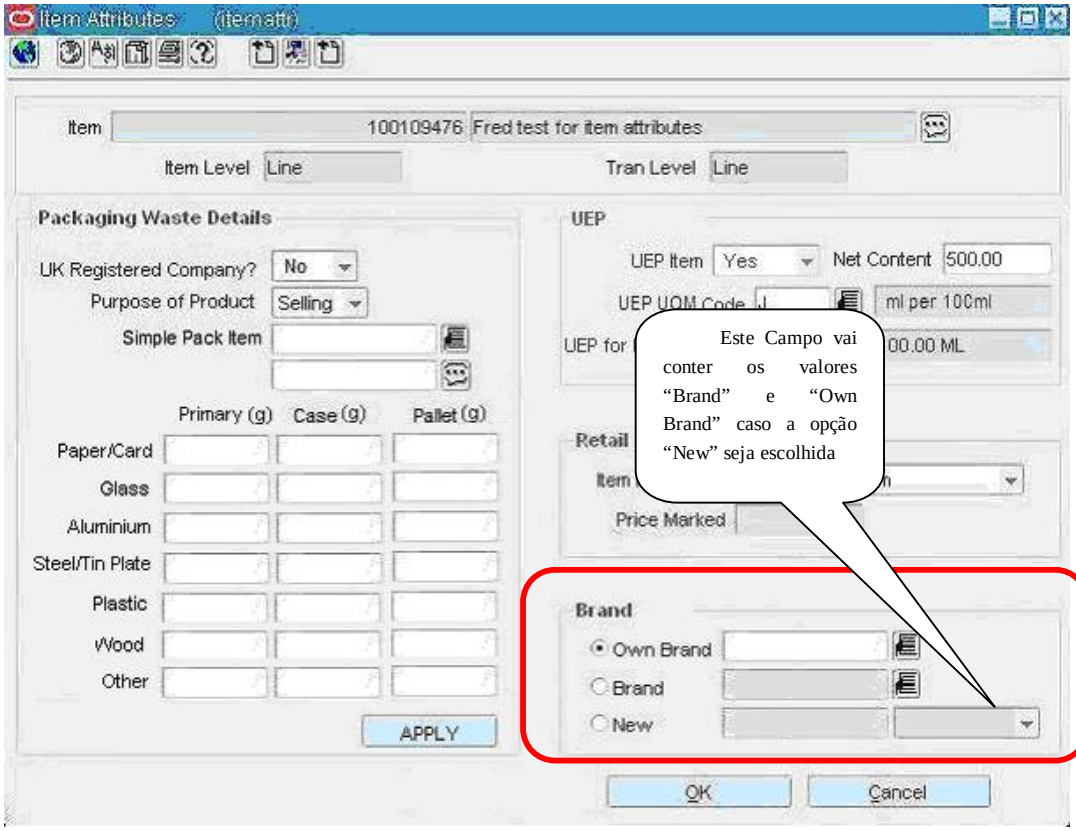


Imagem 17: Campos de Brand em destaque

4.2 Desenho Técnico

O desenho técnico de cada modificação divide-se nas seguintes secções: Campos novos do ecrã, funções novas/modificadas do ecrã, gatilhos do ecrã, consultas à base de dados e novas tabelas para manter os dados.

Na Imagem 18 é apresentada a arquitectura, de alto nível, dos ecrãs do ORMS.

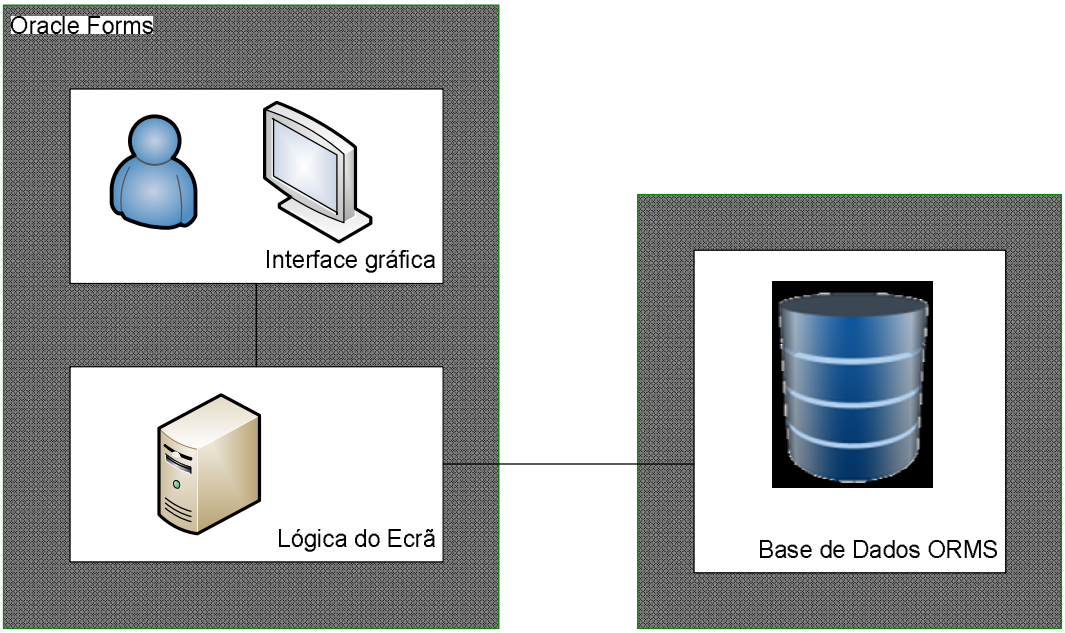


Imagem 18: Oracle Forms

Ao nível da interface gráfica incluem-se os novos campos, sejam botões, lista de valores ou caixas de texto. Ao nível da lógica do ecrã encontram-se os gatilhos, as funções e as consultas à base de dados. Assim, para cada uma das modificações serão apresentados estes pontos.

4.2.1 Modificação 39

Foram considerados os seguintes campos de forma a cumprir com os requisitos do cliente (mais detalhes da implementação estão disponíveis no anexo A.1):

UEP Item: drop down list, com Yes e No, que vai indicar se o preço à unidade é para ser calculado.

UEP Code: este campo vai conter o código atribuído para o artigo que estiver a ser definido. É a partir deste código que é possível determinar quais as unidades de medida para o artigo e para o preço à unidade.

Net Content: neste campo vai ser inserida a quantidade associada ao artigo.

UEP: este campo, que vai estar sempre desactivado, vai mostrar o valor do preço à unidade para o artigo se os valores necessários estiverem definidos. Os valores necessários são o UEP Code, o Net Content e o Retail Price que não é definido neste ecrã.

No que diz respeito a funções, foi preciso alterar algumas existentes e criar novas para que as funcionalidades associadas a cada um dos campos possam existir.

P_FORM_STARTUP: esta função é usada para iniciar o ecrã, assim como as variáveis e campos que lá existem.

P_POP_LISTS: esta função é responsável por preencher todas as listas com os respectivos valores. Por exemplo: popular o Item Price Type ou o UK Registered Company.

P_VIEW: esta função é usada para desactivar todos os campos quando o ecrã se encontra em modo de vista.

P_EDIT_NEW: esta função é usada para activar todos os campos quando o ecrã se encontra em modo de criação ou edição.

P_CALC_UEP_PRIM_ZONE: esta função como o nome indica é responsável por calcular o valor do preço à unidade e preencher o campo UEP com esse valor.

P_CHECK_REQUIRED: esta função é normalmente usada para verificar se todos os campos obrigatórios estão preenchidos à saída do ecrã. Alterada para verificar também os novos campos.

Apenas foi necessário desenhar uma nova consulta à base de dados para garantir o funcionamento do ecrã. Para o correcto funcionamento do ecrã, é necessário obter a lista existente de UEP Codes sempre que o ecrã é inicializado.

Em relação a eventos/gatilhos, foram necessários acrescentar alguns associados maioritariamente aos campos novos.

WHEN-LIST-CHANGED (UEP Item): acontece sempre que o valor definido muda. Se mudar para Yes vai activar os campos novos excepto o UEP. Caso contrario vai fazer desactivar todos os campos.

WHEN-VALIDATE-ITEM (Net Content): sempre que valores são inseridos neste campo este gatilho é iniciado. Se os outros valores estiverem definidos então vai calcular o novo UEP, caso contrário não faz nada.

KEY-LISTVAL (UEP Code): responsável por chamar uma lista de valores, na qual o utilizador pode escolher o código.

WHEN-VALIDATE-ITEM (UEP Code): valida se a informação inserida no campo UEP Code é válida, e recalcula o valor do UEP.

Para um maior detalhe sobre as alterações ao ecrã para a modificação 39, encontra-se disponível no Anexo A.1 parte do documento da modificação entregue ao retalhista.

4.2.2 Modificação 40 e 70

Foram considerados os seguintes campos de forma a cumprir com os requisitos do cliente (mais detalhes da implementação estão disponíveis no anexo A.2):

Item Price Type: esta drop down list vai ter três valores, “Price Related Item”, “Price Marked Item” e “Regular Item”.

Price Marked: este campo vai manter o preço que está marcado no produto. Só vai estar activo quando a opção “Price Marked Item” for escolhida no campo “Item Price Type”.

No que diz respeito a funções, foi preciso alterar algumas existentes e criar novas para que as funcionalidades associadas a cada um dos campos pudessem existir.

P_VIEW: esta função é usada para desactivar todos os campos quando o ecrã se encontra em modo de vista. Alterada para incluir a desactivação destes dois novos campos.

P_POP_LISTS: esta função é responsável por preencher todas as listas com os respectivos valores. Por exemplo: preencher o Item Price Type ou o UK Registered Company. Foi alterada para preencher também a drop down list: Item Price Type.

P_EDIT_NEW: esta função é usada para activar todos os campos quando o ecrã se encontra em modo de criação ou edição. Alterada para activar o campo Item Price Type e o Price Marked.

P_CHECK_REQUIRED: esta função é normalmente usada para verificar se todos os campos obrigatórios estão preenchidos à saída do ecrã. Alterada para verificar também os novos campos. Se o tipo de artigo escolhido for “Price Marked Item”, então o campo “Price Marked” também é obrigatório.

P_EXIT.SAVE_FORM: este programa foi modificado para chamar a função NB_ITEM_ATTRIBUTES_SQL.UPDATE_RETAIL_VALUES para artigos com preço marcado, que vai actualizar o preço de venda de acordo com o valor marcado no produto.

Em relação a eventos/gatilhos, foi necessário acrescentar apenas um associado aos campos novos.

WHEN-LIST-CHANGED (Item Price Type): acontece sempre que o valor definido muda. Se mudar para “Price Marked Item” vai activar o campo “Price Marked”. Caso contrário vai fazer desactivar este mesmo campo.

4.2.3 Modificação 211

A quantidade de campos da modificação 211 é bastaste maior em relação às outras 3 modificações, assim, de forma a cumprir com os requisitos do cliente foram considerados os seguintes campos (mais detalhes da implementação estão disponíveis no anexo A.3):

UK Registered Company: este campo indica se o retalhista paga a um fornecedor do Reino Unido (UK). Se o valor deste campo for “No” então duas colunas novas ficam activas no formulário para o utilizador inserir a informação para pack e para pallet.

Purpose of product: este campo indica se o produto é para ser usado internamente, ou se é para venda. Se o propósito do produto é “Internal” então as três colunas, para inserir as quantidades gastas em embalagem, ficarão desactivadas impedindo assim os utilizadores de inserir qualquer valor.

Simple Pack Item: quando este campo está activado, o utilizador tem a possibilidade de escolher qual o Item Pack que vai ser editado utilizando as colunas “pack” e “pallet”.

Paper/card: aqui o utilizador pode inserir a quantidade (em gramas) de papel envolvido no “Packaging waste details”.

Glass: aqui o utilizador pode inserir a quantidade (em gramas) de vidro envolvido no “Packaging waste details”.

Aluminium: aqui o utilizador pode inserir a quantidade (em gramas) de alumínio envolvido no “Packaging waste details”.

Steel/tin plate: aqui o utilizador pode inserir a quantidade (em gramas) de ferro/estanho envolvido no “Packaging waste details”.

Plastic: aqui o utilizador pode inserir a quantidade (em gramas) de plástico envolvido no “Packaging waste details”.

Wood: aqui o utilizador pode inserir a quantidade (em gramas) de madeira envolvido no “Packaging waste details”.

Other: aqui o utilizador pode inserir a quantidade (em gramas) de outro tipo de material envolvido no “Packaging waste details”.

Own Brand: Este campo vai permitir ao utilizador, através do uso de uma lista de valores, escolher uma marca própria para associar ao item.

Brand: Este campo vai permitir ao utilizador, através do uso de uma lista de valores, escolher uma marca para associar ao item.

New: caso o utilizador não encontrar a marca que pretende, o utilizador poderá criar aqui uma nova marca, e determinar se é própria ou não. Depois de criada, vai aparecer na respectiva lista.

No que diz respeito a funções, foi preciso alterar algumas existentes e criar novas para que as funcionalidades associadas a cada um dos campos possam existir.

INTERNAL_VARIABLES: esta função é usada no início do form é vai ser usada para criar variáveis ao nível do form, necessárias para as funcionalidades do mesmo.

P_FORM_STARTUP: esta função é usada para iniciar o ecrã, assim como as variáveis e campos que lá existem. Normalmente chama outras funções.

P_SET_SCREEN: esta função é responsável por desactivar campos que não são aplicáveis ao artigo que está a ser editado.

P_EDIT_NEW: esta função é usada para activar todos os campos quando o ecra se encontra em modo de criação ou edição.

P_POP_LISTS: esta função é responsável por preencher todas as listas com os respectivos valores. Por exemplo: preencher o Item Price Type ou o UK Registered Company. Alterada para preencher também as seguintes drop down list:: UK Registered Company, Purpose of Product, and Brand Type.

P_CHECK_REQUIRED: esta função é normalmente usada para verificar se todos os campos obrigatórios estão preenchidos à saída do ecrã.

P_EXIT.SAVE_FORM: esta função é normalmente chamada à saída do ecrã. Alterada para chamar a função P_CHECK_REQUIRED.

P_VIEW: esta função é usada para desactivar todos os campos quando o ecrã se encontra em modo de vista. Alterada para incluir os novos campos.

P_UPDATE_SCREEN: este programa vai ser chamado de cada vez que o utilizador executar determinadas acções. Vai ser usado para activar, desactivar campos de acordo com as acções/escolhas do utilizador.

Em relação a eventos/gatilhos, foi necessário acrescentar os seguintes associados aos campos novos.

WHEN-LIST-CHANGED (Purpose of Product): acontece sempre que o valor definido muda. Sempre que mudar o valor escolhido, a função P_UPDATE_SCREEN vai ser chamada.

WHEN-RADIO-CHANGED (Own Brand, Brand, and New): esta acção vai decorrer sempre que a opção escolhida na área das marcas mudar. Sempre que isto acontecer os campos vão ser actualizados de acordo com o botão escolhido.

WHEN-VALIDATE-ITEM (Simple Pack Item): esta função vai ser usada para validar o Item Pack inserido pelo utilizador.

POST-TEXT-ITEM (Simple Pack Item): esta função vai popular as colunas de case e pallet de acordo com o Item Pack inserido pelo utilizador

Foram necessárias três novas consultas de acesso à base de dados para o correcto funcionamento do ecrã. Assim, é necessário obter a seguinte lista de valores para os seguintes pontos:

- **Simple Packs**: todos os packs associados ao item que está a ser definido pelo utilizador;
- **Own Brand**: todas as marcas próprias disponíveis para associar ao artigo;
- **Brand**: todas as marcas disponíveis para associar ao artigo.

4.3 Conclusões

O desenho da solução encontrada para os requisitos apresentados, para este projecto teve duas componentes: o desenho funcional em que é apresentado o futuro funcionamento do ecrã, para que o retalhista possa aprovar as alterações; e o desenho técnico, que aborda as alterações necessárias ao ecrã para que este possa funcionar como desejado.

As alterações apresentadas, centram-se num só ecrã, visto que foi este o trabalho efectuado ao longo do projecto.

5 Conclusões e Trabalho Futuro

Neste último capítulo são apresentadas conclusões do trabalho efectuado, e é abordado o tema de trabalho futuro.

5.1 Satisfação dos Objectivos

O projecto decorreu dentro do planeado. O principal objectivo deste trabalho, que passava por adaptar e implementar o módulo ORMS aos processos de negócio de um grande retalhista, foi cumprido.

Como foi descrito nos capítulos anteriores, a primeira fase foi o levantamento de requisitos, na qual, com a colaboração do retalhista, foram identificadas as principais necessidades deste na criação e manutenção de artigos.

Na fase seguinte, foi iniciado o desenho funcional, sempre com o acompanhamento do retalhista, para que a solução final fosse a solução pretendida. Nesta fase, como é normal, a solução teve várias variantes, chegando-se finalmente à que foi apresentada no capítulo 4.

O desenho técnico revelou-se como o maior desafio deste projecto, pois necessita de aprofundado conhecimento técnico e estrutural do ORMS. O modo de funcionamento dos formulários/ecrãs, dos programas que os suportam, foram conhecimentos adquiridos ao longo deste projecto.

Este projecto permitiu também que o aluno tenha adquirido um conhecimento mais profundo de negócio (sobre um dos processos mais críticos da área de retalho:

gestão de categorias, gestão de produto, negociação e compras, gestão de inventário, gestão de canais de distribuição, definição de preços de venda).

Além disso, o aluno adquiriu conhecimentos sobre as metodologias associadas às fases de um projecto de delivery (levantamento de requisitos, análise funcional, desenho técnico, desenvolvimento, testes e implementação).

5.2 Trabalho Futuro

Completadas as fases de levantamento de requisitos, análise funcional e desenho técnico, seguem-se o desenvolvimento, testes e implementação.

Neste momento, as alterações desenhadas encontram-se em desenvolvimento por uma equipa externa ao projecto. Para que o desenvolvimento decorra como previsto, houve uma passagem de conhecimento entre o aluno, que efectuou o desenho, e a equipa que vai desenvolver.

No fim do desenvolvimento, o projecto de delivery voltará às mãos do aluno, que será responsável pelas fases de testes e implementação. Na fase de testes, cada um dos requisitos será testado ao pormenor, assim como possíveis impactos destas alterações.

Num aspecto mais global, é óbvio que o processo de criação e manutenção de artigos pode ser sempre melhorado. O processo de criação de um artigo continua a ser de certa forma um processo longo. Arranjar maneira de reduzir este tempo de criação mantendo ou melhorando a qualidade e quantidade de informação associada ao artigo seria certamente o caminho a seguir.

Referências

[Ref 1] Wikipédia. Enabler Wipro - Wikipédia, a enciclopédia livre. Disponível em http://pt.wikipedia.org/wiki/Enabler_Wipro

[Ref 2] Celio Ramos. Mercadologia: “Varejo: Que Negócio É Esse?”. Disponível em http://territorio.terra.com.br/canais/vitrine/colunas/materia.asp?codArea=3&materiaID=79

[Ref 3] Wikipédia. ERP - Wikipédia, a enciclopédia livre. Disponível em http://pt.wikipedia.org/wiki/ERP

[Ref 4] Henrique L. Corrêa, Irineu GN Gianesi e Mauro Caon. Planejamento, Programação e Controle da Produção.

[Ref 5] Christopher Koch, Derek Slater e E. Baatz. The ABCs of ERP.

[Ref 6] Wipro Retail. Documentação interna.

Anexo A: Documentação Técnica

Em anexo estão presentes parte da documentação produzida pelo aluno no âmbito deste projecto.

A.1 Modificação 39

A.2 Modificações 40 e 70

A.3 Modificação 211

A new section by name "UEP" will be added to the Item Attributes screen. This section will hold the control fields for holding the Unit Equivalent Price attributes of the item. Based on the user input for Net Content of the item and the UEP UOM Code, the Unit Equivalent Price of the item will be calculated. This value will be calculated every time the form opens and hence not going to be stored anywhere in ORMS. The UEP control fields will be suitably placed in the Item Attributes screen by the developer as shown in the screenshot below:

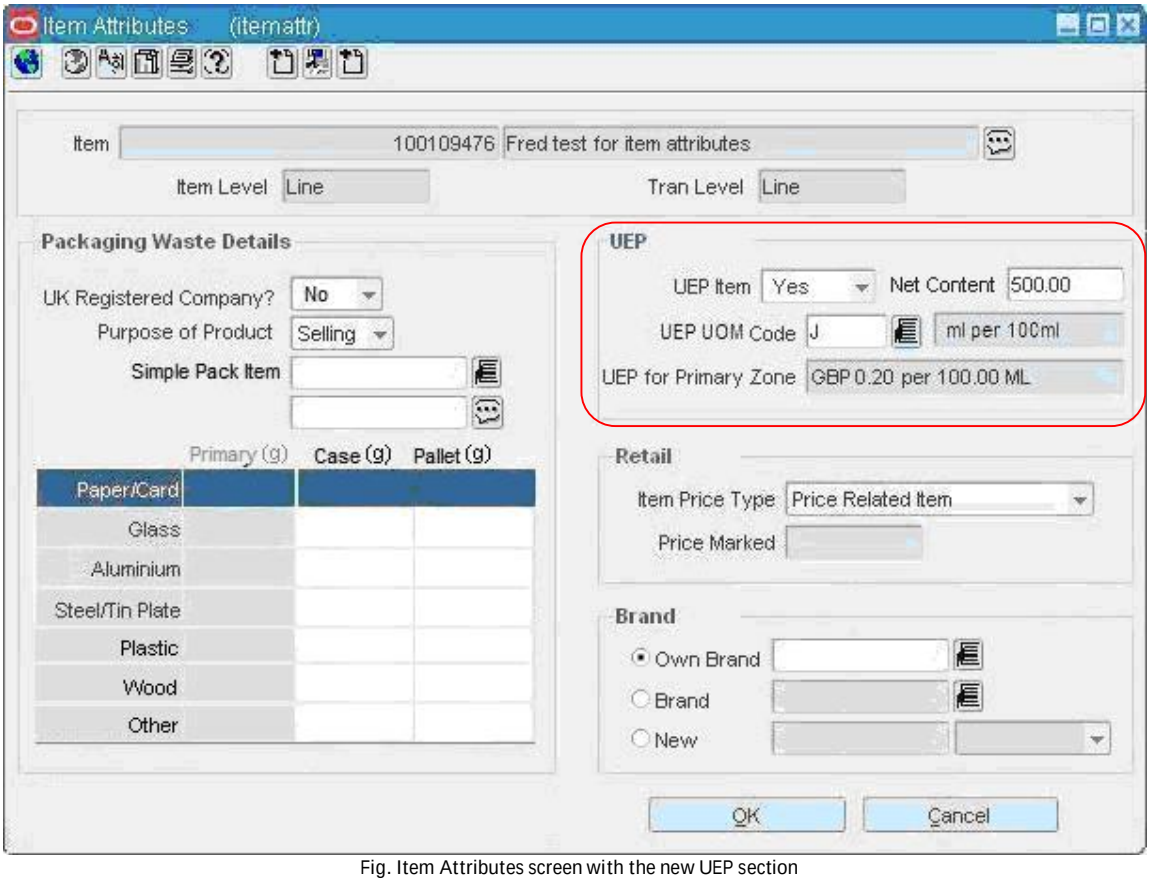


Fig. Item Attributes screen with the new UEP section

The behaviour of these fields is explained in the subsequent section Application Logic Flow. The various new fields (to fetch the input from the user) to be added in the screen are:

1. UEP Item
2. Net Content
3. UEP UOM Code (with push button for LOV and description field)
4. UEP For Primary Zone

Based on the input for the Net Content and UEP UOM Code by the user, the Unit Equivalent Price will be calculated as:

$$\text{Unit Equivalent Price of the item} = \frac{\text{(UEP UOM Code Qty x Unit Retail Price)}}{\text{Net Content}}$$

For e.g., for a UEP UOM Code of "ml per 100ml", item net content of 500ml and Unit Retail Price of £1, the Unit Equivalent Price will be 100 * 1 / 500 = £0.20 per 100ml (as depicted in the screenshot above).

§ Parameters

No change required

§ Canvases

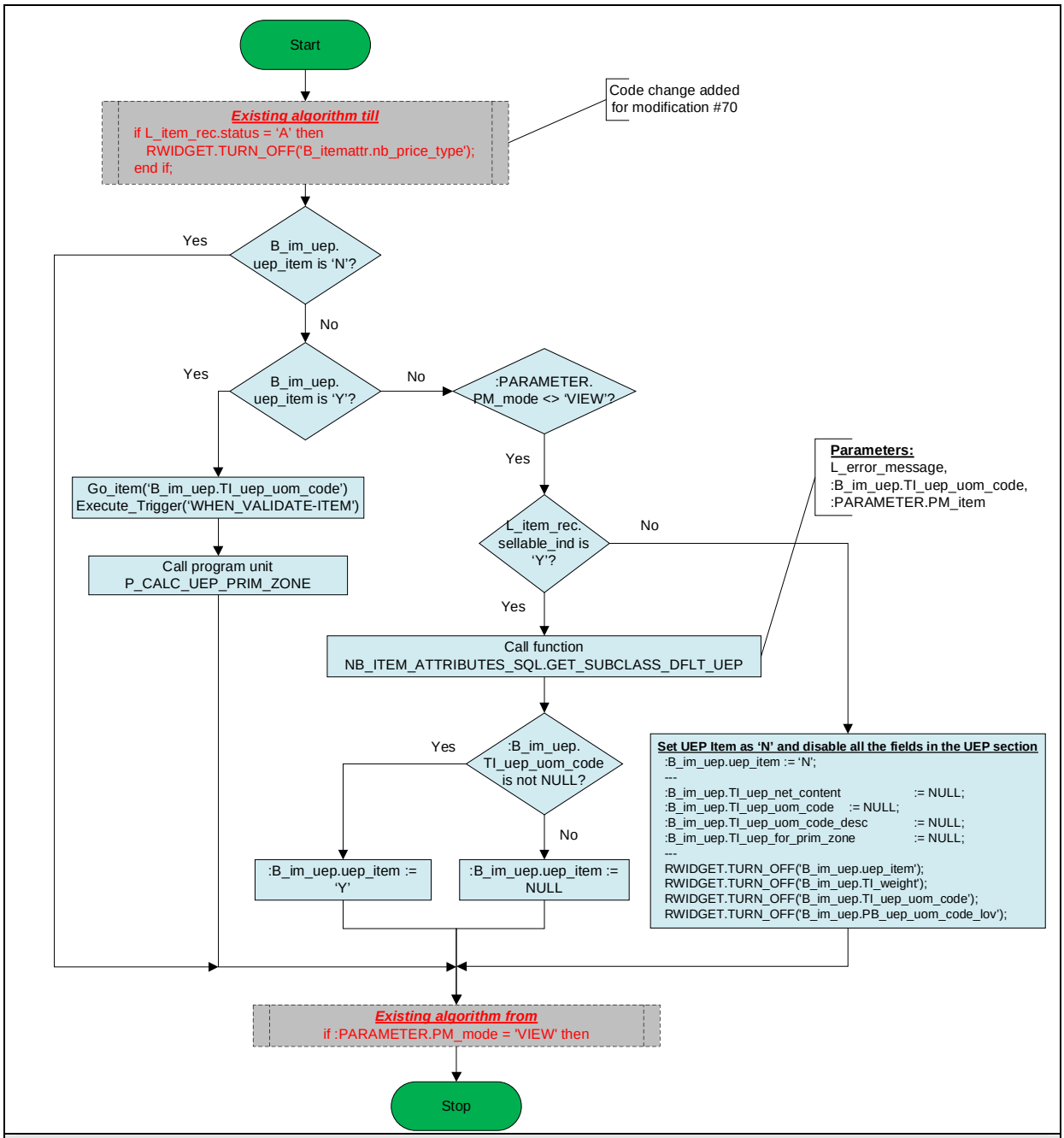
No change required

§ Blocks

This block should be changed to consider the new table NB_ITEM_ATTRIBUTES, instead of the ITEM_ATTRIBUTES table.

§ Program Units

Function Name:	P_FORM_STARTUP [Existing]
Description:	This function is used to initialise the form and its elements and perform initial validations with data population in fields.
Parameters	None
Returns:	None
Algorithm	
The following changes must be placed on the existing algorit hm (changes identified in black, and existing algorithm in red):	



Development details

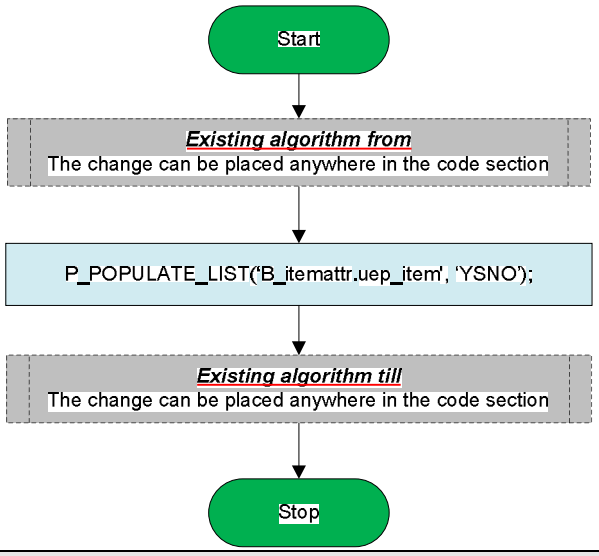
Exception handling:	Already defined
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable as parameter, and then raises the exception FORM_TRIGGER_FAILURE.

Function Name:	P_POP_LISTS [New]
Description:	This function is used to populate the Item Price Type list field during start-up. The same function can

	be used for populating fields to be added as part of future modifications , if any.
Parameters	None
Returns:	None

Algorithm

The following changes must be placed on the existing algorithm (changes identified in black, and existing algorithm in red):



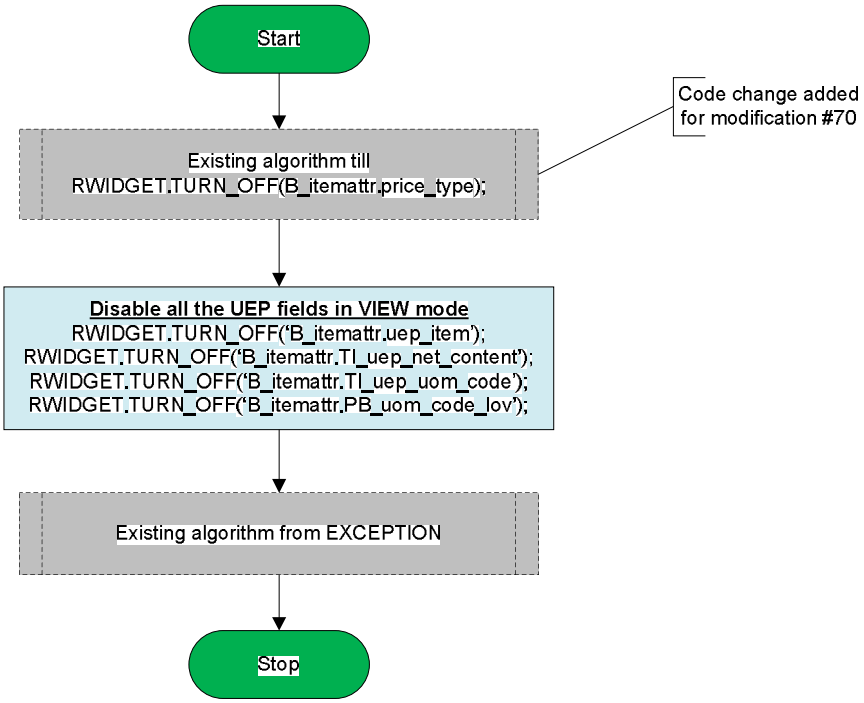
Development details

Exception handling:	Already defined
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable as parameter, and then raises the exception FORM_TRIGGER_FAILURE.

Function Name:	P_VIEW [Existing]
Description:	This function is used to disabling all the form fields when the form is opened in VIEW mode.
Parameters	None
Returns:	None

Algorithm

The following changes must be placed on the existing algorithm (changes identified in black, and existing algorithm in red):

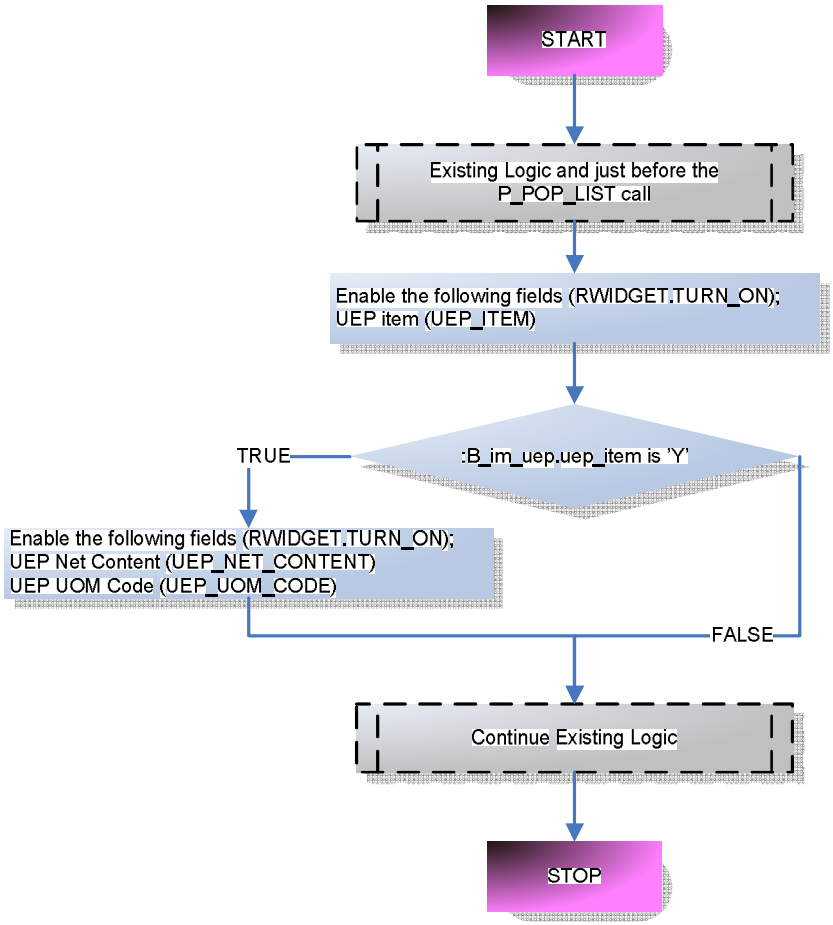


Development details	
Exception handling:	Already defined
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable as parameter, and then raises the exception FORM_TRIGGER_FAILURE.

Program Unit Name:	P_EDIT_NEW [Existing]
Description:	This program unit will be amended to enable the UEP fields
Returns:	BOOLEAN

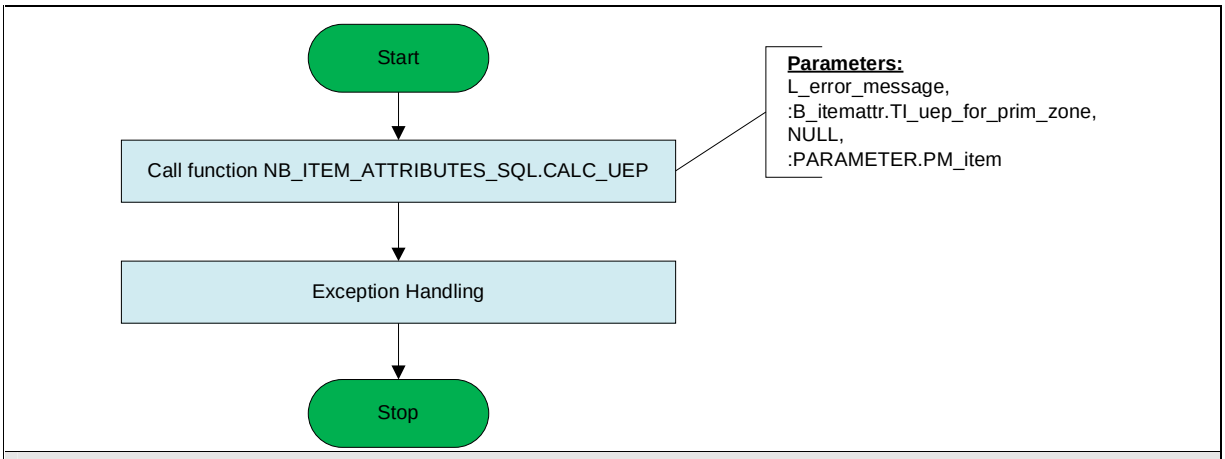
Parameters				
Name	Datatype	IN/OUT	Default	Description
N/A				

Algorithm
The following changes must be placed on the existing algorithm (existing logic is highlighted in red font);



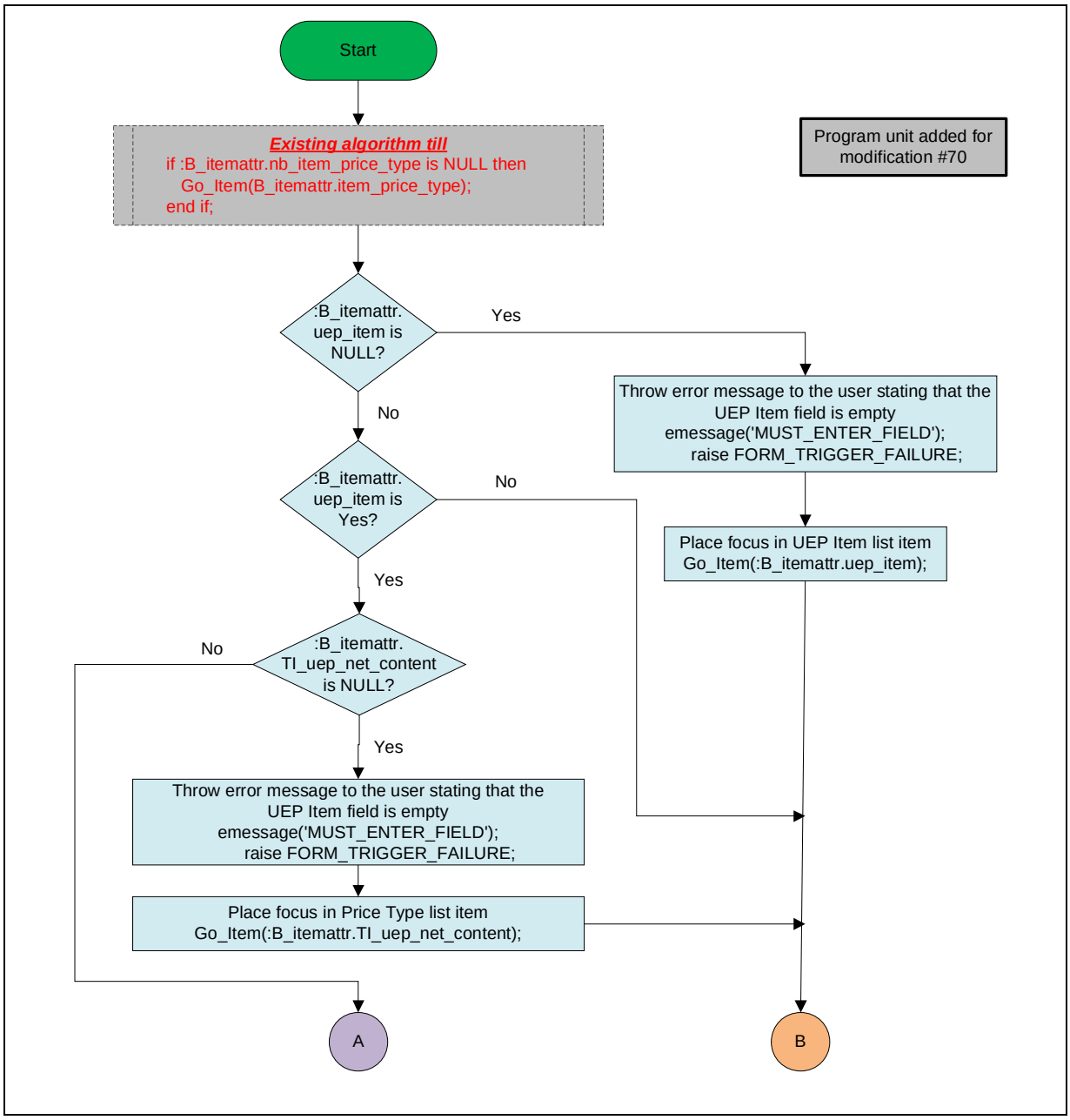
Development details	
Exception handling:	n/a
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_ error_message variable back as parameter, and then raises the exception FORM_TRIGGER_FAILURE.

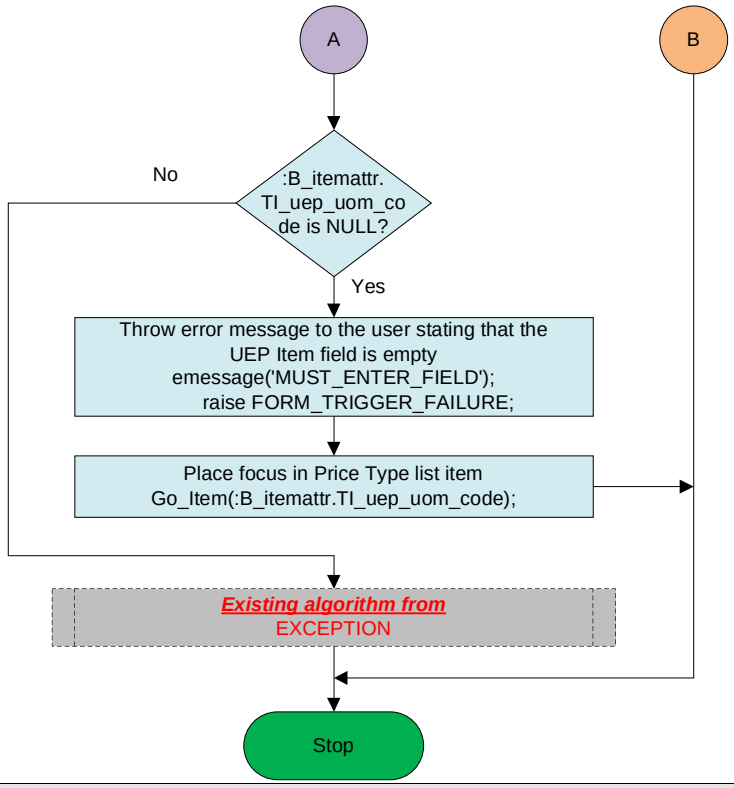
Function Name:	P_CALC_UEP_PRIM_ZONE [New]
Description:	This function is used to calculate the UEP for the Primary Zone. It is fired when all the required inpu ts for calculation of UEP are available.
Parameters	None
Returns:	None
Algorithm	
The following changes must be placed on the existing algorithm (changes identified in black, and existing algorithm in red):	



Development details	
Exception handling:	Already defined
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable as parameter, and then raises the exception FORM_TRIGGER_FAILURE.

Function Name:	P_CHECK_REQUIRED [Existing]
Description:	This function is used to check if all the required fields in the Item Attributes form have been filled by the user.
Parameters	None
Returns:	None
Algorithm	
The following changes must be placed on the existing algorithm (changes identified in black, and existing algorithm in red):	


Start
Existing algorithm till
if :B_itemattr.nb_item_price_type is NULL then
Go_Item(B_itemattr.item_price_type);
end if;
Program unit added for modification #70
:B_itemattr. uep_item is NULL?
Yes
Throw error message to the user stating that the UEP Item field is empty
emessage('MUST_ENTER_FIELD');
raise FORM_TRIGGER_FAILURE;
No
:B_itemattr. uep_item is Yes?
No
Place focus in UEP Item list item
Go_Item(:B_itemattr.uep_item);
Yes
No
:B_itemattr. TI_uep_net_content is NULL?
Yes
Throw error message to the user stating that the UEP Item field is empty
emessage('MUST_ENTER_FIELD');
raise FORM_TRIGGER_FAILURE;
Place focus in Price Type list item
Go_Item(:B_itemattr.TI_uep_net_content);
A
B



Development details

Exception handling:	Already defined
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable as parameter, and then raises the exception FORM_TRIGGER_FAILURE.

§ LOVs

LOV NAME	TITLE	RECORD GROUP	COLUMN MAPPING PROPERTIES	RETURN ITEM	DISPLAY WIDTH	COLUMN TITLE
LOV_UEP_UOM_CODE	List Of UEP Codes	RG_UEP_UOM_CODE	UEP_CODE	B_ITEMATTR.UEP_UOM_CODE	As required	UEP Code
			DESC	B_ITEMATTR.TI_UOM_CODE_DESC	As required	Description
			UEP_UDA	NA	As required	UDA Value

§ Record Groups

NAME	QUERY	COLUMN SPECIFICATION
RG_UEP_UOM_CODE	NB_UEP_UOM_CODES NUUC (PK **UEP_CODE**; UEP_UDA) — nuuc.uep_uda = uv.uda_value; nuuc.uep_id = uv.uda_id — UDA_VALUES UV (UDA_VALUE_DESC DESC)	UEP_CODE DESC UEP_UDA_VALUE

§ Items

BLOCK	ITEM	ITEM TYPE	CANVAS	DATA TYPE	MINIMUM LENGHT	INSERT/ UPDATE	REQUIRED	DATABASE ITEM	DEFAULT VALUE	PROMPT	COMMENTS
B_ITEMATTR	DI_UEP	Display Item	C_ITEMATTR	CHAR(64)	N/A	N/A	N/A	N/A	N/A	UEP	Always visible
B_ITEMATTR	UEP_ITEM	List Item	C_ITEMATTR	CHAR(16)	3	Yes	No	Yes	NULL	UEP Item	Always visible
B_ITEMATTR	UEP_NET_CONTENT	Text Item	C_ITEMATTR	NUMBER(6,2)	6	Yes	No	Yes	N/A	Net Content	Always visible
B_ITEMATTR	UEP_UOM_CODE	Text Item	C_ITEMATTR	CHAR(1)	1	Yes	No	Yes	N/A	UEP UOM Code	Always visible
B_ITEMATTR	PB_UOM_CODE_LOV	Push Button	C_ITEMATTR	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Always visible
B_ITEMATTR	TI_UOM_CODE_DESC	Display Item	C_ITEMATTR	CHAR(64)	20	N/A	N/A	N/A	N/A	N/A	Always visible

B_ITEMATTR	TI_UEP_FOR_PRIM_ZONE	Display Item	C_ITEMATTR	CHAR(80)	30	N/A	N/A	N/A	N/A	UEP for Primary Zone	Always visible

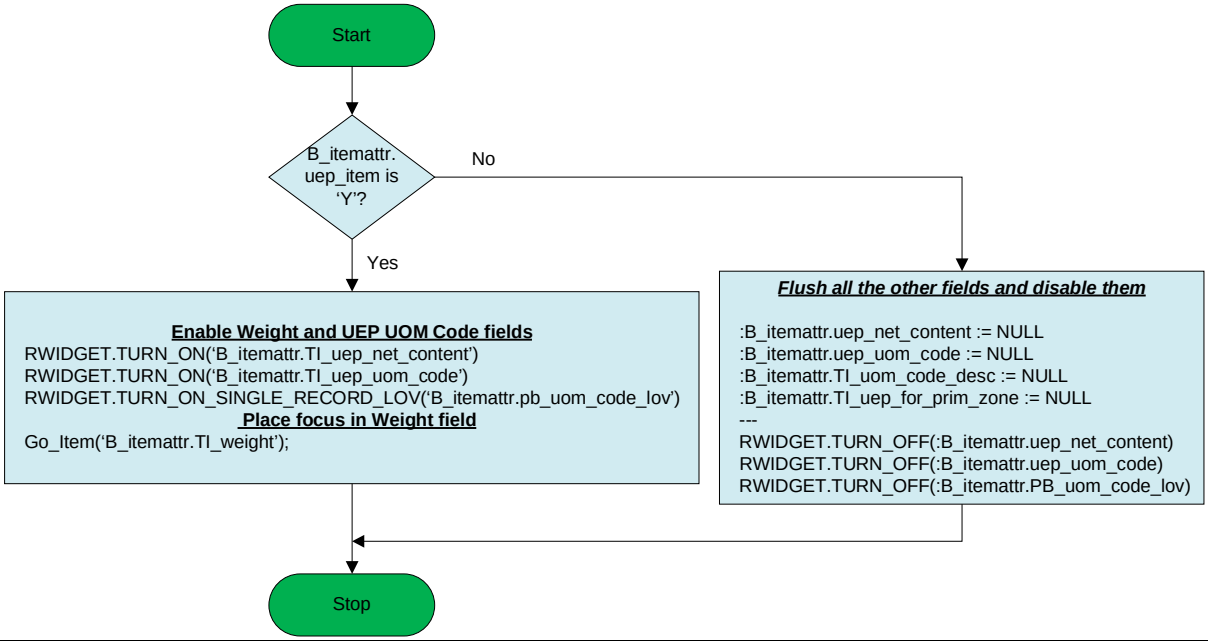
The field minimum length is just a sugges tion, the screen should look as much as possible to the screenshot displayed in the beginning of this chapter

§ Events

Event Name:	B_IM_UEP à NB_UEP_ITEM à WHEN-LIST-CHANGED [New]
Description:	This event is triggered when the UEP Item list is opened and a n option is selected .
Parameters	None
Returns:	None

Algorithm

The following changes must be placed on the existing algorithm (changes identified in black, and existing algorithm in red):



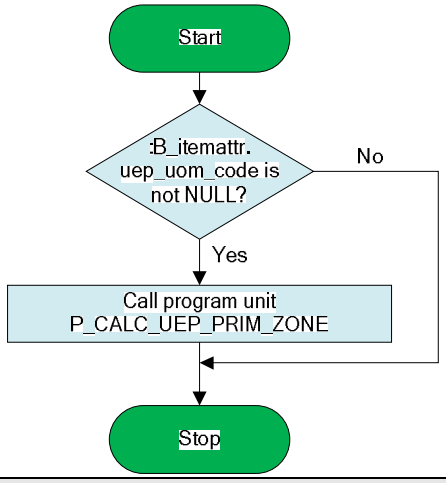
Development details

Exception handling:	Already defined
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable as parameter, and then raises the exception FORM_TRIGGER_FAIL URE.

Event Name:	B_IM_UEP à NB_UEP_NET_CONTENT à WHEN-VALIDATE-ITEM [New]
Description:	This event is triggered when user enters data in the Net Content field and moves to some other field.
Parameters	None
Returns:	None

Algorithm

The following changes must be placed on the existing algorithm (changes identified in black, and existing algorithm in red):

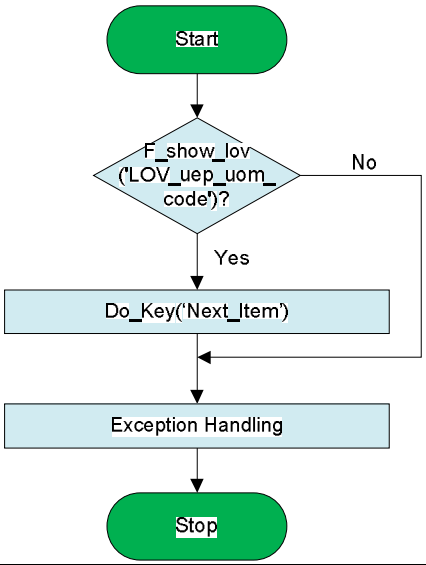


Development details	
Exception handling:	Already defined
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable as parameter, and then raises the exception FORM_TRIGGER_FAILURE.

Event Name:	B_IM_UEP à NB_UEP_UOM_CODE à KEY-LISTVAL [New]
Description:	This event is triggered to show up the LOV.
Parameters	None
Returns:	None

Algorithm

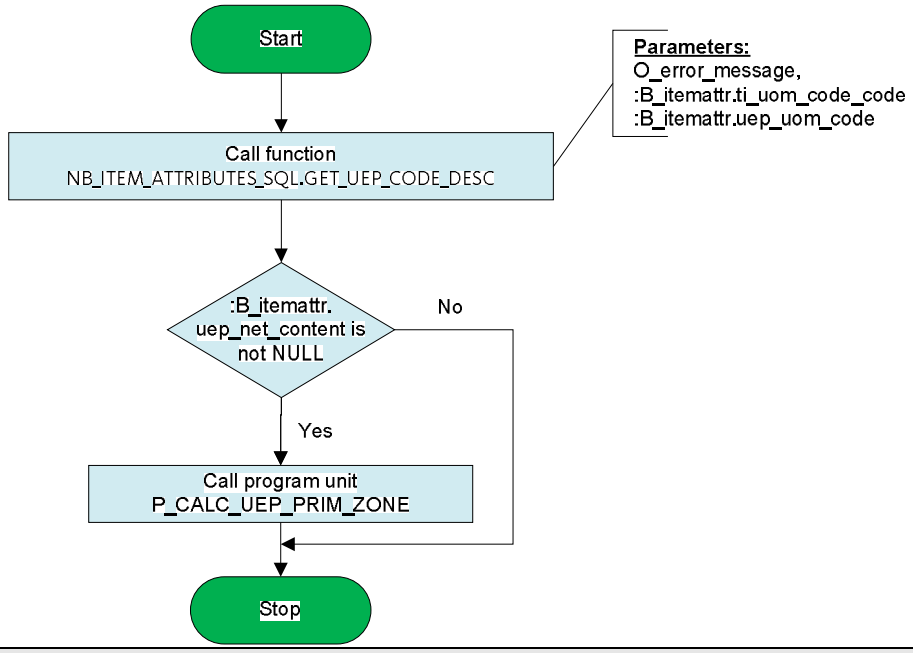
The following changes must be placed on the existing algorithm (changes identified in black, and existing algorithm in red):



Development details	
Exception handling:	Already defined

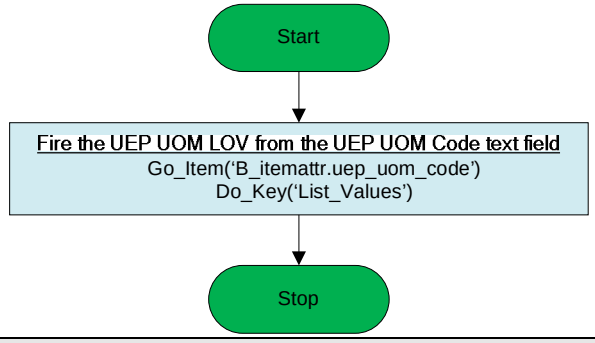
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable as parameter, and then raises the exception FORM_TRIGGER_FAILURE.

Event Name:	B_IM_UEP à NB_UEP_UOM_CODE à WHEN-VALIDATE-ITEM [New]
Description:	This event is triggered when user enters data in the UEP UOM Code field and moves to some other field.
Parameters	None
Returns:	None
Algorithm	
The following changes must be placed on the existing algorithm (changes identified in black, and existing algorithm in red):	



Development details	
Exception handling:	Already defined
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable as parameter, and then raises the exception FORM_TRIGGER_FAILURE.

Event Name:	B_IM_UEP à PB_UOM_CODE_LOV à WHEN-BUTTON-PRESSED [New]
Description:	This event is triggered when the UEP UOM Code LOV button is pressed.
Parameters	None
Returns:	None
Algorithm	
The following changes must be placed on the existing algorithm (changes identified in black, and existing alg orithm in red):	

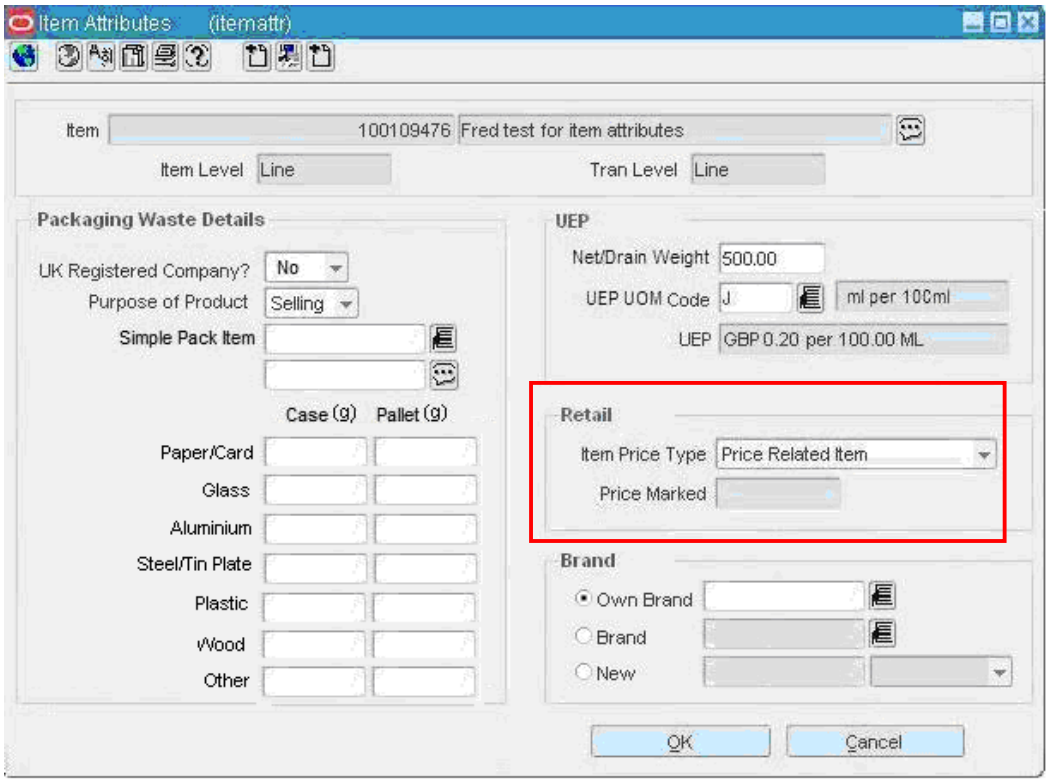


Development details	
Exception handling:	Already defined
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_err or_message variable as parameter, and then raises the exception FORM_TRIGGER_FAILURE.

§ Form Labels/Prompts

The developer is responsible to generate the script to feed suitable form labels/prompts to the database through a script which inserts into FORM_ELEMENTS and FORM_ELEMENTS_LANG tables.

This screen will be modified to accommodate the requirements of several modifications, in particular Mod #39 (Unit Equivalent Price UEP), Mod #40(Price Marked Items), Mod #70 (Price Related Items) and Mod #211 (Product Attributes Integration). Only the changes required for Mod #40 will be me ntioned here. It is assumed all modification required for Mod 211 have been implemented prior.



The Item Attributes screen will be modified to accept the Item Price Type details and for Price Marked Items the corresponding “marked price”. When the OK button is clicked validation of the data entry will occur and when successful the retail values (selling unit retail, standard unit retail etc) will be modified to equal the “marked price”.

- Parameters

  No new parameters are added.

- Canvases

The C_ITEMATTR canvas will be modified to add the ‘Retail’ section, and to include the new Items.

§ Items

Block	Item	Item Type	Canvas	Data Type	Minimum lenght	Insert/ Update	Required	Database Item	FORMAT MASK
B_ITEMATTR	LI_PRICE_TYPE	List Item	C_ITEMATTR	CHAR(30)	30	Yes/Yes	Yes	PRICE_TYPE	
B_ITEMATTR	TI_PRICE_MARKED	Text Item	C_ITEMATTR	NUMBER(20,2)	10	Yes/Yes	Yes	PRICE_MARKED	FM999G990D90
B_ITEMATTR	DI_RETAIL	Display Item	C_ITEMATTR	CHAR(30)	N/A	No/No	Yes		

§ Blocks

No change required

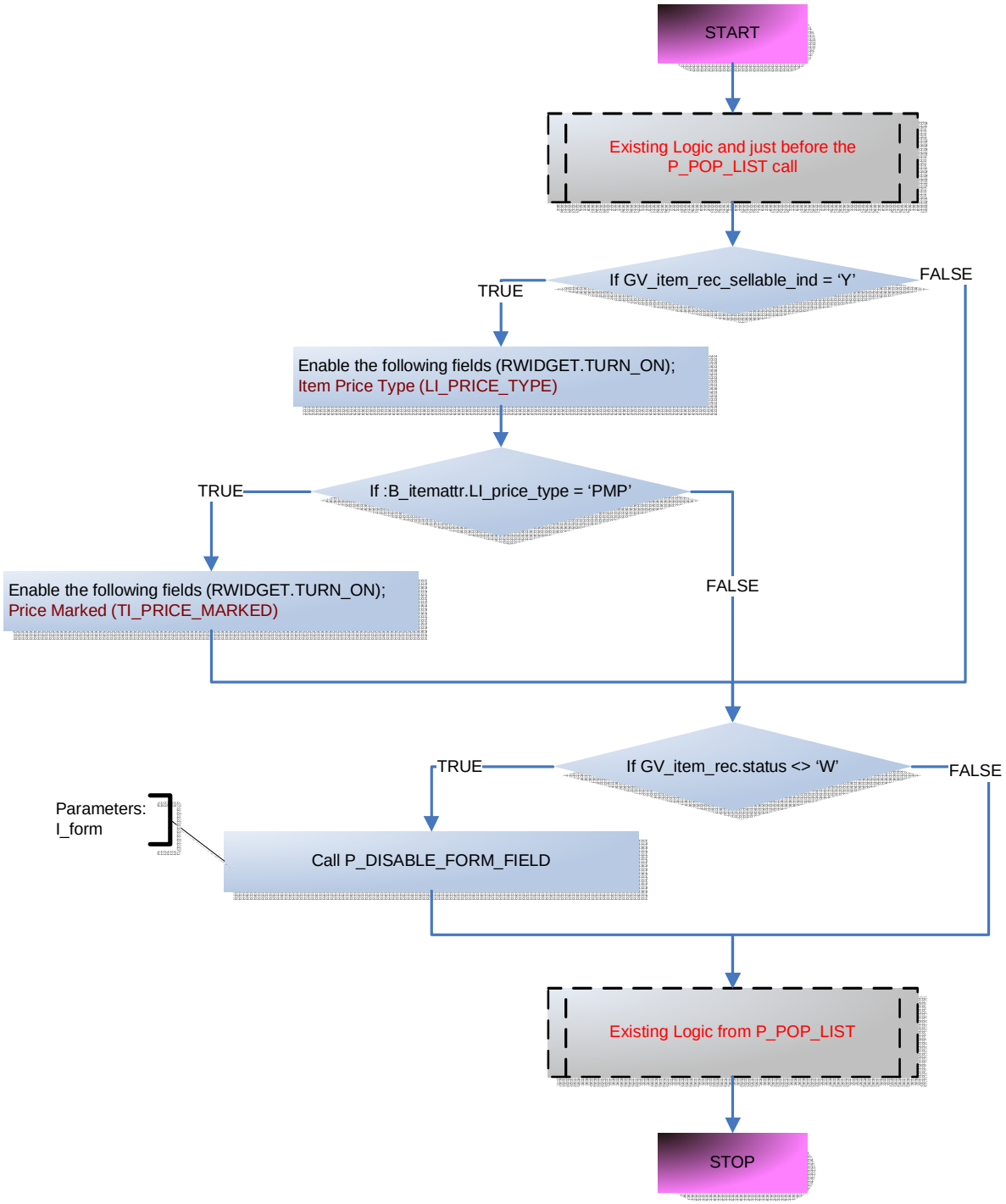
§ Program Units

PROGRAM UNIT NAME:	P_VIEW [EXISTING]			
Description:	This program unit will be amended to disable the 'Item Price Type' and 'Price Marked' fields			
Returns:	BOOLEAN			
Parameters				
Name	Datatype	IN/OUT	Default	Description
N/A				
Algorithm				
Development details				
Insert the following additional details; RWIDGET.TURN_OFF('B_itemattr. LI_price_type'); RWIDGET.TURN_OFF('B_itemattr. TI_price_marked');				
Exception handling:	n/a			
Note:				

PROGRAM UNIT NAME:	P_POP_LIST [EXISTING]			
Description:	This program unit will be amended to populate the Item Price Type list			
Returns:	BOOLEAN			
Parameters				
Name	Datatype	IN/OUT	Default	Description
N/A				
Algorithm				
Development details				
Insert the following additional details; P_POPULATE(':B_itemat tr.LI_price_type','PTYE')				
Exception handling:	n/a			
Note:				

PROGRAM UNIT NAME:	P_EDIT_NEW [EXISTING]			
Description:	This program unit will be amended to enable the Item Price Type and Marked Price fields and to enable the 'Price Marked' field for a super user when the item is not in worksheet status.			
Returns:	BOOLEAN			
Parameters				
Name	Datatype	IN/OUT	Default	Description
N/A				
Algorithm				

The following changes must be placed on the existing algorithm (existing logic is highlighted in red font);



Development details	
Exception handling:	n/a

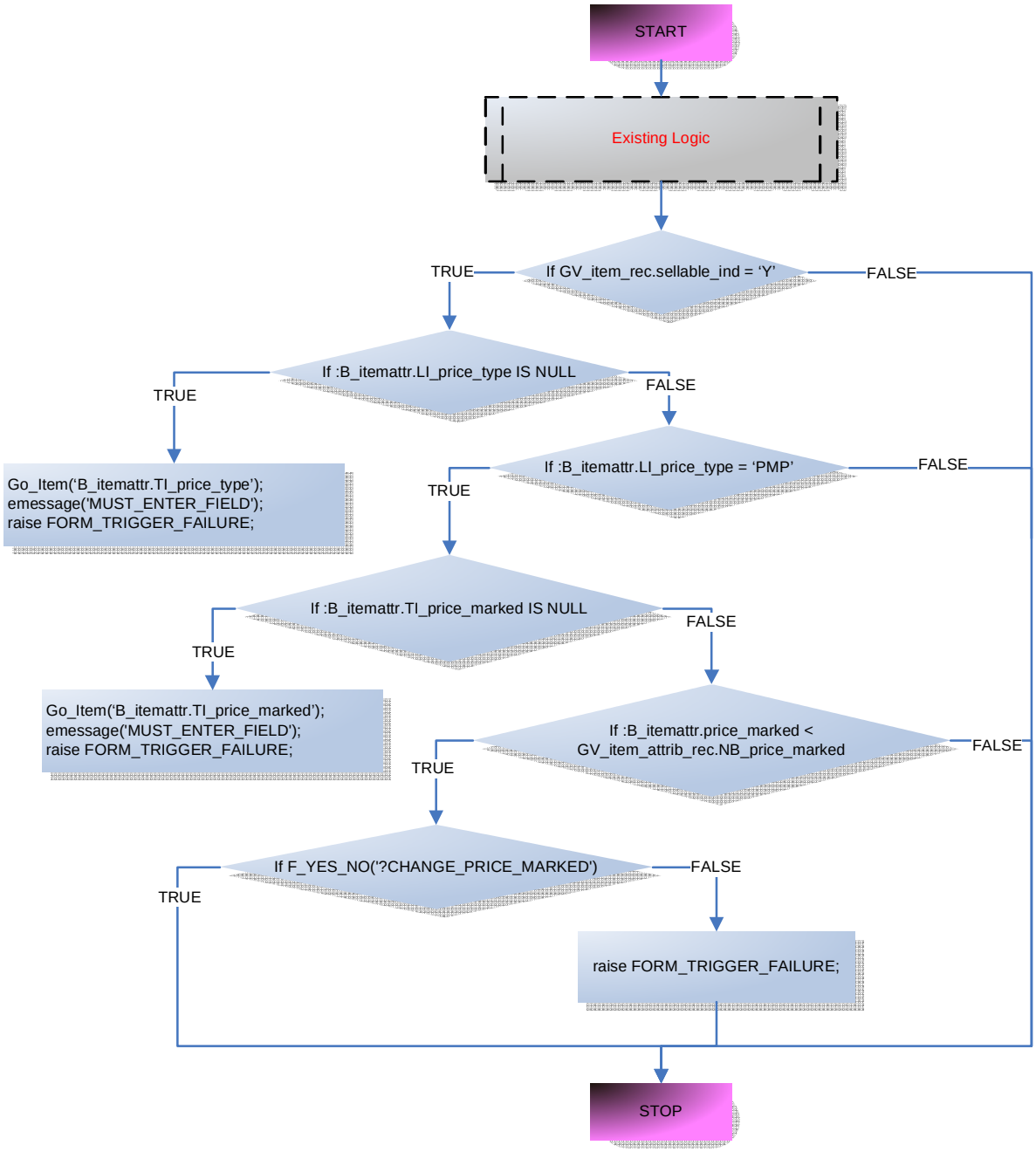
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable back as parameter, and then raises the exception FORM_TRIGGER_FAILURE. It's required to attach the nb_retailer.pll library to the form

PROGRAM UNIT NAME:	P_CHECK_REQUIRED [EXISTING]
Description:	This program unit will be modified to not allow the user to save the I tem Attributes information without providing the 'Item Price Type' details. Furthermore, when the user selects Item Price Type of 'Price Marked Item', the user cannot exit the form by the OK button without providing a "marked price" value. Also, where a superuser updates the "marked price" to a lower value for a price marked item, a warning message will be displayed to confirm the change.
Returns:	BOOLEAN

Parameters				
Name	Datatype	IN/OUT	Default	Description
N/A				

Algorithm

The following changes must be placed on the existing algorithm (existing logic is highlighted in red font);



Development details
As GV_item_attrib_rec.NB_price_marked can be null, use nvl(GV_item_attrib_rec.NB_price_marked, 0) in the above if condition .

Exception handling:	n/a
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable back as parameter, and then raises the exception FORM_TRIG GER_FAILURE.

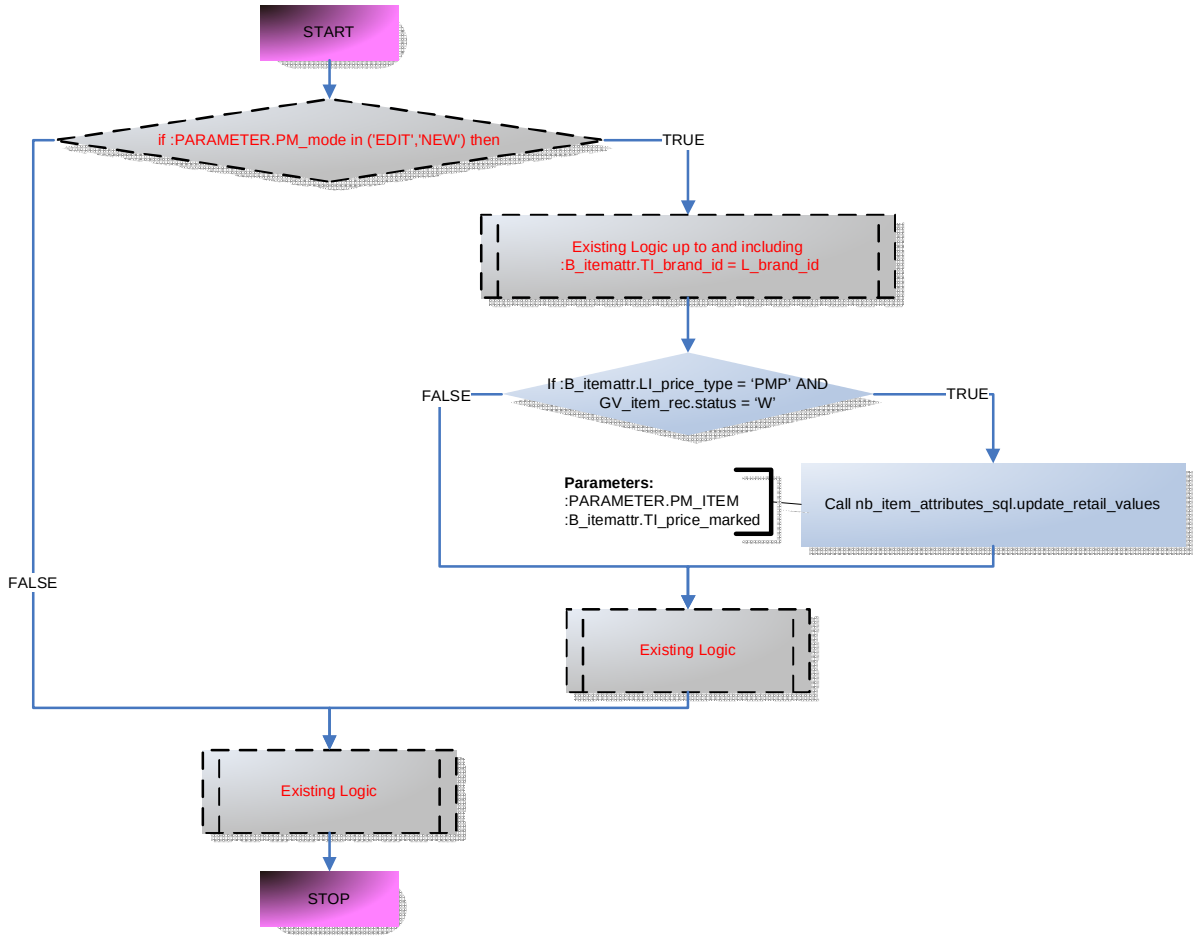
PROGRAM UNIT NAME:	P_EXIT.SAVE_FORM [EXISTING]
Description:	This program unit will be modified to call the NB_ITEM_ATTRIBUTES_SQL.UPDATE_RETAIL_VALUES for price marked items, which will update the retail by zone values to equal the “marked price”, for the base default record, and therefore update the retail values for the item.
Returns:	BOOLEAN

Parameters

Name	Datatype	IN/OUT	Default	Description
N/A				

Algorithm

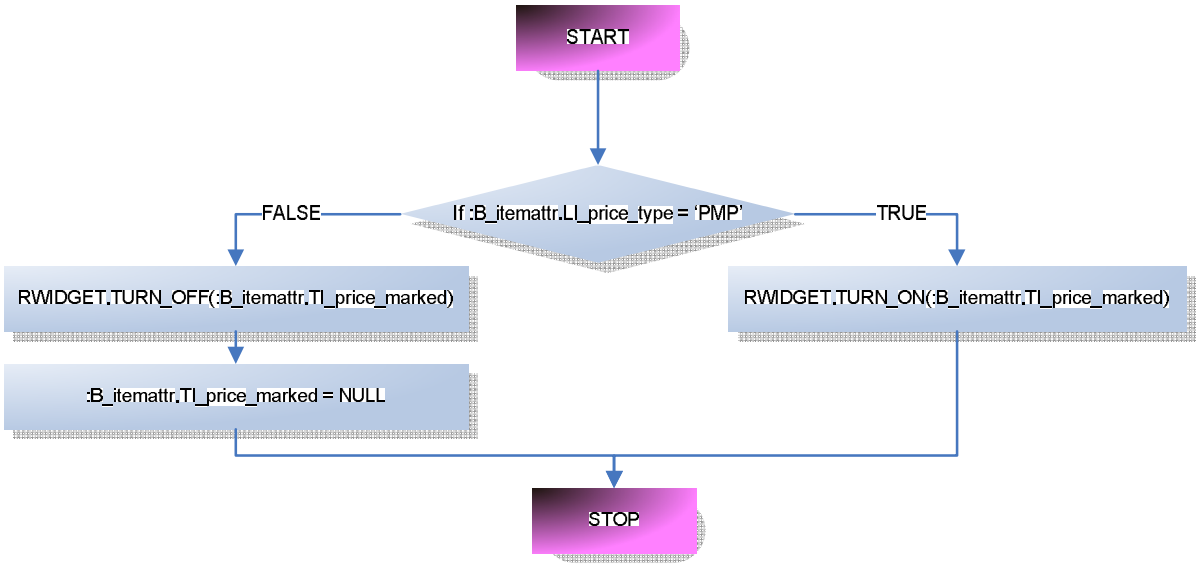
The following changes must be placed on the existing algorithm (existing logic is highlighted in red font);



Development details

Exception handling:	n/a
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of t he L_error_message variable back as parameter, and then raises the exception FORM_TRIGGER_FAILURE.

§ Triggers

PROGRAM UNIT NAME:	WHEN-LIST-CHANGED
Item	:B_itemattr.LI_price_type
Description:	Will enable and disable the price marked text box depending on the item price type.
Parameters:	None
Returns:	BOOLEAN
Algorithm	
	
Development details	
Note, Item Price Type will only be enabled in edit mode for items with worksheet status.	
Exception handling:	n/a
Note:	

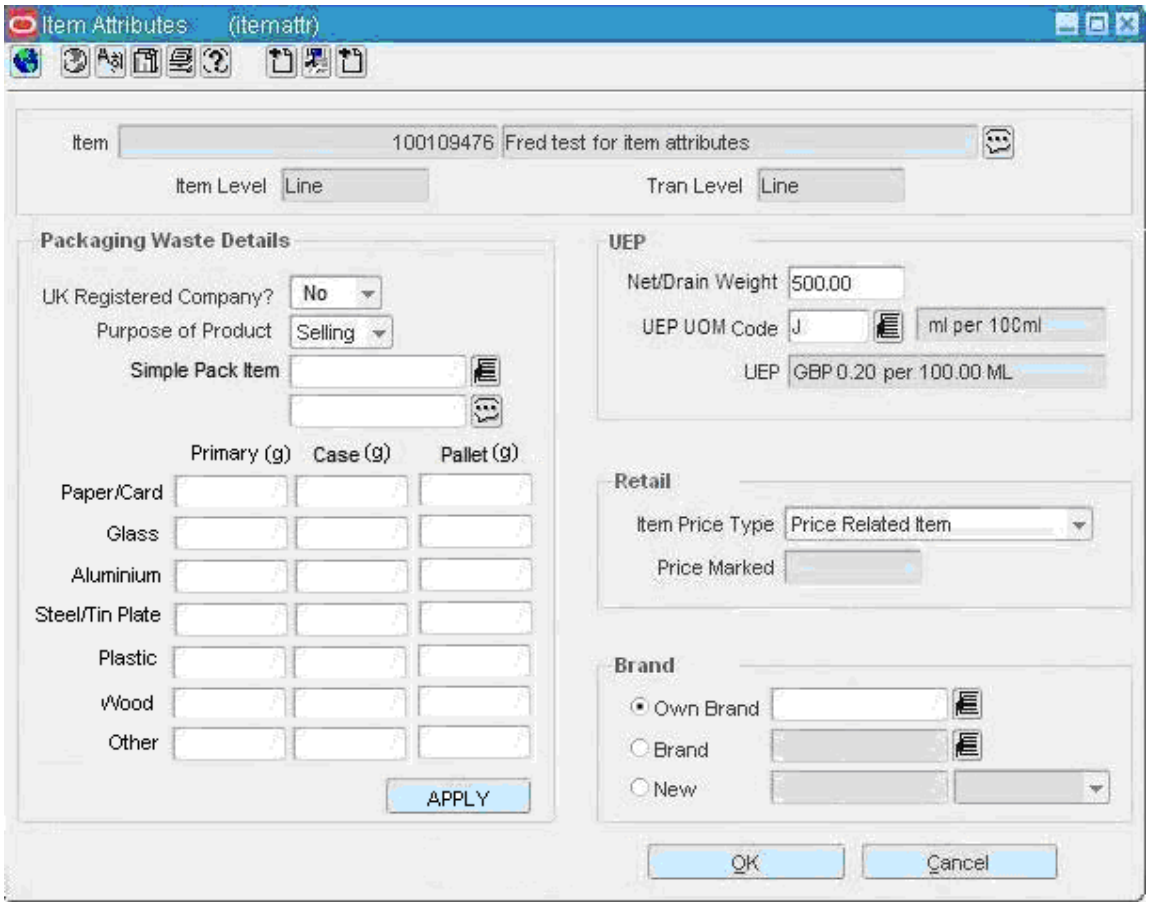
§ LOVs

No change required

§ Record Groups

No change required

The Item Attributes screen will be modified to remove the existing fields and insert new fields corresponding to Mod 211.



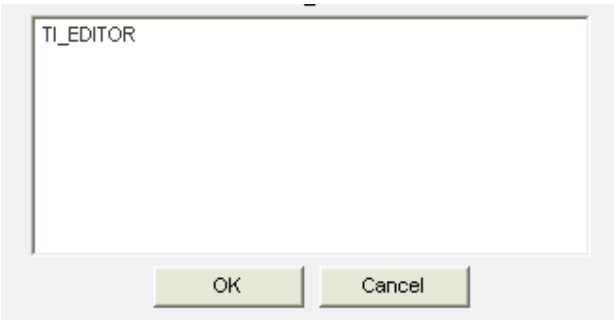
For tab sequence see appendix.

§ Parameters

No new parameters are added.

§ Canvases

C_EDITOR canvas to look as follows;



§ Blocks

No change required

§ Items

NO.	BLOCK	ITEM	ITEM TYPE	CANVAS	DATA TYPE	MINIMUM LENGTH	INSERT/ UPDATE	DATABASE ITEM	DEFAULT VALUE	COMMENTS
1	B_ITEMATTR	LI_UK_REG_COMPANY	List Item	C_ITEMATTR	CHAR(3)	3	Yes/Yes	UK_COMPANY		
2	B_ITEMATTR	LI_PRODUCT_PURPOSE	List Item	C_ITEMATTR	CHAR(10)	10	Yes/Yes	PRODUCT_PURPOSE		
3	B_ITEMATTR	TI_SIMPLE_PACK	Text Item	C_ITEMATTR	VARCHAR2(25)	10				
4	B_ITEMATTR	PB_LOV_SIMPLE_PACK	Push Button	C_ITEMATTR		N/A				
5	B_ITEMATTR	TI_SIMPLE_PACK_DESC	Text Item	C_ITEMATTR	VARCHAR2(250)	30				
6	B_ITEMATTR	PB_SIMPLE_PACK_DESC	Push Button	C_ITEMATTR		N/A				
7	B_ITEM_ATTR	DI_PAPER_CARD	Display Item	C_ITEMATTR	VARCHAR2(20)	N/A				
8	B_ITEM_ATTR	DI_GLASS	Display Item	C_ITEMATTR	VARCHAR2(20)	N/A				
9	B_ITEMATTR	DI_ALUMINIUM	Display Item	C_ITEMATTR	VARCHAR2(20)	N/A				
10	B_ITEMATTR	DI_STEEL_TIN_PLATE	Display Item	C_ITEMATTR	VARCHAR2(20)	N/A				
11	B_ITEMATTR	DI_PLASTIC	Display Item	C_ITEMATTR	VARCHAR2(20)	N/A				
12	B_ITEMATTR	DI_WOOD	Display Item	C_ITEMATTR	VARCHAR2(20)	N/A				
13	B_ITEMATTR	DI_OTHER	Display Item	C_ITEMATTR	VARCHAR2(20)	N/A				
14	B_ITEMATTR	DI_PRIMARY	Display Item	C_ITEMATTR	VARCHAR2(20)	N/A				

15	B_ITEMATTR	DI_CASE	Display Item	C_ITEMATTR	VARCHAR2(20)	N/A				
16	B_ITEMATTR	DI_PALLET	Display Item	C_ITEMATTR	VARCHAR2(20)	N/A				
17	B_ITEMATTR	TI_PAPER_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	PAPER_CARD		
18	B_ITEMATTR	TI_GLASS_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	GLASS		
19	B_ITEMATTR	TI_ALUMINIUM_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	ALUMINIUM		
20	B_ITEMATTR	TI_STEEL_TIN_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	STEEL_TIN_PLATE		
21	B_ITEMATTR	TI_PLASTIC_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	PLASTIC		
22	B_ITEMATTR	TI_WOOD_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	WOOD		
23	B_ITEMATTR	TI_OTHER_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	OTHER		
24	B_ITEMATTR	TI_PAPER_CASE_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	PAPER_CARD_CASE		
25	B_ITEMATTR	TI_GLASS_CASE_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	GLASS_CASE		
26	B_ITEMATTR	TI_ALUMINIUM_CASE_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	ALUMINIUM_CASE		
27	B_ITEMATTR	TI_STEEL_TIN_CASE_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	STEEL_TIN_PLATE_CASE		
28	B_ITEMATTR	TI_PLASTIC_CASE_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	PLASTIC_CASE		
29	B_ITEMATTR	TI_WOOD_CASE__DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	WOOD_CASE		
30	B_ITEMATTR	TI_OTHER_CASE_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	OTHER_CASE		
31	B_ITEMATTR	TI_PAPER_PALLET_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	PAPER_CARD_PALLET		
32	B_ITEMATTR	TI_GLASS_PALLET_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	GLASS_PALLET		
33	B_ITEMATTR	TI_ALUMINIUM_PALLET_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	ALUMINIUM_PALLET		
34	B_ITEMATTR	TI_STEEL_TIN_PALLET_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	STEEL_TIN_PLATE_PALLET		
35	B_ITEMATTR	TI_PLASTIC_PALLET_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	PLASTIC_PALLET		
36	B_ITEMATTR	TI_WOOD_PALLET__DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	WOOD_PALLET		
37	B_ITEMATTR	TI_OTHER_PALLET_DTL	Text Item	C_ITEMATTR	NUMBER(6)	6	Yes/Yes	OTHER_PALLET		

38	B_ITEMATTR	RG_BRAND_TYPE	Radio Group	C_ITEMATTR		N/A				
39	B_ITEMATTR	RB_OWN_BRAND	Radio Button	C_ITEMATTR	VARCHAR2(20)	N/A			'O'	
40	B_ITEMATTR	RB_BRAND	Radio Button	C_ITEMATTR	VARCHAR2(20)	N/A			'B'	
41	B_ITEMATTR	RB_NEW_BRAND	Radio Button	C_ITEMATTR	VARCHAR2(20)	N/A			'N'	
42	B_ITEMATTR	PB_LOV_OWN_BRAND	Push Button	C_ITEMATTR		N/A				
43	B_ITEMATTR	PB_LOV_BRAND	Push Button	C_ITEMATTR		N/A				
44	B_ITEMATTR	LI_BRAND_TYPE	List Item	C_ITEMATTR	VARCHAR2(1)	N/A				
45	B_ITEMATTR	TI_BRAND_ID	Text Box	C_ITEMATTR	NUMBER(4)	N/A	Yes/Yes	BRAND_ID		Field is hidden from user/screen
46	B_ITEMATTR	TI_OWN_BRAND	Text Item	C_ITEMATTR	VARCHAR2(30)	20				
47	B_ITEMATTR	TI_BRAND	Text Item	C_ITEMATTR	VARCHAR2(30)	20				
48	B_ITEMATTR	TI_NEW_BRAND	Text Item	C_ITEMATTR	VARCHAR2(30)	N/A				
49	B_ITEM_ATTR	DI_PACKAGE_WAST_DTL	Display Item	C_ITEMATTR	VARCHAR2(20)	N/A				
50	B_ITEM_ATTR	DI_BRAND	Display Item	C_ITEMATTR	VARCHAR2(20)	N/A				
51	B_ACTION	PB_APPLY	Push Button	C_ITEMATTR	VARCHAR2(10)	N/A				

The field minimum length is just a suggestion, the screen should look as much as possible to the screenshot displayed in the beginning of this chapter

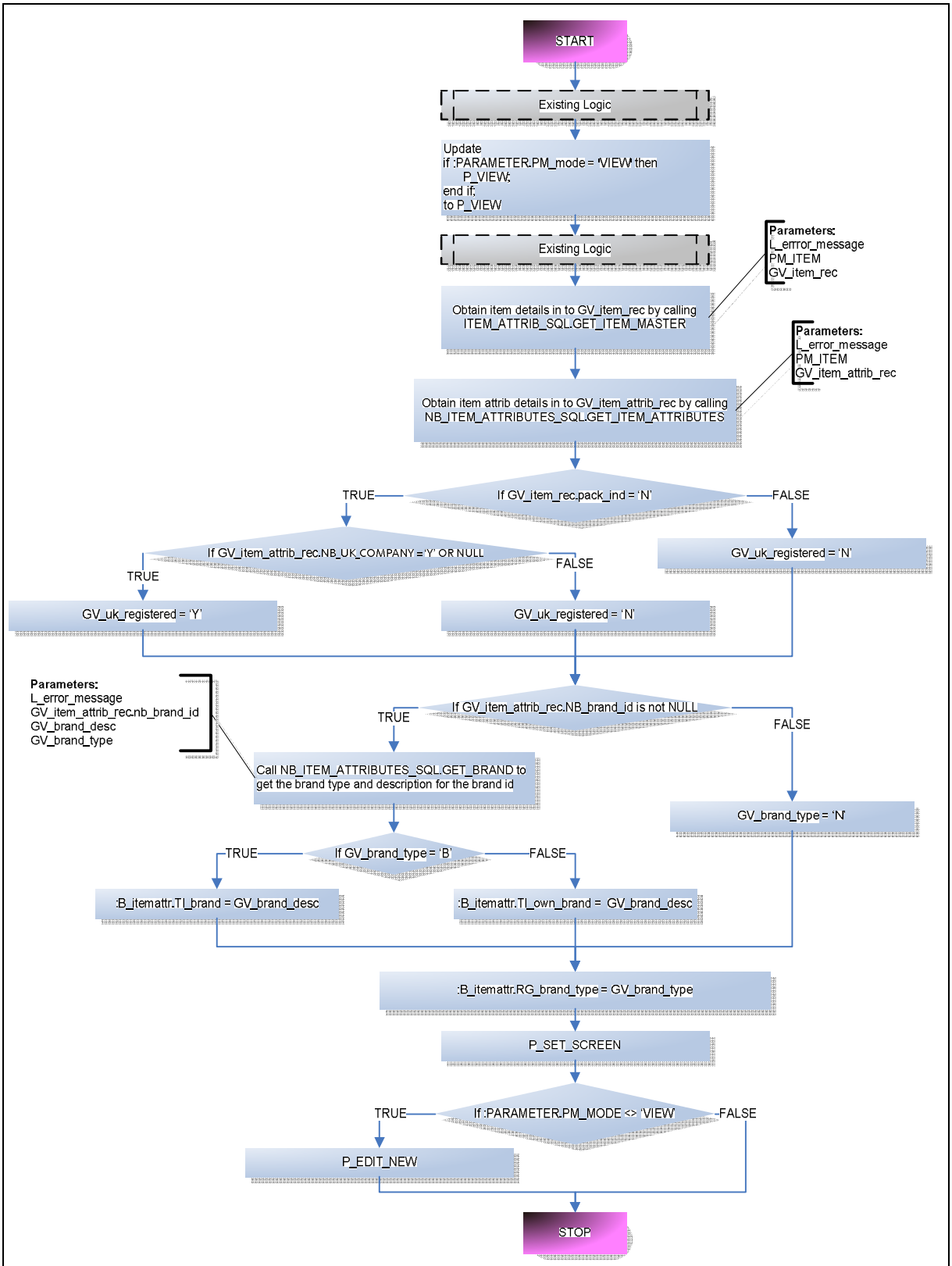
The corresponding mapping of the items on the Item Attributes screen is visualized in the appendix.

§ Program Units

P_EDITOR package from fm_edit.fmb library inserted too.

PROGRAM UNIT NAME:	INTERNAL_VARIABLES [NEW]
Description:	Declare the global variables
Parameters:	none
Returns:	BOOLEAN
Algorithm	
Development details	
GV_item_rec ITEM_MASTER%ROWTYPE GV_item_attrib_rec NB_ITEM_ATTRIBUTES%ROWTYPE GV_uk_registered VARCHAR2(1) GV_brand_desc NB_BRAND.BRAND_DESC%TYPE GV_brand_type VARCHAR2(1) GV_primary_req VARCHAR2(1) = 'N' GV_case_req VARCHAR2(1) = 'N' GV_pallet_req VARCHAR2(1) = 'N' GV_waste_dtl_rec NB_ITEM_ATTRIBUTES_SQL.WASTE_REC%TYPE	
Exception handling:	No change
Note:	

PROGRAM UNIT NAME:	P_FORM_STARTUP [EXISTING]
Description:	This program unit will be amended to obtain item data from ITEM_MASTER and NB_ITEM_ATTRIBUTES, and call the program units to set the screen look
Parameters:	No Change
Returns:	BOOLEAN
Algorithm	

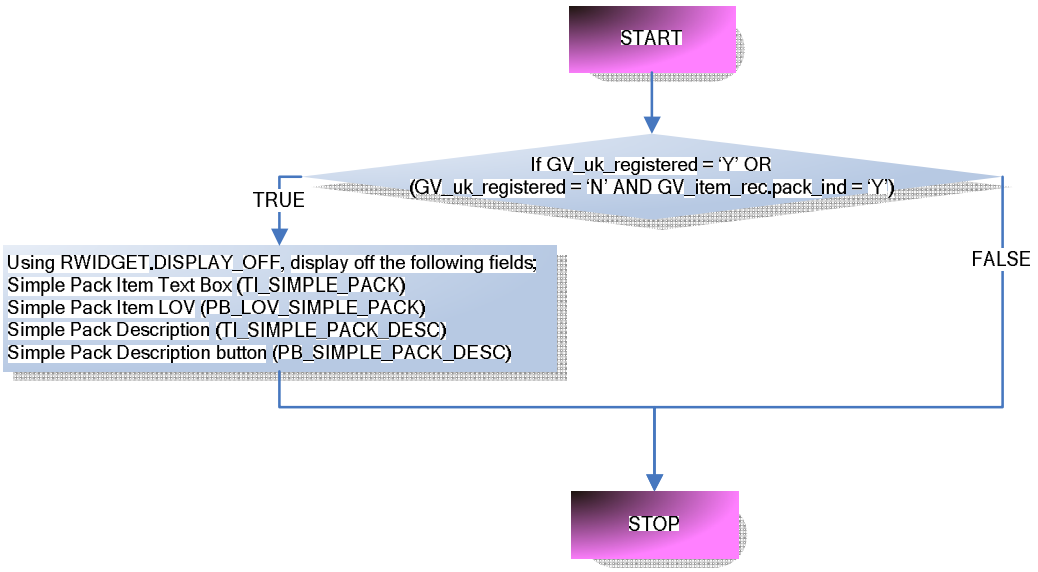


Development details
Where brand type is null it is defau lted to 'N' for New Brand

Exception handling:	No change
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable back as parameter, and then raises the exception FORM_TRIGGER_FAILURE.

PROGRAM UNIT NAME:	P_SET_SCREEN [NEW]
Description:	This program unit will display off fields which are not applicable for the Item
Parameters:	none
Returns:	BOOLEAN

Algorithm

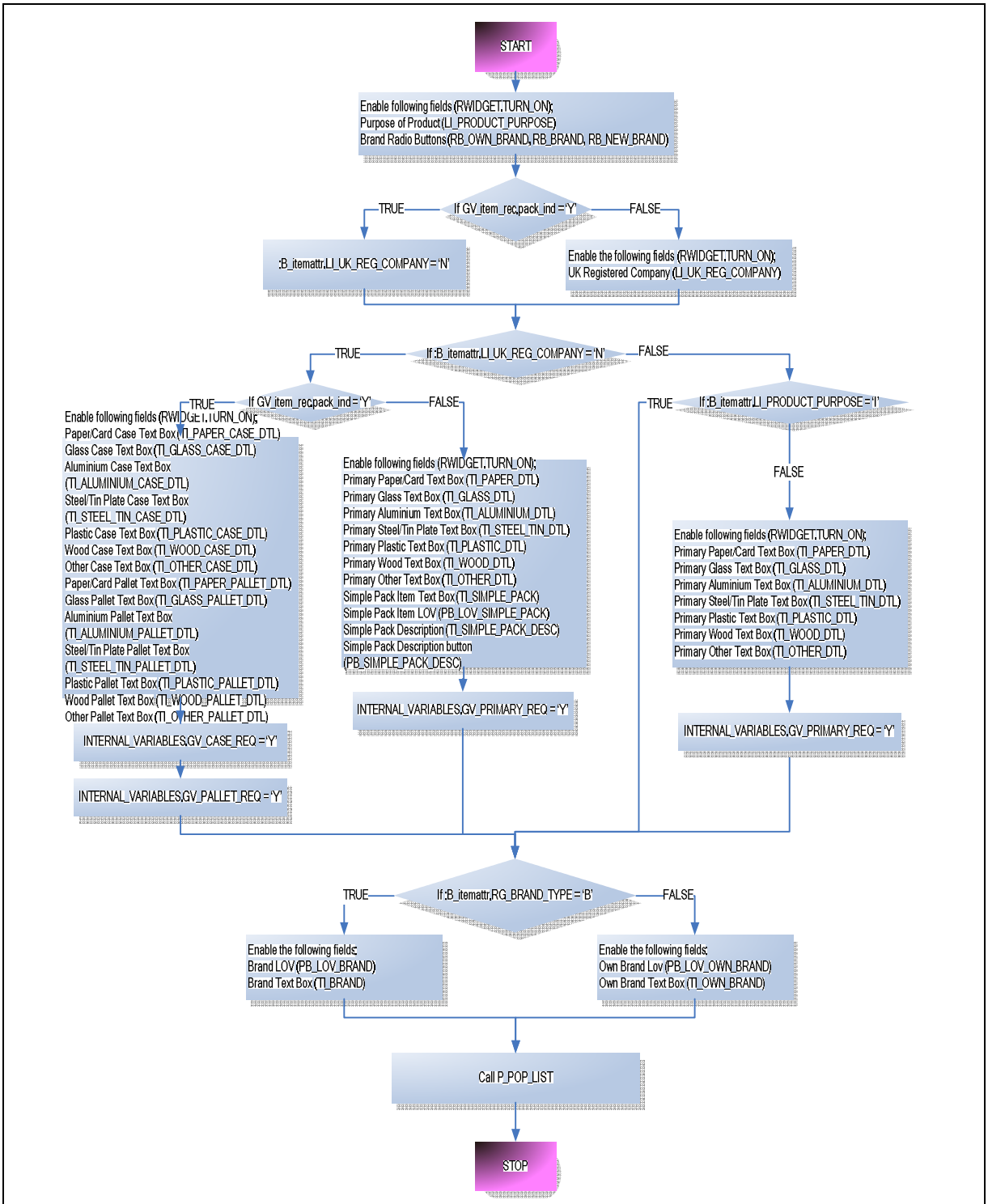


Development details

Exception handling:	No change
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable back as parameter, and then raises the exception FORM_TRIGGER_FAILURE.

PROGRAM UNIT NAME:	P_EDIT_NEW [NEW]
Description:	This program unit will be executed when in New or Edit mode to enable the relevant form fields
Parameters:	none
Returns:	BOOLEAN

Algorithm

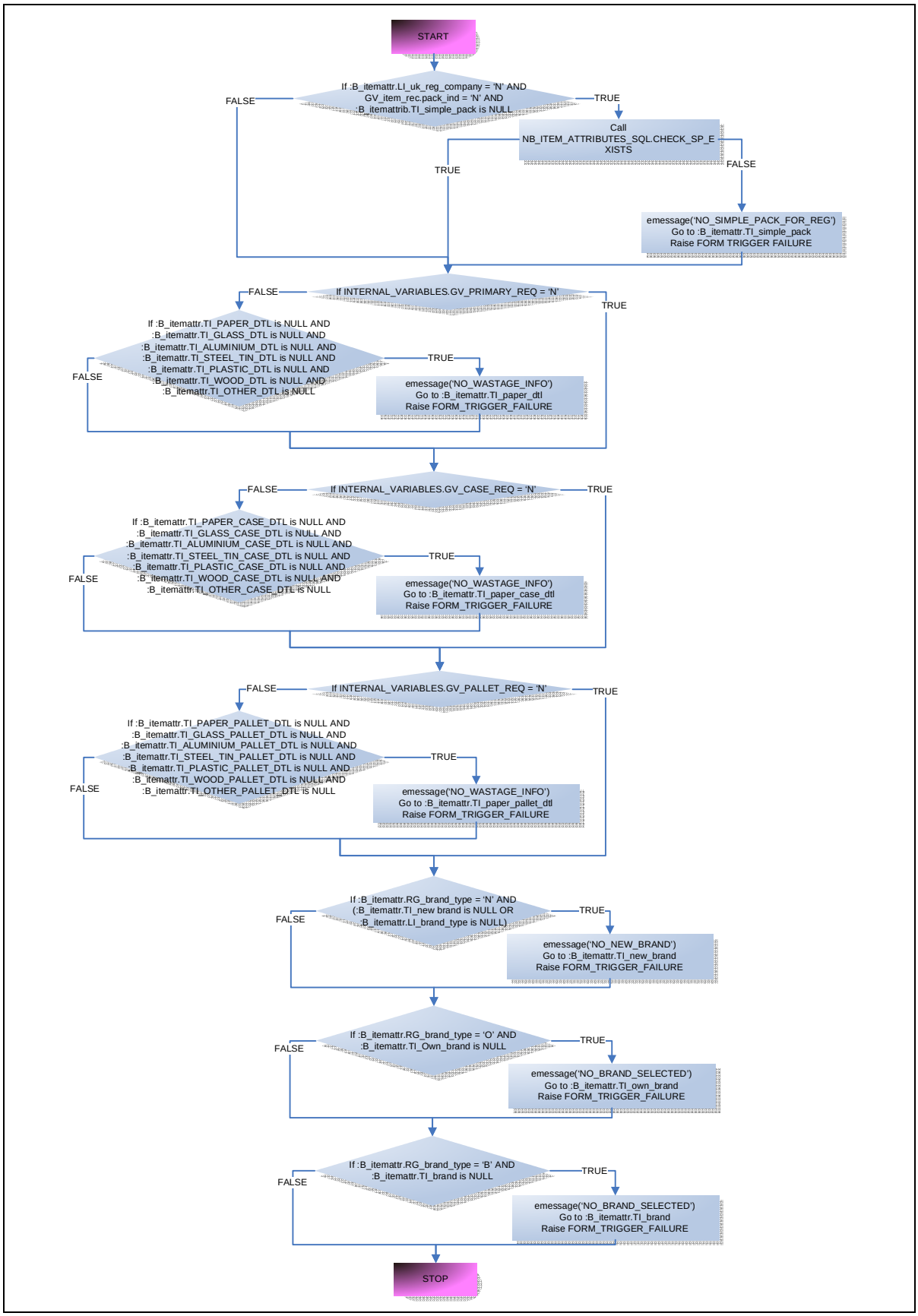


Development details	
Exception handling:	No change

Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable back as parameter, and then raises the exception FORM_TRIGGER_FAILURE.

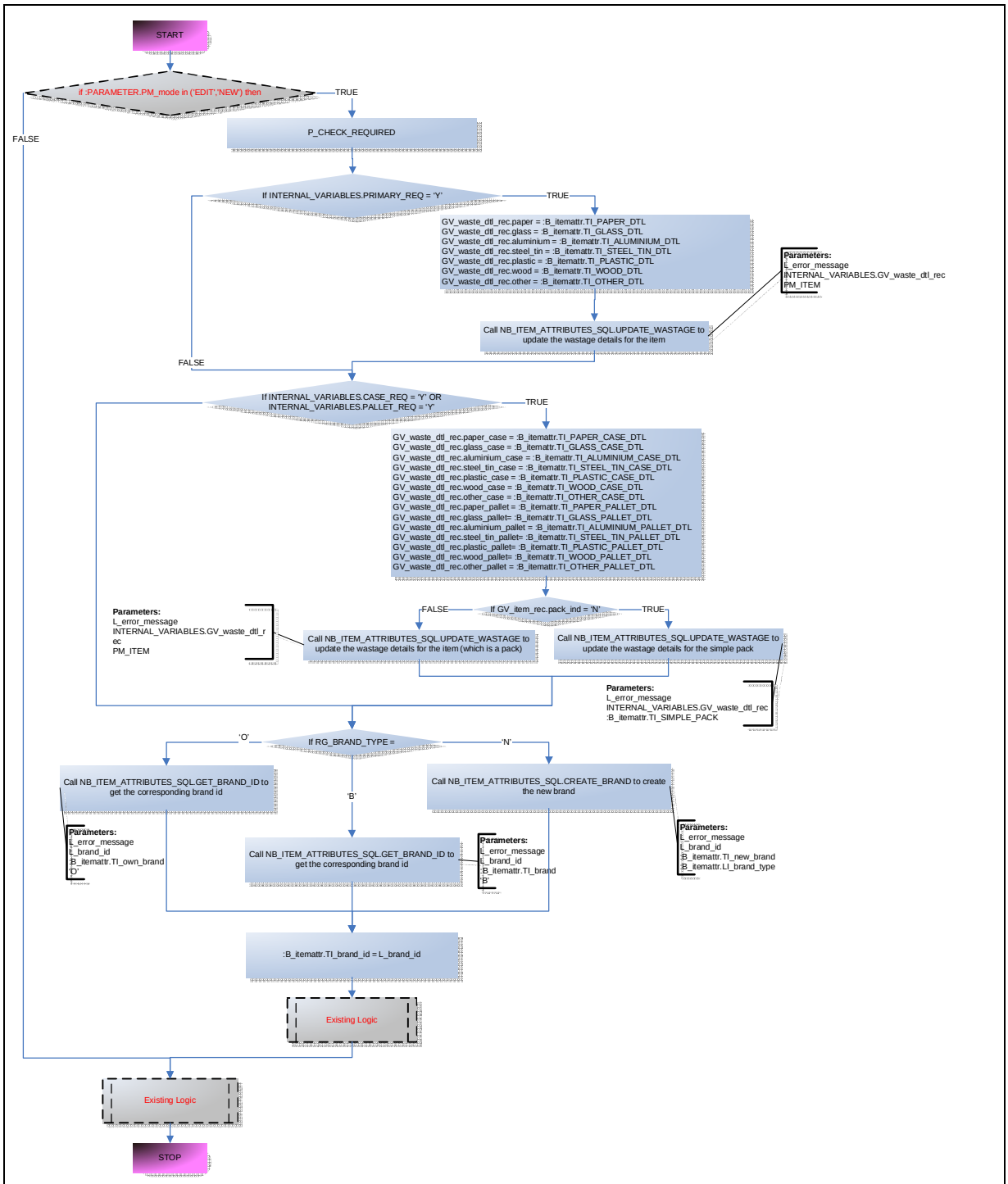
PROGRAM UNIT NAME:	P_POP_LISTS [NEW]
Description:	This program unit will populate all list items
Parameters:	none
Returns:	BOOLEAN
Algorithm	
Development details	
Using P_POPULATE_LIST to populate the following list Items; :B_itemattr.LI_uk_reg_company, YSNO :B_itemattr.LI_product_purpose, IAPP :B_itemattr.LI_brand_type, IABT	
Exception handling:	No change
Note:	Every call to a PL/SQL function should validate the retu rn value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable back as parameter, and then raises the exception FORM_TRIGGER_FAILURE.

PROGRAM UNIT NAME:	P_CHECK_REQUIRED [NE W]
Description:	This program unit w ill validate the entries
Parameters:	None
Returns:	BOOLEAN
Algorithm	


START
If :B_itemattr.LI_uk_reg_company = 'N' AND GV_item_rec.pack_ind = 'N' AND :B_itemattrib.TI_simple_pack is NULL
FALSE
TRUE
Call NB_ITEM_ATTRIBUTES_SQL.CHECK_SP_EXISTS
TRUE
FALSE
emessage('NO_SIMPLE_PACK_FOR_REG')
Go to :B_itemattr.TI_simple_pack
Raise FORM TRIGGER FAILURE
If INTERNAL_VARIABLES.GV_PRIMARY_REQ = 'N'
FALSE
TRUE
If :B_itemattr.TI_PAPER_DTL is NULL AND :B_itemattr.TI_GLASS_DTL is NULL AND :B_itemattr.TI_ALUMINIUM_DTL is NULL AND :B_itemattr.TI_STEEL_TIN_DTL is NULL AND :B_itemattr.TI_PLASTIC_DTL is NULL AND :B_itemattr.TI_WOOD_DTL is NULL AND :B_itemattr.TI_OTHER_DTL is NULL
TRUE
FALSE
emessage('NO_WASTAGE_INFO')
Go to :B_itemattr.TI_paper_dtl
Raise FORM_TRIGGER_FAILURE
If INTERNAL_VARIABLES.GV_CASE_REQ = 'N'
FALSE
TRUE
If :B_itemattr.TI_PAPER_CASE_DTL is NULL AND :B_itemattr.TI_GLASS_CASE_DTL is NULL AND :B_itemattr.TI_ALUMINIUM_CASE_DTL is NULL AND :B_itemattr.TI_STEEL_TIN_CASE_DTL is NULL AND :B_itemattr.TI_PLASTIC_CASE_DTL is NULL AND :B_itemattr.TI_WOOD_CASE_DTL is NULL AND :B_itemattr.TI_OTHER_CASE_DTL is NULL
TRUE
FALSE
emessage('NO_WASTAGE_INFO')
Go to :B_itemattr.TI_paper_case_dtl
Raise FORM_TRIGGER_FAILURE
If INTERNAL_VARIABLES.GV_PALLET_REQ = 'N'
FALSE
TRUE
If :B_itemattr.TI_PAPER_PALLET_DTL is NULL AND :B_itemattr.TI_GLASS_PALLET_DTL is NULL AND :B_itemattr.TI_ALUMINIUM_PALLET_DTL is NULL AND :B_itemattr.TI_STEEL_TIN_PALLET_DTL is NULL AND :B_itemattr.TI_PLASTIC_PALLET_DTL is NULL AND :B_itemattr.TI_WOOD_PALLET_DTL is NULL AND :B_itemattr.TI_OTHER_PALLET_DTL is NULL
TRUE
FALSE
emessage('NO_WASTAGE_INFO')
Go to :B_itemattr.TI_paper_pallet_dtl
Raise FORM_TRIGGER_FAILURE
If :B_itemattr.RG_brand_type = 'N' AND (:B_itemattr.TI_new brand is NULL OR :B_itemattr.LI_brand_type is NULL)
FALSE
TRUE
emessage('NO_NEW_BRAND')
Go to :B_itemattr.TI_new_brand
Raise FORM_TRIGGER_FAILURE
If :B_itemattr.RG_brand_type = 'O' AND :B_itemattr.TI_Own_brand is NULL
FALSE
TRUE
emessage('NO_BRAND_SELECTED')
Go to :B_itemattr.TI_own_brand
Raise FORM_TRIGGER_FAILURE
If :B_itemattr.RG_brand_type = 'B' AND :B_itemattr.TI_brand is NULL
FALSE
TRUE
emessage('NO_BRAND_SELECTED')
Go to :B_itemattr.TI_brand
Raise FORM_TRIGGER_FAILURE
STOP

Development details	
Exception handling:	No change
Note:	Every call to a PL/SQL function s hould validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable back as parameter, and then raises the exception FORM_TRIGGER_FAILURE.

PROGRAM UNIT NAME:	P_EXIT.SAVE_FORM [EX ISTING]
Description:	This program unit will be amended to call the P_CHECK_REQUIRED program unit
Parameters:	none
Returns:	BOOLEAN
Algorithm	



Development details

Exception handling:	No change
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable back as parameter, and then raises the exception FORM_TRIGGER_FAILURE.

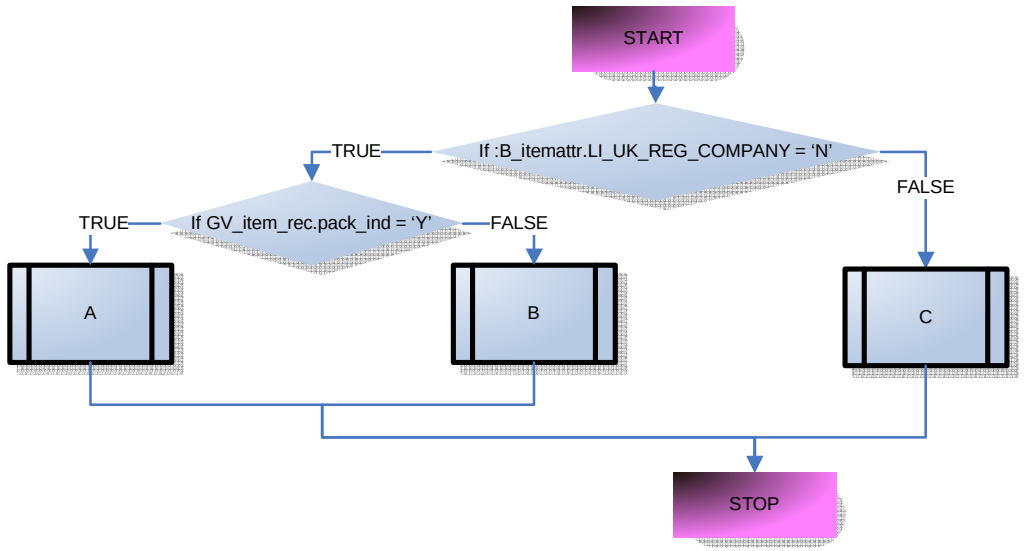
PROGRAM UNIT NAME:	P_VIEW [EXISTING]
Description:	This program unit will be updated to disable the new fields
Parameters:	none
Returns:	BOOLEAN

Algorithm

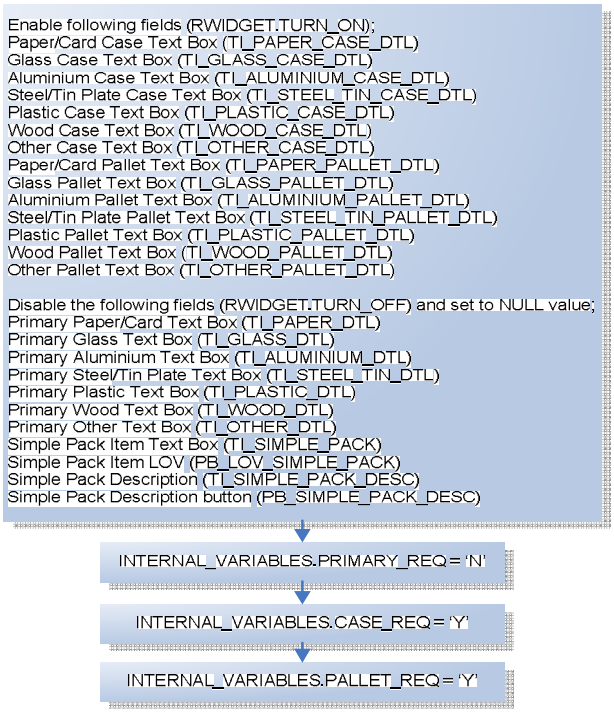
Development details
Using RWIDGET.TURN_OFF, disable the following fields; :B_itemattr.LI_uk_reg_company :B_itemattr.LI_product_purpose :B_itemattr.TI_simple_pack :B_itemattr.PB_lov_simple_pack :B_itemattr.TI_simple_pack_desc :B_itemattr.PB_simple_pack_desc :B_itemattr.TI_paper_dtl :B_itemattr.TI_glass_dtl :B_itemattr.TI_aluminium_dtl :B_itemattr.TI_steel_tin_dtl :B_itemattr.TI_plastic_dtl :B_itemattr.TI_wood_dtl :B_itemattr.TI_other_dtl :B_itemattr.TI_paper_case_dtl :B_itemattr.TI_glass_case_dtl :B_itemattr.TI_aluminium_ case_dtl :B_itemattr.TI_steel_tin_ case_dtl :B_itemattr.TI_plastic_case_dtl :B_itemattr.TI_wood_case_dtl :B_itemattr.TI_other_ case_dtl :B_itemattr.TI_paper_pallet_dtl :B_itemattr.TI_glass_pallet_dtl :B_itemattr.TI_aluminium_pallet_dtl :B_itemattr.TI_steel_tin_pallet_dtl :B_itemattr.TI_plastic_pallet_dtl :B_itemattr.TI_wood_pallet__dtl :B_itemattr.TI_other_p allet_dtl :B_itemattr.RB_own_brand :B_itemattr.RB_brand :B_itemattr.RB_new_brand :B_itemattr.PB_lov_own_brand :B_itemattr.PB_lov_brand :B_itemattr.LI_brand_type :B_itemattr.TI_brand_id :B_itemattr.TI_own_brand :B_itemattr.TI_brand :B_itemattr.TI_new_brand :B_action.PB_apply

Exception handling:	No change
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable back as parameter, and then raises t he exception FORM_TRIGGER_FAILURE.

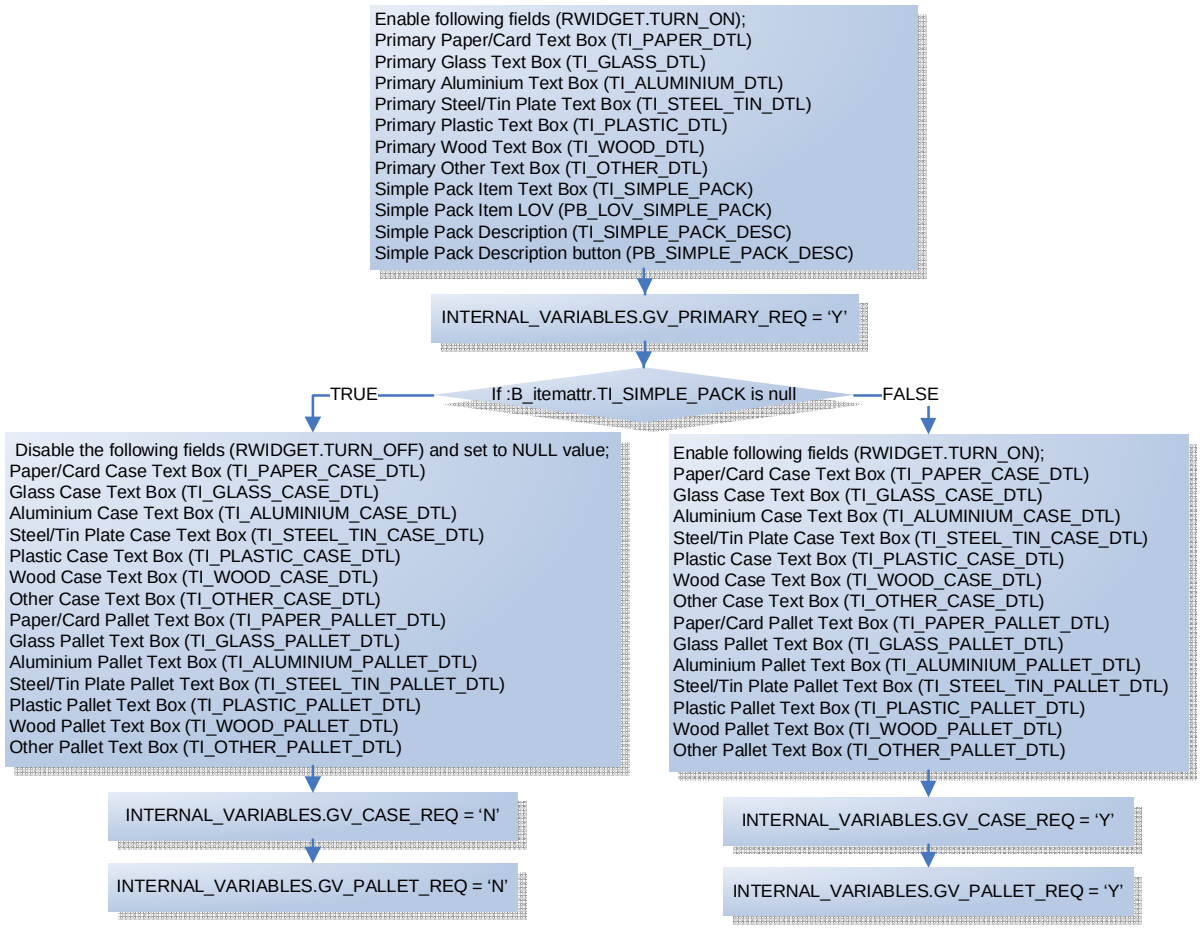
PROGRAM UNIT NAME:	P_UPDATE_SCREEN [NEW]
Description:	This program unit will be called to update the screen, enabling or disabling fields to correspond to the selection chosen by the user.
Parameters:	none
Returns:	BOOLEAN
Algorithm	



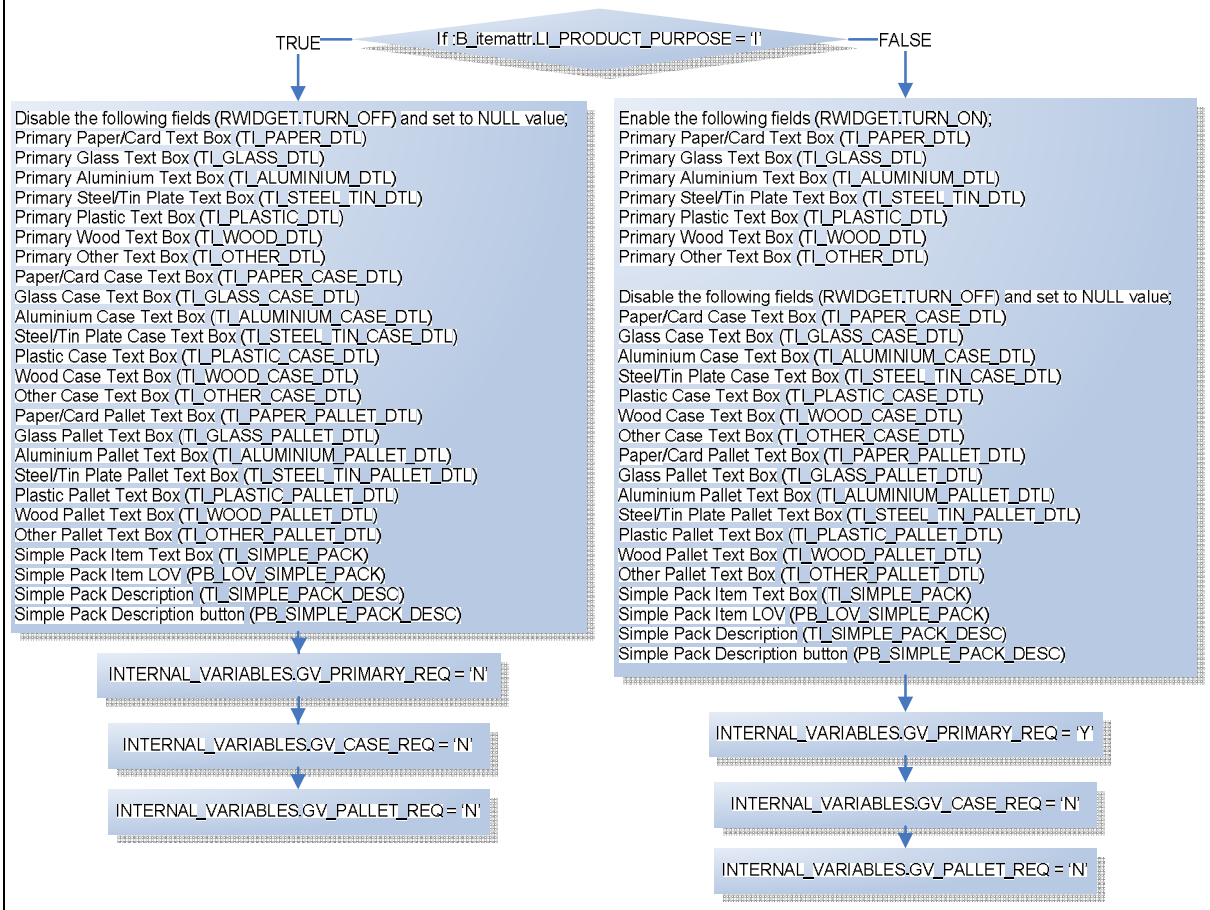
Where A is;


Enable following fields (RWIDGET.TURN_ON);
Paper/Card Case Text Box (TI_PAPER_CASE_DTL)
Glass Case Text Box (TI_GLASS_CASE_DTL)
Aluminium Case Text Box (TI_ALUMINIUM_CASE_DTL)
Steel/Tin Plate Case Text Box (TI_STEEL_TIN_CASE_DTL)
Plastic Case Text Box (TI_PLASTIC_CASE_DTL)
Wood Case Text Box (TI_WOOD_CASE_DTL)
Other Case Text Box (TI_OTHER_CASE_DTL)
Paper/Card Pallet Text Box (TI_PAPER_PALLET_DTL)
Glass Pallet Text Box (TI_GLASS_PALLET_DTL)
Aluminium Pallet Text Box (TI_ALUMINIUM_PALLET_DTL)
Steel/Tin Plate Pallet Text Box (TI_STEEL_TIN_PALLET_DTL)
Plastic Pallet Text Box (TI_PLASTIC_PALLET_DTL)
Wood Pallet Text Box (TI_WOOD_PALLET_DTL)
Other Pallet Text Box (TI_OTHER_PALLET_DTL)
Disable the following fields (RWIDGET.TURN_OFF) and set to NULL value;
Primary Paper/Card Text Box (TI_PAPER_DTL)
Primary Glass Text Box (TI_GLASS_DTL)
Primary Aluminium Text Box (TI_ALUMINIUM_DTL)
Primary Steel/Tin Plate Text Box (TI_STEEL_TIN_DTL)
Primary Plastic Text Box (TI_PLASTIC_DTL)
Primary Wood Text Box (TI_WOOD_DTL)
Primary Other Text Box (TI_OTHER_DTL)
Simple Pack Item Text Box (TI_SIMPLE_PACK)
Simple Pack Item LOV (PB_LOV_SIMPLE_PACK)
Simple Pack Description (TI_SIMPLE_PACK_DESC)
Simple Pack Description button (PB_SIMPLE_PACK_DESC)
INTERNAL_VARIABLES.PRIMARY_REQ = 'N'
INTERNAL_VARIABLES.CASE_REQ = 'Y'
INTERNAL_VARIABLES.PALLET_REQ = 'Y'

Where B is;


Enable following fields (RWIDGET.TURN_ON);
Primary Paper/Card Text Box (TI_PAPER_DTL)
Primary Glass Text Box (TI_GLASS_DTL)
Primary Aluminium Text Box (TI_ALUMINIUM_DTL)
Primary Steel/Tin Plate Text Box (TI_STEEL_TIN_DTL)
Primary Plastic Text Box (TI_PLASTIC_DTL)
Primary Wood Text Box (TI_WOOD_DTL)
Primary Other Text Box (TI_OTHER_DTL)
Simple Pack Item Text Box (TI_SIMPLE_PACK)
Simple Pack Item LOV (PB_LOV_SIMPLE_PACK)
Simple Pack Description (TI_SIMPLE_PACK_DESC)
Simple Pack Description button (PB_SIMPLE_PACK_DESC)
INTERNAL_VARIABLES.GV_PRIMARY_REQ = 'Y'
If :B_itemattr.TI_SIMPLE_PACK is null
TRUE
FALSE
Disable the following fields (RWIDGET.TURN_OFF) and set to NULL value;
Paper/Card Case Text Box (TI_PAPER_CASE_DTL)
Glass Case Text Box (TI_GLASS_CASE_DTL)
Aluminium Case Text Box (TI_ALUMINIUM_CASE_DTL)
Steel/Tin Plate Case Text Box (TI_STEEL_TIN_CASE_DTL)
Plastic Case Text Box (TI_PLASTIC_CASE_DTL)
Wood Case Text Box (TI_WOOD_CASE_DTL)
Other Case Text Box (TI_OTHER_CASE_DTL)
Paper/Card Pallet Text Box (TI_PAPER_PALLET_DTL)
Glass Pallet Text Box (TI_GLASS_PALLET_DTL)
Aluminium Pallet Text Box (TI_ALUMINIUM_PALLET_DTL)
Steel/Tin Plate Pallet Text Box (TI_STEEL_TIN_PALLET_DTL)
Plastic Pallet Text Box (TI_PLASTIC_PALLET_DTL)
Wood Pallet Text Box (TI_WOOD_PALLET_DTL)
Other Pallet Text Box (TI_OTHER_PALLET_DTL)
INTERNAL_VARIABLES.GV_CASE_REQ = 'N'
INTERNAL_VARIABLES.GV_PALLET_REQ = 'N'
Enable following fields (RWIDGET.TURN_ON);
Paper/Card Case Text Box (TI_PAPER_CASE_DTL)
Glass Case Text Box (TI_GLASS_CASE_DTL)
Aluminium Case Text Box (TI_ALUMINIUM_CASE_DTL)
Steel/Tin Plate Case Text Box (TI_STEEL_TIN_CASE_DTL)
Plastic Case Text Box (TI_PLASTIC_CASE_DTL)
Wood Case Text Box (TI_WOOD_CASE_DTL)
Other Case Text Box (TI_OTHER_CASE_DTL)
Paper/Card Pallet Text Box (TI_PAPER_PALLET_DTL)
Glass Pallet Text Box (TI_GLASS_PALLET_DTL)
Aluminium Pallet Text Box (TI_ALUMINIUM_PALLET_DTL)
Steel/Tin Plate Pallet Text Box (TI_STEEL_TIN_PALLET_DTL)
Plastic Pallet Text Box (TI_PLASTIC_PALLET_DTL)
Wood Pallet Text Box (TI_WOOD_PALLET_DTL)
Other Pallet Text Box (TI_OTHER_PALLET_DTL)
INTERNAL_VARIABLES.GV_CASE_REQ = 'Y'
INTERNAL_VARIABLES.GV_PALLET_REQ = 'Y'

Where C is;



Development details	
Exception handling:	No change
Note:	Every call to a PL/SQL function should validate the return value. If is FALSE, invokes the EMESSAGE instruction, passing the value of the L_error_message variable back as parameter, and then raises the exception FORM_TRIGGER_FAILURE.

§ Triggers

PROGRAM UNIT NAME:	WHEN-LIST-CHANGED
Item	:B_itemattr.LI_uk_reg_company
Description:	
Parameters:	None
Returns:	BOOLEAN
Algorithm	

Development details	
Call P_UPDATE_SCREEN	
Exception handling:	n/a
Note:	

PROGRAM UNIT NAME:	WHEN-LIST-CHANGED
Item	:B_itemattr.LI_product_purpose
Description:	
Parameters:	None
Returns:	BOOLEAN
Algorithm	
Development details	
Call P_UPDATE_SCREEN	
Exception handling:	n/a
Note:	

PROGRAM UNIT NAME:	KEY-DOWN
Item	RG_BRAND_TYPE
Description:	
Parameters:	None
Returns:	BOOLEAN
Algorithm	
Development details	

```
If :B_itemattr.RG_brand_type = 'O' then
    :B_itemattr.RG_brand_type = 'B'
    RDWIDGET.TURN_ON(:B_itemattr.TI_BRAND)
    RDWIDGET.TURN_ON(:B_itemattr.LOV_BRAND)
    RWIDGET.TURN_OFF(:B_itemattr.TI_OWN_BRAND )
    RWIDGET.TURN_OFF(:B_itemattr.LOV_OWN_BRAND)
    :B_itemattr.TI_own_brand = NULL
    :B_itemattr.TI_brand = NULL

    If GV_item_rec.NB_brand_type = 'B'
        :B_itemattr.TI_brand = GV_item_rec.NB_brand

If :B_itemattr.RG_br and_type = 'B' then
    :B_itemattr.RG_brand_type = 'N'
    RDWIDGET.TURN_ON(:B_itemattr.TI_NEW_BRAND)
    RDWIDGET.TURN_ON(:B_itemattr.LI_BRAND_TYPE)
    RWIDGET.TURN_OFF(:B_itemattr.TI_BRAND)
    RWIDGET.TURN_OFF(:B_itemattr.LOV_BRAN  D);
    :B_itemattr.TI_brand = NULL
```

Exception handling:	n/a
Note:	

PROGRAM UNIT NAME:	KEY-UP
Item	RG_BRAND_TYPE
Description:	
Parameters:	None
Returns:	BOOLEAN
Algorithm	
Development details	

```
If :B_itemattr.RG_brand_type = 'B' then
    :B_itemattr.RG_brand_type = 'O'
    RDWIDGET.TURN_OFF(:B_itemattr.TI_BRAND)
    RDWIDGET.TURN_OFF(:B_itemattr.LOV_BRAND)
    RWIDGET.TURN_ON(:B_itemattr.TI_OWN_BRAND)
    RWIDGET.TURN_ON(:B_itemattr.LOV_OWN_BRAND)
    :B_itemattr.TI_own_brand = NULL
    :B_itemattr.TI_brand = NULL

    If GV_item_rec.NB_brand_type = 'O'
        :B_itemattr.TI_own_brand = GV_item_rec.NB_brand

If :B_itemattr.RG_brand_type = 'N' then
    :B_itemattr.RG_brand_type = 'B'
    RDWIDGET.TURN_OFF(:B_itemattr.TI_NEW_BRAND)
    RDWIDGET.TURN_OFF(:B_itemattr.LI_BRAND_TYPE)
    RWIDGET.TURN_ON(:B_itemattr.TI_BRAND)
    RWIDGET.TURN_ON(:B_itemattr.LOV_BRAND);
    :B_itemattr.TI_new_brand = NULL
    :B_itemattr.LI_brand_ type = NULL
    :B_itemattr.TI_brand = NULL

    If GV_item_rec.NB_brand_type = 'B'
        :B_itemattr.TI_brand = GV_item_rec.NB_brand
```

Exception handling:	n/a
Note:	

PROGRAM UNIT NAME:	WHEN-RADIO-CHANGED
Item	RG_BRAND_TYPE
Description:	
Parameters:	None
Returns:	BOOLEAN
Algorithm	
Development details	

```
If :B_itemattr.RG_brand_type = 'O' then
    RDWIDGET.TURN_OFF(:B_itemattr.TI_NEW_BRAND)
    RDWIDGET.TURN_OFF(:B_itemattr.LI_BRAND_TYPE)
    RDWIDGET.TURN_OFF(:B_ite mattr.TI_BRAND)
    RDWIDGET.TURN_OFF(:B_itemattr.LOV_BRAND)
    RWIDGET.TURN_ON(:B_itemattr.TI_OWN_BRAND)
    RWIDGET.TURN_ON(:B_itemattr.LOV_OWN_BRAND)
    :B_itemattr.TI_own_brand = NULL
    :B_itemattr.TI_brand = NULL
    :B_itemattr.TI_new_brand = NULL
    :B_itemattr.LI_brand_type = NULL

    If GV_item_rec.NB_brand_type = 'O'
        :B_itemattr.TI_own_brand = GV_item_rec.NB_brand

If :B_itemattr.RG_brand_type = 'B' then
    RDWIDGET.TURN_OFF(:B_itemattr.T I_NEW_BRAND)
    RDWIDGET.TURN_OFF(:B_itemattr.LI_BRAND_TYPE)
    RDWIDGET.TURN_ON(:B_itemattr.TI_BRAND)
    RDWIDGET.TURN_ON(:B_itemattr.LOV_BRAND)
    RWIDGET.TURN_OFF(:B_itemattr.TI_OWN_BRAND)
    RWIDGET.TURN_OFF(:B_itemattr.LOV_OWN  _BRAND)
    :B_itemattr.TI_own_brand = NULL
    :B_itemattr.TI_brand = NULL
    :B_itemattr.TI_new_brand = NULL
    :B_itemattr.LI_brand_type = NULL

    If GV_item_rec.NB_brand_type = 'B'
        :B_itemattr.TI_brand = GV_item  _rec.NB_brand

If :B_itemattr.RG_brand_type = 'N' then
    RDWIDGET.TURN_ON(:B_itemattr.TI_NEW_BRAND)
    RDWIDGET.TURN_ON(:B_itemattr.LI_BRAND_TYPE)
    RDWIDGET.TURN_OFF(:B_itemattr.TI_BRAND)
    RDWIDGET.TURN_OFF(:B_itemattr.LOV_BRAND)
    RWIDGET.TURN_OFF(:B_itemattr.TI_OWN_BRAND)
    RWIDGET.TURN_OFF(:B_itemattr.LOV_OWN_BRAND)
    :B_itemattr.TI_own_brand = NULL
    :B_itemattr.TI_brand = NULL
    :B_itemattr.TI_new_brand = NULL
    :B_itemattr.LI_brand_type = NU LL
```

Exception handling:	n/a
Note:	

PROGRAM UNIT NAME:	WHEN-BUTTON-PRESSED [NEW]
Item	PB_LOV_OWN_BRAND

Description:	
Parameters:	None
Returns:	BOOLEAN
Algorithm	
Development details	
Go_Item('B_itemattr.TI_own_brand'); Do_Key('List_Values');	
Exception handling:	n/a
Note:	

PROGRAM UNIT NAME:	KEY-LISTVAL [NEW]
Item	TI_own_brand
Description:	
Parameters:	None
Returns:	BOOLEAN
Algorithm	
Development details	

if F_SHOW_LOV('LOV_own_brand') then
Go_Item('B_itemattr.PB_OK');
end if;	

Exception handling:	n/a
Note:	

PROGRAM UNIT NAME:	WHEN-BUTTON-PRESSED [NEW]
Item	PB_LOV_BRAND
Description:	
Parameters:	None
Returns:	BOOLEAN
Algorithm	
Development details	
Go_Item('B_itemattr.TI_own_brand'); Do_Key('List_Values');	

Exception handling:	n/a
Note:	

PROGRAM UNIT NAME:	KEY-LISTVAL [NEW]
Item	TI_brand
Description:	
Parameters:	None
Returns:	BOOLEAN
Algorithm	
Development details	

if F_SHOW_LOV('LOV_brand ') then
Go_Item('B_itemattr.PB_OK');
end if;	

Exception handling:	n/a
Note:	

PROGRAM UNIT NAME:	WHEN-BUTTON-PRESSED [NEW]
Item	PB_LOV_SIMPLE_PACK
Description:	
Parameters:	None
Returns:	BOOLEAN
Algorithm	
Development details	
Go_Item('B_itemattr.TI_simple_pack'); Do_Key('List_Values');	
Exception handling:	n/a
Note:	

PROGRAM UNIT NAME:	KEY-LISTVAL [NEW]
Item	TI_simple_pack
Description:	
Parameters:	None
Returns:	BOOLEAN

Algorithm
Development details

if F_SHOW_LOV('LOV_simple_pack ') then
Go_Item('B_itemattr.PB_OK');
end if;

Exception handling:	n/a
Note:	

PROGRAM UNIT NAME:	WHEN-BUTTON-PRESSED [NEW]
Item	PB_SIMPLE_PACK_DESC
Description:	
Parameters:	None
Returns:	BOOLEAN

Algorithm
Development details

if :B_itemattr.TI_simple_pack_desc is NOT NULL then
P_EDITOR.CALL_EDITOR('B_ itemattr.TI_simple_pack_desc',
'N',
'LABL',
'DESC',
NULL,
NULL);
end if;

Exception handling:	n/a
Note:	

PROGRAM UNIT NAME:	KEY-LISTVAL [NEW]
Item	TI_simple_pack_desc
Description:	
Parameters:	None
Returns:	BOOLEAN

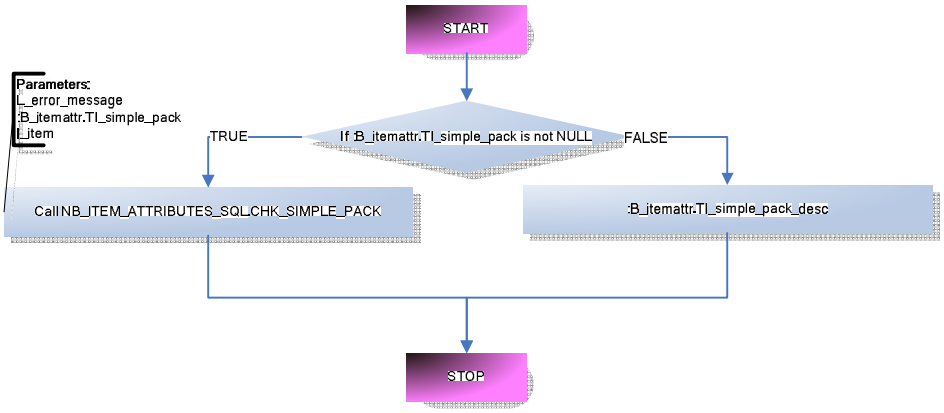
Algorithm
Development details

```
Go_Item('B_head.TI_simple_pack');
if Form_Success then
  Do_Key('List_Values');
end if;
```

Exception handling:	n/a
Note:	

PROGRAM UNIT NAME:	WHEN-VALIDATE-ITEM
Item	:B_itemattr. TI_simple_pack
Description:	Will check the inputted value is a valid simple pack for the regular item
Parameters:	None
Returns:	BOOLEAN

Algorithm



Development details

Exception handling:	n/a
Note:	

PROGRAM UNIT NAME:	POST-TEXT-ITEM
Item	:B_itemattr. TI_simple_pack
Description:	
Parameters:	None
Returns:	BOOLEAN

Algorithm

Development details

Call P_UPDATE_SCREEN

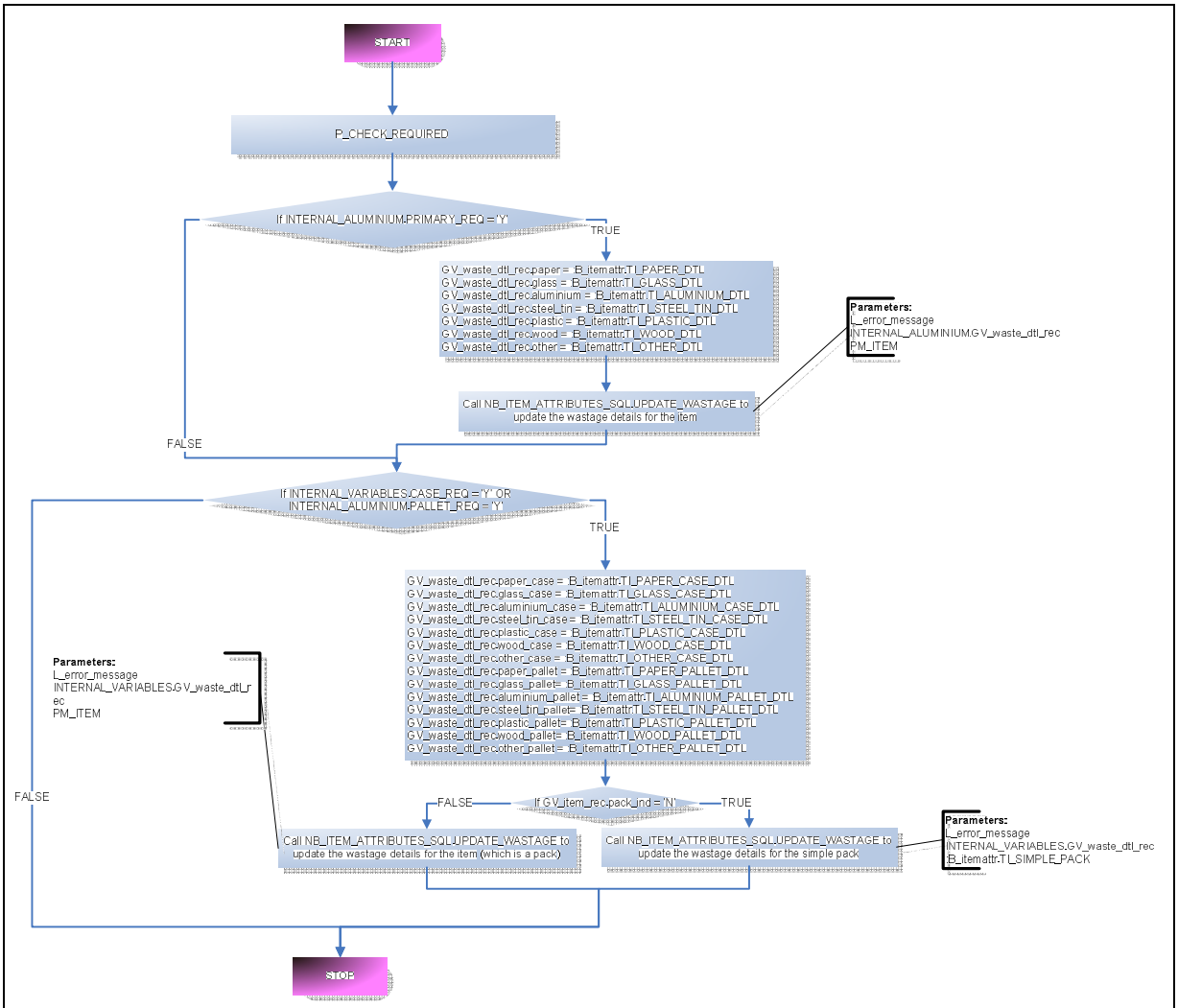
Call NB_ITEM_ATTRIBUTES.GET_ITEM_ATTRIBUTES (O_error_message,
O_item_attrib_rec,
:B_itemattr.TI_simple_pack)

Populate the following fields with the results of the previous function:

:B_itemattr.TI_paper_case_dtl := O_ITEM_ATTRIB_REC.PAPER_CARD_CASE
:B_itemattr.TI_glass_case_dtl := O_ITEM_ATTRIB_REC.GLASS_CASE
:B_itemattr.TI_aluminium_case_dtl : = O_ITEM_ATTRIB_REC.ALUMINIUM_CASE
:B_itemattr.TI_steel_tin_ case_dtl := O_ITEM_ATTRIB_REC.STEEL_TIN_PLATE_CASE
:B_itemattr.TI_plastic_case_dtl := O_ITEM_ATTRIB_REC.PLASTIC_CASE
:B_itemattr.TI_wood_case_dtl := O_ITEM_ATTRIB_REC.WOOD_CASE
:B_itemattr.TI_other_case_dtl := O_ITEM_ATTRIB_REC.OTHER_CASE
:B_itemattr.TI_paper_pallet_dtl := O_ITEM_ATTRIB_REC.PAPER_CARD_PALLET
:B_itemattr.TI_glass_pallet_dtl := O_ITEM_ATTRIB_REC.GLASS_ PALLET
:B_itemattr.TI_aluminium_pallet_dtl := O_ITEM_ATTRIB_REC.ALUMINIUM_ PALLET
:B_itemattr.TI_steel_tin_ pallet_dtl := O_ITEM_ATTRIB_REC.STEEL_TIN_PLATE_ PALLET
:B_itemattr.TI_plastic_ pallet_dtl := O_ITEM_ATTRIB_REC.PLASTIC_ PALLET
:B_itemattr.TI_wood_ pallet_dtl := O_ITEM_ATTRIB_REC.WOOD_ PALLET
:B_itemattr.TI_other_ pallet_dtl := O_ITEM_ATTRIB_REC.OTHER_ PALLET

Exception handling:	n/a
Note:	

PROGRAM UNIT NAME:	WHEN-BUTTON-PRESSED [NEW]
Item	PB_APPLY
Description:	
Parameters:	None
Returns:	BOOLEAN
Algorithm	



Development details

Exception handling:	n/a
Note:	

§ LOVs

Name:	LOV_SIMPLE_PACK	
Description:	LOV which returns the pack items associated to the regular item	
Record Group	RG_SIMPLE_PACK_GROUP	
Column Mapping		
Record Group field	Return Item	Column Title
SIMPLE_PACK	:B_itemattr.TI_simple_pack	Pack Number
DESC	:B_itemattr.TI_simple_pack_desc	Description

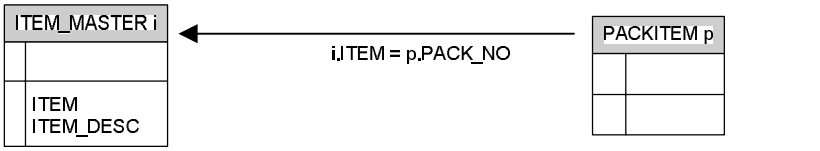
Notes:	

Name:	LOV_OWN_BRAND	
Description:	LOV which returns the own brand descriptions	
Record Group	RG_OWN_BRAND	
Column Mapping		
Record Group field	Return Item	Column Title
BRAND_DESC	:B_itemattr.TI_own_brand	Brand
Notes:		

Name:	LOV_BRAND	
Description:	LOV which returns the brand descriptions	
Record Group	RG_BRAND	
Column Mapping		
Record Group field	Return Item	Column Title
BRAND_DESC	:B_itemattr.TI_brand	Brand
Notes:		

§ Record Groups

Name:	RG_SIMPLE_PACK_GROUP	
Description:	Record group to retrieve all pack items associated to the regular item	
Fields		
Name	Formula	Description
SIMPLE_PACK	I.ITEM	Pack number
DESC	I.ITEM_DESC	Pack description
Query		



i.ITEM_DESC like '%' || UPPER (:B_itemattr.TI_simple_pack_desc) || '%' AND
i.SIMPLE_PACK_IND = 'Y'

p.ITEM = :PARAMETER.PM_ITEM

Notes:	

Name:	RG_OWN_BRAND	
Description:	Record group to retrieve all own brands	
Fields		
Name	Formula	Description

BRAND_DESC	BRAND_DESC	Brand Description

Query

b.BRAND_TYPE = ‘O’

Notes:	

Name:	RG_BRAND
Description:	Record group to retrieve all own brands

Fields

Name	Formula	Description
BRAND_DESC	BRAND_DESC	Brand Description

Query

b.BRAND_TYPE = ‘B’

Notes:	