SAP Business Workflow
Desarrollo

Objetivos del Curso

Objetivos

Son objetivos de este curso

- Identificar todos los puntos en workflow donde se requiere programación
- Crear tipos de objetos nuevos y extender los existentes
- Crear módulos de funciones para determinación de papeles
- Programar eventos
- Programar módulos de funciones para el control de eventos
- Gestionar y monitorear el sistema de workflow en tiempo de ejecución

Perfiles

A quienes va dirigido este curso

- Analistas programadores con muy buen conocimiento y experiencia ABAP
- Analistas programadores con muy buen conocimiento en OOP

Contenidos

- Introducción
- Definición e Implementación de Business Objects

Introducción

- Repaso: ¿Qué es un sistema workflow?
  - Es un sistema que entrega trabajos (tareas)
    - En la secuencia correcta
    - Con toda la información necesaria
    - En el momento correcto
    - A la gente responsable
  - Relacionando estas tareas de manera automática.
  - Control independiente de la aplicación, de las actividades entre las transacciones.
- Repaso: Que cosas **NO** hace el sistema Workflow.
  - Simplificar transacciones complejas (menús, pantallas, etc.)
  - Proveer procesos de negocio eficientes de manera automática (esta tarea se la dejamos a los modeladores del sistema de workflow)
  - Una vez que una aplicación es llamada por el sistema de workflow el control lo tendrá la aplicación y no el sistema de workflow.

Introducción

- Tareas en el sistema de workflow

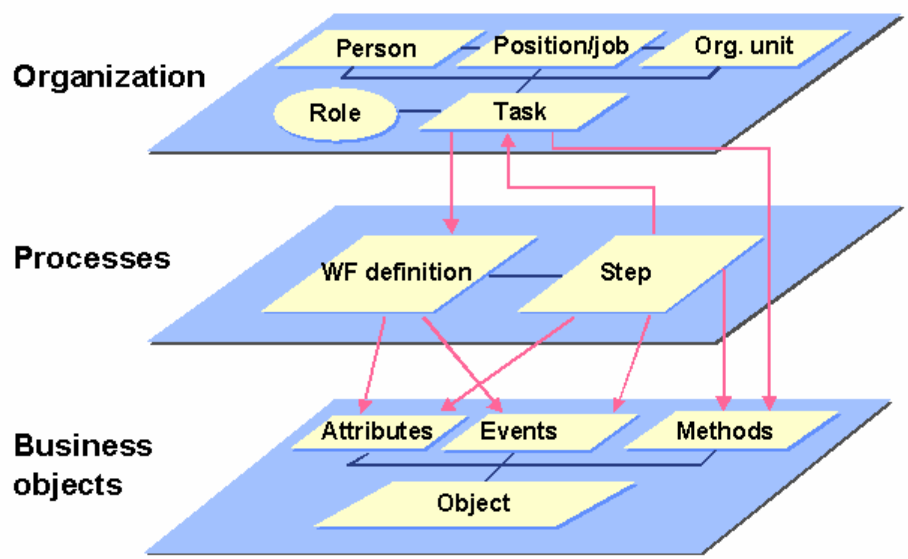
  - Definición del proceso
    - "¿Qué sucede y en qué orden?"
    - Utilización del Workflow Builder y Definición de Tareas
    - Resultado: el proceso se ejecuta en la secuencia correcta
  - Modelización de la organización
    - "¿Quién hace que?"
    - Modelo Organizativo y Definición de Papeles
    - Resultado: El trabajo es realizado por las personas responsables
  - Encapsulación de la Aplicación
    - "¿Cuáles son los objetos que se necesitan?
    - Business Object Builder y Business Object Repository
    - Resultado: El trabajo se envía con la información necesaria

Introducción

- Tareas en el sistema de workflow
  - Soporte al usuario final
    - "¿Qué tengo que hacer hoy?"
    - Herramientas: Business Workplace
    - Resultado: El trabajo se envía a la gente que corresponde en el momento que corresponde
  - Control del proceso
    - "¿Qué pasa cuando…?"
    - Herramientas: Workflow Manager y Workitem Manager
    - Resultado: El trabajo es realizado en la secuencia correcta en el momento correcto
  - Evaluación del proceso
    - "¿Quién hizo que y cuando?
    - Herramientas: Reportes y Análisis

Introducción

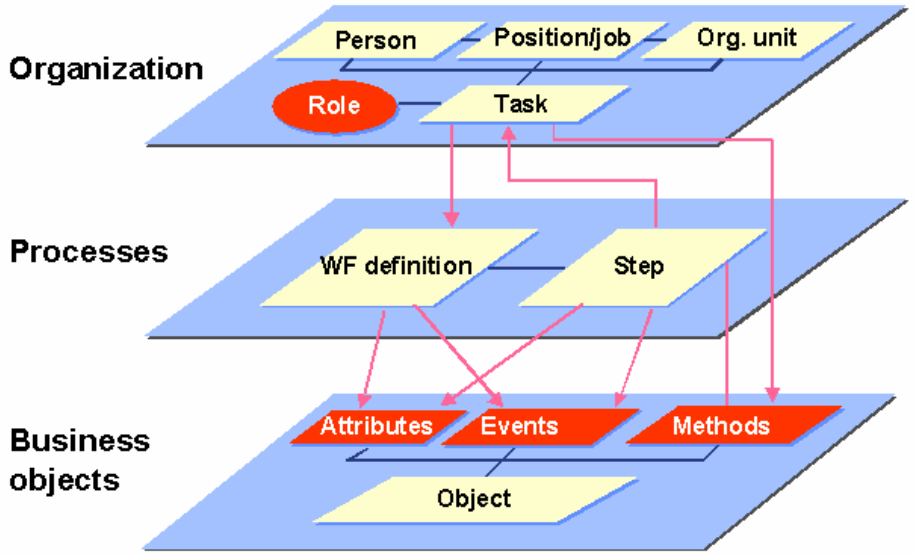
- Arquitectura



Nota: A **à** B significa que el elemento A utiliza el elemento B

Introducción

- Áreas de la arquitectura donde requeriremos programación



Nota: otras áreas donde puede requerirse programación son:

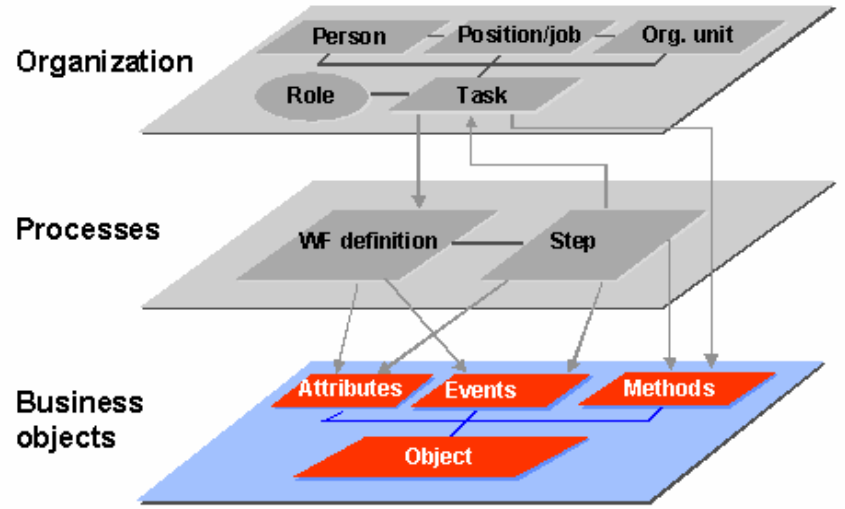
- Administración
- Reportes
- Creación de Workitems.

Contenidos

- Introducción
- Definición e Implementación de Business Objects

Definición e Implementación de Business Objects

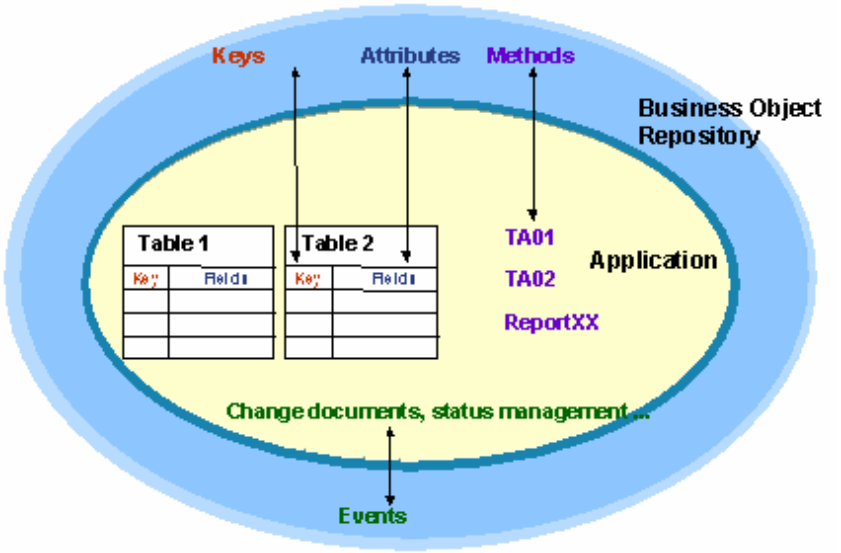
- Comprende el desarrollo en los siguientes elementos de la arquitectura



- Las herramientas que se aprenderán a utilizar en esta unidad son:
  - El BOR (Business Object Repository) para navegar entre los datos
  - El BOB (Business Object Builder) como herramienta de desarrollo

Definición e Implementación de Business Objects

- ¿Por qué utilizamos tecnología orientada a objetos?
  - Principalmente por 2 motivos:
    - Permite simplificar el proceso de modelado del workflow
    - Es una interfase estándar para el entorno de ejecución del workflow
  - Esto se resume en la encapsulación. Los “datos” utilizados en el workflow (tablas) se encapsulan en elementos clave y atributos del objeto, mientras que los programas, funciones, transacciones, etc. se encapsulan en métodos.



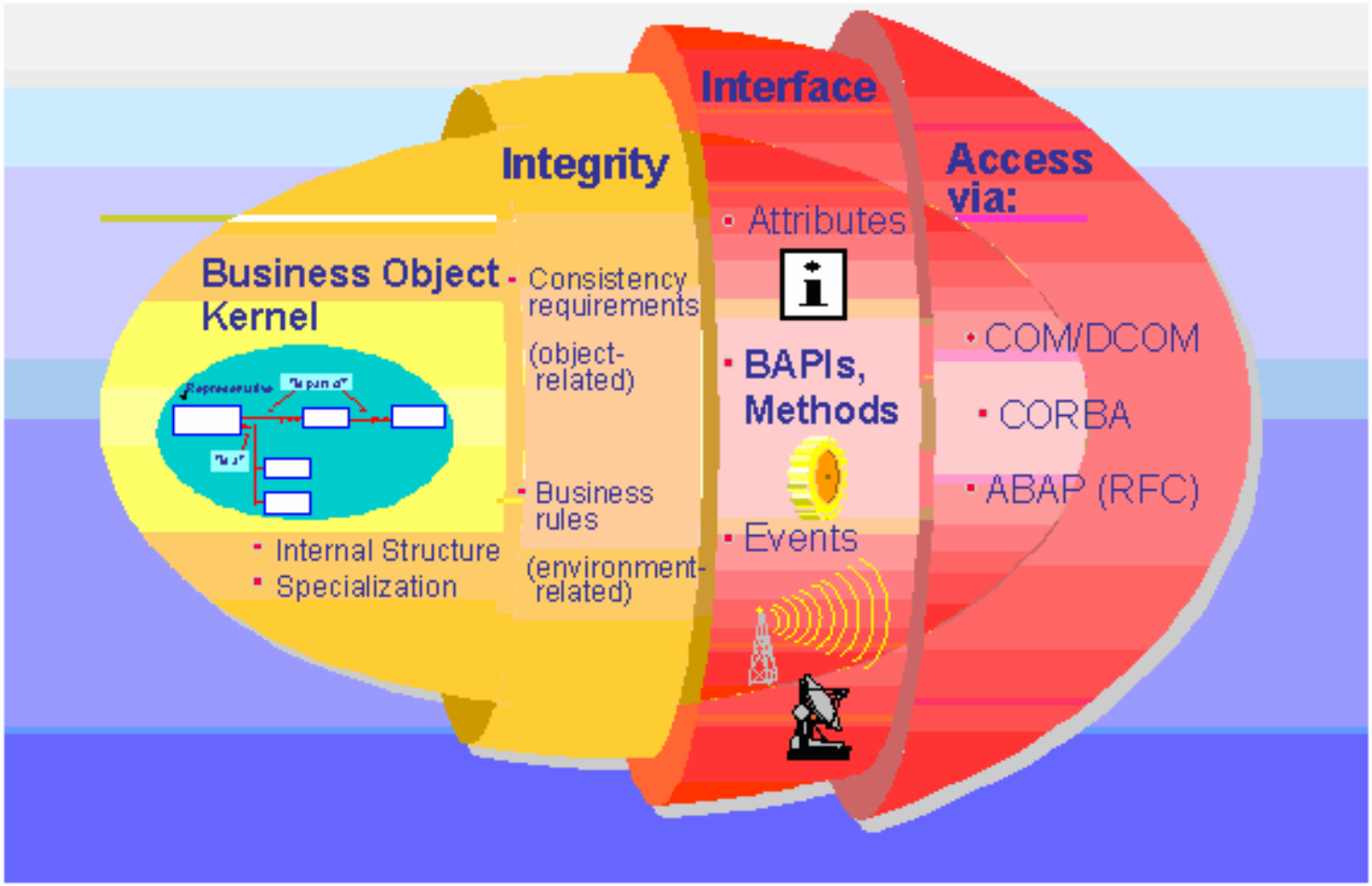
Definición e Implementación de Business Objects

- ¿Dónde se utilizan los objetos en un workflow?
  - Los objetos se utilizan en:
    - ***Tareas***: para ejecutar métodos
    - ***Estructuras de control***: para consultar atributos
    - ***Operaciones en los contenedores***: para consultar atributos
    - ***Pasos de espera o disparadores de eventos***: recibiendo o generando eventos
    - ***Funciones de verificación y de determinación de agentes***: para consultar atributos
    - ***Funciones receptoras***: para consultar atributos
    - ***Papeles***: para consultar atributos
    - ***Métodos secundarios***: para ejecutar métodos

Definición e Implementación de Business Objects

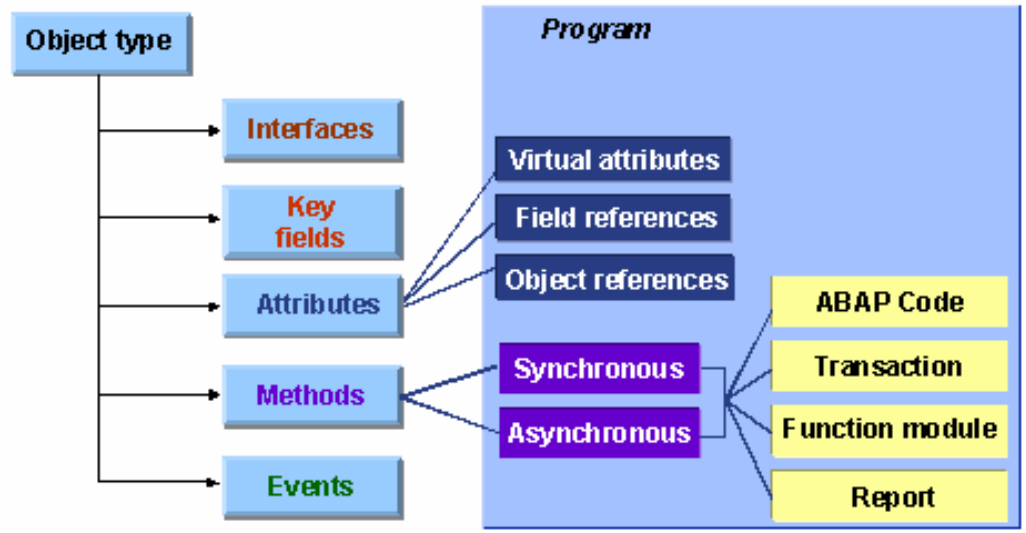
- Relaciones entre objetos
  - Herencia
    - Se utiliza para realizar extensiones funcionales.
    - La relación responde a la frase “es un tipo de”.
    - Por ejemplo: Documento **à** Documento Contable
  - Composición
    - Se utiliza para componer un objeto con otros objetos
    - La relación responde a la frase “es parte de”
    - Por ejemplo: Orden **à** Posicion de la orden
  - Asociación
    - Se utiliza para relacionar dos objetos a través de una clave externa
    - La relación responde a la frase “en relacion a”
    - Por ejemplo: Material **à** Solicitud de pedido

Definición e Implementación de Business Objects



Definición e Implementación de Business Objects

- Object-Type (Tipo de Objeto): Definición



  - Los tipos de objetos pueden consultarse a través del Business Object Repository (BOR) la cual es independiente de mandante.
  - Cada tipo de objeto esta asociado a una clase de desarrollo y, no obstante, a un componente de la aplicación

Definición e Implementación

- **La estructura interna de los Business Objects responde a un conjunto de leyes.**
- **Un business object está definido por los siguientes elementos:**
  - **ID del Objeto:** Identificador único
  - **Campo clave:** Número
  - **Nombre:** Designación semántica única
  - **Componentes:** Estructura interna diferenciada para los objetos de relación “es parte de”
  - **Subtipos:** Especialización de un objeto “es un”
  - **Atributo:** Fecha de entrada, aprobado por, ingresado por, etc.
  - **Métodos:** Implementación de los métodos aplicable al objeto
  - **Eventos:** Eventos que generan al objeto

Definición e Implementación de Business Objects

- Object-Type (Tipo de Objeto): Propiedades

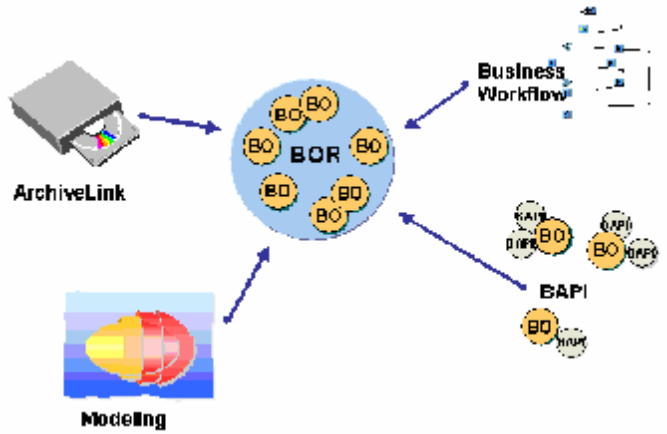
  – En la metodología de desarrollo orientada a objetos las clases tienen determinadas propiedades de las cuales enumeramos:

    - **Encapsulamiento de datos**: consiste en abstraer los datos al modelador del workflow (que puede no ser un programador). Quiere decir que el que este diseñando el workflow no tiene por que saber que tablas, programas, transacciones, etc. están detrás de la ejecución del workflow.
    - **Herencia**: esto significa que los elementos clave, los atributos, métodos y eventos de un tipo de objeto se pasaran a los subtipos que definamos para que de esta manera podamos "extender" la definicion del objeto. Esta propiedad esta orientada a la "reusabilidad" del codigo.
    - **Polimorfismo**: dependiendo del tipo de objeto, el "object manager" siempre selecciona la implementacion de los atributos o metodos que correspondan. Estos elementos siempre se desarrollan utilizando el principio de "late binding".

  – La definición de los tipos de objetos se hace a través del Business Object Builder (BOB).

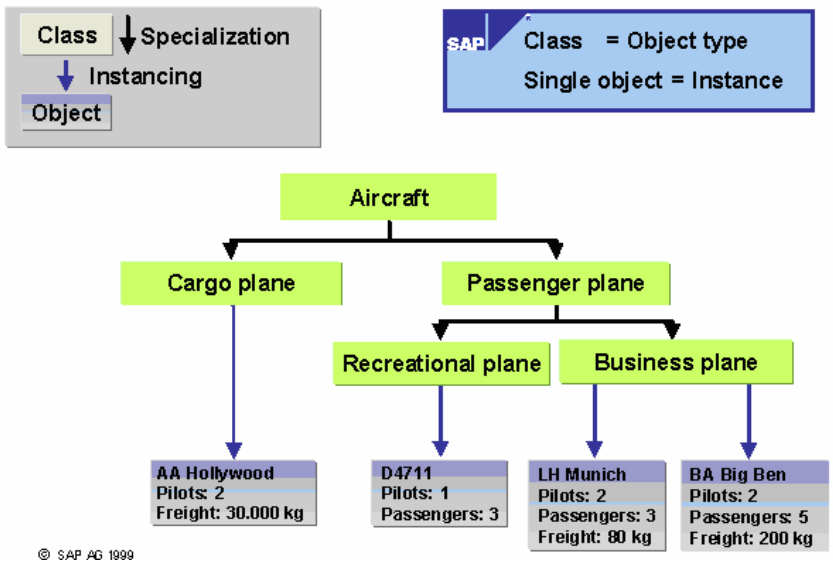
Definición e Implementación de Business Objects

- Business Object Repository (BOR)
  - El BOR es un entorno de desarrollo y ejecución completo, que permite manejar los siguientes tipos de objetos:
    - Objetos del negocio: Incluyen objetos como "Cliente", "Material", y "Código de Compañía". Proveen tanto una visión de alto nivel orientada a los negocios como una interface de programación del sistema R/3.
    - Objetos técnicos: Incluyen textos, notas, ítems de trabajo y documentos, así como objetos de escritorio como textos, gráficos y hojas de cálculo.
    - Metaobjects: Cada objeto tiene un atributo “Tipo de Objeto” que hace referencia al metaobjects al que está asignado. Los métodos, atributos y eventos disponibles para un objeto en particular pueden ser recuperados desde su “Tipo de Objeto”.



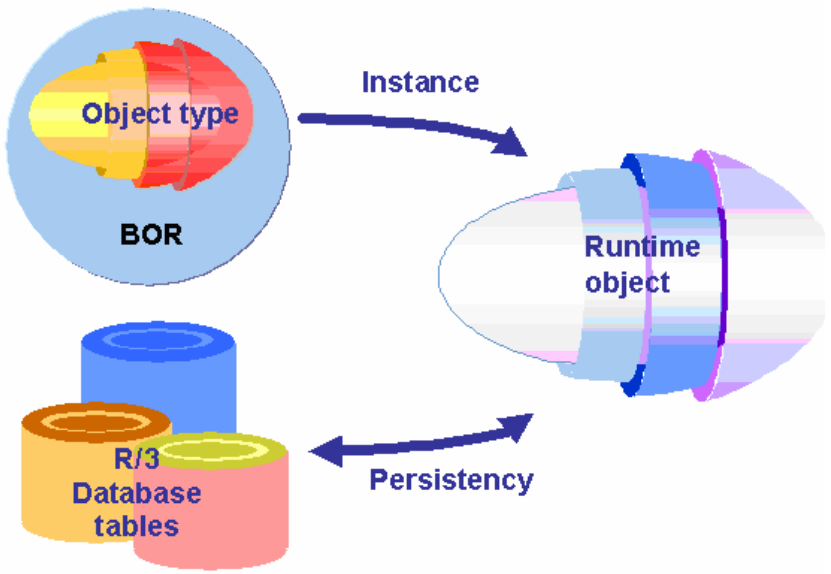
Definición e Implementacion de Business Objects

- Instancias (Objectos)
  - Un tipo de objeto (clases) describe un objeto de negocio abstracto, los datos que le pertenecen, métodos, etc.
  - Los tipos de objetos pueden ser hederados. Los tipos de objetos heredados contienen todos los métodos y campos de datos de los objetos de los cuales heredan.
  - Esto permite el polimorfismo, que combina los atributos de distintos objetos especializados. El polimorfismo puede ser utilizado también por interfaces definidas (atributos y métodos) de objetos.
  - La herencia múltiple no es actualmente soportada directamente por los objetos de negocio de SAP. Un objeto puede implementar varias interfaces.
  - Las instancias de un objeto de negocio contienen datos actuales, por ejemplo, una orden de cliente. Asi pueden existir muchas instancias de un tipo de objeto.



Definición e Implementación de Business Objects

- Instancias (Objectos)
  - Para trabajar con un objeto de negocio, debe primero crearse una instancia del objeto.
  - Cuando se crea una instancia, existen dos tipos de objetos: persistente y no persistente.
  - Los objetos persistentes contienen datos de las bases de datos R/3. Tienen un identificador único determinado por el campo clave.
  - Los objetos no persistentes no están ligados a entradas concretas de base de datos. Por ejemplo, son utilizados para visualizar datos, o para crear nuevos objetos de negocio persistentes.



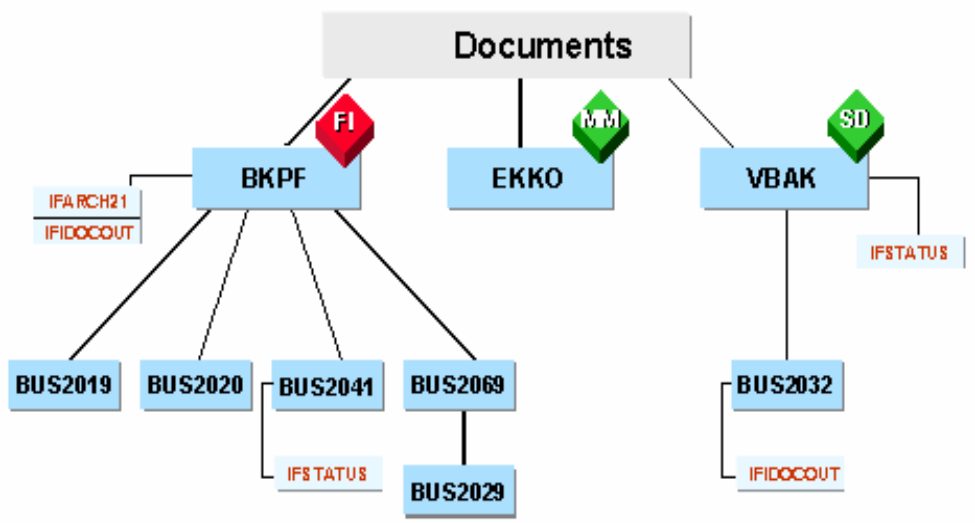
Definición e Implementación de Business Objects

- Interfases: Definición

  - La utilización de interfaces en el desarrollo de tipos de objetos nos permitirá asegurar un entorno común para todos los objetos.
  - La definición de una interfase es similar a la definición de un tipo de objeto con la diferencia que no se implementaran ninguno de los elementos contenidos en la interfase (es decir que es un molde que no contiene código ABAP)
  - Un tipo de objeto que “implemente” una inteface deberá entonces definir (codificar) todos los atributos y metodos que la interface propone.
  - Las interfaces se heredan
  - Las interfaces en SAP se utilizan para reemplazar la herencia multiple. Esto se da por que es mas sencillo de mantenerlas que un tipo de objeto.

Definición e Implementación de Business Objects

- Interfases: Ejemplos



BUS2069 Documento contable
BUS2029 Precios
EKKO Documento de compras
VBAK Documento de ventas
BUS2032 Orden de venta

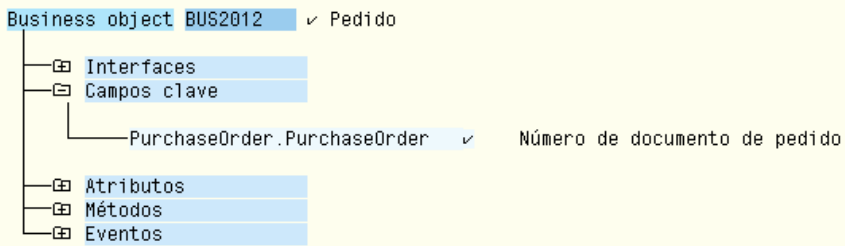
→

IFSTATUS Genera eventos para gestion de status
IFARCH21 Interface para archive link
IFIDOCOUT Procesamiento de salida de IDOCs

Definición e Implementación de Business Objects

- Tipos de Objetos – Elementos: Campos Clave (Key)

  – Un objeto se identifica “univocamente” de otro a través de su clave.
  – Una clave puede estar compuesta de uno o mas campos
  – Hacen referencia a un campo clave de una tabla de la aplicación subyacente
  – El mandante no es necesario en las claves
  – Deben ser campos tipo carácter (CHAR).
  – Los campos clave concatenados pueden contener un máximo de 70 caracteres.



Definición e Implementación de Business Objects

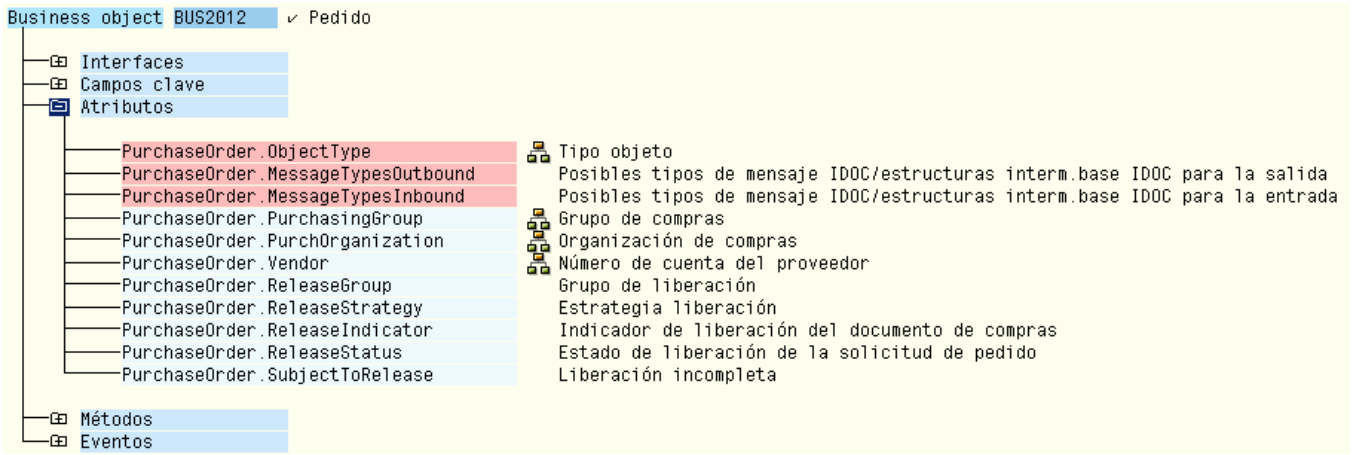
- Tipos de Objetos – Elementos: Atributos

  – Un atributo de un objeto representa determinada característica que este objeto puede llegar a tener.

  – En SAP los atributos pueden ser de base de datos o virtuales.

  – En cuanto a su definición pueden estar relacionados a un tipo de dato de la base de datos o a un tipo de objeto (para asociaciones o composiciones)

  – Pueden ser de una línea o varias líneas (single-line o multiple-line)



Definición e Implementación de Business Objects

- Tipos de Objetos – Elementos: Atributos Virtuales

  – Un atributo virtual es un atributo “calculado”, es decir que se requiere programación para poder obtenerlo (no viene directamente de la base de datos).

  – El Business Object Builder no puede generar el código para un atributo virtual automáticamente (como sucede con los atributos de base de datos) por lo que deben ser “explicitamente” codificados.

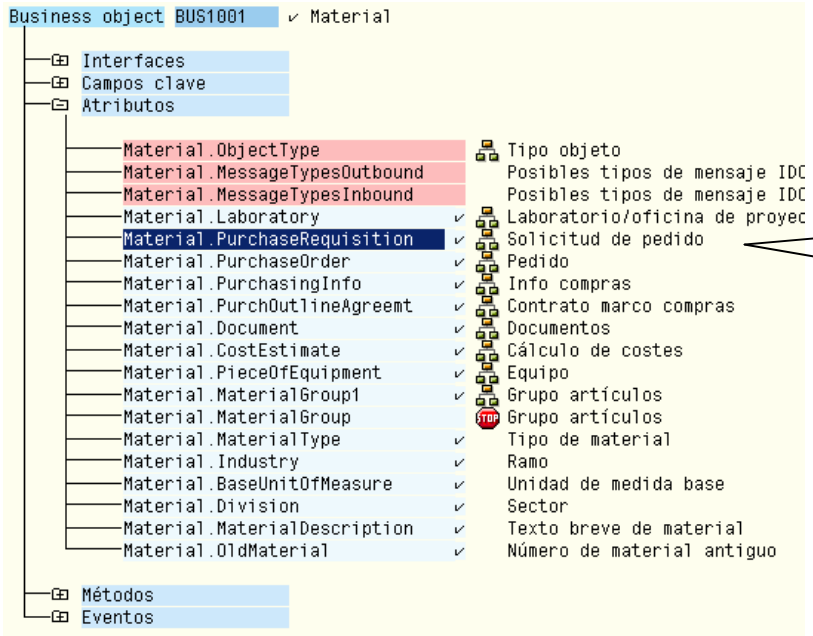
  – Ejemplos donde son necesarios atributos virtuales:
    - Datos dependientes de lenguaje
    - Datos dependientes del tiempo
    - Referencias a objetos
    - Calculo de valores netos - brutos

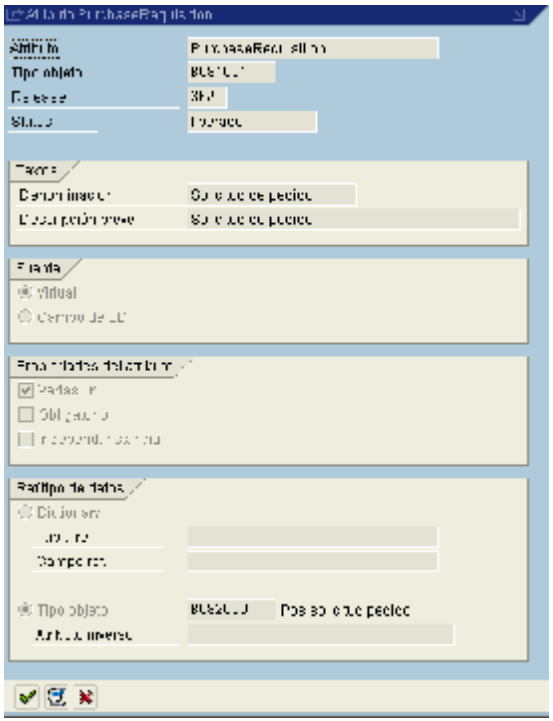
  – Por razones de rendimiento del sistema deberíamos utilizar atributos virtuales para leer atributos en pasos de fondo (background). Pero esto no es estrictamente obligatorio.

Definición e Implementación de Business Objects

- Tipos de Objetos – Elementos: Atributos Multi-lineales (multi-line)

  – Se corresponden a lo que en ABAP denominamos “tablas internas”
  – Pueden contener campos de base de datos u objetos.
  – Generalmente son atributos virtuales.





Definición e Implementación de Business Objects

- Tipos de Objetos – Elementos: Métodos (Sincrónicos y Asincrónicos)

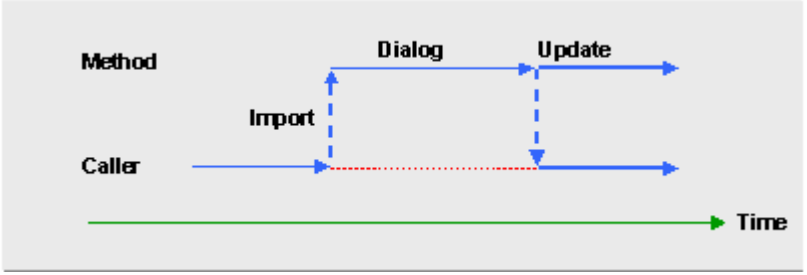
  – Los métodos son las actividades que podemos llevar a cabo sobre un objeto

  – Pueden ser sincrónicos o asincrónicos
    - En los “Sincrónicos” el resultado es enviado directamente al programa que lo ejecuto. Además es posible la utilización de excepciones.
    - En los métodos “Asincrónicos” el resultado siempre es enviado de vuelta al programa llamador en la forma de un evento. Solo se permiten algunas exceptiones (restringidas). El método en si mismo no espera un posible evento finalizador.

  – Ambos tipos de métodos pueden recibir datos a través de parámetros (parámetros de importación)

Definición e Implementación de Business Objects

- Tipos de Objetos – Elementos: Métodos Asincrónicos (detalle)



  - Notas
    - Las líneas azules continuas representan el control del flujo
    - Las líneas azules punteadas representan transferencias de datos
    - La línea punteada roja representa el tiempo de espera del programa llamador.

  - Se necesita una sincronización de los procesos controlada por el programa llamador.
  - El método debe enviar eventos al programa llamador para que este se sincronice.
  - Todos los métodos que hacen escrituras en la base de datos a través de tareas de actualización.

Definición e Implementación de Business Objects

- Tipos de Objetos – Elementos: Método (comunicación)
  - Comunicación vía parámetros
    - Import
    - Export
  - Comunicación a través de resultados
  - Comunicación vía excepciones
    - Error temporal
    - Error de Sistema
    - Error de Aplicación
  - Comunicación vía eventos
- Los métodos sincrónicos se comunican mediante parámetros, resultados y excepciones.
- Los métodos asincrónicos se comunican mediante parámetros y luego mediante un evento.

Definición e Implementación de Business Objects

- Tipos de Objetos – Elementos: Método (comunicación)

  – Los parámetros de tipo export deberían prohibirse. Lo mejor es comunicarse a través de resultados.

  – Si durante la ejecución del método el objeto al que se esta haciendo referencia cambia en la base de datos, este objeto debe refrescarse (una manera de comunicar el resultado) para ello existe la macro SWC_REFRESH_OBJECT.

  – El tipo de excepción que se envíe al sistema de workflow determinara la manera en que se continúe o no la ejecución del workflow (dejando el workitem en estado erróneo o en proceso)

  – Los resultados por excepciones se programan mediante la macro EXIT_RETURN. Otras macros utilizadas para crear excepciones son EXIT_OBJECT_NOT_FOUND y EXIT_CANCELLED. Esta última se utiliza cuando el usuario cancela la ejecución de un método (generando un error temporal).

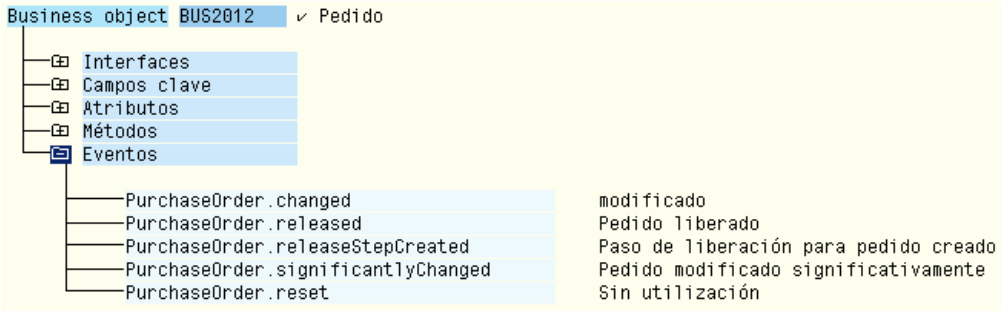
Definición e Implementación de Business Objects

- Tipos de Objetos – Elementos: Eventos

  – Un evento se utiliza principalmente para indicar que algo a sucedido. Estos son indispensables para iniciar o terminar workflows.

  – La definición del evento se hace en el Business Object Builder, pero su implementación se hace con otras herramientas, por ello la documentación de los eventos es “indispensable”.

  – Los eventos llevan y traen parámetros. Los parámetros pueden ser definidos por el usuario (explícitamente) o standards los cuales no se definen (objeto lanzador, usuario que lanza el objeto, fecha, hora, etc.).

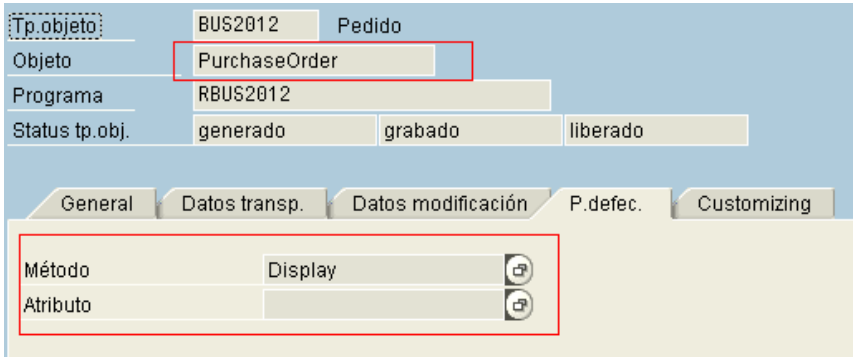


Definición e Implementación de Business Objects

- Estado de un Tipo de Objeto
  - Modelado
    - En este estado el tipo de objeto no se puede “instanciar”. Es decir no se puden generar objetos para este tipo.
  - Implementado
    - Solo para pruebas, uso interno o posiblemente inestable
  - Liberado
    - Liberado para ser utilizado por el cliente. Solo se podrán realizar ampliaciones pero no modificar radicalmente el tipo.
  - Obsoleto
    - El tipo de objeto ha sido reemplazado por otro.

Definición e Implementación de Business Objects

- Desarrollo de un Tipo de Objeto: Datos Generales
  - Nombre del objeto
    - Debe ser un nombre descriptivo para todos los objetos de este tipo. Cuando utilicemos un objeto como elemento del contenedor de workflow este será el nombre que nos aparecerá para definir la variable del contenedor.
  - Método por defecto
    - Este método es el que se utiliza para visualizar el objeto (por ejemplo cuando tenemos el objeto como un link en el workplace del workitem y al hacer doble click nos muestra el contenido de ese objeto o nos lleva a la pantalla para visualizar el objeto)
  - Atributo por defecto
    - Es un atributo identificativo del objeto instanciado. Se utiliza para cuando se quieren emitir listados de objetos para que aparezca ese dato por defecto.



Definición e Implementación de Business Objects

- Desarrollo de un Tipo de Objeto: Campos Clave

  – Cada tipo de objeto tiene un programa que lo implementa.

  – El desarrollo de un tipo de objetos se basa en MACROS que se encuentran en el include <OBJECT>, por eso todos los programas que implementan un tipo de objeto empiezan con la sentencia “include <object>.”

  – El Business Object Builder siempre utiliza referencias a objetos para trabajar. Estas referencias las utiliza para leer y manipular los datos de la aplicación. En el programa del tipo de objeto podemos crear una referencia a un objeto con la macro SWC_CREATE_OBJECT.

  – El programa del tipo de objeto puede utilizar una referencia del objeto que e esta ejecutando, es decir una referencia a si mismo. Esta variable se llama SELF.

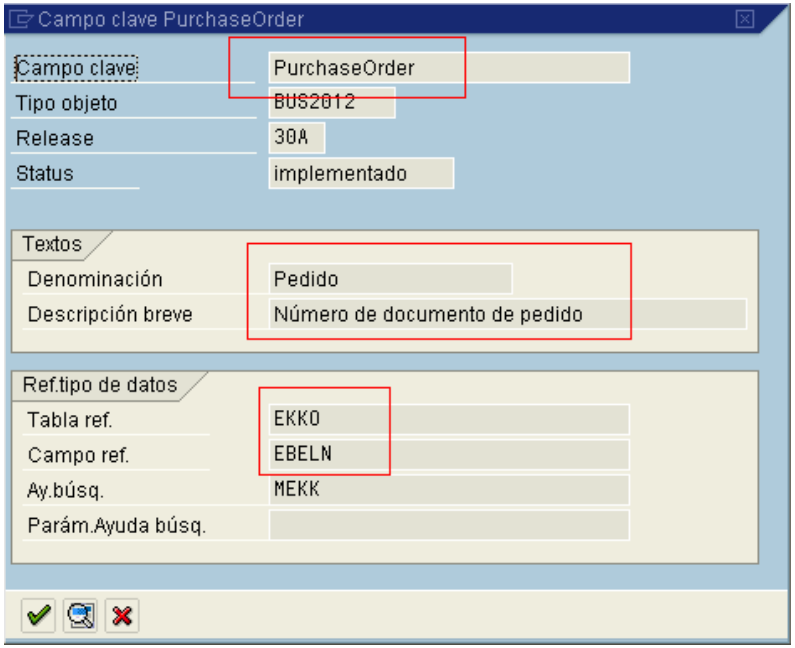
  – Las claves se definen con el Business Object Builder y el programa se genera automáticamente (no es necesario programar la definición de las claves).

Definición e Implementación de Business Objects

- Desarrollo de un Tipo de Objeto: Campos Clave

  – Para crear una clave debemos indicar el nombre (en ingles) una descripción (en cualquier idioma), y una referencia a un tipo de dato de una tabla de la aplicación.

  – Luego en el programa del tipo de objeto vemos que la clave se define entre las sentencias BEGIN OF KEY y END OF KEY.




```
INCLUDE <OBJECT>.
BEGIN_DATA OBJECT. " Do not change.. DATA is generated
* only private members may be inserted into structure private
DATA:
" begin of private,
"   to declare private attributes remove comments and
"   insert private attributes here ...
" end of private,
  BEGIN OF KEY,
      PURCHASEORDER LIKE EKKO-EBELN,
  END OF KEY,
```

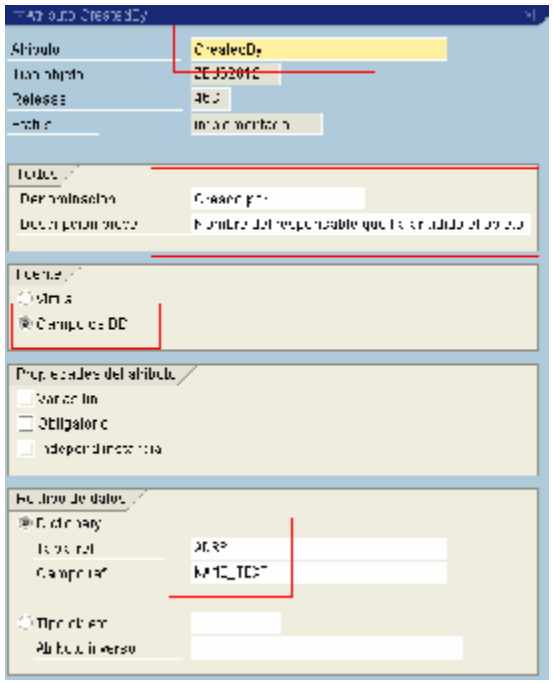
Definición e Implementación de Business Objects

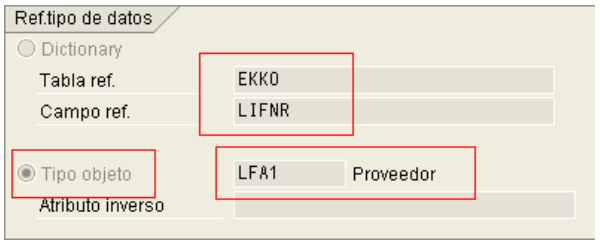
- Desarrollo de un Tipo de Objeto: Atributos de Base de Datos

- Cuando creamos un atributo de base de datos el sistema automáticamente nos propondrá un código básico que podremos terminar para que funcione.

- Para crear el atributo le damos un nombre (en ingles), una descripción (en cualquier idioma), marcamos el atributo como base de datos y le damos una referencia a una tabla y un campo de la tabla.

- Si colocamos como referencia un objeto deberemos tener en cuenta que el objeto tenga como clave un solo campo (que este definido en la tabla que en los campos tabla y campo indiquemos.



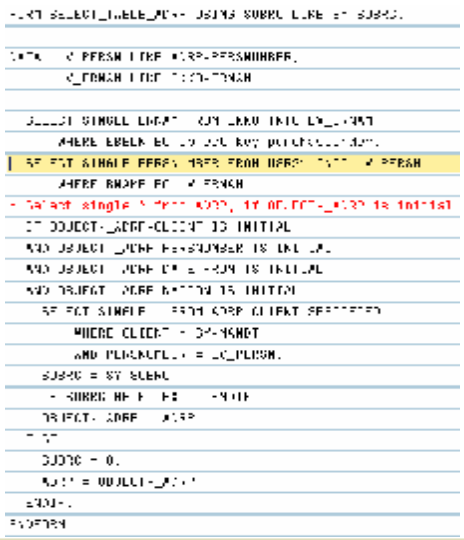


Definición e Implementación de Business Objects

- Desarrollo de un Tipo de Objeto: Atributos de Base de Datos

  – Al crear el atributo el sistema nos propone definir automáticamente el programa.
  – Luego este programa podrá ser reutilizado para todos los atributos que apunten a la misma tabla.
  – El atributo estará definido por las sentencias GET_TABLE_PROPERTY y END_PROPERTY.
  – De no encontrar nada el sistema utiliza la excepción EXIT_OBJECT_NOT_FOUND.

```
GET_TABLE_PROPERTY ADRP.
DATA SUBRC LIKE SY-SUBRC.
* Fill TABLES ADRP to enable Object Manager Access to Table Properties
  PERFORM SELECT_TABLE_ADRP USING SUBRC.
  IF SUBRC NE 0.
    EXIT_OBJECT_NOT_FOUND.
  ENDIF.
END_PROPERTY.
```



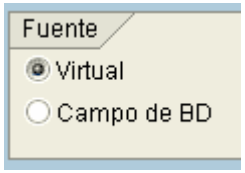
Definición e Implementación de Business Objects

- Desarrollo de un Tipo de Objeto: Atributos Virtuales

- Al crear el atributo virtual el sistema no generara el código automáticamente pero si nos presentara donde debemos colocar el programa (entre las clausulas GET_PROPERTY y END_PROPERTY.

- El atributo se crea mediante el programa (como una variable) y luego se coloca como resultado en el container del objeto mediante la macro SWC_SET_ELEMENT.

- En el caso de que el atributo virtual sea un objeto deberemos instanciar un objeto (conociendo como completar la clave) mediante la macro SWC_CREATE_OBJECT.




```
begin_data object.
  begin of key,
    ...
  end of key,
  salesgroup type swc_object.
end_data object.

get_property salesgroup changing container.

  select single * from vbak
     where vbeln = object-key-salesdocument.

* create object(-reference)
  swc_create_object
      object-salesgroup 'TVKGR' vbak-vkgrp.

* write object(-reference) into container
  swc_set_element container
      'SalesGroup' object-salesgroup.

end_property.
```

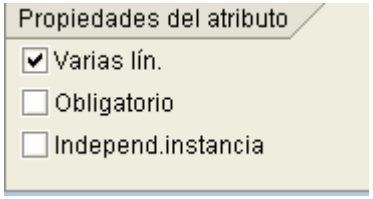
Definición e Implementación de Business Objects

- Desarrollo de un Tipo de Objeto: Atributos de Múltiples Líneas

- Al crear el atributo de múltiples líneas el sistema tampoco generara el código automáticamente pero si nos presentara donde debemos colocar el programa.

- En el caso de los atributos múltiples se trabaja al atributo como una tabla interna.

-El atributo se colocará en el contenedor con la macro SWC_SET_TABLE.




```
begin_data object.
  begin of key,
    ...
  end of key,
  items type swc_object occurs 0.
end_data object.

get_property items changing container.

* Declare data
  tables vbap.
  data item type swc_object.
  data: begin of vbap_key,
          vbeln like vbap-vbeln,
          posnr like vbap-posnr,
        end of vbap_key.
  data vbap_tab like vbap occurs 0 with header line.

* Select data
  select * from vbap into table vbap_tab
     where vbeln = object-key-salesdocument.
  vbap_key-vbeln = object-key-salesdocument.

* Create object reference
  loop at vbap_tab.
      vbap_key-posnr = vbap_tab-posnr.
      swc_create_object item 'VBAP' vbap_key.
      append item to object-items.
  endloop.

* Assign object reference to container element
  swc_set_table container 'Items' object-items.

end_property.
```

Definición e Implementación de Business Objects

- Desarrollo de un Tipo de Objeto: Métodos Sincrónicos

- Para crear un método sincrónico debemos marcarlo como tal en la pantalla de atributos del método.

- En el programa del tipo de objeto la definición del método se encuadra dentro de las sentencias BEGIN_METHOD y END_METHOD.

- En el caso que el método modifique algún atributo del objeto deberemos ejecutar la macro SWC_REFRESH_OBJECT para borrar el buffer del objeto.

- En el ejemplo se muestra como llamar a una transacción con un método.


```
begin_method display changing container.

  set parameter id 'AUN' field object-key-salesdocument.
  call transaction 'VA03' and skip first screen.

end_method.
```

Definición e Implementación de Business Objects

- Desarrollo de un Tipo de Objeto: Métodos Asincrónicos

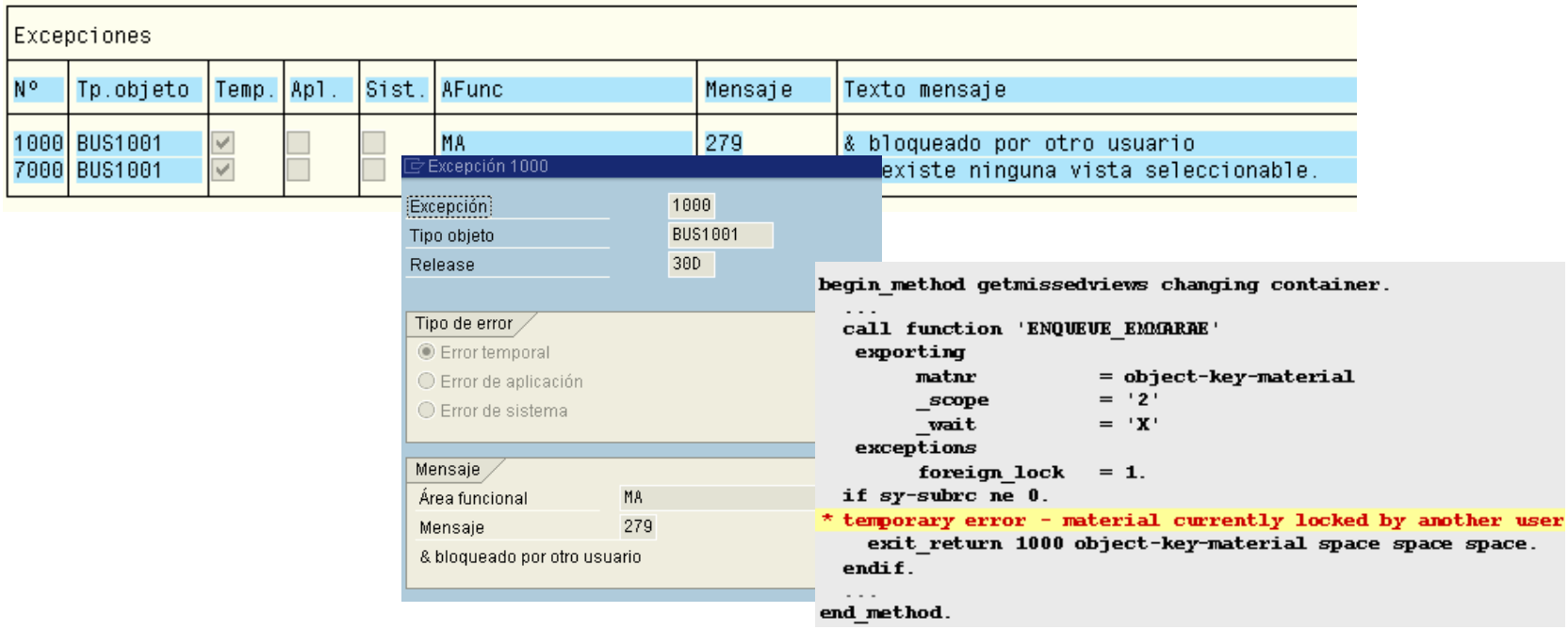
  - En el caso de los métodos asincrónicos no será necesario llamar la macro SWC_REFRESH_OBJECT por que el contexto del objeto siempre se pierde al llamar el método.

  - La ejecución del evento terminador debe darse en la transacción o el programa que ejecuta el método.

  - La tarea de workflow que llame a un objeto asincrónico SIEMPRE esperará el evento terminador. Por este motivo debemos tener cuidado que el programa / transacción / modulo de funcione / etc que ejecute el método siempre lance un evento.

Definición e Implementación de Business Objects

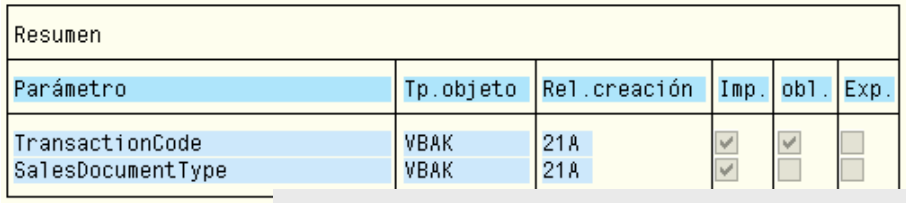
- Desarrollo de un Tipo de Objeto: Implementando excepciones en los métodos



- Las excepciones se definen para cada método. El sistema no generará ningún tipo de código para la excepción.
- La excepción se debe corresponder a un mensaje tipo T100 (con 4 parámetros).
- Para llamar la excepción se usa la macro EXIT_RETURN.

Definición e Implementación de Business Objects

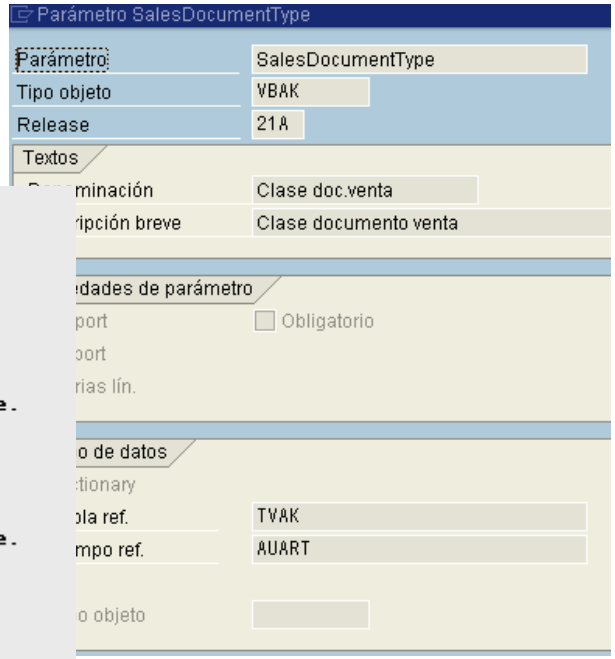
- Desarrollo de un Tipo de Objeto: Completando parámetros de los métodos



Resumen

Parámetro	Tp.objeto	Rel.creación	Imp.	obl.	Exp.
TransactionCode	VBAK	21A	☑	☑	☐
SalesDocumentType	VBAK	21A	☑	☐	☐

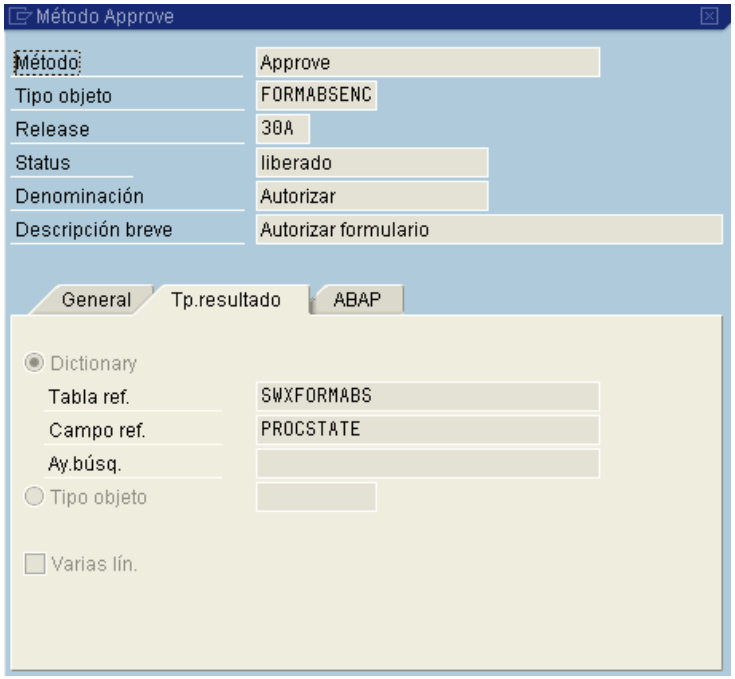
```
begin_method create changing container.

  data: salesdocumenttype like tvak-auart,
        transactioncode like t180-tcode.

  swc_get_element container
      'SalesDocumentType' salesdocumenttype.
  set parameter id 'AAT' field salesdocumenttype.

  swc_get_element container
      'TransactionCode' transactioncode.

  set parameter id 'AAT' field salesdocumenttype.
  call transaction transactioncode.
  .
  .
  .
```



- Para cada método puede o no definirse uno o mas parámetros de entrada y salida.
- Para leer elementos del container se utilizan las macros SWC_GET_ELEMENT (para elementos de una línea) o SWC_GET_TABLE (para los multilínea). A su vez la macro SET PARAMETER ID la utilizaremos para copiar los datos extraídos al parámetro deseado.

Definición e Implementación de Business Objects

- Desarrollo de un Tipo de Objeto: Completando resultados de los métodos




```
begin_method approve changing container.

  data: proc_state like swxformabs-procstate.

  call function 'SWX_FORMABS_APPROVE'
     exporting
            formnumber      = object-key-number
     importing
            proc_state      = proc_state
     exceptions
            form_not_found = 01
            aborted         = 02.
  case sy-subrc.
    when 00.
      swc_set_element container result proc_state.
    when 01.
      exit_return 1301 space space space space.
    when 02.
      exit_cancelled.
  endcase.

end_method.
```

- Los parámetros de resultado se completan con la macro SWC_SET_ELEMENT y el elemento siempre se llama “RESULT”. El elemento RESULT esta dentro del include OBJECTS por lo que no es necesario definirlo explicitament.

Definición e Implementación de Business Objects

- Utilizando un Objeto en un programa: Accediendo a los Atributos

  - Para utilizar un Business Object en un programa es necesario incluir el include **<CNTN1>.** Este include es utilizado en el include <OBJECT> por lo que no se incluye en el programa del tipo de objeto.

  - Antes de acceder a un atributo debemos crear una instancia de un tipo de objeto. Para ello usamos la macro **SWC_CREATE_OBJECT**.

  - Una vez creado el objeto podremos acceder a sus atributos con las macros **SWC_GET_PROPERTY** o **SWC_GET_TABLE_PROPERTY** (para atributos multi-linea)

  - Los atributos no pueden modificarse por macros. Solo podrán modificarse llamando a métodos que implementen el cambio.

```
include <CNTN01>.

parameters: sales_order_key like vbak-vbeln.

data: sales_order type swc_object,
      sales_group type swc_object.

* create object(-reference)
  swc_create_object
      sales_order 'Z_BUS2032' sales_order_key.

  if sy-subrc ne 0.
    " do your own error handling
  endif.

* read attribute SalesGroup of order
  swc_get_property sales_order 'SalesGroup' sales_group.
```

Definición e Implementación de Business Objects

- Utilizando un Objeto en un programa: Accediendo a los Métodos

- Para utilizar un Business Object en un programa es necesario incluir el include **<CNTN1>.** Este include es utilizado en el include <OBJECT> por lo que no se incluye en el programa del tipo de objeto.

- Antes de acceder a un método debemos crear una instancia de un tipo de objeto. Para ello usamos la macro **SWC_CREATE_OBJECT**.

- Antes de llamar el método debemos completar los parámetros de entrada (import)

-Se llama al método con la macro **SWC_CALL_METHOD.**

- Finalmente se consultan los resultados o los parámetros de salida (export).

```
include <CNTN01>.

data: sales_order type swc_object,
      comp_curreny  like vbak-waerk default ...,
      new_value     like vbak-netwr.
      swc_container method_container.

* create object(-reference)
  swc_create_object sales_order
      'BUS2032' sales_oder_key.
  if sy-subrc ne 0.
    " do your own error handling
  endif.


* set import parameters for this method
  swc_set_element method_container
      'CompCurrency' comp_currency.

* call method now
  swc_call_method sales_order
     'ConvertToCompCurrency' method_container.

* read export parameter
  swc_get_element method_container
      'NetvalueInCompCurrency' new_value.
```

Definición e Implementación de Business Objects

- Resumen de MACROS

  – Todas las macros para manipular objetos se encuentran en el programa include **<CNTN01>.**

  – Todos los objetos pueden manipularse fuera del sistema workflow via macros.

- Referencia a un Objeto
  – Declaración
    - DATA: <OBJ_REF> TYPE SWC_OBJECT
  – Creación
    - SWC_CREATE_OBJECT <OBJ_REF> <OBJ_TYPE> <OBJ_CLAVE>
- Acceso a Atributos
  – SWC_GET_[TABLE]_PROPERTY <OBJ_REF> <ATRIBUTO> <VALOR>
- Acceso a un Metodo
  – SWC_CALL_METHOD <OBJ_REF> <METODO> <CONTAINER CON VARIABLES>

Definición e Implementación de Business Objects

- Resumen de MACROS

- Disparando Excepciones
  - EXIT_RETURN <NRO_EXCEPCION> <VAR1> ….. <VAR4>
  - EXIT_OBJECT_NOT_FOUND
  - EXIT_CANCELLED

- Tomar el tipo y la clave de un objeto
  - SWC_GET_OBJECT_TYPE <OBJ_REF> <OBJ_TYPE>
  - SWC_GET_OBJECT_KEY <OBJ_REF> <OBJ_KEY>

- Refrescar el objeto
  - SWC_REFRESH_OBJECT <OBJ_REF>

Definición e Implementación de Business Objects

- Resumen de MACROS – macros para manejar el contenedor de datos

- Definición e Inicialización
  - SWC_CONTAINER <variable>
  - SWC_CREATE_CONTAINER <variable>

- Leer y Escribir en el contenedor
  - SWC_GET/SET_ELEMENT <contenedor> <elemento> <valor>
  - SWC_GET/SET_TABLE <contenedor> <elemento> <tabla interna>

Definición e Implementación de Business Objects

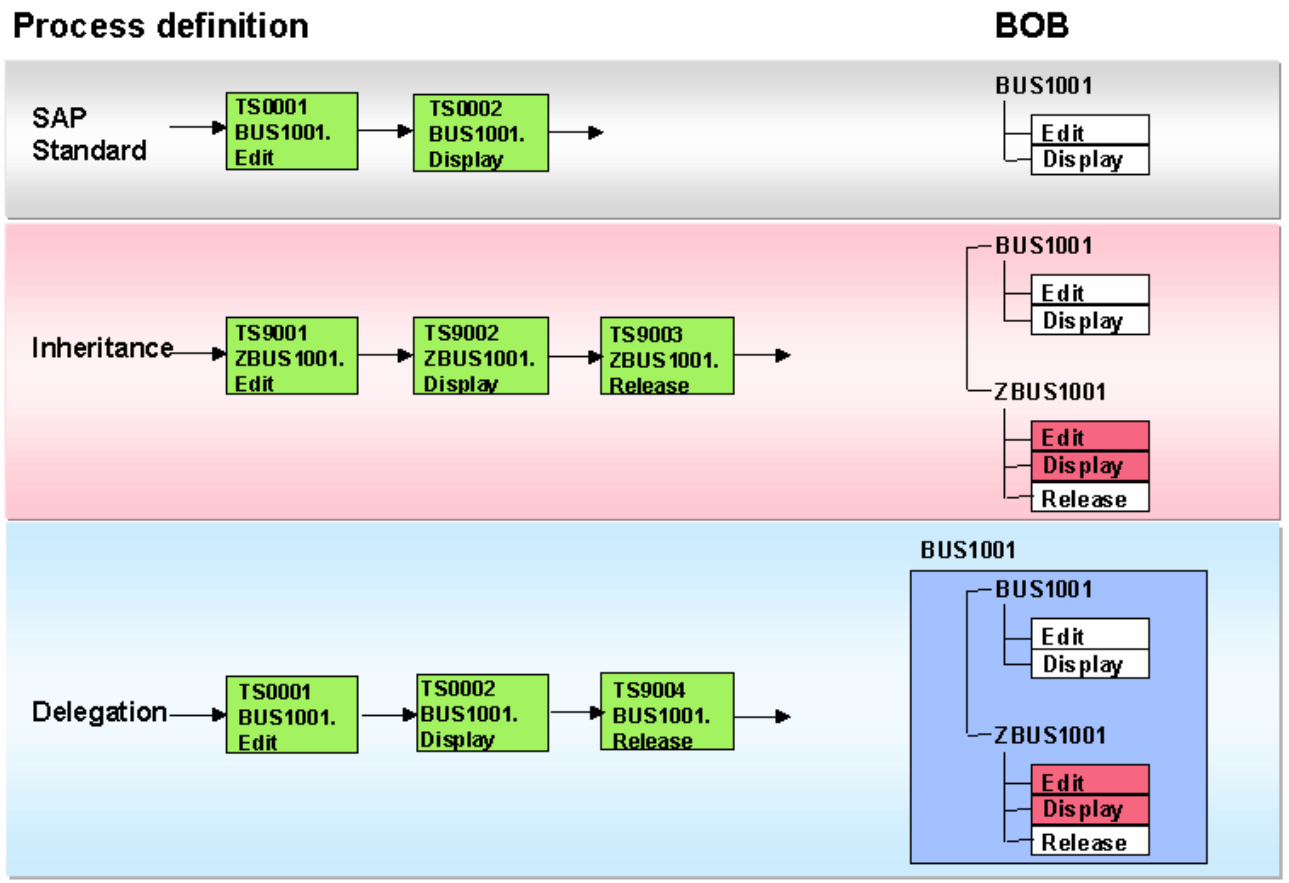
- Delegación

  - **Problema**
    - Como podemos crear nuestras propias extensiones de objetos para poder usar en tareas, eventos, etc. De un objeto creado por SAP sin tener que cambiar TODAS las tareas, eventos, etc.?
  - **Solución**
    - Definir un Sub-Tipo (herencia) y delegarlo en el supertipo
    - La delegación hace que el sub-tipo “cubra” al supertipo
    - De esta manera podemos seguir haciendo referencia al supertipo en las tareas, eventos, etc.

  - Si creamos un sub-tipo y no lo delegamos entonces los programas, tareas, eventos, etc que usen al supertipo no se enterarán de las extensiones que hagamos en el sub-tipo.

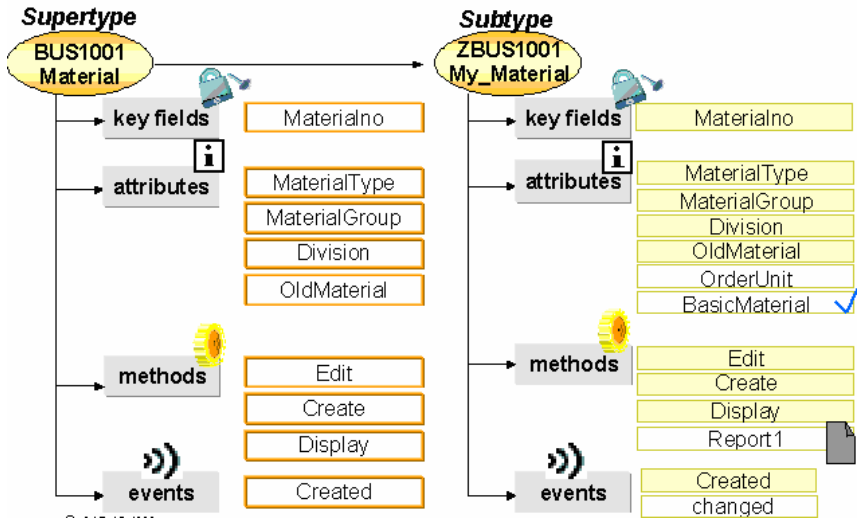
Definición e Implementación de Business Objects

- Delegación



Definición e Implementación de Business Objects

- **Es posible utilizar la DELEGACIÓN para realzar la funcionalidad de los tipos de ojetos SAP.**
- **Los subtipos ofrecen la oportunidad de**
  - **Atributos adicionales**
  - **Métodos adicionales**
  - **Eventos adicionales**
- **Cada tipo de objeto y todos los componentes tiene asignado uno de los cuatro estados posibles:**
  - **Modelado: no existe programa para asignado aún.**
  - **Implementado: el programa ha iniciado pero no finalizado oficialmente.**
  - **Liberado: el programa puede ser ejecutado por todos**
  - **Obsoleto: no utilizar más.**



Definición e Implementación de Business Objects

- Tipos de Objetos Específicos
  - SELFITEM provee la funcionalidad para enviar un correo desde el workflow.
  - SOFM para implementar objetos SAPOffice
  - STD_TEXT puede ser utilizado para integrar texto SAP en un correo enviado en un workflow.
  - Use el tipo de objeto TSTC para ejecutar una transaccón en un paso de workflow.
  - Use el tipo de objeto TRDIR para ejecutar un reporte o programa en un paso workflow.