DIAGRAM SOFTWARE

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

PROGRAMACIÓN DE OPENERP/ODOO

3. WORKFLOWS

José Zambudio Bernabeu
jose.zambudio@diagram.net

NubeaERP global cloud solutions

INTRODUCCIÓN

- Un workflow (flujo de trabajo)

> Es el estudio de los aspectos operacionales de una actividad de trabajo: cómo se estructuran las tareas, cómo se realizan, cuál es su orden correlativo, cómo se sincronizan, cómo fluye la información que soporta las tareas y cómo se le hace seguimiento al cumplimiento de las tareas.

- Más específicamente, un workflow es un grafo en el que los nodos son llamados “activitidades” (activities) y los arcos, que comunican dichos nodos, son llamados “transiciones” (transitions).
  - Las actividades definen que hacer en OpenERP para dicho registro, por ejemplo, modificar un estado o enviar emails.
  - Las transiciones controlan como el workflow progresa de una actividad a otra.
- Además, en un workflow podemos añadir condiciones, señales, triggers a transiciones…

INTRODUCCIÓN

- Como hemos visto hasta ahora, el registro de un workflow viene dado por un record al que vinculamos las actividades y transiciones definidas en este.

```
<record id="workflow_sessionworkflow0" model="workflow">
    <field eval="1" name="on_create"/>
    <field name="name">session workflow</field>
    <field name="osv">openacademy.session</field>
</record>
```

```
<record id="workflow_activity_confirmed1" model="workflow.activity">
    <field name="kind">function</field>
    <field name="name">confirmed</field>
    <field name="join_mode">XOR</field>
    <field name="wkf_id" ref="workflow_sessionworkflow0"/>
    <field eval="0" name="flow_stop"/>
    <field name="split_mode">XOR</field>
    <field name="action">action_confirm()</field>
    <field eval="0" name="flow_start"/>
</record>
```

INTRODUCCIÓN

```
<record id="workflow_activity_draft2" model="workflow.activity">
    <field name="kind">function</field>
    <field name="name">draft</field>
    <field name="join_mode">XOR</field>
    <field name="wkf_id" ref="workflow_sessionworkflow0"/>
    <field eval="0" name="flow_stop"/>
    <field name="split_mode">XOR</field>
    <field name="action">action_reset()</field>
    <field eval="1" name="flow_start"/>
</record>
```

```
<record id="workflow_transition_24" model="workflow.transition">
    <field name="signal">signal_confirm</field>
    <field name="act_from" ref="workflow_activity_draft2"/>
    <field name="act_to" ref="workflow_activity_confirmed1"/>
    <field name="condition">True</field>
</record>
```

INTRODUCCIÓN

- Los workflow son declarados en modelos que poseen un field llamado “state”.
- Atributos:
  - **id=“…”**: Id único para identificar el workflow
  - **model=“workflow”**
  - **name=“…”**: Nombre para el workflow.
  - **osv=“…”**: Modelo al que estamos definiendo el workflow
  - **on_create=“…”**: Indicamos que cree el workflow al crear un registro.

INTRODUCCIÓN

- Siempre definiremos un workflow vinculándolo a un modelo en particular. Esto lo haremos definiendo el field “osv” del record, para el modelo “workflow”
- Los métodos que especifiquemos en las actividades deben definirse en el modelo al que hemos vinculado el workflow.
- En cada nodo (actividad) podemos indicarle a OpenERP si se trata de un nodo fin o de inicio: flow_start, flow_stop.
- Indicamos “on_create” para indicar a OpenERP que dicho workflow debe ser instanciado para cada record nuevo de nuestro modelo. En caso contrario, este debería ser iniciado por otros métodos, por ejemplo por código.

TRANSICTIONS

- Las transiciones nos indican el camino que va a llevar el flujo de trabajo a través de las actividades.
- Cuando una actividad se completa el engine de workflows intentará pasar a la siguiente actividad mediante las transiciones que salgan de la actividad completada. Por tanto las actividades serán procesadas tan pronto como la actividad predecesora este completa.
  - Por otra parte… Podemos indicar al engine que no ejecute las transiciones directamente, sino que espere ciertos eventos, como una **condición**, una **señal** o un **trigger**

TRANSITIONS

- Atributos:
  - **id=“…”**: Id único de la transición.
    - Por convención:
      - trans_“estado desde”_“estado hasta”
  - **model=“workflow.transition”**
  - **act_from=“…”**: Id de la actividad desde
  - **act_to=“…”**: Id de la actividad hasta
  - **signal=“…”**: Nombre de la señal que ejecutará la transición.
  - **condition=“…”**: Condición a evaluar para poder ejecutar la transición.
  - **trigger_model=“…”**: Nombre del modelo para el trigger.
  - **trigger_expr_id=“…”**: Lista de Id’s del modelo trigger_model que ejecutarán la transición.

CONDITIONS - TRANSITIONS

- …Cuando una actividad se ha completado, el engine comprueba las transiciones para acceder a la siguiente actividad…
- En caso de no tener señales o triggers definidos, y tener la transición una condición, el engine evaluará esta, en caso de dar True la transición se ejecuta pasando a la siguiente actividad. En caso de dar False, la transición se evaluará cada vez que el registro se modifique (o en caso de realizar una llamada específica por un método).
- Por defecto el valor de la condición es True.
- Pueden existir varias condiciones (varias líneas), en cuyo caso la última es la que decide si se cumple o no la condición.

CONDITIONS - TRANSITIONS

```
<record id="workflow_transition_26" model="workflow.transition">
    <field name="act_from" ref="workflow_activity_draft2"/>
    <field name="act_to" ref="workflow_activity_confirmed1"/>
    <field name="condition">instructor_id and taken_seats_pct &gt;= 50</field>
</record>
```

```
<record id="trans_router_picking" model="workflow.transition">
    <field name="act_from" ref="act_router"/>
    <field name="act_to" ref="act_picking"/>
    <field name="condition">has_stockable_product()</field>
</record>
```

```
<record id="trans_confirm_mto_purchase" model="workflow.transition">
    <field name="act_from" ref="procurement.act_confirm_mto"/>
    <field name="act_to" ref="act_buy"/>
    <field name="condition">check_buy() and check_supplier_info()</field>
</record>
```

SIGNALS - TRANSITIONS

- …Cuando una actividad se ha completado, el engine comprueba las transiciones para acceder a la siguiente actividad…
- En caso de tener especificada una señal, aunque la condición de la transición sea True, esta no se procesará hasta que no se produzca la señal.
- Una de las formas de ejecutar una señal es mediante un botón. Indicando el nombre de la señal en el atributo “name”. Una vez se hace click sobre el botón, la señal se envía al engine y este ejecuta las condiciones, si estas se cumplen el registro cambia de actividad.

SIGNALS - TRANSITIONS

```
<record id="trans_draft_confirmed" model="workflow.transition">
    <field name="act_from" ref="act_draft"/>
    <field name="act_to" ref="act_confirmed"/>
    <field name="signal">purchase_confirm</field>
</record>
<record id="trans_draft_sent" model="workflow.transition">
    <field name="act_from" ref="act_draft"/>
    <field name="act_to" ref="act_sent"/>
    <field name="signal">send_rfq</field>
</record>
<record id="trans_sent_confirmed" model="workflow.transition">
    <field name="act_from" ref="act_sent"/>
    <field name="act_to" ref="act_confirmed"/>
    <field name="signal">purchase_confirm</field>
</record>
<record id="trans_sent_cancel" model="workflow.transition">
    <field name="act_from" ref="act_sent"/>
    <field name="act_to" ref="act_cancel"/>
    <field name="signal">purchase_cancel</field>
</record>
```

TRIGGERS - TRANSITIONS

- …Cuando una actividad se ha completado, el engine comprueba las transiciones para acceder a la siguiente actividad…
- En caso de evaluarse las condiciones y estas dar un False el engine no pasaría por la transición a la siguiente actividad. A no ser, que definamos los triggers.
- Los triggers, funcionan del mismo modo que en los fields.function. Indicamos una lista de id's de un modelo ya indicado, los cuales cuando sean modificados ejecutarán nuestra transición.

TRIGGERS - TRANSITIONS

```python
def procurement_lines_get(self, cr, uid, ids, *args):
    res = []
    for order in self.browse(cr, uid, ids, context={}):
        for line in order.order_line:
            if line.procurement_id:
                res.append(line.procurement_id.id)
    return res
```

```xml
<record id="trans_ship_ship_end" model="workflow.transition">
    <field name="act_from" ref="act_ship"/>
    <field name="act_to" ref="act_ship_end"/>
    <field name="trigger_model">procurement.order</field>
    <field name="trigger_expr_id">procurement_lines_get()</field>
    <field name="condition">test_state('finished')</field>
</record>
```

ACTIVITIES

- Atributos.
  - **id=“…”**: Id único de la actividad.
  - **model=“workflow.activity”**
  - **“wkf_id” ref=“…”**: Id del workflow al que pertenece.
  - **flow_start**: Indicamos si es el primer estado del workflow.
  - **flow_stop**: Indicamos si es el último estado del workflow.
  - **name**: Indicamos el estado que representa la actividad.
  - **kind**: Indicamos el tipo de acción a realizar cuando se procesa la actividad
    - **dummy**: No realiza ninguna acción.
    - **function**: Ejecuta una función
    - **subflow**: Ejecutar un subflow…
    - **stopall**: Parar el workflow.
  - **action**: Nombre de la función a ejecutar.
  - **subflow_id**: Id del subflow a ejecutar.

FLOW_START && FLOW_STOP - ACTIVITIES

- flow_start && flow_stop
  - Se procesan cuando el workflow es instanciado.
  - Multiples actividades pueden tener flow_start=True. Evaluando de esta forma todas las transiciones.
  - Un workflow es considerado completado cuando todos sus flow_stop tienen valor True.

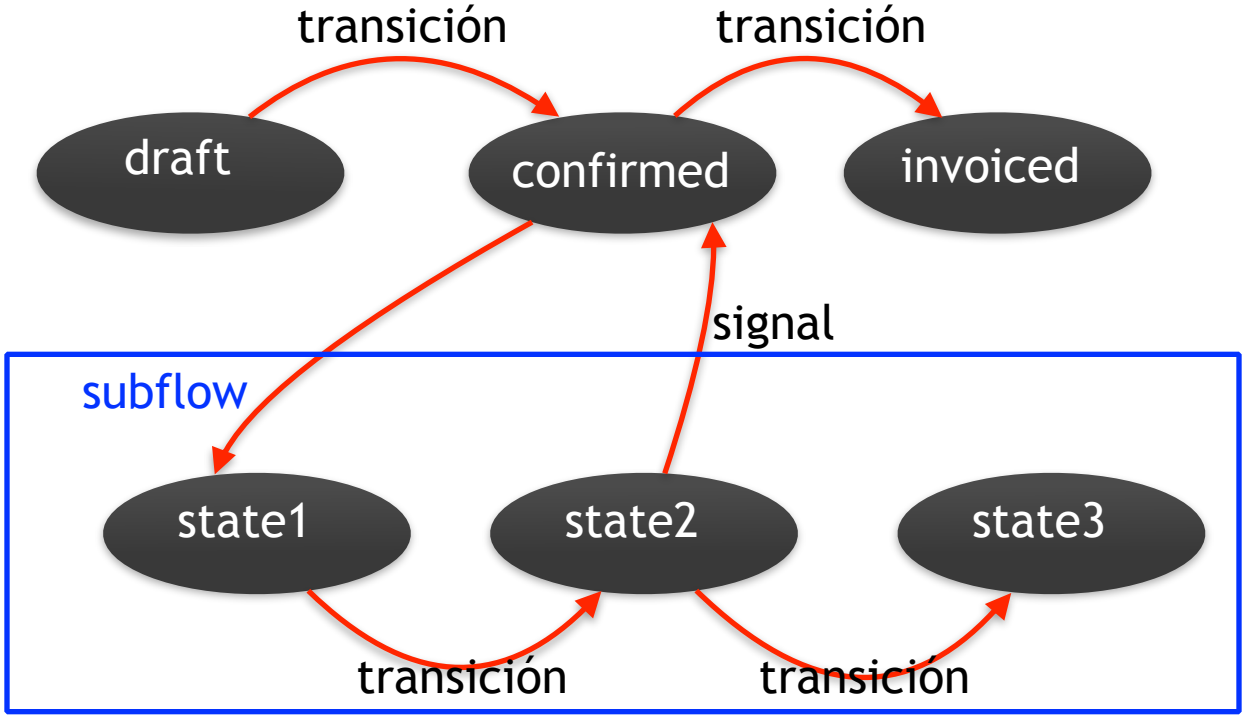
SUBFLOW - ACTIVITIES

- Una actividad puede tener un workflow incrustado (“embed”), al que se le llama subflow.
- La actividad será considerada como completa cuando el subflow se ha completado.
- Recordamos que un workflow se completa cuando TODOS los flow_stop de dicho workflow son TRUE.

```
<record id="act_picking" model="workflow.activity">
    <field name="wkf_id" ref="purchase_order"/>
    <field name="name">picking</field>
    <field name="kind">subflow</field>
    <field name="subflow_id" search="[('osv','=','stock.picking')]"/>
    <field name="action">action_picking_create()</field>
</record>
```

SIGNAL FROM SUBFLOW - ACTIVITIES

- Cuando una actividad contiene un subflow, este tiene la posibilidad de enviar un signal a la actividad padre.



SIGNAL FROM SUBFLOW - ACTIVITIES

```
<record id="act_picking_done" model="workflow.activity">
    <field name="wkf_id" ref="purchase_order"/>
    <field name="name">picking_done</field>
    <field name="action">picking_done()</field>
    <field name="kind">function</field>
    <field name="signal_send">subflow.delivery_done</field>
</record>
```

SPLIT MODE - ACTIVITIES

- Una actividad puede tener varias transiciones hacia varias actividades. Con el atributo split_mode indicamos como deben ser ejecutadas dichas transiciones:
  - **XOR**: Se ejecutarán todas las transiciones y tan pronto como 1 de ellas tenga como condición True, parará la ejecución del resto, evaluando esta.
  - **OR**: Se ejecutarán todas las transiciones y se evaluarán todas aquellas con condición True.
  - **AND**: Se ejecutarán todas las transiciones y sólo se evaluarán cuando todas ellas tengan como condición True.

JOIN MODE - ACTIVITIES

- Al igual que split_mode indica cuando evaluar las transiciones de salida, join_mode nos indica cuando y si debe procesarse una actividad según las transiciones de entrada:
  - **XOR**: En el momento que una transición de entrada tenga la condición True, procesará la actividad.
  - **AND**: La actividad no se procesará hasta que todas las transiciones de entrada no tengan la condición True.

OPENACADEMY

Activities

Activity name

Activity action

Transition

Transition
signal /condition