Process Communication In Petriflow
A Case Study

Milan Mladoniczky, Gabriel Juhás, and Juraj Mažári
An E-Shop example
A process-driven application

- **A use case of a e-shop application**
  An e-shop application created via processes modeled in Petriflow.

- **Application entities as processes**
  Entities, object, of the application expressed as processes.

- **E-shop functionalities as processes**
  All e-shop functionality, like taking orders, registration, shipment of a merchandise, modeled as processes.
An E-Shop example
A process-driven application

✔ A use case of a e-shop application
   An e-shop application created via processes modeled in Petriflow.

✔ Application entities as processes
   Entities, object, of the application expressed as processes.

✔ E-shop functionalities as processes
   All e-shop functionality, like taking orders, registration, shipment of a merchandise, modeled as processes.
The E-Shop processes
Customer entity process

Data-set
ID : String Object Id
Name : String
Address : String
Orders : List<String Object Id>

Current instance = Logged User
Television
Television entity process

Data-set
ID : String Object Id
Model : String

One instance = One TV in stock
Order
Ordering functionality process

Data-set
ID : String Object Id
Television Models : Enumeration
Television reference : String Object Id
Customer reference : String Object Id
Shipment reference : String Object Id
Data-set

ID : String Object Id
Address : String
Television : String Object Id

Shipment
Shipping functionality process
User story

Login → Create and order → Pick a TV → Confirm the choice → Create a shipment
Login

- Registration
- Login
- Logout
- Create a Order
- Show Profile
Create and order

- Registration
- Login
- Logout
- Show profile
- Create an order
Create and order

```xml
<transition>
  <id>4</id>
  <title>Create an order</title>
  <event type="finish">
    <actions phase="post">
      <action>
        orders:f.orders;
        orders <<= createCase(identifier:"order", title:"A new television");
      </action>
    </actions>
  </event>
</transition>
```
Pick a TV

Ship

Show
Pick a TV

<event type="assign">
  <actions phase="pre">
    <action>
      customer:f.customer;
      change customer value {
        return loggedUser().id;
      }
    </action>
  </actions>
</event>
Pick a TV

<event type="assign">
    <actions phase="pre">
        <action>
            customer:f.customer;
            change customer value {
                return loggedUser().id;
            }
        </action>
        <action>
            models:f.tv_models;
            change models choices {
                def tvCases = findCases({
                    it.process.eq("television")
                    .and(it.activePlaces.contains("In stock"))});
                return tvCases.collect({it.dataSet.get("model")}).unique();
            }
        </action>
    </actions>
</event>
Pick a TV

- Arrive
- In stock
- Sell
- Sold
Pick a TV

<event type="assign">
  <actions phase="pre">
    <action>
      customer:f.customer;
      change customer value {
        return loggedUser().id;
      };
    </action>
    <action>
      models:f.tv_models;
      change models choices {
        def tvCases = findCases{
          it.process.eq("television")
          .and(it.tasks.contains({it.transition == “Sell”}));
        return tvCases.collect({it.dataSet.get("model").value}).unique();
      };
    </action>
  </actions>
</event>
Pick a TV

```xml
<data type="enumeration">
  <id>tv_models</id>
  <title>Television models</title>
  <action trigger="set">
    selected: f.this,
    tvRef: f.television;
    def tv = findCase({it.process.eq("television")
      .and(it.dataSet.get("model").value.eq(selected.value))});
    assignTask(transitionId:"Sell", useCase:tv);
    change tvRef value {
      return tv.id;
    }
  </action>
</data>
```
Pick a TV

Order

Pick a TV

Ship

Show

Television

Arrive

In stock

Sell

Sold
Pick a TV

<data type="enumeration">
  <id>tv_models</id>
  <title>Television models</title>
  <action trigger="set">
    selected:f.this,
    tvRef:f.television;
    def tv = findCase({it.dataSet.get("model").eq(selected.value)});
    if(tv){
      assignTask(transitionId:"Sell", useCase:tv);
      change tvRef value {
        return tv.id;
      }
    }
  </action>
</data>
Pick a TV
Pick a TV

<data type="enumeration">
  <id>tv_models</id>
  <title>Television models</title>
  <action trigger="set">
    selected:f.this,
    tvRef:f.television;
    if(tvRef.value){
      def oldTv = findCase({it.id.eq(tvRef.value)});
      cancelTask(transitionId:"Sell", useCase:oldTv);
    }
    def tv = findCase({it.dataSet.get("model").eq(selected.value)});
    if(tv){
      assignTask(transitionId:"Sell", useCase:tv);
      change tvRef value {
        return tv.id;
      };
    }
  </action>
</data>
<event type="finish">
  <actions phase="post">
    <action>
      tvRef:f:television;
      def tv = findCase({it.id.eq(tvRef.value)});
      finishTask(transitionId:"Sell", useCase:tv);
    </action>
  </actions>
</event>
Confirm the choice

<event type="finish">
  <actions phase="post">
    <action>
      tvRef:f.television;
      def tv = findCase({it.id.eq(tvRef.value)});
      finishTask(transitionId:"Sell", useCase:tv);
    </action>
  </actions>
</event>
Create a shipment

```xml
<event type="finish">
  <actions phase="post">
    <action> ... </action>
    <action>
      tvRef:f.television,
      shipRef:f.shipment;
      def ship = createCase(identifier:"shipment", "Order "+useCase.id+" shipment");
      change shipRef value {
        return ship.id;
      };
      def shipTask = assignTask(transitionId:"Initialize", useCase:ship);
      setData(shipTask,[
        "address":loggedUser().dataSet.get("address").value,
        "television":tvRef.value
      ]);
    </action>
  </actions>
</event>
```
Create a shipment

[Diagram showing a process flow with nodes labeled Initialize, Check for defect, Repack, and Send]
THANK YOU