Petriflow in Actions:
Events Call Actions Call Events

Juraj Mažári, Gabriel Juhás, and Milan Mladoniczky

STU FEI

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

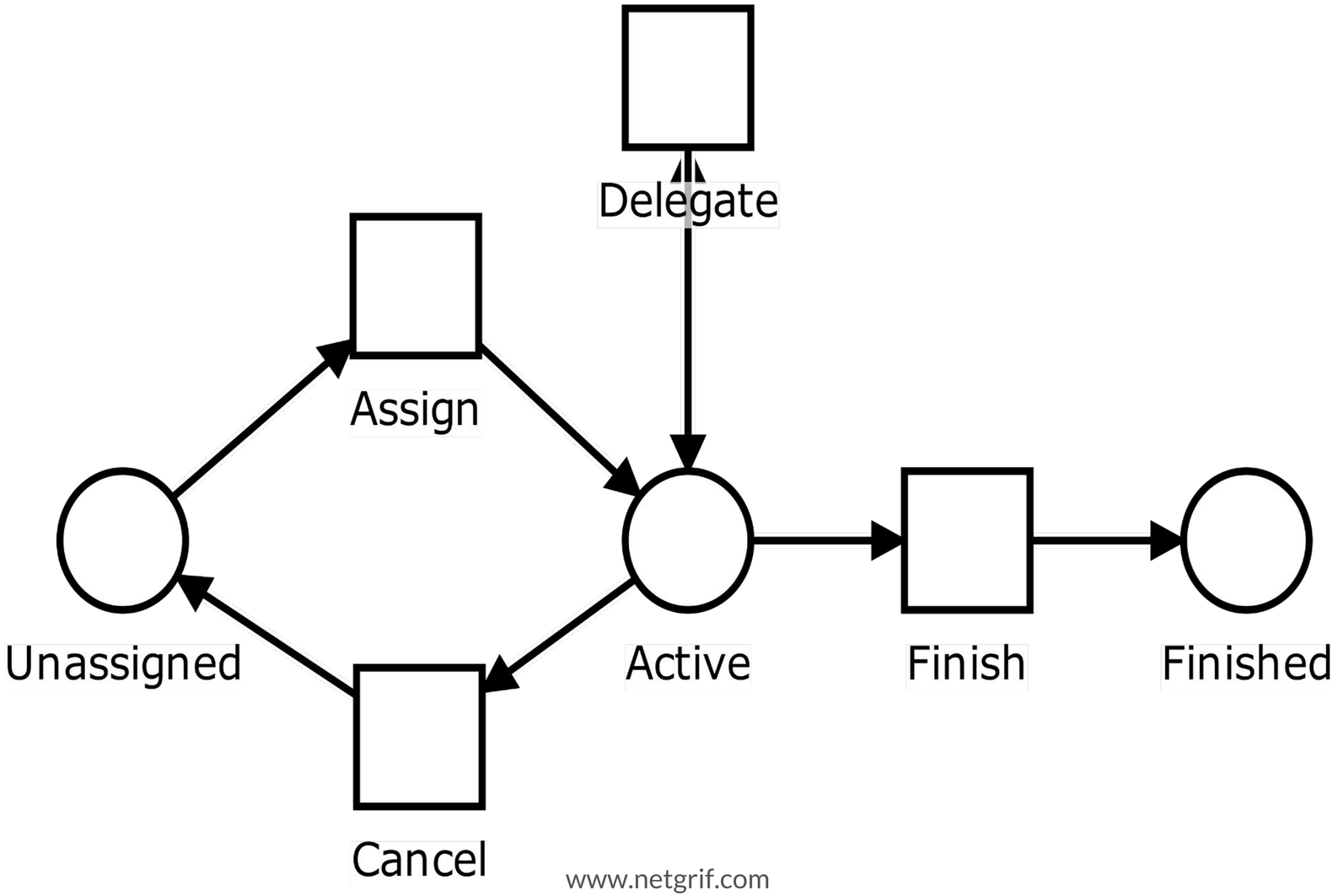
Petriflow

Petri Net + Roles + Data + Actions

www.netgrif.com

Petriflow

Task net



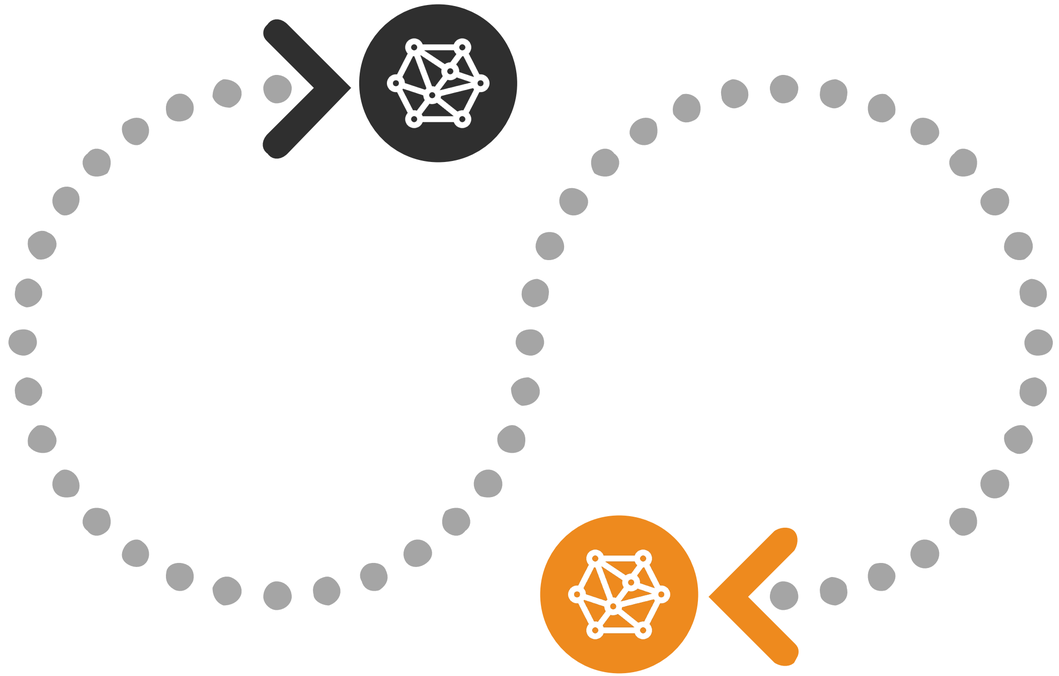
Inter-Process Communication

Interface between models

- Transitions and its events
- Each task event can be triggered by user or from another process using actions

Actions

- Groovy code executed at different places in the model
- Can be triggered by event
- Can trigger event



Actions

Triggering actions on event

Assign task
Send notification email that work has begun

Delegate task
Delegate all my tasks to someone else

Cancel task
Close an exam question when timeruns out

Finish task
Inform parent case that you have finished your work

Read data
Generate new version of PDF file

Save data
Calculate price after setting a discount

Before event
If failure in execution of the action should prevent the event

After event
If actions should be executed after the change in marking caused by the event

Finding objects

Search for any entity

```
List<T> find'T's(Closure<Predicate> predicate)
List<Case> findCases(Closure<Predicate> predicate)

T find'T'(Closure<Predicate> predicate)
Case findCase(Closure<Predicate> predicate)
```

Find all

Find one

Finding objects

Search for any entity

```
List<Case> cases = findCases( {
 it.dataSet.get( ”email” ).eq( ”mazari@netgrif.com” )
} )
Case aCase = findCase ( {
 it.author.id.eq( loggedUser().id )
} )
```

Find all

Find one

Creating new instances


```
Case createCase(
        String identifier, String title = null,
        String color = "", User author = system
)
Case createCase(
        PetriNet net,
        String title = net.defaultCaseName,
        String color = "",
        User author = userService.loggedOrSystem
)
```

Creating new instances


```
Case aCase = createCase( ”insurance” )

Case aCase = createCase( ”insurance”, ”My insurance” )

Case aCase = createCase(
      identifier : ”insurance”,
      author : someUser
)
```

Triggering task events


```
Task assignTask(String transitionId,
        User user = userService.loggedOrSystem,
        Case useCase = currentCase)
Task assignTask(Task task,
        User user = userService.loggedOrSystem)
...
cancelTask(...)
finishTask(...)
```

Triggering task events


```
if (email.value != null ) {
   def user = findUser( it.email.eq ( email.value ) )
   def nC = createCase(
      identifier : ”some_net”,
      author : user
   )
   assignTask( ”first_task”, user, nC )
}
```

Reading & writing data


```
Map<String, Field> getData(Task task, User user =
        userService.loggedOrSystem)

Map<String, Field> getData(Transition transition,
        User user = userService.loggedOrSystem)

Map<String, Field> getData(String trId, Case useCase,
        User user = userService.loggedOrSystem)
```

Reading & writing data


```
setData(Task task, Map dataSet, User user =
        userService.loggedOrSystem)

setData(Transition transition, Map dataSet,
        User user = userService.loggedOrSystem)

setData(String trId, Case useCase, Map dataSet,
        User user = userService.loggedOrSystem)
```

Reading & writing data


```
def useCase = findCase( { it.title.eq ( "Limits" ) } )
def data = getData( "view_limit", useCase )
change actual_limit value {
   data[ "remote_limit" ].value
}

def aCase = findCase( { it.title.eq ( "Limits" ) } )
assignTask( "edit_limit", loggedUser(), aCase )
setData( "edit_limit", aCase, ["newlimit" : 10000] )
```

Conclusion

- Inter-process communication is the future of Petriflow
- Simple enough for customers to understand it and also simple to model at the same time
- Example applications:
  - hotel reservations,
  - **online shopping**,
  - accounting information system,
  - ...