Fear and loathing in Alfresco workflow development

Alexey Ermakov @ ITD Systems

Workflow development lifecycle

- Create a process using plugin for Eclipse
- Create workflow model and l10n
- Create Share config and l10n
- Write some Java code
- Deploy to test server

Not that we needed all that for the development, but once you get locked into a serious business, the tendency is to push it as far as you can

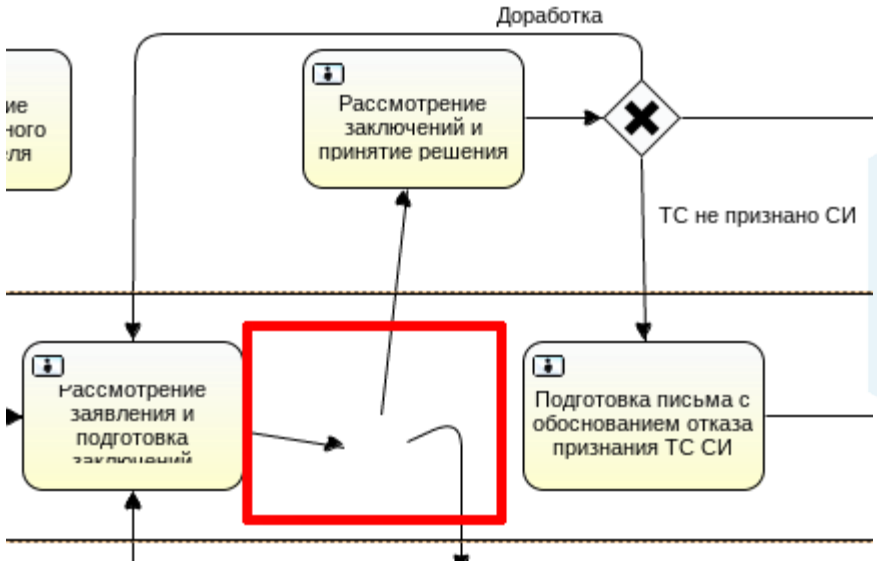
Workflow development lifecycle

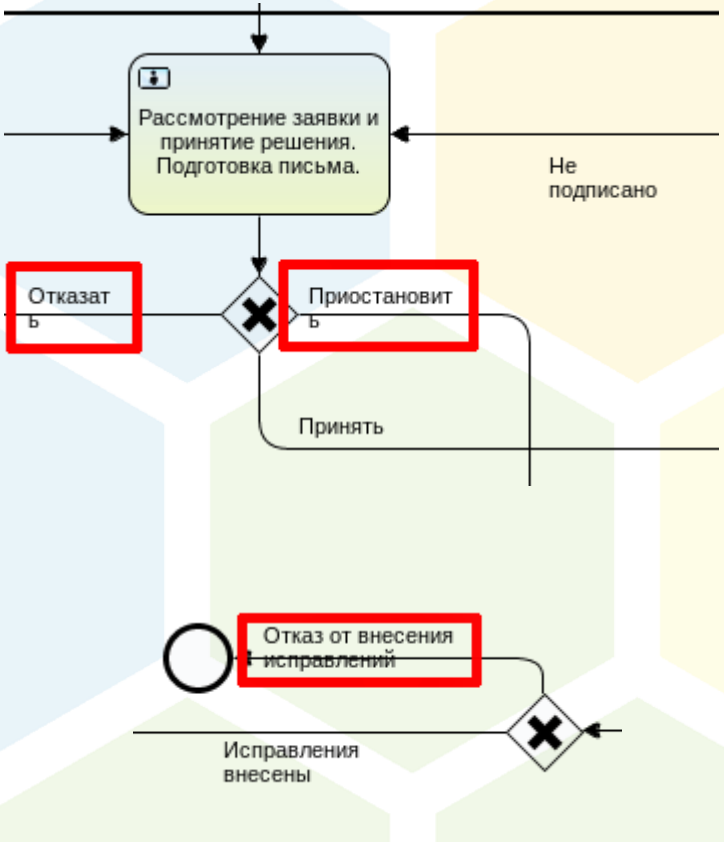
- Create a process using plugin for Eclipse
- Create workflow model and l10n
- Create Share config and l10n
- Write some Java code
- Deploy to test server

Not that we needed all that for the development, but once you get locked into a serious business, the tendency is to push it as far as you can

Step 1: create a process using Eclipse

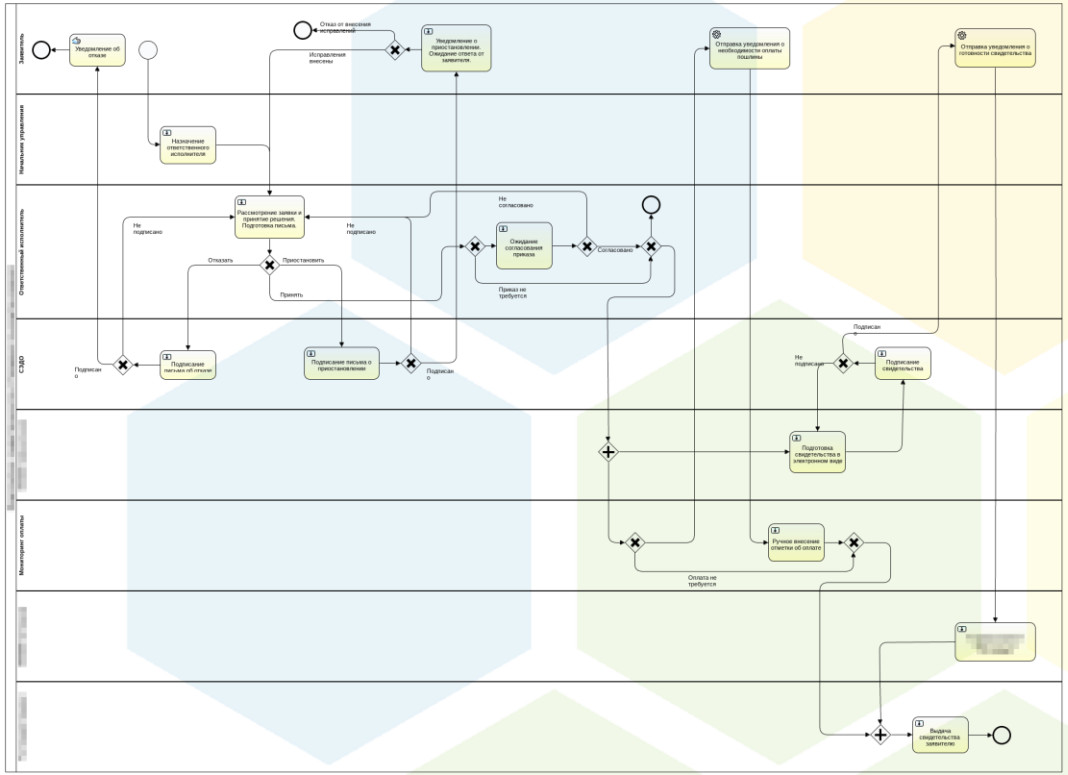
It’s quite easy until something goes wrong…





Step 2: create workflow model

- For each task create type
- Specify outcome constraints for all tasks followed by exclusive gateways
- Add necessary aspects
- Override property values



aconfgen: generated task model

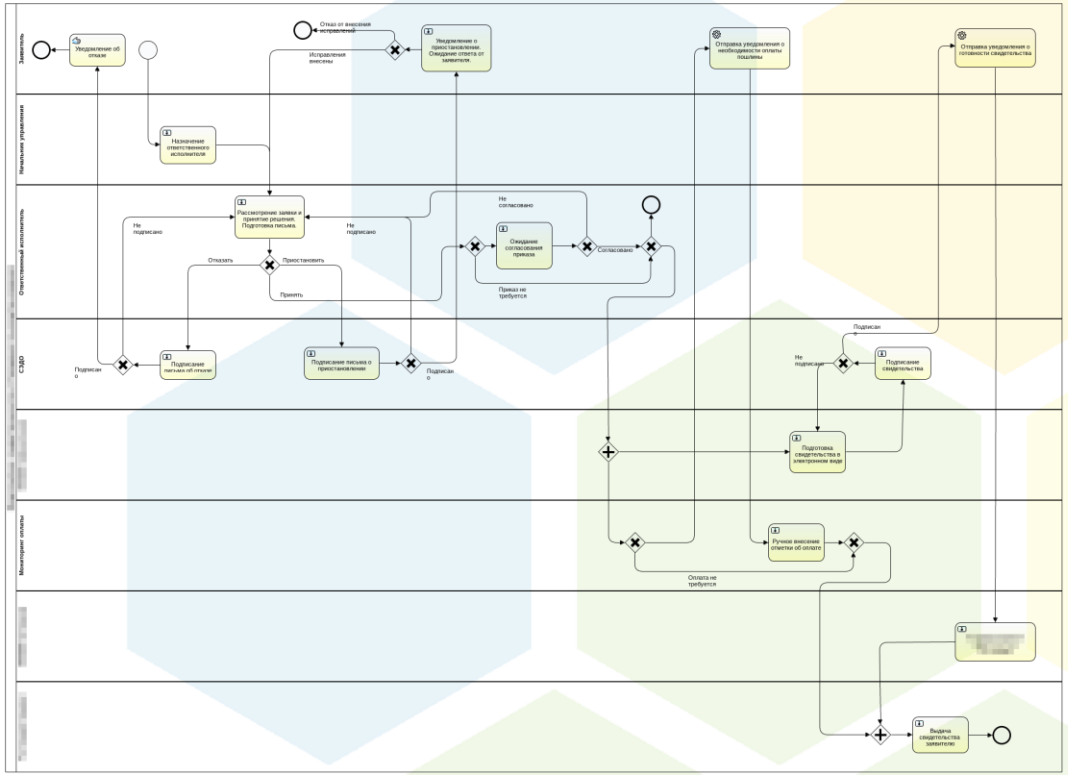
```
<model ... name="guwf:samplemodel">
  <imports>...</imports>
  <namespaces>...</namespaces>
  <types>
    <type name="gu2wf:makeDecisionTask">
      <parent>bpm:activitiOutcomeTask</parent>
...
            <constraint type="LIST" name="...">
              <parameter name="allowedValues">
                <list>
                  <value>signRejection</value>
                  <value>signSuspension</value>
                  <value>exclusivegateway7</value>
                </list>
...
```

aconfgen: generated task model l10n

```
guwf_samplemodel.title=
guwf_samplemodel.description=
guwf_samplemodel.type.gu2wf_prepareDigitalCetificateTask.title=
guwf_samplemodel.type.gu2wf_prepareDigitalCetificateTask.description=
...
guwf_samplemodel.property.gu2wf_prepareDigitalCetificateTaskOutcome.title=
guwf_samplemodel.property.gu2wf_prepareDigitalCetificateTaskOutcome.description=
...
listconstraint.gu2wf_makeDecisionTaskOutcomeConstraint.signRejection=
listconstraint.gu2wf_makeDecisionTaskOutcomeConstraint.signSuspension=
...
```

Step 3: create Share config and l10n

- For each task create form config
- Create form config for start task
- Create l10n for all **label-id**’s in config



aconfgen: generated Share config skeleton

```
<config evaluator="task-type" condition="gu2wf:makeDecisionTask">
...
       <field-visibility>
          <show id="gu2wf:makeDecisionTaskOutcome"/>
...
        </field-visibility>
        <appearance>
          <set id="info" appearance="" label-id="workflow.set.task.info"/>
...
          <field id="..." set="response" label-id="..."/>
...
        </appearance>
...
  </config>
```

aconfgen: generated Share localization

```
workflow.field.packageItems=
workflow.field.gu2wf_makeDecisionTaskOutcome=
workflow.field.transitions=
```

Step 4: write some Java code

It’s quite easy until you try to do something non-trivial…

- Java foundation APIs are too basic
- You have to use Activiti API to create something non-trivial
- You have to use strange tricks to create something non-trivial
- Most probably you will not find helpful examples on forums if you do something non-trivial

Step 4: write some Java code

It’s quite easy until you try to do something non-trivial…

- Java foundation APIs are too basic
- You have to use Activiti API to create something non-trivial
- You have to use strange tricks to create something non-trivial
- Most probably you will not find helpful examples on forums if you do something non-trivial

Simple task: create a task listener

```
<activiti:taskListener event="create" class="...">

  <activiti:field name="script">

        <activiti:string>

          if (typeof bpm_workflowDueDate != 'undefined')

              task.dueDate = bpm_workflowDueDate;

        </activiti:string>

  </activiti:field>

</activiti:taskListener>
```

Simple task: solution

- Create you own BPMNParseHandler
- Inject it to Activiti
- Create all desired task listeners when process definition is being parsed

Straight and forward, right?

Step 4.5: write some more Java code

To reduce amount of problems caused by typos it’s better to use

```
delegateTask.getExecution().setVariable(Constants.MYWF_PROP_WHATEVER, "42");
```

instead of

```
delegateTask.getExecution().setVariable("mywf_propWhatever", "42");
```

Step 4.5: write some more Java code

Constants generated from model definition:

```
public static final QName ASSOC_P06WF_SYSTEM_USAGE_SUMMARY = ...
public static final QName ASSOC_P06WF_SYSTEM_CHARACTERISTICS = ...
public static final QName ASSOC_P06WF_SYSTEM_SOFTWARE_INFORMATION = ...
public static final QName ASSOC_P06WF_SYSTEM_USAGE_REQUIREMENTS = ...
public static final QName ASSOC_P06WF_SYSTEM_CONNECTION_SCHEME = ...
public static final QName ASSOC_P06WF_SYSTEM_INTERNAL_MEASUREMENTS = ...
public static final QName ASSOC_P06WF_ADDITIONAL_MATERIALS = ...
```

Step 5: deploy

Recommended ways of running Alfresco during development:

- Deploy AMPs / JARs to installed Alfresco
- Use mvn –P run install

What we definetely need:

- Official Docker support

Step 5: deploy

Recommended ways of running Alfresco during development:

- Deploy AMPs / JARs to installed Alfresco
- Use mvn –P run install

What we definetely need:

- Official Docker support

Summary

- Alfresco should pay some attention to Activiti designer
- Activiti integration should be revised
- Workflow APIs should provide not only basic functionality

Summary

- Alfresco should pay some attention to Activiti designer
- Activiti integration should be revised
- Workflow APIs should provide not only basic functionality

My current plans

- Port aconfgen to Java and build a Maven plugin
- Create Maven plugin that generates Java representation of content models
- Configure auto-builds for my Docker images

Any help is appreciated

Speaker contacts

aconfgen: github.com/fufler/aconfgen
IRC: fufler
Email: aermakov@itdhq.com