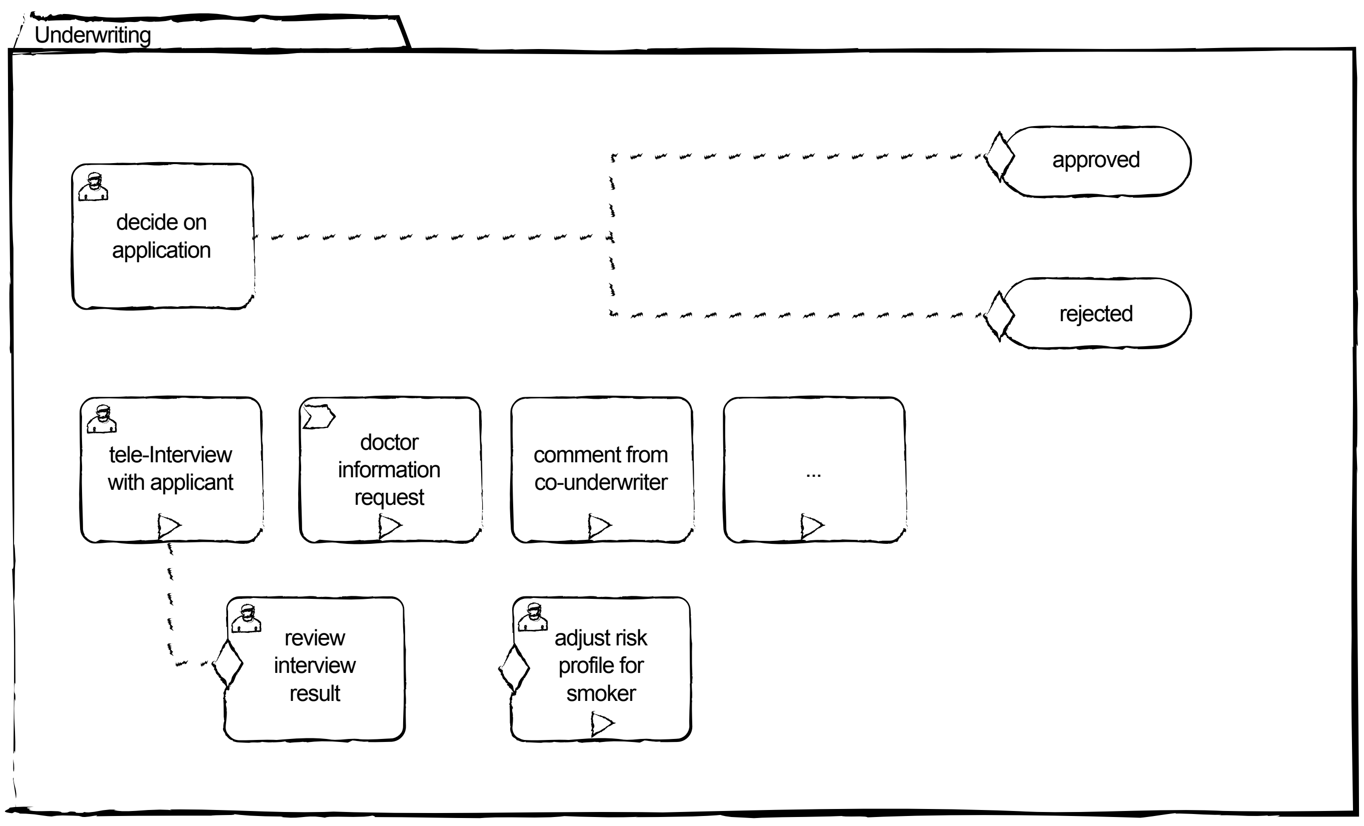
Case Management and CMMN for Developers

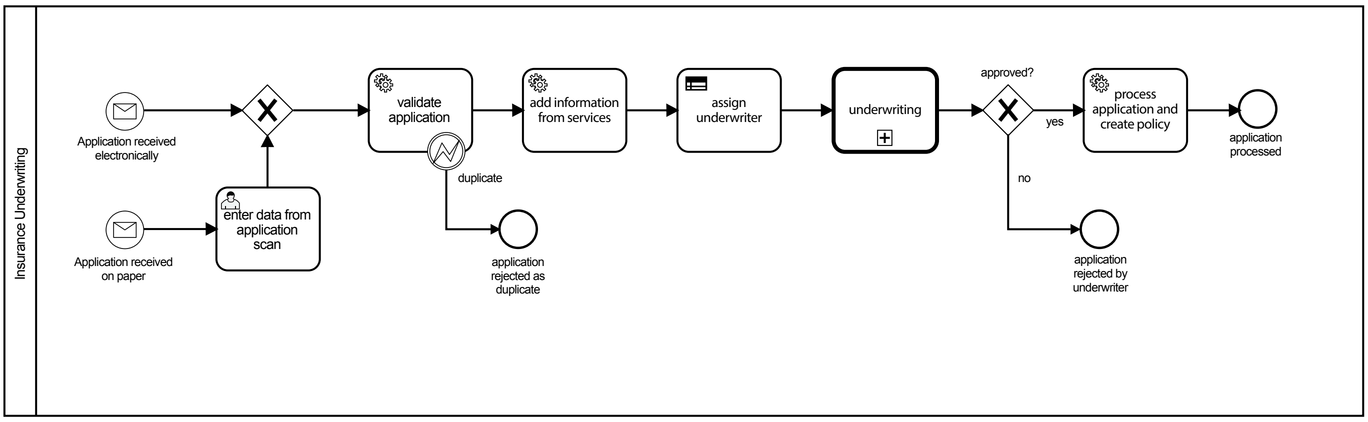
Workflow engines have arrived in developer toolboxes. The BPMN 2.0 standard is typically used. When automating, however, one might come across situations in which humans must be able to affect the process.


Underwriting
decide on application
approved
rejected
tele-Interview with applicant
doctor information request
comment from co-underwriter
...
review interview result
adjust risk profile for smoker

Adding flexibility to BPMN (Business Process Model and Notation) processes is sometimes difficult. Therefore, the Object Management Group has created its own standard CMMN 1.0 (Case Management Model and Notation), which – just like BPMN – is executable in workflow engines. This article introduces CMMN and shows its implementation using the Camunda open source BPM system as an example.

Acceptance (or rejection) of private health insurance shall serve as a specific example. The complete source code, including the associated BPMN and CMMN models is available on GitHub[1] and can be executed or tried directly in Camunda's open source BPM platform.[2]

At first glance, the "underwriting" process design looks quite structured and can therefore be mapped with BPMN:


Insurance Underwriting
Application received electronically
Application received on paper
enter data from application scan
validate application
duplicate
application rejected as duplicate
add information from services
assign underwriter
underwriting
approved?
yes
no
process application and create policy
application processed
application rejected by underwriter

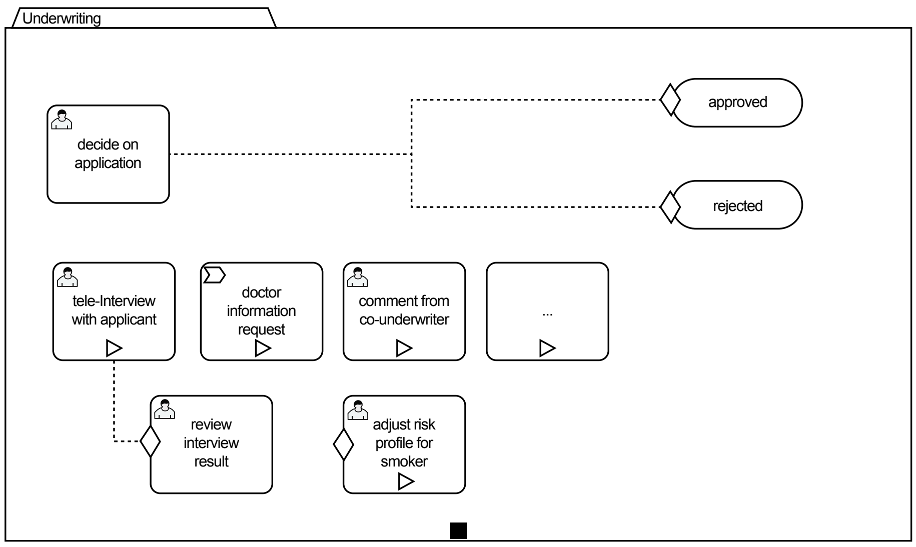
Parts of the "underwriting" process can be easily modeled in BPMN (Fig. 1).

The resulting model can be executed directly by a process engine. The first difficulties arise when looking at the real "underwriting", which is represented in the process by a "Call Activity". At this stage, the clerks have several options. They can contact the GP to find out about pre-existing conditions, which in turn may require contacting a specialist. For larger risks they might need the advice of a colleague and ask for his assessment. Maybe he suggests calling the customer to retrieve information and so on. These activities cannot be forced into a standardized order. Instead, the clerk will need to be assigned the decision making power. In BPMN, this can only be represented with workarounds which will ensure that the model explodes when it comes to more complex scenarios.

1 https://github.com/camunda/camunda-consulting/tree/master/showcases/underwriting
2 http://camunda.org

Enter CMMN

This is where CMMN can help. The standard was released in May 2014 by the OMG in Version 1.0 and is implemented in version 7.2 of the Camunda BPM platform. The following figure shows the case definition for underwriting:


Underwriting
decide on application
approved
rejected
tele-Interview with applicant
doctor information request
comment from co-underwriter
...
review interview result
adjust risk profile for smoker

The unstructured part of the underwriting process can be modeled in CMMN (Fig. 2).

The symbols are similar to those of BPMN. In addition to the case as a whole there are human tasks (resulting in tasks for a task list), process tasks (when BPMN processes are started within a case), milestones and more. All notation elements can be looked up online.[3]

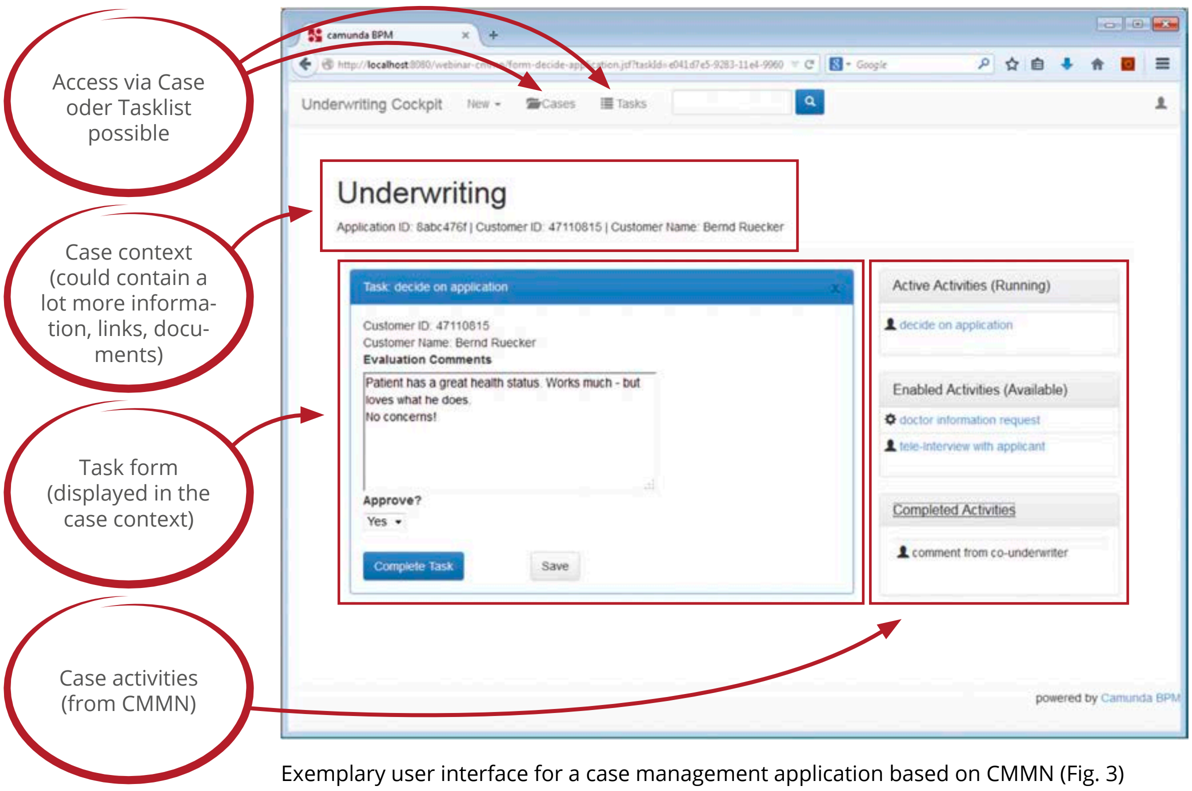
Just like BPMN 2.0, the CMMN 1.0 model is stored as an XML file and can be executed directly by the engine. So far, there is only one graphical modeling tool that masters CMMN – the CMMN Web Modeler.[4] With increasing acceptance of CMMN this should change quickly though. Camunda is already planning a modeling tool based on its bpmn.io project.[5]

The logic behind CMMN can be illustrated best when you look at the resulting user interface of a system:

3 http://docs.camunda.org/latest/api-references/cmmn10/

4 http://cmmnwebmodeler.com/

5 http://bpmn.io/


Access via Case oder Tasklist possible
Case context (could contain a lot more information, links, documents)
Task form (displayed in the case context)
Case activities (from CMMN)
camunda BPM
Underwriting Cockpit
New
Cases
Tasks
Complete Task
Save
powered by Camunda BPM

http://localhost:8080/webinar-cmmn/form-decide-application.jsf?taskId=e041d7e5-9283-11e4-9960

Underwriting

Application ID: 8abc476f | Customer ID: 47110815 | Customer Name: Bernd Ruecker

Task: decide on application

Customer ID: 47110815
Customer Name: Bernd Ruecker
Evaluation Comments
Patient has a great health status. Works much - but loves what he does.
No concerns!

Approve?
Yes

Active Activities (Running)
- decide on application

Enabled Activities (Available)
- doctor information request
- tele-interview with applicant

Completed Activities
- comment from co-underwriter

Exemplary user interface for a case management application based on CMMN (Fig. 3)

The figure above shows a GUI implemented with JavaServer Faces (the source code is also included in the GitHub project) that was extracted from the pilot of a real project. It is divided into several areas:

- Context: To enable the user (case management uses the term “knowledge workers”) to make good decisions, he or she needs to have access to as much information about the current case as possible. This can, for example, be master data, linked entities, documents or notes.
- Activity overview: A central part of CMMN are the activities (e.g. human task or process tasks). They are subject to a life cycle, which will soon be explained in more detail. This means that activities are either active (“active”, i.e. currently running) or available (“enabled”). Available activities can be started manually by humans. This converts them to an active state.
- Task forms: If a human task is executed, a form is defined which, for example, enables completion of tasks by entering relevant data.

In the life cycle of an activity CMMN, it mainly differentiates between “available”, “enabled”, “active” and “completed”. An activity in the CMMN model can be configured accordingly in order to influence the life cycle. Some activities, for example, show a small play icon. This means that they have to be started manually by humans. Consequently, these tasks are only “enabled” and not directly “active”. Other tasks are activated directly.

Exceptions are activities with a so-called sentry guard. They are marked by a small diamond. The sentry decides whether an activity is made available at all - otherwise it remains “available” and won‘t appear in the user interface. The status “available” at this point is unfortunate and misleading: It is the case that an “available” task must first be switched to “enabled” or “active” before it can be executed. It is therefore unavailable for knowledge workers, unless released by the sentry.

There are two possible condition types for the sentry: The availability of the activity may depend on the completion of another task, which is represented by a dotted line. It‘s Important to note that this is not a sequence flow as in BPMN, but merely visualizes the causal dependency. The sentry can also check any condition while using context data. In the example, the information whether the client is a smoker or not controls whether the option to adjust his risk profile is available. The information depends on the application object in the context of the case. Therefore, the interface does not make the tasks “review interview result” and “adjust risk profiles for smoker” visible by default.

This allows a relatively easy definition of a “menu” for the knowledge worker. Important rules can be added to the model to make the knowledge worker‘s job as simple as possible. At runtime, the person then has control again - within the limits set by CMMN. If stages and milestones are now used as well, one can define very extensive cases.

Real-life areas of application

Typical scenarios for CMMN are found in insurances such as applications or claims settlement processes. Another field of application are medical claims such as treatment of a patient in hospital or in a rehabilitation facility. Otherwise CMMN can also be used for exception handling. In these situations the fully automated process flow can be interrupted in order to clarify a problem. Examples include allocation problems for automated document input processes, reading problems in scanning routes, exception handling in call centers and so on.

CMMN can also be used as a step on the way to a BPMN model as the direct full automation of a hitherto manual process might not be realized by one big bang. In this case, the clerks have more options of engagement via CMMN to increase acceptance. The degree of automation can then gradually be increased. This can be further supported by evaluating the CMMN engine‘s collected audit data.

Looking into the source code

For a better understanding, Camunda BPM will serve as a specific example. Camunda, unlike “zero code suites”, focuses on developer-friendliness. Accordingly, you can start up a process engine with standard configuration and in-memory database with a single line of Java:

```
ProcessEngine engine = new
StandaloneInMemProcessEngineConfiguration().buildProcessEngine();
```

Afterwards you can use the Camunda engine‘s Java API, which is shown in the article on the basis of a JUnit test case. The interface developed in JSF uses the same API to read information or to initiate changes to the case instance. The engine can be just as easily controlled via REST API, so that it can be used in a host of other scenarios. Like for instance in JavaScript, .NET or PHP applications.

The following code shows a section of a test case that is starting a new case instance. If you‘re familiar with Camunda, you will recognize the parallels to BPMN process instances. CMMN was indeed integrated into the existing engine and can be closely interlinked with BPMN.

```
// HashMap containing variables to be added to case instance
VariableMap variables = Variables.createVariables().putValue("application",
new Application());

// create the case instance, this persists it in the database
CaseInstance caseInstance = processEngine.getCaseService().createCaseInstanceByKey(
"underwriting", variables);

// load tasks via Query API and assert that we have one task
// created (corresponding to one active Activity)
List<Task> tasks = processEngine.getTaskService().createTaskQuery().list();
assertEquals(1, tasks.size());

// use camunda assertions to check we created the right task:
assertThat(tasks.get(0)).hasDefinitionKey("PI_humanTaskDecide");

// load all so called "executions" of a Case Instance, basically
// corresponding to Activities in our case
List<CaseExecution> caseExecutions =
processEngine.getCaseService().createCaseExecutionQuery().list();
for (CaseExecution caseExecution : caseExecutions) {
       if (caseExecution.isEnabled()) {
              System.out.println("Possible to start (‘enabled‘): " +
              caseExecution.getActivityName() + " [" +
              caseExecution.getActivityType() + "]");
              } // else...
       }

// complete task - changes the state of the case instance
processEngine.getTaskService().complete(tasks.get(0).getId());

// we could check completed activities now:
processEngine.getHistoryService().
createHistoricCaseActivityInstanceQuery()
.caseInstanceId(caseInstance.getId()).list();
```

Data can be passed to a case instance, and that data can be used by a sentry to formulate conditions. Camunda uses JUEL (Java Unified Expression Language), so that the mentioned example, whether the applicant is a smoker or not, is relatively simple - as long as it can be assumed that there is a corresponding attribute in the “Application” class.

A look at the CMMN XML below shows how an activity with sentry is defined. As you can see, the XML format is not complicated. Furthermore, all activities in CMMN can be reused at different stages in the model. That is why the human task is defined separately and included as a so-called plan item in the case.

```
<cmmn:planItem id="PI_humanTaskSmoker" definitionRef="humanTaskSmoker"
 entryCriteriaRefs="sentrySmoker" />

<cmmn:sentry id="sentrySmoker">
  <cmmn:ifPart>
    <cmmn:condition>
      <cmmn:body>${application.smoker}</cmmn:body>
    </cmmn:condition>
  </cmmn:ifPart>
</cmmn:sentry>
<cmmn:humanTask id="humanTaskSmoker" name="adjust risk profile for
 smoker" isBlocking="true" camunda:assignee="demo"
 camunda:formKey="app:adjust-smoker-risk-profile.jsf" />
```

When executing the unit test, you can look at the status of the activities in the case after the start on the console or graphically via a CMMN monitoring plug-in for the Camunda cockpit[6], the platform's browser based monitoring tool.

Status overview

Activity name	Activity type	Status	Comments
review interview result	Human Task	Available	can only be started after 'tele-interview'
adjust risk profile for smoker	Human Task	Available	can only be started with smokers
approved	Milestone	Available	can only be achieved if 'decide on application' is completed with 'approve'
rejected	Milestone	Available	can only be achieved if 'decide on application' is completed with 'reject'
comment from co-underwriter	Human Task	Enabled	can be started manually
tele-Interview with applicant	Human Task	Enabled	can be started manually
doctor information request	Process Task	Enabled	can be started manually
decide on application	Human Task	Active	running
Underwriting	Case	Active	entire case instance is also running

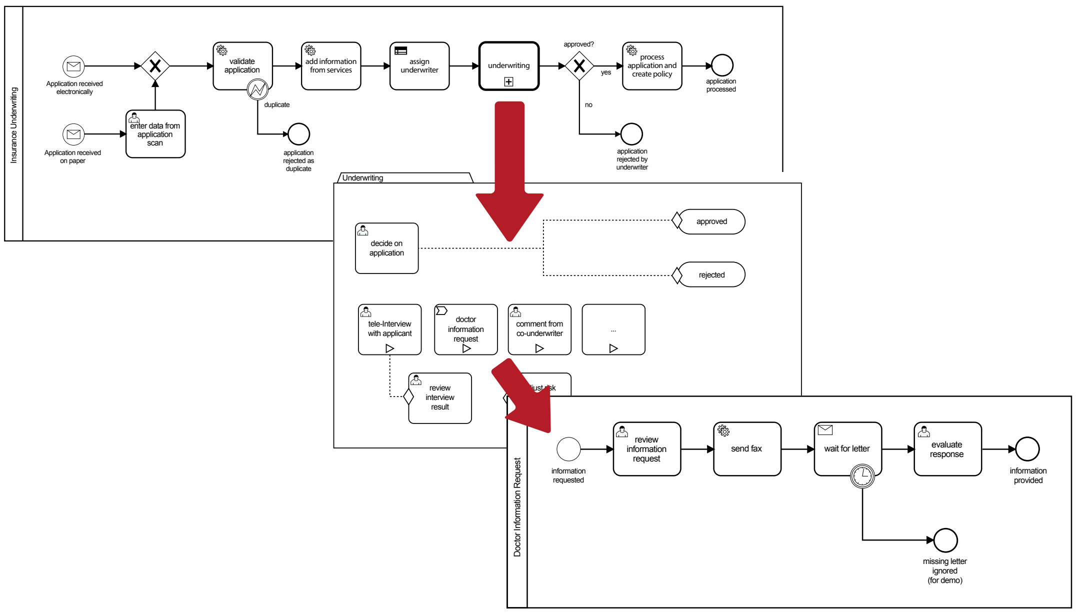
6 https://github.com/camunda/camunda-acm-plugin/tree/master/

If then, for example, the “tele-interview” is conducted with the customer and the corresponding task completed, the activity “review interview result” becomes available. Once all relevant tasks have been completed (there are additional attributes that are not considered further in this article), the case instance ends.

This should suffice as a brief introduction to CMMN, if interested the entire example can be downloaded.[7] To use it, a current Camunda JBoss distribution[8] should be used (JBoss because of the JSF user interface). Alternatively, a CMMN model can be created from scratch. An online tutorial[9] is available.

BPMN + CMMN

There are cases in which business processes should be treated as a whole in CMMN. Experience shows, however, that it is often useful to combine BPMN and CMMN. The following figure shows an example, where the overall process is structured, the decision phase is unstructured and individual activities such as requesting information from the GP is structured again.



Insurance Underwriting
Application received electronically
Application received on paper
enter data from application scan
validate application
duplicate
application rejected as duplicate
add information from services
assign underwriter
underwriting
approved?
yes
no
process application and create policy
application processed
application rejected by underwriter
Underwriting
decide on application
approved
rejected
tele-Interview with applicant
doctor information request
comment from co-underwriter
...
review interview result
Doctor Information Request
information requested
review information request
send fax
wait for letter
evaluate response
information provided
missing letter ignored (for demo)

BPMN and CMMN can be combined in order to always be able to use the right tool for the task (Fig. 4).

7 https://github.com/camunda/camunda-consulting/tree/master/showcases/underwriting
8 http://camunda.org/download/
9 http://docs.camunda.org/latest/guides/getting-started-guides/cmmn/

Currently the standards allow only the link from CMMN to BPMN, but not the other way round. The reason for this is that BPMN has been standardized before CMMN was even available. It can be expected that the other way will also be standardized. Currently, one can realize the cooperation in the reverse direction with workarounds or vendor extensions.

The combination of the two standards is important as it offers the use of the right tool for the right task. There are structured and unstructured processes or process parts, in which one is struggling when using the wrong tool. That is also the reason why Camunda combines both standards in one engine. For the future, it is not completely unrealistic that the standards will converge.

Conclusion

CMMN is a good addition to BPMN for processes that require a high degree of flexibility. The specification combines the experience of many manufacturers and consultants and is very useful. Since the standard is still young, few useable examples and useful tools exist. But this will certainly change this year. Due to the similarity of the symbols to those of BPMN they should quickly gain acceptance. For someone who starts a new project with unstructured process parts, a look at CMMN can be very worthwhile.

Links & Literature

1. https://github.com/camunda/camunda-consulting/tree/master/showcases/underwriting
2. http://camunda.org
3. http://docs.camunda.org/latest/api-references/cmmn10/
4. http://cmmnwebmodeler.com/
5. http://bpmn.io/
6. https://github.com/camunda/camunda-acm-plugin/tree/master/
7. https://github.com/camunda/camunda-consulting/tree/master/showcases/underwriting
8. http://camunda.org/download/
9. http://docs.camunda.org/latest/guides/getting-started-guides/cmmn/