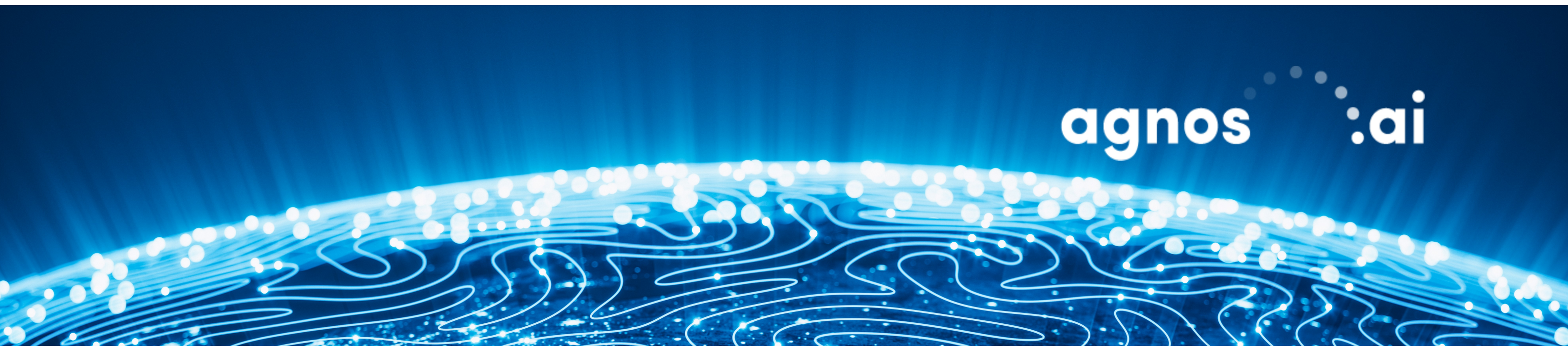
EKG Workflow

Ontology Design for Workflow capability in the Enterprise Knowledge Graph

June 3, 2021, Version 0.1.110, FINAL, Confidential



Contents

0.1 Design Goals

The scope of this service is to support multiple users interacting with each other to take work items through a series of stages of change, review and approval. Key aspects are:

- A controlled transition of stages with both human and automated checking
- Routing, notification and entitlement of different users to take action at each stage
- Visibility of the process and specifically the current stage of any item
- Full auditing of what actions were taken when and by whom

The following represent goals for the Workflow capability. These are not all required of an initial implementation but are intended to be considered as part of the architecture.

- Model-based workflow definition
- Model-based workflow enactment and audit
- Separation between workflow definition and execution
- Workflow has one or more primary items
- Workflow itself has a state and manages the state of items
- Routing based on role or team
  - Allow delegation for a defined time period
- Tied to LDAP
- Logically separate from but intended to work with History
- Start simple but basis for future more sophisticated extension
- Allow (but not require) use with private workspace for non-approved changes
- Time based or periodic events or initialization
- Allow for automated (not manually invoked) transitions based on a condition or timing
- Allow use of workflow-level, system-maintained, contextual variables e.g. $primary, $initiator, $assignee
- Allow use of expressions e.g. for routing, guards etc
  - SHACL (which can incorporate SPARQL) is the prime candidate
- Can interwork, or be triggered by, external flows e.g. Jira, GitHub, CI/CD
- Can use email and other notification
- Supports a browser-based input queue per user
- Browser interface does not require refresh to get updates/notifications (e.g. uses web sockets)
- Manager interface to spot and address blockages
- Metrics
- Incorporate user stories as the actions
- Allow for voting/quorum
- Allow for knowledge worker usage with some degree of flexibility rather than rigid automated processes
- Use provenance to track all inputs
- Including other system interactions
- Allow for multiple Activities within a Workflow
  - Change
  - Approve

  - Read

0.2 Design Approach

0.2.1 Case Management vs Process Management

Case management is complementary to process management but more appropriate to flexible knowledge worker tasks compared to a more traditional and rigid BPNM process flow.

- Allows more flexibility of tasks and sequence – mix of fixed and optional tasks
- Useful for knowledge tasks and optional e.g. compliance related tasks
- More event and condition driven
- Typically associated with one primary artifact
- Tracks all information used ("case file")
- Flexible in who's involved
- Auditing is key
- It fits well with, and provides a framework for, the user story approach
- Can invoke traditional BPNM processes or DMN decisions

There is lots of literature and tooling, including open source. OMG has a standard CMMN[1] though we would not need to use its full power (e.g. multiple layers of nesting) and will start with a more cut-down subset.

Here is sample of the notation:

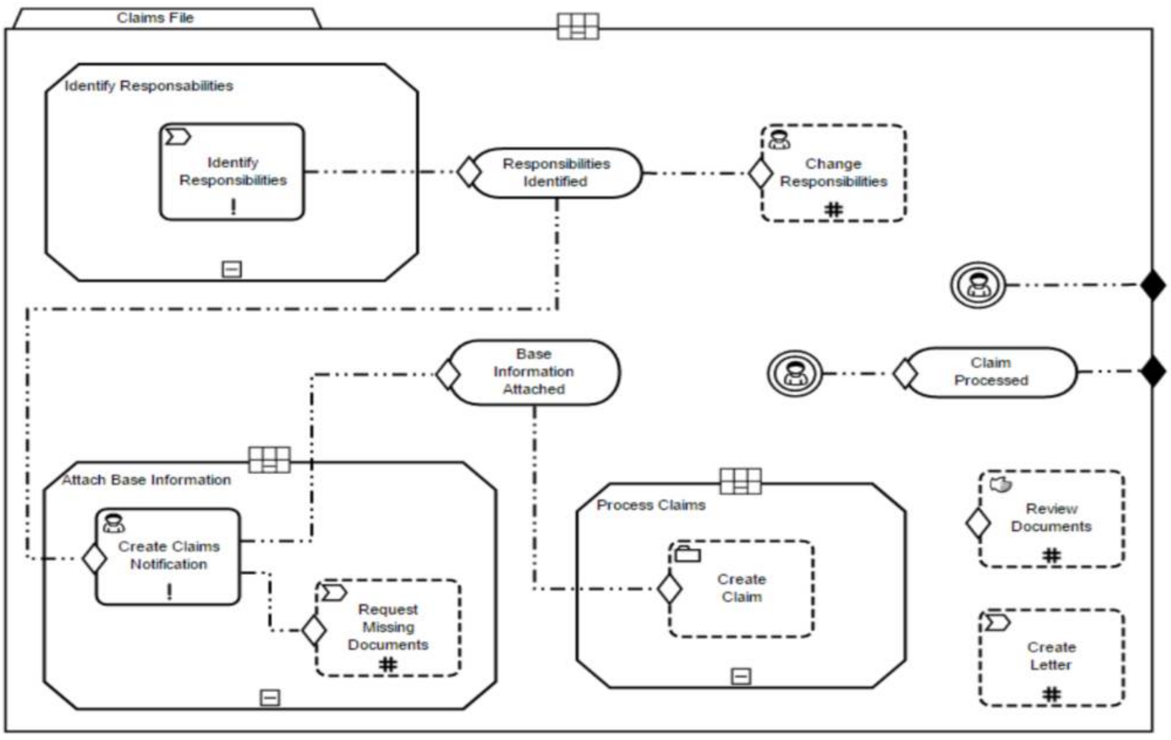

Claims File
Identify Responsabilities
Identify Responsibilities
Responsibilities Identified
Change Responsibilities
Base Information Attached
Claim Processed
Attach Base Information
Create Claims Notification
Request Missing Documents
Process Claims
Create Claim
Review Documents
Create Letter

Figure 6.73 - Claims Management Example

In the above, octagons represent Stages, rounded squares represent Tasks, lozenge shapes represent Milestones and diamonds on the edges of shapes represent entry Criteria – a combination of events (e.g. user actions) or conditions.

[1]See `https://omg.org/spec/CMMN`

0.2.2 State Machines

A state machine is still more rigid than a BPNM process model and controls what transitions can occur between a defined set of states. At any point an item has a current state which determines possible transitions/actions with guards on each. It has similarities with the proposed approach but is more suited to an automation environment than a knowledge worker collaboration environment. As per the example above, the proposed workflow approach can act as a state machine where needed but using Milestones to represent the states.

0.3 Specification

0.3.1 Specification

There are several parts:

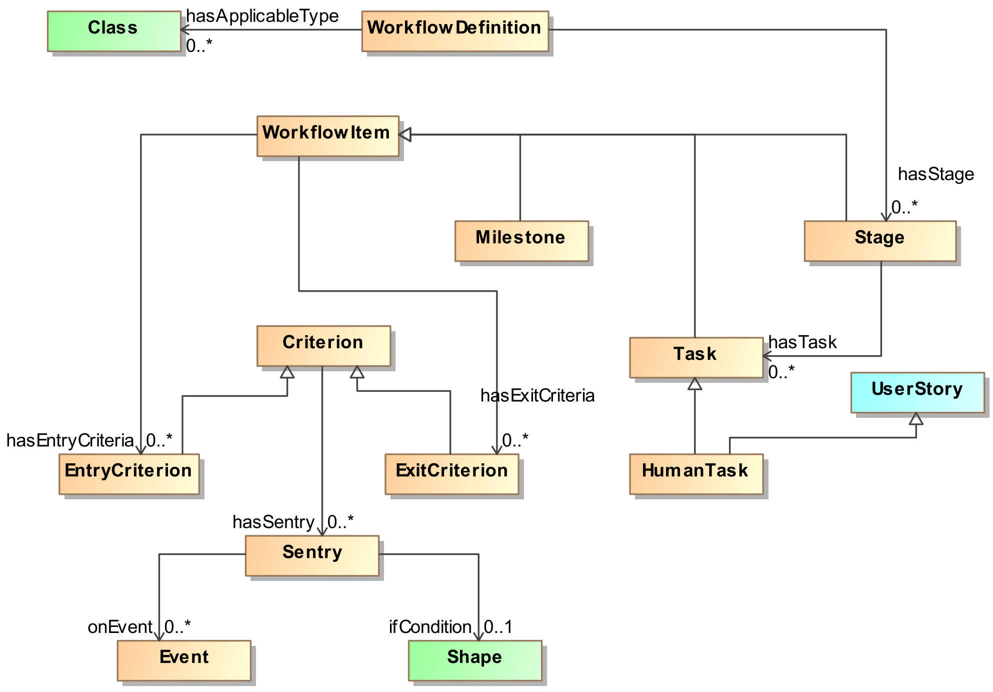
- Workflow definition: defining the process
- Workflow execution: executing the defined process e.g. for approving a particular legal entity change. This will make use of a combination of the Definition together with runtime and instance-specific information
- Auditing: persistently tracking what has happened. This will make use of the existing History service

There will potentially be separate named graphs for each.

0.3.2 Workflow Definition Ontology

This has some simplifications compared to the CMMN metamodel:

- No nesting of tasks/stages
- Fewer events, actions and timers
- No explicit Plan Table to record optional choices



0.3.3 Summary

A *WorkflowDefinition* governs one or more *Classes* of item e.g. LegalEntity. The workflow has a number of *Stages* each of which has a number of *Tasks*. The primary type is a *HumanTask* (other Tasks would be automatic driven by Criteria) which represents a user interacting with software as determined by a UserStory (as already modeled). The *UserStory* determines the entitlement required such as the Persona. A third type of *WorkflowItem* is the *Milestone* which does not involve activity but records an achievement. Each WorkflowItem can have *EntryCriteria* which must be met before that item can be started/is available for the user. And *ExitCriteria* which must be met before the item is deemed complete. The specific condition is represented through a Sentry (or guard) which is passed based on any number of *Events* having occurred and an optional condition having been met. There are a number of built in types of Event which include achievement of Milestones and user actions (e.g. pressing a Submit or Cancel button). The condition is specified by using a SHACL *Shape* which can be used for simple conditions or more sophisticated ones using the power of SPARQL.

Within a Stage a user is able to select (from menu) any HumanTask (UserStory) associated with that Stage and which has its entryCriteria satisfied. Other tasks will automatically be executed when their entryCriteria are satisfied. Many Tasks within Stages related to Approval, might represent stage-specific useful views, queries or reports to run to help the approver reach a decision: a key one being to view the changes (new vs old values) to the legal entity. Hence they might not have entry/exit criteria.

0.3.4 Flow

Specific HumanTasks would represent Approving or Rejecting the proposed change. Approval would be reflected by a user Event which would be one of the entryCriteria for a Milestone. Achieving the Milestone would be an Event which would be combined with other conditions to form the exitCriteria for the Stage as a whole. The flow between Stages is achieved since the same Milestone would be an entryCriteria for the next Stage. Clearly the model allows for more sophisticated flows including branching and looping. For example different approval stages could occur in parallel, and the Reject Event could be in the EntryCriteria re-entering an earlier Edit stage.

```
@base            <https://ekgf.org/ontology/workflow/> .
@prefix :        <https://ekgf.org/ontology/workflow-definition/> .
@prefix hist:    <https://ekgf.org/ontology/history/> .
@prefix story:   <https://ekgf.org/ontology/user-story/> .
@prefix persona: <https://ekgf.org/ontology/persona/> .
@prefix sh:      <https://w3.org/ns/shacl#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix prov:    <http://www.w3.org/ns/prov#> .
@prefix skos:    <http://www.w3.org/2004/02/skos/core#> .

<https://ekgf.org/ontology/workflow/>
  a owl:Ontology ;
  rdfs:label "EKGF Workflow Ontology"@en ;
  rdfs:comment "The EKGF Workflow Ontology"@en ;
  owl:imports persona: , story: ;
.

:WorkflowItem a owl:Class .

:WorkflowDefinition a owl:Class .

:hasApplicableType a owl:ObjectProperty ;
  rdfs:domain :WorkflowDefinition ;
  rdfs:range owl:Class ; # or use concept:Concept here?
.

:hasStage a owl:ObjectProperty ;
  rdfs:domain :WorkflowDefinition ;
  rdfs:range :Stage ;
.

:Stage a owl:Class ;
  rdfs:subClassOf :WorkflowItem ;
.
```

```
:hasTask a owl:ObjectProperty ;
  rdfs:domain :Stage ;
  rdfs:range :Task ;  # provides the persona via story:hasPersona on HumanTask
.

:hasOrganization a owl:ObjectProperty ;
  rdfs:domain :WorkflowStage ;
  rdfs:range :Organization ;  # provides the notification and entitlement
.

:Task a owl:Class ; # can be automatically executed subject to entry criteria
  rdfs:subClassOf :WorkflowItem ;
.

:HumanTask a owl:Class ;
  rdfs:subClassOf :Task, story:UserStory ;
.

:Milestone a owl:Class ;
  rdfs:subClassOf :WorkflowItem ;
.

:hasEntryCriteria a owl:ObjectProperty ;
  rdfs:domain :WorkflowItem ;
  rdfs:range :EntryCriterion ;
.

:hasExitCriteria a owl:ObjectProperty ;
  rdfs:domain :WorkflowItem ;
  rdfs:range :ExitCriterion ;
.

:Criterion a owl:Class .

:EntryCriterion a owl:Class ;
  rdfs:subClassOf :Criterion ;
.

:ExitCriterion a owl:Clas s;
  rdfs:subClassOf :Criterion ;
.

:hasSentry a owl:ObjectProperty ;
  rdfs:domain :Criterion ;
  rdfs:range :Sentry ;
.

:Sentry a owl:Class .

:ifCondition a owl:ObjectProperty ; # only 1
  rdfs:domain :Sentry ;
  rdfs:range sh:Shape ;
.

:onEvent a owl:ObjectProperty ; # may have many, all must occur
  rdfs:domain :Sentry ;
  rdfs:range :Event ;
.

:Event a owl:Class . # Initially a small number of built in events

:MilestoneReached a :Event .

:UserSubmission a :Event .
```

0.3.5 Workflow Instance Ontology

WorkflowInstance represents an instance of a WorkflowDefinition as applied to one or more primary item which are instances of the applicableType for the definition. The running workflow has a number of variables that represent its running state and reflect decisions and intermediate results.

These include:

- Primary items (typically only 1 – such as a Legal Entity)
- Role assignments (to people or groups)
- Active stages (typically only 1; zero if the workflow is complete)
- Milestones achieved
- New events (until processed)
- Timers
- Watched events and timers
- (History service) Activities

The Workflow instances are tracked in the EKG (typically in the *Audit Graph*) and serve as a persistent record. It makes use of the History service for tracking and auditing changes to managed items.

The WorkflowInstance provides the focal point for all workflow-related information. At any point there will be at most one associated with a Session, and there will be a API to access this, from which all the other data can be accessed. The UI will generally display the current session workflow instance name e.g. in title bar or footer.

Generally all state is managed at the WorkflowInstance level: there is no equivalent at the Stage level.

Additionally, the managed item is extended with an additional property to allow tracking of its current state. That is equivalent to the Milestones achieved by associated WorkflowInstances. In addition to the common properties defined here, specific implementations may define workflow-specific properties and variables. These may be accessed by the workflow's Conditions.

```
@base           <https://ekgf.org/ontology/workflow-instance/> .
@prefix:        <https://ekgf.org/ontology/workflow-instance/> .
@prefix wf:     <https://ekgf.org/ontology/workflow-definition/> .
@prefix hist:   <https://ekgf.org/ontology/history/> .
@prefix story:  <https://ekgf.org/ontology/user-story/> .
@prefix persona: <https://ekgf.org/ontology/persona/> .
@prefix sh:     <https://w3.org/ns/shacl#> .
@prefix owl:    <http://www.w3.org/2002/07/owl#> .
@prefix prov:   <http://www.w3.org/ns/prov#> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
@prefix skos:   <http://www.w3.org/2004/02/skos/core#> .
@prefix xsd:    <http://www.w3.org/2001/XMLSchema#> .

<https://ekgf.org/ontology/workflow/>
  a owl:Ontology ;
  rdfs:label "EKGF Workflow Ontology"@en ;
  rdfs:comment "The EKGF Workflow Ontology"@en ;
  owl:imports persona: , story: , hist: , wf: ;
.

:currentState a owl:ObjectProperty ;
  rdfs:domain wf:Milestone ;  # may have multiple values
.

:WorkflowInstance a owl:Class.
# reuse rdfs:label and rdfs:comment
```

```
:hasDefinition a owl:ObjectProperty ;
  rdfs:domain :WorkflowInstance ;
  rdfs:range wf:WorkflowDefinition ;
.

:hasPrimaryItem a owl:ObjectProperty ;
  rdfs:domain :WorkflowInstance ;
.

:hasActiveStage a owl:ObjectProperty ;
  rdfs:domain :WorkflowInstance ;
  rdfs:range wf:Stage ;
.

:hasAssignment a owl:ObjectProperty ;
  rdfs:domain :WorkflowInstance ;
  rdfs:range :Assignment ;
.

:Assignment a owl:Class .

:hasPerson a owl:ObjectProperty ;
  rdfs:domain :Assignment ;
  rdfs:range :User ;
.

:hasRole a owl:ObjectProperty ;
  rdfs:domain :Assignment ;
  rdfs:range cfs:InternaluserId ;
.

:hasHistory a hist:Activity a owl:ObjectProperty ;
  rdfs:domain :WorkflowInstance ;
  rdfs:range wf:Stage ;
.

:hasTimer a owl:ObjectProperty ;
  rdfs:domain :WorkflowInstance ;
  rdfs:range :Timer ;
.

:Timer a owl:Class .

:hasStartTime a owl:DatatypetProperty ;
  rdfs:domain :Timer ;
  rdfs:range xsd:timeStamp ;
.

:hasTriggerTime a owl:DatatypetProperty ;
  rdfs:domain :Timer ;
  rdfs:range xsd:timeStamp ;
.

:hasNewOccurrence a owl:ObjectProperty ;
  rdfs:domain :WorkflowInstanc e;
  rdfs:range wf:Event ;
.

:hasCurrentWatchedEvent a owl:ObjectProperty ;
  rdfs:domain :WorkflowInstance ;
  rdfs:range wf:Event ;
.
```

0.4 Example

0.4.1 Example

This example is for a simple sequential workflow of one initial editing Stage followed by 6 review/approval stages.

0.4.2 Workflow Definition in OWL

Editing starts off in Draft: either for a new Legal Entity or a change to a previously approved one The need for workflow approval is only triggered if certain attributes of the entity are changed. Approvers can reject but not edit the entity Many people can contribute at any stage. There are potentially many use cases that might be available for each.

0.4.3 Workflow Definition in OWL

```
@base         <https://ekg.agnos.ai/id/> .
@prefix :     <https://ekgf.lgt.com/ontology/workflow/> .
@prefix wf:   <https://ekgf.org/ontology/workflow-definition/> .
@prefix ldap: <https://ekgf.org/ontology/ldap/> .
@prefix cs:   <http://purl.org/vocab/changeset/schema#> .
@prefix hist: <https://ekgf.org/ontology/history/> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix legal-entity:  <https://ekg.agnos.ai/id/legal-entity> .

:LegalEntityWorkflow a wf:WorkflowDefinition ;
  wf:hasApplicableType legal-entity:LegalEntity ;
  wf:hasExitCriteria reached Milestone Approved ;
  wf:hasStage
    :DraftPrep,
    :LegalAndTaxReview,
    :ComplianceReview,
    :InternalAuditReview
    :RiskControReview,
    :AccountingReview,
    :MGCReview
.

DraftPrep a wf:Stage;
  wf:hasOrganization ldap:DataMaintenanceUsers ;
  wf:hasTasks ... ; # list of user stories for editing including createEntity, amendEntity
  wf:hasEntryCriteria ... ; # none
  wf:hasExitCriteria ... ; # user selected Submit; basic constraints met
.

Drafted a wf:Milestone ;
  wf:hasEntryCriteria ... ; # DraftPrep exited; workflow-trigger attribute changed
.

Approved a wf:Milestone ;
  wf:hasEntryCriteria ... ; # DraftPrep exited; workflow-trigger attribute not changed
.

ComplianceEditing a wf:Stage ;
  wf:hasOrganization ldap:DataMaintenanceUsers ;
  wf:hasTasks ... ; # list of user stories for editing compliance related fields
  wf:hasEntryCriteria: on Drafted milestone if $entity.isRegulated (SHACL) ;
  wf:hasExitCriteria: MinimalShapeOK on UserSubmit even t;
.
```

```
LegalAndTaxReview a wf:Stage
  wf:hasEntryCriteria: on Drafted milestone if not($entity.isRegulated) ;
  wf:hasExitCriteria: on InComplianceEditing completed ;
  wf:hasTasks ReviewTaxAspects ; # possibly other user stories for tax info/analytics
  wf:hasExitCriteria: MinimalShapeOK on UserSubmit event ;
.
```

0.4.4 Example Execution Flow

- Data maintenance Person (DMP) creates a new or amends an existing LegalEntity
  - ⇒ create a new workflow instance (the only WF definition which has applicableType of LegalEntity, if there were more then user or system could choose)
  - ⇒ create a new hist:Activity
  - ⇒ status of the LegalEntity is Draft (via hasInitialState), everyone can see the changed state
- DMP makes further edits including resuming the next day
  - ⇒ status is still Draft
  - ⇒ History Activity with ChangeSet is built up
- Another DMP makes a few changes
  - ⇒ status is still Draft
  - ⇒ History Activity with ChangeSet and additional contributor is built up (changes not attributed)
- DMP happy and submits for approval
- No trigger attributes were changed
- User confirms that no approval is desired
  - ⇒ status is Approved
- Activity is marked as complete (end date set)
- No trigger attributes were changed
  - ⇒ status is Group Legal and Tax
- Group Legal and Tax gets notified
- Activity is marked as complete (end date set)

0.5 External References

This section provides references to external material that provides supplemental detail or relevant implementations (e.g. of CMMN).

0.5.1 W3C Provenance Ontology (PROV-O)

This is a complex and flexible ontology from which this design borrows the basic concepts of Agent, Activity, Entity and some key properties. See `https://www.w3.org/TR/prov-o/`.

0.5.2 Case Management Model and Notation (CMMN)

The official specification is available at `https://www.omg.org/spec/CMMN`

The following provide background information.

`https://www.researchgate.net/publication/234038280_Data_Centric_BPM_and_the_Emerging_Case_Management_Standard_A_Short_Survey` https://www.researchgate.net/publication/328342272_A_Comparison_of_Flexible_BPMN_and_CMMN_in_Practice_A_Case_Study_on_Component_Release_Processes

https://middlewareblog.redhat.com/2018/06/19/effective-case-management-within-a-bpm-framework/

0.5.3 Flowable

https://flowable.com/open-source/ Open source engine from the Camunda/Activiti family but with pluggable persistence https://github.com/flowable/flowable-mongodb

Interesting fact – the commercial company is a partner of Avaloq.

0.5.4 Camunda

This is an open source BPNM/CMMN engine.

This shows the Architecture difference between History and runtime https://docs.camunda.org/manual/latest/user-guide/process-engine/history/

There is a useful third-party add-on BPM Taskpool https://www.holunda.io/camunda-bpm-taskpool/ It includes a MongoDB materialized view which could be readily adapted to RDF https://www.holunda.io/camunda-bpm-taskpool/wiki/user-guide/components/view-mongo/ Here is an example https://www.holunda.io/camunda-bpm-taskpool/wiki/user-guide/example/

Acronyms

Glossary

EKG system architecture the logical system architecture of Enterprise Knowledge Graph (EKG) is divided into multiple layers or environments.

EKG/Platform a logical system architecture component, the layer of software services that provide and serve the EKG to end-users and other systems. The platform logically is a set services that enforce any of the specified policies in the self-describing datasets (SDDs) that have been published in the EKG. 12, 13

self-describing dataset An EKG is logically composed of a set of *self-describing datasets* that provide information about lineage, provenance, pedigree, maturity, quality, governance, entitlement policies, retention policies, security labels, IP policies, pricing policies, caching policies, organizational ownership, accountabilities, data quality feedback loop, issue management policies and so forth. Any given system owner that connects their data to the EKG becomes in fact a publisher of a self-describing dataset and can therefore control how their data is handled by the EKG/Platform. The EKG/Platform consists of services that enforce any of the specified policies in the self-describing dataset. At the technical level, datasets can be files or streams or API definitions etc. 13

system architecture the conceptual model that defines the structure, behavior, and more views of a system. 13

Index