2. Case description

This MLM challenge involves representing universal properties of process types along w… types, actor types and their various … (Section 2.2), and an application of … in the scope of a particular software engineering process (Section 2.3). Submitted solutions should include bottom-level instances, at least for key types, exemplifying all attributes mentioned in the challenge description. Deviations from the case as described here should be documented in submissions. The case description may be extended but respective rationales should then be provided.

2.1. Overview

Process management is chara… of rules concerning the executio… cesses, tasks, actions or activities … ulating and keeping track of processes, i.e. the enactments of process types, with such pr… referred to as "process instance… literature. For example, in the s… it may be necessary to keep t… …ting, e.g., the fact whether or not code has been *tested*, etc. Further rules impo… the participation of business actors (hu… and artifacts (equipment, documents, to… and specifies depender… engineering domain: (i… *sign*; (ii) *test case desi*… employs a *requiremen*… *cases*; and (iii) *testing* is performed by a *tester*, employs *test cases*, and produces a *test report*.

In other contexts, such as the insurance domain, there may be a need to keep track of which *policy holder submitted an insurance claim*, when it was submitted, which *claims analyst authorized payment* of the insurance premium in response to the claim, how much was claimed, which claims are still pending assessment, how much was paid out for a particular claim, etc.

Submissions to the challenge should focus on the software engineering domain. They may optionally include the insurance domain as well. In the following, we are using the insurance domain for illustrative purposes only.

2.2. Processes, tasks, actors and artifacts

The following general rules pertaining to processes, tasks, actors and artifacts apply for the challenge:

P1) A *process type* (such as *claim handling*) is defined by the composition of one or more *task types* (*receive claim*, *assess claim*, *pay premium*) and their relations.

P2) Ordering constraints between *task types* of a *process type* are established through *gateways*, which may be *sequencing*, *and-split*, *or-split*, *and-join* and *or-join*.

cess type has one *initial task type* (with which all …ecutions begin), and one or more *final task types* (with which all its executions end).

P4) Each *task type* is *created* by an *actor*, who will not necessarily perform it. For example, *Ben Boss created* the task type *assess claim*.

P5) For each *task type*, one may stipulate a set of *actor types* whose instances are the only ones that may *perform* instances of that *task type*. For example, in the *XSure* insurance company, only a *claim handling manager* or a *financial officer* may *authorize payments*.

P6) A *task type* may alternatively be assigned to a particular set of *actors* who are authorized (e.g., *John Smith* and *Paul Alter* may be the only *actors* who are allowed to *assess claims*).

P7) For each *task type* (such as *authorize payment*) one may stipulate the *artifact types* which are *used* and *produced*. For example, *assess claim* uses a *claim* and produces a *claim payment decision*.

P8) *Task types* have an *expected duration* (which is not necessarily respected in particular occurrences).

P9) *Critical task types* are those whose instances are *critical tasks*; each of the latter must be *performed* by a *senior actor* and the artifacts they produce must be associated with a *validation task*.

P10) Each *process type* may be enacted multiple times.

P11) Each *process* comprises one or more *tasks*.

P12) Each *task* has a *begin date* and an *end date*. (e.g., *Assessing Claim 123* has *begin date 01-Jan-19* and *end date 02-Jan-19*).

P13) *Tasks* are associated with *artifacts used* and *produced*, along with *performing actors*.

P14) Every *artifact used* or *produced* in a *task* must instantiate one of the *artifact types* stipulated for the *task type*.

P15) An *actor* may have more than one *actor type* (e.g., *Senior Manager* and *Project Leader*.)

P16) Likewise, an *artifact* may have more than one *artifact type*.

P17) An *actor* who *performs* a *task* must be authorized for that task. Typically, a class of actors is automatically authorized for certain classes of tasks.

P18) *Actor types* may *specialize* other *actor types* in which case all the rules that apply to instances of the specialized *actor type* must apply to instances of the specializing *actor type*. For example, if a *manager* is allowed to *perform tasks* of a certain *task type*, so is a *senior manager*.

P19) All modeling elements, at all levels, must have a *last updated* value of type *time stamp*. This feature should be defined as few times as possible, ideally only once. Respective definitions are exempt from the requirement to have a *last updated* value. Note that this requirement differs from the respective version in [3].

Note that it is not necessary for every *type* in the model to have an instance. It is useful, however, to illustrate the design with a number of instances.

when arrived… or may go further to another Task…

How other things are created, linked ?

Hence there is a perform relationship

Does that mean there is an actual duration ? i.e a start time ?

all instances should be critical or some of them? I choose the

Model de Fahad ?

P10: so many interpretations!

P11-P1x reduce interpretation scope…

P11:task instance!

P9:how? use/ produce /other ?

Not arte fact type

P9:an insta nce or a type ?