20230727 - Thematische werkgroep 3 OSLO Voorwaarden Dienstverlening

>> GT

Intussen deed ik het volgende voor OSLO Voorwaarden Dienstverlening:

- Datavoorbeelden maken (bijgevoegd).
- Datamodel: nazicht en verbeteringen.

Datavoorbeelden:

- In bijlage

Opmerkingen datavoorbeelden:

Hier komen een paar interessante kwesties uit naar voor, te bespreken op de volgende werkgroep, nl:

- Subklassen van Voorwaarde? Wordt aanbevolen in COPSV-AP.
- Strikte koppeling tussen Voorwaarde en Bewijstype in model, zal data kunnen volgen?
- Publieke Dienstverlening (consumptie): niet nodig?
- Hoe aangeven dat agent niet voldoet aan voorwaarde? Maw dat toetsing heeft plaatsgevonden en dat dit het resultaat was.
- Niet expliciet aangeven dat men (als gevolg vh feit dat men aan alle Voorwaarden voldoet) recht heeft op de PubliekeDienstverlening?
- Hoe omgaan met het of-of verhaal van Bewijstypes? Bvb id of paspoort of copie of vreemdelingenkaart of geboortecertificaat, waarbij dit ook nog eens afhangt van kenmerken vd aanvrager.

Datamodel:

- In github

Opmerkingen datamodel:

- Ik checkte tov https://semiceu.github.io/CCCEV/releases/2.00/ en https://semiceu.github.io/CPSV-AP/releases/3.1.1/.
- Voorwaarde.beperking: Betekenis onduidelijk volgens definitie “De effectieve beperking in tekstvorm”. Lijkt technisch veld? Als dit de Voorwaarde is in tekstvorm: daarvoor dient Voorwaarde.beschrijving. Kardinaliteit op 0 gezet.
- Waarom ontbreekt Requirement.isDerivedFrom(ReferenceFramework)? Omdat voorwaarden hier bekeken worden vanuit het perspectief van PubliekeDienstverlening dat binnen een legaal kader werkt? Om dezelfde reden mag ook Voorwaarde.uitgegevenDoor worden geschrapt (zie hieronder).
- TODO: Definities nazien. Bvb definitie van Voorwaarde.isVoorwaardeVan moet zijn “Verwijzing naar de Voorwaarde waarvan dit een ondergeschikte Voorwaarde is.” ipv “Voorwaarden kunnen gegroepeerd worden”.
- TODO: Ook usageNotes ontbreken(in voorgaand geval bvb “Verwijzing naar Voorwaarde ‘Minimumleeftijd 18’ die als ondergeschikte Voorwaarden heeft ‘Leeftijd moet worden verstrekt.’ en ‘De opgegeven leeftijd is correct.’).
- TODO: Ontbrekende tags.
- Waarom ontbreken volgende attributen/rollen bij PubliekeDienstverlening: contactPoint, hasChannel, hasCost, isClassifiedBy, isDescribedAt, isGroupedBy, keyword, produces, relatedService, requires, sector, thematicArea. Opvallend veel rolnamen.

Verbeteringen datamodel:

- Navigatiepijlen verwijderd
- Ontbrekende rolnamen toegevoegd, oa Voorwaarde.heeftBewijstypelijst.
- Camelcase fouten verbeterd, oa Bewijstypelijst ipv BewijsTypeLijst.

- Kardinaliteiten zonder rolnaam verwijderd, oa Bewijstypelijst->Voorwaarde.
- Attributen/rolnamen correct vertaald, oa Voorwaarde.isVoorwaardeVan ipv Voorwaarde.hangtSamenMet.
- Kardinaliteiten minder streng gemaakt: oa Voorwaarde.identificator is nu 0..* ipv 1. Kardinaliteit 0 omdat biet alle Voorwaarden een externe id hebben vh kaliber isbn-nummer of rijksregisternummer en kardinaliteit * omdat meerdere id's mogelijk moeten zijn).
- Kardinaliteit Informatieconcept.uitdrukkingVerwachteWaarde op 0..* gezet: 0 omdat soms niet meer dan een Beschrijving voorhanden is en * opdat eenzelfde concept in meerdere constrainttalen kan uitgedrukt worden?
- Dataset als superklasse van Bewijs.
- LangString ipv TaalString.
- Tijsperiode ipv Tijdsinterval.
- Kardinaliteit van Tijdsperiode.eindtijd op 0 gezet (op bvb identiteitsbewijs staat geen starttijd).
- Bewijs.verstrektDoor en .gemaaktDoor losser gemaakt, deze info is niet steeds beschikbaar of valt samen met deze van gaatOver en uitgegevenDoor.
- Overal Code gebruiken als enumeratie OF specifiee lijsten, niet allebei dooreen. Daarom datatype van BewijsType.bewijstypeClassificatie veranderd naar Code.
- Voorwaarde.uitgegevenDoor: Geschrapt als inverse van Agent.voldoetAan, is daar nl geen inverse van semantisch gezien.
- Voorwaarde.uitgegevenDoor: ook globaal geschrapt aangezien de Voorwaarden hier niet los gezien worden van een PubliekeDienstverlening (uitgever is dan immers de PubliekeOrganisatie die verantwoordelijk is voor de dienst).

>> JL

1. ISSUE:

Hierdoor is het model pas bruikbaar als Voorwaarden aan Bewijzen kunnen gekoppeld worden. Merk op dat in CPSV-AP de directe koppeling PublicService-EvidenceType wel mogelijk is. Echter als we hiermee zouden volstaan kan niet meer automatisch bepaald worden of voor consumptie ve dienst aan alle Voorwaarden is voldaan, we kunnen enkel nog vaststellen of de vereiste bewijsstukken voorhanden zijn.
→ Waarom kan dit niet meer automatisch bepaald worden? Omdat er dan 2 dataconnecties zijn? Kan dit niet bij implementatie worden opgelegd? De link tussen Agent en Bewijs wordt ook bevestigd in de attributen van Bewijs (gaatOver...)

2. ISSUE:

Probleem is dat 1 lijst sowieso niet volstaat wegens verschillende opties: bvb identiteitskaart is goed maar ook paspoort etc… waarbij dit ook nog eens varieert volgens het profiel vd aanvrager (Belg, Niet-belg, Europeaan etc). Volstaat verwijzing naar Voorwaarden om identiteitskaart, paspoort etc te krijgen?
→ De gebruikers vullen op de rechtenverkenner eerst een aantal zaken in (gekoppeld aan SSO van Mijnburgerprofiel), op basis van deze informatie kan er een shortlist komen van de 'publieke dienstverlening' waar ze mogelijks recht op hebben met die voorwaarden aangeduid?

3. ISSUE:

Indien al een dossier gestart is ihkv een PubliekeDienstverlening kan momenteel geen onderscheid gemaakt worden tussen PubliekeDienstverlening als aanbod (gemeente x biedt dienst "Huwelijk" aan) en

PubliekeDienstverlening als consumptie (gemeente x biedt dienst "Huwelijk" aan aan persoon y). → Indien er een dossier wordt gemaakt is er al gecheckt geweest of de Agent voldoet aan de voorwaarden. Het aanbod zal hier allicht voorrang krijgen aangezien de consumptie in het implementatiemodel zit.

4. ISSUE:

Is het ook nodig om te kunnen meedelen dat een Agent niet voldoet aan een Voorwaarde? Impliciet kan een Agent niet genieten de dienstverlening als hij niet aan alle Voorwaarden voldoet, maar dat aan een Voorwaarde niet is voldaan kan betekenen dat 1) de Agent inderdaad niet voldoet, maar ook 2) dat dit nog niet is gecheckt. → Is dit belangrijk voor het datamodel? Kan er een status of een melding worden meegegeven, eventueel bij de implementatie?

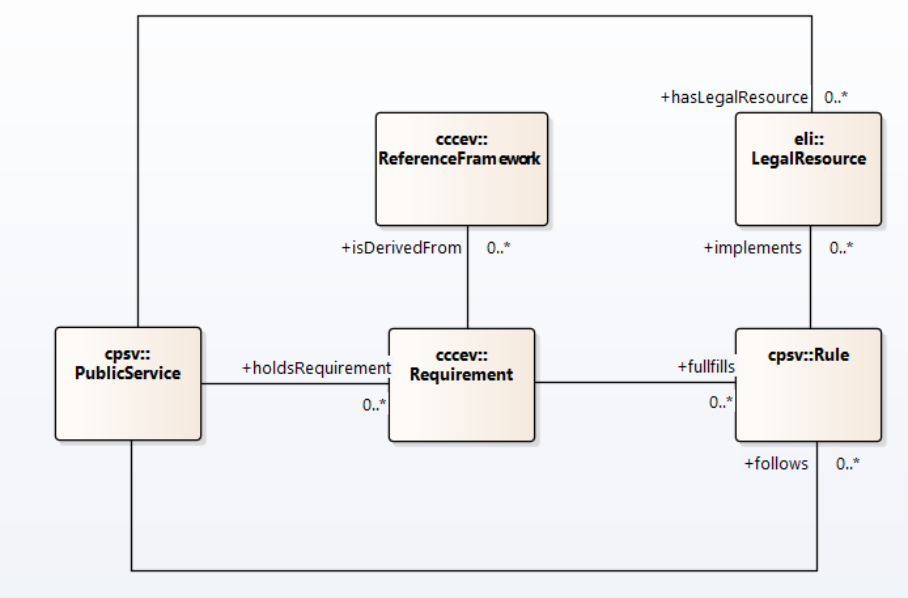
5. Wat bedoel je met "checken in JSONLD-playground na toevoeging contextfile."

20230907 - Thematische werkgroep 4 OSLO Voorwaarden Dienstverlening

>> GT
Er moet dus wel nog wat gebeuren tegen volgende WG Voorwaarden Dienstverlening van 7/9. Namelijk:

1. Oplossing voor de OF relatie tussen Voorwaarden (Voorwaardelijst zoals met Bewijstypelijst?). Welk was het voorbeeld dat werd genoemd en is daar documentatie over? Voorwaarde +18 OF +16 met akkoord rechter?
2. Voorbeeld met subklassen van Voorwaarde correct uitwerken (was nu niet het geval). Plus een meer ingewikkeld voorbeeld? Welk geval werd genoemd?
3. Voorbeeld met verwijzing naar de wetgeving. Uit te zoeken. Feit is dat FormalFramework in CCCEV vervangen is door ReferenceFramework en n CPSV-AP door LegalResource. Kortom: vager in CCCEV en strikter in CPSV-AP.
4. Informatieconcept leeftijdsvereiste strikter uitwerken met geboortedatum tov huwelijksdatum etc. Misschien kan dit ook met SHACL hier.
5. Punt dat niet aan alle Voorwaarden moet voldaan zijn om recht te hebben op de PubliekeDienstverlening? Maar dat klopt eigenlijk niet, want dan is het geen Voorwaarde. Is eigenlijk = punt 1.
6. Alternatief voorbeeld uitwerken dat wat complexer is dan leeftijdsvereiste, bvb https://www.vlaanderen.be/belastingvermindering-voor-de-kosten-voor-kinderopvang.

Over punt 3: Huidige samenhang is deze:



Waarbij:

- ReferenceFramework vervangt het vroegere FormalFramework. Lijkt wat ruimer maar de definitie is niet veranderd. Is vooral omdat ook Requirements moeten geformuleerd kunnen worden die niet in een of andere wetegeving staan, bvb de voorwaarden voor een geldige offerte ofzo.
- LegalResource was vroeger ook FormalFramework. Is dan weer een verenging, wellicht omdat het hier over publieke dienstverlening gaat en men letterlijk niets mag doen in die cointext dat niet ergens in de wet (of daarvan afgeleide reglementen of procedures) is vastgelegd.
- Aangezien het hier gaat over voorwaarden ihkv publieke dienstverlening mag ReferenceFramework weg volgens mij. Wordt sowieso ook gedekt door de fullfills en daarna implements associaties resp tussen Requirement en Rule en tussen Rule en LegalResource.
- Ter info: LegalResource is een referentie naar een wet. Niet te verwarren met de wettekst, dat is een LegalExpression (bvb in één bepaalde taal). Gelieve in het huidig model dus LegaleVerschijningsvorm te vervangen door Rechtsgrond. Zo is het ook gebruikt in OSLO-Besluit, hoewel OSLO-Vlaamse Codex dan weer spreekt van resp Rechtsbron en Verschijningsvorm

---

>> YD
Wat betreft punt 1:
Voor huwelijk specifiek:
http://www.ejustice.just.fgov.be/cgi_loi/loi_a1.pl?DETAIL=1804032130/N&caller=list&row_id=1&numero=11&rech=22&cn=1804032130&table_name=WET&nm=1804032150&la=N&chercher=t&dt=BURGERLI

JK%20WETBOEK&language=nl&choix1=EN&choix2=EN&fromtab=wet_all&nl=n&sql=dt%20contains%20%20%27BURGERLIJK%27%2526%20%27WETBOEK%27and%20actif%20%3D%20%27Y%27&tri=dd%20AS%20RANK%20&trier=afkondiging&imgcn.x=25&imgcn.y=6#LNK0064

Voor niet-belgen geldt ook het wetboek van internationaal privaatrecht:
http://www.ejustice.just.fgov.be/cgi_loi/loi_a1.pl?language=nl&la=N&cn=2004071631&table_name=wet&caller=list&N&fromtab=wet&tri=dd%20AS%20RANK&rech=1&numero=1&sql=%28text%20contains%20%28%27%27%29%29#LNK0061

Wat betreft punt 2:
Ik vermoed dat we huwelijk kunnen nemen, maar dan alle voorwaarden meenemen uit het voorbeeld (5)

Wat betreft punt 3:
Zie ook punt 1, dit is de wetgeving die ze ons doorgestuurd hebben.

Wat betreft punt 4:
Is inderdaad een belangrijk punt om aan te tonen aan de hand van het concept 'Meerderjarigheid' en 'Uitzondering van de rechtbank'. Bijvoorbeeld het voorbeeld leeftijdsvereiste op datum van de trouw zelf.

Wat betreft punt 5:
Klopt, dat gevoel heb ik ook. Ik denk dat we dit punt sowieso moeten samennemen met punt 1. Er zal nooit een voorwaarde zijn waaraan je niet moet voldoen, tenzij je aan een 'voorwaarde x of y' moet voldoen.

Wat betreft punt 6:
Lijkt me inderdaad een goed voorbeeld om uit te werken.

20230810 – Different modelling approaches for public service procedures

Problem definition: broadly speaking, a public service procedure involves a requirement processor and an evidence provider. To enable the processing of an evidence by an application, the procedure requirements and the evidences which support those requirements must be specified.

In terms of application profiles, it is possible that requirement processor and evidence provider do not belong to the same trusted domain. This would be the case if evidences are exchanged across jurisdictions. This application profile is being worked on under the OOTS initiative (technical documentation).

A more simple use case - and application profile - is one where the requirement processor and evidence provider belong to the same trusted domain.

Regardless of the scenario, three technical problems must be addressed to enable the automatic processing of evidences:

- How to convert public service requirements – which are always derived from legal resources – into machine-processable rules? (I)

- How to convert evidences into machine-readable data points? These data points are matched against the rules during processing. (II)
- How to compute which requirements are applicable to a given user? (III)

Addressing these problems comes down to specifying a data model *and* a processing algorithm to handle the incoming evidences. Only so much can be captured in a static data model. An automated public service procedure also requires the definition of the processing algorithm. A choice must be made between a document-based and a fine-grained data-based approach. A data-based approach is more complex but enables a.o. taking into account the state of the agent at the beginning of the process, reasoning and inference capabilities, … A document-based approach is less complex and revolves around the definition of constraints. The latter approach does not leverage the full power of the linked data stack, but is potentially less complex. It is a matter of investigation whether intermediary solutions can be designed.

1. CPSV-AP & CCCEV

CPSV-AP is service provider centric. A cpsv:PublicService is linked to a number of cccev:Requirement. Such cccev:Requirement fulfills a cpsv:Rule which itself implements a eli:LegalResource.

In order to enable the processing of evidences, CCCEV uses the concept of cccev:InformationConcept. It is used as the holder of the machine-processable rule (InformationConcept.expressionOfExpectedValue). The evidence provider provides data points through cccev:SupportedValue.

CCCEV does not specify how public service requirements are to be translated into machine-processable rules (Problem I). More problematic is that neither CPSV-AP nor CCCEV support:

- The notion of a workflow or a procedure – understood as a sequence of steps - which a user is to follow to consume the public service. The (very) limited support for this notion is that multiple cccev:Requirement can be chained together (AND operator). At implementation level, this means that a cpsv:PublicService holds a list of cccev:Requirement. This problem has already been raised in the context of the SDGR implementation, but has not been addressed in the latest releases of the models.
- The direct association between a cpsv:PublicService and cccev:Requirement makes it impossible to model variants (OR operator) of a public service (e.g.: based on categories, …).
- CPSV-AP & CCCEV are oblivious to the state an evidence provider is in (Problem III). This makes it impossible to decide which steps in a workflow could be skipped (NOT operator) for the current user.

CCCEV uses a data-modelling trick to specify which (combination of) cccev:Evidence should be presented to support a given cccev:InformationConcept (Problem II). How to translate an evidence into machine-readable data points is not specified.

2. Single Digital Gateway & OOTS

The OOTS project has its own requirements, namely the exchange and processing of evidences across trust domain boundaries. This is not necessarily a requirement that Voorwaarden Dienstverlening implementers will face. It is still interesting to look at the approach as it re-uses some of the concepts of

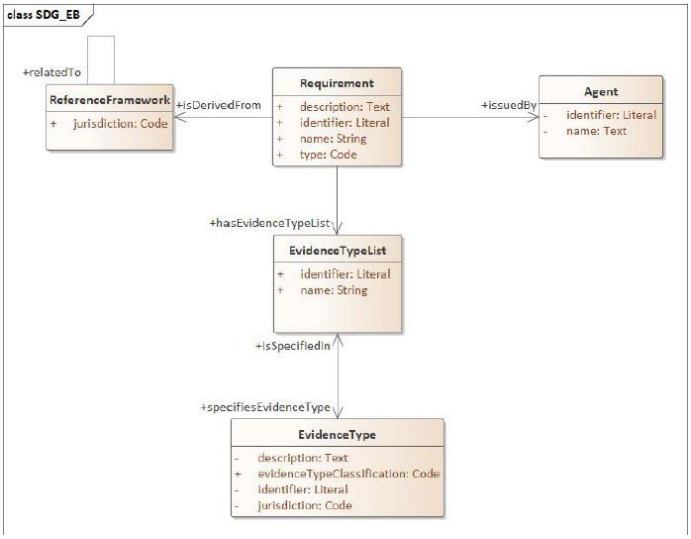
CCCEV. While a part of the architecture re-uses CCCEV, a distinct modelling effort has been made for OOTS (SDG generic metadata model).

"*The evidence exchange consists of a flow of messages based on the Evidence Exchange Data Model (EDM), which comprises the EDM Request (Request from Evidence Requester to Evidence Provider for data or documents) and the EDM Response (Response from Evidence Provider to Evidence Requester to deliver the requested data or documents). The Query Model is fully documented ans supports document-based queries. The EDM […] facilitates the exchange of unstructured and structured types of evidence*" (source).

"*The Exchange Data Models make use of the functional capabilities of the OASIS RegRep V4 Query Protocol, and therefore the request and response messages are being modeled in the form of query models. The query models are illustrated together with a mapping to the appropriate syntax elements of the OASIS RegRep V4 Query Protocol and the syntax elements of the SDG metadata profile, a generic metadata model for the OOTS that is used to enhance the capability of the OASIS RegRep V4 Query Protocol (Syntax Mapping). In this way, for each Exchange Data Model, a specific SDGR Application Profile is defined that consists of syntax mappings to all necessary information entities for Evidence Requests, Evidence Responses and Error Responses. Class diagrams, tables and XML examples of the Evidence Exchange illustrate the shape of the Exchange Data Models.*" (source).

It is not our intention to delve into the details of OOTS. We only assess how Problems I, II and III are addressed by the solution.

Problem I:



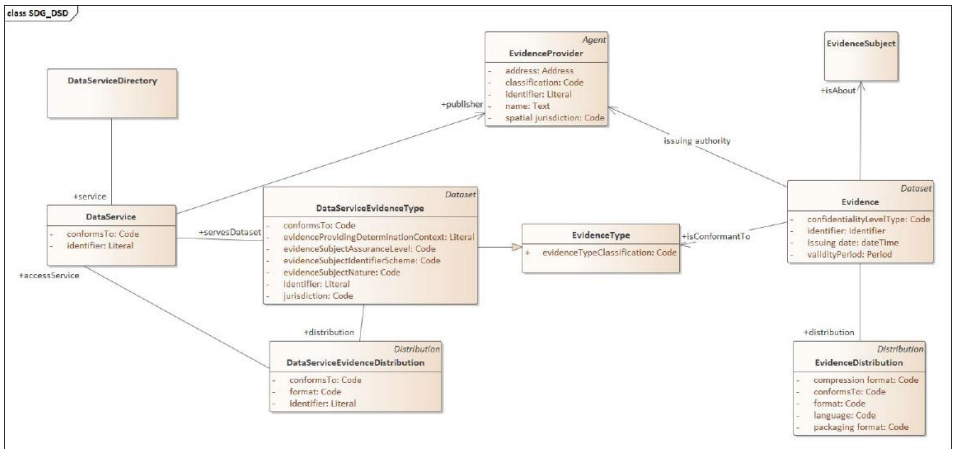
In summary, the cccev:Requiremenent list of the 21 public service procedures in scope of the SDG are specified. The Evidence Broker maps each cccev:Requirement with an cccev:EvidenceType that can prove

that the user fulfills that requirement. The system is not really designed to scale to a large numbers of procedures.

The current focus is on the development of the more basic forms of public procedures in scope of OOTS. For example, the requesting of a proof of registration of birth. This type of public service procedure has probably been chosen given its universality (i.e. no need to distinguish between categories of citizen, …).

An example of the request-response exchange can be found here. It appears that under the current model a variation of requirements will have to be implemented as a distinct public service procedure.

Problem II:



The system is designed around the electronic exchange of documents. The system seems to also support so-called transformations (e.g.: communicate whether someone is above 18 based on certificate of birth data). The response side of OOTS has also been the object of a modelling effort, called the DSD information data model. At this point, it is not clear whether this modelling effort aims at leveraging the modelling work done at the level of SEMIC. In the samples available online, it seems that ad hoc classes are being developed to represent a natural person, a legal person,…

Problem III: OOTS seems to be focus on universal procedures which do not have variations based on user's characteristics.

In conclusion, the OOTS architecture appears to be fairly ad hoc and does not leverage the linked data work made at SEMIC or enabled by the linked data stack.

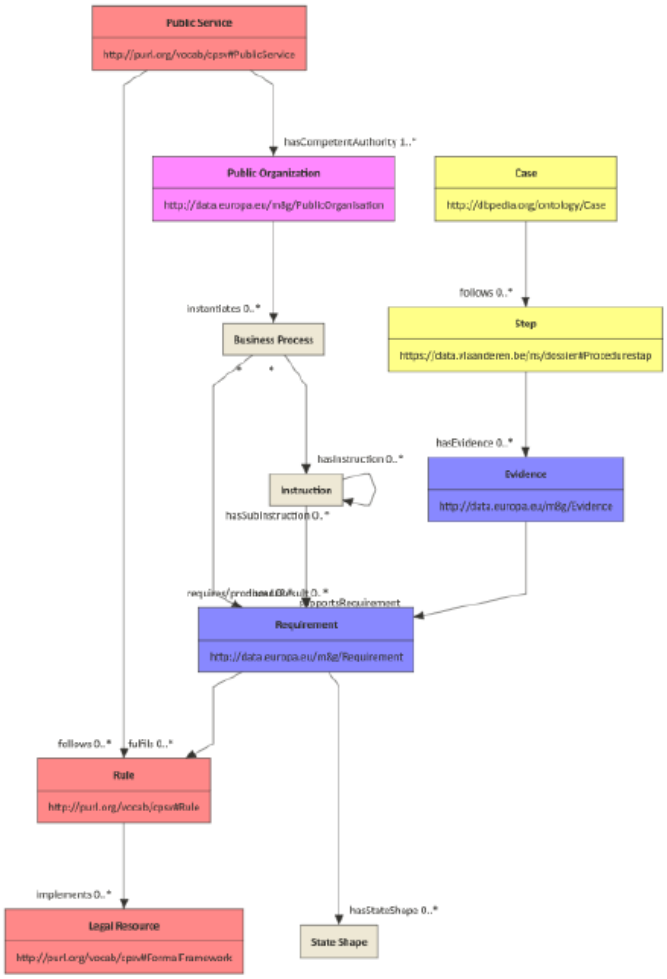
3. OSLO-STEPS

OLSO STEPS is an ambitious project aimed at the modelisation of dynamic workflows for the delivery of public services.

This project is aimed at the modelling of business procedures, understood as dynamic workflows.

It is not our aim to discuss the OSLO STEPS project in detail. We only assess how Problems I and II are addressed by the solution.

Problem I:



(source)

The model is still very much in development. The gist is that a public service is to be decomposed in machine-processable instructions. These instructions form together a business process. A requirement is

the smallest unit of work possible which is scheduled based on pre and post conditions. An instruction takes the form of a https://fast.ilabt.imec.be/ns/oslo-steps#Step.

OSLO STEPS allow public service procedures to be represented as business processes. It is possible to have variants of procedures, which are all encompassed by the same public service.

A state shape is a shape that specifies a set of triples representing a state, and is described as (SHACL) constraints.

Problem II:

The machine-readability of evidences is made possible through the use of RDF. The evidence is presented as a small data graph which is checked against the sh: NodeShape of the requirement.

Problem III:

The system takes into account the current state of the user.

An example can be found here.

4. An intermediary solution

A data example of an alternative solution is available here. This solution leverages the function ontology (prefix fno:). The example: prefix is used when a new term would need to be created.

Problem I:

A public service procedure is presented in formal steps. When a particular step can be turned in a machine-processable process, a set of fno:Functions are defined which implement an fno:Algorithm. The formal fno:Parameters and return values (fno:Output) of these functions are specified. A particular function definition can be linked to a function implementation (fno:Implementation). Ideally, a function should be designed as a reusable, composable block. In order to achieve maximum reusability, implementations should be made available on a function hub. Instead of implementations in particular programming languages, we believe that implementations should be written in SPARQL or a similar query language (GRAPHQL-LD). Often, this preempts additional mapping efforts.

When the procedure is executed (fno:Execution), values are passed as arguments to the functions. As the application profile is about the testing of evidences to requirements, the initial input of the execution must be a value derived from an acceptable eu:Evidence. Thanks to the prov ontology, further transformations of the data are always traceable back to the evidence.

As the user processes through the procedure, the knowledge graph is enriched by more and more data points which are backed by authoritative sources. It is possible that the knowledge graph contains unrelated nodes (no nesting / embedding), which can be used as distinct arguments to functions.

Problem II:

In order to present eu:Evidences, the dcat vocabulary is reused. In the future, more and more authoritative sources will be exposed under the form of base registries. In order to anticipate on this evolution, the authors of CCCEV made eu:Evidence a subclass of dcat:Dataset. Functions, which often take the form of queries, are not fired against the eu:Evidence but a dcat:Distribution in RDF format (e.g.

JSON-LD). The mapping of the dataset to a given distribution needs to be further investigated (how to go from an eidas high authentication based on itsme to an OSLO-Persoon? / based on which identifier can the base registry be queried ?).

An algorithm will need to be divised to parse requirements and submit appropriate evidences. This algorithm must be further investigated as it will have an impact on the data model. We give a first outline hereunder:

1. The algorithm starts with a request to retrieve possible (first) step(s) which are applicable to complete a public service procedure. Assuming that the P-PLAN ontology is used to define the sequence of high-level steps, a query should be fired to retrieve the matching steps based on the progress of the user
2. Of all the steps, the state of the (knowledge graph of the user) should be against the state shape of the step (step:hasStateShape).
3. The dcat:Distribution(s) are drawn from the acceptable eu:Evidence(s). If multiple, the data is gathered in a knowledge graph. The goal of the process is to enrich that graph with data backed by authoritative sources. As long as data point (triple) can be traced back to an eu:Evidence, it can be added to the graph.
4. The processing uses the data from the knowledge graph to pass it as arguments to the data transformation /manipulation functions.
5. One or multiple ouptut values are added to the knowledge graph of the user.
6. Repeat as of step 1.

Outstanding questions:

- Should the P-PLAN and FnO ontologies be combined?
- In ordre to achieve reusability, the FnO ontology causes the creation of many IRIs (on parameters, output values, …). While this is warranted, it should be investigated whether some of the IRIs could be replaced by identifiers without human semantics (e.g. URNs).
- In the current data example, an evidence is linked with high-level p-plan:Activity / dossier:Procedurestap. What if a query / transformation function wants to pivot to another authoritative source? In the current model, this is only possible if multiple evidences have been associated with a requirement and that the data derived from these evidences has been combined in a knowledge graph. Otherwise, a separate activity must be initiated.
- It should be possible to model the high-level sequence of steps in a visualization tool. The output of that tool should be a queriable RDF representation. Is there existing libs / tools which can be used for this?
- In the current data example, the knowledge graph is directly modified (used as input parameter and output value). This introduces the question around mutable vs immutable objects in programming languages. Is it preferable to not modify the knowledge graph directly?

During the last meeting with the core team, the decision was taken to explore OSLO STEPS as the potential way forward.

The original intent of OSLO STEPS is to use the included data model in conjunction with a so-called workflow composer. The workflow composer's function is to compute the steps to be executed. Our understanding at this point is that OSLO STEPS as a data model can however be used without the workflow composer. To be more precise, the algorithm used to compute the sequence of steps to be executed is outside the model (out of scope).

An important contribution of OSLO STEPS is to distinguish between the formal specification of the public service procedure and its execution. The formal specification re-uses the OSLO STEPS (P-PLAN), CCCEV, DCAT and CPSV-AP vocabularies. OSLO STEPS draws heavily upon P-PLAN which is itself an extension of PROV. The execution leverages the OSLO Dossier and P-PLAN (PROV) vocabs. Regarding OSLO Dossier, we note that this vocabulary and application profile already point to PROV. The effort in OSLO Voorwaarden Dienstverlening is to reconcile OSLO STEPS with CCCEV 2.00, thereby enabling the modelling of complex public service procedures, the requirement-evidence exchange and the traceability of execution to formally defined procedure and evidences.

The business logic which happens in each scheduled step is not part of OSLO STEPS but could be modelled with the FnO vocabulary. As of now, we consider this to be out of scope.

The public service procedure starts with a OSLO Persoon (procedures starting with a OSLO Organisatie are considered out of scope for the moment). The OSLO Persoon is a foaf:Agent. Procedures involving multiple foaf:Agents as consumers of the procedure are considered in scope. However, the model only addresses the submission of cccev:Evidences whicg are about the foaf:Agent who initiaties the procedure. Providing and verifying cccev:Evidences on behalf of another foaf:Agent is considered out of scope for the moment and replaced by self-declarations.

1. Initiation

The system represented in the data model only knows about data contained and derived from cccev:Evidences. The system is architected around two phases. In a first phase (composition), the relevant steps of the procedure are selected and scheduled for execution. Evaluations made in this phase are based on the data points derived from the evidences. In a second phase (execution), the business logic associated with the scheduled steps is executed on the data derived from the evidences.

Initially a data graph is composed from the available evidences. As scheduled steps are being executed, the data graph is enriched by the data points outputted from each step.

2. Composition

A cpsv:PublicService holds a reference to a step:Plan (marriage:Plan). This plan is composed of step:Step and p-plan:MultiStep. A p-plan:MultiStep is the representation of a plan that appears as a step of another plan. As soon as a p-plan:MultiStep is encountered it must be expanded to navigate through its step:Steps. A step:Step holds a reference to a step:State which is defined as "*a particular condition something is in. In our case a state could be represented as a set of RDF statements*" (source). A step:State holds a reference to a sh:NodeShape which represents the expected data for that step. SHACL

is a good candidate to express these constrains but we make the choice to keep the model technology independent by working with a reference.

The algorithm walking through the decision tree evaluates at each step whether the step must be scheduled for execution. This happens by evaluating the data graph against the step:StateShape of the step (a step:Step is associated with 1-* step:StateShape). If the evaluation succeeds, the step is scheduled for execution. If not, it is discarded.

From the point of view of CCCEV, a step:Step holds a reference to a cccev:Requirement. Such a requirement has supporting evidences. Regarding evidences, CCCEV specifies that "*the proof can be given through documents or extracts of base registries, independently from its structure, format or medium used to exchange it: a pdf document, a video, a recording, etc.*" (source). An example of an evidence is the record of a citizen in the national register (rijksregister) of Belgium. This data can be distributed in different formats and be conformant to different standards. In OSLO Voorwaarden Dienstverlening, we associate a cccev:Evidence with a dcat:Distribution in RDF format. The exact way this is expressed must be further investigated[1]. All the RDF triples gathered from the evidences constitute the data graph which is evaluated in the process.

3. Execution

The actual execution takes the form of a dossier:Zaak. A step:Step scheduled for execution takes the form of a p-plan:Activity. The p-plan:Activity uses p-plan:Entity which is a fragment of the data graph of the foaf:Agent. Each p-plan:Entity corresponds to a step:StateShape and each p-plan:Activity corresponds to a step:Step.

20230906 – Design decisions

Reconciliation with CCCEV:

- We chose not to rely heavily on multiple inheritance as this would require the creation of new encapsulation classes for this application profile. We chose to use "has-a" relationships to connect the different components. However, it is worth reflecting whether a step:Step should not be made a subclass of cccev:Requirement. This would create a tight relationship between OSLO STEPS and CCCEV.
- We do not work with cccev:InformationConcept and cccev:SupportedValue. cccev:InformationConcept overlaps with step:State and cccev:SupportedValue overlaps with p-plan:Entity.

OSLO STEPS:

- We chose not to work with ComponentLevelStep, ContainerLevelStep and JourneyLevelStep. We rely on a step:Step and p-plan:MultiStep. A p-plan:MultiStep should not be scheduled for execution. If applicable, it should only be expanded to walk through its associated Plan.
- For the moment, we did not work with step:producesState. While this is a hard requirement when working with the workflow composer, it can be left out when working with a less

[1] E.g.: dc:conformsTo pointing to OSLO standard. An alternative would be to also express this as a SHACL shape.

sophisticated tree walking algorithm (e.g. depth-first search, bread-first search). This also reduces the workload in terms of shape constraints which need to be manually drawn up.

- A step:Step can be made part of step:Plan by using the following associations: (a) p-plan:isStepOfPlan and (b) p-plan:isPrecededBy. The former does not assume any order between the steps at the same level. The latter specifies an order, i.e. a dependency between steps. While being a matter of implementation, a step whose parent has been discarded should not be visited when walking through the tree.
- The step:Step is the atomic component which can be reused across plans. When publishing a step:Step for reuse, it should not hold any reference to a step:Plan or (previous) step:Step. An exception is made for step:MultiStep which should always be published with its associated step:Plan.
- The system works by populating the (personal) data graph of the foaf:Agent with data points derived from the cccev:Evidences and from the business logic executed at each step. At present, this is implicit in the model but this should probably be made explicit.

OSLO Dossier:

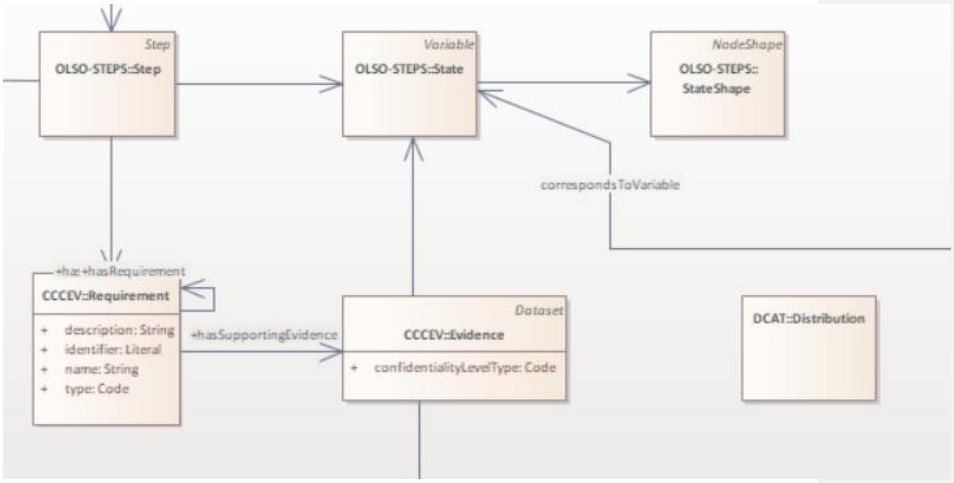
- The interplay between the PROV and DBpedia ontologies is not very clear. I would like to tie Voorwaarden Dienstverlening to OSLO Besluitvorming but the way dossier:Zaak, dossier:ProcedureStap en prov:Activity are supposed to interact is not very clear.

20230920 – Answers to proposition Ben

- I see the benefit of working with both step:requiresState and step:producesState, which gives further hooks to tree walking algorithms. The trade-off is the need to generate even more SHACL shapes to define these states.
- Working with a three-tiered hierarchy (ComponentLevelStep, ContainerLevelStep and JourneyLevelStep) seems like an additional burden. An analyst tasked with formalising a procedure will probably only want to wrap certain Steps into a MultiStep. The definitions of ComponentLevelStep, ContainerLevelStep and JourneyLevelStep are also quite vague.
- In the proposed model, it is suggested to the create a new Class which will inherit from step:State and eu:Requirement (e.g.: ageRequirement). My idea so far was to either:
    - Use has-a relationships

Commented [DB1]: you don't have to, because the 'requiresState' will refer to the same SHACL shape as the 'producesState'

Commented [DB2]: To clarify: "ageRequirement" is an instance of the class eu:Requirement, not a subclass



  - It would be an option to create a new class for the Voorwaarden Dienstverlening AP which inherits both from step:Step and eu:Requirement. However not all steps (which are low-level atomic tasks) are requirements (e.g.: data transformation, manipulation, …). That's the reason why I chose the current has-a relationship.
  - Another new class could be created for the AP which inherits from step:State and eu:InformationConcept. The main advantage here is to be able to link an eu:Evidence to eu:InformationConcept (via eu:supportsConcept) which is a valuable relationship.
- In the proposed model, mention is made of example:BusinessRule (c.f. isAgeAbove18). This seems to point to this node holding a decision table. I am a bit confused by this as I thought that the OSLO STEPS approach relies more on constraint programming as an alternative to rules engine.
- The proposed model imports cpsv:Rule. I would suggest to refrain from linking to that entity. In the current architecture of IDPC for example, cpsv:Rules are not formalised in an atomic way and are essentially text blocks associated with a public service. I would suggest to keep a single link (ex:instantiates) to the unstructured text data.

Aside from these observations, I refer to my other points above.

In the drafting of data examples, I am faced with the difficulty of writing SHACL constraints manually. I hoped to find an automated way to derive constraints from data graphs, but have so far not found a suitable tool.