Data policy as activity network

© Vasily Bunakov

Science and Technology Facilities Council, Harwell Campus, United Kingdom

vasily.bunakov@stfc.ac.uk

Abstract. The work suggests using a network of semantically clear interconnected activities for a formal yet flexible definition of policies in data archives and data infrastructures. The work is inspired by needs of EUDAT Collaborative Data Infrastructure and the case of long-term digital preservation but the suggested policy modelling technique is universal and can be considered for all sorts of data management that require clearly defined policies linked to machine-executable policy implementations.

Keywords: data management, long-term digital preservation, data policy, semantic modelling.

1 Introduction

Problematics of advanced long-term digital preservation [1] has been in focus of many collaborative projects and popular recommendations. However, it has been paid a relatively small attention in domain-specific projects that rely on data archiving, or in projects that develop scalable e-infrastructures aggregating data that comes from different user communities.

One of the problems that long-term digital preservation aims to address is having clear policies for the entire data lifecycle from data ingestion by archive or by e-infrastructure service, through years-long data management with sensible data checks, transformations and moves, to data access and data dissemination to the end users.

One can argue that without clear data policies and means of their validation there is no such a thing as the long-term digital preservation, even in cases when a technology foundation used for an archive or an e-infrastructure is sound and well-supported. At the end of the day, every technology evolves – and at a brisk pace compared to relatively long time when many data assets are going to be useful, so data policies and means of their expression should be semantically clear and in a way more permanent than technology that underpins data management. A strong case for policy-driven digital preservation, with extensive references to the prominent projects and popular methodologies was made in [2].

In practice, quite a few data archives and e-infrastructures end up in a situation when they have got a sound technology for managing data bits, also acquire a decent number of users (which is a popular measure used by funders for their judgement on the e-infrastructure success) but do not have a reasonable data policy, let alone any machine-assisted reasoning over it. The users' trust in the archive or the e-infrastructure may be enough for their daily use but there can be a substantial conceptual and technological gap in regards to data policies formulation, expression and execution.

Some larger projects and e-infrastructures are aware of this gap and do make efforts to close it by working on data policies implementation. An example of such e-infrastructure is EUDAT [3] that has developed a number of operational services [4] and data pilots with user communities, and is now trying to express and apply policies to these services.

The prime candidate for applying data policies in EUDAT is B2SAFE service [5] based on iRODS platform [6]. B2SAFE developers are doing a very good job on building geographically and organizationally distributed data storage with data replication, integrity checks and other routine tasks of data management guided by iRODS machine-executable rules. B2SAFE have made their own effort on policies with the development of Data Policy Manager [7] which is a software module with policies expressed via XML templates. There is a perceived need though of having a more universal solution for policy management across all EUDAT services. The possible policy modelling approaches under consideration are using RuleML[8], SWRL[9] or ProvOne ontology[10] which seems suitable not only for capturing data provenance after the execution of certain actions but also for the forward-looking design of data processing workflows which can then potentially serve as a means of data policy modelling.

This work presents an alternative approach to those mentioned and is based on Research Activity Model [11] which is in fact quite universal and suitable for the expression of all sorts of activities, not necessarily related to research. Research Activity Model is slightly extended and applied to the case of data policy modelling.

The main advantage of this alternative approach is its high modularity which allows modeling policy elements and using them as building blocks for the semantically clear representation of a whole policy. The modularity of policy design is especially important in data infrastructures that commonly aggregate data coming from different user communities, often having their own business models, technical requirements, data formats and data lifecycles which makes it difficult to design and adequately express the crosswalks between community-specific data policies and those for the data infrastructure. Another advantage of the suggested approach is its ability to address the conceptual gap between policy formulation and policy implementation, as it may not be easy to

Proceedings of the XIX International Conference "Data Analytics and Management in Data Intensive Domains" (DAMDID/RCDL'2017), Moscow, Russia, October 10–13, 2017

translate a high-level policy (often in a textual form) into machine-executable policy.

The modularity should allow high levels of inheritance and reuse of policy elements; it also helps to solve specific problems of policy formulation and validation when textually the same policy can be executed in different ways leading to different states of data archive, for which situation we provide an example. The conceptual gap between policy formulation and policy implementation is addressed by a possibility to define policy-related Activities as "black boxes" with (initially) only interfaces defined; this can be hopefully done by policy makers themselves without entirely delegating this policy design phase to policy implementers (software developers).

Implementation of a sensible data policy is a challenging task even within the boundaries of a particular organization. In a situation when the organization is using a collaborative data infrastructure along with its own organization-specific IT services, the implementation of a data policy is going to be even more intricate and is likely to rely on loosely coupled services. An approach to data policy modelling suggested in this work is going to address this challenge, along with the alleviation of the earlier mentioned problems of the policy elements reusability and the policy application results predictability.

The work is inspired by needs of EUDAT Collaborative Data Infrastructure [3] and refers to it for illustration of certain ideas, also the main incentive for the work was modelling policies for the case of long-term digital preservation. However, the suggested modelling technique is universal and can be considered for all archives or e-infrastructures that are interested in all sorts of data management (not only long-term digital preservation) that require a clearly defined policy linked to machine-executable policy implementations.

Conceptual challenges of data policy modelling are discussed first, specifically the problem of policy decomposition into policy elements, then an example is given of how Activity Model can be used for policy modelling. This is followed by suggestions on what IT architecture for data policy management will be required to support the suggested modelling techniques.

2 Data policy and a problem of its decomposition

2.1 Insufficiency of granular policy definition

Data policy is often created as a conventional textual document that contains certain statements about what should or should not be done with data, with implied or sometimes explicit logical "ANDs" and "ORs" that glue statements together in an aggregated policy. This composite nature of policies is why it seems natural to break down the policy document into granular statements, model each statement using some formalism and then execute the statements using some IT solution.

One of the most advanced efforts on data policy decomposition was performed by SCAPE project [12] that created an extensive catalogue of preservation policy elements [13] which are in fact granular textual statements. These granular statements which can be converted, in a pretty straightforward way, in machine-executable statements are called *control policies* in SCAPE. Examples of control policies are: "information on preservation events should use the PREMIS metadata schema" or "original object creation date must be captured". The granular control policies relate to a higher-level *procedural policy* (a procedural policy on Provenance for the current example) which in turn relates to an even higher-level and most abstract *guidance policy* (a policy on Authenticity for the current example). Three-level structure of guidance policies, procedural policies and control policies constitute a very well developed SCAPE digital preservation policy framework.

SCAPE stopped short of the actual implementation of control policies, so when EUDAT [3] decided to use the SCAPE framework for policy considerations, it was also decided to supplement this framework with the catalogue of practical data policies [14] developed by an RDA (Research Data Alliance) Practical Policy Working Group. The practical data policies in this catalogue are expressed as iRODS [6] functons specifically suitable for implementation in EUDAT B2SAFE service [5] based on iRODS platform.

Having well-defined control policies or practical policies is not enough though for semantically clear modelling of a data policy as a whole, as the application (execution) of a policy composed of granular machine-executable statements may lead to quite different outcomes depending on the order in which granular policies are applied.

The problem of policy decomposition is in fact interrelated with the problem of policy validation. To illustrate this, let us consider a simple case when there is a couple of easily identifiable policy statements contained in the same policy document which we want to decompose and validate through execution of two granular policies. Let the statements in a composite policy (perhaps, but not necessarily so, added one to another through some policy update by different policy managers) be:

[1] Image files having size of more than X gigabytes should be stored in file storage A; otherwise they should be stored in file storage B.

[2] Image files of type RAW should be converted in JPG format.

If a certain file of type RAW is more than X gigabytes in size but becomes less than X when converted in JPG then, depending on the higher-level guiding policy and on the order in which these granular policies are applied in the actual service implementation, the result of the combined application of the two granular policies can be any of the following:

1. File is moved as RAW in storage A and remains stored in A as RAW.
2. File is moved as RAW in storage A then converted in JPG and remains stored in A.
3. File is converted in JPG and stored in B.
4. File is moved as RAW in storage A and remains stored in A as RAW; also a copy of it converted in JPG is stored in B.

This is to illustrate that validation of the data policy implementation is hard as any of the listed outcomes may be considered being right or wrong depending on the validator's point of view.

Also let us take into account that policy validation can be based on some statistical selection of samples (so that problematic boundary cases of RAW data sized only slightly over X gigabytes threshold may not be selected in a sample and hence go unnoticed), or that a policy validation procedure allows some tolerance towards small amount of failed policy checks (so that even if a few files have ended up somewhere that a particular policy interpretation considers to be a wrong place, this does not trigger a policy violation alert).

So even if the data policy can be, seemingly successfully, decomposed into granular policies that are easy to define and validate as machine-executable statements, the actual result of the policy implementation does not necessarily match the intentions of policy designers or policy managers, as the backwards process of the policy composition – assembling it from the granular policies (policy elements) – can be performed with substantial variations.

2.2 Possible responses to the challenge of granular policies insufficiency

One possible response to the outlined challenge could be setting up an elaborated policy governance framework, i.e. well-defined business processes that allow human agents (policy managers) to look after the policy implementation, i.e. accumulate and analyse feedback from the environment where the policy is applied and supply the result of this analysis as updated requirements to software developers who work on the actual software implementation of the policy. This approach requires a good organizational culture and a substantial human resource involved in data policy management and in policy implementation; documented requirements will serve as an interface between policy managers and policy implementers. Some "magic" should happen in between so that high-level policy definitions translate into actual policies implementation in software code, this is why policy validation is likely to demand extensive software testing with specific policy-related test cases.

Another possible response is having an elaborated means of expression for the entire data policy (a sophisticated policy modelling language): both for the definition of granular policies and for the definition of logic than binds the granular policies into the whole. An example of this approach is RuleML [8] that is considered a candidate for a detailed expression of data policy in EUDAT e-infrastructure [3]. This approach requires skilled human resource for policy modelling; the modeler and a sophisticated model produced by her becomes then an interface between policy managers and policy implementers (the role of the latter is less prominent than in the first approach, in a sense that software developers should not interpret requirements but just implement – or adopt – a certain engine that executes formal rules defined by the savvy policy modeller).

The third possible response is that a certain formalism is used for the expression and, where necessary, recomposition of granular policies (policy elements) and for their assembling in the whole, with that formalism being reasonably friendly to machines as well as to humans. The humans – policy managers themselves or a not-so-skilled modeller – can use the formalism for a flexible policy definition that can be fairly easily modified depending on the true policy intentions and on the feedback received from the archive or e-infrastructure where the policy is implemented. The role of software developers is then to implement an engine for the formalism (quite similarly to the second approach). The machine just executes the policy expressed using that formalism.

The differences amongst approaches are presented in Table 1; in essence, they are different "weights" (different levels of demand) for the skills of policy managers, policy modellers and policy implementers.

Table 1 Differences amongst policy modelling approaches

Policy modelling approach	**Demands for policy manager skills**	**Demands for policy modeller skills**	**Demands for policy implemen-ter skills**
Policy governance framework + requirement s managemen t + specific software testing	High	None (policy modeler can be replaced by business analyst or/and software tester)	High
Policy modelling language	Low	High	Medium
Formalism for granular policy elements definition and composition	Medium	Medium	Medium

The preferable approach could easily be the third one as it empowers policy modelers themselves with reasonable means of policy expression and therefore can reduce overheads and risks of communicating a policy from policy managers through modelers to implementers. A remote analogy of the third approach could be the proliferation of SQL language that, despite its sophistication, has become a lingua franca of not only software engineers but is widely used by logistics and even sales departments is all sorts of business.

The formalism to be used for data policy expression should not be something as developed as SQL though, neither should it be purely textual: it can be based on the idea of “building blocks” with possible graphical representation of them, hence providing an easy-to-operate semantic wrapper for machine-executable statements. On the other hand (unlike SQL which allows the actual data manipulation), these “building blocks” for data policy definition are likely to remain only a wrapper to the actual machine-executable implementations of granular policies which will be inevitably specific to a particular service even within the same archive or e-infrastructure. As an example, for EUDAT B2SAFE [5] that is based on iRODS platform [6] these granular implementations can be iRODS functions and for other EUDAT services based on other software platforms the policy implementations can be something else. A common semantic wrapper will be then a reasonable means of a clear policy modelling and a clear definition of interfaces between policy “building blocks” across a variety of different IT services.

This work strongly prefers the third approach and suggests considering Activity Model [11] for semantically clear modelling of data policies in all IT services within the same data archive or e-infrastructure, as well as for policy interoperability across different data archives and e-infrastructures.

3 Activity Model as a semantic wrapper for machine-executable policies

3.1 Activity Model in a nutshell

Activity Model [11] was initially suggested for modelling granular research activities and combining them in networks so that, as an example, the output of one Activity can be the input of another one, e.g. these combined Activities may represent certain phases in research data analysis. It has been clear though that Activity Model can suit all sorts of activities as it is pretty generic; as an example, it may well suit for modelling data provenance across different IT services within e-infrastructure.

The main “building block“ of the Activity Model is an “activity cell” represented by Figure 1 with its aspects (that can be thought of as incoming and outcoming relations) explained in Table 2.

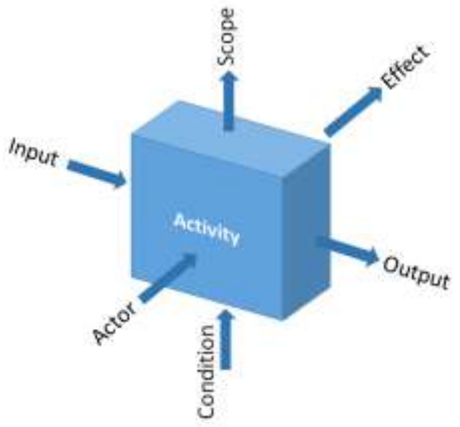

Figure 1 Research activity “cell”; it can be used for semantic definition of any activity

The full RDF serialization of the Activity Model is published in [11]; it is really simple and requires only RDF Schema and an “inverseOf” OWL statement for its expression, i.e. what is often referred to as RDFS Plus.

Table 2 Activity Model aspects explained

Aspect	**Description**	**Examples**	
		Research per se	Research data analysis
Input	Something that is taken in or operated on by Activity	Previous research	Raw data
Output	Something that is intentionally produced by Activity	Raw data	Derived (analyzed) data
Scope	Something that Activity is aimed at or deals with	Sample properties	One or more experiments
Condition	Something that affects or supports Activity, or gives it a specific context	Scientific instrument	IT environment
Actor	Something or somebody who participates in Activity	Investigator	Data analyst
Effect	Something that is a consequence of Activity	Environment pollution	New software module

Activity “cells” can be combined in chains or networks, and not necessarily in a way that the Output of one Activity is the Input to another. As an example, a data management policy can be the Output of one Activity (policy design) and the Condition that affects another Activity, e.g. data replication in the archive.

The model flexibility when any aspect of one Activity can be matched with any aspect of another Activity is supported by the fact that aspects do not have to have types associated with them.

3.2 Proposed extensions of the Activity Model

In order to use Activity Model for data policy modelling, we will need to make a profile of the model by specifying certain types of Activity as subclasses (in case of an RDF serialization of the model – RDFS subclasses). Suggested extensions are presented in Table 3. Conceptually, Generic Data Management Activities should cover the needs of data engineering that are related to machine-interpretable policy implementations, Logical Switch Activities should cover the needs of data analysis and machine-assisted reasoning, and Control Activities should cover the needs of IT services deployment and operation.

Compared to modelling data policies with workflows, the suggested approach based on the definition of policy-related Activities should allow more loosely coupled implementations of policy management IT solutions. As an example, the "data engineering" part of policy implementation represented by Generic Data Management Activity can be performed on a software platform fully controlled by a specific user community or organization (e.g. a research institution), the operation (the actual execution of control statements) represented by Control Activity can be performed by collaborative data infrastructure (e.g. by EUDAT CDI [3]) and the logic of combining policy elements represented by Logical Switch Activity can be performed by either the organization or the data infrastructure, or by a third-party service.

If the policy was modelled by an executable workflow, it would require the presence of all three aspects: data engineering, reasoning and execution – in the same workflow likely operated by a single universal workflow engine. This would mean not only an operational limitation but a conceptual / modelling limitation, too, as all the participants (stakeholders) of policy implementation would have to adhere to the conceptual framework and the format required by the workflow engine. Modeling with interconnected Activities as semantic wrappers to particular implementations leaves more freedom to conceptualize and to operate data policies that are going to be executed by loosely coupled IT services.

Table 3 Additions to the core Activity Model required for data policy modelling

Type to add	Comment / Description
Generic Data Management Activity	Subclass of Activity for data policy definition. It can be considered a semantic wrapper for a variety of data handling Activities, e.g. Activities for data characterization or data transformation.
Logical Switch Activity	Subclass of Activity for logical switches of all sorts
Control Activity	Subclass of Activity for an interface with a particular software platform where policies are executed. This is a semantic wrapper for the actual call to a platform-specific script or function.

Depending on a particular operational environment (software platform where policies are executed), other parts of the Activity Model, e.g. its Inputs, Outputs, or Conditions may require additional semantically clear extensions. However, it is unclear at the moment whether these potentially required extensions should be a part of the universal Activity Model profile for data policies, or it is better to introduce them as necessary, as parts of policy execution engine implementations on particular software platforms.

3.3 Examples of the Activity Model data policies profile application

The role of the suggested model extensions will be clearer by giving an example of their application to the modelling of a particular policy. The example will be a policy with two granular statements about data movements depending on data size and data format that were considered in Section 2.1.

We will need to define first a File Characterization Activity:

```
   @prefix am:
   <http://.../stuff/ActivityModel#> .
   @prefix ampp:
   <http://.../ActivityModel#PolicyProfile> .
   GDMA_FileChar                                  a
ampp:GenericDataPolicyActivity
GDMA_FileChar am:hasInput File
GDMA_FileChar am:hasOutput FileSize
   GDMA_FileChar am:hasOutput FileFormat
   GDMA_FileChar am:hasOutput File
   GDMA_FileChar am:hasScope ImageFiles
   GDMA_FileChar                 am:hasCondition
ServiceInstance
   GDMA_FileChar am:hasActor CertainScript
   GDMA_FileChar am:hasEffect FileCharLog
```

In short, GDPA_FileChar activity takes a file as an input and produces values for the file size and file format (which can be semantically clearly defined as necessary – e.g. with measurement units and format IDs in a file type registry) as outputs; the initial file is passed over as another output. To derive the file size and format, the activity uses CertainScript (which again can be semantically clearly defined as necessary – e.g. with references to a software repository).

As an additional outcome (better defined not as Output but as Effect) of the file characterization activity, we get theFileCharLog log file. The scope of activity is defined as ImageFiles (so that other kinds of files can be handled by differently defined Characterization Activities; what "ImageFiles" actually means can be clearly defined with e.g. a reference to a certain taxonomy entry). The Condition is defined as ServiceInstance (which means that Actor:CertainScript operates in some particular IT service environment).

Mapping of Activity to a particular software implementation can be performed using Activity ID and a reference to a repository with a clear software identity, e.g. a software versioning repository.

The graphic representation of this Characterization Activity (which, in the ideal world, can be designed in a certain authoring tool with graphical user interface and producing the above RDF as a serialization) is illustrated by Figure 2.

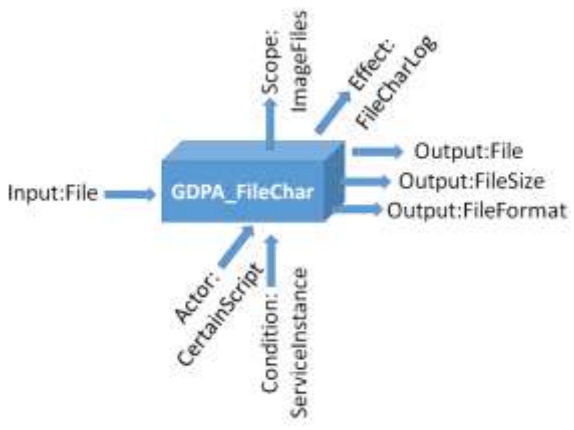

Figure 2 Definition of a Data Policy Activity for image files characterization

The problem of the policy composition out of two granular policies outlined in Section 2.1 can be addressed with the help of other classes of activities that we introduced earlier: Logical Switch and Control. For the sake of simplicity (as we are going just to illustrate it how the policy modelling can be done) we will not be defining all aspects for these activities, e.g. we can omit Scope or Effect but they may be required in a real policy modelling situation.

The Logical Switch activity will take File, FileSize and FileFormat as Inputs, a particular logic of handling file moves to either storage A or B, as well as file conversion, will be Condition. The Activity yields a list of particular control statements (like "move File to storage A", "Convert file in JPG format") as Output. The shape of such defined Logical Switch activity is illustrated by Figure 3.

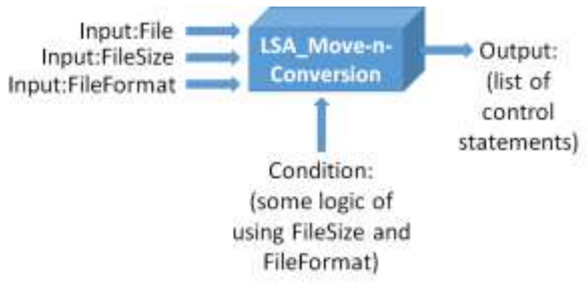
Input:File
Input:FileSize
Input:FileFormat
LSA_Move-n-Conversion
Output: (list of control statements)
Condition: (some logic of using FileSize and FileFormat)

Figure 3 Definition of a Logical Switch Activity for handling image files

The semantically clear definition of a Logical Switch Activity gives an idea of how we suggest to address the problem of a policy composition from granular policy statements. The hope is, if the logic of producing control statements is made explicit, as well as the control statements themselves, this will eliminate the ambiguity of a policy composed of granular policy statements.

A good question is what formalism, if any, will be adequate for the expression of logic in the Condition of the Logical Switch. The short answer is: it depends on the policy engine implementation. In an extreme case, this Condition can be just a mandatory textual explanation (commentary) of the logic implemented by the Actor (which is omitted in the Figure 3), i.e. by an executable function or a procedure or a script for a particular IT platform. Alternatively, rules modelling language or workflow templates (and appropriate engines for them) can be used – yet, in this case, the actual usage of these modelling languages or workflow templates would be limited to the policy logic enwrapped in the Logical Switch Activity, allowing freedom for different implementations of other types of Activities involved in the policy definition.

How to express control statements in the Output is subject to particular implementations, too. The only consideration which is important for the moment – important both from conceptual and from implementation perspectives – is having the list of control statements as a clearly defined interface between Logical Switch Activity and Control Activity.

Control Activity takes the list of control statements as Input and makes platform-specific function or procedure or script calls that implement the control statements. Actors for Control Activity are particular functions / procedures / scripts and the Effects of it are log and error files or messages – whatever is used for traceability in a particular implementation. Condition is, similarly to the file characterization activity definition, a particular software platform or IT service where Actors operate. Figure 4 presents an example of a diagrame for the Control Policy.

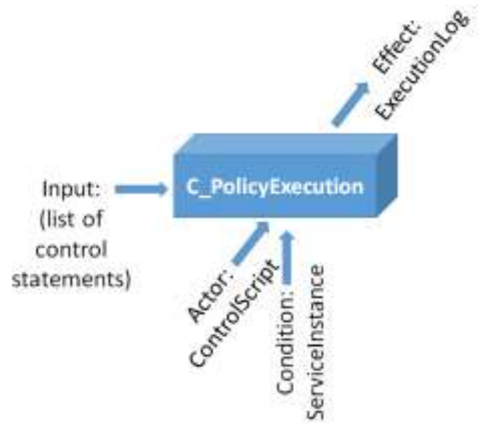
Effect: ExecutionLog
Input: (list of control statements)
C_PolicyExecution
Actor: ControlScript
Condition: ServiceInstance

Figure 4 Definition of a Control Activity for policy execution

Generic Data PolicyActivities (such as data characterization) can be combined with Logical Switch Activities and Control Activities in a chain or a network of activities. For our example, the resulted chain is illustrated by Figure 5. It represents the full model of a certain data policy expressed as a chain of semantically clear activities with interfaces between them, as well as interfaces to activity implementations in particular IT services or software platforms.

It is worth mentioning once again that every aspect in the Figure 5 diagrame (such as File, Size, Format, Script or Log) should be thought of not as a particular artefact or a value but as a semantic wrapper of an artefact or a value. As a particular model serialization, these semantic wrappers can be RDF statements about artefacts or values.

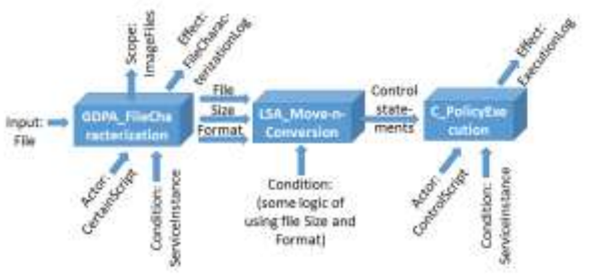
Scope: ImageFiles
Effect: FileCharacterizationLog
Input: File
GDPA_FileCharacterization
File
Size
Format
LSA_Move-n-Conversion
Control state-ments
C_PolicyExecution
Effect: ExecutionLog
Actor: CertainScript
Condition: ServiceInstance
Condition: (some logic of using file Size and Format)
Actor: ControlScript
Condition: ServiceInstance

Figure 5 Example of full policy definition

In real data policy modelling situations, it may be necessary to define more than one instance of each Activity type; as an example, there could be two Data Characterization Activities defined (one for the file size and another for the file format) in place of one in our example. Nevertheless, even differently defined Activities could be combined in a semantically clear network representing the same data policy.

If Activities in Figure 5 are clearly defined and sensibly combined in the Activity network, this eliminates any ambiguity in policy definition and execution exemplified by two interfering granular policies discussed back in Section 2.1 so that the actual result of the policy implementation becomes predictable and can be formally validated.

One of the strengths of the suggested model is a combination of its reasonable expressivity with its high flexibility as it is based on the idea of composition of activities that can be a) modelled differently b) implemented differently and c) operated (executed) differently. In the above example, scripts for file characterization and scripts for policy execution can be implemented using different software and operated by different components of the same service, or by different services, or even by different e-infrastructures.

The actual chain or network of activities, as well as definition of each of them (i.e. definition of all semantic wrappers) could be done in a certain authoring tool with a graphic user interface and RDF as a model serialization format. Development of such a tool has been beyond resources available for this conceptual work; however, such a tool is worth mentioning as one of the elements of an IT architecture that can support data policies formulation, execution and validation.

4 IT architecture for activity-based data policy management

The proposed IT architecture is presented by Figure 6 with the most essential components and information flows (that would constitute a core operational platform for data policy management) designated as filled-in boxes and arrows; more advanced components and flows are designated as dashed boxes and arrows with a blank background.

As already suggested, having policy Activities authoring tools with GUI and possibility to serialize Activity networks in a semantically explicit format such as RDF is essential for good levels of adoption of the suggested approach and therefore such authoring tools should be a part of a sensible IT architecture for data policy management. In addition, what is required is a repository where policy designs can be stored and retrieved from.

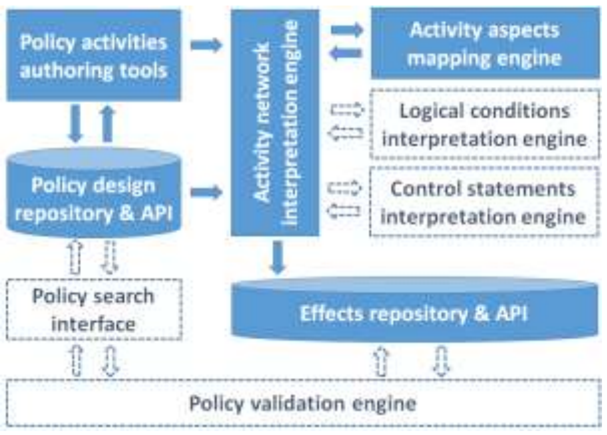
Policy activities authoring tools
Activity network interpretation engine
Activity aspects mapping engine
Logical conditions interpretation engine
Policy design repository & API
Control statements interpretation engine
Policy search interface
Effects repository & API
Policy validation engine

Figure 6 IT architecture for activity-based policy management

Activity network interpretation engine picks up Activity network from the authoring tools or repository and executes them. In order to execute activity networks in a particular IT environment (software platforms and services), a mapping engine is required that maps Activities and their aspects (such as Conditions or Outputs) to configuration files and executable scripts.

In addition to this generic mapping engine, specific engines for logical conditions and control statements can be implemented. Effects repository stores Effect aspects of each Activity; it is a generalization of logging service and contains semantically clear tracks of Activities execution. Policy search interface can be designed for searching and sharing data policies.

For the purposes of data archive or data infrastructure audit, a policy validation engine is required that talks to policy search interface and to Effects repository. The actual validation can be based on matching graphs of artefacts resulted from policies execution with graphs of Activities in the policy design.

5 Conclusion

The problem of data policy modelling with reasonable crosswalks between high-level (read: textual) policies and their machine-executable implementations has yet to find a satisfactory solution. The challenges of policy design and implementation are even bigger when collaborative data infrastructures are operated in combination with the in-house software platforms.

The problem of semantically clear crosswalks and the problem of data policy implementation across organization-specific and external IT services can be addressed by adoption of certain policy modelling techniques and tools. Activity Model [11] can be a reasonable means for the design of such tools, with the idea that data policies can be represented as networks of Activities with interconnected aspects of them.

This work has introduced extensions to the Activity Model in order to make it fit for the task of data policy modelling. An example of using the Activity Model for the definition of a particular data policy has been given, and a possible IT architecture has been considered that can support data policy management based on Activity networks.

Acknowledgements

This work is supported by EUDAT 2020 project that receives funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement No. 654065. The views expressed are those of the author and not necessarily of the project.

References

[1] Giaretta, D. Advanced Digital Preservation. Springer, Heidelberg (2011)

[2] Bunakov, V., Jones, C., Matthews, B., Wilson, M. Data authenticity and data value in policy-driven digital collections. OCLC Systems & Services: International digital library perspectives, vol. 30 issue 4, pp. 212-231 (2014). doi: 10.1108/OCLC-07-2013-0025. Open Access version of the preprint: http://purl.org/net/epubs/work/12299882

[3] EUDAT Collaborative Data Infrastructure. https://www.eudat.eu/eudat-cdi

[4] EUDAT services. https://www.eudat.eu/services-support

[5] EUDAT B2SAFE service. https://www.eudat.eu/b2safe

[6] iRODS: Integrated Rule-Oriented Data System. https://irods.org/

[7] EUDAT Data Policy Manager. https://github.com/EUDAT-B2SAFE/B2SAFE-DPM

[8] RuleML Wiki pages. http://wiki.ruleml.org/index.php/RuleML_Home

[9] SWRL: A Semantic Web Rule Engine. https://www.w3.org/Submission/SWRL/

[10] ProvONE: A PROV Extension Data Model for Scientific Workflow Provenance. http://vcvcomputing.com/provone/provone.html

[11] Bunakov, V. Core semantic model for generic research activity. In 15th All-Russian Conference "Digital Libraries: Advanced Methods and technologies, Digital Collections" (RCDL 2013), Yaroslavl, Russia, 14-17 Oct 2013, CEUR Workshop Proceedings (ISSN 1613-0073) 1108, 79-84 (2013). Persistent URL: http://purl.org/net/epubs/work/10938342

[12] SCAPE: Scalable Preservation Environments project. http://scape-project.eu/

[13] SCAPE Catalogue of Preservation Policy Elements. http://scape-project.eu/wp-content/uploads/2014/02/SCAPE_D13.2_KB_V1.0.pdf

[14] Practical Policy Implementations Report. http://dx.doi.org/10.15497/83E1B3F9-7E17-484A-A466-B3E5775121CC