WFDU-net: A Workflow Notation for Sovereign Data Exchange

Heinrich Pettenpohl[1], Daniel Tebernum[1] and Boris Otto[1,2]
[1]*Data Business, Fraunhofer Institute for Software and Systems Engineering, Dortmund, Germany*
[2]*Industrial Information Management, TU Dortmund University, Dortmund, Germany*

Keywords: Petri Net, Data Sovereignty, WFD-net, Usage Control, CTL*.

Abstract: Data is the main driver of the digital economy. Accordingly, companies are interested in maintaining technical control over the usage of their data at any given time. The International Data Spaces initiative addresses exactly this aspect of data sovereignty with usage control enforcement. In this paper, we introduce the so-called Workflow with Data and Usage control network (WFDU-net) model. The data consumer can visually define his or her workflow using the WFDU-net model and annotate the data operations and context. With model checking we validate that the WFDU-net follows the usage policies defined by the data owner. Afterwards, the compliant WFDU-net can be executed by exporting the WFDU-net in a Petri Net Markup Language (PNML). We evaluated our approach by using our example WFDU-net in a data analytics use case.

1 INTRODUCTION

Data is the main driver of the digital economy. Therefore, the European Union (EU) wants to provide users with tools to process their data and retain full control of their data as it flows within the EU and even across sectors (*The European Data Strategy*, 2020). This kind of control is needed both within an enterprise and within a data ecosystem, when it comes to data sharing between enterprises. Therefore, the International Data Spaces Association (IDSA) developed a reference architecture for sovereign data exchange between enterprises (Otto & Jarke, 2019). Data sovereignty means that the data owner can decide what the data consumer may use the data for. This is a contrast to today's widespread access control, where only access but not use is regulated. Technically, in the IDS, this means that the data owner annotates his or her usage policies to the raw data, so that the systems enforce these policies. Therefore, the IDS use the concept of usage control (Eitel et al., 2019) to provide the data owner with control over his or her data while, at the same time, allowing the data processing on the consumer side. On the consumer side, there is often not only one application processing the data, but a whole chain of applications. To link the applications with each other, workflow engines (Apache Airflow[1], Argo[2]) or message routers (Apache Camel[3], Apache NiFi[4]) or message bus systems (Apache Kafka[5], Apache ActiveMQ[6]) are used today. In the IDS, exemplary Apache Camel with ActiveMQ as well as Apache Airflow are used to exchange data between the applications. To enforce the IDS usage policies, the enforcement tools hooking into the message systems at runtime and check whether the usage policies are being violated and, if so, interrupt the data flow. The applications must also adhere to the terms of use.

The messaging systems do not natively support usage control. The user of these messaging systems mostly have to write down configuration files and use programmatic APIs instead of graphical user interfaces.

In this paper, we want to support the data owner to see what his or her data is being used for and we want to support the data consumer to dynamical create workflows for data processing with a built-in usage policy checking. Therefore, we present a visualization of the workflow, which consists of a control flow and a data flow. After designing the workflow, we use model checking to validate whether the workflow complies with the given usage policies. Finally, the workflow can be executed.

[1] https://airflow.apache.org/
[2] https://argoproj.github.io/projects/argo/
[3] https://camel.apache.org/
[4] https://nifi.apache.org/
[5] https://kafka.apache.org/
[6] https://activemq.apache.org/

Pettenpohl, H., Tebernum, D. and Otto, B.
WFDU-net: A Workflow Notation for Sovereign Data Exchange.
DOI: 10.5220/0010550402310240
In *Proceedings of the 10th International Conference on Data Science, Technology and Applications (DATA 2021)*, pages 231-240
ISBN: 978-989-758-521-0

We will test the execution with the analytics network approach developed by Tebernum *et al.,* which is capable of following usage policies by design. The overall approach allows us to validate the workflow even before it is executed. This provides the user with early feedback already during the design phase. Additionally, we can forward the workflow definition to the data owner to keep the data usage transparent and allow him or her to check the validity himself or herself.

So, in summary, we have three requirements:

- Graphical visualization of data flows
- Proven validation of data flows with reference to usage policies at design time, so that data flows do not simply break off.
- Conversion of the data flows to message systems

The remainder of this paper is structured as follows. Section 2 focuses on related work. Section 3 introduces WFDU-nets. In Section 4, we present a use case that will be realized in the following sections. In Section 5, we formalize the model checking rules and implement them. Section 6 concludes the paper and also provides limitations and future work.

2 RELATED WORK

First, we look at different modelling approaches to select a graphical visualisation which also can prove the usage policies. This will result into our research gap. Then we look at existing usage control solutions and how they enforce policies.

2.1 Petri Nets

In computer science, there are different types of modelling for data in general and workflows in particular. For example, it is possible to use Unified Modeling Language (UML) (Object Management Group [OMG], 2017), Business Process Modeling Notation (BPMN) (Camunda services GmbH et al., 2010), or Event-driven Process Chains (EPC) (Becker et al., 2012). All these modelling languages were extended over the time to fulfil different requirements. However, we will focus on Petri nets for our approach as they are mathematically well-founded and can therefore be better used for analysis purposes in addition to pure modelling (Murata, 1989; van der Aalst, 1998). Furthermore, they can visualize the workflow for the user and it can be used to visualize the execution of a workflow via markings. There are some variations of the original Petri nets (Murata, 1989), for example Coloured Petri Nets (Jensen, 2013), where the colour can be used to attach information to a token. Also, BPMN, EPC, and UML activity diagrams can be translated into Petri nets.

Some work has been done to unify data and the control flow in Petri nets. With Dual Workflow Nets (DFN), Varea et al. (Varea et al., 2006) introduced a Petri net model that combines control and data flows in one Petri net with additionally added data transitions. In contextual nets (Baldan et al., 2012), flow relation itself is used to perform data operations such as reading data within a Petri net. In PN-DO (Petri Nets with Data Operation), Xiang et al. (Xiang et al., 2017) customized a Petri net to generate, update, and write data to places using a specialized flow relation. Another method to add data flows to a Workflow net (WF-net) is proposed with the WFIO-net, where the data is explicitly modelled using specific data places (Cong et al., 2014). This results in a redefined transition fire method, so that standard tools can no longer be used. (Katt et al., 2009) used Coloured Petri nets to define a Usage control Coloured Petri Net (UCPN) with colours for the three main elements of a usage policy: subject, object, and context. The UCPN is used to verify whether a subject is allowed to use an object based on attributes. However, the subject and the context in real world examples are not free flowing, they are directly attached to an application/transition and from our view not a token/colour.

Another method is to include labels (function, context, etc.) in the transitions and use a guard function to select the correct data, as WFD-nets (Workflow Nets with Data) do (Trčka et al., 2009). The guard functions in WFD-nets use write, read, and delete as data operations. WFD-nets are a specialized form of WF-nets (Workflow-nets) (van der Aalst, 1997, 1998). WFD-nets are Petri nets with only one start node and one end node, all other nodes have to be located between the start and end node. The WFD-nets is primarily designed to find data flow errors. Our paper extends the WFD-nets to check usage control policies.

2.2 Usage Control

In the early 2000s, Sandhu and Park introduced usage control (UCON) as an extension of access control (Sandhu & Park, 2003). This initial definition was extended to UCON_{ABC} , which comprises Authorizations, oBligations, and Conditions, referring to attributes of subjects and objects. Lili and Zhigang introduced a CTL logic for usage control enforcement (Lili & Zhigang, 2019). They defined attributes attached to the subject, object, and context to use the CTL logic for checking whether the attributes are fulfilled for some policies. Another approach is proposed in (Zhang et al., 2005) where $\mathrm{UCON}_{\mathrm{ABC}}$ is formalized in Laport's Temporal Logic of Actions (TLA). Basin et al. used first-order temporal logic to monitor usage control policies (Basin et al., 2010; Basin et al., 2011). In (Pretschner et al., 2009), a Linear Time Logic (LTL) dialect is used to analyse policies. For dynamically changing policies, (Elrakaiby & Pang, 2014)introduce an Action Computation Tree Logic (ACTL) approach.

Zrenner et al. presented how usage control can be used for business ecosystems, especially in the IDS (Zrenner et al., 2019). With LUCON there is also a framework to enforce policies in the IDS (Schütte & Brost, 2018), the policies are described in a custom DSL and translated into Prolog to realize enforcement on the data flow.

3 WORKFLOW NETS WITH DATA AND USAGE CONTROL

To model our approach, we are using workflow with data nets (WFD-nets), which are specialized workflow nets (WF-nets). A WF-net itself is a specialized Petri net. Therefore, we start with a first introduction of Petri nets. A Petri net is a graph with two type of nodes: places and transition. The nodes are connected with flow relations, where a flow relation connects a place with a transition or vice versa. The initial markers are tokens, which are placed in places. When a transition fires, it consumes a token from the previous place and generates a new token behind it. A WF-net has exactly one source place and one sink place and all nodes are on a path from this source place to the sink place. We formalize this in the following.

Definition 1. A WF-net is a Petri net $\Sigma = (P, T, F, M_0)$, where:

$P \cap T = \emptyset \; and \; P \cup T \neq \emptyset$ (1)

P is a finite set of **places**. (2)

T is a finite set of **transitions** T such that $P \cap T = \emptyset$. (3)

$F \subseteq (P \times T) \cup (T \times P)$ is a set of directed **flow relation**. (4)

$M_0 : P \rightarrow \mathbb{N}_0$ is the initial marker. (5)

$p_s \in P$ where p_s is one source (start) place such that $\dot{}p_s = \emptyset$ (6)

$p_e \in P$ where p_e is one sink (end) place such that $p_e\dot{} = \emptyset$ (7)

Each node $x \in P \cup T$ is on a path from $p_s$to p_e (8)

$\forall p \in P, M_0(p) = 1$ if $p = p_s$, and otherwise $M_0(p) = 0$. (9)

A WF-net is used to model the control flow of different activities from a starting point to the end of the process. Beside this control flow, a typical workflow also consists of a data flow. Thus, a WFD-net is a WF-net that is extended with read, write, erase, and guard operations for data elements. For example, a transition may read a data element. Therefore, this data element is expected to be written before the transition is executed. The operation write means that the data element gets a new value. If the data element value is already written, it is overwritten by the write operation. Next, the erase operation deletes the value of the data element. Finally, the guard operation is used to check a data element before a transition is executed. For example, the guard tests whether a data element is present or not. Therefore, it reads the value of the data element, whether a value is present or not, and afterwards executes the transition or not.

Definition 2. A WFD-net (Trčka et al., 2009) is a 10-tuple.
$\Sigma_{WFD} = (P, T, F, M_0, D, G_D, Read, Write, Destroy, Guard)$, iff:

(P, T, F, M_0) is a WF-net. (1)

D is a set of data elements. (2)

G_D is a set of guards over D. (3)

Read: $T \rightarrow 2^D$ is the reading data labeling function. (4)

Write: $T \rightarrow 2^D$ is the writing data labeling function. (5)

Erase: $T \rightarrow 2^D$ is the erasing data labeling function. (6)

Next, we will enhance the WFD-net with usage policy related information. To do this, we are looking at what additional elements we need in the WFD-net.

3.1 Usage Policies

A Usage Policy describes what a subject can do on an object in a related context. Therefore, a policy consists of four main elements defined UCON specifications (Park & Sandhu, 2004; Sandhu & Park, 2003): subjects, actions, objects, and context. For our use case, we limit these elements to specific elements of our workflow. A subject is always the application connected to a transition in our WFD-net. Actions are related to three different categories. First, there are subject actions that are performed by a subject, as in our case an application that writes, reads, or erase an object. These operations can easily be extended to more concrete operations like transform, update, etc. in the future. Enforcement actions are actions that have a monitor character, e.g. updating attributes, logging a failed access, or checking a fulfilment of an obligation. Finally, obliged actions are actions that must be fulfilled by the subject. In the following, we call them Policies. In our case, the data elements are the objects. However, the context is still missing in the WFD-net. It describes in which environment the application is running, e.g. inside a company or in a cloud. The specific environment labels must be defined before the data owner has written his or her policies so that he or she knows the labels and can use them.

In the following definition, we connect applications to WFD-net and add their context to the transitions. In addition, we add policies to the data so that a Subject is aware of them while using data elements.

Definition 3. A WFD-usage-control-net is a 15-tuple $\Sigma_{WFDU} = (P, T, F, M_0,\ D,\ G_D,\ Read,\ Write,\ Erase,\ Guard,\ App,\ C,\ Context,\ R,\ Rules)$ iff,

$$(P, T, F, M_0, D, G_D, Read, Write, Erase, Guard) \text{ is a WFD-net.} \tag{1}$$

$$App \text{ is a set of Applications.} \tag{2}$$

$$\text{C is a set of Context elements.} \tag{3}$$

$$T = T_{App} \cap T_C \text{ where } T_{App} \text{ is an application transition and } T_C \text{ is a control flow transition with not data relevant acitivity.} \tag{4}$$

$$\text{Context: } T \rightarrow 2^C \text{ is the context labeling function.} \tag{5}$$

$$\mathcal{R} \text{ is a power set of a set of Policies.} \tag{6}$$

$$\text{DataPolicies: } D \rightarrow 2^{\mathcal{R}} \text{ is a DataPolicies labeling function.} \tag{7}$$

In the following, we will introduce a use case and implement it in an exemplary WFDU-net to make the approach more tangible.

4 USE CASE: METADATA EXTRACTION FOR A DATA CATALOG SYSTEM

Since data is an important asset, enterprises are interested in maintaining a good overview of it. In our case, we assume that the data consumed from the IDS should be listed in a data catalog. The data catalog

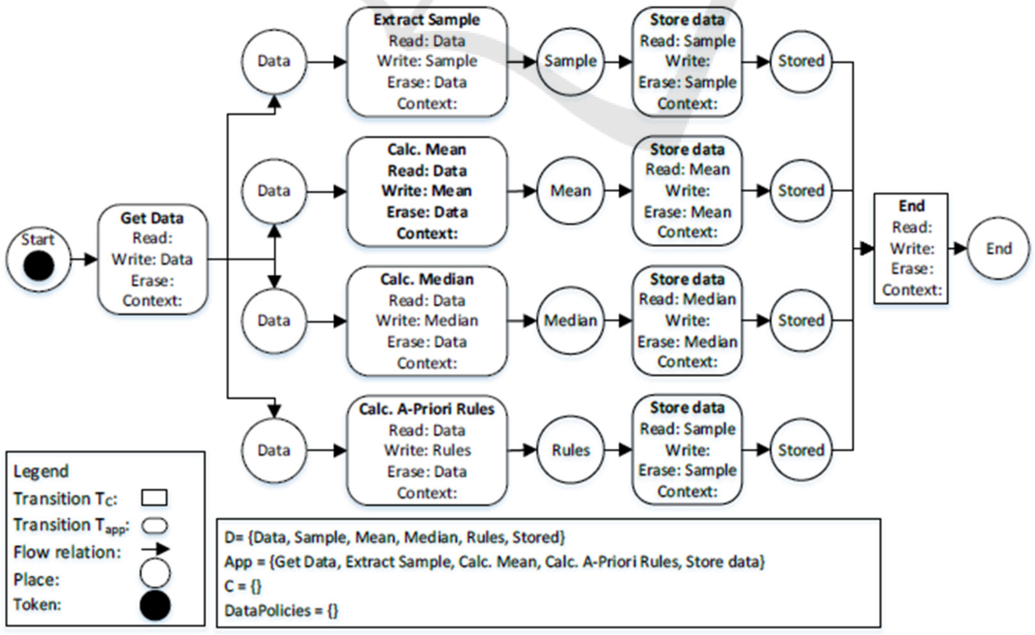


Figure 1: WFD-net for extracting metadata.

stores metadata about the data, thus providing advanced search capabilities to find the needed data more easily. Metadata can contain descriptive, structural, as well as administrative information, which is of great importance for running search queries and finding data (Press, 2004). This can be very heterogeneous depending on the data source. For text files, an automatic detection of keywords may be desirable. For images, an object recognition or knowledge of location may be helpful. For domain-specific data, a more specialized extraction of metadata may be necessary. An enterprise can therefore pre-define specific workflows and implement appropriate analysis tasks to extract metadata for different data types and use cases. A workflow engine located, for example, on an enterprise server or in the cloud can then execute these workflows.

However, there are situations where a workflow cannot simply be executed as it is. This may be the case, for example, if it is necessary to check in advance whether individual steps of the workflow can be applied to certain data. More experienced users will then want to extend, shorten, or even completely customize workflows for their own purposes. Even if the workflows are customizable, the usage policies specified by the data owner in the IDS have to be enforced. For example, confidential management documents should not be analyzed in a third-party cloud application, but only within the company on dedicated computers. Our visual notation helps users understand, model, and validate workflows that can support all these activities.

(Tebernum & Chabrowski, 2020) have designed and implement an analytics network that is located inside the boundaries of an enterprise and can be utilized for the execution of metadata extraction workflows. It is optimized for utilizing already existing computing and storage resources, e.g. edge devices. In addition, it consists of a decentralized data management system that prevents centralized data aggregation and allows data sovereignty to be maintained. Due to the focus on the architectural design, the authors did not provide a solution for a workflow editor with build in formal validation functionality. Workflows can only be written in the appropriate workflow language with expert knowledge and without further validation.

We address this shortcoming by applying the notation presented in this paper to visualize, build, and validate workflows for the proposed analytics network. We also show how data sovereignty properties can be applied.

The following example demonstrates how the notation can be used to support users and apply usage policies. The use case is located in the context of a pharmaceutical enterprise. Much data is generated, processed, bought, and sold on a regular basis. To keep track of all this data and to make it easy searchable, a data catalog is used. With predefined workflows, the necessary metadata is extracted according to data type and data domain. Figure 1 shows a shortened workflow that calculates and extracts standard metadata from a CSV files. For most files, the workflow can be used in this way. Starting from the file to be processed, four tasks are executed concurrently. Each subject action is represented as a rectangular petri net transition with its own read, write and erase functions. We distinguish between rectangles with corners or rounded corners, these represent control flow transitions or applications. The circles represent petri net places in which the tokens/data are located and can be consummated by transitions, or the transitions generate the data for the places behind them. In the figure, a sample is extracted and statistical metrics are calculated. For the more specific task of rule extractions, the analytics network should perform the calculation on a computer belonging to a high-performance group, this should be added to the notation later.

4.1 Usage Policies

The provided data is purchased from the IDS and is linked to some usage policies. One of these policies specifies that only a certain number of copies of the data may exist in the consumer company at the same time. For our example, we assume a fictional case in which the number is limited to a maximum of two copies. Another term of use may specify that the data must not pass national borders. Therefore, it must be ensured that the data is not accidentally analyzed on foreign computers, even if they are part of the enterprise's computer network structure. Due to the terms of use, the standard workflow cannot be applied unchanged. On the one hand, four tasks are performed simultaneously, each holding its own copy of the file. On the other hand, the workflow does not define any geographic restrictions for the computers performing the analysis tasks. Both policies are defined in the following.

Policy 1. Data analysis only in my country: All data processing must be done on computers in France. Therefore, the policy ensures that the operations using the data are only performed on transitions with the context *france*. This is an authorization and obligation rule defined by the UCON specification which authorize a subject to use a resource and determines via the obligation rule what tasks/actions a subject must perform when a resource is used.

Policy 2. Only two copies can be used at the same time: The policy ensures that only two copies of a data set could be used in parallel. In fact, this means that a maximum of only two transitions can executed at the same time. This is a concurrency rule defined by the UCON specification which expresses the way multiple subjects are allowed to use a resource at the same time.

Besides these two policies from the use case above, we also consider generic policies that were specified by the IDS initiative in (Eitel et al., 2019).

Policy 3. Allow/Inhibit the data usage for a specific operation: The policy ensures that a particular operation is allowed or inhibited to use the data. In our case, we can distinguish between read, write, and delete operations. However, it is easy to extend the WFDU-net with more specific operations that can be allowed or inhibited.

Policy 4. Use data and delete it after: After the data is used by an application, it must be ensured that it is deleted.

To check the policies described, we must first define the model checking algorithm for the WFDU-net. The model checking can then be used to check whether the WFDU-net follows these usage policies or break them.

5 FORMALIZATION AND IMPLEMENTATION

First, we unfold the WFDU-net in a similar way to the unfolding of a WFD-net (Trčka et al., 2009). We preprocess our WFDU-net model and split each transition $t \in T$ into its start t_s and its end t_e connected by a place p_t. The transition is executed $exec(t)$ if at least one token is in p_t. To check whether our model and all transitions follow our defined policies, we will define model checking formulas. For this purpose, we will first introduce the Kripke structure and CTL* for our WFDU-net. Then, we will formalize our policies.

5.1 CTL*

CTL* (computation tree logic with branching time temporal logic) is a generalization of the two logics LTL and CTL. CTL* uses a Kripke structure to define states and their relation to each other. In this case, the states are marking positions in the WF-net.

Definition 4. A Kripke structure (Trčka et al., 2009) is a 4-tuple $M = (S, N, L, \rightarrow)$ where:

$$S \text{ is a finite set of states.} \quad (1)$$

$$N \text{ is a non-empty set of atomic propositions.} \quad (2)$$

$$L: S \rightarrow 2^N \text{ is a state labeling function.} \quad (3)$$

$$\rightarrow \subseteq SxS \text{ is a transition relation.} \quad (4)$$

An atomic proposition N is a definition of a marking in a state s. It is defined as $N = \{p \geq i | p \in P, i \in \mathbb{N}$ where $p \geq i$ means that place p holds at least i tokens. The labels of states map the markings to the sets of atomic propositions: $\bigvee_{p \in P}$ *and* $i \in \mathbb{N}, (p \geq i) \in L(m)$ iff $m(p) \geq i$. Therefore, $L(s)$ is the set of atomic propositions that *hold* in s. $s \rightarrow s'$ is the relation from one state marking to another, which is a *step* from a state s to s'. Every *step* is a possible fired transition in our WF-net. A *path* π is then an infinite sequence of states $\pi = s_0, s_1, s_2, \ldots$ such that $s_k \rightarrow s_{k+1}$ for all $0 \leq k < n, s_n \nrightarrow$, and $s_k = s_{k+1}$ for all $k \geq n$. π^k then denotes the path $s_k, s_{k+1}, s_{k+2}, \ldots$.

Definition 5. (Trčka et al., 2009) We define two classes of formulas: state Φ and path Ψ formulas. They are defined by the syntax:

$$\phi ::= n | \neg\phi | \phi \wedge \phi | E\psi$$
$$\psi ::= \phi | \neg\psi | \psi \wedge \psi | X\psi | \psi U \psi$$
with $n \in N, \phi \in \Phi$, and $\psi \in \Psi$.

Definition 6. A state formula ϕ is valid in a state s (notation: $s \vDash \phi$) and path formula ψ is valid on a path π (notation: $\pi \vDash \psi$) (Trčka et al., 2009) when:

$$s \vDash n \text{ iff } n \in L(s) \quad (1)$$

$$s \vDash \neg\phi \text{ iff } s \nvDash \phi \quad (2)$$

$$s \vDash \phi_1 \wedge \phi_2 \text{ iff } s \vDash \phi_1 \text{ and } s \vDash \phi_2 \quad (3)$$

$$s \vDash E\psi \text{ iff there exists a path } \pi \text{ from } s \text{ such that } \pi \vDash \psi \quad (4)$$

$$\pi \vDash \phi \text{ iff } s \text{ is the first state of } \pi \text{ and } s \vDash \phi \quad (5)$$

$$\pi \vDash \neg\psi \text{ iff } \pi \nvDash \psi \quad (6)$$

$$\pi \vDash \psi_1 \wedge \psi_2 \text{ iff } \pi \vDash \psi_1 \text{and } \pi \vDash \psi_2 \quad (7)$$

$$\pi \vDash X\psi \text{ iff } \pi^1 \vDash \psi \quad (8)$$

$$\pi \vDash \psi U \psi' \text{iff there exists a } j \geq 0 \text{ such that } \pi^j \vDash \psi', \text{ and } \pi^k \vDash \psi \text{ for all } 0 \leq k < j. \quad (9)$$

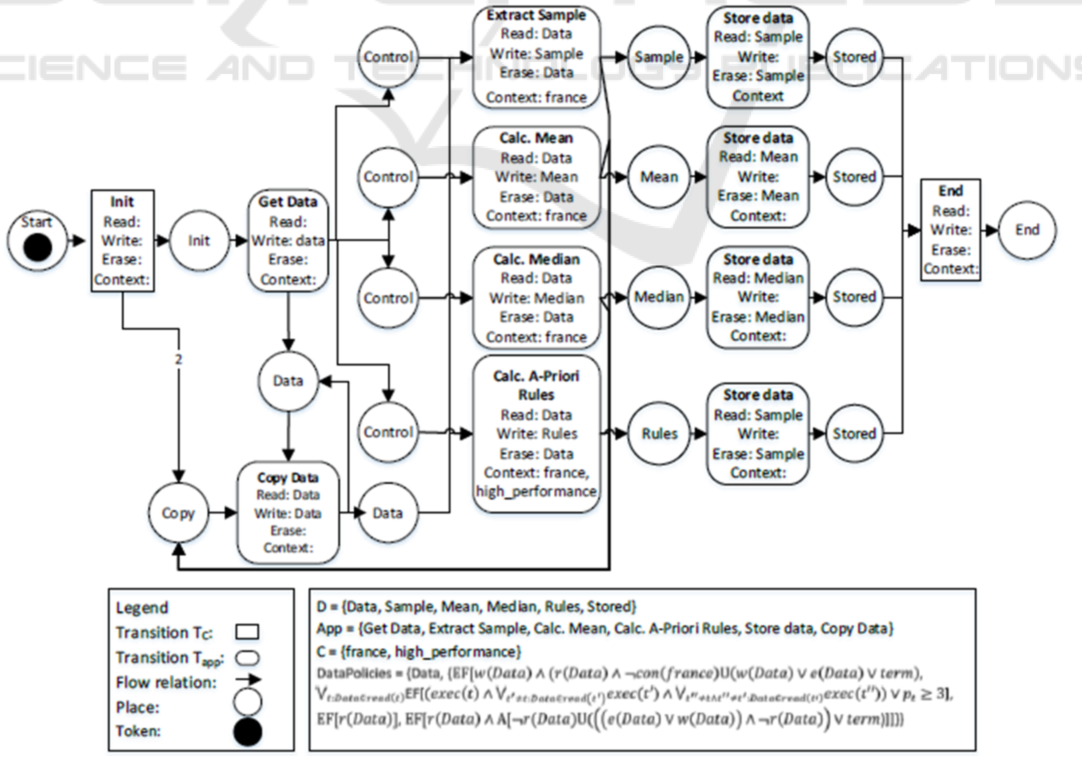
The above definitions are used to notate some informal definition (Trčka et al., 2009): Aψ is "ψ *holds* along all paths"; Eψ is "ψ *holds* along at least one path"; Fψ is "ψ *holds* in future"; EFψ is "ψ *holds* in some future state"; ψUψ' is "ψ *holds* until ψ' holds".

To define our policies as formula, we need to define the atomic proposition. Therefore, we create some abbreviations. To formulate that the transition t is now executed and write the data element $d \in D$, we define $w(d) = \bigvee_{t:d \in write(t)} exec(t)$. For read and erase, the definition is $r(d) = \bigvee_{t:d \in read(t)} exec(t)$ and $e(d) = \bigvee_{t:e \in erase(t)} exec(t)$. To formulate that the transition is executed in a special context $c \in C$, we define $con(c) = \bigvee_{t:c \in context(t)} exec(t)$. We also define $change(t)$ for the set $\{d | d \in write(t) \cup erase(t)\ \}$ and $use(t)$ for the set $\{d | d \in read(t) \cup write(t) \cup erase(t)\}$. The final state of the workflow is denoted *term,* defined by $end = 1 \wedge \bigwedge_{p \in P\{end\}} (p = 0)$.

5.2 Model Checking Formulas

First, we have to check if the underlying WFD-net is valid. The data operations are only labeled to the transitions, so we have to use the formalized anti-patterns from (Trčka et al., 2009), especially to check missing data (is data read before it is written?) and inconsistent data (is data written or erased while it is being used in parallel?). Only if no data is missing and no inconsistent data is possible, we can ensure that the policies can be enforced. Missing data is defined as $\mathrm{E}[\left(\neg w(d)\mathrm{U}\left(r(d) \vee e(d)\right)\right) \vee \mathrm{F}\left[e(d) \wedge \left(\neg w(d)\mathrm{U}\left(r(d) \vee e(d)\right)\right)\right]]$. In the first half, the formula states that there is a path where no writing to d happens until reading d or erasing d takes place. The second part states that d is erased and not written again until it is read or erased. Inconsistent data is defined as $\bigvee_{t \in T: d \in change(t)} \mathrm{EF}[(exec(t) \wedge \bigvee_{t' \neq t: d \in use(t')} exec(t')) \vee p_t \geq 2]$. This means for a transition t where d is changed during execution, there is at least one more transition t' where data is used. The last part of the formula captures the fact that a changing transition can be executed more than once in parallel. Our policies are defined as follows.

Policy 1. Data Analysis Only in My Country. In our case, the transitions with the context *france* are in the country. To violate the policy, we have to check if there is at least one path with read data and not $c = france$ until the data is overwritten or erased or the workflow terminates. This results in the formula $\mathrm{EF}[w(d) \wedge (r(d) \wedge \neg con(c)\mathrm{U}(w(d) \vee e(d) \vee term)$.



D = {Data, Sample, Mean, Median, Rules, Stored}
App = {Get Data, Extract Sample, Calc. Mean, Calc. A-Priori Rules, Store data, Copy Data}
C = {france, high_performance}
DataPolicies = {Data, {$\mathrm{EF}[w(Data) \wedge (r(Data) \wedge \neg con(france)\mathrm{U}(w(Data) \vee e(Data) \vee term)$,
$\bigvee_{t: Data \in read(t)} \mathrm{EF}[(exec(t) \wedge \bigvee_{t' \neq t: Data \in read(t')} exec(t') \wedge \bigvee_{t'' \neq t \wedge t'' \neq t': Data \in read(t'')} exec(t'')) \vee p_t \geq 3]$,
$\mathrm{EF}[r(Data)]$, $\mathrm{EF}[r(Data) \wedge \mathrm{A}[\neg r(Data)\mathrm{U}(\left(\left(e(Data) \vee w(Data)\right) \wedge \neg r(Data)\right) \vee term)]]]\}$

Figure 2: Policy compliant WFDU-net example for metadata extraction.

Policy 2. Only Two Copies Can be used at the Same Time. With the purchased data set, only two transition should be executed at the same time. Therefore, we reformulate the policy to "Only two reads in parallel are allowed". To violate the policy, there has to be a path where the data is read three times in parallel.

- Read the data once: $EF[w(d) \wedge r(d)]$. Since we check missing data in advance, we do not have to check the write operation, as the previous check would have already failed. Therefore, the formula is:
  $$\mathrm{EF[r(d)]} = \bigvee_{t:d\in read(t)} \mathrm{EF}[exec(t)].$$
- Read the data twice:
  $$\bigvee_{t:d\in read(t)} \mathrm{EF}[(exec(t) \wedge \bigvee_{t\neq t:d\in read(t)} exec(t)) \vee p_t \geq 2].$$
- Read the data three times:
  $$\bigvee_{t:d\in read(t)} \mathrm{EF}[(exec(t) \wedge \bigvee_{t'\neq t:d\in read(t')} exec(t') \wedge \bigvee_{t''\neq t\wedge t''\neq t':d\in read(t')} exec(t'')) \vee p_t \geq 3].$$

The formula for n times is similar to the previous approaches.

Policy 3. Allow/Inhibit the Data Usage for a Specific Operation. Firstly, we focus on inhibit the read operation. To violate this policy, there is a path where the data is read: $\mathrm{EF}[r(d)]$. The formula for writing or erasing data is built the same way. For the validation that an operation is allowed, nothing needs to be done, because it does not matter if the data is used or not.

Policy 4. Use Data and Delete it After. The policy is violated if there is a path where the data is read ($\mathrm{EF}[r(d) \wedge \ldots]$) such that, in all possible continuations of this path, no reading takes place until the data gets erased, overwritten, or the workflow terminates ($\mathrm{A}[\neg r(d) \mathrm{U}(\Big(\big(e(d) \vee w(d)\big) \wedge \neg r(d)\Big) \vee term\,)]$). As we assume that the operations are executed in the order read, write, and erase, we have to ensure, that the data is not read in a task before it is erased or overwritten. The result formula is: $\mathrm{EF}[r(d) \wedge \mathrm{A}[\neg r(d) \mathrm{U}(\Big(\big(e(d) \vee w(d)\big) \wedge \neg r(d)\Big) \vee term)]]$.

5.3 Implementation

We used CPN Tools to model our WFDU-net and check the patterns and policies from the previous section. Since the given use case is not yet following any policies, the model checking will fail. Figure 2 presents the redesigned compliant WFDU-net of Figure 1. Please note that there are also other ways than the proposed one to model the WFDU-net compliant to the policies. We insert the labels to annotate that the application is running inside the company in France. First, an initial step named Init was inserted. This allows a control flow before the data flow, both to fetch the data exactly once (Get Data) and to control that each subject action (Extract Sample, Calc. Mean, Calc. Median, Calc A-Priori Rules) is executed exactly once (see places to the left of the subject actions). When initializing, two control flow tokens are also created at the bottom left to create exactly two copies of the data set. In the process, the consummated data set is always created again to the place above the Copy Data Transition itself. The two resulting data copies are each used by one of the subject action transitions. After completion of the action, these again generate a new copy control flow token to enable another copy of the data set. This is done until each subject action is executed exactly once, see control flow tokens from the very beginning (to the left of the subject actions). This ensures that there are always a maximum of two data sets for simultaneous processing.

After the model is created and checked, we export the Petri net to the Petri Net Markup Language (PNML) which is an XML-based format. By transpiling this into native workflow languages like Argo or Airflow, the workflow can then be executed. We tested this using the analytics network from (Tebernum & Chabrowski, 2020).

6 CONCLUSIONS

We presented an approach to model the data flow for data with usage policies. With this approach we tackle our three requirements: (1) Graphical visualisation of data flows; (2) Proven validation of data flows with reference to usage policies at design time, so that data flows do not simply break off; (3) Conversion of the data flows to message systems. Therefore, we developed a Workflow with Data and Usage control network (WFDU-net) based on the WFD-net approach (Trčka et al., 2009), so that also usage policies can be validated, next to data flow errors.

To test our approach, we presented a use case which is based on a real-life case of a pharmaceutical company that buys data from the International Data Spaces and has to enforce the given usage policies. These policies where converted to CTL* logic and then executed in CPN-Tools. We redesign the given workflow of the underlying analytics network to be compliant with the given policies, so that the model checking is valid. Afterwards, existing workflow engines and analytic networks can use the model. In this context, we have shown that the approach can test both context-based and parallel events. For this purpose, all possible actions are calculated, transferred into a Kripke structure and then checked.

One limitation is that the approach cannot test temporal obligations at design time. Another limitation of our approach is that only the whole petri net can be validated with the given usage policies. Checking partial sections, for example if the policy is changed during the data flow, is not possible. We can circumvent this fact by dividing the data flow into appropriate sections.

In the future, we will provide more policy definitions. In addition, we will introduce our own tooling to read the IDS policies and automatically generate the CTL* formulas for the model checking.

ACKNOWLEDGEMENTS

This work was funded by the German Federal Ministry of Education and Research (BMBF) in context of the InDaSpacePlus project (no. 01IS17031) and by the Fraunhofer-Cluster of Excellence »Cognitive Internet Technologies«.

REFERENCES

Baldan, P., Bruni, A., Corradini, A., König, B., Rodríguez, C., & Schwoon, S. (2012). Efficient unfolding of contextual Petri nets. *Theoretical Computer Science*, *449*, 2–22.

Basin, D., Harvan, M., Klaedtke, F., & Zălinescu, E. (2011). MONPOLY: Monitoring usage-control policies. In *International conference on runtime verification* (pp. 360–364). Springer.

Basin, D., Klaedtke, F., & Müller, S. (2010). Policy monitoring in first-order temporal logic. In *International Conference on Computer Aided Verification* (pp. 1–18). Springer.

Becker, J., Probandt, W., & Vering, O. (2012). Modellierungssprachen. In J. Becker, W. Probandt, & O. Vering (Eds.), *Grundsätze ordnungsmäßiger Modellierung* (pp. 4–30). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-30412-5_2

Camunda services GmbH, IBM Corp., OMG, I., PNA Group, SAP AG, & Trisotech, I. (June 2010). *BPMN 2.0 by Example: Version 1.0*. https://www.omg.org/cgi-bin/doc?dtc/10-06-02

Cong, L. I., ZENG, Q., & Hua, D. (2014). Formulating the data-flow modeling and verification for workflow: A Petri net based approach. *International Journal of Science and Engineering Applications*, *3*, 107–112.

Eitel, A., Jung, C., Kühnle, C., Bruckner, F., Brost, G., Birnstill, P., Nagel, R., Bader, S., & Steinbuß, S. (2019). *Usage Control in the International Data Space: Position Paper* [Version 2.0]. https://www.internationaldataspaces.org/wp-content/uploads/2020/06/IDSA-Position-Paper-Usage-Control-in-IDS-2.0.pdf

Elrakaiby, Y., & Pang, J. (2014). Dynamic analysis of usage control policies. In *2014 11th International Conference on Security and Cryptography (SECRYPT)* (pp. 1–13). IEEE.

The European data strategy: Shaping Europe's digital future. (2020). https://ec.europa.eu/commission/presscorner/api/files/attachment/862109/European_data_strategy_en.pdf.pdf

Jensen, K. (2013). *Coloured Petri nets: basic concepts, analysis methods and practical use* (Vol. 1). Springer Science & Business Media.

Katt, B., Zhang, X., & Hafner, M. (2009). Towards a usage control policy specification with Petri nets. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems.* Symposium conducted at the meeting of Springer.

Lili, X., & Zhigang, Z. (2019). Formal Specification of Concurrent Enforcement UCON Model with CTL Logic. In *International Conference on Artificial Intelligence and Security* (pp. 627–641). Springer.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, *77*(4), 541–580.

Object Management Group. (December 2017). *OMG® Unified Modeling Language® (OMG UML®): Version 2.5.1.* https://www.omg.org/spec/UML/2.5.1/PDF

Otto, B., & Jarke, M. (2019). Designing a multi-sided data platform: findings from the International Data Spaces case. *Electronic Markets*, *29*(4), 561–580.

Park, J., & Sandhu, R. (2004). The UCONABC usage control model. *ACM Transactions on Information and System Security (TISSEC)*, *7*(1), 128–174.

Press, N. (2004). *Understanding metadata. National Information Standards Organization*. ISBN1-880124-62-9. Available at: www. niso. org/standards/resources

Pretschner, A., Rüesch, J., Schaefer, C., & Walter, T. (2009). Formal analyses of usage control policies. In *2009 International Conference on Availability, Reliability and Security* (pp. 98–105). IEEE.

Sandhu, R., & Park, J. (2003). Usage control: A vision for next generation access control. In *International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security.* Symposium conducted at the meeting of Springer.

Schütte, J., & Brost, G. S. (2018). LUCON: Data flow control for message-based IoT systems. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE).* Symposium conducted at the meeting of IEEE.

Tebernum, D., & Chabrowski, D. (2020). A Conceptual Framework for a Flexible Data Analytics Network. *Proceedings of the 9th International Conference on Data Science, Technology and Applications*(Volume 1).

Trčka, N., van der Aalst, W. M. P., & Sidorova, N. (2009). Data-flow anti-patterns: Discovering data-flow errors in workflows. In *International Conference on Advanced Information Systems Engineering.* Symposium conducted at the meeting of Springer.

van der Aalst, W. M. P. (1997). Verification of workflow nets. In *International Conference on Application and Theory of Petri Nets.* Symposium conducted at the meeting of Springer.

van der Aalst, W. M. P. (1998). Three good reasons for using a Petri-net-based workflow management system. In *Information and Process Integration in Enterprises* (pp. 161–182). Springer.

Varea, M., Al-Hashimi, B. M., Cortés, L. A., Eles, P., & Peng, Z. (2006). Dual Flow Nets: Modeling the control/data-flow relation in embedded systems. *ACM Transactions on Embedded Computing Systems (TECS)*, *5*(1), 54–81.

Xiang, D., Liu, G., Yan, C., & Jiang, C. (2017). Detecting data-flow errors based on Petri nets with data operations. *IEEE/CAA Journal of Automatica Sinica*, *5*(1), 251–260.

Zhang, X., Parisi-Presicce, F., Sandhu, R., & Park, J. (2005). Formal model and policy specification of usage control. *ACM Transactions on Information and System Security (TISSEC)*, *8*(4), 351–387.

Zrenner, J., Möller, F. O., Jung, C., Eitel, A., & Otto, B. (2019). Usage control architecture options for data sovereignty in business ecosystems. *Journal of Enterprise Information Management*.