Semantic model of the Computing Capacity matching within the FAIR Data Train

BARAN GÜLBEY, University of Twente, The Netherlands

b.gulbey@student.utwente.nl

ABSTRACT

In order to improve healthcare and science, patient data is essential. Especially because health data records are regarded as sensitive information. This is where the concept of FAIR Data Train (FDT) comes into play, where 'Trains' are algorithms that visit 'Data Stations' where health data is stored. By allowing the Train algorithms to access the data locally at each station, the FDT model promises safe and privacy-preserving data analysis. This research identifies the computing capabilities of Data Stations and computing requirements of Trains to support the evaluation of whether Data Stations are capable of running Trains. The relevant computing components are compiled into the 'Computing Environment', which describes the Data Station's computing capabilities and the Train's computing requirements. The Computing Environment and its various computing components form the basis for the design of an RDF semantic model, which is then validated against a set of conditions using the Shapes Constraint Language (SHACL). By achieving an accurate semantic model of the Computing Capacity matching, this work optimizes interoperability and thereby improves the overall efficiency of the FDT. The development of a semantic model for the Computing Environment of Trains and Data Stations will significantly contribute to the overall goals of the FAIR Data Train.

Key Words: FAIR Data Train, Computing Capacity, Semantic Model, Metadata, SHACL, Interoperability, RDF

1 INTRODUCTION

Over the past few years, there has been a big increase in the production of both structured and unstructured data by institutions and individuals around the world. Nowadays, the trend of having a large volume of data is commonly referred to as "big data", where most sectors are searching for ways to utilize it for their benefit [6]. The Healthcare sector in general is one of the biggest data producers worldwide, every patient has their data documented constantly. While traditionally this data is stored in hard copy format, the healthcare industry is slowly transitioning to being fully digital [6]. With sensitive health data becoming digital there are a few issues that have to be dealt with carefully due to safety and legal considerations. For example, regulations like the European GDPR hinder data sharing by having strict requirements regarding the protection of personal (health) data [2]. Nevertheless, these issues are outweighed by the number of possibilities that become available with digitized data. By properly managing health data, a limitless amount of analysis could be done on the big data. Scientists and doctors can use data like never before, improving medical procedures, using resources more efficiently, and saving time in general.

When doing data analysis, there is a lot of data needed. This is a big problem when it comes to health data because of it being sensitive information. Two options are commonly explored: data sharing and centralization. However, both of these solutions are unrealistic options because of their downsides. Data sharing is dangerous because of personal health data being sensitive information, and the increase of cyber-attacks. Data centralization is also not optimal, because of the time and costs it will take for the data to be hosted and maintained.

This is where the concept of the FAIR Data Train (FDT) comes in as a solution. It provides an infrastructure where instead of requesting and receiving datasets, specific questions can be asked which will result in a specific answer [2]. The FDT works by bringing the algorithm to the data instead of traditionally moving the data to the algorithms. The FDT achieves this by having the so-called 'Trains' and 'Data Stations'. The algorithm is portrayed as a Train that 'travels' to Data Stations. The Data Station is a software solution that is managed by a health organization, which possesses health data. At the Data Station, the Train will run its algorithm and start the data analysis. Once it is finished it will travel back to its origin, with the results of the analysis [1]. This way the data can stay where it is in the most privacy-preserving manner.

Interoperability is carefully thought of within the FDT, which makes it possible for different systems (Trains and stations) to work seamlessly together. It would be unfortunate if a Train is deployed but it returns with no results because of unforeseen issues along the way. Metadata definitions play a significant part in the FDT infrastructure, especially the Train and Data Stations where the matching happens. This paper will focus on the Computing Capacity matching between them. Every Data Station has a computer system that controls the data at that location, the system has specific computing power that is available to execute analytics tasks [1]. The Train, however, has certain computing power requirements that need to be minimally available at the Data Station for it to run its algorithm without any performance problems.

1.1 Problem Statement

At the moment, metadata in the FDT is used for facilitating interoperability, however, some aspects of the Train and station matching are not complete or have not been explored yet. A metadata definition of the computing capacities for both the Trains and the stations would enable better metadata matching. This is important because an optimized matching process will lead to the whole FDT infrastructure working more efficiently.

Another aspect of why this is relevant is in the context of the research done by Martinez, where she researches the possibility of staging the FDT to the cloud [8, 12]. Cloud

TScIT 39, July 7, 2023, Enschede, The Netherlands

staging only takes place whenever there is not enough Computing Capacity available at the Data Station to run the Train's algorithm. The dynamic cloud staging is created based on certain computing parameters, this is where the semantic model of this paper will be useful. By having the Train's computing requirements defined as metadata, the system can easily create a virtual machine that has the correct computing resources available. This will make the cloud staging of the FDT more efficient by improving precision and reducing costs.

The goal of this paper is to research and design the semantic model that supports the metadata matching of the Computing Capacity within the FDT. To achieve this, the research will be conducted by answering three research questions.

- **RQ 1:** What are the computing resources that are necessary for the matching process of the Train's and Data Station's Computing Capacity?
- **RQ 2:** How can the semantic model of the Computing Capacity matching be constructed?
- **RQ 3:** How can the semantic model be validated using the Shapes Constraint Language?

First, the Computing Capacity metadata will be defined for the Trains and the Data Stations. This will involve understanding the computing resources of the Trains and the stations, such as central processing units, memory usage, and storage capacity. Once the relevant computing resources are found, the semantic model will be researched. This includes an RDF graph and a table with all the objects, paths, cardinalities, and datatypes. Finally, the semantic model will be validated using the Shapes Constraint Language, using the finished semantic model.

1.2 Structure

The structure of this paper is by first describing the background information of the key concepts of the FDT in section 2, which will build a basis for answering the research questions. Section 3 is for the methodology where the approach is explained to answer each research question. In section 4 the results and findings of the methodology will be discussed. Lastly, in section 6 the main conclusions are presented and recommendations for future work.

2 BACKGROUND

In this section, the knowledge module is constructed. This knowledge will build a basis for the methodology section and the results, which will be used to answer the research questions.

2.1 Personal Health Train

The FDT started in 2016 with the name Personal Health Train. It was an initiative of the Dutch Techcenter for Life Sciences, in collaboration with multiple Dutch research institutions. The concept of the FDT was announced with a video animation [16] portraying the idea of an infrastructure where data analysis on sensitive health data becomes more accessible and safe. This concept is built on the idea of enabling bringing algorithms to data rather than data to algorithms.

2.1.1 Train

"The set of all artifacts required to execute the distributed algorithm and return the results is called a 'Train'" [1]. A Train portrays a particular data request within the FDT infrastructure. The Train is loaded with a lot of different metadata information, where everything is described not only the Train itself but also what it expects from the Data Station and the required data. The actual algorithm is stored within the Train's Payload, this depends on the type of Train. At the moment there are four types of supported Trains: The Message Train, The API Train, the Script Train, The Query Train, and the Container Train. For example, the payload of the query Train could be a Python script or an R script, and the Container Train has a Docker Image as the payload.

2.1.2 Data Station

A Data Station is a software solution that can be managed by a health organization, such as a hospital or a health care provider. At this station, data is stored that is ready to be retrieved and used for an algorithm. The station provides a secure environment and a system with Computing Capacity for executing data analysis.

2.1.3 Computing Environment

When a Train gets deployed, it will have a certain algorithm that will process data and do analysis. This algorithm requires computing power for it to successfully run at a Data Station. The algorithm will take place in the Data Station's system, which is why it needs to ensure that the algorithm can be run. The system of the Data Station has computing components that could answer if it can handle the Train's requirements. To be able to match the Computing Environment requirements of the Train with the Computing Environment components at the Data Station, a metadata definition of it has to be made. The Computing Environment will consist of certain computing components that are necessary and sufficient as a basis to be able to match a Train with a Data Station. A Computing Environment could consist of certain parts of the central processing units, the ram, the graphical processing unit, the storage, and more. These components will be the basis for the Computing Environment and its semantic model.

2.2 Principles of FAIR

In recent times, there has been growing interest in the usage of FAIR Principles. For example, the innovative concept of the FDT is about handling health data in a FAIR way. The FAIR principles give guidelines to make digital resources Findable, Accessible, Interoperable, and Reusable [15]. In the context of the FDT, the FAIR principles ensure that health data is effectively managed and utilized. One of the key concepts of the FDT is the interoperability principle, which makes it possible for different systems (Trains and stations) to work seamlessly together. For the FDT, it ensures: **Findability** by having assigned unique identifiers and providing comprehensive metadata for a Train for example. **Accessibility** by making rare health data more available for data analysis. **Interoperability** by having all sorts of algorithm work at all Data Stations. **Reusability** by designing the Trains to be reused at multiple Data Stations [1].

2.3 Cloud Staging of the FAIR Data Train

The algorithm might not always be able to process the data at the Data Station directly. In that case, the possibility for

staging the data to the cloud, so the required processing can still be executed. This possibility is researched by Martinez [8, 12], where she first builds the framework for staging personal health Trains in the cloud and later builds an architecture that makes the FDT support dynamic cloud staging of the data.

The framework was first proposed by using novel technologies together with AWS for the hosting. One big discussion point of the cloud staging was that the data would have to be copied and sent to the cloud for the data analysis to proceed. This would neglect the main idea of the FDT that the data could stay where it is to preserve the safety of sensitive health data. However, Martinez promised that "the data are still within the data source realm and control, keeping their privacy." [8]. The proposed system would also comply with the main legal regulations for processing personal data in the cloud to keep the information as secure and private as possible, given that the cloud environment does not have technical problems or does not have been hacked [8].

In the second paper, she researches and implements the architecture of dynamically deploying a staging site in the FDT. The paper set out an architecture that would enable the dynamic staging of a Data Station in the cloud whenever the Data Station would not have the required amount of computing power that the Train requires for its algorithm to run successfully. Currently, the Train has two possible outcomes whenever it arrives at the Data Station. The first is when the Data Station has enough Computing Capacity that is required by the Train, the second is when the data has insufficient Computing Capacity that is required. With the help of Martinez's paper, it is now possible to still use that data for analysis with the help of the cloud.

2.4 Technologies

This research makes use of various technologies, ranging from semantic web tools and metadata systems. These technologies form the base for creating and validating the proposed semantic model for matching computing capacities in the FDT. The following sections will explain more about these technologies and why they are important for the research.

2.4.1 Semantic technologies

In order to give meaning to digital content in a way that computers can evaluate and comprehend, semantic technologies have been an essential component of this study. They are the foundation of the Semantic Web, an extension of the internet created to provide data with more context and better machine comprehension. The semantic web is advantageous for the FDT, ensuring smooth interoperability between Data Stations and Trains, but it also lays a foundation for future-proofing these processes. The value of the Semantic Web's capacity to organize and clarify data increases as data volume and complexity continues to rise.

Furthermore, the structure of the Semantic Web is dependent on key standards, such as the Resource Description Framework (RDF) [7] and the Web Ontology Language (OWL) [11]. RDF serves as a conduit for data exchange on the web, effectively mapping relationships between different pieces of data. OWL is a tool used to describe or explain complicated information structures and the connections between them. This allows datasets to have more detail and variety in the information they contain. These technologies together form the basis of the proposed Computing Environment model. By building the Semantic Web, they provide a structured, standardized, and scalable framework for managing data within the FDT infrastructure. They not only ensure efficient communication and data exchange but also enable more precise data analysis, thereby enhancing the overall efficacy of the system.

Semantic technologies help to look ahead, keeping the approach for managing data within the FDT up to date and adaptable, ready for whatever the future of data has in store. It is about providing meaningful identifiers to data and maintaining easy communication across systems, to make the data-handling approach strong regardless of what comes in the way.

2.4.2 Metadata management

Metadata is a term used to describe information about other data. This type of data helps understand the contents of a dataset and makes it easier to process and analyze. In this research, a detailed metadata definition for the Computing Capacity of the Data Station and the requirements of the Train will be created. To make sure this metadata is reliable, Shapes Constraint Language (SHACL) is used. SHACL helps to validate RDF graphs against certain conditions, which is important for making sure the Computing Environment's metadata fits with the requirements of the FDT.

3 METHODOLOGY

In this section, the methodologies are addressed that are employed during this research to handle the three research questions of this paper. The approach is designed to be practical and flexible, adapting as required to ensure each research question is sufficiently addressed. The methodology is broken down into three primary parts, each corresponding to a specific research question.

To answer the first research question, several existing research papers are reviewed. This review is not as in-depth as a traditional literature review but instead focuses on identifying and understanding the necessary computing resources. The materials fall into two categories; First of all, a few key papers on hardware components will be examined to better understand the necessary resources for a Computing Environment, such as central processing units (CPU), memory, and storage capacity. Secondly, several documents detailing the PHT model will be examined. With the knowledge of the domain, the computing requirements and capacities can be identified based on its use cases and constraints. This early research leads to the identification of a comprehensive list of computing components that can accurately define the Computing Environment for the matching.

Using the identified list of computing components, a semantic model will be created to answer the second research question. This model sets out how computing components are connected inside the FDT concept, acting as a map or blueprint. The semantic model offers an RDF-based graphical representation of the Data Station's capacity and the Train's requirements, that will allow an automatic matching procedure. The creation of the semantic model follows the principles of the Resource Description Framework (RDF). The RDF's entity-path graph visualization provides a convenient way of illustrating how each

component of the Computing Environment (entity) is related to the others via paths. Each entity-path pair will be further described in a metadata scheme table.

Additionally, this paper will also take steps to make the architecture of dynamic cloud staging more accurate and efficient. The architecture would be able to read the Train's computing requirements and specifically stage a cloud environment that has enough computing power. This will make the cloud staging of the FDT more efficient by improving precision and reducing costs.

To answer the third research question, the semantic model will be validated by applying the Shapes Constraint Language (SHACL). SHACL is a description model that allows for the definition of conditions that the data must satisfy for the model to be by definition valid. It helps enforce constraints such as cardinality and datatypes, among others. This research question makes sure the model is reliable and follows the relevant requirements. The methodological approach uses domain-specific knowledge from literature review and technical expertise in semantic modeling. This comprehensive approach makes sure that the semantic model from the second research question and its SHACL definitions are both theoretically and practically applicable.

4 RESULTS

The first step of the results section is to find the most important and relevant computer components that can be used in the semantic model. This step is the foundation for creating the model, and later on validating it using the SHACL language. The results of all these steps will form the goal of this research paper: Researching and designing the semantic model that supports the metadata matching of the Computing Capacity within the FDT.

4.1 Identification of Computing Resources

Every computer system is made up of many parts. Each system is distinct because of the many different components, and every small detail in these components. A basic system consists of at least a processing unit, a memory unit, a data storage unit, and an output unit [10]. Details of the memory unit like the storage capacity, its generation, and the storage speed, make each component distinct and thus define the computer system.

In the FDT model, each Data Station system is different because they are managed by different health organizations. The Train, on the other hand, carries specific needs that must be met by the Data Station's computer system to run its algorithm correctly and without problems. Therefore, making sure that the Train's needs match the Data Station's capacity is important in the FDT model. To help with this, a semantic model that shows all the important computing components and how they relate to each other needs to be made.

4.1.1 Central processing unit

The Central Processing Unit (CPU) is one of the primary components of a computer system. It is responsible for interpreting and executing instructions in a computer program, and the CPU is an integral part of the computational functions [7]. For the FDT, both the number of cores and clock speed can be identified as important attributes. The number of cores relates to the parallel processing capabilities, impacting the speed at which the algorithm can run [10]. Alternatively, properties like cache memory [11], CPU architecture, and the number of threads are seen to be less significant. This is mostly because having all of them offers an amount of detail that is not required for the matching process. To match the computing needs of the Train with the processing power of the Data Station, the core count and clock speed are necessary to accurately describe the CPU in its basis.

4.1.2 Graphical processing unit

The Graphical Processing Unit is primarily designed for image rendering and performing very efficient parallel processing with lots of cores [10]. However, GPUs can also be used for data processing. The advantage of GPUs is that they can do several calculations at once, which makes them ideal for the complex matrix operations frequently used in machine learning applications [9]. They are also essential for deep learning due to their skill at processing massive data structures. For the FDT, both the number of GPU cores and the VRAM capacity are of importance. The number of cores influences parallel processing capabilities, while VRAM determines the volume of data the GPU can handle simultaneously. GPU speed is considered an unnecessary detail for the FDT matching process because the cores and VRAM give enough details about the GPUs capacities.

4.1.3 Random access memory

The Random Access Memory is the main memory of the computer. RAM functions as the temporary data storage of the CPU, enabling quick access to important information [7]. By having important information temporarily stored in the RAM, the overall speed of the computer becomes faster and works more efficiently. For the FDT, the RAM capacity of the Data Station is important as it determines the volume of data that can be simultaneously processed in the system. Other parts of the RAM are less important for the FDT matching process and thus will not be added to the model. These parts are the RAM speed and generation which could be interesting for deeper analysis.

4.1.4 Storage

The part of the computer where information is stored for long-term storage and access [7, 10]. Storage is where the FDT health organization stores its data. For the matching process of the FDT, it is important to know how much available storage there is. Having storage available is necessary for temporarily reserving storage space for the data analysis of the algorithm to take place. For the semantic model in its current state, it does not add value to add the type of drive that is used. Not only because a data center could be very big with many different drives, but the type does not change the way data will be processed in general.

4.1.5 Data transfer rate (Throughput)

The Data Transfer Rate also called the Throughput, is the rate at which data can be transferred within the computer system from one part to the other [10]. In Chapter 6 of the "Computing with Data" book written by G. Lebanon, it is said that: "Processing data at scale requires considerations of performance, throughput, and correctness." [7]. This rate is critical in the FDT context, as it affects how quickly data can be moved from storage to the CPU and RAM for data processing.

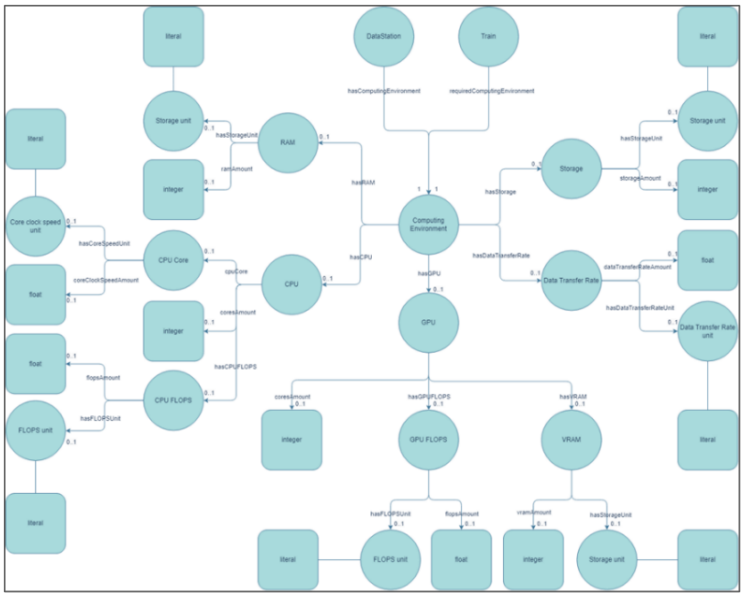

Figure 1 Computing Environment Semantic Model

4.1.6 FLOPS

FLOPS is a measure of a computer system's performance [7]. In the FDT, it can serve as an indicator of both CPU and GPU performance [14]. The FLOPS amount is something that is less accessible because it needs to be calculated. When available, FLOPS allows for a more accurate matching process by providing a universal measure of processing capacity.

4.2 Semantic Model

The goal of this section is to create an accurate semantic model of Computing Capacity matching by going into the approach and findings related to the second research question. By focusing on the key components from section 4.1 that make up the Computing Environment, it will be ensured that computing capacities are effectively matched between Trains and Data Stations.

4.2.1 From components to Computing Environment

Let's start by going back to the components that are identified in the previous section. These are not just chosen at random, they are chosen to provide an accurate representation of the computer environment. Grouping components that are related to each other helps to create a clear and organized structure. The groups that have been made are the 'CPU', 'GPU', 'RAM', 'Storage', and 'Data Transfer Rate'. All these are the parent entities to their children entity which they relate to. For example, the CPU is positioned as the parent entity of the cores, clock speed, and CPU FLOPSs. Just like the CPU, the GPU is the parent entity of the cores, VRAM, and GPU FLOPSs. The RAM, Storage, and Data Transfer Rate are all parents of two entities, which are the amount and their corresponding unit, all of which are further explained in section 4.2.2.

4.2.2 The semantic model

The creation of the semantic model starts with the definition of the primary entities and their relationships with each other. This forms the foundation of the semantic model. Building upon this, each primary entity is dissected into its parts, as determined by section 4.1's findings. These components were then grouped to make an accurate depiction of the Computing Environment.

At the highest level of the graph, there are the already existing Train and Data Station entities of the FDT ontology, these are connected to the central entity: "Computing Environment". The Train and the Data Station are added to the graph to show their relationships to the Computing Environment. The Train has the path 'requiredComputingEnvironment', which makes it clear that the Train has a Computing Environment metadata list consisting of required components that are needed to run its algorithm. The Data Station has the path 'hasComputingEnvironment', which shows that the Data Station has a set of components that together would form the Computing Environment of the Data Station. This list 'tells' the Train what it is capable of and thus if it can run the algorithm at the Data Station. The central entity branches out into each of the component groups: CPU, GPU, RAM, Storage, and Data Transfer Rate, each having its child entities.

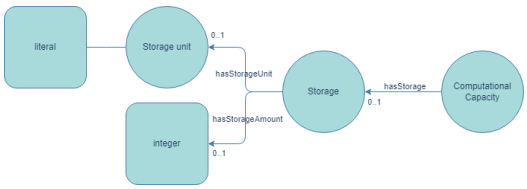

Figure 2 The Storage branch 'singled-out' of the semantic model

The model is pretty complex at first glance, so to simplify things, in Figure 2 the Storage 'branch' is singled out from the full semantic model. This smaller graph should help in understanding the overall structure. The Storage is related to the Computing Environment entity with the path hasStorage. Which indicates that there is a Storage entity stored in the Computing Environment metadata. This storage is connected to two child entities, the amount of storage and the storage unit. The amount of storage is the integer value of how much storage there is. The Storage Unit is the Unit in which the amount of storage is measured. Therefore, if there is 200 GB of storage in its Computing Environment, the 200 is the integer value and the GB is the unit, stored as a literal.

Table 1 FAIR Data Train Computing Environment metadata schema

Property	Path	Datatype	Description
Data Station	fdt:isDataStation		A software solution that can be managed by a health organization, where it hosts and controls specific health datasets within the FAIR Data Train infrastructure.
Train	fdt:isTrain		A privacy-preserving algorithm that travels to Data Stations within the FAIR Data Train infrastructure for data analysis.
Computing Environment	cev:hasComputingEnvironment / cev:requiredComputingEnvironment		The combination of hardware resources either available at a Data Station or required by a Train for performing data analysis tasks.
CPU	cev:hasCPU		Central Processing Unit, the primary hardware for processing instructions within a computer system.
CPU Cores amount	cev:cpuCoresAmount	integer	The number of independent processing units within the CPU. Measured with an integer amount (n).
CPU Core	cev:cpuCore		The individual processing units in a CPU, whose speed is defined by the clock rate.
CPU Clock Speed	cev:coreClockSpeedAmount	float	The operating speed of the CPU cores.
Core clock speed unit	cev:hasCoreSpeedUnit	literal	Measured in Gigahertz (GHz).
RAM	cev:hasRAM		Random Access Memory, is the volatile memory used by a computer for temporary storage of data during processing.
RAM Amount	cev:ramAmount	integer	The amount of RAM.
Storage	cev:hasStorage		The part of the computer where information is stored for long-term storage and access.
Storage amount	cev:storageAmount	integer	The amount of data storage space.
Storage unit	cev:hasStorageUnit	literal	Measured in Gigabyte (GB).
Data Transfer Rate	cev:hasDataTransferRate		The speed at which data can be transferred within a computer system.
Data Transfer Rate amount	cev:hasDataTransferRateAmount	float	The amount of data transfer speed.
Data Transfer Rate unit	cev:hasDataTransferRateUnit	literal	Measured in Gigabit per second (Gb/s).
GPU	cev:hasGPU		Graphics Processing Unit, is a processor dedicated to complex mathematical computations, like machine learning data analysis.
GPU Cores amount	cev:gpuCoresAmount	integer	The number of processing units within the GPU. Measured with an integer amount (n).
VRAM	cev:hasVRAM		Specialized memory is used by the GPU to store image and video data.
VRAM amount	cev:vramAmount	integer	The amount of VRAM within the GPU.
GPU FLOPS	cev:hasFLOPS		A measure of GPU computing performance, based on calculations performed per second.
CPU FLOPS	cev:hasFLOPS		A measure of CPU computing performance, based on calculations performed per second.
FLOPS amount	cev:flopsAmount	float	The amount of FLOPS.
FLOPS unit	cev:hasFLOPSUnit	literal	Measured in Teraflops (TFLOPS).

Moving on to the whole semantic model in Figure 1, patterns can be noticed that seem similar to the Storage entity. The Data Transfer Rate entity and the RAM entity are similar in structure to the Storage. Both have two child entities, which are the respective amount and the units. The CPU and GPU are more complex entities than these other three. The CPU and GPU both have three child entities that are related. The CPU has the child entities: the number of cores, Core Clock Speed, and CPU FLOPS. Where the number of cores is stored as an integer. The Core Clock Speed has the amount stored as a float, and its corresponding units as a literal. The GPU has the child entities: the amount of cores, VRAM, and GPU FLOPS. Where the amount of cores is stored as an integer, just like the CPU. The VRAM is stored as an integer, and its corresponding units as a literal. Both the CPU and the GPU have the FLOPS as a child entity, this is stored as a float and has the FLOPS Unit as a literal.

Every entity within this model is labeled with its cardinality, data type, and prefixes for each ontology. The cardinality of zero to one signifies that not every component is required to exist in the Computing Environment, offering flexibility in terms of which components are present. However, the more components are included, the more accurate the depiction of the system's Computing Environment requirements or capacity. Each entity in the model is distinguished by a unique ontology prefix. Which is the cev., and the FAIR Data Train (fdt.) [5] for the already existing Train and Data Station, thereby ensuring clarity and ease of identification

4.2.3 Metadata scheme

Besides the semantic model, Table 1 is made. This is a metadata scheme that is made alongside the semantic model to provide a more accessible view of the model [3]. This metadata scheme is formatted as a table, detailing individual entity information like the data types, and their descriptions.

In the path column, the ontology prefixes can be identified. These prefixes are essential in maintaining clarity

and coherence within the model. In line with the existing FAIR Data Train (FDT) infrastructure [4], the "Train" and "Data Station" entities have the prefix "fdt". However, all other entities have the custom-created prefix "cev", which stands for Computing Environment Vocabulary and can be found on the w3id website[1].

4.3 Semantic Model Validation Using SHACL

The third and final research question is about validating the semantic model using the Shapes Constraint Language (SHACL). SHACL provides a way to verify that the RDF graphs follow a set of conditions [3]. These conditions, or constraints, are expressed as shapes and provide the requirements for metadata definition.

4.3.1 SHACL and its Role in Semantic Model Validation

SHACL is a language recommended by the World Wide Web Consortium (W3C) [13] for defining and validating RDF graphs and semantic models. It provides a way to express conditions that data should satisfy, making it ideal for checking the integrity and correctness of RDF data. In the context of this research, SHACL provides the necessary tools to verify if the Computing Environment of Trains and Data Stations matches the defined shapes as represented in the semantic model.

```
@prefix fdt: <http://w3id.org/fdt/> .
@prefix cev: <http://w3id.org/cev/> .
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

# Shape for a Train
fdt:TrainShape a sh:NodeShape ;
    sh:targetClass fdt:Train ;
    sh:property [
        sh:path fdt:requiredComputingEnvironment ;
        sh:node cev:ComputingEnvironmentShape ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ].

# Shape for Computing Environment
cev:ComputingEnvironmentShape a sh:NodeShape ;
    sh:property [
        sh:path cev:hasCPU ;
        sh:node cev:CPUShape ;
        sh:minCount 0 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path cev:hasGPU ;
        sh:node cev:GPUShape ;
        sh:minCount 0 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path cev:hasRAM ;
        sh:node cev:RAMShape ;
        sh:minCount 0 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path cev:hasStorage ;
        sh:node cev:StorageShape ;
        sh:minCount 0 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path cev:hasDataTransferRate ;
        sh:node cev:DataTransferRateShape ;
        sh:minCount 0 ;
        sh:maxCount 1 ;
    ].
```

Figure 3 SHACL of the Train Shape and Computing Environment Shape

4.3.2 SHACL Implementation for the Semantic Model

The SHACL validation used for the semantic model includes definitions (shapes) for each primary entity of the model (Train, Data Station, Computing Environment, CPU, GPU, RAM, Storage, and Data Transfer Rate). Each shape specifies the expected properties, their cardinality (minCount and maxCount), and data type. The validation ensures that the data in the semantic model adheres to the structural requirements and data types as laid out in the shapes.

In Figure 3 a limited version of the SHACL is visible, while the full SHACL can be seen in the GitHub repository[2], It can be seen that the "Train" shape requires a "requiredComputingEnvironment" property that references a "ComputingEnvironmentShape" node, and it must exist exactly once (minCount 1, maxCount 1). Within the "ComputingEnvironmentShape", the component group shapes are defined. Each of these has a variety of properties such as core counts, FLOPS, VRAM amounts, and storage units, with each having its cardinality and datatype defined.

5 DISCUSSION

The results section aimed to develop a semantic model to support the metadata matching of the Computing Capacity within the FDT infrastructure. The methodological approach starts with identifying the relevant computing components for this model. These components are then used for the semantic model creation and validation using SHACL.

Computer systems are highly diverse due to their various amount of components, and each component has its details. With elements like a processing unit, memory unit, data storage unit, and output unit being basic parts of any system, it is the specific details within these components that differentiate one system from another. Such diversity is seen in the FDT model since all the Data Stations are maintained by different health organizations. Each of these stations is a system with its own set of computing components that must be matched with the computing requirements of the Train for the algorithm to work properly. Therefore, the goal is to develop a semantic model detailing the relevant computing components and their relationships.

Relevant components such as the CPU, GPU, RAM, Storage, Data Transfer Rate, and FLOPS for the CPU and GPU are identified. Each of these components has specific attributes that play important roles within the FDT data analysis context. While some attributes are not identified, such as cache memory, and the CPU architecture in the case of the CPU. Attributes like the number of cores and clock speed are enough to accurately describe the computing power of the CPU. Similar evaluations are made for the other components, choosing the components necessary for the matching process without overcomplicating the model by adding every detail.

Components that are not chosen are the network speed and the operating system. Recall the main idea behind the FDT where the Train goes to the data. The data does not have to be sent over the network for this reason, making the network speed irrelevant to the semantic model of the Computing Environment. The operating software could be relevant for future improvements of the semantic model. As this could be a requirement for certain programs that need to be run at the station, like the Docker software of the specific Container Train. Therefore, while the operating software could be a relevant component of the semantic model, it is not necessary for the current Computing Environment of the FDT.

The semantic model is designed to combine these critical components and provide a structure that supports effective matching between Trains and Data Stations. The model's parent-child relationships provide a structured

[1] www.w3id.org/cev

[2] https://github.com/baran2411/ComputingEnvironment

graph and address the components that are identified as relevant. This resulted in a semantic model that is flexible in terms of the components present, yet provides an accurate depiction of the system's Computing Environment requirements or capacity. The semantic model is flexible by having the units and amounts separately in the metadata. This gives the ability to store all the components with various unit sizes, for example, the storage can be stored in 'MB', 'GB', and 'TB'. This ability to adapt to advancements in technology ensures the longevity of the model.

The final step is the validation of the semantic model using the SHACL language. SHACL provides a way to express conditions that data should satisfy, making it ideal for checking the integrity and correctness of RDF data. It is a validation layer ensuring that the semantic model adheres to the structural requirements and data types as laid out in the shapes. The already existing ontology prefix "fdt" for the FAIR Data Train, and newly made "cev" for Computing Environment Variables, are important in providing an easy method for identification and interpretation.

5.1 Limitations

The research was not without limitations. The exclusion of certain computing components like CPU threads and GPU speed from the semantic model could impact its applicability in scenarios where these factors matter. The model's design is yet to be tested in real-world situations, leaving potential challenges unexplored. Also, the structural validation offered by SHACL may not be sufficient for complex scenarios that require deep validation rules.

Nevertheless, this research offers valuable insights into the design of a semantic model to match a Train and Data Stations within the FDT. The systematic identification of relevant computing components and their representation within a flexible, organized structure provides a significant contribution to the field. It establishes a good basis for future research to refine the model and validate it within practical applications, improving the efficiency of data processing in diverse Computing Environments.

6 CONCLUSION

The goal of this research is to research and design a semantic model that supports the metadata matching of Computing Capacity within the FDT. This paper ensures that the computing needs of the Train can match the capacities of the Data Station, therefore providing the tools to make data processing and analysis more efficient. Furthermore, this semantic model allows a more effective dynamic cloud staging process. The goal of this research paper is to construct around three main research questions.

The first research question dealt with identifying the most relevant computing components that need to be included in the semantic model. It is found that the relevant computing components to consider are the Central Processing Unit, Graphical Processing Unit, Random Access Memory, Storage, Data Transfer Rate (Throughput), and FLOPS. Each component has unique attributes which define the general Computing Environment of a Data Station.

The second research question is about the creation of the semantic model itself. The finalized semantic model represents a structured view of the Computing Environment of both Trains and Data Stations within the FDT model. The model successfully combines and relates the relevant computing components, resulting in a detailed model of the Computing Environment. The model also allows the necessary flexibility for variations in the specific configurations of different Trains and Data Stations.

The third research question focused on the validation of the semantic model using the SHACL language. Using SHACL, conditions are set to ensure that the defined model meets the standards for accuracy. The SHACL validation checks that all data in the semantic model adheres to the structural requirements and data types as laid out in the defined shapes.

6.1 Future Work

There are several areas where this work could be expanded. For one, the semantic model could be extended to accommodate the requirements of the specific types of Trains, such as Query Trains and Container Trains, which could have their different computing requirements. In addition, the model could be tested for compatibility with the dynamic cloud staging system, by checking the two possible states a Train can enter during its matching process.

Furthermore, certain computing components could be required in all cases. For example, by determining that a Train should always match with at least the CPU Cores. Having certain components always be required could improve the efficiency, however, this also leads to it being a bit less accessible. This could be further researched.

Lastly, the practical application of the semantic model could be tested in real-world scenarios, assessing its effectiveness in accurately matching computing capacities and needs within the FDT infrastructure. Future research and practical application would shed more light on the full potential of the semantic model in optimizing the data processing within the FAIR Data Train framework.

REFERENCES

[1] Beyan, O. et al. 2020. Distributed Analytics on Sensitive Medical Data: The Personal Health Train. *Data Intelligence*. 2, 1–2 (Jan. 2020), 96–107. DOI:https://doi.org/10.1162/dint_a_00032.

[2] Choudhury, A. et al. 2020. Personal Health Train on FHIR: A Privacy Preserving Federated Approach for Analyzing FAIR Data in Healthcare. *Machine Learning, Image Processing, Network Security and Data Sciences* (Singapore, 2020), 85–95.

[3] FAIR Data Point specifications: *https://specs.fairdatapoint.org/*. Accessed: 2023-06-23.

[4] FAIR Data Train specifications: *https://specs.fairdatatrain.org/*. Accessed: 2023-06-23.

[5] Index of /fdt: *https://w3id.org/fdt/*. Accessed: 2023-06-25.

[6] Kumar, S. and Singh, M. 2019. Big Data Analytics for Healthcare Industry: Impact, Applications, and Tools. *Big Data Mining and Analytics*. 2, (Mar. 2019), 48–57. DOI:https://doi.org/10.26599/BDMA.2018.9020031.

[7] Lebanon, G. and El-Geish, M. 2018. Essential Knowledge: Hardware. *Computing with Data: An Introduction to the Data Industry*. G. Lebanon and M. El-Geish, eds. Springer International Publishing. 7–36.

[8] Martinez, V.G. et al. 2021. A Framework for Staging Personal Health Trains in the Cloud. *Proceedings of the 17th International Conference on Web Information Systems and Technologies, WEBIST 2021, October 26-28, 2021*. SCITEPRESS. 133–144.

[9] Pandey, M. et al. 2022. The transformational role of GPU computing and deep learning in drug discovery. *Nature Machine Intelligence*. 4, 3 (Mar. 2022), 211–221. DOI:https://doi.org/10.1038/s42256-022-00463-x.

[10] Patterson, D.A. and Hennessy, J.L. 2012. *Computer Organization and Design: The Hardware/Software Interface*. Elsevier.

[11] Ryabko, B. and Rakitskiy, A. 2017. Application of the Computer Capacity to the Analysis of Processors Evolution. arXiv.

[12] Santos, L.O.B. da S. et al. 2023. Personal Health Train Architecture with Dynamic Cloud Staging. *SN Computer Science*. 4, 1 (Jan. 2023), 14. DOI:https://doi.org/10.1007/s42979-022-01422-4.

[13] Shapes Constraint Language (SHACL): 2017. *https://www.w3.org/TR/shacl/*. Accessed: 2023-06-25.

[14] Welten, S. et al. 2021. DAMS: A Distributed Analytics Metadata Schema. *Data Intelligence*. 3, 4 (Oct. 2021), 528–547. DOI:https://doi.org/10.1162/dint_a_00100.

[15] Wilkinson, M.D. et al. 2018. A design framework and exemplar metrics for FAIRness. *Scientific Data*. 5, 1 (Jun. 2018), 180118. DOI:https://doi.org/10.1038/sdata.2018.118.

[16] 2015. *Personal Health Train - English*.